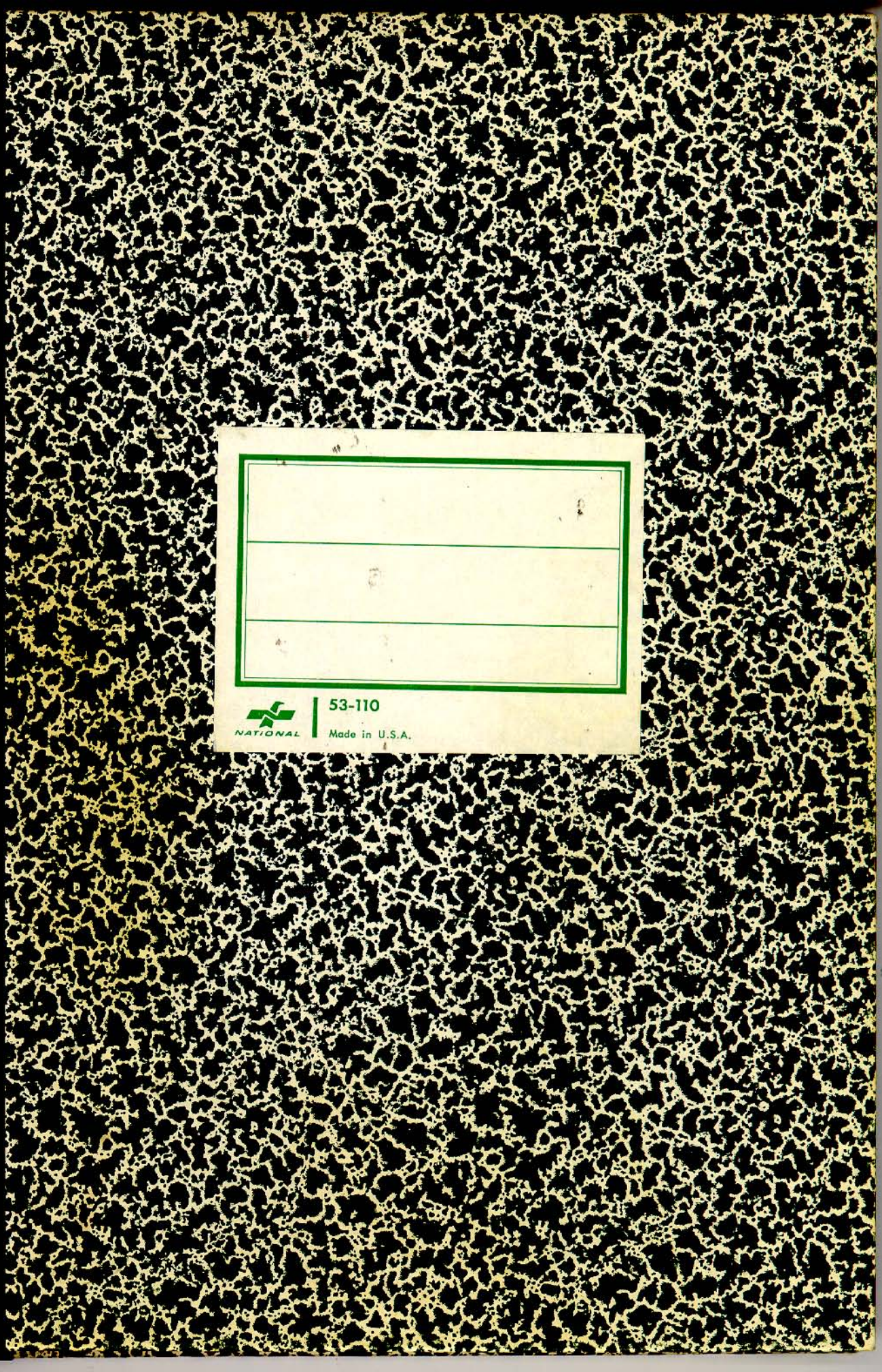
NATIONAL
53-110
Made in U.S.A.

Microcode Operation Tables

CPU SRC (2901)

M2	M1	M0	D	OPERANDS R	S
0	0	0	0	A	Q
0	0	1	1	A	B
0	1	0	2	0	Q
0	1	1	3	0	B
1	0	0	4	0	A
1	0	1	5	D	A
1	1	0	6	D	Q
1	1	1	7	D	0

A & B are general register Ports

Q is special register

D is the external Data Port

CPU FCTN (2901)

M5	M4	M3	D	Operation		
0	0	0	0	R PLUS S	R+S	S+R
0	0	1	1	S MINUS R	S-R	S-R
0	1	0	2	R MINUS S	R-S	
0	1	1	3	R OR S	R∨S	
1	0	0	4	R AND S	R∧S	
1	0	1	5	$\overline{R}$ AND S	$\overline{R}$∧S	
1	1	0	6	R XOR S	R⊻S	
1	1	1	7	R XNOR S	$\overline{R \veebar S}$	

CPU DST (Internal to 2901)

M8	M7	M6	D	RAM shift	RAM Load	Q-Reg shift	Q-Reg Load	Y Output	RAM shift R_0	RAM shift R_3	Q shift Q_0	Q shift Q_3
0	0	0	0	–	–	None	ALU	F	X	X	X	X
0	0	1	1	–	–	–	–	F	X	X	X	X
0	1	0	2	None	ALU	–	–	A	X	X	X	X
0	1	1	3	None	ALU	–	–	F	X	X	X	X
1	0	0	4	$R_{i+1} \to R_i$	ALU F_{i+1}	$Q_{i+1} \to Q_i$	Q Reg Q_{i+1}	F	F_0	IN_3	Q_0	IN_3
1	0	1	5	$R_{i+1} \to R_i$	ALU F_{i+1}	–	–	F	F_0	IN_3	Q_0	X
1	1	0	6	$R_i \to R_{i+1}$	ALU F_{i-1}	$Q_i \to Q_{i+1}$	Q Reg Q_{i-1}	F	IN_0	F_2	IN_0	Q_2
1	1	1	7	$R_i \to R_{i+1}$	ALU F_{i-1}	–		F	IN_0	F_2	X	Q_2

CPU FUNCTION TABLES

Logical

Octal m543	Octal m210	Group	Function
4	0	AND	A ∧ Q
4	1		A ∧ B
4	5		D ∧ A
4	6		D ∧ Q
3	0	OR	A ∨ Q
3	1		A ∨ B
3	5		D ∨ A
3	6		D ∨ Q
6	0	XOR	A ∀ Q
6	1		A ∀ B
6	5		D ∀ A
6	6		D ∀ Q
7	0	XNOR	$\overline{A \forall Q}$
7	1		$\overline{A \forall B}$
7	5		$\overline{D \forall A}$
7	6		$\overline{D \forall Q}$
7	2	INVERT	$\overline{Q}$
7	3		$\overline{B}$
7	4		$\overline{A}$
7	7		$\overline{D}$
6	2	PASS	Q
6	3		B
6	4		A
6	7		D
3	2	PASS	Q
3	3		B
3	4		A
3	7		D
4	2	"ZERO"	0
4	3		0
4	4		0
4	7		0
5	0	MASH	$\overline{A} \wedge Q$
5	1		$\overline{A} \wedge B$
5	5		$\overline{D} \wedge A$
5	6		$\overline{D} \wedge Q$

AND

B \ A	0	1
0	0	0
1	0	1

ARITHMETIC

Octal m543	Octal m210	CN = 0 GROUP	CN = 0 FNCT	CN = 1 GROUP	CN = 1 FNCT
0	0	ADD	A + Q	ADD + 1	A + Q + 1
0	1		A + B		A + B + 1
0	5		D + A		D + A + 1
0	6		D + Q		D + Q + 1
0	2	PASS	Q	INCREMENT	Q + 1
0	3		B		B + 1
0	4		A		A + 1
0	7		D		D + 1
1	2	DECREMENT	Q - 1	PASS	Q
1	3		B - 1		B
1	4		A - 1		A
2	7		D - 1		D
2	2	1's Complement	-Q - 1	2's Complement (Negate)	- Q
2	3		-B - 1		- B
2	4		-A - 1		- A
1	7		- D - 1		- D
1	0	SUBTRACT 1's Comp	Q - A - 1	SUBTACT 2's Comp	Q - A
1	1		B - A - 1		B - A
1	5		A - D - 1		A - D
1	6		Q - D - 1		Q - D
2	0		A - Q - 1		A - Q
2	1		A - B - 1		A - B
2	5		D - A - 1		D - A
2	6		D - Q - 1		D - Q

13 June 77
ARB

External Register Destination

M 11	M 10	M 9	D	
0	0	0	0	NONE
0	0	1	1	DATA Register (16 Bits)
0	1	0	2	ADDRESS Register (16 Bits)
0	1	1	3	ADDRESS EXTENSION Register (6 LS Bits)
1	0	0	4	PROCESSOR STATUS Register (12 MS Bits)
1	0	1	5	Cycle Counter (9 LS Bits)
1	1	0	6	INstruction Register (16 Bits)
1	1	1	7	Stack Overflow Register

DATA PORT SELECTOR

M 15	M 14	M 13	M 12	D			
0	0	0	0	0	Data Register	⟨15:00⟩	word
0	0	0	1	1	Data Register	⟨07:04⟩⟨15:08⟩	SWAB bytes
0	0	1	0	2	Data Register	⟨00:15⟩	Exchange Bits
0	0	1	1	3	Data Register	[07] ⟨07:04⟩	Sign extend Byte
0	1	0	0	4	Data Register	[05] ⟨05:04⟩	Sign extend 6 Bits
0	1	0	1	5	Instruction Register	⟨15:00⟩	word
0	1	1	0	6	INSTRUCTION Register	[07] ⟨07:04⟩	Sign extend Byte
0	1	1	1	7	INSTRUCTION Register	-0- ⟨05:04⟩	last 6 Bits
1	0	0	0	10_8	SCratch Register	⟨15:04⟩	word
1	0	0	1	11	Priority Vectors	-0- ⟨07:04⟩	Byte
1	0	1	0	12	Processor Status	⟨15:04⟩	word
1	0	1	1	13	Stack Limit Register	⟨15:8⟩⟨340_8⟩	word
1	1	0	0	14	Cycle Counter	-0- ⟨07:04⟩	Byte
1	1	0	1	15	* Microcode Constant	⟨MC47:MC32⟩	word
1	1	1	0	16	Console Switches 1	⟨SW15:SW00⟩	word
1	1	1	1	17	Console Switches 2	⟨OPER⟩-0-⟨SW21:SW16⟩	word

* if Microcode Constant is selected - Condition codes are disabled during the microcycle.

13 June 77
ARB

A Register Select

MC23	MC20	
0	0	0
0	1	1
1	0	2
1	1	3

0 * SRC Register Selected from Instruction Decoding ++

1 * DST Register Selected from Instruction Decoding +++

2 Register Selected by Data input Port <Ø3:Ø0>

3 * Register selected by <MC19:MC16>

B Register Select

MC27	MC26	
0	0	0
0	1	1
1	0	2
1	1	3

0 *SRC Register Selected from Instruction Decoding +

1 *DST Register Selected from Instruction Decoding +

2 Register Selected by Data Input Port <Ø3:Ø0>

3 Register Selected by <MC25:MC22>

* SRC, DST, and <MC19:MC16> are modified during the CUSER Mode such that a register 6₈ access is transposed to a register 16₈ access. A CB bit set disables this transposition.

++ LSB of SRC is OR'D with MC22

+++ LSB of DST is OR'D with MC16

Arithmetic Carry In Control

m31	m30	m29	m28	O	
0/1	0	0	0	0	'0'
	0	0	1	1	C – Carry Bit of PSR
	0	1	0	2	V – Overflow / Parity Bit of PSR
	0	1	1	3	Z – Zero Bit of PSR
	1	0	0	4	N – Sign Bit of PSR
	1	0	1	5	C' – previous cycle carry
	1	1	0	6	WR67S – Word or R6, R7, $R16_8$, $R17_8$ of Source [4]
	1	1	1	7	WR67D – Word or R6, R7, $R16_8$, $R17_8$ of Destination [4]

With m31 = 0 above carry in's

With m31 = 1 Complement carry in's

13 June 77

ARB

PROCESSOR STATUS BIT N SELECTOR

M35	M34	M33	M32	N_8		
0	0	0	0	$\emptyset$	N -	Current PSR Bit N
0	0	0	1	1	Z -	Current PSR Bit Z
0	0	1	0	2	V -	Current PSR Bit V
0	0	1	1	3	C -	Current PSR Bit C
0	1	0	0	4	SetN -	Instruction Decoded Set Bit N
0	1	0	1	5	CLRN -	Instruction Decoded clear Bit N
0	1	1	0	6	CPUO(3) -	Bit 3 output of ALU
0	1	1	1	7	N ∀ CPUN -	Current PSR Bit N ∀ present N state
1	0	0	0	10_8	CPUN -	present cycle N state (sign)
1	0	0	1	11	CPUZ -	present cycle Z state (zero)
1	0	1	0	12	CPUV -	present cycle V state (overflow)
1	0	1	1	13	CPUC -	present cycle C state (carry)
1	1	0	0	14	CPUN ∀ C -	
1	1	0	1	15	CPUN ∀ CPUV -	
1	1	1	0	16	'$\emptyset$' -	
1	1	1	1	17	'1' -	

C,V,Z,&N Processor status clocking is inhibited during a Data port selectcode of 15_8 (13_{10})

PROCESSOR STATUS BIT V SELECTOR

M39	M38	M37	M36	N_8	
0	0	0	0	0	CPUP – present cycle parity state of Low Byte. Even gives 1
0	0	0	1	1	Z – Current PSR Bit Z
0	0	1	0	2	V – Current PSR Bit V
0	0	1	1	3	C – Current PSR Bit C
0	1	0	0	4	SET V – Instruction Decoded Set V Bit
0	1	0	1	5	CLR V – Instruction Decoded CLR V Bit
0	1	1	0	6	CPUO<1> – Bit 1 output of ALU
0	1	1	1	7	N ∀ CPUN – Current PSR Bit N ∀ present N state
1	0	0	0	10_8	CPUN – present cycle N state (sign)
1	0	0	1	11	CPUZ – present cycle Z state (zero)
1	0	1	0	12	CPUV – present cycle V state (overflow)
1	0	1	1	13	CPUC – present cycle C state (carry)
1	1	0	0	14	CPUN ∀ C –
1	1	0	1	15	CPUN ∀ CPUV –
1	1	1	0	16	'Ø' –
1	1	1	1	17	'1' –

13 June 77
ARB

PROCESSOR STATUS BIT Z Select

IR3	IR2	IR1	IR0	N_8			
0	0	0	0	$\emptyset$	N	–	Current PSR Bit N
0	0	0	1	1	Z	–	Current PSR Bit Z
0	0	1	0	2	V	–	Current PSR Bit V
0	0	1	1	3	C	–	Current PSR Bit C
0	1	0	0	4	SET Z	–	Instruction Decoded Set Z
0	1	0	1	5	CLR Z	–	Instruction Decoded CLR Z.
0	1	1	0	6	CPUO<2>	–	Bit 2 output of ALU
0	1	1	1	7	N ≠ CPUN	–	current PSR bit N ≠ present CPUN state
1	0	0	0	10_8	CPUN	–	present cycle N state (sign)
1	0	0	1	11	CPUZ	–	present cycle Z state (zero)
1	0	1	0	12	CPUV	–	present cycle V state (overflow)
1	0	1	1	13	CPUC	–	present cycle C state (carry)
1	1	0	0	14	Z ∧ CPUZ	–	multiple word zero check
1	1	0	1	15	CPUN ≠ CPUV	–	
1	1	1	0	16	'$\emptyset$'	–	
1	1	1	1	17	'1'	–	

Processor status clocking is inhibited during a Data Port select code of 15_8

PROCESSOR STATUS Bit C Select

M 4 7	M 4 6	M 4 5	M 4 4	No			
0	0	0	0	0	LSBR	–	Least Significant Bit of ALU
0	0	0	1	1	$\overline{CPUC}$	–	present cycle $\overline{C}$ state
0	0	1	0	2	V	–	Current PSR Bit V
0	0	1	1	3	C	–	Current PSR Bit C
0	1	0	0	4	SetC	–	Instruction decoded Set C
0	1	0	1	5	CLRC	–	Instruction decoded clear C
0	1	1	0	6	CPUO <Ø>	–	Bit Ø output of ALU
0	1	1	1	7	$\overline{CPUZ}$	–	present cycle $\overline{Z}$ state
1	0	0	0	10₈	CPUN	–	present cycle N state
1	0	0	1	11	CPUZ	–	present cycle Z state
1	0	1	0	12	CPUV	–	present cycle V state
1	0	1	1	13	CPUC	–	present cycle C state
1	1	0	0	14	LSBQ	–	Least significant Bit of Q-reg (shift)
1	1	0	1	15	MSBQ	–	most significant Bit of Q reg (shift)
1	1	1	0	16	'Ø'	–	
1	1	1	1	17	'1'	–	

13 June 77

ARD

ALU Shift/Rotate Source Data

m50 m49 m48	N_8	RAM Shifting	RIN_0 or RIN_3 Data
m53 m52 m51		Q-Reg Shifting	QIN_0 or QIN_3 Data
0 0 0	0	'0'	
0 0 1	1	'1'	
0 1 0	2	MSBR - Most significant Bit of Ram (CPUN)	
0 1 1	3	MSBQ - Most significant Bit of Q Reg	
1 0 0	4	C - current PSR Carry state	
1 0 1	5	CPUN ⊻ CPUV -	
1 1 0	6	LSBR - Least significant bit of Ram	
1 1 1	7	LSBQ - Least significant bit of Q Reg	

m54 - CB - Console Bit

a) disables NPR transfers on the Bidirectional BUS

b) disables Force Level Processor Traps

c) disables interrupts during DATI IR's

d) disables address transformation of Register codes

PROCESSOR MODE SELECT

m57	m56	m55		ALU output BUS
0	0	0	LB<07:04> - Low Processor Byte (8 Bits)	Low 16 Bits
0	0	1	LW<15:04> - Low Processor word (16 Bits)	Low 16 Bits
0	1	0	UB<23:16> - upper word Byte (8 Bits)	upper 16 Bits
0	1	1	UW<31:16> - upper word (16 Bits)	upper 16 Bits
1	0	0	LW/LB - conditional word/byte operation	Low 16 Bits
1	0	1	UW/UB - conditional word/byte operation	upper 16 Bits
1	1	0	3B<23:04> - 24 Bit processor operation	Lower 16 Bits
1	1	1	4B<31:04> - 32 Bit processor operation	Lower 16 Bits

PROCESSOR MATH SELECT

Modifies M1 and M3 coding

m59	m58		
0	0	NORMAL ALU Coding	NORMAL
0	1	current PSR C Bit → M1 [C=1 0+B; C=0 A+B] with m3=0, m0=1	Restoring Division
1	0	current PSR $\bar{C}$ Bit → M1 [C=1 A+B; C=0 0+B]	Multiplication
1	1	current PSR $\bar{C}$ Bit → M3 [C=1 R+S; C=0 S-R] with m4, m5 = 00	Non Restoring Division

14 June 77

ARB

PROCESSOR CLOCK Control

m61	m60		
0	0	—	Free Running Clock
0	1	WI/O H	if I/O is Busy clock is held in high state at beginning of microcycle
1	0	WI/O L	if I/O is Busy clock is held in Low state at end of Microcycle
1	1	HALT	if I/O is Busy a WI/O H is performed followed by a Halt of the CPU clock - control reverts to micro switches on the console

m62 CCL — Cycle Counter Clock - cycle counter is clocked at beginning of current microcycle

m63 CFP — Clear Force Priority Logic - Must be used to clear Force Logic after an abort to a Force priority.

Microcontrol Condition Selector

m68 m67 m66 m65 m64	N_8	Select inverses with m69 = 1		
0 0 0 0 0	0	C	C set (Carry)	Current PSW Status
0 0 0 0 1	1	V	V set (overflow)	
0 0 0 1 0	2	Z	Z set (Zero)	
0 0 0 1 1	3	N	N set (Negative)	
0 0 1 0 0	4	CVZ		
0 0 1 0 1	5	C∀N		
0 0 1 1 0	6	V∀N		
0 0 1 1 1	7	(V∀N)VZ		
0 1 0 0 0	10_8	C'		previous ALU operation status
0 1 0 0 1	11	V'		
0 1 0 1 0	12	Z'		
0 1 0 1 1	13	N'		
0 1 1 0 0	14	C'VZ'		
0 1 1 0 1	15	C'∀N'		
0 1 1 1 0	16	V'∀N'		
0 1 1 1 1	17	(V'∀N')VZ		
1 0 0 0 0	20_8	'Ø'		
1 0 0 0 1	21	Trace Bit		
1 0 0 1 0	22	LOOKAHEAD CONDITION	(LKDC)	
1 0 0 1 1	23	PF Down		
1 0 1 0 0	24	BRANCH	Instruction Decoded	
1 0 1 0 1	25	Interrupt Pending		
1 0 1 1 0	26	SRC Ø	even source register	
1 0 1 1 1	27	Byte	Byte operation Instruction Decoded	
1 1 0 0 0	30_8	MMGT Enabled		
1 1 0 0 1	31	CUSER Mode		
1 1 0 1 0	32	PUSER Mode		
1 1 0 1 1	33	Lookahead in progress	(LKD)	
1 1 1 0 0	34	IED		
1 1 1 0 1	35	BUS BR Interrupt		
1 1 1 1 0	36	External Interrupt Line		
1 1 1 1 1	37	Counter ≠ Ø		

indicated states give a '1' Logic level

14 June 77
ARB

I/O Control Selector

for I/O operation IO9 = 1 (i.e M79 = 1)

M73	M72	M71	M70		
0	0	0	0	*	DATI (delayed)
0	0	0	1	*	DATIP (delayed)
0	0	1	0	*	DATO (delayed)
0	0	1	1	*	DATOB (delayed)
0	1	0	0	*	DATI.IR " (delayed)
0	1	0	1	*	Lookahead DATI IR (delayed)
0	1	1	0		INTERRUPT INPUT (delayed)
0	1	1	1		BUS INIT (delayed)
1	0	0	0	*	DATI (immediate)
1	0	0	1	*	DATIP (immediate)
1	0	1	0	*	DATO (immediate)
1	0	1	1	*	DATO B (immediate)
1	1	0	0	*	DATI IR (immediate)
1	1	0	1	*	Lookahead DATI IR (immediate)
1	1	1	0		INTERRUPT INPUT (immediate)
1	1	1	1		INTERRUPT CLOCH (immediate)

* Control options effect BUS cycle

I/O Control Options

m74 BE — Byte enable bit —
if BYTE bit of INSTRUCTION Code is set
this Bit controls selection of Byte or Word
I/O sequence

m75 NDSP — NDA or Special enable bit
if NDA Bit of Instruction code is set
then this enable bit will abort I/O operations
except INIT & Interrupt clock
should not be used during Interrupt inputs

if Special Bit of Instruction code is set
then this enable transforms a DATIP into
a DATI operation

Memory Management

m 7 8	m 7 7	m 7 6		
0	0	0	Current Mode specified by the PSR	enabled by MM6T Enable Bit or MM6T Maintenance Bit
0	0	1	Previous mode specified by the PSR	
0	1	0	Kernel Mode	
0	1	1	User Mode	
1	0	0	Current Mode	enabled by MM6T Enable Bit disabled during Maintenance
1	0	1	Previous Mode	
1	1	0	Kernel Mode	
1	1	1	MM6T Disabled	

14 June 77
ABD

Non I/O Control Selector

For IO9 = ϕ (ie M79 = ϕ)

M78 CLR - Clears the programmed priority states during first half of microcycle

M77 Set level E ϕ

M76 Set level E1

M75 Set level E2

M74 Set level E3

M73 Set level E4

M72 Set level E5

M71 Set level E6

M70 Set level E7

Micro Controller Operation Control

	m95	m94	m93	m92		
	0	0	0	0	Branch <91:80> V condition	BROC
	0	0	0	1	Cond. Branch <91:80> V <N><C> else Next	CBNC
	0	0	1	0	Cond. Branch <91:80> V <Z><V> else Next	CBZV
	0	0	1	1	Cond. Branch <91:80> V <N><Z><V><C> else Next	CBNZVC
	0	1	0	0	Cond. Branch <91:80> else repeat	CBR
	0	1	0	1	Cond. Branch <91:80> else Next	CBN
	0	1	1	0	Cond. JSR <91:80> else Repeat	CJR
	0	1	1	1	Cond JSR <91:80> else Next	CJN
*	1	0	0	0	Cond Next else Repeat	CNR
*	1	0	0	1	Cond Popstack Next always	CPS
*	1	0	1	0	Cond RTS else Repeat	CRR
*	1	0	1	1	cond. RTS else Next	CRN
*	1	1	0	0	Cond. Branch Icode <107:96> current priority Next always	CBIC
*	1	1	0	1	Cond JSR Icode <107:96> current priority Next always	CJIC
*	1	1	1	0	Cond. Branch Icode <107:96> Next always clock's priority, branches next Icode level	CBIN
*	1	1	1	1	Cond JSR Icode <107:96> Next always clocks priority, JSR's next Icode level	CJIN

* See Icode specifications and special functions allowed when m95 = 1

14 June 77
ARS

ICODE Specifications

the Branches and JSR's to the Icode levels have several selectable features specified by bit's 91 & 90 of the normal <91:80> micro branch address

m 9 1	m 9 0	
0	0	} - Branch/JSR address is taken from the Internal Vector Registers + Vector Extension of the PCR
0	1	
1	0	- Branch/JSR address is taken from the Scratch Registers if in PED priority else uses Vector Registers + Extension
1	1	- Branch/JSR address is taken from the selected Instruction decoded Address if in the PED priority else uses Vector Registers + Extension

*

Vector Registers — 1 set of 32 priority addressed branch addresses in ROM

8 sets of 32 priority addressed branch addresses in RAM

sets are selectable by the PCR

Special Functions (m95 = 1 only)

m89 — clear internal interrupt level

m88	m87			
0	0	–	No change	ILN
0	1	–	Enable Interrupt Logic immediately	ILE
1	0	–	Disable Interrupt Logic immediately	ILD
1	1	–	Disable Interrupt Logic – enabled after next Instruction Register Decode Operation	ILDE

m86 — Enable writing of ALU output into selected SCRATCH Register

m85, m84, m83 — Write Address in Scratch Registers

m82, m81, m80 — Read Address in Scratch Registers

14 June 77
ARO

First Operational Console Emulator -

48 instructions to execute

Load Address

Examine contents of Memory / Registers

Deposit data into Memory / Registers

Continue program execution

Start program execution

19 June 77

ARS

Console Routine same CC, LW
overhead registers: R10, R11, R12, and R17

0	WIOH	D = IR	place I
1		R10 = #174000	put Mask
2		R12 = 0 CBN(1) [4]	initialize
3	[GO]	R11 ∧ R10 , CB	Mask switch
4		R11 = SW2 CBN(Z) [GO]	load Regi
5		R10 ∧ SW2 , CB	check switch
6		CBN($\overline{Z}$) [5]	if swith
7		R12 = PSR , CC = R12 CJR(1) [TST]	
10$_8$		SL R11	
11	[LAD]	SL R11 CBN($\overline{N}$) [EX]	
12		AD = R17 = SW1 , C = 0, N = 0 LW	
13		ADX = R17 = SW2 CBR(1) [END] UW	
14	[EX]	SL R11 CBN($\overline{N}$) [CONT]	
15		C = 1 , N = 0 CJN (C) [INC]	
16		AD = R17 LW	
17		D → R17 LW	
20$_8$		R12 = PSR , CC = R12 CBNZV(Z) [20$_8$]	
21		ADX = R17 , UW , CB, DATI CBR(1) [END+1]	
22		D = R<D> , UW CBR(1) [END+1]	
23		D = R<D> , LW CBR(1) [END+1]	
24	[CONT]	SL R11 CBN($\overline{N}$) [START]	
25		AD = R7 LW	
26		ADX = R7 UW	
27		R12 = PSR , CC = R12 , CB, DATIR CBR(1) [0]	
30$_8$	[START]	SL R11 CBN($\overline{N}$) [DEP]	
31		AD = R7 = R17 , LW, INIT	
32	WIOH	ADX = R7 = R17 , UW, DATIR CBR(1) [0]	

Instruction Register Contents in D register for display
osh word for switches LAD, EX, Cont, STRT, Deposit in Register 177_8
routine condition codes to Ø, branch to [4]
witch, and check for switches depressed
Regi ter 11 with switch positions, Branch [GO] if none set, else next
witch for release
h has not been released Branch to [5] else continue

store program CC's in R12, set console CC's from R12, go test for Reg Add.

test sign bit and shift word higher (×2)

test sign bit, shift, branch to [EX] if LAD switch not depressed
move switch address to R17, set codes for new Address
move upper switch address to R17, Branch to [END]

test sign bit, shift branch to [CONT] if EX switch not depressed
set codes, JSR to [INC] if this is second time into EX
move address into AD
move address into D
Restore program CC's, store console CC's, Branch to designated Exam function
move address to ADX, Start BUS DATI, BRANCH to END+1
move contents of register addressed by D into D registers $20_8 - 37_8$ Branch to END+1
move contents of register addressed by D into D registers Ø - 17_8 Branch to END+1

shift R11, test sign bit, branch to [STRT] if Continue switch not depressed
Move current Program counter into AD
move current Program counter extension into ADX
restore program CC's load next instruction

test sign bit, branch to [DEP] if START switch not depressed
load new program counter Address, Do a BUS INIT
load new program counter extension, Start execution

20 June 77
ARD

```
33  [DEP]                       CBN(N̄') [END]
34                  C=0, N=1    CJN(N) [INC]
35                  AD=R17 , R11 = SW1 , LW
36                  ADX=R17 , R11 = SW1 , LW
37                  D = R17             LW
40₈                 R12 = PSR , CC = R12        CBNZV(Z) [40 ]
41                  D = SW1 , CB , DATO         CBR(1) [END+1]
42                  D = R<D> = R11 , LW         CBR(1) [END+1]   Reg 20-37₈
43                  D = R<D> = R11 , LW         CBR(1) [END+1]   Reg 0-17₈

44  [END]           R12 = PSR , CC = R12
45  [END+1]  WIOH   R11 = SW2 , CB              CBR(1) [00]

46  [INC]           R17 = R17 + 1 , 3B          CBN(Z) [INC+2].
47                  R17 = R17 + 1 , 3B
(Test) 50₈ [INC+2]  R11 = #20                   CJR(1) [TST]  [Reg]
51                  R11 ∧ R17  LW   V=CPUZ      CRR(1).

(Reg) 52 [TST]      R17 - #77  4B
53                  R17 - #177704  LW           CBN(Z̄') [NO]
54                  #177737 - R17  LW           CBN(N') [NO]
55                                              CBN(N') [NO]
56                  Z = 1                       CRR(1)
57  [NO]            Z = 0                       CRR(1)
```

Branch to END if DEP switch not depressed
set codes for Deposit, JSR to INC if second time into DEP
move Address into AD, load R11 with Deposit data
move Address into ADX, load R11 with Deposit data
move Address into D
restore program CC's, store console CC's, branch to corresponding deposit
move data into D, do DATO ; Branch to END+1
move deposit data (in R11) into Register pointed to by D and into D, Branch to END+1
move deposit data (in R11) into Register pointed to by D and into D, Branch to END+1

Restore program CC's, store console CC's
wait for IO complete, look at switches, Branch to [GO] to test switches

increment address counter by 1, if register address skip next increment
increment address counter by 1
move test Bit into R11, JSR to TST to determine if address is register
test for upper or lower Registers V=1 lower, V=0 upper, RTS

compose address extension with register address
compose address with register address lower bounds, Branch if previous compare not register
compose address with register address upper bounds, Branch if previous compare not register
branch if previous compare not register.
register address
not register address

20 June 77
ARD

Instruction Decoded ROM ICODE

The CPU instruction decoder generates a unique address for each instruction sequence. The Icode address points to a 32 Bit Instruction code word which specifies particular parameters for the complete instruction.

Bits 96 - 107 / MA0 - MA11

Specify a microcode address which the CPU will branch to

OPTIONS: SID Bit of PCR = 0
(Internal Instruction Decoder for PDP11

then during a PED Level 4-7
, CBIN, CBIC, CJIN, CJIC

The CPU will branch to MA0 - MA11

Bits 112 - 115 FRC C1 C2 WT

These Bits control the execution of Instruction Lookahead operations

Immediately following any form of an instruction load but before branching to any programmed priority level ie E0 - E7, IOT, BPT, EMT, TRAP

FRC unconditionally allows a Lookahead instruction Load

C1 conditionally allows a Lookahead instruction Load

20 Nov 77
AB

after any programmed priority branch

C2 Conditionally enables a Lookahead instruction Load

The condition for C1 and C2 is selectable: a) PDP-11 $D\emptyset \wedge \overline{R7}$
b) External coding

WT This Bit controls the timing of the next instruction decode process

a) any DATIR will load the next instruction word into a temp Register

b) if WT = $\emptyset$ then the processor immediately decodes and loads the new ICODE Data

c) if WT = 1 then the processor waits for a WIOH or WIOL before decoding and loading the New Icode Data

This is necessary if the decoding must be delayed until the completion of the current instruction

TEO Level Programmed priorities

these levels are used as trap levels

Bits 112 - 115

TRAP
EMT
BPT
IOT
} Named for PDP-11 Traps.

