COMPANY PRIVATE

INSTRUCTION SET
BREADBOARD MODEL

AUGUST 11, 1969
UPDATED 9-10-69

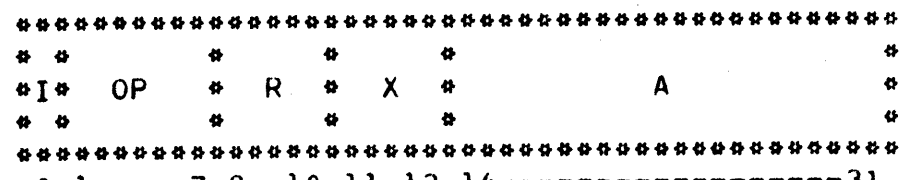
Copy #41

SECTION I

FORMATS AND DESCRIPTIONS

1. INSTRUCTION FORMATS AND ADDRESSING TYPES
2. REGISTERS
3. ADDRESS SPACE
4. DATA FORMATS
5. STACK MARKER WORD FORMATS
6. PROGRAM LABEL FORMAT
7. DATA LABEL FORMATS
8. TABLES
9. EXTERNAL DIRECTORY
10. PROGRAM SEGMENT TABLE (PST)
11. DATA SEGMENT TABLE (DST)
12. INDIRECT ADDRESSING AND ARRAY REFERENCES
13. I/O PROGRAMMING
14. I/O DEVICE STATUS TABLE
15. INTERRUPT TABLE
16. INTERRUPT PROCESSING
17. TRAP PROCESSING
18. BUS FORMATS
19. CONDITION CODE PATTERNS
20. MODES OF OPERATION

1. INSTRUCTION FORMATS AND ADDRESSING TYPES



I = INDIRECT BIT

OP = OPERATION CODE (0-127)

R = REGISTER REFERENCE (FIRST OPERAND)

X = REGISTER REFERENCE (INDEX REGISTER)

A = ADDRESS FIELD

MEMORY ADDRESS (M) COMPUTATION:

THE MEMORY ADDRESS SPECIFIED IN AN INSTRUCTION CONSISTS OF 3 PARTS: THE A FIELD, I FIELD, AND X FIELD. IN GENERAL, THE A FIELD SPECIFIES A DIRECT ADDRESS, THE I FIELD (1 BIT) INDICATES INDIRECT ADDRESSING, AND THE X FIELD INDICATES AN INDEX REGISTER WHOSE VALUE IS TO BE ADDED. THE DEFINITION OF THE A FIELD IS AS FOLLOWS:

TYPE	CODING	DIRECT ADDRESS	RANGE
DB+	A[14-15]=00	DB+A[16-31]	[DB,DB+65535]
IMMEDIATE	A[14-15]=01	(SEE EXPLANATION ON NEXT PAGE)	
PB+	A[14-15]=10	PB+A[18-31]	[PB,PB+16383]
R	A[14-17]=1100	A[29-31]	[R0-R7]
DST/PST	A[14-17]=1101	A[20-31]	[0,4095]
Q+	A[14-18]=11100	Q+A[19-31]	[Q,Q+8181]
Q-	A[14-18]=11101	Q-2↑12+A[19-31]	[Q-1,Q-8192]
S+	A[14-18]=11110	S+A[19-31]	[S,S+8191]
S-	A[14-18]=11111	S-2↑12+A[19-31]	[S-1,S-8192]

NOTE: WHEN DST/PST - TYPE ADDRESSING IS USED, BRANCH AND CALL INSTRUCTIONS INDEX INTO THE PST, AND ALL OTHER INSTRUCTIONS INDEX INTO THE DST.

IF I=0 AND X#0, THE CONTENT OF X IS ADDED TO THE ADDRESS TO PRODUCE A FINAL ADDRESS. IN DOUBLEWORD INSTRUCTIONS (X) IS DOUBLED BEFORE BEING ADDED, AND IN THE BYTE INSTRUCTIONS LDB AND STB IT IS QUARTERED.

IN CERTAIN INSTRUCTIONS, IF R-TYPE ADDRESSING IS USED AND I=0, THE EFFECTIVE ADDRESS IS GIVEN BY THE CONTENT OF REGISTER R, WHICH MUST BE A PROGRAM OR DATA LABEL (TO BE DESCRIBED BELOW). THESE INSTRUCTIONS ARE:

LDB	BRE	BRNN	PPDS	PCAL
STB	BRP	BRZ	LAR	ECAL
BCC	BRN	BRNZ	LBAR	SIO
BRO	BRNP	PSDS	LAS	

MEANING OF IMMEDIATE FIELD:

IF THE INSTRUCTION USES FULL-WORD OPERANDS, THE I BIT OF THE INSTRUCTION IS PLACED IN BITS 0-15 OF THE OPERAND, AND BITS 16-31 OF THE INSTRUCTION BECOME BITS 16-31 OF THE OPERAND. THE RANGE OF OPERANDS IS [-65536,+65535].

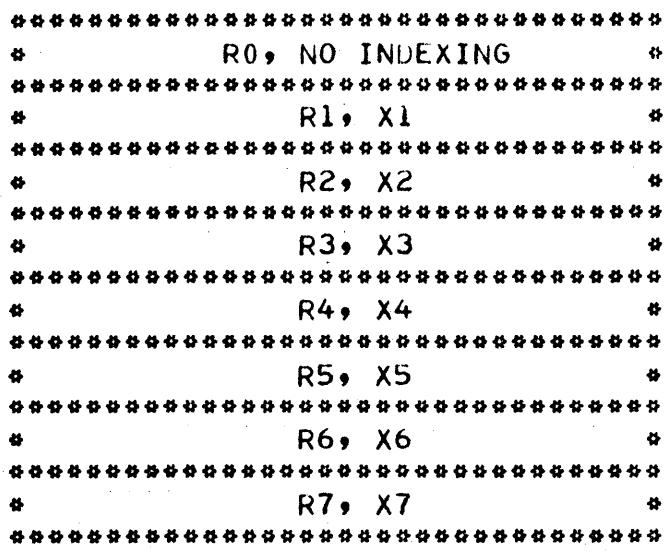
FOR DOUBLEWORD INSTRUCTIONS, THE I BIT OF THE INSTRUCTION IS PLACED IN THE ENTIRE FIRST WORD AND BITS 0-15 OF THE SECOND WORD OF THE OPERAND, AND BITS 16-31 OF THE INSTRUCTION BECOME BITS 16-31 OF THE SECOND WORD OF THE OPERAND. THE RANGE OF OPERANDS IS [-65536,+65535].

FOR THE LDB INSTRUCTION, BITS 24-31 OF THE INSTRUCTION FORM THE OPERAND.

FOR FLOATING POINT INSTRUCTIONS (INCLUDING DOUBLE PRECISION), THE I BIT OF THE INSTRUCTION BECOMES BIT 0 OF THE OPERAND, AND BITS 16-31 OF THE INSTRUCTION BECOME BITS 1-16 OF THE OPERAND. BITS 17-31 OF THE OPERAND, AND THE ENTIRE SECOND WORD (IF ANY), ARE SET TO ZEROS. THE RANGE OF OPERANDS IS APPROXIMATELY [2↑(-256),2↑256] AND [-2↑(-256),-2↑256].

FOR INSTRUCTIONS FOR WHICH AN IMMEDIATE OPERAND IS NOT MEANINGFUL, SUCH AS STOR, A TRAP RESULTS.

2. REGISTERS



THE EIGHT GENERAL-PURPOSE 32-BIT REGISTERS MAY BE USED AS OPERANDS OR AS INDEX REGISTERS. IF INDEX REGISTER 0 IS SPECIFIED, NO INDEXING OCCURS.

PB: PROGRAM BASE REGISTER; 20 BITS
CONTAINS THE ABSOLUTE ADDRESS OF THE FIRST LOCATION OF THE CODE SEGMENT BEING EXECUTED.

DB: DATA BASE REGISTER; 20 BITS
CONTAINS THE ABSOLUTE ADDRESS OF THE FIRST LOCATION (BOTTOM) OF THE STACK.

S: STACK POINTER REGISTER; 20 BITS
CONTAINS THE ABSOLUTE ADDRESS OF THE LAST USED CORE LOCATION OF THE STACK (TOP OF THE STACK).

Q: STACK MARKER POINTER; 20 BITS
CONTAINS THE ABSOLUTE ADDRESS OF THE CURRENT STACK MARKER BEING USED WITHIN THE STACK.

Z: STACK LIMIT POINTER; 20 BITS
CONTAINS THE ABSOLUTE ADDRESS OF THE LAST WORD OF MEMORY AVAILABLE TO THE STACK.

P: PROGRAM COUNTER; 20 BITS
CONTAINS THE ABSOLUTE ADDRESS OF THE INSTRUCTION BEING EXECUTED.

DST: DATA SEGMENT TABLE REGISTER; 20 BITS
CONTAINS THE ABSOLUTE ADDRESS OF THE FIRST WORD OF THE DATA SEGMENT TABLE.

DSL: DATA SEGMENT LENGTH REGISTER; 12 BITS
CONTAINS THE LENGTH OF THE DATA SEGMENT TABLE.

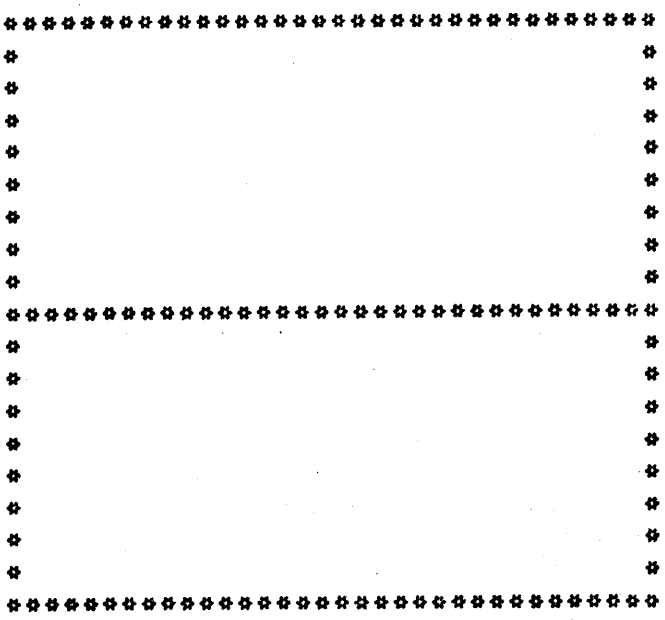
STA: STATUS REGISTER; 32 BITS
INCLUDES:

XNO (8 BIT EXTERNAL NO.)
SNO (10 BIT SEGMENT NO.)
CC (3 BIT CONDITION CODE)
PM (PRIVILEGED MODE BIT)
ID (INTERRUPT DISABLE BIT)
TD (TRACE DISABLE BIT)
MOD (4 BIT MODULE NO.)

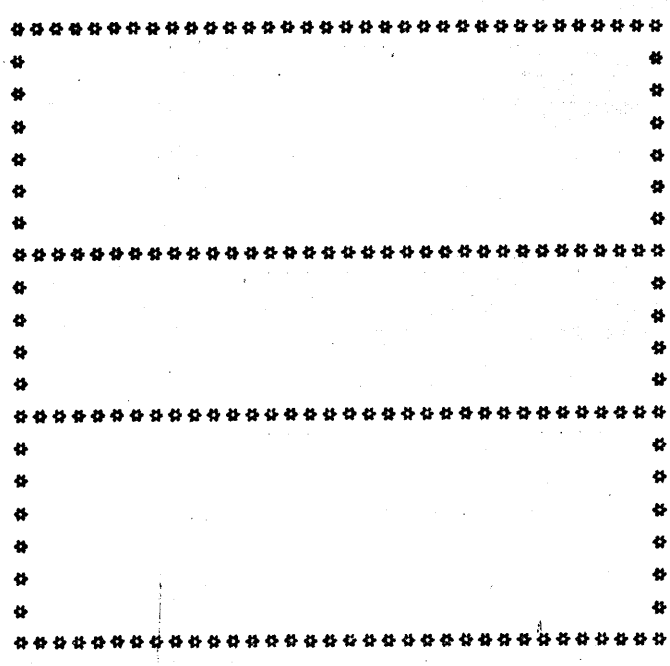
3. ADDRESS SPACE

(HIGHER ADDRESSES ARE DOWNWARD IN THE FIGURE)

PROGRAM AREA

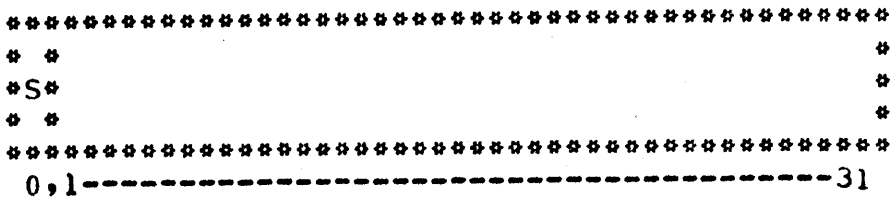


DATA AREA



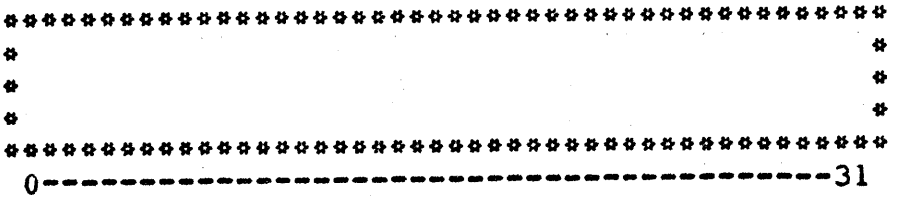
4. DATA FORMATS

FIXED POINT FORMAT: 32 BIT, TWO'S COMPLEMENT



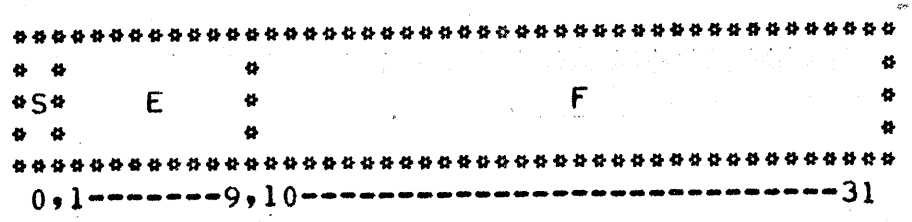
(FOR DOUBLE-WORD FIXED POINT FORMAT, 32 MAGNITUDE BITS ARE ADDED ON THE RIGHT.)

