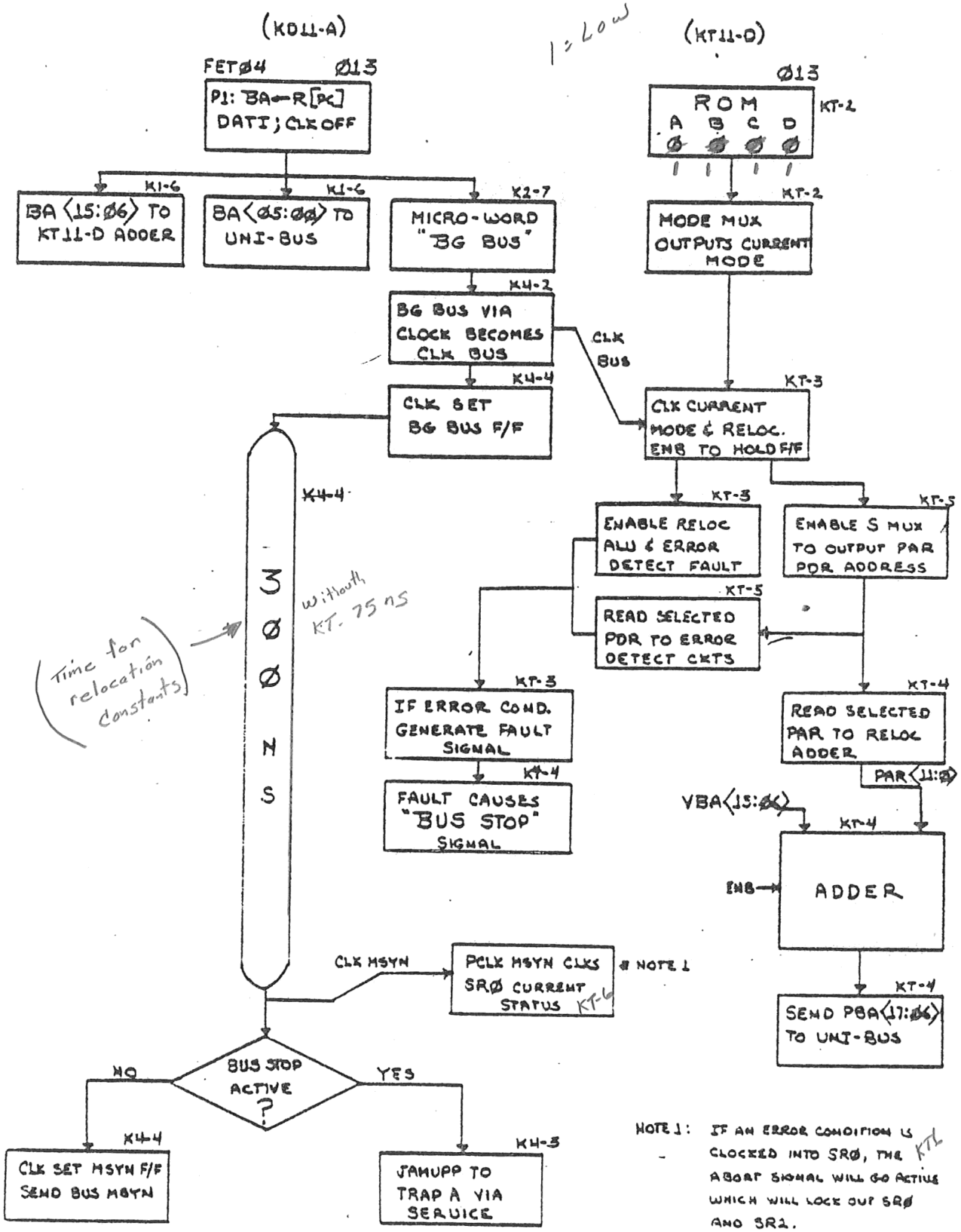
KT11-D MEMORY

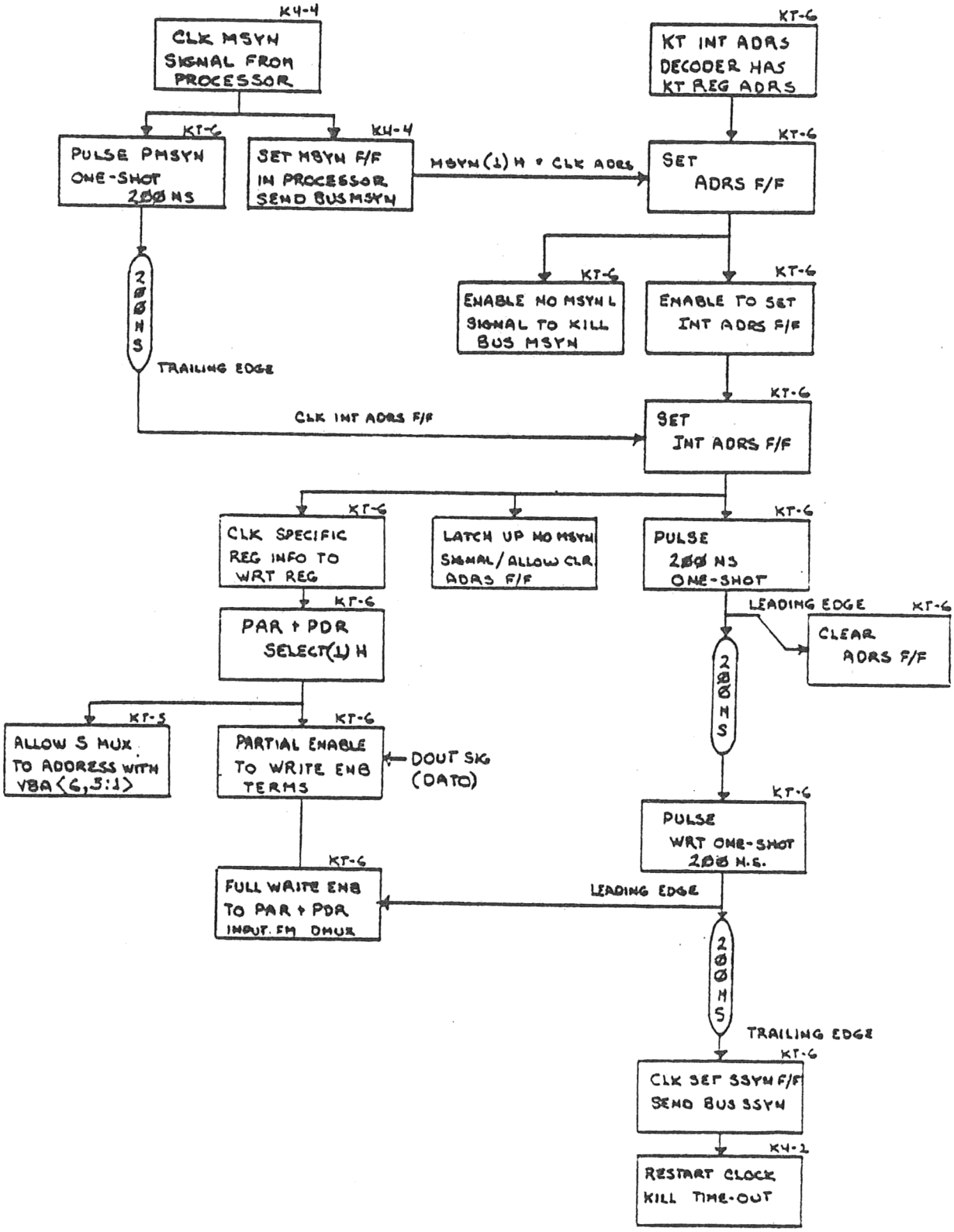
MANAGEMENT HANDOUTS

RELOCATION CYCLE FOR KT11-D

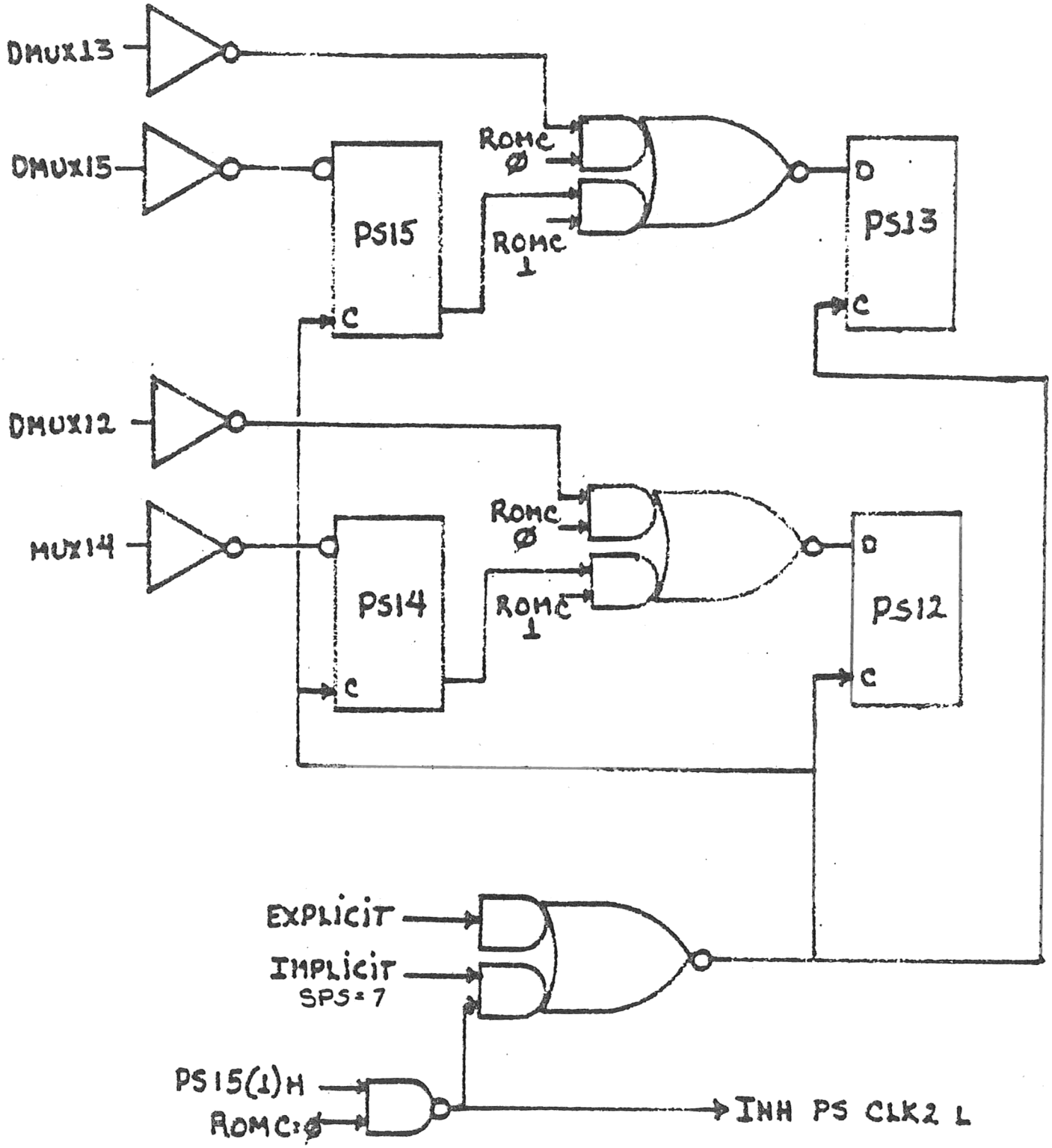


NOTE 1: IF AN ERROR CONDITION IS CLOCKED INTO SRØ, THE ABORT SIGNAL WILL GO ACTIVE WHICH WILL LOCK OUT SRØ AND SR1.

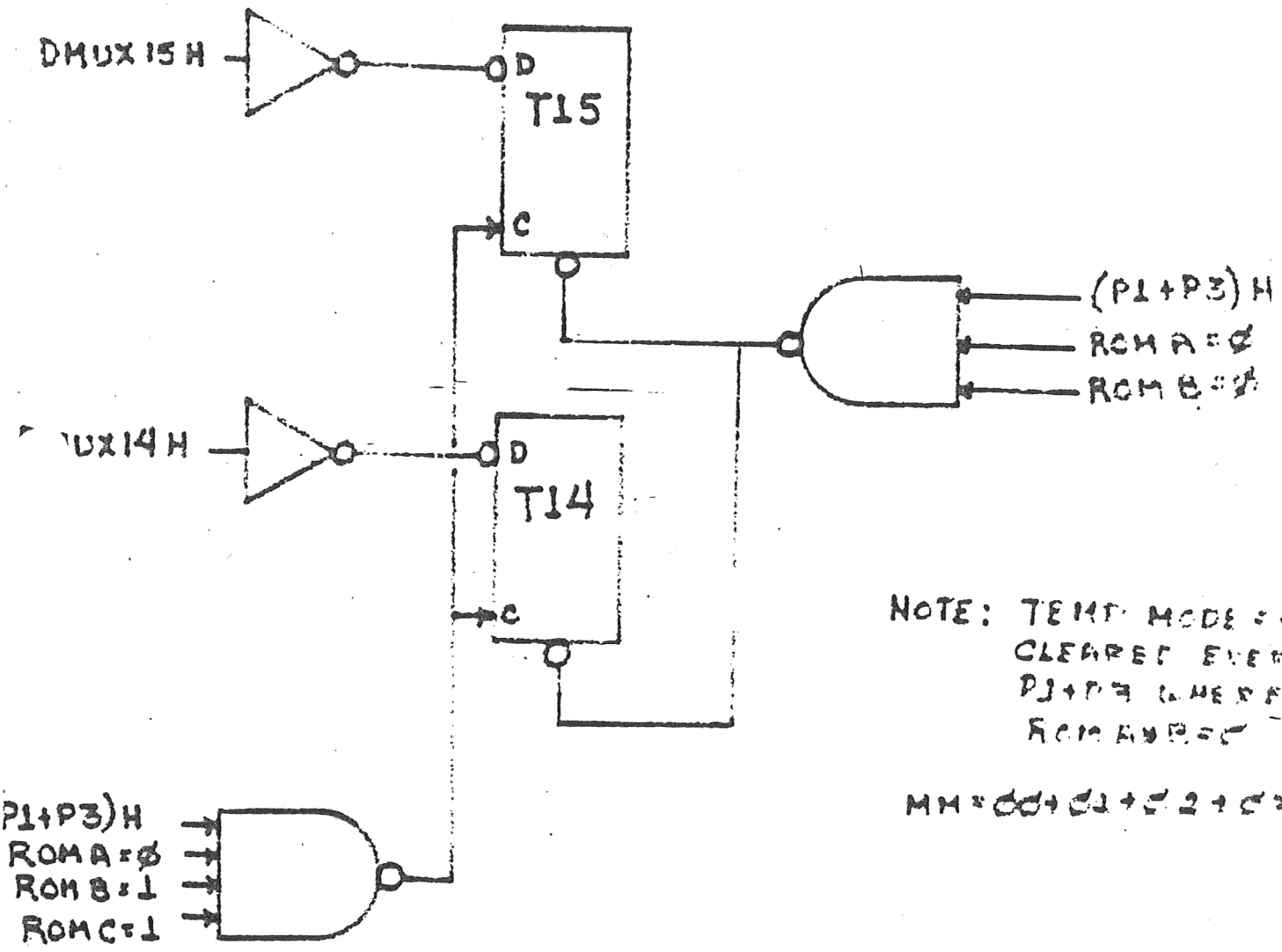
WRITING INTO KT11-D REGISTERS



NOTE: SSYN F/F & INT ADRS F/F WILL CLEAR WHEN MSYN F/F IN KDLL CLEARS


DMUX13
DMUX15
PS15
C
ROMC Ø
ROMC 1
D
PS13
C
DMUX12
MUX14
PS14
C
ROMC Ø
ROMC 1
D
PS12
C
EXPLICIT
IMPLICIT
SPS=7
PS15(1)H
ROMC:Ø
INH PS CLK2 L

PS EXT TEMP



NOTE: TEMP MODE = [illegible]
CLEARED EVERY
P1+P3 WHERE
ROM A = C

MM = ØC + C1 + C2 + C3

· TEMP F/F CLOCK EVERY
P1+P3 WHERE ROM A = C
* ROM B * C = 1

MM = ØC + C7

OUTLINE OF MM FAULT AND TRAP FLOWS:

ASSUME:

CURRENT MODE = USER(11)
PREVIOUS MODE = KERNAL(00)
A BUS CYLE IS INITIATED THAT IS GOING TO CAUSE A VIOLATION. EITHER PAGE LENGTH, NON RESIDENT OR READ ONLY.

KD11-A INITIATES BUS CYCLE:

PROCESSOR INITIATES BUS CYCLE BY GATING AN ADDRESS INTO BA REG, BRINGING UP C1/C0 AND BGBUS MICRO-BIT. BGBUS TRANSITS THRU CLOCK (K4-2) AND BECOMES CLK BUS. THIS SIGNAL CLOCKS CURRENT MODE AND RELOCATION ENABLE INTO HOLD F/F^3 (KT-3) WHICH ENABLES ERROR DETECTION CKTS. CLK BUS ALSO SETS BUS F/F(K4-4) WHICH STARTS DOWN 300 NS DELAY TOWARDS MSYN.

MM SENSES FAULT:

AS SOON AS RELOCATION ENABLE IS ACTIVATED (KT-3) THE ERROR CHECKING CKTS WOULD SENSE THE ERROR. THE FAULT LINES ON KT-3 WOULD GO ACTIVE. FAULT H GOES TO K4-4 AND ENABLES BUS STOP AND ODA ERR L. FAULT L GOES TO BUT MUY TO BE USED ON BUT 11 IN TRAP FLOWS.

KD11-A ATTEMPTS TO ACTIVATE BUS CYCLE:

WHEN BUS F/F (4-4) TRANSITS THRU 300 NS DELAY AND ACTIVATES CLK MSYN SIGNAL, THE MSYN F/F DOES NOT SET BECAUSE OF BUS STOP. INSTEAD, THE TIMER F/F (K4-3) IS SET AND A JAMUPP SEQUENCE IS INITIATED. SINCE FAULT H IS DUPLICATING AN ODA SITUATION, THE JAMUPP WILL BE TO ROM ADDRESS 002, WHICH IS SERVICE B. IN THE JAM UPP SEQUENCE, THE SIGNAL JAMUPP H IS ACTIVATED. THIS SIGNAL SETS THE BERR F/F ON K5-4. THIS F/F SERVES TWO MAJOR FUNCTIONS. FIRST IT SETS UP THE BUBC (BUT 24) BITS ON K3-7 TO GET US THRU SERVICE TO TRAP A (010). IT ALSO RECORDS THE BUS ERROR CONDITION TO PRIME THE DUBBER LINE IN CASE ANOTHER BUS ERROR IS ENCOUNTERED IN THE TRAP FLOWS.

CLK MSYN SIGNAL ALSO GOES OUT TO KT11-D ON KT-6 AND PULSES PCLKMSYN ONE SHOT. THIS SIGNAL IS USED TO CLOCK THE FAULT CONDITION INTO SR0 (KT-3) WHICH IN TURN WILL ENABLE ABORT H. ABORT H KILLS ANY FURTHER CLOCKING OF SR0 AND ALSO FREEZES SR2 [VIRTUAL PC] ON KT-8

KD11-A JAMMED TO SERVICE B:

THE PROCESSOR MAKES AN EXCURSION INTO SER11 AND IS REDIRECTED OUT TO TRAP A BY THE BUT 24 BEING SATISFIED BY BERR F/F.

KDJJ-A ENTERS TRAP FLOWS:

A VECTOR OF 4 IS GENERATED IN TRPØ3 BECAUSE FAULT H SIGNAL IS STILL DUPLICATING AN ODD ADDRESS SITUATION. (THIS IS REDUNDANT FOR A MM FAULT) THE BUT JJ IS SATISFIED BY FAULT L FROM MM FOR DIRECTION TO TRPØ5.

IN TRPØ4, THE SBC=15 CAUSE MM VECTOR (25Ø) TO BE GENERATED INTO D REGISTER

IN TRPØ5 R[VECT] GETS THE MM VECT 25Ø

BEFORE PROCEDING INTO THE PROCESSING OF THE MM FAULT A REVIEW OF KT ROM AND HANDLING OF PS EXT BITS IS IN ORDER.

AT THIS POINT, WE HAVE JUST EXECUTED TRPØ5 AND YOU WILL NOTE THAT IN THIS WORD, THE KT ROM BITS A&B=Ø AND THE WORD SPECIFIES P1. THIS TRANSLATION CLEARS THE 'T' F/F[3] ON KT-1 WHICH IN EFFECT SETS THE TEMP MODE TO KERNAL.

NOTE THAT IN THE TRAP FLOWS THE KT ROM IS ON MM=Ø4 FOR MAJORITY OF THE MICRO-WORDS. THIS MEANS THAT IF RELOCATION IS REQUIRED, THE MODE WILL BE TAKEN FROM TEMP F/F[3] (SEE MODE MOX KT-2)

IN ORDER TO UNDERSTAND THE TRAP FLOWS WITH MM ENABLED, YOU MUST REMAIN ACUTELY AWARE OF HOW THE KT ROM BITS ARE AFFECTING THE MICRO-WORD UNDER EXECUTION.

TRPØ8

A BUS CYCLE IS INITIATED IN THIS WORD AND RELOCATION IS REQUIRED. MMØ4 ALLOWS RELOCATION IN THE MODE TAKEN FROM TEMP F/F[3] (KT-2). SINCE TEMP F/F[3] ARE CLEAR, THE MODE IS KERNAL. THE ADDRESS IN BA=252 SO THE PHYSICAL BUS ADDRESS (PBA) IS DEVELOPED USING KERNALS PAR. (KPARØ) SINCE THIS WORD IS CALLING FOR THE NEW PSW AND THIS NEW PSW INCLUDES THE NEW CURRENT MODE, THE NEW CURRENT MODE IS DETERMINED BY KERNAL.

TRPØ9

THE NEW PSW IS BROUGHT IN FROM UNI-BUS VIA DMUX AND CLOCKED INTO R[TEMP](R1Ø) ON THE P1.
NOTE THAT KTROM=Ø6 IN THIS WORD. SINCE NO BUS CYCLE HAS BEEN INITIATED IN THIS WORD, THE MMØ6 IS NOT INTENDED TO DECLARE MODE FOR RELOCATION. THE MMØ6 BITS IN THIS WORD ARE TO CLK TEMP F/F[3] KT-2 (ROM A=Ø * B=1 * C=L * P1) ALLOW CLOCKING DMUX(15:14) INTO TEMP F/F[3]

TRPØ9 (CONT)

IF THE NEW PSW COMING OUT OF KERNAL SPACE xxx252 SPECIFIES THAT NEW CURRENT MODE IS TO BE USER (11) THE TEMP F/F'S WILL BE SET AND THE REST OF THE TRAP FLOWS WILL BE HANDLED BY USER. IF NEW CURRENT MODE IS KERNAL (ØØ), THE TEMP F/F WILL REFLECT THIS AND CONTROL WILL STAY WITH KERNAL. SINCE THE NEW MODE HAS COME OUT OF KERNAL SPACE, IT IS KERNAL THAT HAS MADE DECISION WETHER USER WILL REGAIN CONTROL AT THIS POINT.

TRPIØ

THE STACK POINTER IS AUTO-DECREMENTED IN THIS WORD IN PREPARATION FOR PUSHING THE OLD PSW ONTO THE STACK
NO BUS CYCLE IN THIS WORD SO MMØV NOT USED FOR RELOCATION.
MMØV DOES ALLOW MODE MUX (KT-2) TO OUTPUT THE TEMP F/F'S AND IN THIS CASE THE MODE 1 SELECT LINE IS USED TO DETERMINE WHICH STACK POINTER TO USE [R6 OR R16] IF THE TEMP F/F'S IN TRPØ9 WENT TO USER MODE A HOOK WOULD BE ACTIVE IN GPR ADRS SWITCH KJ-7 AND R16 WOULD BE ADDRESSED. IF TEMP STAYED IN KERNAL MODE SELECT WOULD NOT HOOK R16 IN THIS WORD AND KERNAL SP WOULD BE USED.

TRPII

THE SPS=6 BITS IN THIS WORD ACTIVATES BUS RD FM PS H SIGNAL (K5-2) THIS GATES KD PSW ONTO RD BUS (K5-2) AND ALSO GATE KT PSW EXT BITS ONTO RD BUS (KT-2). THE FULL PSW TRANSITS THRU ALU AND IS CLOCKED INTO D REG ON P2:

SINCE THERE IS A BUS CYCLE INITIATED IN THIS WORD, THE MMØV

BITS A USED TO SELECT THE RELOCATION MODE FOR THE ADDRESS PUT IN BA REG IN TRPIØ. MMØV TAKES MODE FROM TEMP F/F'S AND WILL DETERMINE IF OLD PSW IS PUSHED ON USER OR KERNAL STACK.

TRPI2

THIS IS A DESKEW NO OP FOR DATO IN TRPII (DO NOT DISTURB D REG) MMØV IS USED IN THIS WORD SIMPLY TO KEEP FROM DISTURBING THE TEMP F/F'S (MMØV DOES NOT CLOCK OR CLEAR TEMP F/F'S)

TRPI3

HERE AGAIN THE SP IS DECREMENTED IN PREPARTION OF PUSHING OLD PC ONTO STACK.

MMØV IS USED TO ALLOW HOOK FOR USER'S STACK POINTER IF TEMP =(11) USER MODE

TRPI4 R[PC] IS PUT INTO D REG AND A DATO BUS CYCLE IS INITIATED MMØV SELECTS RELOCATION MODE FROM TEMP F/F AND PC WILL BE PUSHED ON USER OR KERNAL STACK.

TRP15

NOTE THAT NO DESKEWING IS NECESSARY FOR THE DATO INITIATED IN TRP14. THIS IS BECAUSE THIS WORD DOES NOT EFFECT THE CONTENTS OF THE D REG.

THE PURPOSE OF THIS WORD IS NOT TO ACTUALLY PUT THE CONTENTS OF R[TEMP] = NEW PSW INTO THE PSW. THE SPS=Ø IN THIS WORD DOES NOT ALLOW CLOCKING PSW. IT IS THE INTENT SIMPLY TO READ OUT NEW PSW FROM R[TEMP], ALLOW IT TO TRANSIT FROM RD BUS THRU DMUX AND SETTLE DOWN BEFORE ACTUALLY CLOCKING IT IN ON THE NEXT MICRO-WORD MMØ4 IN THIS WORD LEAVES TEMP UNDISTURBED.

TRP16

THIS WORD CONTINUES THE GATING OF R[TEMP] ON THE DMUX AND BRINGS UP SPS=7
SPS=7 & PL CLOCKS DMUX INTO KD PSW (K5-2)

INH PS CLR 2 CANNOT GO ACTIVE BECAUSE ROMC=1 IN THIS WORD SO PS⟨15:14⟩ ← DMUX⟨15:14⟩, PS⟨13:12⟩ ← PS⟨15:14⟩ (KT-2) THIS TRANSFERS NEW CURRENT MODE INTO PSW AND TRANSFERS CURRENT MODE TO PREVIOUS MODE.

MMØ2 ALSO CLEARS TEMP F/F'S WITH (PJ & ROM A&B=Ø) THIS SETS TEMP UNCONDITIONALLY BACK TO KERNAL FOR USE IN TRAP 2Ø. (NO RELOC OR MODE REQUIRED IN THIS WORD)

TRP2Ø

A BUS CYCLE IS INITIATED IN THIS WORD TO GET A NEW PC FROM xxx25Ø.
MMØ4 TAKES RELOCATION FROM TEMP F/F'S WHICH ARE ØØ KERNAL. PBA = 25Ø + KPARØ AND NEW PC COMES FROM KERNAL SPACE, JUST AS THE NEW PSW DID.

TRP21

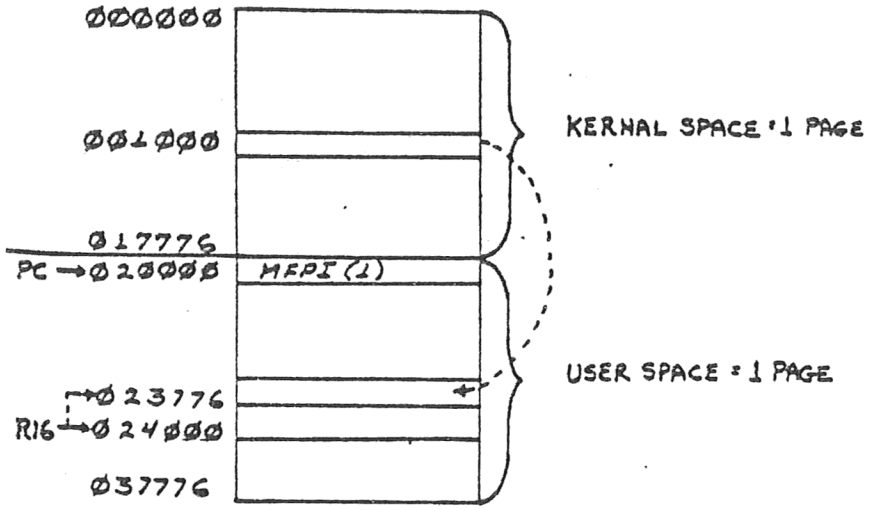
THE NEW PC FROM KERNAL SPACE IS GATED INTO PC

NOTE: FROM HERE, A PASS THRU SERVICE TO FETCH WHERE MM=ØØ AND RELOCATION WILL BE DETERMINED BY THE CURRENT MODE GATED IN ON TRP16.

ASSUME THAT THE FOLLOWING CONDITIONS EXIST WITHIN KDII-A:

R1 = 001000
R6 = 002000
R16 = 004000
R7 = 000000
KPAR0 = 0000
KPDR0 = 1 PAGE/UP/READ-WRITE
UPAR0 = 0200
UPDR0 = 1 PAGE/UP/READ-WRITE
PSW = 14xxxx CURRENT MODE = USER / PREVIOUS MODE = KERNAL

ALLOTED SPACE AS SHOWN BELOW (ADDRESSES ARE PHYSICAL)

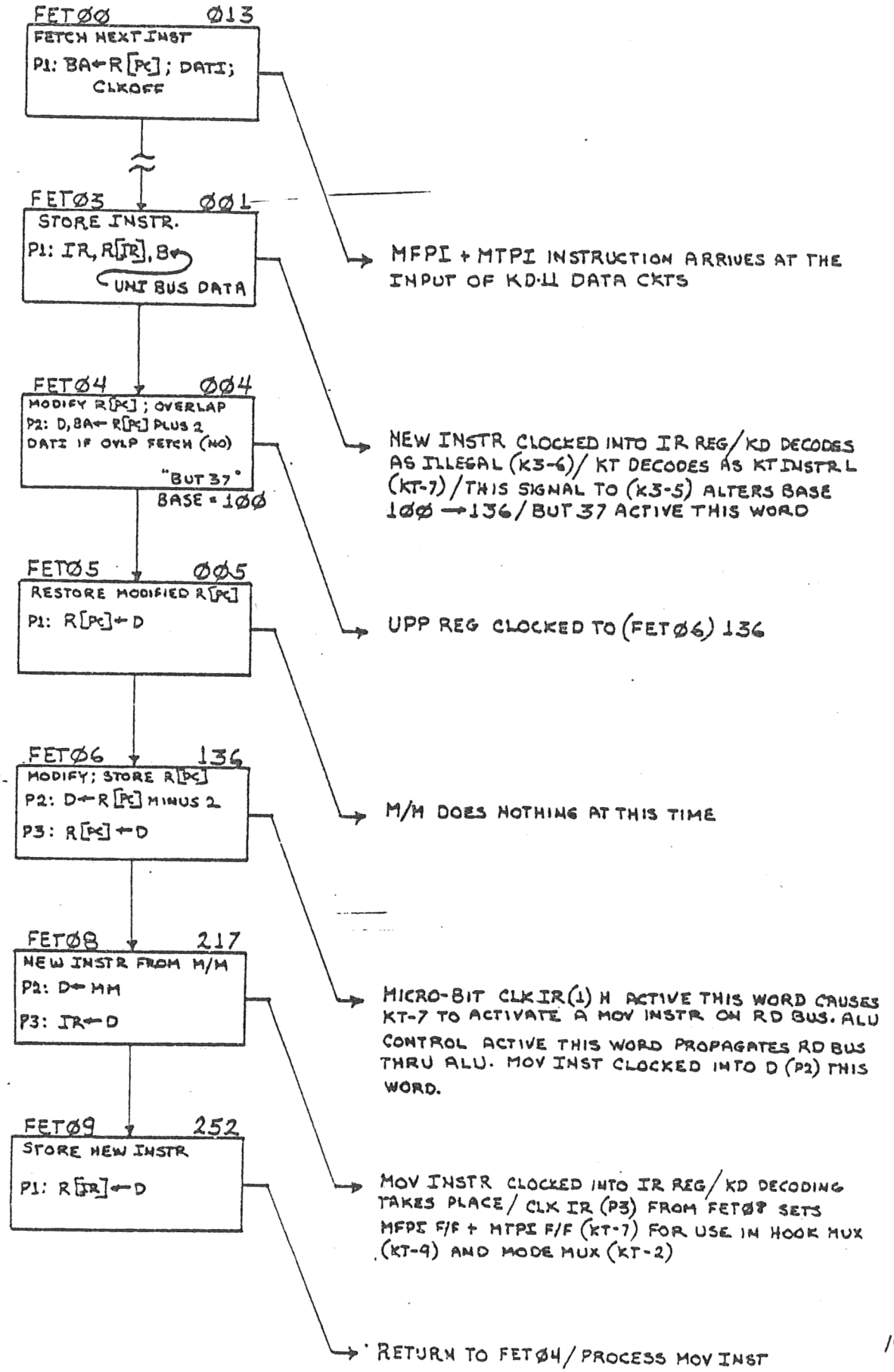


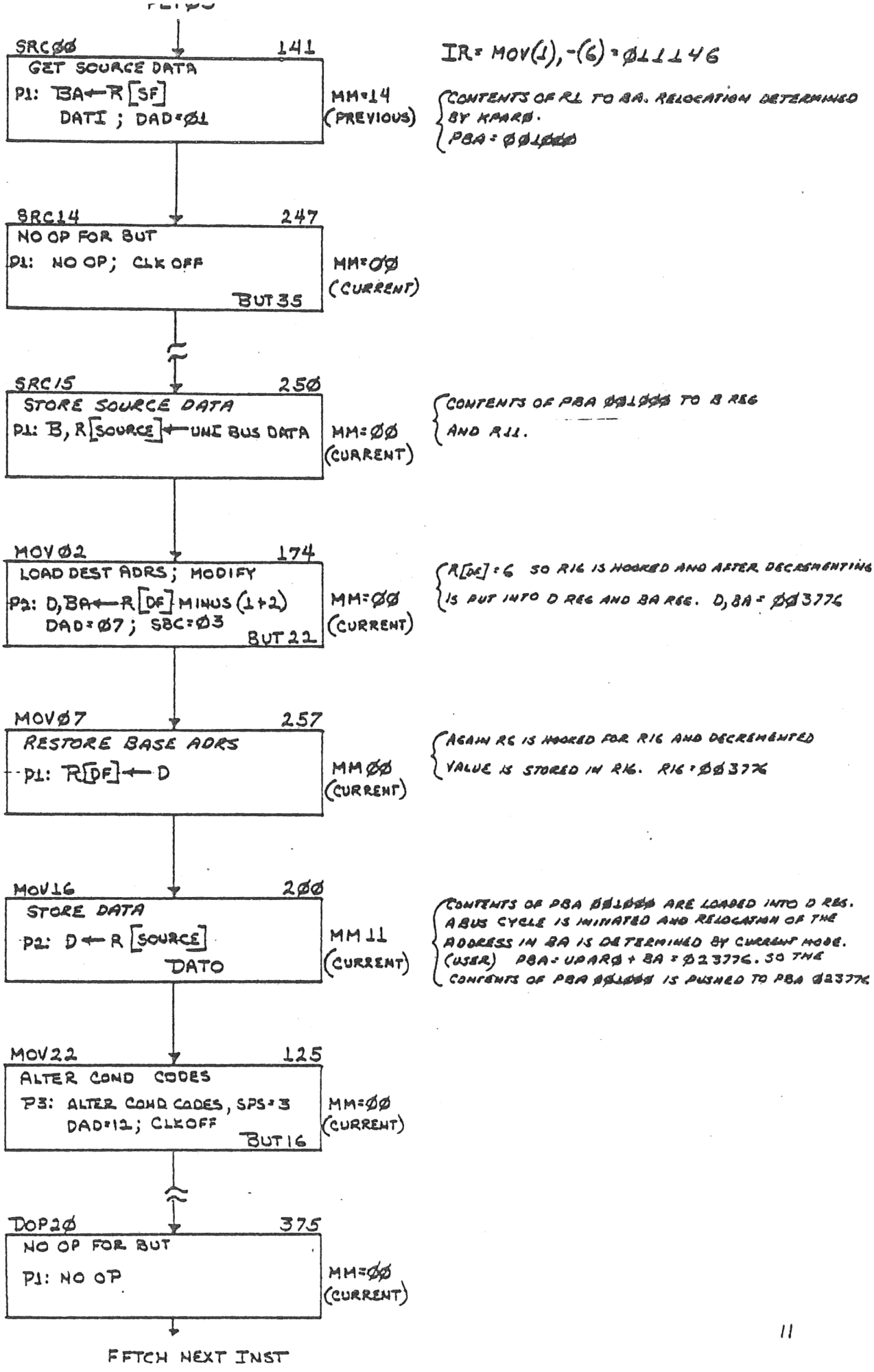
WHEN THE USER FETCHES AND EXECUTES THE MFPI (1) INSTRUCTION AT PHYSICAL LOCATION 020000 THE RESULT WILL BE TO MOVE THE WORD FROM LOCATION 001000 (KERNAL SPACE) AND PUSH IT INTO LOCATION 023776 (USER SPACE) USING R16 (USER SP) IN THE AUTO-DECREMENT MODE.

THE MFPI(1) INSTRUCTION WILL BE CHANGED TO MOV (1), -(6) DURING KT FETCH RECYCLE. THEN THE MOV INSTR WILL BE EXECUTED. DURING EXECUTION OF MOV INST THE ADDRESS OF THE SOURCE OPERAND WILL BE DETERMINED BY THE PREVIOUS MODE (KERNAL) AND THE ADDRESS OF THE DESTINATION WILL BE DETERMINED USING THE CURRENT MODE (USER). ALSO, A DESTINATION HOOK WILL BE ACTIVE SO THAT R16 WILL BE USED INSTEAD OF R6 FOR DESTINATION ADDRESS.

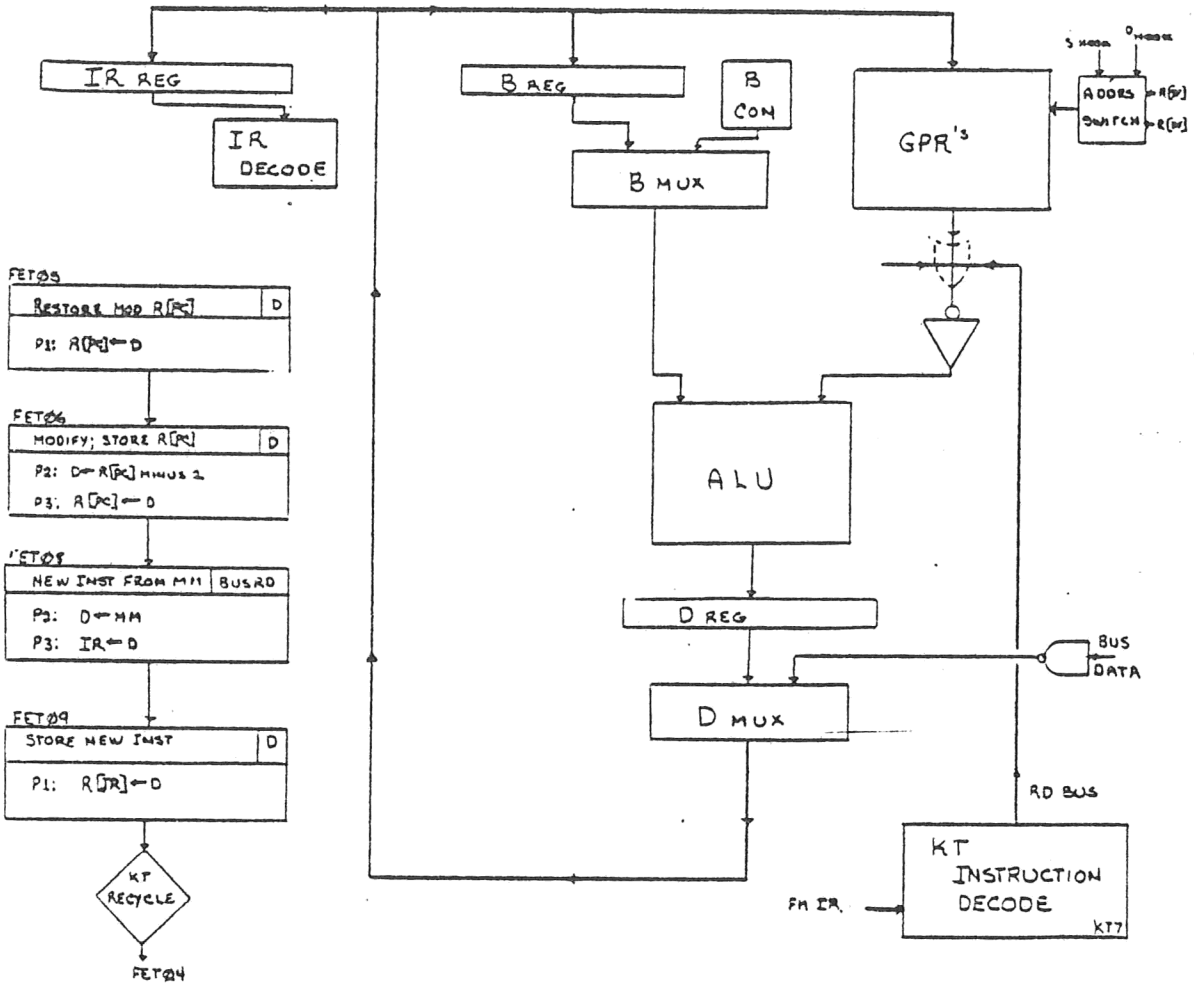
KT FETCH RECYCLE

MFPI OR MTPI



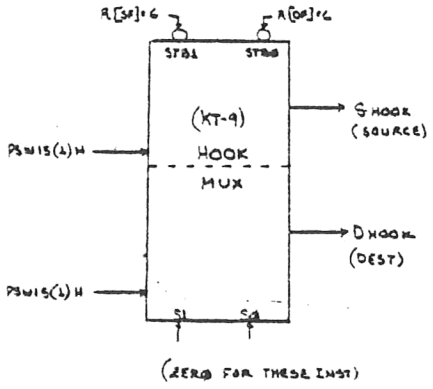

SRCØØ
141
GET SOURCE DATA
P1: BA←R[SF]
DATI ; DAD=Ø1
MM=14
(PREVIOUS)
IR= MOV(I),-(6) =ØIIIØ6
CONTENTS OF RL TO BA. RELOCATION DETERMINED BY KMARØ.
PBA = ØØIØØØ
SRC14
247
NO OP FOR BUT
P1: NO OP; CLK OFF
BUT35
MM=ØØ
(CURRENT)
SRC15
25Ø
STORE SOURCE DATA
P1: B, R[SOURCE]←UNI BUS DATA
MM=ØØ
(CURRENT)
CONTENTS OF PBA ØØIØØØ TO B REG AND RII.
MOVØ2
174
LOAD DEST ADRS; MODIFY
P2: D,BA←R[DF] MINUS (1+2)
DAD=Ø7; SBC=Ø3
BUT22
MM=ØØ
(CURRENT)
R[DE]=6 SO RI6 IS HOOKED AND AFTER DECREMENTING IS PUT INTO D REG AND BA REG. D,BA = ØØ377C
MOVØ7
257
RESTORE BASE ADRS
P1: R[DF]←D
MM ØØ
(CURRENT)
AGAIN R6 IS HOOKED FOR R16 AND DECREMENTED VALUE IS STORED IN R16. R16=ØØ37C
MOV16
2ØØ
STORE DATA
P2: D←R[SOURCE]
DATO
MM 11
(CURRENT)
CONTENTS OF PBA ØØIØØØ ARE LOADED INTO D REG. A BUS CYCLE IS INITATED AND RELOCATION OF THE ADDRESS IN BA IS DETERMINED BY CURRENT MODE. (USER) PBA=UBARØ + BA = Ø2377C. SO THE CONTENTS OF PBA ØØIØØØ IS PUSHED TO PBA Ø2377C
MOV22
125
ALTER COND CODES
P3: ALTER COND CODES, SPS=3
DAD=12; CLKOFF
BUT16
MM=ØØ
(CURRENT)
DOP2Ø
375
NO OP FOR BUT
P1: NO OP
MM=ØØ
(CURRENT)
FFTCH NEXT INST

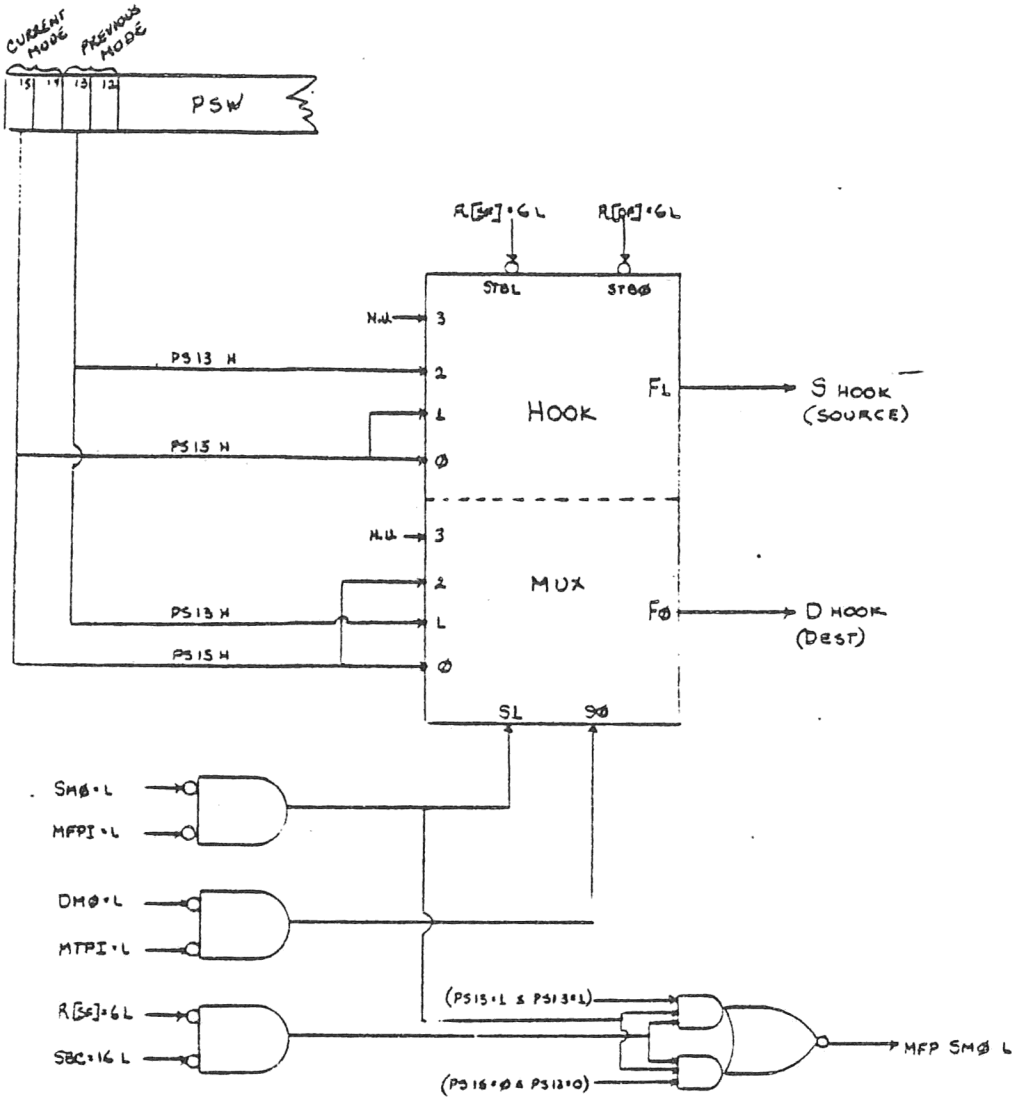
"MEMORY MANAGEMENT"



FETCHED INST	REVISED INST	PSW CURRENT	PSW PREVIOUS	D HOOK	S HOOK	INST EXECUTED	COMMENT
MTPI (1) 006611	MOV(6)+,(1) 012611	00	00	L	L	MOV (6)+, (1)	KERNAL STACK → KERNAL SPACE
		11	11	L	H	MOV(16)+, (1)	USER STACK → USER SPACE
		00	11	L	L	MOV (6)+, (1)	KERNAL STACK → USER SPACE
		11	00	L	H	MOV(16)+, (1)	USER STACK → KERNAL SPACE

MFPI (1) 006511	MOV(1),-(6) 011146	00	00	L	L	MOV(1),-(6)	KERNAL SPACE → KERNAL STACK
		11	11	H	L	MOV(1),-(16)	USER SPACE → USER STACK
		00	11	L	L	MOV(1),-(6)	USER SPACE → KERNAL STACK
		11	00	H	L	MOV(1),-(16)	KERNAL SPACE → USER STACK





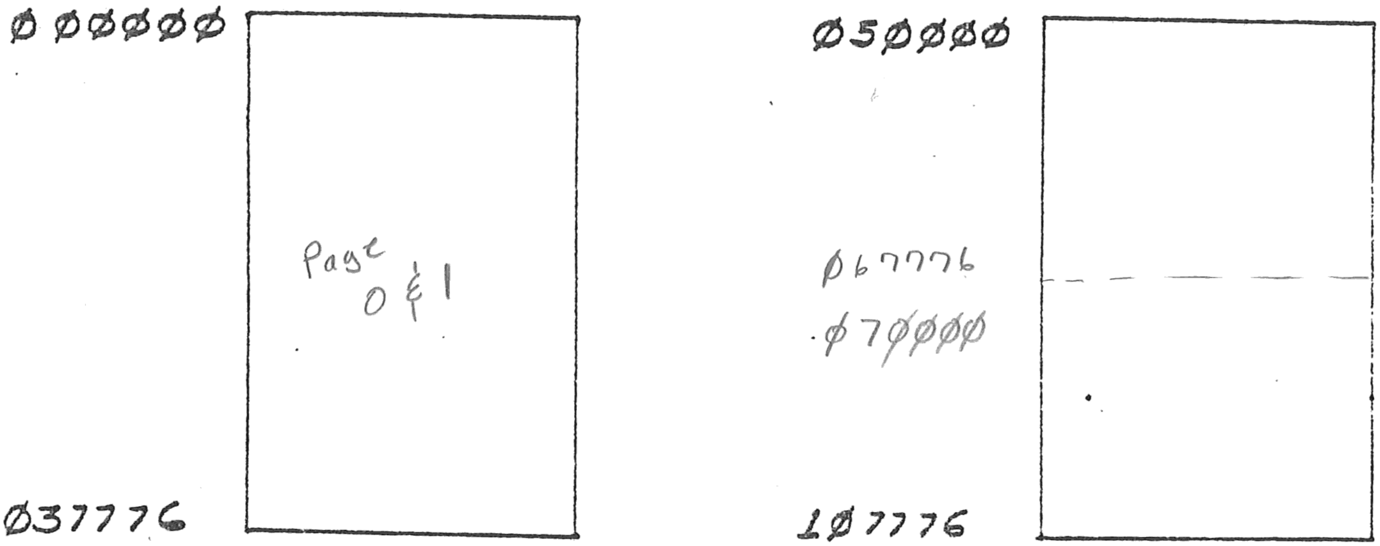
FETCHED INST	REVISED INST	PSW CURRENT	PSW PREVIOUS	D HOOK	S HOOK	MFP SMØL	INST EXECUTED	COMMENTS
MFPI%06 ØØ65Ø6	MOV%6,-(6) Ø1Ø646	Ø Ø	Ø Ø	L	L	L *	MOV%6,-(6)	KERNAL SP → KERNAL STACK
		1 1	1 1	H	H	L *	MOV%16,-(16)	USER SP → USER STACK
		Ø Ø	1 1	L	H	H	MOV%16,-(6)	USER SP → KERNAL STACK
		1 1	Ø Ø	H	L	H	MOV%6,-(16)	KERNAL SP → USER STACK
MTPI%06 ØØ66Ø6	MOV(6)+,%6 Ø126Ø6	Ø Ø	Ø Ø	L	L	H	MOV(6)+,%6	KERNAL STACK → KERNAL SP
		1 1	1 1	H	H	H	MOV(16)+,%16	USER STACK → USER SP
		Ø Ø	1 1	H	L	H	MOV(6)+,%16	KERNAL STACK → USER SP
		1 1	Ø Ø	L	H	H	MOV(16)+,%6	USER STACK → KERNAL SP

* MFP SMØ L INHIBITS DECREMENTATION OF STACK POINTER PRIOR TO DATI

1 QUIZ ON RELOCATION

CONSIDER THAT KERNAL IS BRINGING IN A USER PROGRAM AND WANTS TO INSTALL THIS PROGRAM AS SHOWN BELOW. SHOW HOW AND WHAT KERNAL WOULD SET UP UPAR^S AND UPDR^S BEFORE GIVING OVER EXECUTION CONTROL TO USER GIVE USER FULL ACCESS TO HIS ALLOTTED PHYSICAL SPACE.

USER VIRTUAL — USER PHYSICAL



077776

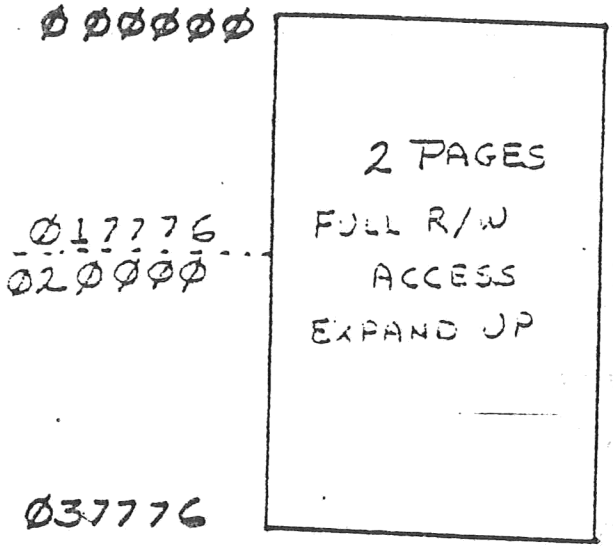
50

UPARØ = 5ØØ	UPDRØ = Ø774Ø6
UPAR1 = 7ØØ	UPDR1 = 000016
UPAR2 =	UPDR2 = 0
UPAR3 =	UPDR3 =
UPAR4 =	UPDR4 =
UPAR5 =	UPDR5 =
UPAR6 =	UPDR6 =
UPAR7 =	UPDR7 =

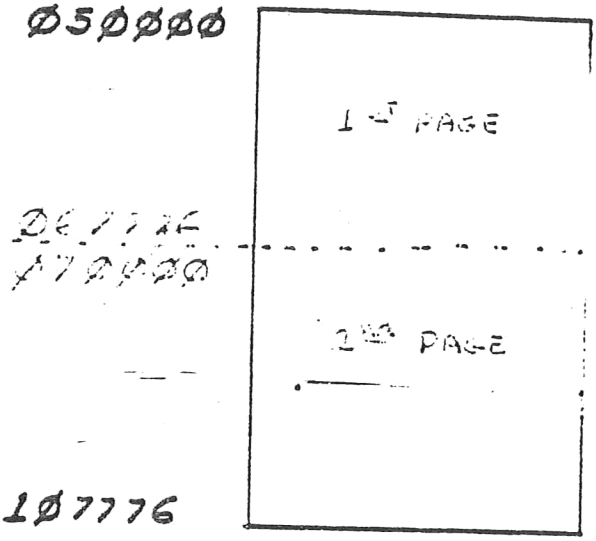
#1 QUIZ ON RELOCATION

CONSIDER THAT KERNAL IS BRINGING IN A USER PROGRAM AND WANTS TO INSTALL THIS PROGRAM AS SHOWN BELOW SHOW HOW AND WHAT KERNAL WOULD SET UP UPAR'S AND UPDR'S BEFORE GIVING OVER EXECUTION CONTROL TO USER GIVE USER FULL ACCESS TO HIS ALLOTTED PHYSICAL SPACE.

USER VIRTUAL



USER PHYSICAL



UPAR0 = 0500

UPAR1 = 0700

UPAR2 =

UPAR3 =

UPAR4 =

UPAR5 =

UPAR6 =

UPAR7 =

NOT USED

UPDR 0 = 077776

UPDR 1 = 077772

UPDR 2 = NOT RELOCATED

UPDR 3 = NOT RELOCATED

UPDR 4 = NOT RELOCATED

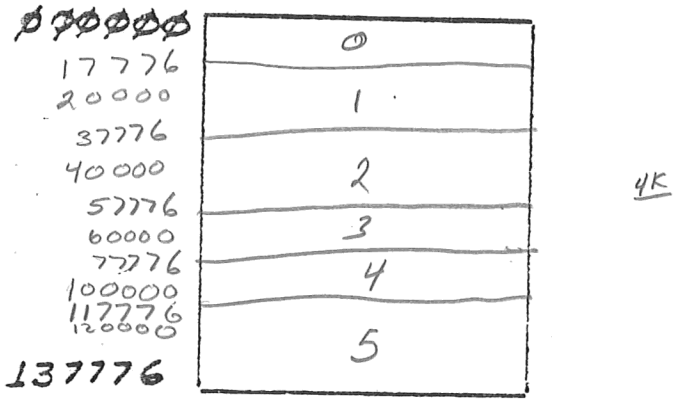
UPDR 5 = NOT RELOCATED

UPDR 6 = NOT RELOCATED

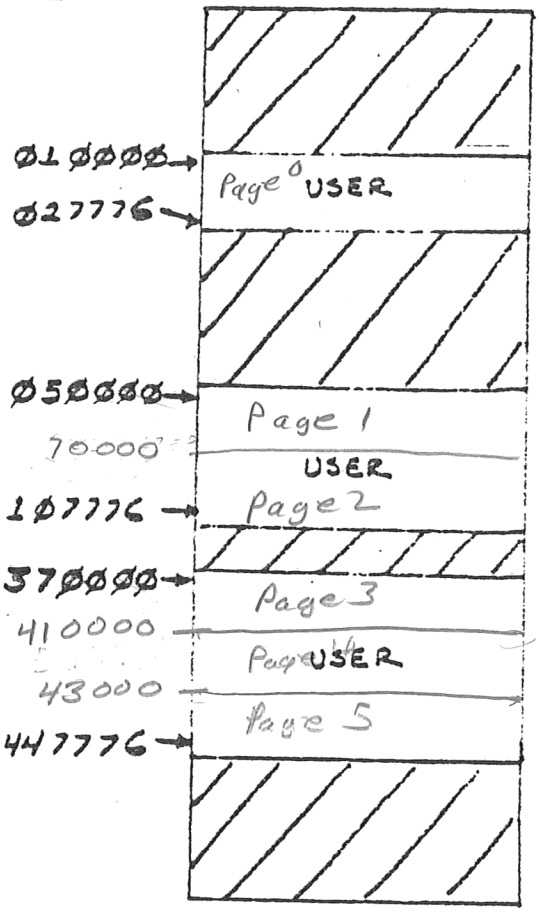
UPDR 7 = NOT RELOCATED

KERNAL IS BRINGING IN THE VIRTUAL USER PROGRAM SHOWN BELOW. LIMITED PHYSICAL SPACE IS AVAILABLE AS SHOWN ON PHYSICAL ADDRESS MAP BELOW. SET UP UPAR^S AND UPDR^S TO ALLOW THIS USER TO UTILIZE EXISTING PHYSKAL SPACE.

USER VIRTUAL



USER PHYSICAL



17776
17776

1070

UPAR0 = 100
UPAR1 = 500
UPAR2 = 700
UPAR3 = 3700
UPAR4 = 4100
UPAR5 = 4300
UPAR6 = { Not used
UPAR7 = {

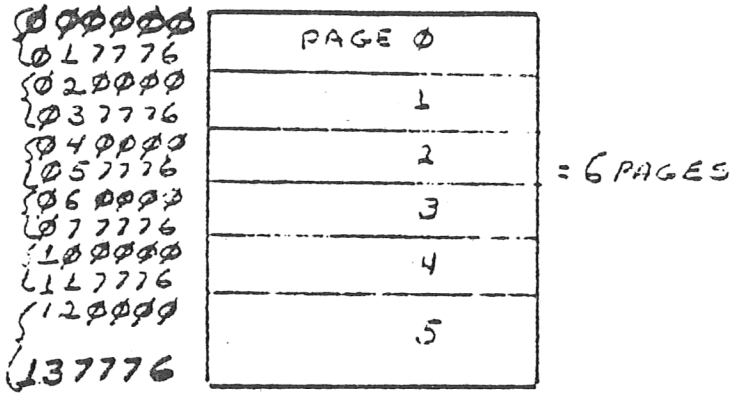
UPDR0 = 16
UPDR1 = 16
UPDR2 = 16
UPDR3 = 16
UPDR4 = 16
UPDR5 = 16
UPDR6 = 0
UPDR7 = 0

NOTE: MAP OUT UPDR^S NOT TO BE USED WITH TERM: NOH RESIDENT.

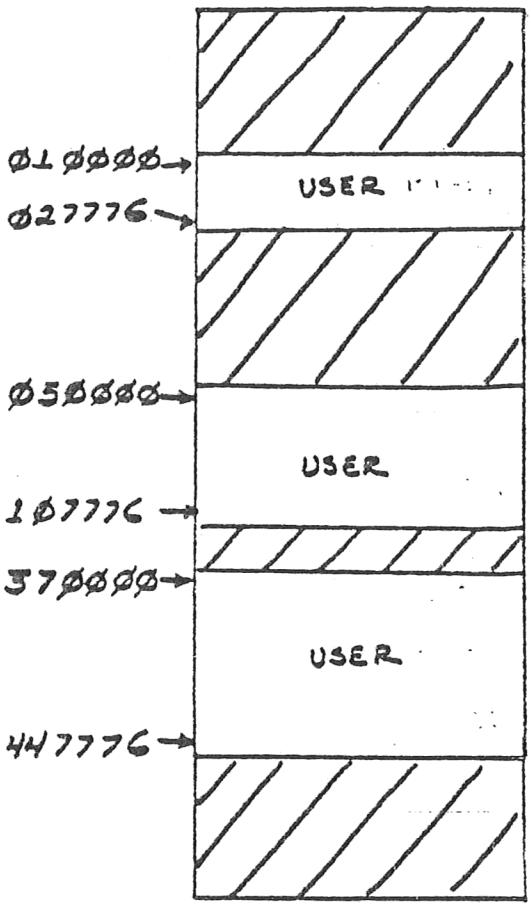
#2 QUIZ ON RELOCATION

KERNAL IS BRINGING IN THE VIRTUAL USER PROGRAM SHOWN BELOW. LIMITED PHYSICAL SPACE IS AVAILABLE AS SHOWN ON PHYSICAL ADDRESS MAP BELOW. SET UP UPAR'S AND UPDR'S TO ALLOW THIS USER TO UTILIZE EXISTING PHYSICAL SPACE.

USER VIRTUAL



USER PHYSICAL



UPAR0 = 0100
UPAR1 = 0500
UPAR2 = 0700
UPAR3 = 3700
UPAR4 = 4100
UPAR5 = 4300
UPAR6 = } NOT USED
UPAR7 = }

UPDR0 = 077406
UPDR1 = 077406
UPDR2 = 077406
UPDR3 = 077406
UPDR4 = 077406
UPDR5 = 077406
UPDR6 = NON-RESIDENT
UPDR7 = NON-RESIDENT

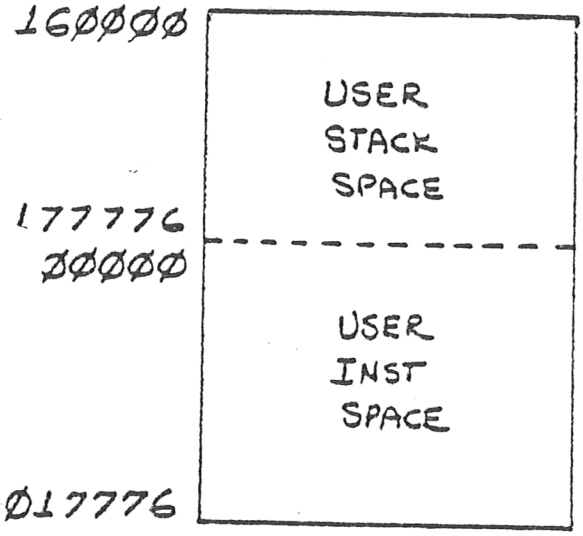
000016
000016 indicates
000016 no upwards
000016 expansion
000016
000016
0
0

NOTE: MAP OUT UPDR'S NOT TO BE USED WITH TERM: NOH RESIDENT.