these Priority levels Branch to the Vector Address + Extension

PEO level Programmed Priorities

these levels are used for subroutine Segments in instruction execution

Bits 127 - 124 'E0_E7

The highest priority is E0, lowest E7

These levels have Multiple options for branching

a) External/Internal Instruction Decoded Address MA0 - MA11

b) address in the Vector Adress + Extension

c) address from the Scratch Register

JONW77
ARO

119 TI - Trace Trap inhibit bit -
will inhibit level 12$_8$ Traps
when the T Bit is set

I/O Control Options

116 NDA - Enables an NDSP I/O operation to
be immediately aborted

117 SP - Enables an NDSP I/O operation to
transform a DATIP to a DATI operation

118 BYT - Enables a BYT I/O operation to
perform BYT or related BUS transactions

Enables conditional word/byt processor
operations to perform Byt ops.

```
        .PAGE
        .TITLE  ARB MICROASSEMBLER PROGRAM
        ;
        ;
        ;*******************************************************
        ;*                                                     *
        ;*      MICROASSEMBLER PROGRAM                         *
        ;*              FOR THE                                *
        ;*      MICROPROGRAMMED DIGITAL COMPUTER               *
        ;*                                                     *
        ;*              WRITTEN       JUNE 1979                *
        ;*              BY      ALAN R. BALDWIN                *
        ;*                                                     *
        ;*      MODIFIED FOR RT-11 V4.0 (FB/XM)                *
        ;*                            APRIL 1981               *
        ;*                                                     *
        ;*******************************************************
        ;
        ;
        .IIF    DF,NOPRNT       .NLIST          ;NO PRINTING
        ;
        .PAGE
        .SBTTL  CHARACTER DEFINITION TABLE
        ;
        CTRL.I  =11
        SPC     =40
        CR      =15
        LF      =12
        FF      =14
        COMMA   =54
        EQUAL   =75
        LTPARN  =50
        RTPARN  =51
        LFTBKT  =133
        RTBKT   =135
        LFTANG  =74
        RTANG   =76
        MINUS   =55
        PLUS    =53
        SMICLN  =73
        DIVIDE  =57
        COLON   =72
        ;
        .PAGE
        .SBTTL  ALL THOSE CONSTANTS
        ;
        SPSP=20040      ;DOUBLE SPACE
        M.SPC=46440
        I.SPC=44440
        V.SPC=53040
        TERMS=40.       ;GP TERM LIMIT
        ;
        .PAGE
        .SBTTL  VARIABLE DEFINITIONS
        ;
        ;LINE SCANNING STATUS WORD
        ;
LSTAT:  .WORD   0       ;LISTING STATUS
CDSTAT: .WORD   0       ;CURRENT STATUS
        MCODE   =1      ;MCODE DEFINED
        ICODE   =2      ;ICODE DEFINED
        VCODE   =4      ;VCODE DEFINED
        X4      =10
        NP      =20     ;WRITE INTO B-REG INDICATOR
        QR      =40     ;STORE IN Q-REG INDICATOR
        NM      =100    ;R-REG SHIFT INDICATOR
        NO      =200    ;Q-REG SHIFT INDICATOR
        NR      =400    ;AR: ENTRY INDICATOR
        NS      =1000   ;CC: ENTRY INDICATOR
        NT      =2000   ;PC: ENTRY INDICATOR
        NU      =4000   ;BR: ENTRY INDICATOR
        MI      =10000  ;INTERRUPT CONFLICT FLAG
        MS      =20000  ;SCRATCH REGISTER CONFLICT FLAG
        MP      =40000  ;INSTRUCTION SEQUENCE CONFLICT FLAG
        M.      =100000 ;NO CODE FLAG
        ;
```

```
C1:     .WORD   0       ;SCANNING POINTERS
C2:     .WORD   0
P1:     .WORD   0
P2:     .WORD   0
FBND:   .WORD   0       ;FIRST CHAR FOR LIST
LBND:   .WORD   0       ;LAST CHAR+1 FOR LIST
NA:     .WORD   0       ;GROUP POINTER
NB:     .WORD   0       ;CHARACTER POINTER
NC:     .WORD   0       ;CURRENT CHARCTER LOCATION
ND:     .WORD   0       ;FSBR90 TEMP
NE:     .WORD   0       ;FSBR90 TEMP
NF:     .WORD   0       ;FIRST CHARACTER OF GROUP
NG:     .WORD   0       ;TEMPORARY RESULT
NH:     .WORD   0       ;EQUAL SIGN POSITION
NI:     .WORD   0       ;MULTIPLE EXTERNAL REGISTER INDICATOR
NJ:     .WORD   0       ;MULTIPLE SCRATCH REGISTER INDICATOR
NK:     .WORD   0       ;RIGHT TERMINATOR POSITION
NL:     .WORD   0       ;LAST CHARACTER OF GROUP
NQ:     .WORD   0       ;BEFORE/AFTER '=' INDICATOR
NV:     .WORD   0       ;TEMPORARY CONSTANT
NW:     .WORD   0       ;FSBR90 TEMP
NX:     .WORD   0       ;SECONDARY OF
NY:     .WORD   0       ;SECONDARY CW
NZ:     .WORD   0       ;FSBR90 TEMP
EQ:     .WORD   0       ;EQUAL SIGN COUNTER
DP:     .WORD   0       ;PREVIOUS DATA PORT VALUE
PA:     .WORD   0       ;PREVIOUS A TERM VALUE
PB:     .WORD   0       ;PREVIOUS B TERM VALUE
PK:     .WORD   0       ;ARITHMETIC & LOGICAL OPERATION HASH CODE
II:     .WORD   0       ;TERM REMAINING TO BE EVALUATED INDICATOR
CW:     .WORD   0       ;OPERATION BIT PATTERN
OF:     .WORD   0       ;CHARACTER OPERATION VALUE
TV:     .WORD   0       ;CHARACTER VALUE
TX:     .WORD   0       ;LOCATION OF CHARACTER
TY:     .WORD   0       ;USED IN D136.
VW:     .WORD   0       ;VECTOR TEMP
VX:     .WORD   0       ;VECTOR TEMP
VY:     .WORD   0       ;VECTOR TEMP
VZ:     .WORD   0       ;VECTOR TEMP
ER:     .WORD   0       ;ERROR FLAG
ME:     .WORD   0       ;FIRST ENTRY INDICATOR
.SC:    .WORD   0       ;SEQUENCE #
        ;
        ;MICROCODE SET
        ;
MBASE:  .WORD   6
M.MPC:  .WORD   0
MW0::   .WORD   0
MW1::   .WORD   0
MW2::   .WORD   0
MW3::   .WORD   0
MW4::   .WORD   0
MW5::   .WORD   0
        ;
        ;INSTRUCTION CODE SET
        ;
IBASE:  .WORD   2
I.MPC:  .WORD   0
MW6::   .WORD   0
MW7::   .WORD   0
        ;
        ;VECTOR SET
        ;
VE:     .WORD   0       ;VECTOR #
VBASE:  .WORD   1
VECT:   .WORD   0
VECTV:  .BLKW   64.
        ;
        ;
IISTNG: .BLKW   4       ;STRING
STRING: .BLKW   4       ;STRING
X:      .WORD   0       ;LOOP INDEX
GP1:    .WORD   0       ;GP(X)
GP2:    .WORD   0       ;GP(X+1)
CQ:     .WORD   0       ;CONDITION TABLE ADDRESS
GP:     .BLKW   1+TERMS ;GP POINTERS
```

```
GPSYM:  .BLKW   4*TERMS ;GP SYMBOLS
ERRLOG: .WORD   0       ;USED BY INTERNAL ERROR ROUTINES
ERRTBL: .BLKB   10.     ;10 ERROR FLAGS
SAVESP: .WORD   0       ;TEMPORARY FOR STACK POINTER
        ;
CSTRNG: .BLKB   81.     ;CSI INPUT STRING
        .EVEN
        ;
DEXT:   .RAD50  "MIC"   ;DEFAULT EXTENSIONS
        .RAD50  "WCS"
        .RAD50  "LST"
        .WORD   0
        ;
.IAREA:
.ICHAN: .BYTE   0,10
.IBLK:  .WORD   0
.IBUFF: .WORD   IBUFF
.IWCNT: .WORD   256.
        .WORD   0
TWORDS: .WORD   0
RDERRF: .WORD   0
IBUFF:  .BLKW   256.
        ;
OAREA:  .BYTE   0,11
OBLK:   .WORD   0
OBUFF:  .WORD   OBJBUF
OWCNT:  .WORD   0
        .WORD   0
OBJBUF: .BLKW   256.
        ;
ASCBUF: .BLKB   80.     ;ASCII BUFFER FOR DATA
        ;
LAREA:  .BYTE   1,11
LBLK:   .WORD   0
LBUFF:  .WORD   LSTBUF
LBCNT:  .WORD   0
        .WORD   0
LSTBUF: .BLKW   256.
        ;
        LPAGE=60.       ;LINES PER PAGE
LINCNT: .WORD   0       ;CURRENT LINE COUNT
SBENTR: .WORD   0       ;SUBTITLE ENTRY INDICATOR
PASCNT: .WORD   0       ;PASS COUNTER
        ;
        .NLIST  BIN
HDR:    .ASCII  <CR><FF>
TITLE:  .ASCII  /                       /
ID:     .ASCII  / MICROASSEMBLER V02 /
DATE:   .ASCII  /          /
TIME:   .ASCII  /          /
PAGE:   .ASCIZ  /  PAGE   0    /
EHDR:   .ASCIZ  //
SBTTL:  .BLKB   80.     ;CHARACTER STRING
SBTTL1: .BLKB   80.     ;CHARACTER STRING
TBLCON: .ASCIZ  /TABLE OF CONTENTS/<CR><LF>
TBLSYM: .ASCIZ  /SYMBOL TABLE/<CR><LF>
        .LIST   BIN
        .EVEN
        ;
CRLFCH: .BYTE   CR,LF
        ;
T.AREA: .WORD   0
TMH:    .WORD   0
TML:    .WORD   0
HRTBL:  .WORD   3       ;HIGH ORDER HRS
        .WORD   0       ;HIGH ORDER MINS
        .WORD   0       ;HIGH ORDER SECS
        .WORD   45700   ;LOW ORDER HRS
        .WORD   7020    ;LOW ORDER MINS
        .WORD   74      ;LOW ORDER SECS
        ;
SCNTM1: .WORD   0
SCNTMP: .WORD   0       ;TEMPORARY
        .NLIST  BIN
SCNLOW: .ASCII  /          /
SCNVAL: .ASCII  /          /
```

```
        .LIST   BIN
        ;
        .PAGE
        .SBTTL  EXTERNAL MACROS
        ;
        .MCALL  .READW, .WRITW
        .MCALL  .CSIGEN,.PURGE
        .MCALL  .DATE,  .WAIT
        .MCALL  .CLOSE, .TTYOUT
        .MCALL  .TTYIN, .GTIM
        .MCALL  .TTINR, .RCTRLO
        .MCALL  .PRINT
        .MCALL  .GTLIN          ;RT-V4 INPUT
        ;
        .PAGE
        .SBTTL  GENERAL SUBROUTINE CALLING MACRO
        ;
.MACRO  CALL    .NAME,P1,P2,P3,P4,P5,P6,P7,P8
 .NLIST SRC
 .NARG  .N
 .N=.N-1
 MARK=6400
        MOV     %5,-(%6)
 .IIF   GE,.N-8.,       MOV     P8,-(%6)
 .IIF   GE,.N-7.,       MOV     P7,-(%6)
 .IIF   GE,.N-6.,       MOV     P6,-(%6)
 .IIF   GE,.N-5.,       MOV     P5,-(%6)
 .IIF   GE,.N-4.,       MOV     P4,-(%6)
 .IIF   GE,.N-3.,       MOV     P3,-(%6)
 .IIF   GE,.N-2.,       MOV     P2,-(%6)
 .IIF   GE,.N-1.,       MOV     P1,-(%6)
        MOV     #MARK+.N,-(%6)
        MOV     %6,%5
        JSR     %7,.NAME
 .LIST  SRC
.ENDM   CALL
        ;
.MACRO  SAVL1
 .NLIST SRC
        MOV     %1,-(%6)
        MOV     2(%5),%1
 .LIST  SRC
.ENDM   SAVL1
        ;
.MACRO  SAVL2
 .NLIST SRC
        MOV     %2,-(%6)
        MOV     4(%5),%2
 .LIST  SRC
.ENDM   SAVL2
        ;
.MACRO  SAVL3
 .NLIST SRC
        MOV     %3,-(%6)
        MOV     6(%5),%3
 .LIST  SRC
.ENDM   SAVL3
        ;
.MACRO  SAVL4
 .NLIST SRC
        MOV     %4,-(%6)
        MOV     10(%5),%4
 .LIST  SRC
.ENDM   SAVL4
        ;
.MACRO  SAVL5
 .NLIST SRC
        MOV     %5,-(%6)
        MOV     12(%5),%5
 .LIST  SRC
.ENDM   SAVL5
        ;
        RES0=12600      ;MOV    (%6)+,%0
        RES1=12601
        RES2=12602
        RES3=12603
```

```
        RES4=12604
        RES5=12605
        SAV0=10046      ;MOV    %0,-(%6)
        SAV1=10146
        SAV2=10246
        SAV3=10346
        SAV4=10446
        SAV5=10546
        ;
        .PAGE
        .SBTTL  UTILLITY MACROS
        ;
.MACRO  ERROR   .A
        MOV     .A,%0
        JSR     %7,TALYER
.ENDM   ERROR
        ;
.MACRO  CHKII
 .NLIST SRC
        TST     II              ;ANY MORE EVALUATION ?
        BEQ     .+6             ;IF NOT - SKIP
        JSR     %7,D123.
 .LIST  SRC
.ENDM   CHKII
        ;
.MACRO  UPDTPK  ..A
 .NLIST SRC
        MOV     PK,%0           ;GET VALUE
        ASH     #4,%0           ;SHIFT 4 PLACES
        BIS     ..A,%0          ;UPDATE VALUE
        MOV     %0,PK           ;SAVE RESULT
 .LIST  SRC
.ENDM   UPDTPK
        ;
.MACRO  GTIVAL  ..A,..B,..C
 .NLIST SRC
        MOV     ..B,%0
        ADD     ..C,%0
        ASL     %0
        ADD     ..A,%0
        MOV     (%0),%0
 .LIST  SRC
.ENDM   GTIVAL
        ;
.MACRO  FOR     ..A,..B,..C,..D,?..E,?..F
 .NLIST SRC
        MOV     ..B,..A
..E:    CMP     ..C,..A
        BCS     ..F
        JSR     %7,..D
        INC     ..A
        BR      ..E
..F:
 .LIST  SRC
.ENDM   FOR
        ;
        .PAGE
        .SBTTL  ENTRY POINT
        ;
        ;
RENTER: BR      PURGE           ;UPON REENTRY GO PURGE I/O CHANNELS
ENTER:  BIS     #20000,@#44     ;SET REENTER BIT IN JSW
        CLR     OBLK            ;INIT OUTPUT HANDLERS
        CLR     OWCNT
        CLR     LBLK
        CLR     LBCNT
TTIN:   .RCTRLO                 ;RESET ^O
        MOV     %6,SAVESP       ;SAVE STACK POINTER
        .CSIGEN #END,#DEXT,#0,#CSTRNG
        BCC     SWITCH          ;NO ERRORS - SWITCH
        MOV     @#52,%0         ;GET ERROR
        BNE     SW.A            ;NOT ILL CMND - SKIP
        .PRINT  #ERR49
        BR      PURGE
SW.A:   DEC     %0
        BNE     SW.B            ;NOT 'NO DEVICE' - SKIP
```

```
        .PRINT  #ERR50
        BR      PURGE
SW.B:   DEC     %0
        BNE     SW.C
        .PRINT  #ERR51
        BR      PURGE
SW.C:   DEC     %0
        BNE     SW.D            ;NOT 'FULL DIRECTORY' - SKIP
        .PRINT  #ERR44
        BR      PURGE
SW.D:   DEC     %0
        BNE     SW.E            ;NOT 'NO FILE' - SKIP
        .PRINT  #ERR40
        BR      PURGE
SW.E:   .PRINT  #ERR51          ;CSI ERROR
        BR      PURGE
SWITCH: MOV     #377,LSTAT      ;PRESET LISTING STATUS
        MOV     (%6)+,%0        ;SAVE POINTER
        MOV     %6,%1
CSI.A:  DEC     %0              ;ANY SWITCHES ?
        BLT     CSI.D           ;IF NOT - SKIP
        MOV     (%1)+,%2        ;GET SWITCH
        CMPB    #'L,%2          ;AN L SWITCH ?
        BNE     CSI.C           ;IF NOT - AN ERROR
        TST     %2              ;ANY ASSOCIATED VALUE ?
        BPL     CSI.A           ;IF NOT - LOOP FOR NEXT SWITCH
        CMPB    LSTAT,#377      ;STILL PRESET ?
        BNE     CSI.B           ;IF NOT - SKIP
        CLRB    LSTAT           ;ELSE SCRAP PRESETS
CSI.B:  BIS     (%1)+,LSTAT     ;SET FLAGS
        BR      CSI.A           ;LOOP FOR NEXT SWITCH
CSI.C:  .PRINT  #ERR48          ;BAD SWITCH
        MOV     SAVESP,%6       ;RESET STACK
        JMP     RENTER          ;TRY RESTART
CSI.D:  MOV     SAVESP,%6       ;RESTORE STACK POINTER
        .WAIT   #3              ;CHECK FOR INPUT FILE
        BCC     ENT.A           ;NO ERROR - SKIP
        .PRINT  #ERR40          ;NO INPUT FILE
PURGE:  .PURGE  #0
        .PURGE  #1
        .PURGE  #2
        JMP     ENTER           ;TRY TO RESTART
ENT.A:  CALL    BLDTIM,#TIME+2  ;GO FIND START TIME
        CALL    BLDDAT,#DATE+1  ;GO FIND DATE
        JSR     %7,CLRSYM       ;INITIALIZE SYMBOL TABLE
        MOV     #INBUFF,.IBUFF  ;SET UP FOR FIRST DATA LOAD
        JSR     %7,SREAD        ;GO READ
        CMP     RDERRF,#1       ;FATAL ERROR ?
        BNE     ENT.B           ;IF NOT - SKIP
        .PRINT  #ERR41          ;BAD FILE ?
        BR      PURGE
ENT.B:  JSR     %7,D20.         ;INITIALIZE VALUES
        JSR     %7,D21.         ;INITIALIZE MORE
        MOV     #IBUFF,.IBUFF   ;RESET POINTER
        MOV     TWORDS,%0       ;BYTES READ
        ADD     #INBUFF,%0      ;COMPUTE LAST POSITION
        MOV     %0,IBUFND       ;SAVE POSITION
        CALL    FSBR64,#INBUFF,IBUFND,#COLON
        MOV     %0,C2           ;SAVE POSITION OF ':'
        BNE     ENT.C           ;IF FOUND ':' - SKIP
        .PRINT  #ERR41          ;BAD FILE ?
        JMP     PURGE
ENT.C:  SUB     #2,%0           ;BACK UP 2 CHARACTERS
        MOV     %0,P2           ;SAVE POSITION
        ;
        .PAGE
        .SBTTL  FIRST PASS OF MICROASSEMBLER
        ;
        CLR     PASCNT          ;INDICATE FIRST PASS
        MOV     #" 0,ERCNT1     ;RESET ERROR COUNTS
        MOV     #" 0,ERCNT2
        MOVB    #1,LSTAT+1      ;INDICATE TBLCON
D25.:   JSR     %7,D50.         ;SCAN STRING
        CMP     #20,%0          ;EN: ?
        BEQ     D25.I           ;THIS PASS FINISHED
        CMP     %0,#21          ;**: ?
```

```
        BLT     D25.            ;NOT CONTROL - THEN SCAN AGAIN
        BGT     D25.C           ;CONTROL - THEN BRANCH AHEAD
        JSR     %7,ERRCHK       ;UPDATE TO THIS POINT
        TST     ME              ;FIRST ENTRY AFTER . SET ?
        BNE     D25.A           ;NO - GO UPDATE MPC
        INC     ME              ;ELSE MAKE SECOND
        BR      D25.B           ;DONOT UPDATE MPC
D25.A:  INC     MPC             ;UPDATE COUNTER
D25.B:  JSR     %7,EVAL         ;EVALUATE NEW TERM
        BR      D25.            ;LOOP SOME MORE
D25.C:  JSR     %7,ERRCHK       ;UPDATE TO THIS POINT
        TST     ME              ;NEED MPC UPDATE ?
        BEQ     D25.D           ;NO - SKIP
        INC     MPC             ;ELSE UPDATE MPC
        CLR     ME              ;RESET FLAG
D25.D:  JSR     %7,EVAL         ;EVALUATE CONTROLS
        BR      D25.            ;AND SCAN SOME MORE
D25.I:  JSR     %7,ERRCHK       ;UPDATE TO THIS POINT
        ;
        .PAGE
        .SBTTL  BETWEEN PASSES SETUP
        ;
        JSR     %7,D21.         ;RESET PARAMETERS
        MOV     ERCNT2,ERCNT1   ;MOV COUNT TO PASS 1
        MOV     #" 0,ERCNT2     ;RESET ERROR COUNT
        .WAIT   #1              ;NEED LIST ?
        BCS     D25.T           ;IF NOT - SKIP
        MOVB    #2,LSTAT+1      ;INDICATE SBTTL
        MOV     #SBTTL1,%1      ;MOVE SBTTL1 TO SBTTL
        MOV     #SBTTL,%2
        MOV     #40.,%0         ;BUFFER IS 80. CHARS
D25.R:  MOV     (%1)+,(%2)+
        DEC     %0              ;MORE ?
        BGT     D25.R           ;LOOP UNTIL DONE
        BITB    #12,LSTAT       ;BINARY OR SOURCE LISTING ?
        BEQ     D25.S           ;IF NOT - SKIP HEADER
        CLR     LINCNT          ;NEW PAGE
        CALL    PNTLST,#HDR
        BITB    #2,LSTAT        ;SOURCE LISTING ?
        BEQ     D25.S           ;IF NOT - SKIP SBTTL
        CALL    PNTLST,#SBTTL
D25.S:  CALL    PNTLST,#EHDR
D25.T:  MOV     #INBUFF,.IBUFF  ;SET LOCATION
        JSR     %7,SREAD        ;READ FIRST BLOCK
        MOV     #IBUFF,.IBUFF   ;RESET POINTER
        MOV     TWORDS,%0       ;BYTES READ
        ADD     #INBUFF,%0      ;COMPUTE LAST POSITION
        MOV     %0,IBUFND       ;SAVE POSITION
        CALL    FSBR64,#INBUFF,IBUFND,#COLON
        MOV     %0,C2           ;SAVE POSITION
        SUB     #2,%0
        MOV     %0,P2           ;SAVE
        ;
        .PAGE
        .SBTTL  SECOND PASS OF MICROASSEMBLER
        ;
        INC     PASCNT          ;INDICATE SECOND PASS
D25.J:  JSR     %7,D50.         ;FIND SEGMENT
        MOV     C1,LBND         ;RESET POINTER FOR LIST
        CMP     %0,#20          ;THIS AN EN: ?
        BEQ     PASFIN          ;PASS FINISHED
        CMP     %0,#21          ;**: ?
        BLT     D25.L           ;NOT CONTROLS - GO EVALUATE
        BGT     D25.M           ;CONTROLS - BRANCH
        JSR     %7,OUTCHK       ;ELSE **:
        TST     ME              ;FIRST CHECK ?
        BNE     D25.K           ;IF NOT - GO UPDATE MPC
        INC     ME              ;ELSE MAKE SECOND
        BR      D25.J           ;EVALUATION WAS DONE ON PASS 1
D25.K:  INC     MPC             ;UPDATE COUNTER
        BR      D25.J           ;SCAN SOME MORE
D25.L:  JSR     %7,EVAL         ;EVALUATE TERMS
        BR      D25.J           ;SCAN SOME MORE
D25.M:  JSR     %7,OUTCHK       ;CHECK TO THIS POINT
        TST     ME              ;NEED MPC UPDATE ?
        BEQ     D25.N           ;IF NOT - SKIP
```

```
        INC     MPC             ;ELSE UPDATE MPC
        CLR     ME              ;RESET FLAG
D25.N:  JSR     %7,EVAL         ;EVALUATE THIS TERM
        BR      D25.J           ;SCAN SOME MORE
        ;
        ;PASS  FINISHED CHECKS
        ;
PASFIN: ADD     #4,LBND         ;GET PAST EN: STATEMENT
        JSR     %7,OUTCHK       ;CHECK EVERY THING
        JSR     %7,PNTSMT       ;NOW PRINT SYMBOL TABLE
ERLIST: .WAIT   #1              ;NEED LIST ?
        BCS     OBJFIN          ;IF NOT - SKIP
        CALL    PNTLST,#ERRSET  ;<CR><LF>
        CALL    PNTLST,#PASS1
        CALL    PNTLST,#PASS2
        CALL    PNTLST,#ERRSET
        CALL    PNTLST,#CSTRNG
        CALL    PNTLST,#ERRSET
        ;
OBJFIN: .WAIT   #0              ;FILE OPEN ?
        BCS     LSTFIN          ;IF NOT - SKIP
        TST     OWCNT           ;ANY WORDS LEFT ?
        BEQ     OBJF.A          ;IF NOT - JUST CLOSE FILE
        MOV     #OAREA,%0
        JSR     %7,WRITE        ;WRITE LAST BLOCK
OBJF.A: .CLOSE  #0              ;MAKE FILE PERMANENT
        ;
LSTFIN: .WAIT   #1              ;FILE OPEN ?
        BCS     PASEND          ;IF NOT - FINISHED
LSTF.A: MOV     LBCNT,%0        ;ANY BYTES TO TRANSFER ?
        BEQ     LSTF.C          ;IF NOT - JUST CLOSE FILE
        ASR     LBCNT           ;MAKE A WORD COUNT
        BCS     LSTF.B          ;IF EVEN - SKIP
        MOVB    #CR,LBUFF(%0)   ;ONE LAST CHAR
LSTF.B: MOV     #LAREA,%0
        JSR     %7,WRITE        ;WRITE DATA
        .WAIT   #1              ;WAIT FOR TRANSFER COMPLETE
LSTF.C: .CLOSE  #1              ;MAKE FILE PERMANENT
        ;
PASEND: .PURGE  #2              ;NO FILE
        .RCTRLO                 ;ENABLE CONSOLE
        .PRINT  #ERRSET         ;<CR><LF>
        .PRINT  #PASS1          ;LIST TO CONSOLE
        .PRINT  #PASS2
        .PRINT  #ERRSET
        .PRINT  #CSTRNG
        .PRINT  #ERRSET
        JMP     ENTER           ;GO START OVER
        ;
        .NLIST  BIN
PASS1:  .ASCII  /ERRORS DETECTED IN PASS 1  =/
ERCNT1: .ASCII  / 0/<0><0>
PASS2:  .ASCII  /ERRORS DETECTED IN PASS 2  =/
ERCNT2: .ASCII  / 0/<0><0>
        .LIST   BIN
        ;
        .PAGE
        .SBTTL  PAGE ROUTINE
        ;
.PG:    .WAIT   #1              ;NEED LIST ?
        BCS     .PGZ            ;IF NOT - SKIP
        MOVB    #SPC,PAGE+13    ;CLEAR PAGE OVERFLOW
        MOV     #PAGE+11,%0     ;ADDRESS OF PAGE NUMBER
.PGC:   INCB    (%0)            ;UPDATE PAGE COUNT
        CMPB    (%0),#':        ;DIGIT OVERFLOW ?
        BCS     .PGD            ;IF NOT - SKIP
        MOVB    #'0,(%0)        ;RESET DIGIT
        CMPB    #SPC,-(%0)      ;PRECEEDING DIGIT A SPACE ?
        BNE     .PGC            ;IF NOT - THEN GO UPDATE
        MOVB    #'0,(%0)        ;ELSE SET TO A 0
        BR      .PGC            ;AND GO UPDATE
        ;
.PGD:   JSR     %7,.SB          ;DO SUBTITLE ROUTINE
        ;
        TST     PASCNT          ;FIRST PASS ?
        BEQ     .PGZ            ;IF SO - SKIP
```

```
        BITB    #2,LSTAT        ;WANT SOURCE LISTING ?
        BEQ     .PGZ            ;IF NOT - SKIP
        CLR     LINCNT          ;CLEAR LINE COUNT
        CALL    PNTLST,#HDR     ;PRINT HEADER
        CALL    PNTLST,#SBTTL   ;AND SUBTITLE
        CALL    PNTLST,#EHDR
.PGZ:   RTS     %7              ;FINISHED
        ;
        .PAGE
        .SBTTL  SUB TITLE ROUTINE
        ;
.SB:    .WAIT   #1              ;LIST NEEDED ?
        BCS     .SBZ            ;IF NOT - SKIP
        CALL    GETASC,#SBTTL,#79.
        CMP     #SBTTL,%0       ;SUBTITLE FOUND ?
        BEQ     .SBZ            ;IF NOT - DO NOTHING
        CLRB    (%0)            ;TERMINATE LIST
        TST     PASCNT          ;FIRST PASS ?
        BNE     .SBZ            ;IF NOT - SKIP
        TST     SBENTR          ;FIRST ENTRY ?
        BNE     .SBA            ;IF NOT - SKIP
        INC     SBENTR          ;NOT FIRST
        CALL    GETASC,#SBTTL1,#79.
        CLRB    (%0)
        BIT     #1,LSTAT        ;WANT TBLCON ?
        BEQ     .SBA            ;IF NOT - SKIP
        CLR     LINCNT          ;CLEAR LINE COUNT
        CALL    LSTOUT,#HDR,#PAGE-1
        CALL    PNTLST,#EHDR
        CALL    PNTLST,#TBLCON
        CALL    PNTLST,#EHDR
.SBA:   BIT     #1,LSTAT        ;WANT TBLCON ?
        BEQ     .SBZ            ;IF NOT - SKIP
        CALL    LSTOUT,#PAGE,#PAGE+15
        CALL    PNTLST,#SBTTL
.SBZ:   RTS     %7              ;FINISHED
        ;
        .PAGE
        .SBTTL  TITLE ROUTINE
        ;
.TT:    CALL    GETASC,#TITLE,#26.
.TTA:   CMP     %0,#TITLE+24.   ;UP TO END ?
        BCC     .TTZ            ;IF SO - SKIP
        MOVB    #SPC,(%0)+
        BR      .TTA            ;LOOP UNTIL FINISHED
.TTZ:   RTS     %7              ;FINISHED
        ;
        .PAGE
        .SBTTL  GET ASCII STRING ROUTINE
        ;
GETASC: SAVL1                   ;%1 IS BUFFER LOCATION
        SAVL2                   ;%2 IS MAXIMUM CHARS
        SAV3                    ;STRING BOUND
        SAV4                    ;STRING BOUND
        MOV     P1,%3           ;LOWER BOUND
        MOV     P2,%4           ;UPPER BOUND
        DEC     %4
GTSC.A: INC     %3              ;POINT TO FIRST CHARACTER
        CMP     %4,%3           ;SCAN FINISHED ?
        BCS     GTSC.Z          ;IF SO - LEAVE
        MOVB    (%3),%0         ;GET CHARACTER
        CMPB    #CTRL.I,%0      ;^I, SP, CR, OR LF ?
        BEQ     GTSC.A          ;SKIP THESE CHARACTERS
        CMPB    #SPC,%0
        BEQ     GTSC.A
        CMPB    #CR,%0
        BEQ     GTSC.A
        CMPB    #LF,%0
        BEQ     GTSC.A
        INC     %4
GTSC.B: MOVB    -(%4),%0
        CMPB    #CTRL.I,%0
        BEQ     GTSC.B
        CMPB    #SPC,%0
        BEQ     GTSC.B
        CMPB    #CR,%0
```

```
        BEQ     GTSC.B
        CMPB    #LF,%0
        BEQ     GTSC.B
GTSC.C: CMP     %4,%3           ;END OF SCAN ?
        BCS     GTSC.Z          ;IF SO - SKIP
        MOVB    (%3)+,(%1)+     ;PLACE BYTE
        DEC     %2              ;SPACE LEFT ?
        BGT     GTSC.C          ;LOOP IF SO
GTSC.Z: MOV     %1,%0           ;SAVE NEXT POSITION
        RES4
        RES3
        RES2
        RES1
        RTS     %5              ;FINISHED
        ;
```

```
        .PAGE
        .SBTTL  OUTPUT DATA SCANNER FOR FILE GENERATION
        ;
ERRCHK: CMP     ERRLOG,#ERRTBL  ;ANY ERRORS ?
        BNE     OUTC.C          ;IF SO - GO PROCESS
        JMP     OUTC.F          ;GO FINISH
OUTCHK: JSR     %7,D45.         ;GO CHECK ALL DEFAULTS & CONFLICTS
        ;
        .WAIT   #0              ;NEED DATA ?
        BCS     OUTC.C          ;IF NOT - SKIP TRANSFERS
        BIT     #MCODE,CDSTAT   ;MCODE DEFINED ?
        BEQ     OUTC.A          ;IF NOT - SKIP
        MOV     MPC,M.MPC       ;CURRENT COUNTER
        CALL    OBJOUT,#8.,#MBASE
OUTC.A: BIT     #ICODE,CDSTAT   ;ICODE DEFINED ?
        BEQ     OUTC.B          ;IF NOT - SKIP
        MOV     MPC,I.MPC       ;CURRENT MPC
        CALL    OBJOUT,#4,#IBASE
OUTC.B: BIT     #VCODE,CDSTAT   ;VECTORS DEFINED
        BEQ     OUTC.C          ;IF NOT - SKIP
        MOV     VECT,%0         ;# DEFINED
        ASL     %0              ;2 WORDS PER ENTRY
        ADD     #2,%0           ;VBASE AND VECT WORDS
        CALL    OBJOUT,%0,#VBASE
        ;
OUTC.C: .WAIT   #1              ;NEED LIST FILE
        BCS     OUTC.D          ;IF NOT - SKIP
        BICB    #200,LSTAT      ;CLEAR ERROR FLAG
        CMP     ERRLOG,#ERRTBL  ;ANY ERRORS ?
        BEQ     OUT.C0          ;IF NOT SKIP
        BISB    #200,LSTAT      ;SET ERROR FLAG
OUT.C0: JSR     %7,LSTERR       ;PRINT ERRORS - IF ANY
        BIT     #210,LSTAT      ;WANT BINARY LIST ?
        BEQ     OUT.C4          ;IF NOT - SKIP
        BIT     #MCODE,CDSTAT   ;MCODE DEFINED ?
        BEQ     OUT.C1          ;IF NOT - SKIP
        MOV     MPC,M.MPC       ;SET UP PC
        CALL    PRINTV,#M.MPC,#M.SPC,#6
        CALL    LSTOUT,#ASCBUF+6,%0
OUT.C1: BIT     #ICODE,CDSTAT   ;ICODE DEFINED ?
        BEQ     OUT.C2          ;IF NOT - SKIP
        MOV     MPC,I.MPC       ;SET UP PC
        CALL    PRINTV,#I.MPC,#I.SPC,#2
        CALL    LSTOUT,#ASCBUF+6,%0
OUT.C2: BIT     #VCODE,CDSTAT   ;VECTORSS DEFIED ?
        BEQ     OUT.C4          ;IF NOT - SKIP
        MOV     #VECTV,%2       ;ADDRESS
        MOV     VECT,%1         ;NUMBER OF TERMS
OUT.C3: CALL    PRINTV,%2,#V.SPC,#1
        ADD     #4,%2           ;UPDATE ADDRESS
        CALL    LSTOUT,#ASCBUF+6,%0
        DEC     %1              ;FINISHED ?
        BGT     OUT.C3          ;LOOP UNTIL FINISHED
OUT.C4: BIT     #202,LSTAT      ;WANT SOURCE LIST ?
        BEQ     OUTC.F          ;IF NOT - SKIP
        JSR     %7,SCNLST       ;SCAN LIST
        CALL    LSTOUT,%1,%2
        BR      OUTC.F          ;GO FINISH
        ;
OUTC.D: CMP     ERRLOG,#ERRTBL  ;ANY ERRORS ?
        BEQ     OUTC.F          ;IF NOT - FINISHED
        JSR     %7,PNTERR       ;ELSE PRINT TO CONSOLE
        BIT     #MCODE,CDSTAT   ;MCODE DEFINED ?
        BEQ     OUT.D1          ;IF NOT - SKIP
        MOV     MPC,M.MPC       ;SET UP PC
        CALL    PRINTV,#M.MPC,#M.SPC,#6
        .PRINT  #ASCBUF+6
OUT.D1: BIT     #ICODE,CDSTAT   ;ICODE DEFINED ?
        BEQ     OUT.D2          ;IF NOT - SKIP
        MOV     MPC,I.MPC       ;SET UP PC
        CALL    PRINTV,#I.MPC,#I.SPC,#2
        .PRINT  #ASCBUF+6
OUT.D2: BIT     #VCODE,CDSTAT   ;VECTORS DEFINED ?
        BEQ     OUT.D4          ;IF NOT SKIP
        MOV     #VBASE,%2       ;SET ADDRESS
        MOV     VECT,%1         ;NUMBER OF TERMS
```

```
OUT.D3: CALL    PRINTV,%2,#V.SPC,#1
        ADD     #4,%2           ;UPDATE ADDRESS
        .PRINT  #ASCBUF+6
        DEC     %1              ;FINISHED ?
        BGT     OUT.D3          ;LOOP UNTIL FINISHED
OUT.D4: JSR     %7,SCNLST
OUTC.E: .TTYOUT (%1)+           ;SEND TO THE CONSOLE
        CMP     %2,%1           ;FINISHED ?
        BCC     OUTC.E          ;LOOP UNTIL FINISHED
        .TTYOUT #CR
        .TTYOUT #LF
OUTC.F: JSR     %7,D20.B        ;CLEAR LINE RESULTS
        MOV     LBND,FBND       ;RESET LISTING POINTERS
        RTS     %7              ;FINISHED
        ;
        ;SCAN LIST ROUTINE
        ;
SCNLST: MOV     FBND,%1         ;GET FIRST BYTE ADDRESS
        MOV     LBND,%2
SCNL.A: DEC     %2              ;UPDATE ADDRESS
        CMP     %2,%1           ;END OF STRING ?
        BLO     SCNL.B          ;IF SO - SKIP
        CMPB    (%2),#SPC       ;SKIP SPACES AND ^I'S
        BEQ     SCNL.A
        CMPB    (%2),#CTRL.I
        BEQ     SCNL.A
SCNL.B: MOV     %2,LBND         ;REDEFINE LBND
        INC     LBND
        RTS     %7              ;FINISHED
        ;
        .PAGE
        .SBTTL  PRINTV ROUTINE
        ;
        ;       CALL    PRINTV,ADDRESS,ASCIIMASK,TERMS
        ;
PRINTV: SAVL1           ;%1 IS ADDRESS OF VALUE
        SAVL2           ;%2 IS ASCII MASK
        SAVL3           ;%3 IS # OF TERMS
        SAV4            ;%4 IS COUNTER
        SAV5            ;%5 IS ASCII BUFFER ADDRESS
        MOV     #ASCBUF,%5
        JSR     %7,ICONV6       ;MAKE 6 CHAR STRING
        MOV     %2,(%5)+        ;PLACE ASCII MASK
        MOV     #SPSP,(%5)+     ;TWO SPACES
PTV.A:  JSR     %7,ICONV6       ;ASCII STRING
        DEC     %3              ;MORE TO DO ?
        BGT     PTV.A           ;IF SO - LOOP
        MOV     CRLFCH,(%5)+    ;<CR><LF>
        MOVB    #200,(%5)
        MOV     %5,%0
        DEC     %0
        RES5
        RES4
        RES3
        RES2
        RES1
        RTS     %5              ;RETURN AND CLEAR STACK
        ;
ICONV6: SAV3                    ;SAVE %3
        MOV     #SPSP,(%5)+     ;TWO SPACES
        MOV     #SPSP,(%5)+
        MOV     (%1)+,%3        ;GET INTEGER VALUE
        MOV     #6,%4           ;6 CHARS TO COMPUTE
        CLR     %0              ;INITIALIZE VALUE
        BR      SH1             ;SKIP TO ENTRY
SH3:    CLR     %0
        ASL     %3
        ROL     %0
        ASL     %3
        ROL     %0
SH1:    ASL     %3
        ROL     %0
        ADD     #60,%0          ;COMPUTE CHARACTER
        MOVB    %0,(%5)+        ;PLACE CHARACTER
        DEC     %4              ;FINISHED ?
        BGT     SH3             ;LOOP UNTIL ALL 6 CHARS DONE
```

```
        RES3                    ;RESTORE %3
        RTS     %7              ;FINISHED
        ;
        .PAGE
        .SBTTL  INPUT FILE HANDLER
        ;
GETINP: SAV1
        SAV2
        MOV     #INBUFF,%1      ;STRING BUFFER
        MOV     FBND,%2         ;CURRENT POINTER
        MOV     %2,%0           ;BEGINNING OF CURRENT INSTRUCTION
        SUB     #INBUFF,%0      ;COMPUTE RELOCATION CONSTANT
        BEQ     GET.C           ;IF =0 - SKIP MOVING
        SUB     %0,C1           ;UPDATE POINTERS
        SUB     %0,C2
        SUB     %0,P1
        SUB     %0,P2
        SUB     %0,FBND
        SUB     %0,LBND
        MOV     IBUFND,%0       ;COMPUTE # OF CHARS TO MOVE
        SUB     %2,%0
        BR      GET.B           ;GO START
GET.A:  MOVB    (%2)+,(%1)+     ;RELOCATE STRING
GET.B:  DEC     %0              ;MORE TO DO ?
        BGE     GET.A           ;IF SO - LOOP
        MOV     %1,IBUFND       ;SAVE END POSITION
        ;
        ;GO READ NEW BLOCK OF STRING DATA
        ;
GET.C:  JSR     %7,CREAD
        TST     RDERRF          ;AN ERROR ?
        BEQ     GET.D           ;IF NOT - SKIP
        ERROR   #42.            ;/NO EN:: FOUND/
        MOV     IBUFND,%1       ;ADDRESS OF BUFFER
        MOVB    #SPC,(%1)+      ;PLACE ' EN::' IN BUFFER
        MOVB    #'E,(%1)+
        MOVB    #'N,(%1)+
        MOVB    #':,(%1)+
        MOVB    #':,(%1)+
        BR      GET.G           ;GO FINISH
GET.D:  MOV     TWORDS,%0       ;BYTES READ
        MOV     IBUFND,%1       ;LAST ADDRESS
        MOV     #IBUFF,%2       ;ADDRESS OF READ DATA
        BR      GET.F           ;GO START MOVING
GET.E:  MOVB    (%2)+,(%1)+     ;MOVE BYTE
        CMP     %1,#INBUFF+BFLEN        ;END OF BUFFER
        BCS     GET.F           ;IF NOT - SKIP
        .PRINT  #ERR47          ;INPUT BUFFER FULL
        MOV     SAVESP,%6       ;RESET STACK
        JMP     RENTER          ;TRY RESTART
GET.F:  DEC     %0              ;MORE TO MOVE ?
        BGE     GET.E           ;IF SO - LOOP
GET.G:  MOV     %1,IBUFND       ;SAVE LAST POSITION
        RES2
        RES1
        RTS     %7              ;FINISHED
        ;
        ;BASIC READ BLOCK ROUTINE
        ;
SREAD:  MOVB    #3,.ICHAN       ;SET FOR FIRST INPUT CHANNEL
        CLR     .IBLK           ;AND BLOCK 0
CREAD:  MOV     #.IAREA,%0      ;SET UP FOR .READW
        .READW
        MOVB    @#52,RDERRF     ;SAVE ERROR FLAG
        BCS     RD.A            ;BRANCH ON ERROR
        INC     .IBLK           ;SET TO READ NEXT BLOCK
        ASL     %0              ;MAKE BYTE COUNT
        MOV     %0,TWORDS       ;SAVE COUNT
        ADD     .IBUFF,%0       ;FIND BUFFER LOCATION
CR.A:   TSTB    -(%0)           ;SCAN DATA FOR PARTIAL READ
        BNE     CR.B            ;IF NOT A 'NULL' - GOOD DATA
        DEC     TWORDS          ;ELSE ONE LESS CHARACTER IN READ DATA
        BGT     CR.A            ;SCAN UNTIL BUFFER SEARCHED
        BR      CREAD           ;IF BUFFER EMPTY - READ ANOTHER
CR.B:   RTS     %7              ;FINISHED
RD.A:   CMPB    RDERRF,#1       ;WHAT KIND OF ERROR
```

```
        BEQ     RD.B            ;HARD READ ERROR
        BGT     RD.C            ;CHANNEL INACTIVE
        INCB    .ICHAN          ;ELSE HIT EOF ON THIS FILE
        CLR     .IBLK           ;SO SET UP FOR NEXT INPUT FILE
        BR      CREAD           ;TRY NEXT
RD.B:   .PRINT  #ERR43
RD.C:   CLR     TWORDS          ;NOTHING READ
        RTS     %7              ;FINISHED
        ;
        .PAGE
        .SBTTL  OBJECT OUTPUT HANDLER
        ;
OBJOUT: SAVL1
        SAVL2
        SAV3
OBJ.A:  MOV     OWCNT,%3        ;GET CURRENT COUNT
        ASL     %3              ;MAKE BYTE COUNT
        MOV     (%2)+,OBJBUF(%3)        ;STORE WORD
        INC     OWCNT           ;UPDATE COUNT
        CMP     OWCNT,#256.     ;IS IT FULL ?
        BCS     OBJ.B           ;IF NOT - SKIP
        MOV     #OAREA,%0       ;SET UP FOR .WRITW
        JSR     %7,WRITE        ;GO WRITE BLOCK
        CLR     OWCNT           ;RESET WORD COUNT
        INC     OBLK            ;NEXT BLOCK
OBJ.B:  DEC     %1              ;MORE TO TRANSFER ?
        BNE     OBJ.A           ;IF SO - LOOP
        RES3
        RES2
        RES1
        RTS     %5              ;CLEAR STACK AND RETURN
        ;
        ;BASIC WRITE BLOCK ROUTINE
        ;
WRITE:  .WRITW
        BCC     WRT.C           ;NO ERROR - SKIP
        TST     @#52            ;CHECK ERROR
        BNE     WRT.A           ;BRANCH ON HARD ERROR
        .PRINT  #ERR44          ;DEVICE FULL
        BR      WRT.B
WRT.A:  .PRINT  #ERR45          ;HARD ERROR
WRT.B:  MOV     SAVESP,%6       ;RESET STACK
        JMP     RENTER          ;REMOVE FILES AND RESTART
WRT.C:  RTS     %7              ;RETURN IN LINE
        ;
        .PAGE
        .SBTTL  LIST OUTPUT HANDLER
        ;
LSTOUT: SAVL1
        SAVL2
        SAV3
        BR      LST.C           ;GO CHECK ENTRY VALUES
LST.A:  MOV     LBCNT,%3        ;GET BYTE COUNT
        MOVB    (%1),LSTBUF(%3) ;MOV BYTE TO BUFFER
        INC     LBCNT           ;UPDATE COUNT
        CMP     LBCNT,#512.     ;FULL ?
        BCS     LST.B           ;IF NOT - SKIP
        ASR     LBCNT           ;MAKE WORD COUNT
        MOV     #LAREA,%0       ;SET UP FOR .WRITW
        JSR     %7,WRITE        ;GO WRITE BLOCK
        CLR     LBCNT           ;CLEAR BYTE COUNT
        INC     LBLK            ;NEXT BLOCK
LST.B:  CMPB    #LF,(%1)+       ;END OF LINE ?
        BNE     LST.C           ;IF NOT - SKIP
        INC     LINCNT          ;UPDATE LINE COUNT
        CMP     LINCNT,#LPAGE   ;ENOUGH LINES YET ?
        BCS     LST.C           ;IF NOT - SKIP
        CLR     LINCNT          ;STARTING OVER
        MOVB    #'+,PAGE+13     ;PAGE OVERFLOW
        CALL    LSTOUT,#HDR,#PAGE-1
        BITB    #1,LSTAT+1      ;PRINTING TBLCON ?
        BEQ     LST.B1          ;IF NOT - SKIP
        BITB    #1,LSTAT        ;WANT TBLCON ?
        BEQ     LST.B1          ;IF NOT - SKIP
        CALL    PNTLST,#EHDR
        CALL    PNTLST,#TBLCON
```

```
LST.B1: BITB    #2,LSTAT+1      ;PRINTING SBTTL ?
        BEQ     LST.B2          ;IF NOT - SKIP
        BITB    #2,LSTAT        ;WANT SOURCE ?
        BEQ     LST.B2          ;IF NOT - SKIP
        CALL    PNTLST,#PAGE
        CALL    PNTLST,#SBTTL
LST.B2: BITB    #4,LSTAT+1      ;PRINTING TBLSYM ?
        BEQ     LST.B3          ;IF NOT - SKIP
        BITB    #4,LSTAT        ;WANT TBLSYM ?
        BEQ     LST.B3          ;IF NOT - SKIP
        CALL    PNTLST,#EHDR
        CALL    PNTLST,#TBLSYM
LST.B3: CALL    PNTLST,#EHDR
LST.C:  CMP     %2,%1           ;FINISHED ?
        BCC     LST.A           ;IF NOT - LOOP
        RES3
        RES2
        RES1
        RTS     %5              ;CLEAR STACK AND RETURN
        ;
        .PAGE
        .SBTTL  LIST OUTPUT ERROR HANDLER
        ;
LSTERR: CMP     ERRLOG,#ERRTBL  ;ANY ERRORS ?
        BNE     LSTR.A          ;IF SO - SKIP
        RTS     %7              ;ELSE FINISHED
LSTR.A: CALL    PNTLST,#ERRSET
        SAV1
        MOV     ERRLOG,%1       ;GET ADDRESS OF LAST ERROR
LSTR.B: CMP     #ERRTBL,%1      ;ANY MORE ?
        BEQ     LSTR.C          ;IF NOT - SKIP
        CALL    PNTLST,#ERRCDE  ;**** ERROR
        MOVB    -(%1),%0        ;GET CODE NUMBER
        ASL     %0              ;MAKE WORD OFFSET
        CALL    PNTLST,MSGTBL(%0)
        BR      LSTR.B          ;LOOP FOR MORE
LSTR.C: MOV     #ERRTBL,ERRLOG  ;RESET ERROR STACK
        RES1
        RTS     %7              ;FINISHED
        ;
PNTLST: SAVL1
        SAV2
        MOV     %1,%2           ;SAVE FIRST BYTE ADDRESS
        TSTB    (%1)            ;=0 ?
        BEQ     PNTL.B          ;IF SO - ONLY NEED CRLF
PNTL.A: TSTB    (%1)+           ;SEARCH FOR - OR 0
        BGT     PNTL.A          ;SCAN UNTIL FOUND
        SUB     #2,%1           ;LAST CHARACTER BEFORE - OR 0
        CALL    LSTOUT,%2,%1
        TSTB    1(%1)           ;0 OR -
        BMI     PNTL.C          ;IF - SKIP
PNTL.B: CALL    LSTOUT,#CRLFCH,#CRLFCH+1
PNTL.C: RES2
        RES1
        RTS     %5              ;CLEAR STACK AND RETURN
        ;
        .PAGE
        .SBTTL  BUILD TIME ROUTINE
        ;
BLDTIM: .GTIM   #T.AREA,#TMH
        SAVL1
        SAV2
        MOV     #HRTBL,%2       ;DATA TABLE
        MOV     #2,%0           ;THREE TERMS
        BR      BLDT.B          ;GO TO ENTRY
BLDT.A: MOVB    #':,(%1)+       ;PLACE ':'
BLDT.B: MOVB    #'0,(%1)+       ;SET TIMES TO 00
        MOVB    #'0,(%1)+
        BR      BLDT.D          ;SKIP TO ENTRY
BLDT.C: INCB    -1(%1)
        CMPB    -1(%1),#':      ;OVERFLOWED ?
        BCS     BLDT.D          ;IF NOT - SKIP
        MOVB    #'0,-1(%1)      ;RESET COUNT
        INCB    -2(%1)
BLDT.D: SUB     6(%2),TML       ;CHECK FOR DIGIT
        SBC     TMH
```

```
        SUB     (%2),TMH
        BPL     BLDT.C          ;LOOP IF RESULT STILL PLUS
        ADD     6(%2),TML       ;RESTORE TICKS
        ADC     TMH             ;ADD CARRY
        ADD     (%2)+,TMH
        DEC     %0              ;FINISHED ALL TERMS ?
        BGE     BLDT.A          ;LOOP UNTIL FINISHED
        RES2
        RES1
        RTS     %5              ;FINISHED
        ;
        .PAGE
        .SBTTL  DATE EVALUATION
        ;
BLDDAT: SAVL1
        SAV2
        .DATE                   ;GET DATE INFO
        TST     %0              ;DATE SET ?
        BEQ     BLDD.F          ;IF NOT - DO NOTHING
        ADD     #10,%1          ;END OF DATE STRING
        MOV     %0,%2           ;COPY INFO
        BIC     #177740,%2      ;CLEAR ALL BUT YEAR
        MOVB    #'2,(%1)        ;PRESET
        MOVB    #'7,-1(%1)
        BR      BLDD.B          ;GO TO ENTRY
BLDD.A: INCB    (%1)            ;UPDATE YEAR
        CMPB    (%1),#':        ;DIGIT OVERFLOW ?
        BCS     BLDD.B          ;IF NOT - SKIP
        MOVB    #'0,(%1)        ;RESET DIGIT
        INCB    -1(%1)
BLDD.B: DEC     %2              ;FINISHED ?
        BGE     BLDD.A          ;LOOP UNTIL FINISHED
        MOVB    #'-,-2(%1)      ;PLACE '-'
        SUB     #10,%1
        ASH     #-5,%0          ;SHIFT DAY INTO PLACE
        MOV     %0,%2           ;MAKE COPY
        BIC     #177740,%2      ;MASK ALL BUT DAY
        MOVB    #SPC,(%1)+
        MOVB    #'0,(%1)
        BR      BLDD.E          ;GO TO ENTRY
BLDD.C: INCB    (%1)            ;UPDATE DAY
        CMPB    (%1),#':        ;DIGIT OVERFLOW ?
        BCS     BLDD.E          ;IF NOT - SKIP
        MOVB    #'0,(%1)        ;RESET DIGIT
        CMPB    #SPC,-1(%1)
        BNE     BLDD.D
        MOVB    #'0,-1(%1)
BLDD.D: INCB    -1(%1)
BLDD.E: DEC     %2              ;FINISHED ?
        BGE     BLDD.C          ;LOOP UNTIL FINISHED
        MOVB    #'-,1(%1)
        ADD     #2,%1
        ASH     #-5,%0          ;SHIFT MONTH INTO POSITION
        BIC     #177740,%0      ;MASK OUT ALL BUT MONTH
        DEC     %0
        MOV     %0,%2
        ADD     %0,%2
        ADD     %0,%2           ;3*[%0-1]
        ADD     #DATTBL,%2      ;GET TABLE ADDRESS
        MOVB    (%2)+,(%1)+     ;MOV INTO STRING
        MOVB    (%2)+,(%1)+
        MOVB    (%2),(%1)
BLDD.F: RES2
        RES1
        RTS     %5              ;FINISHED
        ;
        .NLIST  BIN
DATTBL: .ASCII  /JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDEC/
        .LIST   BIN
        .EVEN
        ;
        .PAGE
        .SBTTL  SYMBOL TABLE PRINTING
        ;
PNTSMT: .WAIT   #1              ;LIST OUTPUT ?
        BCC     PNTS.A          ;IF SO - SKIP
```

```
        RTS     %7              ;ELSE FINISHED
PNTS.A: BIT     #4,LSTAT        ;CHECK IF WANTED
        BNE     PNTS.B          ;IF SO - SKIP
        RTS     %7              ;ELSE FINISHED
PNTS.B: SAV1
        SAV2
        SAV3
        SAV4
        SAV5
        MOVB    #4,LSTAT+1      ;INDICATE TBLSYM
        CLR     LINCNT          ;CLEAR LINE COUNTER
        CALL    LSTOUT,#HDR,#PAGE-1     ;PRINT HEADER FOR SYMBOL TABLE
        CALL    PNTLST,#EHDR
        CALL    PNTLST,#TBLSYM
        CALL    PNTLST,#EHDR
        MOV     #SCNLOW,SCNTMP  ;SET LOWEST SYMBOL
PNTS.C: MOV     #3,%2           ;SYMBOLS PER LINE
PNTS.D: CALL    SCNJLG,SCNTMP
        TST     %0              ;A SYMBOL ?
        BEQ     PNTS.Z          ;IF NOT - SKIP
PNTS.E: JSR     %7,PNTSYM       ;GO PRINT SYMBOL AND LOCATION
        DEC     %2              ;MORE PER LINE ?
        BLE     PNTS.F          ;IF NOT - SKIP
        CALL    LSTOUT,#SCNLOW,#SCNLOW+11
        BR      PNTS.D          ;LOOP FOR MORE
PNTS.F: CALL    LSTOUT,#CRLFCH,#CRLFCH+1
        BR      PNTS.C          ;LOOP FOR MORE
PNTS.Z: CALL    LSTOUT,#CRLFCH,#CRLFCH+1
        MOV     #SYMTBL,%0      ;POINT TO MPC
        JSR     %7,PNTSYM
        CALL    LSTOUT,#CRLFCH,#CRLFCH+1
        RES5
        RES4
        RES3
        RES2
        RES1
        RTS     %7              ;FINISHED
        ;
PNTSYM: MOV     %0,%1           ;SAVE LOCATION
        ADD     #7,%1           ;END OF SYMBOL
        CALL    LSTOUT,%0,%1
        INC     %1              ;POINT TO LOCATIOON
        MOV     #SCNVAL,%5      ;RESULT ADDRESS
        JSR     %7,ICONV6       ;GENERATE ASCII
        CALL    LSTOUT,#SCNVAL,#SCNVAL+11
        RTS     %7              ;FINISHED
        ;
SCNJLG: SAV1
        SAV2
        SAV3
        SAV4
        CLR     %0
        CLR     SCNTMP          ;CLEAR TEMP
        MOV     #SYMTBL+12,SCNTM1       ;FIRST SYMBOL PAST MPC
SCNJ.A: MOV     SCNTM1,%3       ;GET CURRENT ADDRESS
        MOV     2(%5),%1        ;GET POINTER FROM CALLER
        TST     (%3)            ;END OF TABLE ?
        BEQ     SCNJ.Z          ;IF SO NOTHING
        CMPB    (%3)+,(%1)+     ;ONLY WANT STRINGS LARGER
        BLO     SCNJ.B          ;LOWER
        BHI     SCNJ.C          ;HIGHER
        CMPB    (%3)+,(%1)+
        BLO     SCNJ.B
        BHI     SCNJ.C
        CMPB    (%3)+,(%1)+
        BLO     SCNJ.B
        BHI     SCNJ.C
        CMPB    (%3)+,(%1)+
        BLO     SCNJ.B
        BHI     SCNJ.C
        CMPB    (%3)+,(%1)+
        BLO     SCNJ.B
        BHI     SCNJ.C
        CMPB    (%3)+,(%1)+
        BLO     SCNJ.B
        BHI     SCNJ.C
```

```
        CMPB    (%3)+,(%1)+
        BLO     SCNJ.B
        BHI     SCNJ.C
        CMPB    (%3)+,(%1)+
        BHI     SCNJ.C
SCNJ.B: ADD     #12,SCNTM1      ;UPDATE TABLE ADDRESS
        BR      SCNJ.A          ;CHECK ANOTHER
SCNJ.C: MOV     SCNTMP,%4       ;DEFINED ?
        BNE     SCNJ.D          ;IF SO - SKIP
        MOV     SCNTM1,%0       ;%0 IS CURRENT TERM
        MOV     %0,SCNTMP       ;SAVE POINTER
        BR      SCNJ.B
SCNJ.D: MOV     SCNTM1,%3       ;GET ADDRESS
        CMPB    (%3)+,(%4)+     ;ONLY WANT STRINGS LOWER
        BHI     SCNJ.B          ;HIGHER
        BLO     SCNJ.E          ;LOWER
        CMPB    (%3)+,(%4)+
        BHI     SCNJ.B
        BLO     SCNJ.E
        CMPB    (%3)+,(%4)+
        BHI     SCNJ.B
        BLO     SCNJ.E
        CMPB    (%3)+,(%4)+
        BHI     SCNJ.B
        BLO     SCNJ.E
        CMPB    (%3)+,(%4)+
        BHI     SCNJ.B
        BLO     SCNJ.E
        CMPB    (%3)+,(%4)+
        BHI     SCNJ.B
        BLO     SCNJ.E
        CMPB    (%3)+,(%4)+
        BHI     SCNJ.B
        BLO     SCNJ.E
        CMPB    (%3)+,(%4)+
        BHIS    SCNJ.B
SCNJ.E: MOV     SCNTM1,%0       ;SAVE NEW POINTER
        MOV     %0,SCNTMP       ;SAVE FOR NEXT ENTRY
        BR      SCNJ.B          ;LOOP FOR SOME MORE
SCNJ.Z: RES4
        RES3
        RES2
        RES1
        RTS     %5              ;FINISHED
        ;
        .PAGE
        .SBTTL  INITIALIZATION OF TERMS
        ;
D20.:   JSR     %7,CLRSYM       ;CLEAR SYMBOL TABLE
        JSR     %7,RSTII        ;CLEAR II & IISTNG
        MOV     #12.,%0         ;CLEAR TITLE
        MOV     #TITLE,%1
D20.A:  MOV     #SPSP,(%1)+
        DEC     %0
        BGT     D20.A
        CLR     SBENTR          ;CLEAR SUBTITLE ENTRY INDICATOR
        CLRB    SBTTL           ;CLEAR SUBTITLES
        CLRB    SBTTL1
D20.B:  CLR     CDSTAT          ;END OF LINE CLEARS
        CLR     VECT
        CLR     MW0
        CLR     MW1
        CLR     MW2
        CLR     MW3
        CLR     MW4
        CLR     MW5
        CLR     MW6
        CLR     MW7
        RTS     %7              ;FINISHED
        ;
D21.:   MOV     #SPSP,PAGE+6    ;RESET PAGE COUNT
        MOV     #30440,PAGE+10
        MOV     #SPSP,PAGE+12
        MOV     #ERRTBL,ERRLOG
        MOV     #INBUFF,FBND
        MOV     #INBUFF,LBND
```

```
        CLR     MPC
        CLR     C1
        CLR     C2
        CLR     P1
        CLR     P2
        CLR     ME
        MOV     #-1,VY
        RTS     %7              ;FINISHED
        ;
        .PAGE
        .SBTTL  END OF LINE CHECKS
        ;
D45.:   BIT     #MCODE,CDSTAT   ;NEED TO CHECK DEFAULTS ?
        BEQ     D45.G           ;IF NOT - SKIP ALL
        ;
        ;DEFAULTS FOR AR:, CC:, PC:, & BR:
        ;
        BIT     #NR,CDSTAT      ;NEED AR: ?
        BNE     D45.A           ;IF NOT - SKIP
        BIS     #100,MW0        ;SET AR: DEFAULT
D45.A:  BIT     #NS,CDSTAT      ;NEED CC: ?
        BNE     D45.B           ;IF NOT - SKIP
        BIS     #30440,MW2      ;SET CC: DEFAULT
D45.B:  BIT     #NT,CDSTAT      ;NEED PC: ?
        BNE     D45.C           ;IF NOT - SKIP
        BIS     #200,MW3        ;SET PC: DEFAULT
D45.C:  BIT     #NU,CDSTAT      ;NEED BR: ?
        BNE     D45.D           ;IF NOT - SKIP
        BIS     #100000,MW5     ;SET BR: DEFAULT
        BIS     #60,MW4
        ;
        ;CHECK FOR CONFLICTS
        ;
D45.D:  TST     MW5             ;BIT <15>=1 : NO CONFLICT
        BMI     D45.F           ;FOR SCRATCH REGISTERS & INTERRUPTS
        BIT     #MI,CDSTAT      ;INTERRUPT CONFLICT ?
        BEQ     D45.E           ;IF NOT - SKIP
        ERROR   #26.
D45.E:  BIT     #MS,CDSTAT      ;SCRATCH REGISTER CONFLICT ?
        BEQ     D45.F           ;IF NOT - SKIP
        ERROR   #27.
D45.F:  TST     MW4             ;BIT <15>=0 : NO CONFLICT
        BPL     D45.G           ;IF NONE - BRANCH
        BIT     #MP,CDSTAT      ;INST SEQ CONFLICT ?
        BEQ     D45.G           ;IF NOT - SKIP
        ERROR   #28.
D45.G:  RTS     %7              ;FINISHED
        ;
        .PAGE
        .SBTTL  SEQUENCE SCANNER
        ;
        ;       SCANS TEXT AND DETERMINES THE STRING BOUNDS
        ;       FOR SEQUENCE TERMS
        ;
        ;          P1                   P2
        ;         I                    I
        ;       **:  .   .   .   .  .  AR:
        ;       I                       I
        ;        C1                      C2
        ;
D50.:   MOV     C2,P1           ;SHIFT POINTERS
        MOV     P2,C1
        MOV     P2,LBND         ;TEXT EVALUATED TO LBND
D50.A:  CALL    FSBR64,P1,IBUFND,#COLON
        TST     %0              ;FIND ':' - SKIP
        BNE     D50.B           ;IF FOUND ':' - SKIP
        JSR     %7,GETINP       ;ELSE NEED NEW INPUT
        BR      D50.A           ;NOW RESCAN FOR ':'
D50.B:  MOV     %0,C2           ;SAVE POSITION OF ':'
        SUB     #2,%0
        MOV     %0,P2           ;SAVE NEW POSITION
        TST     P1              ;AT START UP ?
        BEQ     D50.            ;IF SO - LOOP ONCE !
        MOV     C1,%0
        DEC     %0
        CALL    FSBR74,%0,P1,#STRING
```

```
        CALL    SCAN2B,#SEQTBL,#STRING
        TST     ER              ;VALID SEQUENCE ?
        BEQ     D50.C           ;IF SO - SKIP
        ERROR   #1
        BR      D50.            ;GO TRY AGAIN
D50.C:  MOV     %0,.SC          ;SAVE SEQUENCE CODE
        RTS     %7              ;FINISHED
        ;
        ;
        ;EVALUATION DISPATCH ROUTINE
        ;
EVAL:   MOV     .SC,%0          ;GET SEQUENCE #
        DEC     %0
        ASL     %0              ;MAKE INTO WORD OFFSET
        JSR     %7,@SEQEVL(%0)  ;DO SELECTED SEQUENCE
        RTS     %7              ;FINISHED
        ;
        ;ADDRESS TABLE
        ;
SEQEVL: .WORD   D150.   ;CM:
        .WORD   D120.   ;AR:
        .WORD   D125.   ;CC:
        .WORD   D130.   ;PC:
        .WORD   D135.   ;IO:
        .WORD   D140.   ;BR:
        .WORD   D145.   ;SP:
        .WORD   D170.   ;IN:
        .WORD   D175.   ;MA:
        .WORD   D180.   ;PS:
        .WORD   D190.   ;MW:
        .WORD   D185.   ;VE:
        .WORD   0
        .WORD   0
        .WORD   0
        .WORD   D155.   ;EN:
        .WORD   D105.   ;**:
        .WORD   D115.   ;.=:
        .WORD   D165.   ;NA:
        .WORD   .PG     ;PG:
        .WORD   .TT     ;TT:
        .WORD   .SB     ;SB:
        ;
```

```
        .PAGE
        .SBTTL  INTERNAL ERROR HANDLERS
        ;
TALYER: MOVB    %0,@ERRLOG              ;SAVE ERROR CODE
        CMP     ERRLOG,#ERRTBL+5.       ;ALLOW ONLY 10 ENTRIES
        BCC     .TALY1                  ;SKIP IF AT END
        INC     ERRLOG                  ;UPDATE ADDRESS
.TALY1: MOV     #ERCNT2+1,%0    ;ERROR COUNT ADDRESS
        INCB    (%0)            ;UPDATE COUNT
        CMPB    #'9,(%0)        ;DIGIT OVERFLOW ?
        BHIS    .TALY3          ;IF NOT - SKIP
        MOVB    #'0,(%0)        ;RESET DIGIT
        CMPB    #SPC,-(%0)      ;A SPACE ?
        BNE     .TALY2          ;IF NOT - SKIP
        MOVB    #'0,(%0)        ;SET DIGIT = 0
.TALY2: INCB    (%0)            ;UPDATE COUNT
.TALY3: RTS     %7              ;RETURN
        ;
PNTERR: SAV1
        MOV     ERRLOG,%1       ;GET ERROR ADDRESS LIST
PNT.A:  CMP     #ERRTBL,%1      ;ANY ERRORS TO REPORT ?
        BEQ     PNT.B           ;IF NOT - SKIP
        .PRINT  #ERRCDE
        MOVB    -(%1),%0        ;GET ERROR #
        ASL     %0              ;MAKE WORD OFFSET
        .PRINT  MSGTBL(%0)      ;PRINT THE MESSAGE
        BR      PNT.A           ;LOOP TO CHECK FOR MORE
PNT.B:  MOV     #ERRTBL,ERRLOG  ;RESET ADDRESS LIST
        RES1
        RTS     %7              ;FINISHED
        ;
MSGTBL: .WORD   ERR0
        .WORD   ERR1
        .WORD   ERR2
        .WORD   ERR3
        .WORD   ERR4
        .WORD   ERR5
        .WORD   ERR6
        .WORD   ERR7
        .WORD   ERR8
        .WORD   ERR9
        .WORD   ERR10
        .WORD   ERR11
        .WORD   ERR12
        .WORD   ERR13
        .WORD   ERR14
        .WORD   ERR15
        .WORD   ERR16
        .WORD   ERR17
        .WORD   ERR18
        .WORD   ERR19
        .WORD   ERR20
        .WORD   ERR21
        .WORD   ERR22
        .WORD   ERR23
        .WORD   ERR24
        .WORD   ERR25
        .WORD   ERR26
        .WORD   ERR27
        .WORD   ERR28
        .WORD   ERR29
        .WORD   ERR30
        .WORD   ERR31
        .WORD   ERR32
        .WORD   ERR33
        .WORD   ERR34
        .WORD   ERR35
        .WORD   ERR36
        .WORD   ERR37
        .WORD   ERR38
        .WORD   ERR39
        .WORD   ERR40
        .WORD   ERR41
        .WORD   ERR42
        .WORD   ERR43
        .WORD   ERR44
```

```
        .WORD   ERR45
        .WORD   ERR46
        .WORD   ERR47
        .WORD   ERR48
        .WORD   ERR49
        .WORD   ERR50
        .WORD   ERR51
        ;
        .NLIST  BIN
ERRSET: .ASCIZ  //
ERRCDE: .ASCII  /**** ERROR:  /<200>
ERR0:   .ASCIZ  /UNDEFINED ERROR MESSAGE/
ERR1:   .ASCIZ  /INVALID SEQUENCE CODE/
ERR2:   .ASCIZ  /NAME NOT IN SYMBOL TABLE/
ERR3:   .ASCIZ  /INVALID ARITHMETIC OR LOGICAL OPERATION/
ERR4:   .ASCIZ  /CONDITION CODES NOT ALLOWED/
ERR5:   .ASCIZ  /MISSING EQUIVALENCE/
ERR6:   .ASCIZ  /NOT C,V,Z, OR N/
ERR7:   .ASCIZ  /NON EXISTENT CODE/
ERR8:   .ASCIZ  /INVALID TERMINATOR/
ERR9:   .ASCIZ  *INVALID I/O OPERATOR*
ERR10:  .ASCIZ  /NO MATCHING ')' FOUND/
ERR11:  .ASCIZ  /NON EXISTANT BRANCH/
ERR12:  .ASCIZ  /NO MATCHING ']' FOUND/
ERR13:  .ASCIZ  /NO MATCHING '>' FOUND/
ERR14:  .ASCIZ  /MISSING DELIMITER/
ERR15:  .ASCIZ  /INVALID VECTOR DESIGNATION/
ERR16:  .ASCIZ  /GROUP ADDRESS ERROR/
ERR17:  .ASCIZ  /NOT MW0-MW7/
ERR18:  .ASCIZ  /A WRITE INTO B-REGISTER MUST BE SPECIFIED/
ERR19:  .ASCIZ  /INVALID SHIFT TERM/
ERR20:  .ASCIZ  /TWO DIFFERENT RA'S/
ERR21:  .ASCIZ  /TWO DIFFERENT RB'S/
ERR22:  .ASCIZ  /TWO DIFFERENT DATA PORTS/
ERR23:  .ASCIZ  /MULTIPLE EXTERNAL REGISTERS SPECIFIED/
ERR24:  .ASCIZ  /AN R SHIFT MUST BE SPECIFIED WITH A Q SHIFT/
ERR25:  .ASCIZ  /SHIFT OPERATION NOT ALLOWED/
ERR26:  .ASCIZ  /INTERRUPT CONTROL NOT ALLOWED DURING BRANCH/
ERR27:  .ASCIZ  /SCRATCH REGISTER USE NOT ALLOWED DURING BRANCH/
ERR28:  .ASCIZ  *INTERNAL PRIORITY CONTROL NOT ALLOWED DURING I/O*
ERR29:  .ASCIZ  /SYMBOL TABLE FULL/
ERR30:  .ASCIZ  /NO NAME/
ERR31:  .ASCIZ  /MUST BE AN OCTAL CONSTANT/
ERR32:  .ASCIZ  /NOT A OR B PORT/
ERR33:  .ASCIZ  /WRITE TO Q-REG NOT ALLOWED/
ERR34:  .ASCIZ  /LABLE USED MORE THAN ONCE/
ERR35:  .ASCIZ  /R & Q SHIFTS NOT ALLOWED WITH WRITE Q-REG/
ERR36:  .ASCIZ  /B-REG WRITE NOT ALLOWED WITH WRITE Q-REG/
ERR37:  .ASCII  //
ERR38:  .ASCIZ  //
ERR39:  .ASCIZ  //
ERR40:  .ASCIZ  /?NO INPUT FILE?/
ERR41:  .ASCIZ  /?BAD FILE?/
ERR42:  .ASCIZ  /NO EN:: FOUND/
ERR43:  .ASCIZ  /HARD ERROR ON READ/
ERR44:  .ASCIZ  <CR><LF>/?DEVICE FULL?/
ERR45:  .ASCIZ  <CR><LF>/HARD ERROR ON WRITE/
ERR46:  .ASCIZ  /VECTOR GROUP NOT 0-7/
ERR47:  .ASCIZ  /INPUT BUFFER OVERFLOW/
ERR48:  .ASCIZ  /?BAD SWITCH?/
ERR49:  .ASCIZ  /?ILLEGAL COMMAND?/
ERR50:  .ASCIZ  /?NO DEVICE?/
ERR51:  .ASCIZ  /?CSIGEN ERROR?/
        .EVEN
        .LIST   BIN
        ;
        .PAGE
        .SBTTL  ENTER NAME INTO SYMBOL TABLE ROUTINE
        ;
        ;       CALL    PLCSYM,#STRING,VAL
        ;         ROUTINE CHECKS SYMBOL TABLE FOR 'STRING' AND SET'S
        ;       ER FLAG IF ALREADY THERE AND GIVES IT THE NEW VAL.
        ;         IF NAME IS NEW THE 'STRING' IS ADDED
        ;       TO THE TABLE AND GIVEN THE VALUE VAL
        ;
PLCSYM: SAVL1
```

```
        SAV2
        MOV     #SYMTBL,%2      ;LOCATION OF TABLE
PLC.A:  CMP     (%1),(%2)       ;CHECK FOR A MATCH
        BNE     PLC.B
        CMP     2(%1),2(%2)
        BNE     PLC.B
        CMP     4(%1),4(%2)
        BNE     PLC.B
        CMP     6(%1),6(%2)
        BNE     PLC.B
        MOV     4(%5),10(%2)    ;SAVE NEW VAL
        MOV     #1,ER           ;INDICATE ALLREADY HERE
        BR      PLC.D           ;GO FINISH
PLC.B:  ADD     #12,%2          ;UPDATE TABLE ADDRESS
        CMP     %2,#SYMEND      ;END OF TABLE ?
        BCS     PLC.C           ;SKIP IF NOT
        ERROR   #29.
        BR      PLC.D           ;GO FINISH
PLC.C:  TST     (%2)            ;EMPTY SPACE ?
        BNE     PLC.A           ;IF NOT - CONTINUE SCANNING TABLE
        MOV     (%1)+,(%2)+     ;ELSE FOUND SPOT
        MOV     (%1)+,(%2)+     ;AND SAVING SYMBOL
        MOV     (%1)+,(%2)+
        MOV     (%1)+,(%2)+
        MOV     4(%5),(%2)      ;SAVE VAL IN TABLE
        CLR     ER              ;NO ERROR
PLC.D:  RES2
        RES1
        RTS     %5
        ;
        .PAGE
        .SBTTL  CLEAR SYMBOL TABLE ROUTINE
        ;
CLRSYM: MOV     #MPC,%0         ;INITIALIZE MPC TO END
CLR.A:  CLR     (%0)+           ;CLEAR AREA
        CMP     %0,#SYMEND      ;AT END ?
        BCS     CLR.A           ;LOOP UNTIL ALL CLEARED
        RTS     %7              ;FINISHED
        ;
        .PAGE
        .SBTTL  BEFTBL AND AFRTBL SCAN ROUTINE
        ;
        ;       CALL    EQLSCN,#TABLE,#STRING
        ;         SCANS TABLE FOR A MATCH WITH SPECIFIED STRING
        ;       LOADS CW & OF
        ;
EQLSCN: SAVL1
        SAV2
        SAV3
EQL.A:  MOV     %1,%3           ;%3 IS ADDRESS POINTER
        MOV     4(%5),%2        ;POINT AT STRING
        CMP     (%3)+,(%2)+     ;CHECK FOR MATCH
        BNE     EQL.B
        CMP     (%3)+,(%2)+
        BNE     EQL.B
        CMP     (%3)+,(%2)+
        BNE     EQL.B
        MOV     (%3)+,CW        ;ELSE FOUND MATCH
        MOV     (%3),%0         ;SAVE RESULT IN CW & OF
        MOV     %0,OF
EQL.X:  RES3                    ;RESTORE REGISTERS
        RES2
        RES1
        RTS     %5
EQL.B:  ADD     #12,%1          ;UPDATE ADDRESS IN TABLE
        TST     (%1)            ;=0 ?
        BNE     EQL.A           ;IF NOT - LOOP AGAIN
        CLR     CW
        CLR     OF
        CLR %0
        BR      EQL.X           ;GO FINISH UP
        ;
        .PAGE
        .SBTTL  2,4,6, AND 8 BYTE SCAN ROUTINES
        ;
        ;       CALL    SCAN2B,#TABLE,#STRING
```

```
        ;         SAME FORM FOR SCAN4B,SCAN6B, AND SCAN8B
        ;       TABLE IS A TABLE ADDRESS AND STRING IS
        ;       THE SEARCH STRING
        ;
SCAN2B: SAVL1
        SAVL2
        MOV     #SPSP,%0        ;DOUBLE SPACE
SCN2.A: CMP     (%1),(%2)       ;SEARCH FOR MATCH
        BNE     SCN2.C
        CMP     2(%2),%0        ;REST MUST BE SPACES
        BNE     SCN2.C
        CMP     4(%2),%0
        BNE     SCN2.C
        CMP     6(%2),%0
        BNE     SCN2.C
        MOV     2(%1),%0        ;SAVE RESULT
        CLR     ER              ;NO ERROR
SCN2.B: RES2
        RES1
        RTS     %5
SCN2.C: ADD     #4,%1           ;UPDATE ADDRESS
        TST     (%1)            ;END OF TABLE ?
        BNE     SCN2.A          ;LOOP UNTIL FINISHED
        CLR     %0              ;RESULT
        MOV     #1,ER           ;SET ERROR FLAG
        BR      SCN2.B          ;GO FINISH UP
        ;
SCAN4B: SAVL1
        SAVL2
        MOV     #SPSP,%0        ;DOUBLE SPACE
SCN4.A: CMP     (%1),(%2)       ;SEARCH FOR MATCH
        BNE     SCN4.C
        CMP     2(%1),2(%2)
        BNE     SCN4.C
        CMP     4(%2),%0        ;REST MUST BE SPACES
        BNE     SCN4.C
        CMP     6(%2),%0
        BNE     SCN4.C
        MOV     4(%1),%0        ;SAVE RESULT
        CLR     ER              ;NO ERROR
SCN4.B: RES2
        RES1
        RTS     %5
SCN4.C: ADD     #6,%1           ;UPDATE ADDRESS
        TST     (%1)            ;END OF TABLE ?
        BNE     SCN4.A          ;LOOP UNTIL END
        CLR     %0              ;RESULT
        MOV     #1,ER           ;SET ERROR
        BR      SCN4.B          ;GO FINISH UP
        ;
SCAN6B: SAVL1
        SAVL2
        MOV     #SPSP,%0        ;DOUBLE SPACE
SCN6.A: CMP     (%1),(%2)       ;SEARCH FOR MATCH
        BNE     SCN6.C
        CMP     2(%1),2(%2)
        BNE     SCN6.C
        CMP     4(%1),4(%2)
        BNE     SCN6.C
        CMP     6(%2),%0        ;MUST BE SPACES
        BNE     SCN6.C
        MOV     6(%1),%0        ;RESULT
        CLR     ER              ;NO ERROR
SCN6.B: RES2
        RES1
        RTS     %5
SCN6.C: ADD     #10,%1          ;UPDATE TABLE ADDRESS
        TST     (%1)            ;END OF TABLE ?
        BNE     SCN6.A          ;LOOP UNTIL END
        CLR     %0              ;RESULT
        MOV     #1,ER           ;SET ERROR FLAG
        BR      SCN6.B
        ;
SCAN8B: SAVL1
        SAVL2
SCN8.A: CMP     (%1),(%2)       ;SEARCH FOR MATCH
```

```
        BNE     SCN8.C
        CMP     2(%1),2(%2)
        BNE     SCN8.C
        CMP     4(%1),4(%2)
        BNE     SCN8.C
        CMP     6(%1),6(%2)
        BNE     SCN8.C
        MOV     10(%1),%0       ;RESULT
        CLR     ER              ;NO ERROR
SCN8.B: RES2
        RES1
        RTS     %5
SCN8.C: ADD     #12,%1          ;UPDATE TABLE ADDRESS
        TST     (%1)            ;END OF TABLE ?
        BNE     SCN8.A          ;LOOP UNTIL FINISHED
        CLR     %0              ;RESULT
        MOV     #1,ER           ;SET ERROR FLAG
        BR      SCN8.B          ;GO FINISH UP
        ;
        .PAGE
        .SBTTL  FSBR60
        ;
        ;       ENTRY - CALL    FSBR60, FIRST, LAST, #GP, #SYMBS
        ;                  SCANS STRING FOR COMMA'S AND
        ;               GENERATES SYMBOLS BRACKETTED
        ;               BY COMMA'S
        ;                  RETURNS WITH SYMBOL STRINGS IN SYMBS
        ;               AND THE ADDRESS OF EACH SYMBOL IN
        ;               GP(X)
        ;
FSBR60: SAVL1                   ;SAVE REGISTER AND GET DATA
        SAVL2
        SAVL3
        SAVL4
        CALL    FSBR70,%1,%2,%3
        MOV     %0,%1           ;SAVE COUNT
        MOV     %0,(%3)+        ;SAVE COUNT IN GP
        BEQ     .60B            ;IF NONE - FINISHED
.60A:   CALL    FSBR74,(%3)+,2(%3),%4
        MOV     %4,-2(%3)       ;PLACE ADDRESS IN GP
        ADD     #10,%4          ;8. BYTES PER SYMBOL
        DEC     %1              ;GOT ALL SYMBOLS ?
        BGT     .60A            ;LOOP UNTIL ALL FOUND
.60B:   RES4                    ;RESTORE REGISTERS
        RES3
        RES2
        RES1
        RTS     %5              ;RETURN & CLEAR STACK
        ;
        .PAGE
        .SBTTL  FSBR64
        ;
        ;       ENTRY - CALL    FSBR64, FIRST, LAST, CHAR
        ;               SEARCH FOR OCCURENCE OF CHARACTER
        ;               %0 = ADDRESS OF CHARACTER IF FOUND
        ;               ELSE %0 = 0  IF NOT FOUND
        ;
FSBR64: SAVL1                   ;SAVE AND GET DATA
        SAVL2
        SAVL3
        INC     %1              ;FIRST+1
        DEC     %2              ;LAST-1
        BR      .64B            ;GO TO ENTRY POINT
.64A:   CMPB    (%1),%3         ;THIS THE CHARACTER ?
        BEQ     .64C            ;IF SO - FINISHED
        INC     %1              ;UPDATE ADDRESS
.64B:   CMP     %2,%1           ;END OF SCAN ?
        BCC     .64A            ;LOOP UNTIL FINISHED
        CLR     %1              ;NOTHING FOUND
.64C:   MOV     %1,%0           ;RETURN CHARACTER LOCATION
        RES3                    ;RESTORE REGISTERS
        RES2
        RES1
        RTS     %5              ;RETURN & CLEAR STACK
        ;
        .PAGE
```

```
        .SBTTL  FSBR70
        ;
        ;       ENTRY - CALL    FSBR70, FIRST, LAST, #GP
        ;               SEARCH FOR MULTIPLE OCCURENCES OF ','
        ;               FIRST VALUE IS A VIRTUAL ','
        ;               LAST VALUE IS A VIRTUAL ','
        ;               GP IS THE NUMBER OF TERMS
        ;               FOLLOWING VALUES ARE LOCATIONS OF ','
        ;
FSBR70: SAVL1                   ;SAVE AND GET DATA
        SAVL2
        SAVL3
        SAV4                    ;SAVE %4
        MOV     %3,%4           ;SAVE #GP
        MOV     #1,(%3)+        ;SET COUNT = 1
        MOV     %1,(%3)+        ;SET FIRST POINTER
.70A:   CALL    FSBR64, %1,%2,#COMMA
        TST     %0              ;CHARACTER FOUND ?
        BEQ     .70B            ;BRANCH IF NONE FOUND
        MOV     %0,(%3)+        ;ELSE SAVE POSITION OF CHARACTER
        MOV     %0,%1           ;SCAN REMAINDER OF STRING
        INC     (%4)            ;UPDATE TERM COUNT
        BR      .70A            ;SCAN
.70B:   MOV     %2,(%3)         ;SAVE LAST POINTER
        MOV     (%4),%0         ;RETURN # OF TERMS
        RES4                    ;RESTORE REGISTERS
        RES3
        RES2
        RES1
        RTS     %5              ;RETURN & CLEAR STACK
        ;
        .PAGE
        .SBTTL  FSBR74
        ;
        ;       ENTRY - CALL    FSBR74, FIRST, LAST, #STRING
        ;               GENERATES A SYMBOL FROM THE INPUT STRING
        ;               WITH UPTO 8 CHARACTERS FOLLOWED BY SPACES
        ;               %0 = 0 IF NO SYMBOL IS FOUND
        ;               ^I,SP,CR, AND LF CHARACTERS ARE STRIPPED
        ;
FSBR74: SAVL1                   ;SAVE AND GET DATA
        SAVL2
        SAVL3
        INC     %1              ;FIRST+1
        DEC     %2              ;LAST-1
        SAV4                    ;SAVE %4
        MOV     %3,%4
        MOV     #SPSP,%0        ;DOUBLE SPACE
        MOV     %0,(%3)+        ;SET SYMBOL TO /        /
        MOV     %0,(%3)+
        MOV     %0,(%3)+
        MOV     %0,(%3)
        MOV     %4,%3
        ADD     #10,%4          ;8. BYTES IN THE SYMBOL
        BR      .74C            ;CHECK END OF SCAN
.74A:   CMPB    (%1),#CTRL.I    ;A ^I ?
        BEQ     .74B            ;SKIP THESE CHARACTERS !
        CMPB    (%1),#SPC
        BEQ     .74B
        CMPB    (%1),#CR
        BEQ     .74B
        CMPB    (%1),#LF
        BEQ     .74B
        MOVB    (%1),(%3)+      ;ELSE SAVE CHARACTER
        CMP     %3,%4           ;END OF STRING ?
        BCS     .74B            ;IF SO - SKIP
        DEC     %3              ;RESET POINTER
.74B:   INC     %1              ;UPDATE ADDRESS
.74C:   CMP     %2,%1           ;END OF SCAN ?
        BCC     .74A            ;LOOP UNTIL END
        MOVB    @6(%5),%0
        SUB     #SPC,%0         ;%0 = 0 IF NO SYMBOL
        RES4                    ;RESTORE REGISTERS
        RES3
        RES2
        RES1
```

```
        RTS     %5              ;RETURN & CLEAR STACK
        ;
        .PAGE
        .SBTTL  FSBR80
        ;
        ;       ENTRY - CALL    FSBR80, FIRST, LAST
        ;               SCANS UNTIL A TERMINATION CHARACTER IS FOUND
        ;               RETURNS WITH VALUE OF TERMINATION IN TV AND %0
        ;               AND ADDRESS OF CHARACTER IN TX
        ;               IF NONE FOUND %0 = 0 AND TX = LAST
        ;
FSBR80: SAVL1                   ;SAVE AND GET DATA
        SAVL2
        MOV     %2,TX           ;SET TX = LAST
        DEC     %2              ;LAST-1
.80A:   CLR     %0              ;SAY NONE FOUND
        INC     %1              ;UPDATE ADDRESS
        CMP     %2,%1           ;END OF SCAN ?
        BCS     .80D            ;IF SO - BRANCH
.80B:   MOV     #13,%0          ;SET CHARACTER VALUE
        CMPB    (%1),#COMMA
        BEQ     .80C
        INC     %0
        CMPB    (%1),#EQUAL
        BEQ     .80C
        INC     %0
        CMPB    (%1),#LTPARN
        BEQ     .80C
        INC     %0
        CMPB    (%1),#RTPARN
        BEQ     .80C
        INC     %0
        CMPB    (%1),#LFTBKT
        BEQ     .80C
        INC     %0
        CMPB    (%1),#RTBKT
        BEQ     .80C
        INC     %0
        CMPB    (%1),#LFTANG
        BEQ     .80C
        INC     %0
        CMPB    (%1),#RTANG
        BEQ     .80C
        INC     %0
        CMPB    (%1),#MINUS
        BEQ     .80C
        INC     %0
        CMPB    (%1),#PLUS
        BNE     .80A            ;IF NOTHING FOUND GO SCAN NEXT CHARACTER
.80C:   MOV     %1,TX           ;SAVE CHARACTER POSITION
.80D:   MOV     %0,TV           ;SAVE CHARACTER VALUE
        RES2                    ;RESTORE REGISTERS
        RES1
        RTS     %5              ;RETURN & CLEAR STACK
        ;
        .PAGE
        .SBTTL  FSBR90
        ;
        ;       CALL    FSBR90,FIRST,LAST
        ;         RETURNS THE VALUE OF THE BRACKETED STRING
        ;               XXX + NNN
        ;              I---------I
        ;       WHERE XXX IS A SYMBOL NAME OR .
        ;       AND NNN IS AN OCTAL CONSTANT
        ;       A - SIGN MAY BE USED FOR THE + SIGN
        ;
        ;       RESULT LEFT IN %0
        ;
FSBR90: SAVL1
        SAVL2
        CLR     NW              ;INIT VALUES
        CLR     NZ
        CALL    FSBR64,%1,%2,#PLUS
        MOV     #1,ND           ;SAY '+'
        MOV     %0,NE           ;SAVE POSITION
        BNE     .90A            ;IF '+' FOUND - BRANCH
```

```
        CALL    FSBR64,%1,%2,#MINUS
        MOV     #-1,ND          ;SAY '-'
        MOV     %0,NE           ;SAVE POSITION
        BNE     .90A            ;IF '-' FOUND - BRANCH
        CLR     ND              ;SAY NO CONSTANT FOUND
        MOV     %2,NE           ;DUMPY POSITION
.90A:   CALL    FSBR74,%1,NE,#STRING
        TST     %0              ;SYMBOL FOUND ?
        BEQ     .90B            ;IF NOT - SKIP
        CALL    SCAN8B,#SYMTBL,#STRING
        MOV     %0,NW           ;SAVE VALUE
        TST     ER              ;SYMBOL THERE ?
        BEQ     .90B            ;IF SO - SKIP
        ERROR   #2              ;NO SYMBOL
.90B:   TST     ND              ;CONSTANT TO EVALUATE ?
        BEQ     .90F            ;IF NOT - SKIP
        CALL    FSBR74,NE,%2,#STRING
        SAV1
        MOV     #STRING,%1      ;POINTER TO STRING
.90C:   CMPB    #SPC,(%1)       ;CHECK FOR END OF STRING
        BEQ     .90E            ;IF SO - BRANCH
        CMP     #STRING+10,%1   ;END OF TABLE ?
        BEQ     .90E            ;IF SO - BRANCH
        MOVB    (%1)+,%0        ;GET CHARACTER
        SUB     #60,%0          ;MAKE INTO VALUE
        BMI     .90D            ;< 0
        CMP     %0,#70          ;CHECK RANGE
        BCC     .90D            ;> 7
        MOV     %0,-(%6)        ;SAVE %0
        MOV     NZ,%0           ;GET VALUE
        ASH     #3,%0           ;8.*NZ
        ADD     (%6)+,%0        ;ADD VALUE
        MOV     %0,NZ           ;SAVE RESULT
        BR      .90C            ;LOOP
.90D:   ERROR   #31.
        CLR     NZ              ;RESULT=0
.90E:   RES1
.90F:   MOV     NZ,%0           ;GET CONSTANT VALUE
        TST     ND              ;CHECK SIGN
        BPL     .90G            ;SKIP IF '+'
        NEG     %0              ;MAKE NEGATIVE
.90G:   ADD     NW,%0           ;ADD IN SYMBOL VALUE
        RES2
        RES1
        RTS     %5              ;RETURN AND CLEAR STACK
        ;
        .PAGE
        .SBTTL  ARITHMETIC AND LOGICAL EVALUATION
        ;
D120.:  BIS     #MCODE+NR,CDSTAT        ;MCODE & AR:
        CLR     EQ              ;RESET VARIABLES
        CLR     PK
        CLR     PA
        CLR     PB
        CLR     DP
        CLR     NI
        CLR     NJ
        JSR     %7,RSTII        ;RESET II & IISTTNG
        ;
        ;FIND ALL COMMAS
        ;
        CALL    FSBR70,P1,P2,#GP
        ;
        ;EXECUTE D121. FOR EACH TERM
        ;
        GTIVAL  #GP,#1,#0       ;GET GP(1)
        MOV     %0,NL           ;SAVE GP(1)
        FOR     NA,#1,GP,D121.  ;FOR NA=1,GP; DO D121.
        ;
        ;FINISH SCAN & CHECK FOR ERRORS
        ;
        ;CHECK NUMBER OF = 'S
        ;
        CMP     EQ,#2           ;2 OF THEM ?
        BCS     D120.D          ;IF LESS THAN 2 - BRANCH
        ;
```

```
        ;FINAL CHECKS FOR TWO = SIGNS
        ;
        BIT     #NO+NM,CDSTAT   ;R & Q SHIFTS NOT ALLOWED
        BEQ     D120.A
        ERROR   #25.
D120.A: BIT     #QR,CDSTAT      ;NO Q WRITES
        BEQ     D120.B
        ERROR   #33.
D120.B: BIT     #NP,CDSTAT      ;NEED WRITE TO B-REG
        BNE     D120.C
        ERROR   #18.
D120.C: BIC     #700,MW0
        BIS     #200,MW0        ;SET CONTROL BITS
        RTS     %7              ;FINISHED
        ;
        ;FINAL CHECKS FOR LESS THAN TWO = SIGNS
        ;
D120.D: BIT     #NM,CDSTAT      ;AN R-SHIFT ?
        BNE     D120.E
        BIT     #QR,CDSTAT      ;WRITE TO Q-REG ?
        BNE     D120.I
        BIT     #NO,CDSTAT      ;A Q-SHIFT ?
        BNE     D120.L
        BIT     #NP,CDSTAT      ;WRITE TO B-REG ?
        BNE     D120.N
        BIS     #100,MW0        ;DEFAULT NO-OP
        RTS     %7              ;FINISHED
        ;
        ;R-SHIFT CHECKS
        ;
D120.E: BIT     #NO,CDSTAT      ;Q-SHIFT ?
        BEQ     D120.F
        BIC     #100,MW0        ;ALLOW Q-SHIFT
D120.F: BIT     #QR,CDSTAT      ;WRITE TO Q ?
        BEQ     D120.G
        ERROR   #33.            ;WRITE TO Q NOT ALLOWED
D120.G: BIT     #NP,CDSTAT      ;NEED A WRITE TO B-REG
        BNE     D120.H
        ERROR   #18.
D120.H: RTS     %7              ;FINISHED
        ;
        ;WRITE TO Q-REG CHECKS
        ;
D120.I: BIT     #NP,CDSTAT      ;WRITE TO B-REG NOT ALLOWED
        BEQ     D120.J
        ERROR   #36.
D120.J: BIT     #NM+NO,CDSTAT   ;R & Q SHIFTS NOT ALLOWED
        BEQ     D120.K
        ERROR   #35.
D120.K: RTS     %7              ;FINISHED
        ;
        ;ERROR IN Q SHIFT
        ;
D120.L: ERROR   #24.            ;NEED R-SHIFT
        BIT     #NP,CDSTAT      ;WRITE TO B-REG NEEDED
        BNE     D120.M
        ERROR   #18.
D120.M: RTS     %7              ;FINISHED
        ;
        ;WRITE TO B-REG
        ;
D120.N: BIS     #300,MW0        ;SET CONTROL BITS
        RTS     %7              ;FINISHED
        ;
        .PAGE
        .SBTTL  BEFORE AND AFTER '=' EVALUATION
        ;
D121.:  SAV1
        CLR     PK              ;HASH CODE = 0
        ;
        ;GET GP(NA) & GP(NA+1)
        ;
        MOV     NL,NF           ;SET FIRST
        GTIVAL  #GP,NA,#1       ;GET GP(NA+1)
        MOV     %0,NL           ;LAST
        ;
```

```
        ;FIND EQUAL SIGNS
        ;
        CALL    FSBR64,NF,NL,#EQUAL
        MOV     %0,NH           ;EQUAL SIGN POSITION
        BEQ     D121.A          ;IF NONE FOUND - SKIP
        INC     EQ              ;ELSE UPDATE COUNTER
        MOV     #-1,NQ          ;SAY BEFORE '='
        MOV     NF,%0
        INC     %0              ;FIRST+1
        MOV     NH,%1
        DEC     %1              ;LAST-1
        ;
        ;EVALUATE HASH CODES FOR TERMS
        ;
        FOR     NC,%0,%1,D122.
        CHKII                   ;CHECK II FOR EVALUATION
        CLR     PK              ;CLEAR HASH CODE
        MOV     NH,NF           ;NOW AFTER '='
        ;
        ;AFTER '=' EVALUATION
        ;
D121.A: MOV     #1,NQ           ;AFTER '='
        MOV     NF,%0
        INC     %0              ;FIRST+1
        MOV     NL,%1
        DEC     %1              ;LAST-1
        ;
        ;EVALUATE HASH CODES FOR TERMS
        ;
        FOR     NC,%0,%1,D122.
        CHKII                   ;CHECK II FOR EVALUATION
        ;
        ;FINISH UP EVALUATION
        ;
        CMP     EQ,#2           ;SECOND PASS THIS SEGMENT
        BEQ     D121.D          ;DONOT SET ARITHMETIC CODES !
        TST     PK              ;ANY CODE TO EVALUATE ?
        BEQ     D121.D          ;IF NOT - SKIP
        MOV     #ARTABL,%0      ;ADDRESS OF TABLE
D121.B: CMP     (%0),PK         ;THIS IT ?
        BEQ     D121.C          ;IF SO - SKIP
        ADD     #4,%0           ;UPDATE ADDRESS
        TST     (%0)            ;END OF TABLE ?
        BNE     D121.B          ;LOOP IF NOT
        ERROR   #3              ;NOT FOUND
        BR      D121.D
D121.C: BIS     2(%0),MW0       ;SET CONTROL BITS
D121.D: RES1
        RTS     %7              ;FINISHED
        ;
        .PAGE
        .SBTTL  HASH CODE GENERATION
        ;
D122.:  SAV1
        SAV2
        MOVB    @NC,%1          ;GET CHARACTER
        MOV     #IISTNG,%2      ;STRING ADDRESS
        ;
        ;CHECK FOR TERMINATION DELIMITERS
        ;
        CMPB    %1,#CTRL.I      ;CHECK FOR ^I,SP,CR,LF
        BEQ     D122.A
        CMPB    %1,#SPC
        BEQ     D122.A
        CMPB    %1,#CR
        BEQ     D122.A
        CMPB    %1,#LF
        BNE     D122.B          ;NOT DELIMITERS
D122.A: CHKII                   ;CHECK II FOR EVALUATION
        BR      D122.Z          ;GO FINISH
        ;
        ;CHECK FOR '+'
        ;
D122.B: CMPB    %1,#PLUS        ;CHECK FOR '+'
        BNE     D122.C          ;IF NOT - SKIP
        CHKII                   ;CHECK II FOR EVALUATION
```

```
        UPDTPK  #1              ;UPDATE HASHCODE
        BR      D122.Z          ;GO FINISH
        ;
        ;CHECK FOR '-'
        ;
D122.C: CMPB    %1,#MINUS       ;CHECK FOR '-'
        BNE     D122.D          ;IF NOT - SKIP
        CHKII                   ;CHECK II FOR EVALUATION
        UPDTPK  #2              ;UPDATE HASH CODE
        BR      D122.Z          ;GO FINISH
        ;
        ;CHECK FOR ';'
        ;
D122.D: CMPB    %1,#SMICLN      ;CHECK FOR ';'
        BNE     D122.E          ;IF NOT - SKIP
        CHKII                   ;CHECK II FOR EVALUATION
                                ;NO PK UPDATE !
        BR      D122.Z          ;GO FINISH
        ;
        ;CHECK FOR '/'
        ;
D122.E: CMPB    %1,#DIVIDE      ;CHECK FOR '/'
        BNE     D122.F          ;IF NOT - SKIP
        CHKII                   ;CHECK II FOR EVALUATION
        UPDTPK  #11             ;UPDATE HASH CODE
        BR      D122.Z          ;GO FINISH
        ;
        ;CHECK FOR '['
        ;
D122.F: CMPB    %1,#LFTBKT              ;CHECK FOR '['
        BNE     D122.G          ;IF NOT - SKIP
        JSR     %7,D223.        ;DO ROUTINE
        UPDTPK  #5              ;UPDATE HASH CODE
        JSR     %7,RSTII        ;CLEAR II & IISTNG
        BR      D122.Z          ;GO FINISH
        ;
        ;CHECK FOR '('
        ;
D122.G: CMPB    %1,#LTPARN      ;CHECK FOR '('
        BNE     D122.H          ;IF NOT - SKIP
        JSR     %7,D123.        ;GO EVALUATE
                                ;NO HASH CODE UPDATE
        BR      D122.Z          ;GO FINISH
        ;
        ;CHECK FOR '<'
        ;
D122.H: CMPB    %1,#LFTANG      ;CHECK FOR '<'
        BNE     D122.I          ;IF NOT - SKIP
        JSR     %7,D224.        ;DO ROUTINE
        JSR     %7,RSTII        ;CLEAR II & IISTNG
        BR      D122.Z          ;GO FINISH
        ;
        ;BUILD IISTNG
        ;
D122.I: MOV     II,%0           ;BYTE COUNT
        CMP     %0,#10          ;PAST 8. BYTES ?
        BCC     D122.Z          ;IF SO - SKIP
        MOVB    %1,IISTNG(%0)   ;ELSE SAVE CHARACTER
        INC     II              ;UPDATE COUNTER
D122.Z: RES2                    ;RESTORE REGISTERS
        RES1
        RTS     %7              ;FINISHED
        ;
```

```
        .PAGE
        .SBTTL  DISPATCH ROUTINE FOR TERMS
        ;
D123.:  TST     NQ              ;BEFORE OR AFTER
        BGE     D123.A          ;AFTER - SKIP
        CALL    EQLSCN,#BEFTBL,#IISTNG
        BR      D123.B
D123.A: CALL    EQLSCN,#AFRTBL,#IISTNG
D123.B: ASL     %0              ;MAKE WORD OFFSET
        JSR     %7,@DSPTCH(%0)  ;GO TO SELECTED ROUTINE
        ;
        ;CLEAR II & IISTNG ROUTINE
        ;
RSTII:  MOV     #SPSP,%0        ;DOUBLE SPACE
        MOV     %0,IISTNG       ;PLACE SPACES IN IISTNG
        MOV     %0,IISTNG+2
        MOV     %0,IISTNG+4
        MOV     %0,IISTNG+6
        CLR     II              ;NO SYMBOL
        RTS     %7              ;FINISHED
        ;
DSPTCH: .WORD   D210.
        .WORD   D211.
        .WORD   D212.
        .WORD   D213.
        .WORD   D214.
        .WORD   D215.
        .WORD   D216.
        .WORD   D217.
        .WORD   D218.
        .WORD   D219.
        ;
        ;
        .PAGE
        .SBTTL  FURTHER ARITHMETIC EVALUATIONS
        ;
        ;ERROR
        ;
D210.:  ERROR   #7              ;NON EXISTENT CODE
        RTS     %7
        ;
        ;R & Q SHIFT EVALUATIONS
        ;
        ;SCAN FOR ')'
        ;
D211.:  CALL    FSBR64,NC,NL,#RTPARN
        MOV     %0,NK           ;SAVE POSITION OF ')'
        BNE     D211.A          ;SKIP IF FOUND A ')'
        ERROR   #10.            ;ELSE ERROR
        MOV     NL,NC           ;TERMINATE LOOP
        RTS     %7              ;FINISHED
        ;
        ;GET STRING AND EVALUATE TERM
        ;
D211.A: CALL    FSBR74,NC,NK,#STRING
        CALL    SCAN6B,#SHFTBL,#STRING
        TST     ER              ;SYMBOL FOUND ?
        BEQ     D211.B          ;SKIP IF FOUND STRING
        ERROR   #19.            ;ELSE UNKNOWN STRING
        MOV     NL,NC           ;TERMINATE LOOP
        RTS     %7              ;FINISHED
        ;
D211.B: MOV     %0,NG           ;SAVE RESULT FROM TABLE
        MOV     NK,NC           ;SCANNING UPTO ')'
        CMP     #"SQ,IISTNG     ;THIS A Q SHIFT ?
        BEQ     D211.C          ;IF SO - SKIP
        BIS     CW,MW0          ;SET BITS IN RESULT
        BIS     NG,MW3          ;SET R SHIFT SELECT BITS
        BIS     #NM,CDSTAT      ;INDICATE R SHIFT
        RTS     %7              ;FINISHED
D211.C: MOV     NG,%0           ;GET SELECT BITS
        ASH     #3,%0           ;SHIFT INTO PLACE
        BIS     %0,MW3          ;SET Q SHIFT SELECT BITS
        BIS     #NO,CDSTAT      ;INDICATE Q SHIFT
        RTS     %7              ;FINISHED
        ;
```

```
        ;A-REGISTER PROCESSING
        ;
D212.:  UPDTPK  #2              ;UPDATE HASHCODE
        TST     PA              ;A PREVIOUS A TERM ?
        BEQ     D212.B          ;IF NOT - SKIP
        CMP     PA,CW           ;ELSE MUST BE SAME A-REG
        BEQ     D212.A
        ERROR   #20.            ;MULTIPLE A-REG
D212.A: RTS     %7              ;FINISHED
D212.B: MOV     CW,PA           ;SAVE VALUE
        BIS     CW,MW1          ;SET CONTROL BITS
        RTS     %7              ;FINISHED
        ;
        ;B-REGISTER PROCESSING
        ;
D213.:  UPDTPK  #3              ;UPDATE HASH CODE
        TST     NQ              ;BEFORE '=' ?
        BGT     D213.A          ;IF AFTER - SKIP
        BIS     #NP,CDSTAT      ;WRITE INTO B-REG
D213.A: TST     PB              ;A PREVIOUS B TERM ?
        BEQ     D213.C          ;IF SO - SKIP
        CMP     PB,CW           ;MUST BE SAME
        BEQ     D213.B          ;IF SO - SKIP
        ERROR   #21.            ;MULTIPLE B-REG
D213.B: RTS     %7              ;FINISHED
D213.C: MOV     CW,PB           ;SAVE VALUE
        BIS     CW,MW1          ;SET BITS IN CONTROL
        RTS     %7              ;FINISHED
        ;
        ;Q-REGISTER PROCESSING
        ;
D214.:  UPDTPK  #4              ;UPDATE HASH CODE
        TST     NQ              ;BEFORE '=' ?
        BGE     D214.A          ;SKIP IF AFTER
        BIS     #QR,CDSTAT      ;WRITE TO Q REG
D214.A: RTS     %7              ;FINISHED
        ;
        ;DATA PORT PROCESSING
        ;
D215.:  UPDTPK  #5              ;UPDATE HASH CODE
D215.1: TST     DP              ;A PREVIOUS VALUE ?
        BEQ     D215.B          ;IF NOT - SKIP
        CMP     DP,CW           ;MUST BE SAME
        BEQ     D215.A          ;IF SO - SKIP
        ERROR   #22.            ;MORE THAN ONE PORT
D215.A: RTS     %7              ;FINISHED
D215.B: MOV     CW,DP           ;SAVE PORT VALUE
        BISB    CW+1,MW0+1      ;SET PORT BITS
        CMPB    CW+1,#200       ;THIS THE SCRATCH PORT ?
        BNE     D215.C          ;IF NOT - SKIP
        BISB    CW,MW5          ;SET SCRATCH READ SELECT
        BIS     #MS,CDSTAT      ;SCRATCH USED
D215.C: RTS     %7              ;FINISHED
        ;
        ;OPERATION CODES  OR,AND,MASK,XOR, & XNOR
        ;
D216.:  UPDTPK  CW              ;UPDATE HASH CODE
        RTS     %7              ;FINISHED
        ;
        ;DESTINATION ADDRESS EVALUATION
        ;
D217.:  TST     NQ              ;BEFORE '=' ?
        BMI     D217.A          ;IF SO - SKIP
        MOV     CW,PK           ;ELSE SET HASH CODE
        RTS     %7              ;FINISHED
D217.A: CMP     #"SC,IISTNG     ;A SCRATCH REGISTER ?
        BNE     D217.C          ;NO - AN EXTERNAL REGISTER
        ;
        ;SCRATCH REGISTER STORE
        ;
        TST     NJ              ;A PREVIOUS SCRATCH ?
        BEQ     D217.B          ;IF NOT - SKIP
        ERROR   #23.            ;ONLY ONE ALLOWED
        RTS     %7              ;FINISHED
D217.B: INC     NJ              ;INDICATE SCRATCH USED
        BIS     CW,MW5          ;SET SCRATCH WRITE SELECT
```

```
        BIS     #MS,CDSTAT      ;SCRATCH USED
        RTS     %7              ;FINISHED
        ;
        ;EXTERNAL REGISTER STORE
        ;
D217.C: TST     NI              ;A PREVIOUS EXT-REG ?
        BEQ     D217.D          ;IF NOT - SKIP
        ERROR   #23.            ;ONLY ONE ALLOWED
        RTS     %7              ;FINISHED
D217.D: INC     NI              ;WRITE TO EXT REG
        BIS     CW,MW0          ;SET SELECT BITS
        RTS     %7              ;FINISHED
        ;
        ;RA & RB PROCESSING
        ;
D218.:  CALL    FSBR64,NC,NL,#RTPARN
        MOV     %0,NK           ;POSITION OF ')'
        BNE     D218.A          ;BRANCH IF FOUND
        ERROR   #10.            ;NO ')'
        MOV     NL,NC           ;TERMINATE LOOP
        RTS     %7              ;FINISHED
D218.A: CALL    FSBR74,NC,NK,#STRING
        MOV     NK,NC           ;EVALUATED TO NK
        ;
        ;EVALUATE NEW TERM
        ;
        MOV     CW,-(%6)
        MOV     OF,-(%6)
        CALL    EQLSCN,#AFRTBL,#STRING
        MOV     CW,NY
        MOV     OF,NX
        MOV     (%6)+,OF
        MOV     (%6)+,CW
        CMP     #5,NX           ;INPUT PORT SPECIFIER ?
        BEQ     D218.B          ;IF SO - BRANCH
        CMP     #11,NX          ;SRC,SCR!1,DST, OR DST!1 ?
        BEQ     D218.F          ;IF SO - BRANCH
        ERROR   #7
        RTS     %7              ;FINISHED
        ;
        ;INPUT PORT SPECIFIER EVALUATION
        ;
D218.B: CMP     #"RA,IISTNG     ;A-PORT ?
        BNE     D218.C          ;IF NOT - SKIP
        JSR     %7,D212.        ;GO DO A-PORT
        BR      D218.D
D218.C: CMP     #"RB,IISTNG     ;B-PORT ?
        BNE     D218.E          ;THIS IS AN ERROR
        JSR     %7,D213.        ;GO DO B-PORT
D218.D: MOV     NY,CW           ;RESET CODE
        JSR     %7,D215.1       ;FINISH DATA PORT PROCESSING
        RTS     %7              ;FINISHED
D218.E: ERROR   #32.
        RTS     %7              ;FINISHED
        ;
        ;SPECIAL MODES  SRC,SRC!1,DST, & DST!1
        ;
D218.F: CMP     #"RA,IISTNG     ;A-PORT ?
        BEQ     D218.G          ;IF SO - BRANCH
        CMP     #"RB,IISTNG     ;B-PORT ?
        BEQ     D218.K          ;IF SO - BRANCH
        ERROR   #32.
        RTS     %7              ;FINISHED
D218.G: TST     PA              ;PREVIOUS A-PORT ?
        BEQ     D218.H          ;IF NOT - SKIP
        CMP     PA,NY           ;MUST BE SAME
        BEQ     D218.J          ;IF SO - SKIP
        ERROR   #20.            ;TWO A-PORTS
        MOV     NL,NC           ;TERMINATE SCAN
        RTS     %7              ;FINISHED
D218.H: MOV     NY,PA           ;SAVE VALUE
        CMPB    #'!,STRING+3    ;! ?
        BNE     D218.I          ;IF NOT - SKIP
        BIS     #1,MW1          ;ELSE SET OR BIT
D218.I: BIS     NY,MW1          ;SET CONTROL BITS
D218.J: UPDTPK  #2              ;UPDATE HASH CODE
```

```
        RTS     %7              ;FINISHED
D218.K: TST     PB              ;PREVIOUS B-PORT
        BEQ     D218.L          ;IF NOT - SKIP
        CMP     PB,NY           ;MUST BE SAME
        BEQ     D218.N          ;IF SO - SKIP
        ERROR   #21.            ;TWO B-PORTS
        MOV     NL,NC           ;TERMINATE SCAN
        RTS     %7              ;FINISHED
D218.L: MOV     NY,PB           ;SAVE VALUE
        CMPB    #'!,STRING+3    ;! ?
        BNE     D218.M          ;IF NOT - SKIP
        BIS     #100,MW1        ;SET OR BIT
D218.M: MOV     NY,%0           ;GET CONTROL BITS
        ASH     #6,%0           ;SHIFT 6 PLACES
        BIS     %0,MW1          ;SET CONTROL BITS
D218.N: TST     NQ              ;BEFORE '=' ?
        BGE     D218.O          ;IF NOT - SKIP
        BIS     #NP,CDSTAT      ;WRITE INTO B-REG
D218.O: UPDTPK  #3              ;UPDATE HASH CODE
        RTS     %7              ;FINISHED
        ;
        ;ERROR
        ;
D219.:  ERROR   #7
        RTS     %7              ;FINISHED
        ;
        ;CONSTANT EVALUATION
        ;
D223.:  BIT     #NS,CDSTAT      ;CC'S BEEN SPECIFIED ?
        BEQ     D223.A          ;IF NOT - SKIP
        ERROR   #4              ;NOT ALLOWED
        MOV     NL,NC           ;TERMINATE OPERATION
        RTS     %7              ;FINISHED
D223.A: BIS     #NS,CDSTAT      ;SAY MW2 USED
        ;
        ;SEARCH FOR ']'
        ;
        CALL    FSBR64,NC,NL,#RTBKT
        MOV     %0,NK           ;SAVE POSITION
        BNE     D223.B          ;IF ']' FOUND - SKIP
        ERROR   #12.            ;NO ']'
        MOV     NL,NC           ;TERMINATE OPERATION
        RTS     %7              ;FINISHED
D223.B: TST     DP              ;PREVIOUS DATA-PORT ?
        BEQ     D223.C          ;IF NOT - SKIP
        CMPB    DP+1,#320       ;MUST BE DATA PORT
        BEQ     D223.D          ;IF SO - SKIP
        ERROR   #22.
        MOV     NK,NC           ;TERMINATE OPERATION
        RTS     %7              ;FINISHED
D223.C: MOV     #150000,DP      ;SET FOR DATA PORT
        BIS     #150000,MW0     ;SET CONTROL BITS
        CALL    FSBR90,NC,NK
D223.D: BIS     %0,MW2          ;PLACE CONSTANT
        MOV     NK,NC           ;SCANNED TO NK
        RTS     %7              ;FINISHED
        ;
        ;ARITHMETIC CARRY IN PROCESSING
        ;
D224.:  CALL    FSBR64,NC,NL,#RTANG
        MOV     %0,NK           ;POSITION OF '>'
        BNE     D224.A          ;FOUND '>' - BRANCH
        ERROR   #17.            ;NO '>'
        MOV     NL,NC           ;TERMINATE OPERATION
        RTS     %7              ;FINISHED
D224.A: CALL    FSBR74,NC,NK,#STRING
        CALL    SCAN6B,#CRYTBL,#STRING
        BIS     %0,MW1          ;SET CONTROL BITS
        TST     ER              ;%0=0 IF THERE WAS AN ERROR !
        BEQ     D224.B          ;NO ERROR - SKIP
        ERROR   #7
D224.B: MOV     NK,NC           ;SCANNED TO NK
        RTS     %7              ;FINISHED
        ;
        .PAGE
        .SBTTL  **: & .=:  SEQUENCES
```

```
        ;
D105.:  CALL    FSBR74,P1,P2,#STRING
        TST     %0              ;A SYMBOL ?
        BEQ     D105.A          ;BRANCH IF NONE
        CALL    PLCSYM,#STRING,MPC
        TST     ER              ;NAME SHOULD NOT BE THERE
        BEQ     D105.A          ;IF NOT - SKIP
        ERROR   #34.
D105.A: RTS     %7              ;FINISHED
        ;
D115.:  CALL    FSBR90,P1,P2
        MOV     %0,MPC          ;NEW PROGRAM LOCATION
