

**PDP-11/45 maintenance  
reference manual**

**pdp11**

**digital**

**PDP-11/45 maintenance  
reference manual**

1st Edition November 1972

Copyright © 1972 by Digital Equipment Corporation

The material in this manual is for informational purposes and is subject to change without notice.

The following are trademarks of Digital Equipment Corporation, Maynard, Massachusetts:

DEC	PDP
FLIP CHIP	FOCAL
DIGITAL	COMPUTER LAB

## CONTENTS

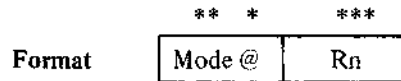
	<b>Pages</b>
CP INSTRUCTION SET . . . . .	1–12
MEMORY MANAGEMENT . . . . .	13–15
SEMICONDUCTOR MEMORY . . . . .	16–23
CORE MEMORY . . . . .	24–25
FLOATING POINT PROCESSOR . . . . .	26–48
OP CODE DETERMINATION . . . . .	49
MEMORY MAP AND PROGRAM LOADERS . . . . .	50–53
ADDRESS MODES . . . . .	54–56
CONSOLE . . . . .	57
KB11-A BLOCK DIAGRAM . . . . .	58–59
MODULE LOCATIONS . . . . .	60
DEVICE REGISTER ADDRESSES . . . . .	61
ASCII CODE . . . . .	62



### INSTRUCTION CARD LEGEND

	OP Fields	Time
n	Byte(1)/Word(0)	Add 150 ns if Destination is an odd byte, except where dst = R7 or where dst mode equals 0 and src, where applicable, is 0.
src	Source Field – 6 Bits	
dst	Destination Field – 6 Bits	Add 90 ns/memory reference if memory management KT11 is in operation.
R	Register – 3 Bits	
FS	Floating Source – 6 Bits	
FD	Floating Destination – 6 Bits	
AC	Floating Accumulator – 2 Bits	
XXX	Offset – 8 Bits	
YY	Offset – 6 Bits	
NN	Count – 6 Bits	
N	Count – 3 Bits	
^	AND	
∨	Inclusive OR	<h4 style="text-align: center;">Condition Codes</h4> <ul style="list-style-type: none"> <li>* Conditionally Set</li> <li>· Not affected</li> <li>0 Cleared</li> <li>1 Set</li> </ul>
⊖	Exclusive OR	
( )	Contents of	
loc	Location	
←	Becomes	
↑	Is Popped from Stack	
↓	Is Pushed onto Stack	
~	Boolean Not	

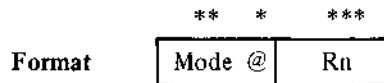
## GENERAL ADDRESSING MODES



- \*Direct/deferred bit for source and destination address
- \*\*Specifies how selected registers are to be used
- \*\*\*Specifies a general register

Mode	Name	Symbol	Function
0	Register	%R	Register contains operand.
1	Register Deferred	@%R or (R)	Register contains the address of the operand.
2	Auto-increment	(R)+	Register contains the address of the operand. Register contents incremented after reference.
3	Auto-increment Deferred	@(R)+	Register is first used as a pointer to a word containing the address of the operand, then incremented (always by 2, even for byte instructions).
4	Auto-decrement	-(R)	Register contents decremented before reference. Register contains the address of the operand.
5	Auto-decrement Deferred	@-(R)	Register is decremented (always by 2, even for byte instructions), then used as a pointer to a word containing the address of the operand.
6	Index	±X(R)	Value X (stored in a word following the instruction) is added to (R) to produce the address of the operand. Neither X nor (R) is modified.
7	Index Deferred	@±X(R) or @(R)	Value X (stored in a word following the instruction) and (R) are added and the sum is used as a pointer to a word containing the address of the operand. Neither X nor (R) is modified.
(±X is an Index value)			

## SPECIAL (PC) ADDRESSING MODES



- \*Direct/deferred bit for source and destination address
- \*\*Specifies how selected register is to be used
- \*\*\*Specifies register 7 (PC)

Mode	Name	Symbol	Function
2	Immediate	#n	Operand follows instruction.
3	Absolute	@#A	A follows instruction is the address of the operand. (A = absolute address.)
6	Relative	A	A is the address of the operand. (A = Index value following the instruction plus updated PC.)
7	Relative Deferred	@A	A is the address of a word containing the address of the operand. (A = Index value following the instruction plus updated PC.)

### BRANCH ADDRESSING

$$\text{OFFSET} = \frac{(\text{effective address}) - (\text{updated PC})}{2}$$

$$\text{EFFECTIVE ADDRESS} = (\text{offset} \times \text{two}) + (\text{updated PC})$$

Branching from location 500		
PC	Instruction	OFFSET
470		373
472		374
474		375
476		376
500	Instruction	377
502		000
504		001
506		002
510		003

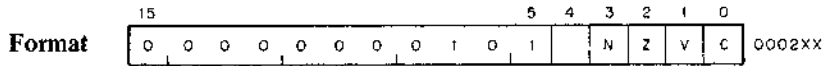
OFFSET – Number of words to branch from updated PC.

EFFECTIVE ADDRESS – The location to branch too.

UPDATED PC – Location of instruction plus two.



**CONDITION CODES OPERATORS: OPR**



11-1463

Condition code operator set or clear condition code bits. Indicated bits of the instruction word (3-0) if = to a ONE, affect the indicated condition code bits NZVC according to bit 4.

Bit 4 = 0 Clear condition code bits  
 Bit 4 = 1 Set condition code bits

Mnemonic	Instruction/Operation	OP Code	Condition Code NZVC	Time
—	No Operation	000240	----	600 ns
CLC	CLear C C ← 0	000241	---0	600 ns
CLV	CLear V V ← 0	000242	--0-	600 ns
CLZ	CLear Z Z ← 0	000244	-0--	600 ns
CLN	CLear N N ← 0	000250	0---	600 ns
CCC	Clear all CC's N, Z, V, C ← 0	000257	0000	600 ns
—	No Operation	000260	----	600 ns
SEC	SEt C C ← 1	000261	---1	600 ns
SEV	SEt V V ← 1	000262	--1-	600 ns
SEZ	SEt Z Z ← 1	000264	-1--	600 ns
SEN	SEt N N ← 1	000270	1---	600 ns
SCC	Set all CC's N, Z, V, C ← 1	000277	1111	600 ns
	Combinations of the above Clear instructions can be ORed together to form combined Clear instructions.			
—	Clear V and C V, C ← 0	000243	--00	600 ns
	Combinations of the above Set instructions can be ORed together to form combined Set instructions.			
—	Set N and V N, V ← 1	000272	1-1-	600 ns

### CONDITIONAL BRANCHES: OPR loc



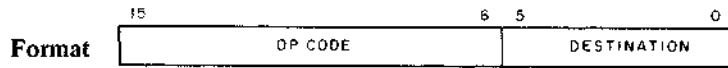
11-1962

The instruction causes a branch to a location defined by the sum of the offset (multiplied by 2) and the current contents of the program counter, if conditions are met.

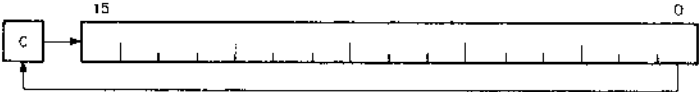
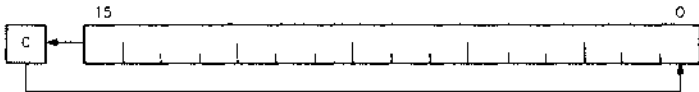
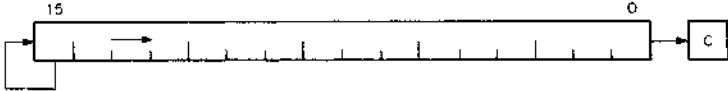
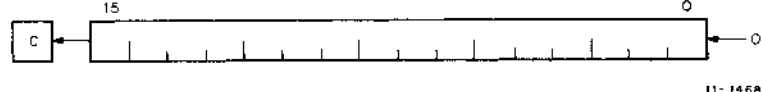
TIME: Branch 600 ns/No Branch 300 ns

Mnemonic	Instruction/Operation	OP Code
BR	BRanch (unconditionally) PC ← loc	0004+XXX
BNE	Branch if Not Equal (zero) PC ← loc if Z=0	0010+XXX
BEQ	Branch if Equal (zero) PC ← loc if Z=1	0014+XXX
BGE	Branch if Greater or Equal (zero) PC ← loc if N=V	0020+XXX
BLT	Branch if Less Than (zero) PC ← loc if N≠V	0024+XXX
BGT	Branch if Greater Than (zero) PC ← loc if Z=0 and N=V	0030+XXX
BLE	Branch if Less than or Equal (zero) PC ← loc if Z=1 or N≠V	0034+XXX
BPL	Branch if PLus PC ← loc if N=0	1000+XXX
BMI	Branch if MINus PC ← loc if N=1	1004+XXX
BHI	Branch if HIGher PC ← loc if C=0 and Z=0	1010+XXX
BLOS	Branch if LOWer or Same PC ← loc if C=1 or Z=1	1014+XXX
BVC	Branch if oVerflow Clear PC ← loc if V=0	1020+XXX
BVS	Branch if oVerflow Set PC ← loc if V=1	1024+XXX
BCC	Branch if Carry Clear PC ← loc if C=0	1030+XXX
BHIS	Branch if HIGher or Same PC ← loc if C=0	1030+XXX
BCS	Branch if Carry Set PC ← loc if C=1	1034+XXX
BLO	Branch if LOWer PC ← loc if C=1	1034+XXX

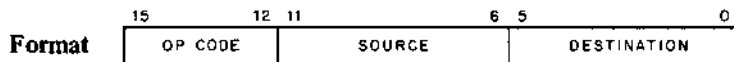
**SINGLE OPERAND INSTRUCTIONS: OPR dst**



11-1464

Mnemonic	Instruction/Operation	OP Code	Condition Code NZVC	Time
CLR(B)	CLear (Byte) (dst) ← 0	n050DD	0100	300 ns
COM(B)	COMplement (Byte) (dst) ← ~(dst)	n051DD	**01	300 ns
INC(B)	INCrement (Byte) (dst) ← (dst) + 1	n052DD	***.	300 ns
DEC(B)	DECrement (Byte) (dst) ← (dst) - 1	n053DD	***.	300 ns
NEG(B)	NEGate (Byte) (dst) ← ~(dst) + 1	n054DD	****	300 ns
ADC(B)	ADd Carry (Byte) (dst) ← (dst) + (c)	n055DD	****	300 ns
SBC(B)	SuBtract Carry (Byte) (dst) ← (dst) - (c)	n056DD	****	300 ns
TST(B)	TeST (Byte) (dst) ← (dst)	n057DD	**00	300 ns
ROR(B)	ROtate Right (Byte) (dst) ← (dst) 1 place right with (c)	n060DD	****	300 ns
				
ROL(B)	ROtate Left (Byte) (dst) ← (dst) 1 place left with (c)	n061DD	****	300 ns
				
ASR(B)	Arithmetic Shift Right (Byte) (dst) ← (dst) shifted 1 place right	n062DD	****	300 ns
				
ASL(B)	Arithmetic Shift Left (Byte) (dst) ← (dst) shifted 1 place left	n063DD	****	300 ns
				
SXT	Sign eXTend (dst) ← 0 if N=0 (dst) ← -1 if N=1	0067DD	.*0-	300 ns
SWAB	SWAp Bytes (dst byte 0) ← (dst byte 1) (dst byte 1) ← (dst byte 0)	0003DD	**00	300 ns

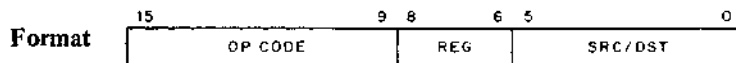
**DOUBLE OPERAND INSTRUCTIONS: OPR src, dst**



11-1469

Mnemonic	Instruction/Operation	OP Code	Condition Code NZVC	Time
MOV(B)	MOVe (Byte) $(dst) \leftarrow (src)$	n1SSDD	**0 -	300 ns
CMP(B)	CoMPare (Byte) $(src) + \sim (dst) + 1$ $(src), (dst)$ unaffected	n2SSDD	****	300 ns
BIT(B)	BIt Test (Byte) $(dst) \wedge (src)$ $(dst), (src)$ unaffected	n3SSDD	**0 -	300 ns
BIC(B)	BIt Clear (Byte) $(dst) \leftarrow \sim (src) \wedge (dst)$	n4SSDD	**0 -	300 ns
BIS(B)	BIt Set (Byte) $(dst) \leftarrow (src) \wedge (dst)$	n5SSDD	**0 -	300 ns
ADD	ADD $(dst) \leftarrow (src) + (dst)$	06SSDD	****	300 ns
SUB	SUBtract $(dst) \leftarrow (dst) + \sim (src) + 1$	16SSDD	****	300 ns

**REGISTER SRC/DST INSTRUCTIONS: OPR src, R**



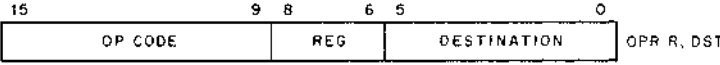
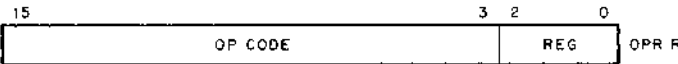
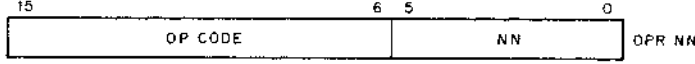
11-1470

Mnemonic	Instruction/Operation	OP Code	Condition Code NZVC	Time
MUL	MULTIply $R, RVI \leftarrow (R) \times (src)$	070RSS	**0*	3.3 $\mu$ s
DIV	DIVIDe quotient R $\leftarrow \frac{(R), (RVI)}{(src)}$ remainder RVI	071RSS	****	6.9-7.5 $\mu$ s
ASH	ArithmetiC SHift $R \leftarrow (R)$ Arith shifted N places right or left	072RSS	****	750 ns+
ASHC	ArithmetiC SHift Combined $R, RVI \leftarrow (R), (RVI)$ Arith shifted (two words) N places right or left	073RSS	****	750 ns+

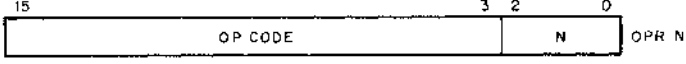
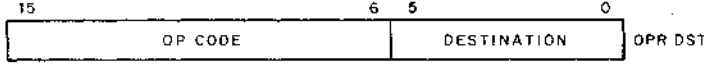
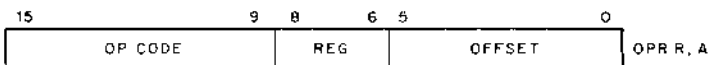
11-1471

XOR	Exclusive OR $(dst) \leftarrow R \vee (dst)$ Note: Syntax format is XOR R, dst	074RDD	**0 -	300 ns
-----	--	--------	-------	--------

### SUBROUTINE INSTRUCTIONS

Mnemonic	Instruction/Operation	OP Code	Condition Code NZVC	Time
JSR	Jump to SubRoutine tmp ← (dst) ↓(SP) ← (reg) reg ← (PC) PC ← (tmp)	004RDD	----	1.5 μs
<b>Format</b>				
RTS	ReTurn from Subroutine PC ← (reg) reg ← (SP)↑	00020R	11-1472 ----	1.2 μs
<b>Format</b>				
MARK	MARK SP ← (PC) + (2xN) PC ← (R5) R5 ← (SP)↑	0064NN	11-1473 ----	900 ns
<b>Format</b>	Note: NN=number of parameters 			

### PROGRAM CONTROL INSTRUCTIONS

Mnemonic	Instruction/Operation	OP Code	Condition Code NZVC	Time
SPL	Set Priority Level PSW(7-5) ← N	00023N	----	600 ns
<b>Format</b>	Note: Kernal mode only 			
JMP	JuMP PC ← dst	0001DD	11-1475 ----	600 ns
<b>Format</b>				
SOB	Subtract One and Branch R ← (R) - 1 PC ← (PC) - (2xOFFSET) if result ≠ 0 PC ← (PC) if result = 0	077RYY	11-1476 ----	750 ns
<b>Format</b>	Note: Branch back only if R≠0 			

**OPERATE INSTRUCTIONS: OPR**

<b>Mnemonic</b>	<b>Instruction/Operation</b>	<b>OP Code</b>	<b>Condition Code NZVC</b>	<b>Time</b>
HLT	HaLT CP ← Halt	000000	----	750 ns
WAIT	WAIT wait for interrupt	000001	----	
RTI	ReTurn from Interrupt PC ← (SP)↑ PSW ← (SP)↑	000002	****	1.5 μs
BPT	BreakPoint Trap ↓(SP) ← (PSW) ↓(SP) ← (PC) PC ← (loc 14) PSW ← (loc 16)	000003	****	2.25 μs
IOT	I/O Trap ↓(SP) ← (PSW) ↓(SP) ← (PC) PC ← (loc 20) PSW ← (loc 22)	000004	****	2.25 μs
RESET	RESET BUS INIT ← TRUE for 10 ms	000005	----	10 ms
RTT	ReTurn from Trap PC ← (SP)↑ PSW ← (SP)↑	000006	****	1.5 μs
EMT	EMulator Trap ↓(SP) ← (PSW) ↓(SP) ← (PC) PC ← (loc 30) PSW ← (loc 32)	104000 - 104377	****	2.25 μs
TRAP	TRAP ↓(SP) ← (PSW) ↓(SP) ← (PC) PC ← (loc 34) PSW ← (loc 36)	104400 - 104777	****	2.25 μs

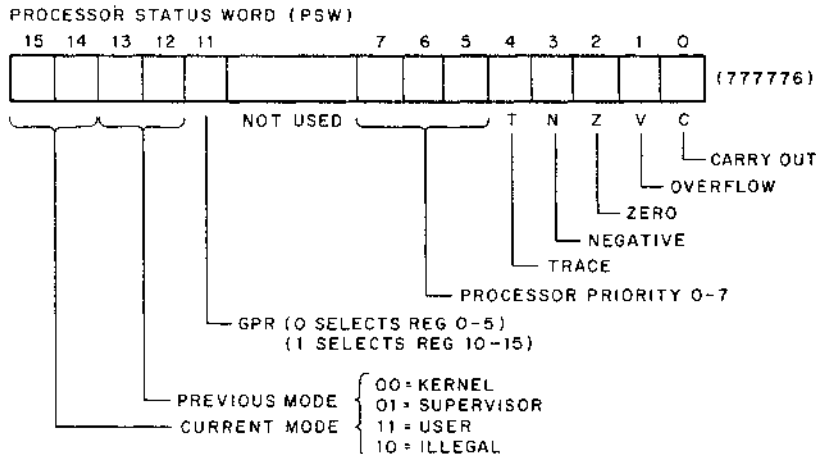
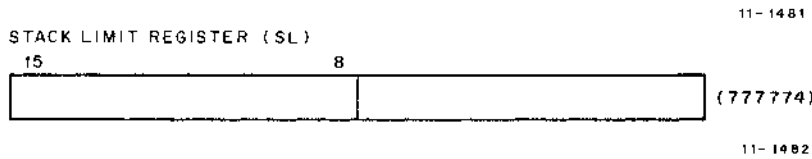
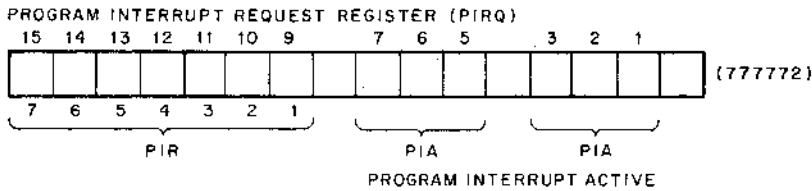
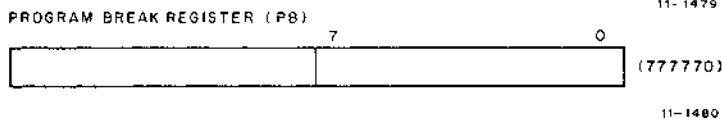
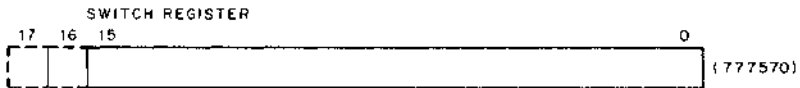
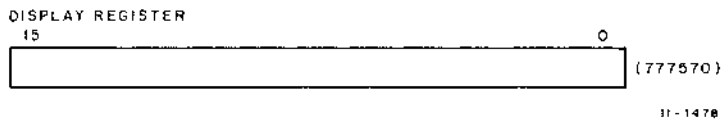
- Notes:
1. HALT issued in SUPERVISOR or USER mode will generate a trap to vector 4.
  2. SPL or RESET issued in SUPERVISOR or USER mode will be a NO OP.
  3. BPT, IOT, EMT and TRAP push old PC and old PSW onto stack of mode you are going to.

## PROCESSOR REGISTER ADDRESSES

### GENERAL REGISTERS

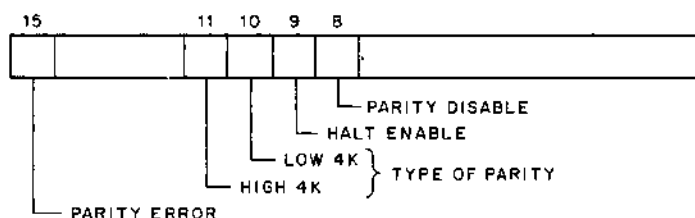
R0	(000000)	R10	(000010)
R1	(000001)	R11	(000011)
R2	(000002)	R12	(000012)
R3	(000003)	R13	(000013)
R4	(000004)	R14	(000014)
R5	(000005)	R15	(000015)
R6 KERNEL SP	(000006)	R16 SUPER SP	(000016)
R7 PC	(000007)	R17 USER SP	(000017)

(addressable only by console)



11-1483

## MEMORY PARITY CONTROL REGISTER



11-1484

### Addressing

0 - 8K	772100
8K - 16K	772102
16K - 24K	772104
24K - 32K	772106
32K - 40K	772110
40K - 48K	772112
48K - 56K	772114
56K - 64K	772116
64K - 72K	772120
72K - 80K	772122
80K - 88K	772124
88K - 96K	772126
96K - 104K	772130
104K - 112K	772132
112K - 120K	772134
120K - 128K	772136

Bits 11 and 10 are associated with the high-order 4K and low-order 4K of this memory address bank. When set to a 1, they specify odd parity for their respective half banks; when clear, even parity.