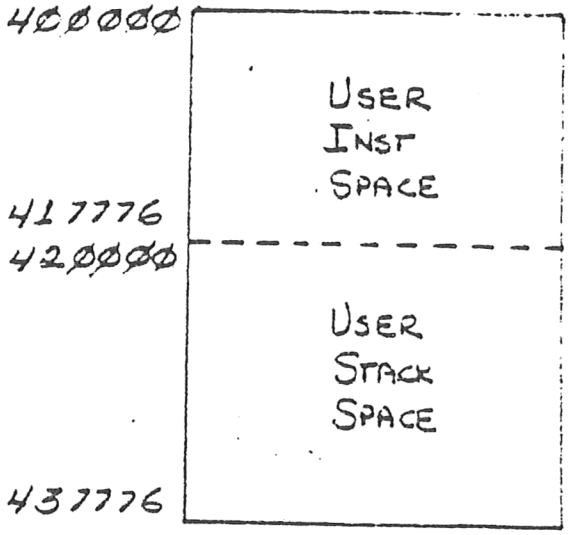
#3 Quiz on Relocation

Consider kernal is bringing in a user's program shown below. Set up the UPAR's and UPDR's to execute and control the physical program.

User Virtual



User Physical

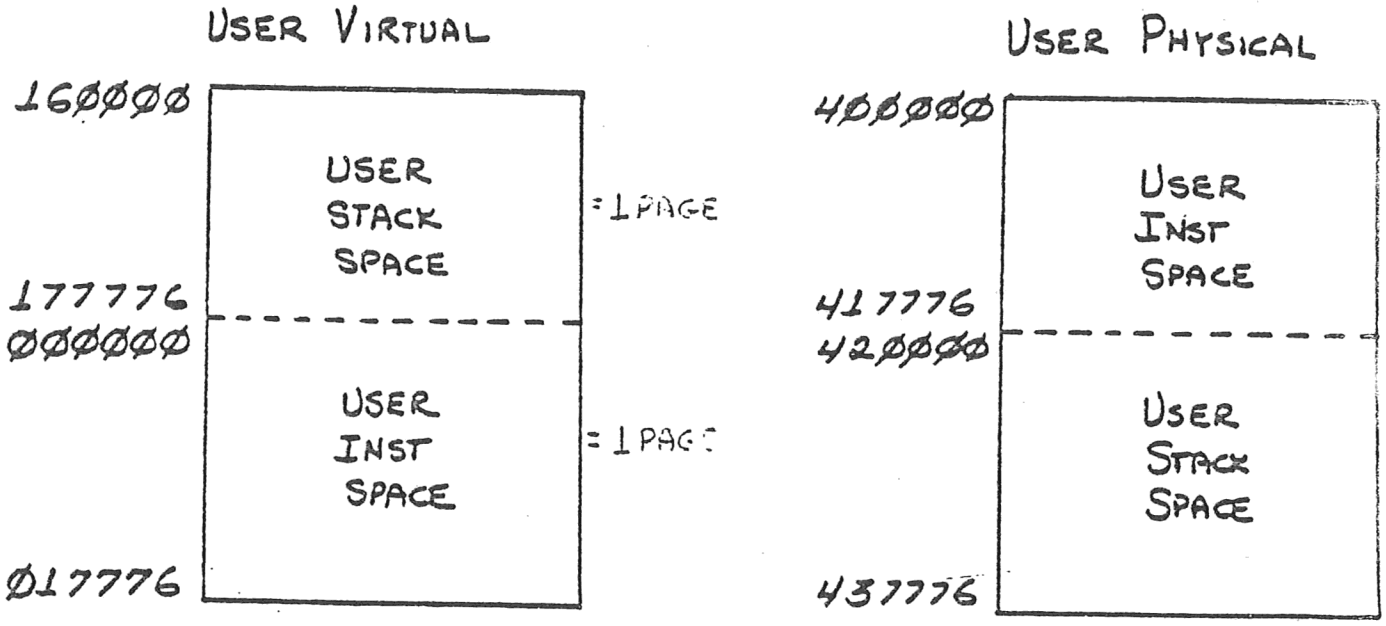


UPAR0 =
UPAR1 =
UPAR2 =
UPAR3 =
UPAR4 =
UPAR5 =
UPAR6 =
UPAR7 =

UPDR0 =
UPDR1 =
UPDR2 =
UPDR3 =
UPDR4 =
UPDR5 =
UPDR6 =
UPDR7 =

#3 Quiz on Relocation

Consider kernal is bringing in a user's program shown below. Set up the UPAR's and UPDR's to execute and control the physical program.



UPAR0 = 4000
UPAR1 =
UPAR2 =
UPAR3 =
UPAR4 =
UPAR5 =
UPAR6 =
UPAR7 = 4200

NOT USED

1 page/R1w/Exp 12

UPDR0 = 077406
UPDR1 = NON-RESIDENT
UPDR2 = [illegible]
UPDR3 = [illegible]
UPDR4 = [illegible]
UPDR5 = [illegible]
UPDR6 = NON-RESIDENT
UPDR7 = 000016

1 page, R/W, Exp 1

KT11-D EXAM

1. A PAGE DESCRIPTOR REGISTER CONTAINS Ø72414 WHICH OF THE BELOW VIRTUAL ADDRESSES WILL CAUSE A VIOLATION?

   A. Ø155ØØ
   B. Ø165ØØ
   C. Ø175ØØ
   D. Ø2ØØØØ

2. MEMORY MANAGEMENT FAULTS IN USER MODE TRAP TO LOCATION 25Ø IN USER SPACE RATHER THE KERNAL SPACE. THE PROBLEM COULD BE CHIP:

   A. E69
   B. E86
   C. E59
   D. E54

3. THE KT11-D IS INSTALLED PROPERLY. RELOCATION IS ENABLED WHEN THE INSTRUCTION SHOWN BELOW IS EXECUTED. THE PROGRAM WILL JUMP TO PHYSICAL LOC_____

   A. 12316
   B. 13316
   C. 53ØØ
   D. 44ØØ

   6ØØØ - JMP @ - (3)
   R3: 15232Ø
   777776 = Ø3ØØØ2
   UPARØ = Ø
   UPAR6 = 2Ø
   KPARØ = Ø
   KPAR6 = 1Ø
   12316 = 53ØØ
   13316 = 44ØØ

4. WHICH OF THE KT ROM SIGNALS LISTED BELOW ENABLES RELOCATING ONLY THE DESTINATION ADDRESSES WHEN IN THE MAINTENENCE MODE?

A. ROM A
B. ROM B
C. ROM C
D. ROM D

5. IT IS DETERMINED THAT PAGE LENGTH VIOLATIONS DO NOT CAUSE A TRAP TO 25Ø. ALL OTHER TYPES OF VIOLATIONS DO TRAP PROPERLY. THE PROBLEM IS MOST LIKELY CHIP:

A. E61 PIN 6
B. E62 PIN 12
C. E73 PIN 12
D. F3Ø PIN 5

6. RELOCATION IS DISABLED AND THE INSTRUCTION SHOWN BELOW CAUSES A BUS TIMEOUT TRAP. THE PROBLEM IS MOST LIKELY CHIP:

A. E97 PIN 3
B. E88 PIN 8
C. E95 PIN 4
D. E8Ø PIN 6

5ØØ - MOV #2Ø, @#1723ØØ
5Ø2 - 2Ø
5Ø4 - 1723ØØ

7. EXECUTION OF A RESET INSTRUCTION IN USER MODE CAUSES THE KDJ1-A TO HALT. THE PROBLEM IS MOST LIKELY CHIP: (RELOCATION IS ENABLED)

A. E1Ø1 PIN 8
B. E1Ø1 PIN 6
C. E1ØØ PIN 4
D. E98 PIN 4

8. THE INSTRUCTION SHOWN BELOW WILL_____ A WORD ON TO/OFF OF THE_____ STACK

A. PUSH/USER
B. POP/KERNAL
C. PUSH/KERNAL
D. POP/USER

5ØØØ - MFPI (2)

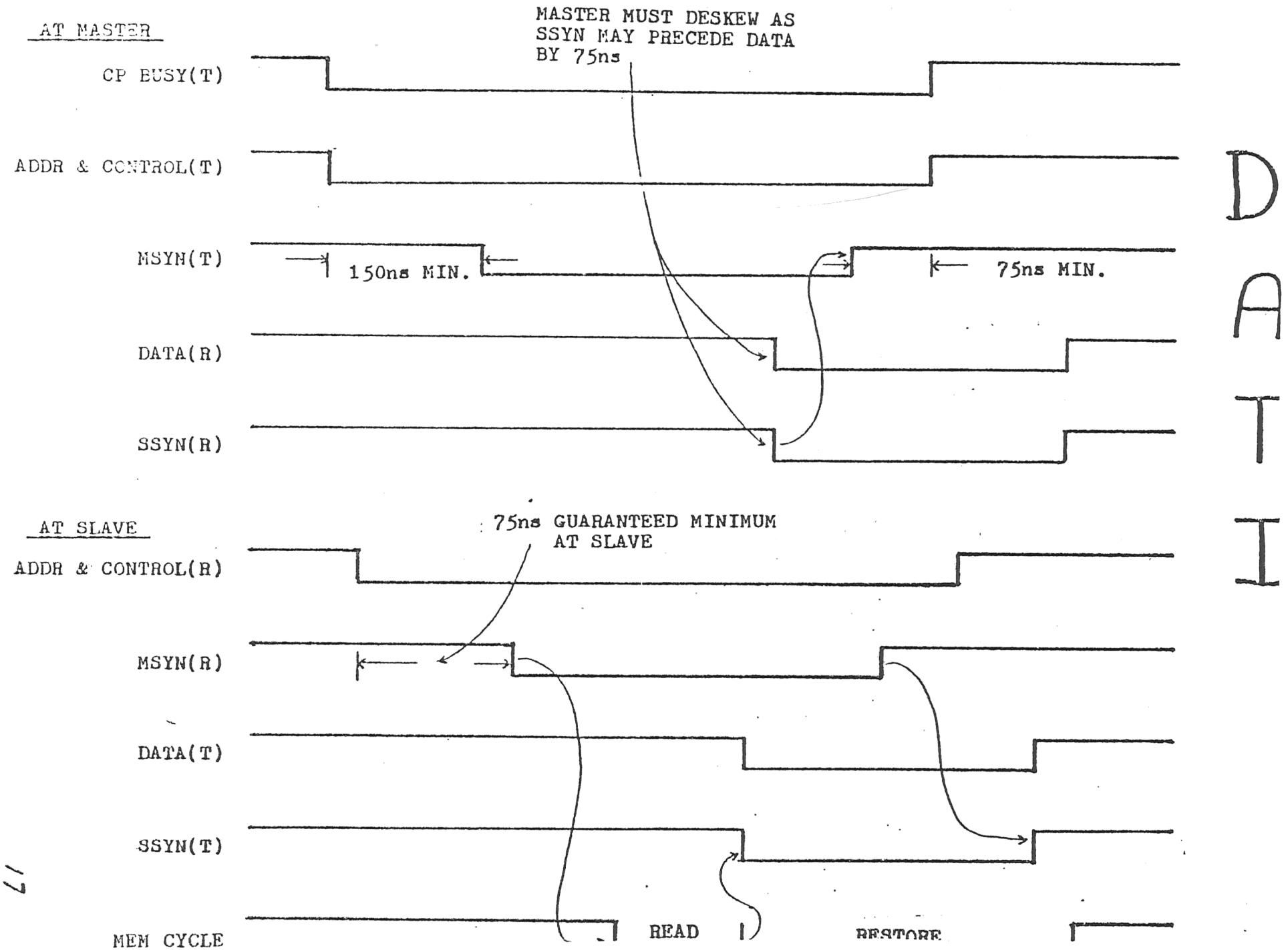
777776 = 14ØØ12

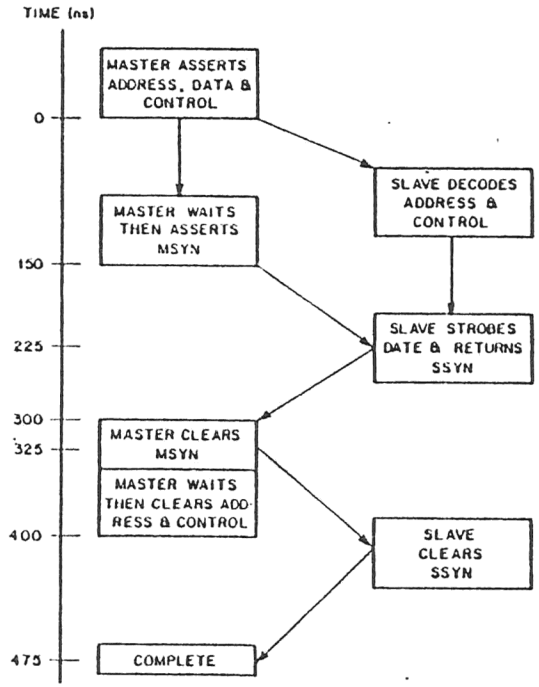
9. IN THE KTJJ-D, THE MAXIMUM PAGE SIZE IS

A. 128 BLOCKS
B. 4096 WORDS
C. 8192 BYTES
D. ALL OF THE ABOVE

10 THE PROCESSOR IS RUNNING IN USER MODE, WHICH OF THE BELOW INSTRUCTIONS WILL ACTIVATE THE SIGNAL K4-2 CLK IR TWICE?

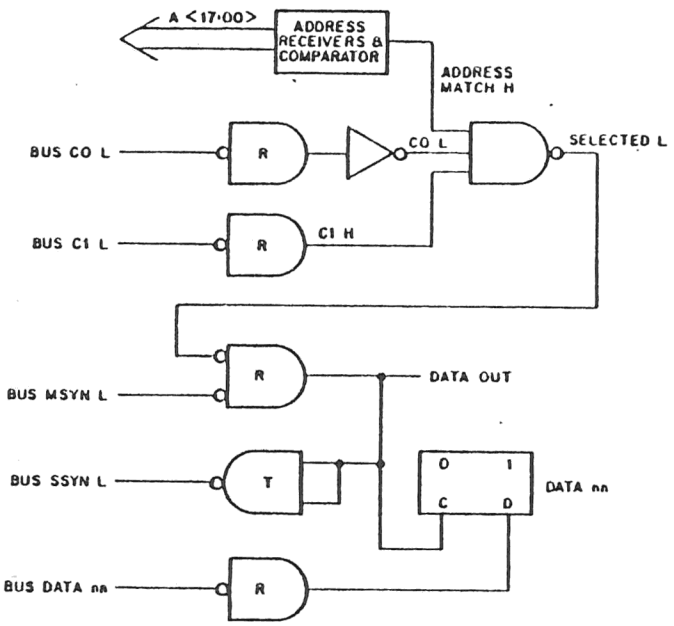
A. RESET
B. MTPI
C. MFPI
D. ALL OF THE ABOVE


AT MASTER
MASTER MUST DESKEW AS SSYN MAY PRECEDE DATA BY 75ns
CP BUSY(T)
ADDR & CONTROL(T)
MSYN(T)
150ns MIN.
75ns MIN.
DATA(R)
SSYN(R)
AT SLAVE
75ns GUARANTEED MINIMUM AT SLAVE
ADDR & CONTROL(R)
MSYN(R)
DATA(T)
SSYN(T)
MEM CYCLE
READ
RESTORE
DATI

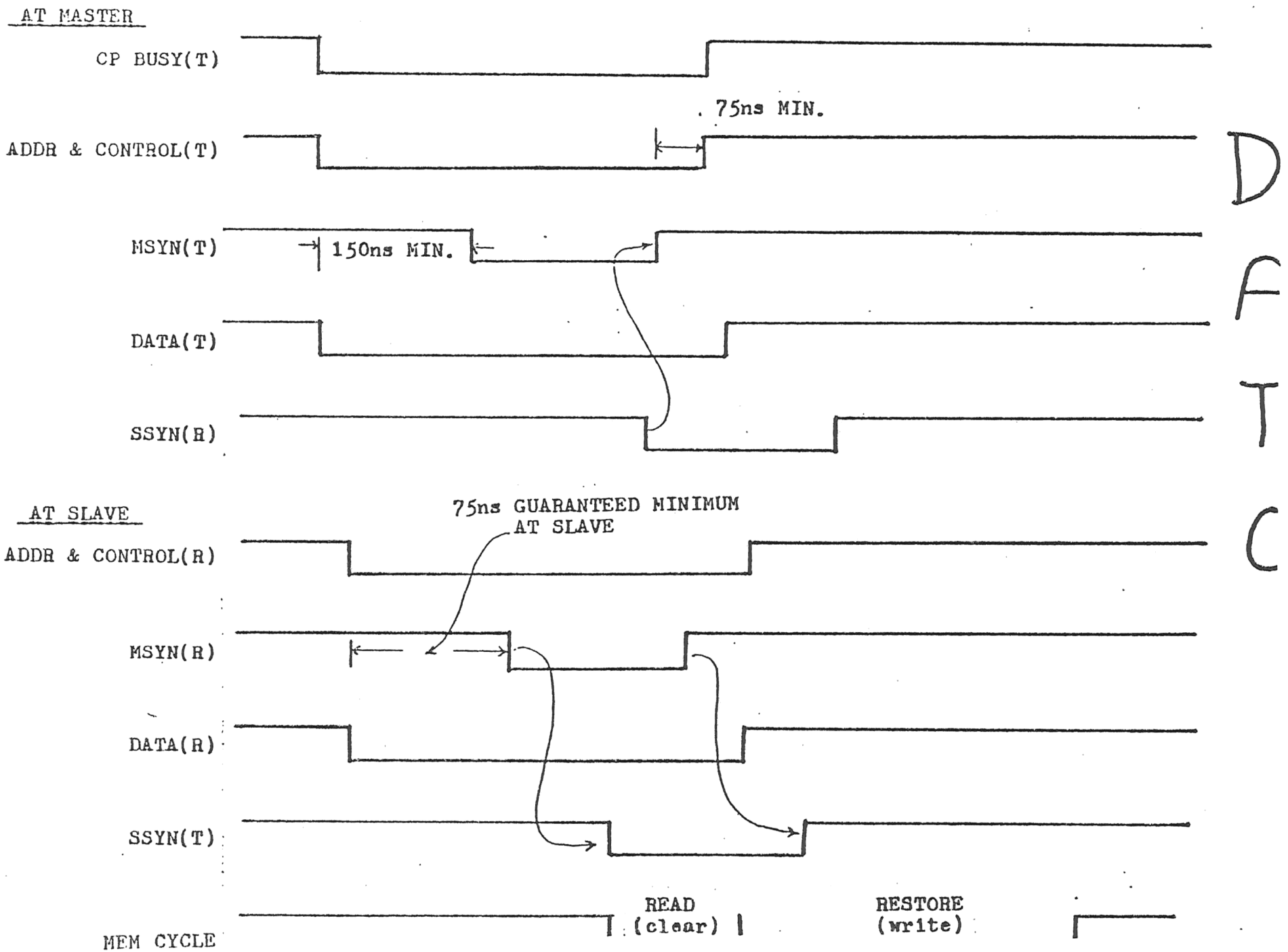


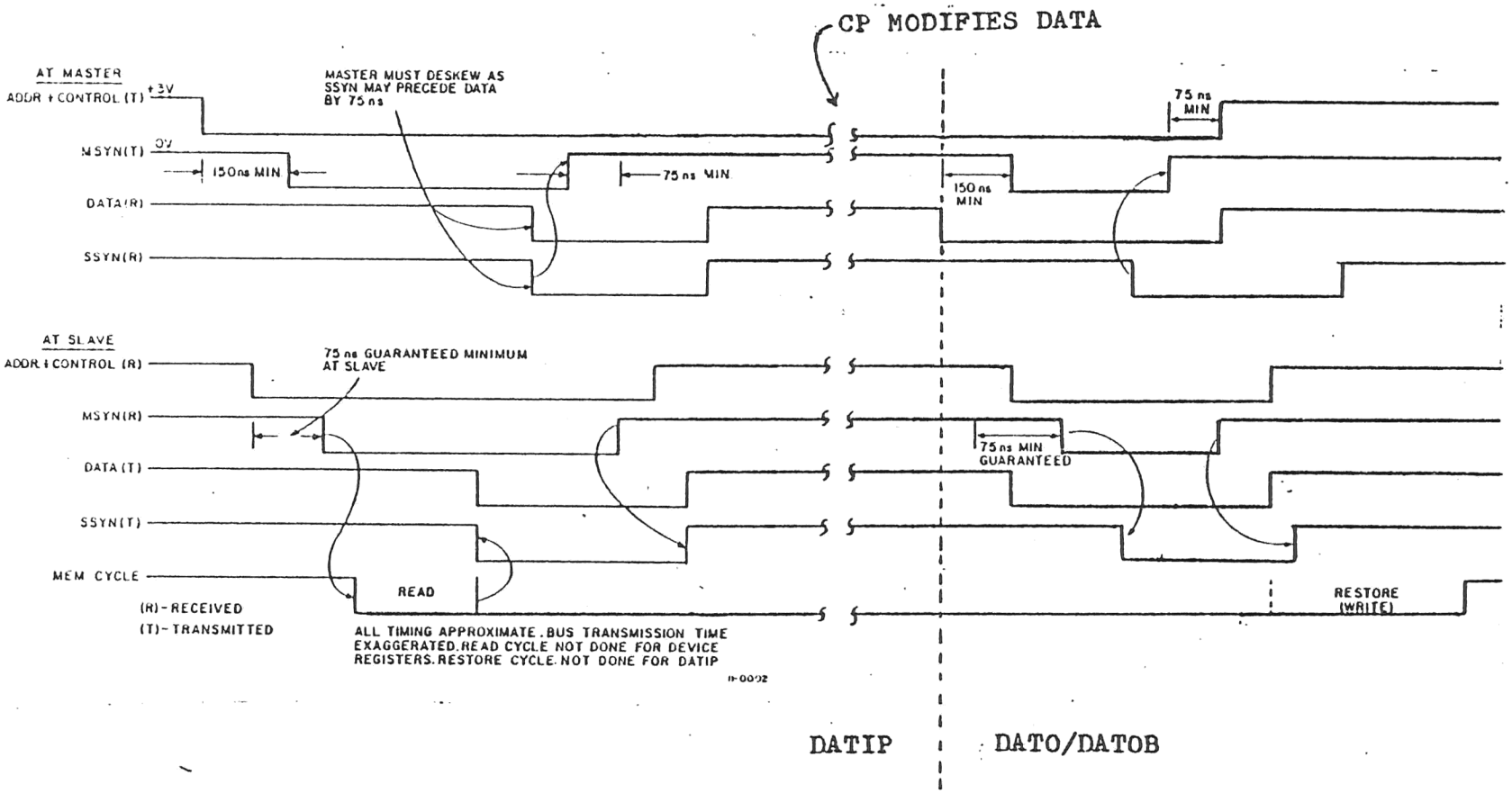
NOTES: Worst case propagation delay of 75 ns is shown. Total time = 475 ns for a single transfer. Next cycle may begin at 400 ns when master may assert new address and control. Maximum sustained transfer rate is 2.5 million words/sec.

Typical DATO Timing Flow



Slave Logic For DATO (Fast Response)


AT MASTER
CP BUSY(T)
. 75ns MIN.
ADDR & CONTROL(T)
MSYN(T)
150ns MIN.
DATA(T)
SSYN(R)
AT SLAVE
75ns GUARANTEED MINIMUM
AT SLAVE
ADDR & CONTROL(R)
MSYN(R)
DATA(R)
SSYN(T)
READ
(clear)
RESTORE
(write)
MEM CYCLE


CP MODIFIES DATA
AT MASTER
ADDR + CONTROL (T)
+3V
0V
MSYN(T)
150ns MIN
MASTER MUST DESKEW AS SSYN MAY PRECEDE DATA BY 75 ns
75 ns MIN
DATA(R)
SSYN(R)
75 ns MIN
150 ns MIN
AT SLAVE
ADDR + CONTROL (R)
75 ns GUARANTEED MINIMUM AT SLAVE
MSYN(R)
DATA (T)
SSYN(T)
MEM CYCLE
READ
75 ns MIN GUARANTEED
RESTORE (WRITE)
(R)-RECEIVED
(T)-TRANSMITTED
ALL TIMING APPROXIMATE. BUS TRANSMISSION TIME EXAGGERATED. READ CYCLE NOT DONE FOR DEVICE REGISTERS. RESTORE CYCLE NOT DONE FOR DATIP
DATIP
DATO/DATOB

DATIP

PRIORITY TRANSFER (BECOMING BUS MASTER)

IN ORDER FOR A DEVICE TO RECIEVE OR SEND INFORMATION (DATA TRANSFER) OVER THE UNI-BUS, THAT DEVICE MUST FIRST BECOME MASTER OF THE BUS. THERE ARE TWO DISTINCT WAYS IN WHICH A DEVICE CAN DO THIS AND THE METHOD BY WHICH THE DEVICE GAINS CONTROL OF THE BUS IS DETERMINED BY WHAT TYPE OF DEVICE IT IS.

NPR

CERTAIN DEVICES HAVE SUFFICIENT CIRCUITRY BUILT INTO THIER INTERFACE TO FUNCTION INDEPENDENTLY ON THE UNI-BUS ONCE THEY GAIN CONTROL. THESE DEVICES ARE CALLED DMA (DIRECT MEMORY ACCESS) DEVICES AND ONCE THEY BECOME BUS-MASTER, THEY CAN INPUT OR OUTPUT TO/FROM MEMORY WITHOUT PROCESSOR INTERVENTION. THESE DEVICES ARE USUALLY THE FASTER PERIPHERALS. SEE INSTRUCTION CARD

1. DEC TAPE
2. DEC DISK (RC11)
3. DEC DISK (RF11)

NOTE THAT EACH OF THESE DEVICES HAS THE FOLLOWING

1. WORD COUNT REG
2. CURRENT ADDRESS REG

ONCE THESE REGISTERS ARE SET UP, ALL THE DEVICE NEEDS IS CONTROL OF THE BUS TO DO ITS DATA XFER. THESE DEVICES WILL ASK FOR BUS CONTROL OVER THE SINGLE NPR LINE.

IT IS THE INTENT HERE THAT DEVICES USING THE NPR LINE ARE FAST AND SELF SUFFICIENT, SO WHEN THE PROCESSOR SEES AN NPR REQUEST, IT MUST HONOR THAT REQUEST VERY QUICKLY OR A CONDITION MAY ARISE WHERE DATA WILL BE LOST. SUCH AS THE DISK READING INFORMATION AND NOT BEING ABLE TO HOLD IT IN ITS BUFFER REGISTER FOR LONG PERIODS OF TIME

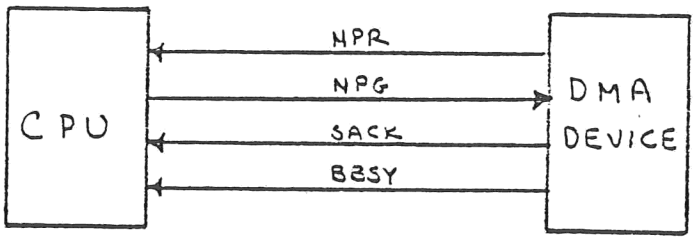
THE PROCESSOR WILL HONOR AN NPR REQUEST AFTER EACH BUS CYCLE. ACTUALLY THE PROCESSOR WILL HESITATE BEFORE ENTERING A BUS CYCLE JUST TO SEE IF AN NPR IS PRESENT. UPON RECOGNITION THAT THE NPR LINE IS ACTIVE, THE PROCESSOR WILL IMMEDIATELY ASSERT THE NPG LINE (EXCEPT WHEN PROCESSOR IS INVOLVED IN A DATIP BUS CYCLE) THE PROCESSOR DOES NOT KNOW WHICH DEVICE ASSERTED NPR, IT JUST SENDS THE NPG(GRANT)

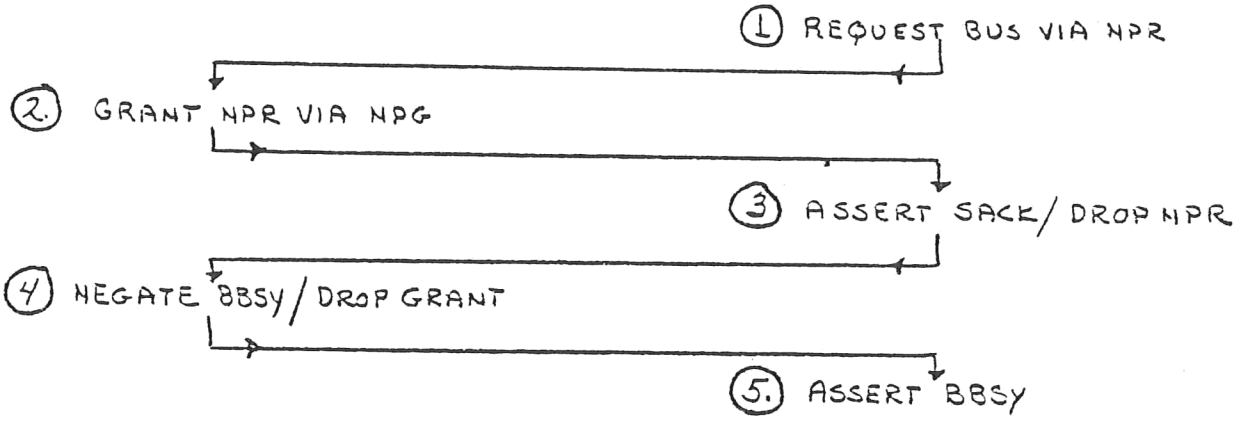
ALTHOUGH THERE MAY BE SEVERAL NPR DEVICES OUT ON THE UNI-BUS AND A CONDITION MAY ARISE WHERE EACH OF THEM REQUESTED THE BUS AT THE SAME TIME, THE PROCESSOR ONLY RECOGNIZES THAT AN NPR IS PRESENT AND SENDS NPG. DMA DEVICES ARE ARRANGED ON THE UNI-BUS IN A PRIORITY SCHEME WITH ONE ANOTHER. AND IT IS SIMPLY THE PHYSICAL PROXIMITY OF THE DEVICE TO THE PROCESSOR THAT WILL DETERMINE ITS PRIORITY AGAINST ALL OTHER DMA DEVICES. THE CLOSER THE DEVICE, THE HIGHER ITS PRIORITY. THIS IS ACCOMPLISHED BY RUNNING THE BUS NPG LINE SERIALLY THRU THE DEVICE AND CAUSING A BLOCK NPG IF THAT DEVICE IS ASSERTING NPR. THIS IS SHOWN ON PRIORITY SCHEME BLOCK DIAGRAM.

ONCE THE PROCESSOR ASSERTS THE NPG (GRANT) LINE A 10 USEC TIME OUT IS STARTED IN THE PROCESSOR. THE PROCESSOR HAS HUNG OUT THE GRANT BUT IS NOT GOING TO WAIT FOREVER FOR THE DEVICE TO RESPOND TO IT. THE PROPER RESPONSE FROM THE DEVICE WOULD BE ASSERTION OF THE SACK LINE IF THE PROCESSOR SEES THIS RESPONSE WITHIN 10 USECS OF ASSERTING GRANT THE PROCESSOR WILL KILL THE TIMEOUT AND CONTINUE ITS DIALOUGE OF GRANTING THE BUS.

UPON RECIEPT OF SACK THE PROCESSOR WILL NEGATE BBSY WHICH IS A SIGNAL TO THE DEVICE THAT PROCESSOR IS YIELDING CONTROL OF THE BUS.

A GENERAL OUTLINE OF NPR PRIORITY TRANSFER WILL SERVE AS A REVIEW.





NOW THE DEVICE IS BUS MASTER AND THE PROCESSOR IS LOCKED OUT OF BUS USAGE UNTIL THE DEVICE CHOOSES TO DROP BBSY. HOWEVER, THE PROCESSOR WILL ARBITRATE VIA THESE PRIORITY TRANSFER LINES FOR THE NEXT BUS MASTER.

THE DEVICE WILL NOW ACCOMPLISH A DATA XFER AND AT ITS CONCLUSION WILL GIVE UP BUS MASTERSHIP BY SIMPLY NEGATING BBSY. WHEN THE PROCESSOR SEES BBSY DROP, IT MAY HAVE ALREADY HAD A DIALOUGE WITH ANOTHER DMA DEVICE IN WHICH CASE THE NEXT DMA DEVICE WILL ASSERT BBSY AND BE BUS MASTER. IF NO NPRS WERE ACTIVE WHILE THIS DEVICE WAS DOING ITS DATA TRANSFER, THE PROCESSOR MAY SNEAK IN FOR A BUS CYCLE BEFORE THIS DEVICE GAINS THE BUS FOR ITS NEXT TRANSFER. THIS IS SHOWN ON PRIORITY TRANSFER TIMING DIAGRAM.

SUMMARY OF NPR PRIORITY TRANSFER.

1. WHEN DOES PROCESSOR GRANT NPRS ?
2. WHAT IS PROCESSORS RESPONSE TO NPRS ?
3. IF MORE THEN ONE DMA IS REQUESTING THE BUS, WHICH WILL GET IT ?
4. HOW LONG WILL PROCESSOR HOLD OUT THE GRANT ?
5. WHAT IS DEVICE RESPONSE TO NPG ?
6. WHAT DOES PROCESSOR DO WITH SACK ?
7. WHAT DOES DEVICE DO WHEN BBSY DROPS ?
8. WHEN DEVICE GAINS BUS CONTROL VIA NPR, WHAT DOES THAT DEVICE INTEND TO DO ?
9. HOW DOES PROCESSOR REGAIN BUS MASTERSHIP .
10. HOW MANY DATA TRANSFERS WILL DEVICE DO BEFORE RELEASING BUS ?

PRIORITY TRANSFER

BR

THE PRECEDING DISCUSSION CONCERNED IT SELF WITH DEVICES GAINING BUS MASTERSHIP VIA NPR ROUTE. OTHER DEVICES, USUALLY INEXPENSIVE, SLOWER DEVICES (NON DMA) HAVE NO INTERFACE CIRCUITRY FOR DOING AN INDEPENDENT DATA TRANSFER. THESE DEVICES RELY ON THE PROCESSOR TO TO ACTUALLY DO THE TRANSFER FOR THEM. AS AN EXAMPLE, A TELETYPE MAY HAVE A CHARACTER IT WANTS TO PUT INTO MEMORY. ALL THAT TELETYPE CAN DO, IS TO GET THE ATTENTION OF THE PROCESSOR VIA ONE OF 4 BR (BUS REQUEST) LINES. THE DIALOUGE THAT WILL TAKE PLACE HERE IS PRETTY MUCH THE SAME AS NPR BUT THE INTENT IS TOTALLY DIFFERENT.

THE PRIORITY SCHEME FOR BR DEVICES IS TWO-FOLD

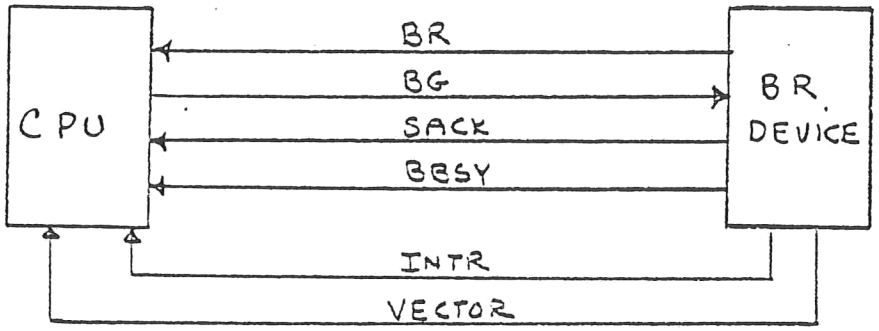
PRIMARY PRIORITY = BR LINE ASSIGNED. (ASCENDING ORDER)
SECONDARY PRIORITY = PHYSICAL PROXIMITY

A REVIEW OF PRIORITY SCHEME FOR BR WILL SERVE HERE.

CHART PRIORITIES FOR NON-LINEAR ARRAY

NPR	A	B	C	D	E	F
BR	D	F	E	A	C	B

THE AIM OF THE BR SEQUENCE IS TO SIMPLY GRANT BUS MASTER SHIP TO A DEVICE SO THAT THE DEVICE CAN INTERRUPT THE PROCESSOR. THE FOLLOWING IS THE SEQUENCE OF EVENTS.



1 DEVICE ASSERTS BR

2 CPU ASSERTS GRANT { GRANTED AT END OF INSTRUCTION PRIORITY ARBITRATION WITH PSW AND OTHER BR'S }

3. DEVICE ASSERTS SACK/DROPS BR

4 CPU DROPS BBSY & GRANT

5 DEVICE ASSERTS BBSY/DROPS SACK BRINGS UP INTR LINE AND PUTS VECTOR ON DATA LINES

6. CPU VECTORS TO SERVICE DOES DATA XFER, ASSERTS SSYN TO CAUSE BR DEVICE TO DROP BBSY

NOTE: WHEN CPU RECIEVES INTR, CPU WILL SEND SSYN WHICH WILL CAUSE DEVICE TO DROP BBSY/INTR/VECTOR ADRS. THIS MAKES CPU BUS MASTER.

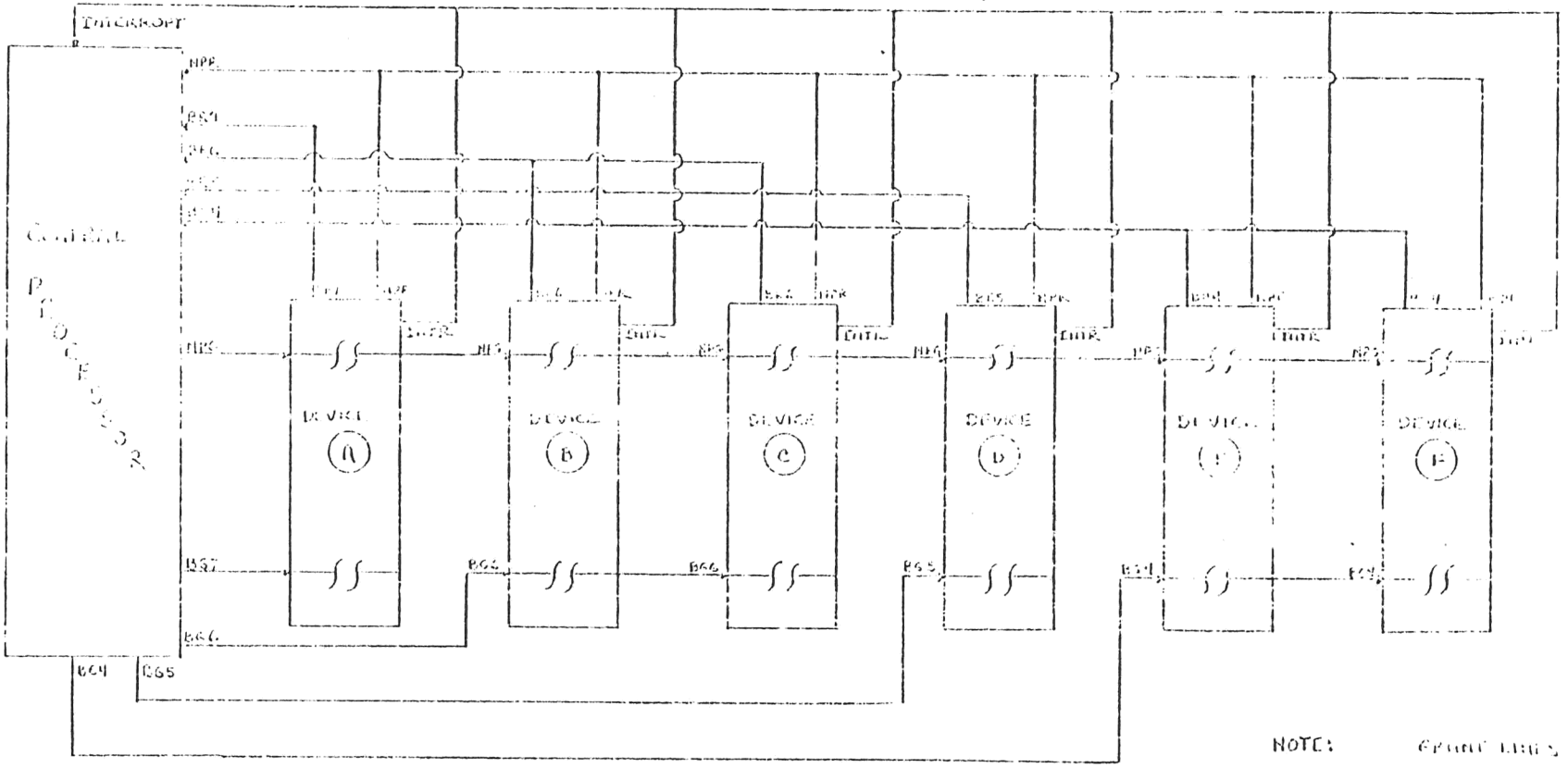
PRIORITY SCHEME

LINEAR

"NPR" HONORED AFTER EACH BUSS CYCLE (EXCEPT FOR DATIP)

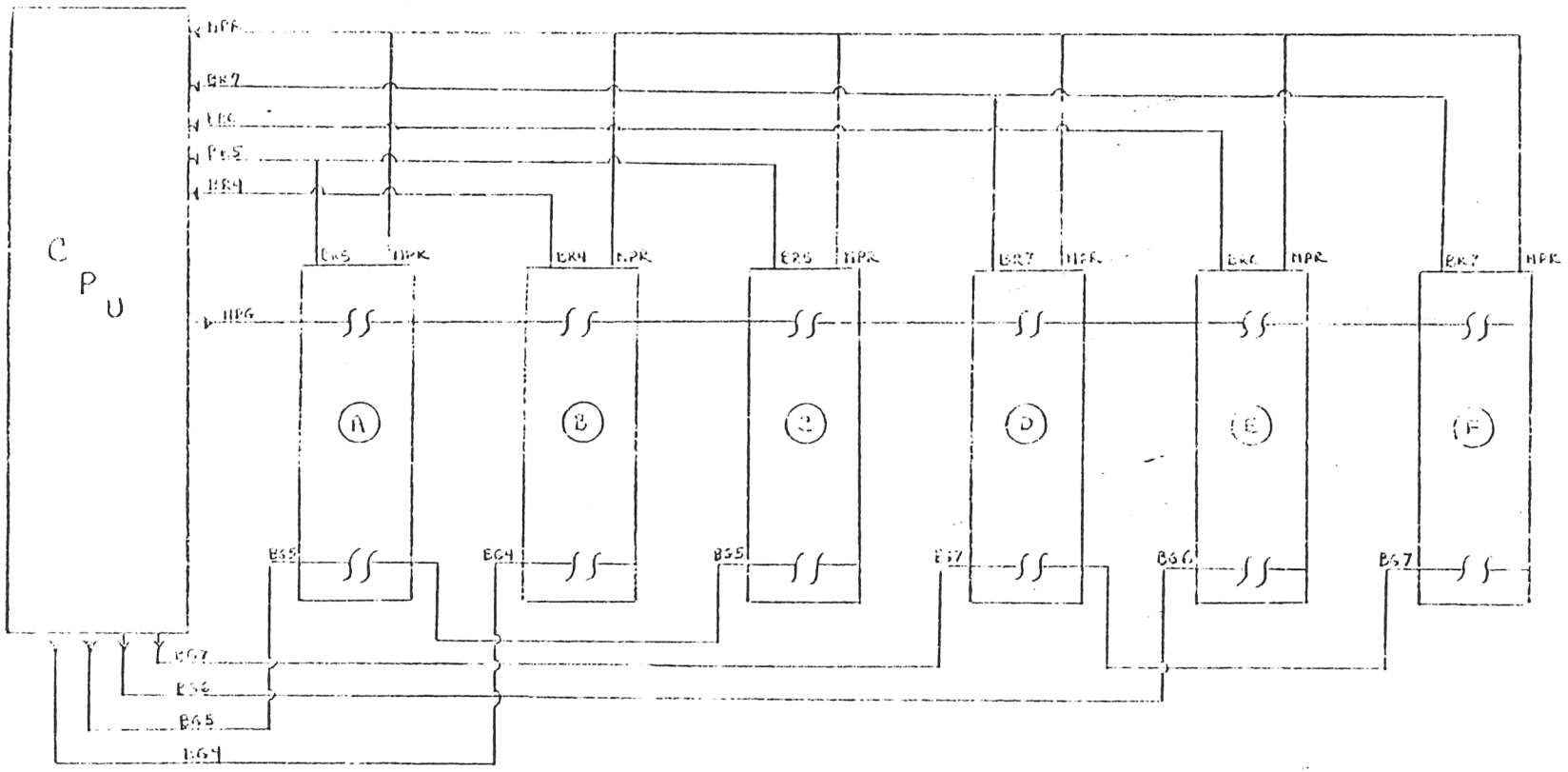
"BR" HONORED AFTER EXECUTION OF EACH INSTRUCTION

"INTR" LINE CANNOT GO ACTIVE UNTIL DEVICE CAPTURES BUSS VIA BR LINE

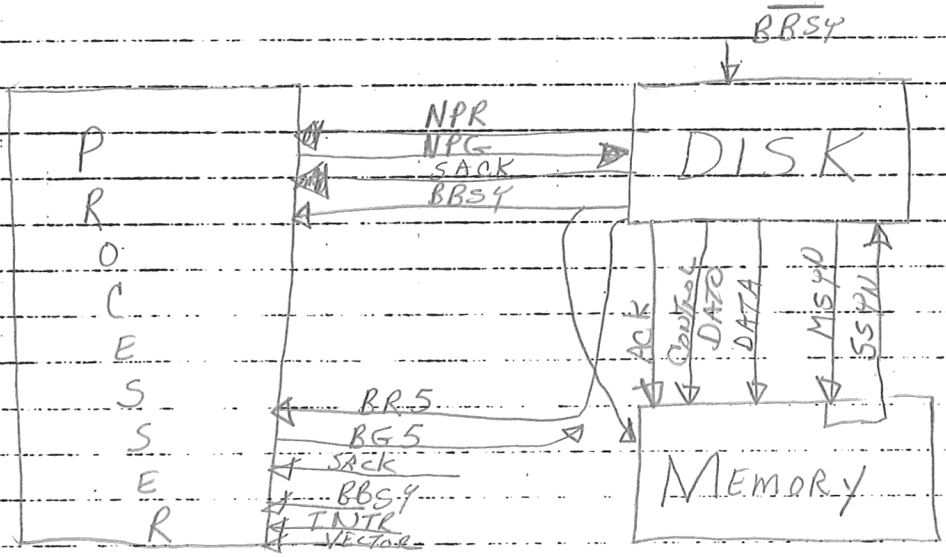


NOTE: [illegible] GOING LINES GO HIGH WHEN TRUE

PRIORITY SCHEME
NON-LINEAR

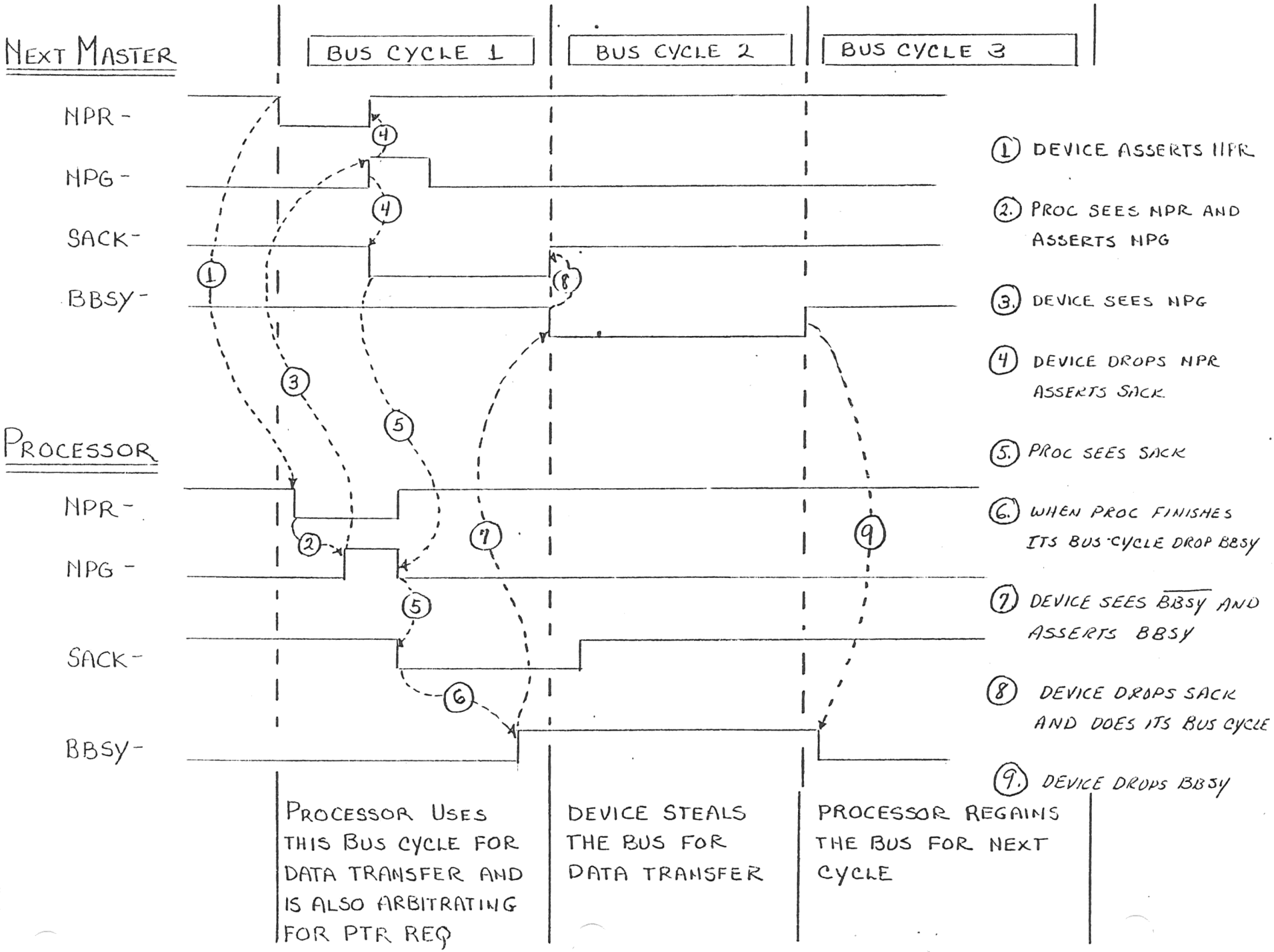


THIS DIAGRAM SHOWS HOW THE PROCESSOR CAN BE DOING A DATA TRANSFER BUS CYCLE AND AT THE SAME TIME BE ARBITRATING A PRIORITY REQUEST.



C1	C0	
0	0	DATI
0	1	DATIP
1	0	DATO
1	1	DATOB

- PRIORITY TRANSFER -



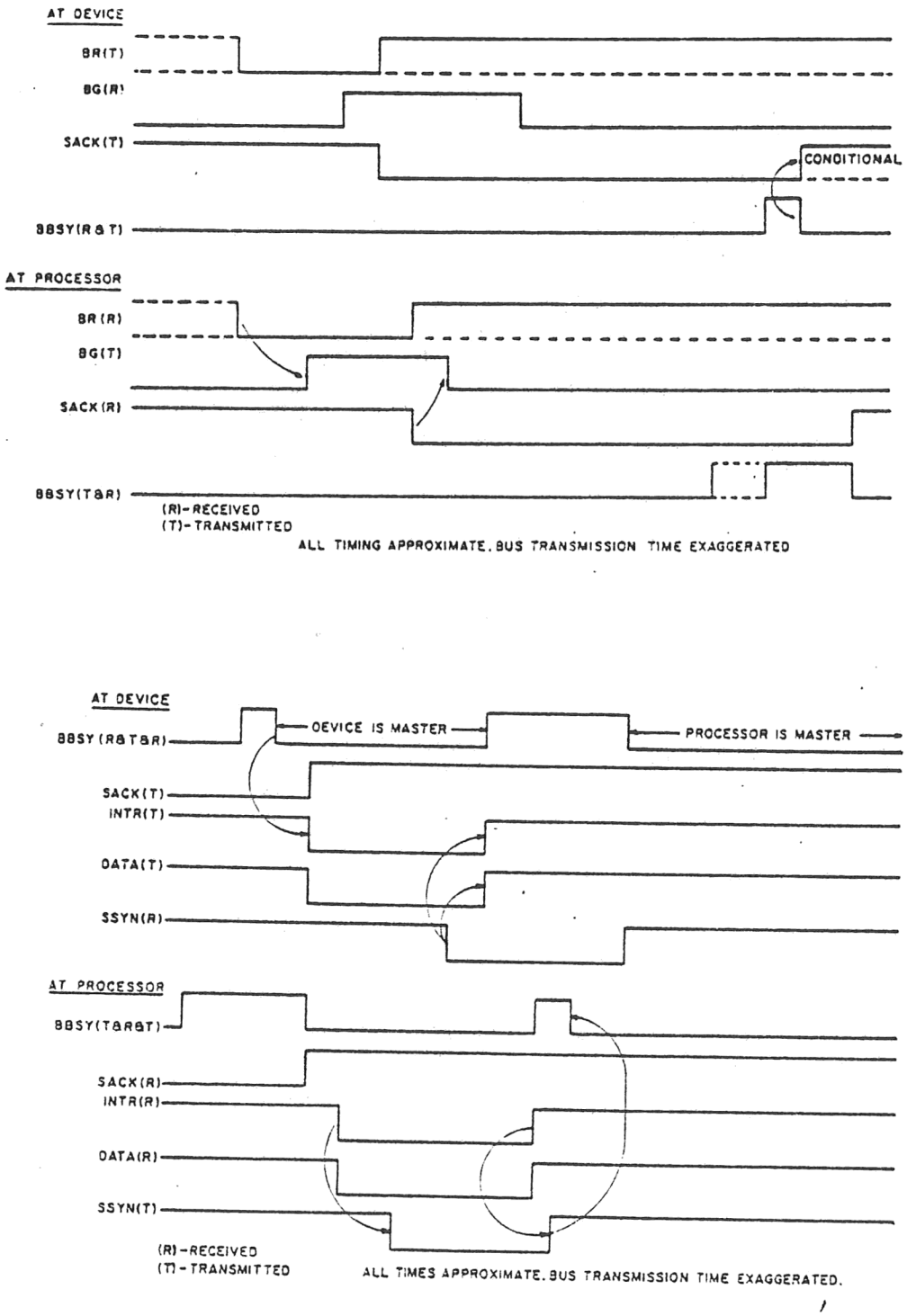
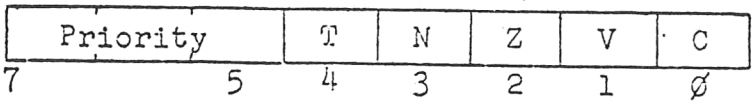


Figure 2-7 INTR Timing Diagram

PROCESSOR STATUS WORD



Where: Priority — specifies 1 of 8 levels of processor priority, which in turn determines the BR priority level(s) on which the BUS may be gained.

T — Trace Bit causes an automatic trap at the completion of the instruction following the one that set the T-bit.

N — A Condition Code indicating that the result of the last instruction affecting it was negative.

Bit 15 1/Ø Negative/Positive Word
Bit 7 1/Ø Negative/Positive Byte

Z — A Condition Code indicating that the result of the last instruction affecting it was zero.

V — A Condition Code indicating that the result of of the last arithmetic operation exhibited arithmetic overflow (a change in sign when quantities of like sign are added). Also used elsewhere as indicated in the PDP-11 Handbook. Exceptions also noted in the Handbook.

C — A Condition Code indicating that the result of the last instruction affecting it had a carry out of Bit 15 (Bit 7 if Byte). Exceptions noted in the PDP-11 Handbook.

The Status Register may be explicitly addressed at 777776.

The Condition Codes in the Status Word are not implicitly affected for a MOV #n,177776 or for any other instruction whose purpose is to explicitly modify the Processor Status Word.

QUIZ DAY 1

1. WHICH ONE OF THE PROCESSOR'S GENERAL PURPOSE REGISTERS HOLDS THE ADDRESS FROM WHICH EACH INSTRUCTION IS FETCHED.

A - R4
B - R5
C - R6
(D) - R7

2. THE CONTENTS OF THE PROCESSOR STATUS WORD IS 000311_8. A REQUEST FOR BUS MASTERSHIP CAN BE GRANTED IF MADE ON

A - BR6 OR BR7
B - NPR ONLY
C - BR7 ONLY
(D) - BR7 OR NPR

3. WHICH OF THE FOLLOWING UNI-BUS SIGNAL LINES IS NOT USED DURING A DATA TRANSFER BUS TRANSACTION?

A - BUS D03
B - BUS MSYN
C - BUS. SSYN
(D) - BUS SACK

4. A DEVICE MAY ASSERT BUS INTR LINE

A - AT ANY TIME
(B) - ONLY AFTER BECOMING BUS MASTER VIA BR LINE
C - ONLY AFTER BECOMING BUS MASTER VIA NPR LINE
D - ANY TIME AFTER BECOMING BUS MASTER

5. DURING A BR PRIORITY TRANSFER SEQUENCE, THE DEVICE RESPONDS TO THE PROCESSOR'S GRANT SIGNAL BY _____ THE _____ LINE

(A) - ASSERTING, SACK
B - DROPPING, BBSY
C - ASSERTING, BBSY
D - DROPPING, SACK

6. WHICH OF THE BELOW LISTED DEVICES ENJOYS THE HIGHEST PRIORITY FOR REQUESTING CONTROL OF THE UNI-BUS VIA BR LINE? (SEE INSTRUCTION CARD)

A - DISK 5
B - LINE PRINTER 4
(C) - LINE CLOCK 6
D - TELETYPE 4

7. THE PROCESS OF TAKING ELEMENTS OFF OF THE STACK IS CALLED

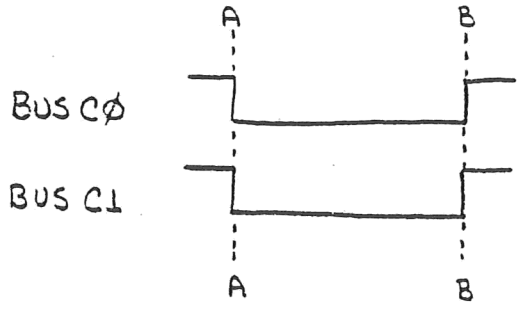
A - PUSHING
(B) - POPPING
C - MOVING
D - TRANSFERRING

PC is lowest between the PC & PSW. when taking them off the stack

8. IF MORE THEN ONE DEVICE IS CONNECTED TO THE SAME BR LINE, WHAT FACTOR DETERMINES WHICH DEVICE HAS THE HIGHER PRIORITY

A - THE NUMBER OF ITS GRANT LINE
B - THE TYPE OF DEVICE
(C) - THE PHYSICAL DISTANCE FROM PROCESSOR
D - THE SPEED AT WHICH THE DEVICE CAN ASSERT BBSY LINE

9. ASSUME THE PROCESSOR IS BUS MASTER AND IS EXHIBITING THE FOLLOWING CONTROL LINE LEVELS ON THE UNI-BUS. WHAT TYPE OF MEMORY CYCLE IS BEING PERFORMED BETWEEN POINTS A AND B



control lines are high when asserted low!

A.- DATI 00
B.- DATIP 01
C.- DATO 10
(D).- DATOB 11

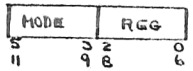
10. DURING A DATI BUS TRANSACTION, MEMORY DOES NOT RESPOND TO PROCESSOR'S MSYN. WHAT WILL PROCESSOR DO?

