15644

PRIMOS

Internal Structure

COURSE NOTES

D O S V M (internal structure) C O U R S E

DAY 1

10.30 - 12.00	Introduction to the concepts of DOSVM Virtual Machines
14.00 - 15.30	Supervisor calls
16.00 - 17.00	I/O virtualisation. Interrupt handling

DAY 2

9.00 - 10.00	Drivers for ASR, PTR/P and Serial Printer
10.30 - 12.00	AMLC driver
14.00 - 15.30	The Scheduler
16.00 - 17.00	Internal commands

DAY 3

9.00 - 10.00	The file system - SEARCH, ATTACH, PRWFIL
10.30 - 12.00	Internal operation of the file system; associative buffers
14.00 - 15.30	Drivers for line printer, card reader, mag tape.
16.00 - 17.00	A preview of PRIMOS 4 and the P400

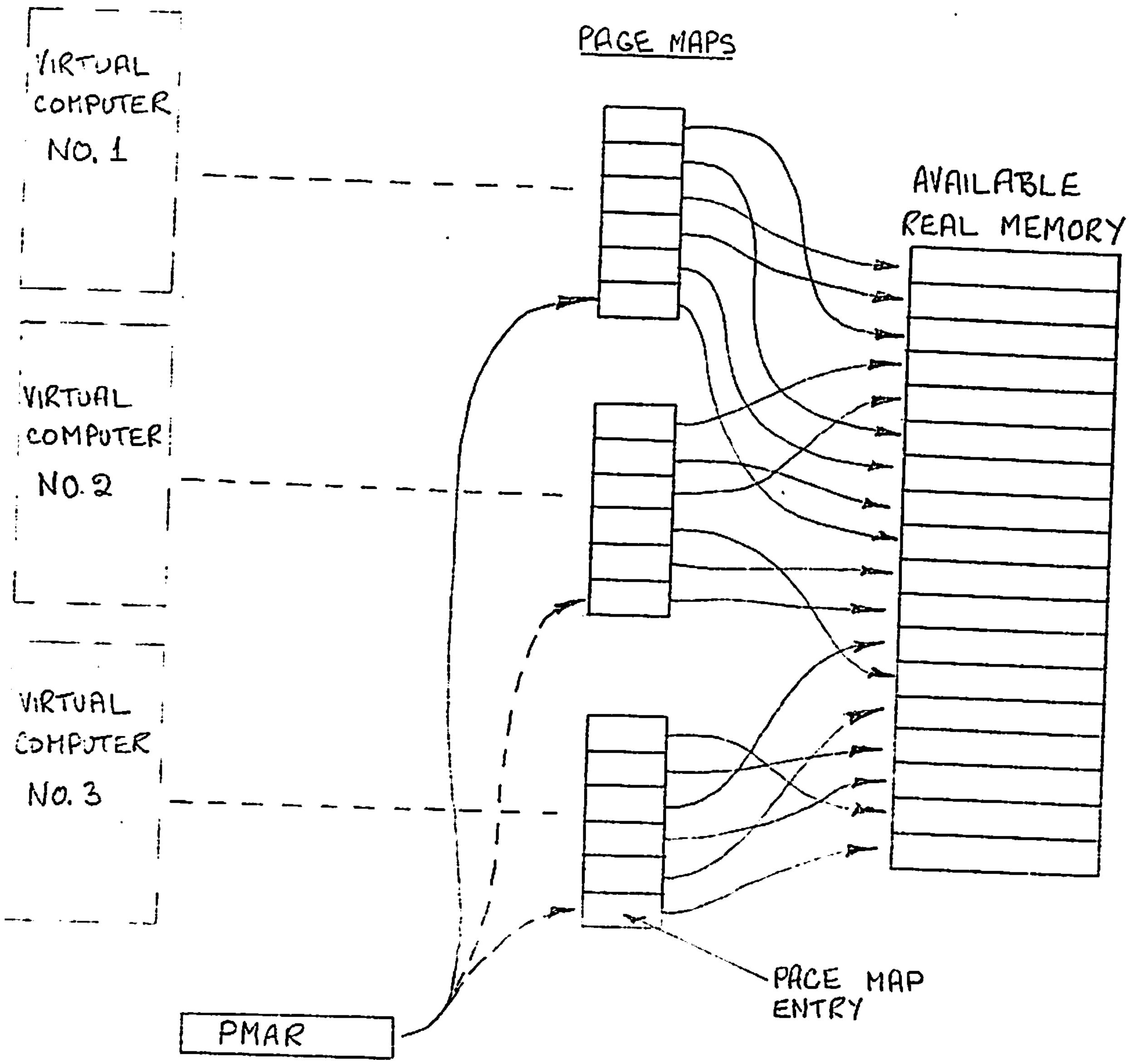
Bob Slater

DOSVM PROVIDES UP TO 32 VIRTUAL COMPUTERS ON ONE PHYSICAL COMPUTER. DOSVM ITSELF IS ONE OF THESE VIRTUAL COMPUTERS.