        BIS     #M.,CDSTAT      ;ENTRY INDICATOR
        RTS     %7              ;FINISHED
        ;
        .PAGE
        .SBTTL  CC: PROCESSING
        ;
D125.:  BIS     #MCODE,CDSTAT   ;MCODE DEFINED
        BIT     #NS,CDSTAT      ;MW2 USED ?
        BEQ     D125.A          ;IF NOT - SKIP
        ERROR   #4
        RTS     %7              ;FINISHED
        ;
        ;FIND ALL COMMA'S
        ;
D125.A: CALL    FSBR70,P1,P2,#GP
        ;
        ;DO D126. FOR EVERY TERM
        ;
        GTIVAL  #GP,#1,#0       ;GET GP(1)
        MOV     %0,GP2          ;SAVE VALUE
        FOR     X,#1,GP,D126.
        BIS     #NS,CDSTAT      ;ENTRY INDICATOR
        RTS     %7              ;FINISHED
        ;
        ;GET GP(X) & GP(X+1)
        ;
D126.:  MOV     GP2,GP1         ;SET FIRST
        GTIVAL  #GP,X,#1        ;GET GP(X+1)
        MOV     %0,GP2          ;SAVE GP(X+1)
        ;
        ;FIND EQUAL SIGN
        ;
        CALL    FSBR64,GP1,GP2,#EQUAL
        MOV     %0,NH           ;SAVE POSITION OF '='
        BNE     D126.A          ;FOUND '=' - SKIP
        ERROR   #5              ;NO '='
        RTS     %7              ;FINISHED
        ;
        ;GET STRING AND CHECK IF IT IS N,Z,V, OR C
        ;
D126.A: CALL    FSBR74,GP1,NH,#STRING
        CALL    SCAN2B,#PSWCCS,#STRING
        TST     ER              ;THERE ?
        BEQ     D126.B          ;IF SO - SKIP
        ERROR   #6              ;WRONG CODE
        RTS     %7
D126.B: MOV     %0,CQ           ;SAVE SCANNING TABLE
        ;
        ;GET CODE STRING
        ;
        CALL    FSBR74,NH,GP2,#STRING
        CALL    SCAN6B,CQ,#STRING
        TST     ER              ;VALID CODE ?
        BEQ     D126.C          ;IF SO - SKIP
        ERROR   #7              ;UNDEFINED CODE
        RTS     %7              ;FINISHED
D126.C: BIS     %0,MW2          ;SET CONTROL BITS
        RTS     %7              ;FINISHED
        ;
        .PAGE
        .SBTTL  PROCESSOR CONTROL EVALUATION
        ;
        ;GET STRINGS FOR EVERY TERM
        ;
```

```
D130.:  BIS     #MCODE,CDSTAT   ;MCODE DEFINED
        CALL    FSBR60,P1,P2,#GP,#GPSYM
        ;
        ;EVALUATE EACH TERM
        ;
        FOR     X,#1,GP,D131.
        BIS     #NT,CDSTAT      ;ENTRY INDICATOR
        RTS     %7              ;FINISHED
        ;
        ;CHECK TABLE FOR TERMS
        ;
D131.:  GTIVAL  #GP,X,#0        ;GET GP(X)
        CALL    SCAN4B,#PRCTBL,%0
        TST     ER              ;DEFINED ?
        BEQ     D131.A          ;IF SO - SKIP
        ERROR   #7              ;UNDEFINED CODE
        RTS     %7
D131.A: BIS     %0,MW3          ;SET CONTROL BITS
        RTS     %7              ;FINISHED
        ;
        .PAGE
        .SBTTL  I/O PROCESSING
        ;
        ;SCAN FOR A TERMINATION CHARACTER
        ;
D135.:  BIS     #MCODE,CDSTAT   ;MCODE DEFINED
        CALL    FSBR80,P1,P2
        CALL    FSBR74,P1,TX,#STRING
        CALL    SCAN8B,#BUSTBL,#STRING
        TST     ER              ;CODE FOUND ?
        BEQ     D135.A          ;IF SO - BRANCH
        ERROR   #9.             ;BAD I/O
        RTS     %7              ;FINISHED
D135.A: BIS     %0,MW4          ;SET I/O BITS
        TST     TV              ;=0 ?
        BNE     D135.B          ;IF MORE - SKIP
        RTS     %7              ;ELSE FINISHED
D135.B: CMP     #15,TV          ;WAS TERMINATION  '(' ?
        BEQ     D136.           ;IF SO - SKIP
        ERROR   #8.             ;GOOFED
        RTS     %7              ;FINISHED
        ;
        ;CONTINUE SCAN FOR ( TERM )
        ;
D136.:  CALL    FSBR64,TX,P2,#RTPARN
        MOV     %0,TY           ;SAVE POSITION
        BNE     D136.A          ;SKIP IF FOUND ')'
        ERROR   #10.            ;NO ')'
        RTS     %7              ;FINISHED
D136.A: CALL    FSBR60,TX,TY,#GP,#GPSYM
        FOR     X,#1,GP,D137.
        RTS     %7              ;FINISHED
        ;
        ;EVALUATE EACH TERM
        ;
D137.:  GTIVAL  #GP,X,#0        ;GET GP(X)
        CALL    SCAN4B,#MMGTBL,%0
        TST     ER              ;CODE FOUND ?
        BEQ     D137.A          ;IF SO - SKIP
        ERROR   #7              ;UNDEFINED CODE
        RTS     %7              ;FINISHED
D137.A: BIS     %0,MW4          ;SET CONTROL BITS
        RTS     %7              ;FINISHED
        ;
        .PAGE
        .SBTTL  BRANCHING EVALUATION
        ;
D140.:  BIS     #MCODE,CDSTAT   ;MCODE DEFINED
        BIS     #NU,CDSTAT      ;ENTRY INDICATOR
        ;
        ;FIND TERMINATION CHARACTER
        ;
        CALL    FSBR80,P1,P2
        ;
        ;CONVERT TERM TO STRING & VERIFY
        ;
```

```
        CALL    FSBR74,P1,TX,#STRING
        CALL    SCAN6B,#MCBTBL,#STRING
        TST     ER              ;VALID CODE ?
        BEQ     D140.A          ;IF SO - SKIP
        ERROR   #11.            ;NO SUCH BRANCH
        RTS     %7              ;FINISHED
D140.A: BIS     %0,MW5          ;SET CONTROL BITS
D140.B: MOV     TV,%0           ;MORE TO CHECK ?
        BNE     D140.C          ;IF SO - SKIP
        RTS     %7              ;FINISHED
D140.C: CMP     #15,%0          ;WAS TERMINATION '(' ?
        BNE     D140.D          ;IF NOT - SKIP
        JSR     %7,D141.        ;SCAN (TERM)
        BR      D140.G          ;GO FINISH
D140.D: CMP     #17,%0          ;WAS TERMINATION '['
        BNE     D140.E          ;IF NOT - SKIP
        JSR     %7,D142.        ;SCAN [TERM]
        BR      D140.G          ;GO FINISH
D140.E: CMP     #21,%0          ;WAS TERMINATION '<' ?
        BNE     D140.F          ;IF NOT - SKIPTO ERROR
        JSR     %7,D143.        ;SCAN <TERM>
        BR      D140.G          ;GO FINISH
D140.F: ERROR   #8.             ;WRONG TERMINATION
        RTS     %7              ;FINISHED
D140.G: CMP     TX,P2           ;FINISHED YET ?
        BCS     D140.H          ;IF NOT - SKIP
        RTS     %7              ;ELSE DONE
D140.H: CALL    FSBR80,TX,P2
        BR      D140.B          ;LOOP
        ;
        ;SCAN TERM IN ()
        ;
D141.:  CALL    FSBR64,TX,P2,#RTPARN
        MOV     %0,TY           ;SAVE POSITION
        BNE     D141.A          ;SKIP IF ')' FOUND
        ERROR   #10.            ;NO ')'
        MOV     P2,TX           ;TERMINATE SCAN
        RTS     %7              ;FINISHED
D141.A: CALL    FSBR74,TX,TY,#STRING
        CALL    SCAN6B,#BRCTBL,#STRING
        TST     ER              ;FIND IT ?
        BEQ     D141.B          ;IF SO - SKIP
        ERROR   #7
        MOV     TY,TX           ;TERMINATE SCAN
        RTS     %7              ;FINISHED
D141.B: BIS     %0,MW4          ;SET CONTROL BITS
        MOV     TY,TX           ;TERMINATE SCAN
        RTS     %7              ;FINISHED
        ;
        ;SCAN TERM []
        ;
D142.:  CALL    FSBR64,TX,P2,#RTBKT
        MOV     %0,TY           ;SAVE POSITION
        BNE     D142.A          ;SKIP IF FOUND ']'
        ERROR   #12.            ;NO ']'
        MOV     P2,TX           ;TERMINATE SCAN
        RTS     %7              ;FINISHED
D142.A: CALL    FSBR90,TX,TY
        BIC     #170000,%0      ;ONLY 12 BITS WORTH
        BIS     %0,MW5          ;SET BITS
        MOV     TY,TX
        RTS     %7              ;FINISHED
        ;

        ;SCAN TERM <>
        ;
D143.:  CALL    FSBR64,TX,P2,#RTANG
        MOV     %0,TY           ;SAVE POSITION
        BNE     D143.A          ;SKIP IF FOUND '>'
        ERROR   #13.            ;NO '>'
        MOV     TY,TX           ;TERMINATE SCAN
        RTS     %7              ;FINISHED
D143.A: CALL    FSBR74,TX,TY,#STRING
        CALL    SCAN4B,#IBCTBL,#STRING
        TST     ER              ;FIND IT ?
        BEQ     D143.B          ;IF SO - SKIP
```

```
        ERROR   #7
        MOV     TY,TX           ;TERMINATE SCAN
        RTS     %7              ;FINISHED
D143.B: BIS     %0,MW5          ;SET BITS
        MOV     TY,TX
        RTS     %7              ;FINISHED
        ;
        .PAGE
        .SBTTL  SPECIAL MODES EVALUATION
        ;
        ;GET STRINGS FOR ALL TERMS
        ;
D145.:  BIS     #MCODE,CDSTAT   ;MCODE DEFINED
        CALL    FSBR60,P1,P2,#GP,#GPSYM
        FOR     X,#1,GP,D146.
        RTS     %7              ;FINISHED
        ;
        ;EVALUATE EACH TERM
        ;
D146.:  GTIVAL  #GP,X,#0        ;GET GP(X)
        CALL    SCAN4B,#SPLTBL,%0
        TST     ER              ;FIND TERM ?
        BEQ     D146.A          ;IF SO - SKIP
        ERROR   #7              ;NOT THERE
        RTS     %7              ;FINISHED
D146.A: TST     %0              ;CHECK SIGN OF RESULT
        BMI     D146.B          ;IF NEGATIVE - BRANCH
        BIS     %0,MW5          ;SET BITS
        BIS     #MI,CDSTAT      ;CONFLICT FLAG - I/O OR INST SEQ
        RTS     %7              ;FINISHED
D146.B: NEG     %0              ;MAKE +
        BIS     %0,MW4          ;SET BITS
        BIS     #MP,CDSTAT      ;CONFLICT FLAG - MICROBRANCH OR INTERRUPTS/SCRATCH REGISTER
        RTS     %7              ;FINISHED
        ;
        .PAGE
        .SBTTL  CM:, EN:, & NA: EVALUATION
        ;
        ;COMMENTS
        ;
D150.:  RTS     %7              ;FINISHED
        ;
        ;END OF PROGRAM
        ;
D155.:  RTS     %7              ;FINISHED
        ;
        ;NAME EVALUATION
        ;
D165.:  CALL    FSBR70,P1,P2,#GP
        GTIVAL  #GP,#1,#0       ;GET GP(1)
        MOV     %0,GP2          ;SAVE INITIAL VALUE
        ;
        ;EVALUATE EACH TERM
        ;
        FOR     X,#1,GP,D166.
        BIS     #M.,CDSTAT
        RTS     %7              ;FINISHED
        ;
        ;EVALUATE EACH NAME SPECIFIED
        ;
D166.:  MOV     GP2,GP1         ;SET UP BOUNDS
        GTIVAL  #GP,X,#1        ;GET GP(X+1)
        MOV     %0,GP2          ;SAVE UPPER BOUNDS
        ;
        ;FIND EQUAL SIGN
        ;
        CALL    FSBR64,GP1,GP2,#EQUAL
        MOV     %0,NH           ;SAVE POSITION OF '='
        BNE     D166.A          ;IF FOUND '=' - SKIP
        ERROR   #5              ;NO '='
        RTS     %7              ;FINISHED
        ;
        ;GET VALUE FOR NAME
        ;
D166.A: CALL    FSBR90,NH,GP2
        MOV     %0,NG           ;SAVE TEMPORARILY
```

```
        ;
        ;GET NAME STRING AND INSERT IN SYMBOL TABLE
        ;
        CALL    FSBR74,GP1,NH,#STRING
        TST     %0              ;IS THERE A STRING ?
        BNE     D166.B          ;IF SO - SKIP
        ERROR   #30.
        RTS     %7              ;FINISHED
D166.B: CALL    PLCSYM,#STRING,NG
        BIS     #M.,CDSTAT
        RTS     %7              ;FINISHED
        ;
        .PAGE
        .SBTTL  IN:, MA:, & PS: EVALUATION
        ;
        ;INSTRUCTION MODE EVALUATION
        ;
D170.:  BIS     #ICODE,CDSTAT   ;ICODE DEFINED
        CALL    FSBR60,P1,P2,#GP,#GPSYM
        FOR     X,#1,GP,D171.
        RTS     %7              ;FINISHED
        ;
        ;EVALUATE EACH TERM
        ;
D171.:  GTIVAL  #GP,X,#0        ;GET GP(X)
        CALL    SCAN4B,#INSTBL,%0
        TST     ER              ;TERM THERE ?
        BEQ     D171.A          ;IF SO - SKIP
        ERROR   #7
        RTS     %7              ;FINISHED
D171.A: TST     %0              ;TEST SIGN OF RESULT
        BMI     D171.B          ;IF [-] - SKIP
        BIS     %0,MW7          ;SET BITS
        RTS     %7              ;FINISHED
D171.B: NEG     %0              ;MAKE +
        BIS     %0,MW6          ;SET BITS
        RTS     %7              ;FINISHED
        ;
        ;MICROBRANCH ADDRESS EVALUATION
        ;
D175.:  BIS     #ICODE,CDSTAT   ;ICODE DEFINED
        CALL    FSBR90,P1,P2
        BIC     #170000,%0      ;ONLY 12 BITS WORTH
        BIS     %0,MW6          ;SET BITS
        RTS     %7              ;FINISHED
        ;
        ;PRIORITY SEQUENCE EVALUATION
        ;
D180.:  BIS     #ICODE,CDSTAT   ;ICODE DEFINED
        CALL    FSBR60,P1,P2,#GP,#GPSYM
        FOR     X,#1,GP,D181.
        RTS     %7              ;FINISHED
        ;
        ;EVALUATE EACH TERM
        ;
D181.:  GTIVAL  #GP,X,#0        ;GET GP(X)
        CALL    SCAN4B,#PSQTBL,%0
        TST     ER              ;TERM THERE ?
        BEQ     D181.A          ;IF SO - SKIP
        ERROR   #7
        RTS     %7              ;FINISHED
D181.A: BIS     %0,MW7          ;SET BITS
        RTS     %7              ;FINISHED
        ;
        .PAGE
        .SBTTL  VECTOR EVALUATION
        ;
D185.:  JSR     %7,OUTCHK       ;UPDATE TO THIS POINT
        BIS     #VCODE,CDSTAT   ;VCODE DEFINED
        BIS     #M.,CDSTAT
        CALL    FSBR70,P1,P2,#GP
        GTIVAL  #GP,#1,#0       ;GET GP(1)
        MOV     %0,GP2          ;SAVE FIRST VALUE
        ;
        ;EVALUATE EACH TERM
        ;
```

```
        FOR     X,#1,GP,D186.
        RTS     %7              ;FINISHED
        ;
        ;EVALUATE TERMS  VECT =  V1 / V2
        ;
D186.:  MOV     GP2,GP1         ;SET LOWER  BOUNDS
        GTIVAL  #GP,X,#1        ;GET GP(X+1)
        MOV     %0,GP2          ;SAVE UPPER BOUNDS
        ;
        ;FIND '='
        ;
        CALL    FSBR64,GP1,GP2,#EQUAL
        MOV     %0,NH           ;SAVE POSITION
        BNE     D186.A          ;IF FOUND '=' - BRANCH
        ERROR   #5
        RTS     %7              ;FINISHED
        ;
        ;CHECK VECTOR SPECIFIER
        ;
D186.A: CALL    FSBR74,GP1,NH,#STRING
        CALL    SCAN4B,#VECTBL,#STRING
        TST     ER              ;TERM DEFINED ?
        BEQ     D186.B          ;IF SO - SKIP
        ERROR   #15.
        RTS     %7              ;FINISHED
D186.B: MOV     %0,VE           ;SAVE OFFSET
        BPL     D186.D          ;IF VECTORS - SKIP
        ;
        ;VECTOR GROUP SPECIFIER
        ;
        CALL    FSBR90,NH,GP2   ;ELSE VECTOR GROUP
        MOV     %0,VE           ;SAVE RESULT
        BMI     D186.C          ;IF <0 - ERROR
        CMP     %0,#10
        BHIS    D186.C          ;IF >7 - ERROR
        MOV     VECT,%0         ;ADDRESS
        ASH     #2,%0           ;4 BYTES PER ENTRY
        MOV     #-1,VECTV(%0)   ;PLACE SPECIAL #
        ADD     #2,%0
        MOV     VE,VECTV(%0)    ;PLACE GROUP #
        INC     VECT
        MOV     #-1,VY          ;RESET BLOCK SPECIFIER
        RTS     %7              ;FINISHED
D186.C: ERROR   #46.            ;VECTOR GROUP NOT 0-7
        RTS     %7              ;FINISHED
        ;
        ;FIND '/'
        ;
D186.D: CALL    FSBR64,NH,GP2,#DIVIDE
        MOV     %0,VW           ;SAVE POSITION OF '/'
        BNE     D186.E          ;IF FOUND '/' - BRANCH
        ERROR   #14.
        RTS     %7              ;FINISHED
        ;
        ;EVALUATE INTERRUPT VECTOR
        ;
D186.E: CALL    FSBR90,VW,GP2
        MOV     %0,VZ           ;SAVE RESULT
        ;
        ;EVALUATE VECTOR BRANCH ADDRESS
        ;
        CALL    FSBR90,NH,VW
        MOV     %0,VX           ;SAVE RESULT
        TST     VY              ;DEFINED ?
        BPL     D186.F          ;IF SO - SKIP
        MOV     VX,VY           ;SET PARAMETER
D186.F: CMPB    VY+1,VX+1       ;MUST BE SAME 256. WORD BLOCK
        BEQ     D186.G          ;IF SO - SKIP
        ERROR   #16.
D186.G: MOV     VECT,%0         ;COUNT
        ASH     #2,%0           ;4 BYTES PER VECTOR
        MOV     VE,VECTV(%0)    ;PLACE VALUE
        ADD     #2,%0           ;NEXT ADDRESS
        MOVB    VZ,VECTV(%0)    ;SET LOW BYTE
        INC     %0
        MOVB    VX,VECTV(%0)    ;SET HIGH BYTE
```

```
        INC     VECT            ;NUMBER OF TERMS
        RTS     %7              ;FINISHED
        ;
        .PAGE
        .SBTTL  MICROWORD EVALUATION
        ;
        ;FIND ALL COMMA'S
        ;
D190.:  BIS     #ICODE,CDSTAT   ;ICODE DEFINED
        CALL    FSBR70,P1,P2
        GTIVAL  #GP,#1,#0       ;GET GP(1)
        MOV     %0,GP2          ;LOWER BOUNDS
        ;
        ;DO D191. FOR EACH TERM
        ;
        FOR     X,#1,GP,D191.
        RTS     %7              ;FINISHED
        ;
        ;EVALUATE EACH TERM
        ;
D191.:  MOV     GP2,GP1         ;SET LOWER
        GTIVAL  #GP,X,#1        ;GET GP(X+1)
        MOV     %0,GP2          ;SET UPPER
        ;
        ;SCAN FOR '='
        ;
        CALL    FSBR64,GP1,GP2,#EQUAL
        MOV     %0,NH           ;SAVE POSITION
        BNE     D191.A          ;IF FOUND '=' - BRANCH
        ERROR   #5
        RTS     %7              ;FINISHED
D191.A: CALL    FSBR74,GP1,NH,#STRING
        CALL    SCAN4B,#MCWTBL,#STRING
        TST     ER              ;VALID TERM ?
        BEQ     D191.B          ;IF SO - SKIP
        ERROR   #17.
        RTS     %7              ;FINISHED
D191.B: MOV     %0,NG           ;SAVE RESULT
        CALL    FSBR90,NH,GP2
        BIS     %0,@NG          ;SET BITS IN APPROPRIATE MW
        RTS     %7              ;FINISHED
        ;
```

```
        .PAGE
        .SBTTL  GLOBLES AND BUFFERS
        ;
        BFLEN=2048.     ;LENGTH OF SCANNING STRING
INBUFF: .BLKB   BFLEN           ;INPUT STRING
IBUFND: .WORD   0
        ;
        NSYMBS=400.     ;SYMBOL TABLE ENTRIES
        .NLIST  BIN
SYMTBL: .ASCII  /.       /      ;THE PROGRAM COUNTER !
        .LIST   BIN
MPC:    .WORD   0
        .BLKW   5*<NSYMBS-1>    ;SPACE FOR (NSYMBS-1) SYMBOLS !
        SYMEND=.                ;SYMBOL TABLE END
        .WORD   0       ;TABLE TERMINATION
        ;
        .PAGE
        .SBTTL  SEQUENCE DEFINITIONS
        ;
        .NLIST  BIN
SEQTBL: .ASCII  /CM/    ;COMMENTS
        .WORD   1
        .ASCII  /AR/    ;ARITHMETIC
        .WORD   2
        .ASCII  /CC/    ;CONDITION CODES
        .WORD   3
        .ASCII  /PC/    ;PROCESSOR CONTROL
        .WORD   4
        .ASCII  /IO/    ;INPUT/OUTPUT
        .WORD   5
        .ASCII  /BR/    ;MICRO BRANCHING
        .WORD   6
        .ASCII  /SP/    ;SPECIAL CONTROL
        .WORD   7
        .ASCII  /IN/    ;INSTRUCTION MODE
        .WORD   10
        .ASCII  /MA/    ;MICROBRANCH ADDRESSING OOF INSTRUCTION
        .WORD   11
        .ASCII  /PS/    ;PRIORITY SEQUENCE
        .WORD   12
        .ASCII  /MW/    ;EXPLICIT MICROWORD DEFINITION VIA BIT SETTING
        .WORD   13
        .ASCII  /VE/    ;VECTOR SPECIFIER
        .WORD   14
        .ASCII  /EN/    ;END OF PROGRAM
        .WORD   20
        .ASCII  /**/    ;BEGINNING OF LINE
        .WORD   21
        .ASCII  /.=/    ;MICROPROGRAM COUNTER
        .WORD   22
        .ASCII  /NA/    ;NAME SPECIFIER
        .WORD   23
        .ASCII  /PG/    ;PAGE OPERATOR
        .WORD   24
        .ASCII  /TT/    ;TITLE DEFINITION
        .WORD   25
        .ASCII  /SB/    ;SUBTITLE DEFINITION
        .WORD   26
        .WORD   0       ;TABLE TERMINATION
        .LIST   BIN
        ;
        .PAGE
        .SBTTL  SHIFT DEFINITIONS
        ;
        .NLIST  BIN
SHFTBL: .ASCII  /0     /        ;0
        .WORD   0
        .ASCII  /1     /        ;1
        .WORD   1
        .ASCII  /MSBR  /        ;MOST SIGNIFICANT BIT OF REGISTER IN ALU
        .WORD   2
        .ASCII  /MSBQ  /        ;MOST SIGNIFICANT BIT OF Q REGISTER
        .WORD   3
        .ASCII  /C     /        ;C BIT OF PSR
        .WORD   4
        .ASCII  /PNXPV /        ;(ALU N BIT) XOR (ALU V BIT)
```

```
        .WORD   5
        .ASCII  /LSBR  /        ;LEAST SIGNIFICANT BIT OF REGISTER IN ALU
        .WORD   6
        .ASCII  /LSBQ  /        ;LEAST SIGNIFICANT BIT OF Q REGISTER
        .WORD   7
        .WORD   0               ;TABLE TERMINATION
        .LIST   BIN
        ;
        .PAGE
        .SBTTL  CARRY-IN DEFINITIONS
        ;
        .NLIST  BIN
CRYTBL: .ASCII  *0     *        ;0
        .WORD   0
        .ASCII  *1     *        ;1
        .WORD   100000
        .ASCII  *C     *        ;PSR CARRY BIT
        .WORD   10000
        .ASCII  *V     *        ;PSR OVERFLOW BIT
        .WORD   20000
        .ASCII  *Z     *        ;PSR ZERO BIT
        .WORD   30000
        .ASCII  *N     *        ;PSR SIGN BIT
        .WORD   40000
        .ASCII  *C'    *        ;PREVIOUS CYCLE ALU CARRY
        .WORD   50000
        .ASCII  *SR67W *        ;SOURCE REGISTER 6 OR 7 OR WORD [L]
        .WORD   60000
        .ASCII  *DR67W *        ;DESTINATION REGISTER 6 OR 7 OR WORD [L]
        .WORD   70000
        .ASCII  */0    *        ;COMPLEMENTS
        .WORD   100000
        .ASCII  */1    *
        .WORD   0
        .ASCII  */C    *
        .WORD   110000
        .ASCII  */V    *
        .WORD   120000
        .ASCII  */Z    *
        .WORD   130000
        .ASCII  */N    *
        .WORD   140000
        .ASCII  */C'   *
        .WORD   150000
        .ASCII  */SR67W*
        .WORD   160000
        .ASCII  */DR67W*
        .WORD   170000
        .WORD   0               ;TABLE TERMINATION
        .LIST   BIN
        ;
        .PAGE
        .SBTTL  PROCESSOR MODE DEFINITIONS
        ;
        .NLIST  BIN
PRCTBL: .ASCII  /WIOH/  ;WAIT FOR IO CLOCK [H]
        .WORD   10000
        .ASCII  /WIOB/  ;SAME AS WIOH
        .WORD   10000
        .ASCII  /WIOL/  ;WAIT FOR IO CLOCK [L]
        .WORD   20000
        .ASCII  /WIOE/  ;SAME AS WIOL
        .WORD   20000
        .ASCII  /HALT/  ;STOP PROCESSOR
        .WORD   30000
        .ASCII  /CCL /  ;STEP COUNTER BY 1
        .WORD   40000
        .ASCII  /CFP /  ;CLEAR FEO PRIORITIES
        .WORD   100000
        .ASCII  /MO  /  ;NO SPECIAL MATH CONTROL
        .WORD   0
        .ASCII  /MR  /  ;FOR RESTORING DIVISION
        .WORD   2000
        .ASCII  /MM  /  ;FOR MULTIPLICATION
        .WORD   4000
        .ASCII  /MNR /  ;FOR NON-RESTORING DIVISION
```

```
        .WORD   6000
        .ASCII  /LB  /  ;LOW BYTE <7:0>
        .WORD   0
        .ASCII  /LW  /  ;LOW WORD <15:0>
        .WORD   200
        .ASCII  /UB  /  ;UPPER BYTE <23:16>
        .WORD   400
        .ASCII  /UW  /  ;UPPER WORD <31:16>
        .WORD   600
        .ASCII  /LWB /  ;LOW WORD/BYTE  <15:0>/<7:0>
        .WORD   1000
        .ASCII  /UWB /  ;UPER WORD/BYTE  <31:16>/<23:16>
        .WORD   1200
        .ASCII  /3B  /  ;THREE BYTES  <23:0>
        .WORD   1400
        .ASCII  /4B  /  ;4 BYTES  <31:0>
        .WORD   1600
        .ASCII  /CB  /  ;CONTROL BIT (CONSOLE BIT)
        .WORD   100
        .WORD   0               ;TABLE TERMINATION
        .LIST   BIN
        ;
        .PAGE
        .SBTTL  BUS CONTROL OPERATIONS
        ;
        .NLIST  BIN
BUSTBL: .ASCII  /DATI    /      ;DATA INPUT
        .WORD   100000
        .ASCII  /DATIP   /      ;DATA INPUT PAUSE-OUTPUT
        .WORD   100100
        .ASCII  /DATO    /      ;DATA OUTPUT
        .WORD   100200
        .ASCII  /DATOB   /      ;BYTE ENABLE OUTPUT
        .WORD   100300
        .ASCII  /DATIR   /      ;INSTRUCTION FETCH
        .WORD   100400
        .ASCII  /LDATIR  /      ;LOOK-AHEAD INSTRUCTION FETCH
        .WORD   100500
        .ASCII  /INT     /      ;INTERRUPT VECTOR INPUT
        .WORD   170600
        .ASCII  /INIT    /      ;BUS INITIALIZATION
        .WORD   170700
        .ASCII  /IDATI   /      ;IMMEDIATE MODE OPERATIONS
        .WORD   101000
        .ASCII  /IDATIP  /
        .WORD   101100
        .ASCII  /IDATO   /
        .WORD   101200
        .ASCII  /IDATOB  /
        .WORD   101300
        .ASCII  /IDATIR  /
        .WORD   101400
        .ASCII  /ILDATIR /
        .WORD   101500
        .ASCII  /IINT    /
        .WORD   171600
        .ASCII  /INTCK   /      ;IMMEDIATE INTERRUPT CHECK
        .WORD   171700
        .WORD   0               ;TABLE TERMINATION
        .LIST   BIN
        ;
        .PAGE
        .SBTTL  MEMORY MANAGEMENT CONTROL
        ;
        .NLIST  BIN
MMGTBL: .ASCII  /BYT /  ;ENABLE BYTE OPERATION
        .WORD   2000
        .ASCII  /NDSP/  ;ENABLE NDA OR SP MODES
        .WORD   4000
        .ASCII  /CM  /  ;CURRENT MODE
        .WORD   0
        .ASCII  /PM  /  ;PREVIOUS MODE
        .WORD   10000
        .ASCII  /K   /  ;KERNAL MODE
        .WORD   20000
        .ASCII  /U   /  ;USER MODE
```

```
        .WORD   30000
        .ASCII  /CMI /  ;CURRENT MODE (INHIBITED BY MAINTENANCE)
        .WORD   40000
        .ASCII  /PMI /  ;PREVIOUS MODE (INHIBITED BY MAINTENANCE)
        .WORD   50000
        .ASCII  /KI  /  ;KERNAL MODE (INHIBITED BY MAINTENANCE)
        .WORD   60000
        .ASCII  /OFF /  ;MEMORY MANAGEMENT OFF
        .WORD   70000
        .WORD   0               ;TABLE TERMINATION
        .LIST   BIN
        ;
        .PAGE
        .SBTTL  MICROBRANCH DEFINITIONS
        ;
        .NLIST  BIN
MCBTBL: .ASCII  /BROC  /        ;BRANCH ORRED WITH CONDITION
        .WORD   0
        .ASCII  /CBNC  /        ;COND. BRANCH (ADD. ORRED WITH NC)
        .WORD   10000
        .ASCII  /CBZV  /        ;COND. BRANCH (ADD. ORRED WITH ZV)
        .WORD   20000
        .ASCII  /CBNZVC/        ;COND. BRANCH (ADD. ORRED WITH NZVC)
        .WORD   30000
        .ASCII  /CBR   /        ;COND. BRANCH ELSE REPEAT
        .WORD   40000
        .ASCII  /CBN   /        ;COND. BRANCH ELSE NEXT
        .WORD   50000
        .ASCII  /CJR   /        ;COND. JSR ELSE REPEAT
        .WORD   60000
        .ASCII  /CJN   /        ;COND. JSR ELSE NEXT
        .WORD   70000
        .ASCII  /CNR   /        ;COND. NEXT ELSE REPEAT
        .WORD   100000
        .ASCII  /CPS   /        ;COND. POP STACK  NEXT ALWAYS
        .WORD   110000
        .ASCII  /CRR   /        ;COND. RETURN ELSE REPEAT
        .WORD   120000
        .ASCII  /CRN   /        ;COND. RETURN ELSE NEXT
        .WORD   130000
        .ASCII  /CBIC  /        ;COND. BRANCH VIA INSTRUCTION - CURRENT PRIORITY
        .WORD   140000
        .ASCII  /CJIC  /        ;COND. JSR VIA INSTRUCTION - CURRENT PRIORITY
        .WORD   150000
        .ASCII  /CBIN  /        ;CBIC - NEXT PRIORITY
        .WORD   160000
        .ASCII  /CJIN  /        ;CJIC - NEXT PRIORITY
        .WORD   170000
        .WORD   0               ;TABLE TERMINATION
        .LIST   BIN
        ;
        ;INSTRUCTION MODE ADDRESS SELECT DEFINITIONS
        ;
        .NLIST  BIN
IBCTBL: .ASCII  /VEC /  ;USE VECTOR REGISTERS
        .WORD   0
        .ASCII  /SCR /  ;USE SCRATCH REGISTERS
        .WORD   4000
        .ASCII  /INST/  ;USE INSTRUCTION DECODE
        .WORD   6000
        .WORD   0       ;TABLE TERMINATION
        .LIST BIN
        ;
        .PAGE
        .SBTTL  BRANCH CONDITIONALS
        ;
        .NLIST  BIN
        ;PROCESSOR STATUS REGISTER BRANCHES
BRCTBL: .ASCII  *C     *        ;C
        .WORD   0
        .ASCII  *BCS   *
        .WORD   0
        .ASCII  *BLO   *
        .WORD   0
        .ASCII  *V     *        ;V
        .WORD   1
```

```
        .ASCII  *BVS    *
        .WORD   1
        .ASCII  *Z      *       ;Z
        .WORD   2
        .ASCII  *BEQ    *
        .WORD   2
        .ASCII  *N      *       ;N
        .WORD   3
        .ASCII  *BMI    *
        .WORD   3
        .ASCII  *COZ    *       ;C OR Z
        .WORD   4
        .ASCII  *BLOS   *
        .WORD   4
        .ASCII  *CXN    *       ;C XOR N
        .WORD   5
        .ASCII  *VXN    *       ;V XOR N
        .WORD   6
        .ASCII  *BLT    *
        .WORD   6
        .ASCII  *VNZ    *       ;Z OR   N XOR V
        .WORD   7
        .ASCII  *BLE    *
        .WORD   7
        ;
        ;PREVIOUS MICROCYCLE CONDITIONALS
        .ASCII  *C'     *       ;C'
        .WORD   10
        .ASCII  *BCS'   *
        .WORD   10
        .ASCII  *BLO'   *
        .WORD   10
        .ASCII  *V'     *       ;V'
        .WORD   11
        .ASCII  *BVS'   *
        .WORD   11
        .ASCII  *Z'     *       ;Z'
        .WORD   12
        .ASCII  *BEQ'   *
        .WORD   12
        .ASCII  *N'     *       ;N'
        .WORD   13
        .ASCII  *BMI'   *
        .WORD   13
        .ASCII  *COZ'   *       ;C' OR Z'
        .WORD   14
        .ASCII  *BLOS'  *
        .WORD   14
        .ASCII  *CXN'   *       ;C' XOR N'
        .WORD   15
        .ASCII  *VXN'   *       ;V' XOR N'
        .WORD   16
        .ASCII  *BLT'   *
        .WORD   16
        .ASCII  *VNZ'   *       ;Z' OR   N' XOR V'
        .WORD   17
        .ASCII  *BLE'   *
        .WORD   17
        ;
        ;VARIOUS TESTABLE BITS
        .ASCII  *0      *       ;0
        .WORD   20
        .ASCII  *TB     *       ;TRACE BIT
        .WORD   21
        .ASCII  *LKDC   *       ;LOOK-AHEAD CONDITION
        .WORD   22
        .ASCII  *PFD    *       ;POWER FAIL DOWN
        .WORD   23
        .ASCII  *IBRD   *       ;BRANCH INSTRUCTION DECODE
        .WORD   24
        .ASCII  *IP     *       ;INTERRUPT PENDING
        .WORD   25
        .ASCII  *SC0    *       ;EVEN NUMBERED SOURCE REGISTER
        .WORD   26
        .ASCII  *BYT    *       ;BYT CONDITIONAL
        .WORD   27
```

```
        .ASCII  *MMGT  *        ;MEMORY MANAGEMENT ENABLE
        .WORD   30
        .ASCII  *CUM   *        ;CURRENT USER MODE
        .WORD   31
        .ASCII  *PUM   *        ;PREVIOUS USER MODE
        .WORD   32
        .ASCII  *LKD   *        ;INSTRUCTION LOOOK-AHEAD
        .WORD   33
        .ASCII  *IEO   *        ;INTERNAL MODE INTERRRUPT PENDING
        .WORD   34
        .ASCII  *BINT  *        ;BUS LEVEL INTERRUPT
        .WORD   35
        .ASCII  *EIL   *        ;EXTERNAL INTERRUPT LINE
        .WORD   36
        .ASCII  *CNTR  *        ;MICROCYCLE COUNTER
        .WORD   37
        .ASCII  *1     *        ;1
        .WORD   60
        ;
        ;COMPLEMENTS OF BRANCH CONDITIONALS
        .ASCII  */C    *
        .WORD   40
        .ASCII  *BCC   *
        .WORD   40
        .ASCII  *BHIS  *
        .WORD   40
        .ASCII  */V    *
        .WORD   41
        .ASCII  *BVC   *
        .WORD   41
        .ASCII  */Z    *
        .WORD   42
        .ASCII  *BNE   *
        .WORD   42
        .ASCII  */N    *
        .WORD   43
        .ASCII  *BPL   *
        .WORD   43
        .ASCII  */COZ  *
        .WORD   44
        .ASCII  *BHI   *
        .WORD   44
        .ASCII  */CXN  *
        .WORD   45
        .ASCII  */VXN  *
        .WORD   46
        .ASCII  *BGE   *
        .WORD   46
        .ASCII  */VNZ  *
        .WORD   47
        .ASCII  *BGT   *
        .WORD   47
        .ASCII  */C'   *
        .WORD   50
        .ASCII  *BCC'  *
        .WORD   50
        .ASCII  *BHIS' *
        .WORD   50
        .ASCII  */V'   *
        .WORD   51
        .ASCII  *BVC'  *
        .WORD   51
        .ASCII  */Z'   *
        .WORD   52
        .ASCII  *BNE'  *
        .WORD   52
        .ASCII  */N'   *
        .WORD   53
        .ASCII  *BPL'  *
        .WORD   53
        .ASCII  */COZ' *
        .WORD   54
        .ASCII  *BHI'  *
        .WORD   54
        .ASCII  */CXN' *
        .WORD   55
```

```
        .ASCII  */VXN' *
        .WORD   56
        .ASCII  *BGE'  *
        .WORD   56
        .ASCII  */VNZ' *
        .WORD   57
        .ASCII  *BGT'  *
        .WORD   57
        .ASCII  */0    *
        .WORD   60
        .ASCII  */TB   *
        .WORD   61
        .ASCII  */LKDC *
        .WORD   62
        .ASCII  */PFD  *
        .WORD   63
        .ASCII  */IBRD *
        .WORD   64
        .ASCII  */IP   *
        .WORD   65
        .ASCII  */SC0  *
        .WORD   66
        .ASCII  */BYT  *
        .WORD   67
        .ASCII  */MMGT *
        .WORD   70
        .ASCII  */CUM  *
        .WORD   71
        .ASCII  */PUM  *
        .WORD   72
        .ASCII  */LKD  *
        .WORD   73
        .ASCII  */IEO  *
        .WORD   74
        .ASCII  */BINT *
        .WORD   75
        .ASCII  */EIL  *
        .WORD   76
        .ASCII  */CNTR *
        .WORD   77
        .ASCII  */1    *
        .WORD   20
        .WORD   0       ;TABLE TERMINATION
        .LIST   BIN
        ;
        .PAGE
        .SBTTL  SPECIAL OPERATIONS
        ;
        .NLIST  BIN
SPLTBL: .ASCII  /CI  /  ;CLEAR INTERRUPT
        .WORD   1000
        .ASCII  /ILE /  ;
        .WORD   200
        .ASCII  /ILD /  ;
        .WORD   400
        .ASCII  /ILDE/  ;
        .WORD   600
        .ASCII  /ILN /  ;
        .WORD   0
        .ASCII  /ST0 /  ;LOWEST PROGRAMMABLE PRIORITY
        .WORD   -100
        .ASCII  /SE7 /
        .WORD   -100
        .ASCII  /ST1 /
        .WORD   -200
        .ASCII  /SE6 /
        .WORD   -200
        .ASCII  /ST2 /
        .WORD   -400
        .ASCII  /SE5 /
        .WORD   -400
        .ASCII  /ST3 /
        .WORD   -1000
        .ASCII  /SE4 /
        .WORD   -1000
        .ASCII  /ST4 /
```

```
        .WORD   -2000
        .ASCII  /SE3 /
        .WORD   -2000
        .ASCII  /ST5 /
        .WORD   -4000
        .ASCII  /SE2 /
        .WORD   -4000
        .ASCII  /ST6 /
        .WORD   -10000
        .ASCII  /SE1 /
        .WORD   -10000
        .ASCII  /ST7 /  ;HIGHEST PROGRAMMABLE PRIORITY
        .WORD   -20000
        .ASCII  /SE0 /
        .WORD   -20000
        .ASCII  /CLR /  ;CLEAR ALL PROGRAMMED PRIORITIES
        .WORD   -40000
        .WORD   0       ;TABLE TERMINATION
        .LIST   BIN
        ;
        .PAGE
        .SBTTL  MICROCODE WORD DEFINITIONS
        ;
        .NLIST  BIN
MCWTBL: .ASCII  /MW0 /  ;MW0 - MW7
        .WORD   MW0
        .ASCII  /MW1 /
        .WORD   MW1
        .ASCII  /MW2 /
        .WORD   MW2
        .ASCII  /MW3 /
        .WORD   MW3
        .ASCII  /MW4 /
        .WORD   MW4
        .ASCII  /MW5 /
        .WORD   MW5
        .ASCII  /MW6 /
        .WORD   MW6
        .ASCII  /MW7 /
        .WORD   MW7
        .WORD   0       ;TABLE TERMINATION
        .LIST   BIN
        ;
        .PAGE
        .SBTTL  VECTOR DEFINITIONS
        ;
        .NLIST  BIN
VECTBL: .ASCII  /VGRP/  ;VECTOR GROUP
        .WORD   -1
        .ASCII  /V0  /  ;V0-V37  VECTOR BRANCH ADDRESSES
        .WORD   0
        .ASCII  /V1  /
        .WORD   1
        .ASCII  /V2  /
        .WORD   2
        .ASCII  /V3  /
        .WORD   3
        .ASCII  /V4  /
        .WORD   4
        .ASCII  /V5  /
        .WORD   5
        .ASCII  /V6  /
        .WORD   6
        .ASCII  /V7  /
        .WORD   7
        .ASCII  /V10 /
        .WORD   10
        .ASCII  /V11 /
        .WORD   11
        .ASCII  /V12 /
        .WORD   12
        .ASCII  /V13 /
        .WORD   13
        .ASCII  /V14 /
        .WORD   14
        .ASCII  /V15 /
```

```
        .WORD   15
        .ASCII  /V16 /
        .WORD   16
        .ASCII  /V17 /
        .WORD   17
        .ASCII  /V20 /
        .WORD   20
        .ASCII  /V21 /
        .WORD   21
        .ASCII  /V22 /
        .WORD   22
        .ASCII  /V23 /
        .WORD   23
        .ASCII  /V24 /
        .WORD   24
        .ASCII  /V25 /
        .WORD   25
        .ASCII  /V26 /
        .WORD   26
        .ASCII  /V27 /
        .WORD   27
        .ASCII  /V30 /
        .WORD   30
        .ASCII  /V31 /
        .WORD   31
        .ASCII  /V32 /
        .WORD   32
        .ASCII  /V33 /
        .WORD   33
        .ASCII  /V34 /
        .WORD   34
        .ASCII  /V35 /
        .WORD   35
        .ASCII  /V36 /
        .WORD   36
        .ASCII  /V37 /
        .WORD   37
        .WORD   0       ;TABLE TERMINATION
        .LIST   BIN
        ;
        .PAGE
        .SBTTL  BEFORE '=' ARITHMETIC DEFINITIONS
        ;
        .NLIST  BIN
BEFTBL: .ASCII  /Q     /        ;Q REGISTER
        .WORD   0,4             ;CODE,SEQUENCE
        .ASCII  /OUT   /        ;ALU OUTPUT
        .WORD   0,7
        .ASCII  /D     /        ;DATA REGISTER
        .WORD   1000,7
        .ASCII  /AD    /        ;ADDRESS REGISTER
        .WORD   2000,7
        .ASCII  /ADX   /        ;ADDRESS EXTENSION REGISTER
        .WORD   3000,7
        .ASCII  /PSR   /        ;PROCESSOR STATUS REGISTER
        .WORD   4000,7
        .ASCII  /CNTR  /        ;CYCLE COUNTER
        .WORD   5000,7
        .ASCII  /IR    /        ;INSTRUCTION REGISTER
        .WORD   6000,7
        .ASCII  /SOR   /        ;STACK OVERFLOW RREGISTER
        .WORD   7000,7
        .ASCII  /SCR0  /        ;SCRATCH REGISTERS 0-7
        .WORD   100,7
        .ASCII  /SCE7  /
        .WORD   100,7
        .ASCII  /SCR1  /
        .WORD   110,7
        .ASCII  /SCE6  /
        .WORD   110,7
        .ASCII  /SCR2  /
        .WORD   120,7
        .ASCII  /SCE5  /
        .WORD   120,7
        .ASCII  /SCR3  /
        .WORD   130,7
```

```
        .ASCII  /SCE4  /
        .WORD   130,7
        .ASCII  /SCR4  /
        .WORD   140,7
        .ASCII  /SCE3  /
        .WORD   140,7
        .ASCII  /SCR5  /
        .WORD   150,7
        .ASCII  /SCE2  /
        .WORD   150,7
        .ASCII  /SCR6  /
        .WORD   160,7
        .ASCII  /SCE1  /
        .WORD   160,7
        .ASCII  /SCR7  /
        .WORD   170,7
        .ASCII  /SCE0  /
        .WORD   170,7
        .ASCII  /R0B   /        ;REGISTERS 0-17 B-PORT
        .WORD   6000,3
        .ASCII  /R1B   /
        .WORD   6100,3
        .ASCII  /R2B   /
        .WORD   6200,3
        .ASCII  /R3B   /
        .WORD   6300,3
        .ASCII  /R4B   /
        .WORD   6400,3
        .ASCII  /R5B   /
        .WORD   6500,3
        .ASCII  /R6B   /
        .WORD   6600,3
        .ASCII  /R7B   /
        .WORD   6700,3
        .ASCII  /R10B  /
        .WORD   7000,3
        .ASCII  /R11B  /
        .WORD   7100,3
        .ASCII  /R12B  /
        .WORD   7200,3
        .ASCII  /R13B  /
        .WORD   7300,3
        .ASCII  /R14B  /
        .WORD   7400,3
        .ASCII  /R15B  /
        .WORD   7500,3
        .ASCII  /R16B  /
        .WORD   7600,3
        .ASCII  /R17B  /
        .WORD   7700,3
        .ASCII  /RB    /        ;B-PORT REGISTER
        .WORD   4000,10
        .WORD   0               ;TABLE TERMINATION
        .LIST   BIN
        ;
        .PAGE
        .SBTTL  AFTER '=' ARITHMETIC DEFINITIONS
        ;
        .NLIST  BIN
AFRTBL: .ASCII  /D     /        ;DATA REGISTER
        .WORD   1,5
        .ASCII  /DSWB  /        ;SWAP BYTES OF DATA REGISTER
        .WORD   10000,5
        .ASCII  /DEX   /        ;EXCHANGE BITS
        .WORD   20000,5
        .ASCII  /DSX7  /        ;SIGN EXTEND DATA FROM BIT 7
        .WORD   30000,5
        .ASCII  /DSX5  /        ;SIGN EXTEND DATA FROM BIT 5
        .WORD   40000,5
        .ASCII  /IR    /        ;INSTRUCTION REGISTER
        .WORD   50000,5
        .ASCII  /ISX7  /        ;SIGN EXTEND INSTRUCTION FROM BIT 7
        .WORD   60000,5
        .ASCII  /IR5   /        ;BITS <5:0> OF INSTRUCTION REGISTER
        .WORD   70000,5
        .ASCII  /PV    /        ;PRIORITY VECTOR BITS <7:0>
```

```
        .WORD   110000,5
        .ASCII  /PSR   /        ;PROCESSOR STATUS REGISTER
        .WORD   120000,5
        .ASCII  /SLR   /        ;STACK LIMIT REGISTER
        .WORD   130000,5
        .ASCII  /CNTR  /        ;CYCLE COUNTER
        .WORD   140000,5
        .ASCII  /SW1   /        ;SWITCHES <15:0>
        .WORD   160000,5
        .ASCII  /SW2   /        ;SWITCHES <21:16> AND CONSOLE CONTROL
        .WORD   170000,5
        .ASCII  /SCR0  /        ;SCRATCH REGISTERS 0-7
        .WORD   100000,5
        .ASCII  /SCE7  /
        .WORD   100000,5
        .ASCII  /SCR1  /
        .WORD   100001,5
        .ASCII  /SCE6  /
        .WORD   100001,5
        .ASCII  /SCR2  /
        .WORD   100002,5
        .ASCII  /SCE5  /
        .WORD   100002,5
        .ASCII  /SCR3  /
        .WORD   100003,5
        .ASCII  /SCE4  /
        .WORD   100003,5
        .ASCII  /SCR4  /
        .WORD   100004,5
        .ASCII  /SCE3  /
        .WORD   100004,5
        .ASCII  /SCR5  /
        .WORD   100005,5
        .ASCII  /SCE2  /
        .WORD   100005,5
        .ASCII  /SCR6  /
        .WORD   100006,5
        .ASCII  /SCE1  /
        .WORD   100006,5
        .ASCII  /SCR7  /
        .WORD   100007,5
        .ASCII  /SCE0  /
        .WORD   100007,5
        .ASCII  /R0A   /        ;REGISTERS 0-17 A-PORT
        .WORD   60,2
        .ASCII  /R1A   /
        .WORD   61,2
        .ASCII  /R2A   /
        .WORD   62,2
        .ASCII  /R3A   /
        .WORD   63,2
        .ASCII  /R4A   /
        .WORD   64,2
        .ASCII  /R5A   /
        .WORD   65,2
        .ASCII  /R6A   /
        .WORD   66,2
        .ASCII  /R7A   /
        .WORD   67,2
        .ASCII  /R10A  /
        .WORD   70,2
        .ASCII  /R11A  /
        .WORD   71,2
        .ASCII  /R12A  /
        .WORD   72,2
        .ASCII  /R13A  /
        .WORD   73,2
        .ASCII  /R14A  /
        .WORD   74,2
        .ASCII  /R15A  /
        .WORD   75,2
        .ASCII  /R16A  /
        .WORD   76,2
        .ASCII  /R17A  /
        .WORD   77,2
        .ASCII  /R0B   /        ;REGISTERS 0-17 B-PORT
```

```
        .WORD   6000,3
        .ASCII  /R1B   /
        .WORD   6100,3
        .ASCII  /R2B   /
        .WORD   6200,3
        .ASCII  /R3B   /
        .WORD   6300,3
        .ASCII  /R4B   /
        .WORD   6400,3
        .ASCII  /R5B   /
        .WORD   6500,3
        .ASCII  /R6B   /
        .WORD   6600,3
        .ASCII  /R7B   /
        .WORD   6700,3
        .ASCII  /R10B  /
        .WORD   7000,3
        .ASCII  /R11B  /
        .WORD   7100,3
        .ASCII  /R12B  /
        .WORD   7200,3
        .ASCII  /R13B  /
        .WORD   7300,3
        .ASCII  /R14B  /
        .WORD   7400,3
        .ASCII  /R15B  /
        .WORD   7500,3
        .ASCII  /R16B  /
        .WORD   7600,3
        .ASCII  /R17B  /
        .WORD   7700,3
        .ASCII  /RA    /        ;A-PORT REGISTER
        .WORD   40,10
        .ASCII  /RB    /        ;B-PORT REGISTER
        .WORD   4000,10
        .ASCII  /SRC   /        ;SRC FROM INSTRUCTION
        .WORD   0,11
        .ASCII  /DST   /        ;DST FROM INSTRUCTION
        .WORD   20,11
        .ASCII  /SRC!1 /        ;SRC ORRED WITH 1
        .WORD   0,11
        .ASCII  /DST!1 /        ;DST ORRED WITH 1
        .WORD   20,11
        .ASCII  /Q     /        ;Q REGISTER
        .WORD   0,4
        .ASCII  /OR    /        ;OPERATION CODES
        .WORD   4,6
        .ASCII  /AND   /
        .WORD   5,6
        .ASCII  /MASK  /
        .WORD   6,6
        .ASCII  /XOR   /
        .WORD   7,6
        .ASCII  /XNOR  /
        .WORD   10,6
        .ASCII  /SRU   /
        .WORD   700,1
        .ASCII  /SRD   /
        .WORD   500,1
        .ASCII  /SQ    /
        .WORD   100,1
        .ASCII  /0     /
        .WORD   1,7
        .ASCII  /NOOP  /
        .WORD   6,7
        .WORD   0               ;TABLE TERMINATION
        .LIST   BIN
        ;
        .PAGE
        .SBTTL  ARITHMETIC FUNCTION TABLE
        ;
        .NLIST BIN
ARTABL: .WORD   1,      42      ;0
        .WORD   2,      34      ;A
        .WORD   3,      33      ;B
        .WORD   4,      32      ;Q
```

```
        .WORD   5,      37      ;D
        .WORD   41,     4       ;A+
        .WORD   61,     3       ;B+
        .WORD   101,    2       ;Q+
        .WORD   121,    7       ;D+
        .WORD   42,     14      ;A-
        .WORD   62,     13      ;B-
        .WORD   102,    12      ;Q-
        .WORD   122,    27      ;D-
        .WORD   1042,   24      ;-A-
        .WORD   1062,   23      ;-B-
        .WORD   1102,   22      ;-Q-
        .WORD   1122,   17      ;-D-
        .WORD   20501,  0       ;A+Q+
        .WORD   40441,  0       ;Q+A+
        .WORD   20461,  1       ;A+B+
        .WORD   30441,  1       ;B+A+
        .WORD   50441,  5       ;D+A+
        .WORD   20521,  5       ;A+D+
        .WORD   50501,  6       ;D+Q+
        .WORD   40521,  6       ;Q+D+
        .WORD   41042,  10      ;Q-A-
        .WORD   21102,  20      ;A-Q-
        .WORD   31042,  11      ;B-A-
        .WORD   21062,  21      ;A-B-
        .WORD   21122,  15      ;A-D-
        .WORD   51042,  25      ;D-A-
        .WORD   41122,  16      ;Q-D-
        .WORD   51102,  26      ;D-Q-
        .WORD   1104,   30      ;A OR Q
        .WORD   2102,   30      ;Q OR A
        .WORD   1103,   31      ;A OR B
        .WORD   1502,   31      ;B OR A
        .WORD   2502,   35      ;D OR A
        .WORD   1105,   35      ;A OR D
        .WORD   2504,   36      ;D OR Q
        .WORD   2105,   36      ;Q OR D
        .WORD   1124,   40      ;A AND Q
        .WORD   2122,   40      ;Q AND A
        .WORD   1123,   41      ;A AND B
        .WORD   1522,   41      ;B AND A
        .WORD   2522,   45      ;D AND A
        .WORD   1125,   45      ;A AND D
        .WORD   2524,   46      ;D AND Q
        .WORD   2125,   46      ;Q AND D
        .WORD   1144,   50      ;A MASK Q
        .WORD   1143,   51      ;A MASK B
        .WORD   2542,   55      ;D MASK A
        .WORD   2544,   56      ;D MASK Q
        .WORD   111124, 50      ;/A AND Q
        .WORD   42622,  50      ;Q AND /A
        .WORD   111123, 51      ;/A AND B
        .WORD   32622,  51      ;B  AND /A
        .WORD   112522, 55      ;/D AND A
        .WORD   22625,  55      ;A AND /D
        .WORD   112524, 56      ;/D AND Q
        .WORD   42625,  56      ;Q AND /D
        .WORD   1164,   60      ;A XOR Q
        .WORD   2162,   60      ;Q XOR A
        .WORD   1163,   61      ;A XOR B
        .WORD   1562,   61      ;B XOR A
        .WORD   2562,   65      ;D XOR A
        .WORD   1165,   65      ;A XOR D
        .WORD   2564,   66      ;D XOR Q
        .WORD   2165,   66      ;Q XOR D
        .WORD   1204,   70      ;A XNOR Q
        .WORD   2202,   70      ;Q XNOR A
        .WORD   1203,   71      ;A XNOR B
        .WORD   1602,   71      ;B XNOR A
        .WORD   2602,   75      ;D XNOR A
        .WORD   1205,   75      ;A XNOR D
        .WORD   2604,   76      ;D XNOR Q
        .WORD   2205,   76      ;Q XNOR D
        .WORD   224,    72      ;/Q
        .WORD   223,    73      ;/B
        .WORD   222,    74      ;/A
```

```
        .WORD   225,    77      ;/D
        .WORD   0               ;TABLE TERMINATION
        .LIST BIN
        ;
        .PAGE
        .SBTTL  PRIORITY SEQUENCE DEFINITIONS
        ;
        .NLIST  BIN
PSQTBL: .ASCII  /E0  /  ;LEVELS E0-E7
        .WORD   100000
        .ASCII  /E1  /
        .WORD   40000
        .ASCII  /E2  /
        .WORD   20000
        .ASCII  /E3  /
        .WORD   10000
        .ASCII  /E4  /
        .WORD   4000
        .ASCII  /E5  /
        .WORD   2000
        .ASCII  /E6  /
        .WORD   1000
        .ASCII  /E7  /
        .WORD   400
        .ASCII  /SRC /          ;=E1
        .WORD   40000
        .ASCII  /DST /          ;=E3
        .WORD   10000
        .ASCII  /EX  /          ;=E5
        .WORD   2000
        .ASCII  /RES /          ;=E7
        .WORD   400
        .ASCII  /IOT /          ;IOT TRAP
        .WORD   10
        .ASCII  /BPT /          ;BPT TRAP
        .WORD   4
        .ASCII  /EMT /          ;EMT TRAP
        .WORD   2
        .ASCII  /TRAP/          ;TRAP TRAP
        .WORD   1
        .WORD   0               ;TABLE TERMINATION
        .LIST   BIN
        ;
        .PAGE
        .SBTTL  INSTRUCTION MODE DEFINITIONS
        ;
        .NLIST  BIN
INSTBL: .ASCII  /FRC /          ;FORCE LOOK-AHEAD
        .WORD   100000
        .ASCII  /WT  /          ;WAIT UNTIL FINISHED
        .WORD   -10000
        .ASCII  /C1  /          ;CONDITIONAL LOOK-AHEAD
        .WORD   -40000
        .ASCII  /C2  /          ;CONDITIONAL LOOK-AHEAD
        .WORD   -20000
        .ASCII  /TI  /          ;TRACE TRAP INHIBIT
        .WORD   200
        .ASCII  /BYT /          ;ENABLE BYT OPERATIONS
        .WORD   100
        .ASCII  /SP  /          ;SPECIAL MODE
        .WORD   40
        .ASCII  /NDA /          ;NO DATA ACCESS MODE
        .WORD   20
        .WORD   0               ;TABLE TERMINATION
        .LIST   BIN
        ;
        .PAGE
        .SBTTL  CONDITION CODE DEFINITIONS
        ;
        .NLIST  BIN
PSWCCS: .ASCII  /N /    ;N BIT
        .WORD   NBCODE
        .ASCII  /V /    ;V BIT
        .WORD   VBCODE
        .ASCII  /Z /    ;Z BIT
        .WORD   ZBCODE
```

```
        .ASCII  /C /    ;C BIT
        .WORD   CBCODE
        .WORD   0       ;TABLE TERMINATION
        .LIST   BIN
        ;
        .PAGE
        .SBTTL  PROCESSOR STATUS BIT N CODE DEFINITIONS
        ;
        .NLIST  BIN
NBCODE: .ASCII  /N     /        ;CURRENT PSR BIT N
        .WORD   0
        .ASCII  /Z     /        ;CURRENT PSR BIT Z
        .WORD   1
        .ASCII  /V     /        ;CURRENT PSR BIT V
        .WORD   2
        .ASCII  /C     /        ;CURRENT PSR BIT C
        .WORD   3
        .ASCII  /SET   /        ;INSTRUCTION DECODED SET BIT
        .WORD   4
        .ASCII  /CLR   /        ;INSTRUCTION DECODED CLEAR BIT
        .WORD   5
        .ASCII  /CPO   /        ;BIT 3 OUTPUT OF ALU
        .WORD   6
        .ASCII  /PNXN  /        ;(PSR BIT N) XOR (ALU N BIT)
        .WORD   7
        .ASCII  /PN    /        ;ALU N BIT
        .WORD   10
        .ASCII  /PZ    /        ;ALU Z BIT
        .WORD   11
        .ASCII  /PV    /        ;ALU V BIT
        .WORD   12
        .ASCII  /PC    /        ;ALU C BIT
        .WORD   13
        .ASCII  /PNXC  /        ;(ALU N BIT) XOR (PSR C BIT)
        .WORD   14
        .ASCII  /PNXPV /        ;(ALU N BIT) XOR (ALU V BIT)
        .WORD   15
        .ASCII  /0     /        ;CLEAR BIT
        .WORD   16
        .ASCII  /1     /        ;SET BIT
        .WORD   17
        .ASCII  /MSBR  /        ;HIGH ORDER BIT OF ALU
        .WORD   10
        .WORD   0               ;TABLE TERMINATION
        .LIST BIN
        ;
        .PAGE
        .SBTTL  PROCESSOR STATUS BIT V CODE DEFINITIONS
        ;
        .NLIST  BIN
VBCODE: .ASCII  /LBPAR /        ;LOW BYTE PARITY
        .WORD   0
        .ASCII  /Z     /        ;CURRENT PSR BIT Z
        .WORD   20
        .ASCII  /V     /        ;CURRENT PSR BIT V
        .WORD   40
        .ASCII  /C     /        ;CURRENT PSR BIT C
        .WORD   60
        .ASCII  /SET   /        ;INSTRUCTION DECODED SET BIT
        .WORD   100
        .ASCII  /CLR   /        ;INSTRUCTION DECODED CLEAR BIT
        .WORD   120
        .ASCII  /CPO   /        ;BIT 1 OUTPUT OF ALU
        .WORD   140
        .ASCII  /PNXN  /        ;(PSR BIT N) XOR (ALU N BIT)
        .WORD   160
        .ASCII  /PN    /        ;ALU N BIT
        .WORD   200
        .ASCII  /PZ    /        ;ALU Z BIT
        .WORD   220
        .ASCII  /PV    /        ;ALU V BIT
        .WORD   240
        .ASCII  /PC    /        ;ALU C BIT
        .WORD   260
        .ASCII  /PNXC  /        ;(ALU N BIT) XOR (PSR C BIT)
        .WORD   300
```

```
        .ASCII  /PNXPV /        ;(ALU N BIT) XOR (ALU V BIT)
        .WORD   320
        .ASCII  /0     /        ;CLEAR BIT
        .WORD   340
        .ASCII  /1     /        ;SET BIT
        .WORD   360
        .ASCII  /MSBR  /        ;HIGH ORDER BIT OF ALU
        .WORD   200
        .WORD   0               ;TABLE TERMINATION
        .LIST BIN
        ;
        .PAGE
        .SBTTL  PROCESSOR STATUS BIT Z CODE DEFINITIONS
        ;
        .NLIST  BIN
ZBCODE: .ASCII  /N     /        ;CURRENT PSR BIT N
        .WORD   0
        .ASCII  /Z     /        ;CURRENT PSR BIT Z
        .WORD   400
        .ASCII  /V     /        ;CURRENT PSR BIT V
        .WORD   1000
        .ASCII  /C     /        ;CURRENT PSR BIT C
        .WORD   1400
        .ASCII  /SET   /        ;INSTRUCTION DECODED SET BIT
        .WORD   2000
        .ASCII  /CLR   /        ;INSTRUCTION DECODED CLEAR BIT
        .WORD   2400
        .ASCII  /CPO   /        ;BIT 2 OUTPUT OF ALU
        .WORD   3000
        .ASCII  /PNXN  /        ;(PSR BIT N) XOR (ALU N BIT)
        .WORD   3400
        .ASCII  /PN    /        ;ALU N BIT
        .WORD   4000
        .ASCII  /PZ    /        ;ALU Z BIT
        .WORD   4400
        .ASCII  /PV    /        ;ALU V BIT
        .WORD   5000
        .ASCII  /PC    /        ;ALU C BIT
        .WORD   5400
        .ASCII  /PZAZ  /        ;(ALU Z BIT) AND (PSR Z BIT)
        .WORD   6000
        .ASCII  /PNXPV /        ;(ALU N BIT) XOR (ALU V BIT)
        .WORD   6400
        .ASCII  /0     /        ;CLEAR BIT
        .WORD   7000
        .ASCII  /1     /        ;SET BIT
        .WORD   7400
        .ASCII  /MSBR  /        ;HIGH ORDER BIT OF ALU
        .WORD   4000
        .WORD   0               ;TABLE TERMINATION
        .LIST BIN
        ;
        .PAGE
        .SBTTL  PROCESSOR STATUS BIT C CODE DEFINITIONS
        ;
        .NLIST  BIN
CBCODE: .ASCII  /LSBR  /        ;LEAST SIGNIFICANT BIT OF ALU
        .WORD   0
        .ASCII  /MSBR  /        ;HIGH ORDER BIT OF ALU
        .WORD   100000
        .ASCII  /PCB   /        ;COMPLEMENT OF ALU Z BIT
        .WORD   10000
        .ASCII  /V     /        ;CURRENT PSR BIT V
        .WORD   20000
        .ASCII  /C     /        ;CURRENT PSR BIT C
        .WORD   30000
        .ASCII  /SET   /        ;INSTRUCTION DECODED SET BIT
        .WORD   40000
        .ASCII  /CLR   /        ;INSTRUCTION DECODED CLEAR BIT
        .WORD   50000
        .ASCII  /CPO   /        ;BIT 0 OUTPUT OF ALU
        .WORD   60000
        .ASCII  /PZB   /        ;COMPLEMENT OF ALU Z BIT
        .WORD   70000
        .ASCII  /PN    /        ;ALU N BIT
        .WORD   100000
```

```
        .ASCII  /PZ    /        ;ALU Z BIT
        .WORD   110000
        .ASCII  /PV    /        ;ALU V BIT
        .WORD   120000
        .ASCII  /PC    /        ;ALU C BIT
        .WORD   130000
        .ASCII  /LSBQ  /        ;LEAST SIGNIFICANT BIT OF Q
        .WORD   140000
        .ASCII  /MSBQ  /        ;MOST SIGNIFICANT BIT OF Q
        .WORD   150000
        .ASCII  /0     /        ;CLEAR BIT
        .WORD   160000
        .ASCII  /1     /        ;SET BIT
        .WORD   170000
        .WORD   0               ;TABLE TERMINATION
        .LIST BIN
        ;
        .IIF    DF,NOPRNT       .LIST           ;RESTORE PRINTING

        END=.
        .END    ENTER
```

```
SET ERROR NONE
DELETE/NOQ BLUXHI.WCS,BLUXHI.LST
SET ERROR ERROR
!
R MICRO
BLUXHI[40],BLUXHI[600]=BLUXHI,EISX,FISX,FPPX,SPCLX,ENDX
^C
```

```
        TT:     BLUCPU MICROCODE

        SB:     TITLE PAGE

        CM:     EMULATION OF THE PDP 11/34 COMPUTER
                CODE DATED MARCH 1980
                ALAN R. BALDWIN

        CM:     CORRECTION TO MTPS (USER MODE) - MAY 1980
        CM:     CORRECTION TO MTOUT (MAINTENANCE MODE) - MAY 1980
        CM:     CORRECTION OF RTI/RTT INSTRUCTIONS - JULY 1980
        CM:     REWRITE OF TRAP HANDLERS - JULY 1980
        CM:     ADDITION OF RESERVED INSTRUCTION HANDLER - JULY 1980
        CM:     ADDITION OF SPL INSTRUCTION - JULY 1980
        CM:     CORRECTION OF RSRVINST/RSRVSTAT - MAY 1986

        CM:     PROGRAM MICROCODE ORIGIN
        NA:             MORIGIN = +6000

        CM:     INSTRUCTION CODE ORIGIN
        NA:             IORIGIN = +6000


        PG:     TRAP PROCESSING

        .=: MORIGIN+0

        CM: TRAP SEQUENCE ENTRY POINT
            RESERVED INSTRUCTIONS
            TRAP, EMT, IOT, BPT INSTRUCTIONS

**: RES
        AR: AD;R10B=PV  PC: LW, WIOL
        IO: DATI(KI)

**:     AR: R10B=R10A-[+2]-<1>  BR: CJN(1) [PULL+2]

**: PUSH
        IO: IDATO(CMI)          BR: CJN(1) [PUSHA]

**: RESEND
        AR: AD=R7A      PC: LW, WIOL
        IO: INTCK       BR:  CBIC(0)

**:     AR: SCR0=0      PC: LW, CFP
        IO: IDATIR(CMI)

**:     AR: AD;R7B=R7A+[+2]+<0> PC: LW, WIOL
        IO: INTCK       BR: CBR(1) [BGN+2]

**: PUSHA
        AR: AD=R6A+[+2]+<0>     PC: LW,WIOL
        SP: CLR

**:     AR: D=R11A        IO: DATO(CMI)
        BR: CRN(MMGT)

**:     AR: SOR=R6A-SLR-<1>
        BR: CRR(1)

**: YSTB
        CM: ENTRY POINT FOR YELLOW STACK AND T BIT TRAPS
        AR: R7B=R7A-[+2]-<1>
        BR: CBN(1) [RES]

**: PF
        CM: POWER FAIL ENTRY POINT
        AR: AD;R10B=PV  PC: LW, WIOL
        IO: DATI(KI)    BR: CJN(1) [PULL]

**:     BR: BROC(/PFD) [PUSH]

        PG:     INTERRUPT HANDLERS

**: INT17
        CM: ENTRY POINT FOR INTERNAL
```

```
        CM: INTERRUPT LEVELS 1 THROUGH 7
        AR: AD;R10B=PV  PC: LW, WIOL
        IO: DATI(KI)    SP: CI

**:     AR: R7B=R7A-[+2]-<1>    BR: CBN(1) [RES+1]

**: BINT
        AR: R10B=[+2]   PC: LW, WIOL
        IO: IINT

**:     AR: AD;R10B=R10A+D+<0>  PC: LW,WIOH
        IO: DATI(KI)    BR: CBN(1) [RES+1]

        PG:     SOURCE MODE SEQUENCE

        .=: MORIGIN+20
        CM: SOURCE MODE CODE
            FOR ALL MODES RESULTS ARE
            (1)  D & R10 CONTAIN DATA FROM ADDRESS
            (2)  STACK OVERFLOW CHECKED IN MODES  4 AND 5
            UPON ENTRY AND EXIT  AD = R7

**: SRC
        CM: (R) IS OPERAND
        AR: D=RA(SRC)   BR:  CBR(1) [SEND]

**:     CM: (R) IS ADDRESS
        AR: AD=RA(SRC)  PC: LW, WIOL
        IO: DATI(CMI,BYT)       BR: CBR(1) [SEND]

**:      CM: (R) IS ADDRESS; (R) + (1 OR 2)
        AR: RB(SRC)=RA(SRC)+[+1]+</SR67W>, AD=RA(SRC)   PC: LW, WIOL
        IO: DATI(CMI,BYT)       BR: CBR(1) [SEND]

**:     CM: (R) IS ADDRESS OF ADDRESS; (R) + 2
        AR: RB(SRC)=RA(SRC)+[+2]+<0>, AD=RA(SRC)        PC: LW, WIOL
        IO: DATI(CMI)   BR: CBR(1) [OS3]

**:     CM: (R) - (1 OR 2); (R) IS ADDRESS
        AR: AD;RB(SRC)=RA(SRC)-[+1]-<SR67W>     PC: LW, WIOL
        IO: DATI(CMI,BYT)       BR: BROC(MMGT) [OS4]

**:     CM: (R) - 2; (R) IS ADDRESS OF ADDRESS
        AR: AD;RB(SRC)=RA(SRC)-[+2]-<1> PC: LW, WIOL
        IO: DATI(CMI)   BR: CBR(1) [OS5]

**:     CM: (R) + X  IS ADDRESS
        AR: R7B=R7A+[+2]+<0>    PC: LW, WIOL
        IO: IDATI(CMI)  BR: CBR(1) [OS6]

**:     CM: (R) + X  IS ADDRESS OF ADDRESS
        AR: R7B=R7A+[+2]+<0>    PC: LW, WIOL
        IO: IDATI(CMI)  BR: CBR(1) [OS7]

        PG:     DESTINATION MODE SEQUENCE

        .=: MORIGIN+30
        CM: DESTINATION MODE CODE
            FOR ALL MODES
            UPON ENTRY
            (1) AD = R7
            (2) R10 CONTAINS DATA OF PRECEEDING SOURCE MODE
            UPON EXIT
            (1) AD & R11 CONTAIN ADDRESS OF DESTINATION DATA
            (2) D CONTAINS DATA AT DESTINATION ADDRESS
            (3) STACK OVERFLOW CHECKED IN MODES 4 & 5

**: DST
        CM: (R) IS OPERAND
        AR: R10B=D, D=RA(DST)   BR: CBIN(1) <INST>

**:     CM: (R) IS ADDRESS
        AR: AD;R11B=RA(DST)     PC: LW, WIOL
        IO: DATIP(CM,BYT,NDSP)  BR: CBIN(1) <INST>

**:     CM: (R) IS ADDRESS; (R) + (1 OR 2)
```

```
        AR: AD;R11B=RA(DST)     PC: LW, WIOL
        IO: DATIP(CM,BYT,NDSP)  BR: CBR(1) [OD2]

**:     CM: (R) IS ADDRESS OF ADDRESS; (R) + 2
        AR: RB(DST)=RA(DST)+[+2]+<0>, AD=RA(DST)        PC: LW, WIOL
        IO: DATI(CMI)   BR: CBR(1) [OD3]

**:     CM: (R) - (1 OR 2); (R) IS ADDRESS
        AR: AD;RB(DST)=RA(DST)-[+1]-<DR67W>     PC: LW, WIOL
        IO: DATIP(CM,BYT,NDSP)  BR: CBR(1) [OD4]

**:     CM: (R) - 2; (R) IS ADDRESS OF ADDRESS
        AR: AD;RB(DST)=RA(DST)-[+2]-<1> PC: LW, WIOL
        IO: DATI(CMI)   BR: CBR(1) [OD5]

**:     CM: (R) + X  IS ADDRESS
        AR: R7B=R7A+[+2]+<0>    PC: LW, WIOL
        IO: IDATI(CMI)  BR: CBR(1) [OD6]

**:     CM: (R) + X  IS ADDRESS OF ADDRESS
        AR: R7B=R7A+[+2]+<0>    PC: LW, WIOL
        IO: IDATI(CMI)  BR: CBR(1) [OD7]

        PG:     SOURCE & DESTINATION COMPLETION

**: OS7
        CM: COMPLETION OF SOURCE MODES
        AR: AD=RA(SRC)+D+<0>  PC: LW, WIOH      IO: DATI(CMI)

**: OS3
        AR: AD=D        PC: LW, WIOH
        IO: DATI(CMI,BYT)       BR: CBR(1) [SEND]

**: OS6
        AR: AD=RA(SRC)+D+<0>
        PC: LW, WIOH    IO: DATI(CMI,BYT)       BR: CBR(1) [SEND]

**: OS5
        AR: AD=D        PC: LW, WIOH
        IO: DATI(CMI,BYT)       BR: CBN(MMGT) [.+2]

**: OS4
        AR: SOR=R6A-SLR-<1>

**: SEND
        AR: R10B=D, AD=R7A      PC: LW, WIOH
        IO: INTCK       BR: CBIN(/LKDC) <INST>

**:     IO: ILDATIR(CMI)        BR: CBIC(1) <INST>

**: OD2
        AR: RB(DST)=RA(DST)+[+1]+</DR67W>
        BR: CBIN(1) <INST>

**: OD7
        AR: AD=RA(DST)+D+<0>    PC: LW, WIOH
        IO: DATI(CMI)

**: OD3
        AR: AD;R11B=D   PC: LW, WIOH
        IO: DATIP(CM,BYT,NDSP)  BR: CBIN(1) <INST>

**: OD4
        AR: R11B=RA(DST)        BR: CBIN(MMGT) <INST>

**:     AR: SOR=R6A-SLR-<1>     BR: CBIC(1) <INST>

**: OD5
        AR: AD;R11B=D   PC: LW, WIOH
        IO: DATIP(CM,BYT,NDSP)  BR: CBIN(MMGT) <INST>

**:     AR: SOR=R6A-SLR-<1>     BR: CBIC(1) <INST>

**: OD6
        AR: AD;R11B=RA(DST)+D+<0>       PC: LW, WIOH
        IO: DATIP(CM,BYT,NDSP)  BR: CBIN(1) <INST>
```

```
        PG:     PULL VECTOR HANDLER

**: PULL
        CM: TRAP HANDLER PULL ROUTINE
        AR: R7B=R7A-[+2]-<1>

**:     AR: R10B=R10A-[+2]-<1>

**:     AR: R10B=D,AD=R10A      PC: LW,WIOH
        IO: DATI(KI)

**:     AR: R11B=PSR,PSR=R10A   PC: LW, WIOL
        CC: C=CPO, V=CPO, Z=CPO, N=CPO

**:     AR: R7B=D, D=R7A        BR: CBN(CUM) [.+2]

**:     AR: AD;R6B=R6A-[+4]-<1> BR: CRR(1)

**:     AR: AD;R16B=R6A-[+4]-<1>        BR: CRR(1)

**: NOBR
        CM: NO BRANCH
        BR: BROC(LKD) [BGN]

**: BR
        CM: BRANCH INSTRUCTION
        AR: AD;R7B=R7B+R10A+<0>
        IO: INTCK       BR: CBR(1) [BGN+3]

        PG:     INSTRUCTION LOAD & DECODE

        .=: MORIGIN+70

**: BGN
        AR: AD=R7A+[+0]+<0>     PC: LW, WIOL
        IO: INTCK       BR: CBR(1) [.+3]

**:     AR: AD;R7B=R7A+[+2]+<0> PC: LW, WIOL
        IO: INTCK       SP: ILE

**:     AR: R10B=ISX7, SRU(0)
        IO: ILDATIR(CMI)        BR: CBIN(1) <INST>

**:     IO: IDATIR(CMI) PC: LW, WIOL
        BR: CBR(1) [BGN+1]

**: INT0
        CM: ENTRY POINT FOR BUS INTERRUPTS
        CM: AND LEVEL 0 INTERNAL INTERRUPTS
        AR: R7B=R7A-[+2]-<1>    BR: CBN(BINT) [BINT]

**:     AR: R7B=R7A+[+2]+<0>    BR: CBN(EIL) [INT17]

**:     AR: AD;R7B=R7A-[+2]-<1>
        IO: INTCK       BR: CBR(1) [BGN+3]

**: FBHINT
        CM: ENTRY POINT FOR 'FBH' INTERRUPTS
            TIME OUT, MEMORY MANAGEMENT, AND ODD ADDRESS TRAPS
            REQUIRE ONE CYCLE DELAY FOR HARDWARE TIMING AFTER 'FBH' CYCLE
        BR: CBR(1) [RES]

        PG:     CONSOLE SWITCH HANDLER

        .=: MORIGIN+100

**: EX1
        CM: RESTORE CC'S, BRANCH TO DESIGNATED EXAM
        AR: R12B=PSR, OUT=R12A  CC: C=CPO, Z=CPO, V=CPO, N=CPO
        BR: CBZV(Z) [.]

**:     CM: MOVE ADDRESS TO ADX, START DATI, BRANCH
        AR: ADX=R17A    PC: UW, CB
        IO: DATI(OFF)   BR: CBR(1) [END+1]
```

```
**:     CM: MOVE REGISTER ADDRESSED BY D INTO D (R20-R37)
        AR: D=RA(D)  PC: UW     BR: CBR(1) [END+1]

**:     CM: MOVE REGISTER ADDRESSD BY D INTO D (R0-R17)
        AR: D=RA(D)     PC: LW  BR: CBR(1) [END+1]

**: DEP1
        CM: RESTORE CC'S, BRANCH TO DEP MODE
        AR: R12B=PSR, OUT=R12A  CC: C=CPO, Z=CPO, V=CPO, N=CPO
        BR: CBZV(Z) [.]

**:     CM: PUT DATA IN D, DO A DATO, BRANCH
        AR: D=SW1       PC: LW, CB
        IO: DATO(OFF)   BR: CBR(1) [END+1]

**:     CM: STORE DATA (R15) IN REG POINTED AT BY D
        AR: D;RB(D)=R15A        PC: UW  BR: CBR(1) [END+1]

**:     CM: STORE DATA (R15) IN REG POINTED AT BY D
        AR: D;RB(D)=R15A        PC: LW  BR: CBR(1) [END+1]

**: CHLT
        CM: CONSOLE INTERRUPT ENTRY POINT
        AR: AD;R7B=R7A-[+2]-<1>
        BR: CBR(1) [CONS]

**: HALT
        CM: HALT INSTRUCTION ENTRY POINT
        BR: CBN(CUM) [RES]

**: CONS
        CM: PUT IR IN D
        AR: D=IR        PC: LW

**:     CM:  MASK WORD FOR CONTROL SWITCHES
        AR: R14B=[+174000]

**:     CM:  INITIALIZE CONSOLE CONDITION CODES
        AR: SCR0;R12B=0

**: GO
        CM:  MASK SWITCH DATA & TEST FOR ACTION
        AR: R15B=SW2 AND R14A   BR: CNR(/Z')    PC: LW, CB

**:     CM:  CHECK SWITCH FOR RELEASE
        AR: SW2 AND R14A        BR: CNR(Z')     PC: LW, CB

**:     CM: EXCHANGE PROGRAM & ROUTINE CC'S, TEST FOR REG
        AR: R12B=PSR, OUT=R12A  CC: C=CPO, Z=CPO, V=CPO, N=CPO
        BR: CJR(1) [TSTR]

**:     CM: TEST SIGN AND SHIFT DATA
        AR: R15B=R15B, SRU(0)

**: LAD
        CM: TEST & SHIFT, BRANCH IF NOT LAD
        AR: R15B=R15B, SRU(0)   BR: CBN(/N') [EX]

**:     CM: SET CODES FOR NEW ADDRESS
        AR: AD;R17B=SW1 CC: C=0, N=0, Z=Z, V=V

**:     CM: MOVE UPPER ADDRESS TO R17 & AD
        AR: ADX;R17B=SW2        PC: UW  BR: CBR(1) [END]

**: EX
        CM: TEST SIGN & SHIFT, IF NOT EX BRANCH
        AR: R15B=R15B, SRU(0)   BR: CBN(/N') [CONT]

**:     CM: SET CODES, JSR TO INCR IF SECOND TIME
        CC: C=1, N=0, V=V, Z=Z  BR: CJN(C) [INCR]

**:     CM: MOVE ADDRESS INTO AD
        AR: AD=R17A

**:     CM: MOVE ADDRESS INTO D
        AR: D=R17A      BR: CBR(1) [EX1]
```

```
**: CONT
        CM: SHIFT & TEST SIGN, IF NOT CONT BRANCH
        AR: R15B=R15B, SRU(0)   BR: CBN(/N') [STRT]

**:     CM: MOVE PC INTO AD, ENABLE INTERRUPT LOGIC
        AR: AD=R7A      SP: ILE PC: LW, CFP

**:     CM: MOVE PCX INTO ADX
        AR: ADX=R7A     PC: UW

**:     CM: RESTORE CC'S, LOAD NEXT INSTRUCTION
        AR: R12B=PSR, OUT=R12A  CC: C=CPO, Z=CPO, V=CPO, N=CPO
        PC: LW, CB      IO: DATIR(CMI)

**:     CM: WAIT FOR IO COMPLETE
        PC: LW, WIOL, CB        BR: CBR(1) [BGN+1]

**: STRT
        CM: SHIFT & TEST SIGN, BRANCH IF NOT STRT
        AR: R15B=R15B, SRU(0)   BR: CBN(/N') [DEP]

**:     CM: LOAD CNTR, SEQUENCE TO CLEAR PRIORITIES
        AR: CNTR=[+177767]
        SP: ILD, CLR    PC: LW, CFP

**:     CM: SEQUENCE PRIORITIES
        AR: PSR=0       CC: C=0, Z=0, V=0, N=0
        PC: LW, CCL     BR: CBIN(0)

**:     CM: REPEAT UNTIL SEQUENCE COMPLETE
        BR: CBN(CNTR) [.-1]

**:     CM: LOAD NEW PC, DO A BUS INIT, ENABLE INTERRUPT LOGIC
        AR: AD;R7B=R17A IO: INIT        SP: ILE PC: LW, CFP

**:     CM: LOAD NEW PCX, LOAD NEXT INSTRUCTION
        AR: ADX;R7B=R17A        PC: UW, WIOL
        IO: DATIR(CMI)  BR: CBR(1) [BGN+1]

**: DEP
        CM: BRANCH IF NOT DEP (ERROR)
        BR: CBN(/N') [END]

**:     CM: SET CODES, INCR IF SECOND TIME
        CC: C=0, N=1, Z=Z, V=V  BR: CJN(N) [INCR]

**:     CM: PUT ADDRESS IN AD, PUT DATA IN R15
        AR: R15B=SW1, AD=R17A   PC: LW

**:     CM: PUT ADDRESS IN ADX, PUT DATA IN R15
        AR: R15B=SW1, ADX=R17A  PC: UW

**:     CM: MOVE ADDRESS INTO D
        AR: D=R17A      BR: CBR(1) [DEP1]

**: END
        CM: RESTORE CC'S
        AR: R12B=PSR, OUT=R12A  CC: C=CPO, Z=CPO, V=CPO, N=CPO

**:     CM: WAIT FOR IO COMPLETE, LOOK AT SWITCHES, BRANCH
        AR: R15B=SW2 AND R14A   PC: LW, WIOH, CB        BR: CBR(1) [GO]

**: INCR
        CM: INCREMENT ADDRESS BY 1, SKIP NEXT IF REG
        AR: R17B=R17B+<1>       PC: 3B  BR: CBN(Z) [.+2]

**:     CM: INCREMENT ADDRESS BY 1
        AR: R17B=R17B+<1>       PC: 3B

**: TSTR
        CM: MOVE TESTF BIT INTO R13, GO TEST FOR REG
        AR: R13B=[+20]  BR: CJR(1) [REG]

**:     CM: TEST FOR UPPER/LOWER REGISTER
        AR: R13B AND R17A       BR: CRR(1)
```

```
        CC: V=PZ, C=C, Z=Z, N=N

**: REG
        CM: TEST ADX FOR REGISTER ADDRESS
        AR: R17A-[+77]-<1>      PC: UB

**:     CM: COMPARE ADDRESS TO LOW BOUND OF REG ADDRESS
        AR: R17A-[+177700]-<1>  PC: LW  BR: CBN(/Z') [NO]

**:     CM: COMPARE ADDRESS TO UPPER BOUNDS OF REG ADDRESS
        AR: [+177737]-R17A-<1>  PC: LW  BR: CBN(N') [NO]

**:     CM: REGISTER ADDRESS
        CC: Z=1, C=C, V=V, N=N  BR: CRN(/N')

**: NO
        CM: NOT REGISTER ADDRESS
        CC: Z=0, C=C, V=V, N=N  BR: CRR(1)

        PG:     DOUBLE OPERAND INSTRUCTIONS

        CM: DOUBLE OPERAND INSTRUCTIONS
            D,S,MODE IS LABLED AS INSTRUCTION
            THE THREE REMAINING MODES D,S0 D0,S & D0,S0
            ARE ACCESSED BY OFFSETS

**: MOV
        CM: MOVE INSTRUCTION
        AR: D=R10A      PC: LW, WIOL    IO: DATO(CM)
        CC: N=PN, Z=PZ, V=0, C=C        BR: CBR(1) [BGN]

**:     AR: D=RA(SRC)   PC: LW, WIOL    IO: DATO(CM)
        CC: N=PN, Z=PZ, V=0, C=C        BR: CBR(1) [BGN]

**:     AR: RB(DST)=D
        CC: Z=PZ, N=PN, V=0, C=C        BR: BROC(LKD) [BGN]

**:     AR: RB(DST)=RA(SRC)
        CC: N=PN, Z=PZ, V=0, C=C        BR: BROC(LKD) [BGN]

**: MOVB
        CM: MOVE BYTE INSTRUCTION
        AR: D=R10A      PC: LB, WIOL    IO: DATOB(CM)
        CC: N=PN, Z=PZ, V=0, C=C        BR: CBR(1) [BGN]

**:     AR: D=RA(SRC)   PC: LB, WIOL    IO: DATOB(CM)
        CC: N=PN, Z=PZ, V=0, C=C        BR: CBR(1) [BGN]

**:     AR: RB(DST)=DSX7        PC: LW
        CC: N=PN, Z=PZ, V=0, C=C        BR: BROC(LKD) [BGN]

**:     AR: D=RA(SRC)   PC: LB  BR: CBR(1) [.-1]

**: CMP
        CM: COMPARE INSTRUCTION
        AR: R10A-D-<1>  PC: LWB, WIOH
        CC: N=PN, Z=PZ, V=PV, C=PCB     BR: CBR(1) [BGN]

**:     AR: RA(SRC)-D-<1>       PC: LWB, WIOH
        CC: N=PN, Z=PZ, V=PV, C=PCB     BR: CBR(1) [BGN]

**:     AR: D-RA(DST)-<1>       PC: LWB
        CC: N=PN, Z=PZ, V=PV, C=PCB     BR: BROC(LKD) [BGN]

**:     AR: RB(SRC)-RA(DST)-<1> PC: LWB
        CC: N=PN, Z=PZ, V=PV, C=PCB     BR: BROC(LKD) [BGN]

**: BIT
        CM: BIT TEST INSTRUCTION
        AR: D AND R10A  PC: LWB, WIOH
        CC: N=PN, Z=PZ, V=0, C=C        BR: CBR(1) [BGN]

**:     AR: D AND RA(SRC)       PC: LWB, WIOH
        CC: N=PN, Z=PZ, V=0, C=C        BR: CBR(1) [BGN]

**:     AR: D AND RA(DST)       PC: LWB
```

```
        CC: N=PN, Z=PZ, V=0, C=C        BR: BROC(LKD) [BGN]

**:     AR: RB(SRC) AND RA(DST) PC: LWB
        CC: N=PN, Z=PZ, V=0, C=C        BR: BROC(LKD) [BGN]

**: BIC
        CM: BIT CLEAR INSTRUCTION
        AR: R10B=D, D=R10A      PC: LWB, WIOH
        BR: CBR(1) [.+4]

**:     AR: R10B=D, D=RA(SRC)   PC: LWB, WIOH
        BR: CBR(1) [.+3]

**:     AR: RB(DST)=/D AND RA(DST)      PC: LWB
        CC: N=PN, Z=PZ, V=0, C=C        BR: BROC(LKD) [BGN]

**:     AR: RB(DST)=/RA(SRC) AND RB(DST)        PC: LWB
        CC: N=PN, Z=PZ, V=0, C=C        BR: BROC(LKD) [BGN]

**:     AR: D=/D AND R10A       PC: LWB, WIOH   IO: DATO(CM,BYT)
        CC: N=PN, Z=PZ, V=0, C=C        BR: CBR(1) [BGN]

**: BIS
        CM: BIT SET INSTRUCTION
        AR: D=R10A OR D PC: LWB, WIOH   IO: DATO(CM,BYT)
        CC: N=PN, Z=PZ, V=0, C=C        BR: CBR(1) [BGN]

**:     AR: D=D OR RA(SRC)      PC: LWB, WIOH IO: DATO(CM,BYT)
        CC: N=PN, Z=PZ, V=0, C=C        BR: CBR(1) [BGN]

**:     AR: RB(DST)=D OR RA(DST)        PC: LWB
        CC: N=PN, Z=PZ, V=0, C=C        BR: BROC(LKD) [BGN]

**:     AR: RB(DST)=RB(DST) OR RA(SRC)  PC: LWB
        CC: N=PN, Z=PZ, V=0, C=C        BR: BROC(LKD) [BGN]

**: ADD
        CM: ADD INSTRUCTION
        AR: D=D+R10A+<0>        PC: LW, WIOH    IO: DATO(CM)
        CC: N=PN, Z=PZ, V=PV, C=PC      BR: CBR(1) [BGN]

**:     AR: D=D+RA(SRC)+<0>     PC: LW, WIOH    IO: DATO(CM)
        CC: N=PN, Z=PZ, V=PV, C=PC      BR: CBR(1) [BGN]

**:     AR: RB(DST)=D+RA(DST)+<0>
        CC: N=PN, Z=PZ, V=PV, C=PC      BR: BROC(LKD) [BGN]

**:     AR: RB(DST)=RB(DST)+RA(SRC)+<0>
        CC: N=PN, Z=PZ, V=PV, C=PC      BR: BROC(LKD) [BGN]

**: SUB
        CM: SUBTRACT INSTRUCTION
        AR: D=D-R10A-<1>        PC: LW, WIOH    IO: DATO(CM)
        CC: N=PN, Z=PZ, V=PV, C=PCB     BR: CBR(1) [BGN]

**:     AR: D=D-RA(SRC)-<1>     PC: LW, WIOH    IO: DATO(CM)
        CC: N=PN, Z=PZ, V=PV, C=PCB     BR: CBR(1) [BGN]

**:     AR: RB(DST)=RA(DST)-D-<1>
        CC: N=PN, Z=PZ, V=PV, C=PCB     BR: BROC(LKD) [BGN]

**:     AR: RB(DST)=RB(DST)-RA(SRC)-<1>
        CC: N=PN, Z=PZ, V=PV, C=PCB     BR: BROC(LKD) [BGN]

        PG:     SINGLE OPERAND INSTRUCTIONS

        CM: SINGLE OPERAND INSTRUCTIONS
            D  MODE IS LABLED AS INSTRUCTION
            D0  MODE IS ACCESSED BY AN OFFSET

**: CLR
        CM: CLEAR INSTRUCTION
        AR: D=0 PC: LWB, WIOL   IO: DATO(CM,BYT)
        CC: Z=1, N=0, V=0, C=0  BR: CBR(1) [BGN]

**:     AR: RB(DST)=0   PC: LWB
```

```
        CC: Z=1, N=0, V=0, C=0  BR: BROC(LKD) [BGN]

**: COM
        CM: COMPLEMENT INSTRUCTION
        AR: D=/D        PC: LWB, WIOH   IO: DATO(CM,BYT)
        CC: N=PN, Z=PZ, V=0, C=1        BR: CBR(1) [BGN]

**:     AR: RB(DST)=/RB(DST)    PC: LWB
        CC: N=PN, Z=PZ, V=0, C=1        BR: BROC(LKD) [BGN]

**: INC
        CM: INCREMENT INSTRUCTION
        AR: D=D+<1>     PC: LWB, WIOH   IO: DATO(CM,BYT)
        CC: N=PN, Z=PZ, V=PV, C=C       BR: CBR(1) [BGN]

**:     AR: RB(DST)=RB(DST)+<1> PC: LWB
        CC: N=PN, Z=PZ, V=PV, C=C       BR: BROC(LKD) [BGN]

**: DEC
        CM: DECREMENT INSTRUCTION
        AR: D=D-<0>     PC: LWB, WIOH   IO: DATO(CM,BYT)
        CC: N=PN, Z=PZ, V=PV, C=C       BR: CBR(1) [BGN]

**:     AR: RB(DST)=RB(DST)-<0> PC: LWB
        CC: N=PN, Z=PZ, V=PV, C=C       BR: BROC(LKD) [BGN]

**: NEG
        CM: NEGATE INSTRUCTION
        AR: D=-D-<1>    PC: LWB, WIOH   IO: DATO(CM,BYT)
        CC: N=PN, Z=PZ, V=PV, C=PZB     BR: CBR(1) [BGN]

**:     AR: RB(DST)=-RB(DST)-<1>        PC: LWB
        CC: N=PN, Z=PZ, V=PV, C=PZB     BR: BROC(LKD) [BGN]

**: ADC
        CM: ADD CARRY INSTRUCTION
        AR: D=D+<C>     PC: LWB, WIOH   IO: DATO(CM,BYT)
        CC: N=PN, Z=PZ, V=PV, C=PC      BR: CBR(1) [BGN]

**:     AR: RB(DST)=RB(DST)+<C> PC: LWB
        CC: N=PN, Z=PZ, V=PV, C=PC      BR: BROC(LKD) [BGN]

**: SBC
        CM: SUBTRACT CARRY INSTRUCTION
        AR: D=D-</C>    PC: LWB, WIOH   IO: DATO(CM,BYT)
        CC: N=PN, Z=PZ, V=PV, C=PCB     BR: CBR(1) [BGN]

**:     AR: RB(DST)=RB(DST)-</C>        PC: LWB
        CC: N=PN, Z=PZ, V=PV, C=PCB     BR: BROC(LKD) [BGN]

**: TST
        CM: TEST INSTRUCTION
        AR: AD=R7B      PC: LW, WIOL    IO: INTCK
        BR: CBR(1) [.+2]

**:     AR: RA(DST)     PC: LWB
        CC: N=PN, Z=PZ, V=0, C=0        BR: BROC(LKD) [BGN]

**:     AR: D   PC: LWB IO: IDATIR(CMI)
        CC: N=PN, Z=PZ, V=0, C=0        BR: CBR(1) [BGN+1]

**: SWAB
        CM: SWAP BYTE INSTRUCTION
        AR: D=DSWB      PC: LB, WIOH    IO: DATO(CM)
        CC: N=PN, Z=PZ, V=0, C=0        BR: CBR(1) [BGN]

**:     AR: D=RA(DST)

**:     AR: RB(DST)=DSWB

**:     AR: RA(DST)     PC: LB
        CC: N=PN, Z=PZ, V=0, C=0        BR: BROC(LKD) [BGN]

**: SXT
        CM: SIGN EXTEND INSTRUCTION
        AR: D=R0B-R0A-</N>      PC: LW, WIOL    IO: DATO(CM)
```

```
        CC: N=PN, Z=PZ, V=0, C=C        BR: CBR(1) [BGN]

**:     AR: RB(DST)=RB(DST)-RA(DST)-</N>
        CC: N=PN, Z=PZ, V=0, C=C        BR: CBR(1) [BGN+1]

**: XOR
        CM: EXCLUSIVE OR INSTRUCTION
        AR: D=D XOR RA(SRC)     PC: LW, WIOH    IO: DATO(CM)
        CC: N=PN, Z=PZ, V=0, C=C        BR: CBR(1) [BGN]

**:     AR: RB(DST)=RB(DST) XOR RA(SRC)
        CC: N=PN, Z=PZ, V=0, C=C        BR: BROC(LKD) [BGN]

**: ROR
        CM: ROTATE RIGHT INSTRUCTION
        AR: R10B=D, SRD(C)      PC: LWB, WIOH
        CC: C=LSBR, N=N, Z=Z, V=V       BR: CBR(1) [SRD]

**:     AR: RB(DST)=RB(DST), SRD(C)     PC: LWB
        CC: C=LSBR, N=N, Z=Z, V=V       BR: CBR(1) [SR0]

**: SRD
        AR: D=R10B      PC: LWB IO: DATO(CM,BYT)
        CC: C=C, N=PN, Z=PZ, V=PNXC     BR: CBR(1) [BGN]

**: SR0
        AR: RB(DST)     PC: LWB
        CC: C=C, N=PN, Z=PZ, V=PNXC     BR: BROC(LKD) [BGN]

**: ROL
        CM: ROTATE LEFT INSTRUCTION
        AR: R10B=D, SRU(C)      PC: LWB, WIOH
        CC: C=MSBR, N=N, Z=Z, V=V       BR: CBR(1) [SRD]

**:     AR: RB(DST)=RB(DST), SRU(C)     PC: LWB
        CC: C=MSBR, N=N, Z=Z, V=V       BR: CBR(1) [SR0]

**: ASR
        CM: ARITHMETIC SHIFT RIGHT INSTRUCTION
        AR: R10B=D, SRD(MSBR)   PC: LWB, WIOH
        CC: C=LSBR, N=N, Z=Z, V=V       BR: CBR(1) [SRD]

**:     AR: RB(DST)=RB(DST), SRD(MSBR)  PC: LWB
        CC: C=LSBR, N=N, Z=Z, V=V       BR: CBR(1) [SR0]

**: ASL
        CM: ARITHMETIC SHIFT LEFT INSTRUCTION
        AR: R10B=D, SRU(0)      PC: LWB, WIOH
        CC: C=MSBR, N=N, Z=Z, V=V       BR: CBR(1) [SRD]

**:     AR: RB(DST)=RB(DST), SRU(0)     PC: LWB
        CC: C=MSBR, N=N, Z=Z, V=V       BR: CBR(1) [SR0]

        PG:     CONTROL, TRAP, AND OTHER INSTRUCTIONS

**: CLRC
        CM: CLEAR CONDITION CODE INSTRUCTIONS
        CC: C=CLR, N=CLR, Z=CLR, V=CLR  BR: BROC(LKD) [BGN]

**: SETC
        CM: SET CONDITION CODE INSTRUCTIONS
        CC: C=SET, N=SET, Z=SET, V=SET  BR: BROC(LKD) [BGN]

**: SOB
        CM: SUBTRACT 1 AND BRANCH IF NOT = 0
        AR: RB(SRC)=RB(SRC)-<0>, AD=R7A

**:     AR: R10B=IR5, SRU(0)
        IO: INTCK       BR: CBN(Z') [BGN+3]

**:     AR: AD;R7B=R7B-R10A-<1>
        IO: DATIR(CMI)  BR: CBR(1) [BGN+1]

**: MARK
        CM: MARK INSTRUCTION
        AR: R10B=IR5+<1>, SRU(0)
```

```
**:     AR: AD;R10B=R6A+R10B+<0>
        IO: DATI(CMI)   BR: CBN(CUM) [.+2]

**:     AR: R6B=R10A+[+2]+<0>   BR: CBR(1) [.+2]

**:     AR: R16B=R10A+[+2]+<0>

**:     AR: AD;R7B=R5A  PC: LW, WIOL    IO: INTCK

**:     AR: R5B=D       IO: IDATIR(CMI) BR: CBR(1) [BGN+1]

**: RTS
        CM: RETURN FROM SUBRUTINE INSTRUCTION
        AR: R10B=R6A+[+2]+<0>, AD=R6A   PC: LW, WIOL
        IO: DATI(CMI)   BR: CBN(CUM) [.+2]

**:     AR: R6B=R10A    BR: CBR(1) [.+2]

**:     AR: R16B=R10A

**:     AR: R7B=RA(DST)

**:     AR: RB(DST)=D   PC: LW, WIOH    BR: CBR(1) [BGN]

**: RTI
        CM: RTI/RTT INSTRUCTIONS
        AR: R10B=R6A+[+2]+<0>,AD=R6A    PC: LW,WIOL
        IO: DATI(CMI)   BR: CBN(CUM) [.+2]

**:     AR: R6B=R6A+[+4]+<0>    BR: CBR(1) [.+2]

**:     AR: R16B=R6A+[+4]+<0>

**:     AR: R7B=D,AD=R10A       PC: LW,WIOH
        IO: DATI(CMI)

**:     CM: SET T BIT & CC'S / DONE IF IN USER MODE
        AR: PSR=D       CC: C=CPO,Z=CPO,V=CPO,N=CPO
        PC: LW,WIOH     BR: CBN(CUM) [BGN]

**:     CM: ELSE RESTORE ALL STATUS FROM STACK
        AR: AD=[+177776]        IO: DATO(OFF)
        BR: CBR(1) [BGN]

**: RESET
        CM: RESET INSTRUCTION
        PC: LW, WIOL    IO: INTCK       BR: CBN(CUM) [BGN+3]

**:     PC: LW, WIOL    IO: INIT        BR: CBR(1) [BGN+3]

**: WAIT
        CM: WAIT FOR INTERRUPT INSTRUCTION
        IO: INTCK       BR: CNR(IP)

**:     AR: R7B=R7A+[+2]+<0>    BR: CBIN(1) <INST>

**: JPR0
        CM: JMP/JSR (0) TRAP
        AR: AD;R10B=[+6]        PC: LW, WIOL
        IO: DATI(KI)    BR: CBN(1) [RES+1]

**: JSR
        CM: JUMP TO SUBRUTINE INSTRUCTION
        AR: D=RA(SRC)   BR: CBN(CUM) [.+2]

**:     AR: AD;R6B=R6A-[+2]-<1> PC: LW, WIOL
        IO: DATO(CMI)   BR: CBR(1) [.+2]

**:     AR: AD;R16B=R6A-[+2]-<1>        PC: LW, WIOL
        IO: DATO(CMI)

**:     AR: RB(SRC)=R7A

**: JMP
        CM: JUMP INSTRUCTION
```

```
        AR: AD;R7B=R11A PC: LW, WIOL
        IO: INTCK       BR: CBR(1) [BGN+3]

        PG:     RESERVED INSTRUCTION TRAP HANDLER

**: RSRVINST
        CM: GET STATUS
        AR: AD=[+177764]        IO: DATI(OFF)

**:     CM: USE STATE 'EX'/'E5' FOR ADDITIONAL CODE
        AR: SCE5=D              PC: LW,WIOH
        SP: CLR,SE5

**: RSRVSTAT
        CM: GO TO ADDITIONAL CODE IF DEFINED
        BR: CJIN(N') <SCR>

**:     CM: ELSE NOT DEFINED
        AR: SCE7=[RES]
        SP: CLR,SE7

**:     CM: RESET CPU STATE TO 'RES'/'E7'
        BR: CBIN(1) <SCR>

        PG:     SPL INSTRUCTION (11/45, 60, & 70)

**: SPL
        CM: SET PRIORITY LEVEL
        AR: R10B=R10B+R10A+<N>
        BR: CBN(CUM) [BGN]

**:     AR: R10B=R10B+R10A+<Z>

**:     AR: R10B=R10B+R10A+<V>

**:     AR: D=R10B+R10A+<C>

**:     AR: AD=[+177776]        IO: DATOB(OFF)
        BR: CBR(1) [BGN]

        PG:     PROCESOR STATUS WORD INSTRUCTIONS

**: MTPS
        CM: MOVE TO PS / DONE IF USER MODE
        AR: D           CC: C=CPO,V=CPO,Z=CPO,N=CPO
        PC: LW, WIOH    BR: CBN(CUM) [BGN]

**:     CM: ELSE IN KERNEL MODE - CHANGE PRIORITY TOO
        AR: AD=[+177776]
        IO: DATOB(OFF)  BR: CBR(1) [BGN]

**: MFPS0
        CM: MOVE FROM PS INSTRUCTION
        AR: D=PSR       BR: CBR(1) [MOVB+2]

**: MFPSX
        AR: D=PSR       CC: C=C, V=0, N=PN, Z=PZ
        PC: LB, WIOL    IO: DATOB(CM)   BR: CBR(1) [BGN]

        PG:     MEMORY MANAGEMENT INSTRUCTIONS

        .=: MORIGIN+347

**: MTPIX
        AR: SCR0=/SCR0

**:     AR: D=R10A      BR: CBN(/Z') [GTSTK]

**:     IO: IDATO(PM)

**: MEND
        AR: R10B=D, AD=R7A      CC: C=C, V=0, N=PN, Z=PZ
        IO: INTCK       BR: CBR(1) [BGN+3]      PC: LW, WIOL

**: MFPIX
        CM: MFPI INSTRUCTION
```

```
        IO: IDATI(PMI)  BR: BROC(CUM) [MFOUT]

**: MFPI0
        AR: R10B=[+106]

**:     AR: R10A-IR-<1> PC: LB  BR: CBN(PUM) [.+2]

**:     AR: D=R6B       BR: BROC(Z') [MFOUT-2]

**:     AR: D=R16B      BR: BROC(Z') [MFOUT-2]

**:     AR: D=RA(DST)   BR: BROC(CUM) [MFOUT]

**:     BR: CBN(CUM) [.+2]

**: MFOUT
        AR: AD;R6B=R6A-[+2]-<1> PC: LW, WIOL
        IO: DATO(CM)    BR: CBR(1) [MEND]

**:     AR: AD;R16B=R6A-[+2]-<1>        PC: LW, WIOL
        IO: DATO(CM)    BR: CBR(1) [MEND]

**: MTPI0
        CM: MTPI INSTRUCTION
        AR: R10B=[+206], AD=R6A IO: DATI(CM)
**:     AR: R10A-IR-<1> PC: LB  BR: CBN(CUM) [.+2]

**:     AR: R6B=R6A+[+2]+<0>    BR: BROC(Z') [MTOUT]

**:     AR: R16B=R6A+[+2]+<0>   BR: BROC(Z') [MTOUT]

**: MTOUT
        AR: RB(DST)=D   CC: C=C, V=0, N=PN, Z=PZ
        PC: LW, WIOH    BR: CBR(1) [BGN]

**:     AR: AD=R7A      PC: LW, WIOL
        IO: INTCK       BR: CBN(PUM) [.+2]

**:     AR: R6B=D       CC: C=C, V=0, N=PN, Z=PZ
        IO: IDATIR(CMI) BR: CBR(1) [BGN+1]

**:     AR: R16B=D      CC: C=C, V=0, N=PN, Z=PZ
        IO: IDATIR(CMI) BR: CBR(1) [BGN+1]

**: GTSTK
        AR: AD=R6A      PC: LW, WIOL
        IO: DATI(CMI)   BR: CBN(CUM) [.+2]

**:     AR: R6B=R6A+[+2]+<0>    SP: SE3, SE5
        BR: CBR(1) [.+2]

**:     AR: R16B=R6A+[+2]+<0>   SP: SE3, SE5

**:     AR: R10B=D, AD=R7A      PC: LW, WIOH
        BR: CBIN(1) <INST>

        NA: MORGEND=.

        PG:     INSTRUCTION CODE SPECIFICATIONS

        .=: IORIGIN+0

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES
```

```
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
```

```
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: HALT        PS: RES
**:     MA: WAIT        PS: EX
**:     MA: RTI PS: EX
**:     MA: RES PS: BPT IN: TI  CM: BPT INSTRUCTION
**:     MA: RES PS: IOT IN: TI  CM: IOT INSTRUCTION
**:     MA:  RESET      PS: EX
**:     MA: RTI PS: EX  IN: TI  CM: RTT INSTRUCTION
**:     MA: RSRVINST    PS: RES
**:     MA: MOV         PS: SRC, DST, EX        IN: NDA
**:     MA: MOV+1       PS: DST, EX     IN: NDA
**:     MA: MOV+2       PS: SRC, EX     IN: C2, WT
**:     MA: MOV+3       PS: EX  IN: C1, WT
**:     MA: MOVB        PS: SRC, DST, EX        IN: NDA, BYT
**:     MA: MOVB+1      PS: DST, EX     IN: NDA, BYT
**:     MA: MOVB+2      PS: SRC, EX     IN: C2, BYT, WT
**:     MA: MOVB+3      PS: EX  IN: C1, BYT, WT
**:     MA: CMP         PS: SRC, DST, EX        IN: SP
**:     MA: CMP+1       PS: DST, EX     IN: SP
**:     MA: CMP+2       PS: SRC, EX     IN: C2, WT
```

```
**:     MA: CMP+3       PS: EX  IN: FRC, WT
**:     MA: CMP         PS: SRC, DST, EX        IN: BYT, SP
**:     MA: CMP+1       PS: DST, EX     IN: BYT, SP
**:     MA: CMP+2       PS: SRC, EX     IN: C2, BYT, WT
**:     MA: CMP+3       PS: EX  IN: FRC, BYT, WT
**:     MA: BIT         PS: SRC, DST, EX        IN: SP
**:     MA: BIT+1       PS: DST, EX     IN: SP
**:     MA: BIT+2       PS: SRC, EX     IN: C2, WT
**:     MA: BIT+3       PS: EX  IN: FRC, WT
**:     MA: BIT         PS: SRC, DST, EX        IN: SP, BYT
**:     MA: BIT+1       PS: DST, EX     IN: SP, BYT
**:     MA: BIT+2       PS: SRC, EX     IN: C2, BYT, WT
**:     MA: BIT+3       PS: EX  IN: FRC, BYT, WT
**:     MA: BIC         PS: SRC, DST, EX
**:     MA: BIC+1       PS: DST, EX
**:     MA: BIC+2       PS: SRC, EX     IN: C2, WT
**:     MA: BIC+3       PS: EX  IN: C1, WT
**:     MA: BIC         PS: SRC, DST, EX        IN: BYT
**:     MA: BIC+1       PS: DST, EX     IN: BYT
**:     MA: BIC+2       PS: SRC, EX     IN: C2, BYT, WT
**:     MA: BIC+3       PS: EX  IN: C1, BYT, WT
**:     MA: BIS         PS: SRC, DST, EX
**:     MA: BIS+1       PS: DST, EX
**:     MA: BIS+2       PS: SRC, EX     IN: C2, WT
**:     MA: BIS+3       PS: EX  IN: C1, WT
**:     MA: BIS         PS: SRC, DST, EX        IN: BYT
**:     MA: BIS+1       PS: DST, EX     IN: BYT
**:     MA: BIS+2       PS: SRC, EX     IN: C2, BYT, WT
**:     MA: BIS+3       PS: EX  IN: C1, BYT, WT
**:     MA: ADD         PS: SRC, DST, EX
**:     MA: ADD+1       PS: DST, EX
**:     MA: ADD+2       PS: SRC, EX     IN: C2, WT
**:     MA: ADD+3       PS: EX  IN: C1,WT
**:     MA: SUB         PS: SRC, DST, EX
**:     MA: SUB+1       PS: DST, EX
**:     MA: SUB+2       PS: SRC, EX   IN: C2, WT
**:     MA: SUB+3       PS: EX  IN: C1, WT
**:     MA: RTS         PS: EX
```

```
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: SPL         PS: EX
**:     MA: CLRC        PS: EX  IN: FRC, WT
**:     MA: CLRC        PS: EX  IN: FRC, WT
**:     MA: SETC        PS: EX  IN: FRC, WT
**:     MA: SETC        PS: EX  IN: FRC, WT
**:     MA: CLR         PS: DST,EX      IN: NDA
**:     MA: CLR+1       PS: EX  IN: C1, WT
**:     MA: CLR         PS: DST, EX     IN: BYT, NDA
**:     MA: CLR+1       PS: EX  IN: BYT, C1, WT
**:     MA: COM         PS: DST, EX
**:     MA: COM+1       PS: EX  IN: C1, WT
**:     MA: COM         PS: DST, EX     IN: BYT
**:     MA: COM+1       PS: EX  IN: BYT, C1, WT
**:     MA: INC         PS: DST, EX
**:     MA: INC+1       PS: EX  IN: C1, WT
**:     MA: INC         PS: DST, EX     IN: BYT
**:     MA: INC+1       PS: EX  IN: BYT, C1, WT
**:     MA: DEC         PS: DST, EX
**:     MA: DEC+1       PS: EX  IN: C1, WT
**:     MA: DEC         PS: DST, EX     IN: BYT
**:     MA: DEC+1       PS: EX  IN: BYT, C1, WT
**:     MA: NEG         PS: DST, EX
**:     MA: NEG+1       PS: EX  IN: C1, WT
**:     MA: NEG         PS: DST, EX     IN: BYT
**:     MA: NEG+1       PS: EX  IN: BYT, C1, WT
**:     MA: ADC         PS: DST, EX
**:     MA: ADC+1       PS: EX  IN: C1, WT
**:     MA: ADC         PS: DST, EX     IN: BYT
**:     MA: ADC+1       PS: EX  IN: BYT, C1, WT
**:     MA: SBC         PS: DST, EX
**:     MA: SBC+1       PS: EX  IN: C1, WT
**:     MA: SBC         PS: DST, EX     IN: BYT
**:     MA: SBC+1       PS: EX  IN: BYT, C1, WT
**:     MA: TST         PS: DST, EX     IN: SP
**:     MA: TST+1       PS: EX  IN: FRC, WT
**:     MA: TST         PS: DST, EX     IN: SP, BYT
```

```
**:     MA: TST+1       PS: EX  IN: BYT, FRC, WT
**:     MA: ROR         PS: DST, EX
**:     MA: ROR+1       PS: EX  IN: C1, WT
**:     MA: ROR         PS: DST, EX     IN: BYT
**:     MA: ROR+1       PS: EX  IN: BYT, C1, WT
**:     MA: ROL         PS: DST, EX
**:     MA: ROL+1       PS: EX  IN: C1, WT
**:     MA: ROL         PS: DST, EX     IN: BYT
**:     MA: ROL+1       PS: EX  IN: BYT, C1, WT
**:     MA: ASR         PS: DST, EX
**:     MA: ASR+1       PS: EX  IN: C1, WT
**:     MA: ASR         PS: DST, EX     IN: BYT
**:     MA: ASR+1       PS: EX  IN: BYT, C1, WT
**:     MA: ASL         PS: DST, EX
**:     MA: ASL+1       PS: EX  IN: C1, WT
**:     MA: ASL         PS: DST, EX     IN: BYT
**:     MA: ASL+1       PS: EX  IN: BYT, C1, WT
**:     MA: MARK        PS: EX
**:     MA: MARK        PS: EX
**:     MA: MTPS        PS: DST, EX     IN: SP, BYT
**:     MA: MTPS        PS: DST, EX     IN: BYT
**:     MA: MFPIX       PS: DST, EX     IN: NDA
**:     MA: MFPI0       PS: EX
**:     MA: MFPIX       PS: DST, EX     IN: NDA CM: MFPD INST.
**:     MA: MFPI0       PS: EX  CM: MFPD INST.
**:     MA: MTPIX       PS: EX  IN: NDA
**:     MA: MTPI0       PS: EX
**:     MA: MTPIX       PS: EX  IN: NDA CM: MTPD INST.
**:     MA: MTPI0       PS: EX  CM: MTPD INST.
**:     MA: SXT         PS: DST, EX     IN: NDA
**:     MA: SXT+1       PS: EX          IN: C1, WT
**:     MA: MFPSX       PS: DST, EX     IN: NDA
**:     MA: MFPS0       PS: EX
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
```

```
**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: XOR         PS: DST, EX

**:     MA: XOR+1       PS: EX  IN: C1, WT

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: SOB PS: EX

**:     MA: SOB PS: EX

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES
```

```
**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: JMP         PS: DST, EX     IN: NDA

**:     MA: JPR0        PS: RES

**:     MA: SWAB        PS: DST, EX

**:     MA: SWAB+1      PS: EX  IN: C1, WT

**:     MA: NOBR        PS: EX  IN: FRC, WT

**:     MA: BR          PS: EX

**:     MA: JSR         PS: DST, EX     IN: NDA

**:     MA: JPR0        PS: RES

**:     MA: RES PS: EMT         IN: TI  CM: EMT INSTRUCTION

**:     MA: RES PS: TRAP        IN: TI  CM: TRAP INSTRUCTION

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

        PG:     VECTOR DEFINITIONS

        VE: VGRP=+0
        VE: V37= CHLT / +37
        VE: V36= CHLT / +36
        VE: V35= CHLT / +35
        VE: V34= FBHINT  / +6
        VE: V33= CHLT / +33
        VE: V32= YSTB / +6
        VE: V31= FBHINT  / +6
        VE: V30= FBHINT  / +252
        VE: V27= CHLT / +0
        VE: V26= CHLT / +0
        VE: V25= CHLT / +0
        VE: V24= CHLT / +0
        VE: V23= CHLT / +0
        VE: V22= CHLT / +0
        VE: V21= CHLT / +0
        VE: V20= RES  / +12
        VE: V17= RES  / +22
        VE: V16= RES  / +16
        VE: V15= RES  / +32
        VE: V14= RES  / +36
        VE: V13= CHLT / +13
        VE: V12= YSTB   / +16
        VE: V11= YSTB   / +6
        VE: V10= PF     / +26
        VE: V7 = INT17 / +102
        VE: V6 = INT17 / +62
        VE: V5 = INT17 / +66
        VE: V4 = INT17 / +172
        VE: V3 = INT17 / +176
        VE: V2 = INT17 / +272
        VE: V1 = INT17 / +276
```

```
        VE: V0 = INT0  / +242
```

```
        PG:     EXTENDED INSTRUCTION SET

        .=: MORGEND
        CM: EXTENDED INSTRUCTION SET (EIS)
            ASH SHIFT ARITHMETICALLY
            ASHC  ARITHMETIC SHIFT COMBINED
            MUL MULTIPLY
            DIV DIVIDE

        MODIFIED MAY 1980 TO USE ONLY R10 & R11 IN 4B MODE

**: ASH
        CM: ARITHMETIC SHIFT INSTRUCTION
        CM: LOAD COUNTER, TEST FOR LEFT / RIGHT SHIFT
        AR: CNTR=DSX5   PC: LW, WIOH
        CC: N=PN, C=0, V=0, Z=0

**:     CM: SET UP FOR 32 BIT SHIFTING
        AR: R10B=0,D=RA(SRC)    PC: LW
        BR: CBN(Z') [END1]

**:     CM: SET UPPER WORD DATA, BRANCH ON RIGHT SHIFT
        AR: R10B=D      PC: UW
        CC: N=PN, C=C, V=V, Z=Z BR: CBN(N) [RTST]

**:     CM: EXTEND SIGN OF RB(SRC) THROUGH Q
        AR: Q=R0A-R0B-</N>      PC: 4B

**: LTST
        CM: LEFT SHIFTING
        AR: R10B=R10B,SRU(0),SQ(MSBR)   PC: 4B, CCL
        CC: C=MSBR, V=0, N=N, Z=0       BR: CNR(/CNTR)

**:     CM: SHIFT LAST SIGN BIT INTO Q REG
        AR: R11B=R10A,SRU(0),SQ(MSBR)   PC: 4B

**:     CM: DID SIGN OF REGISTER CHANGE
        AR: OUT=Q+<N>   PC: 4B
        CC: C=PZB, V=C, N=N, Z=Z

**:     AR: D=R10B      PC: UW
        CC: C=V, V=C, N=0, Z=0  BR: CBR(1) [END1]

**: RTST
        CM: RIGHT SHIFTING
        AR: R10B=R10B,SRD(MSBR)         PC: UW, CCL
        CC: C=LSBR, V=0, N=0, Z=0       BR: CNR(/CNTR)

**:     AR: D=R10B      PC: UW

**: END1
        AR: RB(SRC)=D   PC: LW
        CC: C=C, V=V, N=PN, Z=PZ        BR: CBR(1) [BGN]

**: ASHC
        CM: ARITHMETIC SHIFT COMBINED
        CM: TEST FOR LEFT / RIGHT SHIFT
        AR: CNTR=DSX5   PC: LW, WIOH
        CC: N=PN, C=0, V=0, Z=0

**:     CM: SET UP FOR 32 BIT SHIFTING
        AR: D=RB(SRC)   PC: LW
        BR: CBN(Z') [END2]

**:     AR: R10B=RA(SRC!1)      PC: LW

**:     AR: R10B=D              PC: UW
        CC: N=PN, V=V, C=C, Z=Z BR: CBN(N) [RSHC]

**:     CM: EXTEND SIGN OF RB(SRC) THROUGH Q REG
        AR: Q=R0A-R0B-</N>      PC: 4B

**: LSHC
        CM: LEFT SHIFTING
        AR: R10B=R10B,SRU(0),SQ(MSBR)   PC: 4B, CCL