A - ABORT THE DATI AND PROCEED WITH NEXT INSTRUCTION
B - RETRY BY SENDING MSYN A SECOND TIME
C - HALT THE PROCESSOR AND TURN OFF THE RUN LIGHT
(D) - TRAP TO A FIXED VECTOR ADDRESS

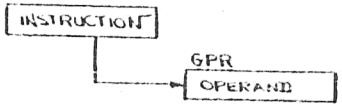
ADDRESSING MODES AND INSTRUCTIONS

DAY 2 & 3

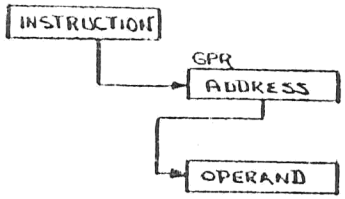
GENERAL REGISTER ADDRESSING



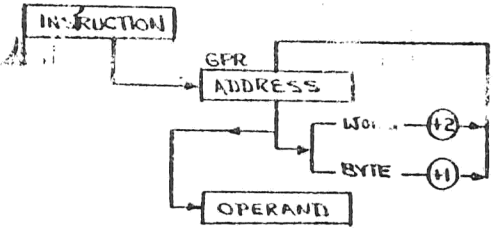
MODE 0 REGISTER OP %Rn



MODE 1 REGISTER DEFERRED OP (Rn)

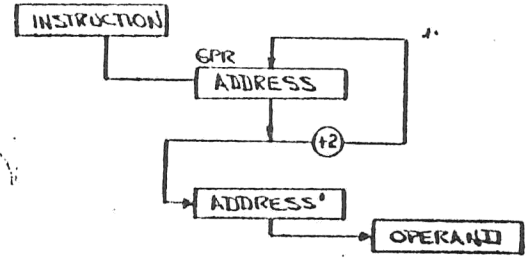


MODE 2 AUTO-INCREMENT OP (Rn)+

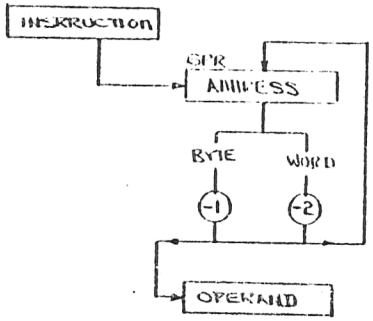


* If R6 or R7 update by +2 always

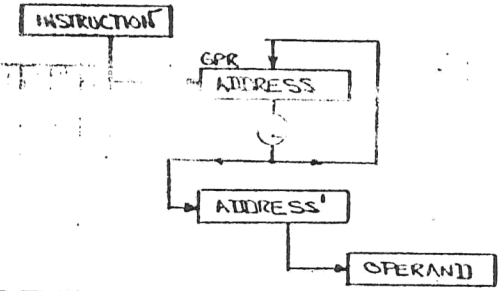
MODE 3 AUTO-INCREMENT DEFERRED OP @(Rn)+



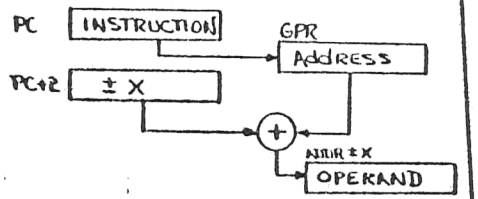
MODE 4 AUTO-DECREMENT OP -(Rn)



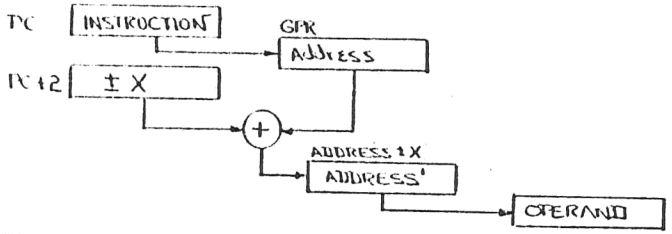
MODE 5 AUTO-DECREMENT DEFERRED OP @-(Rn)



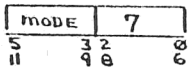
MODE 6 INDEXED OP ±X (Rn)



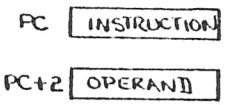
MODE 7 INDEXED DEFERRED OP @±X(Rn)



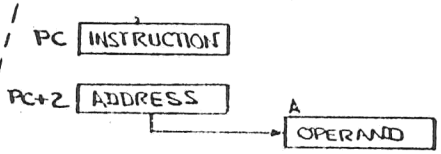
PC REGISTER MODES



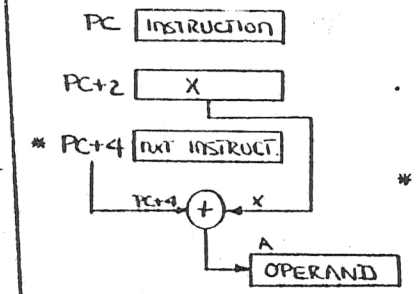
MODE 2 IMMEDIATE OP #n



MODE 3 ABSOLUTE OP@#A



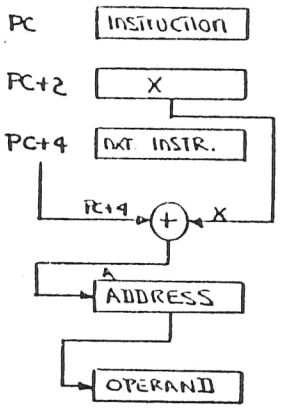
MODE 6 RELATIVE OP A



{A = (PC+4) + X}

or: X = A - (PC+4)

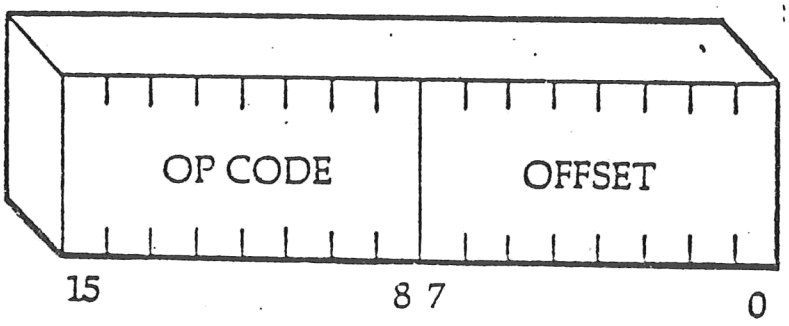
MODE 7 RELATIVE DEFERRED OP @A



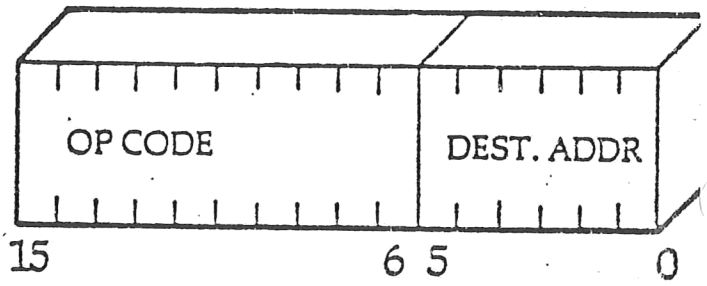
* This could end up being PC+6 if both source & destination of a double operand instruction required an index word.

instructions format

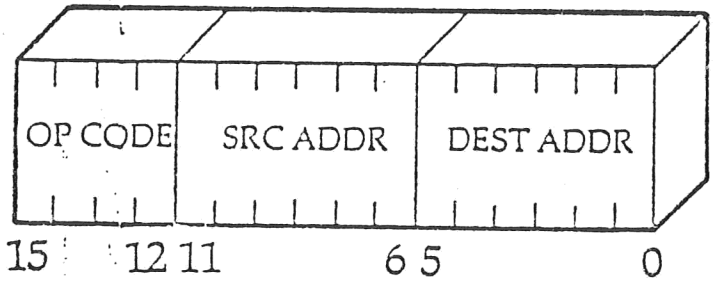
branch instructions



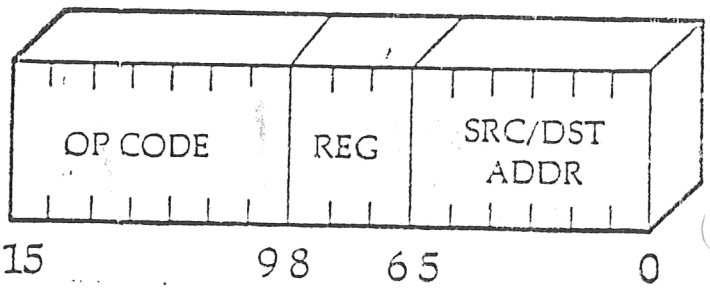
single operand



double operand



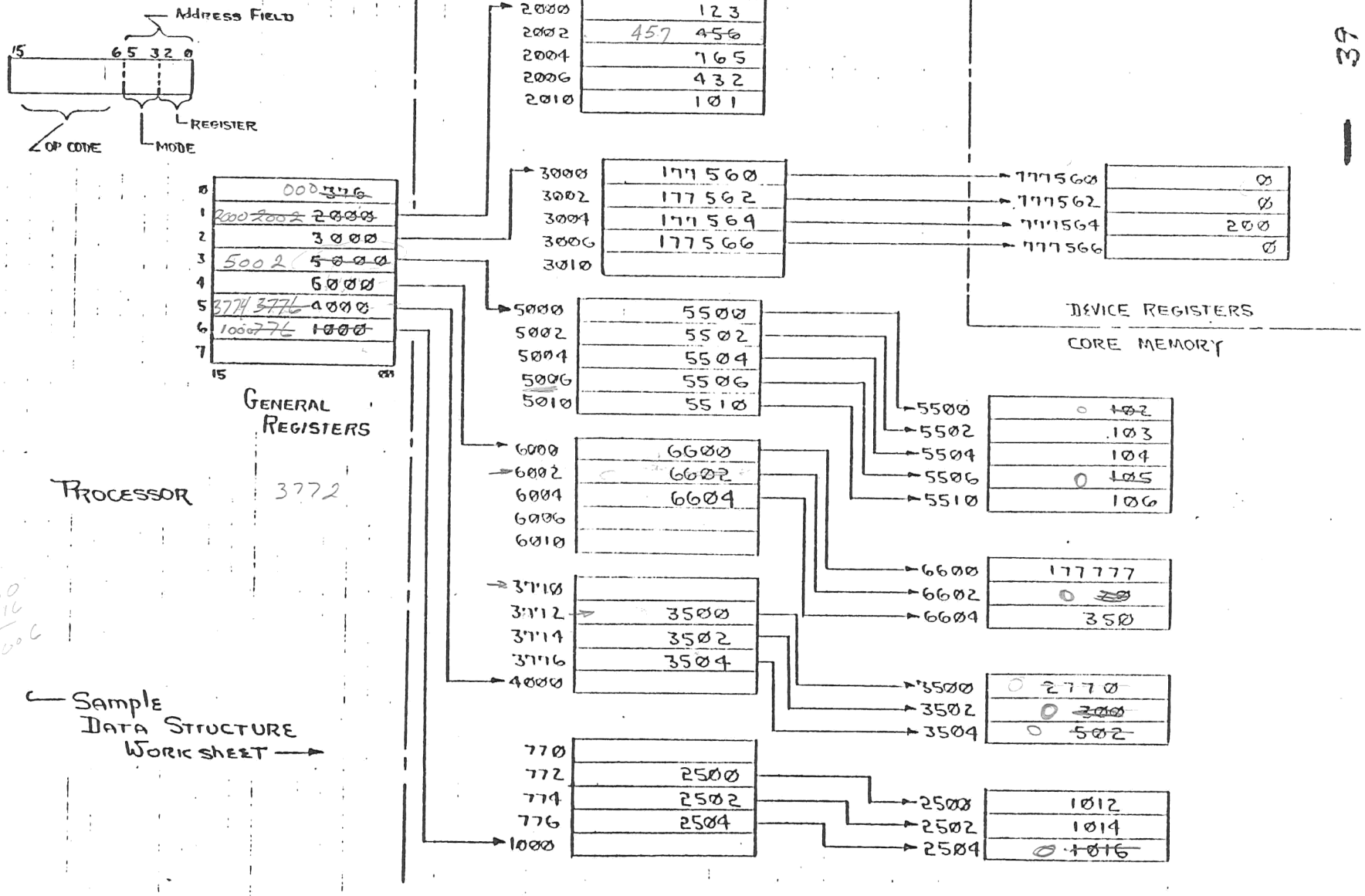
one+half operand



CLASS PROJECT (REG MODES)

REFER TO THE SAMPLE DATA STRUCTURE WORKSHEET AND ANALYZE EACH INSTRUCTION SHOWN BELOW IN THE ORDER LISTED. MAKE CHANGES TO DATA WORKSHEET AS THEY OCCUR IN THE EXECUTION OF EACH INSTRUCTION BECAUSE THE CHANGE IN DATA MAY EFFECT THE NEXT INSTRUCTION IN LINE.

1. CLR %Ø
2. TST (1)+
3. INC (1)
4. TST -(1)
5. CLR @ Ø (2)
   Ø
6. CLR@6 (3)
   6
7. TST (4)+
8. CLR@Ø (4)
   Ø
9. CLR 177772(5)
   177772
10. TST -(5)
11. CLR@ -(5)
12. TST -(6)
13. CLR@ Ø(6)
    Ø
14. TST (6)+
15. CLR @(3)+



Processor 3772

Sample Data Structure Worksheet →

PRACTICE USING COM/NEG/INC/DEC

BEFORE

R1 = 600
R2 = 602
R3 = 604
576 = 177400
600 = 000377
602 = 100100
604 = 177777
606 = 000001

AFTER

R1 = 576
R2 = 602 → 601 → 600
R3 =
576 = 377
600 = 177401
602 = 177500
604 = 177401
606 =

EXECUTE THE FOLLOWING AND SHOW HOW THE ABOVE WAS CHANGED

500 - COM -(1)
502 - NEG 2(1)
504 - 2
506 - INCB -(2)
510 - DECB -(2)
512 - COMB -1(3)
514 - 177777
516 - NEGB (3)
520 - HALT

low order byte

ANSWERS TO COM/NEG/INC/DEC

R1 = 600 → 576
R2 = 602 → 601 → 600
R3 = 604
576 = 177400 → 000377
600 = 000377 → 177401 → 000001 → 000000
602 = 100100 → 077500
604 = 177777 → 177401
606 = 000001

PRACTICE USING SWAB

BEFORE

R1 = 600
R2 = 1000
R3 = 1500
576 = 177000
600 = 055550
602 = 012124
1000 = 137654
1500 = 001100

AFTER

R1 = 576
R2 = 1000
R3 = 40003
576 = 376
600 = 64133
602 = 052624
1000 = 137654
1500 = 40002

EXECUTE THE FOLLOWING AND SHOW HOW THE ABOVE IS CHANGED.

500 - SWAB 2(1)
502 - 2
504 - SWAB -(1)
506 - SWAB 2(1)
510 - 2
512 - SWAB %3
514 - SWAB -36303 (3)
516 - 141475
520 - HALT

15

1 101 000 000

001 000 000 000 011

4 0 0 0 3

012 124

000 010 001 010 100

000 010 000 010 100

0 5 2 0 2 4

055 550

101 101 101 101 000

001 010 010 010 000

0 6 4 1 33

1 111 111 000 000 000

000 000 011 111 110

0 0 0 376

40003
-36303
01500

001 001 000 000

001 000 000 000 010

1

ANSWERS TO SWAB

R1 = 6ØØ → 576
R2 = 1ØØØ
R3 = 15ØØ → Ø4ØØØ3
576 = 177ØØØ → ØØØ376
6ØØ = Ø5555Ø → Ø64133
6Ø2 = Ø12124 → Ø52Ø24
1ØØØ = 137654
15ØØ = ØØ11ØØ → Ø4ØØØ2

012 345

0 001 010 011 100 101 0

312

PRACTICE USING ASR/ASL

BEFORE EXECUTION:

R1 = 600
R2 = 700
R3 = 1000
R4 = 2000
R5 = 3000
600 = 4000
676 = 111000
700 = 5000
1000 = 177776
2000 = 7000
3000 = 177774
4000 = 6
5000 = 012345

AFTER EXECUTION

R1 = 600
R2 = 700 → 677
R3 = 1000
R4 = 4000
R5 = 3000
600 = 4000
676 = 144400
700 = 120000
1000 = 077777
2000 = 7000
3000 = 177774
4000 = 3
5000 = 012312

{EXECUTE THE FOLLOWING PROGRAM AND SHOW WHICH REGISTERS AND MEM LOCATIONS CHANGED ABOVE.}

500 - ASL(2)
502 - ASR(3)
504 - ASL %4
506 - ASR(4)
510 - ASLB 2000(5)
512 - 2000
514 - ASRB -(2)
516 - HALT

1 001 001 000 000 000

001 100 100 100 000 000

1 11

C=1

000 001 010 011 100 101

0 12362

0 1 2 0 0 0

177776

3000
2000
5000

1 111 111 111 111 110

C=0

011 111 111 111 111 110

000 010 000 000 000

0001 010 011 100 101

011 100 101

162

101 110 0101

0 1 2 1 6 2

011 110 010

110

0 00 1 010 011 100 101

0 001 010 011 100 101

001 010 011 100 101

ANSWER TO ASR/ASL

R1 = 600
R2 = 700 → 677
R3 = 1000
R4 = 2000 → 4000
R5 = 3000
600 = 4000
676 = 111000 → 144400
700 = 5000 → 12000
1000 = 177776 → 177777
2000 = 7000
3000 = 177774
4000 = 6 → 3
5000 = 012345 → 012312

0 000 000011 100 100

PRACTICE USING ROR/ROL

BEFORE EXECUTION

R1 = 600
R2 = 600
R3 = 605
R4 = 606
R5 = 610 607
600 = 612
602 = 614
604 = 616
606 = 620
610 = 622
612 = 123456
614 = 1
616 = 100000
620 = 151515
622 = 033312

#30

AFTER EXECUTION

R1 = 600
R2 = 1402
R3 = 302
R4 = 607
R5 = 607
600 = 1424
602 = 614
604 = 616
606 = 310 → 344
610 = 622
612 = 123456
614 = 1
616 = 100000
620 = 151515
622 = 066624

100
37
-30

EXECUTE THE FOLLOWING PROGRAM AND SHOW WHICH REG/MEM LOCATIONS CHANGED. (ASSUME C BIT = 0 TO START)

500 - ROL 12 (5)
502 - 12
504 - ROL (1)
506 - ROR (4)
510 - ROL R2
512 - ROR R3
514 - RORB (4)+
516 - ROLB -(5)
520 - HALT

C=0
0011 0111 0111 0101 0100
0 6 6 6 2 4

612 C=0
0110 0011 0101
3 0 5
C=0
1110 0101 0100
4 2 4

310 C=0
0110 0010 0000
1 0 6 0

620 C=0
0110 0101 0000
3 1 0

NOTE: YOU MUST KEEP TRACK OF PSW (C) BIT

C=0
1 0011 0011 0000
1 0 0 3 4 4

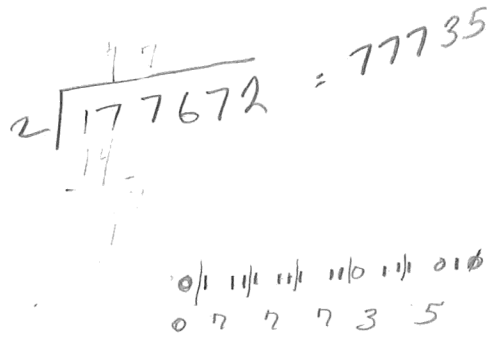
602 C=0
1110 0101 0100
1 4 0 4

C=0
1110 0101 0101
C=1
0110 0010 1011

ANSWERS TO ROR/ROL

R1 = 606
R2 = 602 → 14Ø4
R3 = 605 → 3Ø2
R4 = 606 → 6Ø7
R5 = 610 → 6Ø7
6ØØ = 612 → 1424
6Ø2 = 614
6Ø4 = 616
6Ø6 = 62Ø → 31Ø → 344
61Ø = 622
612 = 123456
614 = 1
616 = 1ØØØØØ
62Ø = 151515
622 = Ø33312 → Ø66624

- C BIT ENDED ON A ZERØ -



1 DETERMINE THE OCTAL CODING FOR EACH INSTRUCTION SHOWN BELOW

(a) INC %2 005202

(b) NEGB @-(3) 105453

(c) ASL (1) 006311

(d) ROR @-5Ø(5) 006075

(f) COM @(3)+ 005133

(g) CLR 25(2) 005062

(h) SWAB-(1) 000341

(I) DEC %3 005303

2. DETERMINE THE SYMBOLIC CODES FOR EACH OF THE BELOW (TWO WORD INSTRUCTIONS CONTAIN TWO ENTRIES)

(a) 1Ø6113 ROLB (3)

(b) ØØ5263 INC -72(R3)
1777Ø6

(c) ØØ5Ø32 CLR @(3)+

(d) ØØ57Ø6 TST %6

(e) ØØ6273 ASR @5ØØ(3)
ØØØ5ØØ

(f) 1Ø5Ø13 CLRB (3)

(g) 1Ø5731 TSTB @(1)+

3 FOR EACH OF THE EXAMPLES BELOW, INDICATE THE ADDRESS OF THE OPERAND

(a) 5ØØØ - NEG #3Ø 5ØØ2

(b) 5ØØØ - INC @#3Ø 3Ø

(c) 5ØØØ - ROR @3Ø 3ØØØ

3Ø = 3ØØØ

000071
-72

ANSWER SHEET

1 DETERMINE THE OCTAL CODING FOR EACH INSTRUCTION SHOWN BELOW

(a) INC %2 — ØØ52Ø2

(b) NEGB @-(3) — 1Ø5453

(c) ASL (1) — ØØ6311

(d) ROR @-5Ø(5) — ØØ6Ø75 / 177730

(e) COM @(3)+ — ØØ5133

(f) CLR 25(2) — ØØ5Ø62 / ØØØØ25

(g) SWAB-(1) — ØØØ341

(h) DEC %3 — ØØ53Ø3

2. DETERMINE THE SYMBOLIC CODES FOR EACH OF THE BELOW (TWO WORD INSTRUCTIONS CONTAIN TWO ENTRIES)

(a) 1Ø6113 — ROLB (3)

(b) ØØ5263 / 177706 — INC -72 (3)

(c) ØØ5Ø32 — CLR @ (2)+

(d) ØØ5706 — TST %6

(e) ØØ6273 / ØØØ5ØØ — ASR @ 5ØØ (3)

(f) 1Ø5Ø13 — CLRB (3)

(g) 1Ø5731 — TSTB @ (1)+

3 FOR EACH OF THE EXAMPLES BELOW, INDICATE THE ADDRESS OF THE OPERAND

(a) 5ØØØ - NEG #3Ø — 5ØØ2

(b) 5ØØØ - INC @#3Ø — 3Ø

(c) 5ØØØ - ROR @ 3Ø — 3ØØØ

3Ø = 3ØØØ

1002

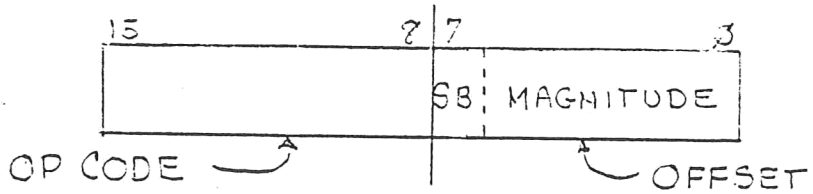
1

176775

176776

BRANCH INST

FORMAT



OPERATION

TEST CONDITION CODE BIT/BITS

TEST SATISFIED = BRANCH TO NEXT INST
TEST NOT SATISFIED = NO BRANCH

OFFSET

SIGNED 8 BIT OPERAND REPRESENTING THE DIRECTION (SIGN) AND THE DISPLACEMENT (MAGNITUDE) TO ALTER THE PROGRAM COUNTER

CALCULATION

THE OFFSET IS EXPRESSED AS A POSITIVE OR NEGATIVE WORD COUNT. WITH 8 BITS, WE HAVE THE FOLLOWING RANGE LIMITATIONS:

1. FORWARD

OFFSET = ØØØ = Ø WORDS FORWARD
OFFSET = 177 = 177_8 WORDS FORWARD

2. BACKWARD (2^s COMP)

OFFSET = 377 = 1 WORD BACKWARD
OFFSET = 2ØØ = 200 WORDS BACKWARD

1000/777 = Branch on self - Tests Unibus

CALCULATION (CONT)

THE PURPOSE OF THE OFFSET IS TO ALTER THE PC. THIS IS DONE BY SIMPLY ADDING THE OFFSET TO THE PC BUT SINCE THE PC IS ACTUALLY A BYTE COUNTER AND THE OFFSET IS IN WORD FORM, THE MACHINE WILL CHANGE THE OFFSET TO BYTE FORM AS FOLLOWS

1. SIGN EXTEND THE OFFSET INTO BITS 8 → 15
2. ADD THIS FACTOR TO ITSELF

THIS PROCEDURE SERVES NOT ONLY TO PUT THE OFFSET IN BYTE FORM BUT ALSO DEVELOPS A 16 BIT OPERAND THAT IS COMPATABLE TO BE ADDED DIRECTLY TO PC. ALSO, THE NEGATIVE OR POSITIVE IDENTITY OF THE OFFSET IS MAINTAINED.

THE MACHINE NOW ADDS THE BYTE OFFSET TO R7 AND PUTS THE SUM IN R7 AND THEN FORCES A FETCH CYCLE

BRANCH ADDRESS = UPDATED PC + 2 (OFFSET)

$$\text{OFFSET} = \frac{\text{BRANCH ADDRESS} - \text{UPDATED PC}}{2}$$

BR. ± Byte offset
BR numeric value
BR Lable

BRANCH PRACTICE

FOR EACH OF THE MACHINE CODED BRANCH INSTRUCTIONS, LIST THE BRANCH ADDRESS AND MNEMONIC CODE.

5ØØ - ØØØ4Ø5 — BRANCH ADRS = 534 — MNEMONIC = BR. +34

5ØØ - ØØØ42Ø — BRANCH ADRS = 542 — MNEMONIC = BR. +42

5ØØ - ØØØ577 — BRANCH ADRS = 400 — MNEMONIC = BR. +400

2. WRITE MACHINE CODES AND BRANCH ADDRESS FOR FOLLOWING

5ØØ - BR. +26 — BRANCH ADRS = 526 — MACH CODE ØØØ412

6ØØ - BR. +1Ø2 — BRANCH ADRS = 7Ø2 — MACH CODE ØØØ44Ø

7ØØ - BR. +262 — BRANCH ADRS = 1162 — MACH CODE ØØØ53Ø

5ØØ - BR. -6 — BRANCH ADRS = 472 — MACH CODE ØØØ774

734 - BR. -1ØØ — BRANCH ADRS = 634 — MACH CODE ØØØ737

522 - BR. -76 — BRANCH ADRS = 424 — MACH CODE ØØØ74Ø

3. WRITE BRANCH ADDRESS AND MNEMONIC CODE

5ØØ - ØØØ776 — BRANCH ADRS 426 — MNEMONIC = BR. -2

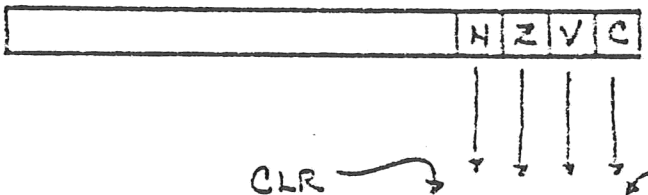
624 - ØØØ677 — BRANCH ADRS 424 — MNEMONIC = BR. -200

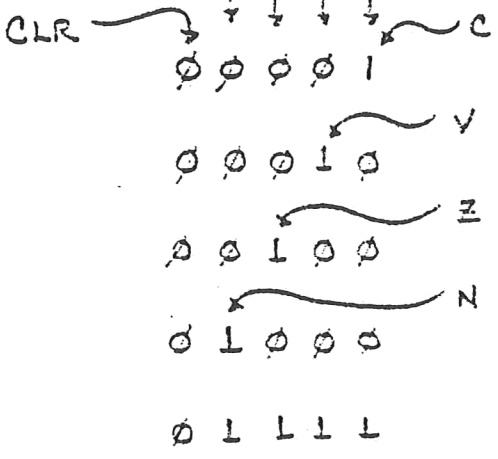
732 - ØØØ7Ø5 — BRANCH ADRS 546 — MNEMONIC = BR. -164

CONDITION CODE OPERATORS

A SET OF CONDITION CODE INSTRUCTIONS TO ALLOW THE PROGRAMMER TO EXPLICITLY ALTER THESE BITS IN THE PSW.

PSW = 177776





CLC = 000241 — 0 0 0 0 1

CLV = 000242 — 0 0 0 1 0

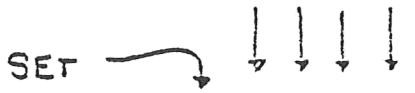
CLZ = 000244 — 0 0 1 0 0

CLN = 000250 — 0 1 0 0 0

CCC = 000257 — 0 1 1 1 1

BIT 4 = 1 (SET C/C INDICATED BY BITS <3:0>)

BIT 4 = 0 (CLR C/C INDICATED BY BITS <3:0>)



SEC = 000261 — 1 0 0 0 1

SEV = 000262 — 1 0 0 1 0

SEZ = 000264 — 1 0 1 0 0

SEN = 000270 — 1 1 0 0 0

SCC = 000277 — 1 1 1 1 1

CONSIDERING THIS FORMAT, IT IS POSSIBLE TO WRITE CONDITION CODE OPERATORS IN MACHINE LANGUAGE THAT CANNOT BE DUPLICATED IN MNUEMONICS: EX 000273 = SET N, V, C

BRANCH QUIZ

(1) CODE A BEQ INSTRUCTION THAT WILL TRANSFER CONTROL TO LOCATION 1750. THE BEQ INSTR. IS AT LOCATION 1776

1776 001764

602
-516
2 164
32

516
64
602

(2.) GIVEN

```
500/ CLR 700      005000
502/ INC 700      005200
504/ BNE. -2      001376
506/ HALT
```

11/010p

LOAD ADDRESS 500, DEPRESS START. HOW MANY TIMES WILL THE INC700 INSTRUCTION BE EXECUTED BEFORE THE MACHINE HALTS?

200000$_8$

(3) WRITE A DIAGNOSTIC ROUTINE TO CHECK THAT THE N, Z, V AND C FLAGS CAN BE ALL SET AND THEN ALL CLEARED. INDICATE A FAILURE BY HALTING AT 000600. INDICATE GOOD TEST BY HALTING AT 000601.

```
277 500 / SCC
001036 502 / BNE. +76
100035 504 / BPL. +74
102034 506 / BVC. +72
103033 510 / BCC. +70
512 / CCC. "
514 / BEQ. +64
516 / BMI. +62
520 / BVS. +60
522 / BCS. +56
524 / JMP @1602
600 / HALT
602 / HALT
```

000257
001432
100431
102430
103426
000137

54

ANSWERS TO BRANCH QUIZ

(1.) = 1776/ BEQ.-26 = ØØ1764

(2.) = $200{,}000_8$ TIMES (UNTIL RØ WRAPS AROUND)

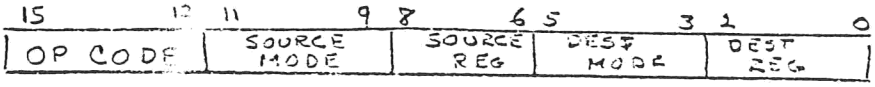
(3)

```
500 - SCC
502 - BNE.+76
504 - BPL.+74
506 - BVC.+72
510 - BCC.+7Ø
512 - CCC
514 - BEQ.+64
516 - BMI.+62
520 - BVS.+6Ø
522 - BCS.+56
524 - JMP @#6Ø2
 }

6ØØ - HALT
6Ø2 - HALT
```

DOUBLE OPERAND INSTRUCTIONS

FORMAT



OP SS DD

ALL 8 GPR MODES AND 4 PC MODES CAN BE USED IN BOTH SOURCE AND DEST
ANY DOP CAN BE EXECUTED IN 144 DIFFERENT WAYS.

GENERAL DOP

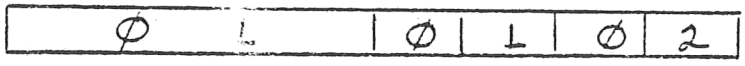
MOV(B)
CMP(B)
ADD
SUB

LOGICAL

BIT(B)
BIC(B)
BIS(B)

MOVE INST

MOV %1, %2



MOVE THE CONTENTS OF SOURCE FIELD INTO DEST FIELD

SOURCE FIELD REMAINS SAME
DEST FIELD LOST

- OVER FOR EXAMPLES

MOVE INSTRUTION EXAMPLES

(1) REG → REG

```
500 - MOV %3, %4
```

(2) REG → MEM

```
500 - MOV %2, @#1000
502 - 1000
```

(3) MEM → REG

```
500 - MOV @#2000, %4
502 - 2000
```

(4) MEM → MEM

```
500 - MOV @#3000, @#4000
502 - 3000
504 - 4000
```

(5) DATA → REG

```
500 - MOV #6, %5
502 - 6
```

(6) DATA → MEM

```
500 - MOV #32, @#6000
502 - 32
504 - 6000
```

(7) DATA → PSW

```
500 - MOV #2, @#177776
502 - 2
504 - 177776
```

(8) BYTE → REG

```
500 - MOVB @#5000, %3
502 - 5000
```

← MOVB * DM0

THIS IS SPECIAL CASE: THE BYTE WILL ALWAYS BE INSERTED IN THE LOW BYTE POSITION OF THE DEST REG AND THE HI BYTE OF THAT REGISTER WILL BE REPLICATED WITH BIT 7, WHICH IS SIGN OF LO BYTE.

DOP INSTRUCTIONS USING PC MODE 6 OR 7

500 - MOV A1, A2
502 - X1
504 - X2

WHEN USING PC MODE 6 WITH A DOUBLE OPERAND INST THE PROGRAMMER SIMPLY LISTS THE SOURCE AND DEST ADDRESSES AND DOES NOT CONCERN HIMSELF WITH THE VALUES THAT MUST BE AT PC+2 AND PC+4.

EXAMPLE

500 - MOV 1000, 2000
502 -
504 -

THE ASSEMBLER WILL CALCULATE THE CORRECT VALUE TO GO INTO 502 AND 504

THE VALUE THAT MUST BE IN 502 IS CALCULATE THUS

X1 = A1 - UPDATED PC = (PC+4)
X1 = 1000 - 504
X1 = 274

THE VALUE THAT MUST BE IN 504:

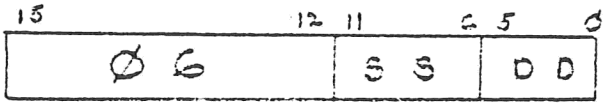
X2 = A2 - UPDATED PC = (PC+6)
X2 = 2000 - 506
X2 = 1272

RESULT:

500 - MOV 1000, 2000
502 - 274
504 - 1272

with 11/40 :
Can't move same register to address specified by register w/ auto incr.
ex. MOV R0, (R0)+

ADD SS, DD

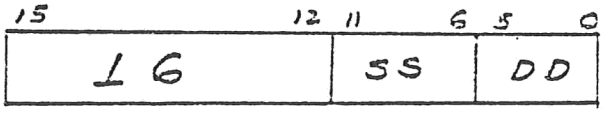


DST ← SRC PLUS DST

ADD THE CONTENTS OF SOURCE TO THE CONTENTS OF DESTINATION AND DELIVER RESULTS TO DESTINATION
OVERFLOW IS POSSIBLE:

OVERFLOW: + PLUS + = − (SUM) OR − PLUS − = + (SUM)

SUB SS, DD

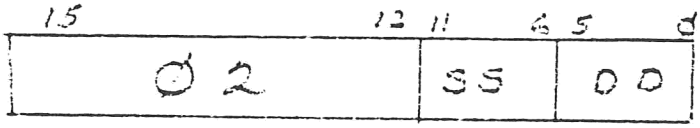


DST ← DST − SRC

SUBTRACT THE CONTENTS OF SOURCE FROM CONTENTS OF DESTINATION AND DELIVER RESULTS TO DESTINATION
OVERFLOW IS POSSIBLE:

OVERFLOW: + MINUS − = − (DIFFERENCE)
OR
− MINUS + = + (DIFFERENCE)

CMP SS, DD



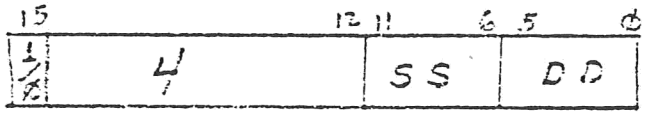
SET PSW ON RESULT ← SRC − DST

THIS IS SIMILIAR TO A SUBTRACT INSTRUCTION EXCEPT DEST IS SUBTRACTED FROM SOURCE AND THE RESULT IS NOT DELIVERED TO DEST.

THIS INSTRUCTION COMPARES ONE OPERAND AGAINST ANOTHER AND SETS CONDITION CODES TO REFLECT THIS COMPARE

BIC INST
BIT CLEAR

BIC[B]



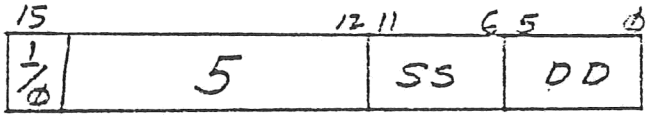
CLEAR ALL BITS IN THE DESTINATION FOR CORRESPONDING "1" BITS IN THE SOURCE. IF BIT IS CLEAR, LEAVE IT CLEARED. SOURCE IS UNCHANGED

EXAMPLE: CLEAR PSW CONDITION CODES

```
500 - BIC #17, @#177776
502 - 17
504 - 177776
```

BIS INST
BIT SET

BIS[B]



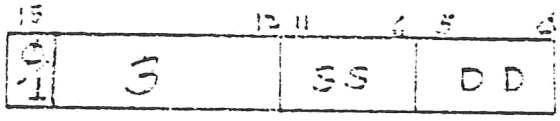
SET ALL BITS IN THE DESTINATION FOR CORRESPONDING "1" BITS IN THE SOURCE. IF BIT IS SET, LEAVE IT SET. SOURCE IS UNCHANGED

EXAMPLE: SET "T" BIT IN PSW WITHOUT AFFECTING C/C

```
500 - BIS #20, @#177776
502 - 20
504 - 177776
```

BIT INST
BIT TEST

BIT[B]



LOGICAL AND BETWEEN SOURCE AND DESTINATION. RESULT IS NOT DELIVERED BUT IS USED TO MANIPULATE THE N & Z BITS IN PSW AND THEREFORE REFLECT THE RESULT OF TEST. NIETHER SOURCE NOR DEST ARE ALTERED BY THIS INST.

N = 1 IF RESULTANT BIT 15 IS SET
Z = 1 IF RESULTANT HAS NO BITS SET

EXAMPLE: TEST LOC 1000 TO SEE IF BITS 0, 1 AND 5 ARE ALL CLEAR

```
500 - BIT #43,@ #1000
502 - 43
504 - 1000
```

IF BITS 0, 1 AND 5 ARE ALL CLEAR Z = 1

IF ANY OF THOSE BITS ARE SET Z = 0

ALL OTHER BITS IN LOC 1000 ARE NOT TESTED

EXECUTE THE FOLLOWING KEEPING TRACK OF LOC 3ØØØ

```
5ØØ - MOV #3ØØØ, %Ø
5Ø2 - 3ØØØ
5Ø4 - CLR(Ø)  (3000)=0
5Ø6 - COM (Ø) (3000)=177777
51Ø - INC (Ø) (3000)= 0
512 - DEC (Ø) (3000)=177777
514 - NEG (Ø) (3000)= 1
516 - TST (Ø) (3000)= 1
52Ø - BEQ .+46  Z,N ≠1
522 - ROL (Ø)   000002 =(3000)
524 - SWAB(Ø)   001000 =(3000)
526 - ROR (Ø)   0  000 = 3000
53Ø - ASL (Ø)
532 - ADD #2, (Ø)
534 - 2
536 - ASR (Ø)
54Ø - MOVB #377, (Ø)
542 - 377
544 - ADD #177ØØØ, (Ø)
546 - 177ØØØ
55Ø - SUB #1ØØ777, (Ø)
552 - 1ØØ777
554 - CMP #77ØØØ, (Ø)
556 - 77ØØØ
56Ø - BEQ .+6
562 - JMP @#5ØØ
564 - 5ØØ
566 - HALT
```

SPOT CHECK ALL INSTR COVERED THUS FAR

3ØØØ = _____

ANSWER SHEET

EXECUTE THE FOLLOWING KEEPING TRACK OF LOC 3ØØØ

```
5ØØ - MOV # 3ØØØ, %Ø ⟶ R Ø = 3ØØØ
5Ø2 - 3ØØØ
5Ø4 - CLR(Ø) 3ØØØ = Ø
5Ø6 - COM (Ø) 3ØØØ = 177777
51Ø - INC (Ø) 3ØØØ = ØØØØØØ
512 - DEC (Ø) 3ØØØ = 177777
514 - NEG (Ø) 3ØØØ = ØØØØØ1
516 - TST (Ø) Z,N BOTH CLEAR
52Ø - BEQ .+46  NO BRANCH
522 - ROL (Ø) 3ØØØ = ØØØØØ2
524 - SWAB(Ø) 3ØØØ = ØØ1ØØØ [C BIT = Ø]
526 - ROR (Ø) 3ØØØ = ØØØ4ØØ [C BIT = Ø]
53Ø - ASL (Ø) 3ØØØ = ØØ1ØØØ
532 - ADD #2, (Ø) 3ØØØ = ØØ1ØØ2
534 - 2
536 - ASR (Ø) 3ØØØ = ØØØ4Ø1
54Ø - MOVB #377, (Ø) 3ØØØ = ØØØ777
542 - 377
544 - ADD # 177ØØØ, (Ø) 3ØØØ = 177777
546 - 177ØØØ
55Ø - SUB #1ØØ777, (Ø) 3ØØØ = Ø77ØØØ
552 - 1ØØ777
554 - CMP # 77ØØØ, (Ø) [Z = 1]
556 - 77ØØØ
560 - BEQ .+6 YES
562 - JMP @ # 5ØØ
564 - 5ØØ
566 - HALT
```

SPOT CHECK ALL INSTR COVERED THUS FAR

3ØØØ = Ø77ØØØ

STUDY GUIDE# ADDRESSING MODES

GIVEN: LOCATION, CONTENT , LOCATION, CONTENT , LOCATION, CONTENT

R3= 3000	3000= 300	300= 30
R4= 4000	4000= 400	400= 40
R7= 7000	6504= 654	654= 64
1000= 100	100= 10	10= 0
1234= 124	124= 14	14= 4
2776= 276	276= 26	26= 6
3500= 350	350= 30	30= 0
4100= 410	410= 40	40= 0

FIND THE FOLLOWING:

	MODE	EFFECTIVE ADRS.	OPERAND	CONTENT OF SPECIFIED REG	PC	NMW (Next memory word)
1. (3)+	2	3000	300	3002	7002	—
2. @-(3)	5	2776	26	2776	7002	—
3. #4	0	4000	4000	—	7002	—
4. 100(4)	6	4100	410	4000	7004	100
5. @(3)+	3	300	30	3002	7002	—
6. 1234 (Lable)	67	1234	124	PC=7004	7004	172230
7. (4)	1	4000	400	4002	7002	—
8. -(3)	4	2776	276	2776	7002	—
9. @100(4)	7	4100	40	4100	7004	100
10. #1000	27	7000	1000	PC=7004	7004	1000
11. -300(4)	6	3500	350	4000	7004	177500
12. @6504	77	6504	64	7004	7004	171274
13. @#1000	37	1000	100	7004	7004	1000
14. @-300(4)	6	350	30	4000	7004	—
15. #3	0	R3	3000	—	7002	—

STUDY GUIDE# ADDRESSING MODES

GIVEN:

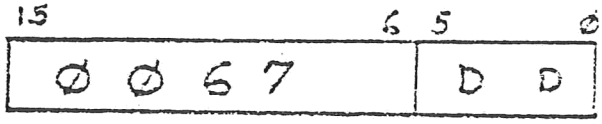
LOCATION	CONTENT	LOCATION	CONTENT	LOCATION	CONTENT
R3=	3000	3000=	300	300=	30
R4=	4000	4000=	400	400=	40
R7=	7000	6504=	654	654=	64
1000=	100	100=	10	10=	0
1234=	124	124=	14	14=	4
2776=	276	276=	26	26=	6
3500=	350	350=	30	30=	0
4100=	410	410=	40	40=	0

FIND THE FOLLOWING:

	MODE	EFFECTIVE ADRS.	OPERAND	CONTENT OF SPECIFIED REG	PC	NMW
1. (3)+	2	[illegible]	[illegible]	[illegible]	[illegible]	—
2. @-(3)	5	276	26	2776	7002	—
3. #4	0	[illegible]	[illegible]	—	[illegible]	—
4. 100(4)	6	[illegible]	[illegible]	[illegible]	[illegible]	100
5. @(3)+	3	[illegible]	[illegible]	[illegible]	[illegible]	—
6. 1234	PC 6	1234	124	PC = 7234	7234	177134
7. (4)	1	[illegible]	[illegible]	[illegible]	[illegible]	—
8. -(3)	4	2776	276	2776	[illegible]	—
9. @100(4)	7	4100	40	4100	[illegible]	114
10. #1000	PC 2	[illegible]	[illegible]	PC = [illegible]	[illegible]	1000
11. -300(4)	6	[illegible]	[illegible]	[illegible]	[illegible]	177534
12. @6504	PC 7	654	64	PC = 7314	[illegible]	177564
13. @#1000	PC 3	1000	100	PC = 7034	7134	1000
14. @-300(4)	7	354	30	4101	[illegible]	177534
15. #3	0	A3	3000	—	[illegible]	—

SXT
SIGN EXTEND

SXT DD



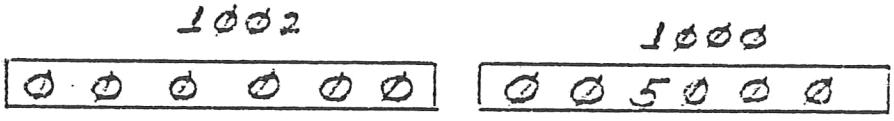
SIGN EXTEND THE PSW N BIT INTO THE DESTINATION

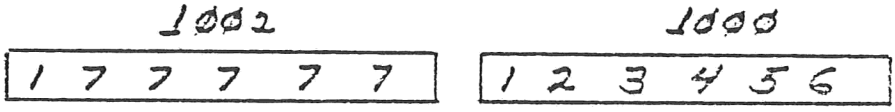
USED TO EXTEND 16 BIT OPERANDS TO 32 BIT OPERANDS TO FACILITATE MULTI PRECISION ARITHMETIC

METHOD:

```
500 - TST @ #1000
502 - SXT @ #1002
```

AFTER EXECUTION OF THE ABOVE THE NUMBER AT LOC 1000 HAS ITS SIGN REPLICATED THROUGHOUT LOC 1002

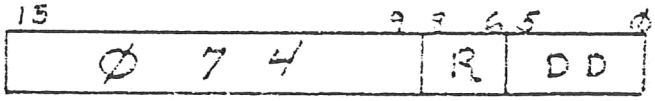




NOTE THAT THE PROGRAMMER NEED NOT KNOW WHAT THE SIGN OF THE NUMBER TO BE EXTENDED IS.

XOR
EXCLUSIVE OR

XOR R, DD



EXCLUSIVE OR (HALF ADD) THE CONTENTS OF R AND THE DESTINATION RETURN THE HALF ADD SUM TO DESTINATION. (R) IS UNCHANGED

METHOD:

5ØØ- XOR 3, (1)

R1: 1ØCØ
1ØCØ = 12ØØ43

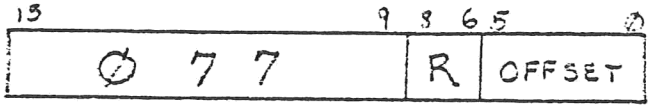
R3: Ø42556

R3 = Ø 1ØØ Ø1Ø 1Ø1 1Ø1 11Ø
1ØØØ = 1 Ø1Ø ØØØ ØØØ 1ØØ Ø11
XOR→1ØØØ 1 11Ø Ø1Ø 1Ø1 ØØ1 1Ø1

IN EFFECT, THIS INSTRUCTION ALLOWS SELECTIVE COMPLEMENTATION OF BITS IN DESTINATION FOR ALL "1" BITS SET IN R.

SOB
SUBTRACT ONE & BRANCH

SOB R, A



DECREMENT REG R BY 1 AND TEST REG = 0

REGISTER IS EQUAL TO ZERO = NO BRANCH
REGISTER NOT EQUAL TO ZERO = BRANCH

CALCULATING BRANCH ADRS:

THE SIX BIT OFFSET FOR THIS INSTRUCTION IS UNSIGNED AND IS CONSIDERED TO BE ALWAYS POSITIVE

BRANCH ADDRESS = UPDATED PC − 2 (OFFSET)

YOU WILL NOTE THAT APPLICATION OF THE ABOVE FORMULA MUST CAUSE THE PC TO BE REDUCED IN VALUE. THE SOB INST CAN ONLY BRANCH BACKWARD AND NEVER FORWARD.

CALCULATE OFFSET

$$\text{OFFSET} = \frac{\text{UPDATED PC} - A}{2}$$

EXAMPLE:

1500 - SOB 3, 1400 R3 = 1

NO BRANCH
R3 = 0

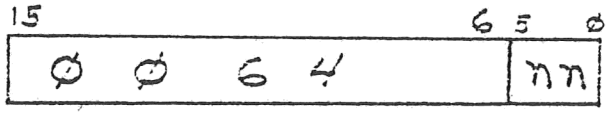
1500 - SOB 3, 1400 R3 = 2

BRANCH TO 1000
R3 = 1

MARK INSTRUCTION

THE MARK INSTRUCTION FACILITATES STACK CLEANUP PROCEDURES WHEN RETURNING FROM A SUB-ROUTINE. STACK CLEANUP SIMPLY MEANS RESETTING THE STACK POINTER TO THE VALUE IT CONTAINED JUST PRIOR TO JUMPING TO THE SUB-ROUTINE. ALSO THE MARK INSTRUCTION ALLOWS THE PROGRAMMER TO USE THE STACK AREA TO PASS PARAMETERS TO HIS SUB-ROUTINE. THE MARK INSTRUCTION IS UNIQUE IN THAT IT IS USUALLY EXECUTES OFF OF THE STACK AND ALSO, IT HAS GPR R5 DEDICATED TO IT.

MARK



nn = PARAMETER COUNT

EXECUTION:

1. SP ← UPDATED PC + 2(nn)
2. PC ← R5
3. R5 ← (R6)+
4. EXECUTE FROM NEW PC

```
START ──► 5000 - MOV %5, -(6)
          5002 - MOV #20, -(6)
          5004 - 20
          5006 - MOV #30, -(6)
          5010 - 30
          5012 - MOV #40, -(6)
          5014 - 40
          5016 - MOV #006403, -(6)
          5020 - 006403
          5022 - MOV %6, %5
          5024 - JSR 7, @ #6000
          5026 - 6000
          5030 - HALT
                 ~
                 ↓

          6000 - HALT
          6002 - RTS 5
```

ASSUME

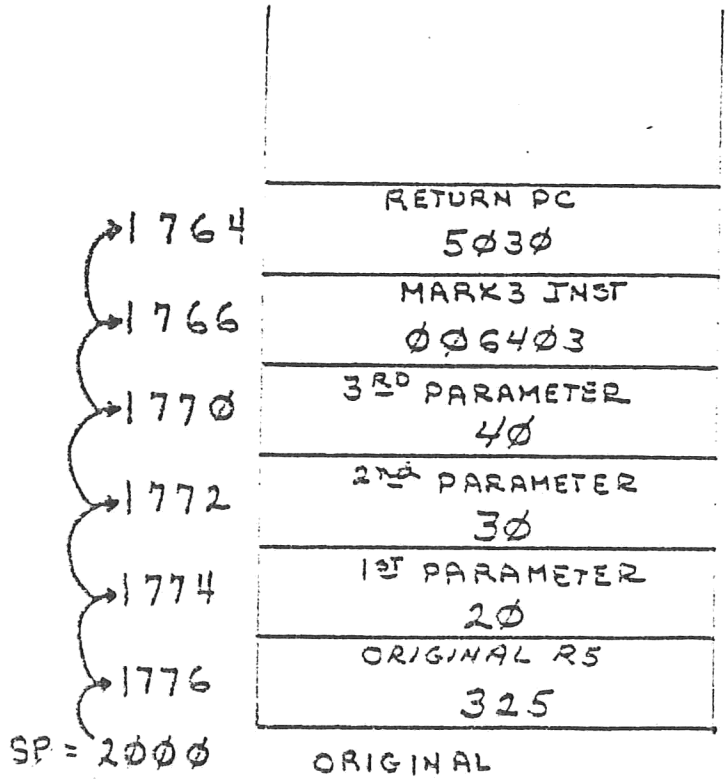
R5: 325
R6: 2000

WHEN PROGRAM HALTS

1. STACK IS AS SHOWN
2. R5 = 1766
3. R6 = 1764
4. R7 (PC) = 6002

NOTE:

R6 POINTS TO RETURN PC
R5 POINTS TO MARK INST.



- PRESS CONTINUE -

WHEN CONTINUE IS PRESSED, THE RTS JUST WILL EXECUTE AS FOLLOWS:

1. TRANSFER CONTENTS OF R5 → PC — PC = 1766
2. POP (6)+ FROM STACK TO R5 — R5 = 5ØEØ, R6 = 177Ø
3. EXECUTE FROM NEW PC

PC = 1766 WHICH IS THE ADDRESS OF THE MARK INSTRUCTION. MARK EXECUTES AS FOLLOWS:

1. SP ← UPDATED PC + 2 (ØN)

   SP ← 177Ø + 2 (3)

   SP = 1776

2. PC ← R5

   PC ← 5Ø3Ø

3. R5 ← (R6)+

   R5 ← 325
   R6 = 2ØØØ

4. EXECUTE (NOTE R5 & R6 = ORIGINAL VALUES)

   HALT.

IMPORTANT POINTS TO REMEMBER

1. A DEFINITE SEQUENCE OF INSTRUCTIONS MUST BE EXECUTED TO PROPERLY USE THE MARK
2. THE MARK INSTRUCTION MUST BE ON THE STACK
3. R[5] MUST POINT TO THE MARK
4. AN RTS SPECIFYING R5 MUST BE USED FOR RETURN
5. THE NET RESULT OF USING THE MARK
   - A. THREE WORDS FEWER ARE REQUIRED IN EACH SUBROUTINE CALL, REQUIRING LESS MEMORY
   - B. SIX BUS CYCLES (MEMORY ACCESSES) ARE SAVED FROM EACH CALL GIVING FASTER CODE.
   - C. ONE LESS WORD IS PUSHED ONTO THE STACK FOR EACH LEVEL OF NESTING, ALSO REDUCING CORE REQUIREMENTS

TRAPS
&
INTERRUPTS
&
I/O

DAY 4

Types of Traps

Types:

1. Programmer Initiated

   The programmer can cause a trap sequence to take place thru use of the following

   A. TRAP instruction
   B. EMT instruction
   C. IOT instruction
   D. BPT instruction
   C. Setting "T" Bit in PSW

2. Machine Initiated

   Should certain conditions exist within the hardware during execution of an instruction the hardware will initiate a trap sequence.

   A. Stack Overflow yellow zone = 400 Red zone = 340
   B. Odd Address Error
   C. Bus Time out
   D Power Fail/Power up
   E. Illegal Instruction
   F Reserved Instruction

TRAP VECTORS
(ASSIGNED)

4 - NEW PC POINTING TO ROUTINE FOR *ILLEGAL INST OR *ERROR
6 - NEW PSW

10 - NEW PC POINTING TO ROUTINE FOR RESERVED INSTRUCTIONS
12 - NEW PSW

14 - NEW PC POINTING TO ROUTINE FOR "T" BIT TRAP
16 - NEW PC

20 - NEW PC POINTING TO ROUTINE FOR IOT INST
22 - NEW PSW

24 - NEW PC POINTING TO ROUTINE FOR PWR FAIL TRAP
26 - NEW PSW

30 - NEW PC POINTING TO ROUTINE FOR EMT INST
32 - NEW PSW

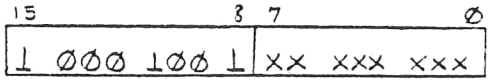
34 - NEW PC POINTINE TO ROUTINE FOR TRAP INST
36 - NEW PSW

* NOTE: ILLEGAL INST. = NON EXISTANT CODE OR SMØ ON JMP INST.

ERROR = 1. ODD ADDRESS ERROR
2. NO SSYN RESPONSE TO MSYN (BUS TIME OUT)
3. STACK OVERFLOW RED OR YELLOW ZONE

PROGRAMMER INITIATED TRAPS

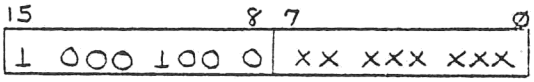
TRAP INST



= 104400 = TRAP0
THRU
104777 = TRAP377
} ALL OF THESE ARE TRAPS

1. PUSH PSW TO STACK
2. PUSH PC TO STACK
3. CONTENTS OF 36 TO PSW
4. CONTENTS OF 34 TO PC

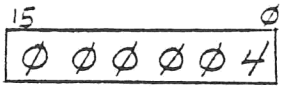
EMT INST



= 104000 = EMT0
THRU
104377 = EMT377
} ALL OF THESE ARE EMT

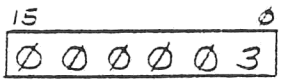
1 PUSH PSW TO STACK
2. PUSH PC TO STACK
3. CONTENTS OF 32 TO PSW
4. CONTENTS OF 30 TO PC

IOT INST



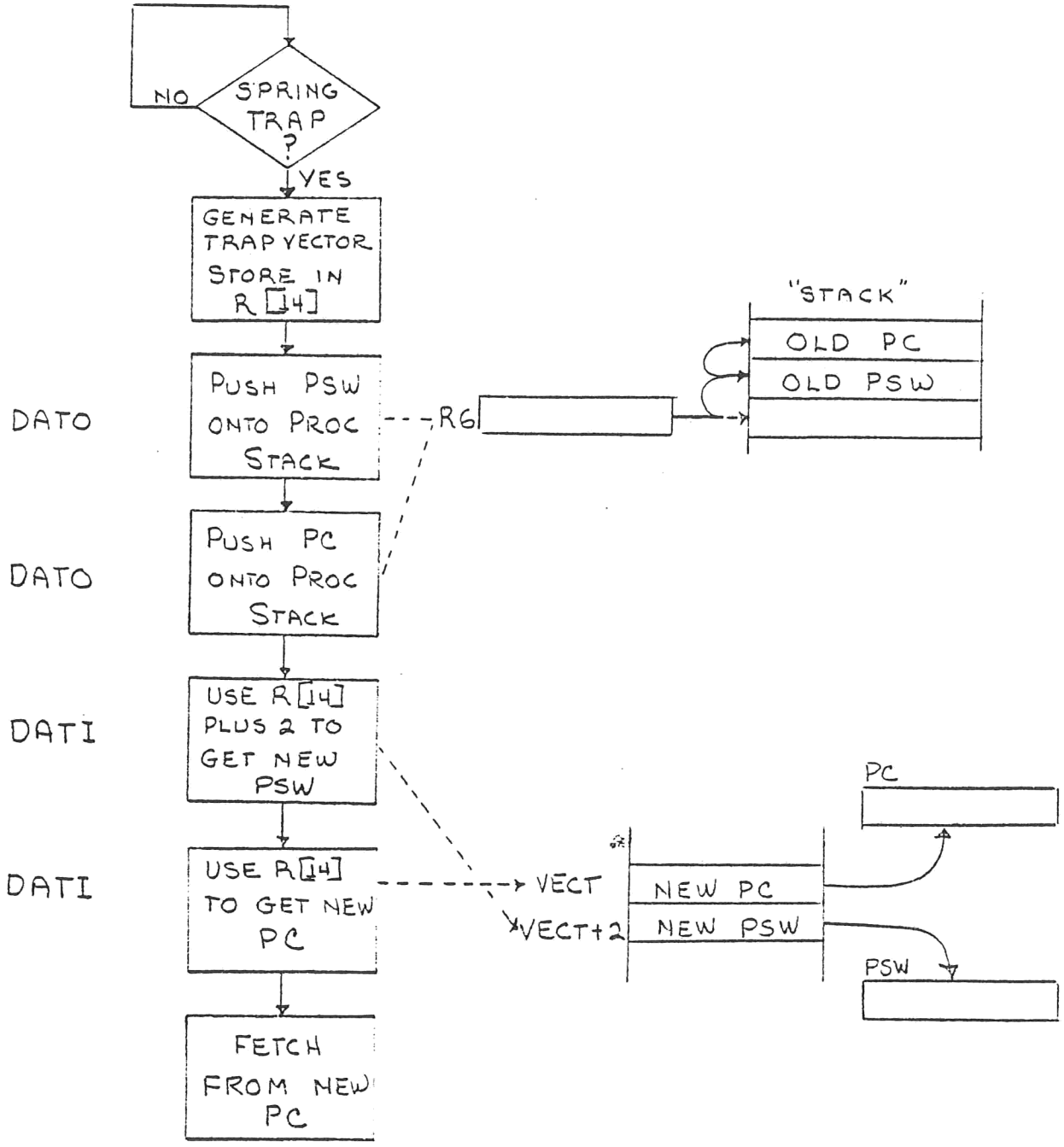
1 PUSH PSW TO STACK
2 PUSH PSW TO STACK
3 CONTENTS OF 22 TO PSW
4 CONTENTS OF 20 TO PC

BPT INST



1. PUSH PSW TO STACK
2. PUSH PC TO STACK
3. CONTENTS OF 16 TO PSW
4. CONTENTS OF 14 TO PC

TRAP FLOW CHART



NOTE: ALL TRAPS OPERATE THE SAME, THE ONLY DIFFERENCE IS:

1. CAUSE OF TRAP
2. VECTOR ADDRESS

EXAMPLE:

ILLEGAL INST TRAP

LOAD ADDRESS = 500 / START

```
500 - MOV #600, %6
502 - 600
504 - MOV #3000, @#4
506 - 3000
510 - 4
512 - MOV #2, @#6
514 - 2
516 - 6
520 - JMP %L   * ILLEGAL *
522 - HALT

3000 - HALT
```

WHEN PROGRAM HALTS LIST WHAT SHOULD BE CONTENTS OF FOLLOWING

R [VECT] =

R6 =

Loc 574 =

Loc 576 =

R7 =

PSW =

ANSWER SHEET

EXAMPLE:

ILLEGAL INST TRAP

LOAD ADDRESS = 500 / START

```
500 - MOV #600, %6
502 - 600
504 - MOV #3000, @#4
506 - 3000
510 - 4
512 - MOV #2, @#6
514 - 2
516 - 6
520 - JMP %L   * ILLEGAL *
522 - HALT

3000 - HALT
```

WHEN PROGRAM HALTS LIST WHAT SHOULD BE CONTENTS OF FOLLOWING

R [VECT] = 4

R6 = 574

Loc 574 = OLD PC = 522

Loc 576 = OLD PSW = 0

R7 = 3002

PSW = NEW PSW = 2

EXAMPLE:

ERROR TRAP

(ODD ADDRESS)

LOAD ADDRESS = 500/START

```
500 - MOV #1000, %6
502 - 1000
504 - MOV #1500, @#4
506 - 1500
510 - 4
512 - MOV #15, @#6
514 - 15
516 - 6
520 - MOV #3, @#5   * ODD ADDRESS ERROR *
522 - 3
524 - 5
526 - HALT

1500 - HALT
```

WHEN PROGRAM HALTS, WHAT SHOULD BE THE CONTENTS OF FOLLOWING!

R[VECT] = 4

R6 =

Loc 774 = OLD PC 526

Loc 776 = OLD PSW

R7 = 502

PSW = 15

ANSWER SHEET
EXAMPLE:

ERROR TRAP

LOAD ADDRESS = 500/START

(ODD ADDRESS)

```
500 - MOV #1000, %6
502 - 1000
504 - MOV #1500, @#4
506 - 1500
510 - 4
512 - MOV #15, @#6
514 - 15
516 - 6
520 - MOV #3, @#5   * ODD ADDRESS ERROR *
522 - 3
524 - 5
526 - HALT

1500 - HALT
```

WHEN PROGRAM HALTS, WHAT SHOULD BE THE CONTENTS OF FOLLOWING:

R[VECT] = 4

R6 = 774

Loc 774 = OLD PC = 526

Loc 776 = OLD PSW = 0

R7 = 1502

PSW = NEW PSW = 15

EXAMPLE:

LOAD ADDRESS = 500/START/ASSUME 4K CORE

ERROR TRAP (BUS TIME OUT)

```
500 - MOV #1500,%6
502 - 1500
504 - MOV #5000,@#4
506 - 5000
510 - 4
512 - CLR @#6
514 - 6
516 - MOV #22,@#37776   * 37776 = NON EXISTANT CORE 4K *
520 - 22
522 - 37776
524 - HALT

5000 - HALT
```

WHEN PROGRAM HALTS, WHAT SHOULD BE CONTENTS OF FOLLOWING:

R[VECT] = 4

R6 = 1474

Loc 1474 = 524 Updated PC

Loc 1476 = OLD PSW

R7 = 5002

PSW = 0

ANSWER SHEET

EXAMPLE:

ERROR TRAP (BUS TIME OUT)

LOAD ADDRESS = 500/START/ASSUME 4K CORE

```
500 - MOV #1500,%6
502 - 1500
504 - MOV #5000,@#4
506 - 5000
510 - 4
512 - CLR @#6
514 - 6
516 - MOV #22,@#37776   * 37776 = NON EXISTANT CORE 4K *
520 - 22
522 - 37776
524 - HALT

5000 - HALT
```

WHEN PROGRAM HALTS, WHAT SHOULD BE CONTENTS OF FOLLOWING:

R[VECT] = 4

R6 = 1474

Loc 1474 = OLD PC = 524

Loc 1476 = OLD PSW = 0

R7 = 5002

PSW = NEW PSW = 0

EXAMPLE

LOAD ADDRESS = 500/START ASSUME 4K CORE

DOUBLE BUSS ERROR

(DUBBER)

(ODD ADDRESS AND BUS TIME OUT)

```
500 - MOV #1475, %6
502 - 1475
504 - MOV #3000, @#4
506 - 3000
510 - 4
512 - CLR @#6
514 - 6
516 - INC @#37776
520 - 37776
522 - HALT

3000 - HALT
```

NOTE: DOUBLE BUSS ERROR IN THIS PROGRAM OCCURS BECAUSE WHILE WE ARE SERVICING A BUS TIME OUT WE ENCOUNTER AN ODD ADDRESS ERROR

WHEN PROGRAM HALTS, WHAT SHOULD BE CONTENTS OF FOLLOWING:

R[VECT] =

R6 (SP) =

Loc 2 =

Loc 0 =

R7 (PC) =

PSW =

ANSWER SHEET

EXAMPLE

LOAD ADDRESS = 500/START ASSUME 4K CORE

DOUBLE BUSS ERROR

(DUBBER)

(ODD ADDRESS AND BUS TIME OUT)

```
500 - MOV #1475, %6
502 - 1475
504 - MOV #3000, @#4
506 - 3000
510 - 4
512 - CLR @#6
514 - 6
516 - INC @#37776
520 - 37776
522 - HALT

3000 - HALT
```

NOTE: DOUBLE BUSS ERROR IN THIS PROGRAM OCCURS BECAUSE WHILE WE ARE SERVICING A BUS TIME OUT WE ENCOUNTER AN ODD ADDRESS ERROR.