IN ORDER TO DO THIS, IT MUST SHARE THE RESOURCES OF THE REAL COMPUTER WITH ALL THE VIRTUAL COMPUTERS. THE FOLLOWING RESOURCES MUST BE SHARED:

1) HIGH SPEED MEMORY
2) C.P. TIME
3) PERIPHERAL DEVICES.

THE PRINCIPLE OF VIRTUAL ADDRESSES AND MEMORY MAPPING FOR MULTIPLE USERS



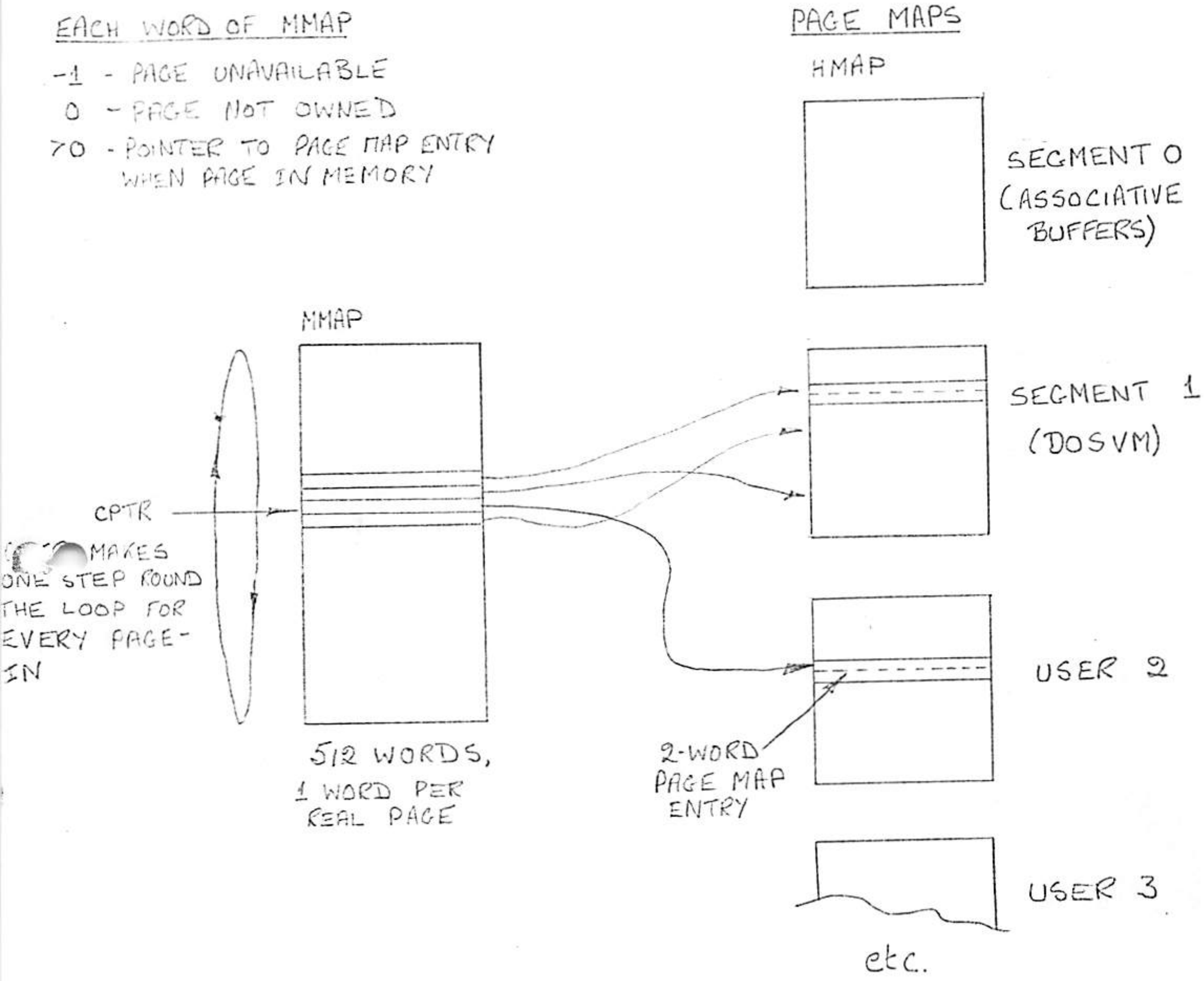
BUT WHAT IF THERE'S NOT ENOUGH REAL MEMORY FOR ALL THE USERS - SOME MEMORY MUST BE SAVED ON DISC - PAGING