        CC: C=MSBR, V=0, N=N, Z=0       BR: CNR(/CNTR)
```

```
**:     CM: SHIFT LAST SIGN BIT INTO Q REG
        AR: R11B=R10A,SRU(0),SQ(MSBR)   PC: 4B

**:     CM: DID SIGN OF REGISTER CHANGE
        AR: OUT=Q+<N>   PC: 4B
        CC: C=PZB, V=C, N=N, Z=Z

**:     AR: D=R10B      PC: UW
        CC: V=C, C=V, N=0, Z=0  BR: CBR(1) [END2-1]

**: RSHC
        CM: RIGHT SHIFTING
        AR: R10B=R10B,SRD(MSBR)         PC: 4B, CCL
        CC: C=LSBR, V=0, N=0, Z=0       BR: CNR(/CNTR)

**:     AR: D=R10B      PC: UW

**:     AR: RB(SRC!1)=R10A      PC: LW

**: END2
        AR: RB(SRC)=D,D=RA(SRC!1)       PC: LW
        CC: C=C, V=V, N=PN, Z=PZ

**:     AR: RB(SRC!1)=D         PC: LW
        CC: C=C, V=V, N=N, Z=PZAZ       BR: CBR(1) [BGN]

**: MUL
        CM: INITIALIZE LOOP COUNTER
        AR: CNTR=[+17]

**:     CM: INITIALIZE RESULT
        AR: R11B=0

**:     CM: LOAD MULTIPLIER
        AR: R10B=D,SRD(0)       PC: LW, WIOH
        CC: C=LSBR, V=V, N=N, Z=Z

**:     CM: ENTER MULTIPLIER SHIFTED ONCE
        AR: Q=R10B

**:     CM: MULTIPLICATION IS PERFORMED BY
        CM: ADDING THE MULTIPLICAND TO THE RESULT
        CM: IF THE MULTIPLIER BIT SHIFTED FROM Q IS A ONE
        CM: ELSE A 0 IS ADDED
        AR: R11B=R11B+RA(SRC)+<0>,SRD(PNXPV),SQ(LSBR)
        CC: C=LSBQ, V=V, N=N, Z=Z       PC: LW, MM, CCL
        BR: CNR(/CNTR)

**:     CM: LAST CYCLE OF TWO'S COMPLEMENT MULTIPLY
        AR: R11B=R11B-RA(SRC)-<1>,SRD(PNXPV),SQ(LSBR)
        CC: C=LSBQ, V=V, N=N, Z=Z       PC: LW, MM

**:     CM: STORE HIGH ORDER RESULT
        AR: RB(SRC)=R11A        PC: LW
        CC: C=0, V=0, N=PN, Z=PZ

**:     CM: STORE LOW ORDER RESULT
        AR: RB(SRC!1)=Q PC: LW
        CC: C=0, V=0, N=N, Z=PZAZ       IO: INTCK

**:     CM: SET UP TO CHECK RANGE
        AR: R10B=[+77777],AD=R7A
        IO: DATIR(CMI)  BR: CBN(N) [NGN]

**: PSN
        CM: CHECK FOR  RESULT > 2^15-1
        AR: OUT=Q-R10A-<1>

**:     AR: OUT=R11A-<C'>
        CC: V=PNXPV, C=C, N=N, Z=Z

**:     AR: OUT=/PSR
        CC: V=CPO, C=C, N=N, Z=Z        BR: CBR(1) [END3]

**: NGN
```

```
        CM: CHECK FOR RESULT < -2^15
        AR: OUT=Q+R10A+<1>

**:     AR: OUT=R11A+<C'>
        CC: V=PNXPV, C=C, N=N, Z=Z

**: END3
        CC: C=V, V=0, N=N, Z=Z  BR: CBR(1) [BGN+1]

**: DIV
        CM: DIVISION IS PERFORMED BY A NON RESTORING
        CM: METHOD. THE COMPLEMENT OF THE QUOTIENT IS DEVELOPED
        CM: IN THE Q REGISTER AND THE REMAINDER IS LEFT IN R10.
        CM: LOAD HIGH ORDER PART OF DIVIDEND
        AR: R10B=RA(SRC)        PC: LW
        CC: C=0, V=0, N=PN, Z=PZ

**:     CM: LOAD LOW ORDER PART OF DIVIDEND, SKIP IF POSITIVE
        AR: Q=RB(SRC!1) PC: LW
        CC: C=0, V=0, N=N, Z=PZAZ       BR: CBN(/N) [.+3]

**:     CM: COMPUTE TWO'S COMPLEMENT OF 32 BIT DIVIDEND
        AR: Q=-Q-<1>

**:     AR: R10B=-R10B-<C'>

**:     CM: LOAD DIVISOR, CHECK IF = 0
        AR: R11B=D      PC: LW, WIOH
        CC: C=PZ, V=0, N=N, Z=Z

**:     CM: SET LOOP COUNTER, SKIP IF DIVISOR POSITIVE
        AR: CNTR=[+17]  BR: CBN(/N') [.+2]

**:     CM: MAKE DIVISOR POSITIVE
        AR: R11B=-R11B-<1>

**:     CM: CHECK FOR OVERFLOW, FALL TO ABORT IF SOURCE = 0
        AR: R10B=R10B-R11A-<1>,SRU(MSBQ),SQ(MSBR)
        CC: C=C, V=PNXPV, N=N, Z=Z      BR: CBN(/C) [.+2]

**:     CM: LEAVE R(SRC) & R(SRC!1), SET CODES
        CC: C=1, V=1, N=N, Z=Z  BR: CBR(1) [BGN]

**:     CM: ABORT IF OVERFLOW, IE. DIVIDEND > DIVISOR
        AR: OUT=/PSR
        CC: C=C, V=CPO, N=N, Z=Z        BR: CBN(/V) [BGN]

**:     CM: SAVE SIGN OF DIVIDEND
        CM: COMPUTE SIGN OF QUOTIENT
        AR: OUT=D       CC: C=0, V=0, N=PNXN, Z=N

**:     CM: FIRST OPERATION IS AN ADD
        AR: R10B=R10B+R11A+<0>,SRU(MSBQ),SQ(MSBR)
        CC: C=PN, V=Z, N=N, Z=0

**:     CM: REPEAT NEXT CYCLE 15 TIMES
        CM: IF THE CARRY FLAG IS A 1 DO AN ADDITION
        CM: IF THE CARRY FLAG IS A 0 DO A SUBTRACTION
        CM: THE CURRENT SIGN BIT IS THE COMPLEMENT OF
        CM: THE DEVELOPED QUOTIENT BIT
        AR: D;R10B=R10B+R11A+</C>,SRU(MSBQ),SQ(MSBR)
        PC: MNR, CCL, LW        CC: C=PN, V=V, N=N, Z=0
        BR: CNR(/CNTR)

**:     CM: GET QUOTIENT, SKIP IF SIGN IS CORRECT
        AR: RB(SRC)=/Q
        CC: C=C, V=V, N=PN, Z=PZ        BR: CBN(/N) [.+2]

**:     CM: CORRECT SIGN OF QUOTIENT
        AR: RB(SRC)=-RB(SRC)-<1>
        CC: C=C, V=V, N=PN, Z=PZ

**:     CM: RESTORE REMAINDER CORRECTION
        CM: IF C = 1 ADD R11  ELSE ADD 0
        CM: CHECK IF SIGN OF REMAINDER NEEDS CORRECTING
        AR: RB(SRC!1)=D+R11A+<0>        PC: MM, LW
```

```
        CC: C=0, V=0, N=N, Z=Z  BR: CBN(/V) [BGN]

**:     CM: CORRECT SIGN OF REMAINDER
        AR: RB(SRC!1)=-RB(SRC!1)-<1>
        CC: C=0, V=0, N=N, Z=Z  BR: CBR(1) [BGN]

        NA: MORGEND=.

        PG:     EXTENDED INSTRUCTION CODES

        .=: IORIGIN+300

**:     MA: MUL PS: DST, EX     IN: SP

**:     MA: MUL PS: DST, EX

**:     MA: DIV PS: DST, EX     IN: SP

**:     MA: DIV PS: DST, EX

**:     MA: ASH PS: DST, EX     IN: SP

**:     MA: ASH PS: DST, EX

**:     MA: ASHC  PS: DST, EX   IN: SP

**:     MA: ASHC  PS: DST, EX
```

```
        PG:     FIS OVERLAY

        CM:       THE FOLLOWING INSTRUCTION CODE
                DEFINITIONS OVERLAY THE ARB11 CODE TO ADD THE FOLLOWING
                FLOATING POINT INSTRUCTIONS.

                PDP-11/35 AND LSI-11 COMPATIBLE

                        FADD    07500R
                        FSUB    07501R
                        FMUL    07502R
                        FDIV    07503R

        CM:     NOTES ON CODING -
                1) CONDITION CODES ARE CHANGED DURING OPERAND
                   ACCESS. MEMORY TRAPS WILL HAVE INVALID CC'S

                2) A.GETARG, B.GETARG, AND ARG.SAV ROUTINES
                   SUPPORT FLOATING AND DOUBLE FORMATS

                3) THE SINGLE PRECISION FLOATING ROUTINES
                        ADD.24
                        MUL.24
                        DIV.24
                        AL.24
                        NORM.24
                   MAY BE CALLED BY THE FPP

                4) DOUBLE PRECISION FLOATING POINT OPERATIONS
                    ARE SUPPORTED BY THE FOLLOWING ROUTINES
                        ADD.56
                        MUL.56
                        DIV.56
                        AL.56
                        NORM.56
                   AND MAY BE CALLED BY THE FPP

        PG:     DEFINE FIS INSTRUCTIONS

        .=:IORIGIN+340

**:     MA: FADD        PS: EX

**:     MA: FSUB        PS: EX

**:     MA: FMUL        PS: EX

**:     MA: FDIV        PS: EX

        .=: MORGEND

        PG:     FDIV

**: FDIV
        CM: GET FLOATING ARGUMENT B
        AR: AD;R10B=RA(DST)     IO: DATI(CMI)
        CC: C=0,Z=0,N=0,V=0     BR: CJN(1) [B.GETARG+1]

**:     CM: ON ZERO ARGUMENT - BRANCH
        AR: AD;RB(DST)=RA(DST)+[+4]+<0>         BR: CBN(C) [DIVZERO]

**:     CM: GET ARGUMENT A
        AR: R10B=RA(DST)
        IO: IDATI(CMI)  BR: CJN(1) [A.GETARG+1]

**:     CM: ON ZERO ARGUMENT - GENERATE A CLEAN ZERO
            PRESET LOOP COUNTER
        AR: CNTR=[+32]          BR: CBN(C) [FIS.SAV]

**:     CM: DO 24-BIT DIVISION  (ROUNDED)
        AR: R14B-R12A-<0>       PC: 4B
        CC: C=PN,N=1,Z=0,V=0    BR: CJN(1) [DIV.24+2]

**:     CM: CHECK EXPONENT
        AR: R4B=R4B+<0>         PC: UW
        BR: CBN(1) [EXPCHK]
```

```
        PG:     FMUL

**: FMUL
        CM: GET FLOATING ARGUMENT B
        AR: AD;R10B=RA(DST)     IO: DATI(CMI)
        CC: C=0,N=0,Z=0,V=0     BR: CJN(1) [B.GETARG+1]

**:     CM: CHECK FOR ZERO ARGUMENT
        AR: AD;RB(DST)=RA(DST)+[+4]+<0>         BR: CBN(C) [FIS.SAV]

**:     CM: GET A ARGUMENT
        AR: R10B=RA(DST)
        IO: IDATI(CMI)          BR: CJN(1) [A.GETARG+1]

**:     CM: CHECK FOR ZERO ARGUMENT
        AR: CNTR=[+30]          BR: CBN(C) [FIS.SAV]

**:     CM: DO 24-BIT MULTIPLY (ROUNDED)
        AR: R14B=R14B,SRD(0)    PC: 4B
        CC: C=LSBR,N=1,Z=0,V=V  BR: CJN(1) [MUL.24+2]

**:     CM: CHECK EXPONENT
        AR: R4B=R4B+<0>         PC: UW
        BR: CBN(1) [EXPCHK]

        PG:     FSUB

**: FSUB
        CM: GET FLOATING ARGUMENT B
        AR: AD;R10B=RA(DST)     IO: DATI(CMI)
        CC: C=0,N=0,Z=0,V=0     BR: CJN(1) [B.GETARG+1]

**:     CM: NEGATE SIGN OF OPERAND
        AR: R5B=R5A XOR [+100000]       PC: UW
        BR: CBN(1) [FADD+1]

        PG:     FADD

**: FADD
        CM: GET FLOATING ARGUMENT B
        AR: AD;R10B=RA(DST)     IO: DATI(CMI)
        CC: C=0,N=0,Z=0,V=0     BR: CJN(1) [B.GETARG+1]

**:     CM: GET A ARGUMENT
        AR: AD;RB(DST)=RA(DST)+[+4]+<0>         IO: DATI(CMI)

**:     AR: R10B=RA(DST)        BR: CJN(1) [A.GETARG+1]

**:     CM: DO 24-BIT ADDITION (ROUNDED)
        AR: R2B                 PC: UW
        CC: C=0,N=1,Z=PZ,V=V    BR: CJN(1) [ADD.24+1]

**:     CM: ON ZERO RESULT (C=1) - GENERATE A CLEAN ZERO
            IF A OR B = 0  (Z=1) - SAVE RESULT IN B
            CHECK EXPONENT
        AR: R4A+<0>     PC: UW
        BR: CBN(COZ) [FIS.SAV]

**: EXPCHK
        CM: BRANCH ON UNDERFLOW, CHECK OVERFLOW
        AR: R4A-[+400]-<1>      PC: UW
        BR: CBN(BLE') [UNDFLO]

**:     CM: BRANCH ON OVERFLOW
        CC: C=0,N=0,Z=0,V=V     BR: CBN(/N') [OVRFLO]

**: FIS.SAV
        CM: SET ADDRESS / GO SAVE ARGUMENT
        AR: R10B=RA(DST)        BR: CJN(1) [ARG.SAV]

**:     CM: WAIT TO START NEXT INSTRUCTION
        AR: AD=R7A+[+0]+<0>     PC: LW,WIOL
        IO: INTCK               BR: CBN(1) [BGN+3]

        PG:     SAVE ARGUMENT ROUTINE
```

```
CM:     IF C=1 A CLEAN ZERO IS GENERATED, ELSE
        R4 (UW) - EXPONENT
        R5 (UW) - FRACTION SIGN
        R14 (4B) - FRACTION
        R15 (4B) - FRACTION
        ARE COMBINED INTO THE STANDARD FLOATING POINT
        FORMAT = FLOATING (V=0)
         SIGN(1-BIT)/EXPONENT(8-BITS)/FRACTION(23-BITS)
        FORMAT = DOUBLE (V=1)
         SIGN(1-BIT)/EXPONENT(8-BITS)/FRACTION(55-BITS)
        AND SAVED AT THE ADDRESS IN R10(LW)

**: ARG.SAV
        CM: MASK FRACTION / GO COMBINE SIGN, EXPONENT, AND FRACTION
        AR: R14B=[+177600] MASK R14A    PC: UW
        BR: CJN(1) [CMB.SEF+1]

**: ARG.SAVA
        CM: SAVE RESULT (R14-4B)
        AR: AD=R10A             IO: DATO(CM)

**:     CM: CLEAR EXTRA FLAGS
        AR: D=R14B              PC: LW,WIOL
        CC: C=0,V=V,N=N,Z=Z

**:     AR: AD=R10A+[+2]+<0>
        IO: DATO(CM)            BR: CRN(/V)

**:     AR: D=R15B              PC: UW,WIOL
        CC: C=0,V=0,N=N,Z=Z

**:     AR: AD=R10A+[+4]+<0>    IO: DATO(CM)

**:     AR: D=R15B              PC: LW,WIOL

**:     AR: AD=R10A+[+6]+<0>
        IO: DATO(CM)            BR: CRN(1)

**: CMB.SEF
        CM: MASK FRACTION
        AR: R14B=[+177600] MASK R14A    PC: UW

**:     CM: GET SIGN OF RESULT / BRANCH IF NOT A CLEAN ZERO
        AR: R5B=R5B,D=R4A               PC: UW
        CC: C=PN,N=PN,Z=Z,V=V           BR: CBN(/C) [.+3]

**:     CM: ELSE A CLEAN ZERO
        AR: D;R14B=0                    PC: 4B
        CC: C=1,N=0,Z=1,V=V

**:     AR: R15B=0                      PC: 4B
        BR: CRR(1)

**:     CM: POSITION EXPONENT AND COMBINE WITH SIGN
        AR: R4B=DSWB,SRD(C)             PC: UW
        CC: C=PZ,Z=PZ,N=N,V=V

**:     CM: COMBINE HIGH ORDER FRACTION WITH EXPONENT AND SIGN
        AR: D;R14B=R14B OR R4A          PC: UW
        BR: CRN(1)

        PG:     FIS TRAP HANDLERS

**: DIVZERO
        CC: V=V,N=1,C=1,Z=0     BR: CBR(1) [FIS.TRAP]

**: UNDFLO
        CC: V=V,N=1,C=0,Z=0     BR: CBR(1) [FIS.TRAP]

**: OVRFLO
        CC: V=V,N=0,C=0,Z=0

**: FIS.TRAP
        CC: V=1,N=N,C=C,Z=Z     BR: CBN(V) [.+2]
```

```
**:     CM: RESET STACK FOR FLOATING
        AR: RB(DST)=RA(DST)-[+4]-<1>
        BR: CBN(1) [.+2]

**:     CM: RESET STACK FOR DOUBLE
        AR: RB(DST)=RA(DST)-[+10]-<1>

**:     AR: AD;R10B=[+246]      PC: LW,WIOL
        IO: DATI(KI)    BR: CBR(1) [RES+1]

        PG:     GET A OPERAND CODE

CM:     THIS CODE GETS THE A ARGUMENT AT THE ADDRESS IN R10 (LW)
        SET V=0 FOR FLOATING / V=1 FOR DOUBLE
        AND EXITS WITH
        C,Z=1 IF EXPONENT = 0
        N=1 IF ARGUMENT < 0
        R2 (UW) = EXPONENT
        R3 (UW) = SIGN OF FRACTION
        R12 (4B) = FRACTION (24-BITS)
        R13 (4B) = FRACTION (32-BITS)

**: A.GETARG
        AR: AD=R10A             IO: DATI(CMI)

**:     CM: CLEAR HIGH FRACTION
        AR: R12B=0              PC: UW

**:     AR: AD=R10A+[+2]+<0>    PC: LW,WIOL

**:     CM: SAVE HIGH ORDER IN TEMPORARY
            GO BUILD SIGN, EXPONENT, AND HIGH FRACTION
        AR: R2B=D,SRU(0)        PC: UW
        CC: C=MSBR,N=PN,Z=0,V=V
        IO: DATI(CMI)           BR: CJN(1) [A.BLDSEF]

**:     CM: GET DOUBLE FRACTION
        AR: AD=R10A+[+4]+<0>    IO: DATI(CMI)

**:     AR: R13B=D              PC: UW
        IO: DATI(CMI)

**:     AR: AD=R10A+[+6]+<0>    PC: LW,WIOL

**:     CM: DUMBY LOAD AD
        AR: R13B=D,AD=R7A       PC: LW,WIOH
        BR: CRR(1)

        PG:     BUILD SIGN, EXPONENT, AND HIGH FRACTION FOR A

CM:       IF V=0, FLOATING MODE - POPS ONE RETURN OFF STACK THEN RETURNS
          IF V=1, DOUBLE MODE - RETURNS TO CALLER

**: A.BLDSEF
        CM: SAVE SIGN
        AR: R3B=0,SRD(C)        PC: UW

**:     CM: MASK IN FRACTION AND HIDDEN BIT
        AR: R12B=R2A,SRD(1)     PC: UB

**:     CM: SAVE LOW FRACTION / DUMBY LOAD AD
        AR: R12B=D,AD=R7A       PC: LW,WIOH

**:     CM: GET EXPONENT INTO R2
        AR: R2B=0,D=R2A         PC: UW
        BR: CPS(/V)

**:     AR: R2B=DSWB            PC: UB
        CC: Z=PZ,C=PZ,N=N,V=V   BR: CRR(1)

        PG:     GET B OPERAND CODE

CM:     THIS CODE GETS THE B ARGUMENT AT THE ADDRESS IN R10 (LW)
        SET V=0 FOR FLOATING / V=1 FOR DOUBLE
        AND EXITS WITH
        C,Z=1 IF EXPONENT = 0
```

```
        N=1 IF ARGUMENT < 0
        R4 (UW) = EXPONENT
        R5 (UW) = SIGN OF FRACTION
        R14 (4B) = FRACTION (24-BITS)
        R15 (4B) = FRACTION (32-BITS)

**: B.GETARG
        AR: AD=R10A     IO: DATI(CMI)

**:     CM: CLEAR HIGH FRACTION
        AR: R14B=0              PC: UW

**:     AR: AD=R10A+[+2]+<0>    PC: LW,WIOL

**:     CM: SAVE HIGH ORDER IN TEMPORARY
            GO BUILD SIGN, EXPONENT, AND HIGH FRACTION
        AR: R4B=D,SRU(0)        PC: UW
        CC: C=MSBR,N=PN,Z=0,V=V
        IO: DATI(CMI)           BR: CJN(1) [B.BLDSEF]

**:     CM: GET DOUBLE FRACTION
        AR: AD=R10A+[+4]+<0>    IO: DATI(CMI)

**:     AR: AD=R10A+[+6]+<0>    PC: LW,WIOL

**:     AR: R15B=D              PC: UW
        IO: DATI(CMI)

**:     CM: DUMBY LOAD AD
        AR: R15B=D,AD=R7A       PC: LW,WIOH
        BR: CRR(1)

        PG:     BUILD SIGN, EXPONENT, AND HIGH FRACTION FOR B

CM:       IF V=0, FLOATING MODE - POPS ONE RETURN OFF STACK THEN RETURNS
          IF V=1, DOUBLE MODE - RETURNS TO CALLER

**: B.BLDSEF
        CM: SAVE SIGN
        AR: R5B=0,SRD(C)        PC: UW

**:     CM: MASK IN FRACTION AND HIDDEN BIT
        AR: R14B=R4A,SRD(1)     PC: UB

**:     CM: SAVE LOW FRACTION / DUMBY LOAD AD
        AR: R14B=D,AD=R7A       PC: LW,WIOH

**:     CM: GET EXPONENT INTO R4
        AR: R4B=0,D=R4A         PC: UW
        BR: CPS(/V)

**:     AR: R4B=DSWB    PC: UB
        CC: Z=PZ,C=PZ,N=N,V=V   BR: CRR(1)

        PG:     24-BIT ADDITION

CM:     ADDITION PERFORMED BETWEEN REGISTERS
        R12/R2/R3 AND R14/R4/R5
        FRACTION RESULT IN R14 (4B)
        EXPONENT RESULT IN R4 (UW)
        SIGN RESULT IN R5 (UW)
        N IS ROUND/TRUNCATE FLAG
        V IS NOT USED
        C = 1 IF RESULT IS ZERO
        Z = 1 IF ARGUMENT A OR B OR RESULT = 0

**: ADD.24
        CM: IS A ARGUMENT = 0 ?
        AR: R2B         CC: Z=PZ,C=0,N=N,V=V    PC: UW

**:     CM: IF A ARGUMENT IS ZERO - B IS RESULT
            AND IF B=0, THEN A CLEAN ZERO WILL BE GENERATED
        AR: R4B         CC: C=PZ,N=N,Z=Z,V=V    PC: UW
        BR: CRN(Z)

**:     CM: BRANCH IF B ARGUMENT IS NOT ZERO
```

```
            CLEAR NORMALIZE SHIFT REGISTER (Q)
        AR: Q=0         PC: 4B
        BR: CBN(/C) [.+4]

**: COPY.24
        CM: ELSE A IS THE RESULT
        AR: R14B=R12A   PC: 4B

**:     AR: R4B=R2A     PC: UW

**:     AR: R5B=R3A     PC: UW
        CC: C=0,Z=1,N=N,V=V     BR: CRR(1)

**:     CM: ALIGN EXPONENTS
        AR: R12B=R12B,SRU(0)    PC: 4B
        BR: CJN(1) [AL.24+1]

**:     CM: CHECK SIGNS OF ARGUMENTS / DO ADDITION
        AR: R5B XOR R3A         PC: UW
        BR: CJN(1) [ADD.24A]

**:     CM: POST NORMALIZE / NORM.24 RETURNS TO CALLER
        AR: R14B=R14B,SRD(0),SQ(LSBR)   PC: 4B
        CC: C=PZ,N=N,V=V,Z=PZ           BR: CBN(1) [NORM.24+2]

**: ADD.24A
        CM: BRANCH IF SIGNS DIFFER
        AR: R5B                 PC: UW
        BR: CBN(N') [.+2]

**:     CM: ELSE COMPUTE SUM
        AR: R14B=R14B+R12A+<0>  PC: 4B
        BR: CRN(1)

**:     CM: OPPOSITE SIGNS - DIDDLE A LITTLE
        BR: CBN(N') [.+2]

**:     CM: NEGATE R12
        AR: R12B=-R12B-<1>      PC: 4B
        BR: CBN(1) [.+2]

**:     CM: NEGATE R14
        AR: R14B=-R14B-<1>      PC: 4B

**:     CM: COMPUTE RESULT
        AR: R14B=R14B+R12A+<0>  PC: 4B
        CC: C=PN,N=N,V=V,Z=Z

**:     CM: SAVE SIGN OF RESULT
        AR: R5B=0,SRD(C)        PC: UW
        BR: CRN(/C)

**:     CM: GET MAGNITUDE
        AR: R14B=-R14B-<1>      PC: 4B
        BR: CRN(1)

        PG:     24-BIT MULTIPLY ROUTINE

CM:     MULTIPLICAND IN R12 (4B)
        MULTIPLIER IN R14 (4B)
        RESULT LEFT IN R14 (4B) NORMALIZED
        EXPONENT RESULT IN R4 (UW)
        SIGN RESULT IN R5 (UW)
        RESULT IS ROUNDED(N=1)/TRUNCATED(N=0)
        V AND Z NOT EFFECTED, C USED

**: MUL.24
        AR: CNTR=[+30]

**:     CM: PRESHIFT MULTIPLIER
        AR: R14B=R14B,SRD(0)    PC: 4B
        CC: C=LSBR,Z=Z,V=V,N=N

**:     CM: LOAD MULTIPLIER
        AR: Q=R14B              PC: 4B
```

```
**:     CM: INITIALIZE RESULT
        AR: R14B=0              PC: 4B

**:     CM: DO 24-BITS OF MULTIPLY
        AR: R14B=R14B+R12A+<0>,SRD(0),SQ(LSBR)  PC: 4B,MM,CCL
        CC: C=LSBQ,V=V,N=N,Z=Z          BR: CNR(/CNTR)

**:     CM: CHECK MOST SIGNIFICANT BIT
        AR: R14B        CC: C=PN,N=N,Z=Z,V=V    PC: 3B

**:     CM: CONDITIONALLY UPDATE B ARGUMENT EXPONENT
        AR: R4B=R4A-[+200]-<C>          PC: UW
        BR: CBN(C) [.+2]

**:     CM: NORMALIZE (SHIFT LEFT) / BRANCH IF TRUNCATING RESULT
        AR: R14B=R14B,SRU(MSBQ),SQ(0)   PC: 4B
        BR: CBN(/N) [.+2]

**:     CM: DO ROUNDING IF NEEDED
        AR: Q                   PC: 4B
        CC: C=PN,N=N,V=V,Z=Z    BR: CJN(N) [NORM.24A]

**:     CM: COMPUTE SIGN OF RESULT / CLEAR C
        AR: R5B=R5B XOR R3A     PC: UW
        CC: C=0,Z=Z,N=N,V=V

**:     CM: COMPUTE EXPONENT
        AR: R4B=R4B+R2A+<0>     PC: UW
        BR: CRN(1)

        PG:     24-BIT DIVISION ROUTINE

CM:     DIVIDEND IN R12 (4B)
        DIVISOR IN R14 (4B)
        RESULT LEFT IN R14 (4B) NORMALIZED
        REMAINDER LEFT IN R12
        EXPONENT RESULT IN R4 (UW)
        SIGN RESULT IN R5 (UW)
        RESULT IS ROUNDED(N=1)/TRUNCATED(N=0)
        V AND Z NOT EFFECTED, C USED

**: DIV.24
        CM: PRESET LOOP COUNTER
        AR: CNTR=[+32]

**:     CM: CHECK DIVISOR > DIVIDEND
        AR: R14B-R12A-<0>       PC: 4B
        CC: C=PN,N=N,Z=Z,V=V

**:     CM: CONDITIONAL EXPONENT UPDATE
        AR: R4B=R4B-R2A-</C>    PC: UW
        CC: C=0,N=N,Z=Z,V=V     BR: CBN(/C) [.+2]

**:     CM: SHIFT DIVISOR
        AR: R14B=R14B,SRU(0)    PC: 4B

**:     CM: EXECUTE 26. CYCLES
        IF C=0  A SUBTRACTION IS DONE
        IF C=1  AN ADDITION IS DONE
        ONE EXTRA BIT IS COMPUTED
        RESULT = /Q

        AR: R12B=R12B+R14A+</C>,SRU(0),SQ(MSBR)
        PC: 4B,MNR,CCL          CC: C=PN,N=N,Z=Z,V=V
        BR: CNR(/CNTR)

**:     CM: GET RESULT / ROUND IF NEEDED
        AR: R14B=/Q,SRD(0)      PC: 4B
        CC: C=LSBR,N=N,Z=Z,V=V  BR: CJN(N) [NORM.24A]

**:     CM: MASK JUNK
        AR: R14B=R14A AND [+377]        PC: UW

**:     CM: COMPUTE SIGN OF RESULT / CLEAR C
        AR: R5B=R5B XOR R3A     PC: UW
        CC: C=0,N=N,Z=Z,V=V
```

```
**:     CM: COMPUTE EXPONENT
        AR: R4B=[+200]-R4A-<1>  PC: UW
        BR: CRN(1)

        PG:     24-BIT OPERAND ALIGNMENT

CM:     ARGUMENTS PRESHIFTED ONE BIT (*2)
        C IS USED
        N,V, AND Z NOT USED

**: AL.24
        CM: PRESHIFT FRACTIONS
        AR: R12B=R12B,SRU(0)    PC: 4B

**:     AR: R14B=R14B,SRU(0)    PC: 4B

**:     CM: COMPUTE EXPONENT DIFFERENCE
        AR: D=R4B-R2A-<1>       PC: UW

**:     CM: RETURN IF ALIGNED
        AR: CNTR;R11B=D         PC: UW
        CC: C=PN,N=N,Z=Z,V=V    BR: CRN(Z')

**:     CM: BRANCH IF A > B  (IE. CNTR < 0)
        CM: IF C=0 THEN R11A-[+20]-<1> IS COMPUTED
        CM: IF C=1 THEN R11A+[+20]+<0> IS COMPUTED
        AR: R11B=R11A+[+20]+</C>        PC: UW,MNR
        BR: CBN(C) [AL.24B]

**: AL.24A
        CM: BRANCH IF 16. OR MORE SHIFTS NEEDED
        AR: R2B=R4A             PC: UW
        CC: C=0,N=N,Z=Z,V=V     BR: CBN(/N') [.+2]

**:     CM: DO SHIFTS
        AR: R12B=R12B,SRD(0)    PC: 4B,CCL
        BR: CRR(/CNTR)

**:     CM: CHECK IF MORE THAN 24. SHIFTS NEEDED
        AR: R11A-[+10]-<0>      PC: UW

**:     CM: BRANCH IF MORE
        AR: CNTR=R11A           PC: UW
        CC: C=PZ,N=N,Z=Z,V=V    BR: CBN(/N') [.+4]

**:     CM: DO 16. BIT SHIFT
        AR: R12B=0,D=R12A       PC: UW

**:     CM: BRANCH IF MORE TO DO
        AR: R12B=D              PC: LW
        BR: CBN(/C) [AL.24A+1]

**:     CM: ELSE FINISHED
        BR: CRR(1)

**:     CM: CLEAR WORD
        AR: R12B=0              PC: 4B
        BR: CRR(1)

**: AL.24B
        CM: BRANCH IF 16. OR MORE SHIFTS NEEDED
        AR: R4B=R2A     PC: UW
        BR: CBN(BLE') [.+2]

**:     CM: DO SHIFTS
        AR: R14B=R14B,SRD(0)    PC: 4B,CCL
        BR: CRR(/CNTR)

**:     CM: CHECK IF MORE THAN 24. SHIFTS NEEDED
        AR: R11A+[+10]+<0>      PC: UW

**:     CM: BRANCH IF MORE
        AR: CNTR=R11A           PC: UW
        CC: C=PZ,N=N,Z=Z,V=V    BR: CBN(N') [.+4]
```

```
**:     CM: DO 16. BIT SHIFT
        AR: R14B=0,D=R14A       PC: UW

**:     CM: BRANCH IF MORE TO DO
        AR: R14B=D              PC: LW
        BR: CBN(/C) [AL.24B+1]

**:     CM: ELSE FINISHED
        BR: CRR(1)

**:     CM: CLEAR WORD
        AR: R14B=0              PC: 4B
        BR: CRR(1)

        PG:     24-BIT POST NORMALIZATION

CM:     EXPECTS ARGUMENTS PRESHIFTED (*2)
        SET N=1 FOR ROUNDING
        OR N=0 FOR TRUNCATING
        C,Z=1 IF RESULT IS ZERO
        V NOT USED

**: NORM.24
        CM: CLEAR SHIFTING REGISTER
        AR: Q=0         PC: 4B

**:     CM: PRESHIFT / CHECK FOR ZERO
        AR: R14B=R14B,SRD(0),SQ(LSBR)           PC: 4B
        CC: C=PZ,Z=PZ,V=V,N=N

**:     CM: PRESET SHIFT COUNTER / RETURN ON ZERO
        AR: CNTR=[+3]   BR: CRN(Z)

**:     CM: PRESHIFT
        AR: R14B=R14B,SRD(0),SQ(LSBR)           PC: 4B
        CC: Z=0,C=0,N=N,V=V

**:     CM: GET Q REGISTER BITS INTO POSITION
        AR: D=Q         PC: UW

**:     AR: Q=DSWB      PC: UW

**:     CM: SHIFT UNTIL A '1' BIT FOUND
        AR: R14B=R14B,SRU(MSBQ),SQ(MSBR)        PC: 3B,CCL
        CC: C=MSBR,N=N,V=V,Z=Z                  BR: CNR(C)

**:     CM: TWO RESTORING SHIFTS
        AR: R14B=R14B,SRD(LSBQ),SQ(LSBR)        PC: 3B

**:     AR: R14B=R14B,SRD(LSBQ),SQ(LSBR)        PC: 3B
        CC: C=LSBR,N=N,Z=Z,V=V

**:     CM: ADJUST EXPONENT
            BRANCH IF ROUNDING
        AR: D=CNTR      BR: CBN(N) [.+2]

**:     CM: COMPUTE EXPONENT AND RETURN
        AR: R4B=R4A+DSX7+<0>            PC: UW
        CC: C=0,Z=0,N=N,V=V             BR: CRN(1)

**:     CM: COMPUTE EXPONENT
        AR: R4B=R4A+DSX7+<0>    PC: UW
        CC: C=C,Z=0,N=N,V=V     BR: CRN(/C)

**: NORM.24A
        CM: AND ROUND RESULT
        AR: R14B=R14B+<C>       PC: 3B
        CC: C=PC,Z=0,N=N,V=V    BR: CRN(/C)

**:     CM: CONDITIONAL UPDATE B EXPONENT
        AR: R4B=R4B+<C>         PC: UW
        CC: C=0,Z=0,N=N,V=V     BR: CRN(/C)

**:     CM: NORMALIZE RESULT
        AR: R14B=R14B,SRD(1)    PC: 3B
        BR: CRN(1)
```

```
        PG:     56-BIT ADDITION

CM:     ADDITION PERFORMED BETWEEN REGISTERS
        R12/R13/R2/R3 AND R14/R15/R4/R5
        FRACTION RESULT IN R14/R15 (8B)
        EXPONENT RESULT IN R4 (UW)
        SIGN RESULT IN R5 (UW)
        RESULT IS ROUNDED(N=1)/TRUNCATED(N=0)
        V NOT USED
        C=1 IF RESULT IS ZERO
        Z=1 IF ARGUMENT A OR B OR RESULT = 0

**: ADD.56
        CM: IS A ARGUMENT = 0 ?
        AR: R2A         CC: Z=PZ,C=0,N=N,V=V    PC: UW

**:     CM: IF A ARGUMENT IS ZERO - B IS RESULT
            AND IF B=0, THEN A CLEAN ZERO WILL BE GENERATED
        AR: R4B         CC: C=PZ,N=N,Z=Z,V=V    PC: UW
        BR: CRN(Z)

**:     CM: BRANCH IF B ARGUMENT IS NOT ZERO
            CLEAR NORMALIZE SHIFT REGISTER (Q)
        AR: Q=0         PC: 4B
        BR: CBN(/C) [.+2]

**: COPY.56
        CM: ELSE A IS THE RESULT
        AR: R15B=R13A   PC: 4B
        BR: CBN(1) [COPY.24]

**:     CM: ALIGN EXPONENTS
        AR: R13B=R13B,SRU(0)    PC: 4B
        CC: C=MSBR,N=N,Z=Z,V=V  BR: CJN(1) [AL.56+1]

**:     CM: CHECK SIGNS OF ARGUMENTS / DO ADDITION
        AR: R5B XOR R3A         PC: UW
        BR: CJN(1) [ADD.56A]

**:     CM: POST NORMALIZE / NORM.56 RETURNS TO CALLER
        AR: R14B=R14B           PC: 4B
        CC: Z=PZ,C=C,N=N,V=V    BR: CBN(1) [NORM.56+2]

**: ADD.56A
        CM: BRANCH IF SIGNS DIFFER
        AR: R5B                 PC: UW
        BR: CBN(N') [.+3]

**:     CM: ELSE COMPUTE SUM
        AR: R15B=R15B+R13A+<0>  PC: 4B

**:     AR: R14B=R14B+R12A+<C'> PC: 4B
        BR: CRN(1)

**:     CM: OPPOSITE SIGNS - DIDDLE A LITTLE
        BR: CBN(N') [.+3]

**:     CM: NEGATE A
        AR: R13B=-R13B-<1>      PC: 4B

**:     AR: R12B=-R12B-<C'>     PC: 4B
        BR: CBN(1) [.+3]

**:     CM: NEGATE B
        AR: R15B=-R15B-<1>      PC: 4B

**:     AR: R14B=-R14B-<C'>     PC: 4B

**:     CM: COMPUTE RESULT
        AR: R15B=R15B+R13A+<0>  PC: 4B

**:     AR: R14B=R14B+R12A+<C'> PC: 4B
        CC: C=PN,N=N,Z=Z,V=V

**:     CM: SAVE SIGN OF RESULT
```

```
        AR: R5B=0,SRD(C)        PC: UW
        BR: CRN(/C)

**:     CM: GET MAGNITUDE
        AR: R15B=-R15B-<1>      PC: 4B

**:     AR: R14B=-R14B-<C'>     PC: 4B
        BR: CRN(1)

        PG:     56-BIT MULTIPLICATION

CM:     MULTIPLICAND IN R12/R13 (8B)
        MULTIPLIER IN R14/R15 (8B)
        FRACTION RESULT IN R14/R15 (8B)
        32-BIT TRUNCATION LEFT IN Q
        EXPONENT RESULT IN R4 (UW)
        SIGN RESULT IN R5 (UW)
        R11 (4B) USED AS A TEMPORARY
        RESULT ROUNDED(N=1)/TRUNCATED(N=0)
        C USED
        N,Z,V NOT USED

**: MUL.56
        CM: SET UP FOR FIRST 32-BIT MULTIPLICATION
        AR: CNTR=[+40]

**:     CM: SAVE HIGH ORDER MULTIPLIER
        AR: R11B=R14A           PC: 4B

**:     CM: INITIALIZE RESULT
        AR: R14B=0              PC: 4B

**:     CM: PRESHIFT MULTIPLIER
        AR: R15B=R15B,SRD(0)    PC: 4B
        CC: C=LSBR,N=N,V=V,Z=Z

**:     CM: LOAD LOW ORDER MULTIPLIER / SKIP IF ZERO
        AR: Q=R15B              PC: 4B
        BR: CBN(Z') [.+2]

**:     CM: AND GO DO 32 BITS OF MULTIPLY
        AR: R15B=0              PC: 4B
        BR: CJN(1) [MUL.56A]

**:     CM: SET UP FOR LAST 24 BITS OF MULTIPLY
        AR: CNTR=[+30]

**:     CM: PRESHIFT MULTIPLIER
        AR: R11B=R11B,SRD(0)    PC: 4B
        CC: C=LSBR,N=N,Z=Z,V=V

**:     CM: DO LAST 24 BITS OF MULTIPLY
        AR: Q=R11B              PC: 4B
        BR: CJN(1) [MUL.56A]

**:     CM: CHECK MSB OF RESULT
        AR: R14B                PC: 3B
        CC: C=PN,N=N,Z=Z,V=V

**:     CM: CONDITIONALLY UPDATE EXPONENT AND FRACTION
        AR: R4B=R4A-[+200]-<C>  PC: UW
        BR: CBN(C) [.+3]

**:     AR: R15B=R15B,SRU(MSBQ),SQ(0)
        CC: C=MSBR,N=N,Z=Z,V=V  PC: 4B

**:     CM: BRANCH IF TRUNCATING
        AR: R14B=R14B,SRU(C)    PC: 4B
        BR: CBN(/N) [.+2]

**:     CM: DO ROUNDING IF REQUIRED
        AR: Q                   PC: 4B
        CC: C=PN,N=N,Z=Z,V=V    BR: CJN(N) [NORM.56A]

**:     CM: COMPUTE SIGN OF RESULT
        AR: R5B=R5B XOR R3A     PC: UW
```

```
        CC: C=0,N=N,Z=Z,V=V

**:     CM: COMPUTE EXPONENT / FINISHED
        AR: R4B=R4B+R2A+<0>     PC: UW
        BR: CRN(1)

**:     CM: RETURN IF CNTR = 0
        AR: R15B=R15B,SRD(C),SQ(LSBR)   PC: 4B,CCL
        CC: C=LSBQ,N=N,Z=Z,V=V  BR: CRN(/CNTR)

**: MUL.56A
        CM: LOW 32 BIT SUM
        AR: R15B=R15B+R13A+<0>  PC: 4B,MM

**:     CM: HIGH 32 BIT SUM
        AR: R14B=R14B+R12A+<C'>,SRD(0)  PC: 4B,MM
        CC: C=LSBR,N=N,Z=Z,V=V  BR: CBN(1) [.-2]

        PG:     56-BIT DIVISION ROUTINE

CM:     DIVIDEND IN R12/R13 (8B)
        DIVISOR IN R14/R15 (8B)
        FRACTION RESULT IN R14/R15 (8B)
        EXPONENT RESULT IN R4 (UW)
        SIGN RESULT IN R5 (UW)
        RESULT ROUNDED(N=1)/TRUNCATED(N=0)
        Z NOT USED
        C USED
        V SET = 1

**: DIV.56
        CM: PRESET COUNTER
        AR: CNTR=[+32]

**:     CM: CHECK DIVISOR > DIVIDEND
        AR: R15B-R13A-<0>               PC: 4B

**:     AR: R14B-R12A-<C'>              PC: 4B
        CC: C=PN,N=N,Z=Z,V=V

**:     CM: CONDITIONAL EXPONENT UPDATE
        AR: R4B=R4A-[+200]-</C>         PC: UW
        BR: CBN(/C) [.+3]

**:     CM: SHIFT DIVISOR
        AR: R15B=R15B,SRU(0)            PC: 4B
        CC: C=MSBR,N=N,Z=Z,V=V

**:     AR: R14B=R14B,SRU(C)            PC: 4B

**:     CM: EXECUTE 26. CYCLES
        AR: R13B=R13B-R15A-<1>,SRU(0)   PC: 4B
        CC: C=0,V=PN,N=N,Z=Z    BR: CJN(1) [DIV.56A+1]

**:     CM: SAVE BITS COMPUTED
        AR: R11B=/Q             PC: 4B

**:     CM: DO REMAINING 32. BITS
        AR: CNTR=[+40]          BR: CJN(1) [DIV.56A]

**:     CM: DO A RIGHT SHIFT TO GET RESULT AND ROUNDING BIT
        AR: R14B=R11A,SRD(0)            PC: 4B
        CC: C=LSBR,N=N,Z=Z,V=V

**:     AR: R15B=/Q,SRD(C)              PC: 4B
        CC: C=LSBR,N=N,Z=Z,V=V

**:     CM: MASK JUNK / ROUND IF REQUIRED
        AR: R14B=R14A AND [+377]        PC: UW
        BR: CJN(N) [NORM.56A]

**:     CM: COMPUTE SIGN OF RESULT
        AR: R5B=R5B XOR R3A             PC: UW
        CC: C=0,N=N,Z=Z,V=V

**:     CM: COMPUTE EXPONENT / FINISHED
```

```
        AR: R4B=R2A-R4B-<1>             PC: UW
        BR: CRN(1)

**:     CM: RETURN IF CNTR=0
        AR: R12B=R12B,SRU(C),SQ(MSBR)   PC: 4B,CCL
        CC: C=MSBR,N=N,Z=Z,V=V          BR: CRN(/CNTR)

**: DIV.56A
        CM: LOW 32. BIT OPERATION
        AR: R13B=R13B+R15A+</C>,SRU(0)  PC: 4B,MNR
        CC: C=C,V=PN,N=N,Z=Z

**:     CM: HIGH 32. BIT OPERATION
        AR: R12B=R12B+R14A+<C'>         PC: 4B,MNR
        CC: C=V,V=1,N=N,Z=Z             BR: CBN(1) [.-2]

        PG:     56-BIT OPERAND ALIGNMENT

CM:     ARGUMENTS PRESHIFTED ONE BIT (*2)
        R11 (UW) USED AS TEMPORARY
        C IS USED
        N,V,Z NOT USED

**: AL.56
        CM: PRESHIFT FRACTIONS
        AR: R13B=R13B,SRU(0)    PC: 4B
        CC: C=MSBR,N=N,Z=Z,V=V

**:     AR: R12B=R12B,SRU(C)    PC: 4B

**:     AR: R15B=R15B,SRU(0)    PC: 4B
        CC: C=MSBR,N=N,Z=Z,V=V

**:     AR: R14B=R14B,SRU(C)    PC: 4B

**:     CM: COMPUTE EXPONENT DIFFERENCE
        AR: D=R4B-R2A-<1>       PC: UW

**:     CM: RETURN IF ALIGNED
        AR: CNTR;R11B=D         PC: UW
        CC: C=PN,N=N,Z=Z,V=V    BR: CRN(Z')

**:     CM: BRANCH IF A>B (IE. CNTR < 0)
        CM: IF C=0 THEN R11A-[+20]-<1> IS COMPUTED
        CM: IF C=1 THEN R11A+[+20]+<0> IS COMPUTED
        AR: R11B=R11A+[+20]+</C>        PC: UW,MNR
        BR: CBN(C) [AL.56B]

**: AL.56A
        CM: BRANCH IF 16. OR MORE SHIFTS REQUIRED
        AR: R2B=R4A             PC: UW
        BR: CBN(/N') [.+4]

**:     CM: ELSE DO SHIFTS
        AR: R12B=R12B,SRD(0)    PC: 4B
        CC: C=LSBR,N=N,Z=Z,V=V

**:     CM: RETURN IF FINISHED
        AR: R13B=R13B,SRD(C)    PC: 4B,CCL
        BR: CRN(/CNTR)

**:     AR: R12B=R12B,SRD(0)    PC: 4B
        CC: C=LSBR,N=N,Z=Z,V=V  BR: CBN(1) [.-1]

**:     CM: CHECK IF 32. OR MORE SHIFTS REQUIRED
        AR: R11B=R11A-[+20]-<1>,D=R11A  PC: UW

**:     CM: SET COUNTER / BRANCH IF MORE
        AR: CNTR=D      CC: C=PZ,N=N,Z=Z,V=V
        BR: CBN(/N') [.+6]

**:     CM: ELSE DO 16. BIT SHIFT
        AR: R12B=0,D=R12A       PC: UW

**:     AR: R12B=D,D=R12A       PC: LW
```

```
**:     AR: R13B=D,D=R13A       PC: UW

**:     AR: R13B=D              PC: LW
        BR: CBN(/C) [AL.56A+1]

**:     BR: CRR(1)

**:     CM: CHECK IF MORE THAN 56. SHIFTS REQUIRED
        AR: R11A-[+30]-<0>      PC: UW

**:     CM: BRANCH IF MORE
        AR: CNTR=R11A           PC: UW
        CC: C=PZ,N=N,Z=Z,V=V    BR: CBN(/N') [.+2]

**:     CM: ELSE DO 32. BIT SHIFT
        AR: R13B=R12A           PC: 4B
        BR: CBN(1) [.+2]

**:     CM: ZERO IF MORE
        AR: R13B=0              PC: 4B
        CC: C=1,N=N,Z=Z,V=V

**:     CM: CLEAR UPPER 32. BITS
        AR: R12B=0              PC: 4B
        CC: C=0,N=N,Z=Z,V=V     BR: CRN(C)

**:     CM: CHECK IF 16. OR MORE SHIFTS LEFT
        AR: R11B=R11A-[+20]-<1> PC: UW
        BR: CBN(1) [AL.56A]

**: AL.56B
        CM: BRANCH IF 16. OR MORE SHIFTS NEEDED
        AR: R4B=R2A             PC: UW
        BR: CBN(BLE') [.+4]

**:     CM: ELSE DO SHIFTS
        AR: R14B=R14B,SRD(0)    PC: 4B
        CC: C=LSBR,N=N,Z=Z,V=V

**:     CM: RETURN IF FINISHED
        AR: R15B=R15B,SRD(C)    PC: 4B,CCL
        BR: CRN(/CNTR)

**:     AR: R14B=R14B,SRD(0)    PC: 4B
        CC: C=LSBR,N=N,Z=Z,V=V  BR: CBN(1) [.-1]

**:     CM: CHECK IF 32. OR MORE SHIFTS NEEDED
        AR: R11B=R11A+[+20]+<0>,D=R11A  PC: UW

**:     CM: SET COUNTER / BRANCH IF 32. OR MORE
        AR: CNTR=D
        CC: C=PZ,N=N,Z=Z,V=V    BR: CBN(BLE') [.+6]

**:     CM: ELSE DO 16. BIT SHIFT
        AR: R14B=0,D=R14A       PC: UW

**:     AR: R14B=D,D=R14A       PC: LW

**:     AR: R15B=D,D=R15A       PC: UW

**:     AR: R15B=D              PC: LW
        BR: CBN(/C) [AL.56B+1]

**:     BR: CRR(1)

**:     CM: CHECK IF MORE THAN 56. SHIFTS NEEDED
        AR: R11A+[+30]+<0>      PC: UW

**:     CM: BRANCH IF MORE
        AR: CNTR=R11A           PC: UW
        CC: C=PZ,N=N,Z=Z,V=V    BR: CBN(N') [.+2]

**:     CM: ELSE DO 32. BIT SHIFT
        AR: R15B=R14A           PC: 4B
        BR: CBN(1) [.+2]
```

```
**:     CM: CLEAR ARGUMENT
        AR: R15B=0              PC: 4B
        CC: C=1,N=N,Z=Z,V=V

**:     AR: R14B=0              PC: 4B
        CC: C=0,N=N,Z=Z,V=V     BR: CRN(C)

**:     CM: CHECK IF 16. OR MORE SHIFTS NEEDED
        AR: R11B=R11A+[+20]+<0> PC: UW
        BR: CBN(1) [AL.56B]

        PG:     56-BIT POST NORMALIZATION

CM:     EXPECTS ARGUMENTS PRESHIFTED (*2)
        V NOT USED
        C,Z = 1 IF RESULT = 0
        RESULT IS ROUNDED(N=1)/TRUNCATED(N=0)

**: NORM.56
        CM: CLEAR SHIFTING REGISTER
        AR: Q=0                 PC: 4B

**:     CM: CHECK FOR ZERO RESULT
        AR: R14B=R14B           PC: 4B
        CC: Z=PZ,C=C,N=N,V=V

**:     CM: BRANCH IF HIGH ORDER NOT ZERO
        AR: R15B=R15B           PC: 4B
        CC: Z=PZAZ,C=C,N=N,V=V  BR: CBN(/Z) [.+12]

**:     CM: CHECK LOWEST ORDER
        AR: Q                   PC: 4B
        CC: Z=PZAZ,C=C,N=N,V=V

**:     CM: RETURN IF RESULT = 0
        AR: D=Q                 PC: UW
        CC: C=1,N=N,Z=Z,V=V     BR: CRN(Z)

**:     CM: ELSE DO A 16. BIT SHIFT
        AR: R15B=D,D=R15A       PC: LW

**:     AR: R15B=D,D=R15A       PC: UW

**:     AR: R14B=D              PC: LW
        CC: Z=PZ,C=0,N=N,V=V

**:     CM: MOVE Q REGISTER BITS
        AR: D=Q                 PC: LW

**:     AR: Q=D                 PC: UW
        BR: CBN(Z') [.+2]

**:     AR: Q=0                 PC: LW

**:     CM: UPDATE EXPONENT
        AR: R4B=R4A-[+20]-<1>   PC: UW
        BR: CBN(1) [NORM.56+2]

**:     CM: PRESET COUNTER
        AR: CNTR=[+2]

**:     CM: PRESIFT TWICE
        AR: R14B=R14B,SRD(0)    PC: 4B
        CC: C=LSBR,N=N,Z=Z,V=V

**:     AR: R15B=R15B,SRD(C),SQ(LSBR)   PC: 4B
        CC: Z=1,C=0,N=N,V=V     BR: CBN(/Z) [.-1]

**:     CM: SHIFT UNTIL A '1' BIT FOUND
        AR: R15B=R15B,SRU(MSBQ),SQ(0)   PC: 4B
        CC: C=MSBR,N=N,Z=Z,V=V  BR: CBN(C) [.+2]

**:     AR: R14B=R14B,SRU(C)    PC: 3B,CCL
        CC: C=MSBR,N=N,Z=Z,V=V  BR: CBN(1) [.-1]

**:     CM: RESTORING SHIFTS
```

```
        AR: R15B=R15B,SRD(C),SQ(LSBR)   PC: 4B

**:     AR: R14B=R14B,SRD(1)    PC: 3B
        CC: C=LSBR,N=N,Z=Z,V=V

**:     AR: R15B=R15B,SRD(C)    PC: 4B
        CC: C=LSBR,N=N,Z=Z,V=V

**:     CM: ADJUST EXPONENT / BRANCH IF ROUNDING
        AR: D=CNTR              BR: CBN(N) [.+2]

**:     CM: COMPUTE EXPONENT AND RETURN
        AR: R4B=R4A+DSX7+<0>    PC: UW
        CC: C=0,Z=0,N=N,V=V     BR: CRN(1)

**:     CM: COMPUTE EXPONENT / RETURN IF ROUNDING BIT = 0
        AR: R4B=R4A+DSX7+<0>    PC: UW
        CC: C=C,Z=0,N=N,V=V     BR: CRN(/C)

**: NORM.56A
        CM: AND ROUND RESULT
        AR: R15B=R15B+<C>       PC: 4B
        CC: C=PC,Z=0,N=N,V=V    BR: CRN(/C)

**:     AR: R14B=R14B+<C>       PC: 3B
        CC: C=PC,Z=0,N=N,V=V    BR: CRN(/C)

**:     CM: CONDITIONALLY UPDATE EXPONENT
        AR: R4B=R4B+<C>         PC: UW
        BR: CRN(/C)

**:     CM: SHIFT FRACTION
        AR: R14B=R14B,SRD(C)    PC: 3B
        CC: C=LSBR,Z=0,N=N,V=V

**:     AR: R15B=R15B,SRD(C)    PC: 4B
        CC: C=0,Z=0,N=N,V=V     BR: CRN(1)

        NA: MORGEND=.
```

```
        PG:     FLOATING POINT PROCESSOR OVERLAY

        CM:       THE FOLLOWING INSTRUCTION CODE
                DEFINITIONS OVERLAY THE ARB11 CODE TO ADD THE FOLLOWING
                FLOATING POINT INSTRUCTIONS -

                PDP-11/34,45,60,70 AND LSI-11/23 COMPATIBLE

                        CFCC    170000
                        SETF    170001
                        SETI    170002
                        SETD    170011
                        SETI    170012
                        LDFPS   1701SRC
                        STFPS   1702DST
                        STST    1703DST
                        CLRF/D  1704FDST
                        TSTF/D  1705FDST
                        ABSF/D  1706FDST
                        NEGF/D  1707FDST
                        MULF/D  171(AC)FSRC
                        MODF/D  171(AC+4)FSRC
                        ADDF/D  172(AC)FSRC
                        LDF/D   172(AC+4)FSRC
                        SUBF/D  173(AC)FSRC
                        CMPF/D  173(AC+4)FSRC
                        STF/D   174(AC)FDST
                        DIVF/D  174(AC+4)FSRC
                        STEXP   175(AC)DST
                        STCFI/L
                        STCDI/L 175(AC+4)DST
                        STCFD
                        STCDF   176(AC)FDST
                        LDEXP   176(AC+4)SRC
                        LDCIF/D
                        LDCLF/D 177(AC)SRC
                        LDCDF/D 177(AC+4)FSRC

        PG:     FPP CODING NOTES

CM:     NOTES ON CODING -

        1) CONDITION CODES ARE CHANGED DURING OPERAND
           ACCESS. MEMORY TRAPS WILL HAVE INVALIDS CC'S

        2) FIS INSTRUCTION SET MUST BE PRESENT TO ALLOW
           ENTRY TO THE FOLLOWING ROUTINES
                A.GETARG
                B.GETARG
                ARG.SAV
                CMB.SEF
                ADD.24
                MUL.24
                DIV.24
                NORM.24
                ADD.56
                MUL.56
                DIV.56
                NORM.56
                FIS.TRAP

        3) THE FOLLOWING ROUTINES MUST BE
           PRESENT FOR RESERVED INSTRUCTION PROCESSING
                RSRVINST
                RSRVSTAT

        4) THE ACCUMULATORS AND CERTAIN STATUS REGISTERS
           ARE KEPT IN MAIN MEMORY. THE LOCATIONS ARE DEFINED HERE

                NA: AC0.FPP=+167000
                NA: AC1.FPP=AC0.FPP+10
                NA: AC2.FPP=AC0.FPP+20
                NA: AC3.FPP=AC0.FPP+30
                NA: AC4.FPP=AC0.FPP+40
                NA: AC5.FPP=AC0.FPP+50
                NA: AC6.FPP=AC0.FPP+60
```

```
                NA: AC7.FPP=AC0.FPP+70

                NA: FEA.FPP=AC0.FPP+100
                NA: FEC.FPP=AC0.FPP+102

        PG:     FPP REGISTER USAGE

CM:     THE FPP MAKES USE OF THE FOLOWING REGISTERS

        THE A ARGUMENT -
                R2 (UW) - EXPONENT
                R3 (UW) - SIGN OF FRACTION
                R12 (4B) - HIGH FRACTION
                R13 (4B) - LOW FRACTION

        THE B ARGUMENT -
                R4 (UW) - EXPONENT
                R5 (UW) - SIGN OF FRACTION
                R14 (4B) - HIGH FRACTION
                R15 (4B) - LOW FRACTION

        FPP STATUS
                R16 (UW) - FLOATING POINT STATUS (WITHOUT FCC'S)
                R6 (UW) - FLOATING POINT CC'S

        TEMPORARYS -
                R0 (UW) - CONDITION CODE SAVE
                R1 (UW) - SCRATCH
                R10 (LW) - ADDRESS CALCULATIONS
                R10 (UW) - SCRATCH
                R11 (4B) - SCRATCH
                Q (4B) - SCRATCH
                SCE5 (SCR2) - INDEX BRANCH REGISTER
                SCR1 (SCE6) - INSTRUCTION POINTER

        PG:     FLOATING INSTRUCTION CODE SPECIFICATIONS

        .=: IORIGIN+0

**:     MA: SET.FPS     PS:  EX

**:     MA: LDFPS       PS: DST,EX      IN: SP

**:     MA: STFPS       PS: DST,EX      IN: NDA

**:     MA: STST        PS: EX

**:     MA: CLRF        PS: EX

**:     MA: TSTF        PS: EX

**:     MA: ABSF        PS: EX

**:     MA: NEGF        PS: EX

**:     MA: MULF        PS: EX

**:     MA: MULF+1      PS: EX

**:     MA: MULF+2      PS: EX

**:     MA: MULF+3      PS: EX

**:     MA: MODF        PS: EX

**:     MA: MODF+1      PS: EX

**:     MA: MODF+2      PS: EX

**:     MA: MODF+3      PS: EX

**:     MA: ADDF        PS: EX

**:     MA: ADDF+1      PS: EX

**:     MA: ADDF+2      PS: EX
```

```
**:     MA: ADDF+3      PS: EX
**:     MA: LDF         PS: EX
**:     MA: LDF+1       PS: EX
**:     MA: LDF+2       PS: EX
**:     MA: LDF+3       PS: EX
**:     MA: SUBF        PS: EX
**:     MA: SUBF+1      PS: EX
**:     MA: SUBF+2      PS: EX
**:     MA: SUBF+3      PS: EX
**:     MA: CMPF        PS: EX
**:     MA: CMPF+1      PS: EX
**:     MA: CMPF+2      PS: EX
**:     MA: CMPF+3      PS: EX
**:     MA: STF         PS: EX
**:     MA: STF+1       PS: EX
**:     MA: STF+2       PS: EX
**:     MA: STF+3       PS: EX
**:     MA: DIVF        PS: EX
**:     MA: DIVF+1      PS: EX
**:     MA: DIVF+2      PS: EX
**:     MA: DIVF+3      PS: EX
**:     MA: STEXP       PS: DST,EX      IN: NDA
**:     MA: STEXP+1     PS: DST,EX      IN: NDA
**:     MA: STEXP+2     PS: DST,EX      IN: NDA
**:     MA: STEXP+3     PS: DST,EX      IN: NDA
**:     MA: STCFI       PS: EX
**:     MA: STCFI+1     PS: EX
**:     MA: STCFI+2     PS: EX
**:     MA: STCFI+3     PS: EX
**:     MA: STCFD       PS: EX
**:     MA: STCFD+1     PS: EX
**:     MA: STCFD+2     PS: EX
**:     MA: STCFD+3     PS: EX
**:     MA: LDEXP       PS: DST,EX      IN: SP
**:     MA: LDEXP+1     PS: DST,EX      IN: SP
**:     MA: LDEXP+2     PS: DST,EX      IN: SP
**:     MA: LDEXP+3     PS: DST,EX      IN: SP
**:     MA: LDCIF       PS: EX
```

```
**:     MA: LDCIF+1     PS: EX

**:     MA: LDCIF+2     PS: EX

**:     MA: LDCIF+3     PS: EX

**:     MA: LDCDF       PS: EX

**:     MA: LDCDF+1     PS: EX

**:     MA: LDCDF+2     PS: EX

**:     MA: LDCDF+3     PS: EX

        PG:     GET ACCUMULATOR INTO A

        .=: MORGEND

CM:     THIS CODE GETS THE A ACCUMULATOR AT THE ADDRESS IN R10 (LW)
        SET V=0 FOR FLOATING / V=1 FOR DOUBLE
        AND EXITS WITH
        C,Z=1 IF EXPONENT = 0
        N=1 IF ACCUMULATOR < 0
        R2 (UW) = EXPONENT
        R3 (UW) = SIGN OF FRACTION
        R12 (4B) = FRACTION (24-BITS)
        R13 (4B) = FRACTION (32-BITS)

**: A.GETACC
        AR: AD=R10A             IO: DATI(OFF)

**:     CM: PRESET HIDDEN BIT OF FRACTION
        AR: R12B=0              PC: UW

**:     AR: AD=R10A+[+2]+<0>    PC: LW,WIOL

**:     CM: SAVE HIGH ORDER IN TEMPORARY
            GO BUILD SIGN, EXPONENT, AND FRACTION
        AR: R2B=D,SRU(0)        PC: UW
        CC: C=MSBR,N=PN,Z=0,V=V
        IO: DATI(OFF)           BR: CJN(1) [A.BLDSEF]

**:     CM: GET DOUBLE FRACTION
        AR: AD=R10A+[+4]+<0>    IO: DATI(OFF)

**:     AR: AD=R10A+[+6]+<0>    PC: LW,WIOL

**:     AR: R13B=D              PC: UW
        IO: DATI(OFF)

**:     CM: DUMBY LOAD AD
        AR: R13B=D,AD=R7A       PC: LW,WIOH
        BR: CRR(1)

        PG:     GET ACCUMULATOR INTO B

CM:     THIS CODE GETS THE B ACCUMULATOR AT THE ADDRESS IN R10 (LW)
        SET V=0 FOR FLOATING / V=1 FOR DOUBLE
        AND EXITS WITH
        C,Z=1 IF EXPONENT IS = 0
        N=1 IF ACCUMULATOR IS < 0
        R4 (UW) = EXPONENT
        R5 (UW) = SIGN OF FRACTION
        R14 (4B) = FRACTION (24-BITS)
        R15 (4B) = FRACTION (32-BITS)

**: B.GTAC
        CM: SET LENGTH FLAG
        AR: R16B                PC: UB
        CC: C=C,N=N,Z=Z,V=PN    BR: CBN(1) [.+2]

**: B.GETACC
        AR: AD=R10A             IO: DATI(OFF)

**:     CM: CLEAR HIGH FRACTION
```

```
        AR: R14B=0              PC: UW

**:     AR: AD=R10A+[+2]+<0>    PC: LW,WIOL

**:     CM: SAVE HIGH ORDER IN TEMPORARY
            GO BUILD SIGN, EXPONENT, AND HIGH FRACTION
        AR: R4B=D,SRU(0)        PC: UW
        CC: C=MSBR,N=PN,Z=0,V=V
        IO: DATI(OFF)           BR: CJN(1) [B.BLDSEF]

**:     CM: GET DOUBLE FRACTION
        AR: AD=R10A+[+4]+<0>    IO: DATI(OFF)

**:     AR: AD=R10A+[+6]+<0>    PC: LW,WIOL

**:     AR: R15B=D              PC: UW
        IO: DATI(OFF)

**:     CM: DUMBY LOAD AD
        AR: R15B=D,AD=R7A       PC: LW,WIOH
        BR: CRR(1)

        PG:     GET OPERAND UTILITY

CM:     SOURCE OPERAND IS LOADED AND CHECKED FOR -0
        IF NOT FROM AN ACCUMULATOR
        TRAP OCCURS TO FIUV.TRP IF -0 AND FIUV=1
        ACCUMULATOR OPERAND IS THEN LOADED

**: GETOPS
        CM: SAVE INSTRUCTION POINTER
        AR: SCR1=R7A-[+2]-<1>

**:     CM: GET SOURCE ARGUMENT
        AR: R10B=IR5,SRD(0)     PC: UW
        BR: CJN(1) [FSRC+1]

**:     CM: SETUP END BRANCH
        AR: SCE5=[.+3]          SP: CLR,SE5

**:     CM: CHECK FOR -0 IF NOT MODE 0
        AR: AD;R10B=R11A        BR: CJN(C) [FIUV.FPP+1]

**:     CM: TRAP IF -0 AND FIUV=1
        AR: SCE5=[BGN]          BR: CBIN(1) <SCR>

**:     CM: ELSE GET ACCUMULATOR
        AR: R1B=[+40]           PC: UW
        IO: IDATI(OFF)          BR: CJN(1) [A.GETACC+1]

**:     CM: SET TRUNCATION FLAG / RETURN TO CALLER
        AR: R16A AND R1B        PC: UW
        CC: C=0,Z=Z,N=PZ,V=V    SP: CLR,SE5
        BR: CRR(1)

        PG:     FPP SOURCE ADDRESSING MODES

CM:     SET C=1 FOR NO DATA ACCESS, ELSE SET C=0
        SET Z=1 FOR FLOATING (V=0) / DOUBLE (V=1)
        SET Z=0 FOR FPS LENGTH

**: FSRC.SAV
        CM: SAVE INSTRUCTION LOCATION
        AR: SCR1=R7A-[+2]-<1>

**: FSRC
        CM: USING DST PART OF INSTRUCTION FIND MODE
        AR: R10B=IR5,SRD(0)     PC: UW

**:     CM: MASK JUNK
        AR: R10B=R10A AND [+34] PC: UW
        BR: CJN(1) [FS.M4+3]

**:     CM: USE SPECIFIED MODE
        AR: R10B=IR5,SRU(0)
        CC: V=V,C=C,N=0,Z=0     BR: CBIN(Z) <SCR>
```

```
**:     CM: ELSE USE FPS LENGTH
        AR: R16B                PC: UB
        CC: C=C,V=PN,N=0,Z=0    BR: CBIC(1) <SCR>

**: FS.M0
        CM: ACCUMULATOR R IS THE OPERAND
        AR: R10B=R10B+R10A+<0>,SRU(0)

**:     CM: RETURN IF ONLY ADDRESS NEEDED
        AR: AD;R10B=R10A+[AC0.FPP]+<0>
        BR: CRN(C)

**:     CM: ELSE GET ACCUMULATOR
        AR: R14B=0              PC: UW
        IO: IDATI(OFF)          BR: CJN(1) [B.GETACC+2]

**:     CM: CLEAR C / NO FIUV FROM ACC
        CC: C=0,N=N,Z=Z,V=V     BR: CRR(1)

        .=: FS.M0+4

**: FS.M1
        CM: (R) IS ADDRESS
        AR: AD;R10B=RA(DST)     BR: CRN(C)

**:     AR: R14B=0              PC: UW
        IO: IDATI(CMI)          BR: CBN(1) [B.GETARG+2]

**: FDST
        CM: CHECK MODE / GO BUILD SEF
        AR: R1B=IR5             PC: UW
        BR: CJN(1) [CMB.SEF]

**:     AR: R1A AND [+70]       PC: UW
        BR: CBN(1) [FS.M7+1]

        .=: FS.M0+10

**: FS.M2
        CM: (R) IS ADDRESS; (R) + (4 OR 10) / IF DST=27, (R) + 2
        CM: MODE 2 / CHECK IMMEDIATE
        AR: R10A-[+56]-<1>

**:     CM: BRANCH IF IMMEDIATE
        AR: RB(DST)=RA(DST)+[+2]+<0>,D=RA(DST)
        BR: CBN(Z') [FS.IM]

**:     AR: RB(DST)=RA(DST)+[+2]+<0>
        BR: CBN(/V) [FS.M5+2]

**:     AR: RB(DST)=RA(DST)+[+4]+<0>
        BR: CBN(1) [FS.M5+2]

        .=: FS.M0+14

**: FS.M3
        CM: (R) IS ADDRESS OF ADDRESS; (R) + 2
        AR: RB(DST)=RA(DST)+[+2]+<0>,AD=RA(DST)
        IO: DATI(CMI)           BR: CBN(1) [FS.M5+2]

**: FS.IM
        CM: IMMEDIATE MODE
        AR: AD;R10B=D           BR: CRN(C)

**:     CM: PUSH DUMBY RETURN IF FLOATING
        AR: R15B=0              PC: 4B
        IO: IDATI(CMI)          BR: CJN(/V) [.+1]

**:     CM: BUILD SEF
        AR: R4B=D,SRU(0)        PC: UW,WIOH
        CC: C=MSBR,N=PN,Z=0,V=V BR: CBN(1) [FS.M4+2]

        .=: FS.M0+20

**: FS.M4
```

```
        CM: (R) - (4 OR 10); (R) IS ADDRESS
        AR: D;RB(DST)=RA(DST)-[+4]-<1>
        BR: CBN(/V) [FS.M5+1]

**:     AR: D;RB(DST)=RA(DST)-[+4]-<1>
        BR: CBN(1) [FS.M5+1]

**:     CM: (FS.IM) B.BLDSEF REMOVES DUMBY RETURN ON FLOATING
            RETURNS TO CALLER OF FSRC
        AR: D;R14B=0            PC: UW
        BR: CBN(1) [B.BLDSEF]

**:     CM: (FSRC) SET UP INDEXED JUMP
        AR: SCE5=R10A+[FS.M0]+<0>       PC: UW
        SP: CLR,SE5             BR: CRR(1)

        .=: FS.M0+24

**: FS.M5
        CM: (R)-2 ; (R) IS ADDRESS OF ADDRESS
        AR: AD;RB(DST)=RA(DST)-[+2]-<1>
        IO: DATI(CMI)

**:     CM: CHECK STACK OVERFLOW
        AR: SOR=R6A-SLR-<1>

**:     CM: RETURN IF ONLY ADDRESS NEEDED
        AR: AD;R10B=D           PC: LW,WIOH
        BR: CRN(C)

**:     AR: R14B=0              PC: UW
        IO: IDATI(CMI)          BR: CBN(1) [B.GETARG+2]

        .=: FS.M0+30

**: FS.M6
        CM: (R) + X IS ADDRESS
        AR: R7B=R7A+[+2]+<0>,AD=R7A
        IO: DATI(CMI)

**:     CM: FINISHED IF ONLY ADDRESS NEEDED
        AR: AD;R10B=RA(DST)+D+<0>       PC: LW,WIOH
        BR: CRN(C)

**:     AR: R14B=0              PC: UW
        IO: IDATI(CMI)          BR: CBN(1) [B.GETARG+2]

**:     CM: (MODE 7)
        AR: AD=RA(DST)+D+<0>    PC: LW,WIOH
        IO: DATI(CMI)           BR: CBN(1) [FS.M5+2]

        .=: FS.M0+34

**: FS.M7
        CM: (R) + X IS ADDRESS OF ADDRESS
        AR: R7B=R7A+[+2]+<0>,AD=R7A
        IO: DATI(CMI)           BR: CBN(1) [FS.M6+3]

        PG:     FLOATING DESTINATION CODE

CM:     STORES RESULT IN ACCUMULATOR OR DESTINATION
        AT ADDRESS IN R10 (LW)
        CONDITION CODES ARE CHANGED
        C=0, N=1 IF NEGATIVE, Z=1 IF EXP=0
        V IS UNCHANGED FLOATING(V=0)/DOUBLE(V=1)


**:     CM: BRANCH IF ACCUMULATOR
        AR: R1A-[+27]-<1>       PC: UW
        BR: CBN(Z') [ACC.SAVA]

**:     CM: BRANCH IF NOT IMMEDIATE
        AR: AD=R10A     BR: CBN(/Z') [ARG.SAVA]

**:     CM: ELSE IMMEDIATE MODE
        IO: IDATO(CM)
```

```
        CC: C=0,V=0,N=N,Z=Z     BR: CRN(1)

        PG:     INTEGER SOURCE

CM:     C=1 FOR NO DATA ACCESS
          RETURNS WITH
                Z=1 IF MODE ZERO
                N=0 IF V=0 OR IMMEDIATE
        ELSE SET C=0
        Z=1 THEN INTEGER(V=0) / DOUBLE(V=1)
        Z=0 THEN MODE IN FPS USED

**: IS.M0
        CM: MODE 0 / REGISTER R IS THE OPERAND
        AR: D=RA(DST)           PC: LW
        CC: N=N,Z=1,C=C,V=V     BR: CRN(C)

**:     CM: ELSE BUILD DATA
        AR: R15B=0              PC: 4B
        CC: N=0,C=C,Z=Z,V=V     BR: CBN(1) [B.GETINT]

**:     CM: (B.GETINT)
        IO: IDATI(CMI)

**:     AR: R15B=D              PC: LW,WIOH
        CC: Z=PZAZ,N=N,C=C,V=V  BR: CRR(1)

        .=: IS.M0+4

**: IS.M1
        CM: MODE 1 / (R) IS ADDRESS
        AR: AD;R10B=RA(DST)     BR: CRN(C)

**:     CM: ELSE GET INTEGER
        AR: R15B=0              PC: 4B
        IO: IDATI(CMI)          BR: CBN(1) [B.GETINT]

**: ISRC.SAV
        CM: SAVE INSTRUCTION PC
        AR: SCR1=R7A-[+2]-<1>

**: ISRC
        CM: SET UP INDEXED JUMP TO MODE
        AR: R10B=IR5,SRD(0)     PC: UW
        BR: CBN(1) [IS.M3+1]

        .=: IS.M0+10

**: IS.M2
        CM: MODE 2 / (R) IS ADDRESS; (R) + (2 OR 4) / IF DST=27, (R) + 2
        AR: RB(DST)=RA(DST)+[+2]+<0>,D=RA(DST)
        BR: CBN(/V) [IS.M5+2]

**:     AR: R10A-[+27]-<1>

**:     CM: BRANCH IF NOT IMMEDIATE
        AR: RB(DST)=RA(DST)+[+2]+<0>
        BR: CBN(/Z') [IS.M5+2]

**:     CM: ELSE RESTORE GENERAL REGISTER
        AR: RB(DST)=RA(DST)-[+2]-<1>
        BR: CBN(1) [IS.M4+3]

        .=: IS.M0+14

**: IS.M3
        CM: MODE 3 / (R) IS ADDRESS OF ADDRESS; (R) + 2
        AR: RB(DST)=RA(DST)+[+2]+<0>,AD=RA(DST)
        IO: DATI(CMI)           BR: CBN(1) [IS.M5+2]

**:     CM: (ISRC) MASK IR
        AR: R10B=R10A AND [+34] PC: UW
        BR: CJN(1) [IS.M4+2]

**:     CM: BRANCH IF USING SET MODE
        AR: R10B=IR5
```

```
        CC: C=C,V=V,N=V,Z=0     BR: CBIN(Z) <SCR>

**:     CM: ELSE USE FPS MODE
        AR: R16A+R16B+<0>       PC: UB
        CC: C=C,V=PN,N=PN,Z=0   BR: CBIC(1) <SCR>

        .=: IS.M0+20

**: IS.M4
        CM: MODE 4 / (R) - (2 OR 4); (R) IS ADDRESS
        AR: D;RB(DST)=RA(DST)-[+2]-<1>
        BR: CBN(/V) [IS.M5+1]

**:     CM: ELSE DOUBLE INTEGER
        AR: D;RB(DST)=RA(DST)-[+2]-<1>
        BR: CBN(1) [IS.M5+1]

**:     CM: (ISRC) BUILD JUMP ADDRESS
        AR: SCE5=R10A+[IS.M0]+<0>       PC: UW
        SP: CLR,SE5             BR: CRR(1)

**:     CM: (MODE 2) SET IMMEDIATE FLAG
        CC: C=C,V=V,N=0,Z=Z     BR: CBN(1) [IS.M5+2]

        .=: IS.M0+24

**: IS.M5
        CM: MODE 5 / (R) - 2 ; (R) IS ADDRESS OF ADDRESS
        AR: AD;RB(DST)=RA(DST)-[+2]-<0>
        IO: DATI(CMI)

**:     CM: CHECK STACK OVERFLOW
        AR: SOR=R6A-SLR-<1>

**:     CM: RETURN IF ONLY ADDRESS NEEDED
        AR: AD;R10B=D           PC: LW,WIOH
        BR: CRN(C)

**:     CM: ELSE GET INTEGER
        AR: R15B=0              PC: 4B
        IO: IDATI(CMI)          BR: CBN(1) [B.GETINT]

        .=: IS.M0+30

**: IS.M6
        CM: MODE 6 / (R) + X IS ADDRESS
        AR: R7B=R7A+[+2]+<0>,AD=R7A
        IO: DATI(CMI)

**:     CM: RETURN IF ONLY ADDRESS WANTED
        AR: AD;R10B=RA(DST)+D+<0>       PC: LW,WIOH
        BR: CRN(C)

**:     CM: ELSE GET INTEGER
        AR: R15B=0              PC: 4B
        IO: IDATI(CMI)          BR: CBN(1) [B.GETINT]

**:     CM: (MODE 7) GET X
        AR: AD=RA(DST)+D+<0>    PC: LW,WIOH
        IO: DATI(CMI)           BR: CBN(1) [IS.M5+2]

        .=: IS.M0+34

**: IS.M7
        CM: MODE 7 / (R) + X IS ADDRESS OF ADDRESS
        AR: R7B=R7A+[+2]+<0>,AD=R7A
        IO: DATI(CMI)           BR: CBN(1) [IS.M6+3]

**: B.GETINT
        CM: CLEAR HIGH FRACTION
        AR: R14B=0              PC: 4B

**:     CM: SET UP ADDRESS
        AR: AD=R10A+[+2]+<0>    PC: LW,WIOL
        BR: CBN(V) [.+2]
```

```
**:     CM: LOAD INTEGER
        AR: R15B=D              PC: LW
        CC: Z=PZ,N=PN,C=PN,V=V  BR: CBN(1) [.+2]

**:     CM: LOAD LONG INTEGER
        AR: R15B=D              PC: UW
        CC: Z=PZ,N=PN,C=PN,V=V  BR: CJN(N) [IS.M0+2]

**:     CM: SAVE SIGN
        AR: R5B=0,SRD(C)        PC: UW
        CC: C=0,N=N,Z=Z,V=V     BR: CRN(/C)

**:     CM: ELSE GET MAGNITUDE
        AR: R15B=-R15B-<1>      PC: LW
        BR: CRN(/V)

**:     AR: R15B=-R15B-<C'>     PC: UW
        BR: CRN(1)

        PG:     SAVE RESULT IN ACCUMULATOR

CM:     IF C=1 A CLEAN ZERO IS GENERATED, ELSE
        R4 (UW) - EXPONENT
        R5 (UW) - SIGN OF FRACTION
        R14 (4B) - HIGH FRACTION
        R15 (4B) - LOW FRACTION
        ARE COMBINED INTO THE STANDARD FLOATING POINT
        FORMAT = FLOATING (V=0)
         SIGN(1-BIT)/EXPONENT(8-BITS)/FRACTION(23-BITS)
        FORMAT = DOUBLE (V=1)
         SIGN(1-BIT)/EXPONENT(8-BITS)/FRACTION(55-BITS)
        AND SAVE IN THE ACCUMULATOR AT THE ADDRESS IN R10 (LW)

**: ACC.SAV
        CM: COMBINE SIGN, EXPONENT, AND FRACTION
        AR: R14B=[+177600] MASK R14A    PC: UW
        BR: CJN(1) [CMB.SEF+1]

**: ACC.SAVA
        CM: SAVE RESULT
        AR: AD=R10A             IO: DATO(OFF)

**:     AR: D=R14B              PC: LW,WIOL
        CC: C=0,N=N,Z=Z,V=V

**:     AR: AD=R10A+[+2]+<0>    IO: DATO(OFF)
        BR: CRN(/V)

**:     CM: DOUBLE MODE
        AR: D=R15B              PC: UW,WIOL
        CC: C=0,V=0,N=N,Z=Z

**:     AR: AD=R10A+[+4]+<0>    IO: DATO(OFF)

**:     AR: D=R15B              PC: LW,WIOL

**:     AR: AD=R10A+[+6]+<0>    IO: DATO(OFF)
        BR: CRR(1)

        PG:     CFCC, SETF, SETI, SETD, AND SETL

**: SET.FPS
        CM: DETERMINE WHICH INSTRUCTION
        AR: R10B=IR5            PC: UW

**:     CM: BRANCH IF CFCC
        AR: R10A-[+1]-<1>       PC: UW
        IO: INTCK               BR: CBN(Z') [CFCC]

**:     CM: BRANCH IF SETF
        AR: R10A-[+2]-<1>       PC: UW
        IO: INTCK               BR: CBN(Z') [SETF]

**:     CM: BRANCH IF SETI
        AR: R10A-[+11]-<1>      PC: UW
        IO: INTCK               BR: CBN(Z') [SETI]
```

```
**:     CM: BRANCH IF SETD
        AR: R10A-[+12]-<1>      PC: UW
        IO: INTCK               BR: CBN(Z') [SETD]

**:     CM: BRANCH IF SETL
        AR: R0B=PSR             PC: UW
        IO: INTCK               BR: CBN(Z') [SETL]

**:     CM: GO CHECK RESERVED INSTRUCTIONS FOR DEFINITION
        AR: AD=[+177764]        IO: DATI(OFF)
        BR: CJN(1) [RSRVSTAT]

**:     CM: ELSE INVALID INSTRUCTION
        AR: SCR1=R7A-[+2]-<1>

**:     CM: SET UP END BRANCH
        AR: SCE5=[BGN+3]                SP: CLR,SE5

**: FOP.TRP
        CM: UNDEFINED FLOATING INSTRUCTION TRAP
        AR: D=[+2]              PC: LW,WIOL
        BR: CJN(1) [FPP.TRP]

**:     CM: GO TO BGN OR TRAP OUT
        AR: AD=R7A+[+0]+<0>     PC: LW,WIOL
        IO: INTCK               BR: CBIN(1) <SCR>

**: CFCC
        CM: TRANSFER CONDITION CODES
        AR: OUT=R6B             PC: UW
        CC: C=CPO,N=CPO,Z=CPO,V=CPO
        IO: DATIR(CMI)          BR: CBR(1) [BGN+1]

**: SETF
        CM: SET FLOATING MODE
        AR: R16B=[+200] MASK R16A       PC: UW
        IO: IDATIR(CMI)         BR: CBR(1) [BGN+1]

**: SETI
        CM: SET INTEGER MODE
        AR: R16B=[+100] MASK R16A       PC: UW
        IO: IDATIR(CMI)         BR: CBN(1) [BGN+1]

**: SETD
        CM: SET DOUBLE MODE
        AR: R16B=R16A OR [+200]         PC: UW
        IO: IDATIR(CMI)         BR: CBN(1) [BGN+1]

**: SETL
        CM: SET LONG INTEGER MODE
        AR: R16B=R16A OR [+100]         PC: UW
        IO: IDATIR(CMI)         BR: CBN(1) [BGN+1]

        PG:     LDFPS AND STFPS

**: LDFPS
        CM: GET NEXT INSTRUCTION
        AR: AD=R7A              PC: LW,WIOL
        IO: INTCK

**:     CM: SAVE FPS
        AR: R16B=D              PC: UW,WIOL
        IO: IDATIR(CMI)

**:     AR: R6B=D               PC: UW
        BR: CBN(1) [BGN+1]

**: STFPS
        CM: MASK FCC'S
        AR: R1B=[+17]           PC: UW

**:     AR: R6B=R1A AND R6B     PC: UW

**:     CM: MASK FPS (LEAVE UNCHANGED)
        AR: R1B=[+30037] MASK R16A      PC: UW
```

```
**:     CM: CHECK FOR MODE 0
        AR: R10B=[+70]

**:     AR: R10A AND IR5

**:     CM: COMBINE FPS & FCC'S
            SKIP IF MODES 1-7
        AR: D=R6B OR R1A        PC: UW
        BR: CBN(/Z') [.+2]

**:     CM: MODE 0
        AR: RB(DST)=D           BR: CBN(1) [BGN]

**:     CM: MODES 1-7
        IO: IDATO(CM)           BR: CBN(1) [BGN]

        PG:     STST

**: STST
        CM: SAVE CC'S / GET ADDRESS
        AR: R0B=PSR             PC: UW
        CC: C=1,N=0,Z=1,V=1     BR: CJN(1) [ISRC]

**:     CM: GET FEC
        AR: AD=[FEC.FPP]        IO: DATI(OFF)
        BR: CBN(/Z) [.+3]

**:     CM: MODE 0
        AR: RB(DST)=D           PC: LW,WIOH

**:     CM: RESTORE CC'S
        AR: OUT=R0B             PC: UW
        CC: C=CPO,N=CPO,Z=CPO,V=CPO
        BR: CBN(1) [BGN]

**:     CM: SAVE FEC / FINISHED IF IMMEDIATE
        AR: AD=R10A             PC: LW,WIOL
        IO: DATO(CM)            BR: CBN(/N) [.-1]

**:     CM: ELSE GET FEA
        AR: AD=[FEA.FPP]        PC: LW,WIOL
        IO: DATI(OFF)

**:     CM: SAVE FEA
        AR: AD=R10A+[+2]+<0>    PC: LW,WIOL
        IO: DATO(CM)            BR: CBN(1) [.-3]

        PG:     CLRF AND TSTF

**: CLRF
        CM: SAVE CC'S / GET ADDRESS WITH CURRENT MODE
        AR: R0B=PSR             PC: UW
        CC: C=1,Z=0,N=0,V=0     BR: CJN(1) [FSRC]

**:     CM: SET END BRANCH
        AR: SCE5=[BGN]          SP: CLR,SE5

**:     CM: SAVE A CLEAN ZERO
        CC: C=1,Z=0,N=0,V=V     BR: CBN(1) [NEGF+4]

**: TSTF
        CM: SAVE CC'S / GET SOURCE DATA
        AR: R0B=PSR             PC: UW
        CC: C=0,N=0,Z=0,V=0     BR: CJN(1) [FSRC.SAV]

**:     CM: SET UP END BRANCH
        AR: SCE5=[BGN]          SP: CLR,SE5

**:     CM: CHECK FOR -0 IF NOT MODE 0
        BR: CJN(C) [FIUV.FPP+1]

**:     CM: MASK FCC'S / GO RESTORE CC'S
        CC: C=0,V=0,N=N,Z=Z     BR: CBN(1) [NEGF+6]

        PG:     ABSF AND NEGF
```

```
**: ABSF
        CM: SAVE CC'S / GET SOURCE DATA
        AR: R0B=PSR             PC: UW
        CC: C=0,N=0,Z=0,V=0     BR: CJN(1) [FSRC.SAV]

**:     CM: SET UP END BRANCH
        AR: SCE5=[BGN]          SP: CLR,SE5

**:     CM: CHECK FOR -0 IF NOT MODE 0
        BR: CJN(C) [FIUV.FPP]

**:     CM: MAKE ABSOLUTE / GO SAVE RESULT
        AR: R5B=0               PC: UW
        BR: CBN(1) [NEGF+4]

**: NEGF
        CM: SAVE CC'S / GET SOURCE DATA
        AR: R0B=PSR             PC: UW
        CC: C=0,N=0,Z=0,V=0     BR: CJN(1) [FSRC.SAV]

**:     CM: SET UP END BRANCH
        AR: SCE5=[BGN]          SP: CLR,SE5

**:     CM: CHECK FOR -0 IF NOT MODE 0
        BR: CJN(C) [FIUV.FPP+1]

**:     CM: NEGATE SIGN / CHECK FOR -0
        AR: R5B=/R5B            PC: UW
        CC: N=PN,C=C,Z=Z,V=V    BR: CJN(Z) [FIUV.FPP+1]

**:     CM: CHECK MODE / GO BUILD SEF
        AR: R1B=IR5             PC: UW
        BR: CJN(1) [CMB.SEF]

**:     AR: R1A AND [+70]       PC: UW
        BR: CJN(1) [FS.M7+1]

**:     CM: SAVE FCC'S / RESTORE CC'S / DO END BRANCH
        AR: R6B=PSR,OUT=R0A     PC: UW
        CC: C=CPO,N=CPO,Z=CPO,V=CPO
        BR: CBIN(1) <SCR>

        PG:     MULF/MULD INSTRUCTIONS

**: MULF
        CM: POINT TO ACCUMULATOR
        AR: R11B=[AC0.FPP]      BR: CBN(1) [.+4]

**:     AR: R11B=[AC1.FPP]      BR: CBN(1) [.+3]

**:     AR: R11B=[AC2.FPP]      BR: CBN(1) [.+2]

**:     AR: R11B=[AC3.FPP]      BR: CBN(1) [.+1]

**:     CM: SAVE CC'S / GET OPERANDS
        AR: R0B=PSR             PC: UW
        CC: C=0,N=0,Z=0,V=0     BR: CJN(1) [GETOPS]

**:     CM: ON ZERO OPERANDS - CLEAN ZERO
        AR: R4B=R4B             PC: UW
        CC: C=1,Z=PZ,N=N,V=V    BR: CBN(Z) [CLRF+2]

**:     BR: CBN(Z) [CLRF+2]

**:     CM: FLOATING MODE
        AR: CNTR=[+30]          BR: CJN(/V) [MUL.24+1]

**:     CM: DOUBLE MODE
        AR: CNTR=[+40]          BR: CJN(V) [MUL.56+1]

**:     CM: GO TO ANSWER
        BR: CBN(1) [ANSWER]

        PG:     MODF/MODD INSTRUCTIONS
```

```
**: MODF
        CM: POINT TO ACCUMULATOR
        AR: R11B=[AC0.FPP]      BR: CBN(1) [.+4]

**:     AR: R11B=[AC1.FPP]      BR: CBN(1) [.+3]

**:     AR: R11B=[AC2.FPP]      BR: CBN(1) [.+2]

**:     AR: R11B=[AC3.FPP]      BR: CBN(1) [.+1]

**:     CM: SAVE CC'S / GET OPERANDS
        AR: R0B=PSR             PC: UW
        CC: C=0,N=0,Z=0,V=0     BR: CJN(1) [GETOPS]

**:     CM: ON ZERO OPERANDS - FINISHED
        AR: R4B=R4B             PC: UW
        CC: C=PZ,Z=Z,N=0,V=V

**:     CM: CLEAR FCC'S
        AR: R6B=0               PC: UW
        BR: CBN(COZ) [MOD.A]

**:     CM: BRANCH ON DOUBLE
        AR: CNTR=[+40]          BR: CBN(V) [.+3]

**:     CM: ELSE CLEAR LOW FRACTIONS
        AR: R13B=0              PC: 4B

**:     AR: R15B=0              PC: 4B

**:     CM: DOUBLE MODE MULTIPLY
        AR: R11B=R14A           PC: 4B
        BR: CJN(1) [MUL.56+2]

**:     CM: USE ONLY 86 BITS OF RESULT
        AR: Q=[+3] MASK Q       PC: LW

**:     CM: COPY RESULT TO ARGUMENT A
        AR: R12B=R14A           PC: 4B

**:     AR: R13B=R15A           PC: 4B

**:     AR: R3B=R5A             PC: UW

**:     CM: AND CHECK EXPONENT
        AR: R2B=R4A             PC: UW
        CC: C=0,N=N,Z=Z,V=V     BR: CJN(1) [EXP.UN+1]

**:     CM: CHECK FOR OVERFLOW / DONE IF C=1
        AR: R6B                 PC: UW
        BR: CBN(C) [MOD.A]

**:     CM: CHECK EXPONENT / DONE ON OVERFLOW
        AR: CNTR=R4A-[+200]-<1> PC: UW
        BR: CBN(/Z') [MOD.B]

**:     CM: IF EXP <= 200 - ONLY FRACTION
        AR: R4A-[+270]-<1>      PC: UW
        BR: CBN(BLE') [MOD.C]

**:     CM: IF EXP < 270 - DO SHIFTING
        AR: R4A-[+337]-<1>      PC: UW
        BR: CBN(BLT') [MODF.A]

**:     CM: IF EXP > 337 - FRACTION = 0
        AR: CNTR=R4A-[+240]-<1> PC: UW
        BR: CBN(BGT') [MOD.B]

**:     CM: ELSE DO SHIFTING
        AR: R12B=R12B,SRU(MSBQ),SQ(0)   PC: 4B,CCL
        BR: CNR(/CNTR)

**:     AR: R13B=Q              PC: 4B

**:     AR: Q=0                 PC: 4B
```

```
**:     AR: R12B=0,D=R12A       PC: UW

**:     CM: SAVE INTEGER PART
        AR: R12B=D              PC: UB
        CC: C=0,N=N,Z=Z,V=V     BR: CJN(1) [INT.SAV+3]

**:     CM: NORMALIZE AND SAVE FRACTION
        AR: R2B=[+200]          PC: UW
        BR: CBN(1) [FRC.SAV]

**: MODF.A
        CM: CLEAR FLOATING
        AR: R14B=0              PC: 4B

**:     AR: R15B=0              PC: 4B

**:     CM: SHIFT PROD TO GET INTEGER IN R14/R15
        AR: R13B=R13B,SRU(MSBQ),SQ(0)   PC: 4B
        CC: C=MSBR,N=N,Z=Z,V=V

**:     AR: R12B=R12B,SRU(C)    PC: 3B
        CC: C=MSBR,N=N,Z=Z,V=V

**:     AR: R15B=R15B,SRU(C)    PC: 4B
        CC: C=MSBR,N=N,Z=Z,V=V

**:     AR: R14B=R14B,SRU(C)    PC: 3B,CCL
        CC: C=0,N=N,Z=Z,V=V     BR: CBN(CNTR) [.-3]

**:     CM: NORMALIZE AND SAVE INTEGER PART
        AR: R4B=[+271]          PC: UW
        BR: CJN(1) [INT.SAV]

**:     CM: NORMALIZE AND SAVE FRACTION
        AR: R2B=[+200]          PC: UW
        BR: CBN(1) [FRC.SAV]

**: MOD.A
        CM: SAVE A CLEAN ZERO INTEGER
        AR: R16B                PC: UB
        CC: C=1,V=PN,N=N,Z=Z    BR: CJN(1) [INT.SAV+3]

**:     CM: SAVE A CLEAN ZERO FRACTION
        AR: R16B                PC: UB
        CC: C=1,V=PN,N=N,Z=Z    BR: CBN(1) [FRC.SAV+10]

**: MOD.B
        CM: SAVE INTEGER PART
        AR: R16B                PC: UB
        CC: V=PN,C=0,N=N,Z=Z    BR: CJN(1) [INT.SAV+3]

**:     CM: SAVE A CLEAN ZERO FRACTION
        AR: R16B                PC: UB
        CC: C=1,V=PN,N=N,Z=Z    BR: CBN(1) [FRC.SAV+10]

**: MOD.C
        CM: SAVE A CLEAN ZERO INTEGER
        AR: R16B                PC: UB
        CC: C=1,V=PN,N=N,Z=Z    BR: CJN(1) [INT.SAV+3]

**:     CM: NORMALIZE AND SAVE FRACTION
        AR: R16B                PC: UB
        CC: C=0,V=PN,N=N,Z=Z    BR: CBN(1) [FRC.SAV+1]

**: INT.SAV
        CM: SAVE Q REGISTER / NORMALIZE INTEGER
        AR: R11B=Q              PC: 4B
        CC: C=0,N=0,Z=0,V=V     BR: CJN(1) [NORM.56]

**:     CM: RESTORE Q REGISTER
        AR: Q=R11A              PC: 4B

**:     CM: SET UP LENGTH FLAG
        AR: R16B                PC: UB
        CC: V=PN,C=C,N=N,Z=Z
```

```
**:     CM: RESET ADDRESS
        AR: R11B=R10A           PC: LW

**:     CM: SAVE INTEGER IN (ACC OR 1)
        AR: R10B=R11A OR [+10]  PC: LW
        BR: CBN(1) [ACC.SAV]

**: FRC.SAV
        CM: SET UP LENGTH FLAG / SKIP ON CLEAN ZERO
        AR: R16B                PC: UB
        CC: V=PN,C=C,N=0,Z=0    BR: CBN(C) [.+10]

**:     CM: COPY R12/R13 TO R14/R15
        AR: R1B=[+40]           PC: UW
        BR: CJN(1) [COPY.56]

**:     CM: SET UP TRUNCATION FLAG / SKIP ON DOUBLE
        AR: R16B AND R1A        PC: UW
        CC: N=PZ,C=C,Z=Z,V=V    BR: CBN(V) [.+5]

**:     CM: NORMALIZE FRACTION
        AR: R4B=R4B+<1>         PC: UW
        CC: C=0,N=0,Z=0,V=V     BR: CJN(1) [NORM.56+1]

**:     CM: SET UP TRUNCATION FLAG / SKIP ON CLEAN ZERO
        AR: R16B AND R1A        PC: UW
        CC: N=PZ,C=C,Z=Z,V=V    BR: CBN(C) [.+4]

**:     CM: CONDITIONAL ROUND
        AR: R15B                PC: 4B
        CC: C=PN,N=N,Z=Z,V=V    BR: CJN(N) [NORM.24A]

**:     CC: C=0,N=0,Z=0,V=V     BR: CBN(1) [.+2]

**:     CM: NORMALIZE INTEGER
        AR: R4B=R4B+<1>         PC: UW
        BR: CJN(1) [NORM.56+1]

**:     CM: SAVE FRACTION
        AR: R10B=R11A           BR: CJN(1) [ACC.SAV]

**:     CM: RESTORE CC'S / BUILD FCC'S
        AR: R0B=PSR,OUT=R0A     PC: UW
        CC: C=CPO,N=CPO,Z=CPO,V=CPO

**:     CM: SET FCC'S / FINISHED / TAP
        AR: R6B=R6B OR R0A      PC: UW
        BR: CBIN(1) <SCR>

        PG:     ADDF/ADDD INSTRUCTIONS

**: ADDF
        CM: POINT TO ACCUMULATOR
        AR: R11B=[AC0.FPP]      BR: CBN(1) [.+4]

**:     AR: R11B=[AC1.FPP]      BR: CBN(1) [.+3]

**:     AR: R11B=[AC2.FPP]      BR: CBN(1) [.+2]

**:     AR: R11B=[AC3.FPP]      BR: CBN(1) [.+1]

**:     CM: SAVE CC'S / GET OPERANDS
        AR: R0B=PSR             PC: UW
        CC: C=0,N=0,Z=0,V=0     BR: CJN(1) [GETOPS]

**:     CM: FLOATING MODE
        AR: R2B=R2B             PC: UW
        CC: C=0,Z=PZ,N=N,V=V
        BR: CJN(/V) [ADD.24+1]

**:     CM: DOUBLE MODE / FALL THROUGH TO ANSWER
        BR: CJN(V) [ADD.56+1]

        PG:     ANSWER ROUTINE

CM:     THIS ROUTINE CHECKS THE ARITHMETIC ANSWER
```

```
        FOR EXPONENT UNDERFLOW/OVERFLOW AND
        SAVES THE RESULT IN THE ACCUMULATOR AT
        ADDRESS R10 (LW)

**: ANSWER
        CM: CLEAR FCC'S / CHECK EXPONENT IF NOT CLEAN ZERO
        AR: R6B=0               PC: UW
        BR: CJN(/C) [EXP.UN+1]

**:     AR: R14B=[+177600] MASK R14A    PC: UW
        BR: CJN(1) [CMB.SEF+1]

**:     AR: AD=R10A             IO: DATO(OFF)
        BR: CJN(1) [ACC.SAV+2]

**:     CM: SAVE CURRENT STATUS / RESTORE CC'S
        AR: R0B=PSR,OUT=R0A     PC: UW
        CC: C=CPO,N=CPO,Z=CPO,V=CPO

**:     CM: SAVE FCC'S / TRAP OUT / FINISHED
        AR: R6B=R6B OR R0A      PC: UW
        BR: CBIN(1) <SCR>

        PG:     LDF/LDD INSTRUCTIONS

**: LDF
        CM: POINT TO ACCUMULATOR
        AR: R11B=[AC0.FPP]      BR: CBN(1) [.+4]

**:     AR: R11B=[AC1.FPP]      BR: CBN(1) [.+3]

**:     AR: R11B=[AC2.FPP]      BR: CBN(1) [.+2]

**:     AR: R11B=[AC3.FPP]      BR: CBN(1) [.+1]

**:     CM: SAVE CC'S / GET SOURCE DATA
        AR: R0B=PSR             PC: UW
        CC: C=0,Z=0,N=0,V=0     BR: CJN(1) [FSRC.SAV]

**:     CM: SET UP END BRANCH
        AR: SCE5=[.+3]          SP: CLR,SE5

**:     CM: CHECK FOR -0 IF NOT MODE 0
        AR: R1B=PSR             PC: UW
        BR: CJN(C) [FIUV.FPP]

**:     CM: MASK STATUS / TRAP ON -0 AND FIUV=1
        AR: R6B=R1A AND [+14]   PC: UW
        BR: CBIN(1) <SCR>

**:     CM: ELSE SAVE IN ACCUMULATOR
        AR: R10B=R11A           CC: C=0,N=N,Z=Z,V=V
        BR: CJN(1) [ACC.SAV]

**:     CM: RESTORE CC'S / FINISHED
        AR: OUT=R0B             PC: UW
        CC: C=CPO,N=CPO,Z=CPO,V=CPO
        BR: CBN(1) [BGN]

        PG:     SUBF/SUBD INSTRUCTIONS

**: SUBF
        CM: POINT TO ACCUMULATOR
        AR: R11B=[AC0.FPP]      BR: CBN(1) [.+4]

**:     AR: R11B=[AC1.FPP]      BR: CBN(1) [.+3]

**:     AR: R11B=[AC2.FPP]      BR: CBN(1) [.+2]

**:     AR: R11B=[AC3.FPP]      BR: CBN(1) [.+1]

**:     CM: SAVE CC'S / GET OPERANDS
        AR: R0B=PSR             PC: UW
        CC: C=0,N=0,Z=0,V=0     BR: CJN(1) [GETOPS]

**:     CM: CHANGE SIGN OF B ARGUMENT / GO TO ADDF
```

```
        AR: R5B=R5A XOR [+100000]       PC: UW
        BR: CBN(1) [ADDF+5]

        PG:     CMPF/CMPD INSTRUCTIONS

**: CMPF
        CM: POINT TO ACCUMULATOR
        AR: R11B=[AC0.FPP]      BR: CBN(1) [.+4]

**:     AR: R11B=[AC1.FPP]      BR: CBN(1) [.+3]

**:     AR: R11B=[AC2.FPP]      BR: CBN(1) [.+2]

**:     AR: R11B=[AC3.FPP]      BR: CBN(1) [.+1]

**:     CM: SAVE CC'S / GET OPERANDS
        AR: R0B=PSR             PC: UW
        CC: C=0,N=0,Z=0,V=0     BR: CJN(1) [GETOPS]

**:     CM: COMPARE EXPONENTS
        AR: R4B-R2A-<1>         PC: UW
        CC: C=0,N=PN,Z=PZ,V=V   SP: CLR

**:     CM: NEGATE AC / BRANCH IF EXPONENTS NOT EQUAL
        AR: R3B=R3A XOR [+100000]       PC: UW
        BR: CBN(/Z) [.+5]

**:     CM: ELSE CHECK ADDITION OF FRACTIONS
        AR: R5B XOR R3A         PC: UW
        BR: CJN(/V) [ADD.24A]

**:     AR: R5B XOR R3A         PC: UW
        BR: CJN(V) [ADD.56A]

**:     CM: CHECK RESULT TO SET FLAGS
        AR: R14B=R14B           PC: 4B
        CC: C=0,Z=PZ,N=0,V=0    BR: CBN(/V) [.+3]

**:     AR: R15B=R15B           PC: 4B
        CC: C=0,Z=PZAZ,N=0,V=0  BR: CBN(1) [.+2]

**:     CM: IF EXP (A>B) SIGN IS SIGN OF A
        AR: R3B                 PC: UW
        CC: C=0,N=PN,Z=0,V=0    BR: CBN(N) [.+2]

**:     CM: SIGN IS SIGN OF (FSRC)/RESULT
        AR: R5B                 PC: UW
        CC: C=0,N=PN,Z=Z,V=0

**:     CM: SAVE FCC'S / RESTORE CC'S
        AR: R6B=PSR,OUT=R0A     PC: UW
        CC: C=CPO,N=CPO,Z=CPO,V=CPO
        BR: CBN(1) [BGN]

        PG:     STF/STD INSTRUCTIONS

**: STF
        CM: POINT TO ACCUMULATOR
        AR: AD;R10B=[AC0.FPP]   IO: DATI(OFF)
        BR: CBN(1) [.+4]

**:     AR: AD;R10B=[AC1.FPP]   IO: DATI(OFF)
        BR: CBN(1) [.+3]

**:     AR: AD;R10B=[AC2.FPP]   IO: DATI(OFF)
        BR: CBN(1) [.+2]

**:     AR: AD;R10B=[AC3.FPP]   IO: DATI(OFF)
        BR: CBN(1) [.+1]

**:     CM: SAVE CC'S
        AR: R0B=PSR             PC: UW

**:     AR: R16B                PC: UB
        CC: C=0,N=0,Z=0,V=PN    BR: CJN(1) [B.GETACC+1]
```

```
**:     CM: GET DESTINATION ADDRESS
        AR: R10B=IR5,SRD(0)     PC: UW
        CC: C=1,N=0,Z=1,V=V     BR: CJN(1) [FSRC+1]

**:     CM: STORE AT DESTINATION
        CC: C=0,N=0,Z=0,V=V
        BR: CJN(1) [FDST]

**:     CM: RESTORE CC'S / FINISHED
        AR: OUT=R0B             PC: UW
        CC: C=CPO,N=CPO,Z=CPO,V=CPO
        BR: CBN(1) [BGN]

        PG:     DIVF/DIVD INSTRUCTIONS

**: DIVF
        CM: POINT TO ACCUMULATOR
        AR: R11B=[AC0.FPP]      BR: CBN(1) [.+4]

**:     AR: R11B=[AC1.FPP]      BR: CBN(1) [.+3]

**:     AR: R11B=[AC2.FPP]      BR: CBN(1) [.+2]

**:     AR: R11B=[AC3.FPP]      BR: CBN(1) [.+1]

**:     CM: SAVE CC'S / GET OPERANDS
        AR: R0B=PSR             PC: UW
        CC: C=0,N=0,Z=0,V=0     BR: CJN(1) [GETOPS]

**:     CM: SET UP END BRANCH
        AR: SCE5=[.+4]          SP: CLR,SE5

**:     CM: CHECK FOR ZERO DIVISOR
        AR: R4B=R4B             PC: UW

**:     AR: D=[+4]              PC: LW,WIOL
        BR: CJN(Z') [FPP.TRP]

**:     CM: TRAP IF DIVISOR = 0
        AR: SCE5=[BGN]          BR: CBIN(1) <SCR>

**:     CM: BRANCH IF ZERO DIVIDEND
        SP: CLR,SE5             BR: CBN(Z) [CLRF+2]

**:     CM: FLOATING MODE
        AR: CNTR=[+32]          BR: CJN(/V) [DIV.24+1]

**:     CM: DOUBLE MODE
        AR: CNTR=[+32]          BR: CJN(V) [DIV.56+1]

**:     CM: GO TO ANSWER
        BR: CBN(1) [ANSWER]

        PG:     STEXP INSTRUCTION

**: STEXP
        CM: POINT TO ACCUMULATOR
        AR: AD=[AC0.FPP]        IO: DATI(OFF)
        BR: CBN(1) [.+4]

**:     AR: AD=[AC1.FPP]        IO: DATI(OFF)
        BR: CBN(1) [.+3]

**:     AR: AD=[AC2.FPP]        IO: DATI(OFF)
        BR: CBN(1) [.+2]

**:     AR: AD=[AC3.FPP]        IO: DATI(OFF)
        BR: CBN(1) [.+1]

**:     CM: SET UP MASK FOR EXPONENT
        AR: R10B=[+77600]

**:     CM: LOAD ADDRESS OF DESTINATION
        AR: R12B=[+200],AD=R11A PC: LW,WIOL

**:     CM: POSITION EXPONENT
```

```
        AR: R10B=D AND R10A,SRU(0)

**:     AR: R10B=IR5,D=R10A

**:     CM: COMPUTE TWO'S COMPLEMEMT FORM
        AR: D=DSWB-R12A-<1>
        CC: C=0,V=0,N=PN,Z=PZ

**:     CM: CHECK MODE
        AR: R10A AND [+70]

**:     CM: SAVE FCC'S / CC'S THE SAME
        AR: R6B=PSR             PC: UW
        BR: CBN(/Z') [.+2]

**:     CM: MODE 0
        AR: RB(DST)=D           BR: CBN(1) [BGN]

**:     CM: MODES 1-7
        IO: IDATO(CM)           BR: CBN(1) [BGN]

        PG:     STCFI,STCFL,STCDI,STCDL INSTRUCTIONS

**: STCFI
        CM: POINT TO ACCUMULATOR
        AR: AD;R10B=[AC0.FPP]   IO: DATI(OFF)
        BR: CBN(1) [.+4]

**:     AR: AD;R10B=[AC1.FPP]   IO: DATI(OFF)
        BR: CBN(1) [.+3]

**:     AR: AD;R10B=[AC2.FPP]   IO: DATI(OFF)
        BR: CBN(1) [.+2]

**:     AR: AD;R10B=[AC3.FPP]   IO: DATI(OFF)
        BR: CBN(1) [.+1]

**:     CM: SAVE CC'S / GET ACCUMULATOR
        AR: R0B=PSR             PC: UW
        CC: C=0,N=0,Z=0,V=0     BR: CJN(1) [B.GTAC]

**:     CM: CLEAR FCC'S / SKIP ON DOUBLE
        AR: R6B=0               PC: UW
        BR: CBN(V) [.+2]

**:     CM: CLEAR LOW FRACTION
        AR: R15B=0              PC: 4B

**:     CM: SAVE DATA FLAGS / GET SOURCE ADDRESS
        AR: R1B=PSR             PC: UW
        CC: C=1,N=0,Z=0,V=0     BR: CJN(1) [ISRC.SAV]

**:     CM: SAVE ISRC MODES / RESTORE DATA FLAGS
        AR: R1B=PSR,OUT=R1A     PC: UW
        CC: C=CPO,N=CPO,Z=CPO,V=V

**:     CM: SET END BRANCH
        AR: SCE5=[BGN]          SP: CLR,SE5

**:     CM: CHECK EXPONENT / BRANCH IF LONG
        AR: CNTR;R4B=R4A-[+200]-<1>     PC: UW
        BR: CBN(V) [STCF.L]

**: STCF.I
        CM: CHECK INTEGER MAGNITUDE
        AR: R4A-[+20]-</N>      PC: UW
        BR: CBN(BLE') [.+3]

**:     CM: SET Q / BRANCH IF IN RANGE
        AR: Q=0                 PC: 4B
        CC: C=0,N=N,Z=Z,V=0     BR: CBN(N') [.+3]

**:     CM: ELSE CONVERSION ERROR / SET FCC
        AR: R6B=[+1]            PC: UW
        BR: CJN(1) [FIC.TRP]
```

```
**:     CM: GENERATE A CLEAN ZERO
        AR: R13B=0              PC: 4B
        CC: C=0,N=0,Z=1,V=0     BR:CBN(1) [.+5]

**:     CM: SHIFT INTEGER INTO POSITION
        AR: R14B=R14B,SRU(0),SQ(MSBR)   PC: 3B,CCL
        BR: CNR(/CNTR)

**:     CM: GET INTEGER / SKIP IF POSITIVE
        AR: R13B=Q              PC: 4B
        BR: CBN(/N) [.+3]

**:     CM: COMPARE TO MOST NEGATIVE NUMBER
        AR: R13A-[+100000]-<0>  PC: LW

**:     CM: CONVERSION ERROR IF R13>100000
        AR: R13B=-R13A-<1>      PC: LW
        BR: CBN(/N') [.-5]

**:     CM: RESTORE ISRC MODES / SAVE CC'S
        AR: R1B=PSR,OUT=R1A     PC: UW
        CC: C=CPO,N=CPO,Z=CPO,V=CPO

**:     CM: PLACE DATA
        AR: D=R13A              PC: LW
        BR: CBN(/Z) [.+2]

**: STCF.M0
        CM: MODE 0
        AR: RB(DST)=D           BR: CBN(1) [.+2]

**:     CM: DESTINATION
        AR: AD=R10A             IO: DATO(CM)

**:     CM: SET CC'S/FCC'S / FINISHED / TRAP
        AR: R6B=R6B OR R1A      PC: UW
        CC: C=CPO,N=CPO,Z=CPO,V=CPO
        BR: CBIN(1) <SCR>

**: STCF.L
        CM: CHECK MAGNITUDE
        AR: R4A-[+40]-</N>      PC: UW
        BR: CBN(BLE') [.+3]

**:     CM: CLEAR Q / BRANCH IF IN RANGE
        AR: Q=0                 PC: 4B
        CC: C=0,N=N,Z=0,V=0     BR: CBN(N') [.+3]

**:     CM: ELSE CONVERSION ERROR / SET FCC
        AR: R6B=[+1]            PC: UW
        BR: CJN(1) [FIC.TRP]

**:     CM: GENERATE A CLEAN ZERO
        AR: R13B=0              PC: 4B
        CC: C=0,N=0,Z=1,V=0     BR: CBN(1) [.+7]

**:     CM: SHIFT INTEGER INTO POSITION
        AR: R15B=R15B,SRU(0),SQ(C)      PC: 4B
        CC: C=MSBR,N=N,Z=0,V=0  BR: CBN(/CNTR) [.+2]

**:     AR: R14B=R14B,SRU(C)    PC: 3B,CCL
        CC: C=MSBR,N=N,Z=0,V=0  BR: CBN(1) [.-1]

**:     CM: GET INTEGER
        AR: R13B=Q              PC: 4B
        CC: C=0,N=N,Z=Z,V=V     BR: CBN(/N) [.+4]

**:     CM: COMPARE TO MOST NEGATIVE
        AR: R12B=0,SRD(1)       PC: 4B

**:     AR: R13B-R12A-<0>       PC: 4B

**:     CM: CONVERSION ERROR IF R13>100000/000000
        AR: R13B=-R13B-<1>      PC: 4B
        BR: CBN(/N') [.-7]
```

```
**:     CM: RESTORE ISRC MODES
        AR: R1B=PSR,OUT=R1A     PC: UW
        CC: C=CPO,N=CPO,Z=CPO,V=CPO

**:     CM: SET DATA / BRANCH ON MODE 0
        AR: D=R13B              PC: UW
        BR: CBN(Z) [STCF.M0]

**:     CM: BRANCH ON IMMEDIATE
        AR: R10B=R10A+[+2]+<0>,AD=R10A
        IO: DATO(CM)            BR: CBN(/N) [STCF.M0+2]

**:     CM: ELSE DOUBLE
        AR: D=R13B              PC: LW,WIOL
        BR: CBN(1) [STCF.M0+1]

        PG:     STCFD/STCDF INSTRUCTIONS

**: STCFD
        CM: POINT TO ACCUMULATOR
        AR: AD;R10B=[AC0.FPP]   IO: DATI(OFF)
        BR: CBN(1) [.+4]

**:     AR: AD;R10B=[AC1.FPP]   IO: DATI(OFF)
        BR: CBN(1) [.+3]

**:     AR: AD;R10B=[AC2.FPP]   IO: DATI(OFF)
        BR: CBN(1) [.+2]

**:     AR: AD;R10B=[AC3.FPP]   IO: DATI(OFF)
        BR: CBN(1) [.+1]

**:     CM: SAVE CC'S / GET ACCUMULATOR
        AR: R0B=PSR             PC: UW
        CC: C=0,N=0,Z=0,V=0     BR: CJN(1) [B.GTAC]

**:     CM: COMPLEMENT LENGTH
        AR: OUT=/PSR            CC: V=CPO,C=C,N=N,Z=Z

**:     CM: SAVE TCC'S / GET DESTINATION ADDRESS
        AR: R1B=PSR             PC: UW
        CC: C=1,Z=1,N=0,V=V     BR: CJN(1) [FSRC.SAV]

**:     CM: SET UP END BRANCH
        AR: SCE5=[BGN]          SP: CLR,SE5

**:     CM: CLEAR FCC'S / RESTORE TCC'S / DO CONVERSION ROUND
        AR: R6B=0,OUT=R1A       PC: UW
        CC: C=CPO,N=CPO,Z=CPO,V=V
        BR: CJN(1) [CNV.RND]

**:     CM: SAVE RESULT
        BR: CJN(1) [FDST]

**:     CM: BUILD FCC'S / RESTORE CC'S
        AR: R0B=PSR,OUT=R0A     PC: UW
        CC: C=CPO,N=CPO,Z=CPO,V=CPO

**:     CM: FINISHED / TRAP
        AR: R6B=R6B OR R0A      PC: UW
        BR: CBIN(1) <SCR>

**: CNV.RND
        CM: SKIP ON DOUBLE SOURCE
        BR: CBN(/V) [.+2]

**:     CM: ELSE CLEAR LOWER FRACTION ON FLOATING SOURCE
        AR: R15B=0              PC: 4B
        BR: CRR(1)

**:     CM: RETURN ON ZERO SOURCE
        AR: R1B=[+40]           PC: UW
        BR: CRN(Z)

**:     CM: CHECK TRUNCATE
        AR: R16A AND R1B        PC: UW
```

```
**:     CM: RETURN IF CHOPPING
        BR: CRN(/Z')

**:     CM: ELSE ROUND FLOATING
        AR: R15B                PC: 4B
        CC: C=PN,N=N,Z=Z,V=V    BR: CJN(1) [NORM.24A]

**:     CM: CHECK OVERFLOW ON ROUNDING
        AR: R4A-[+400]-<1>      PC: UW
        BR: CBN(1) [EXP.OV+1]

        PG:     LDEXP INSTRUCTION

**: LDEXP
        CM: POINT TO ACCUMULATOR
        AR: AD;R10B=[AC0.FPP]   PC: LW,WIOL
        BR: CBN(1) [.+4]

**:     AR: AD;R10B=[AC1.FPP]   PC: LW,WIOL
        BR: CBN(1) [.+3]

**:     AR: AD;R10B=[AC2.FPP]   PC: LW,WIOL
        BR: CBN(1) [.+2]

**:     AR: AD;R10B=[AC3.FPP]   PC: LW,WIOL
        BR: CBN(1) [.+1]

**:     CM: SAVE DATA / GET ACCUMULATOR
        AR: R2B=D               PC: UW
        IO: DATI(OFF)

**:     CM: SAVE CC'S
        AR: R0B=PSR             PC: UW
        CC: C=0,N=0,Z=0,V=0     BR: CJN(1) [B.GTAC]

**:     CM: SET UP END BRANCH
        AR: SCE5=[BGN]          SP: CLR,SE5

**:     CM: SET NEW EXPONENT / SAVE
        AR: R4B=[+200]          PC: UW

**:     AR: R4B=R4B+R2A+<0>     PC: UW
        CC: C=0,N=0,Z=0,V=V     BR: CBN(1) [ANSWER]

        PG:     LDCIF/LDCLF/LCDID/LCDLD INSTRUCTIONS

**: LDCIF
        CM: POINT TO ACCUMULATOR
        AR: R11B=[AC0.FPP]      BR: CBN(1) [.+4]

**:     AR: R11B=[AC1.FPP]      BR: CBN(1) [.+3]

**:     AR: R11B=[AC2.FPP]      BR: CBN(1) [.+2]

**:     AR: R11B=[AC3.FPP]      BR: CBN(1) [.+1]

**:     CM: SAVE CC'S / GET INTEGER
        AR: R0B=PSR             PC: UW
        CC: C=0,N=0,Z=0,V=0     BR: CJN(1) [ISRC.SAV]

**:     CM: NORMALIZE INTEGER
        AR: R4B=[+271]          PC: UW
        BR: CJN(1) [NORM.56]

**:     CM: SET LENGTH / SKIP ON ZERO
        AR: R16B                PC: UB
        CC: C=C,N=N,Z=Z,V=PN    BR: CBN(Z) [.+2]

**:     CM: CHECK ROUNDING IF RESULT FLOATING
        BR: CJN(/V) [CNV.RND+2]

**:     CM: SAVE RESULT
        AR: R10B=R11A           BR: CJN(1) [ACC.SAV]

**:     CM: SAVE FCC'S / RESTORE CC'S
```

```
        AR: R6B=PSR,OUT=R0A     PC: UW
        CC: C=CPO,N=CPO,Z=CPO,V=CPO
        BR: CBN(1) [BGN]

        PG:     LDCDF/LDCFD INSTRUCTIONS

**: LDCDF
        CM: POINT TO ACCUMULATOR
        AR: R11B=[AC0.FPP]      BR: CBN(1) [.+4]

**:     AR: R11B=[AC1.FPP]      BR: CBN(1) [.+3]

**:     AR: R11B=[AC2.FPP]      BR: CBN(1) [.+2]

**:     AR: R11B=[AC3.FPP]      BR: CBN(1) [.+1]

**:     CM: SAVE CC'S / SET UP LENGTH FLAG
        AR: R0B=PSR,D=R16A      PC: UW

**:     CM: GET SOURCE USING SPECIFIED LENGTH
        AR: OUT=/D              PC: LB
        CC: C=0,Z=1,V=PN,N=0    BR: CJN(1) [FSRC.SAV]

**:     CM: SET UP END BRANCH
        AR: SCE5=[.+3]          SP: CLR,SE5

**:     CM: CHECK VARIABLE AND SET FCC'S
        AR: R1B=[+14]           PC: UW
        BR: CJN(C) [FIUV.FPP+1]

**:     CM: SET FCC'S
        AR: R6B=PSR AND R1A     PC: UW
        BR: CBIN(1) <SCR>

**:     CM: SET UP END BRANCH
        AR: SCE5=[BGN]          SP: CLR,SE5

**:     CM: DO CONVERSION ROUND
        AR: R16B                PC: UB
        CC: V=PN,C=C,N=N,Z=Z    BR: CJN(1) [CNV.RND]

**:     CM: ON ZERO EXPONENT, (AC) = EXACT ZERO
        AR: OUT=R0A-R0B-</Z>    PC: UW
        CC: C=PN,N=N,Z=Z,V=V

**:     CM: SAVE IN ACCUMULATOR
        AR: R10B=R11A           BR: CJN(1) [ACC.SAV]

**:     CM: RESTORE CC'S
        AR: R0B=PSR,OUT=R0A     PC: UW
        CC: C=CPO,N=CPO,Z=CPO,V=CPO

**:     CM: FINALIZE FCC'S / FINISHED / TRAP
        AR: R6B=R6B OR R0A      PC: UW
        BR: CBIN(1) <SCR>

        PG:     FIUV - UNDEFINED VARIABLE CHECK

CM:     ROUTINE CHECKS FOR -0 (IE. Z=1 AND N=1)
        AND CHECKS IF FIUV INTERRUPT IS ENABLED
        IF -0 FOUND AND FIUV=0 THEN
                C=1 FOR CLEAN ZERO
        IF -0 FOUND AND FIUV=1 THEN
                C=0 FOR NON CLEAN ZERO
                GO CHECK FID AND SET UP TRAP
        ELSE RETURNS

**: FIUV.FPP
        CM: RETURN IF NOT ZERO
        BR: CRN(/Z)

**:     CM: RETURN IF PLUS / TEST FIUV BIT
        AR: D=R16A AND [+4000]  PC: UW
        BR: CRN(/N)

**:     CM: SET FOR CLEAN ZERO AND RETURN IF FIUV BIT CLEAR
```

```
        AR: D=D
        CC: C=PZ,N=N,Z=Z,V=V    BR: CRN(Z')

**: FIUV.TRP
        CM: ELSE INHIBIT CLEAN ZERO WHEN TRAPPING
            AND GO CHECK FID AND SET UP TRAP
        AR: D=[+14]             PC: LW,WIOL
        BR: CBN(1) [FPP.TRP]

        PG:     EXPONENT UNDERFLOW CHECK

CM:     ROUTINE CHECKS FOR EXPONENT UNDERFLOW (IE. R4 (UW) <= 0)
        AND CHECKS IF FIU INTERRUPT IS ENABLED
        IF UNDERFLOW AND FIU=0
                THEN MASK EXPONENT, SETUP CLEAN ZERO
        IF UNDERFLOW AND FIU=1
                THEN MASK EXPONENT, INHIBIT CLEAN ZERO,
                GO CHECK FID AND SET UP TRAP
        ELSE CHECKS OVERFLOW

**: EXP.UN
        CM: CLEAR FCC'S
        AR: R6B=0               PC: UW

**:     CM: CHECK EXPONENT
        AR: R4B=R4B+<0>         PC: UW

**:     CM: BRANCH IF NO UNDERFLOW
        AR: R16A AND [+2000]    PC: UW
        BR: CBN(BGT') [EXP.OV]

**:     CM: MASK EXPONENT
        AR: R4B=R4A AND [+377]  PC: UW
        BR: CBN(/Z') [.+2]

**:     CM: SETUP FOR A CLEAN ZERO ON FIU=0
        AR: R4B=0               PC: UW
        CC: C=1,Z=Z,N=N,V=V     BR: CRR(1)

**: FIU.TRP
        CM: SET FEC FOR UNDERFLOW
        AR: D=[+12]             PC: LW,WIOL

**:     CM: INHIBIT CLEAN ZERO AND GO SET UP TRAP
        CC: C=0,Z=Z,N=N,V=V     BR: CBN(1) [FPP.TRP]

        PG: EXPONENT OVERFLOW CHECK

CM:     ROUTINE CHECKS FOR EXPONENT OVERFLOW
        AND CHECKS IF FIV INTERRUPT IS ENABLED
        IF OVERFLOW AND FIV=0
                THEN MASK EXPONENT AND SET FOR CLEAN ZERO
        IF OVERFLOW AND FIV=1
                THEN MASK EXPONENT, INHIBIT CLEAN ZERO,
                GO CHECK FID AND SET UP TRAP
        ELSE RETURNS

**: EXP.OV
        CM: CHECK FOR OVERFLOW
        AR: R4A-[+400]-<1>      PC: UW

**:     CM: RETURN IF NO OVERFLOW
        AR: R16A AND [+1000]    PC: UW
        BR: CRN(N')

**:     CM: ELSE MASK EXPONENT
        AR: R4B=R4A AND [+377]  PC: UW
        BR: CBN(/Z') [.+2]

**:     CM: SETUP FOR CLEAN ZERO
        AR: R4B=0               PC: UW
        CC: C=1,Z=Z,N=N,V=V     BR: CBN(1) [.+2]

**: FIV.TRP
        CM: SET FEC FOR OVERFLOW
        AR: D=[+10]             PC: LW,WIOL
```

```
        BR: CJN(1) [FIU.TRP+1]

**:     CM: SET FCC'S (V=1 ON OVERFLOW)
        AR: R6B=[+2]            PC: UW
        BR: CRR(1)

        PG:     FLOATING TRAP PROCESSING

**: FIC.TRP
        CM: SET FEC FOR INTEGER CONVERSION ERROR
        AR: D=[+6]              PC: LW,WIOL

**: FPP.TRP
        CM: SAVE FEC CODE
        AR: AD=[FEC.FPP]        IO: DATO(OFF)

**:     CM: SET FER BIT IN FPS
        AR: R16B=R16A OR [+100000]      PC: UW

**:     CM: SAVE INSTRUCTION LOCATION
        AR: D=SCR1              PC: LW,WIOL

**:     CM: CHECK INTERRUPT DISABLE FID
        AR: R16A AND [+40000]   PC: UW

**:     CM: SAVE FEA / RETURN IF INTERRUPT DISABLED
        AR: AD=[FEA.FPP]        PC: LW,WIOL
        IO: DATO(OFF)           BR: CRN(/Z')

**:     CM: ELSE SET UP TRAP
        AR: SCE5=[TRAP.FPP]     SP: CLR,SE5
        BR: CRR(1)

**: TRAP.FPP
        CM: RESTORE CC'S
        AR: OUT=R0B             PC: UW
        CC: C=CPO,N=CPO,Z=CPO,V=CPO
        BR: CBN(1) [FIS.TRAP+3]

        NA: MORGEND=.
```

```
        PG:     SPCL INSTRUCTIONS

        CM: DEFINE SPCL ENTRY POINT

        .=: IORIGIN+172

**:     MA: SPCL        PS: EX          IN: BYT



        CM:     SPECIAL INSTRUCTION TO ACCESS MICROCODE ROUTINES

        220-223   MICJMP - JUMP TO MICROCODE LOCATION

        THE GENERAL REGISTER R0-R3 CONTAINS THE ABSOLUTE
        MICROCODE ADDRESS TO WHICH CONTROL IS PASSED

        CM: SPECIAL INSTRUCTIONS TO ACCESS ANY MEMORY LOCATION

        224     GETW - GET WORD OF DATA
        225     GETB - GET BYTE OF DATA
        226     PUTW - PUT WORD OF DATA
        227     PUTB - PUT BYTE OF DATA

        SOURCE/DESTINATION DATA IN R0
        LOW 16 BIT ADDRESS IN R1
        HIGH 6 BIT ADDRESS IN R2

        .=: MORGEND

**: SPCL
        CM: GET IR / LOAD AD
        AR: R10B=IR,AD=R1A

**:     CM: CHECK IF MICJMP
        AR: R10A AND [+4]

**:     CM: CHECK IF PUTWB/GETWB / BRANCH IF MICJMP
        AR: R10A AND [+2]       BR: CBN(Z') [MICJMP]

**:     CM: CHECK WORD/BYTE / BRANCH IF PUT
        AR: R10A AND [+1]       BR: CBN(/Z') [PUT.WB]

**: GET.WB
        CM: LOAD ADX / BRANCH IF BYTE
        AR: ADX=R2A             BR: CBN(/Z') [GETB]

**: GETW
        CM: GET WORD INSTRUCTION
        IO: IDATI(OFF)

**:     AR: R0B=D,AD=R7A        PC: LW,WIOH
        CC: C=C,V=0,N=PN,Z=PZ
        IO: INTCK               BR: CBN(1) [BGN+3]

**: GETB
        CM: GET BYTE INSTRUCTION
        IO: IDATI(OFF,BYT)

**:     AR: R0B=DSX7,AD=R7A     PC: LW,WIOH
        CC: C=C,V=0,N=PN,Z=PZ
        IO: INTCK               BR: CBN(1) [BGN+3]

**: PUT.WB
        CM: LOAD ADX / BRANCH IF BYTE
        AR: ADX=R2A             BR: CBN(/Z') [PUTB]

**: PUTW
        CM: PUT WORD INSTRUCTION
        AR: D=R0B       CC: C=C,V=0,N=PN,Z=PZ
        IO: DATO(OFF)   BR: CBN(1) [BGN]

**: PUTB
        CM: PUT BYTE INSTRUCTION
        AR: D=R0B       PC: LB
        CC: C=C,V=0,N=PN,Z=PZ
```

```
        IO: DATO(OFF,BYT)       BR: CBN(1) [BGN]

**: MICJMP
        AR: SCE5=RA(DST)        SP: CLR,SE5

**:     BR: CBIN(1) <SCR>

        NA: MORGEND=.
```

```
        EN::
```