LOGICAL FORMAT, 32 BIT POSITIVE INTEGER



FLOATING POINT FORMATS, SIGN + MAGNITUDE REPRESENTATION

SINGLE PRECISION

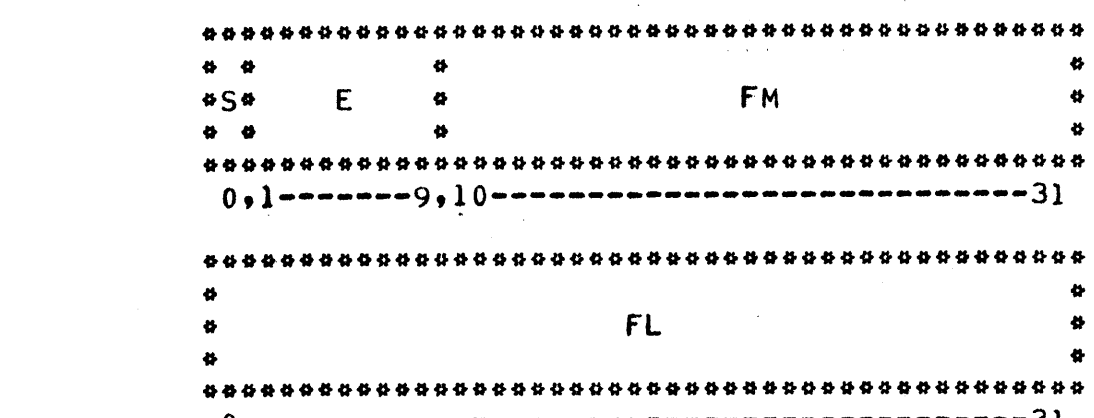


S = FRACTION SIGN BIT (0 FOR POSITIVE, 1 FOR NEGATIVE)

E = EXPONENT+256 (RANGE 0 TO 511)

F = FRACTION (RANGE 0 TO (2↑22) - 1)

DOUBLE PRECISION



S = FRACTION SIGN BIT (0 FOR POSITIVE, 1 FOR NEGATIVE)

E = EXPONENT+256 (RANGE 0 TO 511)

FM = FRACTION MOST SIGNIFICANT (RANGE 0 TO (2↑22)-1)

FL = FRACTION LEAST SIGNIFICANT (RANGE 0 TO (2↑32)-1)

NOTE: IN SIGN + MAGNITUDE REPRESENTATION, THE FRACTION IS ALWAYS POSITIVE WITH THE S BIT CONTAINING THE SIGN OF THE NUMBER. THE BINARY POINT IS ASSUMED TO THE LEFT OF BIT 10 WITH AN IMPLIED LEADING 1 TO THE LEFT OF THE BINARY POINT. THUS ALL FLOATING POINT NUMBERS ARE STORED IN NORMALIZED FORM, BUT NO BIT IS WASTED ON THE LEADING 1, MAKING ALL FRACTION BITS SIGNIFICANT. THE EXCEPTIONS ARE THAT 0 IS A WORD CONTAINING ALL 0'S, AND A ONE IN THE SIGN POSITION WITH THE REMAINDER OF THE WORD CONTAINING ZEROS IS AN UNDEFINED RESULT. THEREFORE:

SINGLE PRECISION

DECIMAL VALUE = (-1)↑S * 2↑(E-256) * (1+F*2↑(-22))
DECIMAL VALUE = 0 WHEN S = E = F = 0

DOUBLE PRECISION

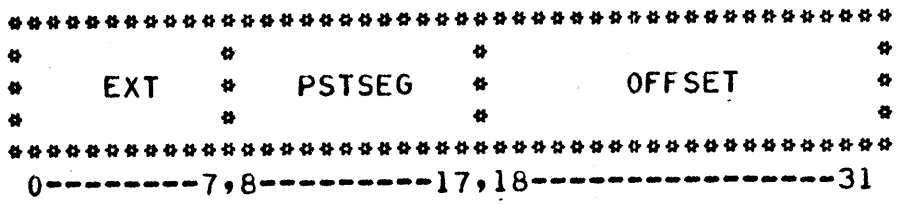
DECIMAL VALUE = (-1)↑S * 2↑(E-256) * (1+FM*2↑(-22)+FL*2↑(-54))
DECIMAL VALUE = 0 WHEN S = E = FM = FL = 0

CASES WHICH MAY CAUSE CONFUSION

```
*****************************************
*0*00000000*00000000000000000000000*  = 0
*****************************************
*0*10000000*00000000000000000000000*  = 1
*****************************************
*1*00000000*00000000000000000000000*  = UNDEFINED
*****************************************
```

5. STACK MARKER WORD FORMATS

FIRST STACK MARKER WORD

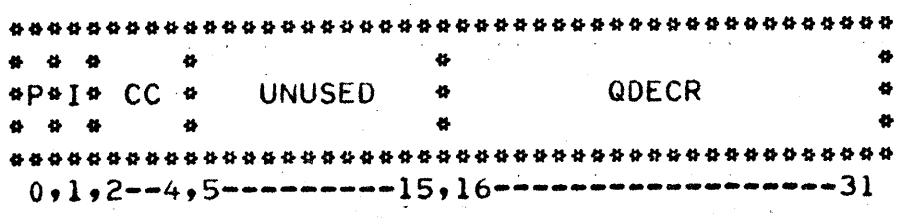


EXT: EXTERNAL INDEX OF CALLER

PSTSEG: PST SEGMENT INDEX OF CALLER

OFFSET: RETURN ADDRESS RELATIVE TO PSTSEG

SECOND STACK MARKER WORD



P: PRIVILEGED MODE FLIP-FLOP SETTING OF CALLER

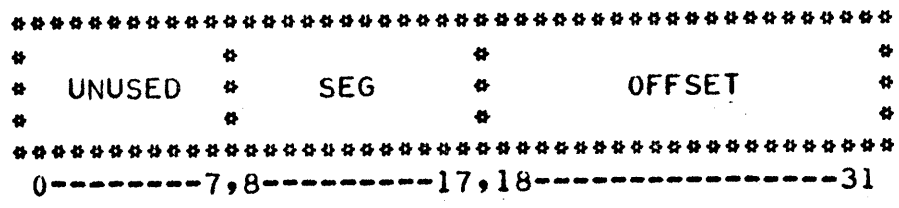
I: INTERRUPT DISABLE FLIP-FLOP SETTING OF CALLER

CC: CONDITION CODE SETTING OF CALLER

QDECR: VALUE TO BE SUBTRACTED FROM Q REGISTER TO OBTAIN
Q VALUE OF CALLER

6. PROGRAM LABEL FORMAT

A PROGRAM LABEL IS USED WHENEVER AN INDIRECT PROGRAM REFERENCE IS NEEDED.



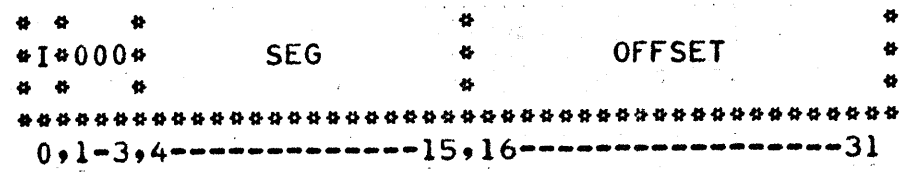
SEG: PST INDEX OF CALLED SEGMENT RELATIVE TO CURRENT PST INDEX, MODULO 1024

OFFSET: ADDRESS RELATIVE TO BASE OF CALLED SEGMENT

7. DATA LABEL FORMATS

ORDINARY DATA LABEL

THIS LABEL IS USED FOR INDIRECT ADDRESSING AND FOR ACCESSING ARRAY ELEMENTS. IF SEG IS NONZERO, IT AUTOMATICALLY TRIGGERS A PROCESS WHEREBY THE INDEX REGISTER SPECIFIED IN THE INSTRUCTION, AND SUCCESSIVE REGISTERS, ARE USED TO INDEX ALONG THE DIMENSIONS OF THE ARRAY.



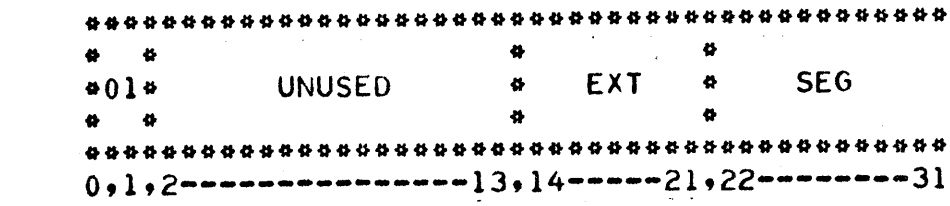
I: INDIRECT BIT. IF I=0, LABEL POINTS DIRECTLY TO AN OPERAND; IF I=1, LABEL POINTS TO ANOTHER LABEL. (NOTE: WHEN MULTI-LEVEL INDIRECT ADDRESSING IS USED, NO MEMBER OF THE CHAIN OF INDIRECT DATA LABELS MAY BE AN ELEMENT OF AN ARRAY OF MORE THAN ONE DIMENSION.)

SEG: THE ADDRESS OF AN ARRAY DESCRIPTION BLOCK, RELATIVE TO THE BEGINNING OF THE DST.

OFFSET: A VALUE USED TO INDEX ALONG THE FIRST DIMENSION OF THE ARRAY, IN ADDITION TO THE INDEX REGISTER.

EXTERNAL CALL DATA LABEL

THIS LABEL IS USED TO CONVERT A NORMAL DATA REFERENCE TO AN EXTERNAL CALL (ECAL) TO THE DESIGNATED SEGMENT.

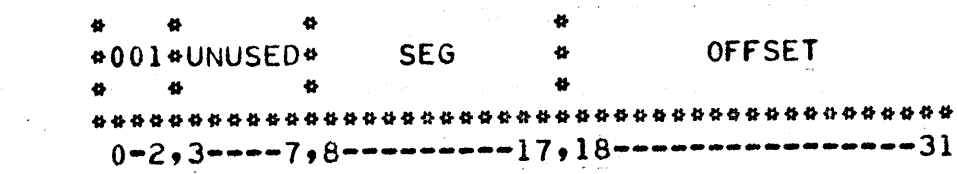


EXT: EXT INDEX OF CALLED SEGMENT

SEG: PST SEGMENT NUMBER OF CALLED SEGMENT

PROCEDURE CALL DATA LABEL

THIS LABEL IS USED TO CONVERT A NORMAL DATA REFERENCE TO A PROCEDURE CALL (PCAL) TO THE DESIGNATED SEGMENT WITHIN THE CURRENT PST.



SEG: PST INDEX OF CALLED SEGMENT RELATIVE TO CURRENT PST INDEX, MODULO 1024

OFFSET: ADDRESS RELATIVE TO BASE OF CALLED SEGMENT.

8. TABLES

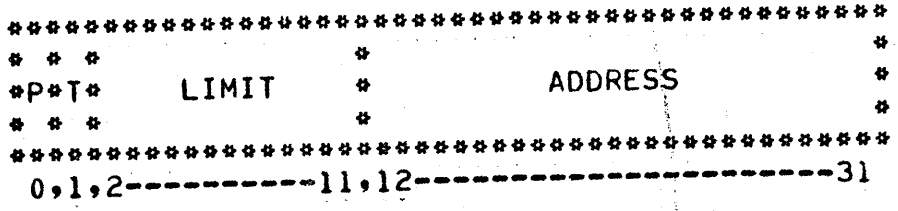
THE FOLLOWING TABLES ARE ALWAYS PRESENT IN CORE:

EXTERNAL DIRECTORY (IN A DEDICATED LOCATION)
PROGRAM SEGMENT TABLES (POSSIBLY SEVERAL)
DATA SEGMENT TABLES (POSSIBLY SEVERAL)
I/O DEVICE STATUS TABLE (IN A DEDICATED LOCATION)
INTERRUPT TABLE (IN A DEDICATED LOCATION)

EACH OF THESE TABLES IS DESCRIBED BELOW.

9. EXTERNAL DIRECTORY

THE EXTERNAL DIRECTORY OCCUPIES 256 WORDS OF CORE IN A DEDICATED LOCATION. IT HAS AN ENTRY, CALLED AN EXTERNAL POINTER, FOR EACH PROGRAM WHICH EXISTS IN THE SYSTEM. THE FORMAT FOR AN EXTERNAL POINTER IS AS FOLLOWS:



P: PRESENCE BIT. P=0 IF PST IS PRESENT AND LOCATED AT "ADDRESS". P=1 IF PST IS NOT PRESENT (CAUSES PRESENCE TRAP).

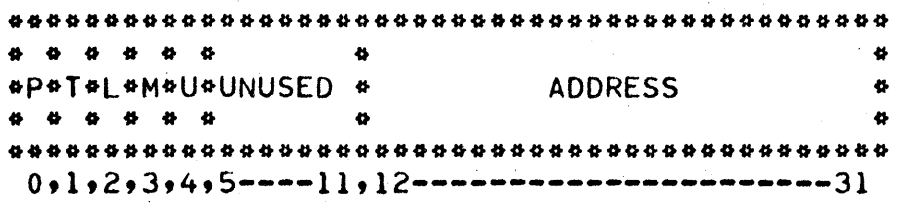
T: TRACE BIT. IF T=0, NO ACTION. IF T=1, TRACE TRAP OCCURS.

LIMIT: LARGEST PST INDEX AVAILABLE TO ECAL INSTRUCTION (ONE LESS THAN LENGTH OF PST.) CAUSES LIMIT-TRAP IF EXCEEDED BY ECAL.

ADDRESS: ABSOLUTE MEMORY LOCATION OF BEGINNING OF PST IF PST IS PRESENT (P=0).

10. PROGRAM SEGMENT TABLE (PST)

EACH PROGRAM WHICH EXISTS IN THE SYSTEM HAS A PROGRAM SEGMENT TABLE WHICH MAY BE UP TO 1024 WORDS IN LENGTH. EACH SEGMENT OF A PROGRAM HAS AN ENTRY, CALLED A PROGRAM SEGMENT CONTROL WORD (PSCW), IN THE PST. THE FORMAT OF A PSCW IS AS FOLLOWS:



P: PRESENCE BIT. P=0 IF SEGMENT IS PRESENT AND LOCATED AT "ADDRESS". P=1 IF SEGMENT IS NOT PRESENT (CAUSES PRESENCE TRAP).

T: TRACE BIT. IF T=0, NO ACTION. IF T=1, TRACE TRAP OCCURS.

L: LINK BIT. IF L=0, THE PSCW POINTS TO THE CODE SEGMENT. IF L=1, THE PSCW POINTS TO ANOTHER PSCW (RELATIVE TO BASE OF PST).

M: MODE BIT. IF M=0(1), THE CODE SEGMENT POINTED TO IS UNPRIVILEGED (PRIVILEGED).

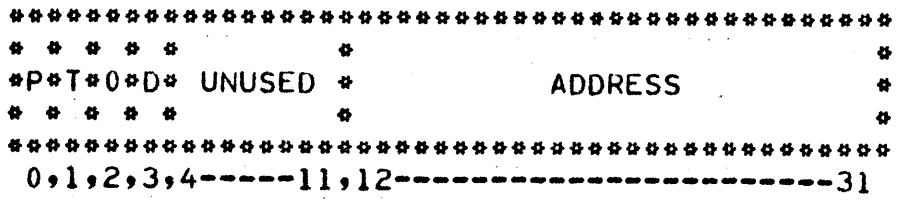
U: "UNCALLABLE" BIT. IF U=1, A TRAP OCCURS WHENEVER THIS PSCW IS REFERENCED BY AN ECAL INSTRUCTION.

ADDRESS: ABSOLUTE MEMORY LOCATION OF BEGINNING OF SEGMENT IF SEGMENT IS PRESENT (P=0).

11. DATA SEGMENT TABLE (DST)

EACH PROGRAM WHICH EXISTS IN THE SYSTEM HAS A DATA SEGMENT TABLE. THE DST DESCRIBES A COLLECTION OF ARRAYS WHICH MAY BE REFERENCED BY THE PROGRAM, EACH OF WHICH HAS AN ARRAY DESCRIPTION BLOCK IN THE DST. EACH ARRAY DESCRIPTION BLOCK CONTAINS ONE DESCRIPTOR, FOLLOWED BY A DIMENSION WORD FOR EACH DIMENSION OF THE ARRAY. THE FORMATS ARE:

DESCRIPTOR



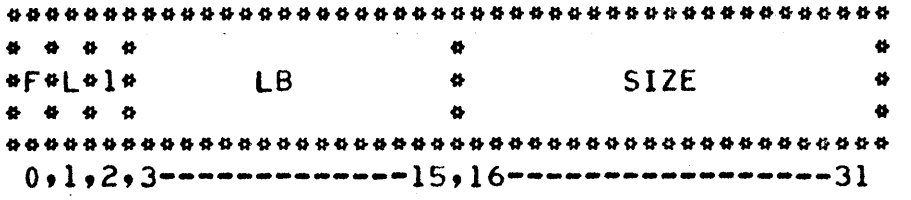
P: PRESENCE BIT. P=0 IF VECTOR POINTED TO IS PRESENT AND LOCATED AT "ADDRESS". P=1 IF VECTOR IS NOT PRESENT (CAUSES PRESENCE TRAP.)

T: TRACE BIT. IF T=0, NO ACTION. IF T=1, TRACE TRAP OCCURS.

D: DIRTY BIT. D=1 IF THE ARRAY POINTED TO BY THIS DESCRIPTOR HAS BEEN ALTERED SINCE IT WAS ENTERED INTO CORE.

ADDRESS: FOR SINGLE-DIMENSIONAL ARRAYS, THE ABSOLUTE ADDRESS OF THE ACTUAL ARRAY. FOR MULTIPLE-DIMENSIONAL ARRAYS, THE ABSOLUTE ADDRESS OF A VECTOR OF SECONDARY DESCRIPTORS, EACH OF WHICH IN TURN POINTS TO ANOTHER VECTOR OF DATA OR SECONDARY DESCRIPTORS. THE SECONDARY DESCRIPTORS HAVE THE SAME FORMAT AS THE ABOVE, AND MAY EXIST ANYWHERE IN MEMORY.

DIMENSION WORD



F: DIMENSION WORD TYPE. F=0 MEANS VARIABLE INDEX TYPE. F=1 MEANS FIXED-INDEX TYPE.

L: LAST DIMENSION BIT. L=0 MEANS ANOTHER DIMENSION WORD FOLLOWS. L=1 MEANS THIS IS THE LAST DIMENSION WORD IN THE ARRAY DESCRIPTOR BLOCK.

VARIABLE-INDEX TYPE FORMAT (F=0)
(THIS TYPE OF DIMENSION WORD ALLOWS THE INDEX ALONG ITS DIMENSION TO BE SPECIFIED BY AN INDEX REGISTER.)

LB: LOWER BOUND OF DIMENSION (-4096 <= LB <= 4095)

SIZE: SIZE OF DIMENSION (0 <= SIZE <= 2↑16-1).

FIXED-INDEX TYPE FORMAT (F=1)
(THIS TYPE OF DIMENSION WORD SETS THE INDEX ALONG ITS DIMENSION EQUAL TO A FIXED OFFSET, RATHER THAN TO THE CONTENT OF AN INDEX REGISTER.)

LB: UNUSED, EXCEPT IF BIT 3 IS ON AND THE INSTRUCTION IS A DOUBLEWORD INSTRUCTION, A TRAP OCCURS.

SIZE: OFFSET USED TO SELECT ELEMENT OF VECTOR (0 <= SIZE <= 2↑16-1).