R.H. SLATER.

PAGE MAP ENTRY

<table>
<tr><th></th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th></tr>
<tr><td>WORD 1</td><td>IN HSM</td><td colspan="2"></td><td>NO COPY ON DISC</td><td>OLD COPY ON DISC</td><td></td><td>WRITE PROT-ECT</td><td colspan="9">PHYSICAL MEMORY PAGE NUMBER</td></tr>
<tr><td>WORD 2</td><td colspan="3">NON-ZERO IF PAGE LOCKED IN MEMORY</td><td>ALT. PAGE DISC</td><td colspan="12">RECORD ADDRESS OF PAGE ON DISC</td></tr>
</table>

ONLY BITS 1 AND 7-16 OF WORD 1 ARE USED BY THE HARDWARE



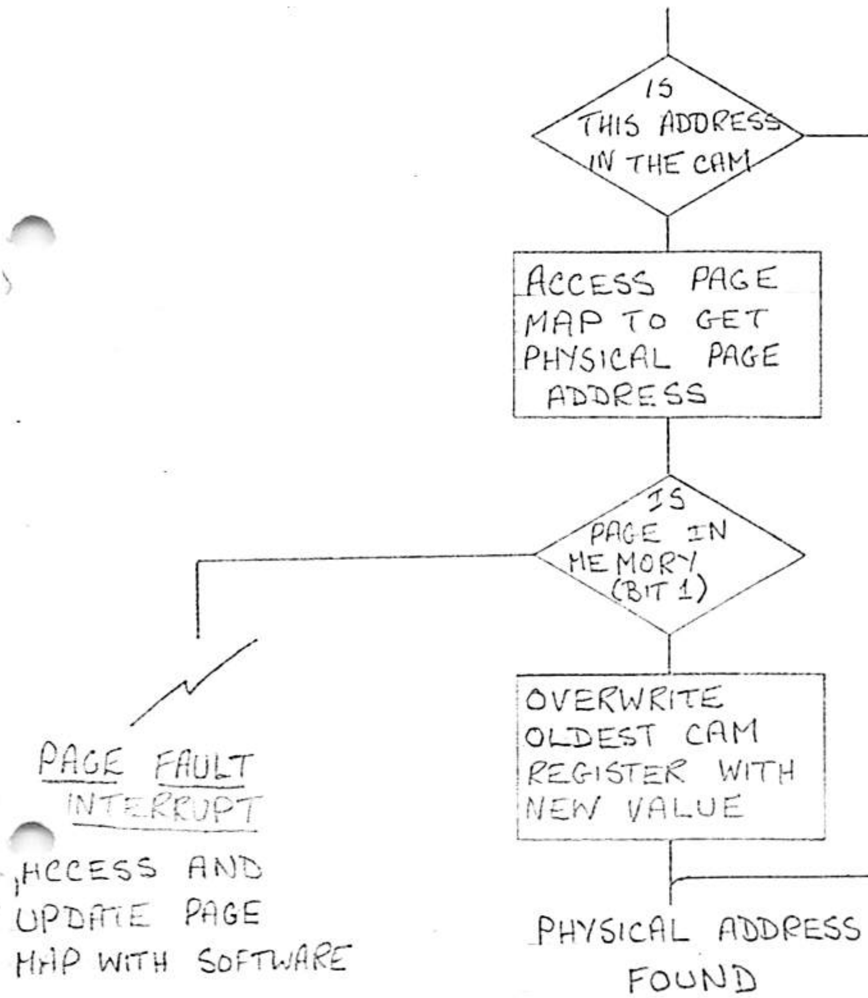
CPTR POINTS TO THE PAGE TO BE WRITTEN OUT TO DISC. THE "ROUND ROBIN" SCHEME MEANS THAT THE PAGE TO BE PAGED OUT IS THE ONE THAT HAS BEEN IN MEMORY THE LONGEST

CAM REGISTERS

AS THE PAGE MAPS ARE STORED IN HIGH SPEED MAIN MEMORY, ACCESSING THEM WILL TAKE ONE FULL MEMORY CYCLE. IN ORDER TO REDUCE THIS TIME, 4 REGISTERS, THE CONTENT ASSOCIATIVE MEMORY REGISTERS (C.A.M.) ARE PROVIDED WHICH CONTAIN THE LAST 4 VIRTUAL ADDRESSES ACCESSED. IF THE VIRTUAL ADDRESS IS FOUND WITHIN THE CAM, INSTRUCTION TIMES ARE INCREASED ONLY BY 80 NSECS.

