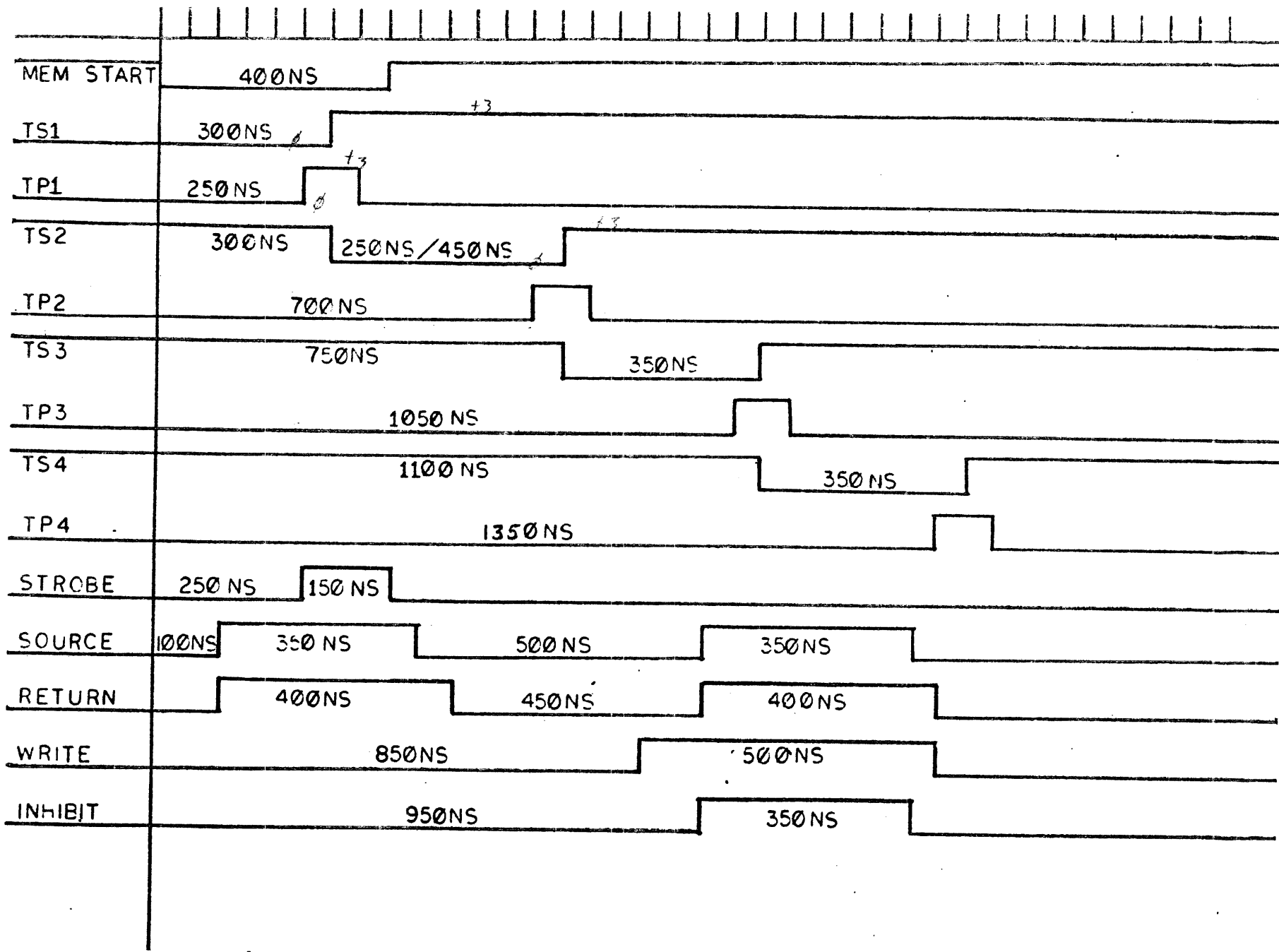
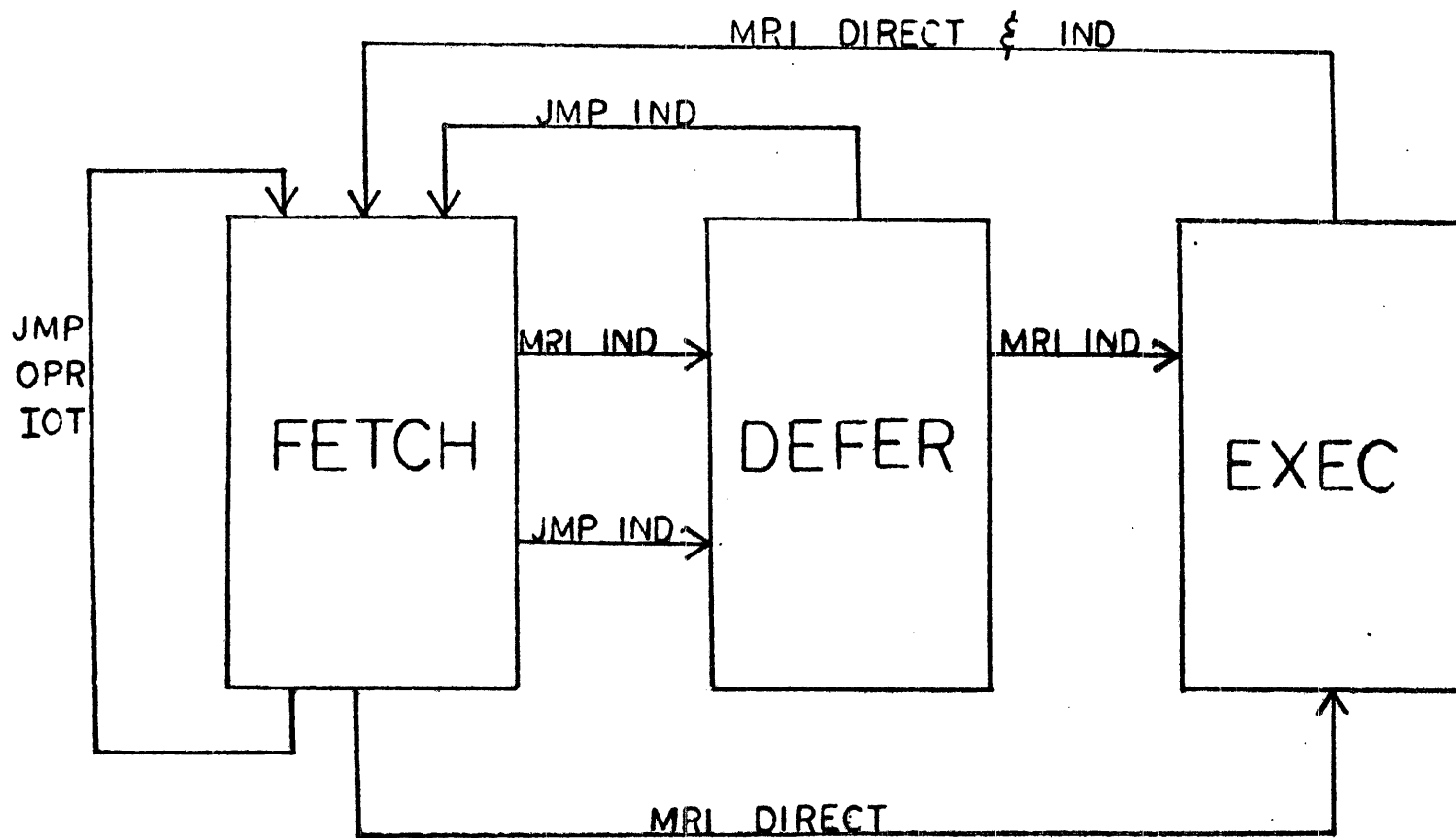


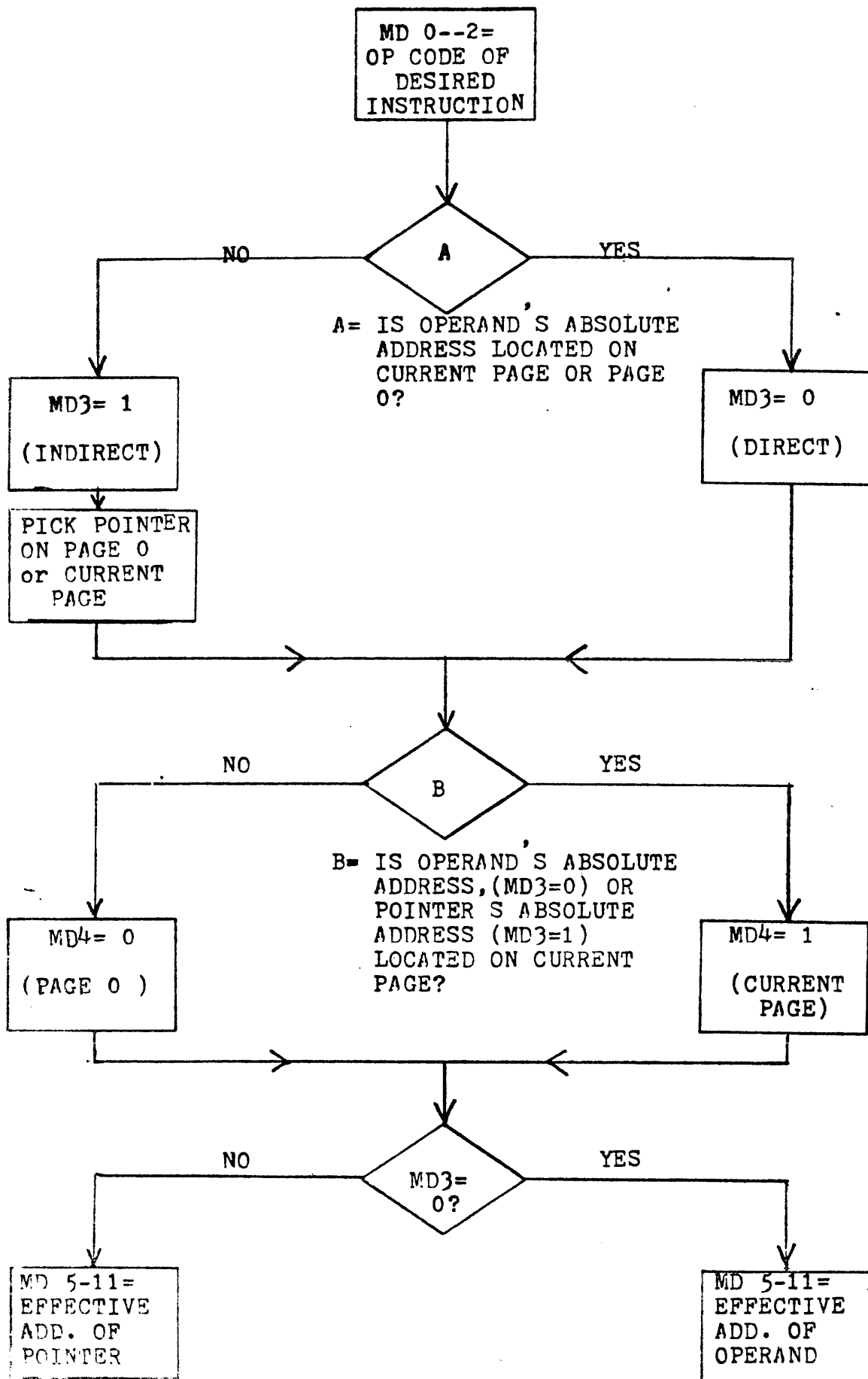
IS
UNABID

ALL TP
AND 100NS
TO 10MS

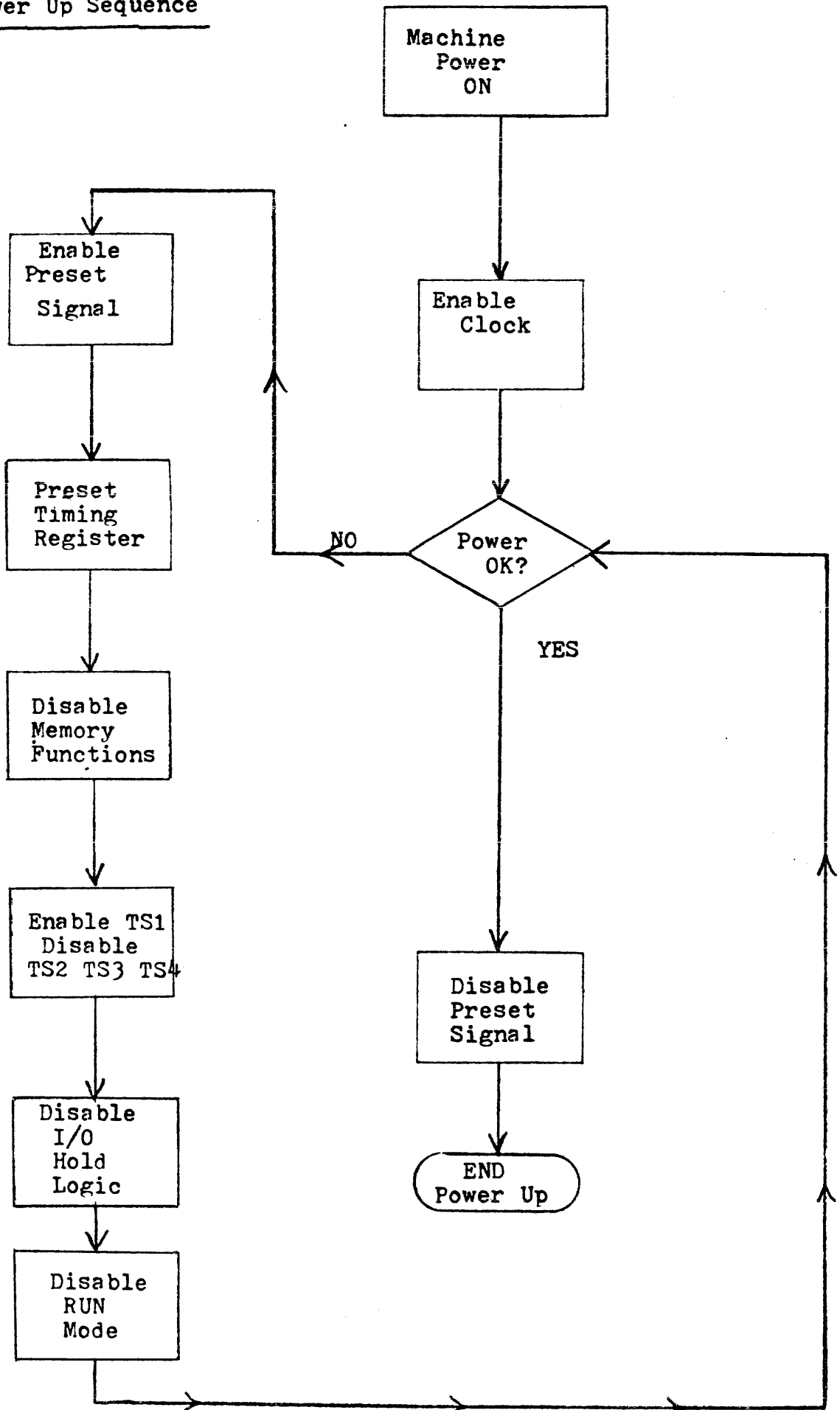


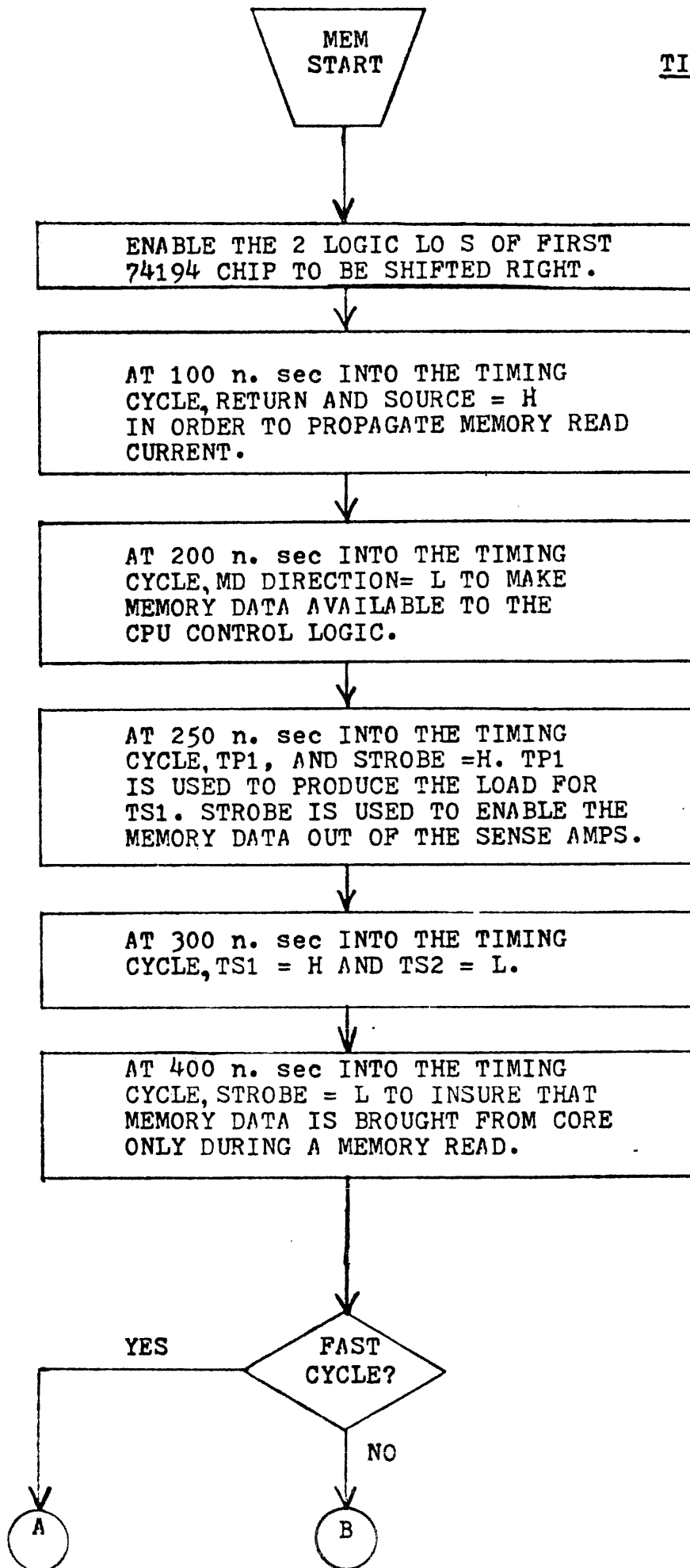


PDP 8/e
MRI INSTRUCTION CODING
FLOW DIAGRAM



Power Up Sequence





A

AT 450 n. sec SOURCE = L,
AT 500 n. sec RETURN = L
IN ORDER TO DISABLE MEMORY
READ CURRENT. ALSO AT 500
n. sec TP2 = H TO PRODUCE
A LOAD FOR TS2.

AT 550 n. sec INTO THE
TIMING CYCLE, TS2 = H, AND
TS3 = L.

AT 650 n. sec INTO THE
TIMING CYCLE, WRITE = H IN
PREPARATION FOR A MEMORY
WRITE.

AT 750 n. sec INTO THE
TIMING CYCLE, SOURCE,
RETURN, AND INHIBIT = H
TO PROPAGATE MEMORY WRITE
CURRENT AND INHIBIT CURRENT.

AT 850 n. sec INTO THE
TIMING CYCLE, TP3 = H TO
PRODUCE A LOAD FOR TS3.

AT 900 n. sec INTO THE
TIMING CYCLE, TS3 = H AND
TS4 = L.

AT 1100 n. sec INTO THE
TIMING CYCLE, SOURCE, AND
INHIBIT = L. AT 1150 n. sec
INTO THE TIMING CYCLE,
RETURN, AND WRITE = L. THIS
IS DONE TO DISABLE MEMORY
WRITE AND INHIBIT CURRENTS.

@ 1200 n.
sec TP4=H

B

AT 450 n. sec INTO THE TIMING
CYCLE, SOURCE = L AND AT 500
n. sec RETURN = L TO DISABLE
MEMORY READ CURRENT.