12. INDIRECT ADDRESSING AND ARRAY REFERENCES

WHEN THE INDIRECT BIT OF AN INSTRUCTION IS 1, THE WORD ADDRESSED BY THE INSTRUCTION CONTAINS A DATA LABEL OR A PROGRAM LABEL. IF THE WORD IS A DATA LABEL, IT SPECIFIES A SEGMENT NUMBER AND AN OFFSET. IF THE SEGMENT NUMBER IS ZERO, THE OFFSET SPECIFIES A WORD DISPLACEMEN RELATIVE TO DB; THIS IS SIMPLE INDIRECT ADDRESSING. MULTIPLE LEVELS OF INDIRECT ADDRESSING ARE ALLOWED. ON THE LAST LEVEL OF INDIRECTION, WHEN A DATA LABEL IS FOUND WITH INDIRECT BIT = 0, THE INDEX REGISTER X SPECIFIED IN THE INSTRUCTION IS ADDED TO THE OFFSET FIELD OF THE DATA LABEL TO FIND THE ACTUAL DATA WORD.

IF THE SEGMENT NUMBER FIELD OF THE DATA LABEL IS NONZERO, IT SPECIFIES A DISPLACEMENT RELATIVE TO THE BEGINNING OF THE DST, AND TRIGGERS AN ARRAY-ACCESSING PROCESS. IN THIS CASE, THE WORD POINTED TO IN THE DST WILL BE THE BEGINNING OF AN ARRAY DESCRIPTION BLOCK CONTAINING A DESCRIPTOR AND SEVERAL DIMENSION WORDS, ONE FOR EACH

DIMENSION OF THE ARRAY. MULTIPLE-DIMENSIONAL ARRAYS ARE STORED AS TREES OF DESCRIPTORS. THE ARRAY DESCRIPTOR POINTS TO (CONTAINS THE ABSOLUTE ADDRESS OF) A VECTOR OF SECONDARY DESCRIPTORS. EACH SECONDAR DESCRIPTOR ON THE LOWEST LEVEL POINTS TO A VECTOR OF ACTUAL DATA WORDS THE NUMBER OF LEVELS OF DESCRIPTORS IN THE CHAIN IS EQUAL TO THE NUMBE OF DIMENSION WORDS IN THE ARRAY DESCRIPTION BLOCK IN THE DST.

THE ARRAY ACCESSING MECHANISM NOW FOLLOWS THE CHAIN OF DESCRIPTORS, INDEXING AT EACH LEVEL BY THE CONTENTS OF AN INDEX REGISTER. THE FIRST DIMENSION IS INDEXED BY THE CONTENT OF THE INDEX REGISTER X SPECIFIED IN THE ORIGINAL INSTRUCTION, PLUS THE OFFSET FIELD OF THE DATA LABEL WHICH POINTED INTO THE DST. THE SECOND DIMENSION IS INDEXED BY THE CONTENT OF REGISTER X+1. THE THIRD DIMENSION IS INDEXED BY THE CONTENT OF REGISTER X+2, ETC. HOWEVER, IF THE INDEX REGISTER X IN THE ORIGINAL INSTRUCTION IS REGISTER 0, NO INDEX REGISTERS ARE USED, AND THE ONLY INDEXING IS ALONG THE FIRST DIMENSION, BY THE OFFSET FIELD OF THE DATA LABEL. ALSO, IF ANY DIMENSION IS INDEXED BY REGISTER 7, NO INDEX REGISTERS ARE USED WITH ANY OF THE FOLLOWING DIMENSIONS (REGISTER WRAPAROUND DOES NOT OCCUR.)

IF ALL DIMENSION WORDS IN THE ARRAY DESCRIPTION BLOCK ARE VARIABLE-INDEX TYPE, THE ARRAY ACCESSING PROCESS PROCEEDS AS ABOVE, AND A BOUNDS-CHECKING PROCESS OCCURS ALONG EACH DIMENSION. EACH DIMENSION WORD CONTAINS A LOWER BOUND AND A SIZE. THE INDEX QUANTITY IS COMPUTED AS DESCRIBED ABOVE, AND THE LOWER BOUND IS SUBTRACTED FROM IT. IF THE ORIGINAL INSTRUCTION WAS A DOUBLEWORD INSTRUCTION AND THIS IS THE LAST DIMENSION OF THE ARRAY, THE RESULT IS THEN MULTIPLIED BY TWO. THIS FINAL RESULT IS NOW USED TO INDEX INTO THE NEXT VECTOR OF DESCRIPTORS OR DATA. IN ADDITION, THIS FINAL RESULT MUST BE LESS THAN OR EQUAL TO THE SIZE FIELD OF THE DIMENSION WORD, OR ELSE AN ARRAY BOUNDS TRAP RESULTS.

HOWEVER, IF ANY DIMENSION WORD IN THE ARRAY DESCRIPTION BLOCK IS FIXED-INDEX TYPE, NO INDEX REGISTER IS USED ALONG ITS DIMENSION; INSTEAD, THE INDEX QUANTITY ALONG ITS DIMENSION IS FIXED AND EQUAL TO THE SIZE FIELD OF THE DIMENSION WORD. THE INDEX REGISTER WHICH WOULD HAVE BEEN USED ALONG THIS DIMENSION IS USED ALONG THE NEXT DIMENSION INSTEAD. IN THIS CASE, NO BOUNDS CHECKING OCCURS, EXCEPT THAT IF BIT OF THE DIMENSION WORD IS ON, AND THE INSTRUCTION IS A DOUBLEWORD INSTRUCTION, AND THIS IS THE LAST DIMENSION OF THE ARRAY, A BOUNDS TRAP OCCURS.

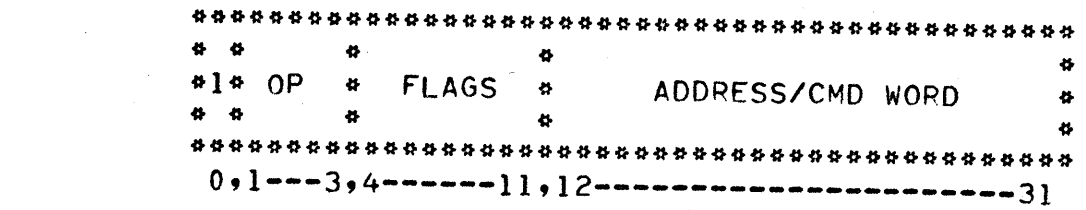
THE SAME OVERALL ARRAY-ACCESSING MECHANISM IS TRIGGERED WHEN AN INSTRUCTION USES DST-RELATIVE ADDRESSING. THE WORD POINTED TO WILL THEN BE THE BEGINNING OF AN ARRAY DESCRIPTION BLOCK, AND THE PROCESS WILL OCCUR AS BEFORE. HOWEVER, IN THIS CASE, THERE IS NO OFFSET FIELD OF THE DATA LABEL TO BE ADDED TO THE INDEX ALONG THE FIRST DIMENSION.

IF ANY INSTRUCTION MODIFIES AN ELEMENT OF AN ARRAY, ALL DESCRIPTORS ALONG THE CHAIN POINTING TO THAT ELEMENT HAVE THEIR "DIRTY" BITS (BIT 3) SET TO 1.

BEFORE ISSUING A START I/O (SIO) INSTRUCTION, THE PROGRAMMER MUST SET UP A PROGRAM IN CORE CONSISTING OF I/O COMMAND WORDS (IOCW'S) AND I/O DATA WORDS (IODW'S) TO SPECIFY THE OPERATIONS DESIRED. EACH IOCW CONTAINS AN OPCODE AND CERTAIN MODIFIER BITS. SOME IOCW'S ARE FOLLOWED BY ONE OR MORE IODW'S, WHICH SPECIFY THE AREA(S) IN MEMORY TO WHICH THEY PERTAIN. EACH IODW CONTAINS A BYTE ADDRESS (22 BITS) AND A BYTE COUNT (9 BITS) OR A WORD ADDRESS (20 BITS) AND A WORD OR RECORD COUNT (11 BITS). WHEN AN SIO COMMAND IS ISSUED, THE I/O PROCESSOR EXECUTES THE FIRST IOCW OF THE I/O PROGRAM SUCCESSIVELY ON EACH OF THE FOLLOWING IODW'S, THEN EXECUTES THE NEXT IOCW ON EACH OF THE IODW'S FOLLOWING IT, ETC., UNTIL AN "END" IOCW IS ENCOUNTERED.

EACH I/O DEVICE HAS A DEVICE STATUS DOUBLEWORD IN A DEDICATED POSITION IN LOWER CORE STORAGE. THE FIRST WORD CONTAINS A POINTER TO THE NEXT IOCW OR IODW TO BE EXECUTED BY THE DEVICE, IF ANY. THE SECOND WORD IS USED ONLY FOR SLOW SPEED DEVICES. IT CONTAINS THE CURRENT BYTE ADDRESS AND BYTE COUNT FOR THE DATA TRANSFER BEING MADE BY THE DEVICE. FOR HIGH SPEED DEVICES, THE WORD ADDRESS AND COUNT ARE STORED IN THE CHANNEL WHICH HANDLES THE DATA TRANSFER FOR THE DEVICE. AS THE DEVICE EXECUTES THE TRANSFER SPECIFIED BY AN IODW, THE DATA ADDRESS IS INCREMENTED AND THE DATA COUNT IS DECREMENTED UNTIL IT REACHES ZERO; THEN THE DEVICE PROCEEDS TO THE NEXT IODW.

IOCW:



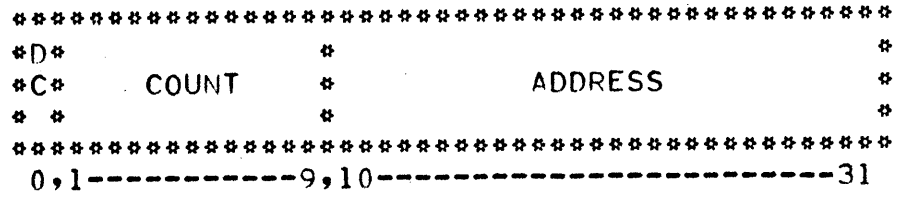
OP: 000 = END
001 = SENSE
010 = CONTROL
011 = JUMP
100 = WRITE
101 = READ
110 = INTERRUPT A CPU

FLAGS: TO BE DEFINED

ADDRESS: ABSOLUTE CORE ADDRESS, USED IN SENSE AND JUMP INSTRUCTIONS.

CMD WORD: 16 BITS TO BE SENT TO DEVICE, USED IN CONTROL INSTRUCTION. (RIGHT JUSTIFIED)

IODW:



DC: DATA CHAINING. IF DC=1, ANOTHER IODW IMMEDIATELY FOLLOWS.

COUNT: NUMBER OF BYTES OR WORDS TO BE PROCESSED.

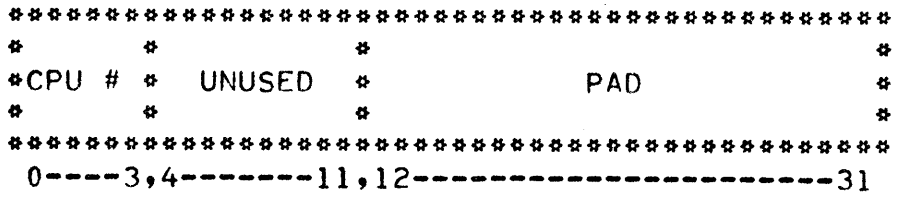
ADDRESS: ABSOLUTE ADDRESS OF FIRST BYTE OR WORD TO BE PROCESSED.

WHEN WORD TRANSFER IS SPECIFIED (HIGH SPEED DEVICES), THE WORD OR RECORD COUNT OCCUPIES BITS 1-11; AND THE WORD ADDRESS OCCUPIES BITS 12-31.

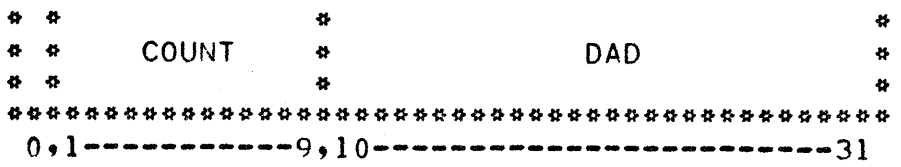
14. I/O DEVICE STATUS TABLE

THE I/O DEVICE STATUS TABLE OCCUPIES 512 WORDS OF CORE IN A DEDICATED LOCATION. IT HAS A DOUBLEWORD, CALLED A DEVICE STATUS DOUBLEWORD, FOR EVERY I/O DEVICE IN THE SYSTEM (UP TO 256 DEVICES.) THE FORMAT OF THE DEVICE STATUS DOUBLEWORD IS:

FIRST WORD



SECOND WORD



CPU #: CPU MODULE NUMBER

PAD: ABSOLUTE CORE ADDRESS OF THE NEXT IOCW OR IODW IN THE I/O PROGRAM NOW BEING EXECUTED, IF ANY.

COUNT: NUMBER OF BYTES OR WORDS REMAINING TO BE PROCESSED UNDER THE CURRENT IODW.

DAD: ABSOLUTE BYTE OR WORD ADDRESS OF THE BYTE OR WORD CURRENTLY BEING PROCESSED.

A HIGH SPEED DEVICE DOES NOT USE THE SECOND WORD. INSTEAD, THE WORD COUNT AND ADDRESS ARE KEPT IN THE CHANNEL REGISTERS. HOWEVER, WHEN THE TRANSFER IS COMPLETE, THE RESIDUAL COUNT AND ADDRESS ARE REMOVED FROM THE CHANNEL AND STORED IN THE SECOND LOCATION OF THE DEVICE STATUS DOUBLEWORD. THE ADDRESS RESIDUE IS 1 HIGHER THAN THE ADDRESS OF THE LAST WORD TRANSFERRED.

15. INTERRUPT TABLE

THE INTERRUPT TABLE OCCUPIES 20 WORDS OF CORE IN A DEDICATED LOCATION. EACH WORD CONTAINS THE ABSOLUTE CORE ADDRESS OF AN INTERRUPT ROUTINE. BRANCHES TO THE 20 ROUTINES ARE CAUSED BY EXTERNAL INTERRUPTS FROM THE 16 MODULES, AND BY THE SPECIAL INTERRUPTS (PARITY ERROR, STACK OVERFLOW, POWER FAIL AND NONRESPONDING MODULE.)

16. INTERRUPT PROCESSING

INTERRUPTS ARE OF TWO TYPES: EXTERNAL INTERRUPTS AND SPECIAL INTERRUPTS. EXTERNAL INTERRUPTS ARE DEFINED AS INPUTS TO THE CPU FROM OTHER MODULES, EXCEPTING THOSE INPUTS WHICH THE CPU HAS REQUESTED AND IS WAITING FOR. SPECIAL INTERRUPTS ARE OF FOUR TYPES: PARITY ERROR, STACK OVERFLOW, POWER FAIL, AND NONRESPONDING MODULE.

WHEN AN INTERRUPT ARRIVES AT THE CPU, INFORMATION DESCRIBING THE INTERRUPT IS LOADED INTO THE INTERRUPT REGISTER. OTHER INTERRUPTS ARE THEN PREVENTED FROM ACCESSING THE INTERRUPT REGISTER UNTIL THE FIRST INTERRUPT HAS BEEN PROCESSED. THE INTERRUPT DISABLE BIT IN THE STATUS REGISTER DISABLES OR ENABLES ALL EXTERNAL INTERRUPTS (SPECIAL INTERRUPTS CANNOT BE DISABLED.) IF ENABLED, AN INTERRUPT WILL BE SERVICED BETWEEN MACHINE INSTRUCTIONS (SOME LONG INSTRUCTIONS ARE INTERRUPTABLE, AND MUST BE RESTARTED.) THE INTERRUPT IS SERVICED AS FOLLOWS:

1. A NEW TWO-WORD STACK MARKER (REGULAR FORMAT) IS PLACED ON THE STACK.

2. THE INTERRUPT DISABLE BIT AND PRIVILEGED MODE BIT IN THE STATUS REGISTER ARE TURNED ON.

3. THE XNO AND SNO FIELDS IN THE STATUS REGISTER ARE SET TO 0.

4. THE PB REGISTER IS SET TO 0.

5. THE INTERRUPTING MODULE NUMBER IS USED TO INDEX INTO THE INTERRUPT TABLE, WHICH CONTAINS THE ABSOLUTE ADDRESS OF THE INTERRUPT-HANDLING ROUTINE. THE TABLE HAS ADDITIONAL ENTRIES FOR THE SPECIAL INTERRUPTS. A BRANCH TO THE ADDRESS GIVEN BY THE TABLE ENTRY OCCURS.