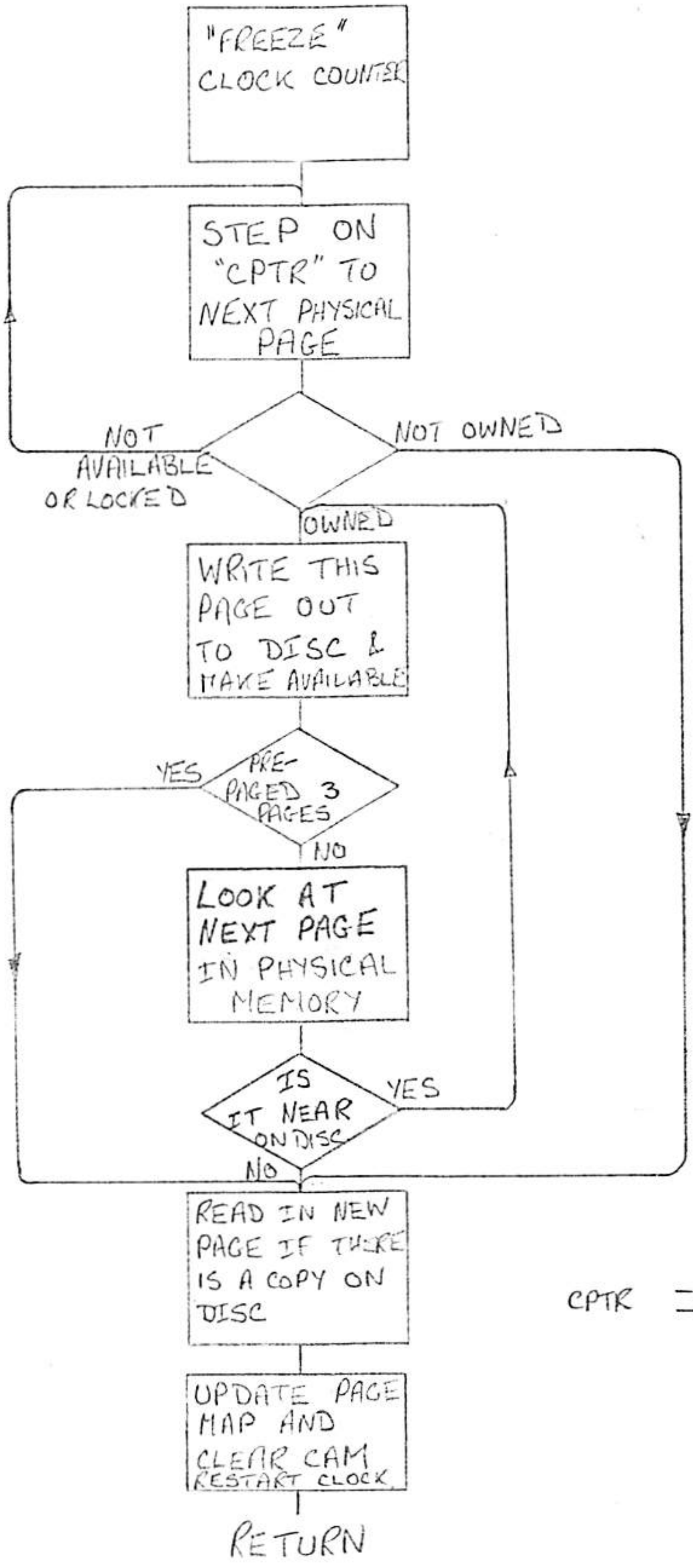
AS EXISTENCE OF A VIRTUAL ADDRESS IN CAM IMPLIES THAT THE PAGE IS MEMORY RESIDENT, EVERY PAGING OPERATION MUST CLEAR THE CAM REGISTERS.