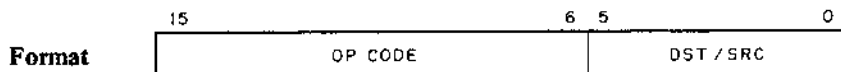
When bit 9 is set, the machine will execute a halt if a parity error occurs; when clear, the machine will perform an effective timeout and interrupt through location 4.

When bit 8 is clear, a parity error will cause an interrupt (or halt as specified in bit 9); if it is set, no action will be taken on a parity error.

When the machine is powered up, the status registers have bit 15 cleared to 0, and the remaining bits set to 1: halt, odd parity enable, parity disable, and no error.



INTER-MODE COMMUNICATIONS: OPR dst or OPR src

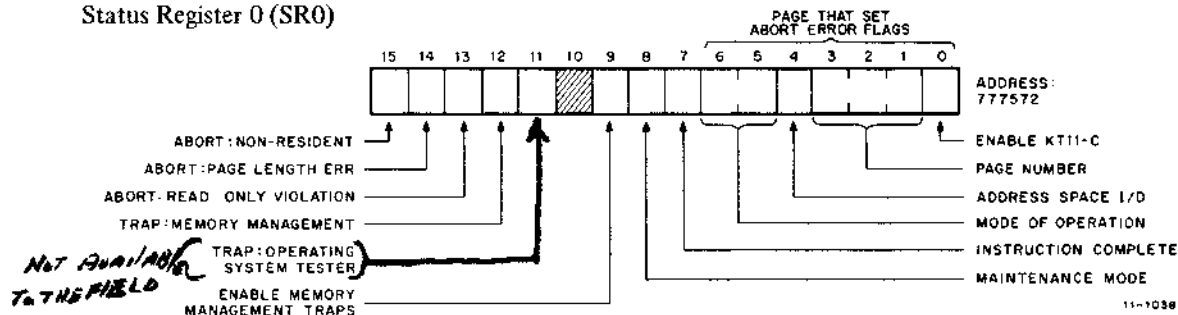


11-1485

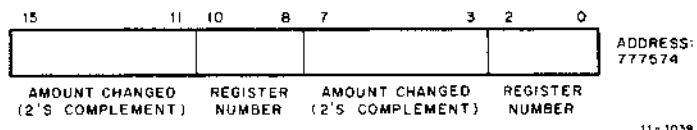
Mnemonic	Instruction/Operation	OP Code	Condition Code NZVC	Time
MFPI	Move From Previous Instruction space (temp) ← (src) ↓(SP) ← (temp)	0065SS	**0 -	1.2 μs
MFPD	Move From Previous Data space (temp) ← (src) ↓(SP) ← (temp)	1065SS	**0 -	1.2 μs
MTPI	Move To Previous Instruction space (temp) ← (SP)↑ (dst) ← (temp)	0066DD	**0 -	900 ns
MTPD	Move To Previous Data space (temp) ← (SP)↑ (dst) ← (temp)	1066DD	**0 -	900 ns

KT11-C MEMORY MANAGEMENT STATUS REGISTER

Status Register 0 (SR0)



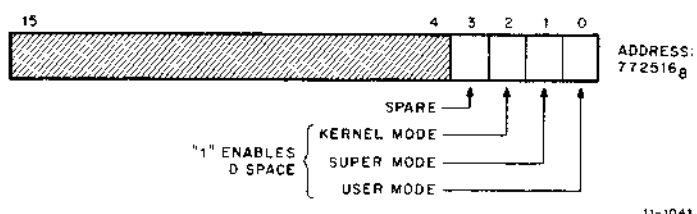
Status Register 1 (SR1)



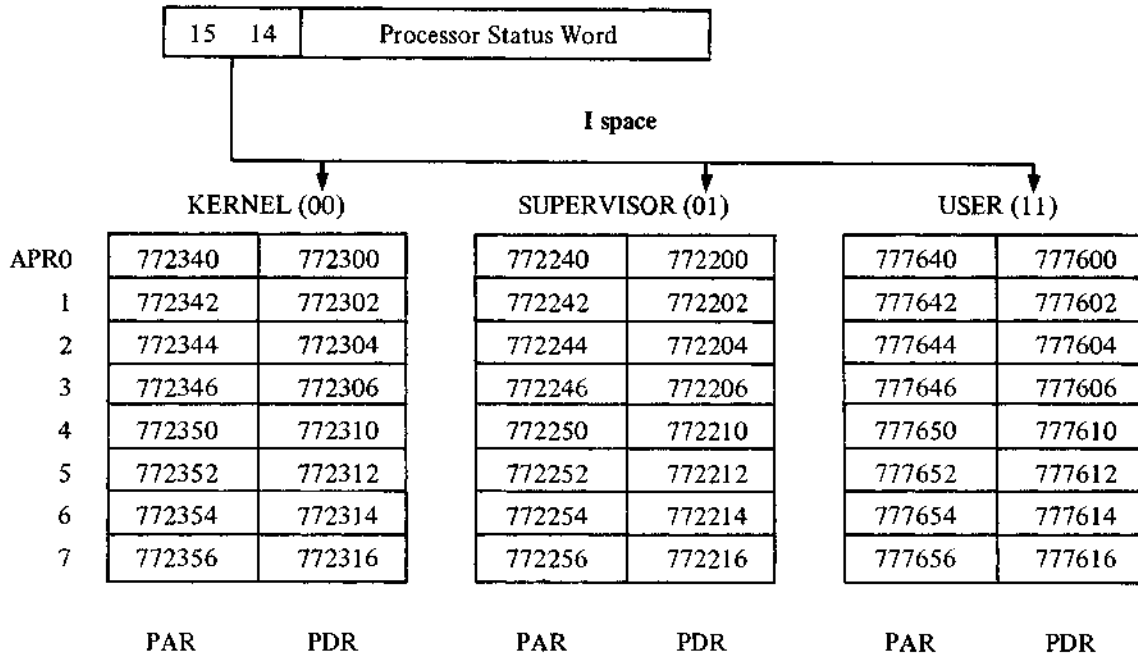
Status Register 2 (SR2)



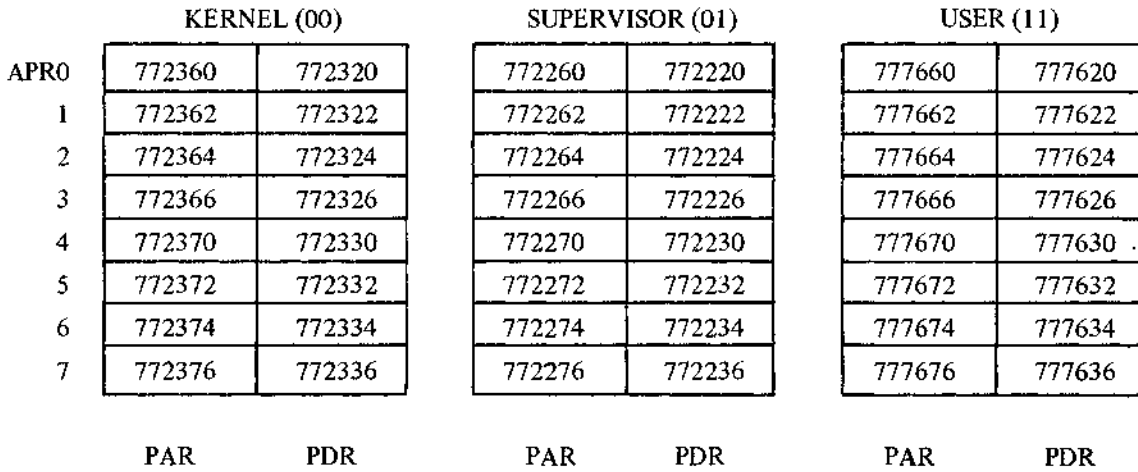
Status Register 3 (SR3)



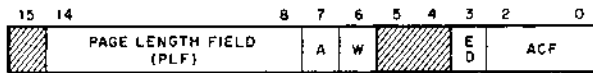
## ACTIVE PAGE REGISTERS



### D space

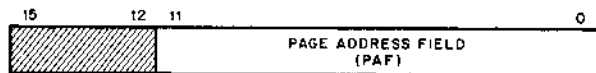


*DESCRIBER*  
**PAGE ADDRESS REGISTER**

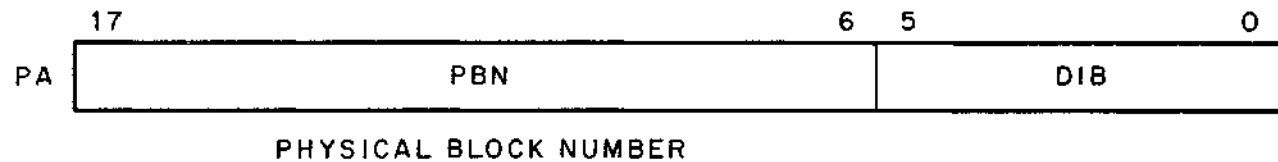
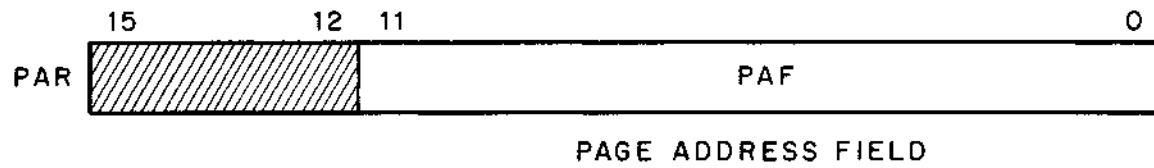
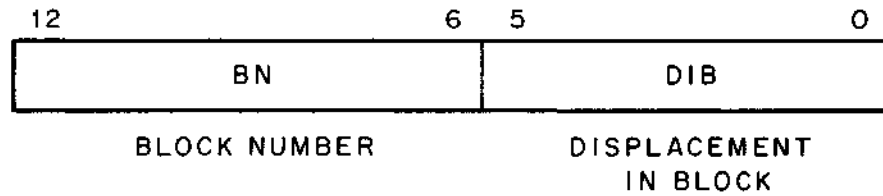
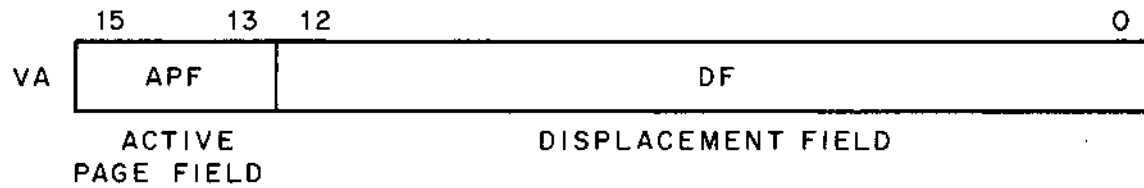


11-1037

*ADDRESS*  
**PAGE DESCRIPTOR REGISTER**



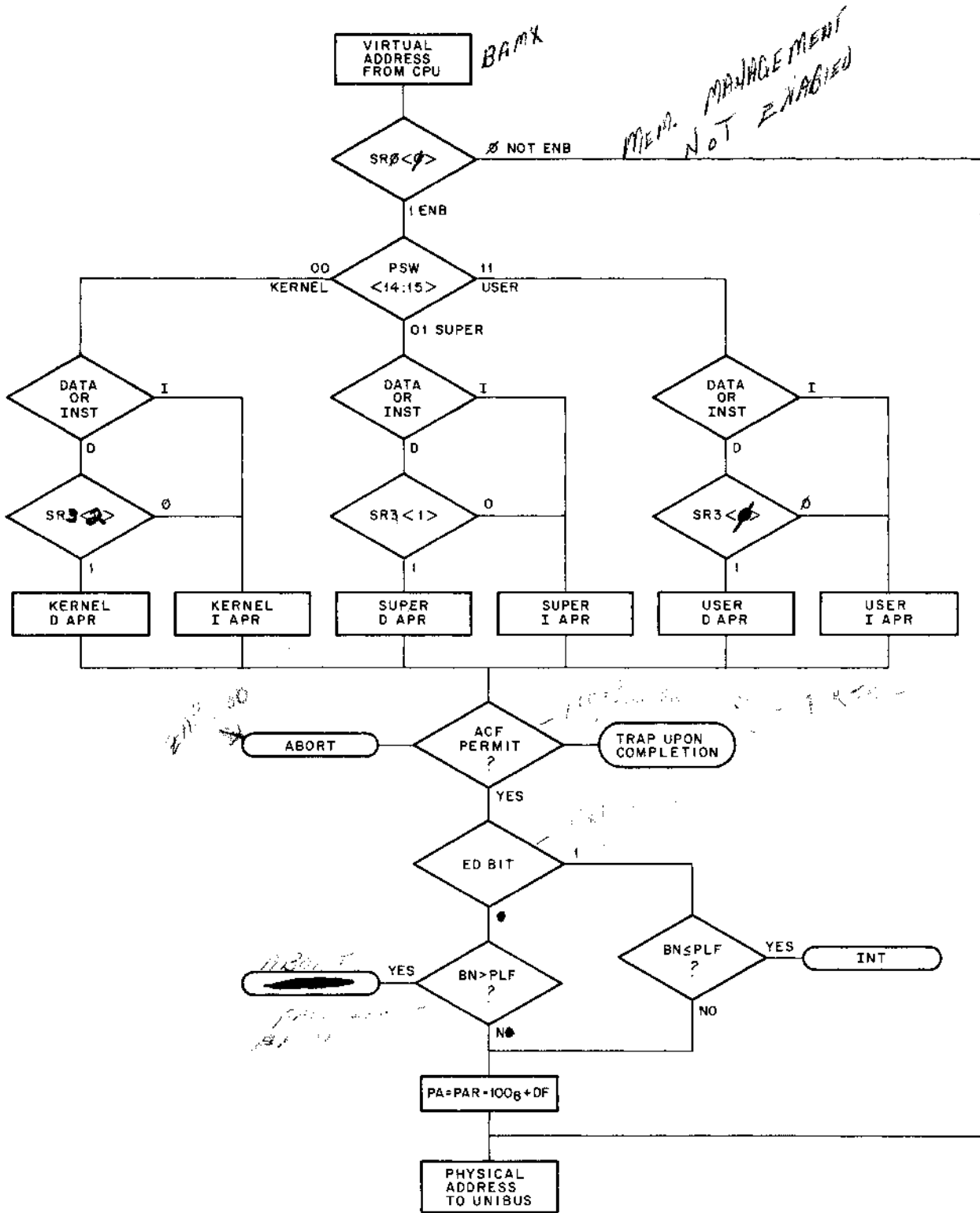
11-1036



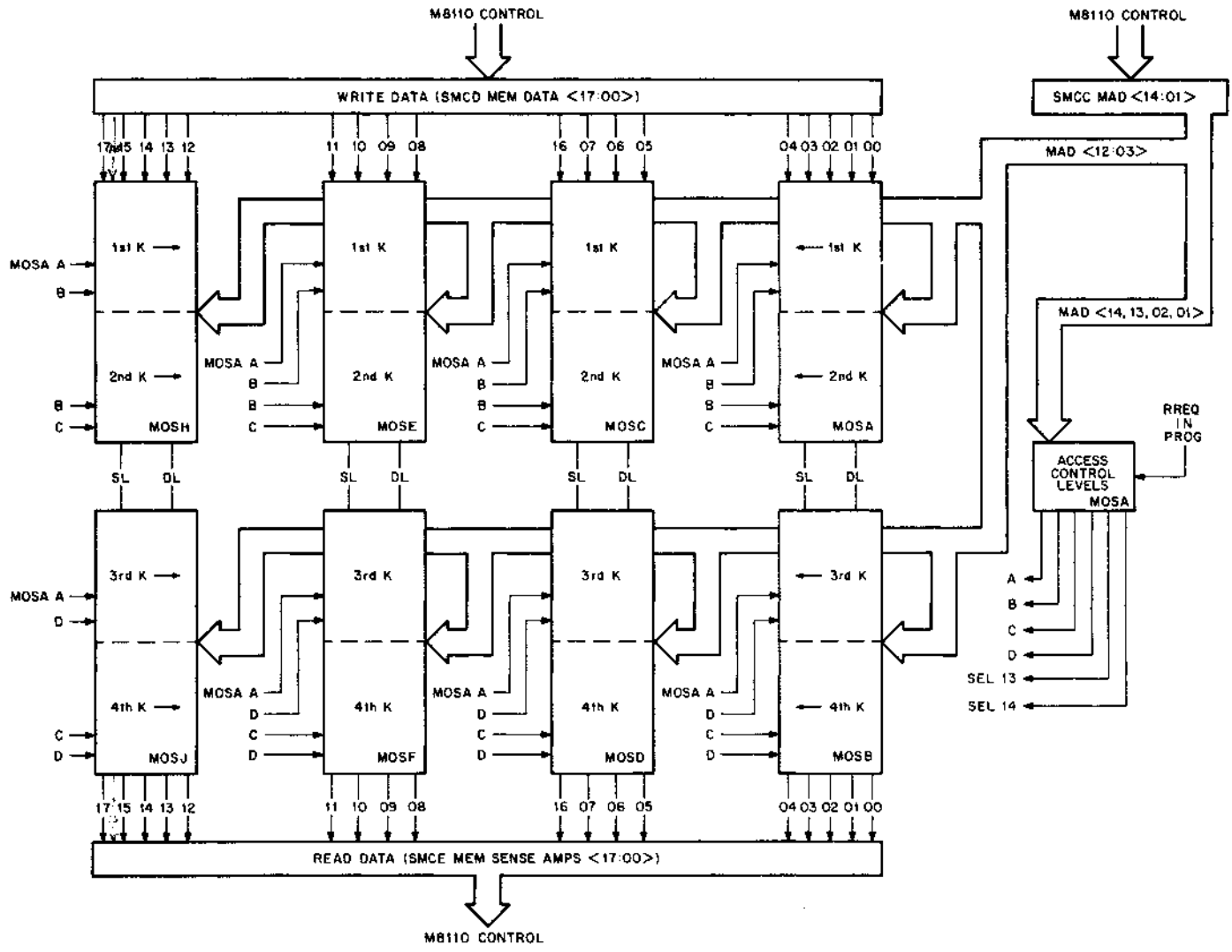
VIRTUAL TO PHYSICAL ADDRESS

11-1486

Virtual to Physical Address



Memory Management



MOS Memory Matrix Block Diagram

**MOS Memory System Configuration**

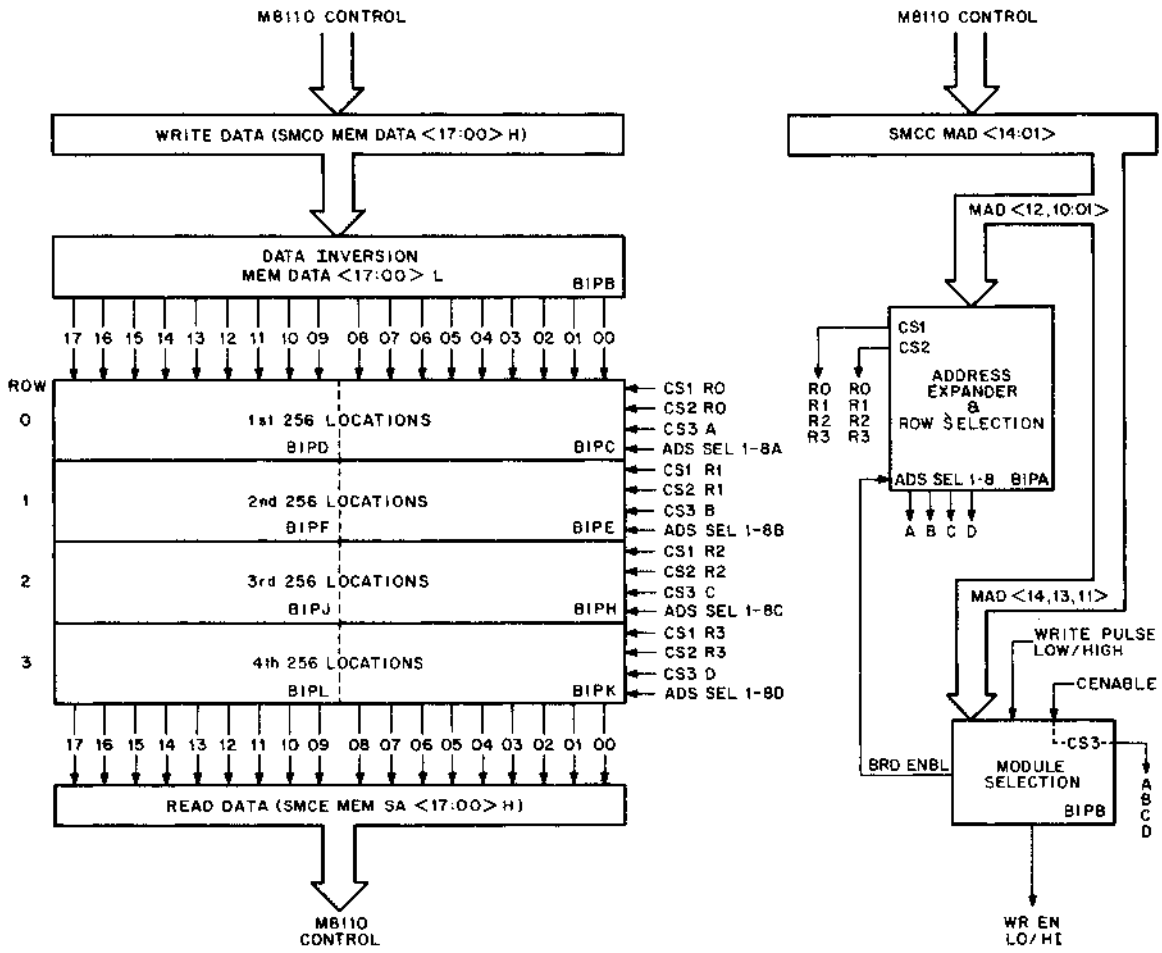
Memory Capacity Option		Option Type Number			
		MS11-BC	MS11-BD	MS11-BM	MS11-BP
		Module Complement			
With Parity	Without Parity	(1)M8110 (1)H746A (1)H744A	(1)M8110	(1)G401	(1) G401YA
4K		1			1
	4K	1		1	
8K		1			2
	8K	1		2	
12K		1			3
	12K	1		3	
16K		1			4
	16K	1		4	
20K		1	1		5
	20K	1	1	5	
24K		1	1		6
	24K	1	1	6	
28K		1	1		7
	28K	1	1	7	
32K		1	1		8
	32K	1	1	8	