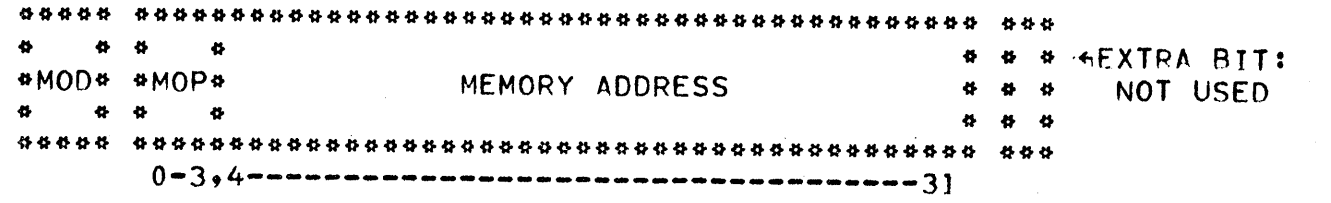
17. TRAP PROCESSING

A TRAP OCCURS WHEN AN UNUSUAL CONDITION IS DETECTED WITHIN THE CPU DURING EXECUTION OF AN INSTRUCTION. EXAMPLES OF TRAP CONDITIONS ARE OVERFLOW, PRESENCE FAULT, MEMORY BOUNDS VIOLATION, ETC.

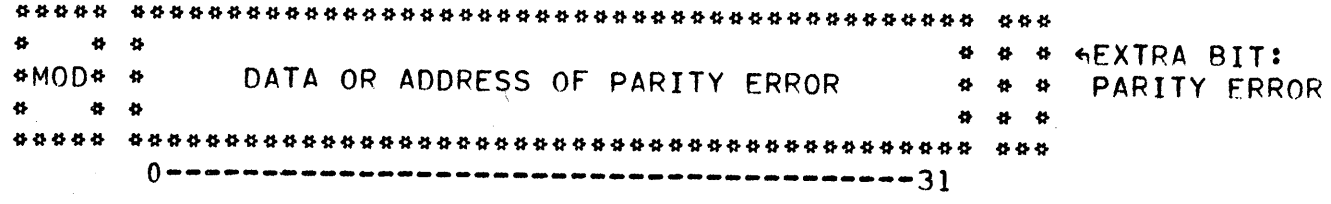
WHEN A TRAP OCCURS, THE CURRENT INSTRUCTION IN THE CPU IS ABORTED. A PARAMETER WORD DESCRIBING THE FAULT IS PLACED ON THE STACK, AND A PCAL IS EXECUTED TO SEGMENT 0 OF THE CURRENT PST, USING AN OFFSET DETERMINED BY THE TYPE OF TRAP.

NOTE THAT THIS IMPLIES THAT ENTRY 0 OF THE PST OF EACH USER MUST POINT TO A COLLECTION OF TRAP-HANDLING ROUTINES. STANDARD ROUTINES MAY BE PROVIDED BY THE OPERATING SYSTEM FOR THIS PURPOSE.

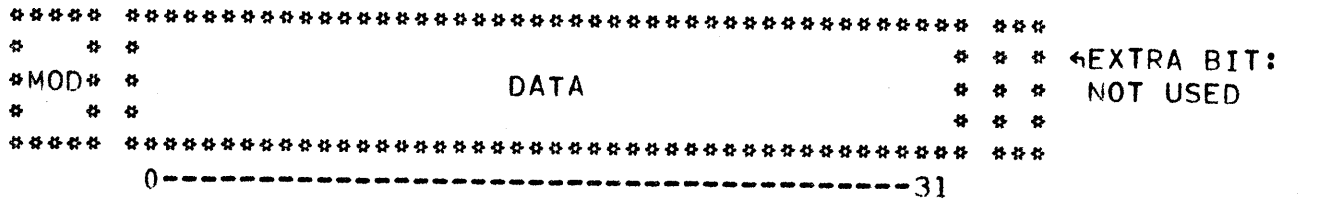
CPU - MEMORY



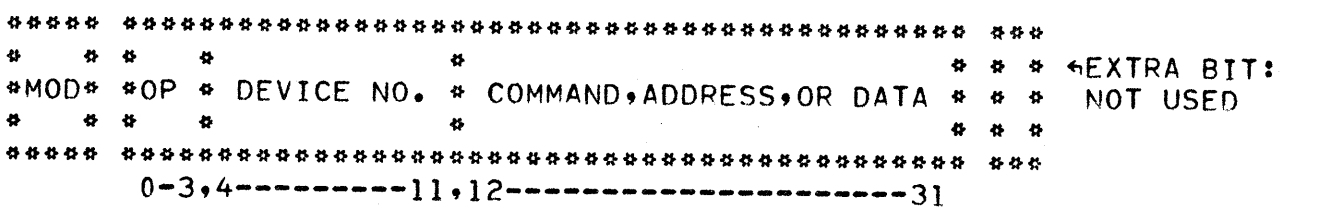
MEMORY - CPU



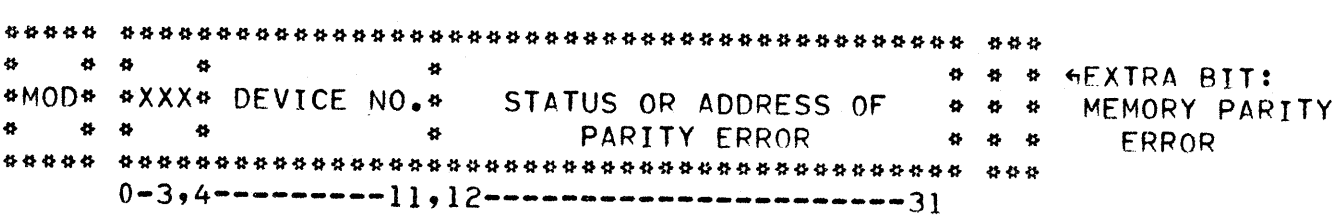
CPU - MEMORY



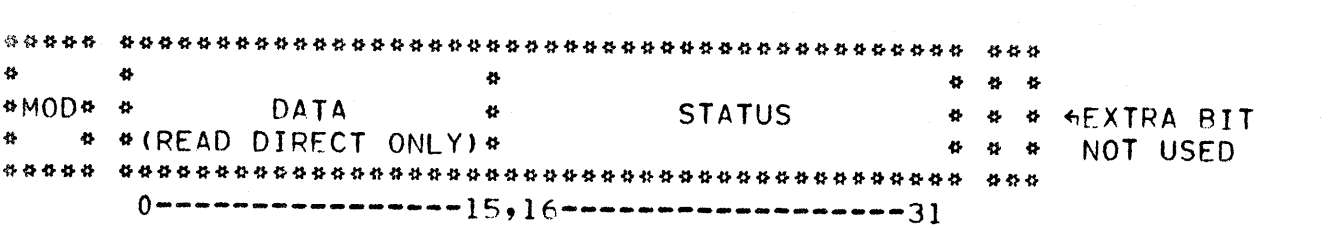
CPU - I/O



INTERRUPTING MODULE - CPU



I/O / CPU



19. CONDITION CODE PATTERNS

AT ALL TIMES, EXACTLY ONE CONDITION CODE BIT IS ON. THE BITS ARE DESIGNATED 0, 1, AND 2.

PATTERN A:

CC = 0 IF RESULT = 0
CC = 1 IF RESULT > 0
CC = 2 IF RESULT < 0

PATTERN B:

CC = 0 IF ALL RESULT BITS ARE 0
CC = 1 IF ANY RESULT BITS ARE 1

PATTERN C:

CC = 0 IF OPERANDS ARE EQUAL
CC = 1 IF OPERAND 1 > OPERAND 2
CC = 2 IF OPERAND 1 < OPERAND 2

20. MODES OF OPERATION

THE MACHINE HAS TWO MODES: PRIVILEGED MODE, AND USER MODE. PRIVILEGED INSTRUCTIONS MAY BE EXECUTED ONLY IN PRIVILEGED MODE. IF A PRIVILEGED INSTRUCTION IS ENCOUNTERED IN USER MODE, A TRAP OCCURS.

SECTION II

1. INDEX OF INSTRUCTIONS
2. DEFINITIONS OF INSTRUCTIONS

IN THE FOLLOWING EXPLANATIONS,

R = THE REGISTER SPECIFIED IN THE R FIELD.

X = THE REGISTER SPECIFIED IN THE X FIELD.

M = THE RESULT OF NORMAL ADDRESS COMPUTATION. NOTE THAT M MAY BE A REGISTER, A MEMORY ADDRESS, OR IMMEDIATE DATA.

A = THE A FIELD OF THE INSTRUCTION.

1. INDEX OF INSTRUCTIONS

LOAD AND STORE INSTRUCTIONS

0. LOAD (LOAD)
1. LLH (LOAD LEFT HALF)
2. LRH (LOAD RIGHT HALF)
3. STOR (STORE)
4. SLH (STORE LEFT HALF)
5. SRH (STORE RIGHT HALF)
6. EXCH (EXCHANGE)

ARITHMETIC AND BOOLEAN INSTRUCTIONS

7. ADD (ADD)
8. SUB (SUBTRACT)
9. MPYE (MULTIPLY EXTENDED)
10. MPY (MULTIPLY)
11. DIVE (DIVIDE EXTENDED)
12. DIV (DIVIDE)
13. AND (AND)
14. IOR (INCLUSIVE OR)
15. XOR (EXCLUSIVE OR)
16. ADDM (ADD TO MEMORY)
17. SUBM (SUBTRACT FROM MEMORY)
18. ANDM (AND MEMORY)
19. IORM (INCLUSIVE OR MEMORY)
20. XORM (EXCLUSIVE OR MEMORY)

LOGICAL INSTRUCTIONS

21. ADDL (ADD LOGICAL)
22. SUBL (SUBTRACT LOGICAL)
23. MPYL (MULTIPLY LOGICAL)
24. DIVL (DIVIDE LOGICAL)

FLOATING POINT INSTRUCTIONS

25. FADD (FLOATING ADD)
26. FSUB (FLOATING SUBTRACT)
27. FMYE (FLOATING MULTIPLY EXTENDED)
28. FMPY (FLOATING MULTIPLY)
29. FDIV (FLOATING DIVIDE)
30. FLT (FLOAT)
31. FIXR (FIX AND ROUND)
32. FIXT (FIX AND TRUNCATE)
33. FNEG (FLOATING NEGATE)

DOUBLE PRECISION FLOATING POINT INSTRUCTIONS

34. DFAD (DOUBLE FLOATING ADD)
35. DFSB (DOUBLE FLOATING SUBTRACT)
36. DFMP (DOUBLE FLOATING MULTIPLY)
37. DFDV (DOUBLE FLOATING DIVIDE)
38. INCM (INCREMENT M)
39. DECM (DECREMENT M)
40. INCR (INCREMENT R)
41. DECR (DECREMENT R)

UNARY OPERATORS

42A. SSP (SET SIGN PLUS)
42B. SSM (SET SIGN MINUS)
42C. CHS (CHANGE SIGN)
42D. NEG (NEGATE
42E. ABS (ABSOLUTE VALUE)
42F. CMPL (COMPLEMENT)
42G. ZERO (ZERO)
42H. TEST (TEST)

DOUBLEWORD INSTRUCTIONS

43. DLOD (DOUBLE LOAD)
44. DSTO (DOUBLE STORE)
45. DADD (DOUBLE ADD)
46. DSUB (DOUBLE SUBTRACT)
47. DCOM (DOUBLE COMPARE)
48. DTST (DOUBLE TEST)

BYTE INSTRUCTIONS

49. LDB (LOAD BYTE)
50. STB (STORE BYTE)
51. EDIT (EDIT BYTES)

COMPARES

52. ACM (ARITHMETIC COMPARE)
53. LCM (LOGICAL COMPARE)
54. FCM (FLOATING COMPARE)
55. DFCM (DOUBLE PRECISION FLOATING COMPARE)
56. BCC (BRANCH ON CONDITION CODE)
57. BRO (BRANCH IF REGISTER ODD)
58. BRE (BRANCH IF REGISTER EVEN)
59. BRP (BRANCH IF REGISTER POSITIVE
60. BRNP (BRANCH IF REGISTER NON-POSITIVE)
61. BRN (BRANCH IF REGISTER NEGATIVE)
62. BRNN (BRANCH IF REGISTER NON-NEGATIVE)
63. BRZ (BRANCH IF REGISTER ZERO)
64. BRNZ (BRANCH IF REGISTER NON-ZERO)

SHIFTS

65. LSL (LEFT SHIFT LOGICAL)
66. RSL (RIGHT SHIFT LOGICAL)
67. LSA (LEFT SHIFT ARITHMETIC)
68. RSA (RIGHT SHIFT ARITHMETIC)
69. LSC (LEFT SHIFT CIRCULAR)
70. RSC (RIGHT SHIFT CIRCULAR)

SPECIAL PURPOSE

71. EXF (EXTRACT FIELD)
72. DPF (DEPOSIT FIELD)
73. MVW (MOVE WORDS)
74. ASCI (ASCII TO INTEGER)
75. BCDI (BINARY-CODED-DECIMAL TO INTEGER)
76. IBCD (INTEGER TO BINARY-CODED-DECIMAL)
77. XEQ (EXECUTE)
78. PSDS (PUSH DATA STACK)
79. PPDS (POP DATA STACK)
80. LDQS (LOAD Q AND S)
81. STQS (STORE Q AND S)
82. STST (STORE STATUS)
83. STRZ (STORE Z)
84. LMS (LOAD MEMORY ONTO STACK)
85. ADS (ADD TO S REGISTER)
86. LAR (LOAD ADDRESS INTO REGISTER)
87. LBAR (LOAD BYTE ADDRESS INTO REGISTER)
88. LAS (LOAD ADDRESS ONTO STACK)
89. LFP (LOAD FROM PROGRAM)
90. PUSH (PUSH)
91. POP (POP)
92. PCAL (PROCEDURE CALL)
93. ECAL (EXTERNAL CALL)
94. EXIT (EXIT)
95. TRAC (TRACE)

PRIVILEGED

96. LSTA (LOAD STATUS)
97. LODZ (LOAD Z)
98. LDDB (LOAD DB)
99. STB (STORE DB)
100. LDST (LOAD DST)
101. SDST (STORE DST)
102. INT (INTERROGATE)
103. PAUS (PAUSE)
104. SIO (START I/O)
105. TIO (TEST I/O)
106. RIO (READ I/O)
107. WIO (WRITE I/O)
108. CIO (COMMAND I/O)

DECIMAL INSTRUCTIONS

109. ADDD (ADD DECIMAL)
110. SUBD (SUBTRACT DECIMAL)

2. DEFINITIONS OF INSTRUCTIONS

LOAD AND STORE INSTRUCTIONS:

0.

LOAD R,M (LOAD)
(R) ← (M)

REGISTER R IS LOADED WITH THE CONTENT OF M.
CC = UNAFFECTED.

1.

LLH R,M (LOAD LEFT HALF)
(R[16-31]) ← (M[0-15])
(R[0-15]) ← 0

BITS 0-15 OF THE CONTENT OF M IS LOADED INTO BITS 16-31 OF R.
BITS 0-15 OF R ARE SET TO ZERO.
CC = UNAFFECTED.

2.

LRH R,M (LOAD RIGHT HALF)
(R[16-31]) ← (M[16-31])
(R[0-15]) ← 0

BITS 16-31 OF THE CONTENT OF M IS LOADED INTO BITS 16-31 OF R.
BITS 0-15 OF R ARE SET TO ZERO.
CC = UNAFFECTED.

3.

STOR R,M (STORE)
(M) ← (R)

THE CONTENT OF R IS STORED IN M.
CC = PATTERN A.

4.

```
SLH  R,M        (STORE LEFT HALF)
     (M[0-15]) ↰ (R[16-31])
```

BITS 16-31 OF R ARE STORED IN BITS 0-15 OF M.
BITS 16-31 OF M ARE UNCHANGED.
CC = PATTERN A ON THE RESULTING WORD M.

5.

```
SRH  R,M        (STORE RIGHT HALF)
     (M[16-31]) ↰ (R[16-31])
     (M[0-15]) ↰ UNAFFECTED
```

BITS 16-31 OF R ARE STORED IN BITS 16-31 OF M.
BITS 0-15 OF M ARE UNCHANGED.
CC = PATTERN A ON THE RESULTING WORD M.

6.

```
EXCH R,M        (EXCHANGE)
     (TEMP) ↰ (R)
     (R) ↰ (M)
     (M) ↰ (TEMP)
```

THE CONTENT OF R AND THE CONTENT OF M ARE EXCHANGED.
CC = PATTERN A ON THE WORD STORED INTO M FROM R.

ARITHMETIC AND BOOLEAN INSTRUCTIONS:

7.

ADD R,M (ADD)
(R) ← (R)+(M)

THE CONTENT OF M IS ADDED TO REGISTER R IN FIXED POINT FORM. THIS INSTRUCTION CAN CAUSE AN INTEGER OVERFLOW TRAP IN WHICH CASE (R) IS UNCHANGED.
CC = UNAFFECTED.

8.

SUB R,M (SUBTRACT)
(R) ← (R)-(M)

THE CONTENT OF M IS SUBTRACTED FROM REGISTER R IN FIXED POINT FORM. THIS INSTRUCTION CAN CAUSE AN INTEGER OVERFLOW TRAP IN WHICH CASE (R) IS UNCHANGED.
CC = UNAFFECTED.

9.