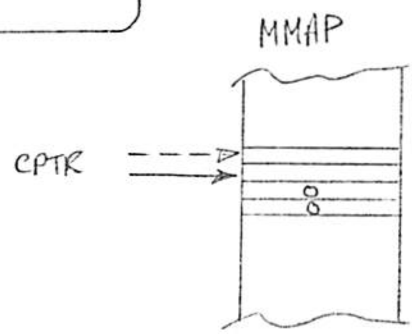
ACTION OF HARDWARE WHEN A USER ACCESSES A VIRTUAL ADDRESS

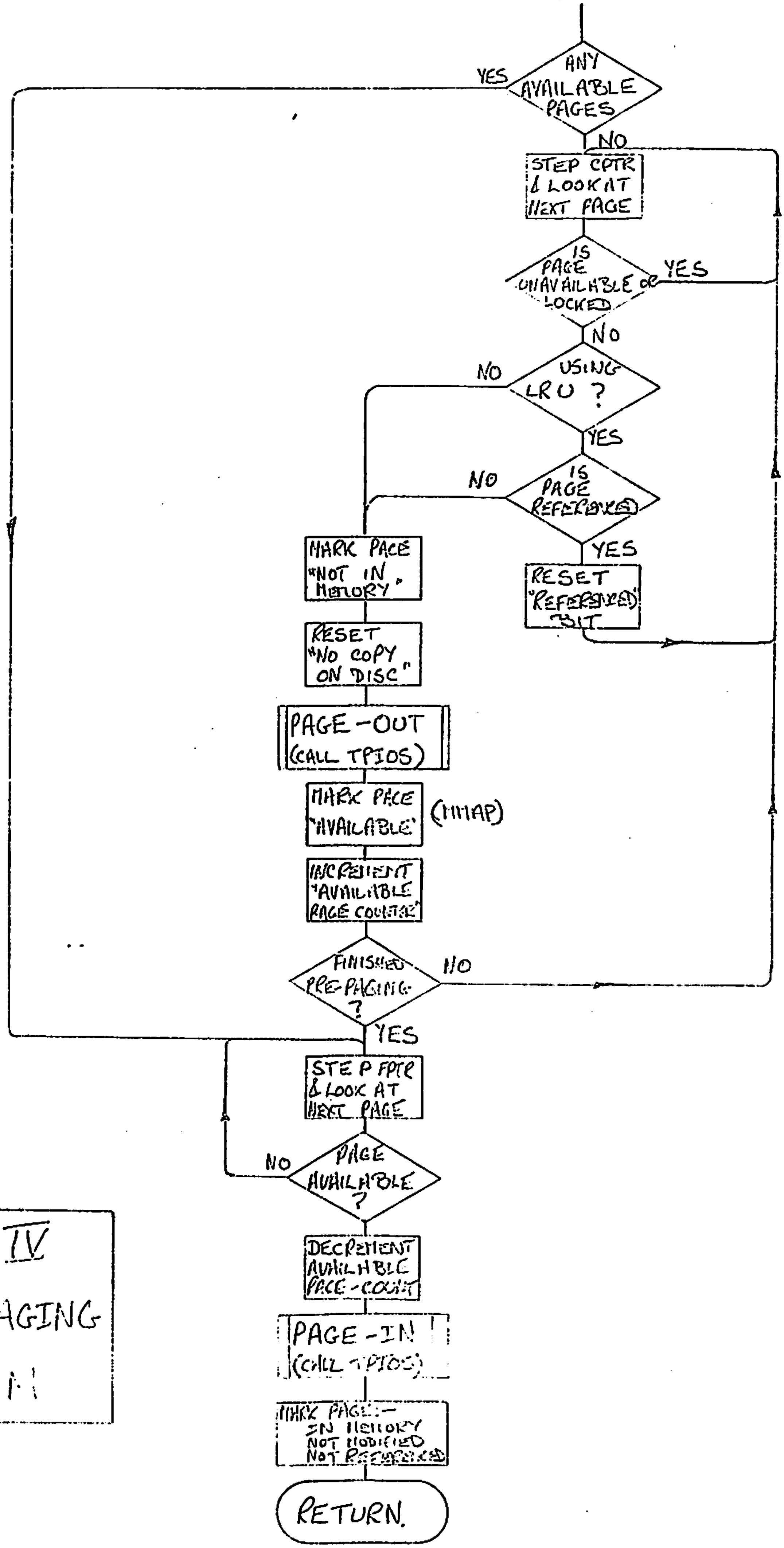


PAGE TURNING IN DOSVM

SUBROUTINE PAGTRN (USER, VIRTUAL ADDRESS)