WHEN PROGRAM HALTS, WHAT SHOULD BE CONTENTS OF FOLLOWING:

R[VECT] = 4

R6 (SP) = 0

LOC 2 = OLD PSW = 0

LOC 0 = OLD PC = 522

R7 (PC) = 3002

PSW = NEW PSW = 0

EXAMPLE:

STACK OVERFLOW

(YELLOW ZONE)

LOAD ADDRESS = 500/START

```
500 - MOV #400, %6
502 - 400
504 - MOV #3000, @#4
506 - 3000
510 - 4
512 - CLR @#6
514 - 6
516 - MOV #3, -(6)
520 - 3
522 - HALT

3000 - HALT
```

WHEN PROGRAM HALTS, WHAT SHOULD BE CONTENTS OF FOLLOWING:

R[VECT] =

R6(SP) =

Loc 376 =

Loc 374 =

Loc 372 =

R7(PC) =

PSW =

ANSWER SHEET
EXAMPLE:

STACK OVERFLOW

(YELLOW ZONE)

LOAD ADDRESS = 500/START

```
500 - MOV #400, %6
502 - 400
504 - MOV #3000, @#4
506 - 3000
510 - 4
512 - CLR @#6
514 - 6
516 - MOV #3, -(6)
520 - 3
522 - HALT

3000 - HALT
```

WHEN PROGRAM HALTS, WHAT SHOULD BE CONTENTS OF FOLLOWING:

R[VECT] = 4

R6(SP) = 372

Loc 376 = 3

Loc 374 = OLD PSW = 0

Loc 372 = OLD PC = 522

R7(PC) = 3002

PSW = NEW PSW = 0

EXAMPLE:

STACK OVERFLOW

(RED ZONE)

LOAD ADDRESS = 500 / START

```
500 - MOV #340, %6
502 - 340
504 - MOV #1000, @#4
506 - 1000
510 - 4
512 - CLR @#6
514 - 6
516 - MOV #15, -(6)
520 - 15
522 - HALT

1000 - HALT
```

WHEN PROGRAM HALTS, WHAT SHOULD BE CONTENTS OF FOLLOWING:

R[VECT] =

R6 (SP) =

Loc 336 =

Loc 2 =

Loc 0 =

R7 (PC) =

PSW =

ANSWER SHEET

EXAMPLE:

STACK OVERFLOW

(RED ZONE)

LOAD ADDRESS = 500 / START

```
500 - MOV #340, %6
502 - 340
504 - MOV #1000, @#4
506 - 1000
510 - 4
512 - CLR @#6
514 - 6
516 - MOV #15, -(6)
520 - 15
522 - HALT

1000 - HALT
```

WHEN PROGRAM HALTS, WHAT SHOULD BE CONTENTS OF FOLLOWING:

R[VECT] = 4

R6 (SP) = 0

Loc 336 = UNCHANGED

Loc 2 = OLD PSW = 0

Loc 0 = OLD PC = 522

R7 (PC) = 1002

PSW = NEW PSW = 0

EXAMPLE

STACK OVERFLOW

(RED ZONE CAUSE BY YELLOW ZONE)

LOAD ADDRESS = 500/START

```
500 - MOV #342, %6
502 - 342
504 - MOV #3000, @#4
506 - 3000
510 - 4
512 - CLR @#6
514 - 6
516 - MOV #50, -(6)
520 - 50
522 - HALT

3000 - HALT
```

WHEN PROGRAM HALTS, WHAT SHOULD BE CONTENTS OF FOLLOWING:

R[VECT] =

R6 (SP) =

Loc 340 =

Loc 336 =

Loc 2 =

Loc 0 =

R7 (PC) =

PSW =

ANSWER SHEET

EXAMPLE

STACK OVERFLOW

(RED ZONE CAUSE BY YELLOW ZONE)

LOAD ADDRESS = 500/START

```
500 - MOV #342, %6
502 - 342
504 - MOV #3000, @#4
506 - 3000
510 - 4
512 - CLR @#6
514 - 6
516 - MOV #50, -(6)
520 - 50
522 - HALT

3000 - HALT
```

WHEN PROGRAM HALTS, WHAT SHOULD BE CONTENTS OF FOLLOWING:

R[VECT] = 4

R6 (SP) = 0

Loc 340 = 50

Loc 336 = UNCHANGED

Loc 2 = OLD PSW = 0

Loc 0 = OLD PC = 522

R7 (PC) = 3002

PSW = NEW PSW = 0

EXAMPLE:

TRAP CATCHER INITIALIZE

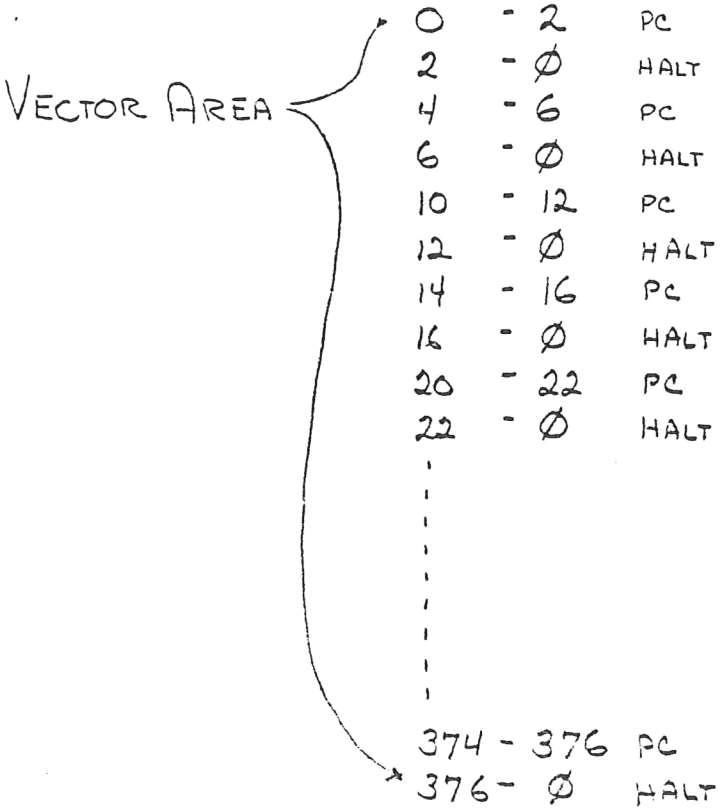
LOAD ADDRESS = 500/START

```
500 - MOV #177776, %0
502 - 177776
504 - CLR %0L
506 - ADD #4, %0
510 - 4
512 - MOV %0, (1)+
514 - CLR (L)+
516 - CMP #400, %0L
520 - 400
522 - BNE. -14
524 - HALT
```

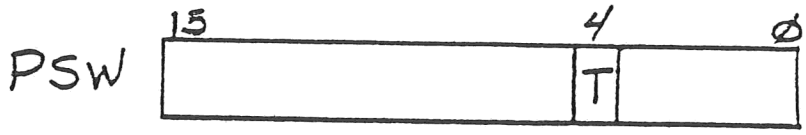
PURPOSE:

TO SET UP VECTOR AREA WITH SERIES OF ADDRESSES THAT POINT TO HALTS. SHOULD MULTIPLE TRAPS OCCUR WE CAN CAUSE A HALT TO OCCUR THAT WILL INDICATE WHAT TYPE OF TRAP IT WAS

RESULT:



OPERATION OF "T" BIT TRAP



IF BIT 4 OF PSW IS SET, A TRAP WILL OCCURR AS FOLLOWS:

1. PSW → STACK AT -(R6)
2. PC → STACK AT -(R6)
3. CONTENTS OF LOC 16 → PSW
4. CONTENTS OF LOC 14 → PC

RULES FOR "T" BIT TRAPPING

1. THE T BIT CAN ONLY BE SET BY AN IMPLICIT OPERATION FROM THE STACK OR VECTOR

EXAMPLE:

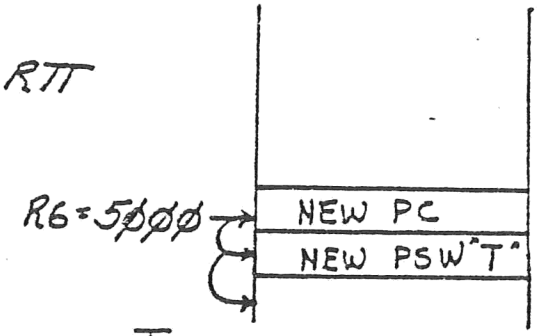
1. WHENEVER THE MACHINE RETRIEVES A NEW PROCESSOR STATUS WORD FROM ANY VECTOR IT IS POSSIBLE THAT THE NEW STATUS WORD COULD SET THE "T" BIT.

5ØØ - EMT
5Ø2

3Ø - NEW PC
32 - NEW PSW "T"

2. WHENEVER THE MACHINE RETRIEVES A NEW PSW FROM THE STACK (IMPLICITLY) IT IS POSSIBLE THAT THIS NEW PSW COULD SET THE "T" BIT.

5ØØ - RTI OR RTT



ONCE THE "T" BIT HAS BEEN SET, A TRACE TRAP IS PENDING. IN ALL CASES, EXCEPT WHEN "T" HAS BEEN SET AS A RESULT OF AN RTT, THE "T" TRAP WILL BE SPRUNG IMMEDIATELY. IE. { PUSH PC AND PSW ON STACK; NEW PC FROM LOC 14, NEW PSW FROM LOC 16 }

IN THE CASE OF SETTING "T" AS A RESULT OF AN RTT THE INSTRUCTION LOCATED AT THE ADDRESS OF THE RETRIEVED PC IS EXECUTED AND THEN THE "T" TRAP IS SPRUNG.

TRACE TRAP ROUTINE

LOAD ADDRESS = 500/START

TRACE START UP:

```
500 - MOV #450,%6
502 - 450
504 - RTI
```

ASSUME:

```
450 - 5000
452 - 20   set 'T' bit in PW
```

PROGRAM TO BE TRACED:

```
5000 - MOV #20,%2
5002 - 20
5004 - ADD %1,%2
5006 - CMP #50,%2
5010 - 50
5012 - HALT
```

```
14 - 6000
16 - 0
```

TRACE ROUTINE:

```
6000 - CMP @(6)+,#0601002
6002 - 0601002
6004 - BEQ .+6
6006 - TST -(6)
6010 - RTT
6012 - CLR (6)
6014 - MOV -(6),%0
6016 - HALT
```

NOTE: THIS ROUTINE SEARCHES THRU PROGRAM LOOKING FOR THE INSTRUCTION ADD %1,%2. EACH INSTRUCTION IN THE PROGRAM IS EXECUTED AFTER IT IS VERIFIED THAT IT IS NOT THE INSTRUCTION WE ARE SEARCHING FOR. WHEN THE TRACED INSTRUCTION IS FOUND, IT IS NOT EXECUTED AND THE MACHINE IS STOPPED WITH THE ADDRESS OF THE INSTRUCTION DISPLAYED IN CONSOLE DATA LIGHTS.

WHENEVER 11/40 halts, R0 is displayed in data lights.

BASIC I/O PROGRAM
[NON-INTERRUPT]

```
500 - MOV #600, %1
502 - 600
504 - TSTB @ #177564
506 - 177564
510 - BPL. -4
512 - MOVB (1)+, @ #177566
514 - 177566
516 - CMP #604, %1
520 - 604
522 - BNE. -16
524 - HALT
```

600 = A, N
602 = Y, V

R6 = 1000

PURPOSE: THE ABOVE PROGRAM PRINTS THE WORD "NAVY" ON THE TELETYPE/DECWRITER

ADVANTAGE: SIMPLICITY IN CODING

DISADVANTAGE: PROCESSOR IS TIED DIRECTLY TO THE TELETYPE AND SPENDS MOST OF THE TIME WAITING FOR PRINTER TO FINISH PRINTING THE CHARACTER.

NOTE: EACH CHARACTER TAKES .1 SECONDS TO PRINT ON TTY. THIS FOUR CHARACTER PRINT TAKES 400,000 MICRO-SECONDS.

BASIC I/O PROGRAMMING

(INTERRUPT)

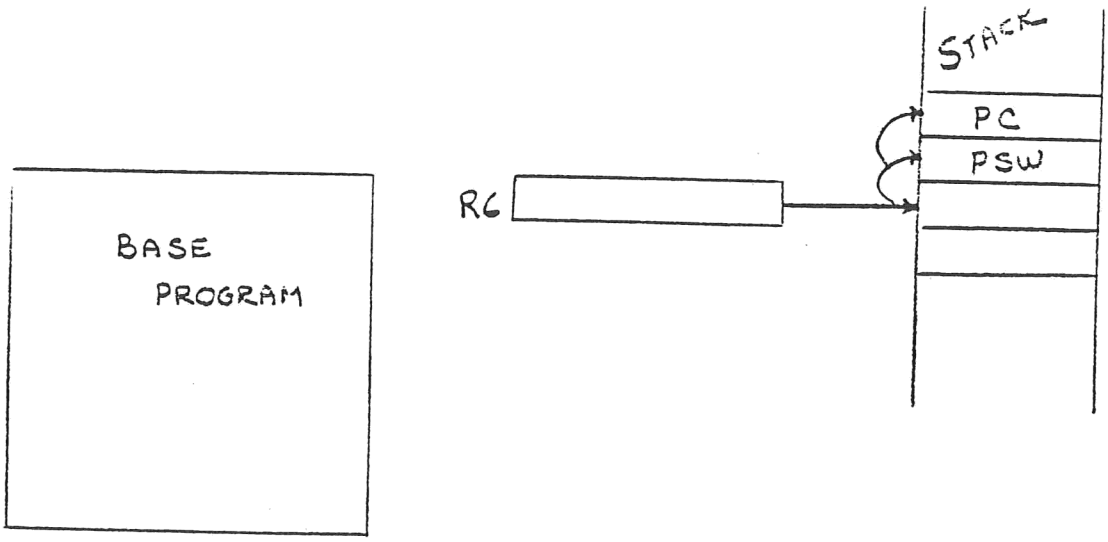
```
500     MOV #2000, %6
502     2000
504     MOV #600, %1
506     600
510     BIS #100, @#177564
512     100
514     177564
516     CLR %2
520     INC %2
522     BR.-2
```

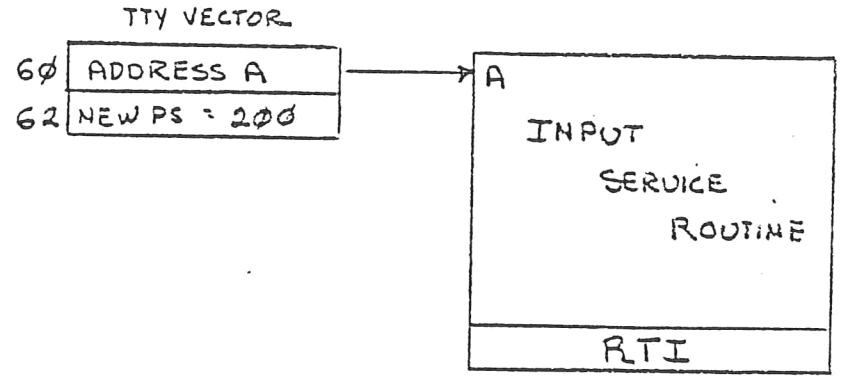
600 = A, N
602 = Y, V

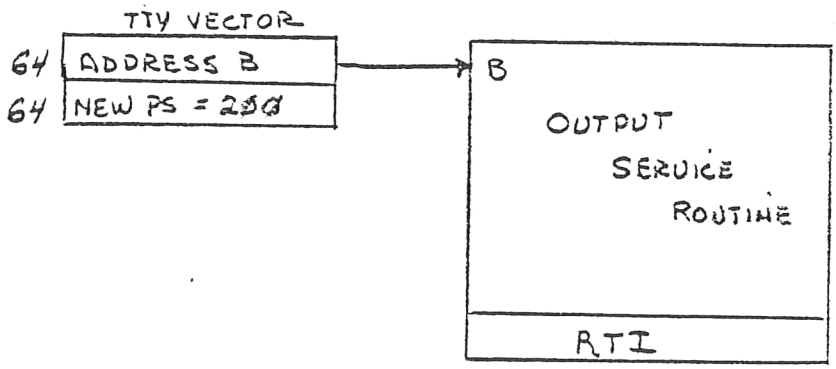
64 = 700
66 = 200

```
700     MOVB (1)+, @#177566
702     177566
704     CMP #4, %1
706     4
710     BEQ.+4
712     RTI
714     CLR @#177564
716     177564
720     HALT
```

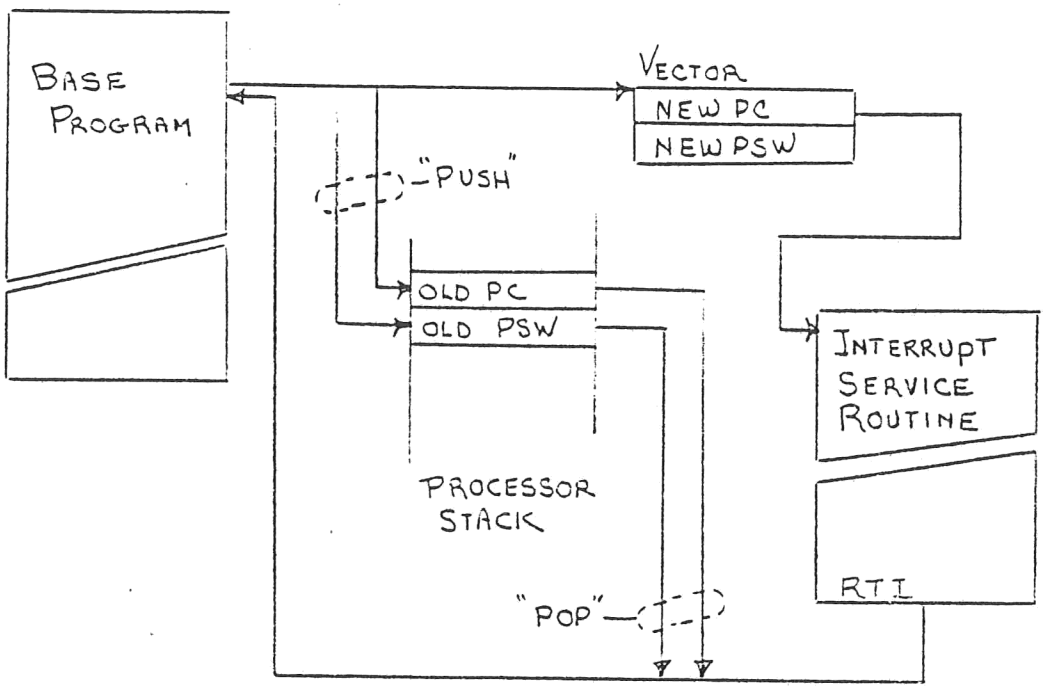
I/O PROGRAMMING USING INTERRUPT

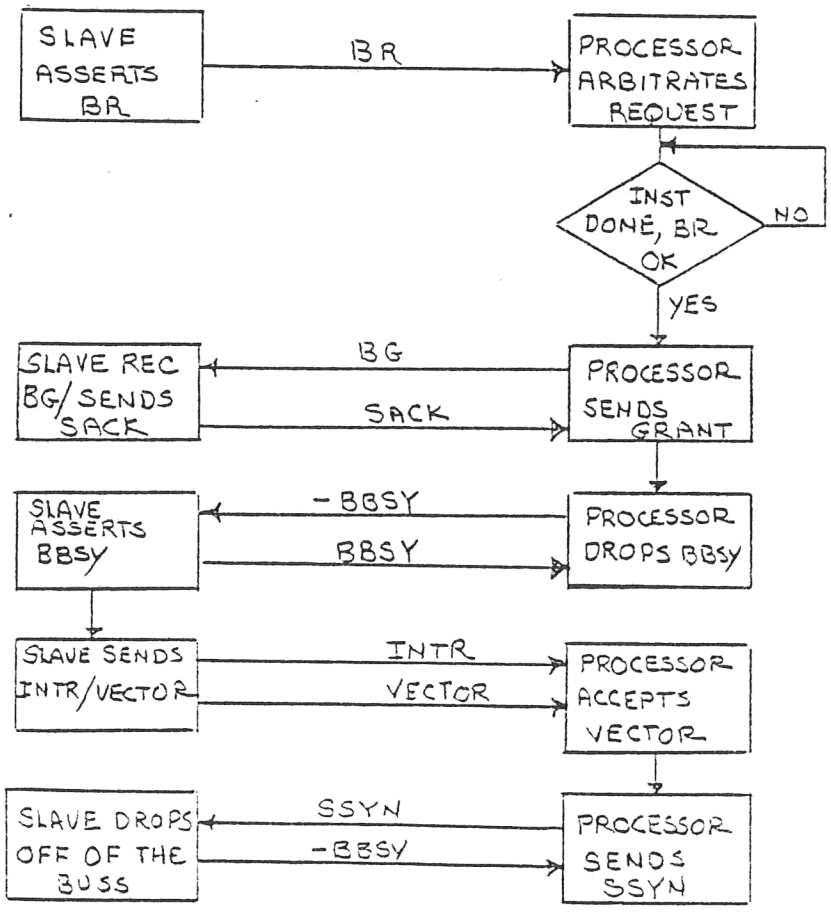




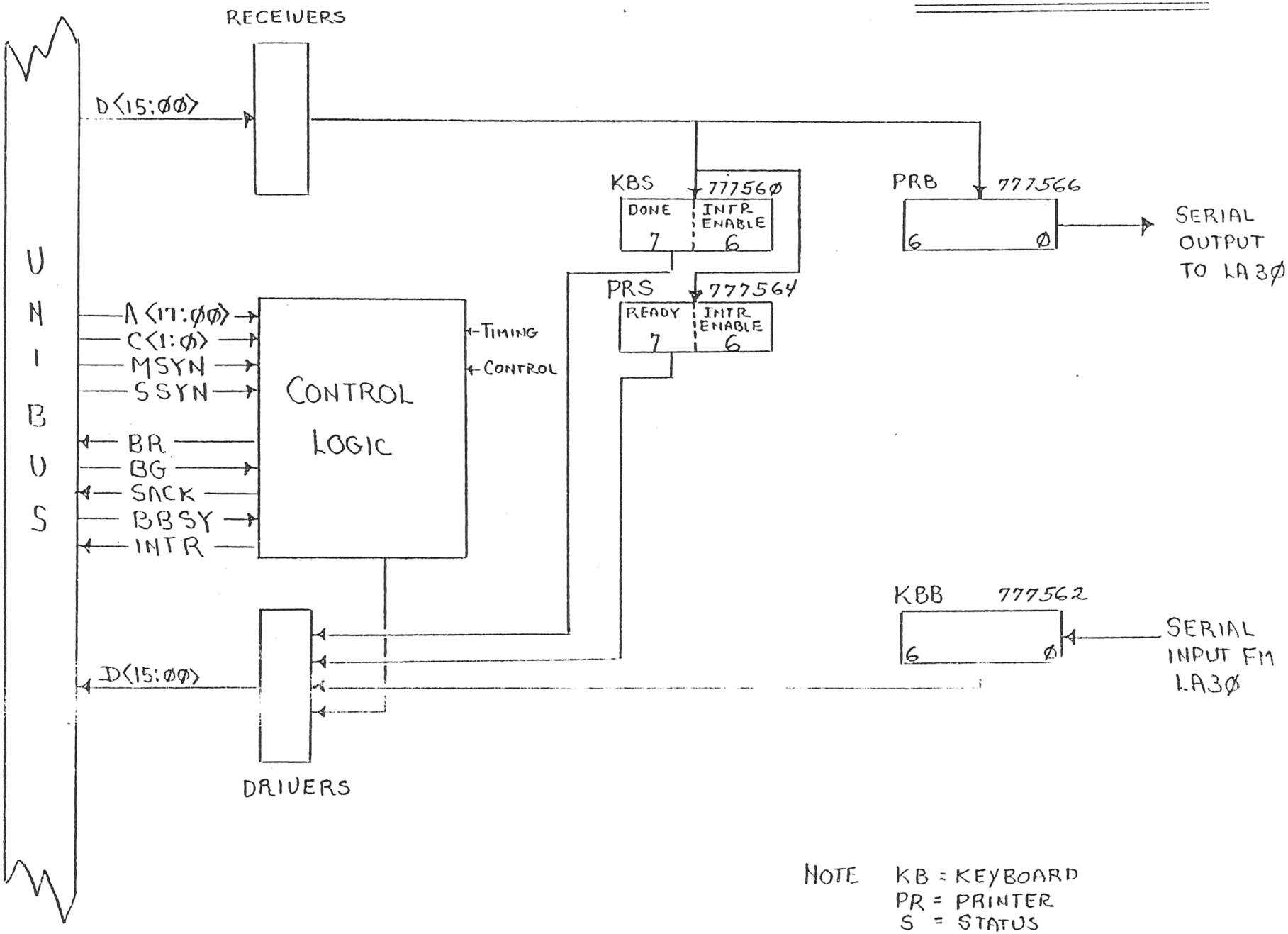


I/O INTERRUPT SEQUENCE



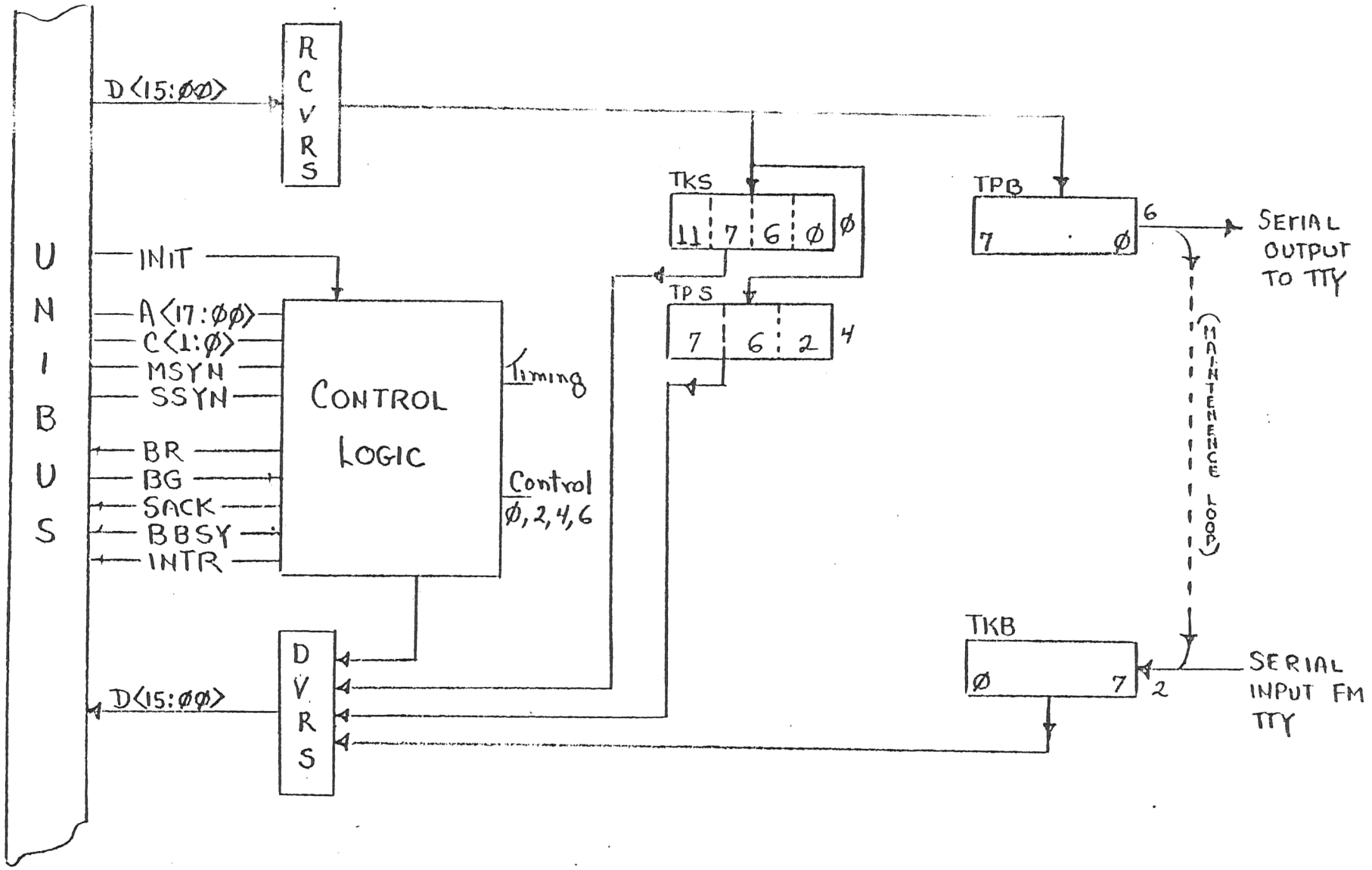


DECWRITER LA30
FUNCTIONAL DIAGRAM



NOTE KB = KEYBOARD
PR = PRINTER
S = STATUS
B = BUFFER

KL11 Teletype Control
Functional Block Diagram



INPUT

TKS = TTY KEYBOARD (READER) STATUS

BIT Ø = READER ENABLE

BIT 6 = INTR ENABLE

BIT 7 = DONE

BIT 11 = BUSY

TKS ADDRESS = 777560

TKB = TTY KEYBOARD (READER) BUFFER

BIT Ø → 7 = 8 BIT DATA HOLDING REG
FROM READER/KEYBOARD TO CPU

TKB ADDRESS = 777562

OUTPUT

TPS = TTY PUNCH (PRINTER) STATUS

BIT 2 = MAINTENCE MODE

BIT 6 = INTR ENABLE

BIT 7 = READY

TPS ADDRESS = 777564

TPB = TTY PUNCH (PRINTER) BUFFER

BIT Ø → 7 = 8 BIT DATA HOLDING
FROM CPU → TTY

TPB ADDRESS = 777566

FDP11/40 MAINTENANCE Quiz 1-4

1. Analyze the following routine:

```
500/ JST 3, @#3000
502/ 3000
504/ HALT
```

R[0] = 1500
R[3] = 1777

```
3000/ HALT
3002/ RTS 3
```

a. Load Address = 500; Depress START. What will be the contents of the following locations after the machine HALTs.?

R[3] = _____
R[6] = _____
R[7] = _____

1476/ _____

b. depress CONTINUE. What will be the contents of the following locations after the machine HALTs ?

R[3] = _____
R[6] = _____
R[7] = _____

1476/ _____

2. Analyze the following Routine:

```
500/ JMP %0
     HALT
```

R[6] = 1700
4/ 3000
6/ 7

```
3000/ HALT
3002/ RTI
```

177776/0

a. Load Address = 500; Depress START. - Determine the contents of the following locations after the machine HALTs.

R[6] = _____
R[7] = _____

1676/ _____
1674/ _____

177776/ _____

b. Depress CONTINUE, Determine the contents of the following locations after the machine HALTs.

R[6] = _____
R[7] = _____

1676/ _____
1674/ _____

177776/ _____

PDP11/40 MAINTENANCE

QUIZ 1-4
ANSWER SHEET

1. Analyze the following routine:

```
500/ JSR 3, @#3000
502/  3000
504/ HALT
```

R[3] = 1500
R[5] = 1777

```
3000/ HALT
3002/ RTS 3
```

a. Load Address = 500; Depress START. What will be the contents of the following locations after the machine HALTs.?

R[3] = 504
R[6] = 1476
R[7] = 3002

1476/ 1777

b. depress CONTINUE. What will be the contents of the following locations after the machine HALTs ?

R[3] = 1777
R[6] = 1500
R[7] = 506

1476/ 1777

---

2. Analyze the following ROUTINE:

```
500/ JMP %0
502/ HALT
```

R[6] = 1700
4/ 3000
6/ 7

```
3000/ HALT
3002/ RTI
```

177776/ 0

a. Load Address = 500; Depress START. - Determine the contents of the following locations after the machine halts.

R[6] = 1674
R[7] = 3002

1676/ 0
1674/ 502

177776/ 7

b. Depress CONTINUE; Determine the contents of the following locations after the machine HALTs.

R[6] = 1700
R[7] = 504

1676/ 0
1674/ 502

177776/ 0

ANALYZE THE FOLLOWING PROGRAM, THEN ANSWER THE QUESTION GIVEN.

```
1500 - MOV #1000,%6
1502 - 1000
1504 - JSR 7,@#2000
1506 - 2000
1510 - HALT

2000 - MOV #3000,@#4
2002 - 3000
2004 - 4
2006 - MOV #17,@#6
2010 - 17
2012 - 6
2014 - JMP %1
2016 - HALT
2020 - RTS 7

3000 - MOV #4000,@#64
3002 - 4000
3004 - 64
3006 - MOV #200,@#66
3010 - 200
3012 - 66
3014 - MOV #100,@#177564
3016 - 100
3020 - 177564
3022 - WAIT
3024 - HALT
3026 - RTI

4000 - MOV #7,@#177566
4002 - 7
4004 - 177566
4006 - CLR @#177564
4010 - 177564
4012 - HALT
4014 - RTI
```

LOAD ADDRESS = 1500
DEPRESS START

DEPRESS CONTINUE THREE TIMES TO COMPLETE THE ROUTINE

DETERMINE THE CONTENTS OF R6 AND R7 AFTER EACH HALT

1ST HALT

R6 = _____
R7 = _____

2ND HALT

R6 = _____
R7 = _____

3RD HALT

R6 = _____
R7 = _____

4TH HALT

R6 = _____
R7 = _____

ANSWER SHEET

ANALYZE THE FOLLOWING PROGRAM, THEN ANSWER THE QUESTION GIVEN

```
1500 - MOV #1000, %6      set stack
1502 - 1000
1504 - JSR 7, @ #2000     push upple 1510 on stack
1506 - 2000
1510 - HALT

2000 - MOV #3000, @ #4
2002 - 3000
2004 - 4
2006 - MOV #17, @ #6      sets PSW
2010 - 17                 774 = 1516
2012 - 6                  774 = PSW
2014 - JMP %1             772 = 2016
2016 - HALT
2020 - RTS 7

3000 - MOV #4000, @ #64
3002 - 4000
3004 - 64
3006 - MOV #200, @ #66
3010 - 200
3012 - 66
3014 - MOV #100, @ #177564
3016 - 100
3020 - 177564
3022 - WAIT
3024 - HALT
3026 - RTI

4000 - MOV #7, @ #177566
4002 - 7
4004 - 177566
4006 - CLR @ #177564
4010 - 177564
4012 - HALT
4014 - RTI
```

LOAD ADDRESS = 1500
DEPRESS START

DEPRESS CONTINUE THREE TIMES TO COMPLETE THE ROUTINE

DETERMINE THE CONTENTS OF R6 AND R7 AFTER EACH HALT

1ST HALT
R6 = 766
R7 = 4014

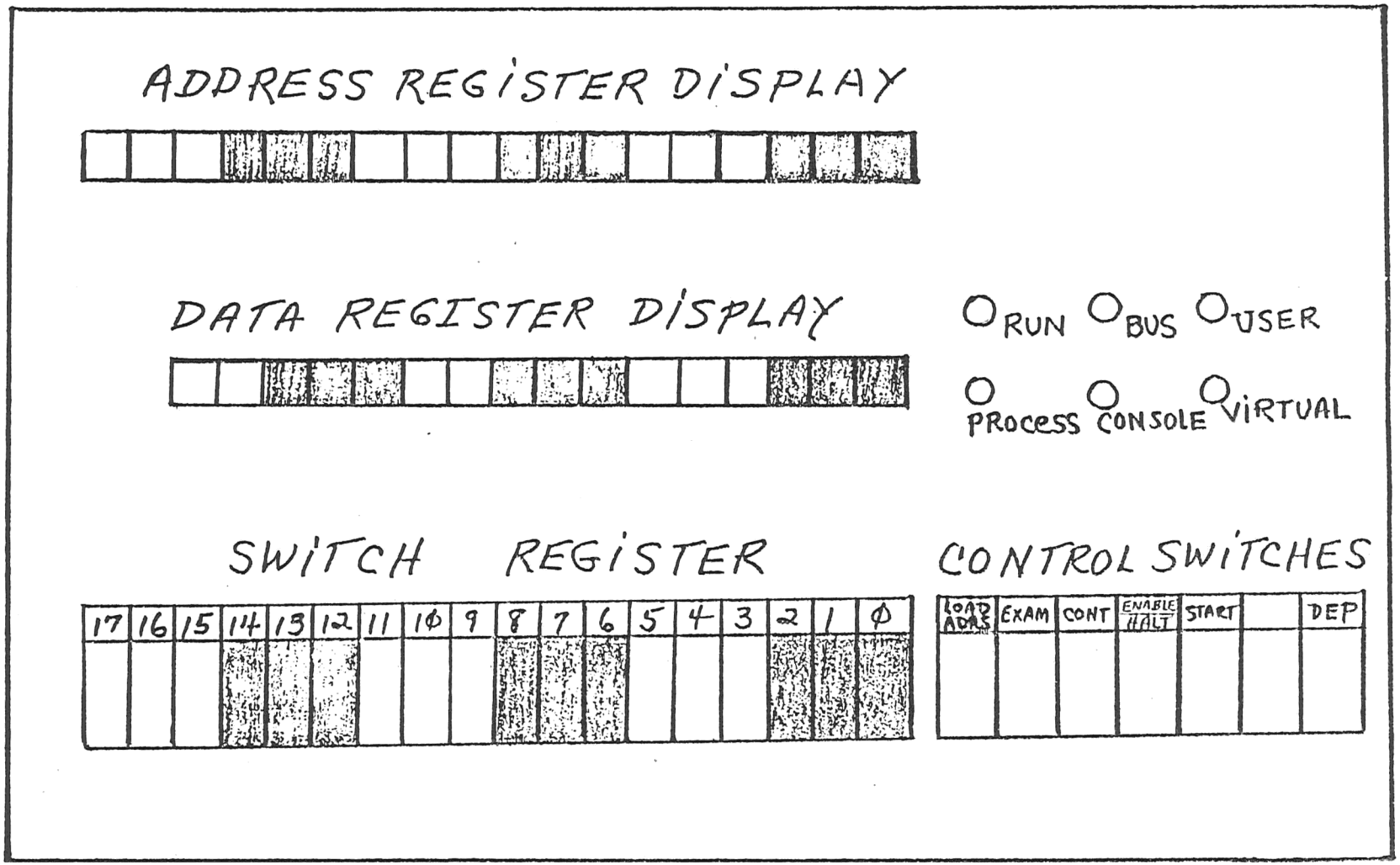
2ND HALT
R6 = 772
R7 = 3026

3RD HALT
R6 = 776
R7 = 2020

4TH HALT
R6 = 1000
R7 = 1512

DIAGNOSTIC
&
LOADERS
DAY 5

PDP-1 CONSOLE



CONSOLE OPERATION

EXAMINE MEMORY

1. HALT THE PROCESSOR
2. SET SR FOR THE DESIRED ADDRESS
3. PRESS THE LOAD ADDRESS KEY
4. PRESS THE EXAMINE KEY

DEPOSIT IN MEMORY

1. HALT THE PROCESSOR
2. SET SR FOR THE DESIRED ADDRESS
3. PRESS THE LOAD ADDRESS KEY
4. SET SR FOR THE DESIRED CONTENT
5. RAISE THE DEPOSIT KEY

RUN PROGRAM

1. HALT THE PROCESSOR
2. SET SR FOR STARTING ADDRESS OF PROGRAM
3. PRESS THE LOAD ADDRESS KEY
4. SET ENABLE/HALT SWITCH TO ENABLE
5. PRESS THE START KEY

BOOTSTRAP LOADER PROGRAM

References: Appendix E
pages 6-2 through 6-8

Location	Content
*XX7744	Ø167Ø1
XX7746	ØØØØ26
XX775Ø	Ø127Ø2
XX7752	ØØØ352
XX7754	ØØ5211
XX7756	1Ø5711
XX776Ø	1ØØ376
XX7762	116162
XX7764	ØØØØØ2
XX7766	*XX74ØØ
XX777Ø	ØØ5267
XX7772	177756
XX7774	ØØØ765
XX7776	YYYYYY**

*The BOOTSTRAP LOADER is designed so that it may be loaded into the highest available memory bank, leaving the largest possible area of contiguous free core for system and user programs (see Memory Allocation sheet). By "filling in the X's" the user indicates where the BOOTSTRAP LOADER to be located in memory.

Location	Memory Bank	Memory Size
Ø17744	Ø	4K
Ø37744	1	8K
Ø57744	2	12K
Ø77744	3	16K
117744	4	2ØK
137744	5	24K
157744	6	28K

For an 8K system, the user will replace XX with Ø3.

**The content of location XX7776 (YYYYYY) should be the address of the device status register for the paper tape reader to be used when loading bootstrap formatted tapes. It is specified as follows:
Teletype (Low Speed) Reader---17756Ø
High Speed Reader---17755Ø

Therefore, if using the high speed reader, replace YYYYYY with 17755Ø.

TABLE A
BOOTSTRAP LOADER PROGRAM

ADDRESS	CONTENTS		MNEMONICS
017 744	016 701	START:	MOV DEVICE, R1
017 746	000 026		
017 750	012 702	LOOP:	MOV#.-LOAD+2, R2
017 752	000 352		
017 754	005 211	ENABLE:	INC@R1
017 756	105 711	WAIT:	TSTB@R1
017 760	100 376		BPL WAIT
017 762	116 162		MOVB 2(R1), LOAD(R2)
017 764	000 002		
017 766	017 400		
017 770	005 267		INC LOOP+2
017 772	177 756		
017 774	000 765	BRNCH:	BR LOOP
017 776	177 560 (TK)		
	or 177 550 (PR)		

read on

THE BOOTSTRAP LOADER (TABLE A)

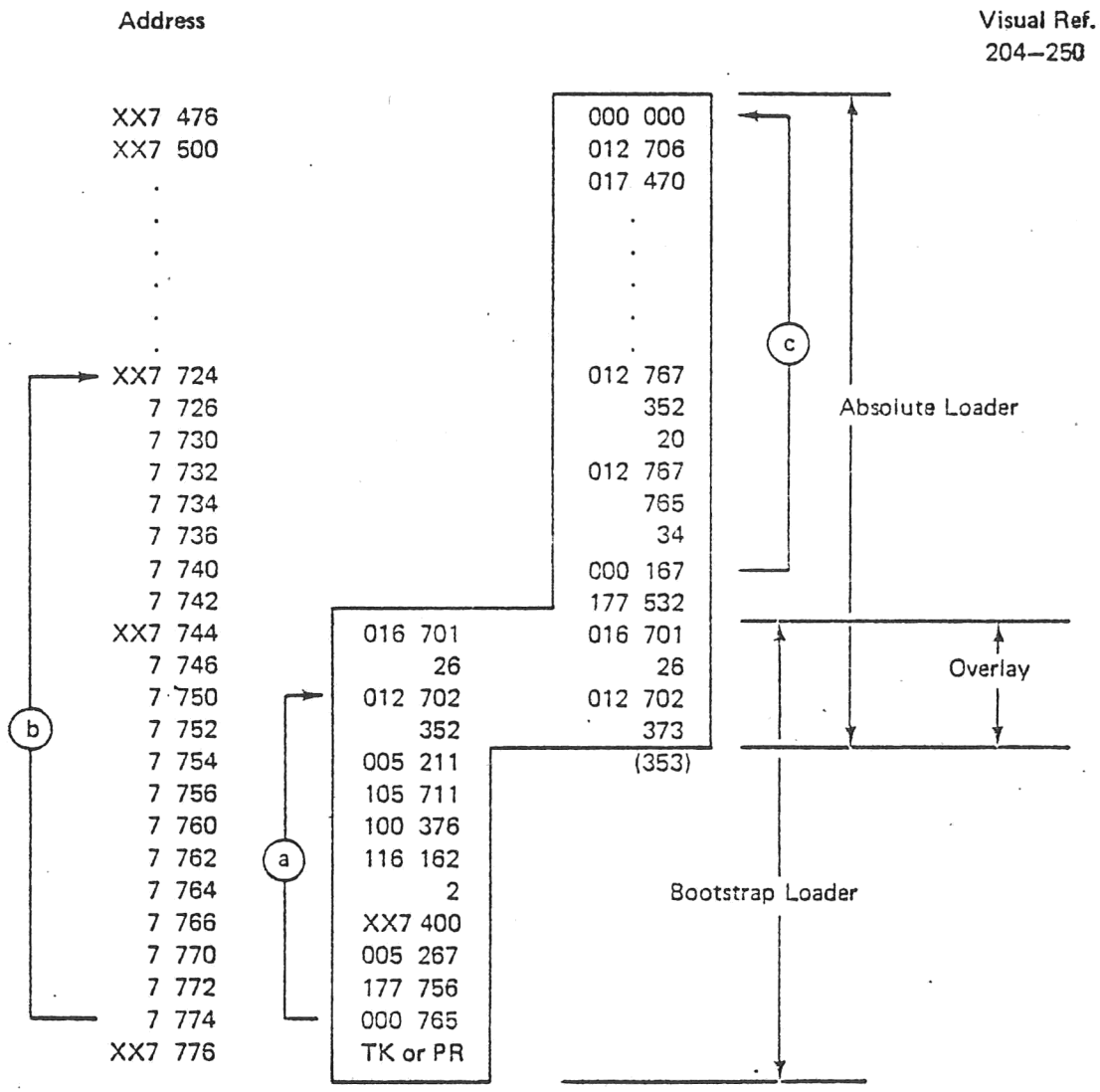
Visual Ref
163–201

Instruction Mnemonic	General Statement	Analysis
START: MOV DEVICE,R1	GPR 1 is loaded with the address of the input device's CSR.	Input device is usually a high speed paper tape reader or teletype®.
LOOP: MOV #.-LOAD+2,R2	GPR 2 is loaded with an address displacement.	
ENABLE: INC @R1	The input device is enabled.	The increment instruction places a logical 1 into the reader enable bit (the LSB).
WAIT: TSTB@R1	The program waits until a byte is read from the absolute loader tape and assembled in the DBR.	Monitors the reader CSR done bit (7). When bit 7 is a logical 1, the DBR is ready to transfer data.
BPL WAIT	When a byte is read, the program counter falls through the branch to next instruction.	When bit 7 is a logical 1, it is the same as having a negative number in the lower byte of the CSR.
MOVB 2(R1),LOAD(R2)	The byte of information assembled in the DBR is moved into memory.	Load and address displacement are summed to develop the storage address.
INC LOOP+2	Address displacement is incremented.	Generates next sequential storage address.
BRNCH: BR LOOP	Returns program counter back to the beginning of program to repeat read and storage sequence.	Unconditional branch instruction.

The relationship between the various instructions and specific memory locations is diagramed in Figure 1.

read on

Figure 1
MEMORY LAYOUT OF BOOTSTRAP & ABSOLUTE LOADER



TK = 177 560 Keyboard
PR = 177 550 Hi Speed Reader
XX is a function of memory size

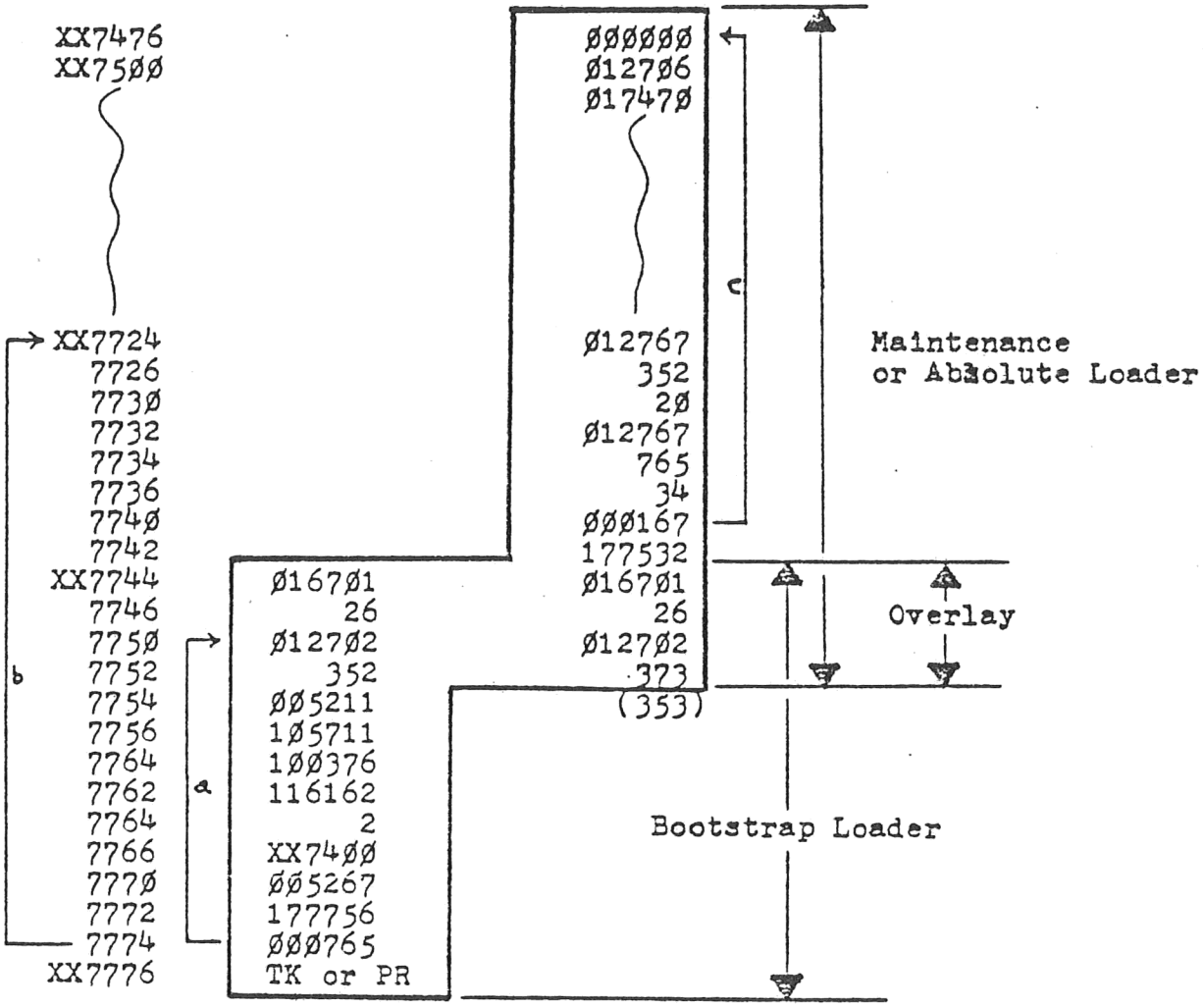
(a) Branch in Loc. XX7 774 returns PC to Loc. XX7 750 before 373 is overlayed in Loc. XX7 752.

(b) Branch in Loc. XX7 774 directs PC to Loc. XX7 724 after 373 is overlayed in Loc. XX7 752.

(c) Absolute loader jumps to Halt in Loc. XX7 476 after restoring Bootstrap Loader.

read on

MEMORY LAYOUT
OF
BOOTSTRAP & MAINTENANCE LOADER

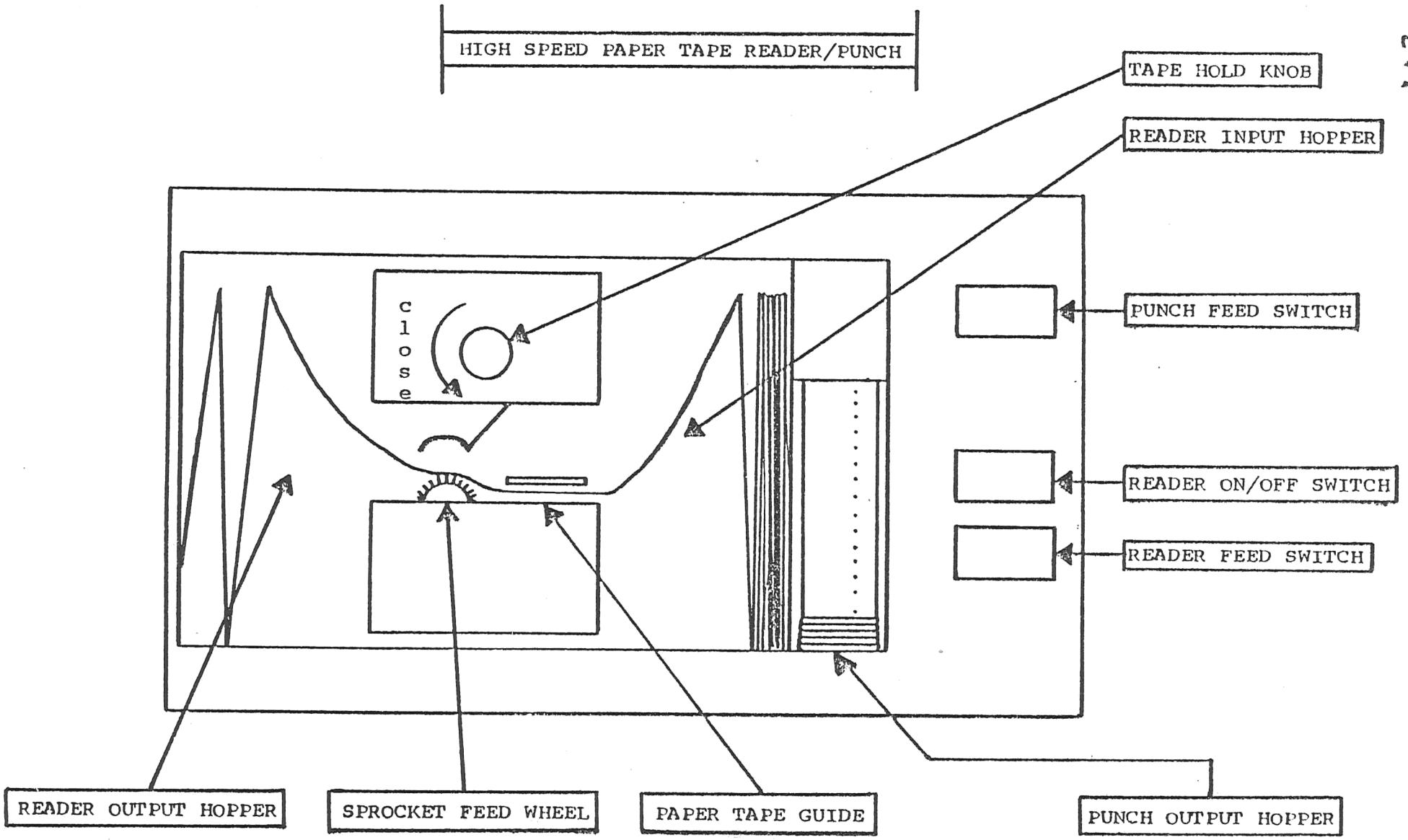


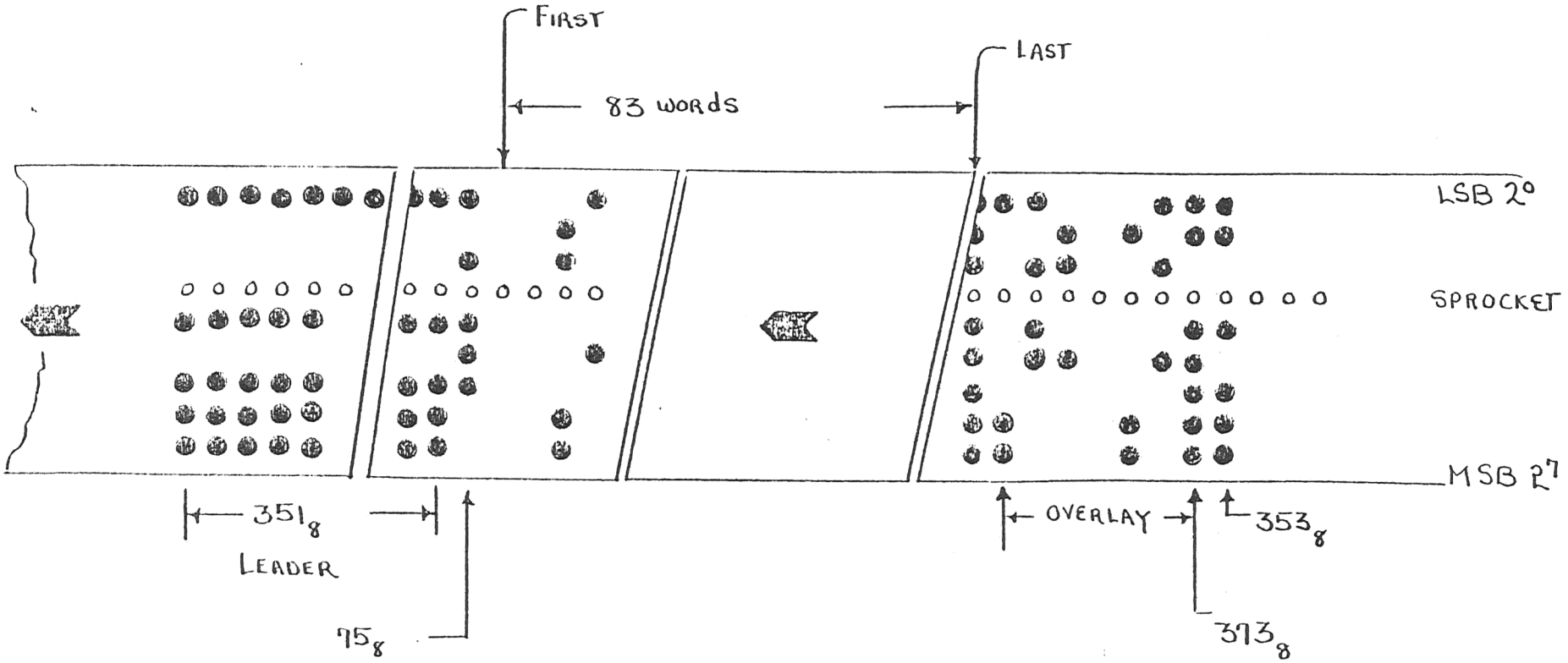
TK=177560 Low Speed Reader
PR=177550 Hi Speed Reader

a Branch in Loc. XX7774 goes to loc. XX7750 before 373 is overlayed in Loc. XX7752.

b Branch in Loc. XX7774 goes to Loc. XX7724 after 373 is overlayed in Loc. XX7752.

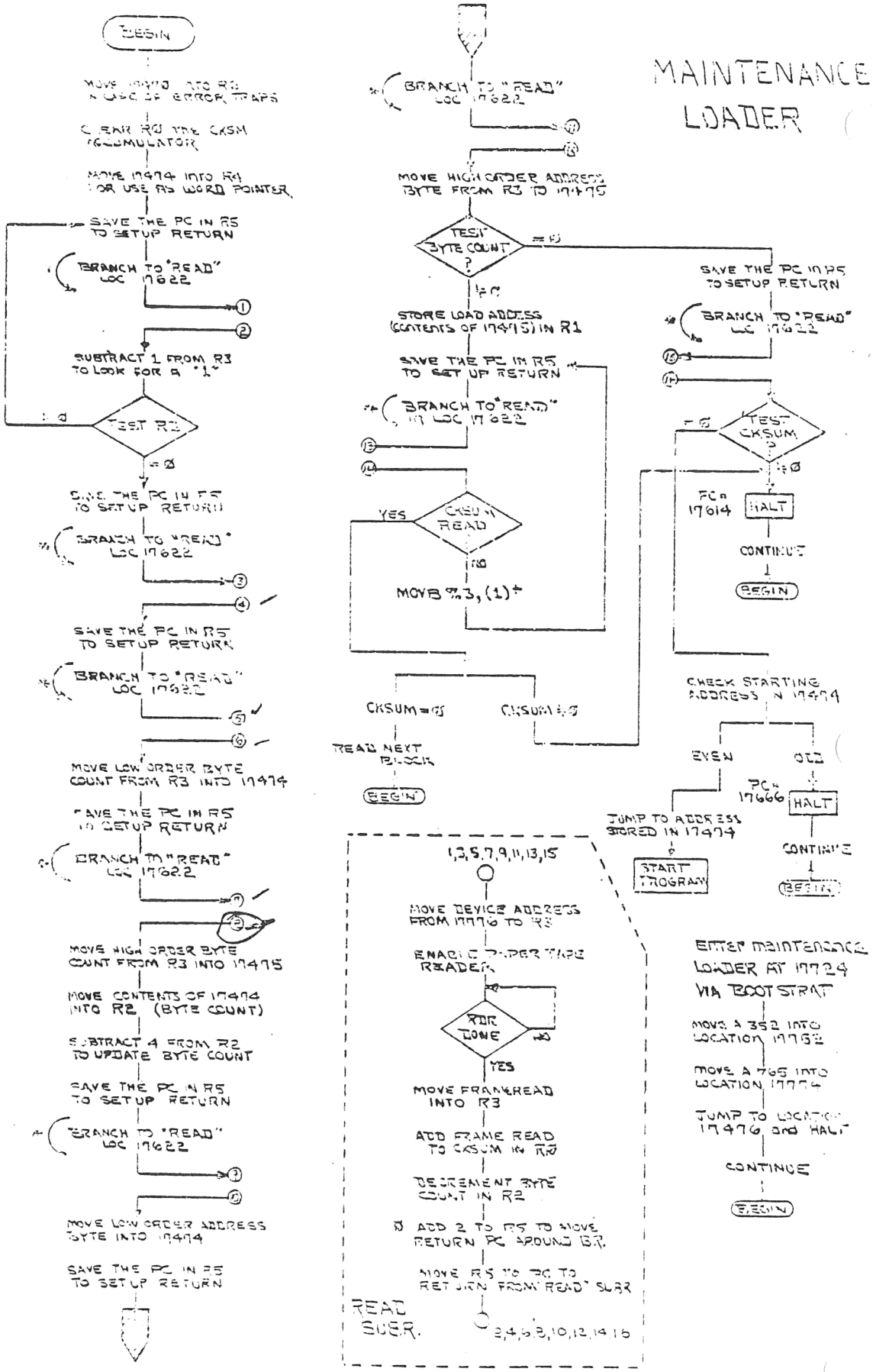
c. Jump to Halt in Loc. XX7476 after restoring Bootstrap Loader.