MPYE R,M (MULTIPLY EXTENDED)
(R,R+1) ← (R)*(M)

THE CONTENT OF R IS MULTIPLIED BY THE CONTENT OF M IN FIXED POINT FORM, AND THE DOUBLE-LENGTH PRODUCT IS LEFT IN R AND R+1.
CC = PATTERN A ON THE DOUBLEWORD RESULT.

10.

MPY R,M (MULTIPLY)
(R) ← (R)*(M)

THE CONTENT OF R IS MULTIPLIED BY THE CONTENT OF M IN FIXED POINT FORM AND THE 32 BIT RESULT IS LEFT IN R. THIS INSTRUCTION CAN CAUSE AN INTEGER OVERFLOW IN WHICH CASE (R) IS UNCHANGED.
CC = UNAFFECTED.

11.

DIVE R,M (DIVIDE EXTENDED)
(R) ← INTEGER((R,R+1)/(M))
(R+1) ← (R,R+1) MODULO(M)

THE DOUBLE-LENGTH INTEGER IN REGISTERS R AND R+1 IS DIVIDED BY THE CONTENT OF M. THE QUOTIENT INTEGER IS LEFT IN R AND THE REMAINDER IN R+1. THIS INSTRUCTION CAN CAUSE AN INTEGER OVERFLOW IN WHICH CASE THE REGISTERS ARE UNCHANGED.
CC = UNAFFECTED.

12.

DIV R,M (DIVIDE)
(R) ← (R)/(M)

THE CONTENT OF R IS DIVIDED BY THE CONTENT OF M AND THE QUOTIENT IS LEFT IN R. THIS INSTRUCTION CAN CAUSE AN INTEGER OVERFLOW IN WHICH CASE THE REGISTER IS UNCHANGED.
CC = UNAFFECTED.

13.

AND R,M (AND)
(R) ← (R)AND(M)

THE CONTENT OF R IS REPLACED BY THE LOGICAL "AND" OF R AND M.
CC = UNAFFECTED.

14.

IOR R,M (INCLUSIVE OR)
(R) ← (R) OR (M)

THE CONTENT OF R IS REPLACED BY THE "INCLUSIVE OR" OF R AND M.
CC = UNAFFECTED.

15.

XOR R,M (EXCLUSIVE OR)
(R) ← ((R)AND NOT(M)) OR (NOT(R)AND(M))

THE CONTENT OF R IS REPLACED BY THE "EXCLUSIVE OR" OF R AND M.
CC = UNAFFECTED.

16.

ADDM R,M (ADD TO MEMORY)
(M) ← (M)+(R)

THE CONTENT OF R IS ADDED TO THE CONTENT OF M IN FIXED POINT FORM. THIS INSTRUCTION CAN CAUSE AN INTEGER OVERFLOW TRAP, IN WHICH CASE M IS UNCHANGED.
CC = PATTERN A.

17.

SUBM R,M (SUBTRACT FROM MEMORY)
(M) ← (M)-(R)

THE CONTENT OF R IS SUBTRACTED FROM THE CONTENT OF M IN FIXED POINT FORM. THIS INSTRUCTION CAN CAUSE AN INTEGER OVERFLOW TRAP, IN WHICH CASE M IS UNCHANGED.
CC = PATTERN A.

18.

ANDM R,M (AND MEMORY)
(M) ← (M)AND(R)

THE CONTENT OF M IS REPLACED BY THE LOGICAL "AND" OF M AND R.
CC = PATTERN B.

19.

IORM R,M (INCLUSIVE OR MEMORY)
(M) ← (M)OR(R)

THE CONTENT OF M IS REPLACED BY THE "INCLUSIVE OR" OF R AND M.
CC = PATTERN B.

20.

XORM R,M (EXCLUSIVE OR MEMORY)
(M) ← ((M)AND NOT(R)) OR (NOT(M)AND(R))

THE CONTENT OF M IS REPLACED BY THE "EXCLUSIVE OR" OF R AND M.
CC = PATTERN B.

LOGICAL INSTRUCTIONS:

21.

ADDL R,M (ADD LOGICAL)
(R) ← (R)+(M) MOD 2↑32

THE CONTENT OF M IS ADDED LOGICALLY TO REGISTER R AS 32 BIT POSITIVE INTEGERS.
CC = 0, IF NO CARRYOUT OF BIT 0 OCCURS.
CC = 1, IF A CARRYOUT OF BIT 0 OCCURS.

22.

SUBL R,M (SUBTRACT LOGICAL)
(R) ← (R)-(M) MOD 2↑32

THE CONTENT OF M IS SUBTRACTED LOGICALLY FROM REGISTER R AS 32 BIT POSITIVE INTEGERS. NOTE THAT A CARRY OUT OF BIT 0 OCCURS IF AND ONLY IF NO BORROW OCCURS.
CC = 0, IF A CARRYOUT OF BIT 0 OCCURS.
CC = 1, IF NO CARRYOUT OF BIT 0 OCCURS.

23.

MPYL R,M (MULTIPLY LOGICAL)
(R,R+1) ← (R)*(M)

THE CONTENT OF R IS MULTIPLIED LOGICALLY BY THE CONTENT OF M AS 32 BIT POSITIVE INTEGERS AND THE 64 BIT POSITIVE PRODUCT IS LEFT IN R AND R+1.
CC = PATTERN B ON THE DOUBLEWORD RESULT.

24.

DIVL R,M (DIVIDE LOGICAL)
(R) ← LOGICAL ((R,R+1)/(M))
(R+1) ← (R,R+1) MODULO (M)

THE 64 BIT POSITIVE DIVIDEND IN REGISTERS R AND R+1 IS DIVIDED LOCICALLY BY THE 32 BIT POSITIVE DIVISOR IN M. THIS INSTRUCTION CAN SET AN INTEGER OVERFLOW TRAP, IN WHICH CASE REGISTERS R AND R+1 ARE UNCHANGED.
CC = UNAFFECTED.

FLOATING POINT INSTRUCTIONS:

NOTE: IF A FLOATING POINT OVERFLOW TRAP OCCURS, THE STORED RESULT WILL BE THE TRUE RESULT DIVIDED BY 2↑512. IF A FLOATING POINT UNDERFLOW TRAP OCCURS, THE STORED RESULT WILL BE THE TRUE RESULT MULTIPLIED BY 2↑512. IF A DIVISION BY ZERO TRAP OCCURS, THE RESULT REGISTER IS UNCHANGED.

25.

FADD R,M (FLOATING ADD)
(R) ← (R)+(M)

THE CONTENT OF M IS ADDED TO REGISTER R IN FLOATING POINT. THIS INSTRUCTION CAN CAUSE FLOATING POINT OVERFLOW OR UNDERFLOW TRAPS.
CC = UNAFFECTED.

26.

FSUB R,M (FLOATING SUBTRACT)
(R) ← (R)-(M)

THE CONTENT OF M IS SUBTRACTED FROM REGISTER R IN FLOATING POINT. THIS INSTRUCTION CAN CAUSE FLOATING POINT OVERFLOW OR UNDERFLOW TRAPS.
CC = UNAFFECTED.

27.

FMYE R,M (FLOATING MULTIPLY EXTENDED)
(R,R+1) ← (R)*(M)

THE CONTENT OF R IS MULTIPLIED BY THE CONTENT OF M AND THE DOUBLE-WORD RESULT IS LEFT IN (R,R+1). THIS INSTRUCTION CAN CAUSE FLOATING POINT OVERFLOW OR UNDERFLOW TRAPS.
CC = PATTERN A ON THE DOUBLEWORD RESULT.

28.

FMPY R,M (FLOATING MULTIPLY)
(R) ← (R)*(M)

THE CONTENT OF R IS MULTIPLIED BY THE CONTENT OF M IN FLOATING POINT. THIS INSTRUCTION CAN CAUSE FLOATING POINT OVERFLOW OR UNDERFLOW TRAPS.
CC = UNAFFECTED.

29.

FDIV R,M (FLOATING DIVIDE)
(R) ← (R)/(M)

THE CONTENT OF R IS DIVIDED BY THE CONTENT OF M IN FLOATING POINT. THIS INSTRUCTION CAN CAUSE FLOATING POINT OVERFLOW, UNDERFLOW, OR DIVISION BY ZERO TRAPS.
CC = UNAFFECTED.

30.

FLT R,M (FLOAT)
(R) ← FLOAT(M)

THE INTEGER IN M IS ROUNDED OFF TO 23 SIGNIFICANT BITS AND THEN FLOATED AND PLACED IN R.
CC = UNAFFECTED.

31.

FIXR R,M (FIX AND ROUND)
(R) ← TRUNCATE((M)+.5*SIGN(M))

THE FLOATING POINT NUMBER IN M IS ROUNDED AND CONVERTED TO FIXED POINT FORM AND PLACED IN R. THIS INSTRUCTION CAN CAUSE AN INTEGER OVERFLOW TRAP.
CC = UNAFFECTED.

32.

FIXT R,M (FIX AND TRUNCATE)
R ← TRUNCATE (M)

THE FLOATING POINT NUMBER IN M IS TRUNCATED AND CONVERTED TO FIXED POINT FORM AND PLACED IN R. THIS INSTRUCTION CAN CAUSE AN INTEGER OVERFLOW TRAP.
CC = UNAFFECTED.

33.

FNEG R,M (FLOATING NEGATE)

THE NUMBER IN M, IF NON-ZERO, HAS THE SIGN BIT INVERTED AND IS THEN STORED IN R. M IS UNCHANGED.
CC = UNAFFECTED.

DOUBLE PRECISION FLOATING POINT INSTRUCTIONS.

34.

DFAD R,M (DOUBLE FLOATING ADD)
(R,R+1) ← (R,R+1)+(M,M+1)

THE CONTENT OF M,M+1 IS ADDED TO REGISTERS R,R+1 IN DOUBLE PRECISION FLOATING POINT. THIS INSTRUCTION CAN CAUSE FLOATING POINT OVERFLOW OR UNDERFLOW TRAPS.
CC = PATTERN A ON THE DOUBLEWORD RESULT.

35.

DFSB R,M (DOUBLE FLOATING SUBTRACT)
(R,R+1) ← (R,R+1)-(M,M+1)

THE CONTENT OF M,M+1 IS SUBTRACTED FROM REGISTERS R,R+1 IN DOUBLE PRECISION FLOATING POINT. THIS INSTRUCTION CAN CAUSE FLOATING POINT OVERFLOW OR UNDERFLOW TRAPS.
CC = PATTERN A ON THE DOUBLEWORD RESULT.

36.

DFMP R,M (DOUBLE FLOATING MULTIPLY)

(R,R+1) ← (R,R+1)*(M,M+1)

THE CONTENT OF R,R+1 IS MULTIPLIED BY THE CONTENT OF M,M+1 IN DOUBLE PRECISION FLOATING POINT. THIS INSTRUCTION CAN CAUSE FLOATING POINT OVERFLOW OR UNDERFLOW TRAPS.
CC = PATTERN A ON THE DOUBLEWORD RESULT.

37.

DFDV R,M (DOUBLE FLOATING DIVIDE)
(R,R+1) ← (R,R+1)/(M,M+1)

THE CONTENT OF R,R+1 IS DIVIDED BY THE CONTENT OF M,M+1 IN DOUBLE PRECISION FLOATING POINT. THIS INSTRUCTION CAN CAUSE FLOATING POINT OVERFLOW, UNDERFLOW, OR DIVISION BY ZERO TRAPS.
CC = PATTERN A ON THE DOUBLEWORD RESULT.

INCREMENT AND TEST INSTRUCTIONS:

38.

```
INCM R,M          (INCREMENT M)
      (M) ← (M)+1
```

THE CONTENT OF M IS INCREMENTED LOGICALLY BY 1.
CC = PATTERN C ON (M):(R) IF R#0
ELSE PATTERN C ON (M):0.

39.

```
DECM R,M          (DECREMENT M)
      (M) ← (M)-1
```

THE CONTENT OF M IS DECREMENTED LOGICALLY BY 1.
CC = IF R#0, PATTERN C ON (M):(R).
ELSE PATTERN C ON (M):0.

40.

```
INCR R,M          (INCREMENT R)
      (R) ← (R)+1
```

THE CONTENT OF R IS INCREMENTED LOGICALLY BY 1.
CC = PATTERN C ON (R):(M)

41.

```
DECR R,M          (DECREMENT R)
      (R) ← (R)-1
```

THE CONTENT OF R IS DECREMENTED LOGICALLY BY 1.
CC = PATTERN C ON (R):(M)

UNARY OPERATORS:

ALL UNARY INSTRUCTIONS SHARE A SINGLE OPCODE. THE EIGHT INSTRUCTIONS ARE DISTINGUISHED BY THEIR R FIELDS.

42A.

```
SSP  M          (SET SIGN PLUS)
     M[0] ← 0
```

THE SIGN BIT OF M IS SET TO 0.
CC = PATTERN A.

42B.

```
SSM  M          (SET SIGN MINUS)
     M[0] ← 1
```

THE SIGN BIT OF M IS SET TO 1.
CC = PATTERN A.

42C.

```
CHS  M          (CHANGE SIGN)
     M[0] ← NOT (M[0])
```

THE SIGN BIT OF M IS COMPLEMENTED.
CC = PATTERN A.

42D.

```
NEG  M          (NEGATE)
     (M) ← -(M)
```

THE CONTENT OF M IS NEGATED.
THIS INSTRUCTION CAN CAUSE AN INTEGER OVERFLOW TRAP, IN WHICH CASE (M) IS UNCHANGED.
CC = PATTERN A.

42E.

```
ABS  M          (ABSOLUTE VALUE)
     (M) ← ABS(M)
```

THE CONTENT OF M IS REPLACED BY ITS ABSOLUTE VALUE.
THIS INSTRUCTION CAN CAUSE AN INTEGER OVERFLOW TRAP, IN WHICH CASE (M) IS UNCHANGED.
CC = PATTERN A.

42F.

CMPL M (COMPLEMENT)
(M) ↰ NOT(M)

THE CONTENT OF M IS REPLACED BY ITS ONE'S COMPLEMENT.
CC = PATTERN B.

42G.

ZERO M (ZERO)
(M) ↰ 0

THE CONTENT OF M IS SET TO ZERO.
CC = UNAFFECTED.

42H.

TEST M (TEST)
CC ↰ PATTERN A (M)

THE CONDITION CODE IS SET TO PATTERN A DEPENDING ON
THE CONTENT OF M.

DOUBLEWORD INSTRUCTIONS:

43.

DLOD R,M (DOUBLE LOAD)
(R,R+1)←(M,M+1)

IN ADDRESS COMPUTATION, THE INDEX REGISTER CONTAINS A DOUBLE-WORD COUNT RATHER THAN A WORD COUNT. THE WORDS IN M AND M+1 ARE LOADED INTO R AND R+1.
CC = PATTERN A ON THE DOUBLEWORD RESULT.

44.

DSTO R,M (DOUBLE STORE)
(M,M+1)←(R,R+1)

IN ADDRESS COMPUTATION, THE INDEX REGISTER CONTAINS A DOUBLE-WORD COUNT RATHER THAN A WORD COUNT. THE WORDS IN R AND R+1 ARE STORED IN M AND M+1.
CC = PATTERN A ON THE DOUBLE-WORD RESULT.

45.

DADD R,M (DOUBLE ADD)
(R,R+1)←(R,R+1)+(M,M+1)

IN ADDRESS COMPUTATION, THE INDEX REGISTER CONTAINS A DOUBLE-WORD COUNT RATHER THAN A WORD COUNT. THE DOUBLE-WORD INTEGERS IN R, R+1 AND M, M+1 ARE ADDED, AND THE RESULT IS LEFT IN R AND R+1. THIS INSTRUCTION CAN CAUSE AN INTEGER OVERFLOW TRAP.
CC = PATTERN A ON THE DOUBLEWORD RESULT.

46.

DSUB R,M (DOUBLE SUBTRACT)
(R,R+1)←(R,R+1)-(M,M+1)

IN ADDRESS COMPUTATION, THE INDEX REGISTER CONTAINS A DOUBLE-WORD COUNT RATHER THAN A WORD COUNT. THE DOUBLE-WORD INTEGER IN M, M+1 IS SUBTRACTED FROM THAT IN R, R+1, AND THE RESULT IS LEFT IN R,R+1. THIS INSTRUCTION CAN CAUSE AN INTEGER OVERFLOW TRAP.
CC = PATTERN A ON THE DOUBLEWORD RESULT.

47.

DCOM R,M (DOUBLE COMPARE)
(R,R+1):(M,M+1)

IN ADDRESS COMPUTATION, THE INDEX REGISTER CONTAINS A DOUBLE-WORD COUNT RATHER THAN A WORD COUNT. THE DOUBLE-WORD INTEGERS IN R, R+1 AND M, M+1 ARE COMPARED NUMERICALLY.
CC = PATTERN C.

48.

DTST M (DOUBLE TEST)
CC ← PATTERN A (M,M+1)

THE CONDITION CODE IS SET TO PATTERN A DEPENDING ON THE DOUBLEWORD CONTENT OF M AND M+1.