PRIMOS IV

REV. 11 PAGING

ALGORITHM

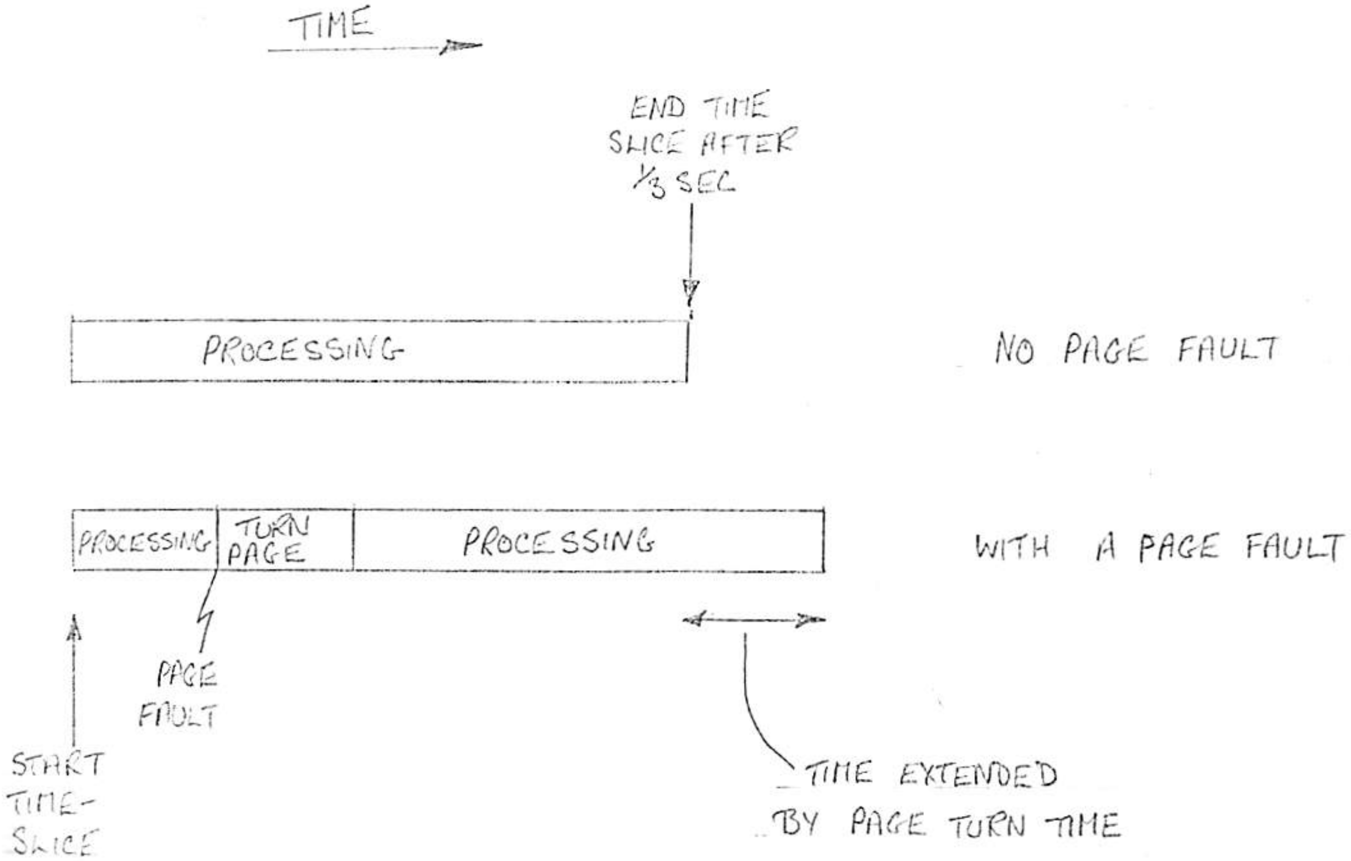
DURING PAGING A FLAG, "PSWI" IS SET WHICH CAUSES THE DOSVM TIME KEEPING ROUTINES TO INCREMENT PAGING TIME INSTEAD OF C.P.U. TIME. A NOMINAL 6 MSECS C.P.U. TIME IS CHARGED TO THE USER FOR EVERY EXECUTION OF "PAGTRN".

PAGRN CALLS THE ROUTINE "TPIOS" TO DO THE BASIC PAGE-DISC READING AND WRITING. TPIOS HANDLES SUCH COMPLICATIONS AS SPLIT DISCS, DIFFERENT RECORD SIZES ON FIXED HEAD DISCS, AND THE USE OF ALTERNATE PAGING DEVICES.