HIGH SPEED PAPER TAPE READER/PUNCH
TAPE HOLD KNOB
READER INPUT HOPPER
close
PUNCH FEED SWITCH
READER ON/OFF SWITCH
READER FEED SWITCH
READER OUTPUT HOPPER
SPROCKET FEED WHEEL
PAPER TAPE GUIDE
PUNCH OUTPUT HOPPER

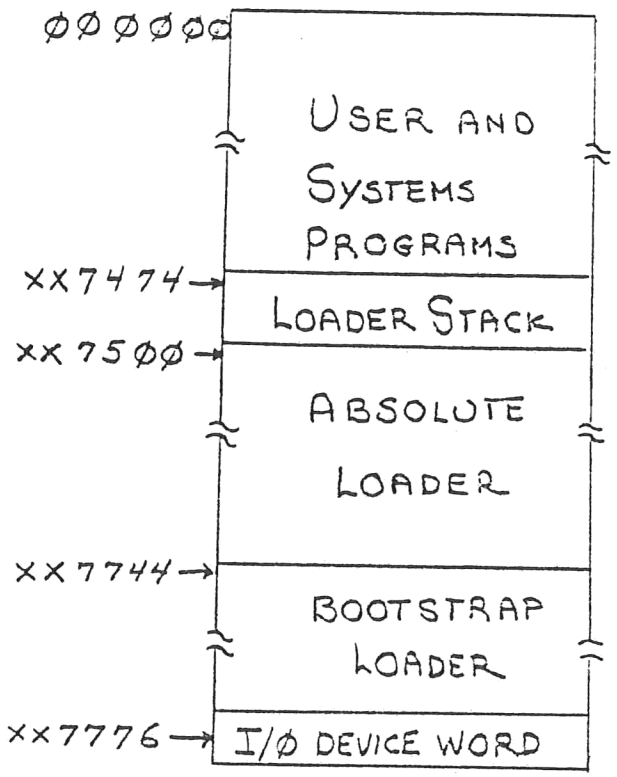


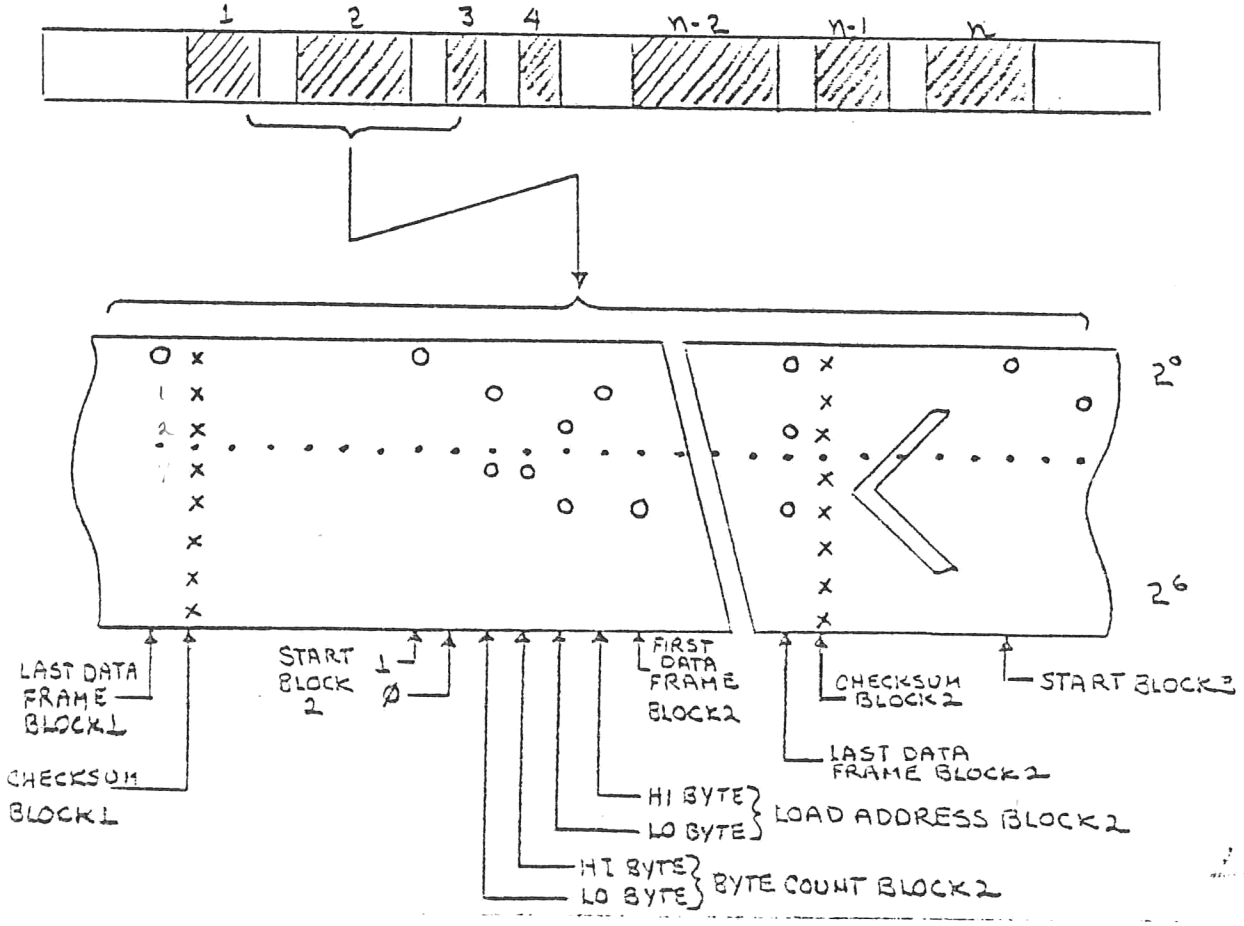
Format of Absolute or Maintenance Loader Tape

MAINTENANCE LOADER



INPUTTING WITH ABSOLUTE LOADER





PDP 11/40

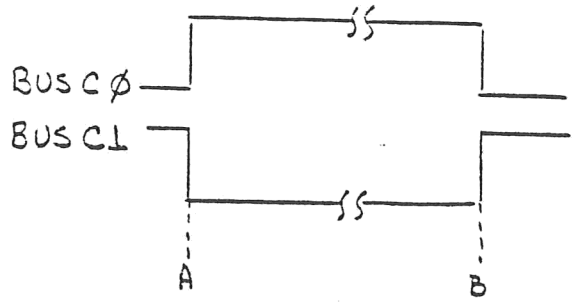
WEEK

1

EXAM

1. REFER TO THE TIMING DIAGRAM SHOWN BELOW. BETWEEN TIMES A AND B THE UNI-BUS IS BEING USED TO PERFORM A BUS TRANSACTION

A. DATI
B. DATIP
(C.) DATO
D. DATOB



2. HOW MANY BUS CYCLES WOULD THE PROCESSOR HAVE TO PERFORM TO FULLY EXECUTE THE FOLLOWING INSTRUCTION? (INCLUDE FETCH)

5ØØ - MOV 1234, @1236
5Ø2 - 53Ø
5Ø4 - 53Ø

A. 4
B. 5
(C.) 6
D. 3

1 Fetch INST.
2. " offset
3 " source data
4 " offset
5 " effective add
6. " Do it!

3 WHICH OF THE BELOW LISTED INSTRUCTIONS IS A THREE WORD INSTRUCTION

A. MOV %2, 5Ø
B. ADD @2Ø, (3)
C. MOV (2), (1)+
(D.) SUB #5Ø, @2ØØ

162777
50
200

4. IF A PROGRAMMER WERE EXPLICITLY PUSHING AND POPPING ITEMS ON TO/OFF OF THE STACK HE WOULD MOST LIKELY USE REGISTER MODES ____ AND ____

A. 2 AND 1
(B.) 4 AND 2
C. 7 AND 1
D. 6 AND 7

5. EXAMINE THE LOCATIONS SHOWN BELOW, WHICH SET OF NUMBERS IS NOT A TRAP CATCHER?

A. 10 = 12
12 = Ø

C. 30 = 32
32 = Ø

(B.) 14 = 16
16 = 20

D. 60 = 62
62 = Ø

6. WHICH INSTRUCTION SHOWN BELOW WILL ENABLE THE DEC WRITER KEYBOARD INTERRUPT?
(ASSUME BUS INIT HAS JUST FINISHED)

A. INC @#177560
B. BIS #400, @#177560
C MOV @#177560, %2
(D) BIS #500, @#177560

EXECUTE THE ROUTINE SHOWN BELOW, ANSWER QUEST #7

```
5000 - CLR %0
5002 - INC %0
5004 - SUB #2, (0)+
5006 - 2
5010 - MOV #5012, @#774
5012 - 5012
5014 - 774
5016 - HALT
```

ASSUME:

R0 = 015
R6 = 1000
LOC 0 = 5020
LOC 2 = 2
LOC 4 = 5016
LOC 6 = 2
LOC 774 = 5010
LOC 776 = 063027

7. WHEN ABOVE PROGRAM HALTS, WHAT WILL BE THE CONTENTS OF LOCATION 774

A. 5016
B. 5020
C. 5010
D. 5012

EXECUTE THE ROUTINE SHOWN BELOW AND ANSWER QUEST #8

```
500 - MOV #600, %6
502 - 600
504 - CLR -(6)
506 - BR. -2
510 - BR. -10
```

ASSUME:

LOC 0 = 512
LOC 2 = 514
LOC 4 = 500
LOC 6 = 0

8. WHAT IS IN R6 WHEN PROGRAM HALTS

A. PROGRAM WILL NOT HALT
B. 0
C. 506
D. 504

EXECUTE THE FOLLOWING MACHINE LANGUAGE PROGRAM AND ANSWER QUESTION # 9

```
START → 77Ø - ØØ4Ø2 | BR.+2
        772 - ØØØØØØ | HALT
        774 - ØØØ137 | JMP 776
        776 - ØØØ776 | BR.-2
       1ØØØ - ØØØØØØ | HALT
```

ASSUME

LOC 4 = 6
LOC 6 = Ø
LOC 10 = 12
LOC 12 = Ø

9. WHICH OF THE STATEMENTS ABOUT THE ABOVE PROGRAM IS TRUE

A. WHEN PROGRAM HALTS, PC = 1Ø
(B) PROGRAM WILL NOT HALT
C WHEN PROGRAM HALTS PC = 14
D WHEN PROGRAM HALTS PC = 1Ø02

CONSIDER THE FOLLOWING INSTRUCTION WHEN ANSWERING QUESTIONS 1Ø

1776- ADD 37774, @2ØØ5

10. WHAT IS THE VALUE THAT SHOULD BE STORED IN LOC 2ØØØ

(A)- 35772
B - 35774
C - 3577Ø
D - 35766

11. EXECUTE THE FOLLOWING PROGRAM AND INDICATE WHAT IS PRINTED ON THE DEC-WRITER

```
500 - MOV #600, %1      012701
502 - 600                600
504 - TSTB @#177564      105737
506 - 177564             177564
510 - BPL .-4            100375
512 - MOV (1)+, @#177566 012137
514 - 177566             177566
516 - CMP #610, %1       022701
520 - 610                610
522 - BNE .-16           001370
524 - HALT               000000
```

ASSUME

600 = 04 B 1 | 10 A 1
602 = 04 B 1 | 502 B
604 = 042 1 | 10 C
606 = 042 | 504 D

A. ABCBCDDE

B. ABCD

C. BCDE

D. ABBCCDDE

12. ANALYZE THE PROGRAM BELOW:

```
500 - MOV #1025, %1
502 - 1025
504 - SWAB %1
506 - HALT
```

WHEN MACHINE HALTS, R1 SHOULD CONTAIN

(A) 012402
B. 102500
C 125000
D. 025001

13. WHICH OF THE BELOW STATEMENTS IS TRUE

A. THE JMP INSTRUCTION SAVES PC ON THE STACK (NO)

B. THE JSR INSTRUCTION CANNOT SAVE PC ON THE STACK

C THE TRAP INSTRUCTION CANNOT SAVE PSW ON THE STACK

(D) THE JSR INSTRUCTION CANNOT SAVE PSW ON THE STACK

ANALYZE THE ROUTINE SHOWN BELOW THEN ANSWER QUESTION 14 AND 15

LOAD ADDRESS = 10500
DEPRESS START

```
10500 - JSR 2, @#1500
10502 - 1500
10504 - BR. +4
10506 - HALT
10510 - HALT
```

ASSUME

R2 = 3156
R6 = 1020

```
1500 - MOV @#177776, %0
1502 - 177776
1504 - HALT
1506 - MOV %0, @#177776
1510 - 177776
1512 - RTS 2
```

14. AFTER FIRST HALT YOU EXAMINE R6, YOU SHOULD OBSERVE

(A) 10504
B. 1506
C. 1022
(D) 1016

15. DEPRESS CONTINUE, THE MACHINE HALTS A SECOND TIME. IF YOU NOW EXAMINE R2 YOU SHOULD OBSERVE:

A. 10512
B. 3156
(C) 10504
D. 1020

16. ANALYZE THE FOLLOWING ROUTINE AND INDICATE THE CONTENTS OF PC WHEN THE MACHINE HALTS.

```
START → 500 - MOV #514, @#4        012737
        502 - 514                      514
        504 - 4                          4
        506 - CLR @#6              005037
        510 - 6                          6
        512 - JMP 700          OP → 111 >MR (1)   000100
        514 - ADD #11, @#512       062737
        516 - 11                        11
        520 - 512                      512
        522 - MOV #6, 70L          012701
        524 - 6                          6
        526 - BR. - 14                 771
```

A. PC = 10

B. PC = 6

C. PC = 4

D. MACHINE WILL NOT HALT

4 = 514

6 = 0

512 = 11

17. REFER TO THE INSTRUCTION SHOWN, WHAT IS THE ADDRESS OF THE OPERAND TO BE DECREMENTED?

500 - DEC @ 50 (2)
502 - 50

ASSUME

R2 = 350
420 = 220
220 = 100

A. 220
B. 420
C. 100
D. R2

18 TO COMPENSATE FOR SKEW ON THE UNI-BUS ADDRESS AND CONTROL LINES, BUS MASTER DELAYS

A. BUS SSYN
B. BUS INIT
C. BUS MSYNA
D. BUS MSYN

19 ON A SYSTEM WITH 24K OF MEMORY, YOU WOULD DEPOSIT A _____ INTO LOCATION _____ IF YOU WANTED TO LOAD THE ABSOLUTE LOADER FROM THE HIGH SPEED READER

A. 17756Ø, 137776
B. 17756Ø, 157776
C. 17755Ø, 137776
D. 17755Ø, 157776

FOR QUESTION 2Ø - REFER TO FIGURE 1

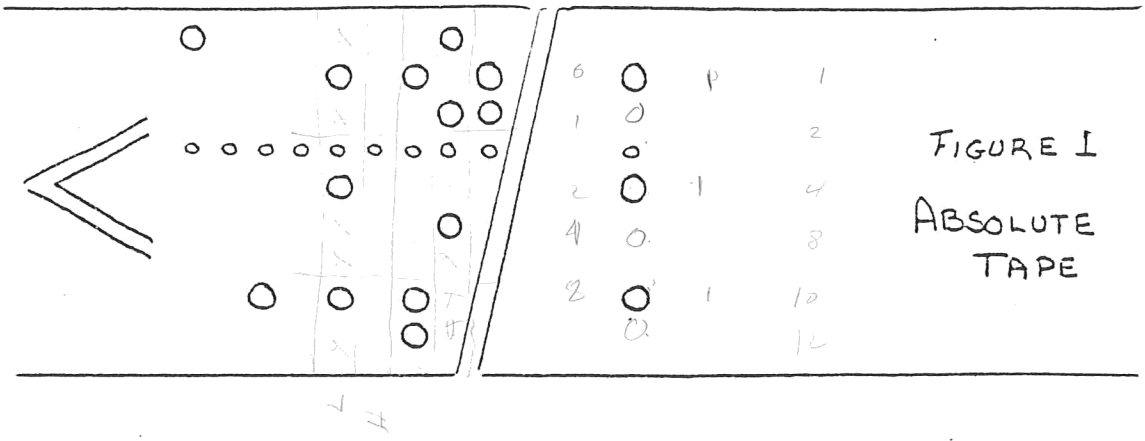


FIGURE 1

ABSOLUTE TAPE

2Ø THE FIRST DATA FRAME WILL BE LOADED INTO LOCATION

A. 1ØØ
B. 112
C. 3Ø2
D. 7ØØ

21. IF THE FIRST DATA WORD IS EXECUTED AS AN INSTRUCTION IT WOULD ALTER

A. RØ
B. R1
C. R2
D. R3

0 1 0 0 0 0 1 0 0 0 0 1 0 1 0 1
Ø 4 1 0 2 5

Unified bus maximizes minicomputer flexibility

Faster memories or an unusual mix of input-output devices can simply be plugged into minicomputers that transmit addresses, data and control information over a single path instead of three

By David Chertkow and Roger Cady, *Digital Equipment Corp., Maynard, Mass.*

☐ To interconnect processors, memories and input-output elements, some of the newest small computers are using a unified bus—a single, long data path comprising several separate signal wires. In these systems, all the devices send and/or receive addresses, data, and control information by means of the same set of signals, and all the devices are connected with the bus by means of a single, standard interface.

As a result, the user can forget his fear of obsolescence, since he can plug in the faster memory or more powerful processor as quickly as technological advances make them available. He can also buy a minimum computer configuration to start with, and expand it as necessary. Or he can tailor a system to suit highly specialized needs: for large batch computation or signal processing, one system could have a large memory and few input-output devices, while for data collection or industrial process control another system might plug many peripherals and relatively little memory into the same bus length.

Such flexibility is in distinct contrast to the traditional system architecture of small computers, which in most cases has called for three separate buses for memory, input-output and direct memory access (DMA). These are usually physically as well as functionally separate, and not designed to permit easy interchange of computer elements.

According to some critics of the unified bus approach, however, the conventional system performs better. For one thing, they say, a unified bus degrades total system performance because the single data path between all components allows only one operation to take place at a time. For another, the bus is said to be very vulnerable to external noise.

Certainly, with just one bus it's possible for only one data transfer to occur at a time. But the same is true of three-bus machines: in conventional minicomputers, the input-output bus and the DMA bus never operate simultaneously, and linking the memory bus to the DMA bus always stalls the central processor. The supposed degradation is therefore more theoretical than actual. And actually, the unified bus may improve system performance, because the bus may be asynchronous—traditional bus structures are synchronized to the memory cycle time, so they may take longer to perform certain operations.

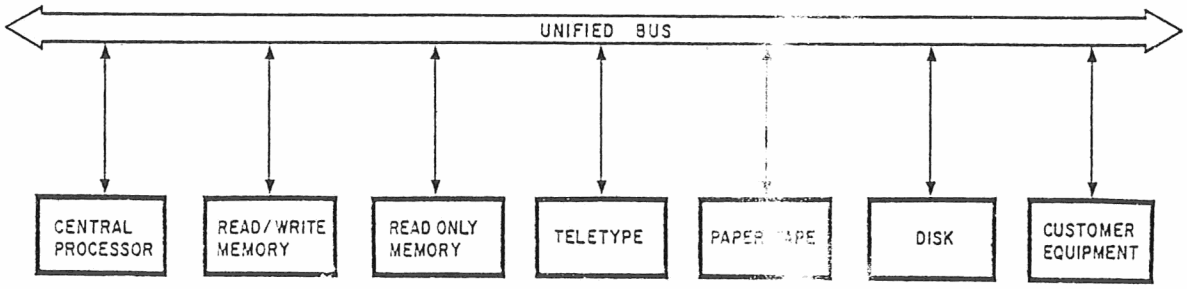
The second objection would hold water only if one were willing to admit that, while memory buses must be secure against noise, input-output buses need not be. But if this were true, the computer could not be used for input-output.

The immunity to real-world noise required by a unified bus is achieved through careful engineering of design aspects like the role of the bus as a transmission line with low-impedance terminations.

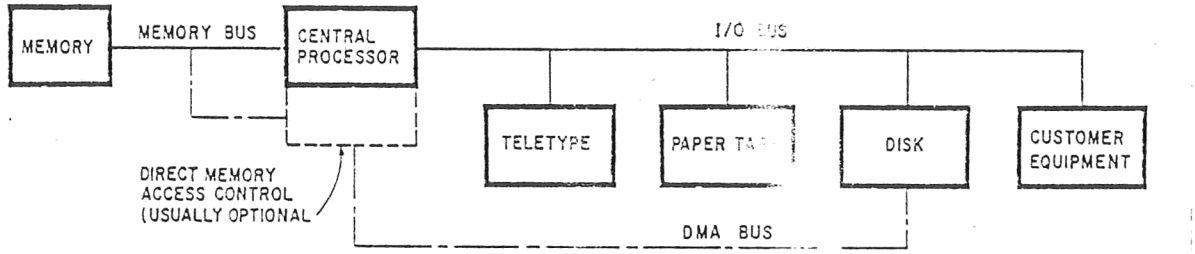
As a physical link between many independent units, the unified bus makes certain demands on the design of the attached devices. For example, because the bus is a transmission line, the receivers that monitor the signals on the line must have only a minimal effect on its distributed impedance. That is, they must have high impedance, if they're not to set up reflections on the bus. And because the bus can be driven from several sources, each output transistor—which may be part of a monolithic structure—should have an open collector, as in Fig. 3.

For these reasons, the usual connection from the collector through a resistance to a power supply is made at the end of the bus, rather than directly at the transistor. In this position it both terminates the line and loads the transistor; in any other position the line would be improperly terminated and reflections would occur. Also, the open collectors permit the drivers to be connected in parallel with other similar collector drivers, as in Fig. 4. This logic is commonly known as a wired OR.

Tri-state logic, recently introduced by a leading semiconductor manufacturer [*Electronics*, Sept. 14, p. 78], might seem an alternative solution to these difficulties. But it causes problems with buses that are electrically long. Though a tri-state scheme works well within a single electronic assembly, it doesn't fit into a properly terminated bidirectional bus scheme: the tri-state device relies on transistors to drive both



1. Unified bus. When a single multiwire path interconnects all units of a computer, the result is a flexible, expandable system that can readily be updated.



2. Traditional architecture. Small computers have usually been built with three separate buses, and so are not easy to adapt to specific applications or to improved technology.

its positive and negative levels, so that the bus cannot be treated simply as a passive terminated transmission line. Also, if the electrical transit time along the bus exceeds the signal rise time, the signal waveform deteriorates. Furthermore, no readily available tri-state devices are capable of sinking the necessary termination current. Finally, since inadequate voltage margins would occur if transistor-transistor logic devices were used as receivers, and since each TTL device would degrade the transmission line signals because of loading, even tri-state drivers would not alleviate the need for a special high-impedance controlled-threshold receiver.

A typical open-collector driver, DEC 8881, has the characteristics listed at far right, while a unified bus receiver can be implemented with a monolithic integrated circuit element such as DEC 380A, also described at right and shown schematically in Fig. 5.

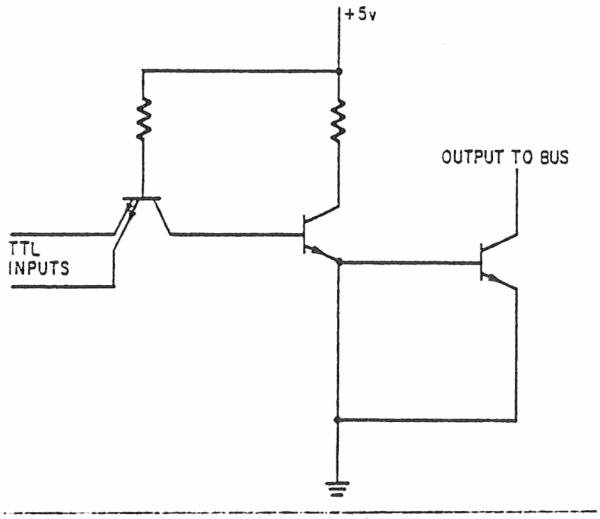
For any combination of interconnected transmitters and receivers in a unified bus system, it's important to know the range of possible values for the load resistance, shown in Fig. 6 as R_L. A large load resistance minimizes power dissipation, but to the detriment of switching speeds and immunity to coupled-in noise. But if the resistance remains below its absolute upper limit, the margins against noise on the power lines are not affected. The terminator must also match the characteristic impedance of the cable employed in the bus interconnection.

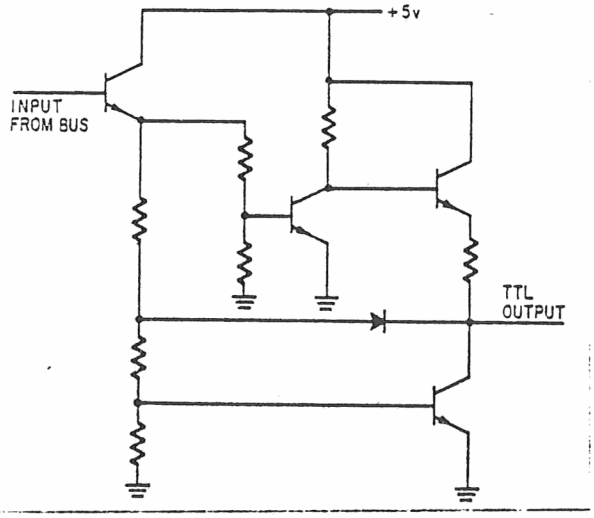
What determines the maximum value of R_L is the total leakage current, or the sum of the leakages through all the transmitters connected to the node, when they are all off, plus the input currents to all the receivers. On passing through the load resistance, this current creates a voltage drop that, if it is too large, forces the voltage at the common collector node below an acceptable minimum. The nominal voltage is 3.5 volts.

The minimum value of R_L depends on the current-sink capability of the transmitter, with no leakage. This current is much larger than the leakage current, and its voltage drop across R_L brings the collector voltage down to between 0.4 and 0.8 volt, depending on the collector-emitter impedance of the transistor.

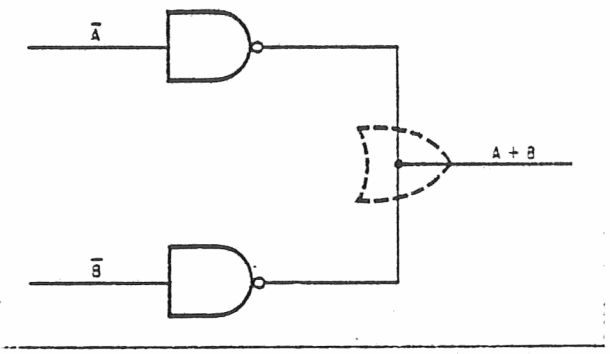
The bus loading is also one of the complex of factors that determines the maximum length of the bus. The others are the distribution of receiver and transmitter taps on the bus, and its physical implementation—coaxial cable, flat cable or twisted pair. Bus lengths of over 100 feet are theoretically possible.

Many new machines are available that use the unified bus in one way or another. Two typical designs are Digital Equipment Corp.'s PDP-11 and PDP-8/E. The PDP-11 is a 16-bit processor, designed to meet the





3. Driver (above left). Open-collector design is essential because several drivers are likely to be connected to the bus. They share a common load resistor.



4. Wired OR (left). Several open-collector drivers whose collectors are electrically common can perform a logic OR function with no additional circuitry.

5. Receiver (above). A high input impedance is essential to these circuits to prevent them from affecting the distributed impedance of the bus.

demand for a modular system for real-time data acquisition, analysis and control. Its unified bussing scheme is called Unibus.

The PDP-11 architecture takes advantage of the Unibus in its method of addressing input-output devices by using the same format for I/O instructions as for memory reference instructions. Memory elements, such as the main core memory or any read-only or solid-state memories* that are attached, have ascending addresses starting at zero, while registers that store the status of individual peripheral devices have descending addresses, starting at the highest possible address.

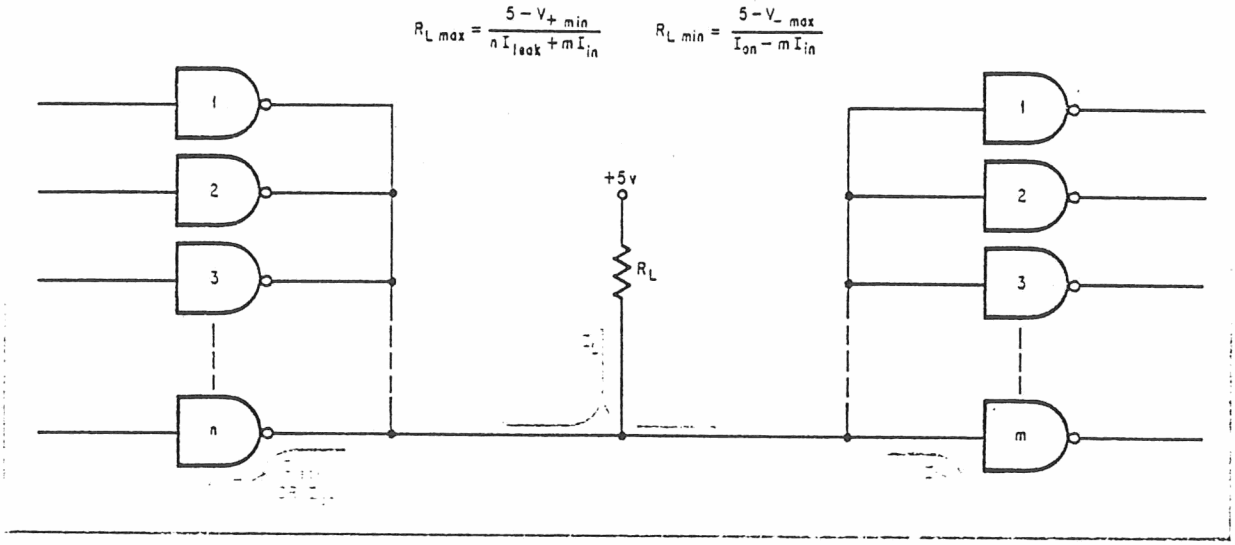
There are generally several thousand memory addresses, but only two—one for data, one for control—for each simple peripheral device, and up to perhaps half a dozen for more complicated equipment like magnetic tapes or disks. The valid memory addresses run from 0 to 65,535; in a typical system, addresses 0 through about 32,000 would indicate locations in memory, addresses from 65,000 up would refer to input-output gear, and the intervening numbers would be reserved for future expansion.

The PDP-11 unified bus consists of 56 signal lines, to which all devices, including the processor, are

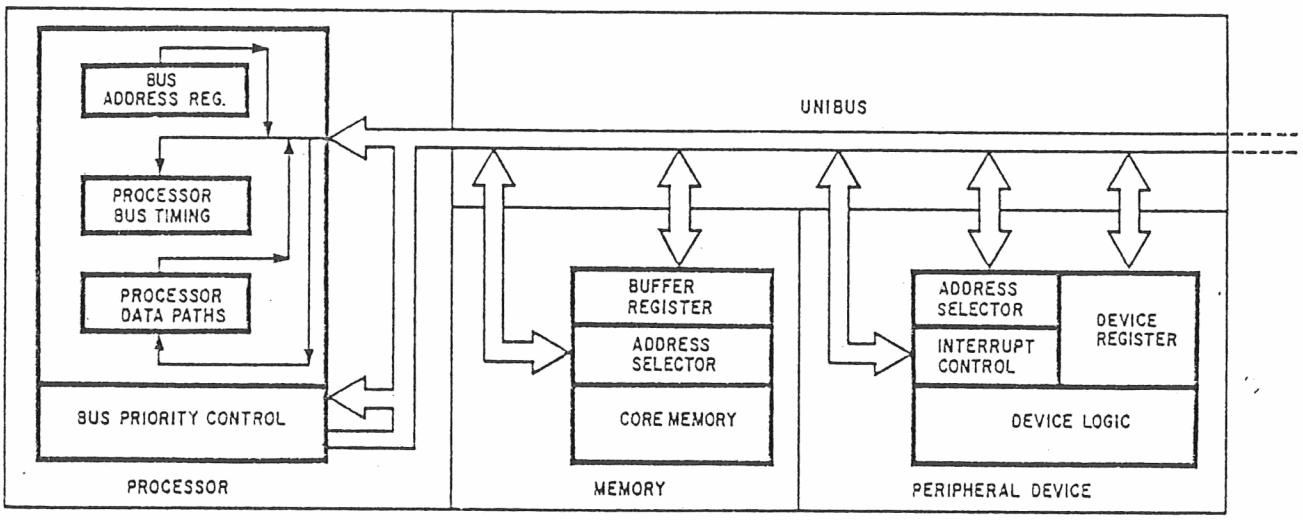
* These are not now available on DEC computers, but will probably be announced in the not too distant future.

Input loading	1.25 TTL unit load
Low output voltage	0.8 V maximum
@ 50 mA sink	0.4-0.6 V typical
High output leakage	
@ 3.5 V	25 μA

Input high threshold	2.5 V maximum
Input low threshold	1.4 V minimum
Input current @ 2.5 V	160 μA maximum
Input current @ 0 V	25 μA maximum
Output drive	7 TTL unit loads



6. **Load resistance.** Total leakage current determines the maximum value of R_L when all the driving transistors are off. Maximum current drawn by one driver when it is on establishes R_L's minimum value.



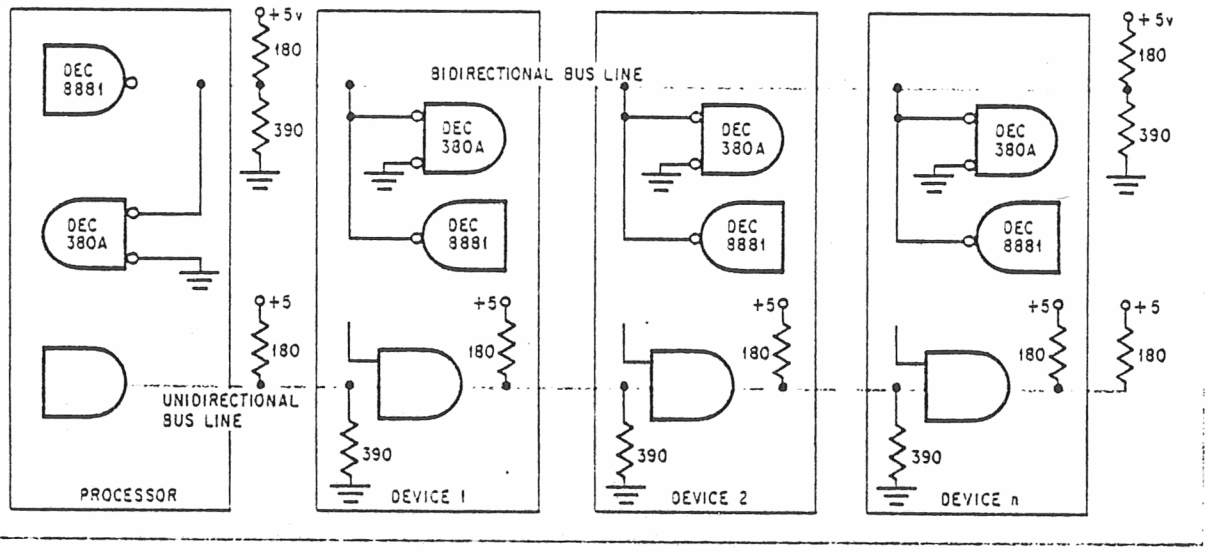
7. **Unibus.** In the PDP-11 there are 56 lines in the unified bus; 51 are bidirectional and the other five are unidirectional. Each peripheral device has from two to six addresses; the memory has several thousand.

connected in parallel, as in Fig. 7. On 51 bidirectional lines signals flow in either direction; these lines are terminated at each end by a resistive divider for each signal, as shown in Fig. 8. The remaining five unidirectional lines are used for priority bus control.

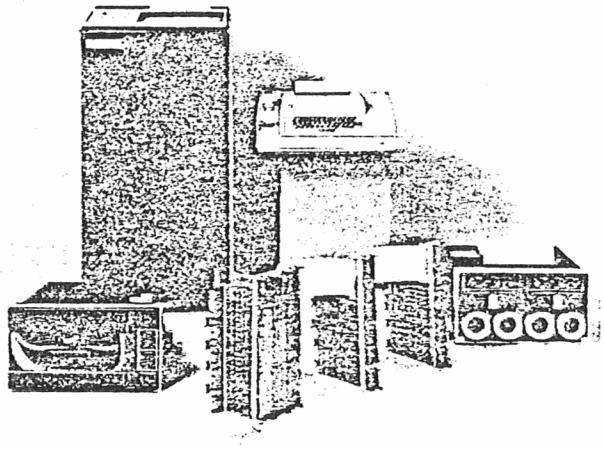
The logic 0 for Unibus signal lines is +3.4 volts; the logic 1 is between ground and +0.8 volt, which is the saturation voltage of the transmitter circuit driving the bus.

Communication between any two devices on the bus is in a master-slave relationship. During any bus operation, one device, the bus master, controls the bus when communicating with another device on the bus, called the slave. For example, the processor, as master, can fetch an instruction from the memory, which is always a slave, or the disk, as master, can transfer data to the memory, as slave. Master-slave relationships are dynamic: the processor, for example, may pass bus control to a disk, whereupon the disk may become master and communicate with a slave memory bank.

When two or more devices try to obtain control of the bus at once, priority circuits decide between them. The priority level of the central processor is programmable, and masks anything of lower priority on the bus. Other devices can have different priority levels, but their levels are fixed at the time of installation. A unit with a high priority level obviously always takes precedence over one with a low priority level, but of units with equal priority levels the one closer to



8. **Terminations.** In the Unibus each bidirectional line has a resistive divider at each end that matches the line's characteristic impedance. Unidirectional lines are short and need only loads, not terminations.



9. **Plug-ins.** The white cable interconnects units in different racks or panels, and plugs in the same way the units do.

the processor on the bus takes precedence over those farther away.

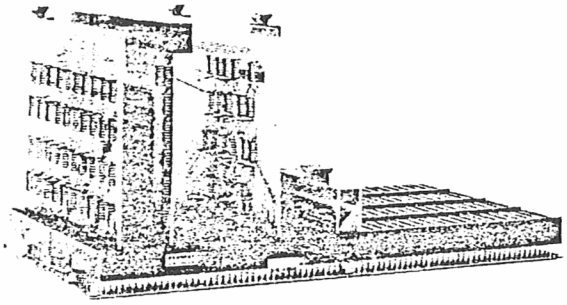
The diagram of the unidirectional line in Fig. 8 hints at this arrangement and its implementation. Suppose the processor has control of the bus when three devices, all of higher priority than the processor, request it. If the requesting devices are of different priority, the processor will grant use of the bus to the one with the highest priority. If they are all of the same priority, all three signals come to the processor along the same bus line, so that it sees only one request signal. Its reply granting priority travels down the bus to the nearest requesting device, passing through any intervening non-requesting devices, which simply regenerate the grant signal. The requesting device takes control of the bus, executes a single bus cycle of a few hundred nanoseconds, and relinquishes the bus. Then the request-grant sequence occurs again, this time going to the second device down the line, which has just been sitting there holding its hand in the air, so to speak. When all higher-priority requests have been granted, control of the bus returns to the lowest-priority device, usually the processor.

The processor usually has lowest priority because in general it can stop whatever it is doing without difficulty. Other devices, however, may be involved in some kind of mechanical motion, or may be connected to a real-time process, either of which requires immediate attention to a request.

The priority determination takes place asynchronously in parallel with data transfer. Every device on the bus except the memory is capable of becoming a bus master.

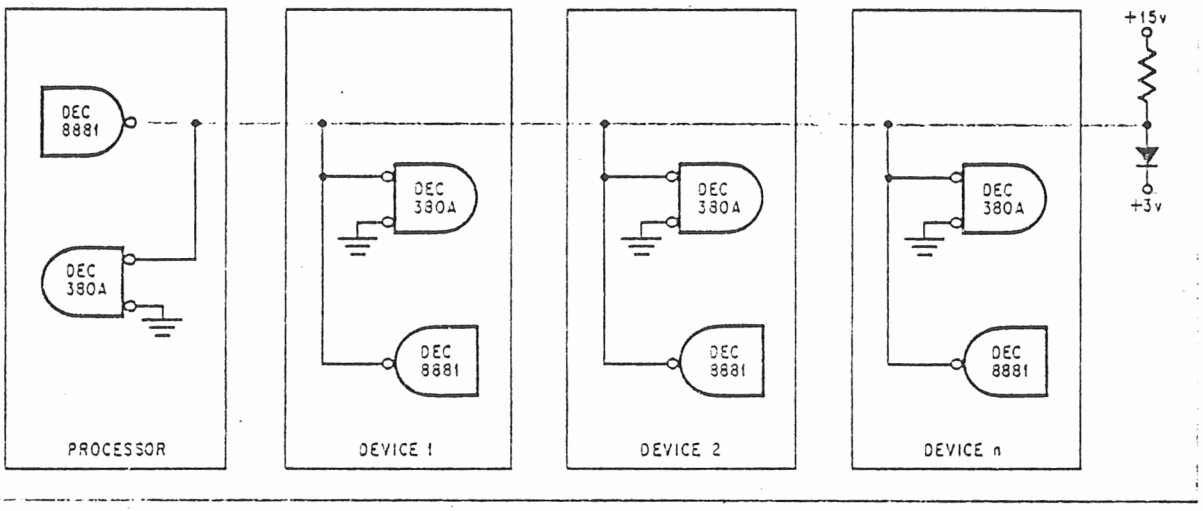
Communication on the bus is interlocked, so that each control signal issued by the master must be acknowledged by a response from the slave to complete the transfer. As a result, communication is independent of the physical bus length and the response time of the master and slave devices. This simplifies the device interface because timing is no longer critical, and slower logic types may be used in, for example, an industrial environment that requires large noise margins in the peripheral equipment. The maximum typical transfer rate on the Unibus is one 16-bit word every 400 ns, or about 2.5 million 16-bit words per second.

With Mylar-wrapped printed circuit cable, the maximum reasonable bus length is 50 feet minus the combined length of all stubs or taps—the wires connecting the actual bus to the receivers and transmitters. In some cases several units, including the processor, may plug into a single back-panel bearing the



10. **Dense package** (left). One Omnibus panel in the PDP-8/E can hold a processor with a 32,000-word memory and up to nine device controllers. More units can be put in a second panel.

11. **Pull-up** (below). Because the Omnibus lines are shorter than those of the Unibus, only a simple resistor-diode pull-up is required at one end of each line. There are 96 lines in the Omnibus.



bus conductors. Alternatively, the processor may be plugged into another panel on the same rack. In other cases a unit on the bus must be in a panel on a separate rack. Bus cables plug into the panel just as the units do, and the stub length runs from the cable plugging point to the card elsewhere on the panel. The photo Fig. 9 illustrates this arrangement.

However, this maximum length of 50 feet is obtainable only if the individual tap lengths are less than 18 inches, and if the loading is not more than one receiver and two transmitters per line. If most loads are concentrated at one end of the bus and a single load is at a distant point, the maximum length could change, provided that the crosstalk of the employed cable is low enough.

The Unibus is limited to a maximum of 20 unit loads. More than 20 would increase the leakage current to a level that, with the maximum allowable load resistance, would drop the output voltage too far below its nominal high state, and so would decrease the noise margin between the high and the low states. In applications that require more than 20 unit loads, a repeater may be installed.

In the PDP-8/E, the latest member of the PDP-8 family of small computers, another form of the unified bus is used. The PDP-8/E is a 12-bit processor with a 1.2-microsecond cycle time. With 4,096 words of memory, it sells for less than $5,000.

In the PDP-8/E the unified bus design, called Omnibus, has a maximum length of 7½ feet, and the computer functional units are densely packed—a processor with 32,000 words of memory and nine input-output controls requires only 10½ inches of panel space. This maximum length of the Omnibus, however, can accommodate two 10½-inch-high mounting boxes with 3½ feet of interconnecting cable, as Fig. 10 shows. With the dense packaging, this bus length is sufficient for the largest PDP-8/E systems.

Omnibus consists of 96 signal lines connected to each module slot. As with the Unibus, functional units like the central processor or memory can be plugged into the Omnibus in any available position. Omnibus signals are terminated at one end with a resistor-diode pull-up, as Fig. 11 shows.

Timing in the PDP-8/E, unlike the PDP-11, is synchronous, with signals generated by a timing module. Since the Omnibus is limited to a length of 7½ feet, bus delays are not a problem.

One of the most exciting aspects of the unified bus approach is the ease with which one may configure multiple processor systems. Since only one bus is necessary, very few building blocks are needed for systems that have multiple processors with shared peripherals or even a shared memory. In such configurations, the individual processors can be task oriented, and yield truly parallel processing with small computers. Moreover, because of the unified bussing, the systems, like the signal processor systems, may be field-expanded modularly and technologically updated for years to come. □

Reprinted from Electronics, December 21, 1970

PDP11/40 CUSTOMER HANDOUTS

WEEK 2

Lab machines

30

96

97

WEEK 2 PDP11/40

TIME	MON	TUES	WED	THURS	FRI
0900	INTRO TO μ PROGRAM DEVICE	KD11 FLOWS	KD11 FLOWS & DISPLAY RULES	LOGIC ANALYSIS	LOGIC ANALYSIS
1000	SUB COMMANDS AND BLOCK DIAGRAM USAGE	MICRO BRANCHING	KD11 CONSOLE OPERATIONS	POWER UP POWER DWN STACK OVFW ODA ERROR	DATA XFER BR SEQ NPR SEQ
1100					
1200	L U N C H				
1300	U WORD FORMAT & CLOCK CKT ANALYZE	KD11 FLOWS & SINGLE CLOCKING KM11	KD11 FLOWS SERVICE & TRAPS	PROBLEM ISOLATION & BACK PLANE	MM-11 MEMORY
1400					
1500	MUXS & GPR ADRS SWITCH	LAB	LAB	TROUBLE SHOOTING LAB	
1600	INTRO TO KD11 FLOWS				REVIEW & COURSE CRITIQUE
1700					

CORRECTION SHEET

1- CHANGE DOPØ7 (SHEET 8) FLOWS FROM DISPLAY D TO DISPLAY R[SF]

2- ADD THE TERM BUTØ6 TO CON19 (SHEET 11) FLOWS

3- REVISE MODULE UTILIZATION KM11A BASIC-F1 OPTION E1

4- BA MUX LOW ORDER INPUTS ARE REVERSED

5-

6-

7-

8-

9-

10-

11-

12-

13-

14-

15-

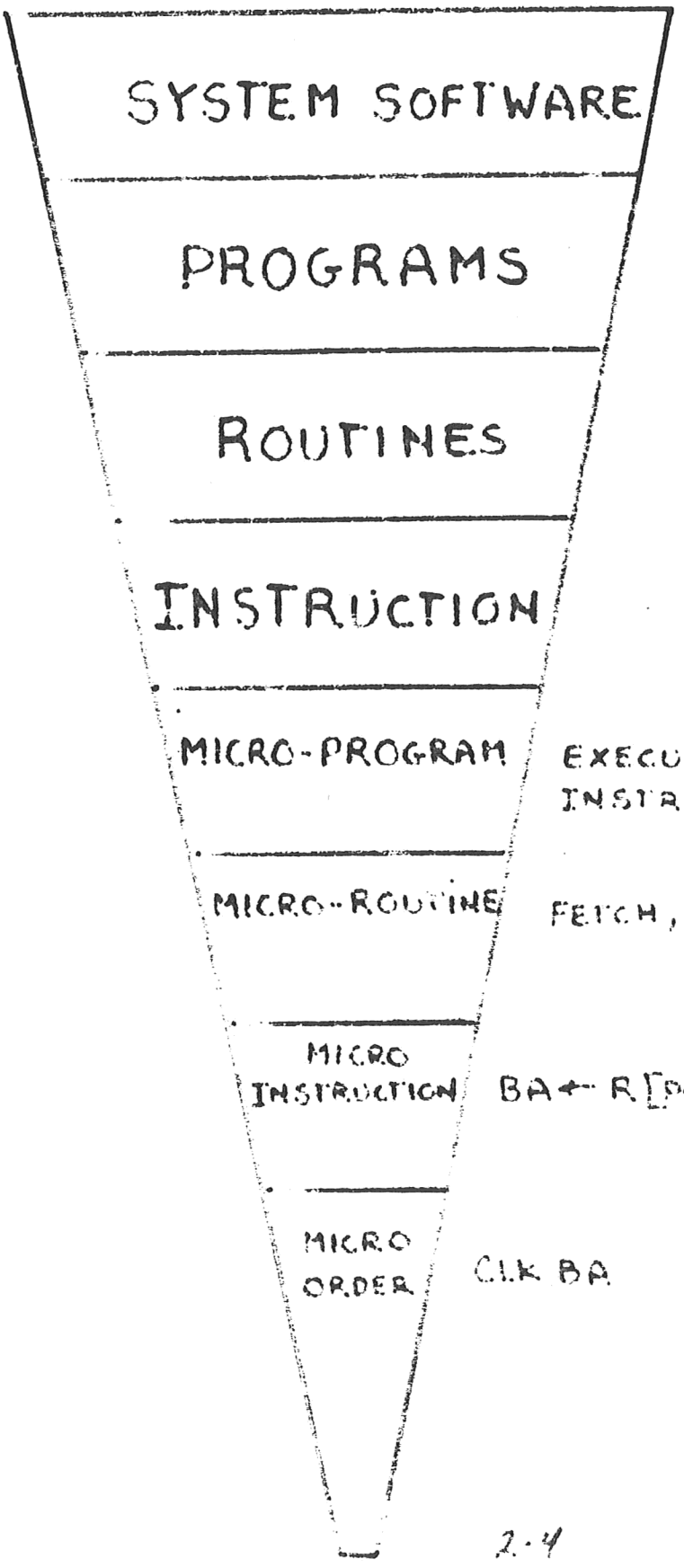
16-

17-

18-

DAY 1

MICRO-PROGRAM 1
INTRODUCTION.



KDJ-11A SUBCOMMAND LIST

GENERAL REGISTERS

RIF (3:Ø) = $ØØ_8$
= $Ø1_8$
= 16_8
RIF (3:Ø) = 17_8

RA (3:Ø) ←RIF (3:Ø)
" " ← IR (8:6)
" " ← IR (2:Ø)
" " ← BA (3:Ø)

READ ←1
WRITE ←1

R(15:Ø8)←DMUX(15:Ø8)
R(Ø7:ØØ)←DMUX(Ø7:ØØ)

ARITHMETIC LOGIC UNIT

AM (15:ØØ)←R(15:ØØ)

BIN(15:ØØ)←BREG(15:ØØ)
BIN(15:ØØ)←BCON(15:ØØ)
BIN(15:Ø8)←BREG(15:Ø8)
BIN(Ø7:ØØ)←BREG(Ø7:ØØ)
BIN(15:Ø8)←BREG(Ø7:ØØ)
BIN(Ø7:ØØ)←BREG(15:Ø8)
BIN(15:ØØ)←BREG(Ø7:ØØ) SEX

BREG(15:ØØ)←DMUX(15:ØØ)

Ain (7:Ø)——PS (7:Ø)

BCON (15:ØØ)←1
←2
←4
←1Ø
BCON (15:ØØ)←177570

Cin←1 Cin←Ø

ALUout = A PLUS CIN
ALUout = $\overline{A}$ PLUS CIN
ALUout = A PLUS B PLUS CIN
ALUout = A MINUS B
ALUout = B PLUS CIN

ALU output

D(15:ØØ)←ALUout (15:ØØ)
DMUX(15:ØØ)←D(15:ØØ)
DMUX(15:ØØ)←R(15:ØØ)
DMUX(15:ØØ)←BUS D (15:ØØ)

DMUX (15:ØØ)←COUT,D (15:Ø1)
PS (7:Ø)←DMUX (7:Ø)
IR (15:ØØ) ←DMUX (15:ØØ)
BA (15:ØØ) ←R(15:ØØ)
BA (15:ØØ) ←ALUout (15:ØØ)

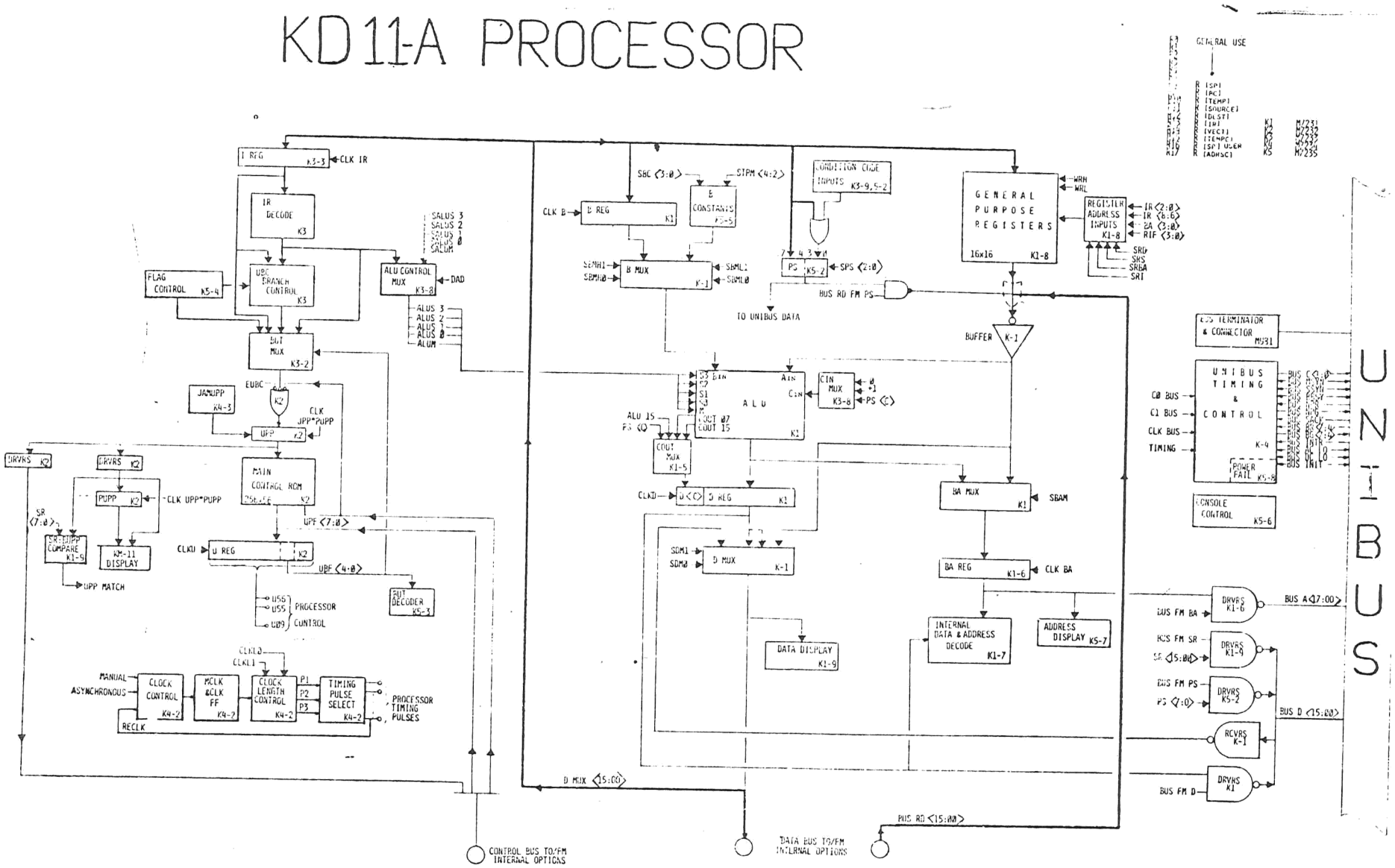
CONTROL AND TIMING

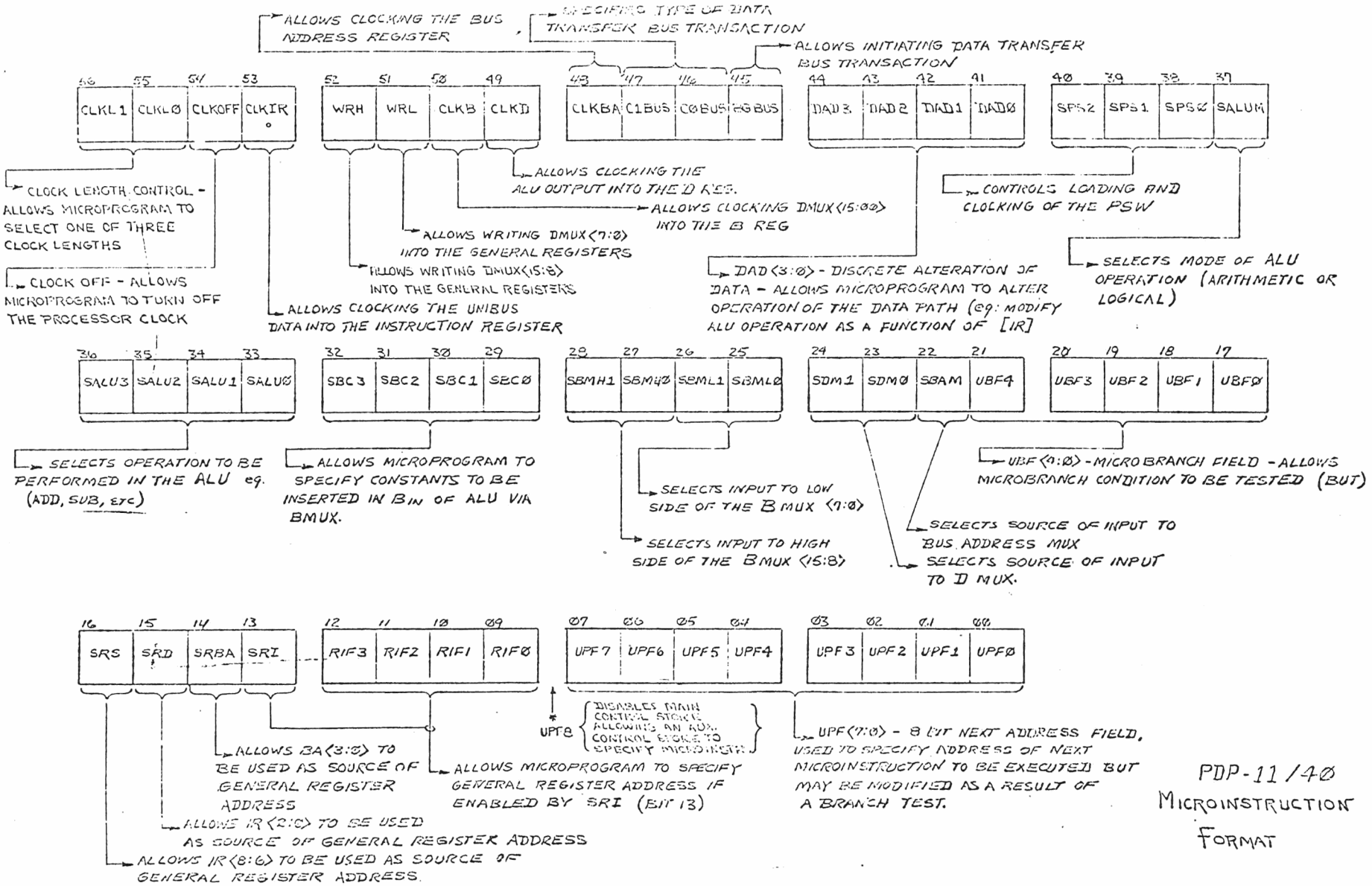
BUS C (1:Ø)←Ø
" "←1
" "←2
" "←3
CLKL←1
" "←2
" "←3
CLKOFF←1

BUS BBSY←1
BUS MSYN←1
BUS A (15:ØØ)←BA (15:ØØ)
BUS A (17:16)←BA (15:13)

BUS D (15:ØØ)←D (15:ØØ)
BUS D (15:ØØ)←Ø,PS (7:Ø)
BUS D (15:ØØ)←SR (15:ØØ)

KD11-A PROCESSOR





PDP-11/40 MICROINSTRUCTION FORMAT

TRNG 3/27/72

SUB-COMMAND EXAMPLES

USING THE KD-11 BLOCK DIAGRAM AND THE SUB COMMAND LIST, WRITE THE SUB-COMMAND SETS REQUIRED TO IMPLEMENT THE FOLLOWING

1. BA ← UNIBUS DATA

2. R[6] ← R[6] MINUS 2

ANSWER TO SUB-COMMAND [illegible]

1. BA ← UNI-BUS DATA

"CLKL 1"

DMUX⟨15:00⟩ ← BUS D⟨15:00⟩

(P1) B⟨15:00⟩ ← DMUX⟨15:00⟩

CLKL2

BMUX⟨15:00⟩ ← B⟨15:00⟩

BIN⟨15:00⟩ ← BMUX⟨15:00⟩

AIN = Ø

CIN = Ø

ALUOUT = BIN + AIN + CIN

BAMUX⟨15:00⟩ ← ALUOUT⟨15:00⟩

(P2) BA⟨15:00⟩ ← BMUX⟨15:00⟩

2. R[6] ← R[6] MINUS 2

"CLKL 3"

RIF⟨3:Ø⟩ ← 6

RA⟨3:Ø⟩ ← RIF⟨3:0⟩

READ ← 1

AIN⟨15:00⟩ ← R⟨15:00⟩

CIN = Ø

BCON⟨15:00⟩ ← 2

BMUX⟨15:00⟩ ← BCON⟨15:00⟩

BIN⟨15:00⟩ ← BMUX⟨15:00⟩

ALUOUT = A MINUS B

(P2) D⟨15:00⟩ ← ALUOUT⟨15:00⟩

DMUX⟨15:00⟩ ← D⟨15:00⟩

R⟨15:00⟩ ← DMUX⟨15:00⟩

(P3) WRITE ← 1

CLK = 6 (P3)

WR = 3

CD = 1

ALU = Ø6

SBC = 2

SBM = 17

SDM = 2

SRX = 1

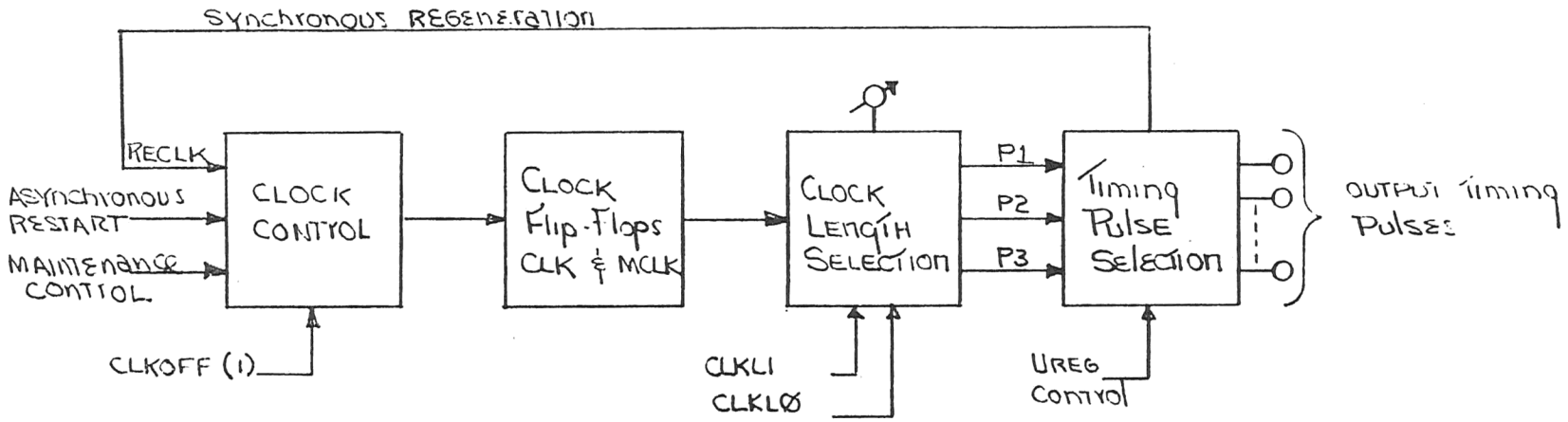
RIF = 6

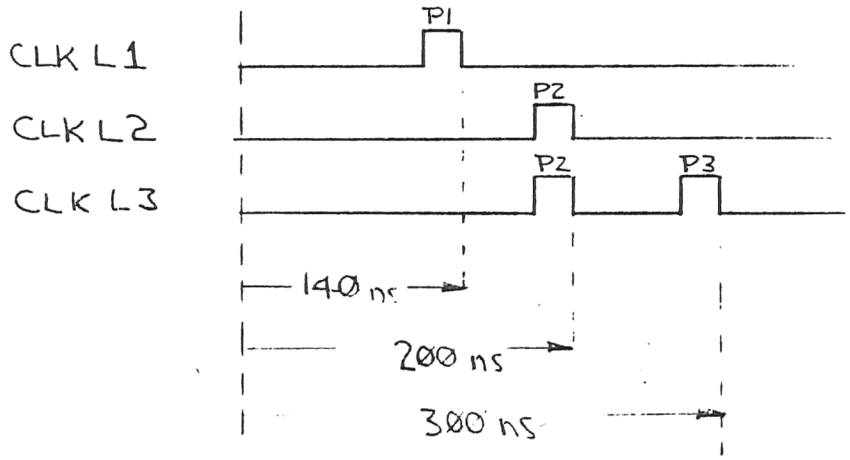
PDP 11/45 MICROINSTRUCTION CODING WORKSHEET

Ø - INACTIVE 1 - ACTIVE

MICROINSTRUCTION	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
1. FET Ø2 (1)	Ø	1	1	Ø	Ø	Ø	Ø	Ø	1	Ø	Ø	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	1	Ø	1	1	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	1
2.																																																										
3.																																																									
4																																																									
5.																																																									
6.																																																									
7																																																									
8.																																																									
9.																																																									
10																																																									
11.																																																									
12.																																																									

B-2





KIIIA

Processor Clock

Block Diagram

BASIC TIMING

PURPOSE:

USED TO GATE INFORMATION INTO REGISTERS AND TO STEP THE MICRO-PROGRAM THRU ITS SEQUENCES:

SELECTION OF CLOCK LENGTH:

THERE ARE THREE UNIQUE CLOCK LENGTHS THAT MAY BE SELECTED BY THE MICRO-PROGRAM. EACH MICRO-WORD WILL SELECT ONE OF THE FOLLOWING CLOCK LENGTHS DEPENDING ON WHAT THAT MICRO-WORD IS TRYING TO DO. THE MICRO-WORD WILL ALWAYS SELECT THE SHORTEST CLOCK LENGTH THAT WILL ACCOMPLISH THE TASK SPECIFIED IN THAT MICRO-WORD.