**MOS Matrix Selected Address Configuration (4 of 16K)**

MAD		REQUIRED JUMPERS		MOS Matrix Memory Address Assignment
14	13	(MAD 14)	(MAD 13)	
0	0	C	A	0-4095
0	1	C	B	4096-8191
1	0	D	A	8192-12,287
1	1	D	B	12,288-16,383

**MOS Matrix Control Level Generation and Selected Memory Address Block (1 of 4K)**

MAD		CONTROL LEVELS GENERATED		Memory Address Block Selected
02	01	(MAD 02)	(MAD 01)	
0	0	MOSA B	• MOSA A	0-1023
0	1	MOSA B	• MOSA C	1024-2047
1	0	MOSA D	• MOSA A	2048-3071
1	1	MOSA D	• MOSA C	3072-4095



11-1325

### Bipolar Memory Matrix

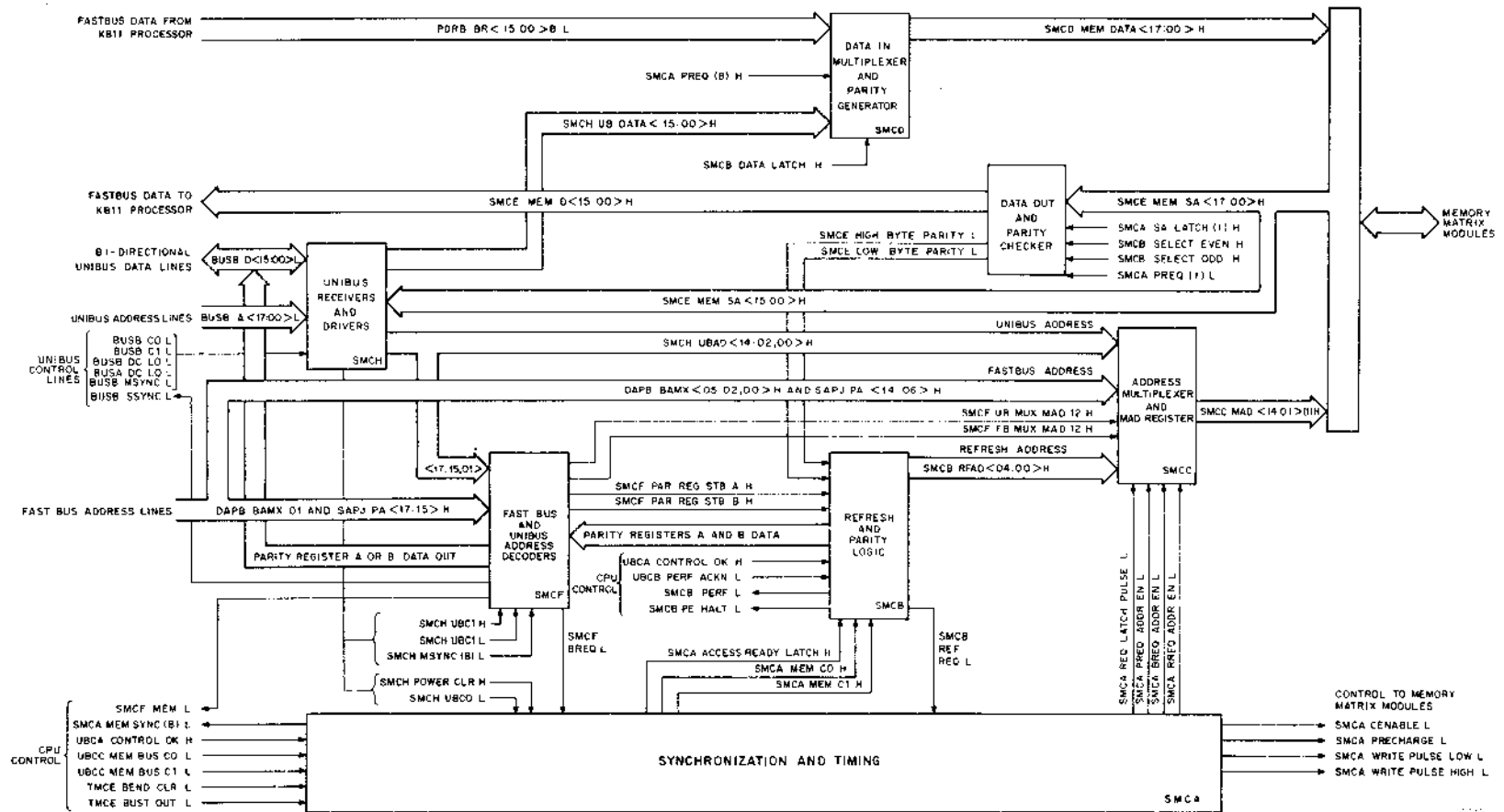
**Bipolar Memory System Configuration**

Memory Capacity Option		Option Type Number		
		MS11-CC	MS11-CM	MS11-CP
		Module Complement		
With Parity	Without Parity	(1)M8110 (2)H744A	(1)M8111	(1)M8111YA
1K		1		1
	1K	1	1	
2K		1		2
	2K	1	2	
3K		1		3
	3K	1	3	
4K		1		4
	4K	1	4	
5K		2		5
	5K	2	5	
6K		2		6
	6K	2	6	
7K		2		7
	7K	2	7	
8K		2		8
	8K	2	8	

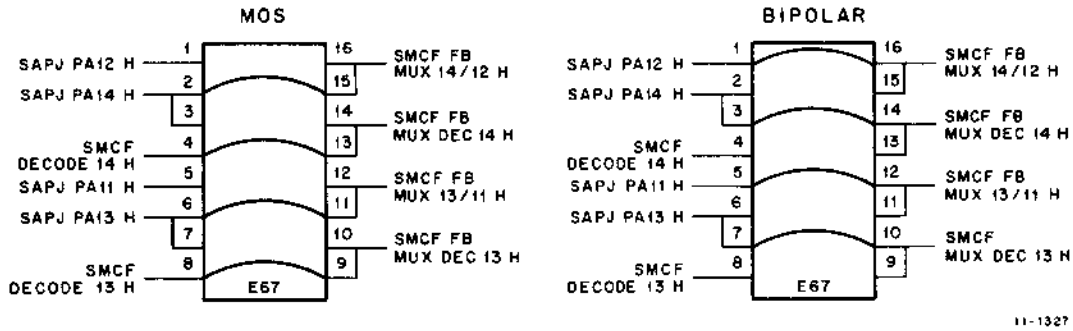
**Bipolar Matrix Selected Address Configuration (1 of 16K)**

MAD				Required Jumpers				Memory Address Assignment
14	13	11	10	(MAD 14)	(MAD 13)	(MAD 11)	(MAD 10)	
0	0	0	0	D	F	H	B	0 to 1023
0	0	0	1	D	F	H	A	1024 to 2047
0	0	1	0	D	F	J	B	2048 to 3071
0	0	1	1	D	F	J	A	3072 to 4095
0	1	0	0	D	E	H	B	4096 to 5119
0	1	0	1	D	E	H	A	5120 to 6143
0	1	1	0	D	E	J	B	6144 to 7167
0	1	1	1	D	E	J	A	7168 to 8191
1	0	0	0	C	F	H	B	8192 to 9215
1	0	0	1	C	F	H	A	9216 to 10,239
1	0	1	0	C	F	J	B	10,240 to 11,263
1	0	1	1	C	F	J	A	11,264 to 12,287
1	1	0	0	C	E	H	B	12,288 to 13,311
1	1	0	1	C	E	H	A	13,312 to 14,335
1	1	1	0	C	E	J	B	14,336 to 15,359
1	1	1	1	C	E	J	A	15,360 to 16,383



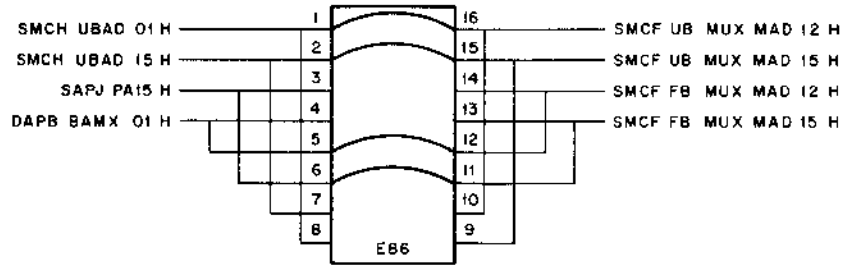


M8110 Semiconductor Memory Control Module, Block Diagram



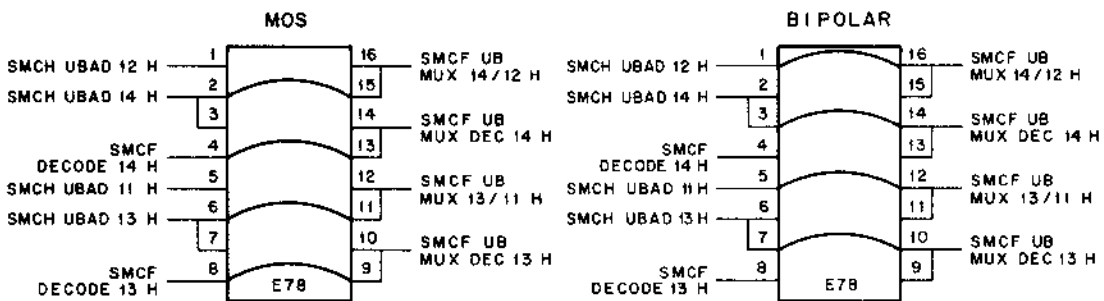
11-1327

**Fastbus Address Multiplexing (14:11), Required E67 Jumpers**



11-1328

**MAD Multiplexing, Required E86 Jumpers**



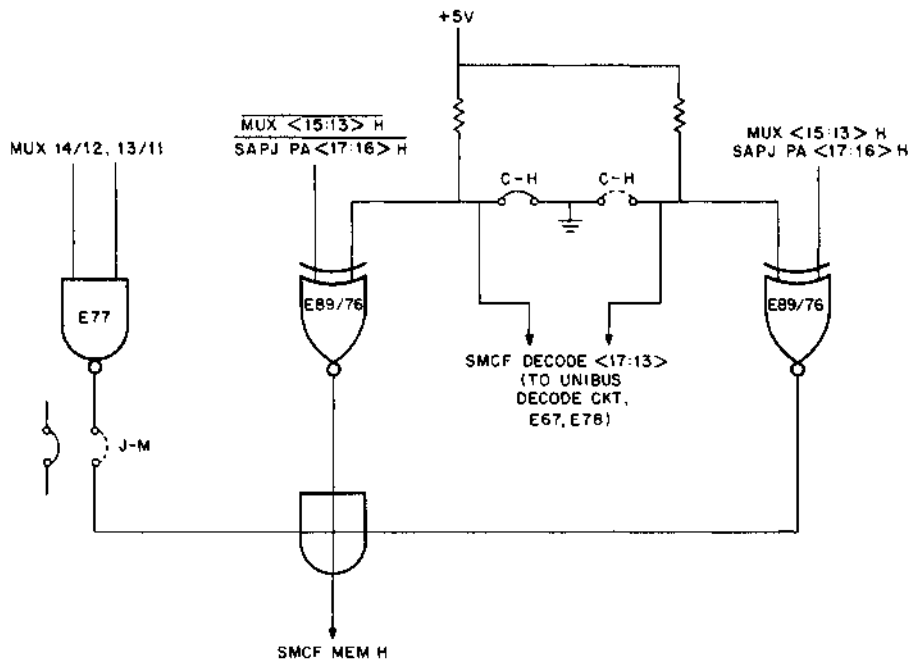
11-1329

**Unibus Address Multiplexing (14:11), Required E78 Jumpers**

**Fastbus/Unibus Memory Address (Assign and Decode)**

Fastbus/Unibus Address Decoder Bits					Memory Address Assignment		M8110 Jumpers (E87) (NOTE 1, NOTE 2)				
17	16	15	14	13	Bipolar	MOS	C	D	E	F	H
0	0	0	0	0	0-4K	0-16K					
0	0	0	0	1	4-8K						X
0	0	0	1	0	8-12K					X	
0	0	0	1	1	12-16K					X	X
0	0	1	0	0	16-20K	16-32K			X		
0	0	1	0	1	20-24K				X		X
0	0	1	1	0	24-28K				X	X	
0	0	1	1	1	28-32K				X	X	X
0	1	0	0	0	32-36K	32-48K		X			
0	1	0	0	1	36-40K			X			X
0	1	0	1	0	40-44K			X		X	
0	1	0	1	1	44-48K			X		X	X
0	1	1	0	0	48-52K	48-64K		X	X		
0	1	1	0	1	52-56K			X	X		X
0	1	1	1	0	56-60K			X	X	X	
0	1	1	1	1	60-64K			X	X	X	X
1	0	0	0	0	64-68K	64-80K	X				
1	0	0	0	1	68-72K		X				X
1	0	0	1	0	72-76K		X			X	
1	0	0	1	1	76-80K		X			X	X
1	0	1	0	0	80-84K	80-96K	X		X		
1	0	1	0	1	84-88K		X		X		X
1	0	1	1	0	88-92K		X		X	X	
1	0	1	1	1	92-96K		X		X	X	X
1	1	0	0	0	96-100K	96-112K	X	X			
1	1	0	0	1	100-104K		X	X			X
1	1	0	1	0	104-108K		X	X		X	
1	1	0	1	1	108-112K		X	X		X	X
1	1	1	0	0	112-116K	112-128K	X	X	X		
1	1	1	0	1	116-120K		X	X	X		X
1	1	1	1	0	120-124K		X	X	X	X	
1	1	1	1	1	124-128K		X	X	X	X	X

- NOTES:
1. "X" denotes jumper to be cut.
  2. Jumpers F and H are left intact for all MOS memory assignments.



11-1326

**Simplified Memory Address Decode (SMCF)**

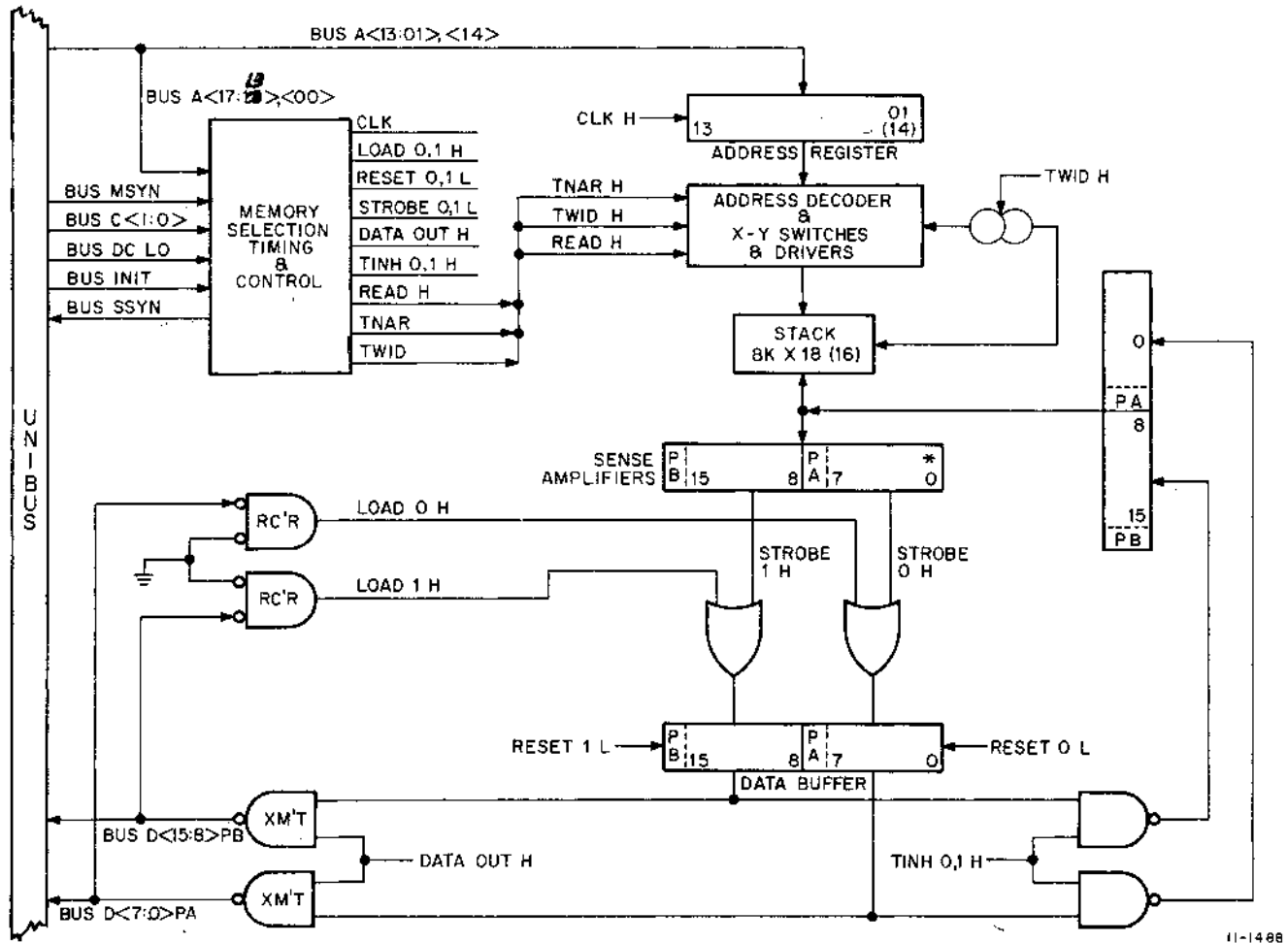
**MOS/Bipolar Module Addressing**

Fastbus/Unibus Memory Address Bits		Memory Address Assignment		Remove Jumpers	
FB MUX 14/12	FB MUX 13/11	MOS	Bipolar	Fastbus Address Select	Unibus Address Select
0	0	0-4095	0-1023	J	N
0	1	4096-8191	1024-2047	K	P
1	0	8192-12287	2048-3071	L	R
1	1	12288-16383	3072-4095	M	S

**MOS/Bipolar Memory Addressing**

No. of Memory Modules in Memory*	Memory Capacity		Remove Jumpers	
	MOS	Bipolar	Fastbus Address Select	Unibus Address Select
1	4K	1K	J	N
2	8K	2K	JK	NP
3	12K	3K	JKL	NPR
4	16K	4K	JKLM	NPRS

\*Connected to one M8110 Control.

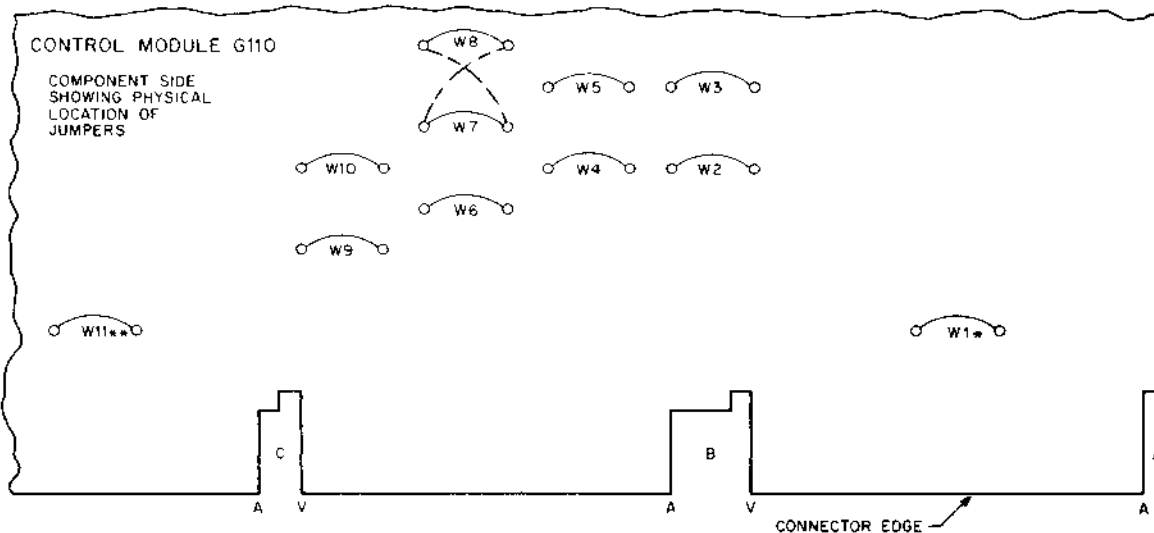
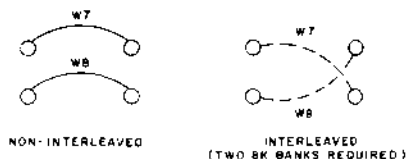


11-1488

MM-11 S, Simplified Block Diagram

Memory Bank (words)	Machine Address (words)	Device Address Jumpers (1)			
		W6 A14 or A01 (2)	W4 A15	W3 A16	W2 A17L
0-8K	00000-037776	In	In	In	In
8-16K	04000-077776	Out	In	In	In
16-24K	10000-137776	In	Out	In	In
24-32K	14000-177776	Out	Out	In	In
32-40K	20000-237776	In	In	Out	In
40-48K	24000-277776	Out	In	Out	In
48-56K	30000-337776	In	Out	Out	In
56-64K	34000-377776	Out	Out	Out	In
64-72K	40000-437776	In	In	In	Out
72-80K	44000-477776	Out	In	In	Out
80-88K	50000-537776	In	Out	In	Out
88-96K	54000-577776	Out	Out	In	Out
96-104K	60000-637776	In	In	Out	Out
104-112K	64000-677776	Out	In	Out	Out
112-120K	70000-737776	In	Out	Out	Out
120-128K	74000-767776	Out	Out	Out	Out

- (1) W5 and W10 must be installed and W9 must be removed.
- (2) The memory can be interleaved as 16K only, using two adjacent contiguously addressed 8K banks. When two 8K banks are interleaved, jumpers W7 and W8 must be in the configuration shown by the dotted lines. Bit A01 goes to the device selector gate controlled by jumper W6. One 8K bank must have W6 installed and the other must have W6 removed. When not interleaved, jumpers W7 and W8 must be in the configuration shown by the solid lines. Bit A14 goes to the device selector gate controlled by jumper W6.