AT 700 n. sec INTO THE TIMING
CYCLE, TP2 = H TO PRODUCE A
LOAD FOR TS2.

AT 750 n. sec INTO THE TIMING
CYCLE, TS2 = H AND TS3 = L.

AT 850 n. sec INTO THE CYCLE
WRITE = H IN PREPARATION FOR A
MEMORY WRITE.

AT 950 n. sec INTO THE TIMING
CYCLE, SOURCE, RETURN, AND
INHIBIT = H TO PROPAGATE
MEMORY WRITE CURRENT AND
INHIBIT CURRENT.

AT 1050 n. sec INTO THE TIMING
CYCLE, TP3 = H TO PRODUCE A
LOAD FOR TS3.

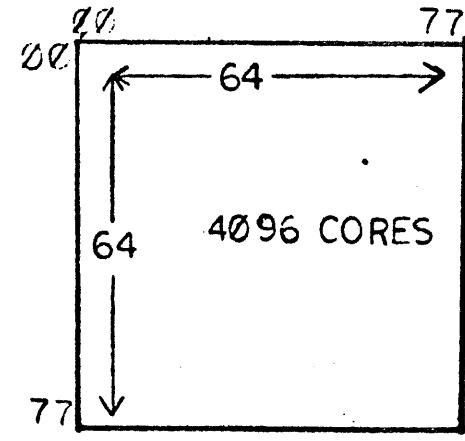
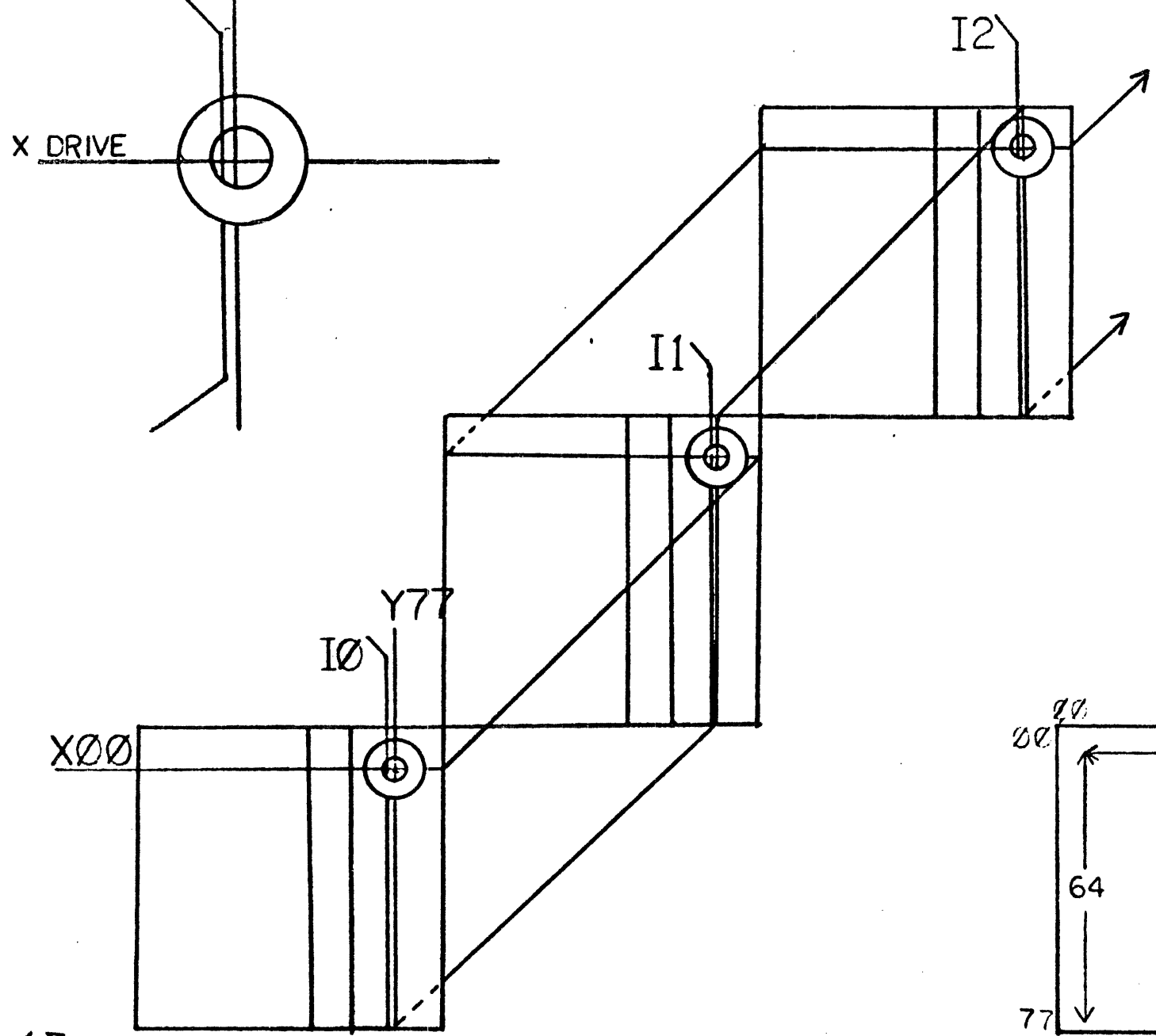
AT 1100 n. sec INTO THE TIMING
CYCLE, TS3 = H AND TS4 = L.

AT 1300 n. sec INTO THE TIMING
CYCLE, SOURCE, AND INHIBIT = L.
AT 1350 n. sec INTO THE TIMING
CYCLE, RETURN, AND WRITE = L.
THIS IS DONE TO DISABLE MEMORY
WRITE AND INHIBIT CURRENTS.

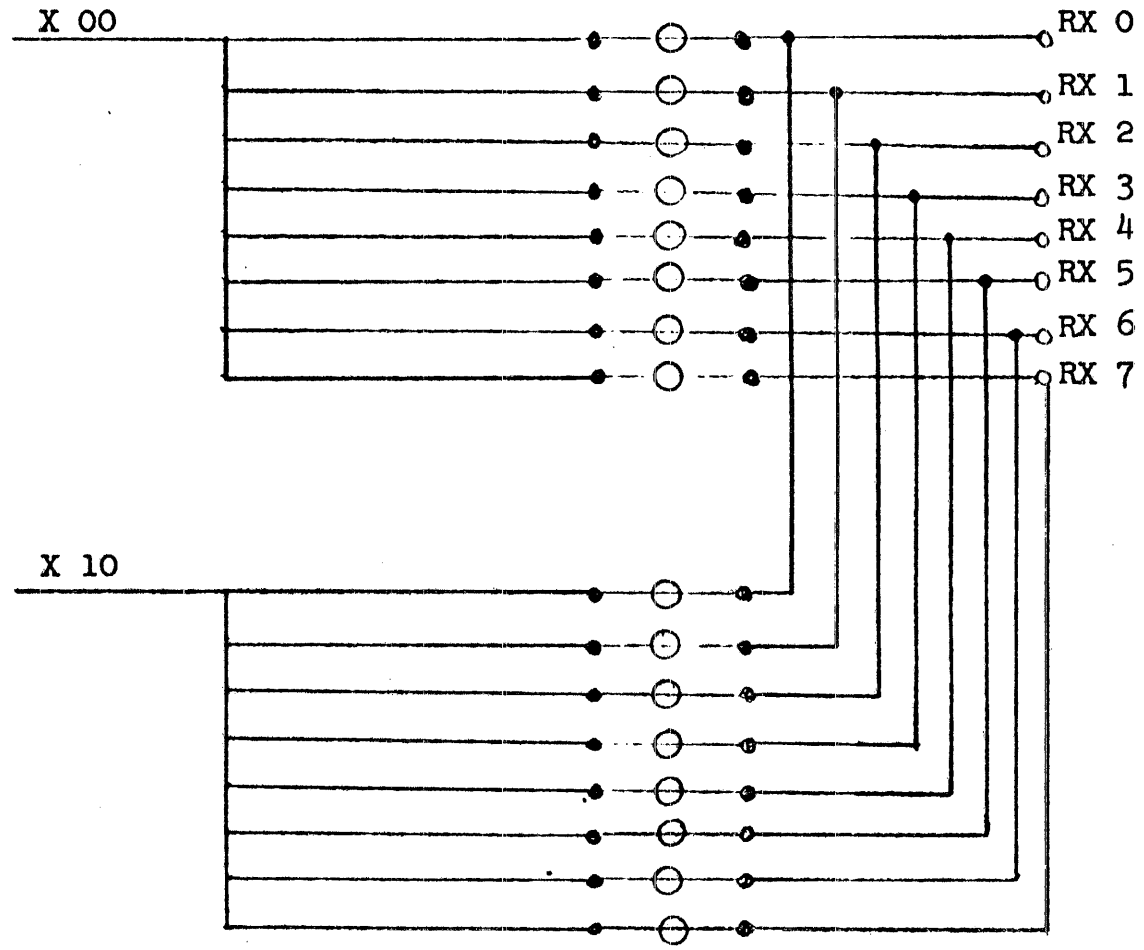
@ 1400 n.
sec TP4=H

100 BIT/SEC DE Y DRIVE

X DRIVE



EXAMPLE OF X READ LINE LAYOUT



DESIGNATES
768 CORES

X READ LINE LAYOUT

MEMORY ADDRESS SELECTION

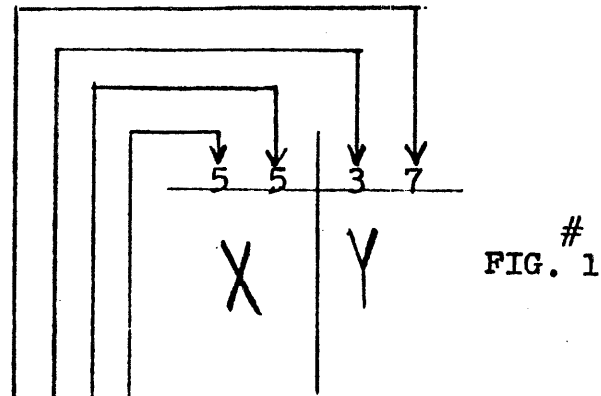


FIG. 1 #

THIS OCTAL DIGIT, WHICH IS COMPRISED OF MA 0---2, WILL SELECT A READ OR WRITE LINE IN THE UPPER LEFT HAND QUADRANT.

THIS OCTAL DIGIT, WHICH IS COMPRISED OF MA 3---5, WILL SELECT A READ OR WRITE LINE IN THE UPPER RIGHT HAND QUADRANT.

THIS OCTAL DIGIT, WHICH IS COMPRISED OF MA 6---8, WILL SELECT A READ OR WRITE LINE IN THE LOWER LEFT HAND QUADRANT.

THIS OCTAL DIGIT, WHICH IS COMPRISED OF MA 9---11, WILL SELECT A READ OR WRITE LINE IN THE LOWER RIGHT HAND QUADRANT.

FIG. 2 #
OUTPUT LINES

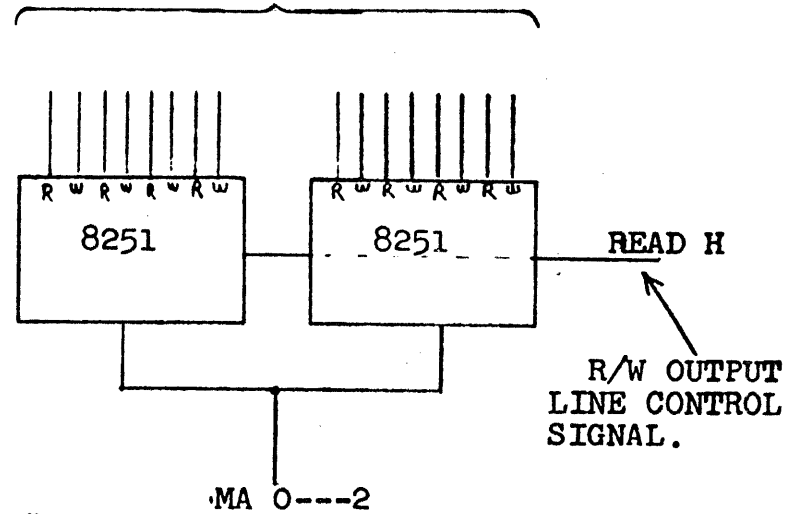
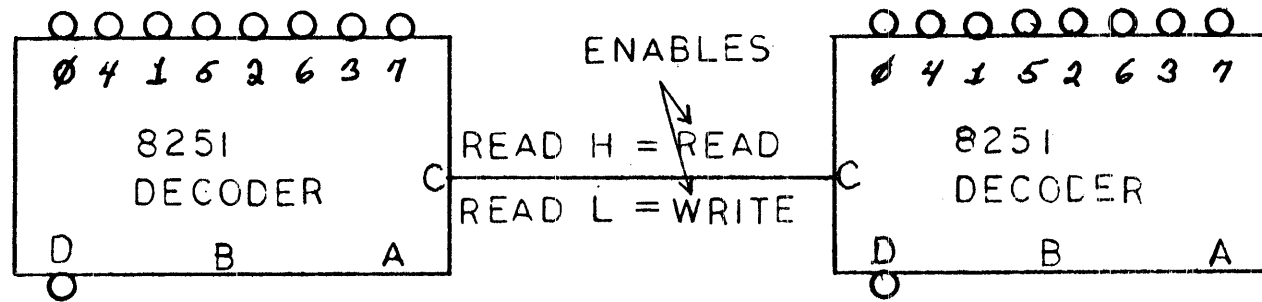


FIG. 2. ILLUSTRATES THE UPPER LEFT HAND QUADRANT OF PRINT 3. EACH 8251 HAS 8 OUTPUT LINES; 4 FOR READ AND 4 FOR WRITE FUNCTIONS. ONE READ OR WRITE LINE(DEPENDING ON THE FUNCTION) MUST BE SELECTED IN ORDER TO SELECT THE DESIRED ADDRESS. THIS IS DONE BY THE COMBINATION OF THE MA BITS (THE OUTPUT LINES WIRED AS SHOWN IN HANDOUT). THE ABOVE INFORMATION IS APPLICABLE TO THE OTHER 3 QUADRANTS OF PRINT 3.

MEMORY ADDRESS DECODER

WHEN READING, USE OUT ACTIVE PINS 4,5,6,7
 WHEN WRITING, USE OUT ACTIVE PINS 0,1,2,3

*inside pins
 READ
 4 5 6 7
 WRITE
 0 1 2 3*

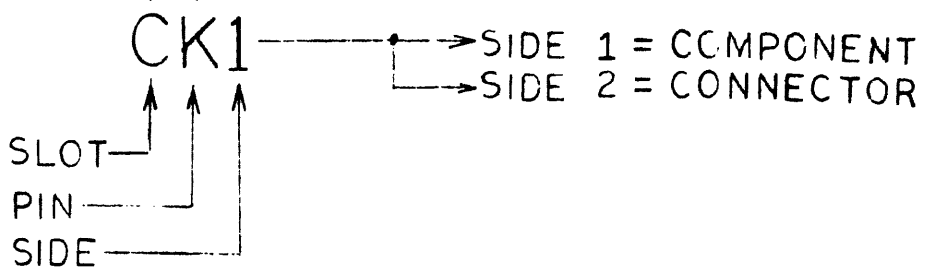
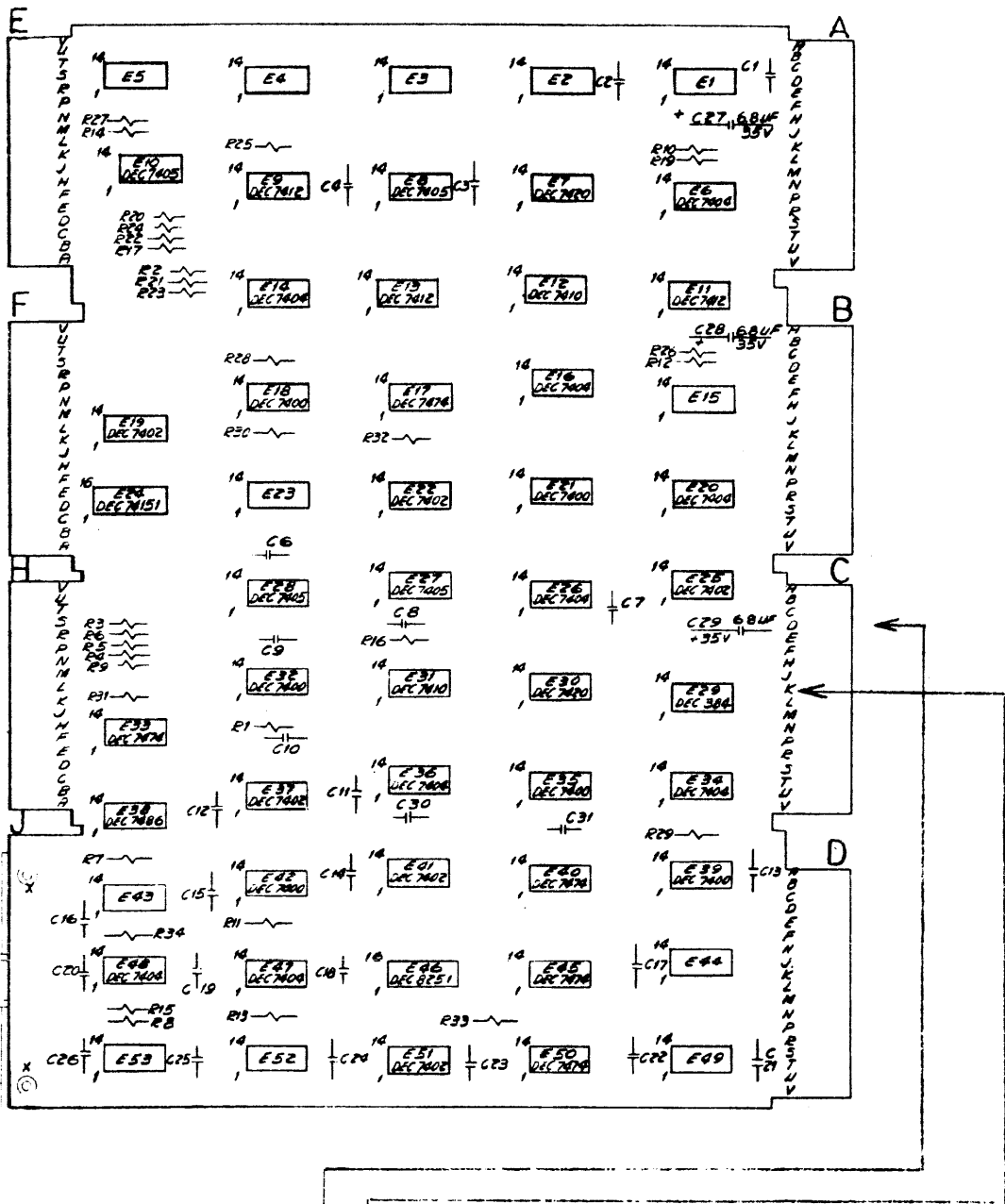


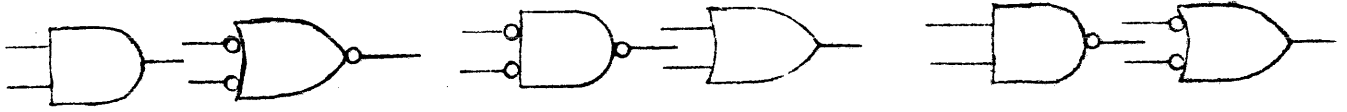
| MA | | | MA INPUT | D B A | CUT ACTIVE PINS | | |
|----|----|----|-------------|-------|-----------------------|---|---|
| 0 | 1 | 2 | | | 4 | 0 | |
| 3 | 4 | 5 | L | L | L | 5 | 1 |
| 6 | 7 | 8 | L | H | L | 6 | 2 |
| 9 | 10 | 11 | L | H | H | 7 | 3 |
| 0 | 0 | 0 | L | L | L | 4 | 0 |
| 0 | 0 | 1 | L | L | H | 5 | 1 |
| 0 | 1 | 0 | L | H | L | 6 | 2 |
| 0 | 1 | 1 | L | H | H | 7 | 3 |