BYTE INSTRUCTIONS:

49.

```
LDB  R,M          (LOAD BYTE)
     IF R-TYPE ADDRESSING IS NOT USED THEN
       BEGIN
       M ← (A)+TRUNCATE(X/4)
       BYTE ADDR←(X) MODULO 4
       (R[0-23])←0
       (R[24-31])←(BYTE [BYTE ADDR] OF (M))
       END
     ELSE
       BEGIN
       (R[0-23])←0
       (R[24-31])←BYTE REFERENCED BY DATA LABEL IN REGISTER X.
       END
```

IN ADDRESS COMPUTATION FOR THIS INSTRUCTION, THE INDEX REGISTER IS ASSUMED TO CONTAIN A NUMBER OF BYTES RATHER THAN A NUMBER OF WORDS. THE RESULT OF THE ADDRESS COMPUTATION IS A BYTE ADDRESS M. THE BYTE AT M IS LOADED INTO THE LOW-ORDER 8 BITS OF R, AND THE HIGH-ORDER 24 BITS OF R ARE SET TO ZERO.
IF R-TYPE ADDRESSING IS USED, THE REGISTER SPECIFIED BY THE LAST 3 BITS OF THE A FIELD CONTAINS A DATA LABEL WHOSE OFFSET FIELD CONTAINS A BYTE COUNT.
CC = UNAFFECTED.

50.

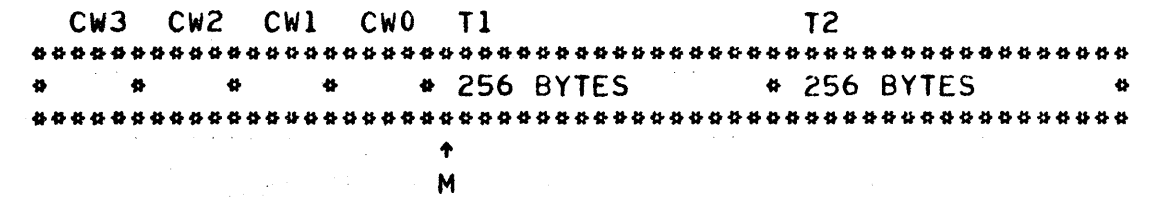
```
STB  R,M          (STORE BYTE)
     IF R-TYPE ADDRESSING IS NOT USED THEN
       BEGIN
       M←(A)+TRUNCATE (X/4)
       BYTE ADDR ← (X) MODULO 4
       (BYTE [BYTE ADDR] OF (M)←(R[24-31])
       END
     ELSE
       BYTE REFERENCED BY DATA LABEL IN REGISTER X ← (R[24-31])
```

IN ADDRESS COMPUTATION FOR THIS INSTRUCTION, THE INDEX REGISTER IS ASSUMED TO CONTAIN A NUMBER OF BYTES RATHER THAN A NUMBER OF WORDS. THE RESULT OF THE ADDRESS COMPUTATION IS A BYTE ADDRESS M. THE LOW-ORDER 8 BITS OF REGISTER R ARE STORED INTO THE PROPER BYTE OF M.
IF R-TYPE ADDRESSING IS USED, THE REGISTER SPECIFIED BY THE LAST 3 BITS OF THE A FIELD CONTAINS A DATA LABEL WHOSE OFFSET FIELD CONTAINS A BYTE COUNT.
CC = UNAFFECTED.

51.

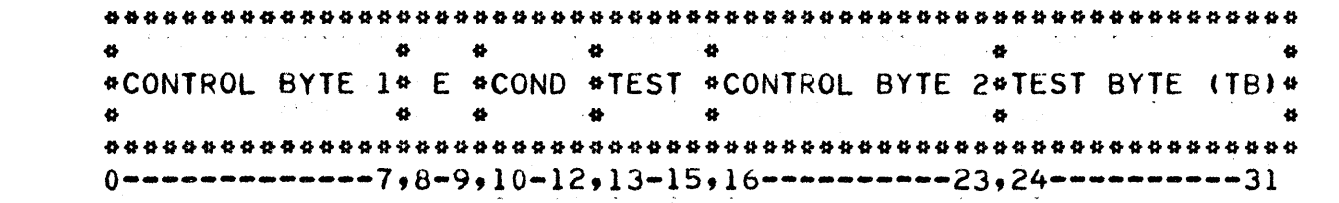
EDIT R,X,M (EDIT BYTES)

REGISTER R CONTAINS A RELATIVE BYTE ADDRESS, CALLED THE SOURCE POINTER. WE WILL REFER TO THE BYTE POINTED TO BY REGISTER R AS R1.
REGISTER R+1 CONTAINS A RELATIVE BYTE ADDRESS, CALLED THE DESTINATION POINTER. WE WILL REFER TO THE BYTE POINTED TO BY REGISTER R+1 AS R2.
REGISTER X CONTAINS AN EDIT CONTROL WORD (FORMAT BELOW).
REGISTER X+1 CONTAINS A BYTE COUNT IN TWOS-COMPLEMENT FORM, WHICH MAY BE INCREMENTED TO ZERO TO TERMINATE THE INSTRUCTION.
M POINTS TO A TRANSLATION TABLE (INDEXING IS NOT ALLOWED.) THE TABLE HAS THE FOLLOWING STRUCTURE:



THE SECTION OF THE TABLE BEGINNING AT LOCATION M IS CALLED T1, AND SPECIFIES A BYTE-TO-BYTE TRANSLATION FUNCTION. ADJACENT TO T1 IN MEMORY MAY BE ANOTHER 256-BYTE TRANSLATION TABLE T2. THE FOUR WORDS PRECEDING T1 IN MEMORY MAY CONTAIN ADDITIONAL CONTROL WORDS TO BE USED BY THE INSTRUCTION.

THE FORMAT OF THE EDIT CONTROL WORD IS:



THE FORMAT OF EACH CONTROL BYTE IS:

BITS	FIELD
0-2	A (ACTION)
3	I1 (INCREMENT REG. R)
4	I2 (INCREMENT REG. R+1)
5	I3 (INCREMENT REG. X+1)
6	H (HALT)
7	C (NEW CONTROL WORD)

THE INSTRUCTION OPERATION IS AS FOLLOWS:

1. THE TEST FIELD DESIGNATES A COMPARISON OF TWO BYTES:

TEST=0	TEST IS TRUE (SEE BELOW)
TEST=1	R1:R2
TEST=2	TB:T1(R1)
TEST=3	TB:T1(R2)
TEST=4	TB:R1
TEST=5	TB:R2
TEST=6	TB:T2(R1)
TEST=7	TB:T2(R2)

2. THE CONDITION CODE IS SET TO PATTERN C ON THE RESULT OF THE TEST, AND IS THEN ANDED WITH THE COND FIELD. IF THE RESULT IS NONZERO, OR IF TEST=0, THE REMAINDER OF THE CYCLE IS CONTROLLED BY CONTROL BYTE 1. OTHERWISE, THE REMAINDER OF THE CYCLE IS CONTROLLED BY CONTROL BYTE 2.

3. THE A FIELD OF THE CONTROL BYTE NOW DETERMINES THE ACTION, WHICH IS THE MOVE OF A SINGLE BYTE:

A=0	NO ACTION
A=1	R1 ← T1(R1)
A=2	R1 ← T2(R1)
A=3	R1 ← TB
A=4	R2 ← R1
A=5	R2 ← T1(R1)
A=6	R2 ← T2(R1)
A=7	R2 ← TB

4. IF THE I1 BIT = 1, INCREMENT REGISTER R.
   IF THE I2 BIT = 1, INCREMENT REGISTER R+1.
   IF THE I3 BIT = 1, INCREMENT REGISTER X+1.
   IF REGISTER X+1 IS THEN 0, SET CC=0 AND TERMINATE THE INSTRUCTION.
   IF THE H BIT = 1, SET CC=1 AND TERMINATE THE INSTRUCTION.
   IF THE C BIT = 1, REPLACE THE CONTENT OF REGISTER X BY THE CONTROL WORD IN LOCATION M-1-E.

5. IF THE INSTRUCTION HAS NOT TERMINATED, RETURN TO STEP 1.

NOTE: REGISTER R, R+1, X, AND X+1 SHOULD NOT OVERLAP.

THE EDIT INSTRUCTION CAN PERFORM CERTAIN FUNCTIONS WITHOUT THE AID OF TRANSLATION TABLES. TWO EXAMPLES ARE GIVEN:

1. TO MOVE A FIXED NUMBER OF BYTES FROM A SOURCE FIELD TO A DESTINATION FIELD, SET UP THE FOLLOWING:

   REGISTER R = SOURCE FIELD POINTER
   REGISTER R+1 = DESTINATION FIELD POINTER
   REGISTER X = 10011100000000000000000000000000
   REGISTER X+1 = TWOS-COMPLEMENT OF BYTE COUNT

2. TO COMPARE A FIXED NUMBER OF BYTES OF FIELD 1 TO FIELD 2, SET UP THE FOLLOWING:

   REGISTER R = FIELD 1 POINTER
   REGISTER R+1 = FIELD 2 POINTER
   REGISTER X = 00011100001000010000001000000000
   REGISTER X+1 = TWOS-COMPLEMENT OF BYTE COUNT

   WHEN THE INSTRUCTION TERMINATES, CC=0 IF THE FIELDS WERE EQUAL. IF THE FIELDS WERE NOT EQUAL, CC=1 AND REGISTERS R AND R+1 POINT TO THE BYTES WHICH COMPARED UNEQUAL.

COMPARES:

52.

ACM R,M (ARITHMETIC COMPARE)
(R):(M)

THE CONTENT OF R IS COMPARED NUMERICALLY WITH THE CONTENT OF M IN FIXED POINT FORM.
CC = PATTERN C.

53.

LCM R,M (LOGICAL COMPARE)
(R):(M)

THE CONTENT OF R IS COMPARED LOGICALLY WITH THE CONTENT OF M, AS 32-BIT POSITIVE NUMBERS.
CC = PATTERN C.

54.

FCM R,M (FLOATING COMPARE)
(R):(M)

THE CONTENT OF R IS COMPARED WITH THE CONTENT OF M AS FLOATING POINT NUMBERS.
CC = PATTERN C.

55.

DFCM R,M (DOUBLE PRECISION FLOATING COMPARE)

THE CONTENT OF R AND R+1 IS COMPARED WITH THE CONTENT OF M AND M+1 AS DOUBLE PRECISION FLOATING POINT NUMBERS.
CC = PATTERN C.

BRANCHES:

56.

BCC R,M (BRANCH ON CONDITION CODE)

A BRANCH TO LOCATION M IS EXECUTED UNDER THE FOLLOWING CONDITIONS:

```
IF R = 0, NEVER BRANCH
   R = 1, BRANCH IF CC = 0
   R = 2, BRANCH IF CC = 1
   R = 3, BRANCH IF CC = 0 OR 1
   R = 4, BRANCH IF CC = 2
   R = 5, BRANCH IF CC = 0 OR 2
   R = 6, BRANCH IF CC = 1 OR 2
   R = 7, ALWAYS BRANCH
```

CC = UNAFFECTED.

57.

BRO R,M (BRANCH IF REGISTER ODD)

IF (R[31])=1 THEN (P)←(M) ELSE (P)←(P)+1

A BRANCH TO LOCATION M IS EXECUTED IF THE NUMBER IN REGISTER R IS ODD.
CC = UNAFFECTED.

58.

BRE R,M (BRANCH IF REGISTER EVEN)

IF (R[31])=0 THEN (P)←(M) ELSE (P)←(P)+1

A BRANCH TO LOCATION M IS EXECUTED IF THE NUMBER IN REGISTER R IS EVEN.
CC = UNAFFECTED.

59.

BRP R,M (BRANCH IF REGISTER POSITIVE)
IF (R)>0 THEN (P)←(M) ELSE (P)←(P)+1

A BRANCH TO LOCATION M IS EXECUTED IF THE NUMBER IN REGISTER R IS POSITIVE.
CC = UNAFFECTED.

60.

BRNP R,M (BRANCH IF REGISTER NON-POSITIVE)
IF (R)<=0 THEN (P)←(M) ELSE (P)←(P)+1

A BRANCH TO LOCATION M IS EXECUTED IF THE NUMBER IN REGISTER R IS LESS THAN OR EQUAL TO ZERO.
CC = UNAFFECTED.

61.

BRN R,M (BRANCH IF REGISTER NEGATIVE)
IF (R)<0 THEN (P)←(M) ELSE (P)←(P)+1

A BRANCH TO LOCATION M IS EXECUTED IF THE NUMBER IN REGISTER R IS LESS THAN ZERO.
CC = UNAFFECTED.

62.

BRNN R,M (BRANCH IF REGISTER NON-NEGATIVE)
IF (R)>=0 THEN (P)←(M) ELSE (P)←(P)+1

A BRANCH TO LOCATION M IS EXECUTED IF THE NUMBER IN REGISTER R IS GREATER THAN OR EQUAL TO ZERO.
CC = UNAFFECTED.

63.

BRZ R,M (BRANCH IF REGISTER ZERO)
IF (R)=0 THEN (P)←(M) ELSE (P)←(P)+1

A BRANCH TO LOCATION M IS EXECUTED IF THE NUMBER IN REGISTER R IS ZERO.
CC = UNAFFECTED.

64.

BRNZ R,M (BRANCH IF REGISTER NON-ZERO)
IF (R)#0 THEN (P)←(M) ELSE (P)←(P)+1

A BRANCH TO LOCATION M IS EXECUTED IF THE NUMBER IN REGISTER R IS NON-ZERO.
CC = UNAFFECTED.

SHIFTS:

65.

LSL R,X,A (LEFT SHIFT LOGICAL)

66.

RSL R,X,A (RIGHT SHIFT LOGICAL)

THE CONTENTS OF REGISTERS R AND X ARE SHIFTED LEFT (RIGHT) TOGETHER A (MOD 64) BITS (R IS ASSUMED TO BE ON THE LEFT) AND THE VACATED BITS ARE SET TO ZEROS. IF R = X, ONLY ONE REGISTER IS SHIFTED.
CC = UNAFFECTED.

67.

LSA R,X,A (LEFT SHIFT ARITHMETIC)

68.

RSA R,X,A (RIGHT SHIFT ARITHMETIC)

THE CONTENTS OF REGISTERS R AND X ARE SHIFTED LEFT (RIGHT) TOGETHER A (MOD 64) BITS (R IS ASSUMED TO BE ON THE LEFT), EXCEPT THAT THE LEFTMOST (SIGN) BIT OF R IS UNCHANGED. VACATED LOW-ORDER BITS ARE SET TO ZERO; VACATED HIGH-ORDER BITS ARE SET TO THE SIGN BIT. IF R = X, ONLY ONE REGISTER IS SHIFTED, AND ITS SIGN BIT IS PRESERVED.
LSA OVERFLOWS IF BIT 1 IS DIFFERENT FROM BIT 0 BEFORE SHIFTING.
CC = UNAFFECTED.

69.

LSC R,X,A (LEFT SHIFT CIRCULAR)

70.

RSC R,X,A (RIGHT SHIFT CIRCULAR)

THE CONTENTS OF REGISTERS R AND X ARE ROTATED LEFT (RIGHT) TOGETHER A (MOD 64) BITS (R IS ASSUMED TO BE ON THE LEFT).
IF R = X, ONLY ONE REGISTER IS ROTATED.
CC = UNAFFECTED.

SPECIAL PURPOSE:

71.

EXF R,X,A (EXTRACT FIELD)

A1 = BITS 22-26 OF A; A2 = BITS 27-31 OF A.
BITS A1, A1+1, ..., A2 OF REGISTER X ARE PLACED IN THE LEAST SIGNIFICANT END OF R. THE REMAINING BITS OF R ARE SET TO ZERO.
CC = UNAFFECTED.

72.

DPF R,X,A (DEPOSIT FIELD)

A1 = BITS 22-26 OF A; A2 = BITS 27-31 OF A.
THE A2-A1+1 LEAST SIGNIFICANT BITS OF X ARE PLACED IN BITS A1, A1+1, ..., A2 OF R. THE REMAINING BITS OF R ARE UNCHANGED.
CC = UNAFFECTED.

73.

MVW R,X (MOVE WORDS)

THE FIELD BEGINNING AT THE SEGMENT ADDRESS IN R IS MOVED INTO THE FIELD BEGINNING AT THE SEGMENT ADDRESS IN R+1. THE WORD COUNT TO BE MOVED IS IN X. DURING EXECUTION, REGISTERS R, R+1, AND X ARE INCREMENTED, UNTIL R AND R+1 POINT TO THE NEXT WORD AFTER THEIR RESPECTIVE FIELDS AND X CONTAINS ZERO. INDIRECT ADDRESSING AND A ARE IGNORED.
CC = UNAFFECTED.

74.