BLUCPU MICROCODE MICROASSEMBLER V02 15-Oct-2007 16:18:43

TABLE OF CONTENTS

TABLE OF CONTENTS

```
        TT:     BLUCPU MICROCODE

        SB:     TITLE PAGE

        CM:     EMULATION OF THE PDP 11/34 COMPUTER
                CODE DATED MARCH 1980
                ALAN R. BALDWIN

        CM:     CORRECTION TO MTPS (USER MODE) - MAY 1980
        CM:     CORRECTION TO MTOUT (MAINTENANCE MODE) - MAY 1980
        CM:     CORRECTION OF RTI/RTT INSTRUCTIONS - JULY 1980
        CM:     REWRITE OF TRAP HANDLERS - JULY 1980
        CM:     ADDITION OF RESERVED INSTRUCTION HANDLER - JULY 1980
        CM:     ADDITION OF SPL INSTRUCTION - JULY 1980
        CM:     CORRECTION OF RSRVINST/RSRVSTAT - MAY 1986

        CM:     PROGRAM MICROCODE ORIGIN
        NA:             MORIGIN = +6000

        CM:     INSTRUCTION CODE ORIGIN
        NA:             IORIGIN = +6000
```

```
        PG:     TRAP PROCESSING

        .=: MORIGIN+0

        CM: TRAP SEQUENCE ENTRY POINT
            RESERVED INSTRUCTIONS
            TRAP, EMT, IOT, BPT INSTRUCTIONS

6000 M      112337    007000    030440    020200    160060    100000
**: RES
        AR: AD;R10B=PV  PC: LW, WIOL
        IO: DATI(KI)

6001 M      150315    107070    000002    000200    000060    076061
**:     AR: R10B=R10A-[+2]-<1>  BR: CJN(1) [PULL+2]

6002 M      000100    000000    030440    000200    141260    076006
**: PUSH
        IO: IDATO(CMI)          BR: CJN(1) [PUSHA]

6003 M      002134    000067    030440    020200    171720    140000
**: RESEND
        AR: AD=R7A      PC: LW, WIOL
        IO: INTCK       BR:  CBIC(0)

6004 M      000142    000000    030440    100200    141460    100100
**:     AR: SCR0=0      PC: LW, CFP
        IO: IDATIR(CMI)

6005 M      152305    006767    000002    020200    171760    046072
**:     AR: AD;R7B=R7A+[+2]+<0> PC: LW, WIOL
        IO: INTCK       BR: CBR(1) [BGN+2]

6006 M      152105    000066    000002    020200    040060    100000
**: PUSHA
        AR: AD=R6A+[+2]+<0>     PC: LW,WIOL
```

```
6007 M      001134    000071    030440    000200    140230    130000
**:     AR: D=R11A        IO: DATO(CMI)
        BR: CRN(MMGT)

6010 M      137115    100066    030440    000200    000060    120000
**:     AR: SOR=R6A-SLR-<1>
        BR: CRR(1)

6011 M      150315    106767    000002    000200    000060    056000
**: YSTB
        CM: ENTRY POINT FOR YELLOW STACK AND T BIT TRAPS
        AR: R7B=R7A-[+2]-<1>
        BR: CBN(1) [RES]

6012 M      112337    007000    030440    020200    160060    076057
**: PF
        CM: POWER FAIL ENTRY POINT
        AR: AD;R10B=PV  PC: LW, WIOL
        IO: DATI(KI)    BR: CJN(1) [PULL]
```

BLUCPU MICROCODE MICROASSEMBLER V02 15-Oct-2007 16:18:43 PAGE 2 +

TRAP PROCESSING

```
6013 M      000100    000000    030440    000200    000063    006002
**:     BR: BROC(/PFD) [PUSH]
```

```
        PG:     INTERRUPT HANDLERS

6014 M      112337    007000    030440    020200    160060    101000
**: INT17
        CM: ENTRY POINT FOR INTERNAL
        CM: INTERRUPT LEVELS 1 THROUGH 7
        AR: AD;R10B=PV  PC: LW, WIOL
        IO: DATI(KI)    SP: CI

6015 M      150315    106767    000002    000200    000060    056001
**:     AR: R7B=R7A-[+2]-<1>    BR: CBN(1) [RES+1]

6016 M      150337    007000    000002    020200    171660    100000
**: BINT
        AR: R10B=[+2]   PC: LW, WIOL
        IO: IINT

6017 M      002305    007070    030440    010200    160060    056001
**:     AR: AD;R10B=R10A+D+<0>  PC: LW,WIOH
        IO: DATI(KI)    BR: CBN(1) [RES+1]
```

```
        PG:     SOURCE MODE SEQUENCE

        .=: MORIGIN+20
        CM: SOURCE MODE CODE
            FOR ALL MODES RESULTS ARE
            (1)  D & R10 CONTAIN DATA FROM ADDRESS
            (2)  STACK OVERFLOW CHECKED IN MODES  4 AND 5
            UPON ENTRY AND EXIT  AD = R7

6020 M      001134    000000    030440    000200    000060    046045
**: SRC
        CM: (R) IS OPERAND
        AR: D=RA(SRC)   BR:  CBR(1) [SEND]

6021 M      002134    000000    030440    020200    142060    046045
**:     CM: (R) IS ADDRESS
        AR: AD=RA(SRC)  PC: LW, WIOL
        IO: DATI(CMI,BYT)       BR: CBR(1) [SEND]

6022 M      152205    160000    000001    020200    142060    046045
**:      CM: (R) IS ADDRESS; (R) + (1 OR 2)
        AR: RB(SRC)=RA(SRC)+[+1]+</SR67W>, AD=RA(SRC)   PC: LW, WIOL
        IO: DATI(CMI,BYT)       BR: CBR(1) [SEND]

6023 M      152205    000000    000002    020200    140060    046041
**:     CM: (R) IS ADDRESS OF ADDRESS; (R) + 2
        AR: RB(SRC)=RA(SRC)+[+2]+<0>, AD=RA(SRC)        PC: LW, WIOL
        IO: DATI(CMI)   BR: CBR(1) [OS3]

6024 M      152315    060000    000001    020200    142030    006044
**:     CM: (R) - (1 OR 2); (R) IS ADDRESS
        AR: AD;RB(SRC)=RA(SRC)-[+1]-<SR67W>     PC: LW, WIOL
        IO: DATI(CMI,BYT)       BR: BROC(MMGT) [OS4]

6025 M      152315    100000    000002    020200    140060    046043
**:     CM: (R) - 2; (R) IS ADDRESS OF ADDRESS
```

```
        IO: DATI(CMI)   BR: CBR(1) [OS5]


6026 M      150305    006767    000002    020200    141060    046042
**:     CM: (R) + X  IS ADDRESS
        AR: R7B=R7A+[+2]+<0>    PC: LW, WIOL
        IO: IDATI(CMI)  BR: CBR(1) [OS6]

6027 M      150305    006767    000002    020200    141060    046040
**:     CM: (R) + X  IS ADDRESS OF ADDRESS
        AR: R7B=R7A+[+2]+<0>    PC: LW, WIOL
        IO: IDATI(CMI)  BR: CBR(1) [OS7]
```

```
        PG:     DESTINATION MODE SEQUENCE

        .=: MORIGIN+30
        CM: DESTINATION MODE CODE
            FOR ALL MODES
            UPON ENTRY
            (1) AD = R7
            (2) R10 CONTAINS DATA OF PRECEEDING SOURCE MODE
            UPON EXIT
            (1) AD & R11 CONTAIN ADDRESS OF DESTINATION DATA
            (2) D CONTAINS DATA AT DESTINATION ADDRESS
            (3) STACK OVERFLOW CHECKED IN MODES 4 & 5

6030 M      001237    007020    030440    000200    000060    166000
**: DST
        CM: (R) IS OPERAND
        AR: R10B=D, D=RA(DST)   BR: CBIN(1) <INST>

6031 M      002334    007120    030440    020200    106160    166000
**:     CM: (R) IS ADDRESS
        AR: AD;R11B=RA(DST)     PC: LW, WIOL
        IO: DATIP(CM,BYT,NDSP)  BR: CBIN(1) <INST>

6032 M      002334    007120    030440    020200    106160    046047
**:     CM: (R) IS ADDRESS; (R) + (1 OR 2)
        AR: AD;R11B=RA(DST)     PC: LW, WIOL
        IO: DATIP(CM,BYT,NDSP)  BR: CBR(1) [OD2]

6033 M      152205    002020    000002    020200    140060    046051
**:     CM: (R) IS ADDRESS OF ADDRESS; (R) + 2
        AR: RB(DST)=RA(DST)+[+2]+<0>, AD=RA(DST)        PC: LW, WIOL
        IO: DATI(CMI)   BR: CBR(1) [OD3]

6034 M      152315    072020    000001    020200    106160    046052
**:     CM: (R) - (1 OR 2); (R) IS ADDRESS
        AR: AD;RB(DST)=RA(DST)-[+1]-<DR67W>     PC: LW, WIOL
```

```
6035 M      152315    102020    000002    020200    140060    046054
**:     CM: (R) - 2; (R) IS ADDRESS OF ADDRESS
        AR: AD;RB(DST)=RA(DST)-[+2]-<1> PC: LW, WIOL
        IO: DATI(CMI)   BR: CBR(1) [OD5]

6036 M      150305    006767    000002    020200    141060    046056
**:     CM: (R) + X  IS ADDRESS
        AR: R7B=R7A+[+2]+<0>    PC: LW, WIOL
        IO: IDATI(CMI)  BR: CBR(1) [OD6]

6037 M      150305    006767    000002    020200    141060    046050
**:     CM: (R) + X  IS ADDRESS OF ADDRESS
        AR: R7B=R7A+[+2]+<0>    PC: LW, WIOL
        IO: IDATI(CMI)  BR: CBR(1) [OD7]
```

```
        PG:     SOURCE & DESTINATION COMPLETION

6040 M      002105    000000    030440    010200    140060    100000
**: OS7
        CM: COMPLETION OF SOURCE MODES
        AR: AD=RA(SRC)+D+<0>  PC: LW, WIOH      IO: DATI(CMI)

6041 M      002137    000000    030440    010200    142060    046045
**: OS3
        AR: AD=D        PC: LW, WIOH
        IO: DATI(CMI,BYT)       BR: CBR(1) [SEND]

6042 M      002105    000000    030440    010200    142060    046045
**: OS6
        AR: AD=RA(SRC)+D+<0>
        PC: LW, WIOH    IO: DATI(CMI,BYT)       BR: CBR(1) [SEND]

6043 M      002137    000000    030440    010200    142030    056045
**: OS5
        AR: AD=D        PC: LW, WIOH
        IO: DATI(CMI,BYT)       BR: CBN(MMGT) [.+2]

6044 M      137115    100066    030440    000200    000060    100000
**: OS4
        AR: SOR=R6A-SLR-<1>

6045 M      002237    007067    030440    010200    171762    166000
**: SEND
        AR: R10B=D, AD=R7A      PC: LW, WIOH
        IO: INTCK       BR: CBIN(/LKDC) <INST>

6046 M      000100    000000    030440    000200    141560    146000
**:     IO: ILDATIR(CMI)        BR: CBIC(1) <INST>

6047 M      150305    172020    000001    000200    000060    166000
**: OD2
```

```
        BR: CBIN(1) <INST>

6050 M      002105    000020    030440    010200    140060    100000
**: OD7
        AR: AD=RA(DST)+D+<0>    PC: LW, WIOH
        IO: DATI(CMI)

6051 M      002337    007100    030440    010200    106160    166000
**: OD3
        AR: AD;R11B=D   PC: LW, WIOH
        IO: DATIP(CM,BYT,NDSP)  BR: CBIN(1) <INST>

6052 M      000334    007120    030440    000200    000030    166000
**: OD4
        AR: R11B=RA(DST)        BR: CBIN(MMGT) <INST>

6053 M      137115    100066    030440    000200    000060    146000
**:     AR: SOR=R6A-SLR-<1>     BR: CBIC(1) <INST>

6054 M      002337    007100    030440    010200    106130    166000
**: OD5
```

SOURCE & DESTINATION COMPLETION

```
        AR: AD;R11B=D   PC: LW, WIOH
        IO: DATIP(CM,BYT,NDSP)  BR: CBIN(MMGT) <INST>

6055 M      137115    100066    030440    000200    000060    146000
**:     AR: SOR=R6A-SLR-<1>     BR: CBIC(1) <INST>

6056 M      002305    007120    030440    010200    106160    166000
**: OD6
        AR: AD;R11B=RA(DST)+D+<0>       PC: LW, WIOH
        IO: DATIP(CM,BYT,NDSP)  BR: CBIN(1) <INST>
```

```
        PG:     PULL VECTOR HANDLER

6057 M      150315    106767    000002    000200    000060    100000
**: PULL
        CM: TRAP HANDLER PULL ROUTINE
        AR: R7B=R7A-[+2]-<1>

6060 M      150315    107070    000002    000200    000060    100000
**:     AR: R10B=R10A-[+2]-<1>

6061 M      002237    007070    030440    010200    160060    100000
**:     AR: R10B=D,AD=R10A      PC: LW,WIOH
        IO: DATI(KI)

6062 M      124237    007170    063146    020200    000060    100000
**:     AR: R11B=PSR,PSR=R10A   PC: LW, WIOL
        CC: C=CPO, V=CPO, Z=CPO, N=CPO

6063 M      001237    006767    030440    000200    000031    056065
**:     AR: R7B=D, D=R7A        BR: CBN(CUM) [.+2]

6064 M      152315    106666    000004    000200    000060    120000
**:     AR: AD;R6B=R6A-[+4]-<1> BR: CRR(1)

6065 M      152315    107666    000004    000200    000060    120000
**:     AR: AD;R16B=R6A-[+4]-<1>        BR: CRR(1)

6066 M      000100    000000    030440    000200    000033    006070
**: NOBR
        CM: NO BRANCH
        BR: BROC(LKD) [BGN]

6067 M      002301    006770    030440    000200    171760    046073
**: BR
        CM: BRANCH INSTRUCTION
        AR: AD;R7B=R7B+R10A+<0>
```

```
        PG:     INSTRUCTION LOAD & DECODE

        .=: MORIGIN+70

6070 M      152105    000067    000000    020200    171760    046073
**: BGN
        AR: AD=R7A+[+0]+<0>     PC: LW, WIOL
        IO: INTCK       BR: CBR(1) [.+3]

6071 M      152305    006767    000002    020200    171760    100200
**:     AR: AD;R7B=R7A+[+2]+<0> PC: LW, WIOL
        IO: INTCK       SP: ILE

6072 M      060737    007000    030440    000200    141560    166000
**:     AR: R10B=ISX7, SRU(0)
        IO: ILDATIR(CMI)        BR: CBIN(1) <INST>

6073 M      000100    000000    030440    020200    141460    046071
**:     IO: IDATIR(CMI) PC: LW, WIOL
        BR: CBR(1) [BGN+1]

6074 M      150315    106767    000002    000200    000035    056016
**: INT0
        CM: ENTRY POINT FOR BUS INTERRUPTS
        CM: AND LEVEL 0 INTERNAL INTERRUPTS
        AR: R7B=R7A-[+2]-<1>    BR: CBN(BINT) [BINT]

6075 M      150305    006767    000002    000200    000036    056014
**:     AR: R7B=R7A+[+2]+<0>    BR: CBN(EIL) [INT17]

6076 M      152315    106767    000002    000200    171760    046073
**:     AR: AD;R7B=R7A-[+2]-<1>
        IO: INTCK       BR: CBR(1) [BGN+3]

6077 M      000100    000000    030440    000200    000060    046000
**: FBHINT
```

```
            TIME OUT, MEMORY MANAGEMENT, AND ODD ADDRESS TRAPS
            REQUIRE ONE CYCLE DELAY FOR HARDWARE TIMING AFTER 'FBH' CYCLE
        BR: CBR(1) [RES]
```

```
        PG:     CONSOLE SWITCH HANDLER

        .=: MORIGIN+100

6100 M      120237    007272    063146    000200    000002    026100
**: EX1
        CM: RESTORE CC'S, BRANCH TO DESIGNATED EXAM
        AR: R12B=PSR, OUT=R12A  CC: C=CPO, Z=CPO, V=CPO, N=CPO
        BR: CBZV(Z) [.]

6101 M      003134    000077    030440    000700    170060    046151
**:     CM: MOVE ADDRESS TO ADX, START DATI, BRANCH
        AR: ADX=R17A    PC: UW, CB
        IO: DATI(OFF)   BR: CBR(1) [END+1]

6102 M      001134    000040    030440    000600    000060    046151
**:     CM: MOVE REGISTER ADDRESSED BY D INTO D (R20-R37)
        AR: D=RA(D)  PC: UW     BR: CBR(1) [END+1]

6103 M      001134    000040    030440    000200    000060    046151
**:     CM: MOVE REGISTER ADDRESSD BY D INTO D (R0-R17)
        AR: D=RA(D)     PC: LW  BR: CBR(1) [END+1]

6104 M      120237    007272    063146    000200    000002    026104
**: DEP1
        CM: RESTORE CC'S, BRANCH TO DEP MODE
        AR: R12B=PSR, OUT=R12A  CC: C=CPO, Z=CPO, V=CPO, N=CPO
        BR: CBZV(Z) [.]

6105 M      161137    000000    030440    000300    170260    046151
**:     CM: PUT DATA IN D, DO A DATO, BRANCH
        AR: D=SW1       PC: LW, CB
        IO: DATO(OFF)   BR: CBR(1) [END+1]

6106 M      001334    004075    030440    000600    000060    046151
**:     CM: STORE DATA (R15) IN REG POINTED AT BY D
```

```
6107 M      001334    004075    030440    000200    000060    046151
**:     CM: STORE DATA (R15) IN REG POINTED AT BY D
        AR: D;RB(D)=R15A        PC: LW  BR: CBR(1) [END+1]

6110 M      152315    106767    000002    000200    000060    046112
**: CHLT
        CM: CONSOLE INTERRUPT ENTRY POINT
        AR: AD;R7B=R7A-[+2]-<1>
        BR: CBR(1) [CONS]

6111 M      000100    000000    030440    000200    000031    056000
**: HALT
        CM: HALT INSTRUCTION ENTRY POINT
        BR: CBN(CUM) [RES]

6112 M      051137    000000    030440    000200    000060    100000
**: CONS
        CM: PUT IR IN D
        AR: D=IR        PC: LW
```

CONSOLE SWITCH HANDLER

```
6113 M      150337    007400    174000    000200    000060    100000
**:     CM:  MASK WORD FOR CONTROL SWITCHES
        AR: R14B=[+174000]

6114 M      000342    007200    030440    000200    000060    100100
**:     CM:  INITIALIZE CONSOLE CONDITION CODES
        AR: SCR0;R12B=0

6115 M      170345    007574    030440    000300    000052    100000
**: GO
        CM:  MASK SWITCH DATA & TEST FOR ACTION
        AR: R15B=SW2 AND R14A   BR: CNR(/Z')    PC: LW, CB

6116 M      170145    000074    030440    000300    000012    100000
**:     CM:  CHECK SWITCH FOR RELEASE
        AR: SW2 AND R14A        BR: CNR(Z')     PC: LW, CB

6117 M      120237    007272    063146    000200    000060    066154
**:     CM: EXCHANGE PROGRAM & ROUTINE CC'S, TEST FOR REG
        AR: R12B=PSR, OUT=R12A  CC: C=CPO, Z=CPO, V=CPO, N=CPO
        BR: CJR(1) [TSTR]

6120 M      000733    007500    030440    000200    000060    100000
**:     CM: TEST SIGN AND SHIFT DATA
        AR: R15B=R15B, SRU(0)

6121 M      000733    007500    030440    000200    000053    056124
**: LAD
        CM: TEST & SHIFT, BRANCH IF NOT LAD
        AR: R15B=R15B, SRU(0)   BR: CBN(/N') [EX]

6122 M      162337    007700    160456    000200    000060    100000
**:     CM: SET CODES FOR NEW ADDRESS
        AR: AD;R17B=SW1 CC: C=0, N=0, Z=Z, V=V
```

```
**:     CM: MOVE UPPER ADDRESS TO R17 & AD
        AR: ADX;R17B=SW2        PC: UW  BR: CBR(1) [END]

6124 M      000733    007500    030440    000200    000053    056130
**: EX
        CM: TEST SIGN & SHIFT, IF NOT EX BRANCH
        AR: R15B=R15B, SRU(0)   BR: CBN(/N') [CONT]

6125 M      000100    000000    170456    000200    000000    076152
**:     CM: SET CODES, JSR TO INCR IF SECOND TIME
        CC: C=1, N=0, V=V, Z=Z  BR: CJN(C) [INCR]

6126 M      002134    000077    030440    000200    000060    100000
**:     CM: MOVE ADDRESS INTO AD
        AR: AD=R17A

6127 M      001134    000077    030440    000200    000060    046100
**:     CM: MOVE ADDRESS INTO D
        AR: D=R17A      BR: CBR(1) [EX1]

6130 M      000733    007500    030440    000200    000053    056135
```

```
CONSOLE SWITCH HANDLER


**: CONT
        CM: SHIFT & TEST SIGN, IF NOT CONT BRANCH
        AR: R15B=R15B, SRU(0)   BR: CBN(/N') [STRT]

6131 M      002134    000067    030440    100200    000060    100200
**:     CM: MOVE PC INTO AD, ENABLE INTERRUPT LOGIC
        AR: AD=R7A      SP: ILE PC: LW, CFP

6132 M      003134    000067    030440    000600    000060    100000
**:     CM: MOVE PCX INTO ADX
        AR: ADX=R7A     PC: UW

6133 M      120237    007272    063146    000300    140460    100000
**:     CM: RESTORE CC'S, LOAD NEXT INSTRUCTION
        AR: R12B=PSR, OUT=R12A  CC: C=CPO, Z=CPO, V=CPO, N=CPO
        PC: LW, CB      IO: DATIR(CMI)

6134 M      000100    000000    030440    020300    000060    046071
**:     CM: WAIT FOR IO COMPLETE
        PC: LW, WIOL, CB        BR: CBR(1) [BGN+1]

6135 M      000733    007500    030440    000200    000053    056143
**: STRT
        CM: SHIFT & TEST SIGN, BRANCH IF NOT STRT
        AR: R15B=R15B, SRU(0)   BR: CBN(/N') [DEP]

6136 M      155137    000000    177767    100200    040060    100400
**:     CM: LOAD CNTR, SEQUENCE TO CLEAR PRIORITIES
        AR: CNTR=[+177767]
        SP: ILD, CLR    PC: LW, CFP

6137 M      004142    000000    167356    040200    000020    160000
**:     CM: SEQUENCE PRIORITIES
        AR: PSR=0       CC: C=0, Z=0, V=0, N=0
        PC: LW, CCL     BR: CBIN(0)
```

```
6140 M      000100    000000    030440    000200    000037    056137
**:     CM: REPEAT UNTIL SEQUENCE COMPLETE
        BR: CBN(CNTR) [.-1]

6141 M      002334    006777    030440    100200    170760    100200
**:     CM: LOAD NEW PC, DO A BUS INIT, ENABLE INTERRUPT LOGIC
        AR: AD;R7B=R17A IO: INIT        SP: ILE PC: LW, CFP

6142 M      003334    006777    030440    020600    140460    046071
**:     CM: LOAD NEW PCX, LOAD NEXT INSTRUCTION
        AR: ADX;R7B=R17A        PC: UW, WIOL
        IO: DATIR(CMI)  BR: CBR(1) [BGN+1]

6143 M      000100    000000    030440    000200    000053    056150
**: DEP
        CM: BRANCH IF NOT DEP (ERROR)
        BR: CBN(/N') [END]

6144 M      000100    000000    160457    000200    000003    076152
**:     CM: SET CODES, INCR IF SECOND TIME
        CC: C=0, N=1, Z=Z, V=V  BR: CJN(N) [INCR]
```

CONSOLE SWITCH HANDLER

```
6145 M      162237    007577    030440    000200    000060    100000
**:     CM: PUT ADDRESS IN AD, PUT DATA IN R15
        AR: R15B=SW1, AD=R17A   PC: LW

6146 M      163237    007577    030440    000600    000060    100000
**:     CM: PUT ADDRESS IN ADX, PUT DATA IN R15
        AR: R15B=SW1, ADX=R17A  PC: UW

6147 M      001134    000077    030440    000200    000060    046104
**:     CM: MOVE ADDRESS INTO D
        AR: D=R17A      BR: CBR(1) [DEP1]

6150 M      120237    007272    063146    000200    000060    100000
**: END
        CM: RESTORE CC'S
        AR: R12B=PSR, OUT=R12A  CC: C=CPO, Z=CPO, V=CPO, N=CPO

6151 M      170345    007574    030440    010300    000060    046115
**:     CM: WAIT FOR IO COMPLETE, LOOK AT SWITCHES, BRANCH
        AR: R15B=SW2 AND R14A   PC: LW, WIOH, CB        BR: CBR(1) [GO]

6152 M      000303    107700    030440    001400    000002    056154
**: INCR
        CM: INCREMENT ADDRESS BY 1, SKIP NEXT IF REG
        AR: R17B=R17B+<1>       PC: 3B  BR: CBN(Z) [.+2]

6153 M      000303    107700    030440    001400    000060    100000
**:     CM: INCREMENT ADDRESS BY 1
        AR: R17B=R17B+<1>       PC: 3B

6154 M      150337    007300    000020    000200    000060    066156
**: TSTR
        CM: MOVE TESTF BIT INTO R13, GO TEST FOR REG
        AR: R13B=[+20]  BR: CJR(1) [REG]
```

```
6155 M      000141    007377    030620    000200    000060    120000
**:     CM: TEST FOR UPPER/LOWER REGISTER
        AR: R13B AND R17A       BR: CRR(1)
        CC: V=PZ, C=C, Z=Z, N=N

6156 M      150115    100077    000077    000400    000060    100000
**: REG
        CM: TEST ADX FOR REGISTER ADDRESS
        AR: R17A-[+77]-<1>      PC: UB

6157 M      150115    100077    177700    000200    000052    056162
**:     CM: COMPARE ADDRESS TO LOW BOUND OF REG ADDRESS
        AR: R17A-[+177700]-<1>  PC: LW  BR: CBN(/Z') [NO]

6160 M      150125    100077    177737    000200    000013    056162
**:     CM: COMPARE ADDRESS TO UPPER BOUNDS OF REG ADDRESS
        AR: [+177737]-R17A-<1>  PC: LW  BR: CBN(N') [NO]

6161 M      000100    000000    037440    000200    000053    130000
**:     CM: REGISTER ADDRESS
        CC: Z=1, C=C, V=V, N=N  BR: CRN(/N')
```

CONSOLE SWITCH HANDLER

6162 M 000100 000000 037040 000200 000060 120000

**: NO

CM: NOT REGISTER ADDRESS

CC: Z=0, C=C, V=V, N=N BR: CRR(1)

```
        PG:     DOUBLE OPERAND INSTRUCTIONS

        CM: DOUBLE OPERAND INSTRUCTIONS
            D,S,MODE IS LABLED AS INSTRUCTION
            THE THREE REMAINING MODES D,S0 D0,S & D0,S0
            ARE ACCESSED BY OFFSETS

6163 M      001134    000070    034750    020200    100260    046070
**: MOV
        CM: MOVE INSTRUCTION
        AR: D=R10A      PC: LW, WIOL    IO: DATO(CM)
        CC: N=PN, Z=PZ, V=0, C=C        BR: CBR(1) [BGN]

6164 M      001134    000000    034750    020200    100260    046070
**:     AR: D=RA(SRC)   PC: LW, WIOL    IO: DATO(CM)
        CC: N=PN, Z=PZ, V=0, C=C        BR: CBR(1) [BGN]

6165 M      000337    002000    034750    000200    000033    006070
**:     AR: RB(DST)=D
        CC: Z=PZ, N=PN, V=0, C=C        BR: BROC(LKD) [BGN]

6166 M      000334    002000    034750    000200    000033    006070
**:     AR: RB(DST)=RA(SRC)
        CC: N=PN, Z=PZ, V=0, C=C        BR: BROC(LKD) [BGN]

6167 M      001134    000070    034750    020000    100360    046070
**: MOVB
        CM: MOVE BYTE INSTRUCTION
        AR: D=R10A      PC: LB, WIOL    IO: DATOB(CM)
        CC: N=PN, Z=PZ, V=0, C=C        BR: CBR(1) [BGN]

6170 M      001134    000000    034750    020000    100360    046070
**:     AR: D=RA(SRC)   PC: LB, WIOL    IO: DATOB(CM)
        CC: N=PN, Z=PZ, V=0, C=C        BR: CBR(1) [BGN]

6171 M      030337    002000    034750    000200    000033    006070
```

```
        CC: N=PN, Z=PZ, V=0, C=C        BR: BROC(LKD) [BGN]

6172 M      001134    000000    030440    000000    000060    046171
**:     AR: D=RA(SRC)   PC: LB  BR: CBR(1) [.-1]

6173 M      000115    100070    014650    011000    000060    046070
**: CMP
        CM: COMPARE INSTRUCTION
        AR: R10A-D-<1>  PC: LWB, WIOH
        CC: N=PN, Z=PZ, V=PV, C=PCB     BR: CBR(1) [BGN]

6174 M      000115    100000    014650    011000    000060    046070
**:     AR: RA(SRC)-D-<1>       PC: LWB, WIOH
        CC: N=PN, Z=PZ, V=PV, C=PCB     BR: CBR(1) [BGN]

6175 M      000125    100020    014650    001000    000033    006070
**:     AR: D-RA(DST)-<1>       PC: LWB
        CC: N=PN, Z=PZ, V=PV, C=PCB     BR: BROC(LKD) [BGN]

6176 M      000111    100020    014650    001000    000033    006070
**:     AR: RB(SRC)-RA(DST)-<1> PC: LWB
```

DOUBLE OPERAND INSTRUCTIONS

```
        CC: N=PN, Z=PZ, V=PV, C=PCB     BR: BROC(LKD) [BGN]

6177 M      000145    000070    034750    011000    000060    046070
**: BIT
        CM: BIT TEST INSTRUCTION
        AR: D AND R10A  PC: LWB, WIOH
        CC: N=PN, Z=PZ, V=0, C=C        BR: CBR(1) [BGN]

6200 M      000145    000000    034750    011000    000060    046070
**:     AR: D AND RA(SRC)       PC: LWB, WIOH
        CC: N=PN, Z=PZ, V=0, C=C        BR: CBR(1) [BGN]

6201 M      000145    000020    034750    001000    000033    006070
**:     AR: D AND RA(DST)       PC: LWB
        CC: N=PN, Z=PZ, V=0, C=C        BR: BROC(LKD) [BGN]

6202 M      000141    000020    034750    001000    000033    006070
**:     AR: RB(SRC) AND RA(DST) PC: LWB
        CC: N=PN, Z=PZ, V=0, C=C        BR: BROC(LKD) [BGN]

6203 M      001237    007070    030440    011000    000060    046207
**: BIC
        CM: BIT CLEAR INSTRUCTION
        AR: R10B=D, D=R10A      PC: LWB, WIOH
        BR: CBR(1) [.+4]

6204 M      001237    007000    030440    011000    000060    046207
**:     AR: R10B=D, D=RA(SRC)   PC: LWB, WIOH
        BR: CBR(1) [.+3]

6205 M      000355    002020    034750    001000    000033    006070
**:     AR: RB(DST)=/D AND RA(DST)      PC: LWB
        CC: N=PN, Z=PZ, V=0, C=C        BR: BROC(LKD) [BGN]

6206 M      000351    002000    034750    001000    000033    006070
```

```
        CC: N=PN, Z=PZ, V=0, C=C        BR: BROC(LKD) [BGN]

6207 M      001155    000070    034750    011000    102260    046070
**:     AR: D=/D AND R10A       PC: LWB, WIOH   IO: DATO(CM,BYT)
        CC: N=PN, Z=PZ, V=0, C=C        BR: CBR(1) [BGN]

6210 M      001135    000070    034750    011000    102260    046070
**: BIS
        CM: BIT SET INSTRUCTION
        AR: D=R10A OR D PC: LWB, WIOH   IO: DATO(CM,BYT)
        CC: N=PN, Z=PZ, V=0, C=C        BR: CBR(1) [BGN]

6211 M      001135    000000    034750    011000    102260    046070
**:     AR: D=D OR RA(SRC)      PC: LWB, WIOH IO: DATO(CM,BYT)
        CC: N=PN, Z=PZ, V=0, C=C        BR: CBR(1) [BGN]

6212 M      000335    002020    034750    001000    000033    006070
**:     AR: RB(DST)=D OR RA(DST)        PC: LWB
        CC: N=PN, Z=PZ, V=0, C=C        BR: BROC(LKD) [BGN]

6213 M      000331    002000    034750    001000    000033    006070
```

DOUBLE OPERAND INSTRUCTIONS

```
**:     AR: RB(DST)=RB(DST) OR RA(SRC)  PC: LWB
        CC: N=PN, Z=PZ, V=0, C=C        BR: BROC(LKD) [BGN]

6214 M      001105    000070    134650    010200    100260    046070
**: ADD
        CM: ADD INSTRUCTION
        AR: D=D+R10A+<0>        PC: LW, WIOH    IO: DATO(CM)
        CC: N=PN, Z=PZ, V=PV, C=PC      BR: CBR(1) [BGN]

6215 M      001105    000000    134650    010200    100260    046070
**:     AR: D=D+RA(SRC)+<0>     PC: LW, WIOH    IO: DATO(CM)
        CC: N=PN, Z=PZ, V=PV, C=PC      BR: CBR(1) [BGN]

6216 M      000305    002020    134650    000200    000033    006070
**:     AR: RB(DST)=D+RA(DST)+<0>
        CC: N=PN, Z=PZ, V=PV, C=PC      BR: BROC(LKD) [BGN]

6217 M      000301    002000    134650    000200    000033    006070
**:     AR: RB(DST)=RB(DST)+RA(SRC)+<0>
        CC: N=PN, Z=PZ, V=PV, C=PC      BR: BROC(LKD) [BGN]

6220 M      001125    100070    014650    010200    100260    046070
**: SUB
        CM: SUBTRACT INSTRUCTION
        AR: D=D-R10A-<1>        PC: LW, WIOH    IO: DATO(CM)
        CC: N=PN, Z=PZ, V=PV, C=PCB     BR: CBR(1) [BGN]

6221 M      001125    100000    014650    010200    100260    046070
**:     AR: D=D-RA(SRC)-<1>     PC: LW, WIOH    IO: DATO(CM)
        CC: N=PN, Z=PZ, V=PV, C=PCB     BR: CBR(1) [BGN]

6222 M      000315    102020    014650    000200    000033    006070
**:     AR: RB(DST)=RA(DST)-D-<1>
        CC: N=PN, Z=PZ, V=PV, C=PCB     BR: BROC(LKD) [BGN]
```

```
**:     AR: RB(DST)=RB(DST)-RA(SRC)-<1>

        CC: N=PN, Z=PZ, V=PV, C=PCB     BR: BROC(LKD) [BGN]
```

```
        PG:     SINGLE OPERAND INSTRUCTIONS

        CM: SINGLE OPERAND INSTRUCTIONS
            D  MODE IS LABLED AS INSTRUCTION
            D0  MODE IS ACCESSED BY AN OFFSET

6224 M      001142    000000    167756    021000    102260    046070
**: CLR
        CM: CLEAR INSTRUCTION
        AR: D=0 PC: LWB, WIOL   IO: DATO(CM,BYT)
        CC: Z=1, N=0, V=0, C=0  BR: CBR(1) [BGN]

6225 M      000342    002000    167756    001000    000033    006070
**:     AR: RB(DST)=0   PC: LWB
        CC: Z=1, N=0, V=0, C=0  BR: BROC(LKD) [BGN]

6226 M      001177    000000    174750    011000    102260    046070
**: COM
        CM: COMPLEMENT INSTRUCTION
        AR: D=/D        PC: LWB, WIOH   IO: DATO(CM,BYT)
        CC: N=PN, Z=PZ, V=0, C=1        BR: CBR(1) [BGN]

6227 M      000373    002000    174750    001000    000033    006070
**:     AR: RB(DST)=/RB(DST)    PC: LWB
        CC: N=PN, Z=PZ, V=0, C=1        BR: BROC(LKD) [BGN]

6230 M      001107    100000    034650    011000    102260    046070
**: INC
        CM: INCREMENT INSTRUCTION
        AR: D=D+<1>     PC: LWB, WIOH   IO: DATO(CM,BYT)
        CC: N=PN, Z=PZ, V=PV, C=C       BR: CBR(1) [BGN]

6231 M      000303    102000    034650    001000    000033    006070
**:     AR: RB(DST)=RB(DST)+<1> PC: LWB
        CC: N=PN, Z=PZ, V=PV, C=C       BR: BROC(LKD) [BGN]
```

```
**: DEC
        CM: DECREMENT INSTRUCTION
        AR: D=D-<0>     PC: LWB, WIOH   IO: DATO(CM,BYT)
        CC: N=PN, Z=PZ, V=PV, C=C       BR: CBR(1) [BGN]

6233 M      000313    002000    034650    001000    000033    006070
**:     AR: RB(DST)=RB(DST)-<0> PC: LWB
        CC: N=PN, Z=PZ, V=PV, C=C       BR: BROC(LKD) [BGN]

6234 M      001117    100000    074650    011000    102260    046070
**: NEG
        CM: NEGATE INSTRUCTION
        AR: D=-D-<1>    PC: LWB, WIOH   IO: DATO(CM,BYT)
        CC: N=PN, Z=PZ, V=PV, C=PZB     BR: CBR(1) [BGN]

6235 M      000323    102000    074650    001000    000033    006070
**:     AR: RB(DST)=-RB(DST)-<1>        PC: LWB
        CC: N=PN, Z=PZ, V=PV, C=PZB     BR: BROC(LKD) [BGN]

6236 M      001107    010000    134650    011000    102260    046070
**: ADC
```

SINGLE OPERAND INSTRUCTIONS

```
        CM: ADD CARRY INSTRUCTION
        AR: D=D+<C>     PC: LWB, WIOH   IO: DATO(CM,BYT)
        CC: N=PN, Z=PZ, V=PV, C=PC      BR: CBR(1) [BGN]

6237 M      000303    012000    134650    001000    000033    006070
**:     AR: RB(DST)=RB(DST)+<C> PC: LWB
        CC: N=PN, Z=PZ, V=PV, C=PC      BR: BROC(LKD) [BGN]

6240 M      001127    110000    014650    011000    102260    046070
**: SBC
        CM: SUBTRACT CARRY INSTRUCTION
        AR: D=D-</C>    PC: LWB, WIOH   IO: DATO(CM,BYT)
        CC: N=PN, Z=PZ, V=PV, C=PCB     BR: CBR(1) [BGN]

6241 M      000313    112000    014650    001000    000033    006070
**:     AR: RB(DST)=RB(DST)-</C>        PC: LWB
        CC: N=PN, Z=PZ, V=PV, C=PCB     BR: BROC(LKD) [BGN]

6242 M      002133    006700    030440    020200    171760    046244
**: TST
        CM: TEST INSTRUCTION
        AR: AD=R7B      PC: LW, WIOL    IO: INTCK
        BR: CBR(1) [.+2]

6243 M      000134    000020    164750    001000    000033    006070
**:     AR: RA(DST)     PC: LWB
        CC: N=PN, Z=PZ, V=0, C=0        BR: BROC(LKD) [BGN]

6244 M      000137    000000    164750    001000    141460    046071
**:     AR: D   PC: LWB IO: IDATIR(CMI)
        CC: N=PN, Z=PZ, V=0, C=0        BR: CBR(1) [BGN+1]

6245 M      011137    000000    164750    010000    100260    046070
**: SWAB
        CM: SWAP BYTE INSTRUCTION
```

```
        CC: N=PN, Z=PZ, V=0, C=0        BR: CBR(1) [BGN]

6246 M      001134    000020    030440    000200    000060    100000
**:     AR: D=RA(DST)

6247 M      010337    002000    030440    000200    000060    100000
**:     AR: RB(DST)=DSWB

6250 M      000134    000020    164750    000000    000033    006070
**:     AR: RA(DST)     PC: LB
        CC: N=PN, Z=PZ, V=0, C=0        BR: BROC(LKD) [BGN]

6251 M      001111    146060    034750    020200    100260    046070
**: SXT
        CM: SIGN EXTEND INSTRUCTION
        AR: D=R0B-R0A-</N>      PC: LW, WIOL    IO: DATO(CM)
        CC: N=PN, Z=PZ, V=0, C=C        BR: CBR(1) [BGN]

6252 M      000311    142020    034750    000200    000060    046071
**:     AR: RB(DST)=RB(DST)-RA(DST)-</N>
        CC: N=PN, Z=PZ, V=0, C=C        BR: CBR(1) [BGN+1]
```

SINGLE OPERAND INSTRUCTIONS

```
6253 M      001165    000000    034750    010200    100260    046070
**: XOR
        CM: EXCLUSIVE OR INSTRUCTION
        AR: D=D XOR RA(SRC)     PC: LW, WIOH    IO: DATO(CM)
        CC: N=PN, Z=PZ, V=0, C=C        BR: CBR(1) [BGN]

6254 M      000361    002000    034750    000200    000033    006070
**:     AR: RB(DST)=RB(DST) XOR RA(SRC)
        CC: N=PN, Z=PZ, V=0, C=C        BR: BROC(LKD) [BGN]

6255 M      000537    007000    000440    011004    000060    046257
**: ROR
        CM: ROTATE RIGHT INSTRUCTION
        AR: R10B=D, SRD(C)      PC: LWB, WIOH
        CC: C=LSBR, N=N, Z=Z, V=V       BR: CBR(1) [SRD]

6256 M      000533    002000    000440    001004    000060    046260
**:     AR: RB(DST)=RB(DST), SRD(C)     PC: LWB
        CC: C=LSBR, N=N, Z=Z, V=V       BR: CBR(1) [SR0]

6257 M      001133    007000    034710    001000    102260    046070
**: SRD
        AR: D=R10B      PC: LWB IO: DATO(CM,BYT)
        CC: C=C, N=PN, Z=PZ, V=PNXC     BR: CBR(1) [BGN]

6260 M      000133    002000    034710    001000    000033    006070
**: SR0
        AR: RB(DST)     PC: LWB
        CC: C=C, N=PN, Z=PZ, V=PNXC     BR: BROC(LKD) [BGN]

6261 M      000737    007000    100440    011004    000060    046257
**: ROL
        CM: ROTATE LEFT INSTRUCTION
        AR: R10B=D, SRU(C)      PC: LWB, WIOH
```

```
6262 M      000733    002000    100440    001004    000060    046260
**:     AR: RB(DST)=RB(DST), SRU(C)     PC: LWB
        CC: C=MSBR, N=N, Z=Z, V=V       BR: CBR(1) [SR0]

6263 M      000537    007000    000440    011002    000060    046257
**: ASR
        CM: ARITHMETIC SHIFT RIGHT INSTRUCTION
        AR: R10B=D, SRD(MSBR)   PC: LWB, WIOH
        CC: C=LSBR, N=N, Z=Z, V=V       BR: CBR(1) [SRD]

6264 M      000533    002000    000440    001002    000060    046260
**:     AR: RB(DST)=RB(DST), SRD(MSBR)  PC: LWB
        CC: C=LSBR, N=N, Z=Z, V=V       BR: CBR(1) [SR0]

6265 M      000737    007000    100440    011000    000060    046257
**: ASL
        CM: ARITHMETIC SHIFT LEFT INSTRUCTION
        AR: R10B=D, SRU(0)      PC: LWB, WIOH
        CC: C=MSBR, N=N, Z=Z, V=V       BR: CBR(1) [SRD]
```

SINGLE OPERAND INSTRUCTIONS

```
6266 M      000733    002000    100440    001000    000060    046260
**:     AR: RB(DST)=RB(DST), SRU(0)     PC: LWB
        CC: C=MSBR, N=N, Z=Z, V=V       BR: CBR(1) [SR0]
```

```
        PG:     CONTROL, TRAP, AND OTHER INSTRUCTIONS

6267 M      000100    000000    052525    000200    000033    006070
**: CLRC
        CM: CLEAR CONDITION CODE INSTRUCTIONS
        CC: C=CLR, N=CLR, Z=CLR, V=CLR  BR: BROC(LKD) [BGN]

6270 M      000100    000000    042104    000200    000033    006070
**: SETC
        CM: SET CONDITION CODE INSTRUCTIONS
        CC: C=SET, N=SET, Z=SET, V=SET  BR: BROC(LKD) [BGN]

6271 M      002213    000067    030440    000200    000060    100000
**: SOB
        CM: SUBTRACT 1 AND BRANCH IF NOT = 0
        AR: RB(SRC)=RB(SRC)-<0>, AD=R7A

6272 M      070737    007000    030440    000200    171712    056073
**:     AR: R10B=IR5, SRU(0)
        IO: INTCK       BR: CBN(Z') [BGN+3]

6273 M      002311    106770    030440    000200    140460    046071
**:     AR: AD;R7B=R7B-R10A-<1>
        IO: DATIR(CMI)  BR: CBR(1) [BGN+1]

6274 M      070707    107000    030440    000200    000060    100000
**: MARK
        CM: MARK INSTRUCTION
        AR: R10B=IR5+<1>, SRU(0)

6275 M      002301    007066    030440    000200    140031    056277
**:     AR: AD;R10B=R6A+R10B+<0>
        IO: DATI(CMI)   BR: CBN(CUM) [.+2]

6276 M      150305    006670    000002    000200    000060    046300
**:     AR: R6B=R10A+[+2]+<0>   BR: CBR(1) [.+2]
```

```
6277 M      150305    007670    000002    000200    000060    100000
**:     AR: R16B=R10A+[+2]+<0>

6300 M      002334    006765    030440    020200    171760    100000
**:     AR: AD;R7B=R5A  PC: LW, WIOL    IO: INTCK

6301 M      000337    006500    030440    000200    141460    046071
**:     AR: R5B=D       IO: IDATIR(CMI) BR: CBR(1) [BGN+1]

6302 M      152205    007066    000002    020200    140031    056304
**: RTS
        CM: RETURN FROM SUBRUTINE INSTRUCTION
        AR: R10B=R6A+[+2]+<0>, AD=R6A   PC: LW, WIOL
        IO: DATI(CMI)   BR: CBN(CUM) [.+2]

6303 M      000334    006670    030440    000200    000060    046305
**:     AR: R6B=R10A    BR: CBR(1) [.+2]

6304 M      000334    007670    030440    000200    000060    100000
**:     AR: R16B=R10A
```