READ WRITE

| MA | | | MA INPUT | D B A | OUT ACTIV. PINS | | |
|----|----|----|-------------|-------|-----------------------|---|---|
| 0 | 1 | 2 | | | 4 | 0 | |
| 3 | 4 | 5 | L | L | H | 5 | 1 |
| 6 | 7 | 8 | L | H | L | 6 | 2 |
| 9 | 10 | 11 | L | H | H | 7 | 3 |
| 1 | 0 | 0 | L | L | L | 4 | 0 |
| 1 | 0 | 1 | L | L | H | 5 | 1 |
| 1 | 1 | 0 | L | H | L | 6 | 2 |
| 1 | 1 | 1 | L | H | H | 7 | 3 |

READ WRITE

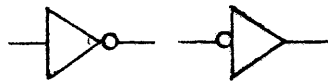
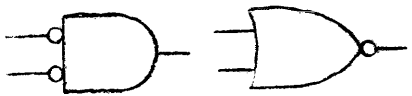




| A | B | C |
|---|---|---|
| H | H | H |
| H | L | L |
| L | H | L |
| L | L | L |

| A | B | C |
|---|---|---|
| H | H | H |
| H | L | H |
| L | H | H |
| L | L | L |

| A | B | C |
|---|---|---|
| H | H | L |
| H | L | H |
| L | H | H |
| L | L | H |



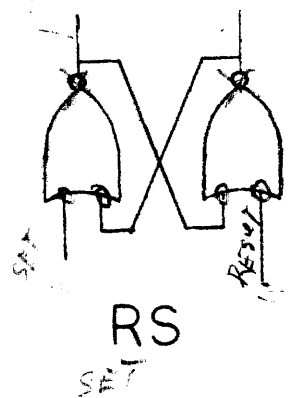
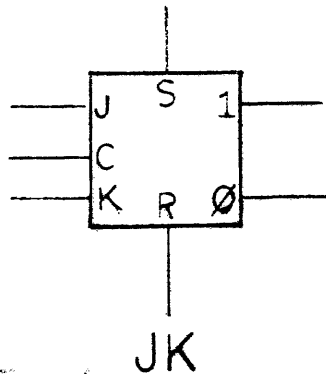
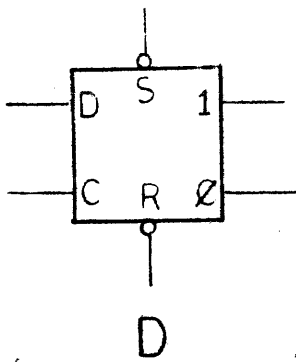
| A | B | C |
|---|---|---|
| H | H | L |
| H | L | L |
| L | H | L |
| L | L | H |

| | | |
|---|---|---|
| H | L | H |
| L | H | L |

| A | B | C |
|---|---|---|
| H | H | L |
| H | L | H |
| L | H | H |
| L | L | L |

Direct Set

*closes to 1 side
1 is H when direct set*



A High on S or R is a direct set

" " " D is 'H' on 1 side Low on 0 side

To Clear: A Low on D L on 1 side H on 0 side

ADDER LOGIC TRUTH TABLES

| ENO | EN1 | EN2 | RESULT |
|-----|-----|-----|--------|
| L | L | L | PC |
| L | L | H | MD |
| L | H | L | MQ |
| L | H | H | MA |
| H | X | X | ZERO |

| DATA T | DATA F | RESULT |
|--------|--------|------------|
| L | L | COMPLEMENT |
| L | H | TRUE |
| H | L | ZERO |

| PAGE | RIGHT | LEFT | TWICE | RESULT |
|------|-------|------|-------|--------------|
| L | L | L | L | CURRENT PAGE |
| X | L | L | H | AND |
| X | L | H | L | RIGHT 2 |
| X | L | H | H | RIGHT 1 |
| X | H | L | L | LEFT 2 |
| X | H | L | H | LEFT 1 |
| X | H | H | L | SWAP |
| X | H | H | H | NO SHIFT |
| H | L | L | L | PAGE ZERO |

I/O SIGNALS

| CO | C1 | C2 | RESULT |
|----|----|----|-------------------|
| L | L | L | DATA → PC |
| L | L | H | DATA → AC |
| L | H | L | PC + DATA → PC |
| L | H | H | AC → DATA, CLR AC |
| H | L | L | DATA → PC |
| H | L | H | AC/DATA ored → AC |
| H | H | L | PC + DATA → PC |
| H | H | H | AC/DATA ored → AC |

ADDER CONDITIONS

| LEFT | RIGHT | CAR IN | SUM | CAR OUT |
|------|-------|--------|-----|---------|
| H | H | H | H | H |
| H | L | H | L | H |
| L | L | H | H | L |
| H | H | L | L | H |
| H | L | L | H | *L |
| L | L | L | L | L |

| RIGHT | LEFT | TWICE | PAGE Z | USE |
|-------|------|-------|--------|--------------|
| L | L | L | L | CURRENT PAGE |
| L | L | H | X | AND |
| L | H | L | X | RTR |
| L | H | H | X | RAR |
| H | L | L | X | RTL |
| H | L | H | X | RAL |
| H | H | L | X | BYTE SWAP |
| H | H | H | X | NO SHIFT |
| L | L | L | H | PAGE ZERO |

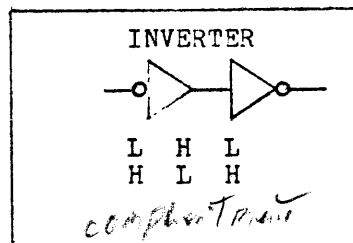
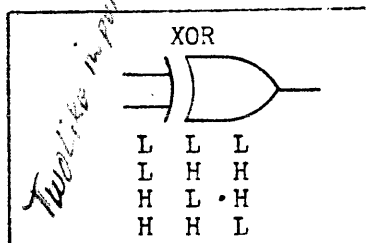
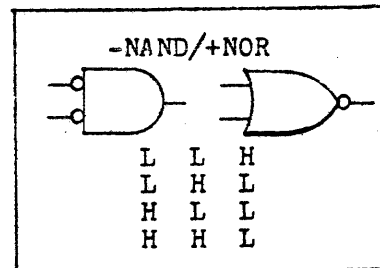
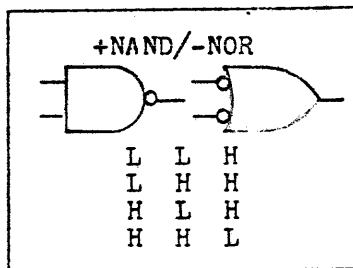
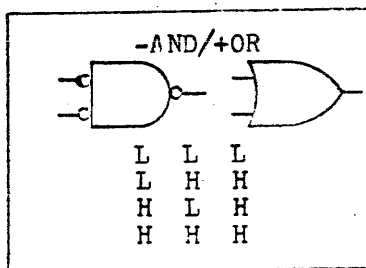
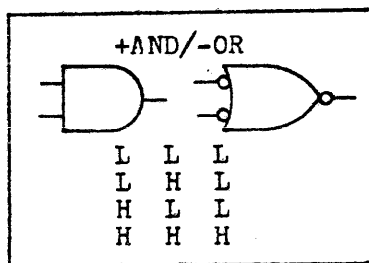
| LEFT | RIGHT | CI | SUM | CO |
|------|-------|----|-----|----|
| H | H | H | H | H |
| H | L | H | L | H |
| L | L | H | H | L |
| H | H | L | L | H |
| H | L | L | H | L |
| L | L | L | H | L |

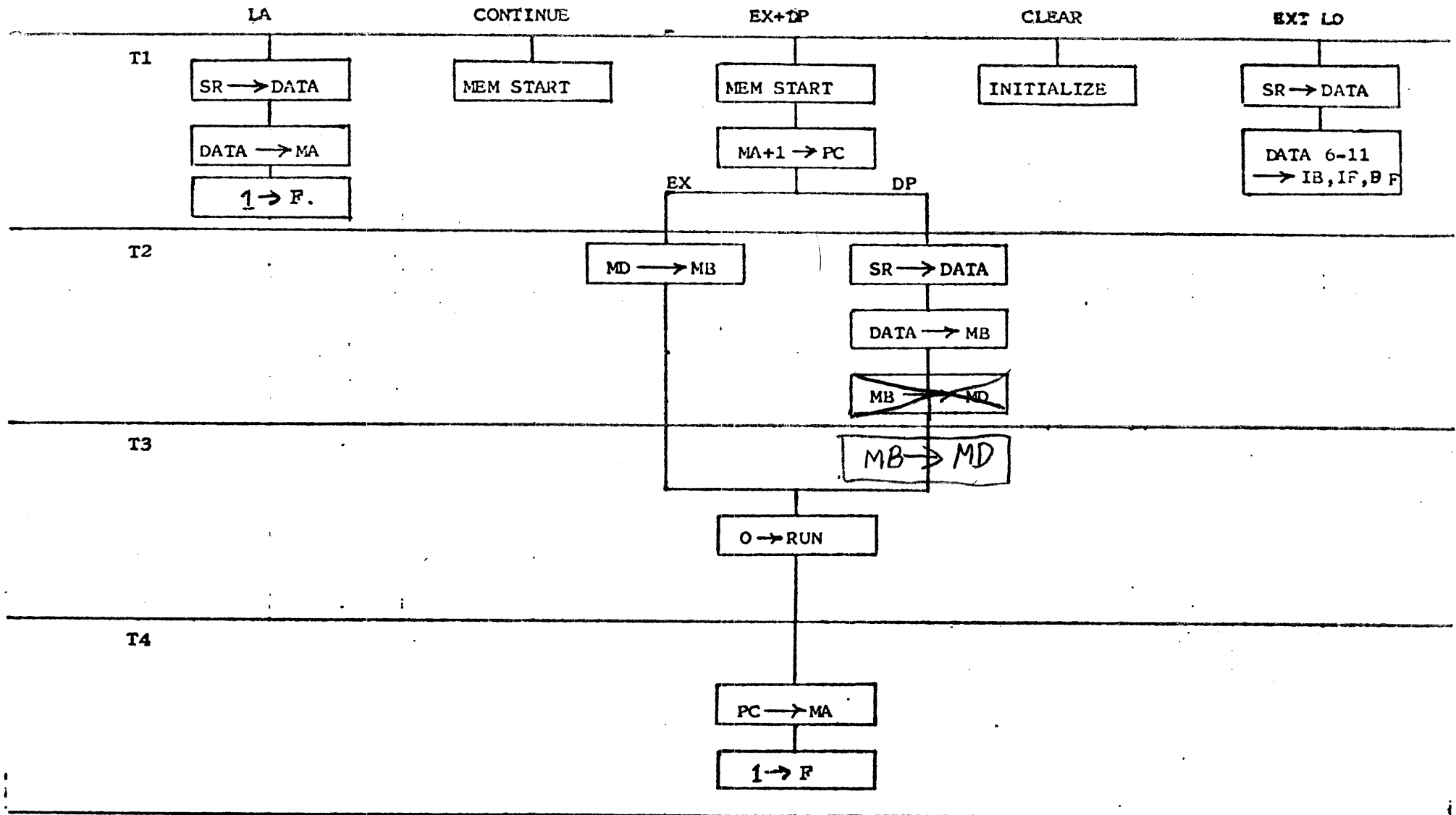
| ENO | EN1 | EN2 | ADDER IN |
|-----|-----|-----|----------|
| L | L | L | PC |
| L | L | H | MD |
| L | H | L | MQ |
| L | H | H | MA |
| H | X | X | ZERO |

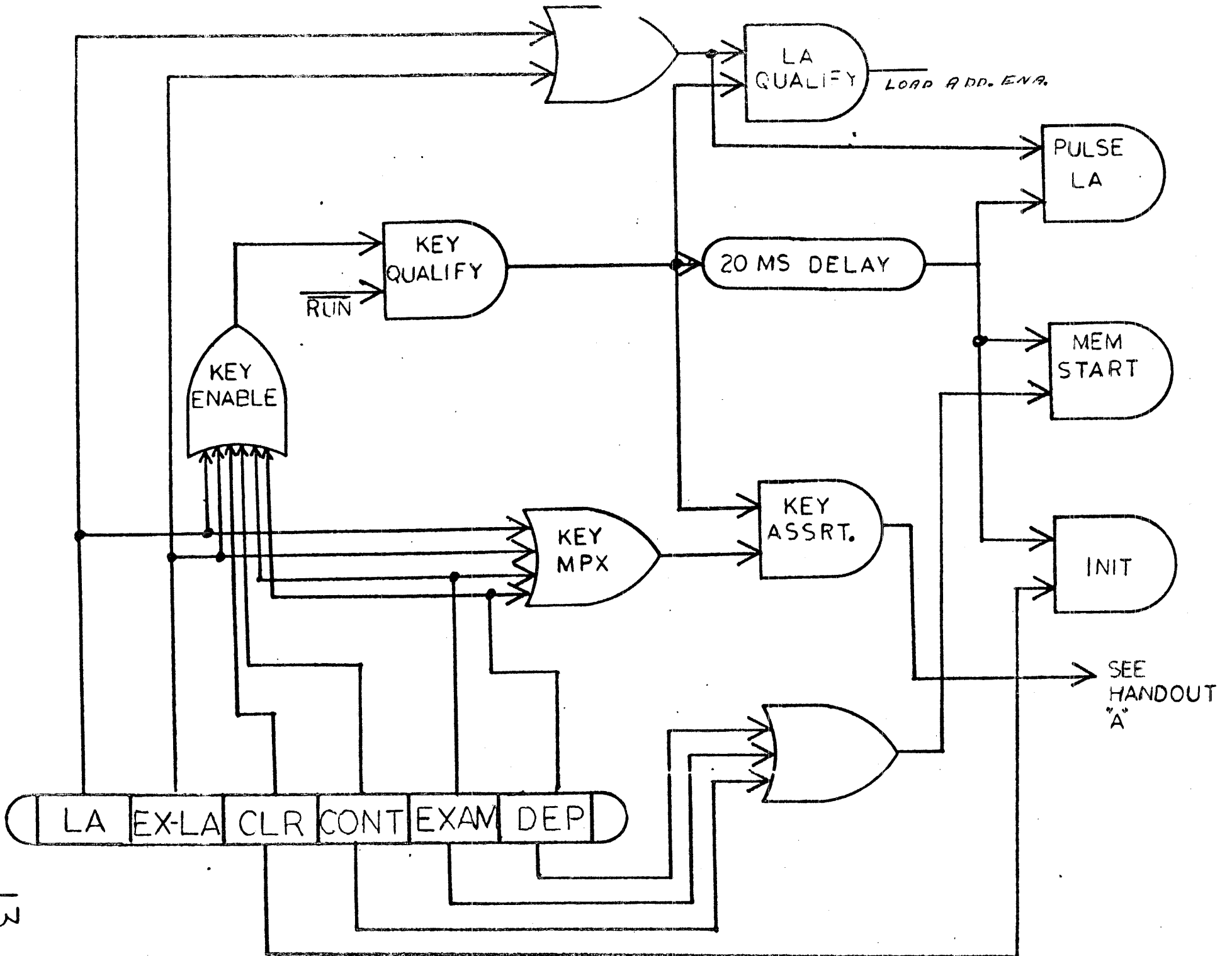
| DATA T | DATA F | ADDER IN |
|--------|--------|----------|
| L | L | DATA BUS |
| L | H | DATA BUS |
| H | L | ZERO |

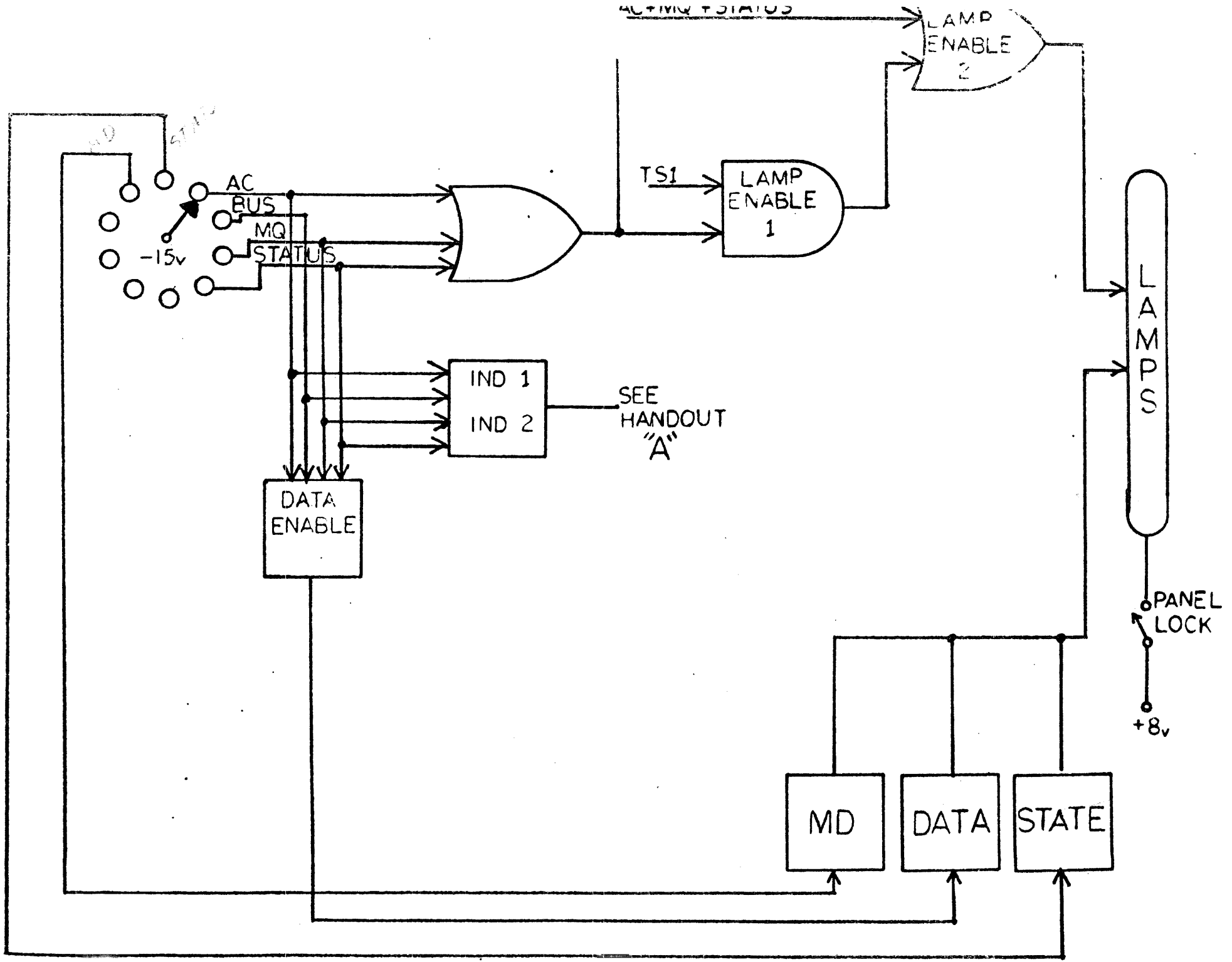
| IND1 | IND2 | RESULT |
|------|------|----------|
| L | L | AC → BUS |
| L | H | BUS |
| H | L | MQ → BUS |
| H | H | STATUS |

| CO | C1 | C2 | RESULT |
|----|----|----|-------------------|
| L | L | L | DATA → PC |
| L | L | H | DATA → AC |
| L | H | L | PC + DATA → PC |
| L | H | H | AC → DATA, 0 → AC |
| H | L | L | DATA → PC |
| H | L | H | AC ored DATA → AC |
| H | H | L | PC + DATA → PC |
| H | H | H | AC ored DATA → AC |









KEY FUNCTION SIGNALS

HANDOUT A

| IND 1 | IND 2. | Result |
|-------|--------|--------------|
| L | L | AC → BUS |
| L | H | BUS → BUS |
| H | L | MQ → BUS |
| H | H | STATUS → BUS |

KEY LA

| | | | |
|-----|-----------------|---|------------------------------|
| (A) | KEY CONTROL | H | Qualification for MA LOAD |
| (B) | MS DISABLE | L | Disables MAJOR STATES and IR |
| (C) | MD DIR. IN. | L | NO use for KEY LA |
| (D) | BK DATA CONTROL | L | NO use for KEY LA |
| (E) | SR----BUS | H | Enables SR to the DATA BUS |