ASCI R,X (ASCII TO INTEGER)

THE ASCII FIELD BEGINNING AT THE SEGMENT BYTE ADDRESS CONTAINED IN X+1 IS CONVERTED TO INTEGER FORMAT UNTIL A NON-NUMERIC CHARACTER IS ENCOUNTERED OR THE MAXIMUM NUMBER OF CHARACTERS TO CONVERT (CONTAINED IN X), IS COMPLETED. THE RESULT IS ADDED TO REGISTERS R AND R+1. DURING EXECUTION, X IS INCREMENTED TO ZERO OR UNTIL A NON-NUMERIC CHARACTER IS ENCOUNTERED. REGISTERS R AND R+1 SHOULD BE INITIALIZED TO THE DESIRED VALUE (USUALLY ZERO) BEFORE ISSUING THIS INSTRUCTION. THIS INSTRUCTION CAN SET ARITHMETIC OVERFLOW.
CC = UNAFFECTED.

75.

BCDI R,M (BINARY-CODED-DECIMAL TO INTEGER)

THE CONTENT OF M IS INTERPRETED AS EIGHT 4-BIT BINARY-CODED-DECIMAL DIGITS. THIS NUMBER IS CONVERTED TO A BINARY INTEGER AND PLACED IN R. INVALID BCD CODES ARE TREATED AS ZEROS; HOWEVER, AN INTEGER OVERFLOW TRAP TAKES PLACE IF AN INVALID BCD CODE IS FOUND.
CC = UNAFFECTED.

76.

IBCD R,M (INTEGER TO BINARY-CODED-DECIMAL)

THE POSITIVE INTEGER IN M IS CONVERTED TO BINARY-CODED-DECIMAL FORM (EIGHT 4-BIT DIGITS) AND PLACED IN R AND R+1.
CC = UNAFFECTED.

77.

XEQ R,M (EXECUTE)

THE INSTRUCTION IN M IS FETCHED, "INCLUSIVE-ORED" WITH THE CONTENT OF R, AND EXECUTED. THE INSTRUCTION LEFT IN M IS UNCHANGED. IF R=0, THE INSTRUCTION IS NOT MODIFIED.
CC = SET BY THE EXECUTED INSTRUCTION.

78.

PSDS R,M (PUSH DATA STACK)

THE ADDRESS M POINTS TO A DATA LABEL. THE INDEX FIELD OF THIS WORD IS INCREMENTED, AND THE WORD IS THEN USED AS IF THE INSTRUCTION WAS AN INDIRECT STORE. AN ARRAY BOUNDS TRAP OCCURS IF THE DATA STACK AREA OVERFLOWS.
CC=UNAFFECTED.

79.

PPDS R,M (POP DATA STACK)

THE ADDRESS M POINTS TO A DATA LABEL. A LOAD THROUGH THIS WORD IS PERFORMED, AND THEN THE INDEX FIELD OF THE DATA LABEL IS DECREMENTED.
CC = UNAFFECTED.

80.

LDQS M (LOAD Q AND S)
(Q)←(M[16-31])+(DB)
(S)←(M[0-15])+(DB)

REGISTER DB IS ADDED TO THE CONTENT OF EACH HALFWORD OF M, AND THE RESULTS ARE LOADED INTO Q AND S.
CC = UNAFFECTED.

81.

STQS M (STORE Q AND S)
(M[0-15])←(S)-(DB)
(M[16-31])←(Q)-(DB)

REGISTER DB IS SUBTRACTED FROM REGISTERS Q AND S. Q-DB IS STORED IN BITS 16-31 OF M; S-DB IS STORED IN BITS 0-15 OF M. REGISTERS DB, Q, AND S ARE UNCHANGED.
CC = UNAFFECTED.

82.

STST M (STORE STATUS)
(M)↰(STATUS)

THE STATUS WORD IS STORED IN M.
CC = UNAFFECTED.

83.

STRZ M (STORE Z)
(Z) ↰ (M)-DB

REGISTER DB IS SUBTRACTED FROM REGISTERS Z. Z-DB IS STORED IN M; REGISTERS DB AND Z ARE UNCHANGED.
THIS IS A PRIVILEGED INSTRUCTION.
CC = UNAFFECTED.

84.

LMS M (LOAD MEMORY ONTO STACK)
S ↰ S+1
(S)↰(M)

THE STACK POINTER S IS INCREMENTED BY ONE. THE CONTENT OF M IS PLACED INTO THE CORE WORD POINTED TO BY S.
THIS INSTRUCTION CAN CAUSE A STACK OVERFLOW TRAP.
CC = PATTERN A.

85.

ADS M (ADD TO S REGISTER)
(S) ↰ (S)+(M)

THE CONTENT OF M IS ADDED TO THE S REGISTER. A STACK INTEGRITY TRAP RESULTS IF THE CONDITION Q<=S<=Z DOES NOT HOLD AFTER THE ADDITION
CC = UNAFFECTED.

86.

LAR R,M (LOAD ADDRESS INTO REGISTER)
R[0-11]↰0;
R[12-31]↰M;

THE RELATIVE ADDRESS M IS COMPUTED,WITH MULTILEVEL INDIRECT ADDRESSING ALLOWED,AND PLACED IN THE LOW ORDER BITS OF R.
CC = UNAFFECTED.

87.

LBAR R,M (LOAD BYTE ADDRESS INTO REGISTER)

THIS INSTRUCTION IS IDENTICAL TO LAR EXCEPT THAT THE REGISTER IS SHIFTED TWO BITS LEFT AFTER LOADING. IF INDEXING IS USED, REGISTER X CONTAINS A BYTE ADDRESS WHICH IS ADDED TO R AFTER THE LEFT SHIFT.
CC = UNAFFECTED.

88.

LAS R,M (LOAD ADDRESS ONTO STACK)
S←S+1
(S)←M

THE STACK POINTER S IS INCREMENTED BY ONE. THE RELATIVE ADDRESS M IS COMPUTED, WITH MULTILEVEL INDIRECT ADDRESSING ALLOWED, AND STORED IN THE CORE WORD POINTED TO BY S.
CC = UNAFFECTED.

89.

LFP R,M (LOAD FROM PROGRAM)

THE WORD IN M IS A PROGRAM LABEL. THE WORD IT REFERENCES IS LOADED INTO R.
CC = UNAFFECTED.

90.

```
PUSH R,X        (PUSH)
     TEMP ← R;
     WHILE TEMP # X DO
          BEGIN
               S ← S+1;
               (S) ← (TEMP);
               TEMP ← TEMP+1 MOD 8
          END;
     S ← S + 1;
     (S) ← (TEMP);
```

ALL THE WORDS FROM REGISTER R TO REGISTER X ARE PUSHED ONTO THE STACK. IF BIT 31 OF THE INSTRUCTION IS 1 AND THE MACHINE IS IN PRIVILEGED MODE, STACK OVERFLOW IS NOT CHECKED.
CC = UNAFFECTED.

91.

```
POP  R,X        (POP)
     TEMP ← X;
     WHILE TEMP # R DO
          BEGIN
               (TEMP) ← (S);
               S ← S-1;
               TEMP ← TEMP-1 MOD 8;
          END;
     (TEMP) ← (S);
     S ← S - 1;
```

THE STACK IS POPPED AND ITS DATA IS STORED IN REGISTERS FROM X DOWN TO R.
CC = UNAFFECTED.

92.

```
PCAL R,M         (PROCEDURE CALL)
     IF R=0 THEN
          BEGIN
               S ← S+1;
               (S) ← PROGRAM LABEL REFERENCE TO P+1;
               S ← S+1;
               (S) ← CC, IDFF, PMFF, S-Q;
               Q ← S;
               P ← M;
          END
     ELSE
          BEGIN
               (R) ← PROGRAM LABEL REFERENCE TO P+1;
               P ← M;
          END;
```

A BRANCH TO M OCCURS. IF R=0, A NEW STACK MARKER IS CREATED AND PLACED ON THE STACK. OTHERWISE, THE RETURN ADDRESS IS PLACED IN R.
CC = UNAFFECTED.

93.

ECAL A (EXTERNAL CALL)

THE OPERATION OF THIS INSTRUCTION IS IDENTICAL TO THAT OF PCAL, EXCEPT THAT THE TRANSFER IS MADE TO THE SEGMENT WHOSE XNO IS GIVEN BY BITS 14-21 OF THE INSTRUCTION AND WHOSE SNO IS GIVEN BY BITS 22-31 OF THE INSTRUCTION. THE INDIRECT BIT IS IGNORED IN THIS INSTRUCTION.
CC = UNAFFECTED.

94.

EXIT A (EXIT)

THIS INSTRUCTION IS USED TO RETURN FROM ROUTINES CALLED BY PCAL AND ECAL. THE STACK MARKER WORD IN (Q-1) SPECIFIES THE RETURN ADDRESS. CC IS RESTORED FROM THE CC FIELD IN (Q). IF PMFF=1, THE IDFF AND THE PMFF BITS IN (Q) ARE USED TO RESET IDFF AND PMFF. FINALLY, S IS SET TO Q-A AND Q IS SET TO Q-(Q[16-31]) (THE QDECR FIELD OF THE STACK MARKER WORD.) THE INDIRECT BIT IS IGNORED IN THIS INSTRUCTION.
CC = UNAFFECTED.

95.

TRAC (TRACE)

THE TRACE DISABLE BIT OF THE STATUS REGISTER IS SET EQUAL TO BIT 0 OF THE INSTRUCTION WORD.
CC = UNAFFECTED.

PRIVILEGED: (PRIVILEGED INSTRUCTIONS ARE EXECUTABLE ONLY IN PRIVILEGED MODE)

96.

LSTA M (LOAD STATUS)
(STATUS)↰(M)

THE STATUS WORD IS LOADED FROM THE CONTENT OF M.
THIS IS A PRIVILEGED INSTRUCTION.

97.

LODZ M (LOAD Z)
(Z) ↰ (M)+DB

REGISTER DB IS ADDED TO THE CONTENT OF M, AND THE RESULT IS LOADED INTO Z.
THIS IS A PRIVILEGED INSTRUCTION.
CC = UNAFFECTED.

98.

LDDB M (LOAD DB)
DB ↰ (M)

THE CONTENT OF M IS LOADED INTO REGISTER DB. THIS IS A PRIVILEGED INSTRUCTION.
CC = UNAFFECTED.

99.

STDB M (STORE DB)
M ↰ (DB)

THE CONTENT OF DB IS STORED INTO M. THIS IS A PRIVILEGED INSTRUCTION.
CC = UNAFFECTED.

100.

LDST M (LOAD DST)
DST ↰ (M)[12-31]
DSL ↰ (M)[0-11]

THE DST AND DSL REGISTERS ARE LOADED FROM THE WORD CONTAINED IN M.
THIS IS A PRIVILEGED INSTRUCTION.
CC = UNAFFECTED.

101.

SDST M (STORE DST)
M[0-11] ↰ DSL
M[12-31] ↰ DST

THE DST AND DSL REGISTERS ARE STORED INTO LOCATION M. THIS IS
A PRIVILEGED INSTRUCTION.
CC = UNAFFECTED.

102.

INT M (INTERROGATE)
M ↰ (INR)

THE CONTENT OF THE INTERRUPT REGISTER IS PLACED IN M.
THIS IS A PRIVILEGED INSTRUCTION.
CC = UNAFFECTED.

103.

PAUS (PAUSE)

THE MACHINE IDLES UNTIL AN INTERRUPT OCCURS.
THIS IS A PRIVILEGED INSTRUCTION.
CC = UNAFFECTED.

104.

SIO R,M (START I/O)

THE LEFT 12 BITS OF REGISTER R CONTAIN THE I/O PROCESSOR NUMBER (4 BITS) AND THE DEVICE NUMBER (8 BITS). M IS THE ADDRESS OF AN I/O PROGRAM WHICH IS TO BE EXECUTED BY THE I/O PROCESSOR. AN I/O STATUS HALFWORD IS RETURNED TO THE RIGHT HALF OF REGISTER R BY THE I/O PROCESSOR.
THIS IS A PRIVILEGED INSTRUCTION.
CC = UNAFFECTED.

105.

TIO R (TEST I/O)

THE LEFT 12 BITS OF REGISTER R CONTAIN THE I/O PROCESSOR NUMBER (4 BITS) AND THE DEVICE NUMBER (8 BITS). A HALFWORD CONTAINING THE STATUS OF THE DEVICE IS RETURNED TO THE RIGHT HALF OF REGISTER R.
THIS IS A PRIVILEGED INSTRUCTION.
CC = UNAFFECTED.

106.

RIO R,M (READ I/O)

THE LEFT 12 BITS OF REGISTER R CONTAIN THE I/O PROCESSOR NUMBER (4 BITS) AND THE DEVICE NUMBER (8 BITS). IF SUCCESSFUL, A DATA INBOUND HALFWORD IS READ DIRECTLY FROM THIS DEVICE INTO THE RIGHT HALF OF M. THE LEFT HALF OF M IS ZEROED. PROCESSING DOES NOT CONTINUE UNTIL THE READ IS COMPLETED. THE DEVICE STATUS HALFWORD IS RETURNED TO THE RIGHT HALF OF REGISTER R.
THIS IS A PRIVILEGED INSTRUCTION.
CC = UNAFFECTED.

107.

WIO R,M (WRITE I/O)

THE LEFT 12 BITS OF REGISTER R CONTAIN THE I/O PROCESSOR NUMBER (4 BITS) AND THE DEVICE NUMBER (8 BITS). A HALFWORD IS WRITTEN DIRECTLY FROM THE RIGHT HALFWORD OF M INTO THE DEVICE DATA OUTBOUND HALFWORD. PROCESSING DOES NOT CONTINUE UNTIL THE WRITE IS COMPLETED. THE DEVICE STATUS HALFWORD IS RETURNED TO THE RIGHT HALF OF REGISTER R.
THIS IS A PRIVILEGED INSTRUCTION.
CC = UNAFFECTED.

108.

CIO R,M (COMMAND I/O)

THE LEFT 12 BITS OF REGISTER R CONTAIN THE I/O PROCESSOR NUMBER (4 BITS) AND THE DEVICE NUMBER (8 BITS). A HALFWORD IS WRITTEN DIRECTLY FROM THE RIGHT HALFWORD OF M INTO THE DEVICE COMMAND REGISTER. PROCESSING CONTINUES IMMEDIATELY. THIS COMMAND CAN ALSO BE USED TO SEND A HALFWORD TO ANOTHER CPU OR TO ANY OTHER MODULE WHOSE NUMBER IS PLACED IN REGISTER R.
THIS IS A PRIVILEGED INSTRUCTION.
CC = UNAFFECTED.

DECIMAL INSTRUCTIONS

109.

ADDD R,M (ADD DECIMAL)
(R) ← (M) + (R)

IN ADDRESS COMPUTATION FOR THIS INSTRUCTION, THE INDEX REGISTER IS ASSUMED TO CONTAIN A NUMBER OF BYTES RATHER THAN A NUMBER OF WORDS. ADDRESS COMPUTATION RESULTS IN A BYTE ADDRESS M. THE LOW-ORDER FOUR BITS OF THE BYTE AT M ARE ADDED DECIMALLY TO THE LOW-ORDER FOUR BITS OF REGISTER R. THE CARRY-IN IS TAKEN AS 0 OR 1 ACCORDING TO WHETHER THE CONDITION CODE IS 0 OR NONZERO. THE CARRY-OUT IS LOADED INTO THE CONDITION CODE. INVALID DIGITS ARE TREATED AS ZEROS, AND CAUSE A TRAP.
CC = 0 IF NO CARRY-OUT
CC = 1 IF A CARRY-OUT OCCURS.

110.

SUBD R,M (SUBTRACT DECIMAL)
(R) ← (R) - (M)

IN ADDRESS COMPUTATION FOR THIS INSTRUCTION, THE INDEX REGISTER IS ASSUMED TO CONTAIN A NUMBER OF BYTES RATHER THAN A NUMBER OF WORDS. ADDRESS COMPUTATION RESULTS IN A BYTE ADDRESS M. THE LOW-ORDER FOUR BITS OF THE BYTE AT M ARE SUBTRACTED DECIMALLY FROM THE LOW-ORDER FOUR BITS OF REGISTER R. IF THE CONDITION CODE IS 1, A BORROW-IN IS ASSUMED (THE CONTENT OF R IS DECREMENTED BY 1 BEFORE SUBTRACTING.) IF THE SUBTRACTION RESULTS IN A BORROW-OUT (BECAUSE (R) < (M)), THE CONDITION CODE IS SET TO 1; ELSE THE CONDITION CODE IS SET TO 0. INVALID DIGITS ARE TREATED AS ZEROS, AND CAUSE A TRAP.
CC = 1 IF BORROW-OUT.
CC = 0 IF NO BORROW-OUT.