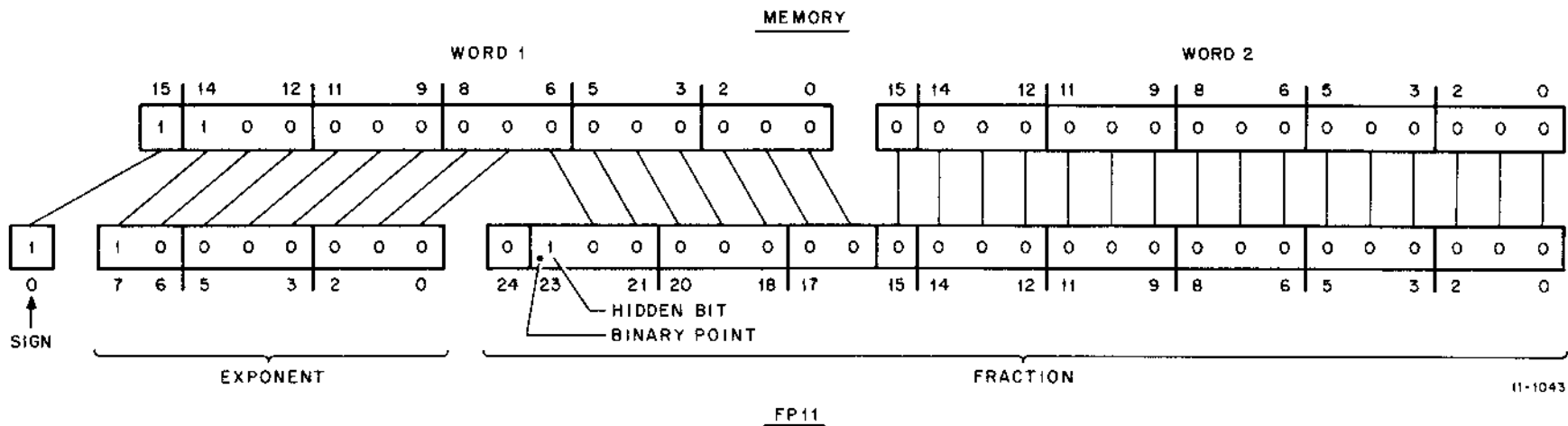
\* Jumper W1 is for test purposes only. It must be installed for normal operation.

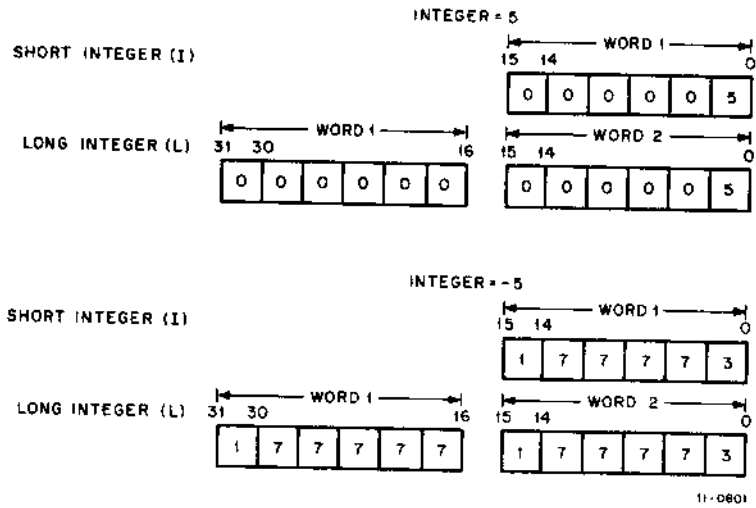
\*\* Jumper W11 should be removed for normal operation. When installed the memory responds to DAT1 only, regardless of state of control lines COD and CO1.

NOTE:  
Jumpers W5, W7, and W8 must remain in the factory installed positions.

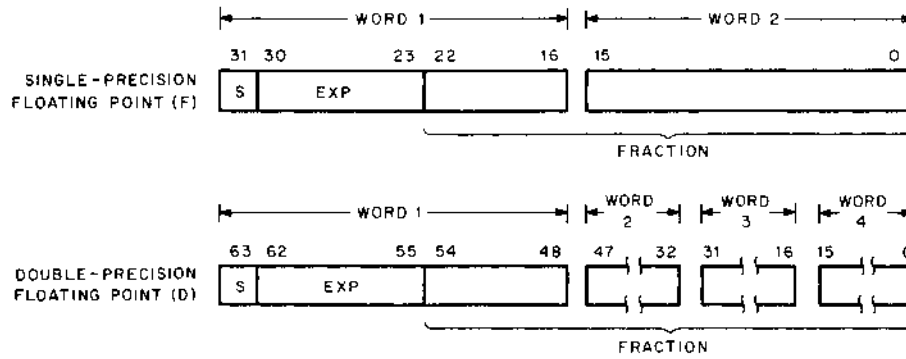
11 1149

## Device Decoding Guide





### Integer Formats



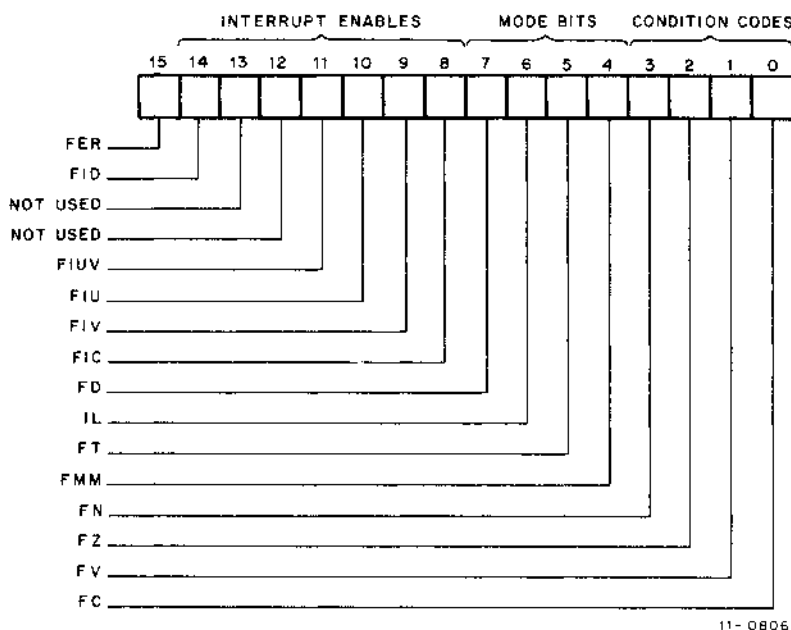
S = Sign

EXP = Exponent in excess  $200_8$  notation

Fraction = 23 or 55 bit fraction in sign and magnitude format.  
 Binary point between bits 22 and 23 for F format or between bits 54 and 55 for D format.

### Floating-Point Data Formats





### Status Register Format

**FER** – This bit indicates an error condition of the FP11.

**FID (Floating Interrupt Disable)** – All interrupts by the FP11 are disabled when this bit is on.

**FIUV (Floating Interrupt on Undefined Variable)** – When this bit is set and a minus 0 is obtained from memory, an interrupt occurs. If the bit is not set, minus 0 can be loaded and stored; however, any arithmetic operation is treated as if it were a positive 0.

**FIU (Floating Interrupt on Underflow)** – When this bit is set, an underflow condition causes a floating underflow interrupt. The result of the operation causing the interrupt is correct except for the exponent, which is off by  $400_8$ . If the FIU bit is not set and underflow occurs, the result is set to zero.

**FIV (Floating Interrupt on Overflow)** – When this bit is set, floating overflow causes an interrupt. The result of the operation causing the interrupt is correct except for the exponent, which is off by  $400_8$ . If the FIV bit is not set, the result of the operation is the same; the only difference is that the interrupt does not occur.

**FIC (Floating Interrupt on Integer Conversion Error)** – When this bit is set, and the Store Convert Floating to Integer instruction causes FC to be set (indicating a conversion error), an interrupt occurs. When a conversion error occurs, the destination register is cleared and the source register is untouched. When FIC is reset, the

result of the operation is the same; however, no interrupt occurs.

**FD (Double-Precision Mode Bit)** – This bit, when set, specifies double-precision format and, when reset, specifies single-precision format.

**IL (Long-Precision Integer Mode Bit)** – This bit is employed during conversion between integer and floating-point format. If set, double-precision, 2's complement integer format of 32 bits is specified; if reset, single-precision 2's complement integer of 16 bits is specified.

**FT (Truncate Bit)** – This bit, when set, causes the result of any floating-point operation to be truncated rather than rounded.

**FMM (Maintenance Mode Bit)** – This bit is used to enable special maintenance logic.

**FC, FV, FZ, and FN** – These bits are the four floating-point condition codes, which can be loaded in the CPU's C, V, Z, and N condition codes, respectively. This is accomplished by the Copy Floating Condition Codes (CFCC) instruction. To determine how each instruction affects the condition codes, refer to the instruction description in the *PDP-11 Handbook*.

For the Store Convert Floating to Integer instruction (which converts a floating-point number to an integer), the FC bit is set if the resulting integer is too large to be stored in the specified register.

## PROCESSING OF FLOATING-POINT EXCEPTIONS

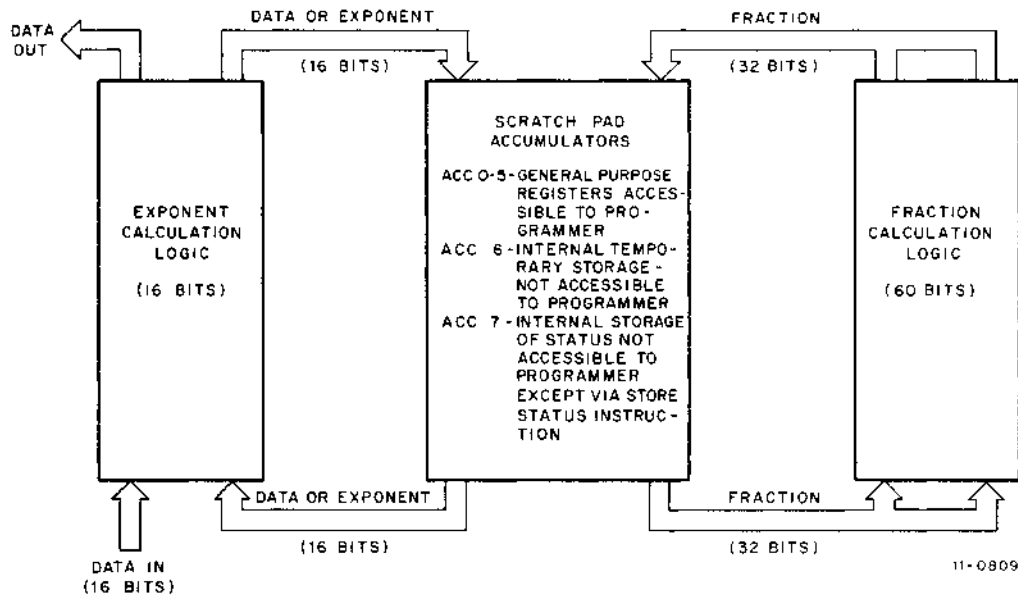
A total of seven possible interrupts can occur. These seven possible interrupt exceptions are encoded in the FP11 Exception Code Register (FEC). The interrupt exception codes represent an offset into a dispatch table, which routes the program to the right error handling routine. The dispatch table is a function of the software. The offset for each exception code is shown below followed by a brief description.

FP11 Exception Code (Base 8)	Definition
2	Floating Op Code Error – The FP11 causes an interrupt for an erroneous op code if the FID bit is not set.
4	Floating Divide by Zero – Division by zero causes an interrupt if the FID bit is not set.
6	Floating Integer Conversion Error
10	Floating Overflow
12	Floating Underflow
14	Floating Undefined Variable
16	Microbreak Trap

**NOTE**

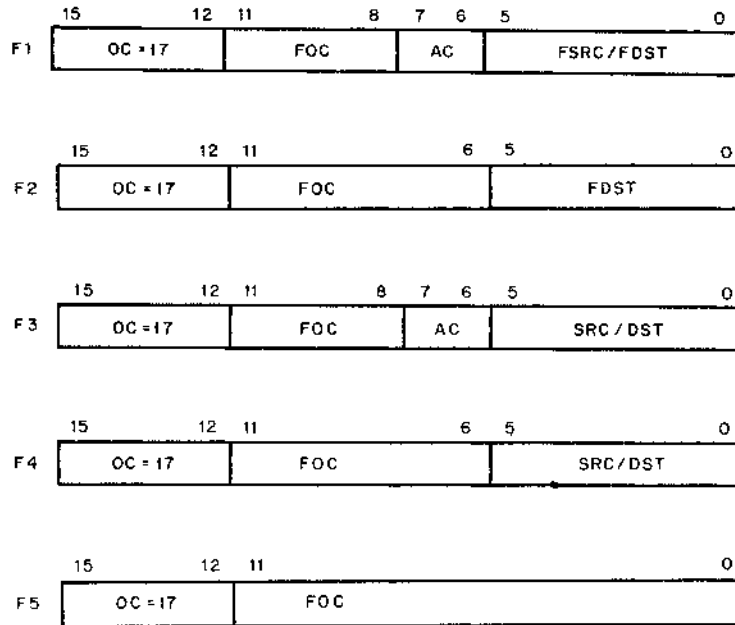
The traps for exception codes 6, 10, 12, and 14 can be enabled in the FPU's Program Status Register.

In addition to the FEC register, the FP11 contains a 16-bit Floating Exception Address register (FEA), which stores the address of the last floating-point instruction that caused a floating-point exception.



**FP11 Simplified Block Diagram**

## FP11 INSTRUCTION FORMATS



11 - 0800

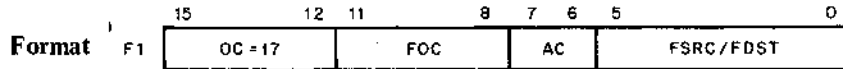
The 2-bit AC field (bits 6 and 7) allows selection of scratch pad accumulators 0 through 3 only. If address mode 0 is specified with formats F1 or F2, bits 2 through 0 are used to select the floating-point accumulator. Only accumulators 5 through 0 can be accessed in this manner. If accumulators 6 or 7 are specified, the FP11 traps if the interrupt is enabled.

The fields of the various instruction formats

Mnemonic	Description
OC	Operation Code — All floating-point instructions are designated by a 4-bit op code of $17_8$ .
FOC	Floating Operation Code — The number of bits in this field varies with the format and is used to specify the actual floating-point operation.
SRC	Source — A 6-bit source field identical to that in a PDP-11 instruction.
DST	Destination — A 6-bit destination field identical to that in a PDP-11 instruction.
FSRC	Floating Source — A 6-bit field used only in format F1. It is identical to SRC, except in mode 0 when it references a floating-point accumulator rather than a CPU general register.
FDST	Floating Destination — A 6-bit field used in formats F1 and F2. It is identical to DST, except in mode 0 when it references a floating-point accumulator instead of a CPU general register.
AC	Accumulator — A 2-bit field used only in formats F3 and F1 to specify accumulators 0 through 3.

Instruction Format	Instruction	Mnemonic
F1	ADD	ADDF FSRC, AC
	LOAD	ADDD FSRC, AC
		LDF FSRC, AC
		LDD FSRC, AC
	SUBTRACT	SUBF FSRC, AC
		SUBD FSRC, AC
	COMPARE	CMPF AC, FDST
		CMPD AC, FDST
	MULTIPLY	MULF FSRC, AC
		MULD FSRC, AC
	MODULO	MODF FSRC, AC
		MODD FSRC, AC
	STORE	STF AC, FDST
		STD AC, FDST
	DIVIDE	DIVF FSRC, AC
		DIVD FSRC, AC
	LOAD CONVERT	LDCFD FSRC, AC
		LDCDF FSRC, AC
F1	STORE CONVERT	STCFD AC, FDST
		STCDF AC, FDST
F2	CLEAR	CLRF FDST
		CLRD FDST
	TEST	TSTF FDST
		TSTD FDST
	ABSOLUTE	ABSF FDST
		ABSD FDST
F2	NEGATE	NEGF FDST
		NEGD FDST
F3	LOAD EXPONENT	LDEXP SRC, AC
	LOAD CONVERT INTEGER TO FLOATING	LCDIF SRC, AC
		LDCID SRC, AC
		LCDLF SRC, AC
		LDCLD SRC, AC
	STORE EXPONENT	STEXP AC, DST
F3	STORE CONVERT FLOATING TO INTEGER	STCFI AC, DST
		STCFL AC, DST
		STCDI AC, DST
		STCDL AC, DST
F4	LOAD FP11's PROGRAM STATUS	LDFPS SRC
	STORE FP11's PROGRAM STATUS	STFPS DST
F4	STORE FP11's STATUS	STST DST
F5	COPY FLOATING CONDITION CODES	CFCC
	SET FLOATING MODE	SETF
	SET INTEGER MODE	SETI
	LOAD UBREAK REGISTER	LDUB
	LOAD SHIFT COUNTER	LDSC
	STORE AR REGISTER IN AC0	STAO
	MAINTENANCE RIGHT SHIFT	MRS
	STORE QR REGISTER IN AC0	STOQ
	SET DOUBLE MODE	SETD
F5	SET LONG INTEGER MODE	SETL

**DOUBLE OPERAND INSTRUCTIONS: OPR FSRC, AC  
OPR AC, FDST**

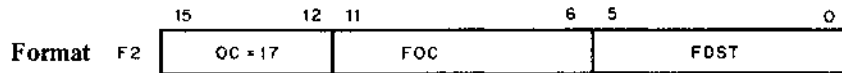


Mnemonic	Instruction/Operation	OP Code
MULF FSRC, AC MULD FSRC, AC	<p>Floating Multiply</p> <p><math>AC \leftarrow (AC) * (FSRC)</math> if <math>[(AC) * (FSRC)] \geq LOLIM</math>;  else <math>AC \leftarrow 0</math></p> <p><math>FC \leftarrow 0</math></p> <p><math>FV \leftarrow 1</math> if <math>(AC) &gt; UPLIM</math>; else <math>FV \leftarrow 0</math></p> <p><math>FZ \leftarrow 1</math> if <math>(AC) = 0</math>; else <math>FZ \leftarrow 0</math></p> <p><math>FN \leftarrow 1</math> if <math>(AC) &lt; 0</math>; else <math>FN \leftarrow 0</math></p>	171000 + AC * 100 + FSRC
MODF FSRC, AC MODD FSRC, AC	<p>Floating Modulo</p> <p><math>AC \vee 1 \leftarrow</math> integer part of <math>[(AC) * (FSRC)]</math></p> <p><math>AC \leftarrow</math> fractional part of <math>(AC) * (FSRC) - (AC \vee 1)</math> if  <math>  (AC) * (FSRC)   \geq LOLIM</math> or <math>FIU = 1</math>; else <math>AC \leftarrow 0</math></p> <p><math>FC \leftarrow 0</math></p> <p><math>FV \leftarrow 1</math> if <math>(AC) &gt; UPLIM</math>; else <math>FV \leftarrow 0</math></p> <p><math>FZ \leftarrow 1</math> if <math>(AC) = 0</math>; else <math>FZ \leftarrow 0</math></p> <p><math>FN \leftarrow 1</math> if <math>(AC) &lt; 0</math>; else <math>FN \leftarrow 0</math></p> <p>The product of (AC) and (FSRC) is 48 bits in single-precision floating-point format or 59 bits in double-precision floating-point format. The integer part of the product <math>[(AC) * (FSRC)]</math> is found and stored in <math>AC \vee 1</math>. The fractional part is then obtained and stored in AC. Note that multiplication by 10 can be done with zero error, allowing decimal digits to be stripped off with no loss in precision.</p>	171400 + AC * 100 + FSRC
ADDF FSRC, AC ADDD FSRC, AC	<p>Floating Add</p> <p><math>AC \leftarrow (AC) + (FSRC)</math> if <math>\{(AC) + (FSRC)\} &gt; UPLIM</math>  else <math>AC \leftarrow 0</math></p> <p><math>FC \leftarrow 0</math></p> <p><math>FV \leftarrow 1</math> if <math>(AC) &gt; UPLIM</math>; else <math>FV \leftarrow 0</math></p> <p><math>FZ \leftarrow 1</math> if <math>(AC) = 0</math>; else <math>FZ \leftarrow 0</math></p> <p><math>FN \leftarrow 1</math> if <math>(AC) &lt; 0</math>; else <math>FN \leftarrow 0</math></p>	172000 + AC * 100 + FSRC
LDF FSRC, AC LDD FSRC, AC	<p>Floating Load</p> <p><math>AC \leftarrow (FSRC)</math></p> <p><math>FC \leftarrow 0</math></p> <p><math>FV \leftarrow 0</math></p> <p><math>FZ \leftarrow 1</math> if <math>(AC) = 0</math>; else <math>FZ \leftarrow 0</math></p> <p><math>FN \leftarrow 1</math> if <math>(AC) &lt; 0</math>; else <math>FN \leftarrow 0</math></p>	172400 + AC * 100 + FSRC
SUBF FSRC, AC SUBD FSRC, AC	<p>Floating Subtract</p> <p><math>AC \leftarrow (AC) - (FSRC)</math> if <math>[(AC) - (FSRC)] \geq LOLIM</math>  else <math>AC \leftarrow 0</math></p> <p><math>FC \leftarrow 0</math></p> <p><math>FV \leftarrow 1</math> if <math>(AC) &gt; UPLIM</math>; else <math>FV \leftarrow 0</math></p> <p><math>FZ \leftarrow 1</math> if <math>(AC) = 0</math>; else <math>FZ \leftarrow 0</math></p> <p><math>FN \leftarrow 1</math> if <math>(AC) &lt; 0</math>; else <math>FN \leftarrow 0</math></p>	173000 + AC * 100 + FSRC