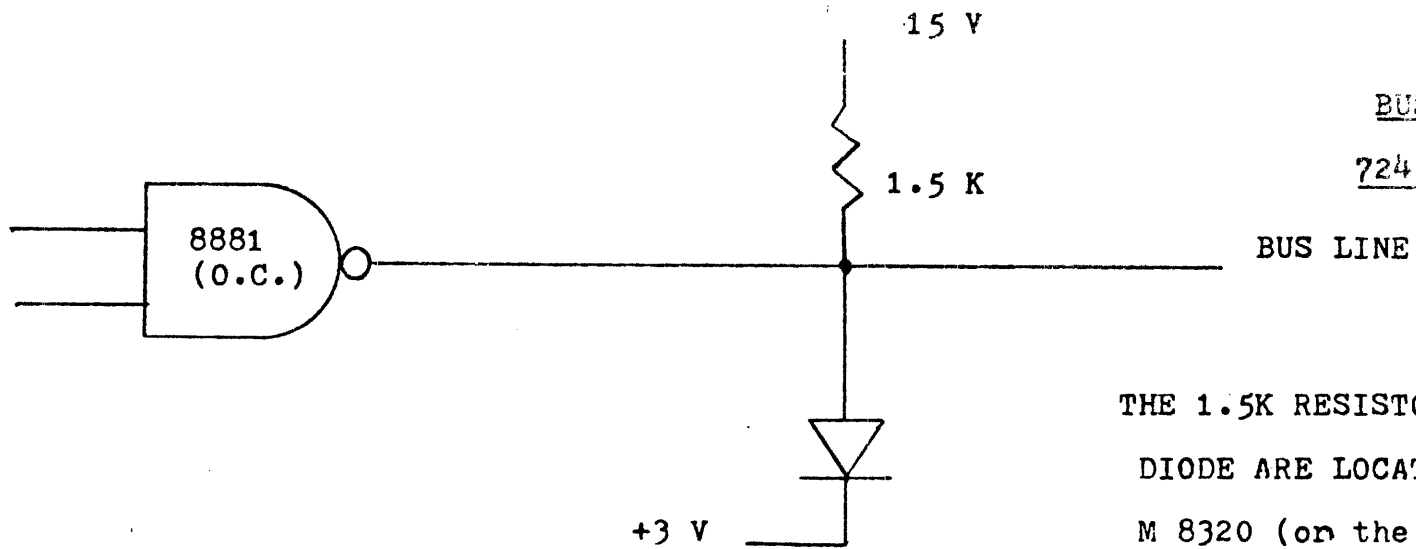
KEY EXAM

| | | | |
|-----|-----------------|---|------------------------------|
| (A) | KEY CONTROL | L | Allows one timing cycle only |
| (B) | MS DISABLE | L | Disables MAJOR STATES and IR |
| (C) | MD DIR, IN. | L | Enables MEM → MD LINES |
| (D) | BK DATA CONTROL | L | Enables MD → MB |
| (E) | SR----BUS | L | Disables SR → BUS |

KEY DEP

| | | | |
|-----|-----------------|---|------------------------------|
| (A) | KEY CONTROL | L | Allows one timing cycle only |
| (B) | MS DISABLE | L | Disables MAJOR STATES and IR |
| (C) | MD DIR. IN. | H | Disables MEM → MD LINES |
| (D) | BK DATA CONTROL | H | Disables MD → MB |
| (E) | SR----BUS | H | Enables SR → BUS |

BUS LOGIC CONCEPT
AND
724 POWER SUPPLY REQUIREMENTS

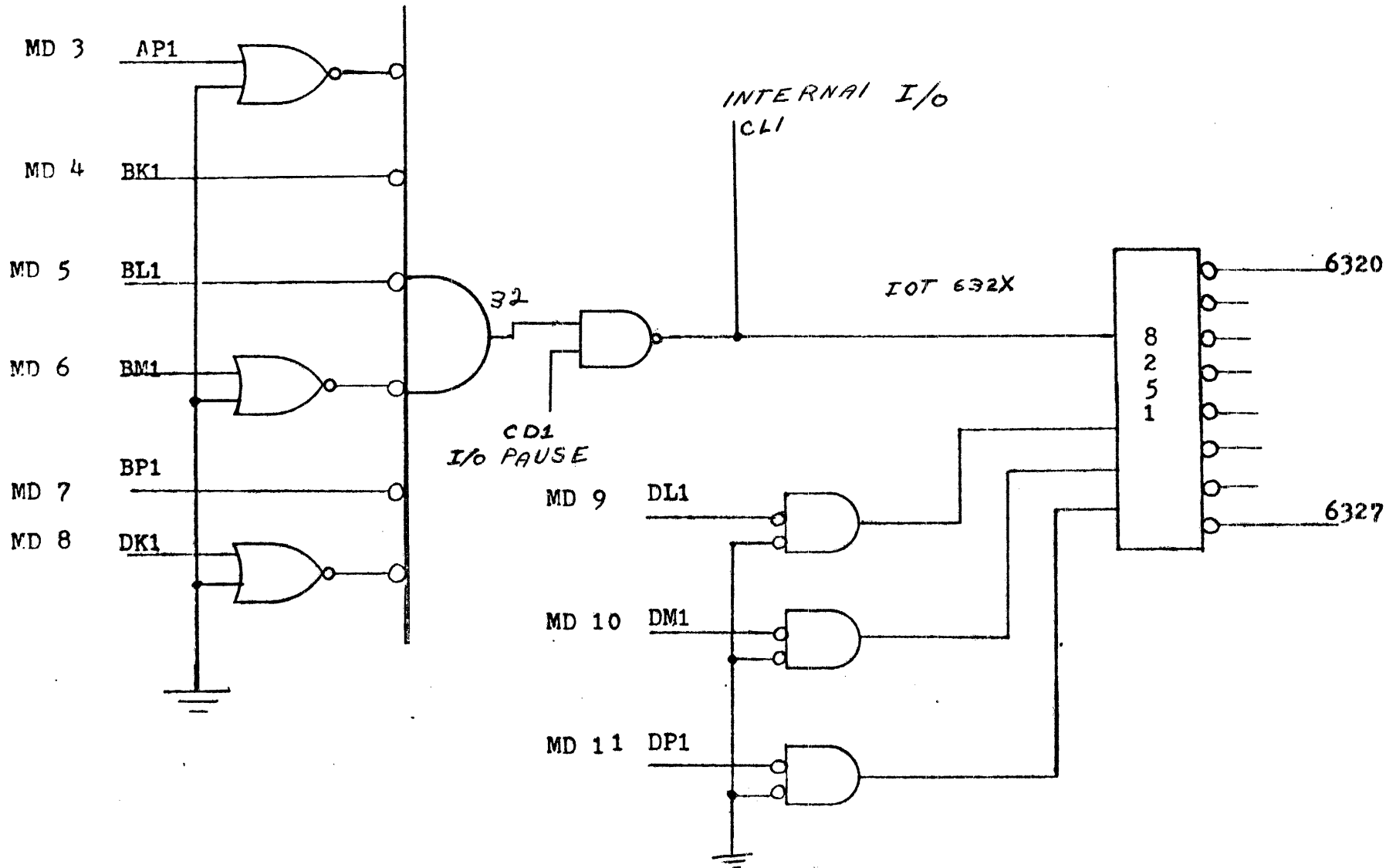


THE 1.5K RESISTOR AND THE DIODE ARE LOCATED ON THE M 8320 (on the M 832) BUS LOADS CARD.

H724 POWER SUPPLY

| OUTPUT VOLTAGE | +15 V | +5 V | -15 V | +8 V |
|----------------|-------|------|-------|--------|
| CURRENT OUTPUT | 1 A | 20 A | .8 A | 2 A |
| CURRENT USED | .8 A | 6 A | 4.5 A | 1.25 A |
| UNUSED CURRENT | .2 A | 14 A | 3.5 A | |

INTERN.
DEVICE CODE SELECTION
(632X)



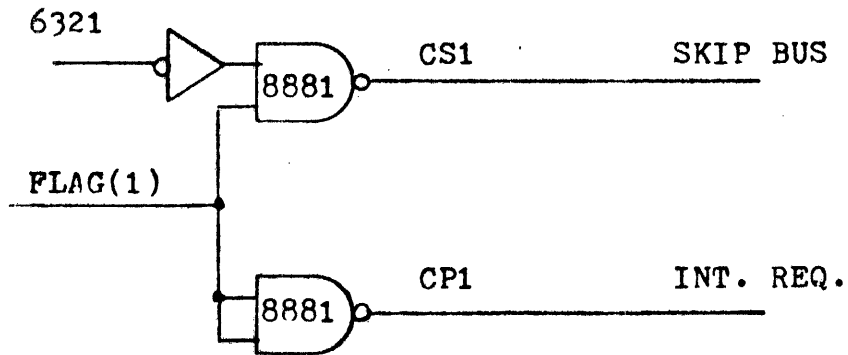
INTERNAL I/O

EXAMPLE INSTRUCTION SET

| <u>INSTRUCTION</u> | <u>FUNCTION</u> |
|--------------------|-------------------------------------------------------------------------------------------------------------|
| 6321----- | CHECK FLAG AND SKIP IF =(1) |
| 6322----- | SKIP ONCE IF FLAG "A" = 1, SKIP ONCE IF FLAG "B" = 1, SKIP TWICE IF FLAG "A" AND FLAG "B" ARE = 1. |
| 6324----- | TRANSFER AC TO DEVICE |
| 6325----- | TRANSFER AC TO DEVICE, 0---AC |
| 6326----- | TRANSFER DATA FROM DEVICE TO AC. |
| 6327----- | TRANSFER DATA TO THE PC. |

GATING FOR SKIP AND INT. REQ.

FIG. 1

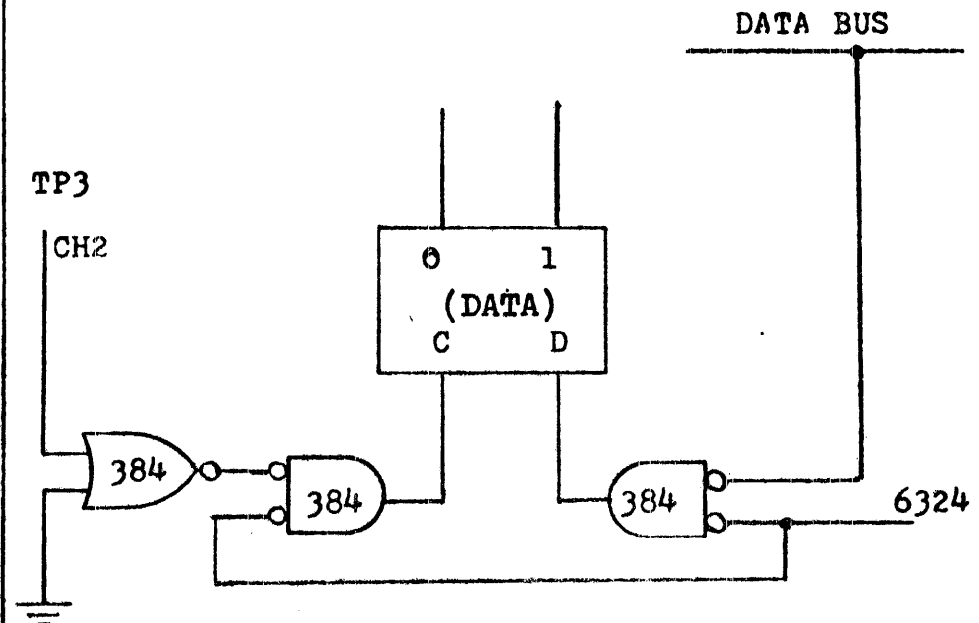


THE PURPOSE OF THIS GATING IS TO ALLOW SENSING OF SKIP CONDITIONS AND ACKNOWLEDGING INTERRUPT REQUESTS FROM THE DEVICE.

I/O OUTPUT TRANSFER GATING

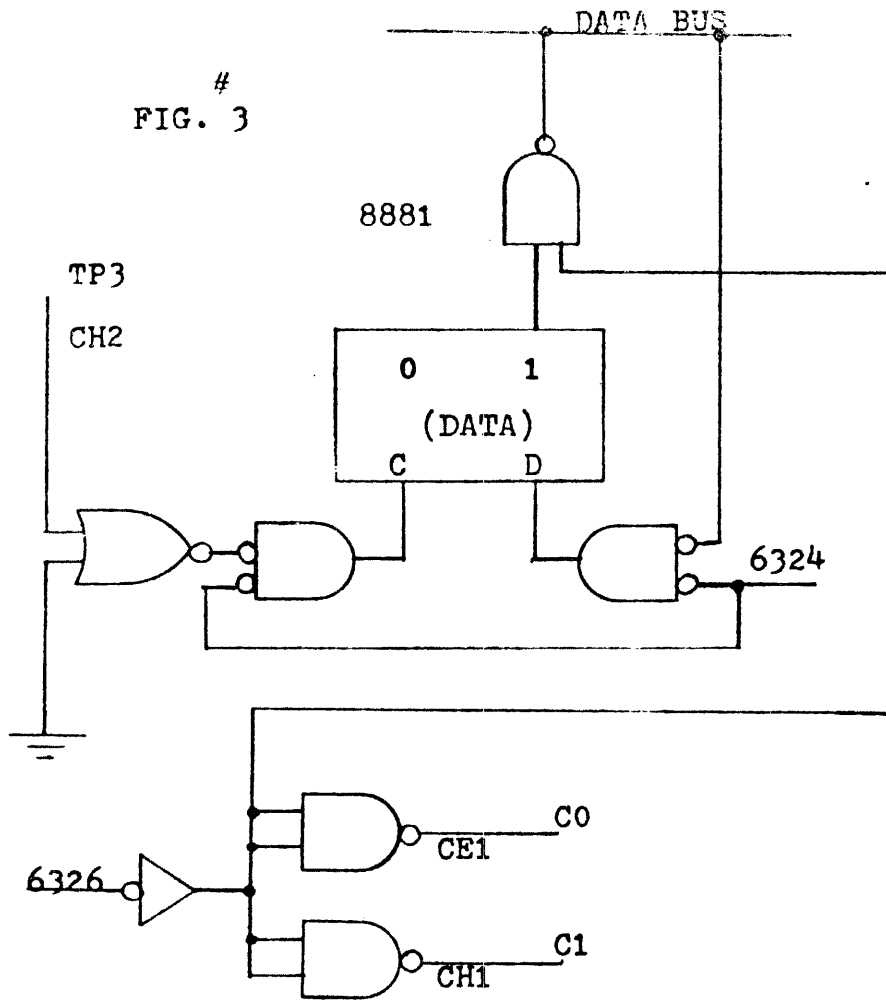
CO C1 C2
H H H = AC √ DATA ---- AC

FIG. 2



THE PURPOSE OF THIS GATING IS TO ALLOW THE TRANSFER OF THE AC TO A REGISTER IN THE DEVICE.

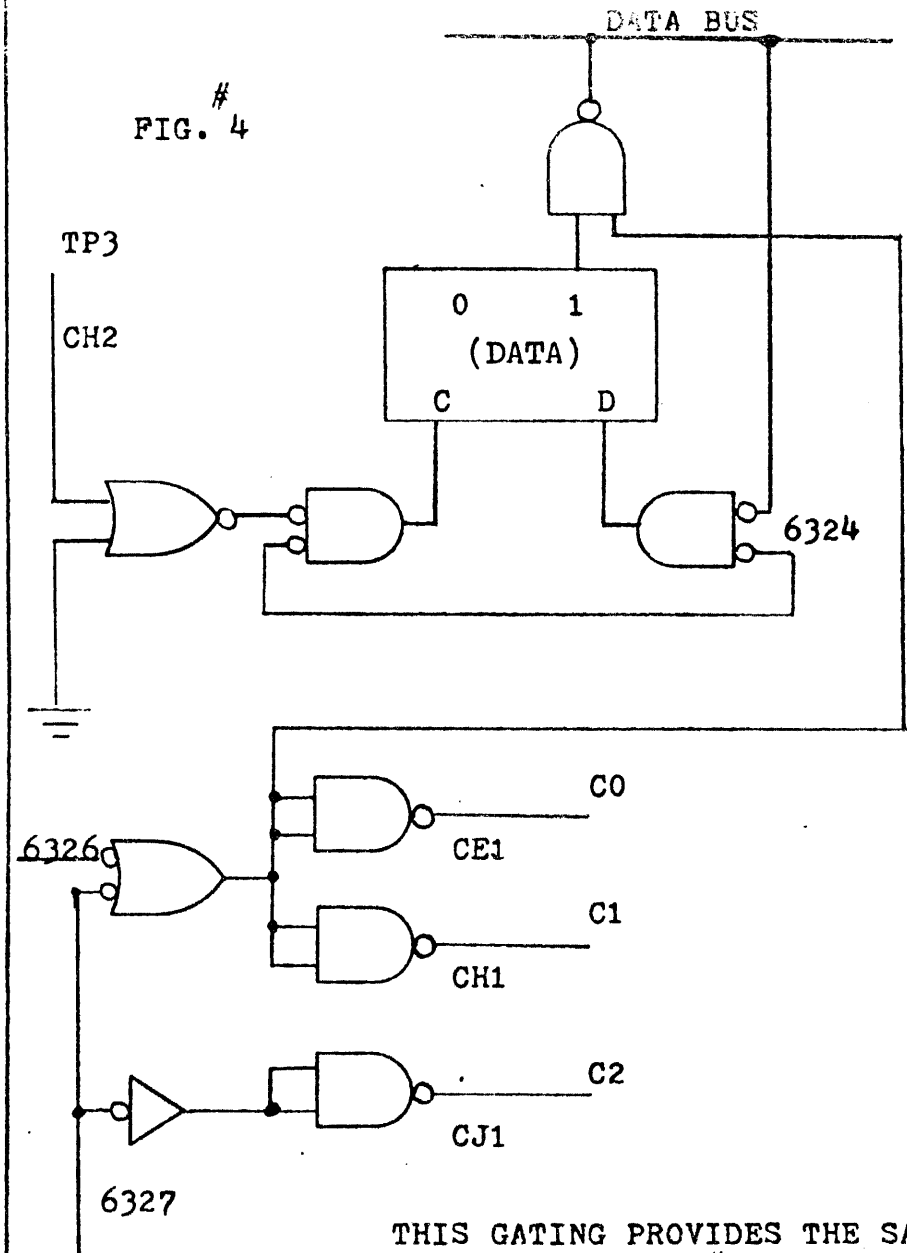
FIG. 3



THIS GATING HAS THE SAME FUNCTION AS THAT OF FIG.#2 PLUS THE DATA FROM THE DEVICE REGISTER CAN BE TRANSFERED TO THE AC.