RECYCLING THE CLOCK:

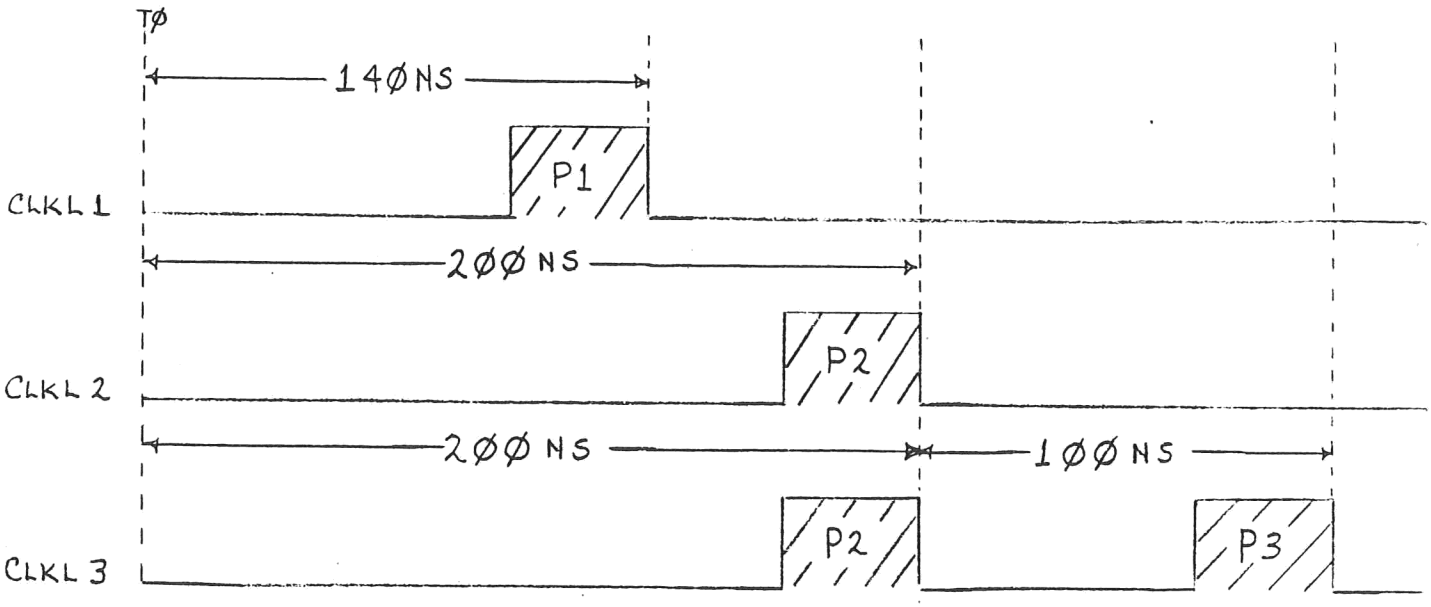
THE CLOCK WILL AUTO-MATICALLY RESTART ITSELF ON THE TRAILING EDGE OF THE LAST CLOCK PULSE PERMITED IN THE PRECEEDING CLOCK CYCLE. RESTART IS ASSUMED UNLESS THE MICRO-WORD CHOOSES TO STOP THE CLOCK (IE DATA BUS CYCLE)

GENERAL RULES FOR CHOOSING CLOCK LENGTHS:

SINCE THE MICRO-PROGRAM CAN BE STEPPED ALONG BY THE PRESENCE OF ANY OF THE THREE CLOCK LENGTHS THE ONLY FACTOR IN SELECTING CLOCK LENGTHS IS WHICH REGISTER IS INVOLVED IN THIS MICRO-WORD EXECUTION CERTAIN REGISTERS REQUIRE PARTICULAR CLOCK LENGTHS TO BECOME ACTIVE. THE FOLLOWING IS THE REQUIRED CLOCK LENGTH THAT WILL ACTIVATE THE REGISTERS.

LOADING BA REG	CLKL 1 OR CLKL 2
LOADING D REG	CLKL 2
LOADING GPR REG	CLKL 1 OR CLKL 3
LOADING IREG AND BREG	CLKL 1 OR CLKL 3

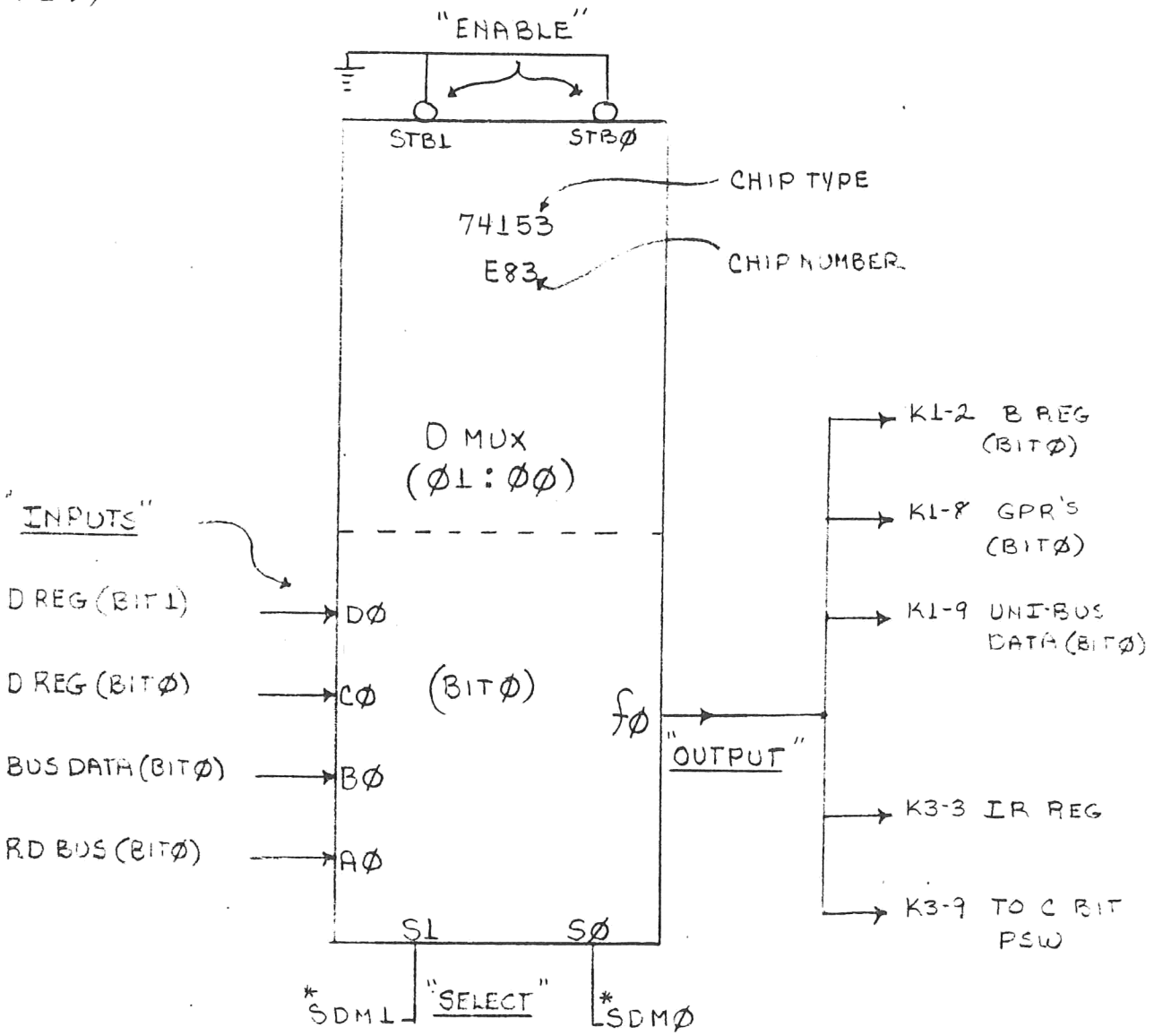
CLOCK CKT OUTPUT



NOTE: WHENEVER CLKL3 IS SPECIFIED BY MICRO-WORD P2 AND P3 ARE BOTH AVAILABLE AT CLOCK OUTPUT.

D MULTIPLEXOR

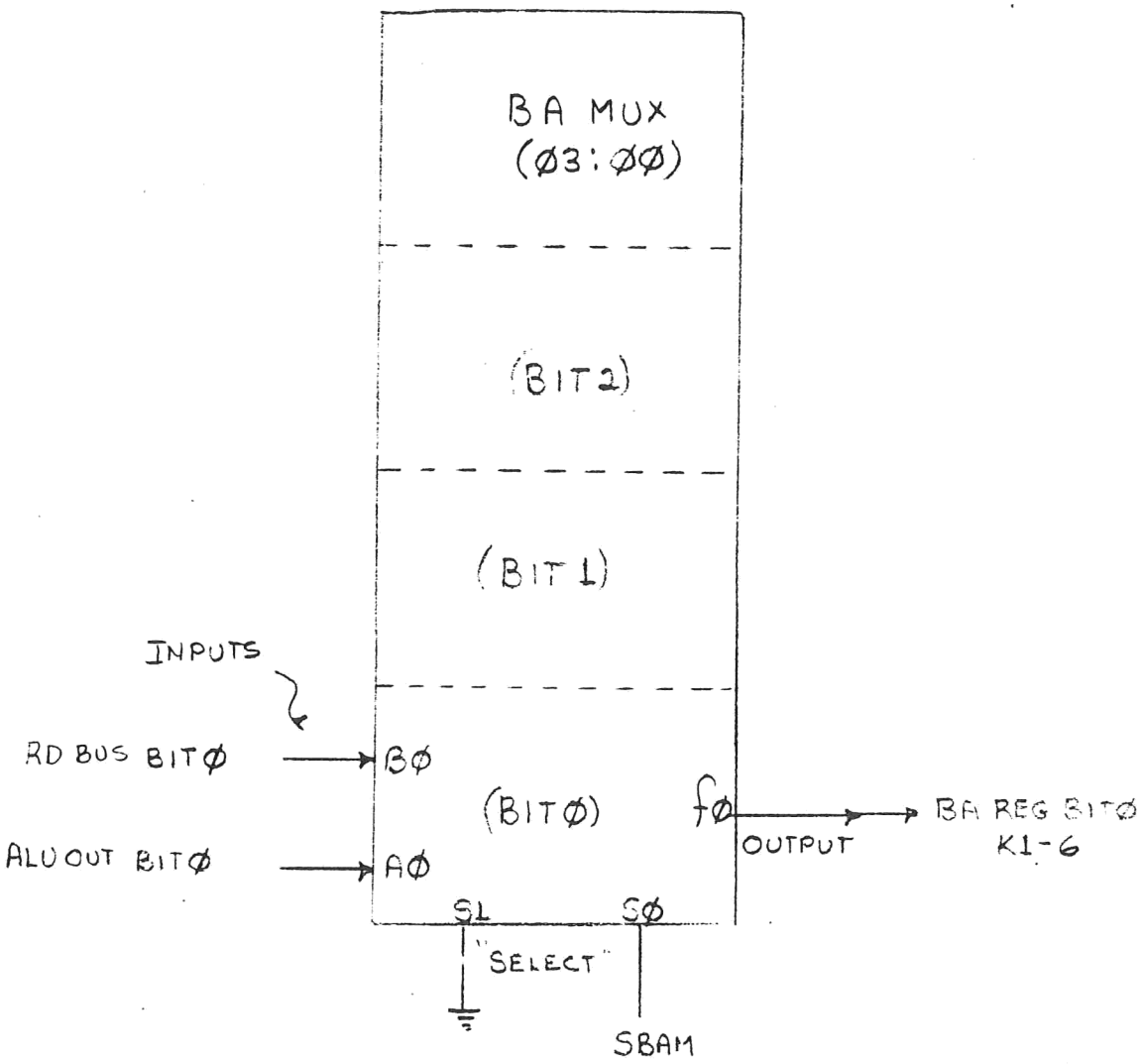
(SHEET K1-2)



S1	SØ	OUTPUT
Ø	Ø	RD BUS BIT Ø
Ø	1	BUS DATA BIT Ø
1	Ø	D REG BIT Ø
1	1	D REG BIT 1

* SDM1/SDMØ = SELECT D MUX BITS FROM MICRO-WORD

BA MULTIPLEXOR
(SHEET K1-2)

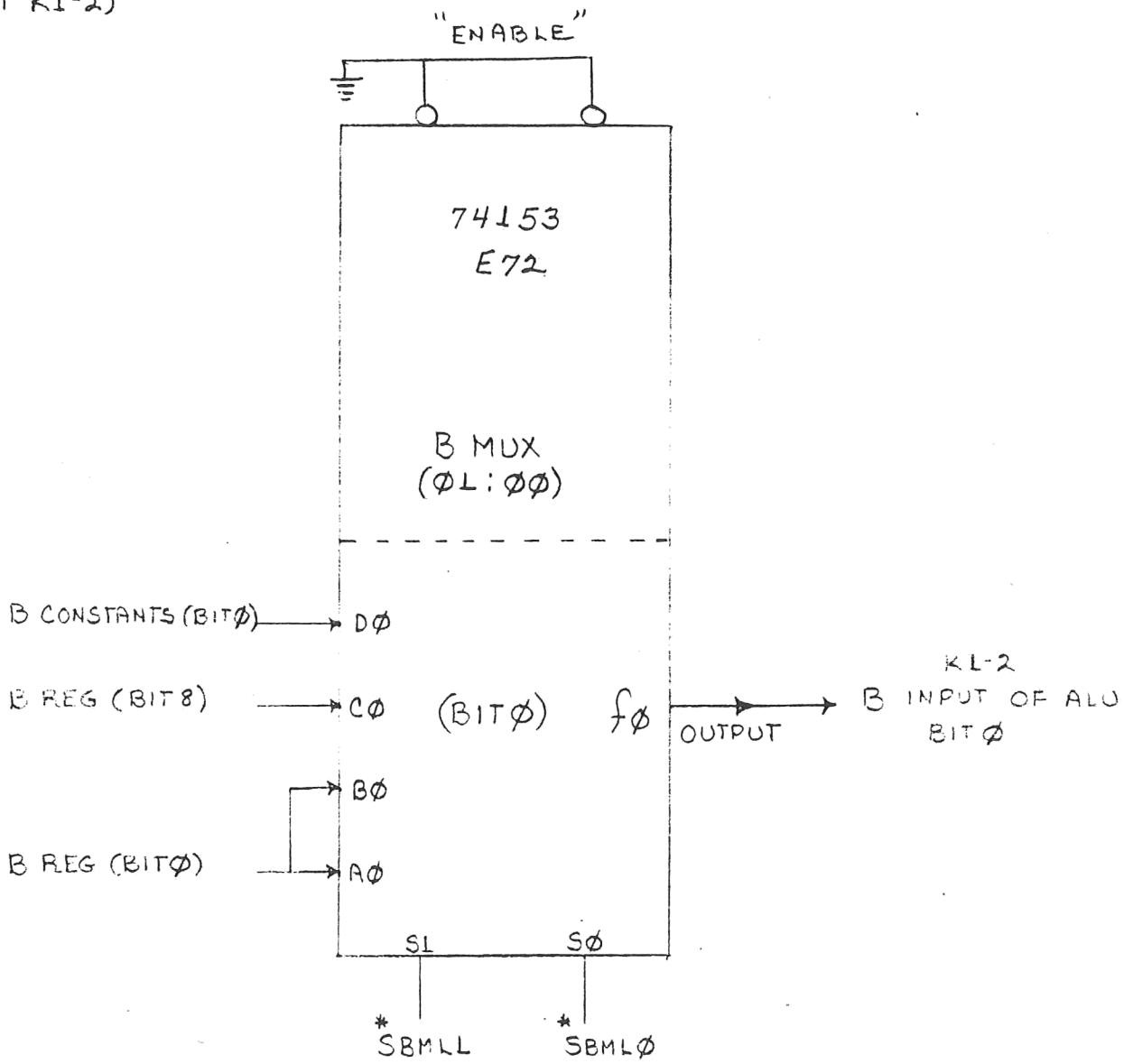


SBAM	OUTPUT
Ø	ALU OUT
1	RD BUS

NOTE: THIS MULTIPLEXOR HAS NO ENABLE

B MULTIPLEXOR

(SHEET K1-2)

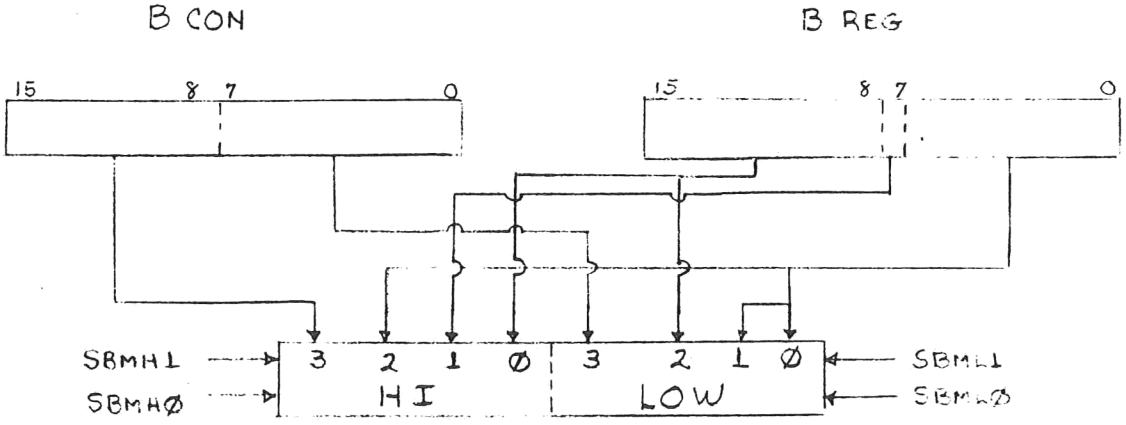


S1	SØ	OUTPUT
Ø	Ø	B REG BITØ
Ø	1	B REG BITØ
1	Ø	B REG BIT 8
1	1	B CONSTANT BITØ

* SBMLL/SBMLØ = SELECT B MUX LO BYTE FROM MICRO-WORD

SBMHL/SBMHØ ARE USED TO CONTROL INPUTS TO B MUX BITS 8→15 SEE SHEET K1-4

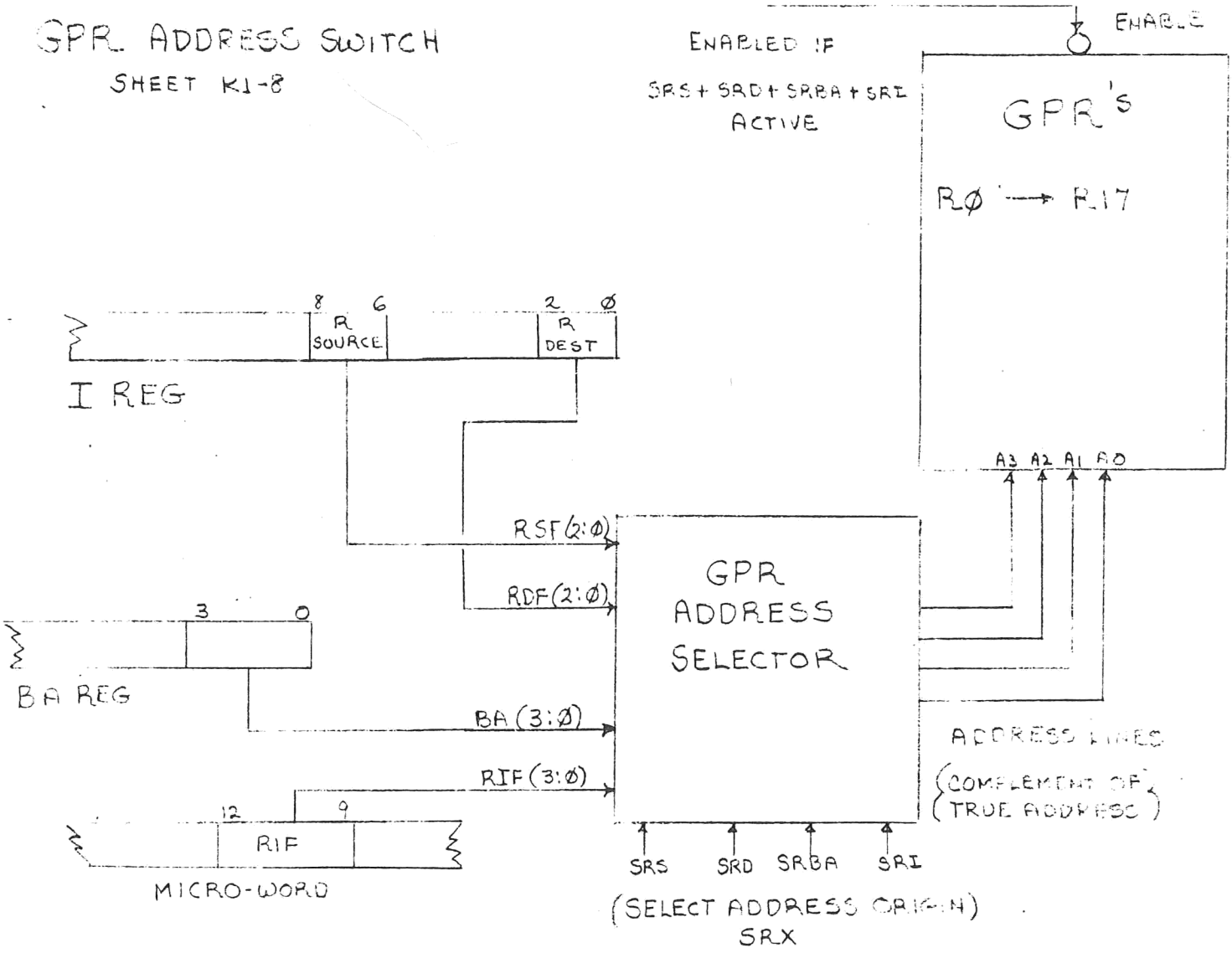
B MUX



	SBMHL	SBMHØ	SBMLL	SBMLØ	
SBM ØØ	Ø	Ø	Ø	Ø	= B REG TO ALU
SBM Ø2	Ø	Ø	L	Ø	= DUPLICATE UPPER BYTE
SBM Ø5	Ø	L	Ø	L	= SIGN EXTEND
SBM 1Ø	L	Ø	Ø	Ø	= DUPLICATE LOW BYTE
SBM 12	L	Ø	L	Ø	= SWAP BYTES
SBM 17	L	L	L	L	= B CON TO ALU

GPR ADDRESS SWITCH

SHEET K1-8



	SRS	SRD	SRBA	SRI	
SRX-ØØ	Ø	Ø	Ø	Ø	NO SELECTION
SRX=Ø1	Ø	Ø	Ø	1	SELECT GPR FROM MICRO-WORD
SRX=Ø2	Ø	Ø	1	Ø	SELECT GPR FROM BA REG
SRX=Ø4	Ø	1	Ø	Ø	SELECT GPR FROM I REG [DF]
SRX-1Ø	1	Ø	Ø	Ø	SELECT GPR FROM I REG [SF]

REVIEW QUIZ
DAY 1

1. ASSUME THE PDP11-40 IS EXECUTING THE BELOW LISTED MICRO-INSTRUCTION, WHAT WOULD BE THE STATE OF SBMH1 AND SBMH0

D ← R[PC] PLUS 2

A. 1, 1
B. 0, 0
C. 1, 0
D. 0, 1

2. ASSUME THE BELOW LISTED MICRO-INSTRUCTION IS UNDER EXECUTION, WHAT IS THE STATE OF THESE BITS IN THE UREG? UREG<16:09>

BA ← R[DEST]

A. 0 0 0 1 0 0 0 1
B. 0 0 0 1 1 0 1 0
C. 0 0 0 1 0 1 0 1
D. 0 0 1 0 0 0 1 0

3. UREG<37:33> = 01100, WHAT IS THE ALU FUNCTION

A. A+B CIN=0
B. A PLUS A CIN ← PS(C)
C. A PLUS A CIN ← 1
D. A CIN=0

4. ROM LOCATION ØØ6 SPECIFIES THAT THE _____ REGISTER SHALL BE DISPLAYED IN THE DATA LIGHTS OF THE CONSOLE. (REFER TO UWORD TABLES)

A- IR REGISTER
B- D REGISTER
C- GPR R-14
(D)- GPR R-13

5. WHICH OF THE BELOW ROM LOCATIONS SPECIFIES CLOCK LENGTH 3, DATI, CLOCK OFF

A. 3ØC
B. 15Ø
C. Ø4Ø
D. 1ØL

6. REFER TO OUTPUT OF THE CLOCK ON K4-2. THE SIGNAL PART P END H IS ACTIVE ON

A. P1 + P2 + P3
B. P1 + P3
(C). P2 + P3
D. P1 + P2

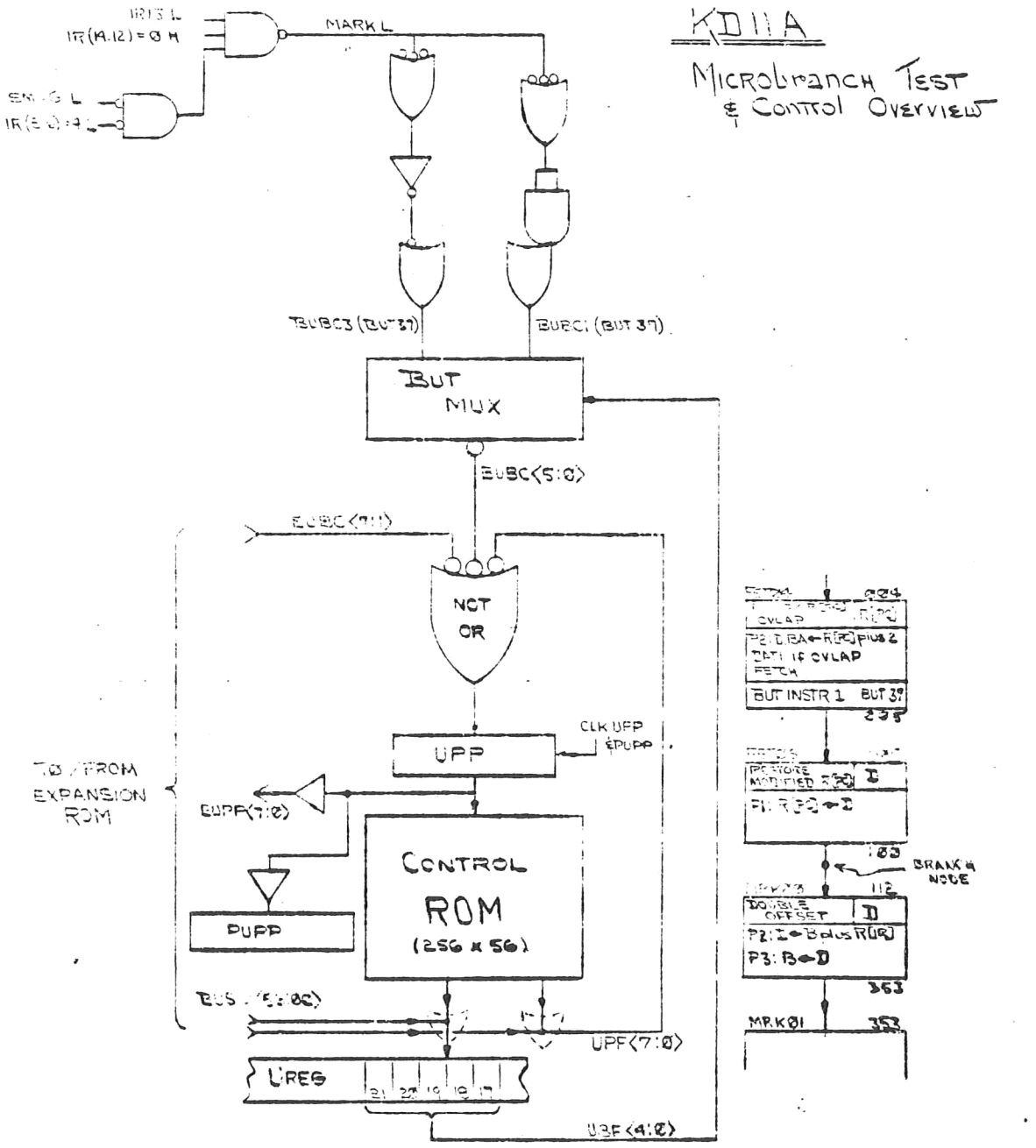
DAY 2

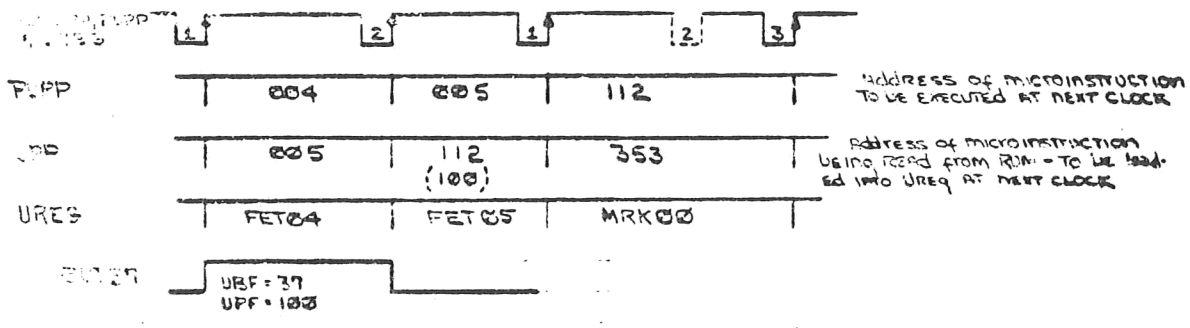
IR = MARK

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Z	Ø	Z	Z	1	1	Ø	1	Ø	Z	x	x	x	x	x	x

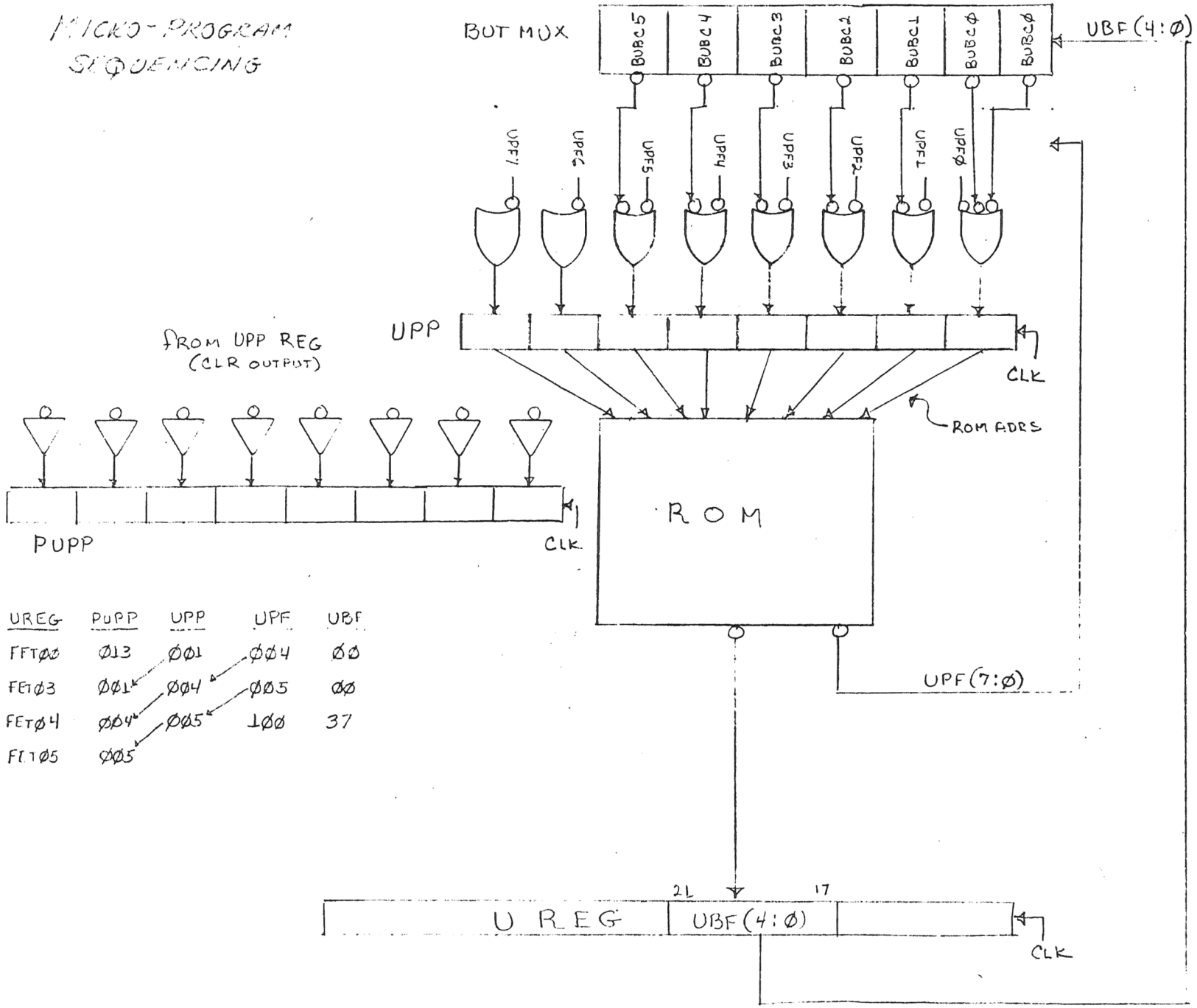
KD11A

Microbranch Test & Control Overview



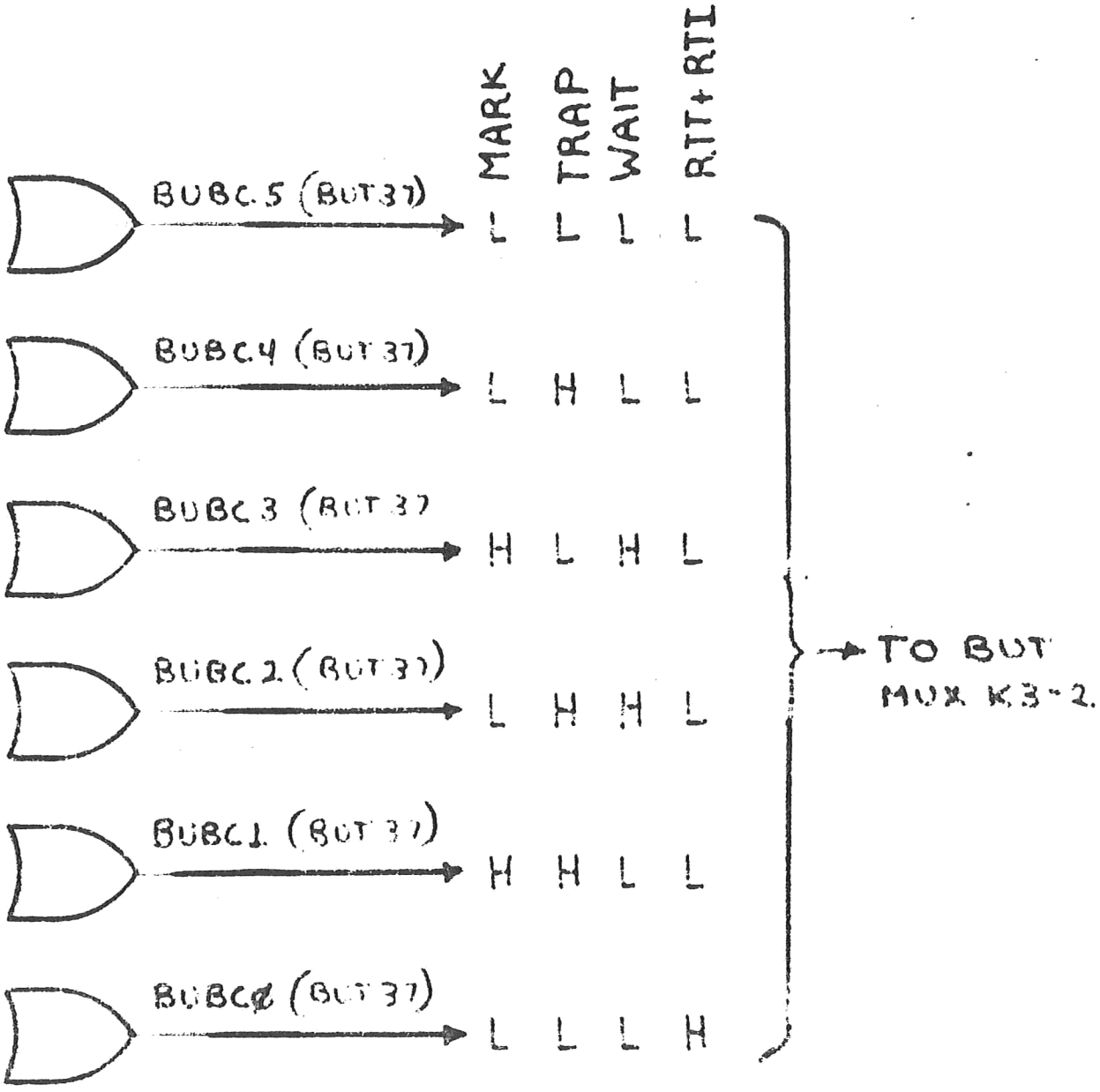


MICRO-PROGRAM SEQUENCING

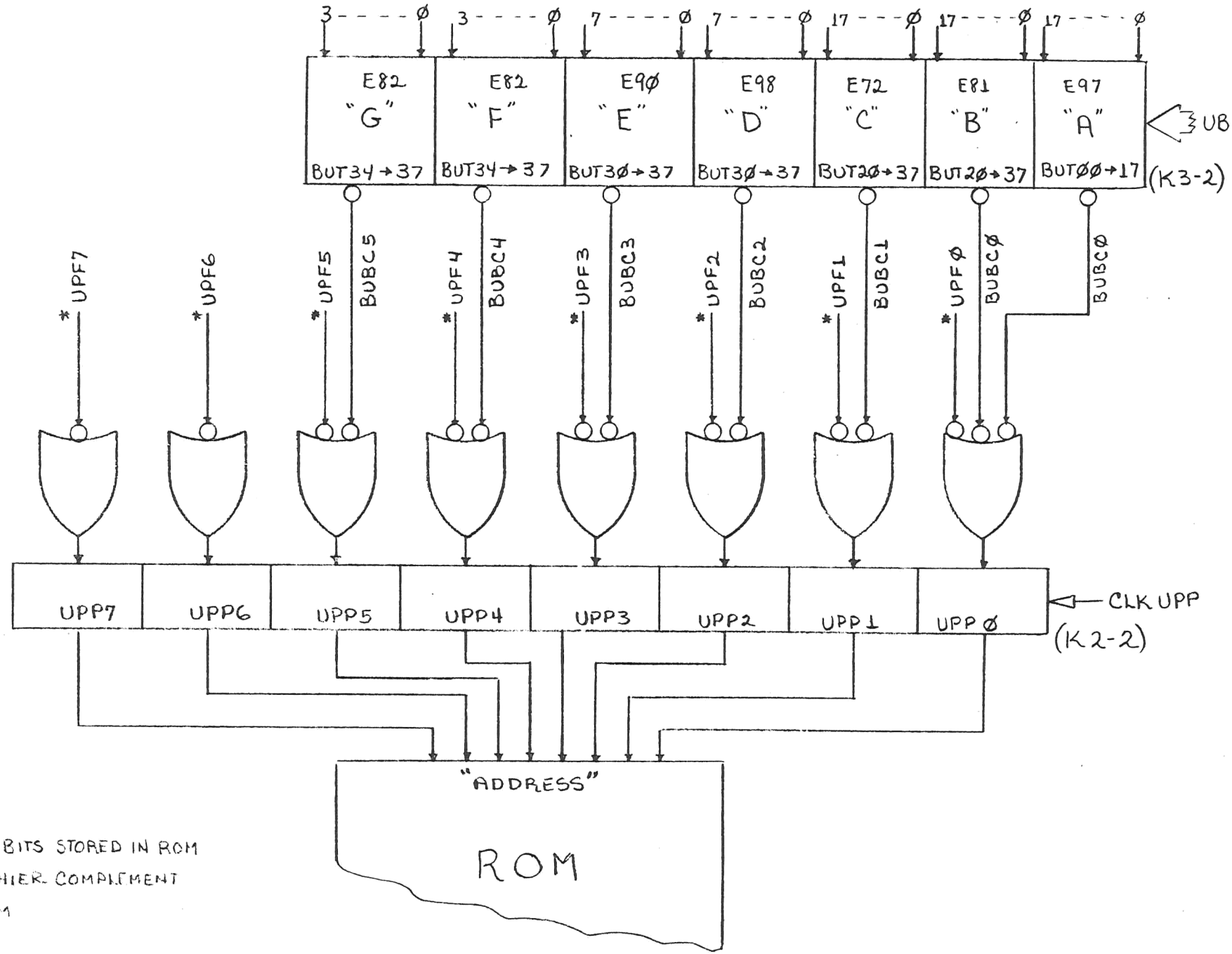


UREG	PUPP	UPP	UPF	UBF
FFT00	013	001	004	00
FET03	001	004	005	00
FET04	004	005	100	37
FET05	005			

OUT PUT OF IR
DECODE TO CONTROL BUT MUX [K3-5]



MICRO-BRANCHING



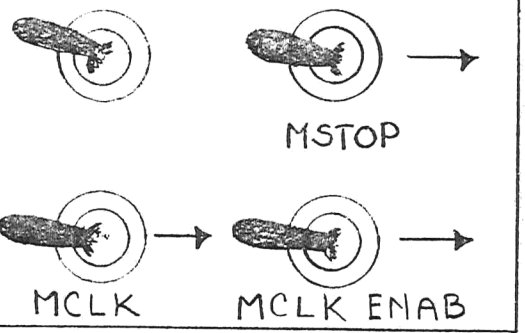
* NOTE:
UPF BITS STORED IN ROM IN THIER COMPLEMENT FORM

KD11-A

MAINTENENCE MODULE

{PLUGS INTO SLOT F1 OF KD11A}

OCTAL Value	NOW			NEXT			
0	PUPP 6	PUPP 3	PUPP Ø	BUPP 6	BUPP 3	BUPP Ø	C
1	PUPP 7	PUPP 4	PUPP 1	BUPP 7	BUPP 4	BUPP 1	V
2	PUPP 8	PUPP 5	PUPP 2	BUPP 8	BUPP 5	BUPP 2	Z
			TRAP	SSYN	MSYN	T	N



INDICATORS

SWITCHES

PUPP ⟨8:Ø⟩ = ADDRESS OF THE MICRO-INSTRUCTION STORED IN U REG.

BUPP ⟨8:Ø⟩ = ADDRESS OF NEXT MICRO-INSTRUCTION BEING READ OUT

MSTOP = ALLOWS STOPPING THE PROCESSOR WHEN MATCH BUPP = SR ⟨8:Ø⟩

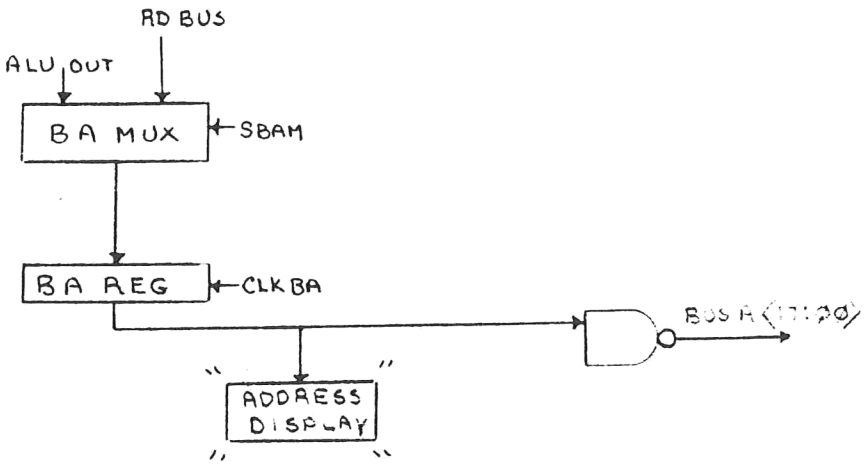
MCLK ENAB = DISABLES PROCESSOR CLOCK AND ALLOWS SINGLE CLOCKING THE MICRO-PROGRAM WITH MCLK SWITCH

MCLK = SINGLE CLOCK SWITCH, EACH TIME ON GENERATES ONE CLOCK CYCLE

DAY 3

SINGLE CLOCK DISPLAY RULES

ADDRESS DISPLAY

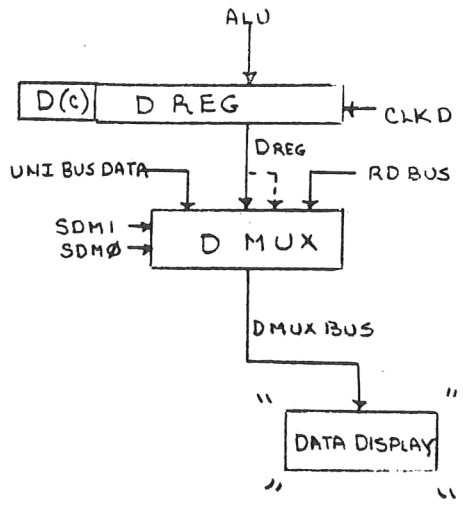


SINCE ADDRESS DISPLAY LIGHTS ARE FED DIRECTLY BY CONTENTS OF BA REG YOU NEED ONLY DETERMINE THE CONTENTS OF BA REG.

RULE:

IF MICRO-WORD SPECIFIES BA←X, YOU WILL NOT SEE X IN BA NOR WILL X BE DISPLAYED IN THE ADDRESS LIGHTS UNTIL THE NEXT CLOCK PULSE. THE ADDRESS LIGHTS WILL DISPLAY THE CONTENTS OF BA REG ESTABLISHED IN THE LAST MICRO-WORD THAT ALTERED THE CONTENTS OF BA REG.

DATA DISPLAY



THE DATA DISPLAY LIGHTS ARE DRIVEN BY THE OUTPUT OF THE D MUX SINCE THE D MUX CAN BE SELECTED TO OUTPUT FROM ONE OF FOUR INPUTS, THE MICRO-WORD FLOWS WILL INDICATE WHICH INPUT LINE IS BEING DISPLAYED THE DATA DISPLAY SOURCE IS INDICATED IN THE SMALL BOX, UPPER RIGHT OF EACH MICRO-WORD OF THE FLOWS.

RULES:

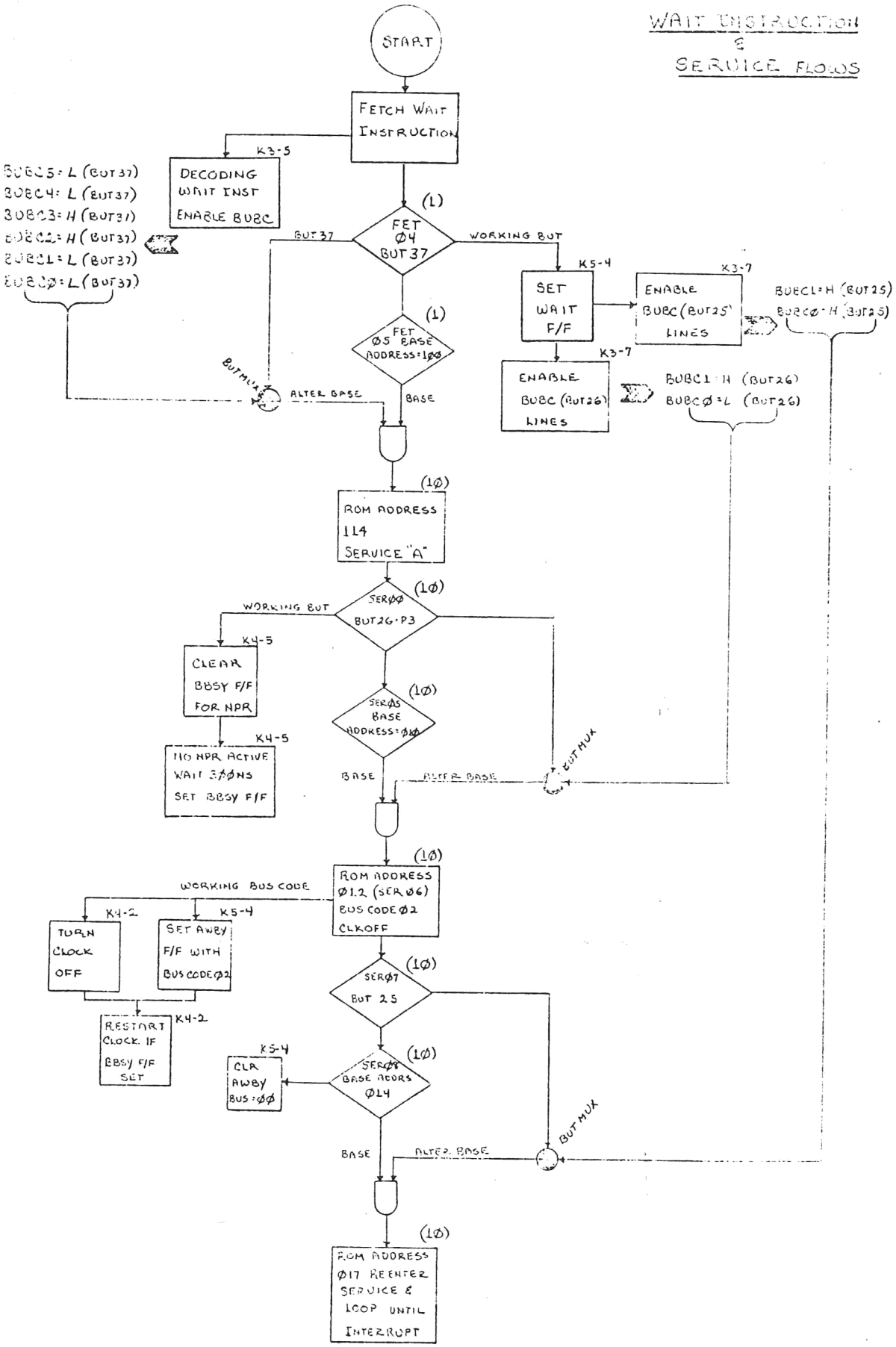
IF THE DATA DISPLAY INDICATOR, IN THE MICRO-WORD, SPECIFIES UNI-BUS DATA OR ANY OF THE 16 GPRS, THAT DATA WILL BE DISPLAYED IN THAT MICRO-WORD.

IF THE DATA DISPLAY INDICATOR, IN THE MICRO-WORD, SPECIFIES DISPLAY D AND THAT MICRO-WORD ALSO SPECIFIES D ← X, X WILL NOT BE SEEN IN D NOR WILL IT BE DISPLAYED UNTIL THE NEXT CLOCK PULSE. THE DATA LIGHTS WILL DISPLAY THE VALUE IN D ESTABLISHED IN THE LAST MICRO-WORD THAT ALTERED THE CONTENTS OF D.

KM-11 DISPLAY

IF THE MICRO-WORD SPECIFIES DATI OR DATO, MSYN LIGHT ON THE KM11 MAINTENENCE MODULE WILL NOT LIGHT UNTIL NEXT CLOCK PULSE. YOU WILL NOTE THAT SSYN ON MAINTENENCE MODULE SHOULD ALSO BE LIGHTED WHEN MSYN IS ON.

WAIT INSTRUCTION & SERVICE FLOWS



BUBC1 & BUBCØ TRANSLATOR (K3-7)

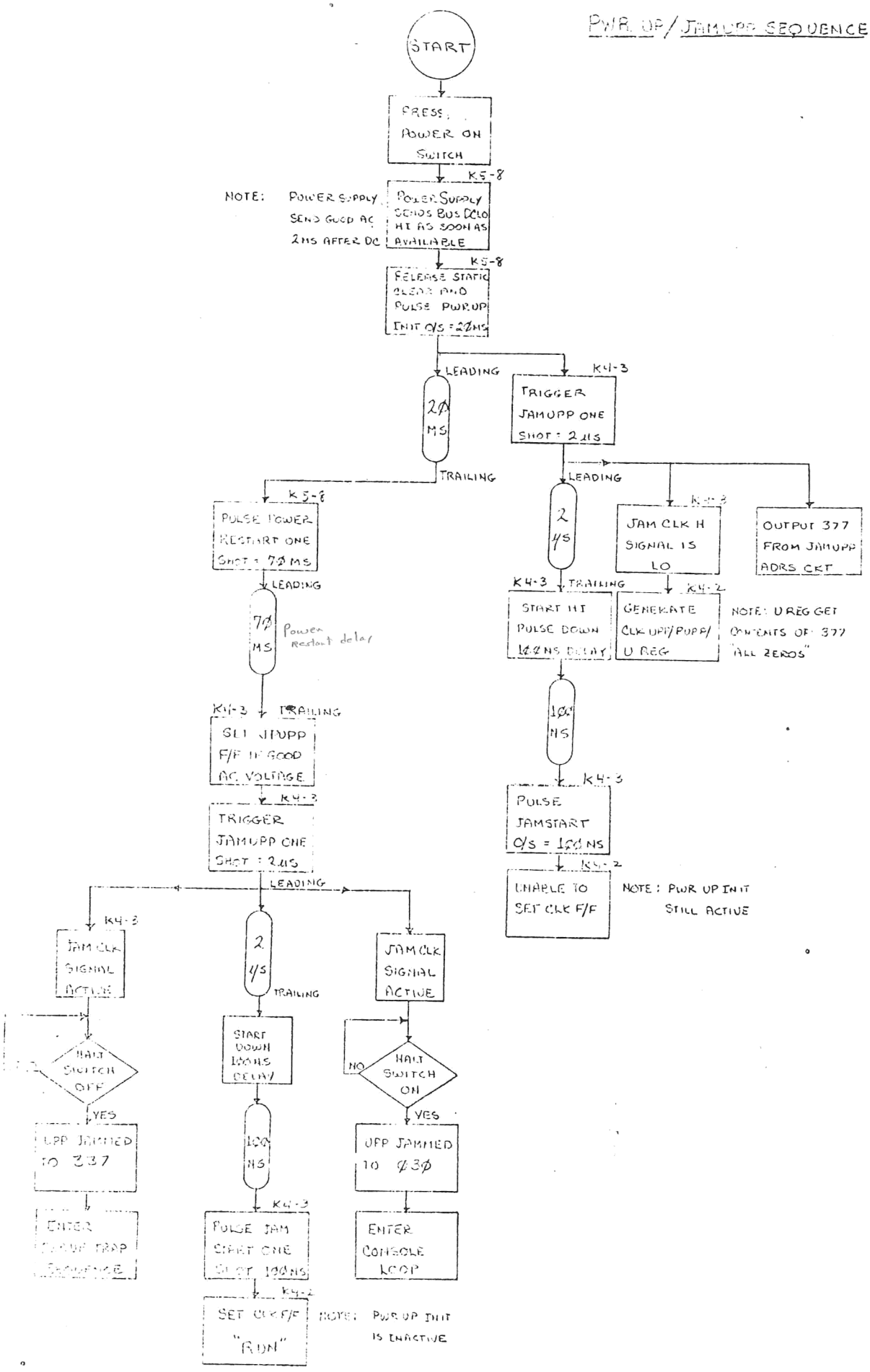
(FOR USE IN SERVICE FLOW)

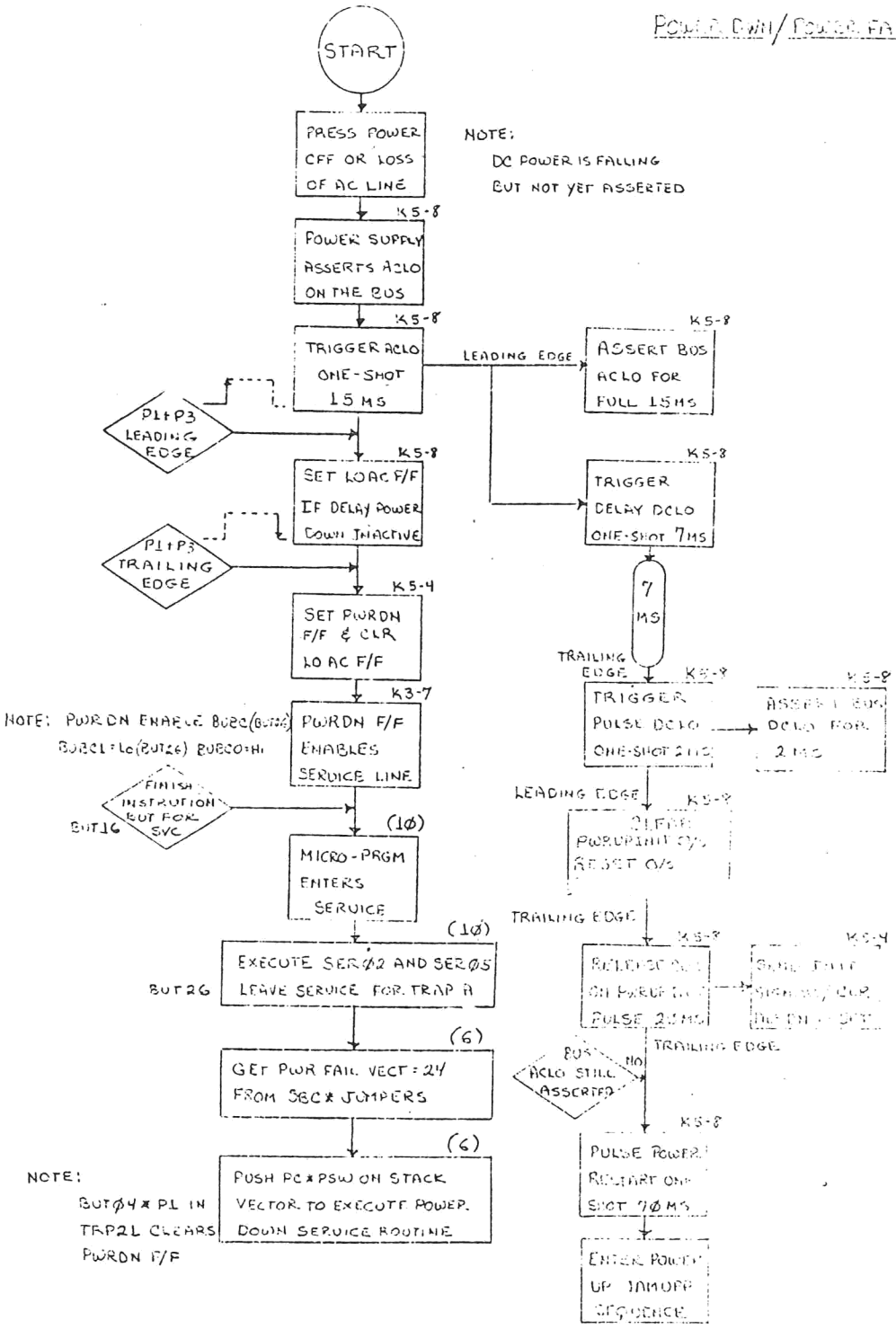
CONDITION	BUBC1 (BUT26)	BUBCØ (BUT26)	BASE ADDRS	BRANCH ADDRS
BERR PS(T) OVFLW PWRDWN	Lo	Lo	Ø1Ø	Ø1Ø TRAP A
CBR	Lo	Hi	Ø1Ø	Ø11 CONSOLE B
BR+WAIT	Hi	Lo	Ø1Ø	Ø12 SERØ6
No RQST	Hi	Hi	Ø1Ø	Ø13 FETCH A

CONDITION	BUBC1 (BUT25)	BUBCØ (BUT25)	BASE ADDRS	BRANCH ADDRS
BR	Lo	Lo	Ø14	Ø14 SERØ9
WAIT	Hi	Hi	Ø14	Ø17 SERVICE C
No RQST	Hi	Lo	Ø14	Ø16 FETCH C

DAY 4

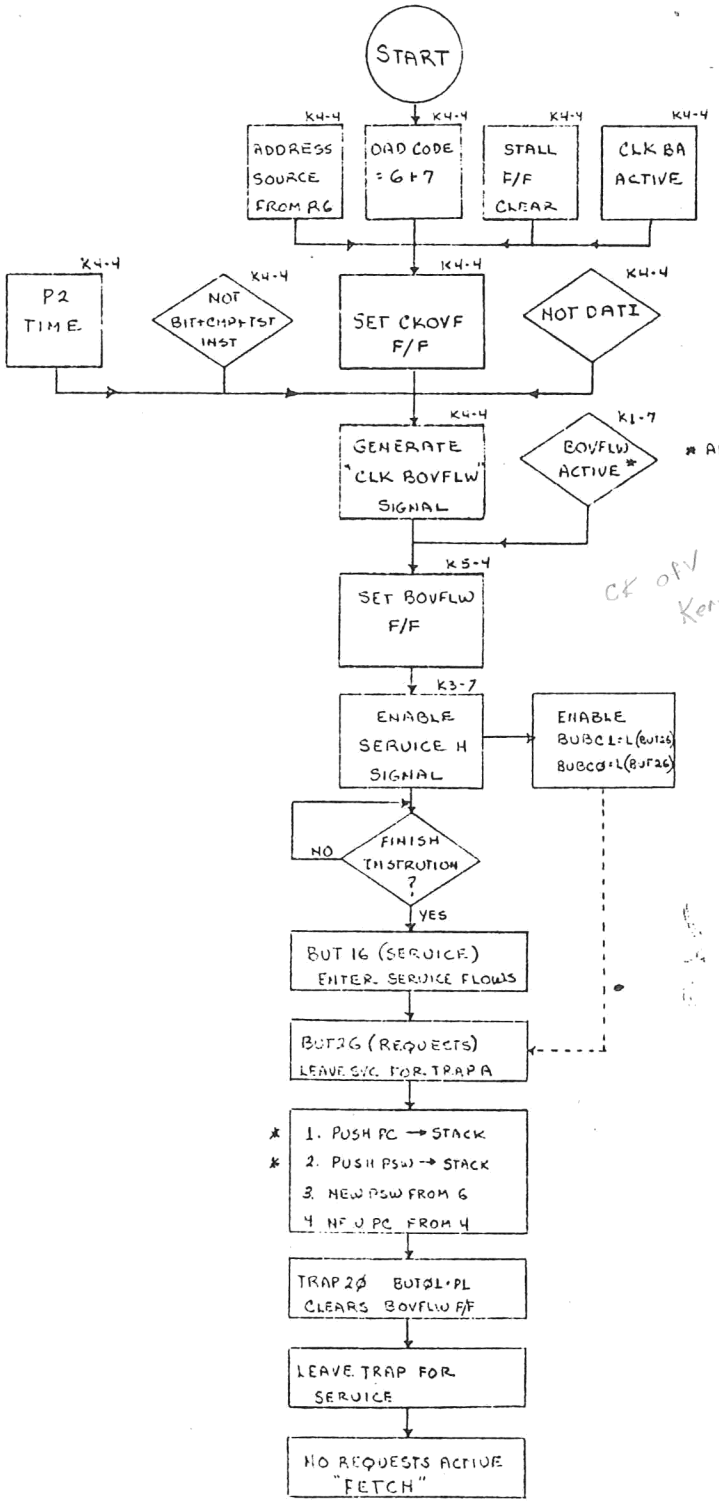
PWR UP/JAMUPP SEQUENCE




START
PRESS POWER OFF OR LOSS OF AC LINE
NOTE:
DC POWER IS FALLING
BUT NOT YET ASSERTED
K5-8
POWER SUPPLY ASSERTS ACLO ON THE BUS
K5-8
TRIGGER ACLO ONE-SHOT 15 MS
LEADING EDGE
K5-8
ASSERT BUS ACLO FOR FULL 15MS
P1+P3 LEADING EDGE
K5-8
SET LOAC F/F IF DELAY POWER DOWN INACTIVE
K5-8
TRIGGER DELAY DCLO ONE-SHOT 7MS
P1+P3 TRAILING EDGE
K5-4
SET PWRDN F/F & CLR LO AC F/F
7 MS
TRAILING EDGE
K5-8
TRIGGER PULSE DCLO ONE-SHOT 2 MS
K5-8
ASSERT BUS DCLO FOR 2 MS
K3-7
PWRDN F/F ENABLES SERVICE LINE
NOTE: PWRDN ENABLE BUS2(BUT26) BUS1=Lo(BUT26) BUS0=Hi
FINISH INSTRUCTION BUT FOR SVC
BUT16
(1Ø)
MICRO-PRGM ENTERS SERVICE
LEADING EDGE
K5-8
CLEAR PWRUP INIT F/F RESET F/F
TRAILING EDGE
K5-8
K5-4
(1Ø)
EXECUTE SERØ2 AND SERØ5 LEAVE SERVICE FOR TRAP A
BUT26
(6)
GET PWR FAIL VECT=24 FROM SBC* JUMPERS
(6)
PUSH PC*PSW ON STACK VECTOR TO EXECUTE POWER DOWN SERVICE ROUTINE
NOTE:
BUTØ4* P1 IN TRP2L CLEARS PWRDN F/F
PUSH
ACLO STILL ASSERTED?
NO
K5-8
PULSE POWER RESTART ONE SHOT 7Ø MS
ENTER POWER UP TRAP SEQUENCE

YELLOW ZONE

STACK OVERFLOW



YELLOW ZONE
* ADDRESS < 400 BUT > 336

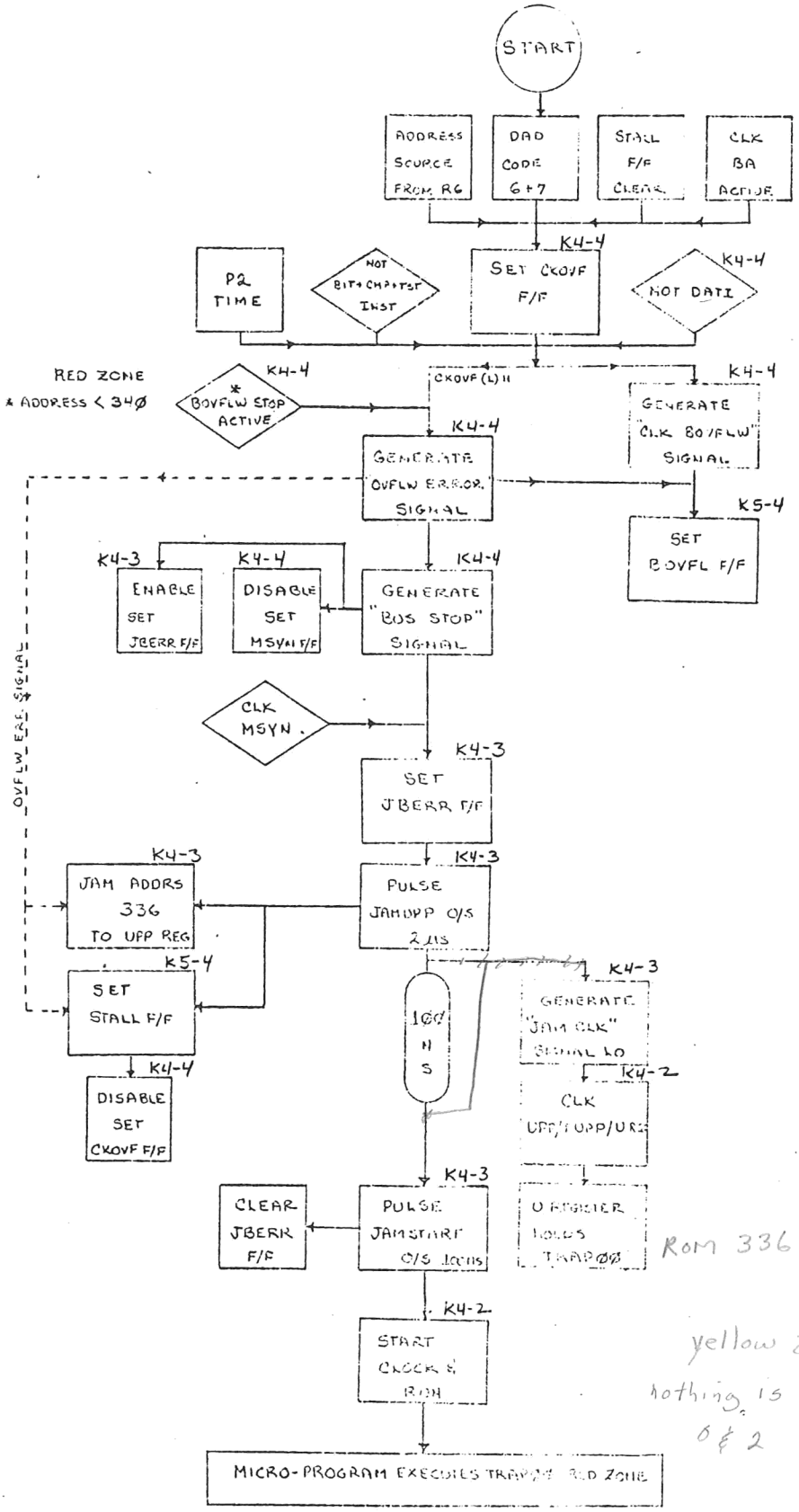
CK OFV in Kernal mode

* NOTE

TWO ADDITIONAL PUSHES ON THE STACK ARE DONE IN THE YELLOW ZONE TRAP ROUTINE AND ARE SENSED BY CKOVF F/F BUT DO NOT MATTER UNLESS A RED ZONE VIOLATION OCCURS DURING YELLOW ZONE PUSH

RED ZONE

STACK OVERFLOW



ROM 336

yellow zone means nothing is in location 0 & 2

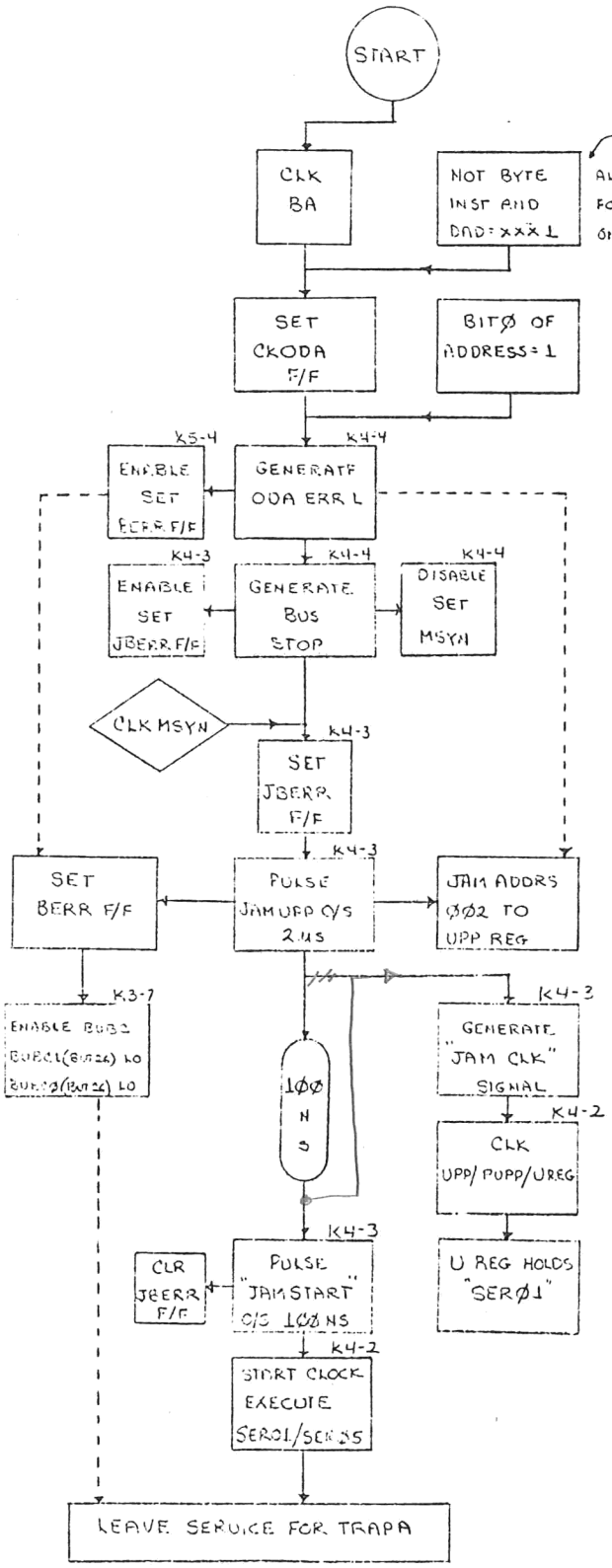
NOTE:

CLKBAIN TRP02 CLEARS CKOVF F/F (STOPS CLOCK FROM GOING NO STALL F/F SET)

BUT Ø1 * PL IN TRP 2Ø CLEARS BOVFL F/F

BUTØ4 * P1 IN TRP 21 CLEARS STALL F/F (ENABLES SENSING OVERFLOW)

ODD ADDRESS ERROR



NOTE

BERR F/F IS CLEARED BY BUTØ3· PL IN TRP16
IF ANOTHER ODA OCCURS UP TO THIS POINT
YOU HAVE A DOUBLE BUS ERROR (DUBBER)

PDP11/40 Quiz
Trouble Isolation

ANALYZE THE PROBLEM SYMPTOMS GIVEN BELOW AND IN EACH CASE, GIVE AT LEAST ONE IC (MODULE NUMBER, CHIP NUMBER AND PIN NUMBER) THAT COULD CAUSE THE PROBLEM.