**DOUBLE OPERAND INSTRUCTIONS: OPR FSRC, AC (Cont.)**  
**OPR AC, FDST**

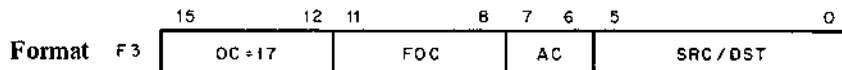
Mnemonic	Instruction/Operation	OP Code
CMPF FSRC, AC CMPD FSRC, AC	Floating Compare $(FSRC) - (AC)$ $FC \leftarrow 0$ $FV \leftarrow 0$ $FZ \leftarrow 1$ if $(FSRC) - (AC) = 0$ ; else $FZ \leftarrow 0$ $FN \leftarrow 1$ if $(FSRC) - (AC) < 0$ ; else $FN \leftarrow 0$	173400 + AC * 100 + FDST
STF AC, FDST STD AC, FDST	Floating Store $FDST \leftarrow (AC)$ $FC \leftarrow FC$ $FV \leftarrow FV$ $FZ \leftarrow FZ$ $FN \leftarrow FN$	174000 + AC * 100 + FDST
DIVF FSRC, AC DIVD FSRC, AC	Floating Divide $AC \leftarrow (AC)/(FSRC)$ if $[(AC)/(FSRC)] > LOLIM$ ; else $AC \leftarrow 0$ $FC \leftarrow 0$ $FV \leftarrow 1$ if $(AC) > UPLIM$ $FZ \leftarrow 1$ if $(AC) = 0$ ; else $FZ \leftarrow 0$ $FN \leftarrow 1$ if $(AC) < 0$ ; else $FN \leftarrow 0$	174400 + AC * 100 + FSRC
STCFD AC, FDST STCDF AC, FDST	Store Convert from Floating to Double or Double to Floating $FDST \leftarrow C_{F,D} \vee D_{,F} (AC)$ $FC \leftarrow 0$ $FV \leftarrow 1$ if $(AC) > UPLIM$ ; else $FV \leftarrow 0$ $FZ \leftarrow 1$ if $(AC) = 0$ ; else $FZ \leftarrow 0$ $FN \leftarrow 1$ if $(AC) < 0$ ; else $FN \leftarrow 0$	176000 + AC * 100 + FDST F,D – single-precision to double-precision floating D,F – double-precision to single-precision floating
LDCDF FSRC, AC LDCFD FSRC, AC	Load Convert Double to Floating or Floating to Double $AC \leftarrow C_{F,D} \vee D_{,F} (FSRC)$ $FC \leftarrow 0$ $FV \leftarrow 1$ if $(AC) > UPLIM$ ; else $FV \leftarrow 0$ $FZ \leftarrow 1$ if $(AC) = 0$ ; else $FZ \leftarrow 0$ $FN \leftarrow 1$ if $(AC) < 0$ ; else $FN \leftarrow 0$ If the current format is single-precision floating-point ( $FD = 0$ ), the source is assumed to be a double-precision number and is converted to single precision. If the floating truncate bit is set the number is truncated; otherwise, it is rounded. If the current format is double-precision ( $FD = 1$ ), the source is assumed to be a single-precision number and is loaded left justified in the AC. the lower half of the AC is cleared.	177400 + AC * 100 + FSRC F,D – single-precision to double-precision floating D,F – double-precision to single-precision floating

**SINGLE OPERAND INSTRUCTIONS: OPR FDST**



Mnemonic	Instruction/Operation	OP Code
CLRF FDST CLR D FDST	Clear FDST ← 0 FC ← 0 FV ← 0 FZ ← 1 FN ← 0	170400 + FDST
TSTF FDST TST D FDST	Test FDST ← (FDST) FC ← 0 FV ← 0 FZ ← 1 if (FDST) = 0; else FZ ← 0 FN ← 1 if (FDST) < 0; else FN ← 0	170500 + FDST
ABSF FDST ABS D FDST	Absolute FDST ← – (FDST) if (FDST) < 0; else FDST ← (FDST) FC ← 0 FV ← 0 FZ ← 1 if (FDST) = 0; else FZ ← 0 FN ← 0	170600 + FDST
NEGF FDST NEG D FDST	Negate FDST ← – (FDST) FC ← 0 FV ← 0 FZ ← 1 if (FDST) = 0; else FZ ← 0 FN ← 1 if (FDST) < 0; else FN ← 0	170700 + FDST

**DOUBLE OPERAND INSTRUCTIONS: OPR SRC  
OPR DST**



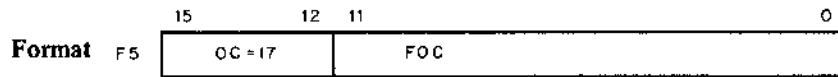
Mnemonic	Instruction/Operation	OP Code
STEXP AC, DST	Store Exponent DST ← AC EXPONENT – 200 FC ← 0 FV ← 0 FZ ← 1 if (DST) = 0; else FZ ← 0 FN ← 1 if (DST) < 0; else FN ← 0 C ← FC V ← FV Z ← FZ N ← FN	175000 + AC * 100 + DST

**DOUBLE OPERAND INSTRUCTIONS: OPR SRC (Cont.)  
OPR DST**

Mnemonic	Instruction/Operation	OP Code
STCFI AC, DST STCFL AC, DST STCDI AC, DST STCDL AC, DST	<p>Store Convert from Floating to Integer Destination receives converted AC if the resulting integer number can be represented in 16 bits (short integer) or 32 bits (long integer). Otherwise, destination is zeroed and C bit is set.</p> <p><math>FV \leftarrow 0</math>  <math>FZ \leftarrow 1</math> if (DST) = 0; else <math>FZ \leftarrow 0</math>  <math>FN \leftarrow 1</math> if (DST) &lt; 0; else <math>FN \leftarrow 0</math>  <math>C \leftarrow FC</math>  <math>V \leftarrow FV</math>  <math>Z \leftarrow FZ</math>  <math>N \leftarrow FN</math></p> <p>When the conversion is to long integer (32 bits) and address mode 0 or immediate mode is specified, only the most significant 16 bits are stored in the destination register.</p>	<p style="text-align: center;">175400 + AC * 100 + DST</p> <p>STCFI – Single float to single integer  STCFL – Single float to long integer  STCDI – Double float to single integer  STCDL – Double float to long integer</p>
LDEXP SRC, AC	<p>Load Exponent  <math>AC\ SIGN \leftarrow (AC\ SIGN)</math>  <math>AC\ EXP \leftarrow (SRC) + 200</math>  <math>AC\ FRACTION \leftarrow (AC\ FRACTION)</math>  <math>FC \leftarrow 0</math>  <math>FV \leftarrow 1</math> if (AC) &gt; UPLIM  <math>FZ \leftarrow 1</math> if (AC) = 0; else <math>FZ = 0</math>  <math>FN \leftarrow 1</math> if (AC) &lt; 0; else <math>FN = 0</math></p>	<p style="text-align: center;">176400 + AC * 100 + SRC</p>
LDCIF SRC, AC LDCID SRC, AC LDCLF SRC, AC LDCLD SRC, AC	<p>Load and Convert from Integer to Floating  <math>AC \leftarrow C_{FL,FD} (SRC)</math>  <math>FC \leftarrow 0</math>  <math>FV \leftarrow 0</math>  <math>FZ \leftarrow 1</math> if (AC) = 0; else <math>FZ \leftarrow 0</math>  <math>FN \leftarrow 1</math> if (AC) &lt; 0; else <math>FN \leftarrow 0</math></p> <p><math>C_{FL,FD}</math> specifies conversion from a 2's complement integer with precision I or L to a floating-point number of precision F or D. If integer flip-flop IL = 0, a 16-bit integer (I) is specified; if IL = 1, a 32 bit integer (L) is specified. If floating-point flip-flop FD = 0, a 32-bit floating-point number (F) is specified; if FD = 1, a 64-bit floating-point number (D) is specified. If a 32-bit integer is specified and addressing mode 0 or immediate mode is used, the 16-bits of the source register are left justified, and the remaining 16-bits are zeroed before the conversion.</p>	<p style="text-align: center;">177000 + AC * 100 + SRC</p> <p>LDCIF – single integer to single float  LDCID – single integer to double float  LDCLF – long integer to single float  LDCLD – long integer to double float</p>

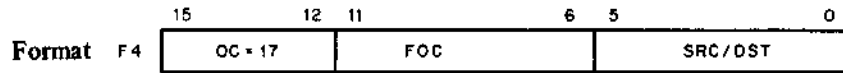


**OPERATE INSTRUCTIONS: OPR**



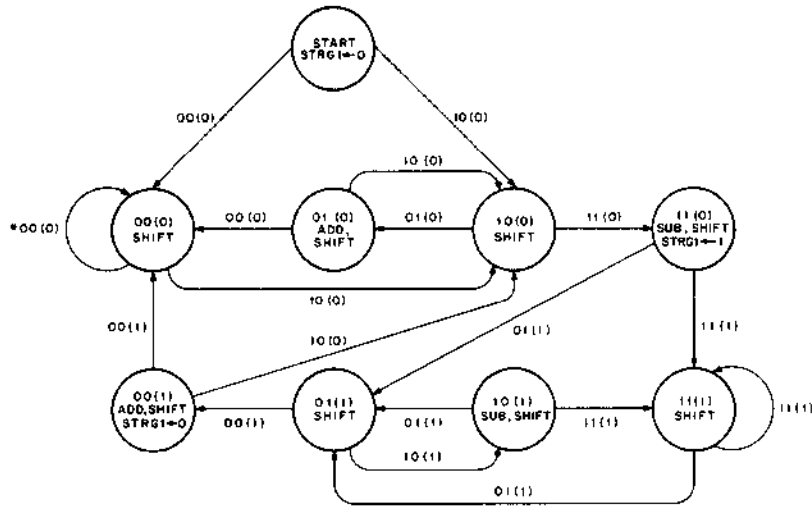
Mnemonic	Instruction/Operation	OP Code
CFCC	Copy Floating Condition Codes C ← FC V ← FV Z ← FZ N ← FN	170000
SETF	Set Floating Mode FD ← 0	170001
SETI	Set Integer Mode FL ← 0	170002
LDUB	Load Microbreak Register This instruction is a maintenance instruction in which the content of register R3 is gated into the UB register. When the control ROM address register matches the contents of the UB register, a scope sync is generated. If the FP11 is in maintenance mode (FMM=1), an interrupt is also generated and the FPU traps to the Ready state. A UB interrupt cannot be generated by the Ready state or by the states that are used to generate the UB interrupt.	170003
LDSC	Load Step Counter This is a maintenance instruction in which the content of register R4 is gated into the step counter, if the FP11 is in maintenance mode (FMM=1). Whenever the step counter is loaded by an LDSC, normal loading via the microprogram is inhibited until the step counter is incremented to zero. This allows partial quotients and products to be formed for diagnostic purposes. If FMM=0, the LDSC acts as a NOP.	170004
STAO	Store AR in AC0 AC0 (54:32) ← AR (57:35) if FD = 0 AC0 (54:0) ← AR (57:3) if FD = 1	170005
MRS	Maintenance Right Shift AR ← AR/2; QR ← QR/2	170006
STQO	Store QR in AC0 BR ← QR; AC (54:32) ← BR (57:35) if FD = 0 AC0 (54:0) ← BR (57:3) if FD = 1	170007
SETD	Set Floating Double Mode FD ← 1	170011
SETL	Set Long Integer Mode FL ← 1	170012

**SINGLE OPERAND INSTRUCTIONS: OPR SRC  
OPR DST**



Mnemonic	Instruction/Operation	OP Code
LDFPS SRC	Load FP11's Program Status Word FPS ← (SRC)	170100 + SRC
STFPS DST	Store FP11's Program Status Word DST ← (FPS)	170200 + DST
STST DST	Store FP11's Status DST ← (FEC) DST + 2 ← (FEA) if not mode 0 or not immediate mode	170300 + DST

Ill Op Code	No Mem Class	Load Class	Store Class	Convert Special*
ROM 214	ROM 234	ROM 254	ROM 274	ROM 270
1. All illegal FP instruction	LDUB M0 LDF M0 STF M0	ADDF ~ M0 SUBF ~ M0 COMPF ~ M0	CLRF ~ M0 STF ~ M0 STCFD ~ M0	STEXP STCFI STCFD
2. If address mode 0 in format F1 or F2 and AC <sub>j</sub> 6 or 7 selected	CLRF M0 LDSC M0 Neg F M0 ABSF ~ M0 TSTF M0 LDFDS M0 LDCIF M0 SETF M0 SETD M0 SETI M0 SETL M0 MRS M0 STQ0 M0 LDEXP M0 ADDF M0 SUBF M0 COMP M0 MULF M0 DIVF M0 LDCDF M0 LDCFD M0	MULF ~ M0 MODF ~ M0 DIV ~ M0 LDCDF ~ M0 LDF ~ M0 LDCFD ~ M0 LDCIF ~ M0 LDFPS ~ M0 CFCC *M0 ~ M0 NEGF ~ M0 ABSF ~ M0 TSTF ~ M0 LDEXP ~ M0	STFPS ~ M0 ~ M0 STST ~ M0 M0 STCFD ~ M0	*Do conversion then use store F1.



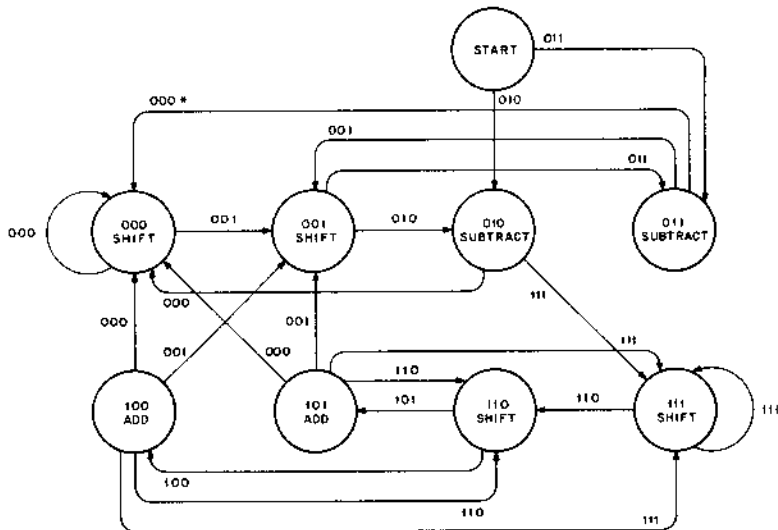
QR3(DBL) QR35(SNG)	QR2(DBL) QR34(SNG)	STRG1	FUNCTION
0	0	0	RIGHT SHIFT QR, AR, INCREMENT SC**
0	1	0	AR ← -BR + AR, RIGHT SHIFT QR, AR, INCREMENT SC
1	0	0	RIGHT SHIFT QR, AR, INCREMENT SC
1	1	0	AR ← AR - BR, RIGHT SHIFT QR, AR, SET STRG1, INCREMENT SC
0	0	1	AR ← AR + BR, RIGHT SHIFT QR, AR, RESET STRG1, INCREMENT SC
0	1	1	RIGHT SHIFT QR, AR, INCREMENT SC
1	0	1	AR ← AR - BR, RIGHT SHIFT QR, AR, INCREMENT SC
1	1	1	RIGHT SHIFT QR, AR INCREMENT SC

\* For double precision format 00(0) = QR3, QR2, (STNG 1)  
 For single precision format 00(0) = QR35, QR34, (STNG 1)

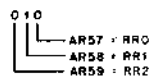
\*\* The step counter is set to the two's complement of the number of bits in the multiplier and is checked for zero after each incrementation.

11-0437

### Multiply State Diagram



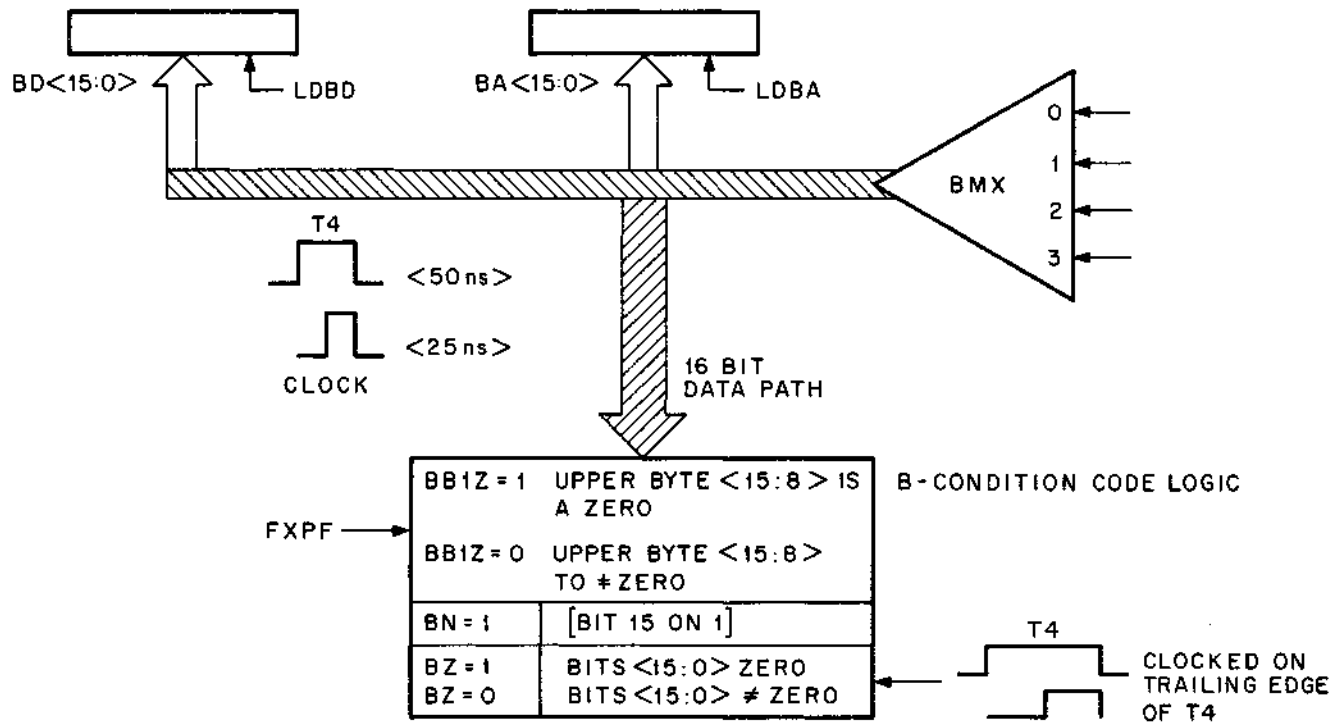
\* Three digits shown throughout state diagram refer to bits AR59, 58, and 57.  
 For example,



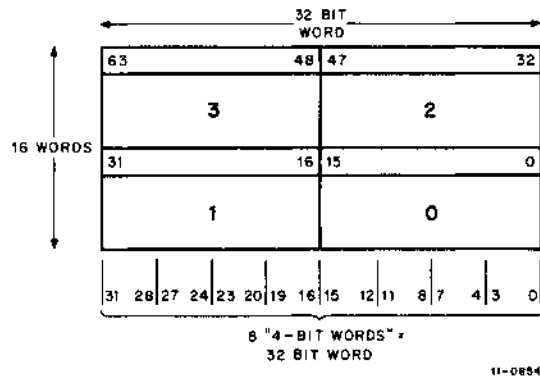
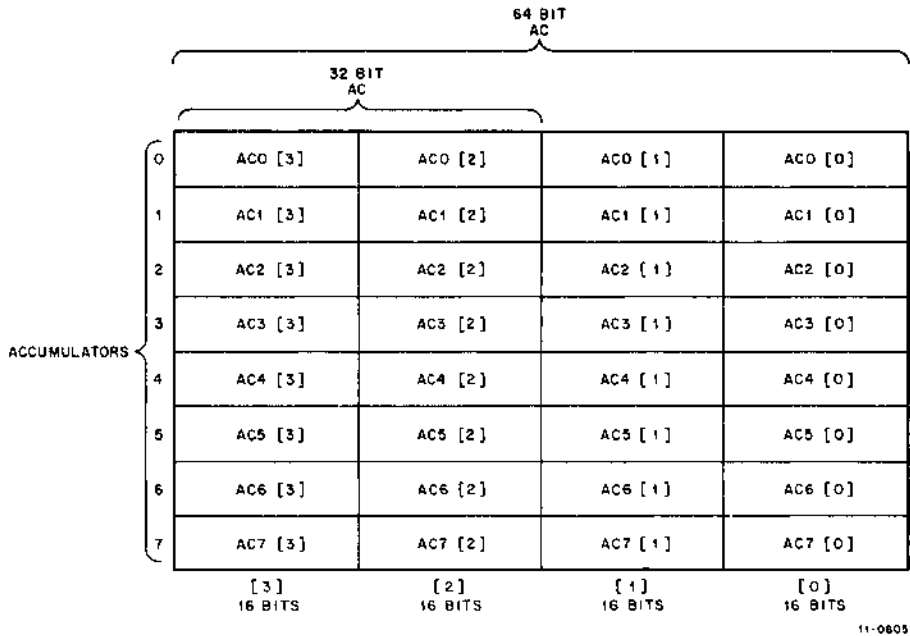
NOTE  
 BR is always positive and normalized.