DATA → AC = Co=L, C1=L, C2=H

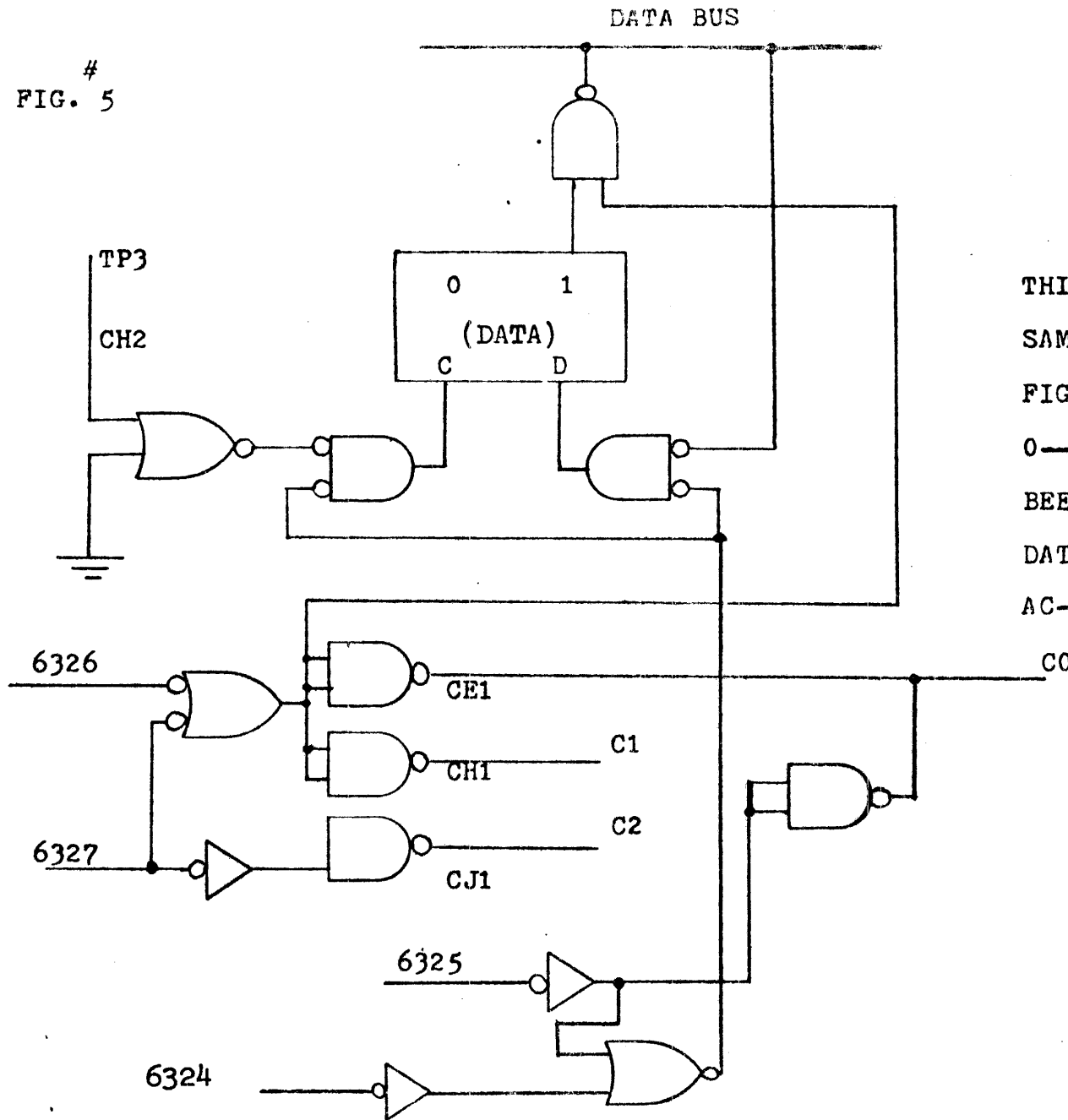
FIG. 4



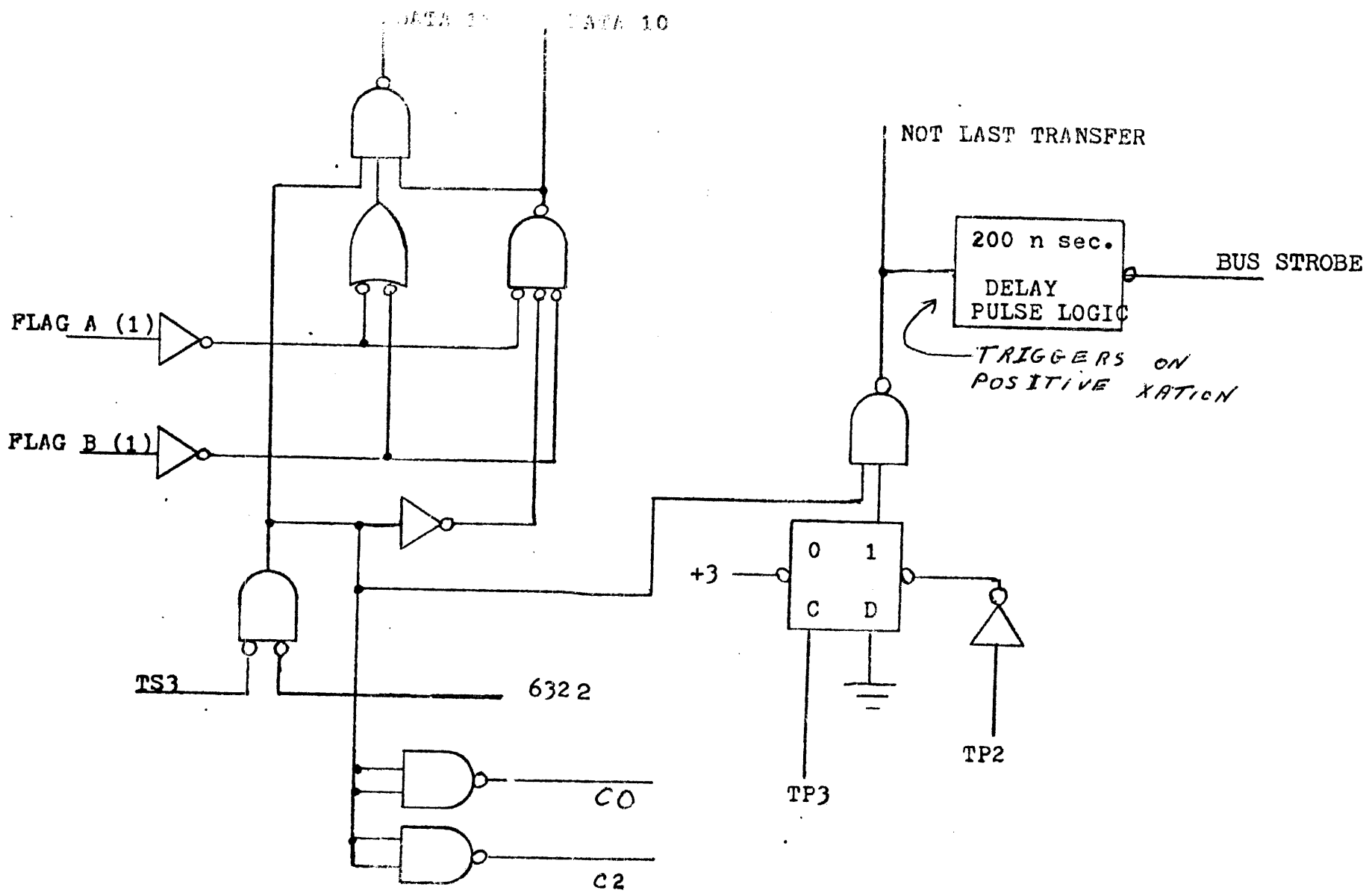
THIS GATING PROVIDES THE SAME # FUNCTIONS AS THOSE OF FIG. 2 and 3 PLUS THE TRANSFERING OF DATA → PC.
DATA → PC = Co=L, C1=L, C2=H

AC → DATA, 0 → AC

FIG. 5



THIS GATING PROVIDES THE
SAME FUNCTIONS AS THOSE OF
FIG. 2, 3, and 4 PLUS THE
0 → AC AFTER THE AC HAS
BEEN TRANSFERRED TO THE
DATA BUS. (0 → AC @ BUS STROBE
AC → DATA, 0 → AC
C0=L, C1=H, C2=H



1. BEFORE THE CPU IS ABLE TO ACKNOWLEDGE ANY INTERRUPT REQUESTS FROM DEVICES, THE INTERRUPT SYSTEM MUST BE ENABLED. THE FOLLOWING IS A PROCEDURE IN ORDER TO ENABLE THE INTERRUPT SYSTEM.
 - a) AN ION INSTRUCTION, 6001, MUST BE PERFORMED. THIS INSTRUCTION WILL "TURN ON" THE INTERRUPT SYSTEM ONLY HALF WAY.
 - b) ANY INSTRUCTION THAT FOLLOWS THE 6001 WILL FULLY ENABLE THE INTERRUPT SYSTEM.

2. WHEN THE INTERRUPT SYSTEM HAS BEEN ENABLED, THE CPU IS CAPABLE OF ACKNOWLEDGING INTERRUPT REQUESTS. THE FOLLOWING IS A GENERAL DESCRIPTION OF ACKNOWLEDGING AN INTERRUPT REQUEST:
 - a) THE CPU WILL ACKNOWLEDGE AN INTERRUPT REQUEST AT TP3 TIME OF ANY MAJOR STATE PROVIDING THAT FETCH IS THE NEXT MAJOR STATE TO OCCUR. BY ACKNOWLEDGING THE INTERRUPT REQUEST THE NORMAL OCCURENCES OF TS4 ARE DISABLED. INSTEAD OF PERFORMING PC---MA (OR PC + 1----MA FOR A SKIP CONDITION) O's ARE FORCED TO THE MA. THEREFORE; THE NEXT ABSOLUTE ADDRESS TO BE REFERENCED IS 0000. ALONG WITH FORCING THE CPU TO GO TO ADDRESS 0000, THE IR IS FORCED TO DECODE A JMS INSTUCTION.

NOW THE CPU IS FORCED TO PERFORM A JMS INSTRUCTION AT ADDRESS 0000. SINCE THE CPU'S IR IS FORCED TO DECODE A JMS AND ABSOLUTE ADDRESS 0000 WAS FORCED TO THE MA, THE CPU CAN ELIMINATE THE NEED FOR A FETCH CYCLE. THE CONTROLLING LOGIC FOR CHANGING MAJOR STATES IS FORCED TO PRODUCE THE CONDITIONS THAT WILL ALLOW THE MAJOR STATE OF EXECUTE TO OCCUR NEXT. THE VALUE OF THE PC (WHICH IS THE NEXT INSTRUCTION'S ADDRESS FOR THE MAIN PROGRAM) WILL BE STORED IN ADDRESS 0000. ADDRESS 0000 IS NOW THE ENTRANCE POINT FOR RETURNING TO THE MAIN PROGRAM. THE FIRST INSTRUCTION OF THE SUBROUTINE WILL BE PERFORMED AT ADDRESS 0001. AT THIS POINT AN INTERROGATION ROUTINE IS EXECUTED (SEE HANDOUT [#] 28).

AS THE JMS IS EXECUTED, THE INT. SYS. IS AUTOMATICALLY SHUT OFF.

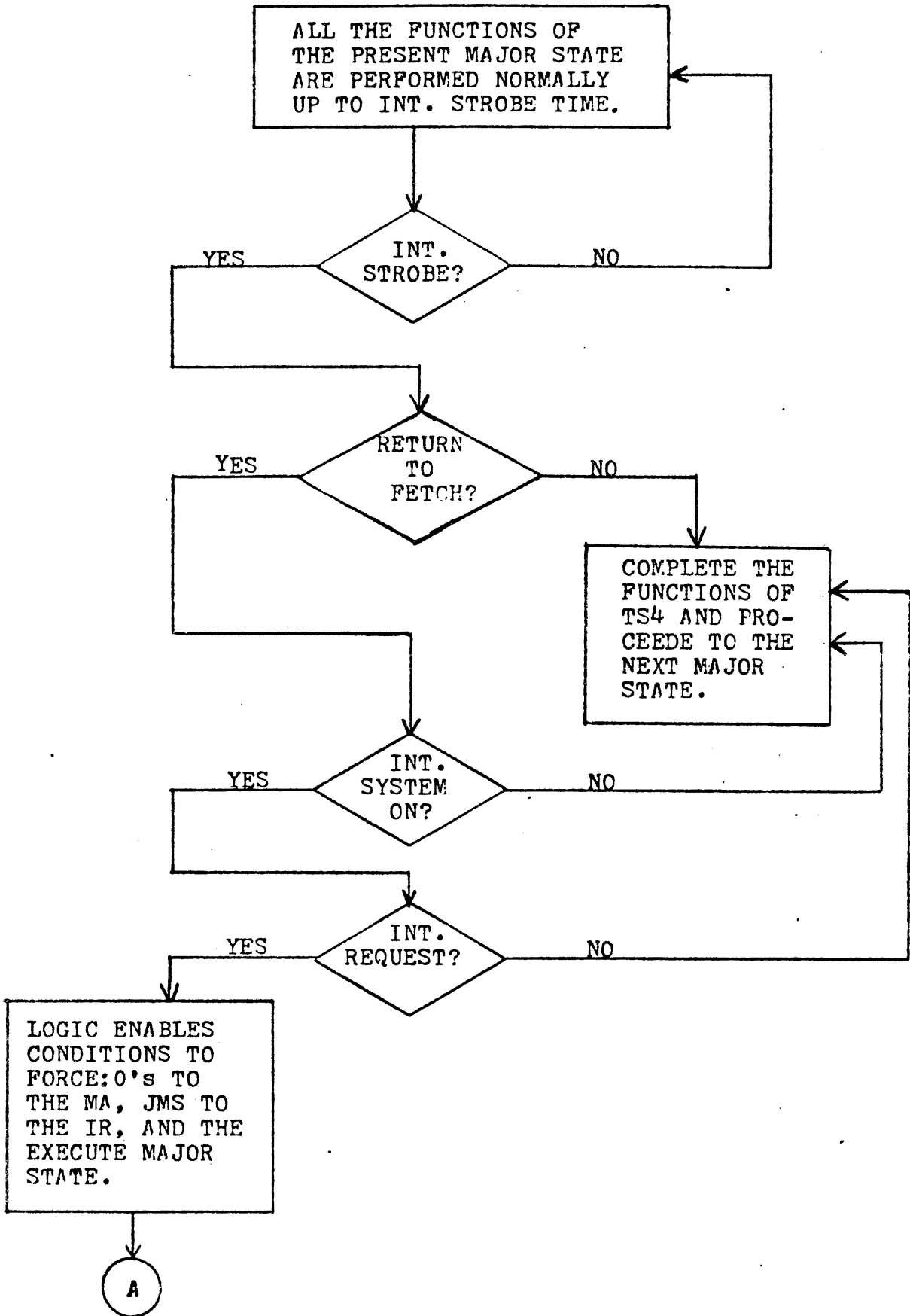
3. UPON THE COMPLETION OF THE INTERROGATION AND THE SERVICE ROUTINES, PROCEDURES FOR RETURNING TO THE MAIN PROGRAM ARE AS FOLLOWS:

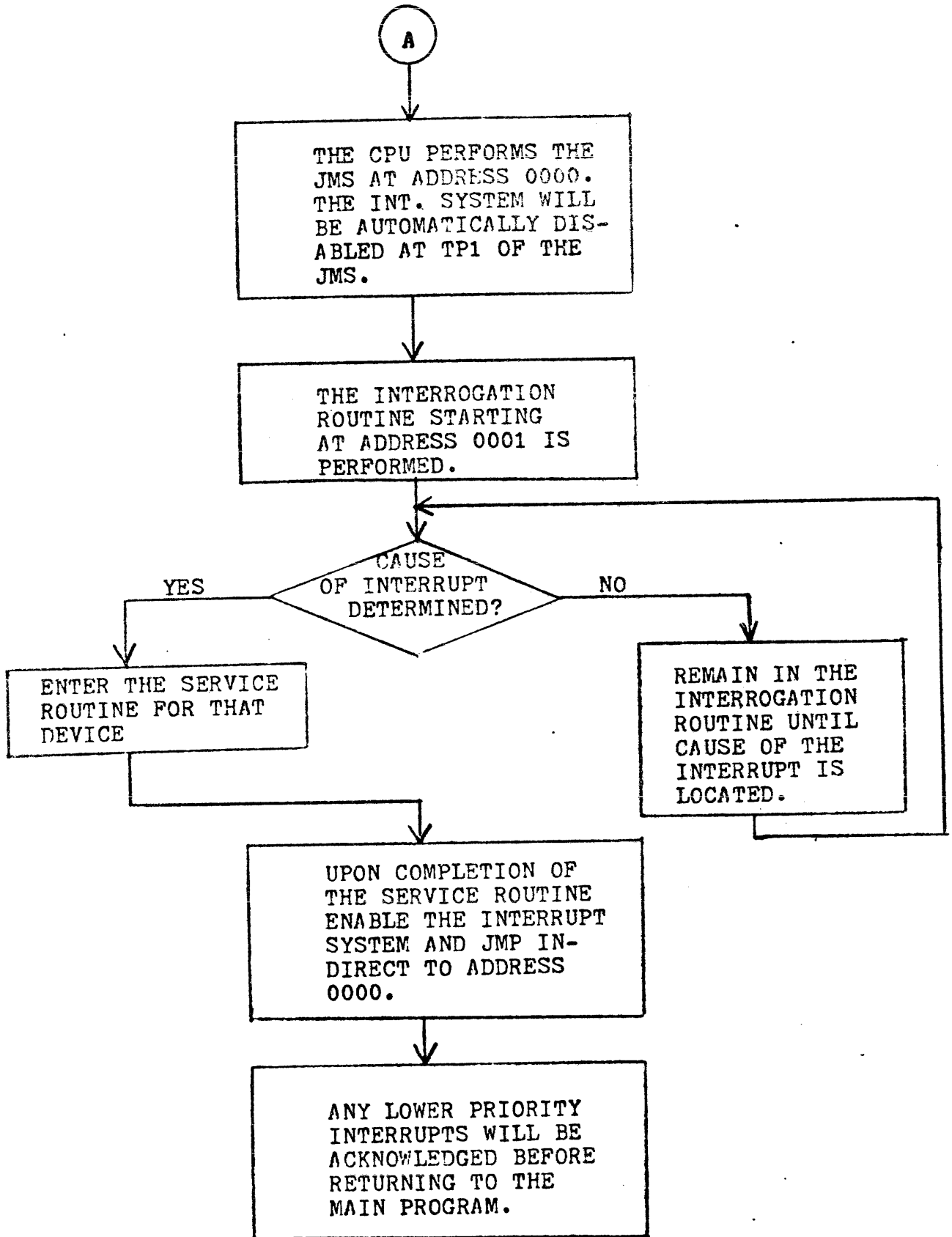
a) BECAUSE THE INTERRUPT SYSTEM WAS DISABLED DURING THE JMS, WHICH ALLOWED THE INTERRUPT BEING PROCESSED NOT TO BE INTERRUPTED BY ANOTHER INTERRUPT, THE INTERRUPT SYSTEM SHOULD BE ENABLED TO ACKNOWLEDGE ANY LOWER PRIORITY INTERRUPTS BEFORE RETURNING TO THE MAIN PROGRAM. THIS IS ACCOMPLISHED BY PERFORMING AN ION

INTERRUPT SEQUENCE DESCRIPTION

SHEET 3

INSTRUCTION AT THE NEXT TO THE LAST ADDRESS OF THE
SERVICE ROUTINE PROGRAM. THE ION INSTRUCTION IS
THEN FOLLOWED BY THE JMP INDIRECT TO ADDRESS 0000.
IF ANY LOWER PRIORITY INTERRUPTS ARE PRESENT AT THIS
TIME, THEY WILL BE ACKNOWLEDGED (ACCORDING TO PRIORITY)
AND THE ORIGINAL CONTENTS OF ADDRESS 0000, WHICH HOLDS
THE PC OF THE FIRST INTERRUPT WILL NOT BE CHANGED.





EXAMPLE PROGRAM

```
0000/ PC (XXXX)
0001/ DCA 3050
0002/ GTF 6004
0003/ DCA 3051
0004/ JMP I 5405
0005/ 7000
```

```
7000/ 6XXX
7001/ SKP 7410
7002/ JMP 5250
7003/ 6XXX
7004/ SKP 7410
7005/ JMP 5300
7006/ 6XXX
7007/ SKP 7410
7010/ JMP 5350
```

DESCRIPTION OF
BASIC INTERROGATION ROUTINE

THE ABOVE PROGRAM IS AN EXAMPLE OF A BASIC INTERROGATION ROUTINE. IN ADDRESS 0000 THE PC (WHICH IS THE ADDRESS OF THE NEXT INSTRUCTION FOR THE MAIN PROGRAM) IS STORED FROM THE FORCED JMS OF THE INTERRUPT SYSTEM. THE DCA 3050 LOCATED IN ADDRESS 0001 IS USED TO STORE THE AC VALUE IN CASE THE DEVICE CAUSING THE INTERRUPT TRANSFERS INFORMATION TO THE AC. THE CONTENTS OF ADDRESS 0002, A GTF, IS USED TO BRING THE VALUE OF THE LINK INTO THE AC. THE GTF IS FOLLOWED BY A DCA 3051 , IN LOCATION 0003, SO THAT THE VALUE OF THE LINK MAY BE STORED IN MEMORY. THE JMP I LOCATED IN ADDRESS 0004 IS USED TO EXIT PAGE "0". THE REASON FOR EXITING PAGE"0" IS THAT IT IS USED FOR AUTO INDEX PURPOSES AS WELL AS DIRECT ACCESS FROM ALL PAGES.