CONTROL, TRAP, AND OTHER INSTRUCTIONS

```
6305 M      000334    006720    030440    000200    000060    100000
**:     AR: R7B=RA(DST)

6306 M      000337    002000    030440    010200    000060    046070
**:     AR: RB(DST)=D   PC: LW, WIOH    BR: CBR(1) [BGN]

6307 M      152205    007066    000002    020200    140031    056311
**: RTI
        CM: RTI/RTT INSTRUCTIONS
        AR: R10B=R6A+[+2]+<0>,AD=R6A    PC: LW,WIOL
        IO: DATI(CMI)   BR: CBN(CUM) [.+2]

6310 M      150305    006666    000004    000200    000060    046312
**:     AR: R6B=R6A+[+4]+<0>    BR: CBR(1) [.+2]

6311 M      150305    007666    000004    000200    000060    100000
**:     AR: R16B=R6A+[+4]+<0>

6312 M      002237    006770    030440    010200    140060    100000
**:     AR: R7B=D,AD=R10A       PC: LW,WIOH
        IO: DATI(CMI)

6313 M      004137    000000    063146    010200    000031    056070
**:     CM: SET T BIT & CC'S / DONE IF IN USER MODE
        AR: PSR=D       CC: C=CPO,Z=CPO,V=CPO,N=CPO
        PC: LW,WIOH     BR: CBN(CUM) [BGN]

6314 M      152137    000000    177776    000200    170260    046070
**:     CM: ELSE RESTORE ALL STATUS FROM STACK
        AR: AD=[+177776]        IO: DATO(OFF)
        BR: CBR(1) [BGN]

6315 M      000100    000000    030440    020200    171731    056073
**: RESET
        CM: RESET INSTRUCTION
```

```
6316 M      000100    000000    030440    020200    170760    046073
**:     PC: LW, WIOL    IO: INIT        BR: CBR(1) [BGN+3]

6317 M      000100    000000    030440    000200    171725    100000
**: WAIT
        CM: WAIT FOR INTERRUPT INSTRUCTION
        IO: INTCK       BR: CNR(IP)

6320 M      150305    006767    000002    000200    000060    166000
**:     AR: R7B=R7A+[+2]+<0>    BR: CBIN(1) <INST>

6321 M      152337    007000    000006    020200    160060    056001
**: JPR0
        CM: JMP/JSR (0) TRAP
        AR: AD;R10B=[+6]        PC: LW, WIOL
        IO: DATI(KI)    BR: CBN(1) [RES+1]

6322 M      001134    000000    030440    000200    000031    056324
**: JSR
        CM: JUMP TO SUBRUTINE INSTRUCTION
```

CONTROL, TRAP, AND OTHER INSTRUCTIONS

```
        AR: D=RA(SRC)   BR: CBN(CUM) [.+2]

6323 M      152315    106666    000002    020200    140260    046325
**:     AR: AD;R6B=R6A-[+2]-<1> PC: LW, WIOL
        IO: DATO(CMI)   BR: CBR(1) [.+2]

6324 M      152315    107666    000002    020200    140260    100000
**:     AR: AD;R16B=R6A-[+2]-<1>        PC: LW, WIOL
        IO: DATO(CMI)

6325 M      000334    000067    030440    000200    000060    100000
**:     AR: RB(SRC)=R7A

6326 M      002334    006771    030440    020200    171760    046073
**: JMP
        CM: JUMP INSTRUCTION
        AR: AD;R7B=R11A PC: LW, WIOL
        IO: INTCK       BR: CBR(1) [BGN+3]
```

```
        PG:     RESERVED INSTRUCTION TRAP HANDLER

6327 M      152137    000000    177764    000200    170060    100000
**: RSRVINST
        CM: GET STATUS
        AR: AD=[+177764]        IO: DATI(OFF)

6330 M      000137    000000    030440    010200    040460    100120
**:     CM: USE STATE 'EX'/'E5' FOR ADDITIONAL CODE
        AR: SCE5=D              PC: LW,WIOH
        SP: CLR,SE5

6331 M      000100    000000    030440    000200    000013    174000
**: RSRVSTAT
        CM: GO TO ADDITIONAL CODE IF DEFINED
        BR: CJIN(N') <SCR>

6332 M      150137    000000    006000    000200    040160    100100
**:     CM: ELSE NOT DEFINED
        AR: SCE7=[RES]
        SP: CLR,SE7

6333 M      000100    000000    030440    000200    000060    164000
**:     CM: RESET CPU STATE TO 'RES'/'E7'
        BR: CBIN(1) <SCR>
```

```
        PG:     SPL INSTRUCTION (11/45, 60, & 70)

6334 M      000301    047070    030440    000200    000031    056070
**: SPL
        CM: SET PRIORITY LEVEL
        AR: R10B=R10B+R10A+<N>
        BR: CBN(CUM) [BGN]

6335 M      000301    037070    030440    000200    000060    100000
**:     AR: R10B=R10B+R10A+<Z>

6336 M      000301    027070    030440    000200    000060    100000
**:     AR: R10B=R10B+R10A+<V>

6337 M      001101    017070    030440    000200    000060    100000
**:     AR: D=R10B+R10A+<C>

6340 M      152137    000000    177776    000200    170360    046070
**:     AR: AD=[+177776]        IO: DATOB(OFF)
        BR: CBR(1) [BGN]
```

```
        PG:     PROCESOR STATUS WORD INSTRUCTIONS


6341 M      000137    000000    063146    010200    000031    056070
**: MTPS
        CM: MOVE TO PS / DONE IF USER MODE
        AR: D           CC: C=CPO,V=CPO,Z=CPO,N=CPO
        PC: LW, WIOH    BR: CBN(CUM) [BGN]

6342 M      152137    000000    177776    000200    170360    046070
**:     CM: ELSE IN KERNEL MODE - CHANGE PRIORITY TOO
        AR: AD=[+177776]
        IO: DATOB(OFF)  BR: CBR(1) [BGN]

6343 M      121137    000000    030440    000200    000060    046171
**: MFPS0
        CM: MOVE FROM PS INSTRUCTION
        AR: D=PSR       BR: CBR(1) [MOVB+2]

6344 M      121137    000000    034750    020000    100360    046070
**: MFPSX
        AR: D=PSR       CC: C=C, V=0, N=PN, Z=PZ
        PC: LB, WIOL    IO: DATOB(CM)   BR: CBR(1) [BGN]
```

```
        PG:     MEMORY MANAGEMENT INSTRUCTIONS

        .=: MORIGIN+347

6347 M      100177    000000    030440    000200    000060    100100
**: MTPIX
        AR: SCR0=/SCR0

6350 M      001134    000070    030440    000200    000052    056374
**:     AR: D=R10A      BR: CBN(/Z') [GTSTK]

6351 M      000100    000000    030440    000200    111260    100000
**:     IO: IDATO(PM)

6352 M      002237    007067    034750    020200    171760    046073
**: MEND
        AR: R10B=D, AD=R7A      CC: C=C, V=0, N=PN, Z=PZ
        IO: INTCK       BR: CBR(1) [BGN+3]      PC: LW, WIOL

6353 M      000100    000000    030440    000200    151031    006362
**: MFPIX
        CM: MFPI INSTRUCTION
        IO: IDATI(PMI)  BR: BROC(CUM) [MFOUT]

6354 M      150337    007000    000106    000200    000060    100000
**: MFPI0
        AR: R10B=[+106]

6355 M      050115    100070    030440    000000    000032    056357
**:     AR: R10A-IR-<1> PC: LB  BR: CBN(PUM) [.+2]

6356 M      001133    006600    030440    000200    000012    006360
**:     AR: D=R6B       BR: BROC(Z') [MFOUT-2]

6357 M      001133    007600    030440    000200    000012    006360
**:     AR: D=R16B      BR: BROC(Z') [MFOUT-2]
```

```
6360 M      001134    000020    030440    000200    000031    006362
**:     AR: D=RA(DST)   BR: BROC(CUM) [MFOUT]

6361 M      000100    000000    030440    000200    000031    056363
**:     BR: CBN(CUM) [.+2]

6362 M      152315    106666    000002    020200    100260    046352
**: MFOUT
        AR: AD;R6B=R6A-[+2]-<1> PC: LW, WIOL
        IO: DATO(CM)    BR: CBR(1) [MEND]

6363 M      152315    107666    000002    020200    100260    046352
**:     AR: AD;R16B=R6A-[+2]-<1>        PC: LW, WIOL
        IO: DATO(CM)    BR: CBR(1) [MEND]

6364 M      152237    007066    000206    000200    100060    100000
**: MTPI0
        CM: MTPI INSTRUCTION
        AR: R10B=[+206], AD=R6A IO: DATI(CM)
6365 M      050115    100070    030440    000000    000031    056367
**:     AR: R10A-IR-<1> PC: LB  BR: CBN(CUM) [.+2]
```

MEMORY MANAGEMENT INSTRUCTIONS

```
6366 M      150305    006666    000002    000200    000012    006370
**:     AR: R6B=R6A+[+2]+<0>    BR: BROC(Z') [MTOUT]

6367 M      150305    007666    000002    000200    000012    006370
**:     AR: R16B=R6A+[+2]+<0>   BR: BROC(Z') [MTOUT]

6370 M      000337    002000    034750    010200    000060    046070
**: MTOUT
        AR: RB(DST)=D   CC: C=C, V=0, N=PN, Z=PZ
        PC: LW, WIOH    BR: CBR(1) [BGN]

6371 M      002134    000067    030440    020200    171732    056373
**:     AR: AD=R7A      PC: LW, WIOL
        IO: INTCK       BR: CBN(PUM) [.+2]

6372 M      000337    006600    034750    000200    141460    046071
**:     AR: R6B=D       CC: C=C, V=0, N=PN, Z=PZ
        IO: IDATIR(CMI) BR: CBR(1) [BGN+1]

6373 M      000337    007600    034750    000200    141460    046071
**:     AR: R16B=D      CC: C=C, V=0, N=PN, Z=PZ
        IO: IDATIR(CMI) BR: CBR(1) [BGN+1]

6374 M      002134    000066    030440    020200    140031    056376
**: GTSTK
        AR: AD=R6A      PC: LW, WIOL
        IO: DATI(CMI)   BR: CBN(CUM) [.+2]

6375 M      150305    006666    000002    000200    002460    046377
**:     AR: R6B=R6A+[+2]+<0>    SP: SE3, SE5
        BR: CBR(1) [.+2]

6376 M      150305    007666    000002    000200    002460    100000
**:     AR: R16B=R6A+[+2]+<0>   SP: SE3, SE5
```

```
6377 M      002237    007067    030440    010200    000060    166000
**:     AR: R10B=D, AD=R7A      PC: LW, WIOH
        BR: CBIN(1) <INST>

        NA: MORGEND=.
```

```
        PG:     INSTRUCTION CODE SPECIFICATIONS

        .=: IORIGIN+0

6000 I      006327    000400
**:     MA: RSRVINST    PS: RES

6001 I      006327    000400
**:     MA: RSRVINST    PS: RES

6002 I      006327    000400
**:     MA: RSRVINST    PS: RES

6003 I      006327    000400
**:     MA: RSRVINST    PS: RES

6004 I      006327    000400
**:     MA: RSRVINST    PS: RES

6005 I      006327    000400
**:     MA: RSRVINST    PS: RES

6006 I      006327    000400
**:     MA: RSRVINST    PS: RES

6007 I      006327    000400
**:     MA: RSRVINST    PS: RES

6010 I      006327    000400
**:     MA: RSRVINST    PS: RES

6011 I      006327    000400
**:     MA: RSRVINST    PS: RES

6012 I      006327    000400
**:     MA: RSRVINST    PS: RES
```

```
6013 I      006327    000400
**:     MA: RSRVINST    PS: RES

6014 I      006327    000400
**:     MA: RSRVINST    PS: RES

6015 I      006327    000400
**:     MA: RSRVINST    PS: RES

6016 I      006327    000400
**:     MA: RSRVINST    PS: RES

6017 I      006327    000400
**:     MA: RSRVINST    PS: RES

6020 I      006327    000400
**:     MA: RSRVINST    PS: RES

6021 I      006327    000400
**:     MA: RSRVINST    PS: RES
```

INSTRUCTION CODE SPECIFICATIONS

```
6022 I      006327    000400
**:     MA: RSRVINST    PS: RES

6023 I      006327    000400
**:     MA: RSRVINST    PS: RES

6024 I      006327    000400
**:     MA: RSRVINST    PS: RES

6025 I      006327    000400
**:     MA: RSRVINST    PS: RES

6026 I      006327    000400
**:     MA: RSRVINST    PS: RES

6027 I      006327    000400
**:     MA: RSRVINST    PS: RES

6030 I      006327    000400
**:     MA: RSRVINST    PS: RES

6031 I      006327    000400
**:     MA: RSRVINST    PS: RES

6032 I      006327    000400
**:     MA: RSRVINST    PS: RES

6033 I      006327    000400
**:     MA: RSRVINST    PS: RES

6034 I      006327    000400
**:     MA: RSRVINST    PS: RES

6035 I      006327    000400
**:     MA: RSRVINST    PS: RES
```

```
6036 I      006327    000400
**:     MA: RSRVINST    PS: RES

6037 I      006327    000400
**:     MA: RSRVINST    PS: RES

6040 I      006327    000400
**:     MA: RSRVINST    PS: RES

6041 I      006327    000400
**:     MA: RSRVINST    PS: RES

6042 I      006327    000400
**:     MA: RSRVINST    PS: RES

6043 I      006327    000400
**:     MA: RSRVINST    PS: RES

6044 I      006327    000400
**:     MA: RSRVINST    PS: RES
```

INSTRUCTION CODE SPECIFICATIONS

```
6045 I      006327    000400
**:     MA: RSRVINST    PS: RES

6046 I      006327    000400
**:     MA: RSRVINST    PS: RES

6047 I      006327    000400
**:     MA: RSRVINST    PS: RES

6050 I      006327    000400
**:     MA: RSRVINST    PS: RES

6051 I      006327    000400
**:     MA: RSRVINST    PS: RES

6052 I      006327    000400
**:     MA: RSRVINST    PS: RES

6053 I      006327    000400
**:     MA: RSRVINST    PS: RES

6054 I      006327    000400
**:     MA: RSRVINST    PS: RES

6055 I      006327    000400
**:     MA: RSRVINST    PS: RES

6056 I      006327    000400
**:     MA: RSRVINST    PS: RES

6057 I      006327    000400
**:     MA: RSRVINST    PS: RES

6060 I      006327    000400
**:     MA: RSRVINST    PS: RES
```

```
6061 I      006327    000400
**:     MA: RSRVINST    PS: RES

6062 I      006327    000400
**:     MA: RSRVINST    PS: RES

6063 I      006327    000400
**:     MA: RSRVINST    PS: RES

6064 I      006327    000400
**:     MA: RSRVINST    PS: RES

6065 I      006327    000400
**:     MA: RSRVINST    PS: RES

6066 I      006327    000400
**:     MA: RSRVINST    PS: RES

6067 I      006327    000400
**:     MA: RSRVINST    PS: RES
```

INSTRUCTION CODE SPECIFICATIONS

```
6070 I      006327    000400
**:     MA: RSRVINST    PS: RES

6071 I      006327    000400
**:     MA: RSRVINST    PS: RES

6072 I      006327    000400
**:     MA: RSRVINST    PS: RES

6073 I      006327    000400
**:     MA: RSRVINST    PS: RES

6074 I      006327    000400
**:     MA: RSRVINST    PS: RES

6075 I      006327    000400
**:     MA: RSRVINST    PS: RES

6076 I      006327    000400
**:     MA: RSRVINST    PS: RES

6077 I      006327    000400
**:     MA: RSRVINST    PS: RES

6100 I      006111    000400
**:     MA: HALT        PS: RES

6101 I      006317    002000
**:     MA: WAIT        PS: EX

6102 I      006307    002000
**:     MA: RTI PS: EX

6103 I      006000    000204
**:     MA: RES PS: BPT IN: TI  CM: BPT INSTRUCTION
```

```
6104 I      006000    000210
**:     MA: RES PS: IOT IN: TI  CM: IOT INSTRUCTION

6105 I      006315    002000
**:     MA:  RESET      PS: EX

6106 I      006307    002200
**:     MA: RTI PS: EX  IN: TI  CM: RTT INSTRUCTION

6107 I      006327    000400
**:     MA: RSRVINST    PS: RES

6110 I      006163    052020
**:     MA: MOV         PS: SRC, DST, EX        IN: NDA

6111 I      006164    012020
**:     MA: MOV+1       PS: DST, EX     IN: NDA

6112 I      036165    042000
**:     MA: MOV+2       PS: SRC, EX     IN: C2, WT
```

INSTRUCTION CODE SPECIFICATIONS

```
6113 I      056166    002000
**:     MA: MOV+3       PS: EX  IN: C1, WT

6114 I      006167    052120
**:     MA: MOVB        PS: SRC, DST, EX        IN: NDA, BYT

6115 I      006170    012120
**:     MA: MOVB+1      PS: DST, EX     IN: NDA, BYT

6116 I      036171    042100
**:     MA: MOVB+2      PS: SRC, EX     IN: C2, BYT, WT

6117 I      056172    002100
**:     MA: MOVB+3      PS: EX  IN: C1, BYT, WT

6120 I      006173    052040
**:     MA: CMP         PS: SRC, DST, EX        IN: SP

6121 I      006174    012040
**:     MA: CMP+1       PS: DST, EX     IN: SP

6122 I      036175    042000
**:     MA: CMP+2       PS: SRC, EX     IN: C2, WT

6123 I      116176    002000
**:     MA: CMP+3       PS: EX  IN: FRC, WT

6124 I      006173    052140
**:     MA: CMP         PS: SRC, DST, EX        IN: BYT, SP

6125 I      006174    012140
**:     MA: CMP+1       PS: DST, EX     IN: BYT, SP

6126 I      036175    042100
**:     MA: CMP+2       PS: SRC, EX     IN: C2, BYT, WT
```

```
6127 I      116176    002100
**:     MA: CMP+3       PS: EX  IN: FRC, BYT, WT

6130 I      006177    052040
**:     MA: BIT         PS: SRC, DST, EX        IN: SP

6131 I      006200    012040
**:     MA: BIT+1       PS: DST, EX     IN: SP

6132 I      036201    042000
**:     MA: BIT+2       PS: SRC, EX     IN: C2, WT

6133 I      116202    002000
**:     MA: BIT+3       PS: EX  IN: FRC, WT

6134 I      006177    052140
**:     MA: BIT         PS: SRC, DST, EX        IN: SP, BYT

6135 I      006200    012140
**:     MA: BIT+1       PS: DST, EX     IN: SP, BYT
```

INSTRUCTION CODE SPECIFICATIONS

```
6136 I      036201    042100
**:     MA: BIT+2       PS: SRC, EX     IN: C2, BYT, WT

6137 I      116202    002100
**:     MA: BIT+3       PS: EX  IN: FRC, BYT, WT

6140 I      006203    052000
**:     MA: BIC         PS: SRC, DST, EX

6141 I      006204    012000
**:     MA: BIC+1       PS: DST, EX

6142 I      036205    042000
**:     MA: BIC+2       PS: SRC, EX     IN: C2, WT

6143 I      056206    002000
**:     MA: BIC+3       PS: EX  IN: C1, WT

6144 I      006203    052100
**:     MA: BIC         PS: SRC, DST, EX        IN: BYT

6145 I      006204    012100
**:     MA: BIC+1       PS: DST, EX     IN: BYT

6146 I      036205    042100
**:     MA: BIC+2       PS: SRC, EX     IN: C2, BYT, WT

6147 I      056206    002100
**:     MA: BIC+3       PS: EX  IN: C1, BYT, WT

6150 I      006210    052000
**:     MA: BIS         PS: SRC, DST, EX

6151 I      006211    012000
**:     MA: BIS+1       PS: DST, EX
```

```
6152 I      036212    042000
**:     MA: BIS+2       PS: SRC, EX     IN: C2, WT

6153 I      056213    002000
**:     MA: BIS+3       PS: EX  IN: C1, WT

6154 I      006210    052100
**:     MA: BIS         PS: SRC, DST, EX        IN: BYT

6155 I      006211    012100
**:     MA: BIS+1       PS: DST, EX     IN: BYT

6156 I      036212    042100
**:     MA: BIS+2       PS: SRC, EX     IN: C2, BYT, WT

6157 I      056213    002100
**:     MA: BIS+3       PS: EX  IN: C1, BYT, WT

6160 I      006214    052000
**:     MA: ADD         PS: SRC, DST, EX
```

INSTRUCTION CODE SPECIFICATIONS

```
6161 I      006215    012000
**:     MA: ADD+1       PS: DST, EX

6162 I      036216    042000
**:     MA: ADD+2       PS: SRC, EX     IN: C2, WT

6163 I      056217    002000
**:     MA: ADD+3       PS: EX  IN: C1,WT

6164 I      006220    052000
**:     MA: SUB         PS: SRC, DST, EX

6165 I      006221    012000
**:     MA: SUB+1       PS: DST, EX

6166 I      036222    042000
**:     MA: SUB+2       PS: SRC, EX   IN: C2, WT

6167 I      056223    002000
**:     MA: SUB+3       PS: EX  IN: C1, WT

6170 I      006302    002000
**:     MA: RTS         PS: EX

6171 I      006327    000400
**:     MA: RSRVINST    PS: RES

6172 I      006327    000400
**:     MA: RSRVINST    PS: RES

6173 I      006334    002000
**:     MA: SPL         PS: EX

6174 I      116267    002000
**:     MA: CLRC        PS: EX  IN: FRC, WT
```

```
6175 I      116267    002000
**:     MA: CLRC        PS: EX  IN: FRC, WT

6176 I      116270    002000
**:     MA: SETC        PS: EX  IN: FRC, WT

6177 I      116270    002000
**:     MA: SETC        PS: EX  IN: FRC, WT

6200 I      006224    012020
**:     MA: CLR         PS: DST,EX      IN: NDA

6201 I      056225    002000
**:     MA: CLR+1       PS: EX  IN: C1, WT

6202 I      006224    012120
**:     MA: CLR         PS: DST, EX     IN: BYT, NDA

6203 I      056225    002100
**:     MA: CLR+1       PS: EX  IN: BYT, C1, WT
```

INSTRUCTION CODE SPECIFICATIONS

```
6204 I      006226    012000
**:     MA: COM         PS: DST, EX

6205 I      056227    002000
**:     MA: COM+1       PS: EX  IN: C1, WT

6206 I      006226    012100
**:     MA: COM         PS: DST, EX     IN: BYT

6207 I      056227    002100
**:     MA: COM+1       PS: EX  IN: BYT, C1, WT

6210 I      006230    012000
**:     MA: INC         PS: DST, EX

6211 I      056231    002000
**:     MA: INC+1       PS: EX  IN: C1, WT

6212 I      006230    012100
**:     MA: INC         PS: DST, EX     IN: BYT

6213 I      056231    002100
**:     MA: INC+1       PS: EX  IN: BYT, C1, WT

6214 I      006232    012000
**:     MA: DEC         PS: DST, EX

6215 I      056233    002000
**:     MA: DEC+1       PS: EX  IN: C1, WT

6216 I      006232    012100
**:     MA: DEC         PS: DST, EX     IN: BYT

6217 I      056233    002100
**:     MA: DEC+1       PS: EX  IN: BYT, C1, WT
```

```
6220 I      006234    012000
**:     MA: NEG         PS: DST, EX

6221 I      056235    002000
**:     MA: NEG+1       PS: EX  IN: C1, WT

6222 I      006234    012100
**:     MA: NEG         PS: DST, EX     IN: BYT

6223 I      056235    002100
**:     MA: NEG+1       PS: EX  IN: BYT, C1, WT

6224 I      006236    012000
**:     MA: ADC         PS: DST, EX

6225 I      056237    002000
**:     MA: ADC+1       PS: EX  IN: C1, WT

6226 I      006236    012100
**:     MA: ADC         PS: DST, EX     IN: BYT
```

INSTRUCTION CODE SPECIFICATIONS

```
6227 I      056237    002100
**:     MA: ADC+1       PS: EX  IN: BYT, C1, WT

6230 I      006240    012000
**:     MA: SBC         PS: DST, EX

6231 I      056241    002000
**:     MA: SBC+1       PS: EX  IN: C1, WT

6232 I      006240    012100
**:     MA: SBC         PS: DST, EX     IN: BYT

6233 I      056241    002100
**:     MA: SBC+1       PS: EX  IN: BYT, C1, WT

6234 I      006242    012040
**:     MA: TST         PS: DST, EX     IN: SP

6235 I      116243    002000
**:     MA: TST+1       PS: EX  IN: FRC, WT

6236 I      006242    012140
**:     MA: TST         PS: DST, EX     IN: SP, BYT

6237 I      116243    002100
**:     MA: TST+1       PS: EX  IN: BYT, FRC, WT

6240 I      006255    012000
**:     MA: ROR         PS: DST, EX

6241 I      056256    002000
**:     MA: ROR+1       PS: EX  IN: C1, WT

6242 I      006255    012100
**:     MA: ROR         PS: DST, EX     IN: BYT
```

```
6243 I      056256    002100
**:     MA: ROR+1       PS: EX  IN: BYT, C1, WT

6244 I      006261    012000
**:     MA: ROL         PS: DST, EX

6245 I      056262    002000
**:     MA: ROL+1       PS: EX  IN: C1, WT

6246 I      006261    012100
**:     MA: ROL         PS: DST, EX     IN: BYT

6247 I      056262    002100
**:     MA: ROL+1       PS: EX  IN: BYT, C1, WT

6250 I      006263    012000
**:     MA: ASR         PS: DST, EX

6251 I      056264    002000
**:     MA: ASR+1       PS: EX  IN: C1, WT
```

INSTRUCTION CODE SPECIFICATIONS

```
6252 I      006263    012100
**:     MA: ASR         PS: DST, EX     IN: BYT

6253 I      056264    002100
**:     MA: ASR+1       PS: EX  IN: BYT, C1, WT

6254 I      006265    012000
**:     MA: ASL         PS: DST, EX

6255 I      056266    002000
**:     MA: ASL+1       PS: EX  IN: C1, WT

6256 I      006265    012100
**:     MA: ASL         PS: DST, EX     IN: BYT

6257 I      056266    002100
**:     MA: ASL+1       PS: EX  IN: BYT, C1, WT

6260 I      006274    002000
**:     MA: MARK        PS: EX

6261 I      006274    002000
**:     MA: MARK        PS: EX

6262 I      006341    012140
**:     MA: MTPS        PS: DST, EX     IN: SP, BYT

6263 I      006341    012100
**:     MA: MTPS        PS: DST, EX     IN: BYT

6264 I      006353    012020
**:     MA: MFPIX       PS: DST, EX     IN: NDA

6265 I      006354    002000
**:     MA: MFPI0       PS: EX
```

```
6266 I      006353    012020
**:     MA: MFPIX       PS: DST, EX     IN: NDA CM: MFPD INST.


6267 I      006354    002000
**:     MA: MFPI0       PS: EX  CM: MFPD INST.


6270 I      006347    002020
**:     MA: MTPIX       PS: EX  IN: NDA


6271 I      006364    002000
**:     MA: MTPI0       PS: EX


6272 I      006347    002020
**:     MA: MTPIX       PS: EX  IN: NDA CM: MTPD INST.


6273 I      006364    002000
**:     MA: MTPI0       PS: EX  CM: MTPD INST.


6274 I      006251    012020
**:     MA: SXT         PS: DST, EX     IN: NDA
```

INSTRUCTION CODE SPECIFICATIONS

```
6275 I      056252    002000
**:     MA: SXT+1       PS: EX          IN: C1, WT

6276 I      006344    012020
**:     MA: MFPSX       PS: DST, EX     IN: NDA

6277 I      006343    002000
**:     MA: MFPS0       PS: EX

6300 I      006327    000400
**:     MA: RSRVINST    PS: RES

6301 I      006327    000400
**:     MA: RSRVINST    PS: RES

6302 I      006327    000400
**:     MA: RSRVINST    PS: RES

6303 I      006327    000400
**:     MA: RSRVINST    PS: RES

6304 I      006327    000400
**:     MA: RSRVINST    PS: RES

6305 I      006327    000400
**:     MA: RSRVINST    PS: RES

6306 I      006327    000400
**:     MA: RSRVINST    PS: RES

6307 I      006327    000400
**:     MA: RSRVINST    PS: RES

6310 I      006253    012000
**:     MA: XOR         PS: DST, EX
```

```
6311 I      056254    002000
**:     MA: XOR+1       PS: EX  IN: C1, WT

6312 I      006327    000400
**:     MA: RSRVINST    PS: RES

6313 I      006327    000400
**:     MA: RSRVINST    PS: RES

6314 I      006327    000400
**:     MA: RSRVINST    PS: RES

6315 I      006327    000400
**:     MA: RSRVINST    PS: RES

6316 I      006271    002000
**:     MA: SOB PS: EX

6317 I      006271    002000
**:     MA: SOB PS: EX
```

INSTRUCTION CODE SPECIFICATIONS

```
6320 I      006327    000400
**:     MA: RSRVINST    PS: RES

6321 I      006327    000400
**:     MA: RSRVINST    PS: RES

6322 I      006327    000400
**:     MA: RSRVINST    PS: RES

6323 I      006327    000400
**:     MA: RSRVINST    PS: RES

6324 I      006327    000400
**:     MA: RSRVINST    PS: RES

6325 I      006327    000400
**:     MA: RSRVINST    PS: RES

6326 I      006327    000400
**:     MA: RSRVINST    PS: RES

6327 I      006327    000400
**:     MA: RSRVINST    PS: RES

6330 I      006327    000400
**:     MA: RSRVINST    PS: RES

6331 I      006327    000400
**:     MA: RSRVINST    PS: RES

6332 I      006327    000400
**:     MA: RSRVINST    PS: RES

6333 I      006327    000400
**:     MA: RSRVINST    PS: RES
```

```
6334 I      006327    000400
**:     MA: RSRVINST    PS: RES

6335 I      006327    000400
**:     MA: RSRVINST    PS: RES

6336 I      006327    000400
**:     MA: RSRVINST    PS: RES

6337 I      006327    000400
**:     MA: RSRVINST    PS: RES

6340 I      006327    000400
**:     MA: RSRVINST    PS: RES

6341 I      006327    000400
**:     MA: RSRVINST    PS: RES

6342 I      006327    000400
**:     MA: RSRVINST    PS: RES
```

INSTRUCTION CODE SPECIFICATIONS

```
6343 I      006327    000400
**:     MA: RSRVINST    PS: RES

6344 I      006327    000400
**:     MA: RSRVINST    PS: RES

6345 I      006327    000400
**:     MA: RSRVINST    PS: RES

6346 I      006327    000400
**:     MA: RSRVINST    PS: RES

6347 I      006327    000400
**:     MA: RSRVINST    PS: RES

6350 I      006327    000400
**:     MA: RSRVINST    PS: RES

6351 I      006327    000400
**:     MA: RSRVINST    PS: RES

6352 I      006327    000400
**:     MA: RSRVINST    PS: RES

6353 I      006327    000400
**:     MA: RSRVINST    PS: RES

6354 I      006327    000400
**:     MA: RSRVINST    PS: RES

6355 I      006327    000400
**:     MA: RSRVINST    PS: RES

6356 I      006327    000400
**:     MA: RSRVINST    PS: RES
```

```
6357 I      006327    000400
**:     MA: RSRVINST    PS: RES


6360 I      006326    012020
**:     MA: JMP         PS: DST, EX     IN: NDA


6361 I      006321    000400
**:     MA: JPR0        PS: RES


6362 I      006245    012000
**:     MA: SWAB        PS: DST, EX


6363 I      056246    002000
**:     MA: SWAB+1      PS: EX  IN: C1, WT


6364 I      116066    002000
**:     MA: NOBR        PS: EX  IN: FRC, WT


6365 I      006067    002000
**:     MA: BR          PS: EX
```

INSTRUCTION CODE SPECIFICATIONS

```
6366 I      006322    012020
**:     MA: JSR         PS: DST, EX     IN: NDA

6367 I      006321    000400
**:     MA: JPR0        PS: RES

6370 I      006000    000202
**:     MA: RES PS: EMT         IN: TI  CM: EMT INSTRUCTION

6371 I      006000    000201
**:     MA: RES PS: TRAP        IN: TI  CM: TRAP INSTRUCTION

6372 I      006327    000400
**:     MA: RSRVINST    PS: RES

6373 I      006327    000400
**:     MA: RSRVINST    PS: RES

6374 I      006327    000400
**:     MA: RSRVINST    PS: RES

6375 I      006327    000400
**:     MA: RSRVINST    PS: RES

6376 I      006327    000400
**:     MA: RSRVINST    PS: RES

6377 I      006327    000400
**:     MA: RSRVINST    PS: RES
```

```
        PG:     VECTOR DEFINITIONS


7777 V      000000
        VE: VGRP=+0
0037 V      044037
        VE: V37= CHLT / +37
0036 V      044036
        VE: V36= CHLT / +36
0035 V      044035
        VE: V35= CHLT / +35
0034 V      037406
        VE: V34= FBHINT  / +6
0033 V      044033
        VE: V33= CHLT / +33
0032 V      004406
        VE: V32= YSTB / +6
0031 V      037406
        VE: V31= FBHINT  / +6
0030 V      037652
        VE: V30= FBHINT  / +252
0027 V      044000
        VE: V27= CHLT / +0
0026 V      044000
        VE: V26= CHLT / +0
0025 V      044000
        VE: V25= CHLT / +0
0024 V      044000
        VE: V24= CHLT / +0
0023 V      044000
        VE: V23= CHLT / +0
0022 V      044000
        VE: V22= CHLT / +0
0021 V      044000
        VE: V21= CHLT / +0
0020 V      000012
        VE: V20= RES  / +12
```

```
        VE: V17= RES  / +22
0016 V      000016
        VE: V16= RES  / +16
0015 V      000032
        VE: V15= RES  / +32
0014 V      000036
        VE: V14= RES  / +36
0013 V      044013
        VE: V13= CHLT / +13
0012 V      004416
        VE: V12= YSTB   / +16
0011 V      004406
        VE: V11= YSTB   / +6
0010 V      005026
        VE: V10= PF     / +26
0007 V      006102
        VE: V7 = INT17 / +102
0006 V      006062
        VE: V6 = INT17 / +62
0005 V      006066
        VE: V5 = INT17 / +66
```

VECTOR DEFINITIONS

```
0004 V      006172
        VE: V4 = INT17 / +172
0003 V      006176
        VE: V3 = INT17 / +176
0002 V      006272
        VE: V2 = INT17 / +272
0001 V      006276
        VE: V1 = INT17 / +276
0000 V      036242
        VE: V0 = INT0  / +242
```

```
        PG:     EXTENDED INSTRUCTION SET

        .=: MORGEND
        CM: EXTENDED INSTRUCTION SET (EIS)
            ASH SHIFT ARITHMETICALLY
            ASHC  ARITHMETIC SHIFT COMBINED
            MUL MULTIPLY
            DIV DIVIDE

        MODIFIED MAY 1980 TO USE ONLY R10 & R11 IN 4B MODE

6400 M      045137    000000    167350    010200    000060    100000
**: ASH
        CM: ARITHMETIC SHIFT INSTRUCTION
        CM: LOAD COUNTER, TEST FOR LEFT / RIGHT SHIFT
        AR: CNTR=DSX5   PC: LW, WIOH
        CC: N=PN, C=0, V=0, Z=0

6401 M      001242    007000    030440    000200    000012    056412
**:     CM: SET UP FOR 32 BIT SHIFTING
        AR: R10B=0,D=RA(SRC)    PC: LW
        BR: CBN(Z') [END1]

6402 M      000337    007000    030450    000600    000003    056410
**:     CM: SET UPPER WORD DATA, BRANCH ON RIGHT SHIFT
        AR: R10B=D      PC: UW
        CC: N=PN, C=C, V=V, Z=Z BR: CBN(N) [RTST]

6403 M      000021    146060    030440    001600    000060    100000
**:     CM: EXTEND SIGN OF RB(SRC) THROUGH Q
        AR: Q=R0A-R0B-</N>      PC: 4B

6404 M      000633    007000    107340    041620    000077    100000
**: LTST
        CM: LEFT SHIFTING
        AR: R10B=R10B,SRU(0),SQ(MSBR)   PC: 4B, CCL
```

```
6405 M      000634    007170    030440    001620    000060    100000
**:     CM: SHIFT LAST SIGN BIT INTO Q REG
        AR: R11B=R10A,SRU(0),SQ(MSBR)   PC: 4B

6406 M      000102    040000    070460    001600    000060    100000
**:     CM: DID SIGN OF REGISTER CHANGE
        AR: OUT=Q+<N>   PC: 4B
        CC: C=PZB, V=C, N=N, Z=Z

6407 M      001133    007000    027076    000600    000060    046412
**:     AR: D=R10B      PC: UW
        CC: C=V, V=C, N=0, Z=0  BR: CBR(1) [END1]

6410 M      000533    007000    007356    040602    000077    100000
**: RTST
        CM: RIGHT SHIFTING
        AR: R10B=R10B,SRD(MSBR)         PC: UW, CCL
        CC: C=LSBR, V=0, N=0, Z=0       BR: CNR(/CNTR)

6411 M      001133    007000    030440    000600    000060    100000
```

```
EXTENDED INSTRUCTION SET


**:     AR: D=R10B      PC: UW


6412 M      000337    000000    034450    000200    000060    046070
**: END1
        AR: RB(SRC)=D   PC: LW
        CC: C=C, V=V, N=PN, Z=PZ        BR: CBR(1) [BGN]


6413 M      045137    000000    167350    010200    000060    100000
**: ASHC
        CM: ARITHMETIC SHIFT COMBINED
        CM: TEST FOR LEFT / RIGHT SHIFT
        AR: CNTR=DSX5   PC: LW, WIOH
        CC: N=PN, C=0, V=0, Z=0


6414 M      001133    000000    030440    000200    000012    056427
**:     CM: SET UP FOR 32 BIT SHIFTING
        AR: D=RB(SRC)   PC: LW
        BR: CBN(Z') [END2]


6415 M      000334    007001    030440    000200    000060    100000
**:     AR: R10B=RA(SRC!1)      PC: LW


6416 M      000337    007000    030450    000600    000003    056424
**:     AR: R10B=D              PC: UW
        CC: N=PN, V=V, C=C, Z=Z BR: CBN(N) [RSHC]


6417 M      000021    146060    030440    001600    000060    100000
**:     CM: EXTEND SIGN OF RB(SRC) THROUGH Q REG
        AR: Q=R0A-R0B-</N>      PC: 4B


6420 M      000633    007000    107340    041620    000077    100000
**: LSHC
        CM: LEFT SHIFTING
        AR: R10B=R10B,SRU(0),SQ(MSBR)   PC: 4B, CCL
        CC: C=MSBR, V=0, N=N, Z=0       BR: CNR(/CNTR)
```

```
6421 M      000634    007170    030440    001620    000060    100000
**:     CM: SHIFT LAST SIGN BIT INTO Q REG
        AR: R11B=R10A,SRU(0),SQ(MSBR)   PC: 4B

6422 M      000102    040000    070460    001600    000060    100000
**:     CM: DID SIGN OF REGISTER CHANGE
        AR: OUT=Q+<N>   PC: 4B
        CC: C=PZB, V=C, N=N, Z=Z

6423 M      001133    007000    027076    000600    000060    046426
**:     AR: D=R10B      PC: UW
        CC: V=C, C=V, N=0, Z=0  BR: CBR(1) [END2-1]

6424 M      000533    007000    007356    041602    000077    100000
**: RSHC
        CM: RIGHT SHIFTING
        AR: R10B=R10B,SRD(MSBR)         PC: 4B, CCL
        CC: C=LSBR, V=0, N=0, Z=0       BR: CNR(/CNTR)

6425 M      001133    007000    030440    000600    000060    100000
**:     AR: D=R10B      PC: UW
```

EXTENDED INSTRUCTION SET

```
6426 M      000334    000170    030440    000200    000060    100000
**:     AR: RB(SRC!1)=R10A      PC: LW

6427 M      001237    000001    034450    000200    000060    100000
**: END2
        AR: RB(SRC)=D,D=RA(SRC!1)       PC: LW
        CC: C=C, V=V, N=PN, Z=PZ

6430 M      000337    000100    036040    000200    000060    046070
**:     AR: RB(SRC!1)=D         PC: LW
        CC: C=C, V=V, N=N, Z=PZAZ       BR: CBR(1) [BGN]

6431 M      155137    000000    000017    000200    000060    100000
**: MUL
        CM: INITIALIZE LOOP COUNTER
        AR: CNTR=[+17]

6432 M      000342    007100    030440    000200    000060    100000
**:     CM: INITIALIZE RESULT
        AR: R11B=0

6433 M      000537    007000    000440    010200    000060    100000
**:     CM: LOAD MULTIPLIER
        AR: R10B=D,SRD(0)       PC: LW, WIOH
        CC: C=LSBR, V=V, N=N, Z=Z

6434 M      000033    007000    030440    000200    000060    100000
**:     CM: ENTER MULTIPLIER SHIFTED ONCE
        AR: Q=R10B

6435 M      000401    007100    140440    044265    000077    100000
**:     CM: MULTIPLICATION IS PERFORMED BY
        CM: ADDING THE MULTIPLICAND TO THE RESULT
        CM: IF THE MULTIPLIER BIT SHIFTED FROM Q IS A ONE
```

```
        AR: R11B=R11B+RA(SRC)+<0>,SRD(PNXPV),SQ(LSBR)
        CC: C=LSBQ, V=V, N=N, Z=Z       PC: LW, MM, CCL
        BR: CNR(/CNTR)

6436 M      000411    107100    140440    004265    000060    100000
**:     CM: LAST CYCLE OF TWO'S COMPLEMENT MULTIPLY
        AR: R11B=R11B-RA(SRC)-<1>,SRD(PNXPV),SQ(LSBR)
        CC: C=LSBQ, V=V, N=N, Z=Z       PC: LW, MM

6437 M      000334    000071    164750    000200    000060    100000
**:     CM: STORE HIGH ORDER RESULT
        AR: RB(SRC)=R11A        PC: LW
        CC: C=0, V=0, N=PN, Z=PZ

6440 M      000332    000100    166340    000200    171760    100000
**:     CM: STORE LOW ORDER RESULT
        AR: RB(SRC!1)=Q PC: LW
        CC: C=0, V=0, N=N, Z=PZAZ       IO: INTCK

6441 M      152237    007067    077777    000200    140403    056445
**:     CM: SET UP TO CHECK RANGE
```

```
EXTENDED INSTRUCTION SET

        AR: R10B=[+77777],AD=R7A
        IO: DATIR(CMI)  BR: CBN(N) [NGN]

6442 M      000110    100070    030440    000200    000060    100000
**: PSN
        CM: CHECK FOR  RESULT > 2^15-1
        AR: OUT=Q-R10A-<1>

6443 M      000114    050071    030720    000200    000060    100000
**:     AR: OUT=R11A-<C'>
        CC: V=PNXPV, C=C, N=N, Z=Z

6444 M      120177    000000    030540    000200    000060    046447
**:     AR: OUT=/PSR
        CC: V=CPO, C=C, N=N, Z=Z        BR: CBR(1) [END3]

6445 M      000100    100070    030440    000200    000060    100000
**: NGN
        CM: CHECK FOR RESULT < -2^15
        AR: OUT=Q+R10A+<1>

6446 M      000104    050071    030720    000200    000060    100000
**:     AR: OUT=R11A+<C'>
        CC: V=PNXPV, C=C, N=N, Z=Z

6447 M      000100    000000    020740    000200    000060    046071
**: END3
        CC: C=V, V=0, N=N, Z=Z  BR: CBR(1) [BGN+1]

6450 M      000334    007000    164750    000200    000060    100000
**: DIV
        CM: DIVISION IS PERFORMED BY A NON RESTORING
        CM: METHOD. THE COMPLEMENT OF THE QUOTIENT IS DEVELOPED
        CM: IN THE Q REGISTER AND THE REMAINDER IS LEFT IN R10.
        CM: LOAD HIGH ORDER PART OF DIVIDEND
```

```
        CC: C=0, V=0, N=PN, Z=PZ

6451 M      000033    000100    166340    000200    000043    056454
**:     CM: LOAD LOW ORDER PART OF DIVIDEND, SKIP IF POSITIVE
        AR: Q=RB(SRC!1) PC: LW
        CC: C=0, V=0, N=N, Z=PZAZ       BR: CBN(/N) [.+3]

6452 M      000022    100000    030440    000200    000060    100000
**:     CM: COMPUTE TWO'S COMPLEMENT OF 32 BIT DIVIDEND
        AR: Q=-Q-<1>

6453 M      000323    057000    030440    000200    000060    100000
**:     AR: R10B=-R10B-<C'>

6454 M      000337    007100    110740    010200    000060    100000
**:     CM: LOAD DIVISOR, CHECK IF = 0
        AR: R11B=D      PC: LW, WIOH
        CC: C=PZ, V=0, N=N, Z=Z

6455 M      155137    000000    000017    000200    000053    056457
**:     CM: SET LOOP COUNTER, SKIP IF DIVISOR POSITIVE
```

```
EXTENDED INSTRUCTION SET

        AR: CNTR=[+17]  BR: CBN(/N') [.+2]

6456 M      000323    107100    030440    000200    000060    100000
**:     CM: MAKE DIVISOR POSITIVE
        AR: R11B=-R11B-<1>

6457 M      000611    107071    030720    000223    000040    056461
**:     CM: CHECK FOR OVERFLOW, FALL TO ABORT IF SOURCE = 0
        AR: R10B=R10B-R11A-<1>,SRU(MSBQ),SQ(MSBR)
        CC: C=C, V=PNXPV, N=N, Z=Z      BR: CBN(/C) [.+2]

6460 M      000100    000000    170760    000200    000060    046070
**:     CM: LEAVE R(SRC) & R(SRC!1), SET CODES
        CC: C=1, V=1, N=N, Z=Z  BR: CBR(1) [BGN]

6461 M      120177    000000    030540    000200    000041    056070
**:     CM: ABORT IF OVERFLOW, IE. DIVIDEND > DIVISOR
        AR: OUT=/PSR
        CC: C=C, V=CPO, N=N, Z=Z        BR: CBN(/V) [BGN]

6462 M      000137    000000    160347    000200    000060    100000
**:     CM: SAVE SIGN OF DIVIDEND
        CM: COMPUTE SIGN OF QUOTIENT
        AR: OUT=D       CC: C=0, V=0, N=PNXN, Z=N

6463 M      000601    007071    107020    000223    000060    100000
**:     CM: FIRST OPERATION IS AN ADD
        AR: R10B=R10B+R11A+<0>,SRU(MSBQ),SQ(MSBR)
        CC: C=PN, V=Z, N=N, Z=0

6464 M      001601    117071    107040    046223    000077    100000
**:     CM: REPEAT NEXT CYCLE 15 TIMES
        CM: IF THE CARRY FLAG IS A 1 DO AN ADDITION
        CM: IF THE CARRY FLAG IS A 0 DO A SUBTRACTION
        CM: THE CURRENT SIGN BIT IS THE COMPLEMENT OF
```

```
        AR: D;R10B=R10B+R11A+</C>,SRU(MSBQ),SQ(MSBR)
        PC: MNR, CCL, LW        CC: C=PN, V=V, N=N, Z=0
        BR: CNR(/CNTR)

6465 M      000372    000000    034450    000200    000043    056467
**:     CM: GET QUOTIENT, SKIP IF SIGN IS CORRECT
        AR: RB(SRC)=/Q
        CC: C=C, V=V, N=PN, Z=PZ        BR: CBN(/N) [.+2]

6466 M      000323    100000    034450    000200    000060    100000
**:     CM: CORRECT SIGN OF QUOTIENT
        AR: RB(SRC)=-RB(SRC)-<1>
        CC: C=C, V=V, N=PN, Z=PZ

6467 M      000305    000171    160740    004200    000041    056070
**:     CM: RESTORE REMAINDER CORRECTION
        CM: IF C = 1 ADD R11  ELSE ADD 0
        CM: CHECK IF SIGN OF REMAINDER NEEDS CORRECTING
        AR: RB(SRC!1)=D+R11A+<0>        PC: MM, LW
        CC: C=0, V=0, N=N, Z=Z  BR: CBN(/V) [BGN]
```

EXTENDED INSTRUCTION SET

```
6470 M      000323    100100    160740    000200    000060    046070
**:     CM: CORRECT SIGN OF REMAINDER
        AR: RB(SRC!1)=-RB(SRC!1)-<1>
        CC: C=0, V=0, N=N, Z=Z  BR: CBR(1) [BGN]

        NA: MORGEND=.
```

```
        PG:     EXTENDED INSTRUCTION CODES

        .=: IORIGIN+300

6300 I      006431    012040
**:     MA: MUL PS: DST, EX     IN: SP

6301 I      006431    012000
**:     MA: MUL PS: DST, EX

6302 I      006450    012040
**:     MA: DIV PS: DST, EX     IN: SP

6303 I      006450    012000
**:     MA: DIV PS: DST, EX

6304 I      006400    012040
**:     MA: ASH PS: DST, EX     IN: SP

6305 I      006400    012000
**:     MA: ASH PS: DST, EX

6306 I      006413    012040
**:     MA: ASHC  PS: DST, EX   IN: SP

6307 I      006413    012000
**:     MA: ASHC  PS: DST, EX
```

```
        PG:     FIS OVERLAY

        CM:       THE FOLLOWING INSTRUCTION CODE
                DEFINITIONS OVERLAY THE ARB11 CODE TO ADD THE FOLLOWING
                FLOATING POINT INSTRUCTIONS.

                PDP-11/35 AND LSI-11 COMPATIBLE

                        FADD    07500R
                        FSUB    07501R
                        FMUL    07502R
                        FDIV    07503R

        CM:     NOTES ON CODING -
                1) CONDITION CODES ARE CHANGED DURING OPERAND
                   ACCESS. MEMORY TRAPS WILL HAVE INVALID CC'S

                2) A.GETARG, B.GETARG, AND ARG.SAV ROUTINES
                   SUPPORT FLOATING AND DOUBLE FORMATS

                3) THE SINGLE PRECISION FLOATING ROUTINES
                        ADD.24
                        MUL.24
                        DIV.24
                        AL.24
                        NORM.24
                   MAY BE CALLED BY THE FPP

                4) DOUBLE PRECISION FLOATING POINT OPERATIONS
                    ARE SUPPORTED BY THE FOLLOWING ROUTINES
                        ADD.56
                        MUL.56
                        DIV.56
                        AL.56
                        NORM.56
                   AND MAY BE CALLED BY THE FPP
```

```
        PG:     DEFINE FIS INSTRUCTIONS

        .=:IORIGIN+340

6340 I      006507    002000
**:     MA: FADD        PS: EX

6341 I      006505    002000
**:     MA: FSUB        PS: EX

6342 I      006477    002000
**:     MA: FMUL        PS: EX

6343 I      006471    002000
**:     MA: FDIV        PS: EX

        .=: MORGEND
```

```
        PG:     FDIV


6471 M      002334    007020    167356    000200    140060    076563
**: FDIV
        CM: GET FLOATING ARGUMENT B
        AR: AD;R10B=RA(DST)     IO: DATI(CMI)
        CC: C=0,Z=0,N=0,V=0     BR: CJN(1) [B.GETARG+1]

6472 M      152305    002020    000004    000200    000000    056536
**:     CM: ON ZERO ARGUMENT - BRANCH
        AR: AD;RB(DST)=RA(DST)+[+4]+<0>         BR: CBN(C) [DIVZERO]

6473 M      000334    007020    030440    000200    141060    076546
**:     CM: GET ARGUMENT A
        AR: R10B=RA(DST)
        IO: IDATI(CMI)  BR: CJN(1) [A.GETARG+1]

6474 M      155137    000000    000032    000200    000000    056516
**:     CM: ON ZERO ARGUMENT - GENERATE A CLEAN ZERO
            PRESET LOOP COUNTER
        AR: CNTR=[+32]          BR: CBN(C) [FIS.SAV]

6475 M      000111    007472    107357    001600    000060    076635
**:     CM: DO 24-BIT DIVISION  (ROUNDED)
        AR: R14B-R12A-<0>       PC: 4B
        CC: C=PN,N=1,Z=0,V=0    BR: CJN(1) [DIV.24+2]

6476 M      000303    006400    030440    000600    000060    056514
**:     CM: CHECK EXPONENT
        AR: R4B=R4B+<0>         PC: UW
        BR: CBN(1) [EXPCHK]
```

```
        PG:     FMUL


6477 M      002334    007020    167356    000200    140060    076563
**: FMUL
        CM: GET FLOATING ARGUMENT B
        AR: AD;R10B=RA(DST)     IO: DATI(CMI)
        CC: C=0,N=0,Z=0,V=0     BR: CJN(1) [B.GETARG+1]

6500 M      152305    002020    000004    000200    000000    056516
**:     CM: CHECK FOR ZERO ARGUMENT
        AR: AD;RB(DST)=RA(DST)+[+4]+<0>         BR: CBN(C) [FIS.SAV]

6501 M      000334    007020    030440    000200    141060    076546
**:     CM: GET A ARGUMENT
        AR: R10B=RA(DST)
        IO: IDATI(CMI)          BR: CJN(1) [A.GETARG+1]

6502 M      155137    000000    000030    000200    000000    056516
**:     CM: CHECK FOR ZERO ARGUMENT
        AR: CNTR=[+30]          BR: CBN(C) [FIS.SAV]

6503 M      000533    007400    007057    001600    000060    076622
**:     CM: DO 24-BIT MULTIPLY (ROUNDED)
        AR: R14B=R14B,SRD(0)    PC: 4B
        CC: C=LSBR,N=1,Z=0,V=V  BR: CJN(1) [MUL.24+2]

6504 M      000303    006400    030440    000600    000060    056514
**:     CM: CHECK EXPONENT
        AR: R4B=R4B+<0>         PC: UW
        BR: CBN(1) [EXPCHK]
```

```
        PG:     FSUB


6505 M      002334    007020    167356    000200    140060    076563
**: FSUB
        CM: GET FLOATING ARGUMENT B
        AR: AD;R10B=RA(DST)     IO: DATI(CMI)
        CC: C=0,N=0,Z=0,V=0     BR: CJN(1) [B.GETARG+1]


6506 M      150365    006565    100000    000600    000060    056510
**:     CM: NEGATE SIGN OF OPERAND
        AR: R5B=R5A XOR [+100000]       PC: UW
        BR: CBN(1) [FADD+1]
```

```
        PG:     FADD


6507 M      002334    007020    167356    000200    140060    076563
**: FADD
        CM: GET FLOATING ARGUMENT B
        AR: AD;R10B=RA(DST)     IO: DATI(CMI)
        CC: C=0,N=0,Z=0,V=0     BR: CJN(1) [B.GETARG+1]

6510 M      152305    002020    000004    000200    140060    100000
**:     CM: GET A ARGUMENT
        AR: AD;RB(DST)=RA(DST)+[+4]+<0>         IO: DATI(CMI)

6511 M      000334    007020    030440    000200    000060    076546
**:     AR: R10B=RA(DST)        BR: CJN(1) [A.GETARG+1]

6512 M      000133    006200    164457    000600    000060    076600
**:     CM: DO 24-BIT ADDITION (ROUNDED)
        AR: R2B                 PC: UW
        CC: C=0,N=1,Z=PZ,V=V    BR: CJN(1) [ADD.24+1]

6513 M      000104    000064    030440    000600    000004    056516
**:     CM: ON ZERO RESULT (C=1) - GENERATE A CLEAN ZERO
            IF A OR B = 0  (Z=1) - SAVE RESULT IN B
            CHECK EXPONENT
        AR: R4A+<0>     PC: UW
        BR: CBN(COZ) [FIS.SAV]

6514 M      150115    100064    000400    000600    000017    056537
**: EXPCHK
        CM: BRANCH ON UNDERFLOW, CHECK OVERFLOW
        AR: R4A-[+400]-<1>      PC: UW
        BR: CBN(BLE') [UNDFLO]

6515 M      000100    000000    167056    000200    000053    056540
**:     CM: BRANCH ON OVERFLOW
        CC: C=0,N=0,Z=0,V=V     BR: CBN(/N') [OVRFLO]
```

```
6516 M      000334    007020    030440    000200    000060    076520
**: FIS.SAV
        CM: SET ADDRESS / GO SAVE ARGUMENT
        AR: R10B=RA(DST)        BR: CJN(1) [ARG.SAV]

6517 M      152105    000067    000000    020200    171760    056073
**:     CM: WAIT TO START NEXT INSTRUCTION
        AR: AD=R7A+[+0]+<0>     PC: LW,WIOL
        IO: INTCK               BR: CBN(1) [BGN+3]
```

```
        PG:     SAVE ARGUMENT ROUTINE

CM:     IF C=1 A CLEAN ZERO IS GENERATED, ELSE
        R4 (UW) - EXPONENT
        R5 (UW) - FRACTION SIGN
        R14 (4B) - FRACTION
        R15 (4B) - FRACTION
        ARE COMBINED INTO THE STANDARD FLOATING POINT
        FORMAT = FLOATING (V=0)
         SIGN(1-BIT)/EXPONENT(8-BITS)/FRACTION(23-BITS)
        FORMAT = DOUBLE (V=1)
         SIGN(1-BIT)/EXPONENT(8-BITS)/FRACTION(55-BITS)
        AND SAVED AT THE ADDRESS IN R10(LW)

6520 M      150355    007474    177600    000600    000060    076531
**: ARG.SAV
        CM: MASK FRACTION / GO COMBINE SIGN, EXPONENT, AND FRACTION
        AR: R14B=[+177600] MASK R14A    PC: UW
        BR: CJN(1) [CMB.SEF+1]

6521 M      002134    000070    030440    000200    100260    100000
**: ARG.SAVA
        CM: SAVE RESULT (R14-4B)
        AR: AD=R10A             IO: DATO(CM)

6522 M      001133    007400    160440    020200    000060    100000
**:     CM: CLEAR EXTRA FLAGS
        AR: D=R14B              PC: LW,WIOL
        CC: C=0,V=V,N=N,Z=Z

6523 M      152105    000070    000002    000200    100241    130000
**:     AR: AD=R10A+[+2]+<0>
        IO: DATO(CM)            BR: CRN(/V)

6524 M      001133    007500    160740    020600    000060    100000
**:     AR: D=R15B              PC: UW,WIOL
```

```
6525 M      152105    000070    000004    000200    100260    100000
**:     AR: AD=R10A+[+4]+<0>    IO: DATO(CM)

6526 M      001133    007500    030440    020200    000060    100000
**:     AR: D=R15B              PC: LW,WIOL

6527 M      152105    000070    000006    000200    100260    130000
**:     AR: AD=R10A+[+6]+<0>
        IO: DATO(CM)            BR: CRN(1)

6530 M      150355    007474    177600    000600    000060    100000
**: CMB.SEF
        CM: MASK FRACTION
        AR: R14B=[+177600] MASK R14A    PC: UW

6531 M      001233    006564    100450    000600    000040    056534
**:     CM: GET SIGN OF RESULT / BRANCH IF NOT A CLEAN ZERO
        AR: R5B=R5B,D=R4A               PC: UW
        CC: C=PN,N=PN,Z=Z,V=V           BR: CBN(/C) [.+3]
```

SAVE ARGUMENT ROUTINE

```
6532 M      001342    007400    177456    001600    000060    100000
**:     CM: ELSE A CLEAN ZERO
        AR: D;R14B=0                    PC: 4B
        CC: C=1,N=0,Z=1,V=V

6533 M      000342    007500    030440    001600    000060    120000
**:     AR: R15B=0                      PC: 4B
        BR: CRR(1)

6534 M      010537    006400    114440    000604    000060    100000
**:     CM: POSITION EXPONENT AND COMBINE WITH SIGN
        AR: R4B=DSWB,SRD(C)             PC: UW
        CC: C=PZ,Z=PZ,N=N,V=V

6535 M      001331    007464    030440    000600    000060    130000
**:     CM: COMBINE HIGH ORDER FRACTION WITH EXPONENT AND SIGN
        AR: D;R14B=R14B OR R4A          PC: UW
        BR: CRN(1)
```

```
        PG:     FIS TRAP HANDLERS

6536 M      000100    000000    177057    000200    000060    046541
**: DIVZERO
        CC: V=V,N=1,C=1,Z=0     BR: CBR(1) [FIS.TRAP]

6537 M      000100    000000    167057    000200    000060    046541
**: UNDFLO
        CC: V=V,N=1,C=0,Z=0     BR: CBR(1) [FIS.TRAP]

6540 M      000100    000000    167056    000200    000060    100000
**: OVRFLO
        CC: V=V,N=0,C=0,Z=0

6541 M      000100    000000    030760    000200    000001    056543
**: FIS.TRAP
        CC: V=1,N=N,C=C,Z=Z     BR: CBN(V) [.+2]

6542 M      150315    102020    000004    000200    000060    056544
**:     CM: RESET STACK FOR FLOATING
        AR: RB(DST)=RA(DST)-[+4]-<1>
        BR: CBN(1) [.+2]

6543 M      150315    102020    000010    000200    000060    100000
**:     CM: RESET STACK FOR DOUBLE
        AR: RB(DST)=RA(DST)-[+10]-<1>

6544 M      152337    007000    000246    020200    160060    046001
**:     AR: AD;R10B=[+246]      PC: LW,WIOL
        IO: DATI(KI)    BR: CBR(1) [RES+1]
```

```
        PG:     GET A OPERAND CODE

CM:     THIS CODE GETS THE A ARGUMENT AT THE ADDRESS IN R10 (LW)
        SET V=0 FOR FLOATING / V=1 FOR DOUBLE
        AND EXITS WITH
        C,Z=1 IF EXPONENT = 0
        N=1 IF ARGUMENT < 0
        R2 (UW) = EXPONENT
        R3 (UW) = SIGN OF FRACTION
        R12 (4B) = FRACTION (24-BITS)
        R13 (4B) = FRACTION (32-BITS)

6545 M      002134    000070    030440    000200    140060    100000
**: A.GETARG
        AR: AD=R10A             IO: DATI(CMI)

6546 M      000342    007200    030440    000600    000060    100000
**:     CM: CLEAR HIGH FRACTION
        AR: R12B=0              PC: UW

6547 M      152105    000070    000002    020200    000060    100000
**:     AR: AD=R10A+[+2]+<0>    PC: LW,WIOL

6550 M      000737    006200    107050    000600    140060    076555
**:     CM: SAVE HIGH ORDER IN TEMPORARY
            GO BUILD SIGN, EXPONENT, AND HIGH FRACTION
        AR: R2B=D,SRU(0)        PC: UW
        CC: C=MSBR,N=PN,Z=0,V=V
        IO: DATI(CMI)           BR: CJN(1) [A.BLDSEF]

6551 M      152105    000070    000004    000200    140060    100000
**:     CM: GET DOUBLE FRACTION
        AR: AD=R10A+[+4]+<0>    IO: DATI(CMI)

6552 M      000337    007300    030440    000600    140060    100000
**:     AR: R13B=D              PC: UW
```

```
6553 M      152105    000070    000006    020200    000060    100000
**:     AR: AD=R10A+[+6]+<0>    PC: LW,WIOL

6554 M      002237    007367    030440    010200    000060    120000
**:     CM: DUMBY LOAD AD
        AR: R13B=D,AD=R7A       PC: LW,WIOH
        BR: CRR(1)
```

```
        PG:     BUILD SIGN, EXPONENT, AND HIGH FRACTION FOR A

CM:       IF V=0, FLOATING MODE - POPS ONE RETURN OFF STACK THEN RETURNS
          IF V=1, DOUBLE MODE - RETURNS TO CALLER

6555 M      000542    006300    030440    000604    000060    100000
**: A.BLDSEF
        CM: SAVE SIGN
        AR: R3B=0,SRD(C)        PC: UW

6556 M      000534    007262    030440    000401    000060    100000
**:     CM: MASK IN FRACTION AND HIDDEN BIT
        AR: R12B=R2A,SRD(1)     PC: UB

6557 M      002237    007267    030440    010200    000060    100000
**:     CM: SAVE LOW FRACTION / DUMBY LOAD AD
        AR: R12B=D,AD=R7A       PC: LW,WIOH

6560 M      001242    006262    030440    000600    000041    110000
**:     CM: GET EXPONENT INTO R2
        AR: R2B=0,D=R2A         PC: UW
        BR: CPS(/V)

6561 M      010337    006200    114440    000400    000060    120000
**:     AR: R2B=DSWB            PC: UB
        CC: Z=PZ,C=PZ,N=N,V=V   BR: CRR(1)
```

```
        PG:     GET B OPERAND CODE

CM:     THIS CODE GETS THE B ARGUMENT AT THE ADDRESS IN R10 (LW)
        SET V=0 FOR FLOATING / V=1 FOR DOUBLE
        AND EXITS WITH
        C,Z=1 IF EXPONENT = 0
        N=1 IF ARGUMENT < 0
        R4 (UW) = EXPONENT
        R5 (UW) = SIGN OF FRACTION
        R14 (4B) = FRACTION (24-BITS)
        R15 (4B) = FRACTION (32-BITS)

6562 M      002134    000070    030440    000200    140060    100000
**: B.GETARG
        AR: AD=R10A     IO: DATI(CMI)

6563 M      000342    007400    030440    000600    000060    100000
**:     CM: CLEAR HIGH FRACTION
        AR: R14B=0              PC: UW

6564 M      152105    000070    000002    020200    000060    100000
**:     AR: AD=R10A+[+2]+<0>    PC: LW,WIOL

6565 M      000737    006400    107050    000600    140060    076572
**:     CM: SAVE HIGH ORDER IN TEMPORARY
            GO BUILD SIGN, EXPONENT, AND HIGH FRACTION
        AR: R4B=D,SRU(0)        PC: UW
        CC: C=MSBR,N=PN,Z=0,V=V
        IO: DATI(CMI)           BR: CJN(1) [B.BLDSEF]

6566 M      152105    000070    000004    000200    140060    100000
**:     CM: GET DOUBLE FRACTION
        AR: AD=R10A+[+4]+<0>    IO: DATI(CMI)

6567 M      152105    000070    000006    020200    000060    100000
**:     AR: AD=R10A+[+6]+<0>    PC: LW,WIOL
```

```
6570 M      000337    007500    030440    000600    140060    100000
**:     AR: R15B=D              PC: UW
        IO: DATI(CMI)

6571 M      002237    007567    030440    010200    000060    120000
**:     CM: DUMBY LOAD AD
        AR: R15B=D,AD=R7A       PC: LW,WIOH
        BR: CRR(1)
```

```
        PG:     BUILD SIGN, EXPONENT, AND HIGH FRACTION FOR B

CM:       IF V=0, FLOATING MODE - POPS ONE RETURN OFF STACK THEN RETURNS
          IF V=1, DOUBLE MODE - RETURNS TO CALLER

6572 M      000542    006500    030440    000604    000060    100000
**: B.BLDSEF
        CM: SAVE SIGN
        AR: R5B=0,SRD(C)        PC: UW

6573 M      000534    007464    030440    000401    000060    100000
**:     CM: MASK IN FRACTION AND HIDDEN BIT
        AR: R14B=R4A,SRD(1)     PC: UB

6574 M      002237    007467    030440    010200    000060    100000
**:     CM: SAVE LOW FRACTION / DUMBY LOAD AD
        AR: R14B=D,AD=R7A       PC: LW,WIOH

6575 M      001242    006464    030440    000600    000041    110000
**:     CM: GET EXPONENT INTO R4
        AR: R4B=0,D=R4A         PC: UW
        BR: CPS(/V)

6576 M      010337    006400    114440    000400    000060    120000
**:     AR: R4B=DSWB    PC: UB
        CC: Z=PZ,C=PZ,N=N,V=V   BR: CRR(1)
```

```
        PG:     24-BIT ADDITION

CM:     ADDITION PERFORMED BETWEEN REGISTERS
        R12/R2/R3 AND R14/R4/R5
        FRACTION RESULT IN R14 (4B)
        EXPONENT RESULT IN R4 (UW)
        SIGN RESULT IN R5 (UW)
        N IS ROUND/TRUNCATE FLAG
        V IS NOT USED
        C = 1 IF RESULT IS ZERO
        Z = 1 IF ARGUMENT A OR B OR RESULT = 0

6577 M      000133    006200    164440    000600    000060    100000
**: ADD.24
        CM: IS A ARGUMENT = 0 ?
        AR: R2B         CC: Z=PZ,C=0,N=N,V=V    PC: UW

6600 M      000133    006400    110440    000600    000002    130000
**:     CM: IF A ARGUMENT IS ZERO - B IS RESULT
            AND IF B=0, THEN A CLEAN ZERO WILL BE GENERATED
        AR: R4B         CC: C=PZ,N=N,Z=Z,V=V    PC: UW
        BR: CRN(Z)

6601 M      000042    000000    030440    001600    000040    056605
**:     CM: BRANCH IF B ARGUMENT IS NOT ZERO
            CLEAR NORMALIZE SHIFT REGISTER (Q)
        AR: Q=0         PC: 4B
        BR: CBN(/C) [.+4]

6602 M      000334    007472    030440    001600    000060    100000
**: COPY.24
        CM: ELSE A IS THE RESULT
        AR: R14B=R12A   PC: 4B

6603 M      000334    006462    030440    000600    000060    100000
**:     AR: R4B=R2A     PC: UW
```

```
6604 M      000334    006563    167440    000600    000060    120000
**:     AR: R5B=R3A     PC: UW
        CC: C=0,Z=1,N=N,V=V     BR: CRR(1)

6605 M      000733    007200    030440    001600    000060    076645
**:     CM: ALIGN EXPONENTS
        AR: R12B=R12B,SRU(0)    PC: 4B
        BR: CJN(1) [AL.24+1]

6606 M      000161    006563    030440    000600    000060    076610
**:     CM: CHECK SIGNS OF ARGUMENTS / DO ADDITION
        AR: R5B XOR R3A         PC: UW
        BR: CJN(1) [ADD.24A]

6607 M      000433    007400    114440    001660    000060    056673
**:     CM: POST NORMALIZE / NORM.24 RETURNS TO CALLER
        AR: R14B=R14B,SRD(0),SQ(LSBR)   PC: 4B
        CC: C=PZ,N=N,V=V,Z=PZ           BR: CBN(1) [NORM.24+2]

6610 M      000133    006500    030440    000600    000013    056612
**: ADD.24A
```

24-BIT ADDITION

```
        CM: BRANCH IF SIGNS DIFFER
        AR: R5B                 PC: UW
        BR: CBN(N') [.+2]

6611 M      000301    007472    030440    001600    000060    130000
**:     CM: ELSE COMPUTE SUM
        AR: R14B=R14B+R12A+<0>  PC: 4B
        BR: CRN(1)

6612 M      000100    000000    030440    000200    000013    056614
**:     CM: OPPOSITE SIGNS - DIDDLE A LITTLE
        BR: CBN(N') [.+2]

6613 M      000323    107200    030440    001600    000060    056615
**:     CM: NEGATE R12
        AR: R12B=-R12B-<1>      PC: 4B
        BR: CBN(1) [.+2]

6614 M      000323    107400    030440    001600    000060    100000
**:     CM: NEGATE R14
        AR: R14B=-R14B-<1>      PC: 4B

6615 M      000301    007472    100440    001600    000060    100000
**:     CM: COMPUTE RESULT
        AR: R14B=R14B+R12A+<0>  PC: 4B
        CC: C=PN,N=N,V=V,Z=Z

6616 M      000542    006500    030440    000604    000040    130000
**:     CM: SAVE SIGN OF RESULT
        AR: R5B=0,SRD(C)        PC: UW
        BR: CRN(/C)

6617 M      000323    107400    030440    001600    000060    130000
**:     CM: GET MAGNITUDE
        AR: R14B=-R14B-<1>      PC: 4B
```

```
        PG:     24-BIT MULTIPLY ROUTINE

CM:     MULTIPLICAND IN R12 (4B)
        MULTIPLIER IN R14 (4B)
        RESULT LEFT IN R14 (4B) NORMALIZED
        EXPONENT RESULT IN R4 (UW)
        SIGN RESULT IN R5 (UW)
        RESULT IS ROUNDED(N=1)/TRUNCATED(N=0)
        V AND Z NOT EFFECTED, C USED

6620 M      155137    000000    000030    000200    000060    100000
**: MUL.24
        AR: CNTR=[+30]

6621 M      000533    007400    000440    001600    000060    100000
**:     CM: PRESHIFT MULTIPLIER
        AR: R14B=R14B,SRD(0)    PC: 4B
        CC: C=LSBR,Z=Z,V=V,N=N

6622 M      000033    007400    030440    001600    000060    100000
**:     CM: LOAD MULTIPLIER
        AR: Q=R14B              PC: 4B

6623 M      000342    007400    030440    001600    000060    100000
**:     CM: INITIALIZE RESULT
        AR: R14B=0              PC: 4B

6624 M      000401    007472    140440    045660    000077    100000
**:     CM: DO 24-BITS OF MULTIPLY
        AR: R14B=R14B+R12A+<0>,SRD(0),SQ(LSBR)  PC: 4B,MM,CCL
        CC: C=LSBQ,V=V,N=N,Z=Z          BR: CNR(/CNTR)

6625 M      000133    007400    100440    001400    000060    100000
**:     CM: CHECK MOST SIGNIFICANT BIT
        AR: R14B        CC: C=PN,N=N,Z=Z,V=V    PC: 3B
```

```
**:     CM: CONDITIONALLY UPDATE B ARGUMENT EXPONENT
        AR: R4B=R4A-[+200]-<C>          PC: UW
        BR: CBN(C) [.+2]

6627 M      000633    007400    030440    001603    000043    056631
**:     CM: NORMALIZE (SHIFT LEFT) / BRANCH IF TRUNCATING RESULT
        AR: R14B=R14B,SRU(MSBQ),SQ(0)   PC: 4B
        BR: CBN(/N) [.+2]

6630 M      000132    000000    100440    001600    000003    076705
**:     CM: DO ROUNDING IF NEEDED
        AR: Q                   PC: 4B
        CC: C=PN,N=N,V=V,Z=Z    BR: CJN(N) [NORM.24A]

6631 M      000361    006563    160440    000600    000060    100000
**:     CM: COMPUTE SIGN OF RESULT / CLEAR C
        AR: R5B=R5B XOR R3A     PC: UW
        CC: C=0,Z=Z,N=N,V=V

6632 M      000301    006462    030440    000600    000060    130000
**:     CM: COMPUTE EXPONENT
```

24-BIT MULTIPLY ROUTINE

```
        AR: R4B=R4B+R2A+<0>     PC: UW
        BR: CRN(1)
```

```
        PG:     24-BIT DIVISION ROUTINE

CM:     DIVIDEND IN R12 (4B)
        DIVISOR IN R14 (4B)
        RESULT LEFT IN R14 (4B) NORMALIZED
        REMAINDER LEFT IN R12
        EXPONENT RESULT IN R4 (UW)
        SIGN RESULT IN R5 (UW)
        RESULT IS ROUNDED(N=1)/TRUNCATED(N=0)
        V AND Z NOT EFFECTED, C USED

6633 M      155137    000000    000032    000200    000060    100000
**: DIV.24
        CM: PRESET LOOP COUNTER
        AR: CNTR=[+32]

6634 M      000111    007472    100440    001600    000060    100000
**:     CM: CHECK DIVISOR > DIVIDEND
        AR: R14B-R12A-<0>       PC: 4B
        CC: C=PN,N=N,Z=Z,V=V

6635 M      000311    116462    160440    000600    000040    056637
**:     CM: CONDITIONAL EXPONENT UPDATE
        AR: R4B=R4B-R2A-</C>    PC: UW
        CC: C=0,N=N,Z=Z,V=V     BR: CBN(/C) [.+2]

6636 M      000733    007400    030440    001600    000060    100000
**:     CM: SHIFT DIVISOR
        AR: R14B=R14B,SRU(0)    PC: 4B

6637 M      000601    117274    100440    047620    000077    100000
**:     CM: EXECUTE 26. CYCLES
        IF C=0  A SUBTRACTION IS DONE
        IF C=1  AN ADDITION IS DONE
        ONE EXTRA BIT IS COMPUTED
        RESULT = /Q
```

```
        AR: R12B=R12B+R14A+</C>,SRU(0),SQ(MSBR)
        PC: 4B,MNR,CCL          CC: C=PN,N=N,Z=Z,V=V
        BR: CNR(/CNTR)

6640 M      000572    007400    000440    001600    000003    076705
**:     CM: GET RESULT / ROUND IF NEEDED
        AR: R14B=/Q,SRD(0)      PC: 4B
        CC: C=LSBR,N=N,Z=Z,V=V  BR: CJN(N) [NORM.24A]

6641 M      150345    007474    000377    000600    000060    100000
**:     CM: MASK JUNK
        AR: R14B=R14A AND [+377]        PC: UW

6642 M      000361    006563    160440    000600    000060    100000
**:     CM: COMPUTE SIGN OF RESULT / CLEAR C
        AR: R5B=R5B XOR R3A     PC: UW
        CC: C=0,N=N,Z=Z,V=V

6643 M      150325    106464    000200    000600    000060    130000
**:     CM: COMPUTE EXPONENT
        AR: R4B=[+200]-R4A-<1>  PC: UW
```

24-BIT DIVISION ROUTINE

BR: CRN(1)

```
        PG:     24-BIT OPERAND ALIGNMENT

CM:     ARGUMENTS PRESHIFTED ONE BIT (*2)
        C IS USED
        N,V, AND Z NOT USED

6644 M      000733    007200    030440    001600    000060    100000
**: AL.24
        CM: PRESHIFT FRACTIONS
        AR: R12B=R12B,SRU(0)    PC: 4B

6645 M      000733    007400    030440    001600    000060    100000
**:     AR: R14B=R14B,SRU(0)    PC: 4B

6646 M      001111    106462    030440    000600    000060    100000
**:     CM: COMPUTE EXPONENT DIFFERENCE
        AR: D=R4B-R2A-<1>       PC: UW

6647 M      005337    007100    100440    000600    000012    130000
**:     CM: RETURN IF ALIGNED
        AR: CNTR;R11B=D         PC: UW
        CC: C=PN,N=N,Z=Z,V=V    BR: CRN(Z')

6650 M      150305    117171    000020    006600    000000    056661
**:     CM: BRANCH IF A > B  (IE. CNTR < 0)
        CM: IF C=0 THEN R11A-[+20]-<1> IS COMPUTED
        CM: IF C=1 THEN R11A+[+20]+<0> IS COMPUTED
        AR: R11B=R11A+[+20]+</C>        PC: UW,MNR
        BR: CBN(C) [AL.24B]

6651 M      000334    006264    160440    000600    000053    056653
**: AL.24A
        CM: BRANCH IF 16. OR MORE SHIFTS NEEDED
        AR: R2B=R4A             PC: UW
        CC: C=0,N=N,Z=Z,V=V     BR: CBN(/N') [.+2]
```

```
**:     CM: DO SHIFTS
        AR: R12B=R12B,SRD(0)    PC: 4B,CCL
        BR: CRR(/CNTR)

6653 M      150115    000071    000010    000600    000060    100000
**:     CM: CHECK IF MORE THAN 24. SHIFTS NEEDED
        AR: R11A-[+10]-<0>      PC: UW

6654 M      005134    000071    110440    000600    000053    056660
**:     CM: BRANCH IF MORE
        AR: CNTR=R11A           PC: UW
        CC: C=PZ,N=N,Z=Z,V=V    BR: CBN(/N') [.+4]

6655 M      001242    007272    030440    000600    000060    100000
**:     CM: DO 16. BIT SHIFT
        AR: R12B=0,D=R12A       PC: UW

6656 M      000337    007200    030440    000200    000040    056652
**:     CM: BRANCH IF MORE TO DO
        AR: R12B=D              PC: LW
        BR: CBN(/C) [AL.24A+1]
```

24-BIT OPERAND ALIGNMENT

```
6657 M      000100    000000    030440    000200    000060    120000
**:     CM: ELSE FINISHED
        BR: CRR(1)

6660 M      000342    007200    030440    001600    000060    120000
**:     CM: CLEAR WORD
        AR: R12B=0              PC: 4B
        BR: CRR(1)

6661 M      000334    006462    030440    000600    000017    056663
**: AL.24B
        CM: BRANCH IF 16. OR MORE SHIFTS NEEDED
        AR: R4B=R2A     PC: UW
        BR: CBN(BLE') [.+2]

6662 M      000533    007400    030440    041600    000077    120000
**:     CM: DO SHIFTS
        AR: R14B=R14B,SRD(0)    PC: 4B,CCL
        BR: CRR(/CNTR)

6663 M      150105    000071    000010    000600    000060    100000
**:     CM: CHECK IF MORE THAN 24. SHIFTS NEEDED
        AR: R11A+[+10]+<0>      PC: UW

6664 M      005134    000071    110440    000600    000013    056670
**:     CM: BRANCH IF MORE
        AR: CNTR=R11A           PC: UW
        CC: C=PZ,N=N,Z=Z,V=V    BR: CBN(N') [.+4]

6665 M      001242    007474    030440    000600    000060    100000
**:     CM: DO 16. BIT SHIFT
        AR: R14B=0,D=R14A       PC: UW

6666 M      000337    007400    030440    000200    000040    056662
```

```
        AR: R14B=D              PC: LW
        BR: CBN(/C) [AL.24B+1]

6667 M      000100    000000    030440    000200    000060    120000
**:     CM: ELSE FINISHED
        BR: CRR(1)

6670 M      000342    007400    030440    001600    000060    120000
**:     CM: CLEAR WORD
        AR: R14B=0              PC: 4B
        BR: CRR(1)
```

```
        PG:     24-BIT POST NORMALIZATION

CM:     EXPECTS ARGUMENTS PRESHIFTED (*2)
        SET N=1 FOR ROUNDING
        OR N=0 FOR TRUNCATING
        C,Z=1 IF RESULT IS ZERO
        V NOT USED

6671 M      000042    000000    030440    001600    000060    100000
**: NORM.24
        CM: CLEAR SHIFTING REGISTER
        AR: Q=0         PC: 4B

6672 M      000433    007400    114440    001660    000060    100000
**:     CM: PRESHIFT / CHECK FOR ZERO
        AR: R14B=R14B,SRD(0),SQ(LSBR)           PC: 4B
        CC: C=PZ,Z=PZ,V=V,N=N

6673 M      155137    000000    000003    000200    000002    130000
**:     CM: PRESET SHIFT COUNTER / RETURN ON ZERO
        AR: CNTR=[+3]   BR: CRN(Z)

6674 M      000433    007400    167040    001660    000060    100000
**:     CM: PRESHIFT
        AR: R14B=R14B,SRD(0),SQ(LSBR)           PC: 4B
        CC: Z=0,C=0,N=N,V=V

6675 M      001132    000000    030440    000600    000060    100000
**:     CM: GET Q REGISTER BITS INTO POSITION
        AR: D=Q         PC: UW

6676 M      010037    000000    030440    000600    000060    100000
**:     AR: Q=DSWB      PC: UW

6677 M      000633    007400    100440    041423    000000    100000
**:     CM: SHIFT UNTIL A '1' BIT FOUND
```

```
        CC: C=MSBR,N=N,V=V,Z=Z                  BR: CNR(C)

6700 M      000433    007400    030440    001467    000060    100000
**:     CM: TWO RESTORING SHIFTS
        AR: R14B=R14B,SRD(LSBQ),SQ(LSBR)        PC: 3B

6701 M      000433    007400    000440    001467    000060    100000
**:     AR: R14B=R14B,SRD(LSBQ),SQ(LSBR)        PC: 3B
        CC: C=LSBR,N=N,Z=Z,V=V

6702 M      141137    000000    030440    000200    000003    056704
**:     CM: ADJUST EXPONENT
            BRANCH IF ROUNDING
        AR: D=CNTR      BR: CBN(N) [.+2]

6703 M      030305    006464    167040    000600    000060    130000
**:     CM: COMPUTE EXPONENT AND RETURN
        AR: R4B=R4A+DSX7+<0>            PC: UW
        CC: C=0,Z=0,N=N,V=V             BR: CRN(1)

6704 M      030305    006464    037040    000600    000040    130000
```

24-BIT POST NORMALIZATION

```
**:     CM: COMPUTE EXPONENT
        AR: R4B=R4A+DSX7+<0>    PC: UW
        CC: C=C,Z=0,N=N,V=V     BR: CRN(/C)

6705 M      000303    017400    137040    001400    000040    130000
**: NORM.24A
        CM: AND ROUND RESULT
        AR: R14B=R14B+<C>       PC: 3B
        CC: C=PC,Z=0,N=N,V=V    BR: CRN(/C)

6706 M      000303    016400    167040    000600    000040    130000
**:     CM: CONDITIONAL UPDATE B EXPONENT
        AR: R4B=R4B+<C>         PC: UW
        CC: C=0,Z=0,N=N,V=V     BR: CRN(/C)

6707 M      000533    007400    030440    001401    000060    130000
**:     CM: NORMALIZE RESULT
        AR: R14B=R14B,SRD(1)    PC: 3B
        BR: CRN(1)
```

```
        PG:     56-BIT ADDITION

CM:     ADDITION PERFORMED BETWEEN REGISTERS
        R12/R13/R2/R3 AND R14/R15/R4/R5
        FRACTION RESULT IN R14/R15 (8B)
        EXPONENT RESULT IN R4 (UW)
        SIGN RESULT IN R5 (UW)
        RESULT IS ROUNDED(N=1)/TRUNCATED(N=0)
        V NOT USED
        C=1 IF RESULT IS ZERO
        Z=1 IF ARGUMENT A OR B OR RESULT = 0

6710 M      000134    000062    164440    000600    000060    100000
**: ADD.56
        CM: IS A ARGUMENT = 0 ?
        AR: R2A         CC: Z=PZ,C=0,N=N,V=V    PC: UW

6711 M      000133    006400    110440    000600    000002    130000
**:     CM: IF A ARGUMENT IS ZERO - B IS RESULT
            AND IF B=0, THEN A CLEAN ZERO WILL BE GENERATED
        AR: R4B         CC: C=PZ,N=N,Z=Z,V=V    PC: UW
        BR: CRN(Z)

6712 M      000042    000000    030440    001600    000040    056714
**:     CM: BRANCH IF B ARGUMENT IS NOT ZERO
            CLEAR NORMALIZE SHIFT REGISTER (Q)
        AR: Q=0         PC: 4B
        BR: CBN(/C) [.+2]

6713 M      000334    007573    030440    001600    000060    056602
**: COPY.56
        CM: ELSE A IS THE RESULT
        AR: R15B=R13A   PC: 4B
        BR: CBN(1) [COPY.24]

6714 M      000733    007300    100440    001600    000060    077001
```

```
        AR: R13B=R13B,SRU(0)    PC: 4B
        CC: C=MSBR,N=N,Z=Z,V=V  BR: CJN(1) [AL.56+1]

6715 M      000161    006563    030440    000600    000060    076717
**:     CM: CHECK SIGNS OF ARGUMENTS / DO ADDITION
        AR: R5B XOR R3A         PC: UW
        BR: CJN(1) [ADD.56A]

6716 M      000333    007400    034440    001600    000060    057053
**:     CM: POST NORMALIZE / NORM.56 RETURNS TO CALLER
        AR: R14B=R14B           PC: 4B
        CC: Z=PZ,C=C,N=N,V=V    BR: CBN(1) [NORM.56+2]

6717 M      000133    006500    030440    000600    000013    056722
**: ADD.56A
        CM: BRANCH IF SIGNS DIFFER
        AR: R5B                 PC: UW
        BR: CBN(N') [.+3]

6720 M      000301    007573    030440    001600    000060    100000
**:     CM: ELSE COMPUTE SUM
```

56-BIT ADDITION

```
        AR: R15B=R15B+R13A+<0>  PC: 4B

6721 M      000301    057472    030440    001600    000060    130000
**:     AR: R14B=R14B+R12A+<C'> PC: 4B
        BR: CRN(1)

6722 M      000100    000000    030440    000200    000013    056725
**:     CM: OPPOSITE SIGNS - DIDDLE A LITTLE
        BR: CBN(N') [.+3]

6723 M      000323    107300    030440    001600    000060    100000
**:     CM: NEGATE A
        AR: R13B=-R13B-<1>      PC: 4B

6724 M      000323    057200    030440    001600    000060    056727
**:     AR: R12B=-R12B-<C'>     PC: 4B
        BR: CBN(1) [.+3]

6725 M      000323    107500    030440    001600    000060    100000
**:     CM: NEGATE B
        AR: R15B=-R15B-<1>      PC: 4B

6726 M      000323    057400    030440    001600    000060    100000
**:     AR: R14B=-R14B-<C'>     PC: 4B

6727 M      000301    007573    030440    001600    000060    100000
**:     CM: COMPUTE RESULT
        AR: R15B=R15B+R13A+<0>  PC: 4B

6730 M      000301    057472    100440    001600    000060    100000
**:     AR: R14B=R14B+R12A+<C'> PC: 4B
        CC: C=PN,N=N,Z=Z,V=V

6731 M      000542    006500    030440    000604    000040    130000
**:     CM: SAVE SIGN OF RESULT
```

```
        BR: CRN(/C)


6732 M      000323    107500    030440    001600    000060    100000
**:     CM: GET MAGNITUDE
        AR: R15B=-R15B-<1>      PC: 4B


6733 M      000323    057400    030440    001600    000060    130000
**:     AR: R14B=-R14B-<C'>     PC: 4B
        BR: CRN(1)
```

```
        PG:     56-BIT MULTIPLICATION

CM:     MULTIPLICAND IN R12/R13 (8B)
        MULTIPLIER IN R14/R15 (8B)
        FRACTION RESULT IN R14/R15 (8B)
        32-BIT TRUNCATION LEFT IN Q
        EXPONENT RESULT IN R4 (UW)
        SIGN RESULT IN R5 (UW)
        R11 (4B) USED AS A TEMPORARY
        RESULT ROUNDED(N=1)/TRUNCATED(N=0)
        C USED
        N,Z,V NOT USED

6734 M      155137    000000    000040    000200    000060    100000
**: MUL.56
        CM: SET UP FOR FIRST 32-BIT MULTIPLICATION
        AR: CNTR=[+40]

6735 M      000334    007174    030440    001600    000060    100000
**:     CM: SAVE HIGH ORDER MULTIPLIER
        AR: R11B=R14A           PC: 4B

6736 M      000342    007400    030440    001600    000060    100000
**:     CM: INITIALIZE RESULT
        AR: R14B=0              PC: 4B

6737 M      000533    007500    000440    001600    000060    100000
**:     CM: PRESHIFT MULTIPLIER
        AR: R15B=R15B,SRD(0)    PC: 4B
        CC: C=LSBR,N=N,V=V,Z=Z

6740 M      000033    007500    030440    001600    000012    056742
**:     CM: LOAD LOW ORDER MULTIPLIER / SKIP IF ZERO
        AR: Q=R15B              PC: 4B
        BR: CBN(Z') [.+2]
```

```
**:     CM: AND GO DO 32 BITS OF MULTIPLY
        AR: R15B=0              PC: 4B
        BR: CJN(1) [MUL.56A]

6742 M      155137    000000    000030    000200    000060    100000
**:     CM: SET UP FOR LAST 24 BITS OF MULTIPLY
        AR: CNTR=[+30]

6743 M      000533    007100    000440    001600    000060    100000
**:     CM: PRESHIFT MULTIPLIER
        AR: R11B=R11B,SRD(0)    PC: 4B
        CC: C=LSBR,N=N,Z=Z,V=V

6744 M      000033    007100    030440    001600    000060    076755
**:     CM: DO LAST 24 BITS OF MULTIPLY
        AR: Q=R11B              PC: 4B
        BR: CJN(1) [MUL.56A]

6745 M      000133    007400    100440    001400    000060    100000
**:     CM: CHECK MSB OF RESULT
        AR: R14B                PC: 3B
```