11-0443

### State Diagram for Divide

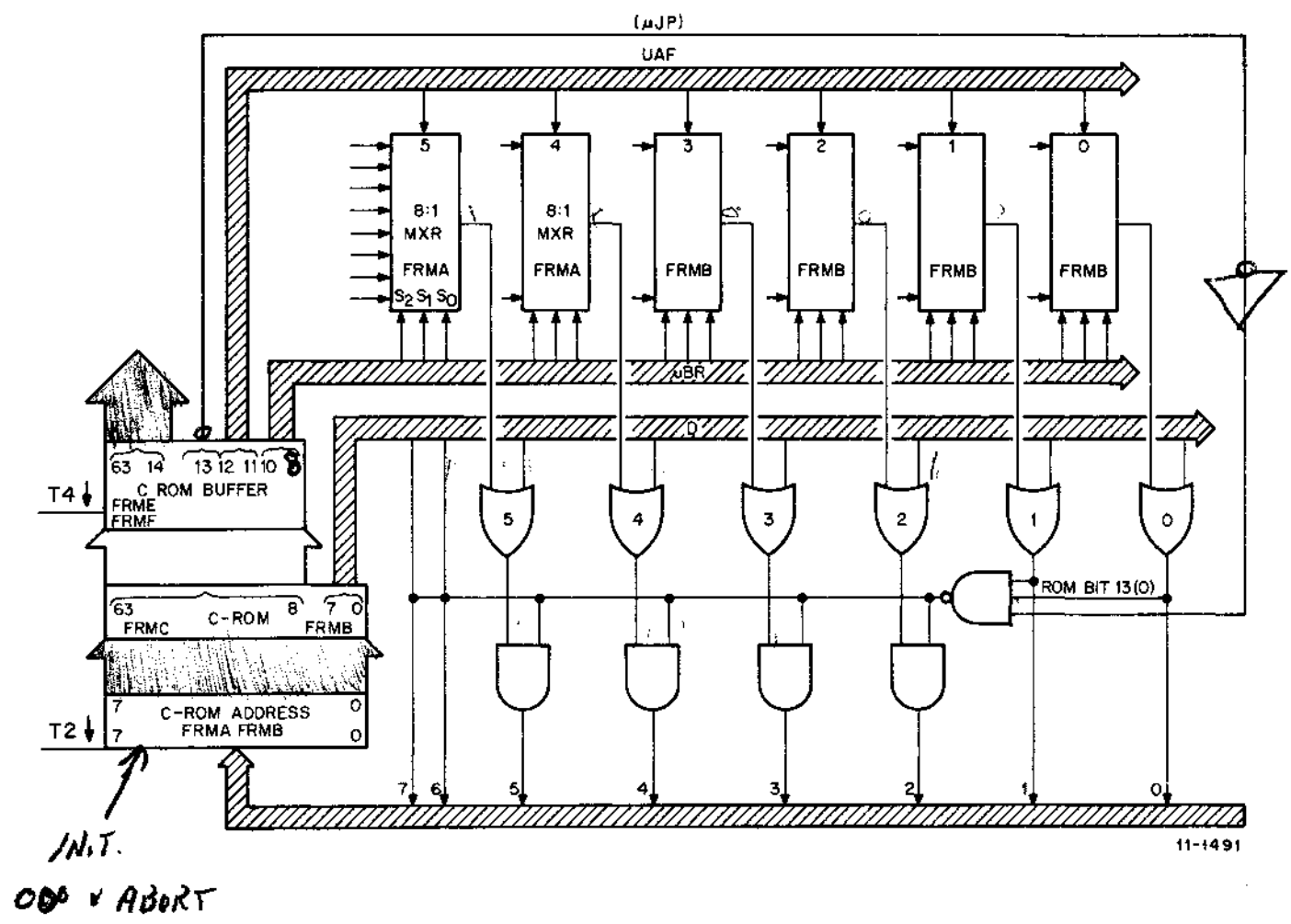


11-1490

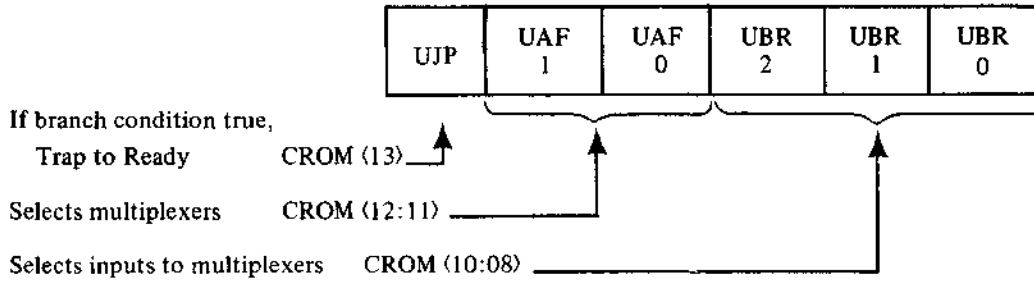


ALU Control Field (ALUC3-ALUC0)	Function	ALU Select Lines				Mode ALUM	Carry in ALUC1	
		ALUS3	ALUS2	ALUS1	ALUS0			
0	~A	0	0	0	0	1	X	
1	~(A V B)	0	0	0	1	1	X	
2	A minus B	0	1	1	0	0	0	Drive ALUS2 low
3	0	0	0	1	1	1	X	
4	~(A ^ B)	0	1	0	0	1	X	
5	~B	0	1	0	1	1	X	
6	A minus B minus 1	0	1	1	0	0	1	
7	A ^ ~B	0	1	1	1	1	X	
10	A plus B plus 1	1	0	0	1	0	0	Drive ALUS0 low
11	A plus B	1	0	0	1	0	1	
12	B	1	0	1	0	1	X	
13	A ^ B	1	0	1	1	1	X	
14	1	1	1	0	0	1	X	
15	A minus 1	1	1	1	1	0	1	Drive ALUS1 low
16	A V B	1	1	1	0	1	X	
17	A	1	1	1	1	1	X	

X = don't care  
0 = low  
1 = high



**6-Bit Branch Bits**



The three UBR bits are applied to each of the six multiplexers and uniquely specify one of the inputs to the multiplexer. If UBR bits 2, 1, and 0 are all 1s, the multiplexer output goes to 0, which indicates no modification takes place. For all other combinations, the multiplexer output goes to a 1 if the selected branch condition is true. The UAF bits specify the multiplexer(s) as follows:

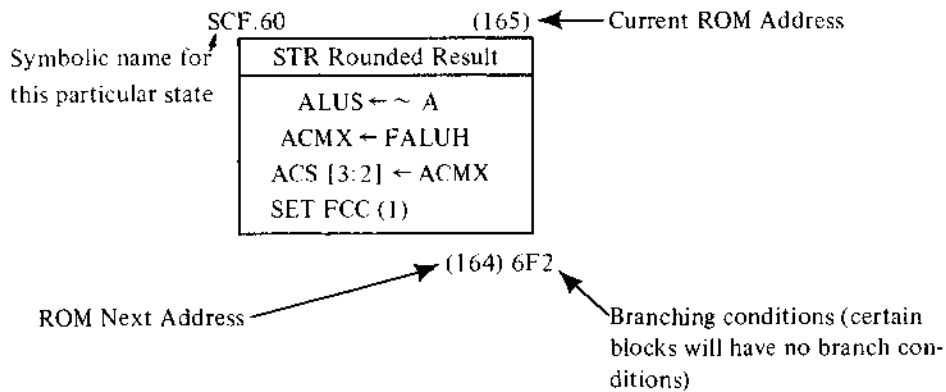
UAF1	UAF0	Multiplexers Selected
0	0	0 through 5 if UBR is even (UBR0 on a 0) 2 through 5 if UBR is odd (UBR0 on a 1)
0	1	0 Multiplexer selected
1	0	1 Multiplexer selected
1	1	Both 0 and 1 Multiplexers selected

**Multiplexer Branching Conditions**

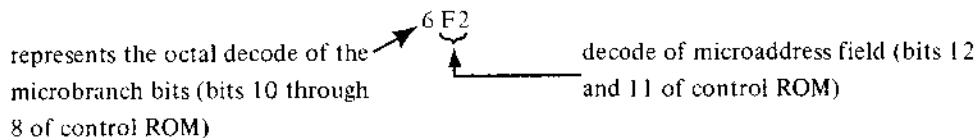
*SEL BY UBR*

	5	4	3	2	1	0	
A	SUB FRAC	FIRD4	FIRD3	FIRD2	FIRD1	FIRD0	0
B	FIR07 (1)	FIR06 (1)	FIR11 (1)	FIR10 (1)	AR50 (0)	SD (1)	1
C	RNG2	RNG1	RNG0	0	BBIZ (1)	BN (0)	2
D	0	0	0	FIU (1)	IL (0)	Immediate	3
E	0	0	0	FT (1)	~(FC A FIC)	FD (0)	4
F	FIRD6	FIRD5	0	~ CONV SP	~(FVA FIV)	M0	5
G	0	0	FIR08 (0)	AR58 (0)	AR59 (0)	BZ (1)	6
H	0	0	0	0	0	0	7

Multiplexer Inputs



The branching conditions are designated as follows:





<b>FRL</b>	Fraction Data Path Low Order	M8115-0-01
<b>FRH</b>	Fraction Data Path High Order	M8114-0-01
<b>FRM</b>	FP ROM and ROM Control	M8112-0-01
<b>FXP</b>	Floating-Point Exponent Data Path	M8113-0-01

The **FRL** group of prints contains the following logic:

1. lower half of **FALU**
2. lower half of **AR**
3. lower half of **BR**
4. lower half of **QR**
5. floating-point status
6. **ACMX**
7. scratch pad (**AC7-0**)
8. **BMX**

The **FRH** group of prints contains the following logic:

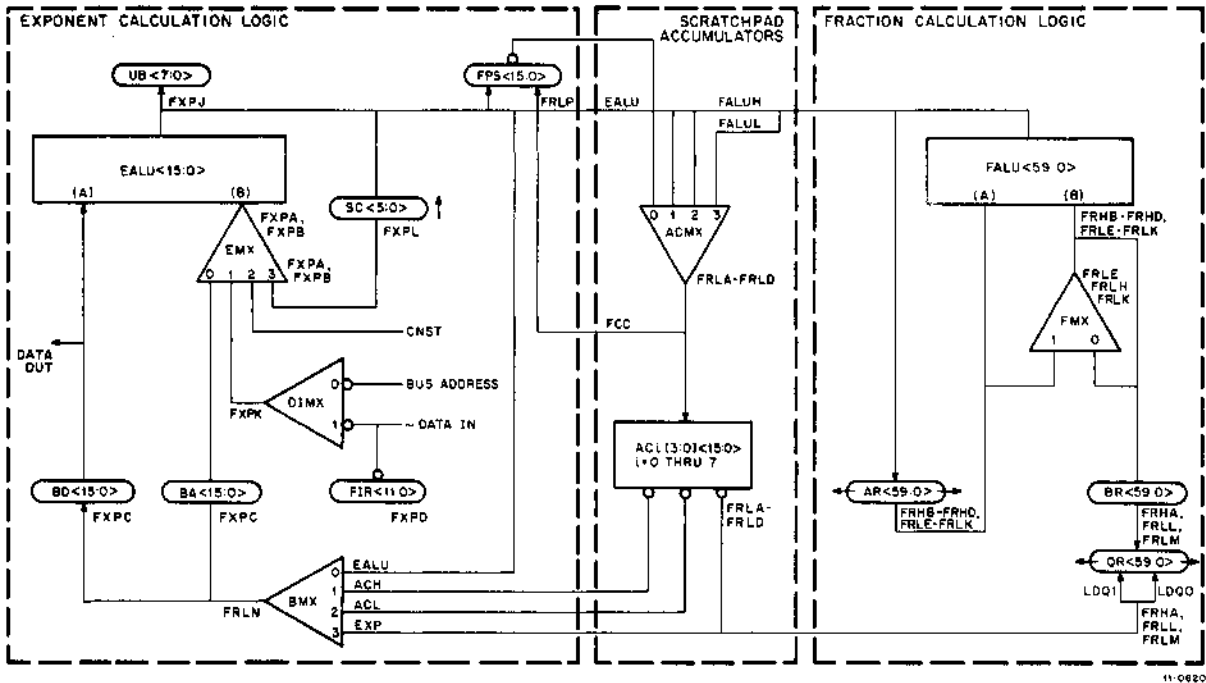
1. upper half of **FALU**
2. upper half of **AR**
3. upper half of **BR**
4. upper half of **QR**
5. clock logic, times states, time pulses
6. sign of source (**SS**) and sign of destination (**SD**) logic
7. fractional control logic

The **FRM** group of prints contains the following logic:

1. control ROM
2. control ROM address register
3. Scratch pad addressing logic
4. ROM multiplexers
5. ROM data buffer
6. interface logic

The **FXP** group of prints contains the following logic:

1. **EALU**
  2. **EMX**
  3. Step counter
  4. **FIR**
  5. **BA** register
  6. **BD** register
  7. **U** break register
  8. **DIMX**
  9. **BR**anching Logic
  10. Range ROM
  11. **FRHE**
1. **MRI** and **MRO** register
  2. **MUL ARITH** flip-flop
  3. Pause logic
  4. **STRG I** flip-flop
  5. **AR** control
  6. **QR** control
  7. **MUL SUB** flip-flop
  8. **AR** clock logic
  9. **QR** clock logic
  10. Sign bit



### DATA PATH DEFINITION

$ACMX0 \langle 31 \rangle \leftarrow \sim BN$ ;  $ACMX0 \langle 30 \rangle \leftarrow BZ$ ;  $ACMX0 \langle 29:16 \rangle \leftarrow 37777$ ;  $ACMX0 \langle 15:0 \rangle \leftarrow FPS$   
 $ACMX1 \langle 31:16 \rangle \leftarrow EALU \langle 15:00 \rangle$ ;  $ACMX1 \langle 15:00 \rangle \leftarrow EALU \langle 15:00 \rangle$   
 $ACMX2 \langle 31 \rangle \leftarrow \sim SD$ ;  $ACMX2 \langle 30:23 \rangle \leftarrow EALU \langle 07:00 \rangle$ ;  $ACMX2 \langle 22:00 \rangle \leftarrow FALU \langle 57:35 \rangle$   
 $ACMX3 \langle 31:00 \rangle \leftarrow FALU \langle 34:03 \rangle$

$BMX0 \langle 15:00 \rangle \leftarrow EALU \langle 15:00 \rangle$   
 $BMX1 \langle 15:00 \rangle \leftarrow AC_i \langle 3 \rangle \langle 15:00 \rangle$  or  $AC_i \langle 1 \rangle \langle 15:00 \rangle$   
 $BMX2 \langle 15:00 \rangle \leftarrow AC_i \langle 2 \rangle \langle 15:00 \rangle$  or  $AC_i \langle 0 \rangle \langle 15:00 \rangle$   
 $BMX3 \langle 15:08 \rangle \leftarrow 0$ ;  $BMX3 \langle 07:00 \rangle \leftarrow AC_i \langle 3:2 \rangle \langle 30:23 \rangle$  or  $AC_i \langle 01:0 \rangle \langle 30:23 \rangle$

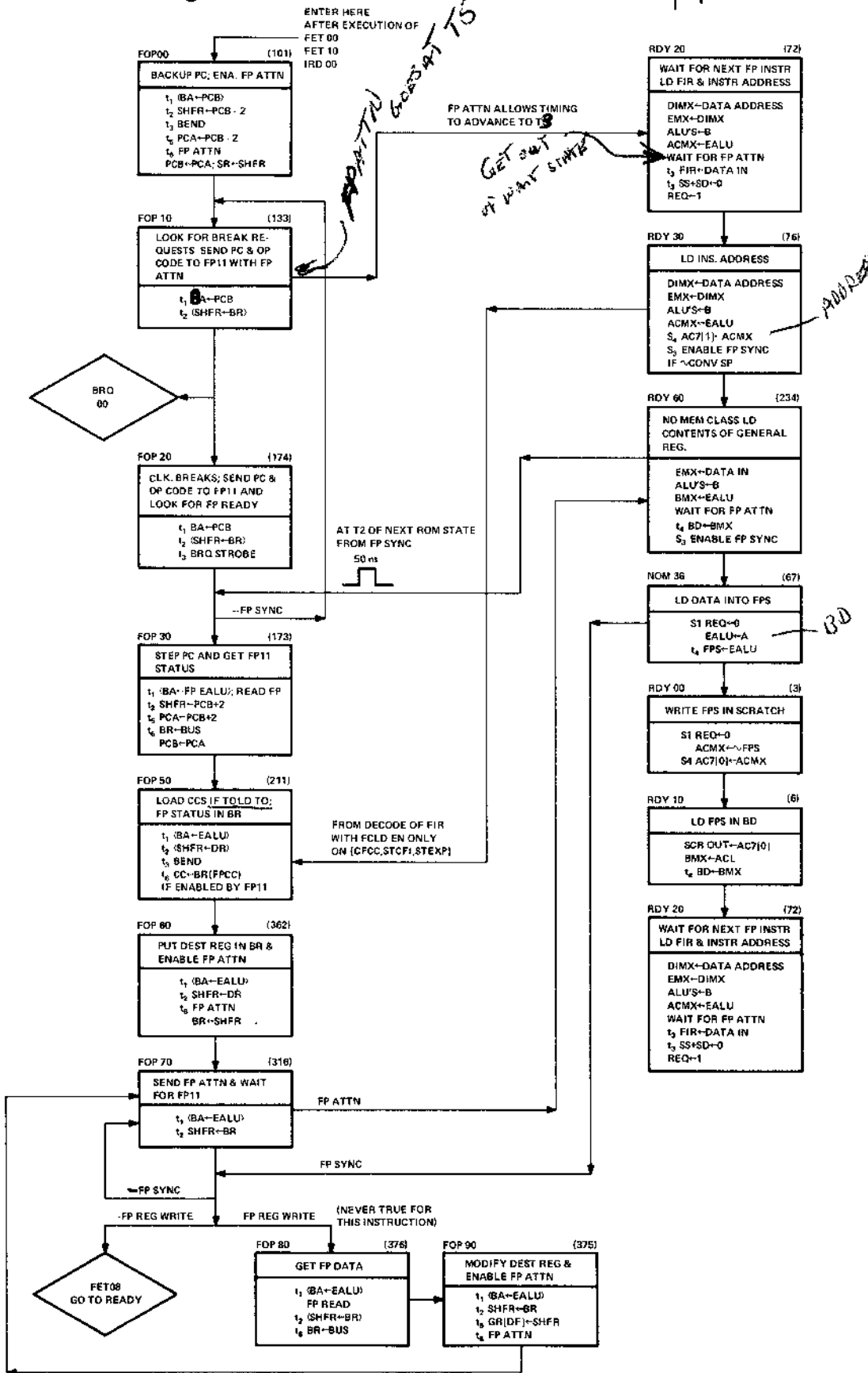
$EMX0 \langle 15:00 \rangle \leftarrow BA \langle 15:00 \rangle$   
 $EMX1 \langle 15:00 \rangle \leftarrow DIMX \langle 15:00 \rangle$   
 $EMX2 \langle 15:00 \rangle \leftarrow CNST \langle 15:00 \rangle$   
 $EMX3 \langle 15:06 \rangle \leftarrow 0$ ;  $EMX3 \langle 05:00 \rangle \leftarrow SC \langle 05:00 \rangle$

$FMX0 \langle 02 \rangle \leftarrow BR \langle 35 \rangle$ ;  $FMX0 \langle 01 \rangle \leftarrow BR \langle 19 \rangle$ ;  $FMX0 \langle 00 \rangle \leftarrow BR \langle 3 \rangle$   
 $FMX1 \langle 02 \rangle \leftarrow AR \langle 34 \rangle$ ;  $FMX1 \langle 01 \rangle \leftarrow 1$ ;  $FMX1 \langle 00 \rangle \leftarrow AR \langle 02 \rangle$

$LDQ1 = QR \langle 59 \rangle \leftarrow 0$ ;  $QR \langle 58 \rangle \leftarrow 1$  if  $AC_i \langle 3:2 \rangle \langle 30:24 \rangle \neq 0$  else  $QR \langle 58 \rangle \leftarrow 0$   
 $QR \langle 57:35 \rangle \leftarrow AC_i \langle 3:2 \rangle \langle 22:0 \rangle$   
 $LDQ0 = QR \langle 34:3 \rangle \leftarrow AC_i \langle 1:0 \rangle \langle 31:0 \rangle$ ;  $QR \langle 2:0 \rangle \leftarrow 0$