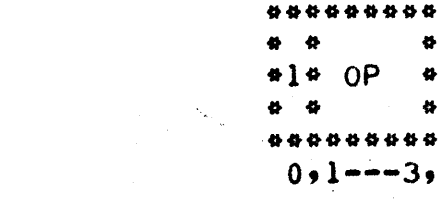
13. I/O PROGRAMMING

FROM 8-11-69 COPY OBSOLETED 9-10-69

TO DO I/O, THE PROGRAMMER MUST SET UP A PROGRAM IN CORE CONSISTING OF I/O COMMAND WORDS (IOCW'S) AND I/O DATA WORDS (IODW'S). EACH IOCW HAS BIT 0 = 1, AND CONTAINS AN OPCODE AND CERTAIN MODIFIER BITS. EACH IOCW MAY BE FOLLOWED BY ONE OR MORE IODW'S, WHICH SPECIFY THE AREA(S) IN MEMORY ON WHICH IT IS TO OPERATE. EACH IODW HAS BIT 0 = 0, AND CONTAINS A BYTE ADDRESS (22 BITS) AND A BYTE COUNT (9 BITS). WHEN AN SIO COMMAND IS ISSUED, THE I/O PROCESSOR EXECUTES THE FIRST IOCW OF THE I/O PROGRAM SUCCESSIVELY ON EACH OF THE FOLLOWING IODW'S, THEN EXECUTES THE NEXT IOCW ON EACH OF THE IODW'S FOLLOWING IT, ETC., UNTIL AN "END" IOCW IS ENCOUNTERED.

EACH I/O DEVICE HAS A DEVICE STATUS DOUBLEWORD (DSD) IN A DEDICATED POSITION IN LOWER CORE STORAGE. THE FIRST WORD CONTAINS A POINTER TO THE IOCW OR IODW CURRENTLY BEING EXECUTED BY THE DEVICE, IF ANY. THE SECOND WORD CONTAINS THE CURRENT BYTE ADDRESS AND BYTE COUNT FOR THE DATA TRANSFER BEING MADE BY THE DEVICE. AS THE DEVICE EXECUTES THE TRANSFER SPECIFIED BY AN IODW, THE BYTE ADDRESS IS INCREMENTED AND THE BYTE COUNT IS DECREMENTED UNTIL IT REACHES ZERO; THEN THE DEVICE PROCEEDS TO THE NEXT IODW.

IOCW:

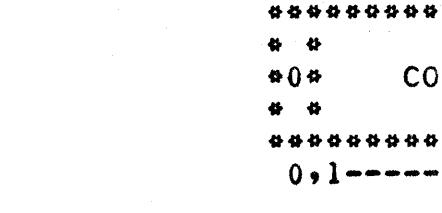


OP:
- 000 = END
- 001 = SENSE
- 010 = CONTROL
- 011 = JUMP
- 100 = WRITE
- 101 = READ
- 110 = INTERRUPT THE CPU WHOSE MODULE NO. IS IN BITS 12-15

FLAGS: NOT YET DEFINED

ADDRESS: ABSOLUTE CORE ADDRESS, USED IN SENSE, CONTROL, AND JUMP INSTRUCTIONS.

IODW:



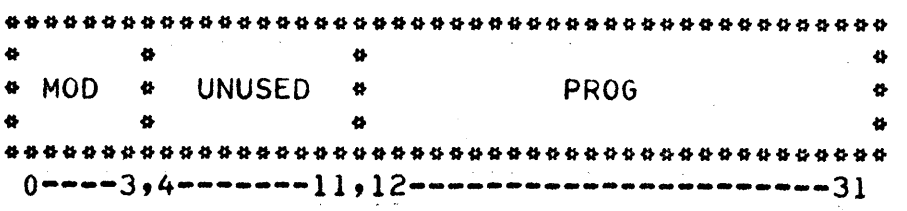
COUNT: NUMBER OF BYTES TO BE PROCESSED.

ADDRESS: ABSOLUTE BYTE ADDRESS OF FIRST BYTE TO BE PROCESSED.

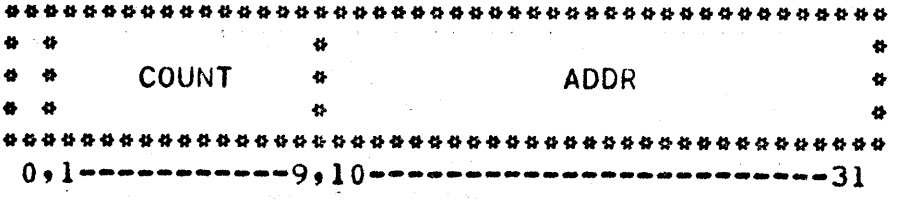
14. I/O DEVICE STATUS TABLE

THE I/O DEVICE STATUS TABLE OCCUPIES 512 WORDS OF CORE IN A DEDICATED LOCATION. IT HAS A DOUBLEWORD, CALLED A DEVICE STATUS DOUBLEWORD, FOR EVERY I/O DEVICE IN THE SYSTEM (UP TO 256 DEVICES.) THE FORMAT OF THE DEVICE STATUS DOUBLEWORD IS:

FIRST WORD



SECOND WORD



MOD: CPU MODULE NUMBER

PROG: ABSOLUTE CORE ADDRESS OF THE CURRENT IOCW OR IODW IN THE I/O PROGRAM NOW BEING EXECUTED, IF ANY.

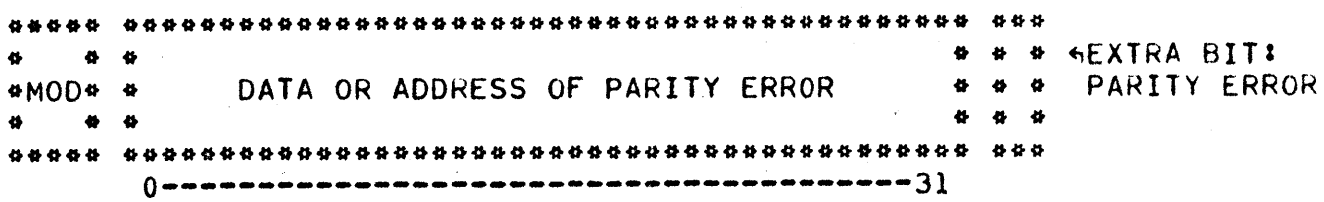
COUNT: NUMBER OF BYTES OR WORDS REMAINING TO BE PROCESSED UNDER THE CURRENT IODW. (IF WORD COUNT, THIS FIELD OCCUPIES BITS 1-11.)

ADDR: ABSOLUTE BYTE OR WORD ADDRESS OF THE BYTE OR WORD CURRENTLY BEING PROCESSED. (IF WORD ADDRESS, THIS FIELD OCCUPIES BITS 12-31.)

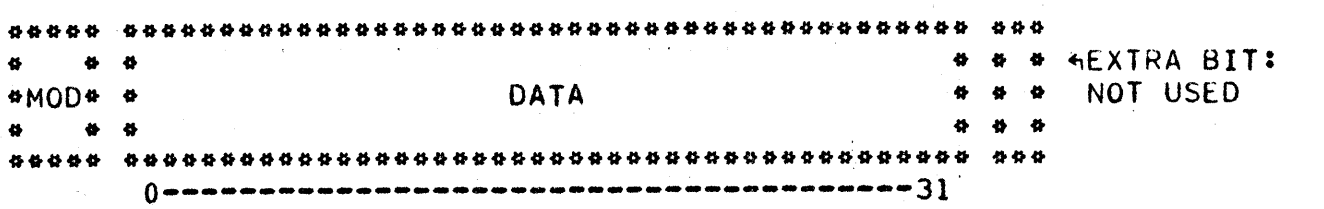
18. BUS FORMATS

CPU - MEMORY

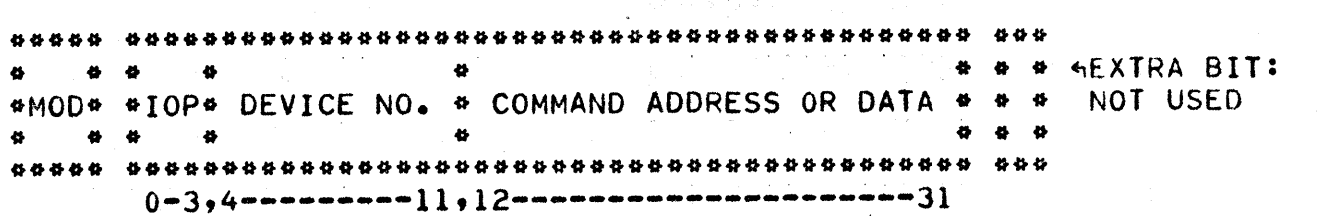
MEMORY - CPU



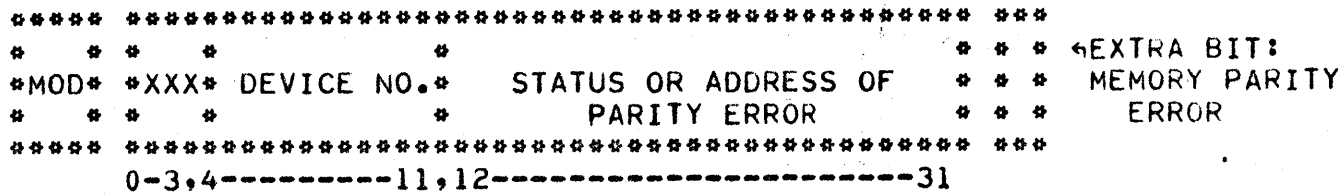
CPU - MEMORY



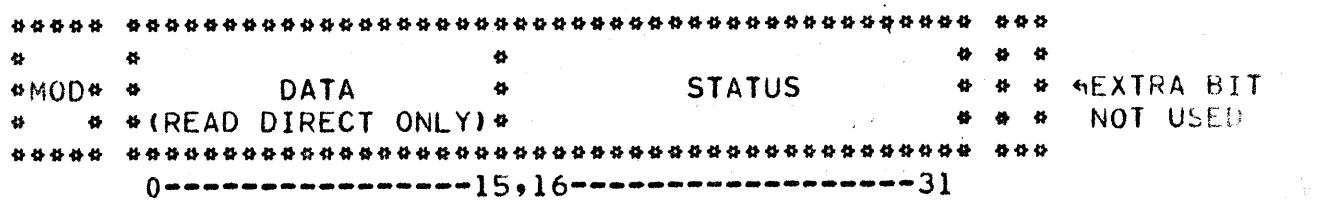
CPU - I/O



INTERRUPTING MODULE - CPU



I/O / CPU



SECTION II

MACHINE INSTRUCTIONS

IN THE FOLLOWING EXPLANATIONS,

R = THE REGISTER SPECIFIED IN THE R FIELD.

X = THE REGISTER SPECIFIED IN THE X FIELD.

M = THE RESULT OF NORMAL ADDRESS COMPUTATION. NOTE THAT M MAY BE A REGISTER, A MEMORY ADDRESS, OR IMMEDIATE DATA.

A = THE A FIELD OF THE INSTRUCTION.

0.

LOAD R,M (LOAD)
(R) ← (M)

REGISTER R IS LOADED WITH THE CONTENT OF M.
CC = UNAFFECTED.

1.

LLH R,M (LOAD LEFT HALF)
(R[16-31]) ← (M[0-15])
(R[0-15]) ← 0

BITS 0-15 OF THE CONTENT OF M IS LOADED INTO BITS 16-31 OF R.
BITS 0-15 OF R ARE SET TO ZERO.
CC = UNAFFECTED.

2.

LRH R,M (LOAD RIGHT HALF)
(R[16-31]) ← (M[16-31])
(R[0-15]) ← 0

BITS 16-31 OF THE CONTENT OF M IS LOADED INTO BITS 16-31 OF R.
BITS 0-15 OF R ARE SET TO ZERO.
CC = UNAFFECTED.

3.

STOR R,M (STORE)
(M) ← (R)

THE CONTENT OF R IS STORED IN M.
CC = PATTERN A.

104.

SIO R,M (START I/O)

THE LEFT 12 BITS OF REGISTER R CONTAIN THE I/O PROCESSOR NUMBER (4 BITS) AND THE DEVICE NUMBER (8 BITS). M IS THE ADDRESS OF AN I/O PROGRAM WHICH IS TO BE EXECUTED BY THE I/O PROCESSOR. AN I/O STATUS HALFWORD IS RETURNED TO THE RIGHT HALF OF REGISTER R BY THE I/O PROCESSOR.
THIS IS A PRIVILEGED INSTRUCTION.
CC = UNAFFECTED.

105.

TIO R (TEST I/O)

THE LEFT 12 BITS OF REGISTER R CONTAIN THE I/O PROCESSOR NUMBER (4 BITS) AND THE DEVICE NUMBER (8 BITS). A HALFWORD CONTAINING THE STATUS OF THE DEVICE IS RETURNED TO THE RIGHT HALF OF REGISTER R.
THIS IS A PRIVILEGED INSTRUCTION.
CC = UNAFFECTED.

106.

HIO R (HALT I/O)

THE LEFT 12 BITS OF REGISTER R CONTAIN THE I/O PROCESSOR NUMBER (4 BITS) AND THE DEVICE NUMBER (8 BITS). THIS DEVICE IS HALTED, AND ITS STATUS HALFWORD IS RETURNED TO THE RIGHT HALF OF REGISTER R.
THIS IS A PRIVILEGED INSTRUCTION.
CC = UNAFFECTED.

107.

RIO R,M (READ I/O)

THE LEFT 12 BITS OF REGISTER R CONTAIN THE I/O PROCESSOR NUMBER (4 BITS) AND THE DEVICE NUMBER (8 BITS). IF SUCCESSFUL, A DATA INBOUND HALFWORD IS READ DIRECTLY FROM THIS DEVICE INTO THE RIGHT HALF OF M. THE LEFT HALF OF M IS ZEROED. PROCESSING DOES NOT CONTINUE UNTIL THE READ IS COMPLETED. THE DEVICE STATUS HALFWORD IS RETURNED TO THE RIGHT HALF OF REGISTER R.
THIS IS A PRIVILEGED INSTRUCTION.
CC = UNAFFECTED.

108.

WIO R,M (WRITE I/O)

THE LEFT 12 BITS OF REGISTER R CONTAIN THE I/O PROCESSOR NUMBER (4 BITS) AND THE DEVICE NUMBER (8 BITS). A HALFWORD IS WRITTEN DIRECTLY FROM THE RIGHT HALFWORD OF M INTO THE DEVICE DATA OUTBOUND HALFWORD. PROCESSING DOES NOT CONTINUE UNTIL THE WRITE IS COMPLETED. THE DEVICE STATUS HALFWORD IS RETURNED TO THE RIGHT HALF OF REGISTER R.
THIS IS A PRIVILEGED INSTRUCTION.
CC = UNAFFECTED.

109.

CIO R,M (COMMAND I/O)

THE LEFT 12 BITS OF REGISTER R CONTAIN THE I/O PROCESSOR NUMBER (4 BITS) AND THE DEVICE NUMBER (8 BITS). A HALFWORD IS WRITTEN DIRECTLY FROM THE RIGHT HALFWORD OF M INTO THE DEVICE COMMAND REGISTER. PROCESSING CONTINUES IMMEDIATELY. THIS COMMAND CAN ALSO BE USED TO SEND A HALFWORD TO ANOTHER CPU OR TO ANY OTHER MODULE WHOSE NUMBER IS PLACED IN REGISTER R.
THIS IS A PRIVILEGED INSTRUCTION.
CC = UNAFFECTED.

DECIMAL INSTRUCTIONS

110.

ADDD R,M (ADD DECIMAL)
(R) ← (M) + (R)

IN ADDRESS COMPUTATION FOR THIS INSTRUCTION, THE INDEX REGISTER IS ASSUMED TO CONTAIN A NUMBER OF BYTES RATHER THAN A NUMBER OF WORDS. ADDRESS COMPUTATION RESULTS IN A BYTE ADDRESS M. THE LOW-ORDER FOUR BITS OF THE BYTE AT M ARE ADDED DECIMALLY TO THE LOW-ORDER FOUR BITS OF REGISTER R. THE CARRY-IN IS TAKEN AS 0 OR 1 ACCORDING TO WHETHER THE CONDITION CODE IS 0 OR NONZERO. THE CARRY-OUT IS LOADED INTO THE CONDITION CODE. INVALID DIGITS ARE TREATED AS ZEROS, AND CAUSE A TRAP.
CC = 0 IF NO CARRY-OUT
CC = 1 IF A CARRY-OUT OCCURS.

111.

SUBD R,M (SUBTRACT DECIMAL)
(R) ← (R) - (M)

IN ADDRESS COMPUTATION FOR THIS INSTRUCTION, THE INDEX REGISTER IS ASSUMED TO CONTAIN A NUMBER OF BYTES RATHER THAN A NUMBER OF WORDS. ADDRESS COMPUTATION RESULTS IN A BYTE ADDRESS M. THE LOW-ORDER FOUR BITS OF THE BYTE AT M ARE SUBTRACTED DECIMALLY FROM THE LOW-ORDER FOUR BITS OF REGISTER R. IF THE CONDITION CODE IS 1, A BORROW-IN IS ASSUMED (THE CONTENT OF R IS DECREMENTED BY 1 BEFORE SUBTRACTING.) IF THE SUBTRACTION RESULTS IN A BORROW-OUT (BECAUSE (R) < (M)), THE CONDITION CODE IS SET TO 1; ELSE THE CONDITION CODE IS SET TO 0. INVALID DIGITS ARE TREATED AS ZEROS, AND CAUSE A TRAP.
CC = 1 IF BORROW-OUT.
CC = 0 IF NO BORROW-OUT.