56-BIT MULTIPLICATION

```
        CC: C=PN,N=N,Z=Z,V=V

6746 M      150315    016464    000200    000600    000000    056751
**:     CM: CONDITIONALLY UPDATE EXPONENT AND FRACTION
        AR: R4B=R4A-[+200]-<C>  PC: UW
        BR: CBN(C) [.+3]

6747 M      000633    007500    100440    001603    000060    100000
**:     AR: R15B=R15B,SRU(MSBQ),SQ(0)
        CC: C=MSBR,N=N,Z=Z,V=V  PC: 4B

6750 M      000733    007400    030440    001604    000043    056752
**:     CM: BRANCH IF TRUNCATING
        AR: R14B=R14B,SRU(C)    PC: 4B
        BR: CBN(/N) [.+2]

6751 M      000132    000000    100440    001600    000003    077100
**:     CM: DO ROUNDING IF REQUIRED
        AR: Q                   PC: 4B
        CC: C=PN,N=N,Z=Z,V=V    BR: CJN(N) [NORM.56A]

6752 M      000361    006563    160440    000600    000060    100000
**:     CM: COMPUTE SIGN OF RESULT
        AR: R5B=R5B XOR R3A     PC: UW
        CC: C=0,N=N,Z=Z,V=V

6753 M      000301    006462    030440    000600    000060    130000
**:     CM: COMPUTE EXPONENT / FINISHED
        AR: R4B=R4B+R2A+<0>     PC: UW
        BR: CRN(1)

6754 M      000433    007500    140440    041664    000077    130000
**:     CM: RETURN IF CNTR = 0
        AR: R15B=R15B,SRD(C),SQ(LSBR)   PC: 4B,CCL
        CC: C=LSBQ,N=N,Z=Z,V=V  BR: CRN(/CNTR)
```

```
6755 M      000301    007573    030440    005600    000060    100000
**: MUL.56A
        CM: LOW 32 BIT SUM
        AR: R15B=R15B+R13A+<0>  PC: 4B,MM

6756 M      000501    057472    000440    005600    000060    056754
**:     CM: HIGH 32 BIT SUM
        AR: R14B=R14B+R12A+<C'>,SRD(0)  PC: 4B,MM
        CC: C=LSBR,N=N,Z=Z,V=V  BR: CBN(1) [.-2]
```

```
        PG:     56-BIT DIVISION ROUTINE

CM:     DIVIDEND IN R12/R13 (8B)
        DIVISOR IN R14/R15 (8B)
        FRACTION RESULT IN R14/R15 (8B)
        EXPONENT RESULT IN R4 (UW)
        SIGN RESULT IN R5 (UW)
        RESULT ROUNDED(N=1)/TRUNCATED(N=0)
        Z NOT USED
        C USED
        V SET = 1

6757 M      155137    000000    000032    000200    000060    100000
**: DIV.56
        CM: PRESET COUNTER
        AR: CNTR=[+32]

6760 M      000111    007573    030440    001600    000060    100000
**:     CM: CHECK DIVISOR > DIVIDEND
        AR: R15B-R13A-<0>               PC: 4B

6761 M      000111    057472    100440    001600    000060    100000
**:     AR: R14B-R12A-<C'>              PC: 4B
        CC: C=PN,N=N,Z=Z,V=V

6762 M      150315    116464    000200    000600    000040    056765
**:     CM: CONDITIONAL EXPONENT UPDATE
        AR: R4B=R4A-[+200]-</C>         PC: UW
        BR: CBN(/C) [.+3]

6763 M      000733    007500    100440    001600    000060    100000
**:     CM: SHIFT DIVISOR
        AR: R15B=R15B,SRU(0)            PC: 4B
        CC: C=MSBR,N=N,Z=Z,V=V

6764 M      000733    007400    030440    001604    000060    100000
```

```
6765 M      000711    107375    160600    001600    000060    076777
**:     CM: EXECUTE 26. CYCLES
        AR: R13B=R13B-R15A-<1>,SRU(0)   PC: 4B
        CC: C=0,V=PN,N=N,Z=Z    BR: CJN(1) [DIV.56A+1]

6766 M      000372    007100    030440    001600    000060    100000
**:     CM: SAVE BITS COMPUTED
        AR: R11B=/Q             PC: 4B

6767 M      155137    000000    000040    000200    000060    076776
**:     CM: DO REMAINING 32. BITS
        AR: CNTR=[+40]          BR: CJN(1) [DIV.56A]

6770 M      000534    007471    000440    001600    000060    100000
**:     CM: DO A RIGHT SHIFT TO GET RESULT AND ROUNDING BIT
        AR: R14B=R11A,SRD(0)            PC: 4B
        CC: C=LSBR,N=N,Z=Z,V=V

6771 M      000572    007500    000440    001604    000060    100000
**:     AR: R15B=/Q,SRD(C)              PC: 4B
```

56-BIT DIVISION ROUTINE

```
        CC: C=LSBR,N=N,Z=Z,V=V

6772 M      150345    007474    000377    000600    000003    077100
**:     CM: MASK JUNK / ROUND IF REQUIRED
        AR: R14B=R14A AND [+377]        PC: UW
        BR: CJN(N) [NORM.56A]

6773 M      000361    006563    160440    000600    000060    100000
**:     CM: COMPUTE SIGN OF RESULT
        AR: R5B=R5B XOR R3A             PC: UW
        CC: C=0,N=N,Z=Z,V=V

6774 M      000321    106462    030440    000600    000060    130000
**:     CM: COMPUTE EXPONENT / FINISHED
        AR: R4B=R2A-R4B-<1>             PC: UW
        BR: CRN(1)

6775 M      000633    007200    100440    041624    000077    130000
**:     CM: RETURN IF CNTR=0
        AR: R12B=R12B,SRU(C),SQ(MSBR)   PC: 4B,CCL
        CC: C=MSBR,N=N,Z=Z,V=V          BR: CRN(/CNTR)

6776 M      000701    117375    030600    007600    000060    100000
**: DIV.56A
        CM: LOW 32. BIT OPERATION
        AR: R13B=R13B+R15A+</C>,SRU(0)  PC: 4B,MNR
        CC: C=C,V=PN,N=N,Z=Z

6777 M      000301    057274    020760    007600    000060    056775
**:     CM: HIGH 32. BIT OPERATION
        AR: R12B=R12B+R14A+<C'>         PC: 4B,MNR
        CC: C=V,V=1,N=N,Z=Z             BR: CBN(1) [.-2]
```

```
        PG:     56-BIT OPERAND ALIGNMENT

CM:     ARGUMENTS PRESHIFTED ONE BIT (*2)
        R11 (UW) USED AS TEMPORARY
        C IS USED
        N,V,Z NOT USED

7000 M      000733    007300    100440    001600    000060    100000
**: AL.56
        CM: PRESHIFT FRACTIONS
        AR: R13B=R13B,SRU(0)    PC: 4B
        CC: C=MSBR,N=N,Z=Z,V=V

7001 M      000733    007200    030440    001604    000060    100000
**:     AR: R12B=R12B,SRU(C)    PC: 4B

7002 M      000733    007500    100440    001600    000060    100000
**:     AR: R15B=R15B,SRU(0)    PC: 4B
        CC: C=MSBR,N=N,Z=Z,V=V

7003 M      000733    007400    030440    001604    000060    100000
**:     AR: R14B=R14B,SRU(C)    PC: 4B

7004 M      001111    106462    030440    000600    000060    100000
**:     CM: COMPUTE EXPONENT DIFFERENCE
        AR: D=R4B-R2A-<1>       PC: UW

7005 M      005337    007100    100440    000600    000012    130000
**:     CM: RETURN IF ALIGNED
        AR: CNTR;R11B=D         PC: UW
        CC: C=PN,N=N,Z=Z,V=V    BR: CRN(Z')

7006 M      150305    117171    000020    006600    000000    057030
**:     CM: BRANCH IF A>B (IE. CNTR < 0)
        CM: IF C=0 THEN R11A-[+20]-<1> IS COMPUTED
        CM: IF C=1 THEN R11A+[+20]+<0> IS COMPUTED
```

```
        BR: CBN(C) [AL.56B]

7007 M      000334    006264    030440    000600    000053    057013
**: AL.56A
        CM: BRANCH IF 16. OR MORE SHIFTS REQUIRED
        AR: R2B=R4A             PC: UW
        BR: CBN(/N') [.+4]

7010 M      000533    007200    000440    001600    000060    100000
**:     CM: ELSE DO SHIFTS
        AR: R12B=R12B,SRD(0)    PC: 4B
        CC: C=LSBR,N=N,Z=Z,V=V

7011 M      000533    007300    030440    041604    000077    130000
**:     CM: RETURN IF FINISHED
        AR: R13B=R13B,SRD(C)    PC: 4B,CCL
        BR: CRN(/CNTR)

7012 M      000533    007200    000440    001600    000060    057011
**:     AR: R12B=R12B,SRD(0)    PC: 4B
        CC: C=LSBR,N=N,Z=Z,V=V  BR: CBN(1) [.-1]
```

56-BIT OPERAND ALIGNMENT

```
7013 M      151215    107171    000020    000600    000060    100000
**:     CM: CHECK IF 32. OR MORE SHIFTS REQUIRED
        AR: R11B=R11A-[+20]-<1>,D=R11A  PC: UW

7014 M      005137    000000    110440    000200    000053    057022
**:     CM: SET COUNTER / BRANCH IF MORE
        AR: CNTR=D      CC: C=PZ,N=N,Z=Z,V=V
        BR: CBN(/N') [.+6]

7015 M      001242    007272    030440    000600    000060    100000
**:     CM: ELSE DO 16. BIT SHIFT
        AR: R12B=0,D=R12A       PC: UW

7016 M      001237    007272    030440    000200    000060    100000
**:     AR: R12B=D,D=R12A       PC: LW

7017 M      001237    007373    030440    000600    000060    100000
**:     AR: R13B=D,D=R13A       PC: UW

7020 M      000337    007300    030440    000200    000040    057010
**:     AR: R13B=D              PC: LW
        BR: CBN(/C) [AL.56A+1]

7021 M      000100    000000    030440    000200    000060    120000
**:     BR: CRR(1)

7022 M      150115    000071    000030    000600    000060    100000
**:     CM: CHECK IF MORE THAN 56. SHIFTS REQUIRED
        AR: R11A-[+30]-<0>      PC: UW

7023 M      005134    000071    110440    000600    000053    057025
**:     CM: BRANCH IF MORE
        AR: CNTR=R11A           PC: UW
        CC: C=PZ,N=N,Z=Z,V=V    BR: CBN(/N') [.+2]
```

```
7024 M      000334    007372    030440    001600    000060    057026
**:     CM: ELSE DO 32. BIT SHIFT
        AR: R13B=R12A           PC: 4B
        BR: CBN(1) [.+2]

7025 M      000342    007300    170440    001600    000060    100000
**:     CM: ZERO IF MORE
        AR: R13B=0              PC: 4B
        CC: C=1,N=N,Z=Z,V=V

7026 M      000342    007200    160440    001600    000000    130000
**:     CM: CLEAR UPPER 32. BITS
        AR: R12B=0              PC: 4B
        CC: C=0,N=N,Z=Z,V=V     BR: CRN(C)

7027 M      150315    107171    000020    000600    000060    057007
**:     CM: CHECK IF 16. OR MORE SHIFTS LEFT
        AR: R11B=R11A-[+20]-<1> PC: UW
        BR: CBN(1) [AL.56A]

7030 M      000334    006462    030440    000600    000017    057034
```

56-BIT OPERAND ALIGNMENT

```
**: AL.56B
        CM: BRANCH IF 16. OR MORE SHIFTS NEEDED
        AR: R4B=R2A             PC: UW
        BR: CBN(BLE') [.+4]

7031 M      000533    007400    000440    001600    000060    100000
**:     CM: ELSE DO SHIFTS
        AR: R14B=R14B,SRD(0)    PC: 4B
        CC: C=LSBR,N=N,Z=Z,V=V

7032 M      000533    007500    030440    041604    000077    130000
**:     CM: RETURN IF FINISHED
        AR: R15B=R15B,SRD(C)    PC: 4B,CCL
        BR: CRN(/CNTR)

7033 M      000533    007400    000440    001600    000060    057032
**:     AR: R14B=R14B,SRD(0)    PC: 4B
        CC: C=LSBR,N=N,Z=Z,V=V  BR: CBN(1) [.-1]

7034 M      151205    007171    000020    000600    000060    100000
**:     CM: CHECK IF 32. OR MORE SHIFTS NEEDED
        AR: R11B=R11A+[+20]+<0>,D=R11A  PC: UW

7035 M      005137    000000    110440    000200    000017    057043
**:     CM: SET COUNTER / BRANCH IF 32. OR MORE
        AR: CNTR=D
        CC: C=PZ,N=N,Z=Z,V=V    BR: CBN(BLE') [.+6]

7036 M      001242    007474    030440    000600    000060    100000
**:     CM: ELSE DO 16. BIT SHIFT
        AR: R14B=0,D=R14A       PC: UW

7037 M      001237    007474    030440    000200    000060    100000
**:     AR: R14B=D,D=R14A       PC: LW
```

```
**:     AR: R15B=D,D=R15A       PC: UW

7041 M      000337    007500    030440    000200    000040    057031
**:     AR: R15B=D              PC: LW
        BR: CBN(/C) [AL.56B+1]

7042 M      000100    000000    030440    000200    000060    120000
**:     BR: CRR(1)

7043 M      150105    000071    000030    000600    000060    100000
**:     CM: CHECK IF MORE THAN 56. SHIFTS NEEDED
        AR: R11A+[+30]+<0>      PC: UW

7044 M      005134    000071    110440    000600    000013    057046
**:     CM: BRANCH IF MORE
        AR: CNTR=R11A           PC: UW
        CC: C=PZ,N=N,Z=Z,V=V    BR: CBN(N') [.+2]

7045 M      000334    007574    030440    001600    000060    057047
**:     CM: ELSE DO 32. BIT SHIFT
        AR: R15B=R14A           PC: 4B
```

56-BIT OPERAND ALIGNMENT

```
        BR: CBN(1) [.+2]

7046 M      000342    007500    170440    001600    000060    100000
**:     CM: CLEAR ARGUMENT
        AR: R15B=0              PC: 4B
        CC: C=1,N=N,Z=Z,V=V

7047 M      000342    007400    160440    001600    000000    130000
**:     AR: R14B=0              PC: 4B
        CC: C=0,N=N,Z=Z,V=V     BR: CRN(C)

7050 M      150305    007171    000020    000600    000060    057030
**:     CM: CHECK IF 16. OR MORE SHIFTS NEEDED
        AR: R11B=R11A+[+20]+<0> PC: UW
        BR: CBN(1) [AL.56B]
```

```
        PG:     56-BIT POST NORMALIZATION

CM:     EXPECTS ARGUMENTS PRESHIFTED (*2)
        V NOT USED
        C,Z = 1 IF RESULT = 0
        RESULT IS ROUNDED(N=1)/TRUNCATED(N=0)

7051 M      000042    000000    030440    001600    000060    100000
**: NORM.56
        CM: CLEAR SHIFTING REGISTER
        AR: Q=0                 PC: 4B

7052 M      000333    007400    034440    001600    000060    100000
**:     CM: CHECK FOR ZERO RESULT
        AR: R14B=R14B           PC: 4B
        CC: Z=PZ,C=C,N=N,V=V

7053 M      000333    007500    036040    001600    000042    057065
**:     CM: BRANCH IF HIGH ORDER NOT ZERO
        AR: R15B=R15B           PC: 4B
        CC: Z=PZAZ,C=C,N=N,V=V  BR: CBN(/Z) [.+12]

7054 M      000132    000000    036040    001600    000060    100000
**:     CM: CHECK LOWEST ORDER
        AR: Q                   PC: 4B
        CC: Z=PZAZ,C=C,N=N,V=V

7055 M      001132    000000    170440    000600    000002    130000
**:     CM: RETURN IF RESULT = 0
        AR: D=Q                 PC: UW
        CC: C=1,N=N,Z=Z,V=V     BR: CRN(Z)

7056 M      001237    007575    030440    000200    000060    100000
**:     CM: ELSE DO A 16. BIT SHIFT
        AR: R15B=D,D=R15A       PC: LW
```

```
**:     AR: R15B=D,D=R15A       PC: UW

7060 M      000337    007400    164440    000200    000060    100000
**:     AR: R14B=D              PC: LW
        CC: Z=PZ,C=0,N=N,V=V

7061 M      001132    000000    030440    000200    000060    100000
**:     CM: MOVE Q REGISTER BITS
        AR: D=Q                 PC: LW

7062 M      000037    000000    030440    000600    000012    057064
**:     AR: Q=D                 PC: UW
        BR: CBN(Z') [.+2]

7063 M      000042    000000    030440    000200    000060    100000
**:     AR: Q=0                 PC: LW

7064 M      150315    106464    000020    000600    000060    057053
**:     CM: UPDATE EXPONENT
        AR: R4B=R4A-[+20]-<1>   PC: UW
        BR: CBN(1) [NORM.56+2]
```

56-BIT POST NORMALIZATION

```
7065 M      155137    000000    000002    000200    000060    100000
**:     CM: PRESET COUNTER
        AR: CNTR=[+2]

7066 M      000533    007400    000440    001600    000060    100000
**:     CM: PRESIFT TWICE
        AR: R14B=R14B,SRD(0)    PC: 4B
        CC: C=LSBR,N=N,Z=Z,V=V

7067 M      000433    007500    167440    001664    000042    057066
**:     AR: R15B=R15B,SRD(C),SQ(LSBR)   PC: 4B
        CC: Z=1,C=0,N=N,V=V     BR: CBN(/Z) [.-1]

7070 M      000633    007500    100440    001603    000000    057072
**:     CM: SHIFT UNTIL A '1' BIT FOUND
        AR: R15B=R15B,SRU(MSBQ),SQ(0)   PC: 4B
        CC: C=MSBR,N=N,Z=Z,V=V  BR: CBN(C) [.+2]

7071 M      000733    007400    100440    041404    000060    057070
**:     AR: R14B=R14B,SRU(C)    PC: 3B,CCL
        CC: C=MSBR,N=N,Z=Z,V=V  BR: CBN(1) [.-1]

7072 M      000433    007500    030440    001664    000060    100000
**:     CM: RESTORING SHIFTS
        AR: R15B=R15B,SRD(C),SQ(LSBR)   PC: 4B

7073 M      000533    007400    000440    001401    000060    100000
**:     AR: R14B=R14B,SRD(1)    PC: 3B
        CC: C=LSBR,N=N,Z=Z,V=V

7074 M      000533    007500    000440    001604    000060    100000
**:     AR: R15B=R15B,SRD(C)    PC: 4B
        CC: C=LSBR,N=N,Z=Z,V=V
```

```
**:     CM: ADJUST EXPONENT / BRANCH IF ROUNDING
        AR: D=CNTR              BR: CBN(N) [.+2]


7076 M      030305    006464    167040    000600    000060    130000
**:     CM: COMPUTE EXPONENT AND RETURN
        AR: R4B=R4A+DSX7+<0>    PC: UW
        CC: C=0,Z=0,N=N,V=V     BR: CRN(1)


7077 M      030305    006464    037040    000600    000040    130000
**:     CM: COMPUTE EXPONENT / RETURN IF ROUNDING BIT = 0
        AR: R4B=R4A+DSX7+<0>    PC: UW
        CC: C=C,Z=0,N=N,V=V     BR: CRN(/C)


7100 M      000303    017500    137040    001600    000040    130000
**: NORM.56A
        CM: AND ROUND RESULT
        AR: R15B=R15B+<C>       PC: 4B
        CC: C=PC,Z=0,N=N,V=V    BR: CRN(/C)


7101 M      000303    017400    137040    001400    000040    130000
**:     AR: R14B=R14B+<C>       PC: 3B
```

56-BIT POST NORMALIZATION

```
        CC: C=PC,Z=0,N=N,V=V    BR: CRN(/C)

7102 M      000303    016400    030440    000600    000040    130000
**:     CM: CONDITIONALLY UPDATE EXPONENT
        AR: R4B=R4B+<C>         PC: UW
        BR: CRN(/C)

7103 M      000533    007400    007040    001404    000060    100000
**:     CM: SHIFT FRACTION
        AR: R14B=R14B,SRD(C)    PC: 3B
        CC: C=LSBR,Z=0,N=N,V=V

7104 M      000533    007500    167040    001604    000060    130000
**:     AR: R15B=R15B,SRD(C)    PC: 4B
        CC: C=0,Z=0,N=N,V=V     BR: CRN(1)

        NA: MORGEND=.
```

```
        PG:     FLOATING POINT PROCESSOR OVERLAY

        CM:       THE FOLLOWING INSTRUCTION CODE
                DEFINITIONS OVERLAY THE ARB11 CODE TO ADD THE FOLLOWING
                FLOATING POINT INSTRUCTIONS -

                PDP-11/34,45,60,70 AND LSI-11/23 COMPATIBLE

                        CFCC    170000
                        SETF    170001
                        SETI    170002
                        SETD    170011
                        SETI    170012
                        LDFPS   1701SRC
                        STFPS   1702DST
                        STST    1703DST
                        CLRF/D  1704FDST
                        TSTF/D  1705FDST
                        ABSF/D  1706FDST
                        NEGF/D  1707FDST
                        MULF/D  171(AC)FSRC
                        MODF/D  171(AC+4)FSRC
                        ADDF/D  172(AC)FSRC
                        LDF/D   172(AC+4)FSRC
                        SUBF/D  173(AC)FSRC
                        CMPF/D  173(AC+4)FSRC
                        STF/D   174(AC)FDST
                        DIVF/D  174(AC+4)FSRC
                        STEXP   175(AC)DST
                        STCFI/L
                        STCDI/L 175(AC+4)DST
                        STCFD
                        STCDF   176(AC)FDST
                        LDEXP   176(AC+4)SRC
                        LDCIF/D
                        LDCLF/D 177(AC)SRC
```

```
        PG:     FPP CODING NOTES

CM:     NOTES ON CODING -

        1) CONDITION CODES ARE CHANGED DURING OPERAND
           ACCESS. MEMORY TRAPS WILL HAVE INVALIDS CC'S

        2) FIS INSTRUCTION SET MUST BE PRESENT TO ALLOW
           ENTRY TO THE FOLLOWING ROUTINES
                A.GETARG
                B.GETARG
                ARG.SAV
                CMB.SEF
                ADD.24
                MUL.24
                DIV.24
                NORM.24
                ADD.56
                MUL.56
                DIV.56
                NORM.56
                FIS.TRAP

        3) THE FOLLOWING ROUTINES MUST BE
           PRESENT FOR RESERVED INSTRUCTION PROCESSING
                RSRVINST
                RSRVSTAT

        4) THE ACCUMULATORS AND CERTAIN STATUS REGISTERS
           ARE KEPT IN MAIN MEMORY. THE LOCATIONS ARE DEFINED HERE

                NA: AC0.FPP=+167000
                NA: AC1.FPP=AC0.FPP+10
                NA: AC2.FPP=AC0.FPP+20
                NA: AC3.FPP=AC0.FPP+30
                NA: AC4.FPP=AC0.FPP+40
```

```
NA: AC6.FPP=AC0.FPP+60
NA: AC7.FPP=AC0.FPP+70


NA: FEA.FPP=AC0.FPP+100
NA: FEC.FPP=AC0.FPP+102
```

```
        PG:     FPP REGISTER USAGE

CM:     THE FPP MAKES USE OF THE FOLOWING REGISTERS

        THE A ARGUMENT -
                R2 (UW) - EXPONENT
                R3 (UW) - SIGN OF FRACTION
                R12 (4B) - HIGH FRACTION
                R13 (4B) - LOW FRACTION

        THE B ARGUMENT -
                R4 (UW) - EXPONENT
                R5 (UW) - SIGN OF FRACTION
                R14 (4B) - HIGH FRACTION
                R15 (4B) - LOW FRACTION

        FPP STATUS
                R16 (UW) - FLOATING POINT STATUS (WITHOUT FCC'S)
                R6 (UW) - FLOATING POINT CC'S

        TEMPORARYS -
                R0 (UW) - CONDITION CODE SAVE
                R1 (UW) - SCRATCH
                R10 (LW) - ADDRESS CALCULATIONS
                R10 (UW) - SCRATCH
                R11 (4B) - SCRATCH
                Q (4B) - SCRATCH
                SCE5 (SCR2) - INDEX BRANCH REGISTER
                SCR1 (SCE6) - INSTRUCTION POINTER
```

```
        PG:     FLOATING INSTRUCTION CODE SPECIFICATIONS

        .=: IORIGIN+0

6000 I      007256    002000
**:     MA: SET.FPS     PS:  EX

6001 I      007276    012040
**:     MA: LDFPS       PS: DST,EX      IN: SP

6002 I      007301    012020
**:     MA: STFPS       PS: DST,EX      IN: NDA

6003 I      007311    002000
**:     MA: STST        PS: EX

6004 I      007320    002000
**:     MA: CLRF        PS: EX

6005 I      007323    002000
**:     MA: TSTF        PS: EX

6006 I      007327    002000
**:     MA: ABSF        PS: EX

6007 I      007333    002000
**:     MA: NEGF        PS: EX

6010 I      007342    002000
**:     MA: MULF        PS: EX

6011 I      007343    002000
**:     MA: MULF+1      PS: EX

6012 I      007344    002000
**:     MA: MULF+2      PS: EX
```

```
6013 I      007345    002000
**:     MA: MULF+3      PS: EX

6014 I      007354    002000
**:     MA: MODF        PS: EX

6015 I      007355    002000
**:     MA: MODF+1      PS: EX

6016 I      007356    002000
**:     MA: MODF+2      PS: EX

6017 I      007357    002000
**:     MA: MODF+3      PS: EX

6020 I      007445    002000
**:     MA: ADDF        PS: EX

6021 I      007446    002000
**:     MA: ADDF+1      PS: EX
```

FLOATING INSTRUCTION CODE SPECIFICATIONS

```
6022 I      007447    002000
**:     MA: ADDF+2      PS: EX

6023 I      007450    002000
**:     MA: ADDF+3      PS: EX

6024 I      007461    002000
**:     MA: LDF         PS: EX

6025 I      007462    002000
**:     MA: LDF+1       PS: EX

6026 I      007463    002000
**:     MA: LDF+2       PS: EX

6027 I      007464    002000
**:     MA: LDF+3       PS: EX

6030 I      007473    002000
**:     MA: SUBF        PS: EX

6031 I      007474    002000
**:     MA: SUBF+1      PS: EX

6032 I      007475    002000
**:     MA: SUBF+2      PS: EX

6033 I      007476    002000
**:     MA: SUBF+3      PS: EX

6034 I      007501    002000
**:     MA: CMPF        PS: EX

6035 I      007502    002000
**:     MA: CMPF+1      PS: EX
```

```
6036 I      007503    002000
**:     MA: CMPF+2      PS: EX

6037 I      007504    002000
**:     MA: CMPF+3      PS: EX

6040 I      007517    002000
**:     MA: STF         PS: EX

6041 I      007520    002000
**:     MA: STF+1       PS: EX

6042 I      007521    002000
**:     MA: STF+2       PS: EX

6043 I      007522    002000
**:     MA: STF+3       PS: EX

6044 I      007530    002000
**:     MA: DIVF        PS: EX
```

FLOATING INSTRUCTION CODE SPECIFICATIONS

```
6045 I      007531    002000
**:     MA: DIVF+1      PS: EX

6046 I      007532    002000
**:     MA: DIVF+2      PS: EX

6047 I      007533    002000
**:     MA: DIVF+3      PS: EX

6050 I      007545    012020
**:     MA: STEXP       PS: DST,EX      IN: NDA

6051 I      007546    012020
**:     MA: STEXP+1     PS: DST,EX      IN: NDA

6052 I      007547    012020
**:     MA: STEXP+2     PS: DST,EX      IN: NDA

6053 I      007550    012020
**:     MA: STEXP+3     PS: DST,EX      IN: NDA

6054 I      007562    002000
**:     MA: STCFI       PS: EX

6055 I      007563    002000
**:     MA: STCFI+1     PS: EX

6056 I      007564    002000
**:     MA: STCFI+2     PS: EX

6057 I      007565    002000
**:     MA: STCFI+3     PS: EX

6060 I      007630    002000
**:     MA: STCFD       PS: EX
```

```
6061 I      007631    002000
**:     MA: STCFD+1     PS: EX

6062 I      007632    002000
**:     MA: STCFD+2     PS: EX

6063 I      007633    002000
**:     MA: STCFD+3     PS: EX

6064 I      007653    012040
**:     MA: LDEXP       PS: DST,EX      IN: SP

6065 I      007654    012040
**:     MA: LDEXP+1     PS: DST,EX      IN: SP

6066 I      007655    012040
**:     MA: LDEXP+2     PS: DST,EX      IN: SP

6067 I      007656    012040
**:     MA: LDEXP+3     PS: DST,EX      IN: SP
```

FLOATING INSTRUCTION CODE SPECIFICATIONS

```
6070 I      007664    002000
**:     MA: LDCIF       PS: EX

6071 I      007665    002000
**:     MA: LDCIF+1     PS: EX

6072 I      007666    002000
**:     MA: LDCIF+2     PS: EX

6073 I      007667    002000
**:     MA: LDCIF+3     PS: EX

6074 I      007676    002000
**:     MA: LDCDF       PS: EX

6075 I      007677    002000
**:     MA: LDCDF+1     PS: EX

6076 I      007700    002000
**:     MA: LDCDF+2     PS: EX

6077 I      007701    002000
**:     MA: LDCDF+3     PS: EX
```

```
        PG:     GET ACCUMULATOR INTO A

        .=: MORGEND

CM:     THIS CODE GETS THE A ACCUMULATOR AT THE ADDRESS IN R10 (LW)
        SET V=0 FOR FLOATING / V=1 FOR DOUBLE
        AND EXITS WITH
        C,Z=1 IF EXPONENT = 0
        N=1 IF ACCUMULATOR < 0
        R2 (UW) = EXPONENT
        R3 (UW) = SIGN OF FRACTION
        R12 (4B) = FRACTION (24-BITS)
        R13 (4B) = FRACTION (32-BITS)

7105 M      002134    000070    030440    000200    170060    100000
**: A.GETACC
        AR: AD=R10A             IO: DATI(OFF)

7106 M      000342    007200    030440    000600    000060    100000
**:     CM: PRESET HIDDEN BIT OF FRACTION
        AR: R12B=0              PC: UW

7107 M      152105    000070    000002    020200    000060    100000
**:     AR: AD=R10A+[+2]+<0>    PC: LW,WIOL

7110 M      000737    006200    107050    000600    170060    076555
**:     CM: SAVE HIGH ORDER IN TEMPORARY
            GO BUILD SIGN, EXPONENT, AND FRACTION
        AR: R2B=D,SRU(0)        PC: UW
        CC: C=MSBR,N=PN,Z=0,V=V
        IO: DATI(OFF)           BR: CJN(1) [A.BLDSEF]

7111 M      152105    000070    000004    000200    170060    100000
**:     CM: GET DOUBLE FRACTION
        AR: AD=R10A+[+4]+<0>    IO: DATI(OFF)
```

```
**:     AR: AD=R10A+[+6]+<0>    PC: LW,WIOL


7113 M      000337    007300    030440    000600    170060    100000
**:     AR: R13B=D              PC: UW
        IO: DATI(OFF)


7114 M      002237    007367    030440    010200    000060    120000
**:     CM: DUMBY LOAD AD
        AR: R13B=D,AD=R7A       PC: LW,WIOH
        BR: CRR(1)
```

```
        PG:     GET ACCUMULATOR INTO B

CM:     THIS CODE GETS THE B ACCUMULATOR AT THE ADDRESS IN R10 (LW)
        SET V=0 FOR FLOATING / V=1 FOR DOUBLE
        AND EXITS WITH
        C,Z=1 IF EXPONENT IS = 0
        N=1 IF ACCUMULATOR IS < 0
        R4 (UW) = EXPONENT
        R5 (UW) = SIGN OF FRACTION
        R14 (4B) = FRACTION (24-BITS)
        R15 (4B) = FRACTION (32-BITS)

7115 M      000133    007600    030600    000400    000060    057117
**: B.GTAC
        CM: SET LENGTH FLAG
        AR: R16B                PC: UB
        CC: C=C,N=N,Z=Z,V=PN    BR: CBN(1) [.+2]

7116 M      002134    000070    030440    000200    170060    100000
**: B.GETACC
        AR: AD=R10A             IO: DATI(OFF)

7117 M      000342    007400    030440    000600    000060    100000
**:     CM: CLEAR HIGH FRACTION
        AR: R14B=0              PC: UW

7120 M      152105    000070    000002    020200    000060    100000
**:     AR: AD=R10A+[+2]+<0>    PC: LW,WIOL

7121 M      000737    006400    107050    000600    170060    076572
**:     CM: SAVE HIGH ORDER IN TEMPORARY
            GO BUILD SIGN, EXPONENT, AND HIGH FRACTION
        AR: R4B=D,SRU(0)        PC: UW
        CC: C=MSBR,N=PN,Z=0,V=V
        IO: DATI(OFF)           BR: CJN(1) [B.BLDSEF]
```

```
**:     CM: GET DOUBLE FRACTION
        AR: AD=R10A+[+4]+<0>    IO: DATI(OFF)

7123 M      152105    000070    000006    020200    000060    100000
**:     AR: AD=R10A+[+6]+<0>    PC: LW,WIOL

7124 M      000337    007500    030440    000600    170060    100000
**:     AR: R15B=D              PC: UW
        IO: DATI(OFF)

7125 M      002237    007567    030440    010200    000060    120000
**:     CM: DUMBY LOAD AD
        AR: R15B=D,AD=R7A       PC: LW,WIOH
        BR: CRR(1)
```

```
        PG:     GET OPERAND UTILITY

CM:     SOURCE OPERAND IS LOADED AND CHECKED FOR -0
        IF NOT FROM AN ACCUMULATOR
        TRAP OCCURS TO FIUV.TRP IF -0 AND FIUV=1
        ACCUMULATOR OPERAND IS THEN LOADED

7126 M      150115    100067    000002    000200    000060    100110
**: GETOPS
        CM: SAVE INSTRUCTION POINTER
        AR: SCR1=R7A-[+2]-<1>

7127 M      070537    007000    030440    000600    000060    077137
**:     CM: GET SOURCE ARGUMENT
        AR: R10B=IR5,SRD(0)     PC: UW
        BR: CJN(1) [FSRC+1]

7130 M      150137    000000    007133    000200    040460    100120
**:     CM: SETUP END BRANCH
        AR: SCE5=[.+3]          SP: CLR,SE5

7131 M      002334    007071    030440    000200    000000    077716
**:     CM: CHECK FOR -0 IF NOT MODE 0
        AR: AD;R10B=R11A        BR: CJN(C) [FIUV.FPP+1]

7132 M      150137    000000    006070    000200    000060    164120
**:     CM: TRAP IF -0 AND FIUV=1
        AR: SCE5=[BGN]          BR: CBIN(1) <SCR>

7133 M      150337    006100    000040    000600    171060    077106
**:     CM: ELSE GET ACCUMULATOR
        AR: R1B=[+40]           PC: UW
        IO: IDATI(OFF)          BR: CJN(1) [A.GETACC+1]

7134 M      000141    006176    160451    000600    040460    120000
**:     CM: SET TRUNCATION FLAG / RETURN TO CALLER
```

```
        CC: C=0,Z=Z,N=PZ,V=V    SP: CLR,SE5

        BR: CRR(1)
```

```
        PG:     FPP SOURCE ADDRESSING MODES

CM:     SET C=1 FOR NO DATA ACCESS, ELSE SET C=0
        SET Z=1 FOR FLOATING (V=0) / DOUBLE (V=1)
        SET Z=0 FOR FPS LENGTH

7135 M      150115    100067    000002    000200    000060    100110
**: FSRC.SAV
        CM: SAVE INSTRUCTION LOCATION
        AR: SCR1=R7A-[+2]-<1>

7136 M      070537    007000    030440    000600    000060    100000
**: FSRC
        CM: USING DST PART OF INSTRUCTION FIND MODE
        AR: R10B=IR5,SRD(0)     PC: UW

7137 M      150345    007070    000034    000600    000060    077165
**:     CM: MASK JUNK
        AR: R10B=R10A AND [+34] PC: UW
        BR: CJN(1) [FS.M4+3]

7140 M      070737    007000    037056    000200    000002    164000
**:     CM: USE SPECIFIED MODE
        AR: R10B=IR5,SRU(0)
        CC: V=V,C=C,N=0,Z=0     BR: CBIN(Z) <SCR>

7141 M      000133    007600    037216    000400    000060    144000
**:     CM: ELSE USE FPS LENGTH
        AR: R16B                PC: UB
        CC: C=C,V=PN,N=0,Z=0    BR: CBIC(1) <SCR>

7142 M      000701    007070    030440    000200    000060    100000
**: FS.M0
        CM: ACCUMULATOR R IS THE OPERAND
        AR: R10B=R10B+R10A+<0>,SRU(0)
```

```
**:     CM: RETURN IF ONLY ADDRESS NEEDED
        AR: AD;R10B=R10A+[AC0.FPP]+<0>
        BR: CRN(C)

7144 M      000342    007400    030440    000600    171060    077120
**:     CM: ELSE GET ACCUMULATOR
        AR: R14B=0              PC: UW
        IO: IDATI(OFF)          BR: CJN(1) [B.GETACC+2]

7145 M      000100    000000    160440    000200    000060    120000
**:     CM: CLEAR C / NO FIUV FROM ACC
        CC: C=0,N=N,Z=Z,V=V     BR: CRR(1)

        .=: FS.M0+4

7146 M      002334    007020    030440    000200    000000    130000
**: FS.M1
        CM: (R) IS ADDRESS
        AR: AD;R10B=RA(DST)     BR: CRN(C)

7147 M      000342    007400    030440    000600    141060    056564
```

```
FPP SOURCE ADDRESSING MODES


**:     AR: R14B=0              PC: UW
        IO: IDATI(CMI)          BR: CBN(1) [B.GETARG+2]


7150 M      070337    006100    030440    000600    000060    076530
**: FDST
        CM: CHECK MODE / GO BUILD SEF
        AR: R1B=IR5             PC: UW
        BR: CJN(1) [CMB.SEF]


7151 M      150145    000061    000070    000600    000060    057177
**:     AR: R1A AND [+70]       PC: UW
        BR: CBN(1) [FS.M7+1]


        .=: FS.M0+10


7152 M      150115    100070    000056    000200    000060    100000
**: FS.M2
        CM: (R) IS ADDRESS; (R) + (4 OR 10) / IF DST=27, (R) + 2
        CM: MODE 2 / CHECK IMMEDIATE
        AR: R10A-[+56]-<1>


7153 M      151205    002020    000002    000200    000012    057157
**:     CM: BRANCH IF IMMEDIATE
        AR: RB(DST)=RA(DST)+[+2]+<0>,D=RA(DST)
        BR: CBN(Z') [FS.IM]


7154 M      150305    002020    000002    000200    000041    057170
**:     AR: RB(DST)=RA(DST)+[+2]+<0>
        BR: CBN(/V) [FS.M5+2]


7155 M      150305    002020    000004    000200    000060    057170
**:     AR: RB(DST)=RA(DST)+[+4]+<0>
        BR: CBN(1) [FS.M5+2]


        .=: FS.M0+14
```

```
7156 M      152205    002020    000002    000200    140060    057170
**: FS.M3
        CM: (R) IS ADDRESS OF ADDRESS; (R) + 2
        AR: RB(DST)=RA(DST)+[+2]+<0>,AD=RA(DST)
        IO: DATI(CMI)           BR: CBN(1) [FS.M5+2]

7157 M      002337    007000    030440    000200    000000    130000
**: FS.IM
        CM: IMMEDIATE MODE
        AR: AD;R10B=D           BR: CRN(C)

7160 M      000342    007500    030440    001600    141041    077161
**:     CM: PUSH DUMBY RETURN IF FLOATING
        AR: R15B=0              PC: 4B
        IO: IDATI(CMI)          BR: CJN(/V) [.+1]

7161 M      000737    006400    107050    010600    000060    057164
**:     CM: BUILD SEF
        AR: R4B=D,SRU(0)        PC: UW,WIOH
        CC: C=MSBR,N=PN,Z=0,V=V BR: CBN(1) [FS.M4+2]
```

FPP SOURCE ADDRESSING MODES

```
        .=: FS.M0+20

7162 M      151315    102020    000004    000200    000041    057167
**: FS.M4
        CM: (R) - (4 OR 10); (R) IS ADDRESS
        AR: D;RB(DST)=RA(DST)-[+4]-<1>
        BR: CBN(/V) [FS.M5+1]

7163 M      151315    102020    000004    000200    000060    057167
**:     AR: D;RB(DST)=RA(DST)-[+4]-<1>
        BR: CBN(1) [FS.M5+1]

7164 M      001342    007400    030440    000600    000060    056572
**:     CM: (FS.IM) B.BLDSEF REMOVES DUMBY RETURN ON FLOATING
            RETURNS TO CALLER OF FSRC
        AR: D;R14B=0            PC: UW
        BR: CBN(1) [B.BLDSEF]

7165 M      150105    000070    007142    000600    040460    120120
**:     CM: (FSRC) SET UP INDEXED JUMP
        AR: SCE5=R10A+[FS.M0]+<0>       PC: UW
        SP: CLR,SE5             BR: CRR(1)

        .=: FS.M0+24

7166 M      152315    102020    000002    000200    140060    100000
**: FS.M5
        CM: (R)-2 ; (R) IS ADDRESS OF ADDRESS
        AR: AD;RB(DST)=RA(DST)-[+2]-<1>
        IO: DATI(CMI)

7167 M      137115    100066    030440    000200    000060    100000
**:     CM: CHECK STACK OVERFLOW
        AR: SOR=R6A-SLR-<1>
```

```
**:     CM: RETURN IF ONLY ADDRESS NEEDED
        AR: AD;R10B=D           PC: LW,WIOH
        BR: CRN(C)

7171 M      000342    007400    030440    000600    141060    056564
**:     AR: R14B=0              PC: UW
        IO: IDATI(CMI)          BR: CBN(1) [B.GETARG+2]

        .=: FS.M0+30

7172 M      152205    006767    000002    000200    140060    100000
**: FS.M6
        CM: (R) + X IS ADDRESS
        AR: R7B=R7A+[+2]+<0>,AD=R7A
        IO: DATI(CMI)

7173 M      002305    007020    030440    010200    000000    130000
**:     CM: FINISHED IF ONLY ADDRESS NEEDED
        AR: AD;R10B=RA(DST)+D+<0>       PC: LW,WIOH
        BR: CRN(C)
```

FPP SOURCE ADDRESSING MODES

```
7174 M      000342    007400    030440    000600    141060    056564
**:     AR: R14B=0              PC: UW
        IO: IDATI(CMI)          BR: CBN(1) [B.GETARG+2]

7175 M      002105    000020    030440    010200    140060    057170
**:     CM: (MODE 7)
        AR: AD=RA(DST)+D+<0>    PC: LW,WIOH
        IO: DATI(CMI)           BR: CBN(1) [FS.M5+2]

        .=: FS.M0+34

7176 M      152205    006767    000002    000200    140060    057175
**: FS.M7
        CM: (R) + X IS ADDRESS OF ADDRESS
        AR: R7B=R7A+[+2]+<0>,AD=R7A
        IO: DATI(CMI)           BR: CBN(1) [FS.M6+3]
```

```
        PG:     FLOATING DESTINATION CODE

CM:     STORES RESULT IN ACCUMULATOR OR DESTINATION
        AT ADDRESS IN R10 (LW)
        CONDITION CODES ARE CHANGED
        C=0, N=1 IF NEGATIVE, Z=1 IF EXP=0
        V IS UNCHANGED FLOATING(V=0)/DOUBLE(V=1)



7177 M      150115    100061    000027    000600    000012    057247
**:     CM: BRANCH IF ACCUMULATOR
        AR: R1A-[+27]-<1>       PC: UW
        BR: CBN(Z') [ACC.SAVA]

7200 M      002134    000070    030440    000200    000052    056521
**:     CM: BRANCH IF NOT IMMEDIATE
        AR: AD=R10A     BR: CBN(/Z') [ARG.SAVA]

7201 M      000100    000000    160740    000200    101260    130000
**:     CM: ELSE IMMEDIATE MODE
        IO: IDATO(CM)
        CC: C=0,V=0,N=N,Z=Z     BR: CRN(1)
```

```
        PG:     INTEGER SOURCE

CM:     C=1 FOR NO DATA ACCESS
          RETURNS WITH
                Z=1 IF MODE ZERO
                N=0 IF V=0 OR IMMEDIATE
        ELSE SET C=0
        Z=1 THEN INTEGER(V=0) / DOUBLE(V=1)
        Z=0 THEN MODE IN FPS USED

7202 M      001134    000020    037440    000200    000000    130000
**: IS.M0
        CM: MODE 0 / REGISTER R IS THE OPERAND
        AR: D=RA(DST)           PC: LW
        CC: N=N,Z=1,C=C,V=V     BR: CRN(C)

7203 M      000342    007500    030456    001600    000060    057237
**:     CM: ELSE BUILD DATA
        AR: R15B=0              PC: 4B
        CC: N=0,C=C,Z=Z,V=V     BR: CBN(1) [B.GETINT]

7204 M      000100    000000    030440    000200    141060    100000
**:     CM: (B.GETINT)
        IO: IDATI(CMI)

7205 M      000337    007500    036040    010200    000060    120000
**:     AR: R15B=D              PC: LW,WIOH
        CC: Z=PZAZ,N=N,C=C,V=V  BR: CRR(1)

        .=: IS.M0+4

7206 M      002334    007020    030440    000200    000000    130000
**: IS.M1
        CM: MODE 1 / (R) IS ADDRESS
        AR: AD;R10B=RA(DST)     BR: CRN(C)
```

```
**:     CM: ELSE GET INTEGER
        AR: R15B=0              PC: 4B
        IO: IDATI(CMI)          BR: CBN(1) [B.GETINT]


7210 M      150115    100067    000002    000200    000060    100110
**: ISRC.SAV
        CM: SAVE INSTRUCTION PC
        AR: SCR1=R7A-[+2]-<1>


7211 M      070537    007000    030440    000600    000060    057217
**: ISRC
        CM: SET UP INDEXED JUMP TO MODE
        AR: R10B=IR5,SRD(0)     PC: UW
        BR: CBN(1) [IS.M3+1]

        .=: IS.M0+10


7212 M      151205    002020    000002    000200    000041    057230
**: IS.M2
        CM: MODE 2 / (R) IS ADDRESS; (R) + (2 OR 4) / IF DST=27, (R) + 2
        AR: RB(DST)=RA(DST)+[+2]+<0>,D=RA(DST)
```

```
INTEGER SOURCE

        BR: CBN(/V) [IS.M5+2]

7213 M      150115    100070    000027    000200    000060    100000
**:     AR: R10A-[+27]-<1>

7214 M      150305    002020    000002    000200    000052    057230
**:     CM: BRANCH IF NOT IMMEDIATE
        AR: RB(DST)=RA(DST)+[+2]+<0>
        BR: CBN(/Z') [IS.M5+2]

7215 M      150315    102020    000002    000200    000060    057225
**:     CM: ELSE RESTORE GENERAL REGISTER
        AR: RB(DST)=RA(DST)-[+2]-<1>
        BR: CBN(1) [IS.M4+3]

        .=: IS.M0+14

7216 M      152205    002020    000002    000200    140060    057230
**: IS.M3
        CM: MODE 3 / (R) IS ADDRESS OF ADDRESS; (R) + 2
        AR: RB(DST)=RA(DST)+[+2]+<0>,AD=RA(DST)
        IO: DATI(CMI)           BR: CBN(1) [IS.M5+2]

7217 M      150345    007070    000034    000600    000060    077224
**:     CM: (ISRC) MASK IR
        AR: R10B=R10A AND [+34] PC: UW
        BR: CJN(1) [IS.M4+2]

7220 M      070337    007000    037042    000200    000002    164000
**:     CM: BRANCH IF USING SET MODE
        AR: R10B=IR5
        CC: C=C,V=V,N=V,Z=0     BR: CBIN(Z) <SCR>

7221 M      000101    007676    037210    000400    000060    144000
**:     CM: ELSE USE FPS MODE
```

```
        CC: C=C,V=PN,N=PN,Z=0   BR: CBIC(1) <SCR>

        .=: IS.M0+20

7222 M      151315    102020    000002    000200    000041    057227
**: IS.M4
        CM: MODE 4 / (R) - (2 OR 4); (R) IS ADDRESS
        AR: D;RB(DST)=RA(DST)-[+2]-<1>
        BR: CBN(/V) [IS.M5+1]

7223 M      151315    102020    000002    000200    000060    057227
**:     CM: ELSE DOUBLE INTEGER
        AR: D;RB(DST)=RA(DST)-[+2]-<1>
        BR: CBN(1) [IS.M5+1]

7224 M      150105    000070    007202    000600    040460    120120
**:     CM: (ISRC) BUILD JUMP ADDRESS
        AR: SCE5=R10A+[IS.M0]+<0>       PC: UW
        SP: CLR,SE5             BR: CRR(1)

7225 M      000100    000000    030456    000200    000060    057230
```

```
INTEGER SOURCE


**:     CM: (MODE 2) SET IMMEDIATE FLAG
        CC: C=C,V=V,N=0,Z=Z     BR: CBN(1) [IS.M5+2]

        .=: IS.M0+24

7226 M      152315    002020    000002    000200    140060    100000
**: IS.M5
        CM: MODE 5 / (R) - 2 ; (R) IS ADDRESS OF ADDRESS
        AR: AD;RB(DST)=RA(DST)-[+2]-<0>
        IO: DATI(CMI)

7227 M      137115    100066    030440    000200    000060    100000
**:     CM: CHECK STACK OVERFLOW
        AR: SOR=R6A-SLR-<1>

7230 M      002337    007000    030440    010200    000000    130000
**:     CM: RETURN IF ONLY ADDRESS NEEDED
        AR: AD;R10B=D           PC: LW,WIOH
        BR: CRN(C)

7231 M      000342    007500    030440    001600    141060    057237
**:     CM: ELSE GET INTEGER
        AR: R15B=0              PC: 4B
        IO: IDATI(CMI)          BR: CBN(1) [B.GETINT]

        .=: IS.M0+30

7232 M      152205    006767    000002    000200    140060    100000
**: IS.M6
        CM: MODE 6 / (R) + X IS ADDRESS
        AR: R7B=R7A+[+2]+<0>,AD=R7A
        IO: DATI(CMI)

7233 M      002305    007020    030440    010200    000000    130000
**:     CM: RETURN IF ONLY ADDRESS WANTED
```

```
        BR: CRN(C)

7234 M      000342    007500    030440    001600    141060    057237
**:     CM: ELSE GET INTEGER
        AR: R15B=0              PC: 4B
        IO: IDATI(CMI)          BR: CBN(1) [B.GETINT]

7235 M      002105    000020    030440    010200    140060    057230
**:     CM: (MODE 7) GET X
        AR: AD=RA(DST)+D+<0>    PC: LW,WIOH
        IO: DATI(CMI)           BR: CBN(1) [IS.M5+2]

        .=: IS.M0+34

7236 M      152205    006767    000002    000200    140060    057235
**: IS.M7
        CM: MODE 7 / (R) + X IS ADDRESS OF ADDRESS
        AR: R7B=R7A+[+2]+<0>,AD=R7A
        IO: DATI(CMI)           BR: CBN(1) [IS.M6+3]

7237 M      000342    007400    030440    001600    000060    100000
```

```
INTEGER SOURCE


**: B.GETINT
        CM: CLEAR HIGH FRACTION
        AR: R14B=0              PC: 4B

7240 M      152105    000070    000002    020200    000001    057242
**:     CM: SET UP ADDRESS
        AR: AD=R10A+[+2]+<0>    PC: LW,WIOL
        BR: CBN(V) [.+2]

7241 M      000337    007500    104450    000200    000060    057243
**:     CM: LOAD INTEGER
        AR: R15B=D              PC: LW
        CC: Z=PZ,N=PN,C=PN,V=V  BR: CBN(1) [.+2]

7242 M      000337    007500    104450    000600    000003    077204
**:     CM: LOAD LONG INTEGER
        AR: R15B=D              PC: UW
        CC: Z=PZ,N=PN,C=PN,V=V  BR: CJN(N) [IS.M0+2]

7243 M      000542    006500    160440    000604    000040    130000
**:     CM: SAVE SIGN
        AR: R5B=0,SRD(C)        PC: UW
        CC: C=0,N=N,Z=Z,V=V     BR: CRN(/C)

7244 M      000323    107500    030440    000200    000041    130000
**:     CM: ELSE GET MAGNITUDE
        AR: R15B=-R15B-<1>      PC: LW
        BR: CRN(/V)

7245 M      000323    057500    030440    000600    000060    130000
**:     AR: R15B=-R15B-<C'>     PC: UW
        BR: CRN(1)
```

```
        PG:     SAVE RESULT IN ACCUMULATOR

CM:     IF C=1 A CLEAN ZERO IS GENERATED, ELSE
        R4 (UW) - EXPONENT
        R5 (UW) - SIGN OF FRACTION
        R14 (4B) - HIGH FRACTION
        R15 (4B) - LOW FRACTION
        ARE COMBINED INTO THE STANDARD FLOATING POINT
        FORMAT = FLOATING (V=0)
         SIGN(1-BIT)/EXPONENT(8-BITS)/FRACTION(23-BITS)
        FORMAT = DOUBLE (V=1)
         SIGN(1-BIT)/EXPONENT(8-BITS)/FRACTION(55-BITS)
        AND SAVE IN THE ACCUMULATOR AT THE ADDRESS IN R10 (LW)

7246 M      150355    007474    177600    000600    000060    076531
**: ACC.SAV
        CM: COMBINE SIGN, EXPONENT, AND FRACTION
        AR: R14B=[+177600] MASK R14A    PC: UW
        BR: CJN(1) [CMB.SEF+1]

7247 M      002134    000070    030440    000200    170260    100000
**: ACC.SAVA
        CM: SAVE RESULT
        AR: AD=R10A             IO: DATO(OFF)

7250 M      001133    007400    160440    020200    000060    100000
**:     AR: D=R14B              PC: LW,WIOL
        CC: C=0,N=N,Z=Z,V=V

7251 M      152105    000070    000002    000200    170241    130000
**:     AR: AD=R10A+[+2]+<0>    IO: DATO(OFF)
        BR: CRN(/V)

7252 M      001133    007500    160740    020600    000060    100000
**:     CM: DOUBLE MODE
        AR: D=R15B              PC: UW,WIOL
```

```
7253 M      152105    000070    000004    000200    170260    100000
**:     AR: AD=R10A+[+4]+<0>    IO: DATO(OFF)

7254 M      001133    007500    030440    020200    000060    100000
**:     AR: D=R15B              PC: LW,WIOL

7255 M      152105    000070    000006    000200    170260    120000
**:     AR: AD=R10A+[+6]+<0>    IO: DATO(OFF)
        BR: CRR(1)
```

```
        PG:     CFCC, SETF, SETI, SETD, AND SETL

7256 M      070337    007000    030440    000600    000060    100000
**: SET.FPS
        CM: DETERMINE WHICH INSTRUCTION
        AR: R10B=IR5            PC: UW

7257 M      150115    100070    000001    000600    171712    057271
**:     CM: BRANCH IF CFCC
        AR: R10A-[+1]-<1>       PC: UW
        IO: INTCK               BR: CBN(Z') [CFCC]

7260 M      150115    100070    000002    000600    171712    057272
**:     CM: BRANCH IF SETF
        AR: R10A-[+2]-<1>       PC: UW
        IO: INTCK               BR: CBN(Z') [SETF]

7261 M      150115    100070    000011    000600    171712    057273
**:     CM: BRANCH IF SETI
        AR: R10A-[+11]-<1>      PC: UW
        IO: INTCK               BR: CBN(Z') [SETI]

7262 M      150115    100070    000012    000600    171712    057274
**:     CM: BRANCH IF SETD
        AR: R10A-[+12]-<1>      PC: UW
        IO: INTCK               BR: CBN(Z') [SETD]

7263 M      120337    006000    030440    000600    171712    057275
**:     CM: BRANCH IF SETL
        AR: R0B=PSR             PC: UW
        IO: INTCK               BR: CBN(Z') [SETL]

7264 M      152137    000000    177764    000200    170060    076331
**:     CM: GO CHECK RESERVED INSTRUCTIONS FOR DEFINITION
        AR: AD=[+177764]        IO: DATI(OFF)
        BR: CJN(1) [RSRVSTAT]
```

```
7265 M      150115    100067    000002    000200    000060    100110
**:     CM: ELSE INVALID INSTRUCTION
        AR: SCR1=R7A-[+2]-<1>

7266 M      150137    000000    006073    000200    040460    100120
**:     CM: SET UP END BRANCH
        AR: SCE5=[BGN+3]               SP: CLR,SE5

7267 M      151137    000000    000002    020200    000060    077737
**: FOP.TRP
        CM: UNDEFINED FLOATING INSTRUCTION TRAP
        AR: D=[+2]              PC: LW,WIOL
        BR: CJN(1) [FPP.TRP]

7270 M      152105    000067    000000    020200    171760    164000
**:     CM: GO TO BGN OR TRAP OUT
        AR: AD=R7A+[+0]+<0>     PC: LW,WIOL
        IO: INTCK               BR: CBIN(1) <SCR>

7271 M      000133    006600    063146    000600    140460    046071
**: CFCC
```

CFCC, SETF, SETI, SETD, AND SETL

```
        CM: TRANSFER CONDITION CODES
        AR: OUT=R6B             PC: UW
        CC: C=CPO,N=CPO,Z=CPO,V=CPO
        IO: DATIR(CMI)          BR: CBR(1) [BGN+1]

7272 M      150355    007676    000200    000600    141460    046071
**: SETF
        CM: SET FLOATING MODE
        AR: R16B=[+200] MASK R16A       PC: UW
        IO: IDATIR(CMI)         BR: CBR(1) [BGN+1]

7273 M      150355    007676    000100    000600    141460    056071
**: SETI
        CM: SET INTEGER MODE
        AR: R16B=[+100] MASK R16A       PC: UW
        IO: IDATIR(CMI)         BR: CBN(1) [BGN+1]

7274 M      150335    007676    000200    000600    141460    056071
**: SETD
        CM: SET DOUBLE MODE
        AR: R16B=R16A OR [+200]         PC: UW
        IO: IDATIR(CMI)         BR: CBN(1) [BGN+1]

7275 M      150335    007676    000100    000600    141460    056071
**: SETL
        CM: SET LONG INTEGER MODE
        AR: R16B=R16A OR [+100]         PC: UW
        IO: IDATIR(CMI)         BR: CBN(1) [BGN+1]
```

```
        PG:     LDFPS AND STFPS

7276 M      002134    000067    030440    020200    171760    100000
**: LDFPS
        CM: GET NEXT INSTRUCTION
        AR: AD=R7A              PC: LW,WIOL
        IO: INTCK

7277 M      000337    007600    030440    020600    141460    100000
**:     CM: SAVE FPS
        AR: R16B=D              PC: UW,WIOL
        IO: IDATIR(CMI)

7300 M      000337    006600    030440    000600    000060    056071
**:     AR: R6B=D               PC: UW
        BR: CBN(1) [BGN+1]

7301 M      150337    006100    000017    000600    000060    100000
**: STFPS
        CM: MASK FCC'S
        AR: R1B=[+17]           PC: UW

7302 M      000341    006661    030440    000600    000060    100000
**:     AR: R6B=R1A AND R6B     PC: UW

7303 M      150355    006176    030037    000600    000060    100000
**:     CM: MASK FPS (LEAVE UNCHANGED)
        AR: R1B=[+30037] MASK R16A      PC: UW

7304 M      150337    007000    000070    000200    000060    100000
**:     CM: CHECK FOR MODE 0
        AR: R10B=[+70]

7305 M      070145    000070    030440    000200    000060    100000
**:     AR: R10A AND IR5
```

```
        PG:     STST

7311 M      120337    006000    177776    000600    000060    077211
**: STST
        CM: SAVE CC'S / GET ADDRESS
        AR: R0B=PSR             PC: UW
        CC: C=1,N=0,Z=1,V=1     BR: CJN(1) [ISRC]

7312 M      152137    000000    167102    000200    170042    057315
**:     CM: GET FEC
        AR: AD=[FEC.FPP]        IO: DATI(OFF)
        BR: CBN(/Z) [.+3]

7313 M      000337    002000    030440    010200    000060    100000
**:     CM: MODE 0
        AR: RB(DST)=D           PC: LW,WIOH

7314 M      000133    006000    063146    000600    000060    056070
**:     CM: RESTORE CC'S
        AR: OUT=R0B             PC: UW
        CC: C=CPO,N=CPO,Z=CPO,V=CPO
        BR: CBN(1) [BGN]

7315 M      002134    000070    030440    020200    100243    057314
**:     CM: SAVE FEC / FINISHED IF IMMEDIATE
        AR: AD=R10A             PC: LW,WIOL
        IO: DATO(CM)            BR: CBN(/N) [.-1]

7316 M      152137    000000    167100    020200    170060    100000
**:     CM: ELSE GET FEA
        AR: AD=[FEA.FPP]        PC: LW,WIOL
        IO: DATI(OFF)

7317 M      152105    000070    000002    020200    100260    057314
**:     CM: SAVE FEA
        AR: AD=R10A+[+2]+<0>    PC: LW,WIOL
```

```
        PG:     CLRF AND TSTF

7320 M      120337    006000    177356    000600    000060    077136
**: CLRF
        CM: SAVE CC'S / GET ADDRESS WITH CURRENT MODE
        AR: R0B=PSR             PC: UW
        CC: C=1,Z=0,N=0,V=0     BR: CJN(1) [FSRC]

7321 M      150137    000000    006070    000200    040460    100120
**:     CM: SET END BRANCH
        AR: SCE5=[BGN]          SP: CLR,SE5

7322 M      000100    000000    177056    000200    000060    057337
**:     CM: SAVE A CLEAN ZERO
        CC: C=1,Z=0,N=0,V=V     BR: CBN(1) [NEGF+4]

7323 M      120337    006000    167356    000600    000060    077135
**: TSTF
        CM: SAVE CC'S / GET SOURCE DATA
        AR: R0B=PSR             PC: UW
        CC: C=0,N=0,Z=0,V=0     BR: CJN(1) [FSRC.SAV]

7324 M      150137    000000    006070    000200    040460    100120
**:     CM: SET UP END BRANCH
        AR: SCE5=[BGN]          SP: CLR,SE5

7325 M      000100    000000    030440    000200    000000    077716
**:     CM: CHECK FOR -0 IF NOT MODE 0
        BR: CJN(C) [FIUV.FPP+1]

7326 M      000100    000000    160740    000200    000060    057341
**:     CM: MASK FCC'S / GO RESTORE CC'S
        CC: C=0,V=0,N=N,Z=Z     BR: CBN(1) [NEGF+6]
```

```
        PG:     ABSF AND NEGF

7327 M      120337    006000    167356    000600    000060    077135
**: ABSF
        CM: SAVE CC'S / GET SOURCE DATA
        AR: R0B=PSR             PC: UW
        CC: C=0,N=0,Z=0,V=0     BR: CJN(1) [FSRC.SAV]

7330 M      150137    000000    006070    000200    040460    100120
**:     CM: SET UP END BRANCH
        AR: SCE5=[BGN]          SP: CLR,SE5

7331 M      000100    000000    030440    000200    000000    077715
**:     CM: CHECK FOR -0 IF NOT MODE 0
        BR: CJN(C) [FIUV.FPP]

7332 M      000342    006500    030440    000600    000060    057337
**:     CM: MAKE ABSOLUTE / GO SAVE RESULT
        AR: R5B=0               PC: UW
        BR: CBN(1) [NEGF+4]

7333 M      120337    006000    167356    000600    000060    077135
**: NEGF
        CM: SAVE CC'S / GET SOURCE DATA
        AR: R0B=PSR             PC: UW
        CC: C=0,N=0,Z=0,V=0     BR: CJN(1) [FSRC.SAV]

7334 M      150137    000000    006070    000200    040460    100120
**:     CM: SET UP END BRANCH
        AR: SCE5=[BGN]          SP: CLR,SE5

7335 M      000100    000000    030440    000200    000000    077716
**:     CM: CHECK FOR -0 IF NOT MODE 0
        BR: CJN(C) [FIUV.FPP+1]

7336 M      000373    006500    030450    000600    000002    077716
```

```
        PG:     MULF/MULD INSTRUCTIONS

7342 M      150337    007100    167000    000200    000060    057346
**: MULF
        CM: POINT TO ACCUMULATOR
        AR: R11B=[AC0.FPP]      BR: CBN(1) [.+4]

7343 M      150337    007100    167010    000200    000060    057346
**:     AR: R11B=[AC1.FPP]      BR: CBN(1) [.+3]

7344 M      150337    007100    167020    000200    000060    057346
**:     AR: R11B=[AC2.FPP]      BR: CBN(1) [.+2]

7345 M      150337    007100    167030    000200    000060    057346
**:     AR: R11B=[AC3.FPP]      BR: CBN(1) [.+1]

7346 M      120337    006000    167356    000600    000060    077126
**:     CM: SAVE CC'S / GET OPERANDS
        AR: R0B=PSR             PC: UW
        CC: C=0,N=0,Z=0,V=0     BR: CJN(1) [GETOPS]

7347 M      000333    006400    174440    000600    000002    057322
**:     CM: ON ZERO OPERANDS - CLEAN ZERO
        AR: R4B=R4B             PC: UW
        CC: C=1,Z=PZ,N=N,V=V    BR: CBN(Z) [CLRF+2]

7350 M      000100    000000    030440    000200    000002    057322
**:     BR: CBN(Z) [CLRF+2]

7351 M      155137    000000    000030    000200    000041    076621
**:     CM: FLOATING MODE
        AR: CNTR=[+30]          BR: CJN(/V) [MUL.24+1]

7352 M      155137    000000    000040    000200    000001    076735
**:     CM: DOUBLE MODE
        AR: CNTR=[+40]          BR: CJN(V) [MUL.56+1]
```

```
        PG:     MODF/MODD INSTRUCTIONS

7354 M      150337    007100    167000    000200    000060    057360
**: MODF
        CM: POINT TO ACCUMULATOR
        AR: R11B=[AC0.FPP]      BR: CBN(1) [.+4]

7355 M      150337    007100    167010    000200    000060    057360
**:     AR: R11B=[AC1.FPP]      BR: CBN(1) [.+3]

7356 M      150337    007100    167020    000200    000060    057360
**:     AR: R11B=[AC2.FPP]      BR: CBN(1) [.+2]

7357 M      150337    007100    167030    000200    000060    057360
**:     AR: R11B=[AC3.FPP]      BR: CBN(1) [.+1]

7360 M      120337    006000    167356    000600    000060    077126
**:     CM: SAVE CC'S / GET OPERANDS
        AR: R0B=PSR             PC: UW
        CC: C=0,N=0,Z=0,V=0     BR: CJN(1) [GETOPS]

7361 M      000333    006400    110456    000600    000060    100000
**:     CM: ON ZERO OPERANDS - FINISHED
        AR: R4B=R4B             PC: UW
        CC: C=PZ,Z=Z,N=0,V=V

7362 M      000342    006600    030440    000600    000004    057417
**:     CM: CLEAR FCC'S
        AR: R6B=0               PC: UW
        BR: CBN(COZ) [MOD.A]

7363 M      155137    000000    000040    000200    000001    057366
**:     CM: BRANCH ON DOUBLE
        AR: CNTR=[+40]          BR: CBN(V) [.+3]

7364 M      000342    007300    030440    001600    000060    100000
```

MODF/MODD INSTRUCTIONS

```
7372 M      000334    006365    030440    000600    000060    100000
**:     AR: R3B=R5A             PC: UW

7373 M      000334    006264    160440    000600    000060    077722
**:     CM: AND CHECK EXPONENT
        AR: R2B=R4A             PC: UW
        CC: C=0,N=N,Z=Z,V=V     BR: CJN(1) [EXP.UN+1]

7374 M      000133    006600    030440    000600    000000    057417
**:     CM: CHECK FOR OVERFLOW / DONE IF C=1
        AR: R6B                 PC: UW
        BR: CBN(C) [MOD.A]

7375 M      155115    100064    000200    000600    000052    057421
**:     CM: CHECK EXPONENT / DONE ON OVERFLOW
        AR: CNTR=R4A-[+200]-<1> PC: UW
        BR: CBN(/Z') [MOD.B]

7376 M      150115    100064    000270    000600    000017    057423
**:     CM: IF EXP <= 200 - ONLY FRACTION
        AR: R4A-[+270]-<1>      PC: UW
        BR: CBN(BLE') [MOD.C]

7377 M      150115    100064    000337    000600    000016    057407
**:     CM: IF EXP < 270 - DO SHIFTING
        AR: R4A-[+337]-<1>      PC: UW
        BR: CBN(BLT') [MODF.A]

7400 M      155115    100064    000240    000600    000057    057421
**:     CM: IF EXP > 337 - FRACTION = 0
        AR: CNTR=R4A-[+240]-<1> PC: UW
        BR: CBN(BGT') [MOD.B]

7401 M      000633    007200    030440    041603    000077    100000
**:     CM: ELSE DO SHIFTING
```

MODF/MODD INSTRUCTIONS

```
7407 M      000342    007400    030440    001600    000060    100000
**: MODF.A
        CM: CLEAR FLOATING
        AR: R14B=0              PC: 4B

7410 M      000342    007500    030440    001600    000060    100000
**:     AR: R15B=0              PC: 4B

7411 M      000633    007300    100440    001603    000060    100000
**:     CM: SHIFT PROD TO GET INTEGER IN R14/R15
        AR: R13B=R13B,SRU(MSBQ),SQ(0)   PC: 4B
        CC: C=MSBR,N=N,Z=Z,V=V

7412 M      000733    007200    100440    001404    000060    100000
**:     AR: R12B=R12B,SRU(C)    PC: 3B
        CC: C=MSBR,N=N,Z=Z,V=V

7413 M      000733    007500    100440    001604    000060    100000
**:     AR: R15B=R15B,SRU(C)    PC: 4B
        CC: C=MSBR,N=N,Z=Z,V=V

7414 M      000733    007400    160440    041404    000037    057411
**:     AR: R14B=R14B,SRU(C)    PC: 3B,CCL
        CC: C=0,N=N,Z=Z,V=V     BR: CBN(CNTR) [.-3]

7415 M      150337    006400    000271    000600    000060    077425
**:     CM: NORMALIZE AND SAVE INTEGER PART
        AR: R4B=[+271]          PC: UW
        BR: CJN(1) [INT.SAV]

7416 M      150337    006200    000200    000600    000060    057432
**:     CM: NORMALIZE AND SAVE FRACTION
        AR: R2B=[+200]          PC: UW
        BR: CBN(1) [FRC.SAV]
```

MODF/MODD INSTRUCTIONS

```
7423 M      000133    007600    170600    000400    000060    077430
**: MOD.C
        CM: SAVE A CLEAN ZERO INTEGER
        AR: R16B                PC: UB
        CC: C=1,V=PN,N=N,Z=Z    BR: CJN(1) [INT.SAV+3]

7424 M      000133    007600    160600    000400    000060    057433
**:     CM: NORMALIZE AND SAVE FRACTION
        AR: R16B                PC: UB
        CC: C=0,V=PN,N=N,Z=Z    BR: CBN(1) [FRC.SAV+1]

7425 M      000332    007100    167056    001600    000060    077051
**: INT.SAV
        CM: SAVE Q REGISTER / NORMALIZE INTEGER
        AR: R11B=Q              PC: 4B
        CC: C=0,N=0,Z=0,V=V     BR: CJN(1) [NORM.56]

7426 M      000034    000071    030440    001600    000060    100000
**:     CM: RESTORE Q REGISTER
        AR: Q=R11A              PC: 4B

7427 M      000133    007600    030600    000400    000060    100000
**:     CM: SET UP LENGTH FLAG
        AR: R16B                PC: UB
        CC: V=PN,C=C,N=N,Z=Z

7430 M      000334    007170    030440    000200    000060    100000
**:     CM: RESET ADDRESS
        AR: R11B=R10A           PC: LW

7431 M      150335    007071    000010    000200    000060    057246
**:     CM: SAVE INTEGER IN (ACC OR 1)
        AR: R10B=R11A OR [+10]  PC: LW
        BR: CBN(1) [ACC.SAV]
```

MODF/MODD INSTRUCTIONS

```
**:     CM: SET UP TRUNCATION FLAG / SKIP ON CLEAN ZERO
        AR: R16B AND R1A        PC: UW
        CC: N=PZ,C=C,Z=Z,V=V    BR: CBN(C) [.+4]

7437 M      000133    007500    100440    001600    000003    076705
**:     CM: CONDITIONAL ROUND
        AR: R15B                PC: 4B
        CC: C=PN,N=N,Z=Z,V=V    BR: CJN(N) [NORM.24A]

7440 M      000100    000000    167056    000200    000060    057442
**:     CC: C=0,N=0,Z=0,V=V     BR: CBN(1) [.+2]

7441 M      000303    106400    030440    000600    000060    077052
**:     CM: NORMALIZE INTEGER
        AR: R4B=R4B+<1>         PC: UW
        BR: CJN(1) [NORM.56+1]

7442 M      000334    007071    030440    000200    000060    077246
**:     CM: SAVE FRACTION
        AR: R10B=R11A           BR: CJN(1) [ACC.SAV]

7443 M      120237    006060    063146    000600    000060    100000
**:     CM: RESTORE CC'S / BUILD FCC'S
        AR: R0B=PSR,OUT=R0A     PC: UW
        CC: C=CPO,N=CPO,Z=CPO,V=CPO

7444 M      000331    006660    030440    000600    000060    164000
**:     CM: SET FCC'S / FINISHED / TAP
        AR: R6B=R6B OR R0A      PC: UW
        BR: CBIN(1) <SCR>
```

```
        PG:     ADDF/ADDD INSTRUCTIONS

7445 M      150337    007100    167000    000200    000060    057451
**: ADDF
        CM: POINT TO ACCUMULATOR
        AR: R11B=[AC0.FPP]      BR: CBN(1) [.+4]

7446 M      150337    007100    167010    000200    000060    057451
**:     AR: R11B=[AC1.FPP]      BR: CBN(1) [.+3]

7447 M      150337    007100    167020    000200    000060    057451
**:     AR: R11B=[AC2.FPP]      BR: CBN(1) [.+2]

7450 M      150337    007100    167030    000200    000060    057451
**:     AR: R11B=[AC3.FPP]      BR: CBN(1) [.+1]

7451 M      120337    006000    167356    000600    000060    077126
**:     CM: SAVE CC'S / GET OPERANDS
        AR: R0B=PSR             PC: UW
        CC: C=0,N=0,Z=0,V=0     BR: CJN(1) [GETOPS]

7452 M      000333    006200    164440    000600    000041    076600
**:     CM: FLOATING MODE
        AR: R2B=R2B             PC: UW
        CC: C=0,Z=PZ,N=N,V=V
        BR: CJN(/V) [ADD.24+1]

7453 M      000100    000000    030440    000200    000001    076711
**:     CM: DOUBLE MODE / FALL THROUGH TO ANSWER
        BR: CJN(V) [ADD.56+1]
```

```
        PG:     ANSWER ROUTINE

CM:     THIS ROUTINE CHECKS THE ARITHMETIC ANSWER
        FOR EXPONENT UNDERFLOW/OVERFLOW AND
        SAVES THE RESULT IN THE ACCUMULATOR AT
        ADDRESS R10 (LW)

7454 M      000342    006600    030440    000600    000040    077722
**: ANSWER
        CM: CLEAR FCC'S / CHECK EXPONENT IF NOT CLEAN ZERO
        AR: R6B=0               PC: UW
        BR: CJN(/C) [EXP.UN+1]

7455 M      150355    007474    177600    000600    000060    076531
**:     AR: R14B=[+177600] MASK R14A    PC: UW
        BR: CJN(1) [CMB.SEF+1]

7456 M      002134    000070    030440    000200    170260    077250
**:     AR: AD=R10A             IO: DATO(OFF)
        BR: CJN(1) [ACC.SAV+2]

7457 M      120237    006060    063146    000600    000060    100000
**:     CM: SAVE CURRENT STATUS / RESTORE CC'S
        AR: R0B=PSR,OUT=R0A     PC: UW
        CC: C=CPO,N=CPO,Z=CPO,V=CPO

7460 M      000331    006660    030440    000600    000060    164000
**:     CM: SAVE FCC'S / TRAP OUT / FINISHED
        AR: R6B=R6B OR R0A      PC: UW
        BR: CBIN(1) <SCR>
```

```
        PG:     LDF/LDD INSTRUCTIONS

7461 M      150337    007100    167000    000200    000060    057465
**: LDF
        CM: POINT TO ACCUMULATOR
        AR: R11B=[AC0.FPP]      BR: CBN(1) [.+4]

7462 M      150337    007100    167010    000200    000060    057465
**:     AR: R11B=[AC1.FPP]      BR: CBN(1) [.+3]

7463 M      150337    007100    167020    000200    000060    057465
**:     AR: R11B=[AC2.FPP]      BR: CBN(1) [.+2]

7464 M      150337    007100    167030    000200    000060    057465
**:     AR: R11B=[AC3.FPP]      BR: CBN(1) [.+1]

7465 M      120337    006000    167356    000600    000060    077135
**:     CM: SAVE CC'S / GET SOURCE DATA
        AR: R0B=PSR             PC: UW
        CC: C=0,Z=0,N=0,V=0     BR: CJN(1) [FSRC.SAV]

7466 M      150137    000000    007471    000200    040460    100120
**:     CM: SET UP END BRANCH
        AR: SCE5=[.+3]          SP: CLR,SE5

7467 M      120337    006100    030440    000600    000000    077715
**:     CM: CHECK FOR -0 IF NOT MODE 0
        AR: R1B=PSR             PC: UW
        BR: CJN(C) [FIUV.FPP]

7470 M      150345    006661    000014    000600    000060    164000
**:     CM: MASK STATUS / TRAP ON -0 AND FIUV=1
        AR: R6B=R1A AND [+14]   PC: UW
        BR: CBIN(1) <SCR>

7471 M      000334    007071    160440    000200    000060    077246
```

```
        PG:     SUBF/SUBD INSTRUCTIONS

7473 M      150337    007100    167000    000200    000060    057477
**: SUBF
        CM: POINT TO ACCUMULATOR
        AR: R11B=[AC0.FPP]      BR: CBN(1) [.+4]

7474 M      150337    007100    167010    000200    000060    057477
**:     AR: R11B=[AC1.FPP]      BR: CBN(1) [.+3]

7475 M      150337    007100    167020    000200    000060    057477
**:     AR: R11B=[AC2.FPP]      BR: CBN(1) [.+2]

7476 M      150337    007100    167030    000200    000060    057477
**:     AR: R11B=[AC3.FPP]      BR: CBN(1) [.+1]

7477 M      120337    006000    167356    000600    000060    077126
**:     CM: SAVE CC'S / GET OPERANDS
        AR: R0B=PSR             PC: UW
        CC: C=0,N=0,Z=0,V=0     BR: CJN(1) [GETOPS]

7500 M      150365    006565    100000    000600    000060    057452
**:     CM: CHANGE SIGN OF B ARGUMENT / GO TO ADDF
        AR: R5B=R5A XOR [+100000]       PC: UW
        BR: CBN(1) [ADDF+5]
```

```
        PG:     CMPF/CMPD INSTRUCTIONS

7501 M      150337    007100    167000    000200    000060    057505
**: CMPF
        CM: POINT TO ACCUMULATOR
        AR: R11B=[AC0.FPP]      BR: CBN(1) [.+4]

7502 M      150337    007100    167010    000200    000060    057505
**:     AR: R11B=[AC1.FPP]      BR: CBN(1) [.+3]

7503 M      150337    007100    167020    000200    000060    057505
**:     AR: R11B=[AC2.FPP]      BR: CBN(1) [.+2]

7504 M      150337    007100    167030    000200    000060    057505
**:     AR: R11B=[AC3.FPP]      BR: CBN(1) [.+1]

7505 M      120337    006000    167356    000600    000060    077126
**:     CM: SAVE CC'S / GET OPERANDS
        AR: R0B=PSR             PC: UW
        CC: C=0,N=0,Z=0,V=0     BR: CJN(1) [GETOPS]

7506 M      000111    106462    164450    000600    040060    100000
**:     CM: COMPARE EXPONENTS
        AR: R4B-R2A-<1>         PC: UW
        CC: C=0,N=PN,Z=PZ,V=V   SP: CLR

7507 M      150365    006363    100000    000600    000042    057514
**:     CM: NEGATE AC / BRANCH IF EXPONENTS NOT EQUAL
        AR: R3B=R3A XOR [+100000]       PC: UW
        BR: CBN(/Z) [.+5]

7510 M      000161    006563    030440    000600    000041    076610
**:     CM: ELSE CHECK ADDITION OF FRACTIONS
        AR: R5B XOR R3A         PC: UW
        BR: CJN(/V) [ADD.24A]
```

CMPF/CMPD INSTRUCTIONS

```
7516 M      120237    006660    063146    000600    000060    056070
**:     CM: SAVE FCC'S / RESTORE CC'S
        AR: R6B=PSR,OUT=R0A     PC: UW
        CC: C=CPO,N=CPO,Z=CPO,V=CPO
        BR: CBN(1) [BGN]
```

```
        PG:     STF/STD INSTRUCTIONS

7517 M      152337    007000    167000    000200    170060    057523
**: STF
        CM: POINT TO ACCUMULATOR
        AR: AD;R10B=[AC0.FPP]   IO: DATI(OFF)
        BR: CBN(1) [.+4]

7520 M      152337    007000    167010    000200    170060    057523
**:     AR: AD;R10B=[AC1.FPP]   IO: DATI(OFF)
        BR: CBN(1) [.+3]

7521 M      152337    007000    167020    000200    170060    057523
**:     AR: AD;R10B=[AC2.FPP]   IO: DATI(OFF)
        BR: CBN(1) [.+2]

7522 M      152337    007000    167030    000200    170060    057523
**:     AR: AD;R10B=[AC3.FPP]   IO: DATI(OFF)
        BR: CBN(1) [.+1]

7523 M      120337    006000    030440    000600    000060    100000
**:     CM: SAVE CC'S
        AR: R0B=PSR             PC: UW

7524 M      000133    007600    167216    000400    000060    077117
**:     AR: R16B                PC: UB
        CC: C=0,N=0,Z=0,V=PN    BR: CJN(1) [B.GETACC+1]

7525 M      070537    007000    177456    000600    000060    077137
**:     CM: GET DESTINATION ADDRESS
        AR: R10B=IR5,SRD(0)     PC: UW
        CC: C=1,N=0,Z=1,V=V     BR: CJN(1) [FSRC+1]

7526 M      000100    000000    167056    000200    000060    077150
**:     CM: STORE AT DESTINATION
        CC: C=0,N=0,Z=0,V=V
```

```
        PG:     DIVF/DIVD INSTRUCTIONS

7530 M      150337    007100    167000    000200    000060    057534
**: DIVF
        CM: POINT TO ACCUMULATOR
        AR: R11B=[AC0.FPP]      BR: CBN(1) [.+4]

7531 M      150337    007100    167010    000200    000060    057534
**:     AR: R11B=[AC1.FPP]      BR: CBN(1) [.+3]

7532 M      150337    007100    167020    000200    000060    057534
**:     AR: R11B=[AC2.FPP]      BR: CBN(1) [.+2]

7533 M      150337    007100    167030    000200    000060    057534
**:     AR: R11B=[AC3.FPP]      BR: CBN(1) [.+1]

7534 M      120337    006000    167356    000600    000060    077126
**:     CM: SAVE CC'S / GET OPERANDS
        AR: R0B=PSR             PC: UW
        CC: C=0,N=0,Z=0,V=0     BR: CJN(1) [GETOPS]

7535 M      150137    000000    007541    000200    040460    100120
**:     CM: SET UP END BRANCH
        AR: SCE5=[.+4]          SP: CLR,SE5

7536 M      000333    006400    030440    000600    000060    100000
**:     CM: CHECK FOR ZERO DIVISOR
        AR: R4B=R4B             PC: UW

7537 M      151137    000000    000004    020200    000012    077737
**:     AR: D=[+4]              PC: LW,WIOL
        BR: CJN(Z') [FPP.TRP]

7540 M      150137    000000    006070    000200    000060    164120
**:     CM: TRAP IF DIVISOR = 0
        AR: SCE5=[BGN]          BR: CBIN(1) <SCR>
```

```
        PG:     STEXP INSTRUCTION

7545 M      152137    000000    167000    000200    170060    057551
**: STEXP
        CM: POINT TO ACCUMULATOR
        AR: AD=[AC0.FPP]        IO: DATI(OFF)
        BR: CBN(1) [.+4]

7546 M      152137    000000    167010    000200    170060    057551
**:     AR: AD=[AC1.FPP]        IO: DATI(OFF)
        BR: CBN(1) [.+3]

7547 M      152137    000000    167020    000200    170060    057551
**:     AR: AD=[AC2.FPP]        IO: DATI(OFF)
        BR: CBN(1) [.+2]

7550 M      152137    000000    167030    000200    170060    057551
**:     AR: AD=[AC3.FPP]        IO: DATI(OFF)
        BR: CBN(1) [.+1]

7551 M      150337    007000    077600    000200    000060    100000
**:     CM: SET UP MASK FOR EXPONENT
        AR: R10B=[+77600]

7552 M      152237    007271    000200    020200    000060    100000
**:     CM: LOAD ADDRESS OF DESTINATION
        AR: R12B=[+200],AD=R11A PC: LW,WIOL

7553 M      000745    007070    030440    000200    000060    100000
**:     CM: POSITION EXPONENT
        AR: R10B=D AND R10A,SRU(0)

7554 M      071237    007070    030440    000200    000060    100000
**:     AR: R10B=IR5,D=R10A

7555 M      011125    100072    164750    000200    000060    100000
```

```
        PG:     STCFI,STCFL,STCDI,STCDL INSTRUCTIONS

7562 M      152337    007000    167000    000200    170060    057566
**: STCFI
        CM: POINT TO ACCUMULATOR
        AR: AD;R10B=[AC0.FPP]   IO: DATI(OFF)
        BR: CBN(1) [.+4]

7563 M      152337    007000    167010    000200    170060    057566
**:     AR: AD;R10B=[AC1.FPP]   IO: DATI(OFF)
        BR: CBN(1) [.+3]

7564 M      152337    007000    167020    000200    170060    057566
**:     AR: AD;R10B=[AC2.FPP]   IO: DATI(OFF)
        BR: CBN(1) [.+2]

7565 M      152337    007000    167030    000200    170060    057566
**:     AR: AD;R10B=[AC3.FPP]   IO: DATI(OFF)
        BR: CBN(1) [.+1]

7566 M      120337    006000    167356    000600    000060    077115
**:     CM: SAVE CC'S / GET ACCUMULATOR
        AR: R0B=PSR             PC: UW
        CC: C=0,N=0,Z=0,V=0     BR: CJN(1) [B.GTAC]

7567 M      000342    006600    030440    000600    000001    057571
**:     CM: CLEAR FCC'S / SKIP ON DOUBLE
        AR: R6B=0               PC: UW
        BR: CBN(V) [.+2]

7570 M      000342    007500    030440    001600    000060    100000
**:     CM: CLEAR LOW FRACTION
        AR: R15B=0              PC: 4B

7571 M      120337    006100    177356    000600    000060    077210
**:     CM: SAVE DATA FLAGS / GET SOURCE ADDRESS
```

STCFI,STCFL,STCDI,STCDL INSTRUCTIONS

```
7576 M      000042    000000    160740    001600    000013    057601
**:     CM: SET Q / BRANCH IF IN RANGE
        AR: Q=0                 PC: 4B
        CC: C=0,N=N,Z=Z,V=0     BR: CBN(N') [.+3]

7577 M      150337    006600    000001    000600    000060    077736
**:     CM: ELSE CONVERSION ERROR / SET FCC
        AR: R6B=[+1]            PC: UW
        BR: CJN(1) [FIC.TRP]

7600 M      000342    007300    167756    001600    000060    057605
**:     CM: GENERATE A CLEAN ZERO
        AR: R13B=0              PC: 4B
        CC: C=0,N=0,Z=1,V=0     BR:CBN(1) [.+5]

7601 M      000633    007400    030440    041420    000077    100000
**:     CM: SHIFT INTEGER INTO POSITION
        AR: R14B=R14B,SRU(0),SQ(MSBR)   PC: 3B,CCL
        BR: CNR(/CNTR)

7602 M      000332    007300    030440    001600    000043    057605
**:     CM: GET INTEGER / SKIP IF POSITIVE
        AR: R13B=Q              PC: 4B
        BR: CBN(/N) [.+3]

7603 M      150115    000073    100000    000200    000060    100000
**:     CM: COMPARE TO MOST NEGATIVE NUMBER
        AR: R13A-[+100000]-<0>  PC: LW

7604 M      000324    107373    030440    000200    000053    057577
**:     CM: CONVERSION ERROR IF R13>100000
        AR: R13B=-R13A-<1>      PC: LW
        BR: CBN(/N') [.-5]
```

STCFI,STCFL,STCDI,STCDL INSTRUCTIONS

```
        CC: C=CPO,N=CPO,Z=CPO,V=CPO
        BR: CBIN(1) <SCR>

7612 M      150115    140064    000040    000600    000017    057615
**: STCF.L
        CM: CHECK MAGNITUDE
        AR: R4A-[+40]-</N>      PC: UW
        BR: CBN(BLE') [.+3]

7613 M      000042    000000    167340    001600    000013    057616
**:     CM: CLEAR Q / BRANCH IF IN RANGE
        AR: Q=0                 PC: 4B
        CC: C=0,N=N,Z=0,V=0     BR: CBN(N') [.+3]

7614 M      150337    006600    000001    000600    000060    077736
**:     CM: ELSE CONVERSION ERROR / SET FCC
        AR: R6B=[+1]            PC: UW
        BR: CJN(1) [FIC.TRP]

7615 M      000342    007300    167756    001600    000060    057624
**:     CM: GENERATE A CLEAN ZERO
        AR: R13B=0              PC: 4B
        CC: C=0,N=0,Z=1,V=0     BR: CBN(1) [.+7]

7616 M      000633    007500    107340    001640    000077    057620
**:     CM: SHIFT INTEGER INTO POSITION
        AR: R15B=R15B,SRU(0),SQ(C)      PC: 4B
        CC: C=MSBR,N=N,Z=0,V=0  BR: CBN(/CNTR) [.+2]

7617 M      000733    007400    107340    041404    000060    057616
**:     AR: R14B=R14B,SRU(C)    PC: 3B,CCL
        CC: C=MSBR,N=N,Z=0,V=0  BR: CBN(1) [.-1]

7620 M      000332    007300    160440    001600    000043    057624
**:     CM: GET INTEGER
```