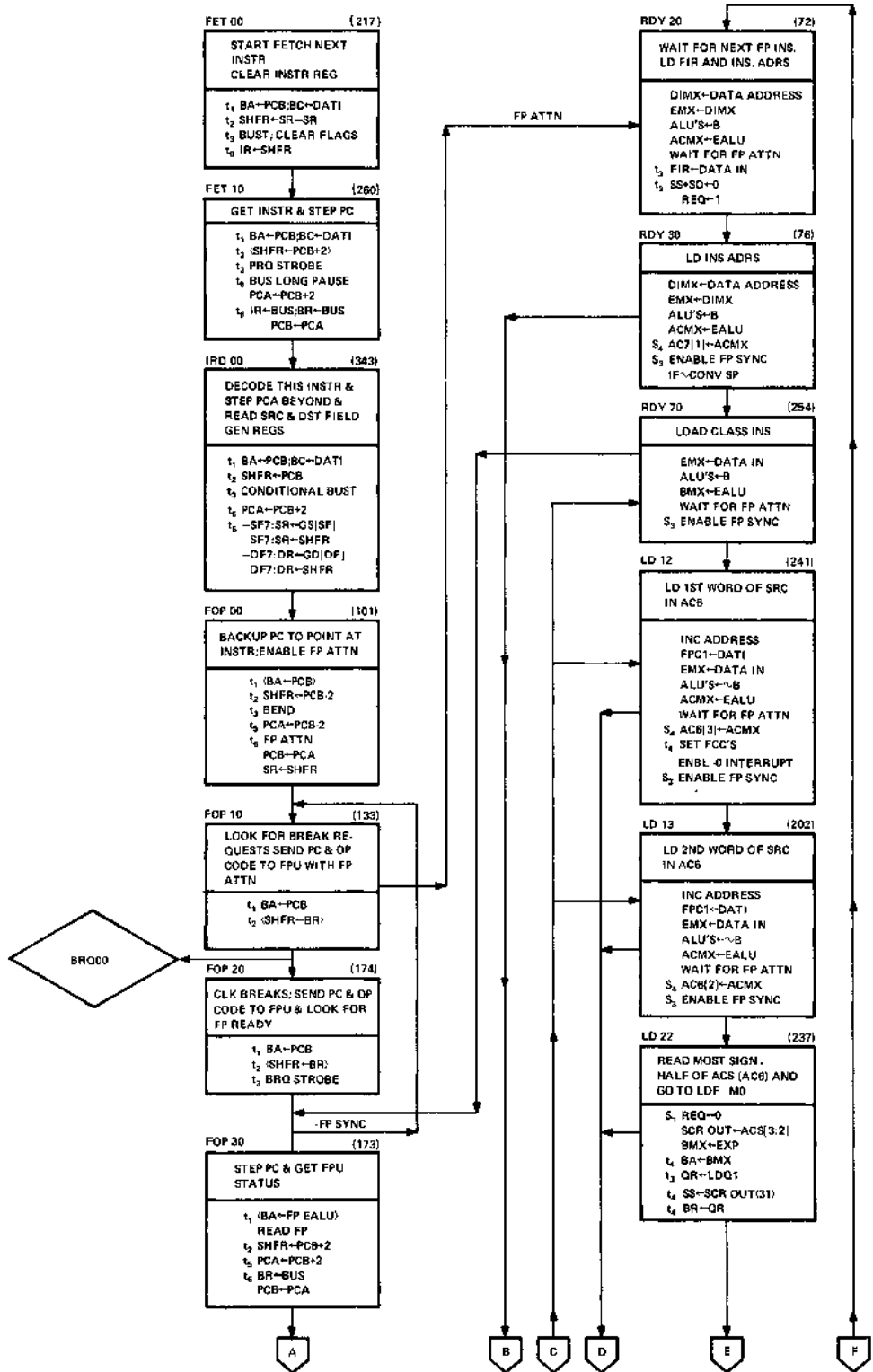
*ADFPS MODE 0  
CP Flow*

*FP Flow*



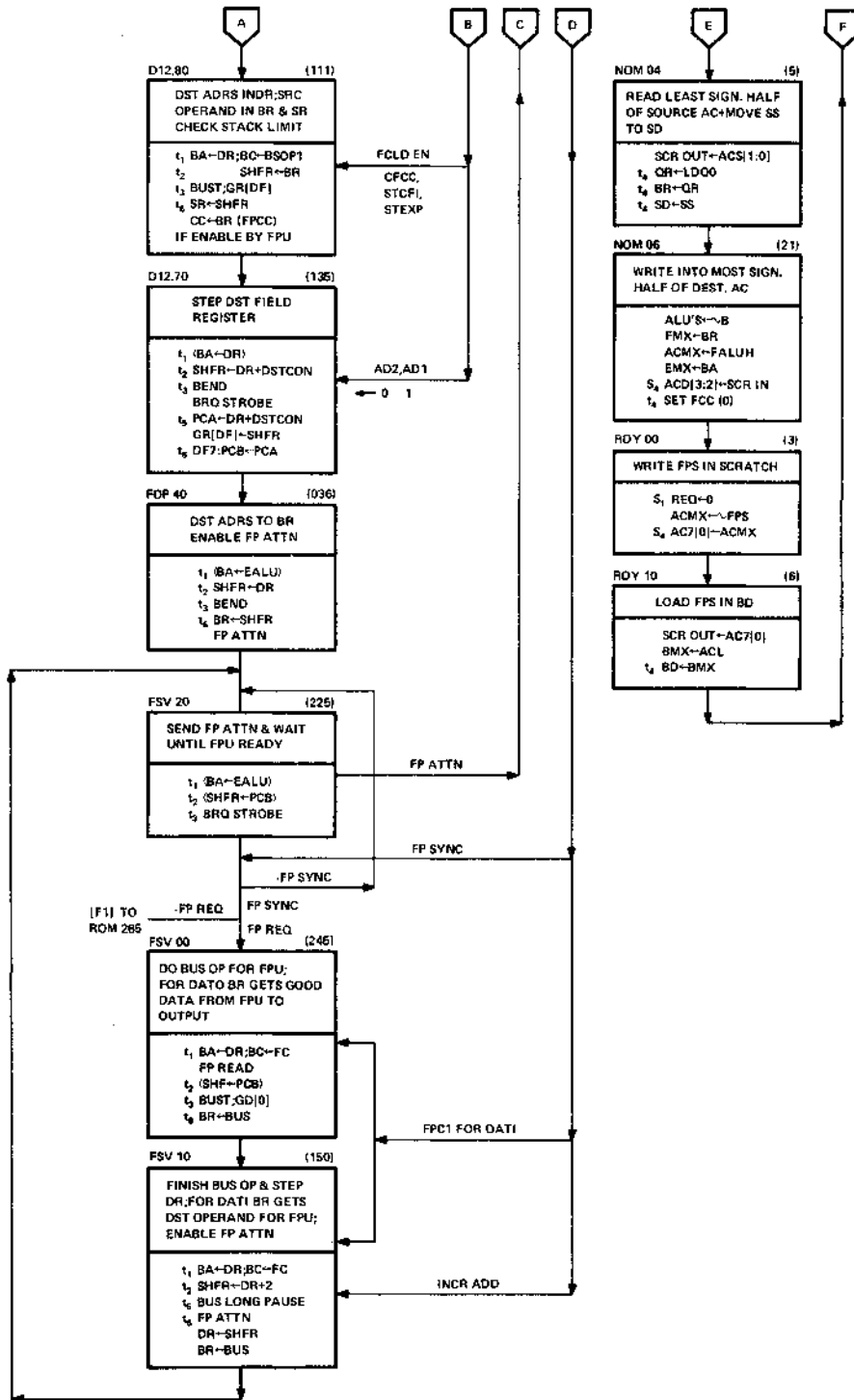
CP/FPP Interflow Mode 0

11-1443



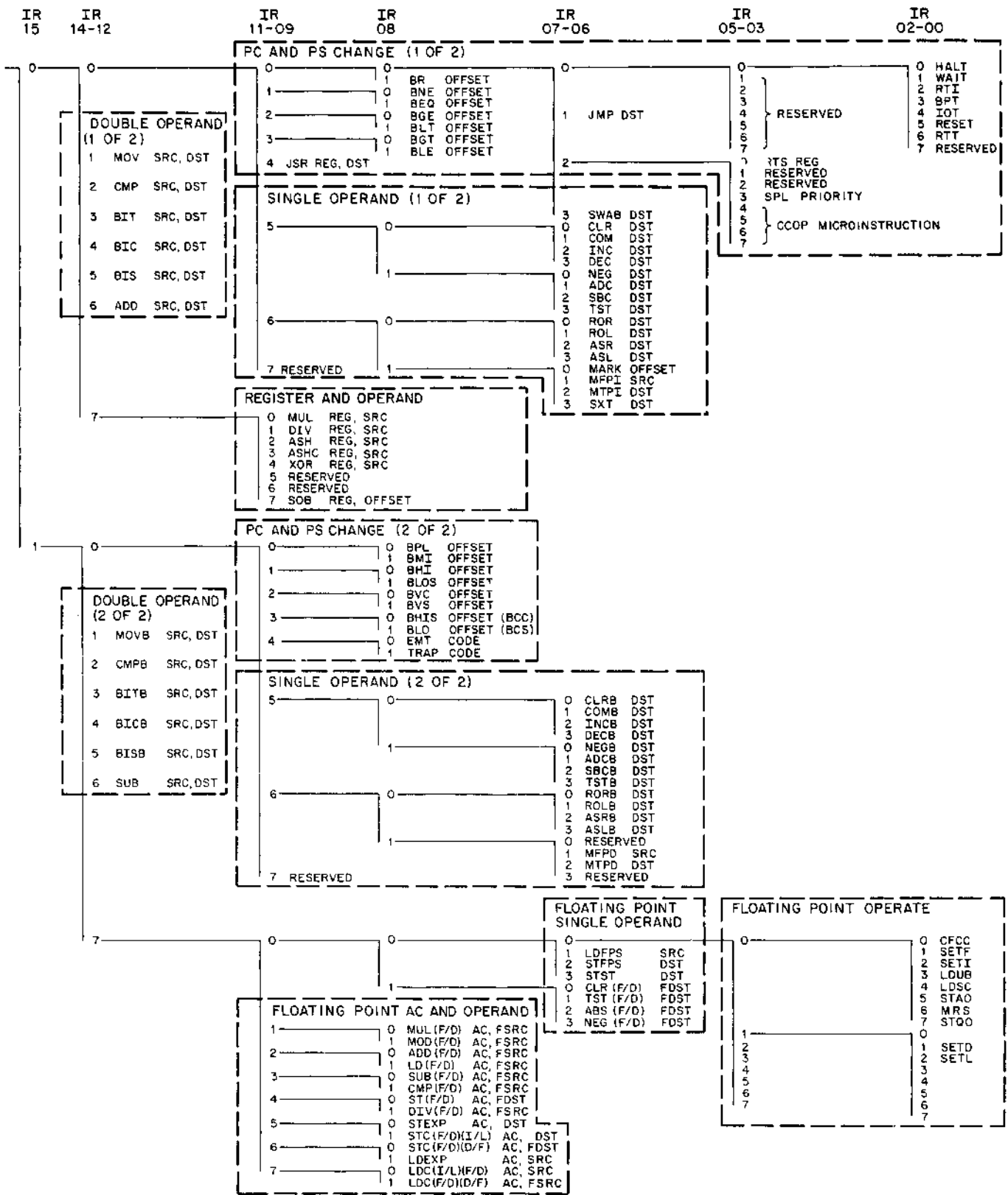
11-1442-A

CP/FPP Interflow Mode 2 (sheet 1 of 2)

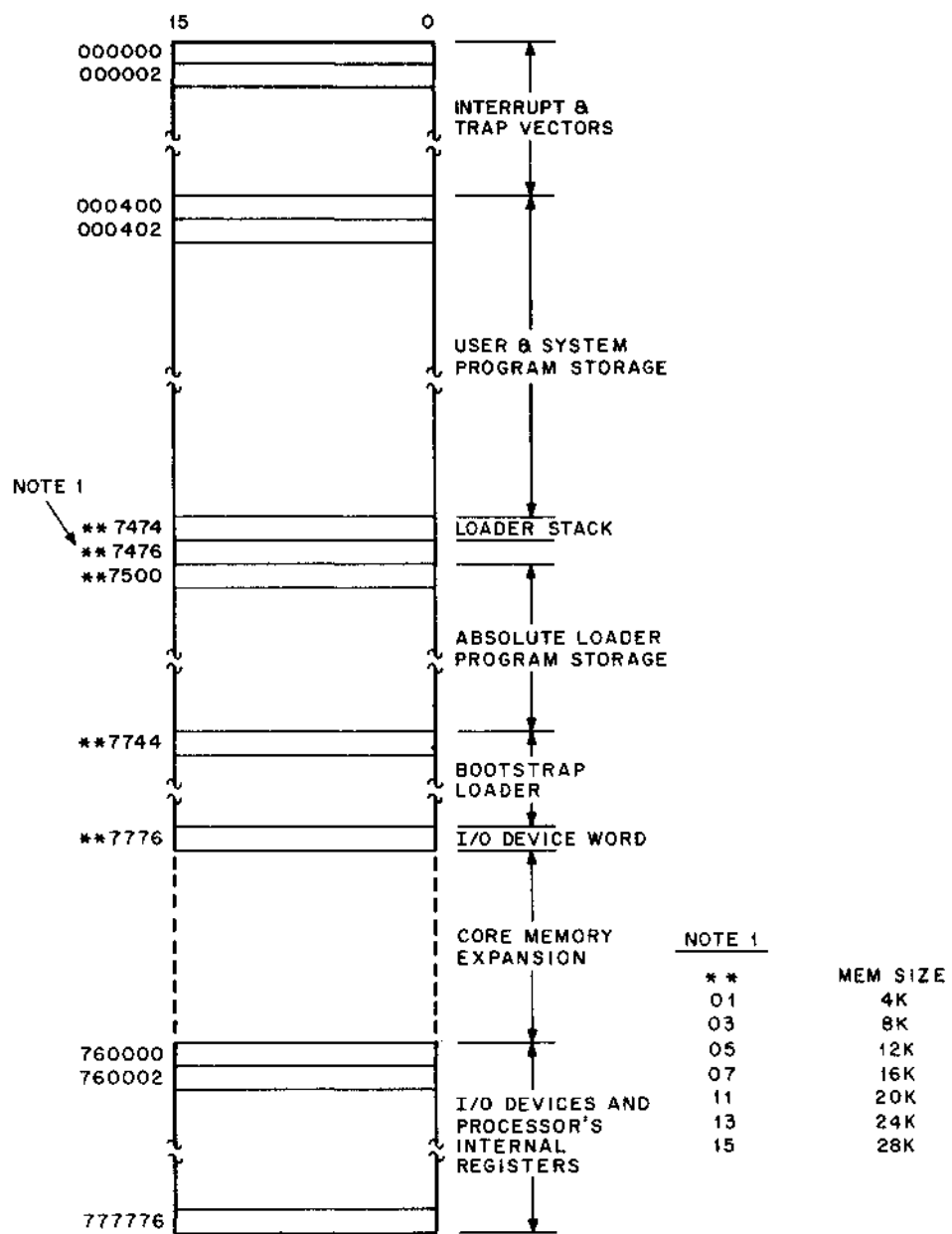


11-1442-8

CP/FPP Interflow Mode 2 (sheet 2 of 2)

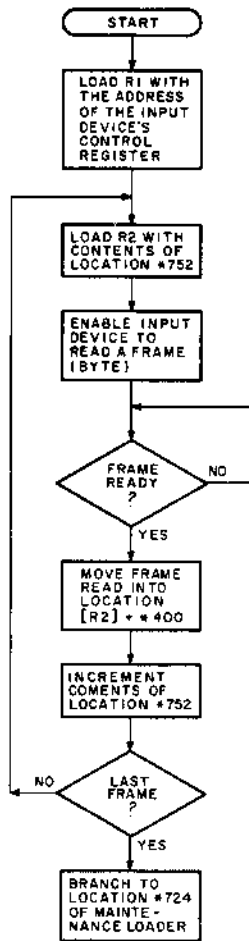


Determination of an Instruction from the Binary Code



11-1492

PDP-11 Typical Core Memory Storage Map



11-1493

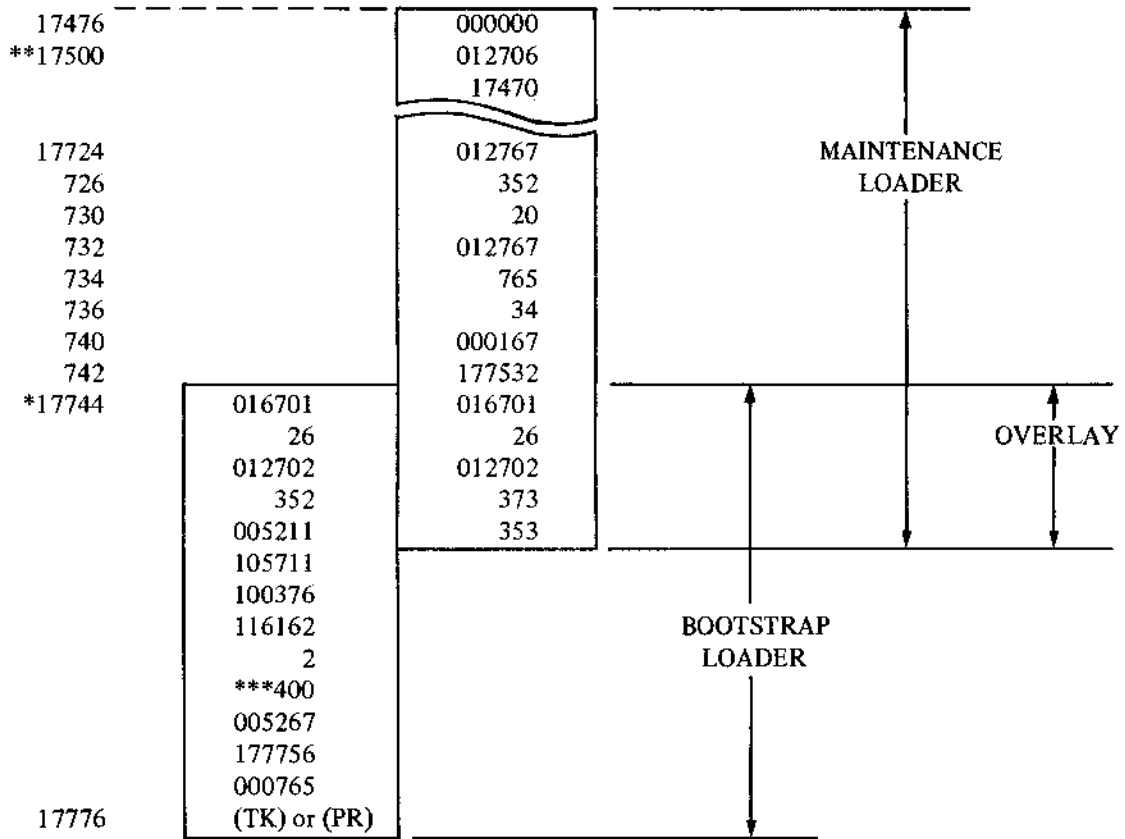
### Bootstrap Loader, Flow Chart

#### Bootstrap Loader Coding

Location	Octal	Symbolic
*744	016701	MOV *776, %1
746	26	
750	012702	MOV #352, %2
752	352	
754	005211	INC (1)
756	105711	TSTB (1)
760	100376	BPL -2
762	116162	MOVB 2(1), *400 (2)
764	2	
766	*400	
770	005267	INC *752
772	177756	
774	000765	BR. -24
*776	177560	(TK)
	or, 177550	(PR)

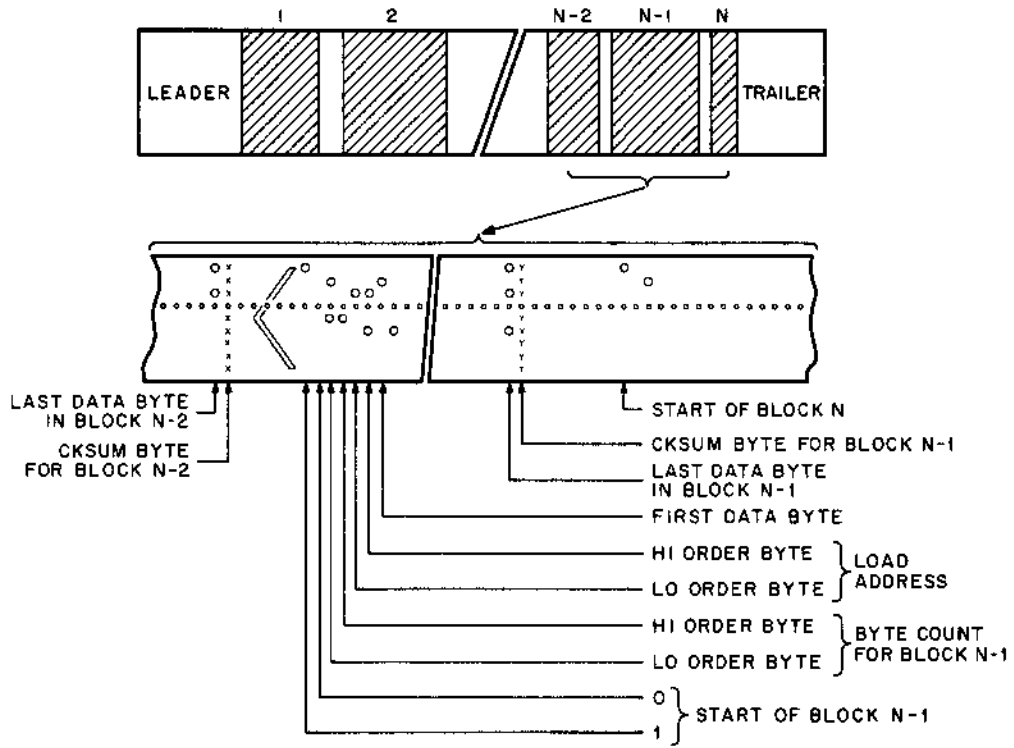
* = 17	for	4K
37		8K
57		12K
77		16K
117		20K
137		24K
157	for	28K