1. THE CLOCK IS DISABLED FROM THE MAINTENENCE BOARD (KM11). YOU POWER UP THE PROCESSOR WITH THE HALT/ENABLE SWITCH IN THE HALT POSITION. UPP IS JAMMED TO ROM LOCATION 337 INSTEAD OF 30.

2. YOU ARE USING MAINT. MODULE TO SINGLE CLOCK THE CONSOLE FUNCTIONS. NO MATTER WHICH SWITCH YOU DEPRESS, YOU DISCOVER THAT ROM ALWAYS STAYS IN THE CONSOLE LOOP (26→46→26→46) ROM WILL NOT GO TO LOCATION 27 WHEN YOU DEPRESS A CONSOLE CONTROL SWITCH.

3. YOU ARE SINGLE CLOCKING THE PROCESSOR TO CHECK OUT THE LOAD ADDRESS SEQUENCE. THE SEQUENCE IS NORMAL UNTIL PUPP GOES TO 051, HERE YOU NOTE THE CONSOLE ADDRESS LIGHTS INDICATE # 17.

should have 177570
get 000017

4. WHEN SINGLE CLOCKING THE PROCESSOR THRU A HALT INSTRUCTION, YOU HAVE NORMAL INDICATIONS UNTIL PUPP = ØØ5, HERE YOU NOTE THAT UPP GOES TO 114 INSTEAD OF 122, WHICH IS THE REQUIRED SEQUENCE FOR A HALT.

FOR QUESTIONS 5 - 9 ASSUME THAT YOU ARE SINGLE CLOCKING THE SUBTRACT INSTRUCTION SHOWN BELOW:

```
5ØØ- SUB #2Ø, @ (3)+
5Ø2- 2Ø
5Ø4- HALT
```

R3 = 3ØØØ
3ØØØ = 5ØØØ
5ØØØ = 4Ø

40
-20

5 IN THE VERY FIRST STEP OF FETCH [FETØ2, LOC 16] WE NOTE THAT THE CONSOLE ADDRESS LIGHTS INDICATE A 3ØØØ INSTEAD OF 5ØØ.

6. NORMAL INDICATION ON THE FIRST THREE ROM STATES BUT WHEN WE CLOCK IN FETØ5, WE NOTE THE CONSOLE DATA LIGHTS INDICATE A 5Ø3

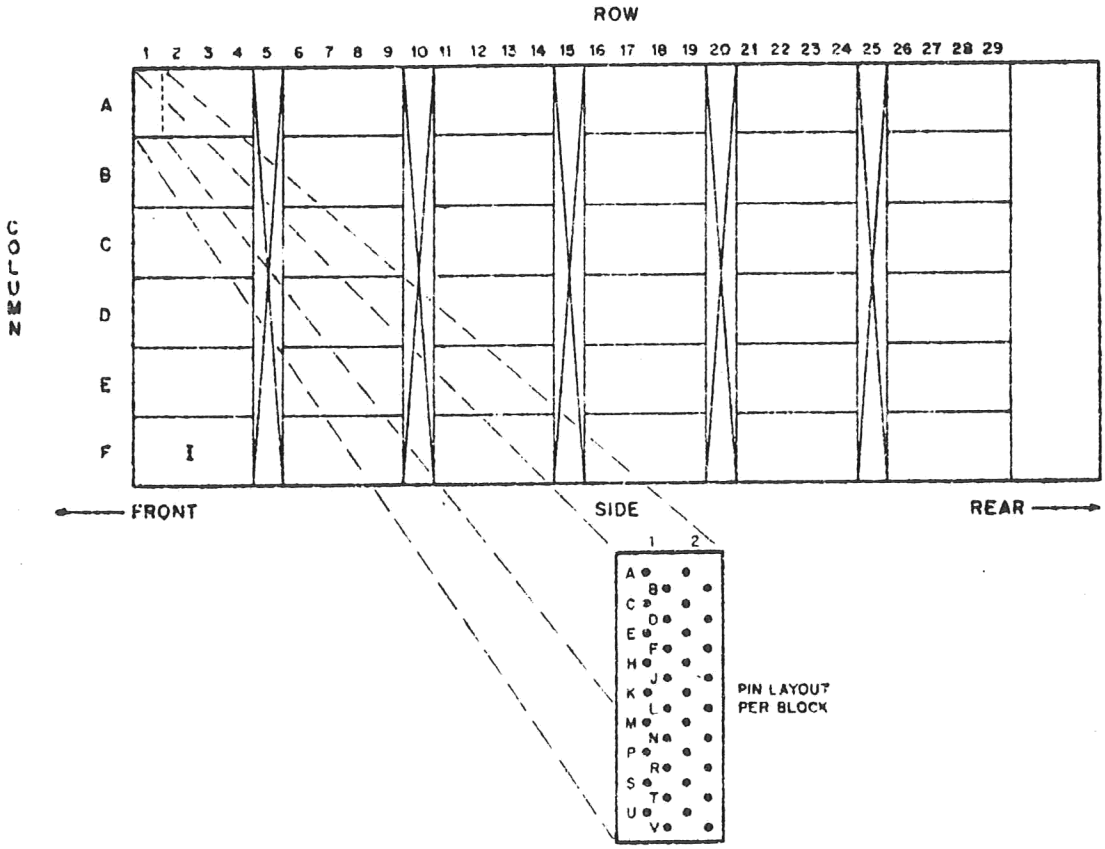
7. NORMAL INDICATIONS UNTIL WE CLOCK IN SRC15 LOC 25Ø. HERE WE NOTE THAT UPP GOES TO 137 INSTEAD OF 162.

8. NORMAL INDICATIONS UNTIL WE CLOCK IN DOP12 LOCATION 367. HERE WE NOTE THE CONSOLE DATA LIGHTS INDICATE A 17

9. IN THE LAST MKRO-WORD DOP20, LOC 375 WE NOTE THAT THE "C" BIT INDICATOR ON THE MAINT MODULE IS LIGHTED.

1. JAMUPP PROBLEM
2. BUT PROBLEM
3. SBC PROBLEM
4. IR DECODE PROBLEM
5. GPR ADDRESS SWITCH
6. INSERTED CARRY ON PC UPDATE
7. GETTING FALSE ODD BYTE SIGNAL
8. GETTING ONE'S COMPLEMENT INSTEAD OF TWO'S COMP
9. PSW C' SHOULD SET IF OVFLW ON SUBTRACT

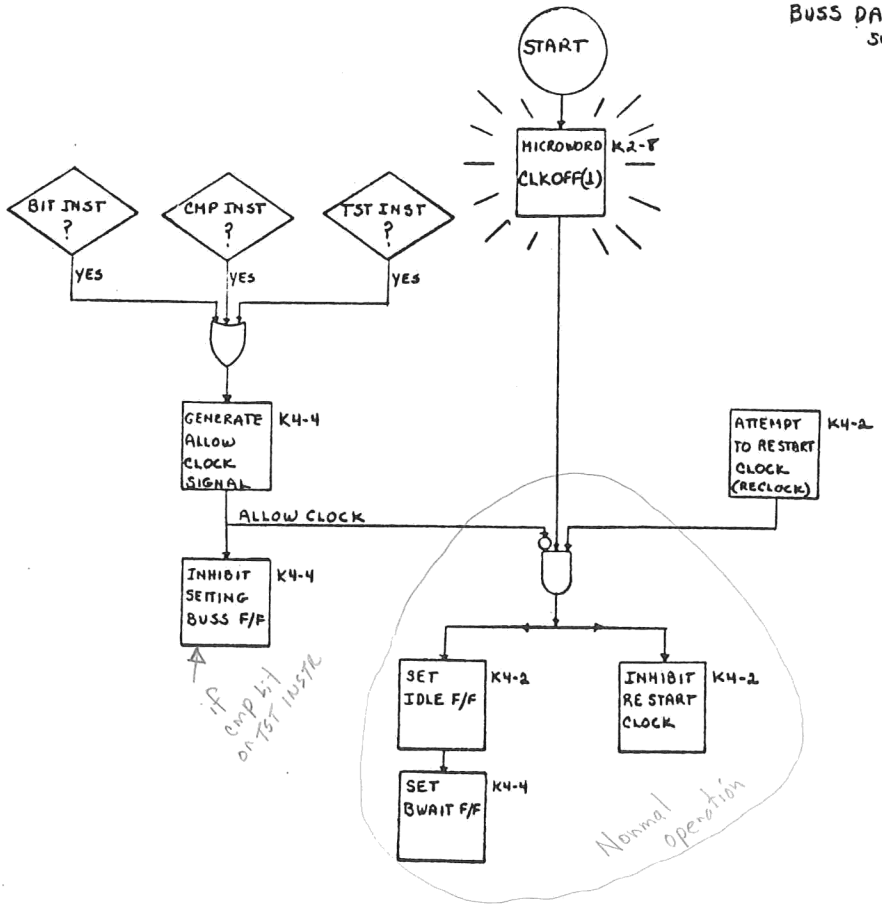
PDP-11 MODULE BLOCK AND PIN LAYOUT



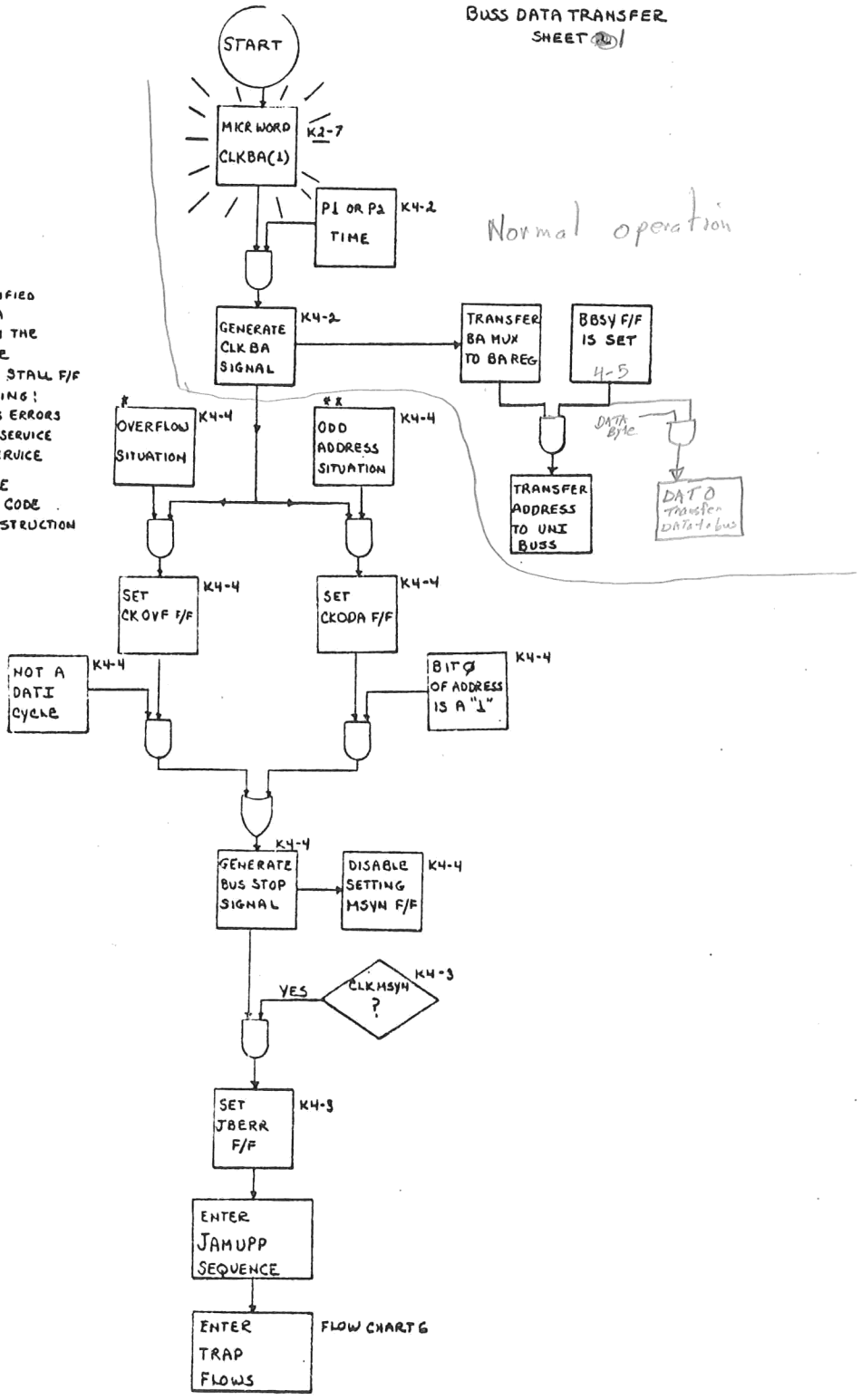
PIN SIDE VIEW

DAY 5

BUSS DATA TRANSFER
SHEET 3



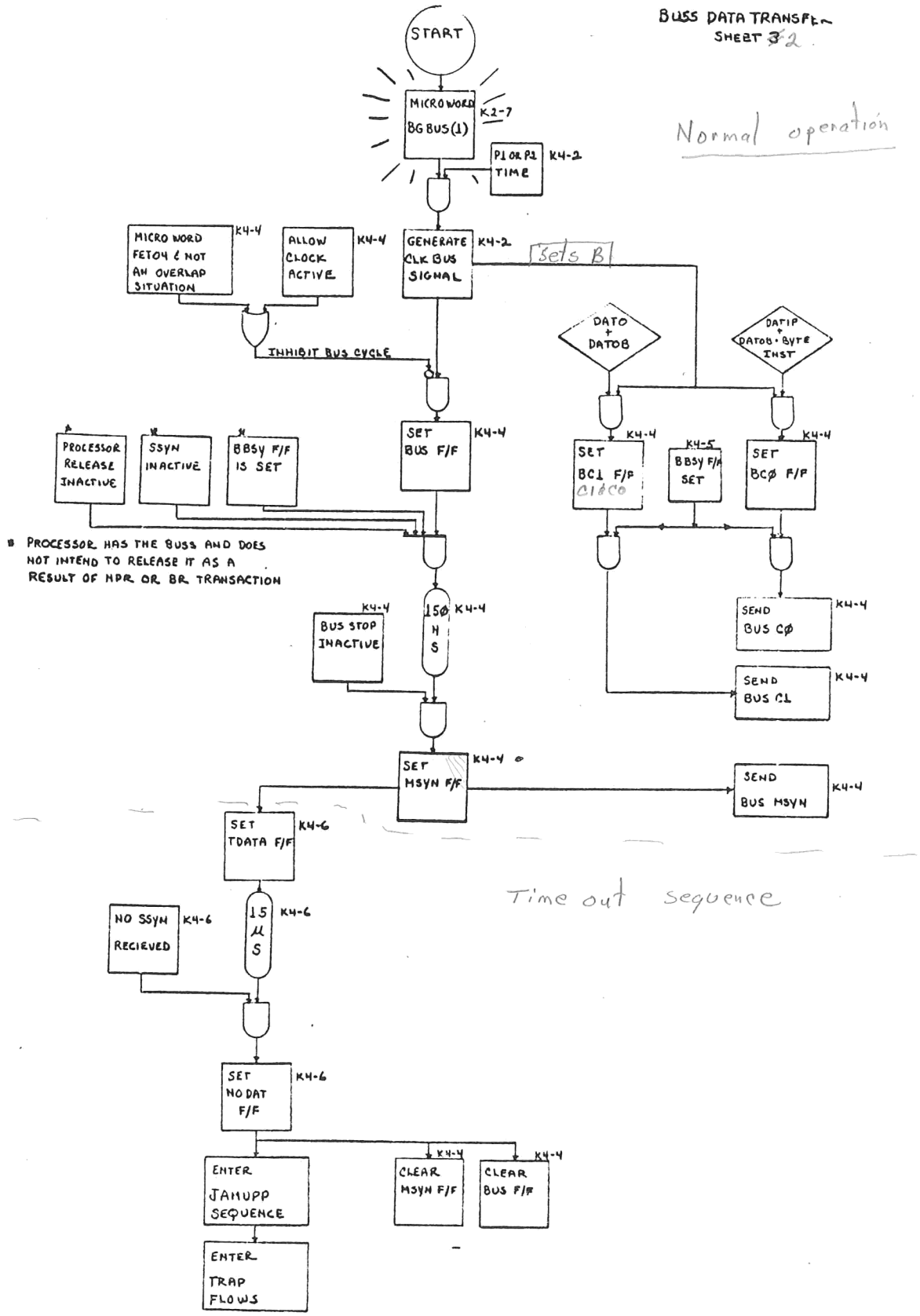
BUSS DATA TRANSFER
SHEET 1



* OVRFLW SITUATIONS ARE SPECIFIED BY U WORD DAD CODE AND A REGISTER ADDRESS OF R6 ON THE UNI-BUS. OVERFLW CAN BE INHIBITED BY SETING THE STALL F/F DURING ONE OF THE FOLLOWING:
  1. DOUBLE BUSS ERRORS
  2. POWER DOWN SERVICE
  3. OVERFLOW SERVICE

** ODD ADDRESS SITUATIONS ARE SPECIFIED BY U WORD DAD CODE AND IR DECODE = BYTE INSTRUCTION

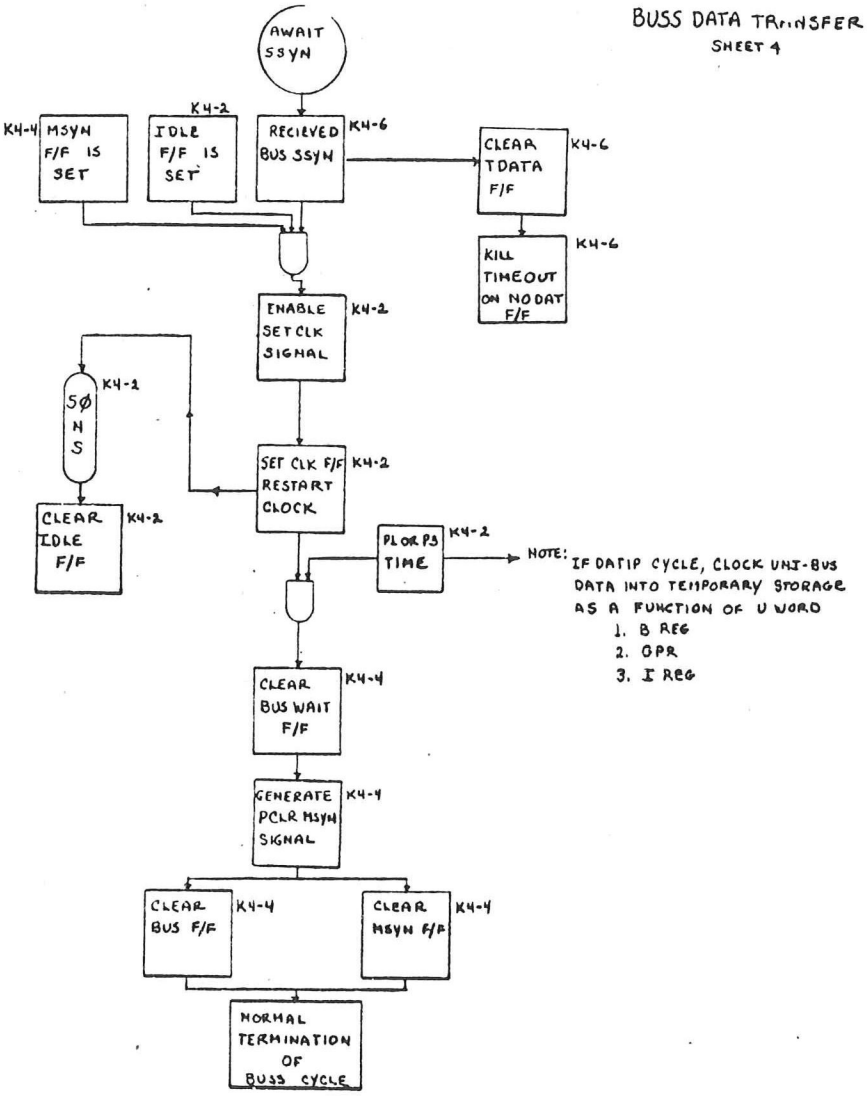
BUSS DATA TRANSFER
SHEET 3/2

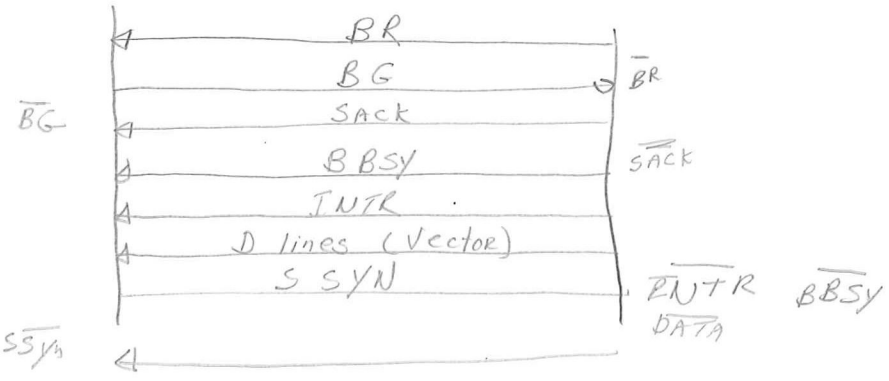


Normal operation

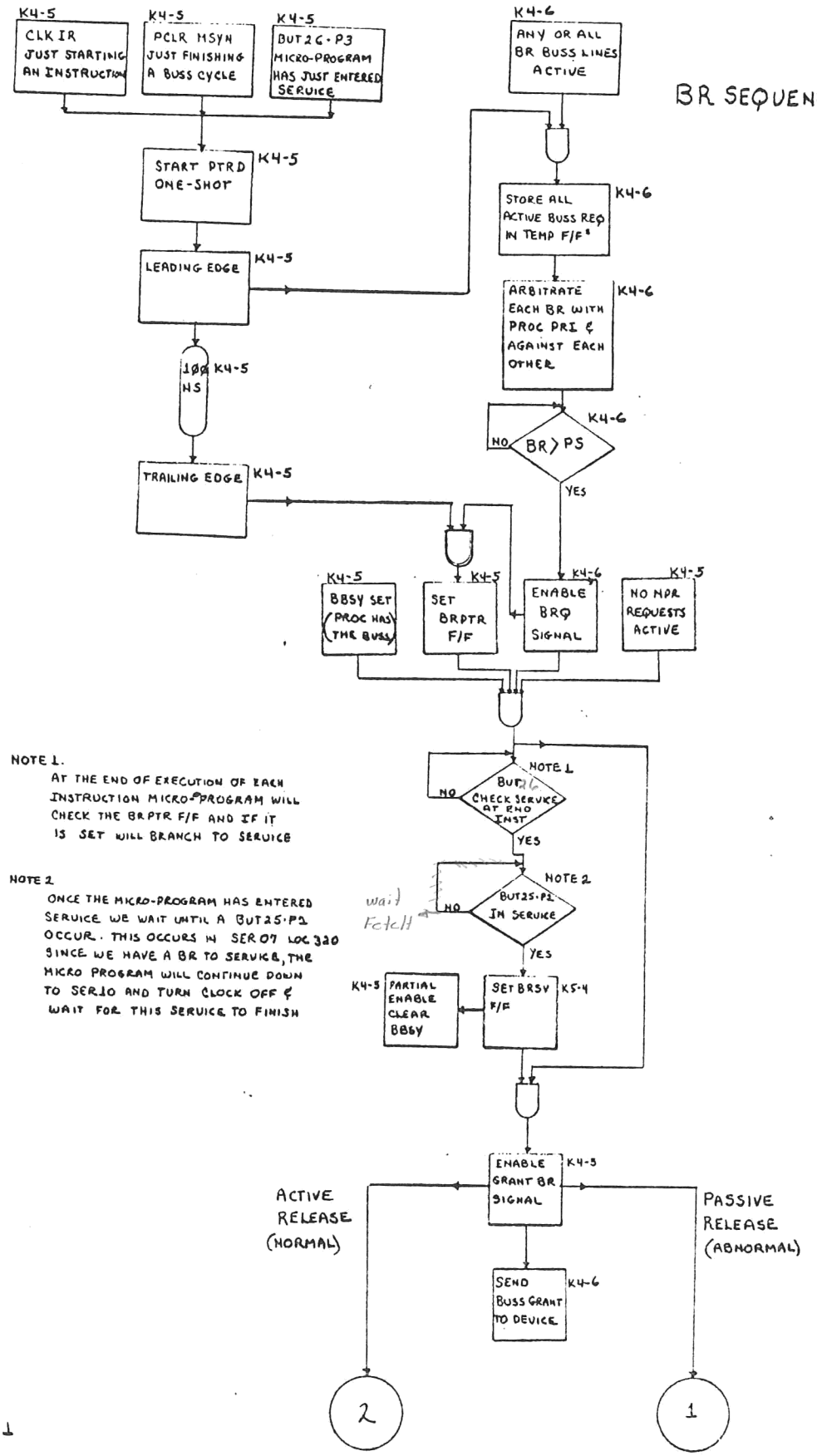
Time out sequence

BUSS DATA TRANSFER
SHEET 4





BR SEQUENCE



NOTE 1.

AT THE END OF EXECUTION OF EACH INSTRUCTION MICRO-PROGRAM WILL CHECK THE BRPTR F/F AND IF IT IS SET WILL BRANCH TO SERVICE

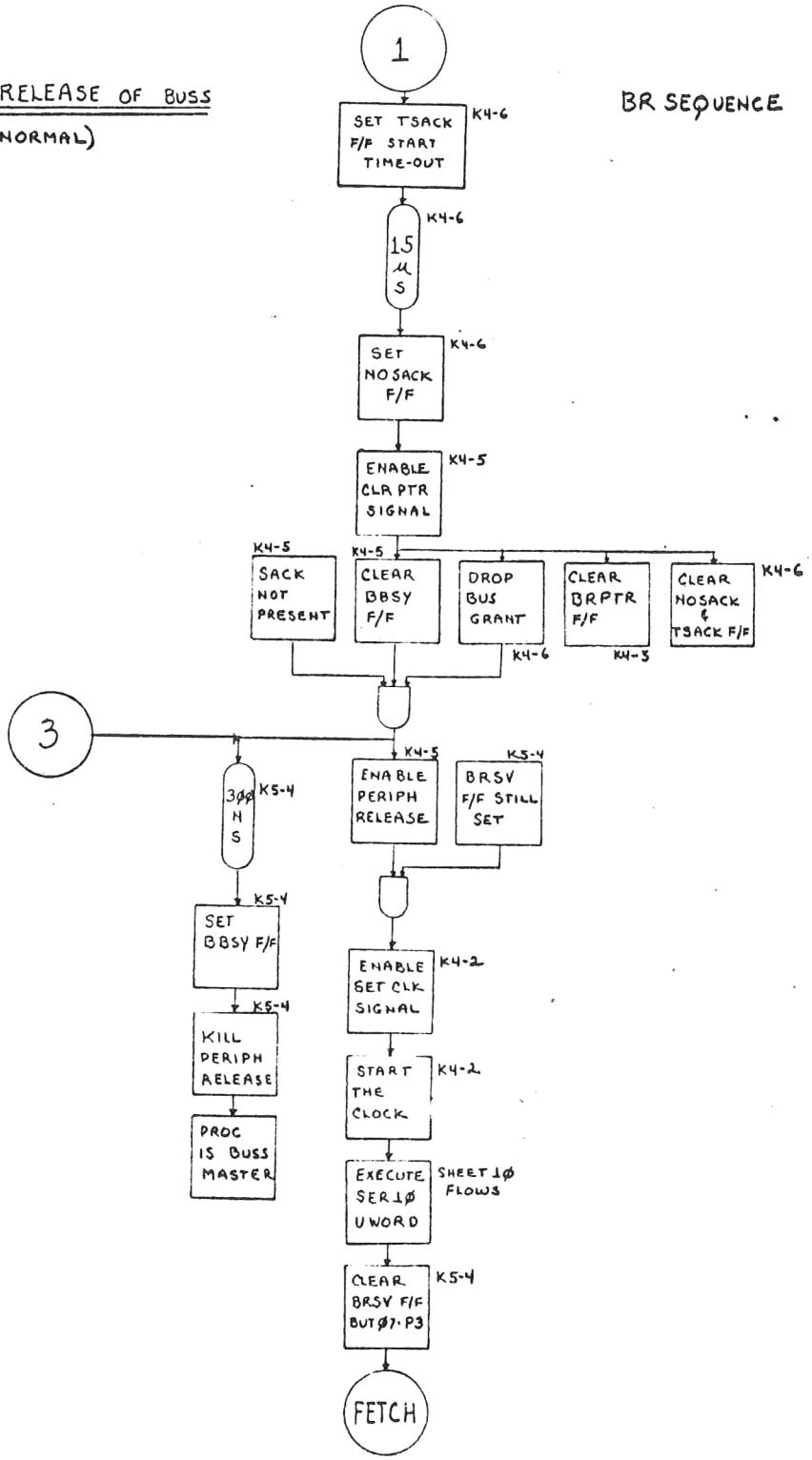
NOTE 2

ONCE THE MICRO-PROGRAM HAS ENTERED SERVICE WE WAIT UNTIL A BUT25·P2 OCCUR. THIS OCCURS IN SER 07 LOC 320 SINCE WE HAVE A BR TO SERVICE, THE MICRO PROGRAM WILL CONTINUE DOWN TO SER10 AND TURN CLOCK OFF & WAIT FOR THIS SERVICE TO FINISH

PASSIVE RELEASE OF BUSS

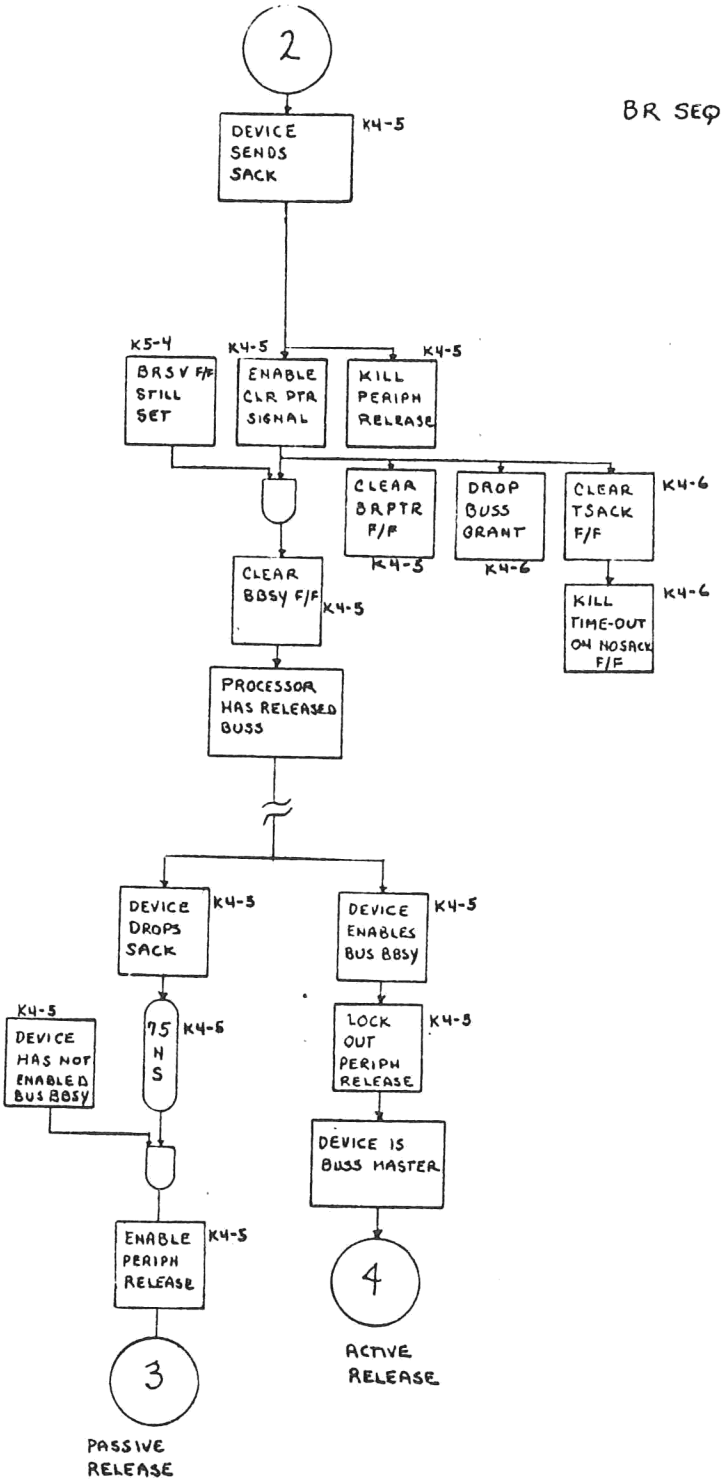
(ABNORMAL)

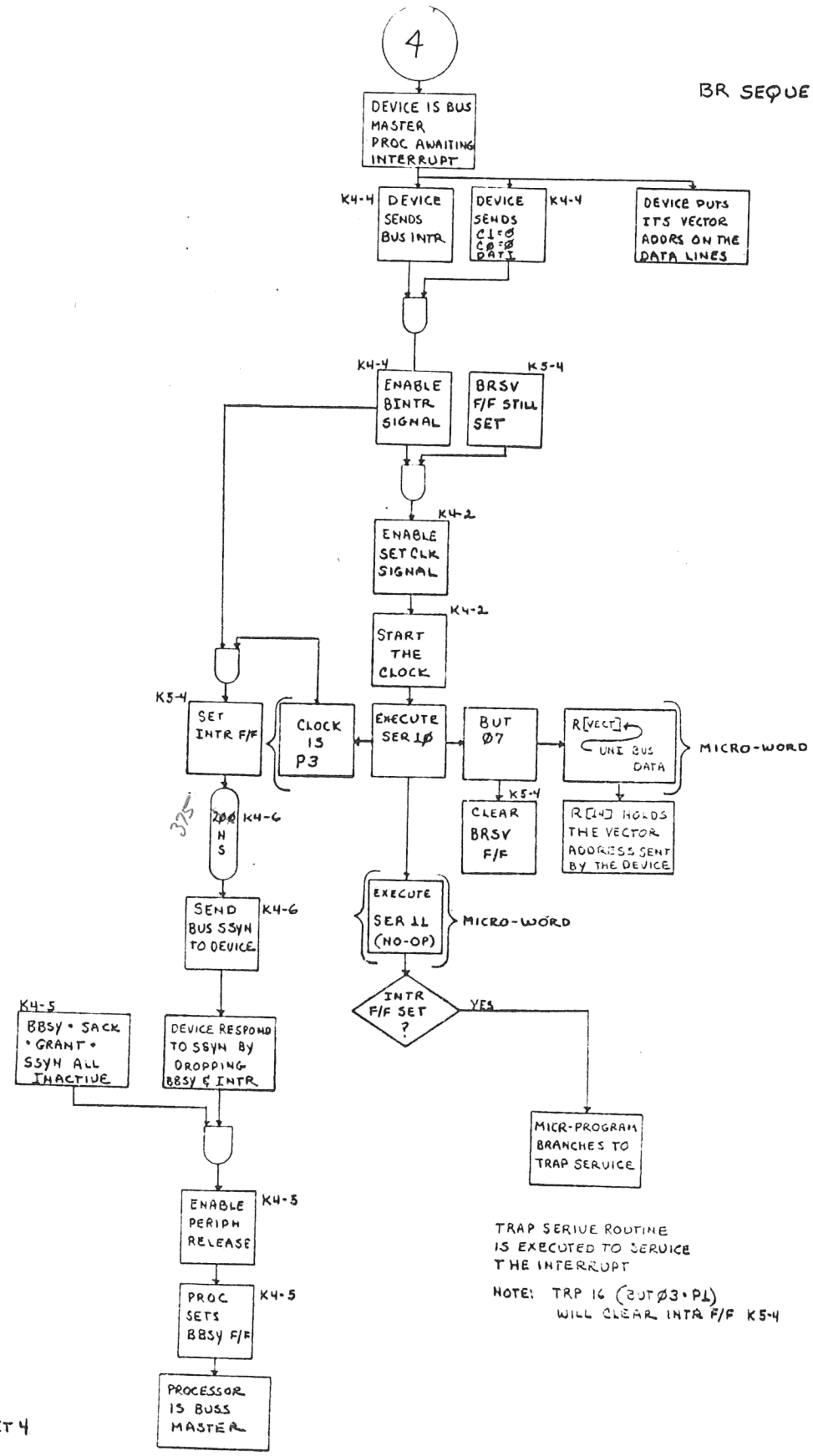
BR SEQUENCE



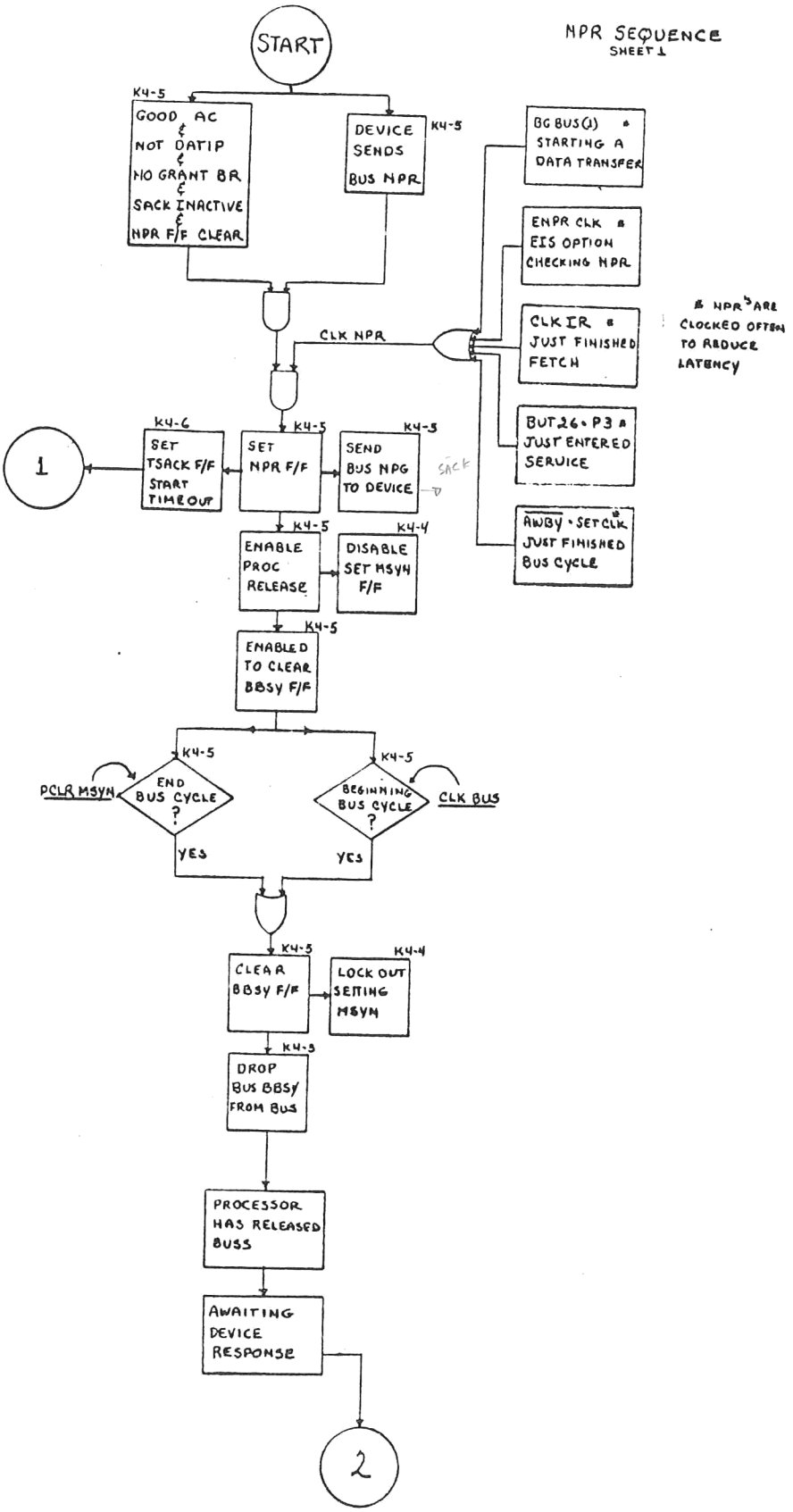
SHEET2

BR SEQUENCE

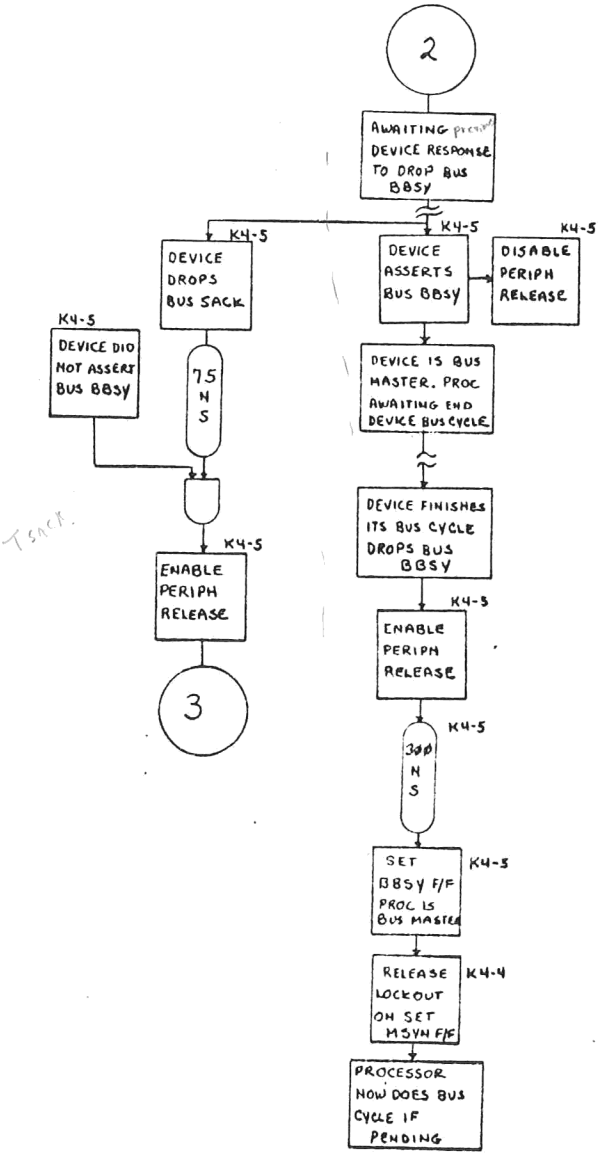

2
DEVICE SENDS SACK
K4-5
K5-4
BRSV F/F STILL SET
K4-5
ENABLE CLR PTR SIGNAL
K4-5
KILL PERIPH RELEASE
CLEAR BRPTR F/F
K4-5
DROP BUSS GRANT
K4-6
CLEAR TSACK F/F
K4-6
KILL TIME-OUT ON NOSACK F/F
K4-6
CLEAR BBSY F/F
K4-5
PROCESSOR HAS RELEASED BUSS
DEVICE DROPS SACK
K4-5
DEVICE ENABLES BUS BBSY
K4-5
K4-5
DEVICE HAS NOT ENABLED BUS BBSY
75 NS
K4-6
LOCK OUT PERIPH RELEASE
K4-5
DEVICE IS BUSS MASTER
ENABLE PERIPH RELEASE
K4-5
3
PASSIVE RELEASE
4
ACTIVE RELEASE


4
BR SEQUENCE
DEVICE IS BUS MASTER PROC AWAITING INTERRUPT
K4-4
DEVICE SENDS BUS INTR
DEVICE SENDS C1=0 C0=0 DATI
K4-4
DEVICE PUTS ITS VECTOR ADDRS ON THE DATA LINES
K4-4
ENABLE BINTR SIGNAL
K5-4
BRSV F/F STILL SET
K4-2
ENABLE SETCLK SIGNAL
K4-2
START THE CLOCK
K5-4
SET INTR F/F
CLOCK IS P3
EXECUTE SER 10
BUT 07
R[VECT] UNI BUS DATA
MICRO-WORD
K5-4
CLEAR BRSV F/F
R[VECT] HOLDS THE VECTOR ADDRESS SENT BY THE DEVICE
200 NS
K4-6
SEND BUS SSYN TO DEVICE
K4-6
EXECUTE SER 11 (NO-OP)
MICRO-WORD
INTR F/F SET ?
YES
K4-5
BBSY • SACK • GRANT • SSYN ALL INACTIVE
DEVICE RESPOND TO SSYN BY DROPPING BBSY & INTR
MICR-PROGRAM BRANCHES TO TRAP SERVICE
ENABLE PERIPH RELEASE
K4-5
PROC SETS BBSY F/F
K4-5
PROCESSOR IS BUSS MASTER
TRAP SERIVE ROUTINE IS EXECUTED TO SERVICE THE INTERRUPT
NOTE: TRP 16 (BUT 03 • P1) WILL CLEAR INTR F/F K5-4
SHEET 4

NPR SEQUENCE
SHEET 1

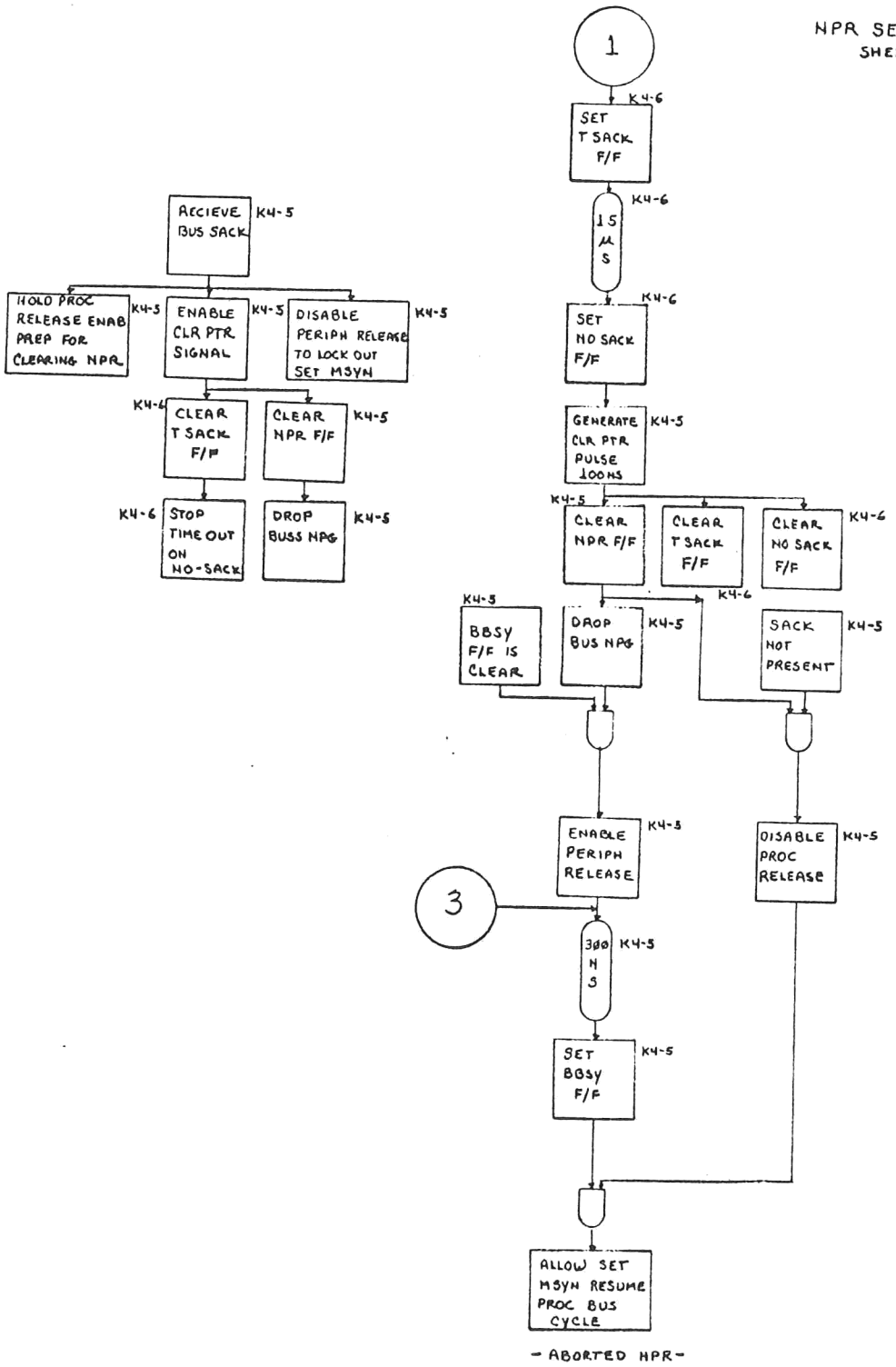


NPR SEQUENCE
SHEET 2



NORMAL TERMINATION

NPR SEQUENCE
SHEET 3



PDP 11/40

WEEK 2

EXAM

1. TO DISPLAY THE UNI-BUS DATA LINES ON THE CONSOLE DATA DISPLAY, BITS U<24:23> OF THE UWORD MUST PROGRAMMED AS:

A. ØØ
B. ØL
C. LØ
D. LL

2. THE MICRO-INSTRUCTION SHOWN BELOW IS BEING EXECUTED. WHAT IS THE STATE OF UREG <16:Ø9>?

SRX (3) | RIF (5)

A. ØØLL ØLØL
B. LØØL LØØL
C. ØØØL LØØL
D. ØØLØ LØLØ

P1: B, R[SOURCE] ← D

3. DURING THE MICRO-INSTRUCTION THAT ENABLES DELIVERING THE RESULTS OF THE INSTRUCTION SHOWN BELOW, WHAT WOULD BE THE STATE OF WRH(1) AND WRL(J) (REFER TO K2-7)

A. H, H
B. H, L
C. L, H
D. L, L

```
600  MOVB (1),%2

R1 = 503
```

4. WHICH OF THE FOLLOWING ELEMENTARY OPERATIONS DICTATES THAT A CLOCK LENGTH OF THREE MUST BE SPECIFIED?

A. D ← R[SP] MINUS 2

B. R[PC] ← D

C. R[SF] ← UNIBUS DATA

D. R[PC] ← R[PC] PLUS 2

5. WHICH OPERATION SHOWN BELOW REQUIRES TWO MICRO-INSTRUCTIONS TO EXECUTE?

A. R[PC] ← R[PC] PLUS 2

B. D ← R[DEST] PLUS UNIBUS DATA

C. B ← R[SOURCE]

D. BA ← R[SF] PLUS B

6. REFER TO K3-2 PRINT. THE MICRO-INSTRUCTION BEING EXECUTED IS ENCODED TO BUT JMP+JSR. WHICH BUT MULTIPLEXOR CHIP IS SELECTED

A. E82

B. E72

C. E81

D. E97

9. THE MICRO-INSTRUCTION CURRENTLY BEING EXECUTED SPECIFIES A CLOCK LENGTH OF THREE. THE NEXT MICRO-INSTRUCTION WILL BE CLOCKED INTO THE UREG AT THE:

A. LEADING EDGE OF P2

B. TRAILING EDGE OF P3

C. LEADING EDGE OF P3

D. TRAILING EDGE OF P2

BEFORE ANSWERING QUESTIONS 8 THRU 10 COMPLETE A MICRO-PROGRAM WORKSHEET FOR THE INSTRUCTION SHOWN BELOW. START AT FET Ø2 AND INCLUDE ALL MICRO-INSTRUCTIONS FROM START TO FINISH.

→ 5ØØ - CMP 7Ø2, (3) R2 = 35Ø
5Ø2 - HALT R3 = 5ØØ

10. IF YOU MANUALLY CLOCK THE MICRO-PROGRAM UNTIL PUPP = 267, YOU SHOULD OBSERVE _____ IN THE BUPP LIGHTS OF THE MAINTENANCE MODULE?

A. 266

B. 22Ø

C. 225

D. 224

9. IF YOU MANUALLY CLOCK UNTIL PUPP = 367 YOU SHOULD OBSERVE _____ IN THE DATA LIGHTS ON THE CONSOLE?

A. Ø2Ø213
B. Ø6Ø134
C. 16Ø Ø35
D. 16Ø 135

10. WHAT SHOULD BE IN THE CONSOLE ADDRESS DISPLAY WHEN PUPP = 161

A. 476
B. 5ØØ
C. 5Ø2
D. 5Ø4

11 YOU ARE MANUALLY CLOCKING THE MICRO-PROGRAM TO VERIFY THE INSTRUCTION SHOWN BELOW. THE NEXT MICRO-WORD EXECUTED AFTER SSLØ3 SHOULD BE

A. DOP16
B. DOP17
C. DOP18
D. DOP19

INCB %2
R2 = 1ØØ

12. DURING EXECUTION OF A ROL2 INSTRUCTION, WHICH DATA FACILITY DOES THE ACTUAL SHIFTING?

A. B MUX

B. D MUX

C B REG

D ALU

13. THE INSTRUCTION SHOWN BELOW IS BEING SINGLE CLOCKED. THE BUT INST L IN FETØ4 SHOULD ACTIVATE MICRO-BRANCH CONTROL SIGNALS

A. BUBCØ AND BUBC1

B. BUBC1 AND BUBC2

C. BUBC3 AND BUBC4

D. BUBC2 AND BUBC4

RTS 3

14 REFER TO SHEET 3 OF FLOWS. DURING EXECUTION OF THE MICRO-INSTRUCTION TAGGED DST16, WHAT SHOULD BE THE STATE OF SBML1 AND SBMLØ ON K2-5

A. L, L

B. H, L

C. L, H

D. H, H

15. CONSIDER THE PRINCIPLE OF MICRO-BRANCHING USED IN THE KD-11A. IF A MICRO-WORD HAS A BASE ADDRESS OF 370, HOW MANY ADDRESS PATHS ARE OPEN TO US IN ADDRESSING THE NEXT MICRO-WORD. ASSUME BUT37

A. 2
B. 3
C. 6
D. 8

16. WHEN SINGLE CLOCKING THE MARK INSTRUCTION, YOU STOP THE SEQUENCE WITH PUPP=354. IF YOU SCOPED SOML(1) AND SOMØ(1) ON K2-5 YOU WOULD EXPECT TO OBSERVE

A. HIGH, HIGH
B. LOW, LOW
C. HIGH, LOW
D. LOW, HIGH

17. YOU ARE CHECKING OUT THE DEPOSIT OPERATION BY DEPOSITING ALL ONES INTO LOCATION 500. YOU MANUALLY CLOCK UNTIL PUPP=67. WHAT SHOULD YOU OBSERVE IN THE CONSOLE ADDRESS DISPLAY?