THE JMP I TAKES THE PROGRAM TO ADDRESS 7000 WHERE THE INTERROGATION ROUTINE BEGINS. THE FIRST PART OF THE PROGRAM WAS WRITTEN FOR "HOUSE CLEANING" PURPOSES. THE INSTRUCTION LOCATED IN ADDRESS 7000, LISTED AS 6XXX, WILL CHECK INTERRUPT FLAG OF THE FIRST DEVICE (PRIORITY 0) FOR A SKIP CONDITION. IF THAT DEVICE'S INTERRUPT FLAG WAS SET, A SKIP CONDITION WOULD RESULT CAUSING THE PROGRAM TO REFERENCE ADDRESS 7002 INSTEAD OF 7001. THE INSTRUCTION AT ADDRESS 7002, AJMP 5250, WILL ENTER THE SERVICE ROUTINE FOR DEVICE PRIORITY 0.

EXAMPLE PROGRAM

IF DEVICE 0'S INTERRUPT FLAG WAS NOT SET, THE PROGRAM WOULD REFERENCE ADDRESS 7001. AN UNCONDITIONAL SKIP, 7410, LOCATED AT ADDRESS 7001 WILL CAUSE THE PROGRAM TO SKIP OVER THE JMP 5250. THUS THE ENTRANCE POINT FOR THE SERVICE ROUTINE OF DEVICE 0 IS SKIPPED OVER.

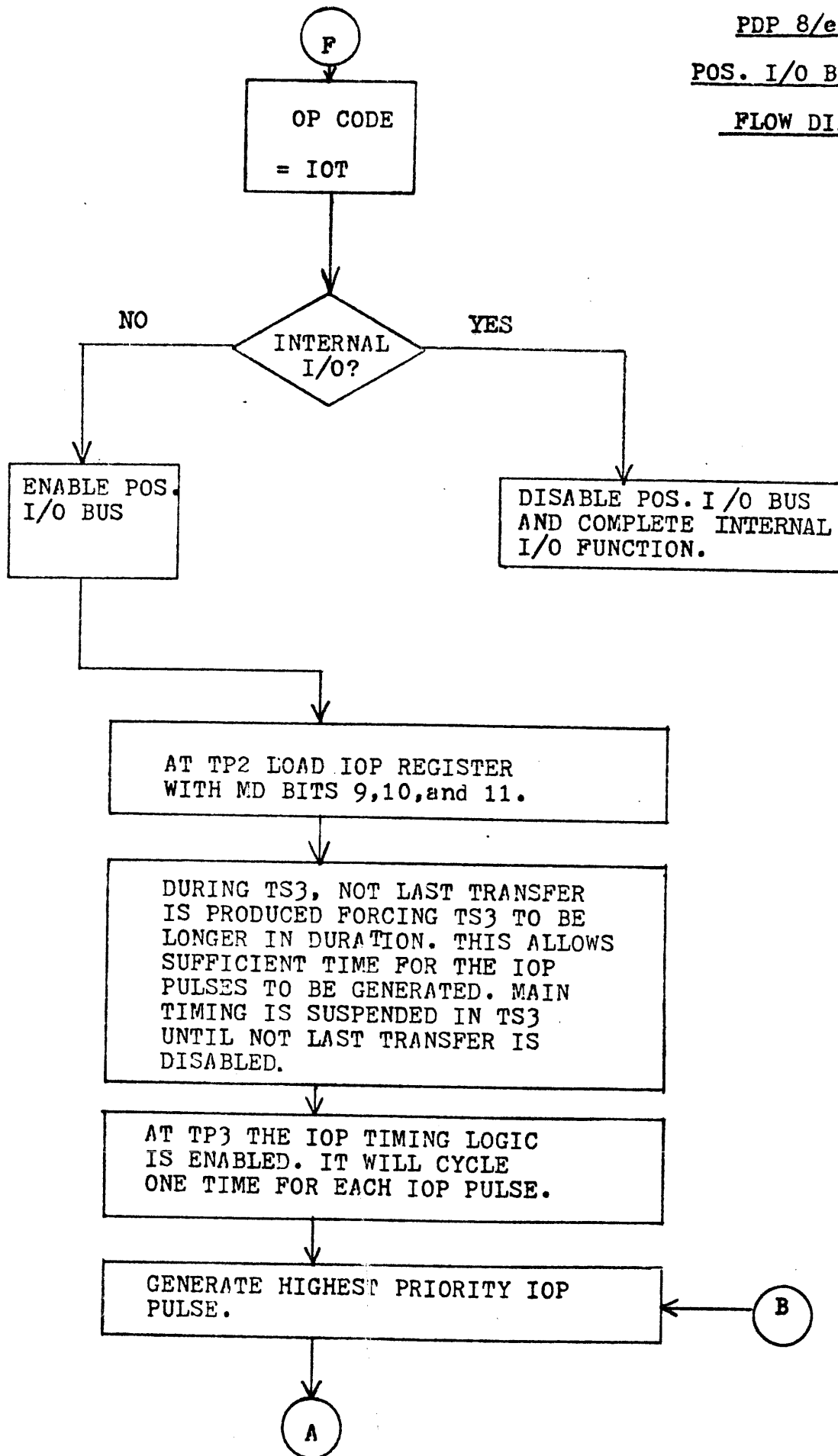
THE SAME SEQUENCE OF CHECKING THE INTERRUPT FLAGS APPLIES TO THE OTHER TWO DEVICES.

THE NEXT PORTION OF THE EXAMPLE PROGRAM DEMONSTRATES HOW TO ENTER THE MAIN PROGRAM AND ALSO ENABLE THE INTERRUPT SYSTEM. THIS PORTION OF THE PROGRAM WILL BE EXECUTED AFTER THE MAIN PORTION OF THE SERVICE ROUTINE HAS BEEN COMPLETED.

(AS AN EXAMPLE, DEVICE PRIORITY 1 WILL BE USED)

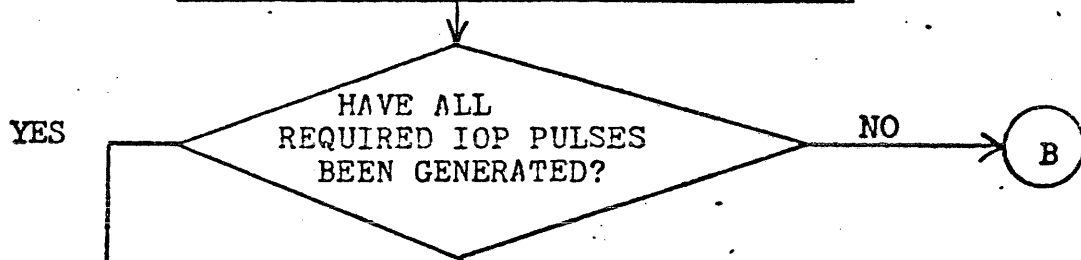
| | | |
|-------|-----|------|
| 7124/ | CLA | 7200 |
| 7125/ | TAD | 1051 |
| 7126/ | RTF | 6005 |
| 7127/ | CLA | 7200 |
| 7130/ | TAD | 1050 |
| 7131/ | ION | 6001 |
| 7132/ | JMP | 5400 |

AT LOCATION 7124, A CLA CLEARS THE AC IN PREPARATION FOR RESTORING THE VALUES THAT WERE STORED IN MEMORY BY THE "HOUSE CLEANING" PORTION OF THE PROGRAM. THE TAD INSTRUCTION, 1051, LOCATED AT ADDRESS 7125 WILL BRING THE ORIGINAL CONDITION OF THE LINK, PRIOR TO THE INTERRUPT, INTO THE AC. FOLLOWING THE TAD INSTRUCTION IS A RTF INSTRUCTION. THIS INSTRUCTION WILL RETURN THE ORIGINAL CONTENTS OF THE LINK, WHICH IS NOW IN THE AC, BACK INTO THE LINK. THE NEXT LOCATION, 7127, HOLDS A CLA INSTRUCTION. THIS WILL BE USED TO CLEAR THE AC IN PREPARATION FOR RESTORING THE ORIGINAL AC, PRIOR TO THE INTERRUPT. AFTER THE AC HAS BEEN CLEARED BY THE CLA, THE TAD INSTRUCTION, 1050, AT ADDRESS 7130 WILL BRING THE ORIGINAL CONTENTS OF THE AC, PRIOR TO THE INTERRUPT, BACK INTO THE AC. THE 6001 INSTRUCTION, ION, WILL PARTIALLY ENABLE THE INTERRUPT SYSTEM. THE LAST INSTRUCTION, 5400, WILL FULLY ENABLE THE INTERRUPT SYSTEM AND ALLOW THE PROGRAM TO RETURN TO ADDRESS 0000. FROM THIS POINT THE PROGRAM CAN ENTER THE MAIN PROGRAM OR ACKNOWLEDGE ANY LOWER PRIORITY INTERRUPTS THAT MAY BE PRESENT.



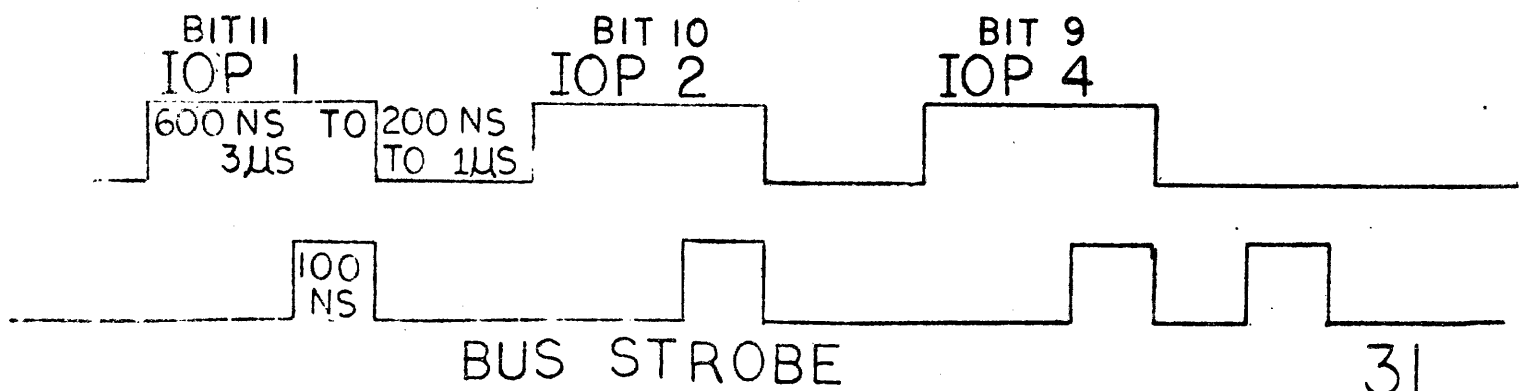


100 n. sec. PRIOR TO THE TERMINATION OF THE IOP PULSE, A BUS STROBE PULSE IS PRODUCED. BUS STROBE PRODUCES AC LOAD IF C2=H OR A PC LOAD IF C2=L.



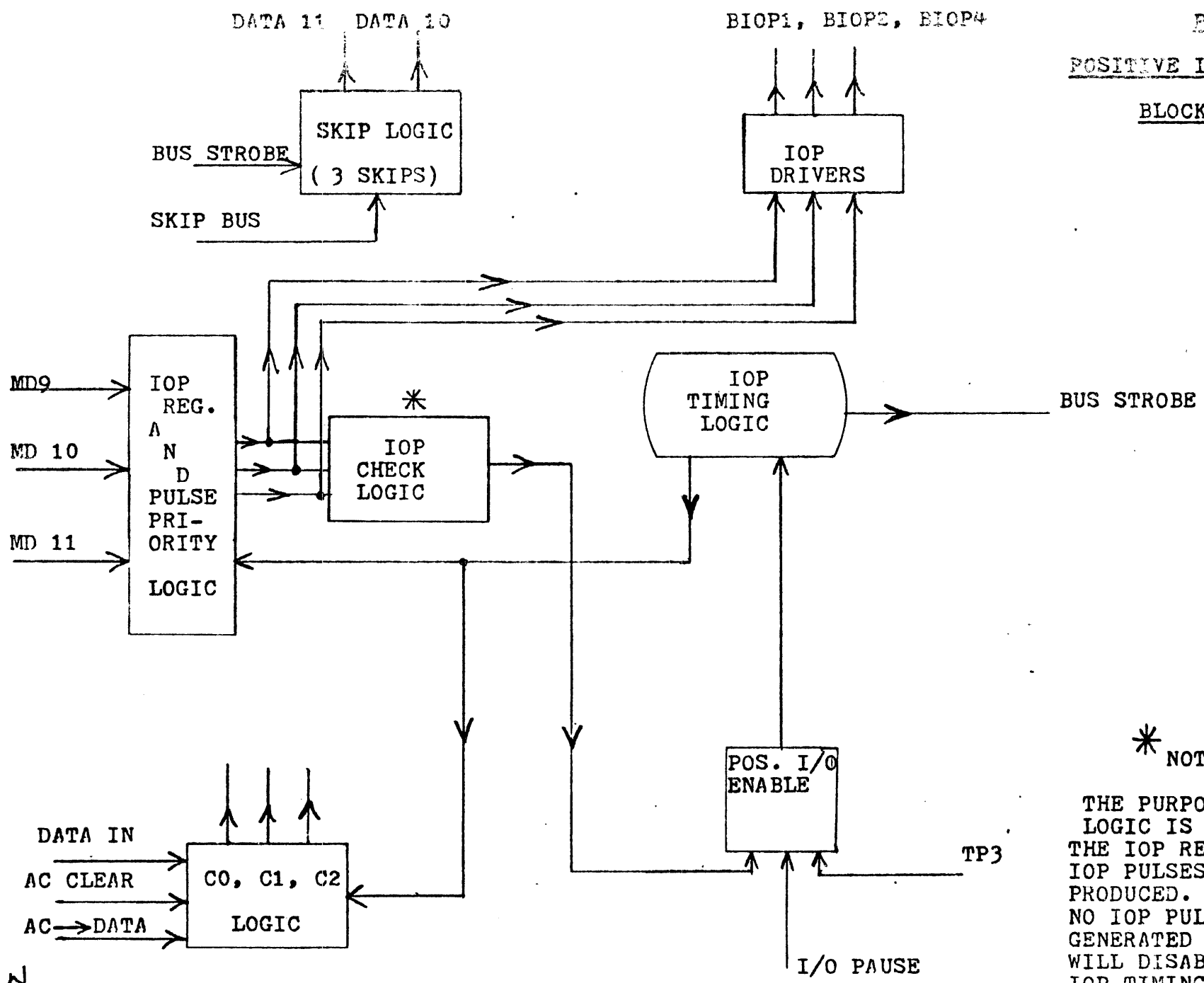
DISABLE NOT LAST TRANSFER. ENABLE IOP TIMING LOGIC TO PRODUCE A "DUMMY CYCLE" (NO IOP PULSES ARE PRODUCED). THE DUMMY CYCLE IS USED TO ADD THE SKIP COUNTS (IF ANY) TO THE PC. (PC +DATA → PC)

BUS STROBE OF THE DUMMY CYCLE THE PC IS LOADED WITH PC + DATA. BECAUSE OF NOT LAST TRANSFER BEING DISABLED, TS3 IS DISABLED AT BUS STROBE AND TS4 IS ENABLED. MAIN TIMING IS NOW FREED TO RESUME THE REMAINING FUNCTIONS.



POSITIVE I/O INTERFACE

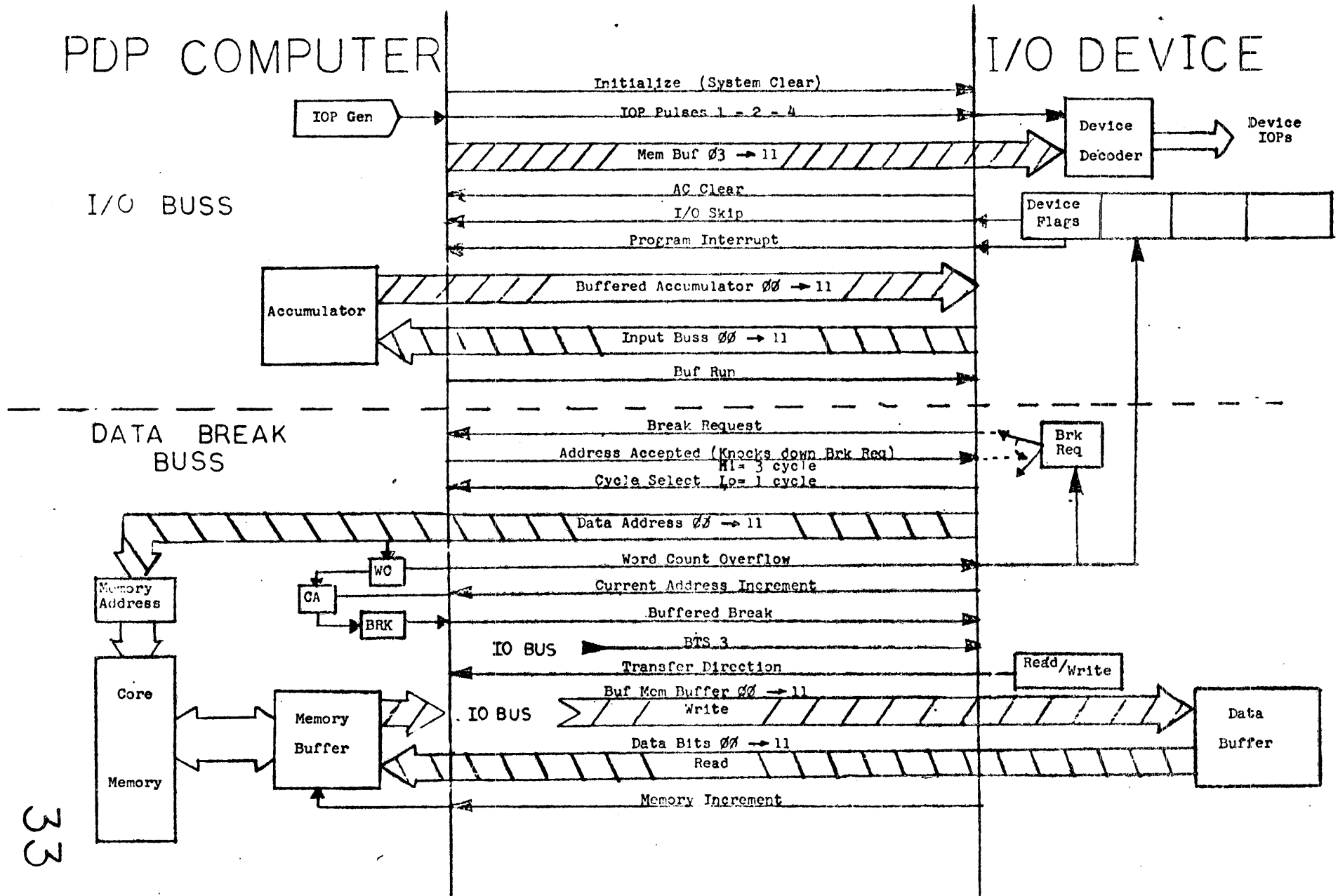
BLOCK DIAGRAM



* NOTE:
 THE PURPOSE OF THIS LOGIC IS TO CHECK THE IOP REGISTR FOR IOP PULSES TO BE PRODUCED. IF THERE ARE NO IOP PULSES TO BE GENERATED THIS LOGIC WILL DISABLE THE IOP TIMING LOGIC.