TK = 177560    Low-Speed Reader  
 PR = 177550    High-Speed Reader

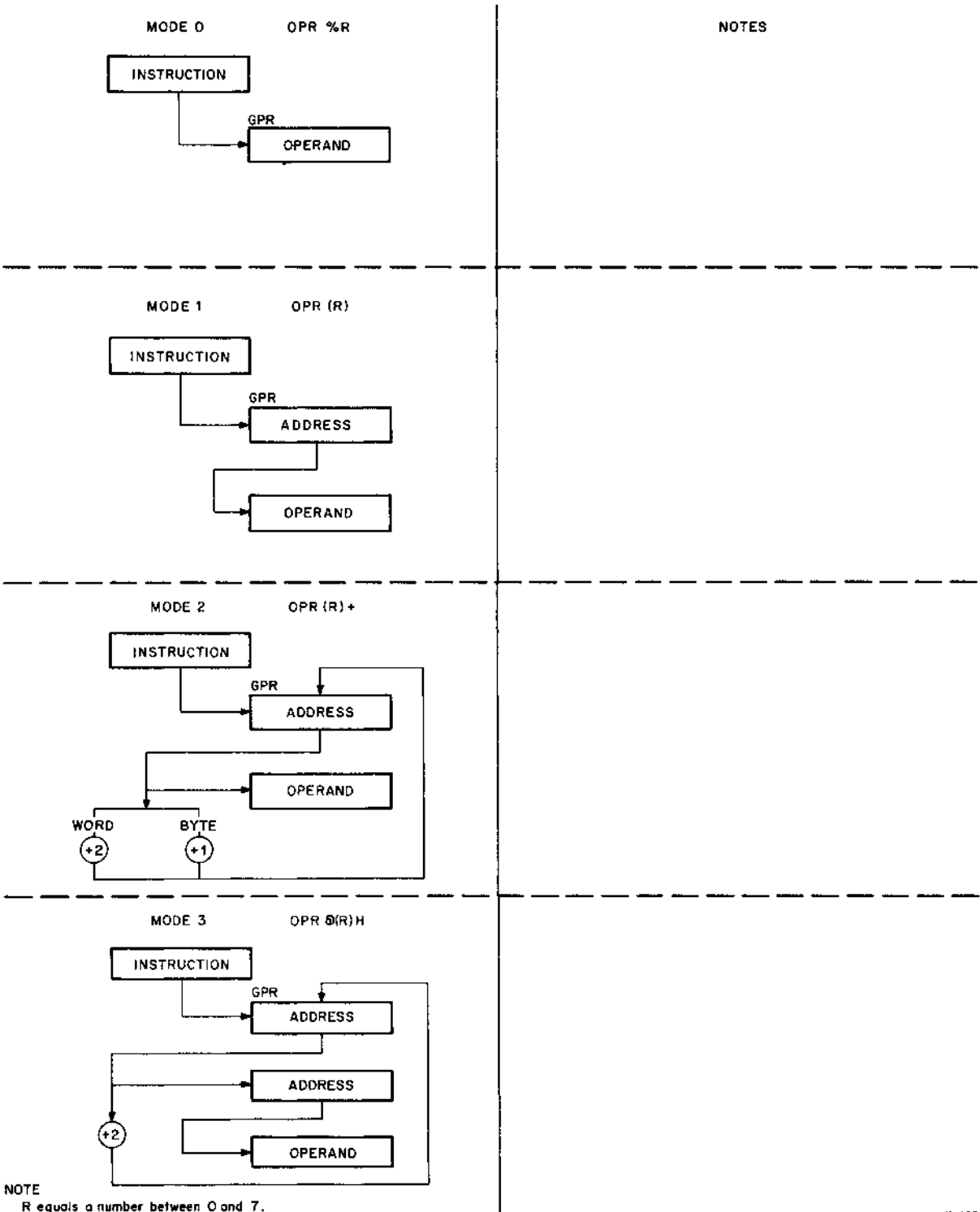
\*Starting address of the Bootstrap Loader  
 \*\*Starting address of the Maintenance Loader



11-1461

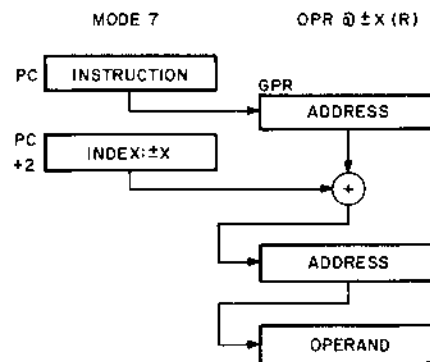
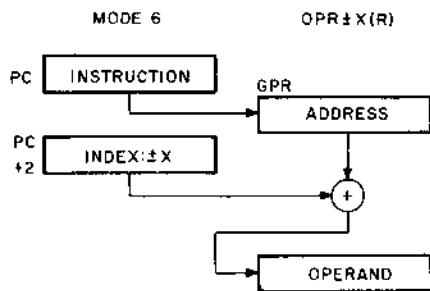
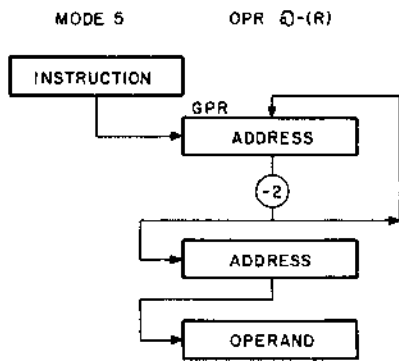
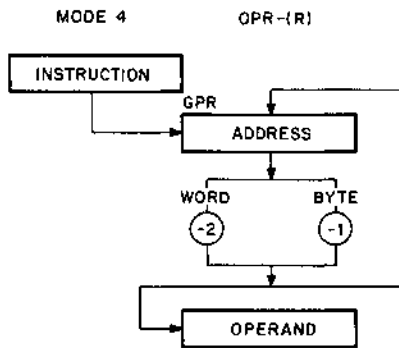
**Absolute Loader Tape Format**

# ADDRESSING MODES

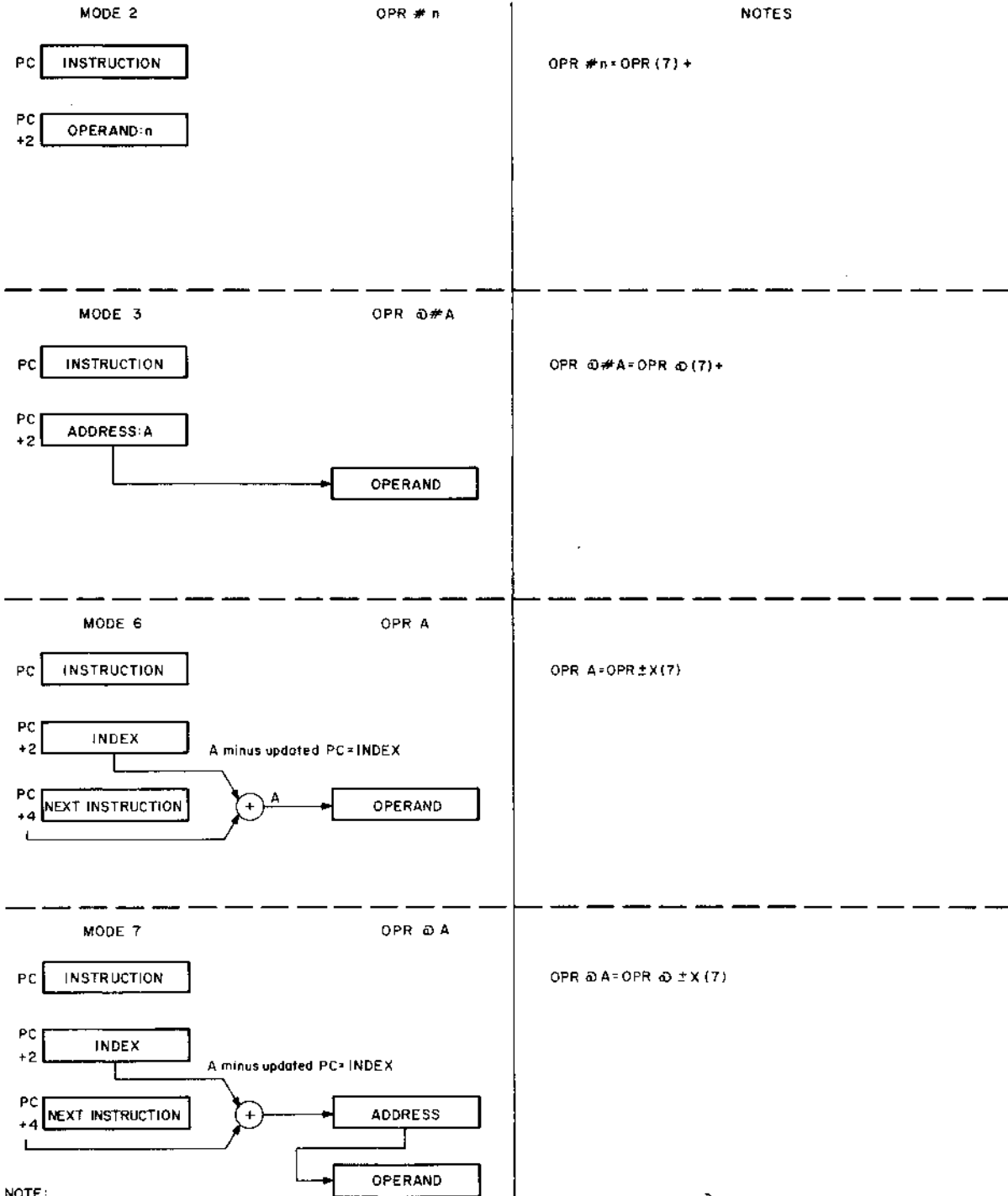


NOTE  
R equals a number between 0 and 7.

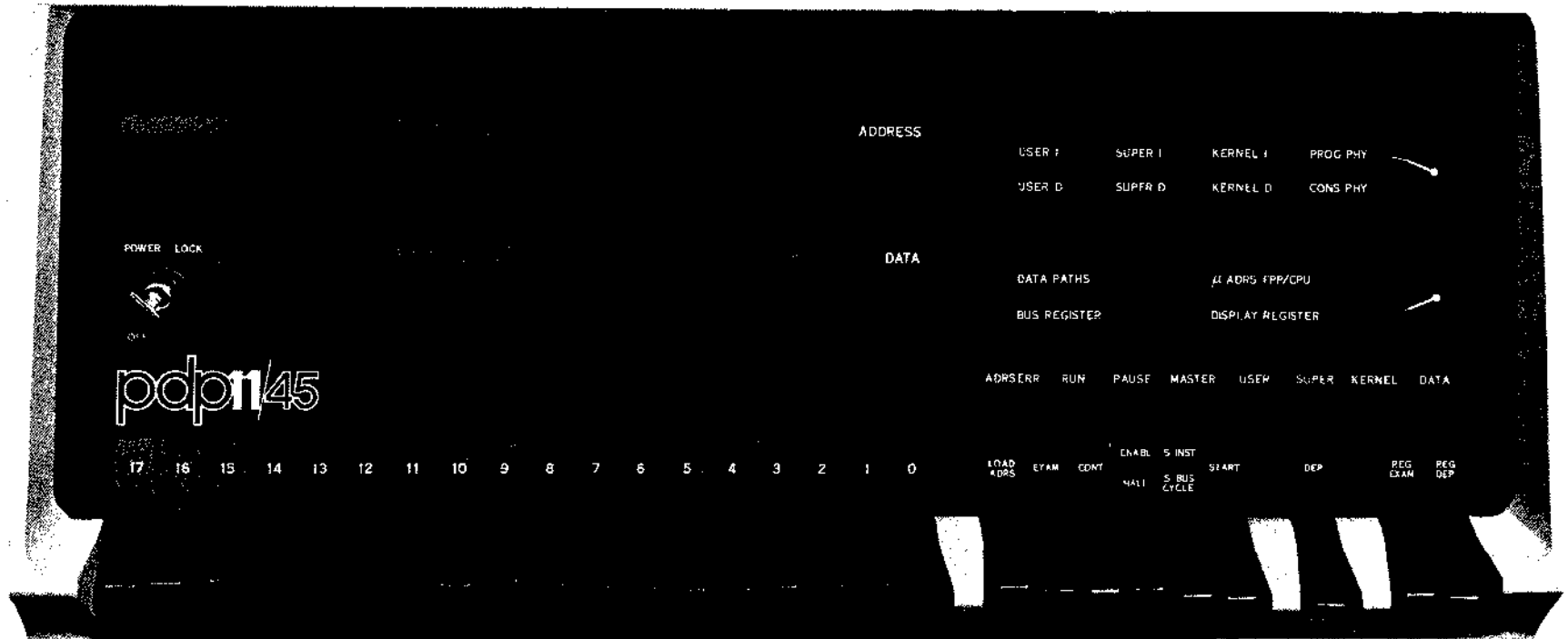
11-1494



PC REGISTER ADDRESSING



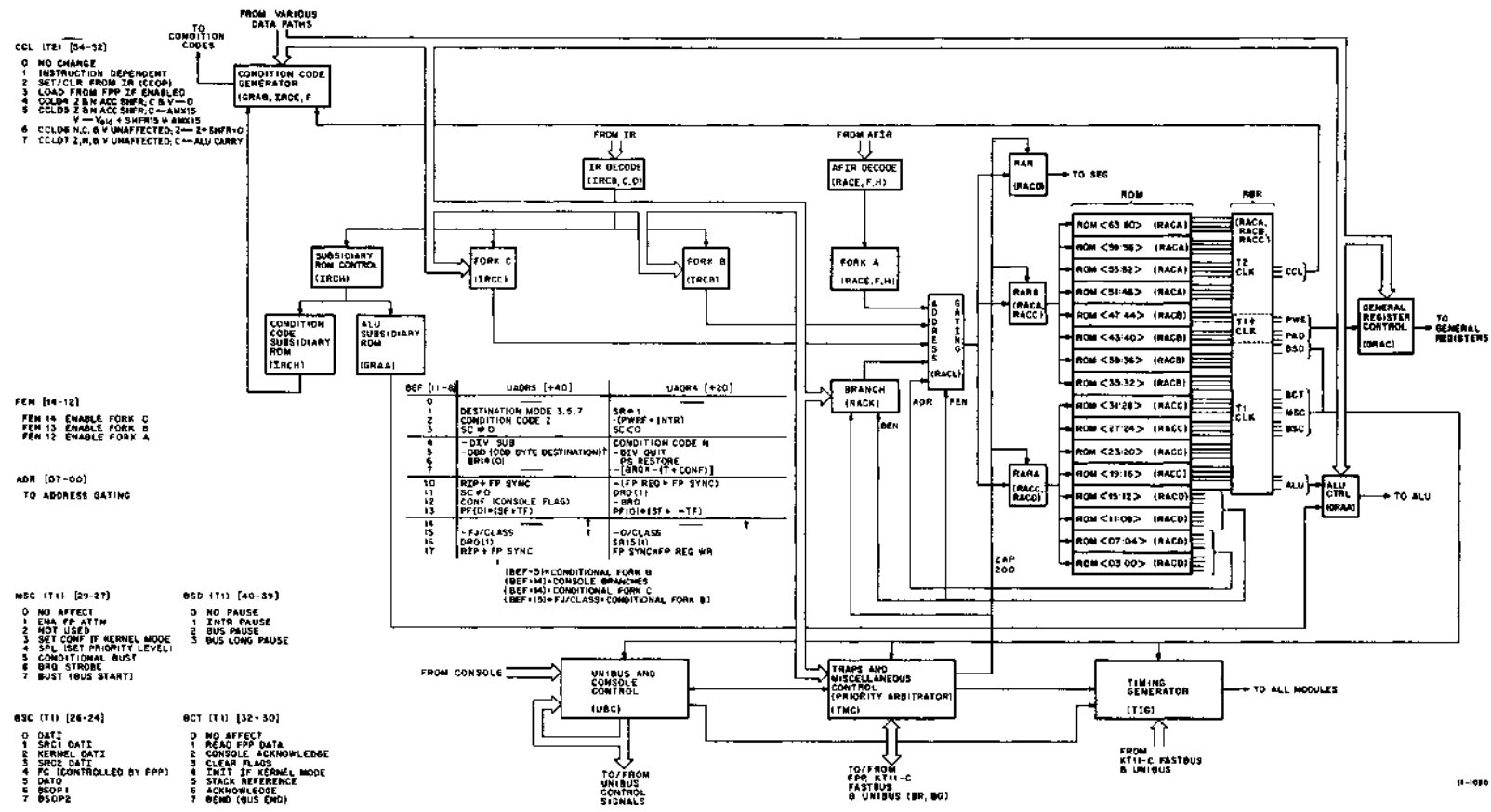
NOTE: The mode specified overrides the fact that the register is the PC (register 7). EXAMPLE: 500/CLR-(7) } Program causes halt at address 500 whose  
 502/777 } content has been offered to = 0s  
 504/400 }  
 506/HALT }



PDP-11/45 Operator's Console

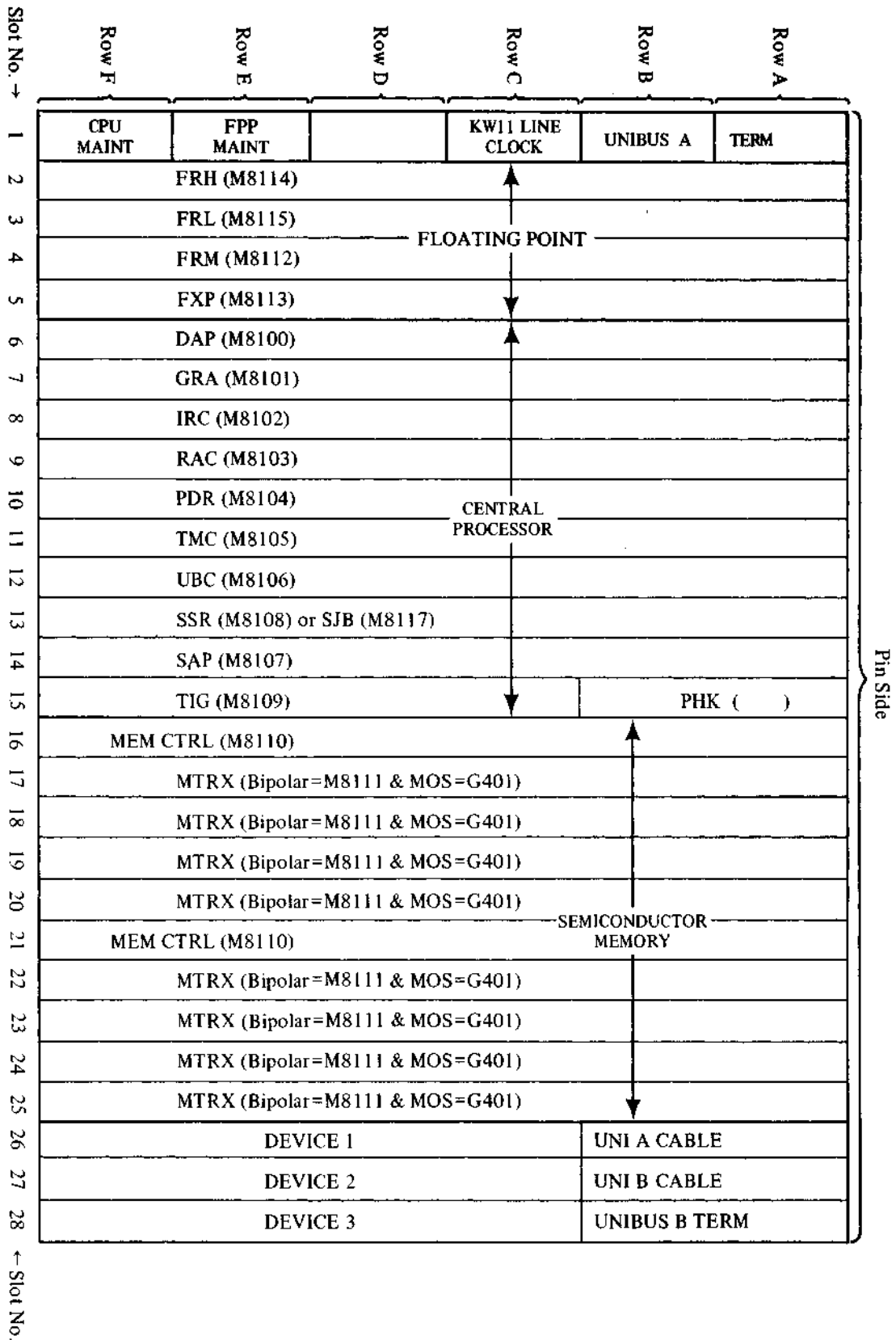


59



KB11-A Central Processor Control Section, Block Diagram





Module Layout

**DEVICE REGISTER ADDRESSES**

<b>Device</b>	<b>CSR</b>	<b>DBR</b>	<b>Vector</b>
Teletype Keyboard	777560	777562	60 BR4
Teletype Printer	777564	777566	64 BR4
Reader (PC11)	777550	777552	70 BR4
Punch (PC11)	777554	777556	74 BR4
Line Clock (KW11-L)	777546	-	100 BR6
Line Printer (LP11)	777514	777516	200 BR4
DECtape (TC11/TU56)	777340	777350	214 BR5
Control	777342		
Word Count	777344		
Current Address	777346		
DECdisk (RC11/RS64)	777444	777456	210 BR5
Look Ahead	777440		
Disk Address	777442		
Word Count	777450		
Current Address	777452		
Maintenance	777454		
DECdisk (RF11/RS11)	777460	777472	204 BR5
Word Count	777462		
Current Address	777464		
Disk Address	777466		
Disk Address Extended	777470		
Maintenance	777476		
DEC Disk Pack (RP11/RS03)	776714	-	254 BR5
Word Count	776716		
Current Address	776720		
Disk Cylinder Address	776722		
Disk Address	776724		
Device Status	776710		
Error Register	776712		
Maintenance Registers	776726		
	776730		
	776732		
Card Reader (CR11/CM11)	777160	777162 777164	
Magnetic Tape (TM11/TU10)	772522	772530	
Byte Count	772524		
Current Address	772526		
Status	772520		
Device Interface (DR11)	772414	772416	
Word Count	772410		
Current Address	772412		

ASCII 7-Bit Octal Code	Char	ASCII 7-Bit Octal Code	Char.	ASCII 7-Bit Octal Code	Char.	ASCII 7-Bit Octal Code	Char.
000	NUL	040	SP	100	@	140	'
001	SOH	041	!	101	A	141	a
002	STX	042	-	102	B	142	b
003	ETX	043	#	103	C	143	c
004	EOT	044	\$	104	D	144	d
005	ENQ	045	%	105	E	145	e
006	ACK	046	&	106	F	146	f
007	BEL	047	'	107	G	147	g
010	BS	050	(	110	H	150	h
011	HT	051	)	111	I	151	i
012	LF	052	·	112	J	152	j
013	VT	053	+	113	K	153	k
014	FF	054	,	114	L	154	l
015	CR	055	-	115	M	155	m
016	SO	056	.	116	N	156	n
017	SI	057	/	117	O	157	o
020	DLE	060	0	120	P	160	p
021	DC1	061	1	121	Q	161	q
022	DC2	062	2	122	R	162	r
023	DC3	063	3	123	S	163	s
024	DC4	064	4	124	T	164	t
025	NAK	065	5	125	U	165	u
026	SYN	066	6	126	V	166	v
027	ETB	067	7	127	W	167	w
030	CAN	070	8	130	X	170	x
031	EM	071	9	131	Y	171	y
032	SUB	072	:	132	Z	172	z
033	ESC	073	;	133	[	173	(
034	FS	074	<	134	\	174	
035	GS	075	=	135	]	175	)
036	RS	076	>	136	↑	176	~
037	US	077	?	137	←	177	DEL

**digital**

**DIGITAL EQUIPMENT CORPORATION  
MAYNARD, MASSACHUSETTS 01754**