STCFI,STCFL,STCDI,STCDL INSTRUCTIONS

```
        AR: D=R13B              PC: UW
        BR: CBN(Z) [STCF.M0]

7626 M      152205    007070    000002    000200    100243    057611
**:     CM: BRANCH ON IMMEDIATE
        AR: R10B=R10A+[+2]+<0>,AD=R10A
        IO: DATO(CM)            BR: CBN(/N) [STCF.M0+2]

7627 M      001133    007300    030440    020200    000060    057610
**:     CM: ELSE DOUBLE
        AR: D=R13B              PC: LW,WIOL
        BR: CBN(1) [STCF.M0+1]
```

```
        PG:     STCFD/STCDF INSTRUCTIONS

7630 M      152337    007000    167000    000200    170060    057634
**: STCFD
        CM: POINT TO ACCUMULATOR
        AR: AD;R10B=[AC0.FPP]   IO: DATI(OFF)
        BR: CBN(1) [.+4]

7631 M      152337    007000    167010    000200    170060    057634
**:     AR: AD;R10B=[AC1.FPP]   IO: DATI(OFF)
        BR: CBN(1) [.+3]

7632 M      152337    007000    167020    000200    170060    057634
**:     AR: AD;R10B=[AC2.FPP]   IO: DATI(OFF)
        BR: CBN(1) [.+2]

7633 M      152337    007000    167030    000200    170060    057634
**:     AR: AD;R10B=[AC3.FPP]   IO: DATI(OFF)
        BR: CBN(1) [.+1]

7634 M      120337    006000    167356    000600    000060    077115
**:     CM: SAVE CC'S / GET ACCUMULATOR
        AR: R0B=PSR             PC: UW
        CC: C=0,N=0,Z=0,V=0     BR: CJN(1) [B.GTAC]

7635 M      120177    000000    030540    000200    000060    100000
**:     CM: COMPLEMENT LENGTH
        AR: OUT=/PSR            CC: V=CPO,C=C,N=N,Z=Z

7636 M      120337    006100    177456    000600    000060    077135
**:     CM: SAVE TCC'S / GET DESTINATION ADDRESS
        AR: R1B=PSR             PC: UW
        CC: C=1,Z=1,N=0,V=V     BR: CJN(1) [FSRC.SAV]

7637 M      150137    000000    006070    000200    040460    100120
**:     CM: SET UP END BRANCH
```

STCFD/STCDF INSTRUCTIONS

```
7644 M      000100    000000    030440    000200    000041    057646
**: CNV.RND
        CM: SKIP ON DOUBLE SOURCE
        BR: CBN(/V) [.+2]

7645 M      000342    007500    030440    001600    000060    120000
**:     CM: ELSE CLEAR LOWER FRACTION ON FLOATING SOURCE
        AR: R15B=0              PC: 4B
        BR: CRR(1)

7646 M      150337    006100    000040    000600    000002    130000
**:     CM: RETURN ON ZERO SOURCE
        AR: R1B=[+40]           PC: UW
        BR: CRN(Z)

7647 M      000141    006176    030440    000600    000060    100000
**:     CM: CHECK TRUNCATE
        AR: R16A AND R1B        PC: UW

7650 M      000100    000000    030440    000200    000052    130000
**:     CM: RETURN IF CHOPPING
        BR: CRN(/Z')

7651 M      000133    007500    100440    001600    000060    076705
**:     CM: ELSE ROUND FLOATING
        AR: R15B                PC: 4B
        CC: C=PN,N=N,Z=Z,V=V    BR: CJN(1) [NORM.24A]

7652 M      150115    100064    000400    000600    000060    057731
**:     CM: CHECK OVERFLOW ON ROUNDING
        AR: R4A-[+400]-<1>      PC: UW
        BR: CBN(1) [EXP.OV+1]
```

```
        PG:     LDEXP INSTRUCTION

7653 M      152337    007000    167000    020200    000060    057657
**: LDEXP
        CM: POINT TO ACCUMULATOR
        AR: AD;R10B=[AC0.FPP]   PC: LW,WIOL
        BR: CBN(1) [.+4]

7654 M      152337    007000    167010    020200    000060    057657
**:     AR: AD;R10B=[AC1.FPP]   PC: LW,WIOL
        BR: CBN(1) [.+3]

7655 M      152337    007000    167020    020200    000060    057657
**:     AR: AD;R10B=[AC2.FPP]   PC: LW,WIOL
        BR: CBN(1) [.+2]

7656 M      152337    007000    167030    020200    000060    057657
**:     AR: AD;R10B=[AC3.FPP]   PC: LW,WIOL
        BR: CBN(1) [.+1]

7657 M      000337    006200    030440    000600    170060    100000
**:     CM: SAVE DATA / GET ACCUMULATOR
        AR: R2B=D               PC: UW
        IO: DATI(OFF)

7660 M      120337    006000    167356    000600    000060    077115
**:     CM: SAVE CC'S
        AR: R0B=PSR             PC: UW
        CC: C=0,N=0,Z=0,V=0     BR: CJN(1) [B.GTAC]

7661 M      150137    000000    006070    000200    040460    100120
**:     CM: SET UP END BRANCH
        AR: SCE5=[BGN]          SP: CLR,SE5

7662 M      150337    006400    000200    000600    000060    100000
**:     CM: SET NEW EXPONENT / SAVE
```

```
        PG:     LDCIF/LDCLF/LCDID/LCDLD INSTRUCTIONS

7664 M      150337    007100    167000    000200    000060    057670
**: LDCIF
        CM: POINT TO ACCUMULATOR
        AR: R11B=[AC0.FPP]      BR: CBN(1) [.+4]

7665 M      150337    007100    167010    000200    000060    057670
**:     AR: R11B=[AC1.FPP]      BR: CBN(1) [.+3]

7666 M      150337    007100    167020    000200    000060    057670
**:     AR: R11B=[AC2.FPP]      BR: CBN(1) [.+2]

7667 M      150337    007100    167030    000200    000060    057670
**:     AR: R11B=[AC3.FPP]      BR: CBN(1) [.+1]

7670 M      120337    006000    167356    000600    000060    077210
**:     CM: SAVE CC'S / GET INTEGER
        AR: R0B=PSR             PC: UW
        CC: C=0,N=0,Z=0,V=0     BR: CJN(1) [ISRC.SAV]

7671 M      150337    006400    000271    000600    000060    077051
**:     CM: NORMALIZE INTEGER
        AR: R4B=[+271]          PC: UW
        BR: CJN(1) [NORM.56]

7672 M      000133    007600    030600    000400    000002    057674
**:     CM: SET LENGTH / SKIP ON ZERO
        AR: R16B                PC: UB
        CC: C=C,N=N,Z=Z,V=PN    BR: CBN(Z) [.+2]

7673 M      000100    000000    030440    000200    000041    077646
**:     CM: CHECK ROUNDING IF RESULT FLOATING
        BR: CJN(/V) [CNV.RND+2]

7674 M      000334    007071    030440    000200    000060    077246
```

```
        PG:     LDCDF/LDCFD INSTRUCTIONS

7676 M      150337    007100    167000    000200    000060    057702
**: LDCDF
        CM: POINT TO ACCUMULATOR
        AR: R11B=[AC0.FPP]      BR: CBN(1) [.+4]

7677 M      150337    007100    167010    000200    000060    057702
**:     AR: R11B=[AC1.FPP]      BR: CBN(1) [.+3]

7700 M      150337    007100    167020    000200    000060    057702
**:     AR: R11B=[AC2.FPP]      BR: CBN(1) [.+2]

7701 M      150337    007100    167030    000200    000060    057702
**:     AR: R11B=[AC3.FPP]      BR: CBN(1) [.+1]

7702 M      121237    006076    030440    000600    000060    100000
**:     CM: SAVE CC'S / SET UP LENGTH FLAG
        AR: R0B=PSR,D=R16A      PC: UW

7703 M      000177    000000    167616    000000    000060    077135
**:     CM: GET SOURCE USING SPECIFIED LENGTH
        AR: OUT=/D              PC: LB
        CC: C=0,Z=1,V=PN,N=0    BR: CJN(1) [FSRC.SAV]

7704 M      150137    000000    007707    000200    040460    100120
**:     CM: SET UP END BRANCH
        AR: SCE5=[.+3]          SP: CLR,SE5

7705 M      150337    006100    000014    000600    000000    077716
**:     CM: CHECK VARIABLE AND SET FCC'S
        AR: R1B=[+14]           PC: UW
        BR: CJN(C) [FIUV.FPP+1]

7706 M      120345    006661    030440    000600    000060    164000
**:     CM: SET FCC'S
```

LDCDF/LDCFD INSTRUCTIONS

```
**:     CM: RESTORE CC'S
        AR: R0B=PSR,OUT=R0A     PC: UW
        CC: C=CPO,N=CPO,Z=CPO,V=CPO

7714 M      000331    006660    030440    000600    000060    164000
**:     CM: FINALIZE FCC'S / FINISHED / TRAP
        AR: R6B=R6B OR R0A      PC: UW
        BR: CBIN(1) <SCR>
```

```
        PG:     FIUV - UNDEFINED VARIABLE CHECK

CM:     ROUTINE CHECKS FOR -0 (IE. Z=1 AND N=1)
        AND CHECKS IF FIUV INTERRUPT IS ENABLED
        IF -0 FOUND AND FIUV=0 THEN
                C=1 FOR CLEAN ZERO
        IF -0 FOUND AND FIUV=1 THEN
                C=0 FOR NON CLEAN ZERO
                GO CHECK FID AND SET UP TRAP
        ELSE RETURNS

7715 M      000100    000000    030440    000200    000042    130000
**: FIUV.FPP
        CM: RETURN IF NOT ZERO
        BR: CRN(/Z)

7716 M      151145    000076    004000    000600    000043    130000
**:     CM: RETURN IF PLUS / TEST FIUV BIT
        AR: D=R16A AND [+4000]  PC: UW
        BR: CRN(/N)

7717 M      001137    000000    110440    000200    000012    130000
**:     CM: SET FOR CLEAN ZERO AND RETURN IF FIUV BIT CLEAR
        AR: D=D
        CC: C=PZ,N=N,Z=Z,V=V    BR: CRN(Z')

7720 M      151137    000000    000014    020200    000060    057737
**: FIUV.TRP
        CM: ELSE INHIBIT CLEAN ZERO WHEN TRAPPING
            AND GO CHECK FID AND SET UP TRAP
        AR: D=[+14]             PC: LW,WIOL
        BR: CBN(1) [FPP.TRP]
```

```
        PG:     EXPONENT UNDERFLOW CHECK

CM:     ROUTINE CHECKS FOR EXPONENT UNDERFLOW (IE. R4 (UW) <= 0)
        AND CHECKS IF FIU INTERRUPT IS ENABLED
        IF UNDERFLOW AND FIU=0
                THEN MASK EXPONENT, SETUP CLEAN ZERO
        IF UNDERFLOW AND FIU=1
                THEN MASK EXPONENT, INHIBIT CLEAN ZERO,
                GO CHECK FID AND SET UP TRAP
        ELSE CHECKS OVERFLOW

7721 M      000342    006600    030440    000600    000060    100000
**: EXP.UN
        CM: CLEAR FCC'S
        AR: R6B=0               PC: UW

7722 M      000303    006400    030440    000600    000060    100000
**:     CM: CHECK EXPONENT
        AR: R4B=R4B+<0>         PC: UW

7723 M      150145    000076    002000    000600    000057    057730
**:     CM: BRANCH IF NO UNDERFLOW
        AR: R16A AND [+2000]    PC: UW
        BR: CBN(BGT') [EXP.OV]

7724 M      150345    006464    000377    000600    000052    057726
**:     CM: MASK EXPONENT
        AR: R4B=R4A AND [+377]  PC: UW
        BR: CBN(/Z') [.+2]

7725 M      000342    006400    170440    000600    000060    120000
**:     CM: SETUP FOR A CLEAN ZERO ON FIU=0
        AR: R4B=0               PC: UW
        CC: C=1,Z=Z,N=N,V=V     BR: CRR(1)

7726 M      151137    000000    000012    020200    000060    100000
```

```
        PG: EXPONENT OVERFLOW CHECK

CM:     ROUTINE CHECKS FOR EXPONENT OVERFLOW
        AND CHECKS IF FIV INTERRUPT IS ENABLED
        IF OVERFLOW AND FIV=0
                THEN MASK EXPONENT AND SET FOR CLEAN ZERO
        IF OVERFLOW AND FIV=1
                THEN MASK EXPONENT, INHIBIT CLEAN ZERO,
                GO CHECK FID AND SET UP TRAP
        ELSE RETURNS

7730 M      150115    100064    000400    000600    000060    100000
**: EXP.OV
        CM: CHECK FOR OVERFLOW
        AR: R4A-[+400]-<1>      PC: UW

7731 M      150145    000076    001000    000600    000013    130000
**:     CM: RETURN IF NO OVERFLOW
        AR: R16A AND [+1000]    PC: UW
        BR: CRN(N')

7732 M      150345    006464    000377    000600    000052    057734
**:     CM: ELSE MASK EXPONENT
        AR: R4B=R4A AND [+377]  PC: UW
        BR: CBN(/Z') [.+2]

7733 M      000342    006400    170440    000600    000060    057735
**:     CM: SETUP FOR CLEAN ZERO
        AR: R4B=0               PC: UW
        CC: C=1,Z=Z,N=N,V=V     BR: CBN(1) [.+2]

7734 M      151137    000000    000010    020200    000060    077727
**: FIV.TRP
        CM: SET FEC FOR OVERFLOW
        AR: D=[+10]             PC: LW,WIOL
        BR: CJN(1) [FIU.TRP+1]
```

```
        PG:     FLOATING TRAP PROCESSING

7736 M      151137    000000    000006    020200    000060    100000
**: FIC.TRP
        CM: SET FEC FOR INTEGER CONVERSION ERROR
        AR: D=[+6]              PC: LW,WIOL

7737 M      152137    000000    167102    000200    170260    100000
**: FPP.TRP
        CM: SAVE FEC CODE
        AR: AD=[FEC.FPP]        IO: DATO(OFF)

7740 M      150335    007676    100000    000600    000060    100000
**:     CM: SET FER BIT IN FPS
        AR: R16B=R16A OR [+100000]      PC: UW

7741 M      101137    000000    030440    020200    000060    100001
**:     CM: SAVE INSTRUCTION LOCATION
        AR: D=SCR1              PC: LW,WIOL

7742 M      150145    000076    040000    000600    000060    100000
**:     CM: CHECK INTERRUPT DISABLE FID
        AR: R16A AND [+40000]   PC: UW

7743 M      152137    000000    167100    020200    170252    130000
**:     CM: SAVE FEA / RETURN IF INTERRUPT DISABLED
        AR: AD=[FEA.FPP]        PC: LW,WIOL
        IO: DATO(OFF)           BR: CRN(/Z')

7744 M      150137    000000    007745    000200    040460    120120
**:     CM: ELSE SET UP TRAP
        AR: SCE5=[TRAP.FPP]     SP: CLR,SE5
        BR: CRR(1)

7745 M      000133    006000    063146    000600    000060    056544
**: TRAP.FPP
```

```
        PG:     SPCL INSTRUCTIONS

        CM: DEFINE SPCL ENTRY POINT

        .=: IORIGIN+172

6172 I      007746    002100
**:     MA: SPCL        PS: EX          IN: BYT



        CM:     SPECIAL INSTRUCTION TO ACCESS MICROCODE ROUTINES

        220-223   MICJMP - JUMP TO MICROCODE LOCATION

        THE GENERAL REGISTER R0-R3 CONTAINS THE ABSOLUTE
        MICROCODE ADDRESS TO WHICH CONTROL IS PASSED

        CM: SPECIAL INSTRUCTIONS TO ACCESS ANY MEMORY LOCATION

        224     GETW - GET WORD OF DATA
        225     GETB - GET BYTE OF DATA
        226     PUTW - PUT WORD OF DATA
        227     PUTB - PUT BYTE OF DATA

        SOURCE/DESTINATION DATA IN R0
        LOW 16 BIT ADDRESS IN R1
        HIGH 6 BIT ADDRESS IN R2

        .=: MORGEND

7746 M      052237    007061    030440    000200    000060    100000
**: SPCL
        CM: GET IR / LOAD AD
        AR: R10B=IR,AD=R1A
```

SPCL INSTRUCTIONS

```
7754 M      002237    006067    034750    010200    171760    056073
**:     AR: R0B=D,AD=R7A        PC: LW,WIOH
        CC: C=C,V=0,N=PN,Z=PZ
        IO: INTCK               BR: CBN(1) [BGN+3]

7755 M      000100    000000    030440    000200    173060    100000
**: GETB
        CM: GET BYTE INSTRUCTION
        IO: IDATI(OFF,BYT)

7756 M      032237    006067    034750    010200    171760    056073
**:     AR: R0B=DSX7,AD=R7A     PC: LW,WIOH
        CC: C=C,V=0,N=PN,Z=PZ
        IO: INTCK               BR: CBN(1) [BGN+3]

7757 M      003134    000062    030440    000200    000052    057761
**: PUT.WB
        CM: LOAD ADX / BRANCH IF BYTE
        AR: ADX=R2A             BR: CBN(/Z') [PUTB]

7760 M      001133    006000    034750    000200    170260    056070
**: PUTW
        CM: PUT WORD INSTRUCTION
        AR: D=R0B       CC: C=C,V=0,N=PN,Z=PZ
        IO: DATO(OFF)   BR: CBN(1) [BGN]

7761 M      001133    006000    034750    000000    172260    056070
**: PUTB
        CM: PUT BYTE INSTRUCTION
        AR: D=R0B       PC: LB
        CC: C=C,V=0,N=PN,Z=PZ
        IO: DATO(OFF,BYT)       BR: CBN(1) [BGN]

7762 M      000134    000020    030440    000200    040460    100120
**: MICJMP
```

BLUCPU MICROCODE MICROASSEMBLER V02 15-Oct-2007 16:18:43

SYMBOL TABLE

A.BLDSEF	006555	A.GETACC	007105	A.GETARG	006545
ABSF	007327	AC0.FPP	167000	AC1.FPP	167010
AC2.FPP	167020	AC3.FPP	167030	AC4.FPP	167040
AC5.FPP	167050	AC6.FPP	167060	AC7.FPP	167070
ACC.SAV	007246	ACC.SAVA	007247	ADC	006236
ADD	006214	ADD.24	006577	ADD.24A	006610
ADD.56	006710	ADD.56A	006717	ADDF	007445
AL.24	006644	AL.24A	006651	AL.24B	006661
AL.56	007000	AL.56A	007007	AL.56B	007030
ANSWER	007454	ARG.SAV	006520	ARG.SAVA	006521
ASH	006400	ASHC	006413	ASL	006265
ASR	006263	B.BLDSEF	006572	B.GETACC	007116
B.GETARG	006562	B.GETINT	007237	B.GTAC	007115
BGN	006070	BIC	006203	BINT	006016
BIS	006210	BIT	006177	BR	006067
CFCC	007271	CHLT	006110	CLR	006224
CLRC	006267	CLRF	007320	CMB.SEF	006530
CMP	006173	CMPF	007501	CNV.RND	007644
COM	006226	CONS	006112	CONT	006130
COPY.24	006602	COPY.56	006713	DEC	006232
DEP	006143	DEP1	006104	DIV	006450
DIV.24	006633	DIV.56	006757	DIV.56A	006776
DIVF	007530	DIVZERO	006536	DST	006030
END	006150	END1	006412	END2	006427
END3	006447	EX	006124	EX1	006100
EXP.OV	007730	EXP.UN	007721	EXPCHK	006514
FADD	006507	FBHINT	006077	FDIV	006471
FDST	007150	FEA.FPP	167100	FEC.FPP	167102
FIC.TRP	007736	FIS.SAV	006516	FIS.TRAP	006541
FIU.TRP	007726	FIUV.FPP	007715	FIUV.TRP	007720
FIV.TRP	007734	FMUL	006477	FOP.TRP	007267
FPP.TRP	007737	FRC.SAV	007432	FS.IM	007157
FS.M0	007142	FS.M1	007146	FS.M2	007152
FS.M3	007156	FS.M4	007162	FS.M5	007166

BLUCPU MICROCODE MICROASSEMBLER V02 15-Oct-2007 16:18:43

SYMBOL TABLE

NEGF	007333	NGN	006445	NO	006162
NOBR	006066	NORM.24	006671	NORM.24A	006705
NORM.56	007051	NORM.56A	007100	OD2	006047
OD3	006051	OD4	006052	OD5	006054
OD6	006056	OD7	006050	OS3	006041
OS4	006044	OS5	006043	OS6	006042
OS7	006040	OVRFLO	006540	PF	006012
PSN	006442	PULL	006057	PUSH	006002
PUSHA	006006	PUT.WB	007757	PUTB	007761
PUTW	007760	REG	006156	RES	006000
RESEND	006003	RESET	006315	ROL	006261
ROR	006255	RSHC	006424	RSRVINST	006327
RSRVSTAT	006331	RTI	006307	RTS	006302
RTST	006410	SBC	006240	SEND	006045
SET.FPS	007256	SETC	006270	SETD	007274
SETF	007272	SETI	007273	SETL	007275
SOB	006271	SPCL	007746	SPL	006334
SR0	006260	SRC	006020	SRD	006257
STCF.I	007575	STCF.L	007612	STCF.M0	007607
STCFD	007630	STCFI	007562	STEXP	007545
STF	007517	STFPS	007301	STRT	006135
STST	007311	SUB	006220	SUBF	007473
SWAB	006245	SXT	006251	TRAP.FPP	007745
TST	006242	TSTF	007323	TSTR	006154
UNDFLO	006537	WAIT	006317	XOR	006253
YSTB	006011				
.	007764				

ERRORS DETECTED IN PASS 1 = 0

ERRORS DETECTED IN PASS 2 = 0

BLUXHI[40],BLUXHI[600]=BLUXHI,EISX,FISX,FPPX,SPCLX,EN
X

```
SET ERROR NONE
DELETE/NOQ BLUXLO.WCS,BLUXLO.LST
SET ERROR ERROR
!
R MICRO
BLUXLO[40],BLUXLO[600]=BLUXLO,EISX,FISX,FPPX,SPCLX,ENDX
^C
```

```
        TT:     BLUCPU MICROCODE

        SB:     TITLE PAGE

        CM:     EMULATION OF THE PDP 11/34 COMPUTER
                CODE DATED MARCH 1980
                ALAN R. BALDWIN

        CM:     CORRECTION TO MTPS (USER MODE) - MAY 1980
        CM:     CORRECTION TO MTOUT (MAINTENANCE MODE) - MAY 1980
        CM:     CORRECTION OF RTI/RTT INSTRUCTIONS - JULY 1980
        CM:     REWRITE OF TRAP HANDLERS - JULY 1980
        CM:     ADDITION OF RESERVED INSTRUCTION HANDLER - JULY 1980
        CM:     ADDITION OF SPL INSTRUCTION - JULY 1980
        CM:     CORRECTION OF RSRVINST/RSRVSTAT - MAY 1986

        CM:     PROGRAM MICROCODE ORIGIN
        NA:             MORIGIN = +0

        CM:     INSTRUCTION CODE ORIGIN
        NA:             IORIGIN = +0


        PG:     TRAP PROCESSING

        .=: MORIGIN+0

        CM: TRAP SEQUENCE ENTRY POINT
            RESERVED INSTRUCTIONS
            TRAP, EMT, IOT, BPT INSTRUCTIONS

**: RES
        AR: AD;R10B=PV  PC: LW, WIOL
        IO: DATI(KI)

**:     AR: R10B=R10A-[+2]-<1>  BR: CJN(1) [PULL+2]

**: PUSH
        IO: IDATO(CMI)          BR: CJN(1) [PUSHA]

**: RESEND
        AR: AD=R7A      PC: LW, WIOL
        IO: INTCK       BR:  CBIC(0)

**:     AR: SCR0=0      PC: LW, CFP
        IO: IDATIR(CMI)

**:     AR: AD;R7B=R7A+[+2]+<0> PC: LW, WIOL
        IO: INTCK       BR: CBR(1) [BGN+2]

**: PUSHA
        AR: AD=R6A+[+2]+<0>     PC: LW,WIOL
        SP: CLR

**:     AR: D=R11A        IO: DATO(CMI)
        BR: CRN(MMGT)

**:     AR: SOR=R6A-SLR-<1>
        BR: CRR(1)

**: YSTB
        CM: ENTRY POINT FOR YELLOW STACK AND T BIT TRAPS
        AR: R7B=R7A-[+2]-<1>
        BR: CBN(1) [RES]

**: PF
        CM: POWER FAIL ENTRY POINT
        AR: AD;R10B=PV  PC: LW, WIOL
        IO: DATI(KI)    BR: CJN(1) [PULL]

**:     BR: BROC(/PFD) [PUSH]

        PG:     INTERRUPT HANDLERS

**: INT17
        CM: ENTRY POINT FOR INTERNAL
```

```
        CM: INTERRUPT LEVELS 1 THROUGH 7
        AR: AD;R10B=PV  PC: LW, WIOL
        IO: DATI(KI)    SP: CI

**:     AR: R7B=R7A-[+2]-<1>    BR: CBN(1) [RES+1]

**: BINT
        AR: R10B=[+2]   PC: LW, WIOL
        IO: IINT

**:     AR: AD;R10B=R10A+D+<0>  PC: LW,WIOH
        IO: DATI(KI)    BR: CBN(1) [RES+1]

        PG:     SOURCE MODE SEQUENCE

        .=: MORIGIN+20
        CM: SOURCE MODE CODE
            FOR ALL MODES RESULTS ARE
            (1)  D & R10 CONTAIN DATA FROM ADDRESS
            (2)  STACK OVERFLOW CHECKED IN MODES  4 AND 5
            UPON ENTRY AND EXIT  AD = R7

**: SRC
        CM: (R) IS OPERAND
        AR: D=RA(SRC)   BR:  CBR(1) [SEND]

**:     CM: (R) IS ADDRESS
        AR: AD=RA(SRC)  PC: LW, WIOL
        IO: DATI(CMI,BYT)       BR: CBR(1) [SEND]

**:      CM: (R) IS ADDRESS; (R) + (1 OR 2)
        AR: RB(SRC)=RA(SRC)+[+1]+</SR67W>, AD=RA(SRC)   PC: LW, WIOL
        IO: DATI(CMI,BYT)       BR: CBR(1) [SEND]

**:     CM: (R) IS ADDRESS OF ADDRESS; (R) + 2
        AR: RB(SRC)=RA(SRC)+[+2]+<0>, AD=RA(SRC)        PC: LW, WIOL
        IO: DATI(CMI)   BR: CBR(1) [OS3]

**:     CM: (R) - (1 OR 2); (R) IS ADDRESS
        AR: AD;RB(SRC)=RA(SRC)-[+1]-<SR67W>     PC: LW, WIOL
        IO: DATI(CMI,BYT)       BR: BROC(MMGT) [OS4]

**:     CM: (R) - 2; (R) IS ADDRESS OF ADDRESS
        AR: AD;RB(SRC)=RA(SRC)-[+2]-<1> PC: LW, WIOL
        IO: DATI(CMI)   BR: CBR(1) [OS5]

**:     CM: (R) + X  IS ADDRESS
        AR: R7B=R7A+[+2]+<0>    PC: LW, WIOL
        IO: IDATI(CMI)  BR: CBR(1) [OS6]

**:     CM: (R) + X  IS ADDRESS OF ADDRESS
        AR: R7B=R7A+[+2]+<0>    PC: LW, WIOL
        IO: IDATI(CMI)  BR: CBR(1) [OS7]

        PG:     DESTINATION MODE SEQUENCE

        .=: MORIGIN+30
        CM: DESTINATION MODE CODE
            FOR ALL MODES
            UPON ENTRY
            (1) AD = R7
            (2) R10 CONTAINS DATA OF PRECEEDING SOURCE MODE
            UPON EXIT
            (1) AD & R11 CONTAIN ADDRESS OF DESTINATION DATA
            (2) D CONTAINS DATA AT DESTINATION ADDRESS
            (3) STACK OVERFLOW CHECKED IN MODES 4 & 5

**: DST
        CM: (R) IS OPERAND
        AR: R10B=D, D=RA(DST)   BR: CBIN(1) <INST>

**:     CM: (R) IS ADDRESS
        AR: AD;R11B=RA(DST)     PC: LW, WIOL
        IO: DATIP(CM,BYT,NDSP)  BR: CBIN(1) <INST>

**:     CM: (R) IS ADDRESS; (R) + (1 OR 2)
```

```
        AR: AD;R11B=RA(DST)     PC: LW, WIOL
        IO: DATIP(CM,BYT,NDSP)  BR: CBR(1) [OD2]

**:     CM: (R) IS ADDRESS OF ADDRESS; (R) + 2
        AR: RB(DST)=RA(DST)+[+2]+<0>, AD=RA(DST)        PC: LW, WIOL
        IO: DATI(CMI)   BR: CBR(1) [OD3]

**:     CM: (R) - (1 OR 2); (R) IS ADDRESS
        AR: AD;RB(DST)=RA(DST)-[+1]-<DR67W>     PC: LW, WIOL
        IO: DATIP(CM,BYT,NDSP)  BR: CBR(1) [OD4]

**:     CM: (R) - 2; (R) IS ADDRESS OF ADDRESS
        AR: AD;RB(DST)=RA(DST)-[+2]-<1> PC: LW, WIOL
        IO: DATI(CMI)   BR: CBR(1) [OD5]

**:     CM: (R) + X  IS ADDRESS
        AR: R7B=R7A+[+2]+<0>    PC: LW, WIOL
        IO: IDATI(CMI)  BR: CBR(1) [OD6]

**:     CM: (R) + X  IS ADDRESS OF ADDRESS
        AR: R7B=R7A+[+2]+<0>    PC: LW, WIOL
        IO: IDATI(CMI)  BR: CBR(1) [OD7]

        PG:     SOURCE & DESTINATION COMPLETION

**: OS7
        CM: COMPLETION OF SOURCE MODES
        AR: AD=RA(SRC)+D+<0>  PC: LW, WIOH      IO: DATI(CMI)

**: OS3
        AR: AD=D        PC: LW, WIOH
        IO: DATI(CMI,BYT)       BR: CBR(1) [SEND]

**: OS6
        AR: AD=RA(SRC)+D+<0>
        PC: LW, WIOH    IO: DATI(CMI,BYT)       BR: CBR(1) [SEND]

**: OS5
        AR: AD=D        PC: LW, WIOH
        IO: DATI(CMI,BYT)       BR: CBN(MMGT) [.+2]

**: OS4
        AR: SOR=R6A-SLR-<1>

**: SEND
        AR: R10B=D, AD=R7A      PC: LW, WIOH
        IO: INTCK       BR: CBIN(/LKDC) <INST>

**:     IO: ILDATIR(CMI)        BR: CBIC(1) <INST>

**: OD2
        AR: RB(DST)=RA(DST)+[+1]+</DR67W>
        BR: CBIN(1) <INST>

**: OD7
        AR: AD=RA(DST)+D+<0>    PC: LW, WIOH
        IO: DATI(CMI)

**: OD3
        AR: AD;R11B=D   PC: LW, WIOH
        IO: DATIP(CM,BYT,NDSP)  BR: CBIN(1) <INST>

**: OD4
        AR: R11B=RA(DST)        BR: CBIN(MMGT) <INST>

**:     AR: SOR=R6A-SLR-<1>     BR: CBIC(1) <INST>

**: OD5
        AR: AD;R11B=D   PC: LW, WIOH
        IO: DATIP(CM,BYT,NDSP)  BR: CBIN(MMGT) <INST>

**:     AR: SOR=R6A-SLR-<1>     BR: CBIC(1) <INST>

**: OD6
        AR: AD;R11B=RA(DST)+D+<0>       PC: LW, WIOH
        IO: DATIP(CM,BYT,NDSP)  BR: CBIN(1) <INST>
```

```
        PG:     PULL VECTOR HANDLER

**: PULL
        CM: TRAP HANDLER PULL ROUTINE
        AR: R7B=R7A-[+2]-<1>

**:     AR: R10B=R10A-[+2]-<1>

**:     AR: R10B=D,AD=R10A      PC: LW,WIOH
        IO: DATI(KI)

**:     AR: R11B=PSR,PSR=R10A   PC: LW, WIOL
        CC: C=CPO, V=CPO, Z=CPO, N=CPO

**:     AR: R7B=D, D=R7A        BR: CBN(CUM) [.+2]

**:     AR: AD;R6B=R6A-[+4]-<1> BR: CRR(1)

**:     AR: AD;R16B=R6A-[+4]-<1>        BR: CRR(1)

**: NOBR
        CM: NO BRANCH
        BR: BROC(LKD) [BGN]

**: BR
        CM: BRANCH INSTRUCTION
        AR: AD;R7B=R7B+R10A+<0>
        IO: INTCK       BR: CBR(1) [BGN+3]

        PG:     INSTRUCTION LOAD & DECODE

        .=: MORIGIN+70

**: BGN
        AR: AD=R7A+[+0]+<0>     PC: LW, WIOL
        IO: INTCK       BR: CBR(1) [.+3]

**:     AR: AD;R7B=R7A+[+2]+<0> PC: LW, WIOL
        IO: INTCK       SP: ILE

**:     AR: R10B=ISX7, SRU(0)
        IO: ILDATIR(CMI)        BR: CBIN(1) <INST>

**:     IO: IDATIR(CMI) PC: LW, WIOL
        BR: CBR(1) [BGN+1]

**: INT0
        CM: ENTRY POINT FOR BUS INTERRUPTS
        CM: AND LEVEL 0 INTERNAL INTERRUPTS
        AR: R7B=R7A-[+2]-<1>    BR: CBN(BINT) [BINT]

**:     AR: R7B=R7A+[+2]+<0>    BR: CBN(EIL) [INT17]

**:     AR: AD;R7B=R7A-[+2]-<1>
        IO: INTCK       BR: CBR(1) [BGN+3]

**: FBHINT
        CM: ENTRY POINT FOR 'FBH' INTERRUPTS
            TIME OUT, MEMORY MANAGEMENT, AND ODD ADDRESS TRAPS
            REQUIRE ONE CYCLE DELAY FOR HARDWARE TIMING AFTER 'FBH' CYCLE
        BR: CBR(1) [RES]

        PG:     CONSOLE SWITCH HANDLER

        .=: MORIGIN+100

**: EX1
        CM: RESTORE CC'S, BRANCH TO DESIGNATED EXAM
        AR: R12B=PSR, OUT=R12A  CC: C=CPO, Z=CPO, V=CPO, N=CPO
        BR: CBZV(Z) [.]

**:     CM: MOVE ADDRESS TO ADX, START DATI, BRANCH
        AR: ADX=R17A    PC: UW, CB
        IO: DATI(OFF)   BR: CBR(1) [END+1]
```

```
**:     CM: MOVE REGISTER ADDRESSED BY D INTO D (R20-R37)
        AR: D=RA(D)  PC: UW     BR: CBR(1) [END+1]

**:     CM: MOVE REGISTER ADDRESSD BY D INTO D (R0-R17)
        AR: D=RA(D)     PC: LW  BR: CBR(1) [END+1]

**: DEP1
        CM: RESTORE CC'S, BRANCH TO DEP MODE
        AR: R12B=PSR, OUT=R12A  CC: C=CPO, Z=CPO, V=CPO, N=CPO
        BR: CBZV(Z) [.]

**:     CM: PUT DATA IN D, DO A DATO, BRANCH
        AR: D=SW1       PC: LW, CB
        IO: DATO(OFF)   BR: CBR(1) [END+1]

**:     CM: STORE DATA (R15) IN REG POINTED AT BY D
        AR: D;RB(D)=R15A        PC: UW  BR: CBR(1) [END+1]

**:     CM: STORE DATA (R15) IN REG POINTED AT BY D
        AR: D;RB(D)=R15A        PC: LW  BR: CBR(1) [END+1]

**: CHLT
        CM: CONSOLE INTERRUPT ENTRY POINT
        AR: AD;R7B=R7A-[+2]-<1>
        BR: CBR(1) [CONS]

**: HALT
        CM: HALT INSTRUCTION ENTRY POINT
        BR: CBN(CUM) [RES]

**: CONS
        CM: PUT IR IN D
        AR: D=IR        PC: LW

**:     CM:  MASK WORD FOR CONTROL SWITCHES
        AR: R14B=[+174000]

**:     CM:  INITIALIZE CONSOLE CONDITION CODES
        AR: SCR0;R12B=0

**: GO
        CM:  MASK SWITCH DATA & TEST FOR ACTION
        AR: R15B=SW2 AND R14A   BR: CNR(/Z')    PC: LW, CB

**:     CM:  CHECK SWITCH FOR RELEASE
        AR: SW2 AND R14A        BR: CNR(Z')     PC: LW, CB

**:     CM: EXCHANGE PROGRAM & ROUTINE CC'S, TEST FOR REG
        AR: R12B=PSR, OUT=R12A  CC: C=CPO, Z=CPO, V=CPO, N=CPO
        BR: CJR(1) [TSTR]

**:     CM: TEST SIGN AND SHIFT DATA
        AR: R15B=R15B, SRU(0)

**: LAD
        CM: TEST & SHIFT, BRANCH IF NOT LAD
        AR: R15B=R15B, SRU(0)   BR: CBN(/N') [EX]

**:     CM: SET CODES FOR NEW ADDRESS
        AR: AD;R17B=SW1 CC: C=0, N=0, Z=Z, V=V

**:     CM: MOVE UPPER ADDRESS TO R17 & AD
        AR: ADX;R17B=SW2        PC: UW  BR: CBR(1) [END]

**: EX
        CM: TEST SIGN & SHIFT, IF NOT EX BRANCH
        AR: R15B=R15B, SRU(0)   BR: CBN(/N') [CONT]

**:     CM: SET CODES, JSR TO INCR IF SECOND TIME
        CC: C=1, N=0, V=V, Z=Z  BR: CJN(C) [INCR]

**:     CM: MOVE ADDRESS INTO AD
        AR: AD=R17A

**:     CM: MOVE ADDRESS INTO D
        AR: D=R17A      BR: CBR(1) [EX1]
```

```
**: CONT
        CM: SHIFT & TEST SIGN, IF NOT CONT BRANCH
        AR: R15B=R15B, SRU(0)   BR: CBN(/N') [STRT]

**:     CM: MOVE PC INTO AD, ENABLE INTERRUPT LOGIC
        AR: AD=R7A      SP: ILE PC: LW, CFP

**:     CM: MOVE PCX INTO ADX
        AR: ADX=R7A     PC: UW

**:     CM: RESTORE CC'S, LOAD NEXT INSTRUCTION
        AR: R12B=PSR, OUT=R12A  CC: C=CPO, Z=CPO, V=CPO, N=CPO
        PC: LW, CB      IO: DATIR(CMI)

**:     CM: WAIT FOR IO COMPLETE
        PC: LW, WIOL, CB        BR: CBR(1) [BGN+1]

**: STRT
        CM: SHIFT & TEST SIGN, BRANCH IF NOT STRT
        AR: R15B=R15B, SRU(0)   BR: CBN(/N') [DEP]

**:     CM: LOAD CNTR, SEQUENCE TO CLEAR PRIORITIES
        AR: CNTR=[+177767]
        SP: ILD, CLR    PC: LW, CFP

**:     CM: SEQUENCE PRIORITIES
        AR: PSR=0       CC: C=0, Z=0, V=0, N=0
        PC: LW, CCL     BR: CBIN(0)

**:     CM: REPEAT UNTIL SEQUENCE COMPLETE
        BR: CBN(CNTR) [.-1]

**:     CM: LOAD NEW PC, DO A BUS INIT, ENABLE INTERRUPT LOGIC
        AR: AD;R7B=R17A IO: INIT        SP: ILE PC: LW, CFP

**:     CM: LOAD NEW PCX, LOAD NEXT INSTRUCTION
        AR: ADX;R7B=R17A        PC: UW, WIOL
        IO: DATIR(CMI)  BR: CBR(1) [BGN+1]

**: DEP
        CM: BRANCH IF NOT DEP (ERROR)
        BR: CBN(/N') [END]

**:     CM: SET CODES, INCR IF SECOND TIME
        CC: C=0, N=1, Z=Z, V=V  BR: CJN(N) [INCR]

**:     CM: PUT ADDRESS IN AD, PUT DATA IN R15
        AR: R15B=SW1, AD=R17A   PC: LW

**:     CM: PUT ADDRESS IN ADX, PUT DATA IN R15
        AR: R15B=SW1, ADX=R17A  PC: UW

**:     CM: MOVE ADDRESS INTO D
        AR: D=R17A      BR: CBR(1) [DEP1]

**: END
        CM: RESTORE CC'S
        AR: R12B=PSR, OUT=R12A  CC: C=CPO, Z=CPO, V=CPO, N=CPO

**:     CM: WAIT FOR IO COMPLETE, LOOK AT SWITCHES, BRANCH
        AR: R15B=SW2 AND R14A   PC: LW, WIOH, CB        BR: CBR(1) [GO]

**: INCR
        CM: INCREMENT ADDRESS BY 1, SKIP NEXT IF REG
        AR: R17B=R17B+<1>       PC: 3B  BR: CBN(Z) [.+2]

**:     CM: INCREMENT ADDRESS BY 1
        AR: R17B=R17B+<1>       PC: 3B

**: TSTR
        CM: MOVE TESTF BIT INTO R13, GO TEST FOR REG
        AR: R13B=[+20]  BR: CJR(1) [REG]

**:     CM: TEST FOR UPPER/LOWER REGISTER
        AR: R13B AND R17A       BR: CRR(1)
```

```
        CC: V=PZ, C=C, Z=Z, N=N

**: REG
        CM: TEST ADX FOR REGISTER ADDRESS
        AR: R17A-[+77]-<1>      PC: UB

**:     CM: COMPARE ADDRESS TO LOW BOUND OF REG ADDRESS
        AR: R17A-[+177700]-<1>  PC: LW  BR: CBN(/Z') [NO]

**:     CM: COMPARE ADDRESS TO UPPER BOUNDS OF REG ADDRESS
        AR: [+177737]-R17A-<1>  PC: LW  BR: CBN(N') [NO]

**:     CM: REGISTER ADDRESS
        CC: Z=1, C=C, V=V, N=N  BR: CRN(/N')

**: NO
        CM: NOT REGISTER ADDRESS
        CC: Z=0, C=C, V=V, N=N  BR: CRR(1)

        PG:     DOUBLE OPERAND INSTRUCTIONS

        CM: DOUBLE OPERAND INSTRUCTIONS
            D,S,MODE IS LABLED AS INSTRUCTION
            THE THREE REMAINING MODES D,S0 D0,S & D0,S0
            ARE ACCESSED BY OFFSETS

**: MOV
        CM: MOVE INSTRUCTION
        AR: D=R10A      PC: LW, WIOL    IO: DATO(CM)
        CC: N=PN, Z=PZ, V=0, C=C        BR: CBR(1) [BGN]

**:     AR: D=RA(SRC)   PC: LW, WIOL    IO: DATO(CM)
        CC: N=PN, Z=PZ, V=0, C=C        BR: CBR(1) [BGN]

**:     AR: RB(DST)=D
        CC: Z=PZ, N=PN, V=0, C=C        BR: BROC(LKD) [BGN]

**:     AR: RB(DST)=RA(SRC)
        CC: N=PN, Z=PZ, V=0, C=C        BR: BROC(LKD) [BGN]

**: MOVB
        CM: MOVE BYTE INSTRUCTION
        AR: D=R10A      PC: LB, WIOL    IO: DATOB(CM)
        CC: N=PN, Z=PZ, V=0, C=C        BR: CBR(1) [BGN]

**:     AR: D=RA(SRC)   PC: LB, WIOL    IO: DATOB(CM)
        CC: N=PN, Z=PZ, V=0, C=C        BR: CBR(1) [BGN]

**:     AR: RB(DST)=DSX7        PC: LW
        CC: N=PN, Z=PZ, V=0, C=C        BR: BROC(LKD) [BGN]

**:     AR: D=RA(SRC)   PC: LB  BR: CBR(1) [.-1]

**: CMP
        CM: COMPARE INSTRUCTION
        AR: R10A-D-<1>  PC: LWB, WIOH
        CC: N=PN, Z=PZ, V=PV, C=PCB     BR: CBR(1) [BGN]

**:     AR: RA(SRC)-D-<1>       PC: LWB, WIOH
        CC: N=PN, Z=PZ, V=PV, C=PCB     BR: CBR(1) [BGN]

**:     AR: D-RA(DST)-<1>       PC: LWB
        CC: N=PN, Z=PZ, V=PV, C=PCB     BR: BROC(LKD) [BGN]

**:     AR: RB(SRC)-RA(DST)-<1> PC: LWB
        CC: N=PN, Z=PZ, V=PV, C=PCB     BR: BROC(LKD) [BGN]

**: BIT
        CM: BIT TEST INSTRUCTION
        AR: D AND R10A  PC: LWB, WIOH
        CC: N=PN, Z=PZ, V=0, C=C        BR: CBR(1) [BGN]

**:     AR: D AND RA(SRC)       PC: LWB, WIOH
        CC: N=PN, Z=PZ, V=0, C=C        BR: CBR(1) [BGN]

**:     AR: D AND RA(DST)       PC: LWB
```

```
        CC: N=PN, Z=PZ, V=0, C=C        BR: BROC(LKD) [BGN]

**:     AR: RB(SRC) AND RA(DST) PC: LWB
        CC: N=PN, Z=PZ, V=0, C=C        BR: BROC(LKD) [BGN]

**: BIC
        CM: BIT CLEAR INSTRUCTION
        AR: R10B=D, D=R10A      PC: LWB, WIOH
        BR: CBR(1) [.+4]

**:     AR: R10B=D, D=RA(SRC)   PC: LWB, WIOH
        BR: CBR(1) [.+3]

**:     AR: RB(DST)=/D AND RA(DST)      PC: LWB
        CC: N=PN, Z=PZ, V=0, C=C        BR: BROC(LKD) [BGN]

**:     AR: RB(DST)=/RA(SRC) AND RB(DST)        PC: LWB
        CC: N=PN, Z=PZ, V=0, C=C        BR: BROC(LKD) [BGN]

**:     AR: D=/D AND R10A       PC: LWB, WIOH   IO: DATO(CM,BYT)
        CC: N=PN, Z=PZ, V=0, C=C        BR: CBR(1) [BGN]

**: BIS
        CM: BIT SET INSTRUCTION
        AR: D=R10A OR D PC: LWB, WIOH   IO: DATO(CM,BYT)
        CC: N=PN, Z=PZ, V=0, C=C        BR: CBR(1) [BGN]

**:     AR: D=D OR RA(SRC)      PC: LWB, WIOH IO: DATO(CM,BYT)
        CC: N=PN, Z=PZ, V=0, C=C        BR: CBR(1) [BGN]

**:     AR: RB(DST)=D OR RA(DST)        PC: LWB
        CC: N=PN, Z=PZ, V=0, C=C        BR: BROC(LKD) [BGN]

**:     AR: RB(DST)=RB(DST) OR RA(SRC)  PC: LWB
        CC: N=PN, Z=PZ, V=0, C=C        BR: BROC(LKD) [BGN]

**: ADD
        CM: ADD INSTRUCTION
        AR: D=D+R10A+<0>        PC: LW, WIOH    IO: DATO(CM)
        CC: N=PN, Z=PZ, V=PV, C=PC      BR: CBR(1) [BGN]

**:     AR: D=D+RA(SRC)+<0>     PC: LW, WIOH    IO: DATO(CM)
        CC: N=PN, Z=PZ, V=PV, C=PC      BR: CBR(1) [BGN]

**:     AR: RB(DST)=D+RA(DST)+<0>
        CC: N=PN, Z=PZ, V=PV, C=PC      BR: BROC(LKD) [BGN]

**:     AR: RB(DST)=RB(DST)+RA(SRC)+<0>
        CC: N=PN, Z=PZ, V=PV, C=PC      BR: BROC(LKD) [BGN]

**: SUB
        CM: SUBTRACT INSTRUCTION
        AR: D=D-R10A-<1>        PC: LW, WIOH    IO: DATO(CM)
        CC: N=PN, Z=PZ, V=PV, C=PCB     BR: CBR(1) [BGN]

**:     AR: D=D-RA(SRC)-<1>     PC: LW, WIOH    IO: DATO(CM)
        CC: N=PN, Z=PZ, V=PV, C=PCB     BR: CBR(1) [BGN]

**:     AR: RB(DST)=RA(DST)-D-<1>
        CC: N=PN, Z=PZ, V=PV, C=PCB     BR: BROC(LKD) [BGN]

**:     AR: RB(DST)=RB(DST)-RA(SRC)-<1>
        CC: N=PN, Z=PZ, V=PV, C=PCB     BR: BROC(LKD) [BGN]

        PG:     SINGLE OPERAND INSTRUCTIONS

        CM: SINGLE OPERAND INSTRUCTIONS
            D  MODE IS LABLED AS INSTRUCTION
            D0  MODE IS ACCESSED BY AN OFFSET

**: CLR
        CM: CLEAR INSTRUCTION
        AR: D=0 PC: LWB, WIOL   IO: DATO(CM,BYT)
        CC: Z=1, N=0, V=0, C=0  BR: CBR(1) [BGN]

**:     AR: RB(DST)=0   PC: LWB
```

```
        CC: Z=1, N=0, V=0, C=0  BR: BROC(LKD) [BGN]

**: COM
        CM: COMPLEMENT INSTRUCTION
        AR: D=/D        PC: LWB, WIOH   IO: DATO(CM,BYT)
        CC: N=PN, Z=PZ, V=0, C=1        BR: CBR(1) [BGN]

**:     AR: RB(DST)=/RB(DST)    PC: LWB
        CC: N=PN, Z=PZ, V=0, C=1        BR: BROC(LKD) [BGN]

**: INC
        CM: INCREMENT INSTRUCTION
        AR: D=D+<1>     PC: LWB, WIOH   IO: DATO(CM,BYT)
        CC: N=PN, Z=PZ, V=PV, C=C       BR: CBR(1) [BGN]

**:     AR: RB(DST)=RB(DST)+<1> PC: LWB
        CC: N=PN, Z=PZ, V=PV, C=C       BR: BROC(LKD) [BGN]

**: DEC
        CM: DECREMENT INSTRUCTION
        AR: D=D-<0>     PC: LWB, WIOH   IO: DATO(CM,BYT)
        CC: N=PN, Z=PZ, V=PV, C=C       BR: CBR(1) [BGN]

**:     AR: RB(DST)=RB(DST)-<0> PC: LWB
        CC: N=PN, Z=PZ, V=PV, C=C       BR: BROC(LKD) [BGN]

**: NEG
        CM: NEGATE INSTRUCTION
        AR: D=-D-<1>    PC: LWB, WIOH   IO: DATO(CM,BYT)
        CC: N=PN, Z=PZ, V=PV, C=PZB     BR: CBR(1) [BGN]

**:     AR: RB(DST)=-RB(DST)-<1>        PC: LWB
        CC: N=PN, Z=PZ, V=PV, C=PZB     BR: BROC(LKD) [BGN]

**: ADC
        CM: ADD CARRY INSTRUCTION
        AR: D=D+<C>     PC: LWB, WIOH   IO: DATO(CM,BYT)
        CC: N=PN, Z=PZ, V=PV, C=PC      BR: CBR(1) [BGN]

**:     AR: RB(DST)=RB(DST)+<C> PC: LWB
        CC: N=PN, Z=PZ, V=PV, C=PC      BR: BROC(LKD) [BGN]

**: SBC
        CM: SUBTRACT CARRY INSTRUCTION
        AR: D=D-</C>    PC: LWB, WIOH   IO: DATO(CM,BYT)
        CC: N=PN, Z=PZ, V=PV, C=PCB     BR: CBR(1) [BGN]

**:     AR: RB(DST)=RB(DST)-</C>        PC: LWB
        CC: N=PN, Z=PZ, V=PV, C=PCB     BR: BROC(LKD) [BGN]

**: TST
        CM: TEST INSTRUCTION
        AR: AD=R7B      PC: LW, WIOL    IO: INTCK
        BR: CBR(1) [.+2]

**:     AR: RA(DST)     PC: LWB
        CC: N=PN, Z=PZ, V=0, C=0        BR: BROC(LKD) [BGN]

**:     AR: D   PC: LWB IO: IDATIR(CMI)
        CC: N=PN, Z=PZ, V=0, C=0        BR: CBR(1) [BGN+1]

**: SWAB
        CM: SWAP BYTE INSTRUCTION
        AR: D=DSWB      PC: LB, WIOH    IO: DATO(CM)
        CC: N=PN, Z=PZ, V=0, C=0        BR: CBR(1) [BGN]

**:     AR: D=RA(DST)

**:     AR: RB(DST)=DSWB

**:     AR: RA(DST)     PC: LB
        CC: N=PN, Z=PZ, V=0, C=0        BR: BROC(LKD) [BGN]

**: SXT
        CM: SIGN EXTEND INSTRUCTION
        AR: D=R0B-R0A-</N>      PC: LW, WIOL    IO: DATO(CM)
```

```
        CC: N=PN, Z=PZ, V=0, C=C        BR: CBR(1) [BGN]

**:     AR: RB(DST)=RB(DST)-RA(DST)-</N>
        CC: N=PN, Z=PZ, V=0, C=C        BR: CBR(1) [BGN+1]

**: XOR
        CM: EXCLUSIVE OR INSTRUCTION
        AR: D=D XOR RA(SRC)     PC: LW, WIOH    IO: DATO(CM)
        CC: N=PN, Z=PZ, V=0, C=C        BR: CBR(1) [BGN]

**:     AR: RB(DST)=RB(DST) XOR RA(SRC)
        CC: N=PN, Z=PZ, V=0, C=C        BR: BROC(LKD) [BGN]

**: ROR
        CM: ROTATE RIGHT INSTRUCTION
        AR: R10B=D, SRD(C)      PC: LWB, WIOH
        CC: C=LSBR, N=N, Z=Z, V=V       BR: CBR(1) [SRD]

**:     AR: RB(DST)=RB(DST), SRD(C)     PC: LWB
        CC: C=LSBR, N=N, Z=Z, V=V       BR: CBR(1) [SR0]

**: SRD
        AR: D=R10B      PC: LWB IO: DATO(CM,BYT)
        CC: C=C, N=PN, Z=PZ, V=PNXC     BR: CBR(1) [BGN]

**: SR0
        AR: RB(DST)     PC: LWB
        CC: C=C, N=PN, Z=PZ, V=PNXC     BR: BROC(LKD) [BGN]

**: ROL
        CM: ROTATE LEFT INSTRUCTION
        AR: R10B=D, SRU(C)      PC: LWB, WIOH
        CC: C=MSBR, N=N, Z=Z, V=V       BR: CBR(1) [SRD]

**:     AR: RB(DST)=RB(DST), SRU(C)     PC: LWB
        CC: C=MSBR, N=N, Z=Z, V=V       BR: CBR(1) [SR0]

**: ASR
        CM: ARITHMETIC SHIFT RIGHT INSTRUCTION
        AR: R10B=D, SRD(MSBR)   PC: LWB, WIOH
        CC: C=LSBR, N=N, Z=Z, V=V       BR: CBR(1) [SRD]

**:     AR: RB(DST)=RB(DST), SRD(MSBR)  PC: LWB
        CC: C=LSBR, N=N, Z=Z, V=V       BR: CBR(1) [SR0]

**: ASL
        CM: ARITHMETIC SHIFT LEFT INSTRUCTION
        AR: R10B=D, SRU(0)      PC: LWB, WIOH
        CC: C=MSBR, N=N, Z=Z, V=V       BR: CBR(1) [SRD]

**:     AR: RB(DST)=RB(DST), SRU(0)     PC: LWB
        CC: C=MSBR, N=N, Z=Z, V=V       BR: CBR(1) [SR0]

        PG:     CONTROL, TRAP, AND OTHER INSTRUCTIONS

**: CLRC
        CM: CLEAR CONDITION CODE INSTRUCTIONS
        CC: C=CLR, N=CLR, Z=CLR, V=CLR  BR: BROC(LKD) [BGN]

**: SETC
        CM: SET CONDITION CODE INSTRUCTIONS
        CC: C=SET, N=SET, Z=SET, V=SET  BR: BROC(LKD) [BGN]

**: SOB
        CM: SUBTRACT 1 AND BRANCH IF NOT = 0
        AR: RB(SRC)=RB(SRC)-<0>, AD=R7A

**:     AR: R10B=IR5, SRU(0)
        IO: INTCK       BR: CBN(Z') [BGN+3]

**:     AR: AD;R7B=R7B-R10A-<1>
        IO: DATIR(CMI)  BR: CBR(1) [BGN+1]

**: MARK
        CM: MARK INSTRUCTION
        AR: R10B=IR5+<1>, SRU(0)
```

```
**:     AR: AD;R10B=R6A+R10B+<0>
        IO: DATI(CMI)   BR: CBN(CUM) [.+2]

**:     AR: R6B=R10A+[+2]+<0>   BR: CBR(1) [.+2]

**:     AR: R16B=R10A+[+2]+<0>

**:     AR: AD;R7B=R5A  PC: LW, WIOL    IO: INTCK

**:     AR: R5B=D       IO: IDATIR(CMI) BR: CBR(1) [BGN+1]

**: RTS
        CM: RETURN FROM SUBRUTINE INSTRUCTION
        AR: R10B=R6A+[+2]+<0>, AD=R6A   PC: LW, WIOL
        IO: DATI(CMI)   BR: CBN(CUM) [.+2]

**:     AR: R6B=R10A    BR: CBR(1) [.+2]

**:     AR: R16B=R10A

**:     AR: R7B=RA(DST)

**:     AR: RB(DST)=D   PC: LW, WIOH    BR: CBR(1) [BGN]

**: RTI
        CM: RTI/RTT INSTRUCTIONS
        AR: R10B=R6A+[+2]+<0>,AD=R6A    PC: LW,WIOL
        IO: DATI(CMI)   BR: CBN(CUM) [.+2]

**:     AR: R6B=R6A+[+4]+<0>    BR: CBR(1) [.+2]

**:     AR: R16B=R6A+[+4]+<0>

**:     AR: R7B=D,AD=R10A       PC: LW,WIOH
        IO: DATI(CMI)

**:     CM: SET T BIT & CC'S / DONE IF IN USER MODE
        AR: PSR=D       CC: C=CPO,Z=CPO,V=CPO,N=CPO
        PC: LW,WIOH     BR: CBN(CUM) [BGN]

**:     CM: ELSE RESTORE ALL STATUS FROM STACK
        AR: AD=[+177776]        IO: DATO(OFF)
        BR: CBR(1) [BGN]

**: RESET
        CM: RESET INSTRUCTION
        PC: LW, WIOL    IO: INTCK       BR: CBN(CUM) [BGN+3]

**:     PC: LW, WIOL    IO: INIT        BR: CBR(1) [BGN+3]

**: WAIT
        CM: WAIT FOR INTERRUPT INSTRUCTION
        IO: INTCK       BR: CNR(IP)

**:     AR: R7B=R7A+[+2]+<0>    BR: CBIN(1) <INST>

**: JPR0
        CM: JMP/JSR (0) TRAP
        AR: AD;R10B=[+6]        PC: LW, WIOL
        IO: DATI(KI)    BR: CBN(1) [RES+1]

**: JSR
        CM: JUMP TO SUBRUTINE INSTRUCTION
        AR: D=RA(SRC)   BR: CBN(CUM) [.+2]

**:     AR: AD;R6B=R6A-[+2]-<1> PC: LW, WIOL
        IO: DATO(CMI)   BR: CBR(1) [.+2]

**:     AR: AD;R16B=R6A-[+2]-<1>        PC: LW, WIOL
        IO: DATO(CMI)

**:     AR: RB(SRC)=R7A

**: JMP
        CM: JUMP INSTRUCTION
```

```
        AR: AD;R7B=R11A PC: LW, WIOL
        IO: INTCK       BR: CBR(1) [BGN+3]

        PG:     RESERVED INSTRUCTION TRAP HANDLER

**: RSRVINST
        CM: GET STATUS
        AR: AD=[+177764]        IO: DATI(OFF)

**:     CM: USE STATE 'EX'/'E5' FOR ADDITIONAL CODE
        AR: SCE5=D              PC: LW,WIOH
        SP: CLR,SE5

**: RSRVSTAT
        CM: GO TO ADDITIONAL CODE IF DEFINED
        BR: CJIN(N') <SCR>

**:     CM: ELSE NOT DEFINED
        AR: SCE7=[RES]
        SP: CLR,SE7

**:     CM: RESET CPU STATE TO 'RES'/'E7'
        BR: CBIN(1) <SCR>


        PG:     SPL INSTRUCTION (11/45, 60, & 70)

**: SPL
        CM: SET PRIORITY LEVEL
        AR: R10B=R10B+R10A+<N>
        BR: CBN(CUM) [BGN]

**:     AR: R10B=R10B+R10A+<Z>

**:     AR: R10B=R10B+R10A+<V>

**:     AR: D=R10B+R10A+<C>

**:     AR: AD=[+177776]        IO: DATOB(OFF)
        BR: CBR(1) [BGN]

        PG:     PROCESOR STATUS WORD INSTRUCTIONS

**: MTPS
        CM: MOVE TO PS / DONE IF USER MODE
        AR: D           CC: C=CPO,V=CPO,Z=CPO,N=CPO
        PC: LW, WIOH    BR: CBN(CUM) [BGN]

**:     CM: ELSE IN KERNEL MODE - CHANGE PRIORITY TOO
        AR: AD=[+177776]
        IO: DATOB(OFF)  BR: CBR(1) [BGN]

**: MFPS0
        CM: MOVE FROM PS INSTRUCTION
        AR: D=PSR       BR: CBR(1) [MOVB+2]

**: MFPSX
        AR: D=PSR       CC: C=C, V=0, N=PN, Z=PZ
        PC: LB, WIOL    IO: DATOB(CM)   BR: CBR(1) [BGN]

        PG:     MEMORY MANAGEMENT INSTRUCTIONS

        .=: MORIGIN+347

**: MTPIX
        AR: SCR0=/SCR0

**:     AR: D=R10A      BR: CBN(/Z') [GTSTK]

**:     IO: IDATO(PM)

**: MEND
        AR: R10B=D, AD=R7A      CC: C=C, V=0, N=PN, Z=PZ
        IO: INTCK       BR: CBR(1) [BGN+3]      PC: LW, WIOL

**: MFPIX
```

```
        CM: MFPI INSTRUCTION
        IO: IDATI(PMI)  BR: BROC(CUM) [MFOUT]

**: MFPI0
        AR: R10B=[+106]

**:     AR: R10A-IR-<1> PC: LB  BR: CBN(PUM) [.+2]

**:     AR: D=R6B       BR: BROC(Z') [MFOUT-2]

**:     AR: D=R16B      BR: BROC(Z') [MFOUT-2]

**:     AR: D=RA(DST)   BR: BROC(CUM) [MFOUT]

**:     BR: CBN(CUM) [.+2]

**: MFOUT
        AR: AD;R6B=R6A-[+2]-<1> PC: LW, WIOL
        IO: DATO(CM)    BR: CBR(1) [MEND]

**:     AR: AD;R16B=R6A-[+2]-<1>        PC: LW, WIOL
        IO: DATO(CM)    BR: CBR(1) [MEND]

**: MTPI0
        CM: MTPI INSTRUCTION
        AR: R10B=[+206], AD=R6A IO: DATI(CM)
**:     AR: R10A-IR-<1> PC: LB  BR: CBN(CUM) [.+2]

**:     AR: R6B=R6A+[+2]+<0>    BR: BROC(Z') [MTOUT]

**:     AR: R16B=R6A+[+2]+<0>   BR: BROC(Z') [MTOUT]

**: MTOUT
        AR: RB(DST)=D   CC: C=C, V=0, N=PN, Z=PZ
        PC: LW, WIOH    BR: CBR(1) [BGN]

**:     AR: AD=R7A      PC: LW, WIOL
        IO: INTCK       BR: CBN(PUM) [.+2]

**:     AR: R6B=D       CC: C=C, V=0, N=PN, Z=PZ
        IO: IDATIR(CMI) BR: CBR(1) [BGN+1]

**:     AR: R16B=D      CC: C=C, V=0, N=PN, Z=PZ
        IO: IDATIR(CMI) BR: CBR(1) [BGN+1]

**: GTSTK
        AR: AD=R6A      PC: LW, WIOL
        IO: DATI(CMI)   BR: CBN(CUM) [.+2]

**:     AR: R6B=R6A+[+2]+<0>    SP: SE3, SE5
        BR: CBR(1) [.+2]

**:     AR: R16B=R6A+[+2]+<0>   SP: SE3, SE5

**:     AR: R10B=D, AD=R7A      PC: LW, WIOH
        BR: CBIN(1) <INST>

        NA: MORGEND=.

        PG:     INSTRUCTION CODE SPECIFICATIONS

        .=: IORIGIN+0

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES
```

```
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
```

```
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: HALT        PS: RES
**:     MA: WAIT        PS: EX
**:     MA: RTI PS: EX
**:     MA: RES PS: BPT IN: TI  CM: BPT INSTRUCTION
**:     MA: RES PS: IOT IN: TI  CM: IOT INSTRUCTION
**:     MA:  RESET      PS: EX
**:     MA: RTI PS: EX  IN: TI  CM: RTT INSTRUCTION
**:     MA: RSRVINST    PS: RES
**:     MA: MOV         PS: SRC, DST, EX        IN: NDA
**:     MA: MOV+1       PS: DST, EX     IN: NDA
**:     MA: MOV+2       PS: SRC, EX     IN: C2, WT
**:     MA: MOV+3       PS: EX  IN: C1, WT
**:     MA: MOVB        PS: SRC, DST, EX        IN: NDA, BYT
**:     MA: MOVB+1      PS: DST, EX     IN: NDA, BYT
**:     MA: MOVB+2      PS: SRC, EX     IN: C2, BYT, WT
**:     MA: MOVB+3      PS: EX  IN: C1, BYT, WT
**:     MA: CMP         PS: SRC, DST, EX        IN: SP
**:     MA: CMP+1       PS: DST, EX     IN: SP
**:     MA: CMP+2       PS: SRC, EX     IN: C2, WT
```

```
**:     MA: CMP+3       PS: EX  IN: FRC, WT
**:     MA: CMP         PS: SRC, DST, EX        IN: BYT, SP
**:     MA: CMP+1       PS: DST, EX     IN: BYT, SP
**:     MA: CMP+2       PS: SRC, EX     IN: C2, BYT, WT
**:     MA: CMP+3       PS: EX  IN: FRC, BYT, WT
**:     MA: BIT         PS: SRC, DST, EX        IN: SP
**:     MA: BIT+1       PS: DST, EX     IN: SP
**:     MA: BIT+2       PS: SRC, EX     IN: C2, WT
**:     MA: BIT+3       PS: EX  IN: FRC, WT
**:     MA: BIT         PS: SRC, DST, EX        IN: SP, BYT
**:     MA: BIT+1       PS: DST, EX     IN: SP, BYT
**:     MA: BIT+2       PS: SRC, EX     IN: C2, BYT, WT
**:     MA: BIT+3       PS: EX  IN: FRC, BYT, WT
**:     MA: BIC         PS: SRC, DST, EX
**:     MA: BIC+1       PS: DST, EX
**:     MA: BIC+2       PS: SRC, EX     IN: C2, WT
**:     MA: BIC+3       PS: EX  IN: C1, WT
**:     MA: BIC         PS: SRC, DST, EX        IN: BYT
**:     MA: BIC+1       PS: DST, EX     IN: BYT
**:     MA: BIC+2       PS: SRC, EX     IN: C2, BYT, WT
**:     MA: BIC+3       PS: EX  IN: C1, BYT, WT
**:     MA: BIS         PS: SRC, DST, EX
**:     MA: BIS+1       PS: DST, EX
**:     MA: BIS+2       PS: SRC, EX     IN: C2, WT
**:     MA: BIS+3       PS: EX  IN: C1, WT
**:     MA: BIS         PS: SRC, DST, EX        IN: BYT
**:     MA: BIS+1       PS: DST, EX     IN: BYT
**:     MA: BIS+2       PS: SRC, EX     IN: C2, BYT, WT
**:     MA: BIS+3       PS: EX  IN: C1, BYT, WT
**:     MA: ADD         PS: SRC, DST, EX
**:     MA: ADD+1       PS: DST, EX
**:     MA: ADD+2       PS: SRC, EX     IN: C2, WT
**:     MA: ADD+3       PS: EX  IN: C1,WT
**:     MA: SUB         PS: SRC, DST, EX
**:     MA: SUB+1       PS: DST, EX
**:     MA: SUB+2       PS: SRC, EX   IN: C2, WT
**:     MA: SUB+3       PS: EX  IN: C1, WT
**:     MA: RTS         PS: EX
```

```
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: SPL         PS: EX
**:     MA: CLRC        PS: EX  IN: FRC, WT
**:     MA: CLRC        PS: EX  IN: FRC, WT
**:     MA: SETC        PS: EX  IN: FRC, WT
**:     MA: SETC        PS: EX  IN: FRC, WT
**:     MA: CLR         PS: DST,EX      IN: NDA
**:     MA: CLR+1       PS: EX  IN: C1, WT
**:     MA: CLR         PS: DST, EX     IN: BYT, NDA
**:     MA: CLR+1       PS: EX  IN: BYT, C1, WT
**:     MA: COM         PS: DST, EX
**:     MA: COM+1       PS: EX  IN: C1, WT
**:     MA: COM         PS: DST, EX     IN: BYT
**:     MA: COM+1       PS: EX  IN: BYT, C1, WT
**:     MA: INC         PS: DST, EX
**:     MA: INC+1       PS: EX  IN: C1, WT
**:     MA: INC         PS: DST, EX     IN: BYT
**:     MA: INC+1       PS: EX  IN: BYT, C1, WT
**:     MA: DEC         PS: DST, EX
**:     MA: DEC+1       PS: EX  IN: C1, WT
**:     MA: DEC         PS: DST, EX     IN: BYT
**:     MA: DEC+1       PS: EX  IN: BYT, C1, WT
**:     MA: NEG         PS: DST, EX
**:     MA: NEG+1       PS: EX  IN: C1, WT
**:     MA: NEG         PS: DST, EX     IN: BYT
**:     MA: NEG+1       PS: EX  IN: BYT, C1, WT
**:     MA: ADC         PS: DST, EX
**:     MA: ADC+1       PS: EX  IN: C1, WT
**:     MA: ADC         PS: DST, EX     IN: BYT
**:     MA: ADC+1       PS: EX  IN: BYT, C1, WT
**:     MA: SBC         PS: DST, EX
**:     MA: SBC+1       PS: EX  IN: C1, WT
**:     MA: SBC         PS: DST, EX     IN: BYT
**:     MA: SBC+1       PS: EX  IN: BYT, C1, WT
**:     MA: TST         PS: DST, EX     IN: SP
**:     MA: TST+1       PS: EX  IN: FRC, WT
**:     MA: TST         PS: DST, EX     IN: SP, BYT
```

```
**:     MA: TST+1       PS: EX  IN: BYT, FRC, WT
**:     MA: ROR         PS: DST, EX
**:     MA: ROR+1       PS: EX  IN: C1, WT
**:     MA: ROR         PS: DST, EX     IN: BYT
**:     MA: ROR+1       PS: EX  IN: BYT, C1, WT
**:     MA: ROL         PS: DST, EX
**:     MA: ROL+1       PS: EX  IN: C1, WT
**:     MA: ROL         PS: DST, EX     IN: BYT
**:     MA: ROL+1       PS: EX  IN: BYT, C1, WT
**:     MA: ASR         PS: DST, EX
**:     MA: ASR+1       PS: EX  IN: C1, WT
**:     MA: ASR         PS: DST, EX     IN: BYT
**:     MA: ASR+1       PS: EX  IN: BYT, C1, WT
**:     MA: ASL         PS: DST, EX
**:     MA: ASL+1       PS: EX  IN: C1, WT
**:     MA: ASL         PS: DST, EX     IN: BYT
**:     MA: ASL+1       PS: EX  IN: BYT, C1, WT
**:     MA: MARK        PS: EX
**:     MA: MARK        PS: EX
**:     MA: MTPS        PS: DST, EX     IN: SP, BYT
**:     MA: MTPS        PS: DST, EX     IN: BYT
**:     MA: MFPIX       PS: DST, EX     IN: NDA
**:     MA: MFPI0       PS: EX
**:     MA: MFPIX       PS: DST, EX     IN: NDA CM: MFPD INST.
**:     MA: MFPI0       PS: EX  CM: MFPD INST.
**:     MA: MTPIX       PS: EX  IN: NDA
**:     MA: MTPI0       PS: EX
**:     MA: MTPIX       PS: EX  IN: NDA CM: MTPD INST.
**:     MA: MTPI0       PS: EX  CM: MTPD INST.
**:     MA: SXT         PS: DST, EX     IN: NDA
**:     MA: SXT+1       PS: EX          IN: C1, WT
**:     MA: MFPSX       PS: DST, EX     IN: NDA
**:     MA: MFPS0       PS: EX
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
```

```
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: XOR         PS: DST, EX
**:     MA: XOR+1       PS: EX  IN: C1, WT
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: SOB PS: EX
**:     MA: SOB PS: EX
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
**:     MA: RSRVINST    PS: RES
```

```
**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: JMP         PS: DST, EX     IN: NDA

**:     MA: JPR0        PS: RES

**:     MA: SWAB        PS: DST, EX

**:     MA: SWAB+1      PS: EX  IN: C1, WT

**:     MA: NOBR        PS: EX  IN: FRC, WT

**:     MA: BR          PS: EX

**:     MA: JSR         PS: DST, EX     IN: NDA

**:     MA: JPR0        PS: RES

**:     MA: RES PS: EMT         IN: TI  CM: EMT INSTRUCTION

**:     MA: RES PS: TRAP        IN: TI  CM: TRAP INSTRUCTION

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

**:     MA: RSRVINST    PS: RES

        PG:     VECTOR DEFINITIONS

        VE: VGRP=+0
        VE: V37= CHLT / +37
        VE: V36= CHLT / +36
        VE: V35= CHLT / +35
        VE: V34= FBHINT  / +6
        VE: V33= CHLT / +33
        VE: V32= YSTB / +6
        VE: V31= FBHINT  / +6
        VE: V30= FBHINT  / +252
        VE: V27= CHLT / +0
        VE: V26= CHLT / +0
        VE: V25= CHLT / +0
        VE: V24= CHLT / +0
        VE: V23= CHLT / +0
        VE: V22= CHLT / +0
        VE: V21= CHLT / +0
        VE: V20= RES  / +12
        VE: V17= RES  / +22
        VE: V16= RES  / +16
        VE: V15= RES  / +32
        VE: V14= RES  / +36
        VE: V13= CHLT / +13
        VE: V12= YSTB   / +16
        VE: V11= YSTB   / +6
        VE: V10= PF     / +26
        VE: V7 = INT17 / +102
        VE: V6 = INT17 / +62
        VE: V5 = INT17 / +66
        VE: V4 = INT17 / +172
        VE: V3 = INT17 / +176
        VE: V2 = INT17 / +272
```

```
        VE: V1 = INT17 / +276
        VE: V0 = INT0  / +242
```

```
        PG:     DIAGNOSTIC HALTS

        CM: MUST BE USED INPLACE OF ENDX.MIC
        .=: +4000

**: DIAG        BR: CBR(1) [INT0]


**:     BR: CBR(1) [INT17]


**:     BR: CBR(1) [INT17]

**:     BR: CBR(1) [INT17]

**:     BR: CBR(1) [INT17]

**:     BR: CBR(1) [INT17]

**:     BR: CBR(1) [INT17]

**:     BR: CBR(1) [INT17]

**:     PC: HALT        BR: CBR(1) [PF]

**:     PC: HALT        BR: CBR(1) [YSTB]

**:     PC: HALT        BR: CBR(1) [YSTB]

**:     PC: HALT        BR: CBR(1) [CHLT]

**:     BR: CBR(1) [RES]

**:     BR: CBR(1) [RES]

**:     BR: CBR(1) [RES]

**:     BR: CBR(1) [RES]

**:     PC: HALT        BR: CBR(1) [RES]

**:     BR: CBR(1) [CHLT]

**:     BR: CBR(1) [CHLT]

**:     BR: CBR(1) [CHLT]

**:     BR: CBR(1) [CHLT]

**:     BR: CBR(1) [CHLT]

**:     BR: CBR(1) [CHLT]

**:     BR: CBR(1) [CHLT]

**:     PC: HALT        BR: CBR(1) [FBHINT]

**:     PC: HALT        BR: CBR(1) [FBHINT]

**:     PC: HALT        BR: CBR(1) [YSTB]

**:     PC: HALT        BR: CBR(1) [CHLT]

**:     PC: HALT        BR: CBR(1) [FBHINT]

**:     PC: HALT        BR: CBR(1) [CHLT]

**:     PC: HALT        BR: CBR(1) [CHLT]

**:     PC: HALT        BR: CBR(1) [CHLT]

        PG:     DIAGNOSTIC VECTOR GROUP

        VE: VGRP=+1

        VE: V0= DIAG+0 / +242
```

```
        VE: V1= DIAG+1 / +276
        VE: V2= DIAG+2 / +272
        VE: V3= DIAG+3 / +176
        VE: V4= DIAG+4 / +172
        VE: V5= DIAG+5 / +66
        VE: V6= DIAG+6 / +62
        VE: V7= DIAG+7 / +102
        VE: V10= DIAG+10 / +26
        VE: V11= DIAG+11 / +6
        VE: V12= DIAG+12 / +16
        VE: V13= DIAG+13 / +13
        VE: V14= DIAG+14 / +36
        VE: V15= DIAG+15 / +32
        VE: V16= DIAG+16 / +16
        VE: V17= DIAG+17 / +22
        VE: V20= DIAG+20 / +12
        VE: V21= DIAG+21 / +0
        VE: V22= DIAG+22 / +0
        VE: V23= DIAG+23 / +0
        VE: V24= DIAG+24 / +0
        VE: V25= DIAG+25 / +0
        VE: V26= DIAG+26 / +0
        VE: V27= DIAG+27 / +0
        VE: V30= DIAG+30 / +252
        VE: V31= DIAG+31 / +6
        VE: V32= DIAG+32 / +6
        VE: V33= DIAG+33 / +33
        VE: V34= DIAG+34 / +6
        VE: V35= DIAG+35 / +35
        VE: V36= DIAG+36 / +36
        VE: V37= DIAG+37 / +37

        EN::
```

```
        PG: DIAGNOSTIC DUMPER

        CM: THIS CODE DUMPS THE FOLLOWING DATA
            BEFORE EACH INSTRUCTION IS EXECUTED

                R0              MMUSR0
                R1              MMUSR1
                R2              MMUSR2
                R3              177770
                R4              177772
                R5              177774
                R6              R16 (USER R6)
                R7              PSW

        .=: MORGEND

**: DIAGNOS
**:     AR: AD=R7B      PC: WIOL
**:     IO: ILDATIR(CMI)
        BR: CBIN(1) <INST>

        AR: AD=[+0]
**:     AR: ADX=[+5]    IO: DATI(OFF)
**:     AR: R15B=D                      PC: LW,WIOH
**:     AR: R15B=R15A AND [+177740]     PC: LW

**:     AR: R15B=R15A+[+2]+<0>, AD=R15A PC: LW,WIOL
**:     AR: ADX=[+4]                    PC: LW
**:     AR: D=R0A       IO: DATO(OFF)   PC: LW

**:     AR: R15B=R15A+[+2]+<0>, AD=R15A PC: LW,WIOL
**:     AR: ADX=[+4]                    PC: LW
**:     AR: D=R1B       IO: DATO(OFF)   PC: LW

**:     AR: R15B=R15A+[+2]+<0>, AD=R15A PC: LW,WIOL
**:     AR: ADX=[+4]                    PC: LW
**:     AR: D=R2B       IO: DATO(OFF)   PC: LW

**:     AR: R15B=R15A+[+2]+<0>, AD=R15A PC: LW,WIOL
**:     AR: ADX=[+4]                    PC: LW
**:     AR: D=R3B       IO: DATO(OFF)   PC: LW

**:     AR: R15B=R15A+[+2]+<0>, AD=R15A PC: LW,WIOL
**:     AR: ADX=[+4]                    PC: LW
**:     AR: D=R4B       IO: DATO(OFF)   PC: LW

**:     AR: R15B=R15A+[+2]+<0>, AD=R15A PC: LW,WIOL
**:     AR: ADX=[+4]                    PC: LW
**:     AR: D=R5B       IO: DATO(OFF)   PC: LW

**:     AR: R15B=R15A+[+2]+<0>, AD=R15A PC: LW,WIOL
**:     AR: ADX=[+4]                    PC: LW
**:     AR: D=R6B       IO: DATO(OFF)   PC: LW

**:     AR: R15B=R15A+[+2]+<0>, AD=R15A PC: LW,WIOL
**:     AR: ADX=[+4]                    PC: LW
**:     AR: D=R7B       IO: DATO(OFF)   PC: LW

**:     AR: AD=[+177572]                PC: WIOL
        IO: DATI(OFF)
**:     AR: R15B=R15A+[+2]+<0>, AD=R15A PC: LW,WIOL
**:     AR: ADX=[+4]    IO: DATO(OFF)   PC: LW

**:     AR: AD=[+177574]                PC: WIOL
        IO: DATI(OFF)
**:     AR: R15B=R15A+[+2]+<0>, AD=R15A PC: LW,WIOL
**:     AR: ADX=[+4]    IO: DATO(OFF)   PC: LW

**:     AR: AD=[+177576]                PC: WIOL
        IO: DATI(OFF)
**:     AR: R15B=R15A+[+2]+<0>, AD=R15A PC: LW,WIOL
**:     AR: ADX=[+4]    IO: DATO(OFF)   PC: LW

**:     AR: AD=[+177770]                PC: WIOL
        IO: DATI(OFF)
**:     AR: R15B=R15A+[+2]+<0>, AD=R15A PC: LW,WIOL
```

```
**:     AR: ADX=[+4]    IO: DATO(OFF)   PC: LW

**:     AR: AD=[+177772]                PC: WIOL
        IO: DATI(OFF)
**:     AR: R15B=R15A+[+2]+<0>, AD=R15A PC: LW,WIOL
**:     AR: ADX=[+4]    IO: DATO(OFF)   PC: LW

**:     AR: AD=[+177774]                PC: WIOL
        IO: DATI(OFF)
**:     AR: R15B=R15A+[+2]+<0>, AD=R15A PC: LW,WIOL
**:     AR: ADX=[+4]    IO: DATO(OFF)   PC: LW

**:     AR: R15B=R15A+[+2]+<0>, AD=R15A PC: LW,WIOL
**:     AR: ADX=[+4]                    PC: LW
**:     AR: D=R16B      IO: DATO(OFF)   PC: LW

**:     AR: AD=[+177776]                PC: WIOL
        IO: DATI(OFF)
**:     AR: R15B=R15A+[+2]+<0>, AD=R15A PC: LW,WIOL
**:     AR: ADX=[+4]    IO: DATO(OFF)   PC: LW

**:     AR: AD=[+0]     PC: WIOL
**:     AR: ADX=[+5]
**:     AR: D=R15B      IO: DATO(OFF)   PC: LW

        NA: MORGEND=.
```

```
        TT:     EXTENDED INSTRUCTION MICROCODE

        SB:     TITLE PAGE

        CM:     EXTENSION MICROCODE FOR THE ARB-11 CPU
                IS INCLUDED IN THIS SECTION.

        CM:     THE MICROCODE DISPATCHER IS LOCATED AT
                ADDRESS 4000 OF THE MICROCODE WCS

        CM:     THE MICROJUMP REGISTER @17777764 MUST BE LOADED
                WITH THIS ADDRESS AND BIT <15> SET TO 1
                TO ENABLE THE EXTENDED INSTRUCTIONS

        CM:     PROGRAM MICROCODE ORIGIN
        NA:             MORIGIN = +4000
        CM:     ARB-11 MADHOUSE ENTRY
        NA:             BGN     = +6070

        PG:     INSTRUCTION DISPATCHER

        .=: MORIGIN+0

**: DISPATCH
        CM: LOAD R10 WITH INSTRUCTION CODE
        AR: R10B=IR

**:     CM: COMPARE TO XM CODE (=77)
        AR: R10A-[+77]-<1>

**:     CM: ON XM CODE GOTO XM
        BR: CBN(Z') [XM]

**:     CM: END OF LIST - RETURN TO RSRVSTAT FOR ERROR PROCESSING
        BR: CRR(1)

        PG:     XM INSTRUCTION


        CM:     MICROCODE IMPLEMENTATION OF FAST MEMORY TRANSFERS
                CODE DATED NOVEMBER 1982
                ALAN R. BALDWIN

        CM:     XM INSTRUCTION CHANGED FROM 7 TO 77
                ON 7-MAR-1984. CODE 7 ASSIGNED THE MFPT
                INSTRUCTION (PDP 11/44) BY DEC.

        CM:     REGISTER USAGE -

                        R0 -    6-BIT ADDRESS EXTENSION (SOURCE)
                        R1 -    16-BIT ADDRESS (SOURCE)

                        R2 -    6-BIT ADDRESS EXTENSION (DESTINATION)
                        R3 -    16-BIT ADDRESS (DESTINATION)

                        R4 -    16-BIT TRANSFER WORD COUNT

        CM:     INSTRUCTION MAY BE SUSPENDED BY A PENDING INTERRUPT
                AND RE-STARTED (CONTINUED)

        CM:     AT INSTRUCTION COMPLETION -
                        <R0/R1> = LAST SRC ADDRESS +2
                        <R2/R3> = LAST DST ADDRESS +2
                          R4    = 0


**: XM
        CM: TEST FOR WORD COUNT=0
        AR: R4B=R4A

**:     CM: EXIT ROUTINE IF COUNT=0
        BR: CBN(Z') [BGN]

**:     CM: LOAD SOURCE ADDRESS AND UPDATE (WORDS)
        AR: R1B=R1A+[+2]+<0>, AD=R1A
```

```
**: XMLOOP
        CM: LOAD SOURCE ADDRESS EXTENSION AND UPDATE
        AR: R0B=R0A+<C'>, ADX=R0A       IO: DATI(OFF)

**:     CM: UPDATE WORD COUNTER
        AR: R4B=R4A-<0>                 CC: Z=PZ

**:     CM: LOAD DESTINATION ADDRESS AND UPDATE (WORDS)
        AR: R3B=R3A+[+2]+<0>, AD=R3A    PC: LW, WIOL
        IO: INTCK

**:     CM: LOAD DESTINATION ADDRESS EXTENSION AND UPDATE
        AR: R2B=R2A+<C'>, ADX=R2A       IO: DATO(OFF)
        BR: CBN(Z) [BGN]

**:     CM: LOAD SOURCE ADDRESS AND UPDATE (WORDS)
        AR: R1B=R1A+[+2]+<0>, AD=R1A    PC: LW, WIOL
        BR: CBN(/IP) [XMLOOP]

**:     CM: RESTORE R1 FOR RE-ENTRY AFTER INTERRUPT
        AR: R1B=R1A-[+2]-<1>            BR: CBIN(1) <INST>


        EN::
```

```
        PG:     SPECIAL MEMORY ACCESS INSTRUCTIONS

        .=: MORGEND
        CM: SPECIAL MEMORY ACCESS INSTRUCTIONS
                STORD INSTRUCTION  0070[SRC] WORD OPERATION
                STORD INSTRUCTION  1070[SRC] BYTE OPERATION
            MOVES DATA FROM SOURCE TO THE ABSOLUTE ADDRESS
            IN THE TOP TWO WORDS OF THE STACK
                LOADD INSTRUCTION  0071[DST] WORD OPERATION
                LOADD INSTRUCTION  1071[DST] BYTE OPERATION
            MOVES DATA TO THE DESTINATION FROM THE ABSOLUTE
            ADDRESS IN THE TOP TWO WORDS OF THE STACK
            (R6)        CONTAINS LOW ORDER 16 BITS OF ADDRESS
            2(R6)  CONTAINS HIGH ORDER 6 BITS OF ADDRESS
                JUMPM INSTRUCTION  0072[SRC] WORD OPERATION
                JUMPM INSTRUCTION  1072[SRC] BYTE OPERATION
            BRANCHES TO THE MICROPROGRAM ADDRESS CONTAINED
            IN THE SOURCE OPERAND
            FOR WORD OPERATIONS A 12 BIT BRANCH ADDRESS IS USED
            FOR BYTE OPERATIONS AN 8 BIT BRANCH ADDRESS IS USED

**: STORD
        CM: STORD INSTRUCTION
        AR: R10B=D      PC: LW, WIOH
        BR: CJR(1) [ABAD]

**:     AR: D=R10B      PC: LWB CC: C=C, V=0, N=PN, Z=PZ
        IO: DATO(OFF,BYT)       BR: CBR(1) [BGN]

**: ABAD
        CM: GET ABSOLUTE ADDRESS FROM STACK
        AR: R12B=R6A+[+2]+<0>,AD=R6A    PC: LW, WIOL
        IO: DATI(CMI)

**:     AR: R12B=D,AD=R12A      PC: LW, WIOH
        IO: DATI(CMI)

**:     AR: AD=R12B     PC: LW, WIOL

**:     AR: ADX=D       BR: CRR(1)

**: LOADD
        CM: LOADD INSTRUCTION
        CM: A BYTE OPERATION ?
        AR: R10B=IR     CC: C=C, V=V, N=PN, Z=Z
        BR: CJR(1) [ABAD]

**:     CM: A MODE 0 OPERATION ?
        AR: R10B=R10A AND [+70]
        IO: IDATI(OFF,BYT)

**:     CM: BRANCH ON MODE ZERO OPERATION
        AR: AD=R11A     PC: LW, WIOL
        BR: CBN(Z') [MD0]

**:     CM: STORE DATA
        AR: D   CC: C=C, V=0, N=PN, Z=PZ        PC: LWB
        IO: IDATO(CM,BYT)       BR: CBR(1) [BGN]

**: MD0
        CM: MODE ZERO; FALL THROUGH ON BYTE OPERATION
        AR: RB(DST)=D   CC: C=C, V=0, N=PN, Z=PZ
        PC: LW  BR: CBN(/N) [BGN]

**:     CM: SIGN EXTEND BYTE
        AR: RB(DST)=DSX7        CC: C=C, V=0, N=PN, Z=PZ
        PC: LW  BR: CBR(1) [BGN]

**: JUMPM
        CM: JUMP TO MICRO ADDRESS INSTRUCION
        AR: SCR6=D      PC: LW, WIOH    SP: CLR, ST6

**:     CM: JUMP TO MICROADDRESS
        BR: CBIN(1) <SCR>

**:     CM: ENTRY POINT FOR BYTE OPERATIONS
```

```
        AR: R10B=[+377] SP: CLR, ST6

**:     AR: SCR6=R10A AND D  PC: WIOH

**:     BR: CBIN(1) <SCR>

        NA: MORGEND=.

        PG:     SPECIAL INSTRUCTION CODES

        .=: IORIGIN+320

**:     MA: STORD       PS: DST, EX     IN: SP

**:     MA: LOADD       PS: DST, EX     IN: NDA

**:     MA: JUMPM       PS: DST, EX     IN: SP

        .=: IORIGIN+330

**:     MA: STORD       PS: DST, EX     IN: SP, BYT

**:     MA: LOADD       PS: DST, EX     IN: NDA, BYT

**:     MA: JUMPM+2     PS: DST, EX     IN: SP, BYT
```