PDP COMPUTER

I/O DEVICE



STATE FUNCTIONS

OF

3 CYCLE BREAK

| WORD COUNT | CURRENT ADDRESS | BREAK |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>THE PURPOSE OF THIS STATE IS TO INCREMENT A MEMORY WORD LOCATED AT A KNOWN MEMORY ADDRESS. THIS KNOWN MEMORY ADDRESS IS SPECIFIED BY THE 3 CYCLE DEVICE. THE MEMORY WORD LOCATED AT THIS ADDRESS IS THE 2'S COMPLEMENTED VALUE OF THE NUMBER TRANSFERS TO BE PERFORMED. IF THE MEMORY WORD IS INCREMENTED TO 0000, <u>WORD COUNT OVERFLOW</u> IS SENT TO THE DEVICE TO INHIBIT ANY MORE BREAK REQ.</p> | <p>THE PURPOSE OF THIS STATE IS TO SPECIFY AN ADDRESS AT WHICH THE TRANSFER TAKES PLACE. THIS FUNCTION CAN BE DONE 2 WAYS. 1) A KNOWN VALUE, LOCATED AT THE INCREMENTED VALUE OF THE MEMORY ADDRESS OF THE WORD COUNT STATE WILL BE INCREMENTED EACH TIME THIS STATE IS REFERENCED. THIS INCREMENTED VALUE IS USED FOR THE MA OF THE BREAK STATE, THUS ALLOWING SEQUENTIAL DATA TRANSFERS FROM MEMORY. 2) THE KNOWN VALUE NEED NOT BE INCREMENTED ALLOWING DATA TRANSFERS AT A CONSTANT ADDRESS. HOWEVER; THIS METHOD REQUIRES A BACKGROUND PROGRAM.</p> | <p>THE PURPOSE OF THIS STATE IS TO PERFORM THE TRANSFERS. THERE ARE 4 FUNCTIONS WHICH MAY BE ACCOMPLISHED</p> <ol style="list-style-type: none">1) DATA MAY BE TRANSFERED FROM THE DEVICE TO THE CPU.2) DATA FROM THE CPU MAY BE TRANSFERED TO THE DEVICE.3) DATA FROM THE DEVICE MAY BE ADDED WITH A MEMORY WORD FROM THE CPU.4) A MEMORY WORD OF THE CPU MAY BE INCREMENTED. <p>ALL 4 FUNCTIONS ARE PERFORMED AT A MEMORY ADDRESS SPECIFIED BY THE CURRENT ADDRESS STATE.</p> |

OPTIONS FOR BREAK MAJOR STATE

WRITE
(DATA OUT)

(MEM) → MDL → PERIPH.

CONTENTS OF MEMORY
SENT TO PERIPH AND
WRITTEN BACK INTO
MEMORY

DI (0)
INC (0)

READ
(DATA IN)

DATA LINES → DATA BUS → MB
MDL → (MEM)

DATA FROM PERIPH
WRITTEN INTO
MEMORY

DI (1)
INC (0)

INC (MEM)

(MEM) → MDL DATA BUS = 1
DATA BUS + MDL → MB → (MEM)

CONTENTS OF MEMORY
+1 WRITTEN BACK INTO
MEMORY
(USED AS A COUNTER)

DI (0)
INC (1)

DATA ADDED TO

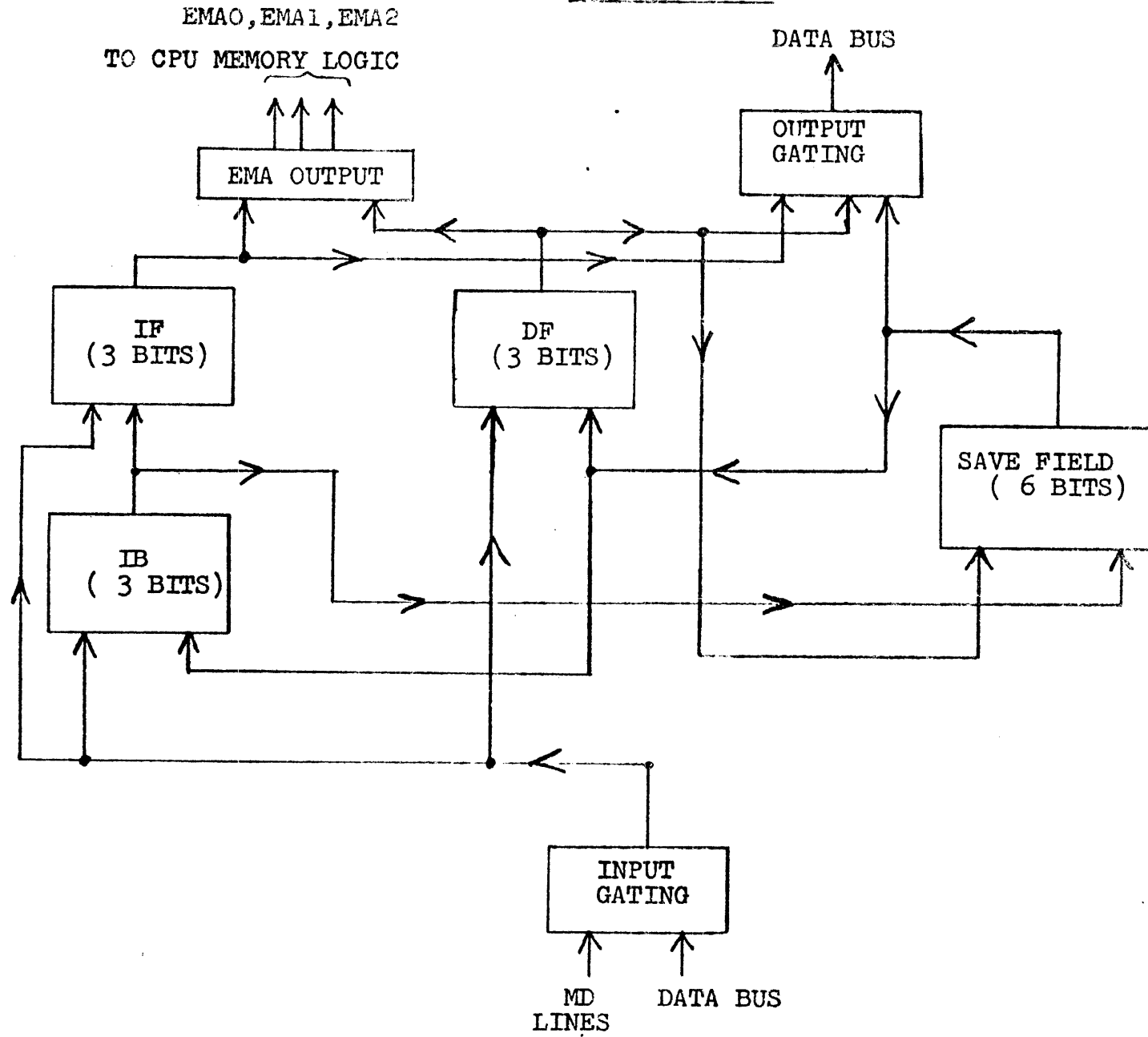
(MEM) → MDL DATA LINES → DB
DATA BUS + MDL → MB → (MEM)

BREAK DATA IS ADDED
TO THE CONTENTS OF
MEMORY AND WRITTEN
BACK INTO MEMORY

DI (1)
INC (1)

EXTENDED MEMORY CONTROL

BLOCK DIAGRAM



INSTRUCTION FIELD AND DATA FIELDREGISTERS

THE FOLLOWING INFORMATION DESCRIBES THE RELATIONSHIP BETWEEN THE CPU MAJOR STATES AND EXTENDED MEMORY FUNCTIONS.

KEY FUNCTIONS: ALTHOUGH THE CPU IS IN NO MAJOR STATE, EXTENDED MEMORY MUST STILL CONTROL WHAT FIELD WILL BE REFERENCED DURING KEY FUNCTIONS. BECAUSE OF THIS, THE PROGRAMMER HAS A CHOICE OF ANY FIELD (LIMITED, OF COURSE, TO THE AMOUNT OF CORE ON THE SYSTEM) TO MANUALLY DEPOSIT OR EXAMINE DATA. THE CONTENTS OF THE IF REGISTER DETERMINES WHICH FIELD WILL BE REFERENCED DURING KEY FUNCTIONS.

FETCH: THE CONTENTS OF THE IF REGISTER WILL DETERMINE WHAT FIELD WILL BE REFERENCED TO OBTAIN INSTRUCTIONS. THIS CONDITION IS TRUE FOR ANY FETCH MAJOR STATE.

DEFER: THE CONTENTS OF THE IF REGISTER WILL DETERMINE FROM WHAT FIELD THE POINTER ADDRESS WILL BE OBTAINED. * HOWEVER; AT TP4 TIME OF AT TP4 TIME OF EACH MAJOR STATE EXT. MEM. DEFER EITHER THE IF OR THE DF REGISTER CAN BE THE REGISTER THAT WILL SPECIFY THE FIELD CONT. LOGIC DECIDES IF THE IF OR DF WILL SPECIFY THE NEXT FIELD TO REFERENCE.

INSTRUCTION FIELD AND DATA FIELDREGISTERS

DEFER----- TO BE REFERENCED FOR THE EXECUTE MAJOR
(cont.) STATE. THIS IS DEPENDENT UPON WHAT TYPE
OF INSTRUCTION IS BEING PERFORMED.

IF AN AND, TAD, ISZ, OR DCA IS
BEING PERFORMED, THE CONTENTS OF THE DF
REGISTER WILL DETERMINE WHICH FIELD WILL
BE REFERENCED TO OBTAIN DATA FOR THE
EXECUTE MAJOR STATE. IF THE CONTENTS OF
THE DF \neq IF THE PROGRAMMER NOT ONLY HAS
THE CAPABILITY OF OBTAINING DATA FROM
ANY MEMORY PAGE; HE ALSO HAS THE
CAPABILITY OF REFERENCING A DIFFERENT
MEMORY FIELD IN ORDER TO OBTAIN DATA.
ON THE OTHER HAND; IF THE DF = IF, A
NORMAL INDIRECT IS PERFORMED IN THE SAME
FIELD THE INSTRUCTION AND THE POINTER WERE
OBTAINED.

IF A JMS OR JMP IS BEING PERFORMED,
THE DF REGISTER WILL NOT SPECIFY THE
FIELD TO BE REFERENCED IN EXECUTE; THE
IF REGISTER WILL. THE REASON FOR THIS IS
THAT A JMP NEVER ENTERS THE EXECUTE STATE.
IF THE DF REGISTER COULD SPECIFY WHICH
FIELD TO REFERENCE DURING THE NEXT MAJOR

INSTRUCTION FIELD AND DATA FIELDREGISTERS

DEFER ----- STATE, WHICH WOULD BE FETCH, THE NEXT
(cont.) INSTRUCTION WOULD BE OBTAINED FROM THE
FIELD SPECIFIED BY THE DF REGISTER. THIS
WOULD DEFEAT THE PURPOSE OF THE DF
REGISTER AS IT IS USED ONLY TO OBTAIN
DATA; NOT INSTRUCTIONS.

BECAUSE THE JMP INDIRECT CANNOT USE
THE DF REGISTER TO SPECIFY A FIELD TO
REFERENCE FOR THE EXECUTE MAJOR STATE,
THE JMS INSTRUCTION IS FORCED TO DO THE
SAME. IF THE JMS INSTRUCTION COULD USE
THE DF REGISTER TO SPECIFY WHICH FIELD TO
REFERENCE FOR THE EXECUTE MAJOR STATE,
THE PC WOULD BE STORED IN THAT PARTICULAR
FIELD. HOWEVER; UPON RETURNING FROM THE
SUBROUTINE, WHICH REQUIRES THE USE OF A
JMP INDIRECT TO THE ADDRESS WHERE THE PC IS
LOCATED, THE FIELD WHERE THE PC IS STORED
COULD NOT BE REFERENCED. WHY? A JMP
INDIRECT CANNOT REFERENCE A FIELD SPECIFIED
BY THE DF REGISTER.

EXECUTE ----- THE EXECUTE MAJOR STATE CAN REDERENCE
A FIELD SPECIFIED BY THE IF OR DF.
(SEE ABOVE INFORMATION OF DEFER FOR EXPLAN-
ATION)

FETCH FOR AND, TAD, ISZ, DCA, JMS INSTRUCTIONS

| Function | Source Req'd | Route Req'd | Dest'n Req'd | Enabling Levels & Boolean Expression | Source Page & Cord | Dest'n Page & Cord | Time | Major State |
|----------------------------|--------------|-------------|--------------|----------------------------------------------------------------------|--------------------|--------------------|------|-------------|
| MA+1→PC | CPMA | +1 | PC | ENOL, EN1H, EN2H = TS1 (BF + KC) | 11H7 | 6C8 | TS1 | FETCH |
| | | | | CARRY IN L = TS1 (BF + KC) | 11H1 | 9 | | |
| O→SKIP | | | | PC LOAD = TP1 (BF + KC) | 10J5 | 6B2 | TP1 | |
| SA→MD | SA | | MD LINES | O→SKIP = BF · TP1 | 10H1 | 9J1 | TS2 | |
| | | | | TIME STROBE = (STROBE · FIELD · WRITE) DELAYED | 5B8 | 5E1-8 | | |
| AC→DATA | AC | | | MD DIR L = 100NS INTO TS1, O→MD DIR | 14C7 | 5E1-8 | | |
| MD→MB | MD LINES | | MB | AC→BUS L = BF · TS2 | 11J3 | 6C6 | TP2 | |
| | | | | ENOL, EN1L, EN2H = (DCA · E · TS2) · BTS2 · (MS DIS + BRK DATA CONT) | 11H5 | 6C8 | | |
| MDO-2→IR | | | IR | MB LOAD = BTP2 | 10H2 | 6C2 | | |
| | | | | LOAD IR = TP2 · F | 12E5 | 12E5 | | |
| | | | | | | | TS3 | |
| | | | | | | | TP3 | |
| MD 5-11 → CPMA | MD5-11 | FOR PAGING | | RIGHT L, LEFT L, TWICE L = (BTS4 · F SET · BF · FE + FD) | 10J5 | 6F8 | TS4 | |
| MD 4(0) O→CPMA 0-4 | 0 | | | PAGE Z H ₂ (MD4(0) · (BTS4 · FSET · BF · FE + FD) | 10J5 | 6F8 | | |
| MD4(1) CPMA 0-4 → CPMA 0-4 | CPMA 0-4 | | | PAGE Z L = MD4(1) · (BTS4 · FSET · BF · FE + FD) | 10J5 | 6F8 | | |
| | | | CPMA | CPMA LOAD = TP4 · MALC | 12H4 | 6F3 | TP4 | |
| MD 3(0) I→E | | | EXECUTE | CPMA LOAD (F · OPR + IOT · MD3(0) · JMP) = E L | 12B7 | 12B4 | | |
| MD 3(1) I→D | | | DEFER | CPMA LOAD (F · OPR+IOT · MD3(1)) = D L | 12B7 | 12B4 | | |

PDP-8E MECHANIZATION TABLE

DEFER FOR AND, TAD, ISZ, DCA, JMS INSTRUCTIONS

| Function | Source Req'd | Route Req'd | Dest'n Req'd | Enabling Levels & Boolean Expression | Source Page & Cord | Dest'n Page & Cord | Time | Major Instruction State |
|----------------------------------|--------------|-------------|--------------|-------------------------------------------------------------------------|--------------------|--------------------|------|-------------------------|
| | | | | | | | TS1 | DEFER |
| | | | | | | | TP1 | |
| SA→MD | SA | | | TIME STROBE = (STROBE · FIELD · WRITE) DELAYED | 5B8 | 5E1-8 | TS2 | |
| | | | MD LINES | MD DIR L = 100NS INTO TS1, 0→MD DIR | 14C7 | 5E1-8 | | |
| MD+1→MB | MD | | | ENOL, EN1L, EN2H = (DCA · E · TS2) · BTS2 · (MS DIS + BRK DATA CONT) | 11H 5 | 6C8 | | |
| | | +1 | | CARRY IN L = BD · BTS2 | 11H1 | 9 | | |
| | | | MB | MB LOAD = BTP2 | 10H2 | 6C2 | TP2 | |
| AUTO INDEX MB→MD MD DIR H | MB | | | MD DIR H = TP2 (D · AUTO INDEX) | 14C7 | 6C3 | | |
| AUTO INDEX MEM→MD MD DIR L | MEM | | | MD DIR L = TP2 (D · AUTO INDEX) | 14C7 | 5E8 | | |
| | | | | | | | TS3 | |
| | | | | | | | TP3 | |
| MD→CPMA | MD | | | ENOL, EN1L, EN2H = BTS4 · (D · JMP) | 11H5 | 6C8 | TS4 | |
| | | | CPMA | CPMA LOAD = TP4 · MALC | 10H4 | 6F3 | TP4 | |
| 1→E | | | EXECUTE | CPMA LOAD · (D · JMP) = E L | 12B7 | 12B4 | | |

PIP-8E MEC. IZATION TABLE

EXECUTE FOR AND, TAD, ISZ, JMS INSTRUCTIONS

| Function | Source Req'd | Route Req'd | Dest'n Req'd | Enabling Levels & Boolean Expression | Source Page & Cord | Dest'n Page & Cord | Time | Major State | Instruc tion |
|-----------------------|--------------|-------------|--------------|----------------------------------------------------------------------------------------------------------------------------------|--------------------|--------------------|------|-------------|---------------|
| | | | | | | | TS1 | Execute | ALL |
| | | | | | | | TP1 | | |
| SA → MD | SA | | | TIME STROBE = (STROBE · FIELD · WRITE) | 5B8 | 5E1-8 | TS2 | | ALL |
| | | | MD LINES | MD DIR L = 100 NS INTO TSI, 0 → MD DIR | 14C7 | 5E1-8 | | | ALL |
| MD → MB | MD | | | ENOL, EN1L, EN2H = (DCA · E · TS2) BTS2 · (MS DIS + BRK DATA CONT) | 11H5 | 6C8 | | | AND TAD |
| MD+1 → MB | MD | | | ENOL, EN1L, EN2H = (DCA · E · TS2) BTS2 · (MS DIS + BRK DATA CONT) | 11H5 | 6C8 | | | ISZ |
| | | +1 | | CARRY IN = ISZ · E · TS2 | 11H2 | 9 | | | |
| AC → DATA | AC | | | AC → BUS = DCA · E · TS2 | 11J3 | 6C6 | | | DCA |
| DATA → MB | | BUS ENABLE | | DATA L, DATA F H = DCA · E · TS2 | 11J7 | 6D6 | | | |
| PC → MB | PC | | | ENOL, EN1L, EN2L = JMS · E · TS2 | 11H6 | 6C8 | | | JMS |
| | | | MB | MB LOAD = BTP2 | 10H2 | 6C2 | TP2 | | ALL |
| MB → MD | MD DIRH | | | MD DIR H = F + (D · AUTO INDEX) | 14C7 | 6C3 | | | ALL |
| CARRY OUT → OVERFLOW | | | | BTP3 · CARRY OUT = OVERFLOW | 10D7 | 10H3 | | | |
| AC ^ MB → AC | AC ^ MB | | | LEFT L, RIGHT L, TWICE H = AND · E · TS3 | 10H6 | 6F8 | TS3 | | AND |
| AC + MD → AC | AC | | | AC → BUS = TAD · E · TS3 | 11J3 | 6C6 | | | TAD |
| | | BUS ENABLE | | DATA T L, DATA F H = TAD · E · TS3 | 11J7 | 6D6 | | | |
| | MD | | | ENO L, EN1L, EN2H = TAD · E · TS3 | 11H5 | 6C8 | | | |
| 0 → AC | 0 | | | ENO H, EN1H, EN2H | 11J5 | 6C8 | | | DCA |
| MA+1 → PC | MA | | | ENO L, EN1H, EN2H = JMS · E · TS3 | 11F6 | 6C8 | | | JMS |
| | | +1 | | CARRY IN = (JMS · E · TS3) MA DIS · ROM ADDRESS | 11H2 | 9 | | | |
| CARRY OUT LINK → LINK | | | LINK | CARRY OUT · OP1 · OPE+IOT · TAD · E · BTP3 · LINK(1) = LINK(0) CARRY OUT · OP1 · OPE+IOT · TAD · E · BTP3 · LINK(0) = LINK(1) | 11E-J7 | 11J7 | TP3 | | TAD |
| OVERFLOW → SKIP | | | | OVERFLOW · (ISZ · E · TS3) · BTP3 = 1 → SKIP | 10H3 | 9J1 | | | ISZ |
| | | | AC | AC LOAD = BTP3 (AND · E + TAD · E + DCA · E + OPR · F) | 10H3 | 6B2 | | | AND TA DCA |
| | | | PC | PC LOAD = JMS · E · BTP3 | 10J4 | 6B2 | | | JMS |

| | | | | | | | | | |
|-----------------------|----|----|-------|-------------------------------------------------------------|------|------|-----|--|-----|
| SKIP PC→CPMA | PC | | | ENOL, EN1L, EN2L = (TS4 · F SET · MS DIS) · (FE+FD) | 11F6 | 6C8 | TS4 | | ALL |
| SKIP PC+1→ CPMA | PC | | | ENOL, EN1L, EN2L = (TS4 · F SET · MS DIS) · (FE+FD) | 11F6 | 6C8 | | | ISZ |
| | | +1 | | CARRY IN = SKIP (1) · (TS4 · F SET MS DIS) | 11H1 | 9 | | | |
| | | | CPMA | TP4 · MALC = CPMA LOAD | 10H4 | 6F3 | TP4 | | ALL |
| 1→F | | | FETCH | (F+D+FE SET + F D SET) = F SET F SET · CPMA LOAD = FL | 12B4 | 12B4 | | | |