A. 177777

B. 5ØØ

C. 1775ØØ

D. 17757Ø

18. REFER TO KL-7 PRINT: WHICH OF THE FOLLOWING BUS ADDRESSES WILL THE SIGNAL BOVFL (ENABLE) STOP?

A. 4ØØ

B. 37Ø

C. 35Ø

D. 33Ø

19. THE MICRO-PROGRAM DOES A JAMUPP TO LOC ØØ2 FOR ANY MICRO-INSTRUCTION THAT SPECIFIES A DATA TRANSFER (DATI OR DATO) YOU ARE GETTING PROPER SSYN RESPONSE TO MSYN. TROUBLE HAS BEEN ISOLATED TO THE K4 MODULE. WHICH CHIP

A. E2Ø

B. E56

C. E38

D. E1Ø

20 WHILE CHECKING OUT THE CONSOLE, YOU NOTE THAT YOU CAN EXAMINE REGISTERS BUT CANNOT EXAMINE ANY LOCATION ON THE BUS. YOU SINGLE CLOCK THRU AN EXAMINE OF LOC 5ØØ AND FIND THAT THE MICRO-PROGRAM SHIPS AROUND EXM Ø6, EXM Ø7 AND EXMØ8 BACK INTO THE CONSOLE LOOP. THE PROBLEM IS MOST LIKELY CHIP

A. E82

B. E98

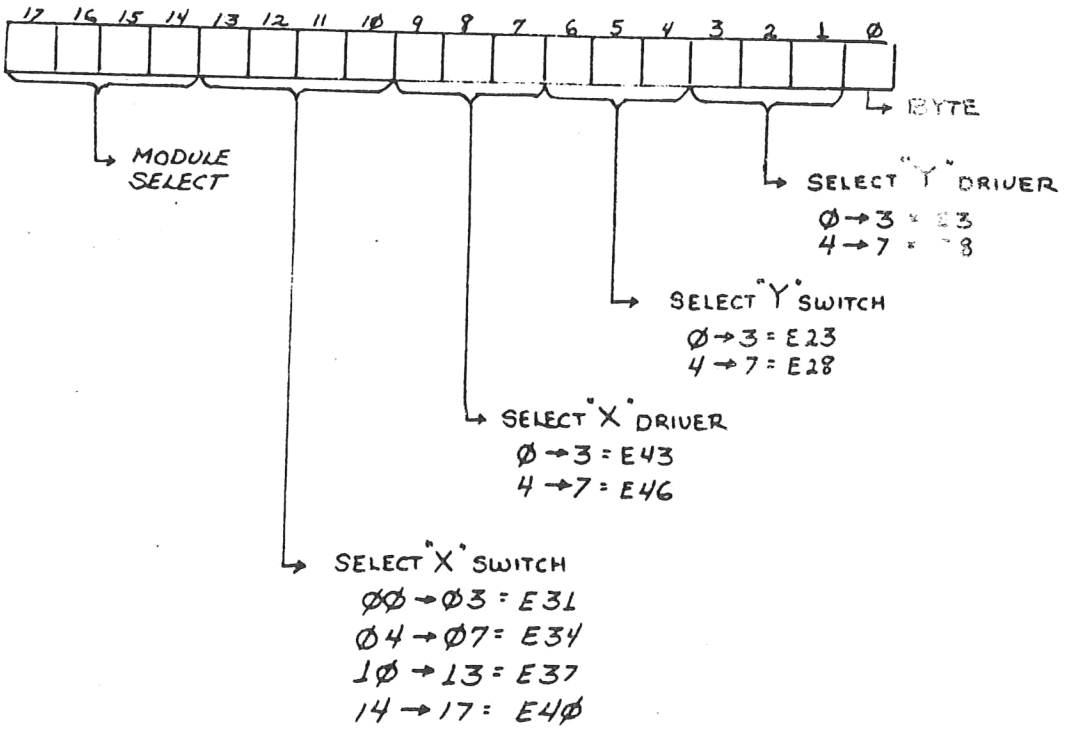
C. E8L

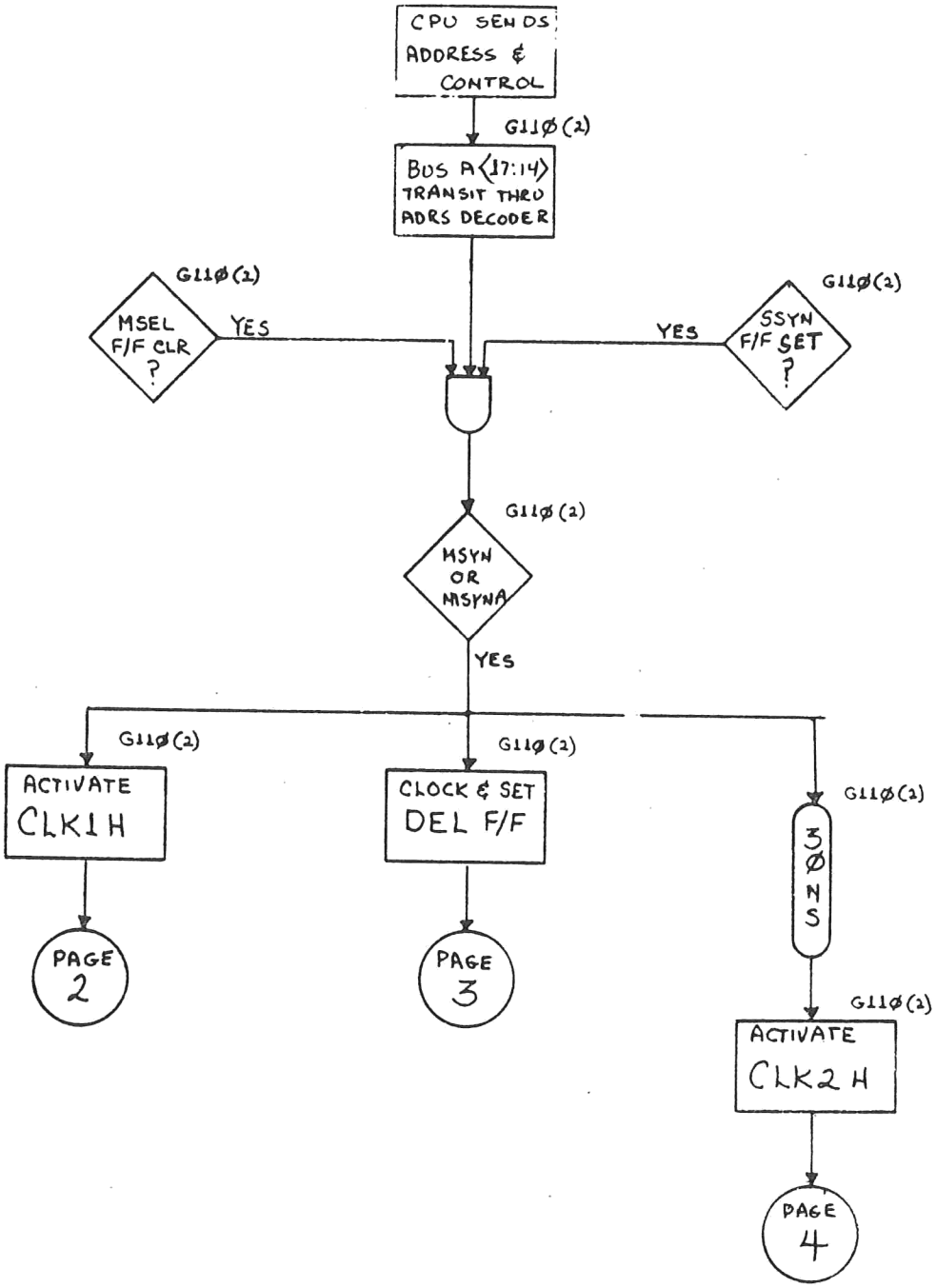
D. E73

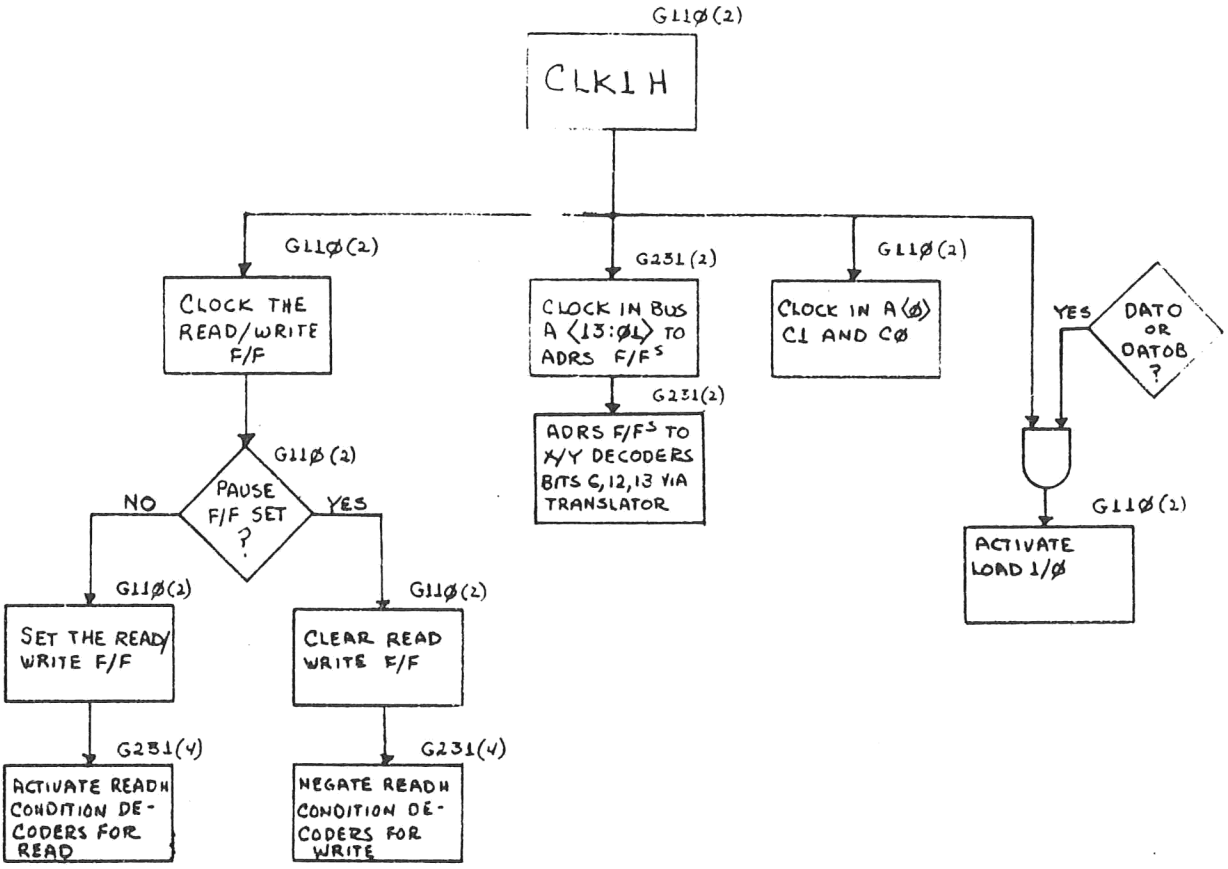
MM-11

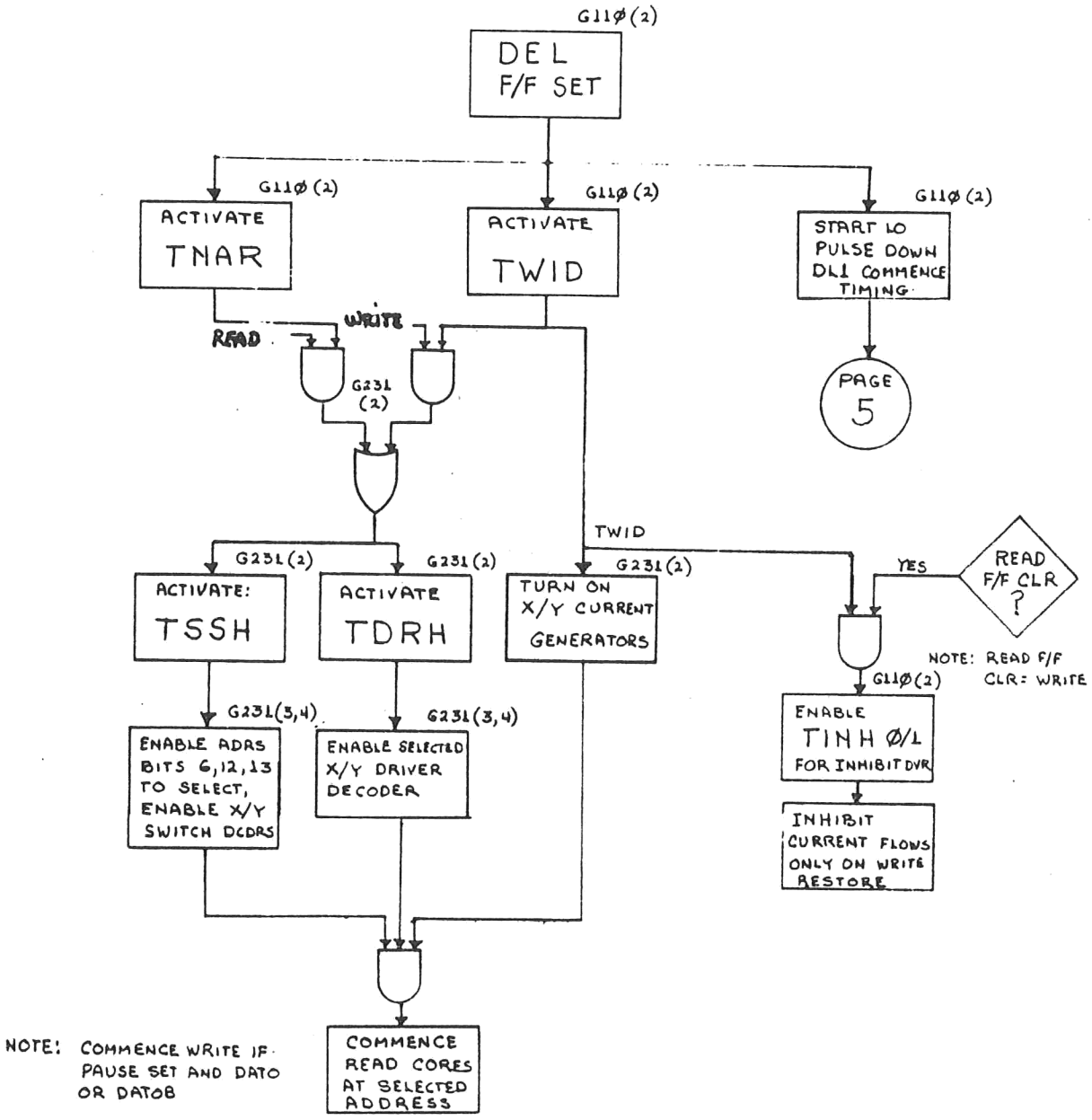
MEMORY

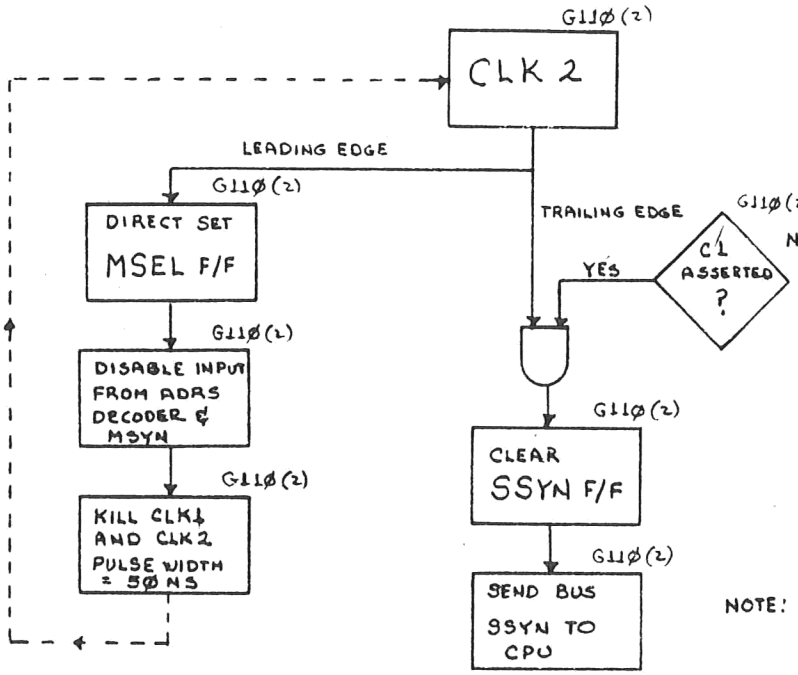
MEMORY ADDRESS FORMAT




CPU SENDS ADDRESS & CONTROL
G11Ø(2)
BUS A⟨17:14⟩ TRANSIT THRU ADRS DECODER
G11Ø(2)
MSEL F/F CLR ?
YES
YES
G11Ø(2)
SSYN F/F SET ?
G11Ø(2)
MSYN OR MSYNA
YES
G11Ø(2)
ACTIVATE CLK1H
G11Ø(2)
CLOCK & SET DEL F/F
G11Ø(2)
3ØNS
G11Ø(2)
ACTIVATE CLK2H
PAGE 2
PAGE 3
PAGE 4


G11Ø(2)
CLK1 H
G11Ø(2)
CLOCK THE READ/WRITE F/F
G11Ø(2)
PAUSE F/F SET ?
NO
YES
G11Ø(2)
SET THE READ/WRITE F/F
G231(4)
ACTIVATE READH CONDITION DE-CODERS FOR READ
G11Ø(2)
CLEAR READ WRITE F/F
G231(4)
NEGATE READH CONDITION DE-CODERS FOR WRITE
G231(2)
CLOCK IN BUS A ⟨13:ØL⟩ TO ADRS F/F'S
G231(2)
ADRS F/F'S TO X/Y DECODERS BITS 6,12,13 VIA TRANSLATOR
G11Ø(2)
CLOCK IN A⟨Ø⟩ C1 AND CØ
YES
DATO OR DATOB ?
G11Ø(2)
ACTIVATE LOAD 1/Ø


G11Ø(2)
DEL F/F SET
G11Ø(2)
ACTIVATE TNAR
G11Ø(2)
ACTIVATE TWID
G11Ø(2)
START LO PULSE DOWN DL1 COMMENCE TIMING
READ
WRITE
G231 (2)
PAGE 5
TWID
G231(2)
G231(2)
G231(2)
ACTIVATE: TSSH
ACTIVATE TDRH
TURN ON X/Y CURRENT GENERATORS
YES
READ F/F CLR ?
NOTE: READ F/F CLR = WRITE
G11Ø(2)
ENABLE TINH Ø/L FOR INHIBIT DVR
INHIBIT CURRENT FLOWS ONLY ON WRITE RESTORE
G231(3,4)
G231(3,4)
ENABLE ADRS BITS 6,12,13 TO SELECT, ENABLE X/Y SWITCH DCDRS
ENABLE SELECTED X/Y DRIVER DECODER
COMMENCE READ CORES AT SELECTED ADDRESS
NOTE: COMMENCE WRITE IF PAUSE SET AND DATO OR DATOB

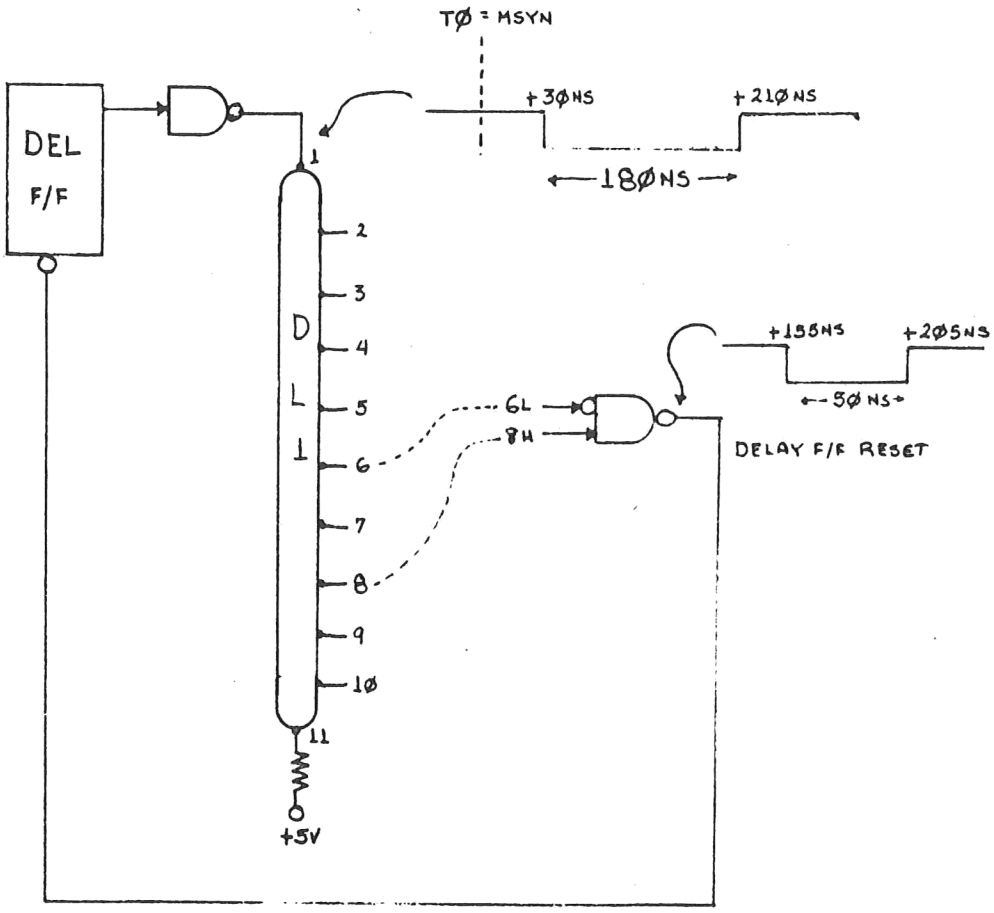


NOTE: C1 IS ASSERTED ON DATO OR DATOB

NOTE: IF THIS WERE A DATO OR DATOB, LOAD 1/Ø H ON PAGE 2 WOULD HAVE GATED DATA INTO DATA REG AND BUS SSYN WOULD BE PROPER RESPONSE. NOTE HOW EARLY IN MEM CYCLE SSYN IS SENT ON DATO/DATOB

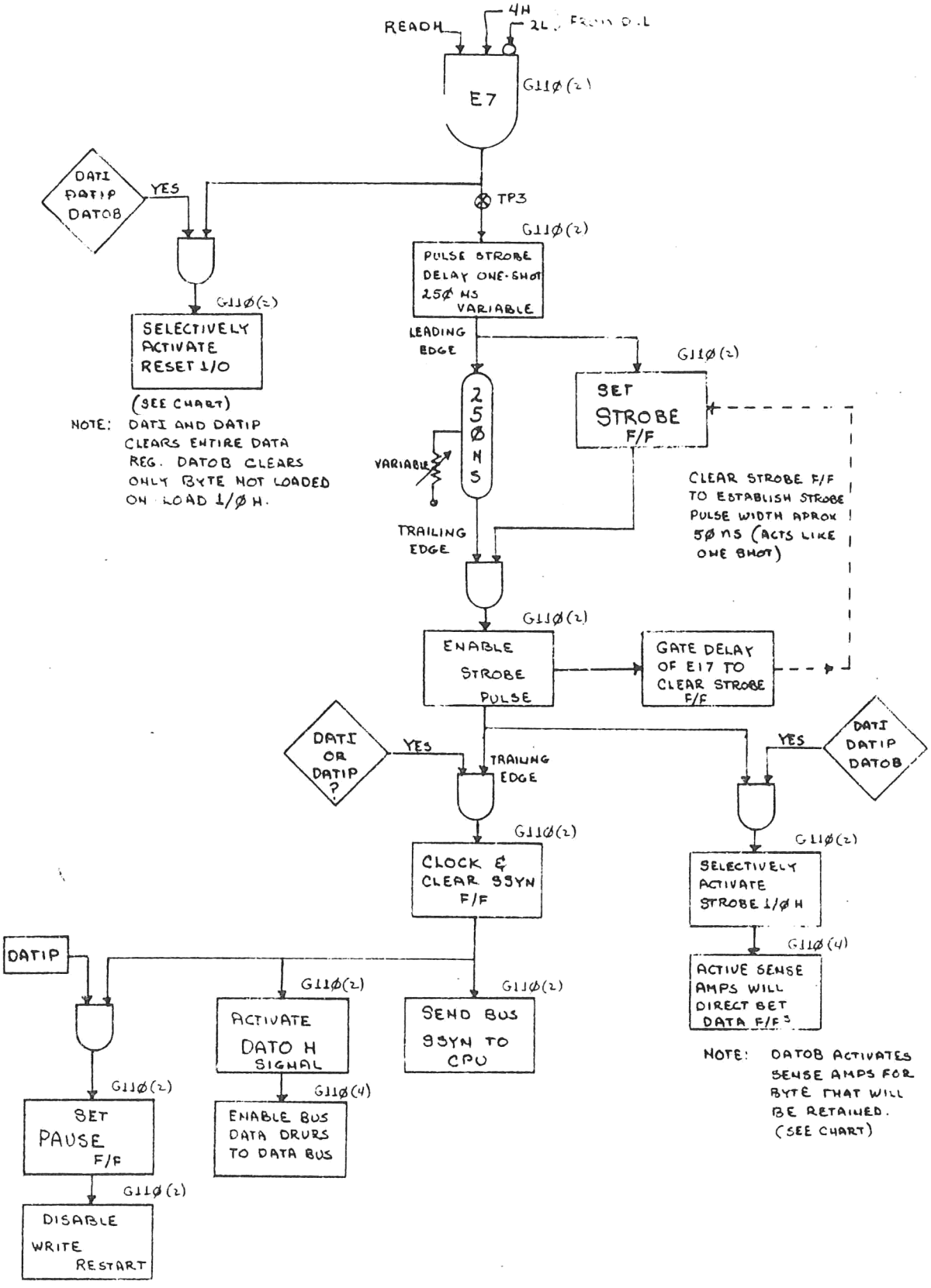
G110(c)

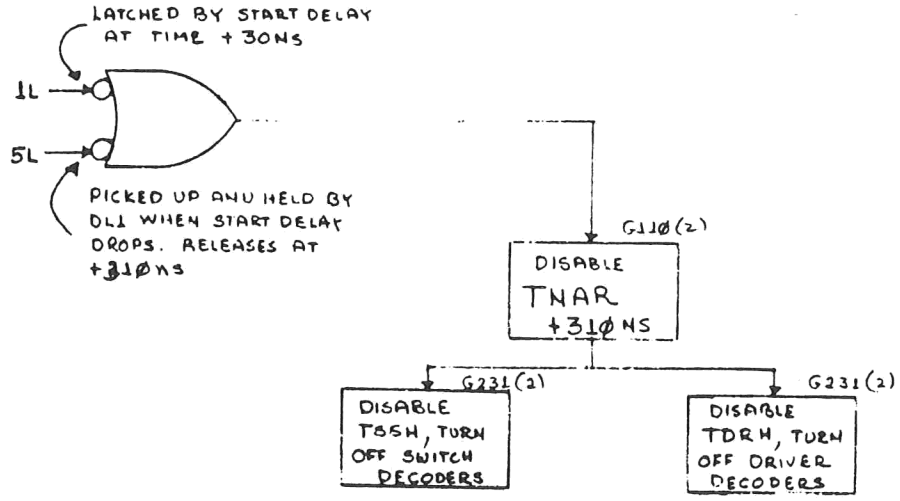
START LO PULSE DOWN DLL COMMENCE TIMING

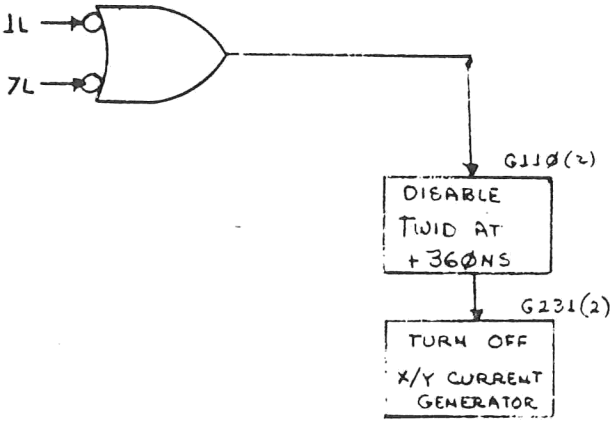


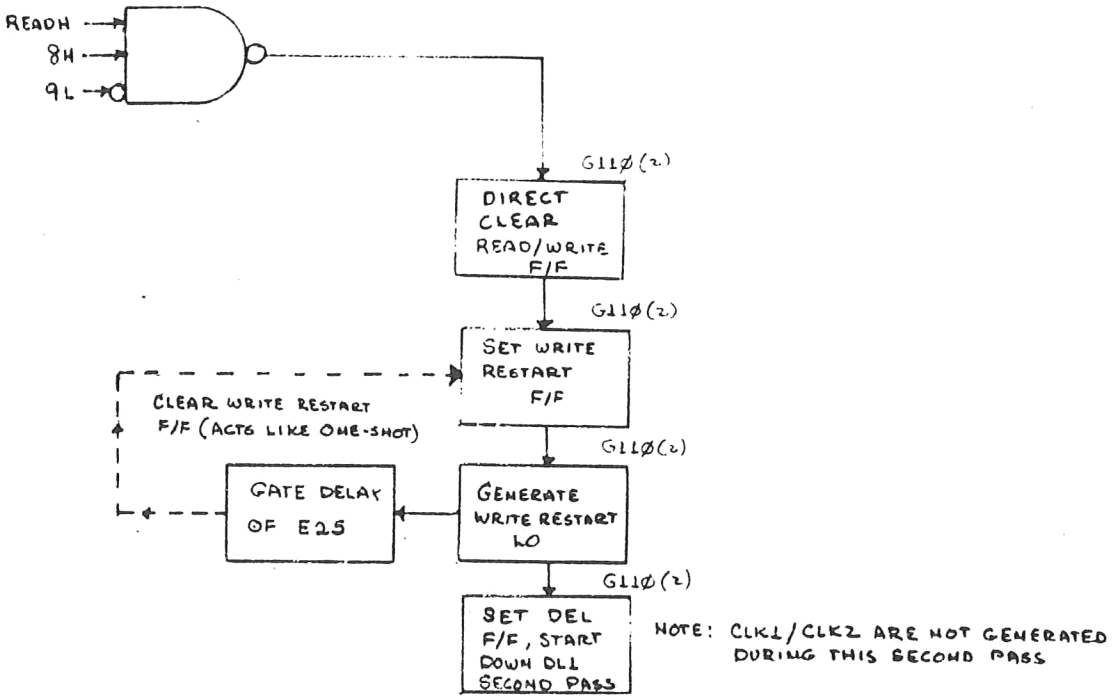
THE PURPOSE OF SHOWING THE ABOVE IS TO ESTABLISH THE RELATIONSHIP BETWEEN THE TIME MSYN (TØ) IS RECIEVED AND THE WAY THE WIDTH OF THE LO PULSE DOWN THE DELAY LINE IS DETERMINED. TO FIND THE APPROXIMATE ELAPSED TIME (REFRENCED TO MSYN) MULTIPLY 25NS X THE TAP NUMBER. THE APPARENT DESCREPENCIES IN TIME ARE CONSUMED IN GATE DELAYS AND RESPONSE TIME.

AS SHOWN, THE WIDTH OF THE LO PULSE COMING DOWN THE DELAY LINE IS ≈ 18ØNS. EACH TAP WILL SEE THIS 18ØNS LO PULSE 25 NS DISPLACED FROM THE PREVIOUS TAP

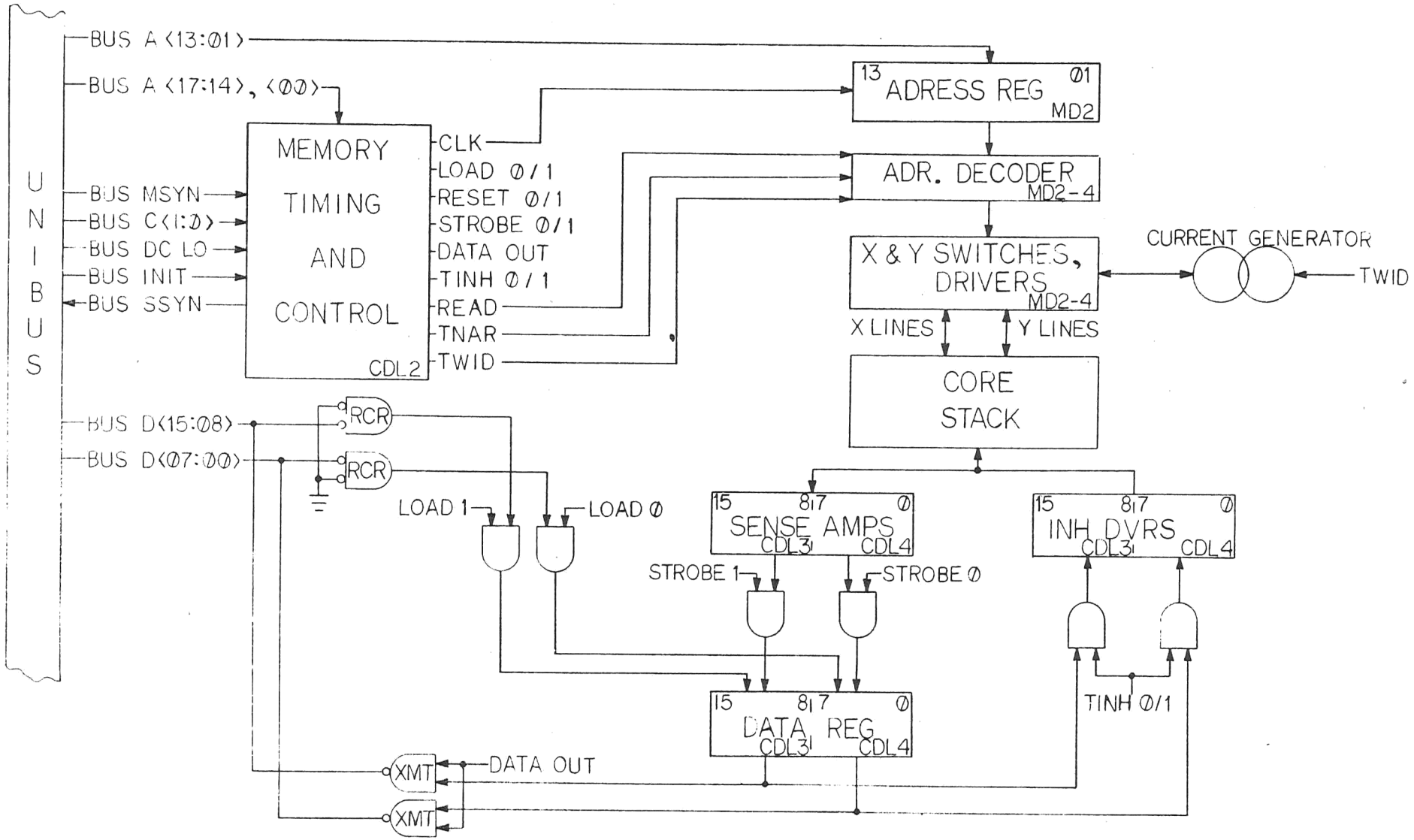

READH
4H
2L FROM D.L
E7
G11Ø(2)
DATI DATIP DATOB
YES
TP3
G11Ø(2)
PULSE STROBE DELAY ONE-SHOT 25Ø NS VARIABLE
G11Ø(2)
SELECTIVELY ACTIVATE RESET 1/O
(SEE CHART)
NOTE: DATI AND DATIP CLEARS ENTIRE DATA REG. DATOB CLEARS ONLY BYTE NOT LOADED ON LOAD 1/Ø H.
LEADING EDGE
G11Ø(2)
SET STROBE F/F
2 5 Ø N S
VARIABLE
CLEAR STROBE F/F TO ESTABLISH STROBE PULSE WIDTH APROX 5Ø NS (ACTS LIKE ONE SHOT)
TRAILING EDGE
G11Ø(2)
ENABLE STROBE PULSE
GATE DELAY OF E17 TO CLEAR STROBE F/F
DATI OR DATIP ?
YES
TRAILING EDGE
YES
DATI DATIP DATOB
G11Ø(2)
CLOCK & CLEAR SSYN F/F
G11Ø(2)
SELECTIVELY ACTIVATE STROBE 1/Ø H
G11Ø(4)
ACTIVE SENSE AMPS WILL DIRECT SET DATA F/F 3
NOTE: DATOB ACTIVATES SENSE AMPS FOR BYTE THAT WILL BE RETAINED. (SEE CHART)
DATIP
G11Ø(2)
ACTIVATE DATO H SIGNAL
G11Ø(2)
SEND BUS SSYN TO CPU
G11Ø(2)
SET PAUSE F/F
G11Ø(4)
ENABLE BUS DATA DRVRS TO DATA BUS
G11Ø(2)
DISABLE WRITE RESTART

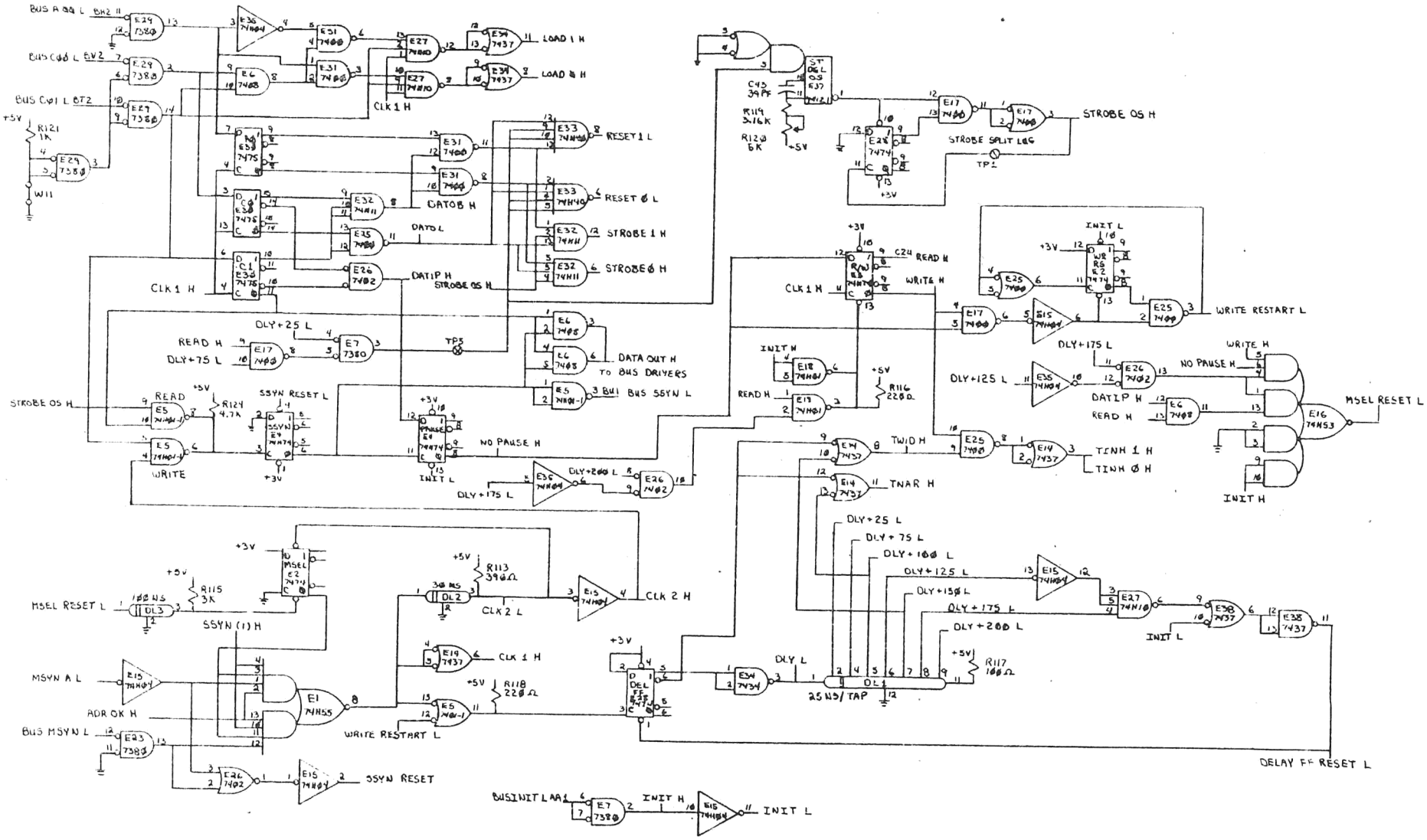

LATCHED BY START DELAY AT TIME +30NS
1L
5L
PICKED UP AND HELD BY DL1 WHEN START DELAY DROPS. RELEASES AT +310NS
G110(2)
DISABLE TNAR +310NS
G231(2)
DISABLE T55H, TURN OFF SWITCH DECODERS
G231(2)
DISABLE TDRH, TURN OFF DRIVER DECODERS


1L
7L
G110(2)
DISABLE TWID AT +360NS
G231(2)
TURN OFF X/Y CURRENT GENERATOR



A SECOND PASS IS MADE DOWN DLL WITH THE READ/WRITE F/F CLEAR. THIS DOES A WRITE RESTORE AT THE SELECTED LOCATION INHIBIT DRIVERS GO ON WITH TWID AND SENSE AMPS ARE DISABLED DURING THIS PASS.


UNIBUS
BUS A<13:01>
BUS A<17:14>, <00>
BUS MSYN
BUS C<1:0>
BUS DC LO
BUS INIT
BUS SSYN
MEMORY TIMING AND CONTROL
CDL2
CLK
LOAD 0/1
RESET 0/1
STROBE 0/1
DATA OUT
TINH 0/1
READ
TNAR
TWID
13 ADRESS REG 01
MD2
ADR. DECODER
MD2-4
X & Y SWITCHES, DRIVERS
MD2-4
CURRENT GENERATOR
TWID
X LINES
Y LINES
CORE STACK
BUS D<15:08>
BUS D<07:00>
RCR
RCR
LOAD 1
LOAD 0
15 8 7 0
SENSE AMPS
CDL3
CDL4
STROBE 1
STROBE 0
15 8 7 0
INH DVRS
CDL3
CDL4
15 8 7 0
DATA REG
CDL3
CDL4
TINH 0/1
XMT
XMT
DATA OUT


BUS A 00 L
BUS C00 L
BUS C01 L
+5V
R121 1K
W11
LOAD 1 H
LOAD 0 H
CLK 1 H
RESET 1 L
RESET 0 L
STROBE 1 H
STROBE 0 H
DATO B H
DATO L
DATI P H
STROBE OS H
DLY+25 L
READ H
DLY+75 L
TP5
DATA OUT H
TO BUS DRIVERS
BUS SSYN L
SSYN RESET L
R124 4.7K
READ
WRITE
NO PAUSE H
INIT L
DLY+200 L
DLY+175 L
C43 39PF
R119 3.16K
R120 6K
STROBE SPLIT LEG
TP1
STROBE OS H
C24 READ H
WRITE H
INIT H
R116 220Ω
DLY+125 L
DATI P H
READ H
NO PAUSE H
WRITE H
MSEL RESET L
WRITE RESTART L
TWID H
TINH 1 H
TINH 0 H
TNAR H
INIT H
DLY+25 L
DLY+75 L
DLY+100 L
DLY+125 L
DLY+150 L
DLY+175 L
DLY+200 L
R117 100Ω
25 NS/TAP
DELAY FF RESET L
MSEL RESET L
100 NS
DL3
R115 3K
SSYN (1) H
R113 390Ω
30 NS
DL2
CLK 2 L
CLK 2 H
CLK 1 H
MSYN A L
ADR OK H
BUS MSYN L
R118 220Ω
WRITE RESTART L
SSYN RESET
BUS INIT L
INIT H
INIT L

AWBY	Await BUS busy
ALUM	ALU Mode Select
ALL DOP	All double Operand instructions
BUS A + NO	A xx where xx is the Bit # of the address
ALL SOP	All single Operand instructions
ADRS	Address
ALU	Arithmetic and Logical Unit
ALUS	ALU Select Bits
BR	BUS Request
BUS SACK	BUS Selection Acknowledge
BRQ	BUS Request
BRPTR	BUS Request - priority transfer request
BCON	Binary constant
BOVFL STOP	Basic Overflow (red zone)
BOVFL	Basic Overflow (yellow zone)
BERR	Bus Error
BRSV	BUS Request service
B INTR	BUS Interrupt
BSSY	Busy
BC	B Constant bits
BA MUX	BUS Address Multiplexer
BUS "D"	BUS Data bits
B + NO	B xx where xx is the bit number of B Register
B MUX	Partitioned Multiplexer (BREG - BCON)
BUS	Unibus
BUS RD	Register Data BUS (BUS Register Data)
BUBC	Basic Micro Branch Control Bits
BUT	Branch Micro Test
BUS FM SR	BUS from switch register
BUS FM PS	BUS from processor status register
BUS NPR	BUS non-processor request
BUS BG	BUS grant to a BR Device
BUS RD FM. PS	Register Data BUS from processor status register
CLK	Clock
CIR	Clock Instruction Register
CB	Clock "B" Register
CD	Clock "D" Register
CBA	Clock BUS Address Register
CONS	Console
CBR	Console BUS Request
CONSL	Console
C DATA	(Data Input to PS (C) F/F)
CLK BA	Clock the BUS address register
COUT MUX	Carry out multiplexer from ALU
COUT	Carry out from ALU
C1 BUS	Control line 1 for data xfers
C0 BUS	Control line 0 for data xfers
CLR	Clear
CIN STR	An instruction that affects condition codes
CC INSTR	Condition code instructions

CASI ADRS	(Example of line not used - necessary for automated drafting system)
DOP	Double Operand
D MUX	Data Multiplexer
D + NO	D xx where xx the bit of D Register
D(C)	Logical HI order one bit extension of "D" Register
DAD	Discrete Alteration of data
DM	Destination Mode
DMO	Destination Mode Zero
DISABLE TSACK	Disable the timing for selection acknowledge time out
DUBBERR	Double BUS error
EUPP	Extended or expansion Micro Program Pointer (EIS)
ESALUM	Extended or expansion selection of ALU Mode (EIS)
ECLK U	Extended or expansion micro word register clock (EIS)
EUBC	Extended or expansion micro branch control (EIS)
IOT	Input/Output Trap
ILL INSTR	Illegal inst. (Inst. that is not executable in the 11/40 under any conditions)
IR	Instruction Register
INH PS CLK	Inhibit Processor Status Register Clocking
JPUP	A F/F that initiates a jam sequence when halt sw, start sw activates
JAM UPP	Jam the micro program pointer with new number
INIT 2	Initialize
KT INSTR	(MM) Memory MGMT Instruction
KT SR	SW16, SW17
MM	Memory Management
MODE SEL 1	(MM) the HI order bit of mode being multiplexed thru the mode mux
NODAT	No Data
NO SACK	No selection acknowledge
OB	Odd Byte
ODA ERR	Odd Address Error
PTRD	Priority transfer delayed
PROC RELEASE	Processor release (when someone drops BBSY)
PART P END	Part of the clock pulses provide a pulse at the end (Clk 2 or Clk 3)
PEND	Pulse at end of any clk selection
PART DOP	Part of the double operand inst
PRIV INSTR	Priviledged instruction - halt or resat instructions
PAST B	Past B input to the ALU
PAST C	What PS(C) was prior
PAST A	The past sign of the A input to ALU
PERIF RELEASE	Peripheral release (when a BR device drops BBSY)
PUPP	Past Micro Program Pointer - address of inst whose contents are now in the UREG
PTR	Priority Transfer
PS	Processor Status
ROTSHF	Rotate and shift instructions
RSVD INSTR	Reserved Instructions - instructions resv for future use
R [SF]	Some GPR (0-7) specified by the source field of instruction currently in IR
R [SOURCE]	GPR 11
RIF	Register immediate field
RG ADRS	- RADRS - Register Address
SLR	Stack limit register address
SR ADRS	Switch Register (Console) Adrs
S HOOK	Source Reg Hook to select R16 instead of R6 when needed
D HOOK	Destination Reg Hook to select R16 instead of R6 when needed

SRI	Select the register (using the RIF) immediate
SM	Source Mode
SR	Switch Register (Console)
SRD	Select the register of the destination-as specified by the dest. field of the inst currently
SRS	Select the register of the source-as specified by the source field of the inst currently in
SDM	Select the D MUX
SBAM	Select the BUS Address MUX
SPS	Select the processor status register
SBC	Select the binary constant (generate a number)
SOP MORE	More of the single operand instructions
SWAB	Swap Bytes
SBML	Select the function of the low byte of B MUX
SXT	Sign extend instruction
SEXB	Sign extend thru the B MUX Reg thru HI Byte
SOP	Single Operand
SPS	Select processor status
SRBA	Select the GPR - (0-17) specified by the LO order 4 bits of the BA Register
SOB	Subtract one and branch
SF	Source Field
SBMH	Select the function of the HI Byte of BMUX
TIME SACK	Time selection acknowledge
UBF	Micro Branch Field
UPP	Micro Program Pointer
WR	Write
XOR	Exclusive "OR"

11/40 PROCESSOR TIMING

CLK U — (JAM CLK + P1 + P2 + P3) — TRAILING EDGE
CLK (UPP & AUPP)

PEND — P1 + P2 + P3 — LEADING EDGE

PART P END — ~~P1~~ + P2 + P3 — LEADING EDGE

PS (P1+P3) — P1 + P3 — LEADING EDGE

(P1+P3) — P1 + P3 — LEADING EDGE

CLK B — P1 + P3 — ~~TRAILING~~ EDGE

WR R — P1 + P3 — LEADING EDGE

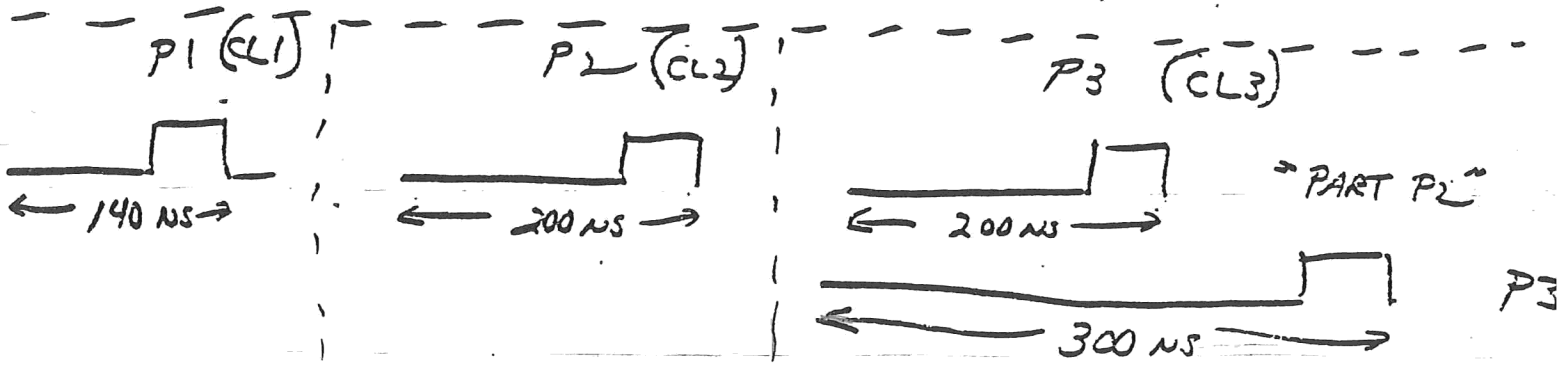
CLK IR — P1 + P3 — TRAILING EDGE

CLK BUS — P1 + P2 — TRAILING EDGE

CLK BA — P1 + P2 — TRAILING EDGE

CLK D — P2 — TRAILING EDGE

NOTE: THERE WILL ONLY BE ONE OF THE "P" PULSES IN ANY GIVEN ROM STATE. THAT IS, IF THERE IS A P1, THERE IS NO P2 OR P3, ETC.



ASYNCHRONOUS RES

IN CERTAIN AREAS OF THE MICRO-PROGRAM THE CLOCK WILL BE TURNED ON BY AN ASYNCHRONOUS CONDITION.

PENORESET: USED TO RESTART THE CLOCK UPON COMPLETION OF THE BUS INIT GENERATED DURING A RESET I

BR*INTR: USED TO RESTART THE CLOCK AFTER RECIEPT OF A BUS INTR. (ACTIVE)

BR*PERIF RELEASE: USED TO RESTART THE CLOCK AFTER IT WAS TURNED OFF TO AWAIT AN INTR BUT NO INTR WAS RECIEVE.

PERIF RELEASE = INACTIVE BUS (PASSIVE)
- NO SACK
- NO GRANT
- NO SSYN
- NO BUS MASTER

BBSY(1)H * AWBY(1)H: CPU WILL SET AWBY F/F AND THEN TURN OFF CLOCK. THE CLOCK WILL START AS DETERMINED BY PROCESSORS BUS BUSY F/F. THIS ALLOWS CPU TO STALL MICRO-PROGRAM UNTIL CPU IS MASTER OF BUS.

JAMSTART*-PWR UP INIT: USED TO START CLOCK DURING ONE OF THE FOLLOWING CONDITIONS:

- PWR UP * -INIT
- BUS ERROR (ODA OR BUS TIME-OUT)
- DUBBER
- RED ZONE
- CNSL RESET

1000 = 005212
R7 = 1000
R2 = 2000

016	P1 : BA ← R(R7) = 1000
	DATI clk off
001	IR, R(IR), BA ← unibus data (5212)
004	P2 : D, BA ← [PC + 2] = 1002
005	P1 : R[1002] ← D
161	BA ← R(DF) = 2000
246	Noop clk off
267	B, R[dest] = R12 ← unibus data = 070707
	DAD = 17
	SO PMORE
	Not neg & not SWAB
220	D ← f DAD [(R12)+1] = 070710
211	PS(C) = 0
367	N Z C V clock dA
375	NO OP

since dest mode in IR = 1

DATIP

DAD = 07 = allows odd address & DATOB for byte INST.

1 + Φ = −Φ
1 * 1 = 1

BA = 2000

2000 = 070710 (write occures)

Flows to complete one micro sequence for increment

P2 : D, BA ← R[SP] minus 2

P3 : R[SP] ← D DAD = 6

CLK = 6

WR = 3

CD = 1

SBC = 01

ALU = 06

CBA = 1

SBM = 0

CBA = 1

SDM = 2

DAD = 6

SRx = 1

RIF = 6

FET 5

"L"
CLK=4
CD=1
ALU=11 = add.
SDM=5 sign extend
SRX=0
RIF=7 → R7
UPF=340

CLK=2
WR=3 } Load Reg.
SRX=1
SDM=2
RIF=7
UPF=100

Branches, MARK, SOB } "B"

0001(11 111)

add 1000 = 000777
R7 = 1000

PC=1000

ROM ϕϕ	BA ← R[PC], DATI, CLKoff
ϕ1	IR, R(IR) B ← UNibus data (000777)
0	R(13)
ϕ4	P2: D BA ← R(PC)+2 = 1002
	determine must go to branch
ϕ5	R(PC) ← D = 1002
111	D ← R(PC) PLUS SEXB (177777)
340	R(PC) ← D ← 1001
341	P2: D ← R(PC) PLUS SEXB (177777)
	1001 + -1 = 1000
	P3: R(PC) ← D = 1000

sign extend 377
1111 111 111

177777
1002
1001

R1 = 600

N = 1

177566 = ~~600~~ = 041101

A B C D

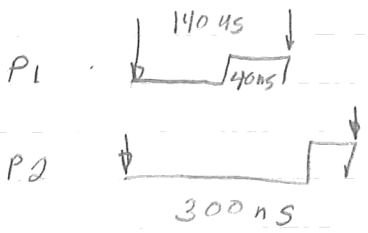
1 0 2
0 1 00 0 0 1 0

0 000 001 0 1 00 010 101

0 0 01 0 1 0 1 0 0 00 0 0 1 0
0 1 2 4 0 2

L1	L6	off	
0	0	X	
0	1	X	P1
1	0	X	P2
1	1	1	P3

Signals are listed as octal words with the off bit in micro code.



→ clock signals

337 D ← 24 DATI
(R14) ← D = 24

334 BA ← 26 DATI

335 PSW ← (26)

332 BA ← 24 DATI

333 R7 ← New PC (024)

123

015

FET A to get FET of 1st Pul

1000/ 061242
R7 = 1000
R2 = 2000
2000/ 070707
1776/ 105050

ADD INSTR

013	BA ← R(PC) = 1000
001	IR, R(IR) ← unibus data (DATI, CLKOFF) = 61242
004	D, BA ← R[PC] + 2 = 1002
005	R[1002] ← D
141	BA ← R[SF] = 2000
247	NO-OP; CLKoff
250	B, R[source] ← unibus = 070707 (R11)
164	D, BA R[DF] − 2 (R12)
260	R12 ← D = 1776
267	B, [R12] ← unibus data 105050

1000/061242
1002/ 000000

Load Adress / Deposit

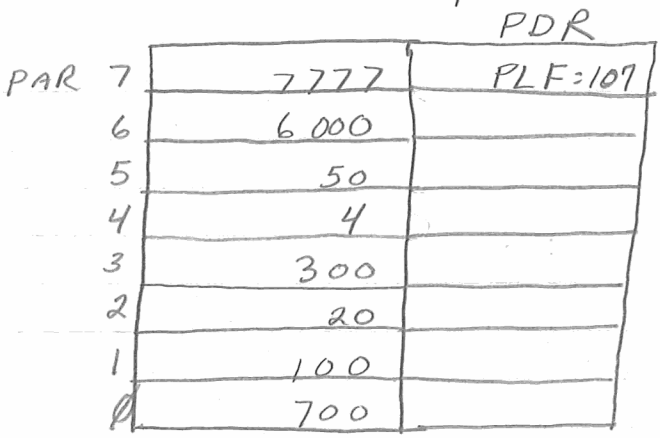
LOAD Address

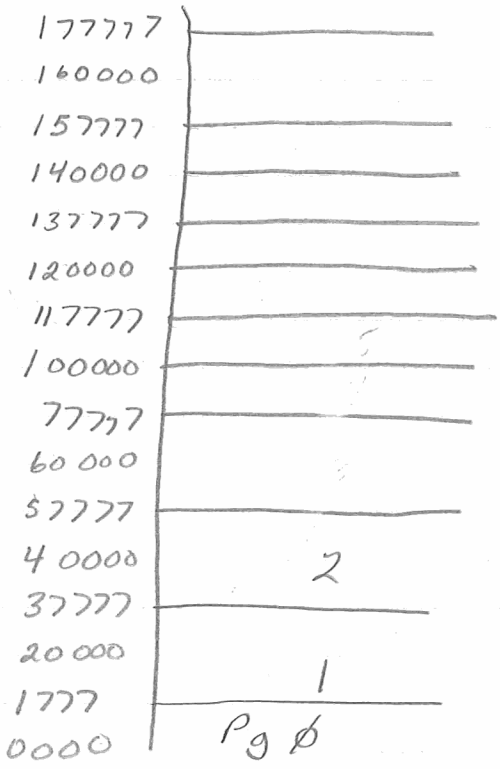
ϕ27	D=ϕ
ϕ44	R (TempC) = ϕ R15
ϕ47	D ← 1
ϕ50	D ← R17
ϕ37	BA ← 77757ϕ
ϕ51	R17 ← 1ϕϕϕ
ϕ52	D, BA ← 1ϕϕϕ
ϕ33	D ← ϕ
ϕ30, 315, 046	

ϕ277	D ← ϕ
ϕ44	(R15) ← ϕ
ϕ47	D ← 1 (R15) ← 1
ϕ50	D ← 1ϕϕϕ
ϕ34	BA ← 1ϕϕϕ
ϕ63	D, BA ← 1ϕϕϕ BA ← 777570 B ← ϕ61242

07ϕ	D ← 061242 Dep=1
072	DAT.ϕ 1000/061242
ϕ63	D, BA ← 10ϕϕ+1 R17 ← 1001
ϕ64	D, BA ← 10ϕ+1 R17 ← 1002
ϕ72	DATϕ 1002/ϕϕϕϕϕϕ

Example.





017677
777700
67777
600000
24777
5000
20377
400
47777
3000
21777

3000
27777

10000
107777

700 00

Example

10000/ 015237

PA. = 100000

10002/ 177566

100002

10004/ N.I.

000 | 10000
70000
100000

PA 100000 = get instr.

PA 17776 = get src add.

PA 70070 get src

PA 100002 get dst

R2 = 30000

027776 = 7

70 = 7

17 566
777700
*017466

PAR | 1000 |

12 bits

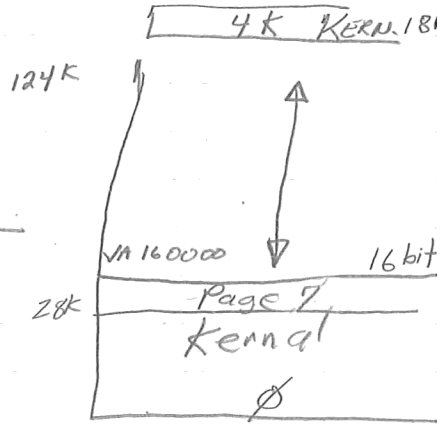
VA | | 6 bits |

PAR | |

= 160100

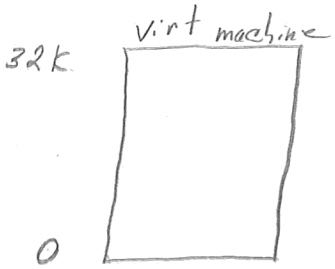
1000

P.A. 260000



KERNAL mode = operating system

EACH USER has a virtual machine



1 110 000 000 000 000 000

PAR 7

∴ 0000 17777

7777 7777

77770000 017677

Base pg Address

= Phy pg = PAR x 100

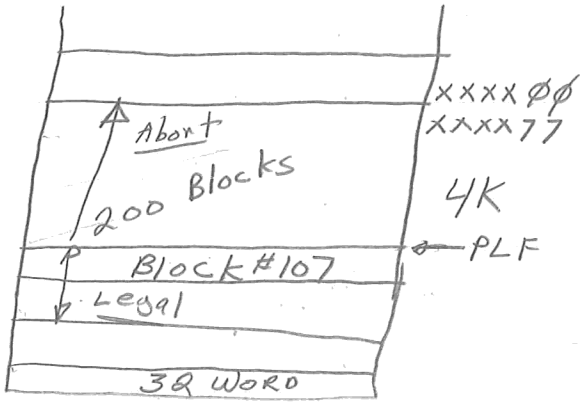
Hi end = Base + 17777

V.A. (Virtual address)

Sel. PAR	Block #	DIB = Displacement IN Block
15 13	12 6	5 0
0 00	0 00000	000 000

Max page size

= 4K words
= 200_8 blocks
0 → 177
1 Block to 200 blocks



(Page length field) PLF is the last legal address up or down
ED = 0 means pg expands upward
ED = 1 pg expands down.

PDR = 16

Last block = -0
ED = down w/ read write

gives full page expanding downward.

TABLE OF LOGIC FUNCTIONS

Function Select				Output Function	
S3	S2	S1	S0	Negative Logic	Positive Logic
L	L	L	L	$F = \overline{A}$	$F = \overline{A}$
L	L	L	H	$F = \overline{AB}$	$F = \overline{A+B}$
L	L	H	L	$F = \overline{A}+B$	$F = \overline{A}B$
L	L	H	H	F = Logical 1	F = Logical 0
L	H	L	L	$F = \overline{A+B}$	$F = \overline{AB}$
L	H	L	H	$F = \overline{B}$	$F = \overline{B}$
L	H	H	L	$F = \overline{A \oplus B}$	$F = A \oplus B$
L	H	H	H	$F = A+\overline{B}$	$F = A\overline{B}$
H	L	L	L	$F = \overline{A}B$	$F = \overline{A}+B$
H	L	L	H	$F = A \oplus B$	$F = \overline{A \oplus B}$
H	L	H	L	F = B	F = B
H	L	H	H	F = A+B	F = AB
H	H	L	L	F = Logical 0	F = Logical 1
H	H	L	H	$F = A\overline{B}$	$F = A+\overline{B}$
H	H	H	L	F = AB	F = A+B
H	H	H	H	F = A	F = A

With mode control (M) high: C_n irrelevant
For positive logic: logical 1 = high voltage
logical 0 = low voltage
For negative logic: logical 1 = low voltage
logical 0 = high voltage

TABLE OF ARITHMETIC OPERATIONS

Function Select				Output Function	
S3	S2	S1	S0	Low Levels Active	High Levels Active
L	L	L	L	F = A minus 1	F = A
L	L	L	H	F = AB minus 1	F = A+B
L	L	H	L	$F = A\overline{B}$ minus 1	$F = A+\overline{B}$
L	L	H	H	F = minus 1 (2's complement)	F = minus 1 (2's complement)
L	H	L	L	$F = A$ plus $[A+\overline{B}]$	$F = A$ plus $A\overline{B}$
L	H	L	H	$F = AB$ plus $[A+\overline{B}]$	$F = [A+B]$ plus $A\overline{B}$
L	H	H	L	F = A minus B minus 1	F = A minus B minus 1
L	H	H	H	$F = A+\overline{B}$	$F = A\overline{B}$ minus 1
H	L	L	L	F = A plus [A+B]	F = A plus AB
H	L	L	H	F = A plus B	F = A plus B
H	L	H	L	$F = A\overline{B}$ plus [A+B]	$F = [A+\overline{B}]$ plus AB
H	L	H	H	F = A+B	F = AB minus 1
H	H	L	L	F = A plus A†	F = A plus A†
H	H	L	H	F = AB plus A	A = [A+B] plus A
H	H	H	L	$F = A\overline{B}$ plus A	$F = [A+\overline{B}]$ plus A
H	H	H	H	F = A	F = A minus 1

With mode control (M) and C_n low
† Each bit is shifted to the next more significant position.

A.16 74181 4-BIT ARITHMETIC UNIT WITH FULL LOOK-AHEAD

The 74181 performs up to 16 arithmetic and 16 logic functions. Arithmetic operations are selected by four function-select lines (S0, S1, S2, and S3) with a low-level voltage at the mode control input (M), and a low-level internal carry. Logical operations are selected by the same four function-select lines with the exception that the mode control input (M) must be high to disable the internal carry.