PDP-8E MECHANIZATION TABLE
 FETCH FOR JMP AND JMPI INSTRUCTIONS

| Function | Source Req'd | Route Req'd | Dest'n Req'd | Enabling Levels & Boolean Expression | Source Page & Cord | Dest'n Page & Cord | Time | Major State | Instruction |
|-----------------------------|--------------|-------------|--------------|----------------------------------------------------------------------|--------------------|--------------------|------|-------------|---------------|
| MA+1 → PC | CPMA | +1 | PC | ENOL, EN1H, EN2H = TS1 (BF + KC) | 11H7 | 6C8 | TS1 | FETCH | ALL |
| | | | | CARRY IN L = TS1 (BF + KC) | 11H1 | 9 | | | |
| | | | | PC LOAD = TP1 (BF + KC) | 10J5 | 6B2 | | | |
| O → SKIP | | | | O → SKIP = BF · TP1 | 10H1 | 9J1 | | | |
| SA → MD | SA | | MD LINES | TIME STROBE = (STROBE · FIELD · WRITE) DELAYED | 5B8 | 5E1-8 | TS2 | | |
| | | | | MD DIR L = 100NS INTO TS1, O → MD DIR | 14C7 | 5E1-8 | | | |
| AC → DATA | AC | | | AC → BUS L = BF · TS2 | 11J3 | 6C6 | | | |
| MD → MB | MD LINES | | MB | ENOL, EN1L, EN2H = (DCA · E · TS2) · BTS2 · (MS DIS + BRK DATA CONT) | 11H5 | 6C8 | TP2 | | |
| | | | | MB load = BTP2 | 10H2 | 6C2 | | | |
| MDO-2 → IR | | | IR | LOAD IR = TP2 · F | 12E5 | 12E5 | | | |
| MD5-11 → FC | MD5-11 | PAGE | | LEFT L, RIGHT L, TWICE L = BTS3 · JMP · BF | 10H6 | 13J2 | TS3 | | |
| | | | | PAGE 2 H = MD4(0) · (BTS3 · JMP · BF) | 10H6 | 6E8 | | | |
| | | | | PAGE 2 L = MD4(1) · (BTS3 · JMP · BF) | 10H6 | 6E8 | | | |
| MD 4(0) O → PCO-4 | 0 | | | | | | | | |
| MD 4(1) MAO-4 → PCO-4 | CPMA 0-4 | | | | | | | | |
| | | | PC | PC LOAD = BTP3 · JMP | 10H5 | 6B2 | TP3 | | |
| MD 3(0) PC → MA | PC | | | ENOL, EN1L, EN2L = (TS4 · F SET · MS DIS) · (FE + FD) | 11H6 | 6C8 | TS4 | | JMP DIRECT |
| MD 3(1) MD 5-11 → MA | MD5-11 | PAGE | | RIGHT L, LEFT L, TWICE L = (BTS4 · F SET · BF · FE + FD) | 10J5 | 13J2 | | | JMP II DIRECT |
| | | | | PAGE 2 H = MD 4(0) · (BTS4 · F SET · BF · FE + FD) | 10J5 | 6F8 | | | |
| MD 4(0) O → CPMA 0-4 | 0 | | | | | | | | |
| MD 4(1) CPMA 0-4 → CPMA 0-4 | CPMA 0-4 | | | PAGE 2 L = MD4(1) · (BTS4 · F SET · BF · FE + FD) | 10J5 | 6F8 | TP4 | | |
| | | | CPMA | CPMA LOAD = TP4 · MALC | 10H4 | 6F3 | | | |
| MD 3(0) 1 → F | | | FETCH | CPMA LOAD (F · OPR + IOT · MD 3(0) · JMP = FL) | 12B7 | 12B4 | | | JMP DIRECT |
| MD 3(1) 1 → D | | | D EFER | CPMA LOAD (F · OPR + IOT · MD 3(1) = DL) | 12B7 | 12B4 | | | JMP II DIRECT |

PDF-8E MECHANIZATION TABLE
DEFER FOR JMP I INSTRUCTION

| Function | Source Req'd | Route Req'd | Dest'n Req'd | Enabling Levels & Boolean Expression | Source Page & Code | Destination Page & Code | Time | Major State |
|----------------------------------|--------------|-------------|--------------|------------------------------------------------------------------|--------------------|-------------------------|------|-------------|
| | | | | | | | TS1 | DEFER |
| | | | | | | | TP1 | |
| SA→MD | SA | | | TIME STROBE = (STROBE·FIELD·WRITE) DELAYED | 5B8 | 5E 1-8 | TS2 | |
| | | | MD LINES | MD DIR L = 100NS INTO TS1, 0→MD DIR | 14C7 | 5E 1-8 | | |
| MD+1→MB | MD | | | ENOL, EN1L, EN2H = (DCA·E·TS2)·BTS2· (MS DIS + BRK DATA CONT) | 11H5 | 6C8 | | |
| | | +1 | | CARRY IN = BD·BTS2 | 11H1 | 9 | | |
| | | | MB | MB LOAD = BTP2 | 10H2 | 6C2 | TP2 | |
| AUTO INDEX MB→MD MD DIR H | MB | | | MB DIR H = TP2 (D · AUTO INDEX) | 14C7 | 6C3 | | |
| AUTO INDEX MEM→MD MD DIR L | MEM | | | MD DIR L = TP2 (D · AUTO INDEX) | 14C7 | 5E8 | | |
| MD→PC | MD | | | ENOL, EN1L, EN2H = JMP · BTS3 · BD | 11F4 | 6C8 | TS3 | |
| | | | PC | PC LOAD = BTP3 · JMP | 10H5 | 6B2 | TP3 | |
| PC→CPMA | PC | | | ENOL, EN1L, EN2L = (TS4 · F SET · MS DIS) · (FE + FD) | 10H6 | 6C8 | TS4 | |
| | | | CPMA | CPMA LOAD = TP4 · WALC | 10H4 | 6F3 | TP4 | |
| 1→F | | | FETCH | JMP · D · CPMA LOAD = FL | 12B7 | 12B4 | | |

| | | | | | | | | |
|---------------------------------------------------------------------------------------|----|---------|-------|--------------------------------------------------------------------|------|------|-----|----------|
| MD 10(1)• MD 8(1)• MD 9(0) RTR | | RIGHT 2 | | LEFT H, RIGHT L, TWICE L = (MD10(1)• OP1•TS3)•(MD 8(0)•OP1•TS3) | 10H7 | 6F8 | | RTR |
| MD 10(1)• MD 8(0)• MD 9(1) RTL | | LEFT 2 | | LEFT L, RIGHT H, TWICE L = (MD10(1)• OP1•RS3)•(MD9(1)•OP1•TS3) | 10H7 | 6F8 | | RTL |
| MD 10(1)• MD 8(0)• MD 9(0) ByTE SWAP ACO-5→ AC6-11 AC6-11→ ACO-5 | | SWAP | | LEFT H, RIGHT H, TWICE L = (MD10(1)• OP1•TS3) | 10H7 | 6F8 | | ByTE SWA |
| DATA→AC | | | AC | AC LOAD = BTP3•(OPR•F) | 10H3 | 6B2 | TP3 | ALL |
| | | | LINK | LINK CLOCK = BTP3•OP1 | 10H8 | 10H8 | | |
| PC→CPMA | PC | | | ENO L, EN1L, EN2L = (TS4•F SET• MS DIS)•(FE + FD) | 11H6 | 6C8 | TS4 | |
| | | | CPMA | CPMA LOAD = TP4•M A L C | 10H4 | 6F3 | TP4 | |
| 1→F | | | FETCH | OPR + IOT•F•CPMA LOAD = F L | 12B7 | 12B4 | | |

PDP-8E MECH. ZATION TABLE
 FETCH FOR OPERATE GROUP TWO INSTRUCTIONS

| Function | Source Req'd | Route Req'd | Dest'n Req'd | Enabling Levels & Boolean Expression | Source Page & Code | Dest'n Page & Code | Time | Major State | Instru tion |
|---------------------|--------------|-------------|--------------|-----------------------------------------------------------------------------------------------------|--------------------|--------------------|------|-------------|--------------------------|
| MA+1→PC | CPMA | | | ENOL, EN1H, EN2H = TS1 (BF + KC) | 11H7 | 6C8 | TS1 | FETCH | ALL |
| | | +1 | | CARRY IN L = TS1 (BF + KC) | 11H1 | 9 | | | |
| | | | PC | PC LOAD = TP1 (BF + KC) | 10J5 | 6B2 | TP1 | | |
| 0→SKIP | 0 | | | 0→SKIP = BF · TP1 | 10H1 | 10J5 | | | |
| SA→MD | SA | | | TIME STROBE = (STROBE · FIELD · WRITE) DELAYED | 5B8 | 5E 1-8 | TS2 | | |
| | | | MD LINES | MD DIR L = 100 NS INTO TS1, 0→MD DIR | 14C7 | 5E 1-8 | | | |
| AC→DATA | AC | | | AC→BUS = BF · TS2 | 11J3 | 6C6 | | | |
| MD→MB | MD LINES | | | ENOL, EN1L, EN2H = (DCA · E · TS2) · BTS2 · (MS DIS + BRK DATA CONT) | 11H5 | 6C8 | | | |
| | | | MB | MB LOAD = BTP2 | 10H2 | 6C2 | TP2 | | |
| MDO-2→IR | | | IR | LOAD IR = TP2 · F | 12E5 | 12E5 | | | |
| MD 4(0) | AC | | | AC→BUS = (MD 7(1) · OPE · TS3) · (BTS3 · MD4(0) · OPR · F) · AC→BUS INH | 11H3 | 6C6 | TS3 | | NOP |
| MD5(1) · ACO(1) | 1 | | | SKIP DATA IN L = OP2 · TS3 · MD5(1) · ACO(1) · MD 8(0) | 10H2 | 10J2 | | | SMA |
| 1→SKIP | | | | | | | | | |
| MD 6(1) · AC = (∅) | 1 | | | SKIP DATA IN L = OP2 · TS3 · MD 6(1) · AC = 0 · MD 8(0) | 10H2 | 10J2 | | | SZA |
| 1→SKIP | | | | | | | | | |
| MD 7(1) · L = 1 | 1 | | | SKIP DATA IN L = OP2 · TS3 · MD 7(1) · LINK(1) · MD 8(0) | 10H2 | 10J2 | | | SNL |
| 1→SKIP | | | | | | | | | |
| MD 8(1) · SKIP→SKIP | 0 | | | SKIP DATA IN H = [OP2 · TS3 · MD 8(1) · (MD6(1) · AC=0) + (MD5(1) · AC 0(1) + (LINK(1) · MD 7(1))] | 10H2 | 10J2 | | | SPA SNA SZL SKP |
| | 1 | | | SKIP DATA IN L = OP2 · TS3 · MD 8(1) · (MD 6(1) · AC = 0 + MD 5(1) · ACO(1) + LINK(1) · MD 7(1)) | 10H2 | 10J2 | | | |
| MD 9(1) | SR | | | SR→BUS = IRO · IR1 · IR2 · MD3(1) · MD 9(1) · MD 11(0) · USER MODE · TS3 | 2 D7 | 2B 3-5 | | | OSR |
| SR→DATA | | | | | | | | | |
| MD 10(1) · C→RUN | 0 | | | RUN DATA IN L = MDO(1) · MD1(1) · MD 2(1) · MD 3(1) · MD 10(1) · MD 11(0) · USER MODE · F | 14B6 | 14D6 | | | HLT |
| DATA→AC | DATA | | | DATA T L, DATA F H = (OPR + IOT) · BTS3 · (C2L + C1L + COH) | 11H8 | 6D6 | | | ALL |

| | | | | | | | | |
|-----------------|----|----|-------|--------------------------------------------------|------|------|-----|-----|
| | | | AC | AC LOAD = OPR·F·BTP3 | 10H3 | 6B2 | TP3 | ALL |
| | | | SKIP | CLOCK SKIP = BTP3 | 10J2 | 10J2 | | ALL |
| SKIP PC→MA | PC | | | ENOL, EN1L, EN2L=(TS4·F SET·MS DIS)· (FE+FD) | 11H6 | 6C8 | TS4 | |
| SKIP PC+1→MA | PC | | | ENOL, EN1L, EN2L,=(TS4·F SET·MS DIS)· (FE+FD) | 11H6 | 6C8 | | |
| | | +1 | | CARRY IN=SKIP(1)·TS4·F SET·MS DIS) | 11H1 | 9 | | |
| | | | CPMA | CPMA LOAD = TP4·MALC | 10H4 | 6F3 | TP4 | |
| 1→F | | | FETCH | OPR+IOT·F·CPMA LOAD = FL | 12B7 | 12B4 | | |

PDP-8E MECHANIZATION TABLE
 FETCH FOR OPERATE GROUP THREE INSTRUCTIONS EXCLUDING EAE

| Function | Source Req'd | Route Req'd | Dest'n Req'd | Enabling Levels & Boolean Expression | Source Page & Code | Dest'n Page & Code | Time | Major State | Instu- tion |
|--------------------------------|--------------|-------------|--------------|---------------------------------------------------------------|--------------------|--------------------|------|-------------|-------------|
| MA+1→PC | CPMA | | | ENOL, EN1H, EN2H = TS1(BF+KC) | 11H7 | 6C8 | TS1 | FETCH | ALL |
| | | +1 | | CARRY IN L = TS1(BF+KC) | 11H1 | 9 | | | |
| | | | PC | PC LOAD = TP1(BF+KC) | 10J5 | 6B2 | TP1 | | |
| O→SKIP | 0 | | | O→SKIP = BF·TP1 | 10H1 | 10J5 | | | |
| SA→MD | SA | | | TIME STROBE=(STROBE·FIELD·WRITE) DELAYED | 5B8 | 5E1-8 | TS2 | | |
| | | | MD LINES | MD DIR L = 100NS INTO TS1, O→MD DIR | 14C7 | 5E1-8 | | | |
| AC→DATA | AC | | | AC→BUS L = BF·TS2 | 11J3 | 6C6 | | | |
| MD→MB | MD LINES | | | ENOL, EN1L, EN2H = (DCA·E·TS2)·BTS2· (MS DIS+BK DATA CONT) | 11H5 | 6C8 | | | |
| | | | MB | MB LOAD = BTP2 | 10H2 | 6C2 | TP2 | | |
| MD0-2→ IR | | | IR | LOAD IR = TP2·F | 12E5 | 12E5 | | | |
| MD 4(0)· MD 7(0) AC→DATA | AC | | | AC→BUS=(BTS3·OPR·F·MD4(0)·(OPE·TS3)· MD7(1)·AC→BUS INH | 11J3 | 6C6 | TS3 | | NOP |
| MD 5(1) MQ→DATA | MQ | | | MQ→BUS=OPE·TS3·MD5(1)·MQ→BUS INH | 11J4 | 6C6 | | | MQA |
| MD 7(1)· MD 4(0) AC→MQ | AC | | | SHL+Ld ENA=BTS3·OPR·F·MD 4(0) | 11H3 | 6B5 | | | SQL |
| MD 7(1)· MD 4(1) O→MQ | 0 | | | SHL+Ld ENA = BTS3·MD4(1)·OPR·F | 11H3 | 6B5 | | | CLA, SQL |
| DATA→AC | DATA | | | DATA T, L, DATA F H = OPR+IOT·BTS3· (C2L+C1I+COH) | 11H8 | 6D6 | | | ALL |
| | | | AC | AD LOAD = OPR·F·BTP3 | 10H3 | 6B2 | TP3 | | |
| | | | MQ | MQ LOAD = BTP3·MD 7(1)·OPE | 10H4 | 6B2 | | | SQL |
| PC→CPMA | PC | | | ENOL, EN1L, EN2L = (TS4·F SET· MS DIS)·(FE+FD) | 11H6 | 6C8 | TS4 | | ALL |
| | | | CPMA | CPMA LOAD = TP4·MALC | 10H4 | 6F3 | TP4 | | |
| 1→F | | | FETCH | OPR+IOT·F·CPMA LOAD=FL | 12B7 | 12B4 | | | |

| FUNCTION | ENO | EN1 | EN2 | DATA T | DATA F | CARRY IN | MD DIR | RIGHT | LEFT | TWICE | PAGE Z | TIME |
|-----------------------------------------------------------------------------------------------------------|-----|-----|-----|--------|--------|----------|--------|-------|------|-------|---------------------------------|------|
| MA+1 → PC | L | H | H | H | L | L | L | H | H | H | L | TS1 |
| AC → DATA MD → MB MD 0-2 → IR | L | L | H | H | L | H | L | H | H | H | L | TS2 |
| ↓ | H | H | H | H | L | H | L | H | H | H | L | TS3 |
| MD 5-11 → CPMA (0) (1) ┌──────────┴──────────┐ 0 → MA 0-4 MA → MA 0-4 | H | H | H | H | L | H | L | L | L | L | MD4L L ----- MD4L H | TS4 |

| FUNCTION | ENO | EN1 | EN2 | DATA T | DATA F | CARRY IN | MD DIR | RIGHT | LEFT | TWICE | PAGE 2 | TIM |
|-------------------------------|-----|-----|-----|-----------|-----------|-------------|-----------|-------|------|-------|-----------|-----|
| AND, TAD, ISZ, DCA, JMS PC | H | H | H | H | L | H | L | H | H | H | L | TS1 |
| AND, TAD | L | L | H | H | L | H | L | H | H | H | L | TS2 |
| ISZ | L | L | H | H | L | L | L | H | H | H | L | TS2 |
| DCA | H | H | H | L | H | H | L | H | H | H | L | TS2 |
| JMS | L | L | L | H | L | H | L | H | H | H | L | TS2 |
| AND | H | H | H | H | L | H | H | L | L | H | L | TS3 |
| TAD | L | L | H | L | H | H | H | H | H | H | L | TS3 |
| ISZ | H | H | H | H | L | H | H | H | H | H | L | TS3 |
| DCA | H | H | H | H | L | H | H | H | H | H | L | TS3 |
| JMS | L | H | H | H | L | L | H | H | H | H | L | TS3 |
| AND, TAD, ISZ, DCA, JMS | L | L | L | H | L | H | H | H | H | H | L | TS4 |

| FUNCTION | ENO | EN1 | EN2 | DATA T | DATA F | CARRY IN | MD DIR | RIGHT | LEFT | TWICE | PAGE Z | TIM |
|------------------|-----|-----|-----|-----------|-----------|-------------|--------------------|-------|------|-------|-----------|-----|
| ↓ | H | X | X | H | L | H | L | H | H | H | L | TS |
| MD + 1 → MB | L | L | H | H | L | L | L | H | H | H | L | TS |
| JMP MD → PC | L | L | H | H | L | H | L | H | H | H | L | TS |
| JMP | H | X | X | H | L | H | AUTO INDEX H | H | H | H | L | |
| MD → CPMA | L | L | H | H | L | H | L | H | H | H | L | |
| JMP PC → CPMA | L | L | L | H | L | H | AUTO INDEX H | H | H | H | L | TS |