THE MAXIMUM NUMBER OF PAGES THAT MAY BE PRE-PAGED (PREPGK) IS SET TO 3 IN STANDARD DOSVM.

PRE-PAGING LOOKS AT UP TO 3 OF THE NEXT AVAILABLE, UNLOCKED PAGES TO SEE IF THEY CAN BE PAGED OUT. IF A PAGE IS FOUND WHOSE DISC RECORD ~~ADDRESS DIFFERS~~ IS ON THE SAME TRACK ~~BY LESS THAN 7~~, IT WILL BE PAGED OUT. IMMEDIATELY A DISC ADDRESS WHICH IS TOO FAR AWAY IS FOUND, PRE-PAGING ATTEMPTS ARE ABORTED. PRE-PAGING ONLY AFFECTS PAGE-OUTS. PAGE-INS STILL DEPEND ON DEMAND.

THE AMOUNT OF PRE-PAGING MAY BE CHANGED FROM 3 BY CHANGING THE LOCATION 'PREPGK'.

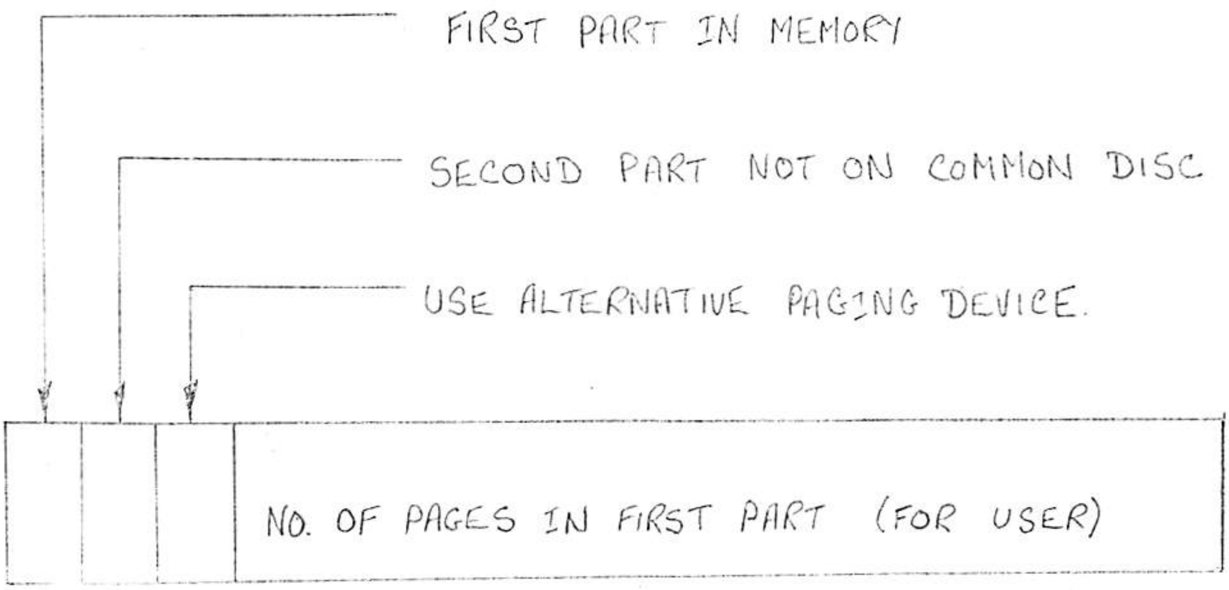


TIME
END TIME
SLICE AFTER
1/3 SEC
PROCESSING
NO PAGE FAULT
PROCESSING
TURN PAGE
PROCESSING
WITH A PAGE FAULT
PAGE
FAULT
START
TIME-
SLICE
TIME EXTENDED
BY PAGE TURN TIME

CUSTOM PAGE MAPS

THE STANDARD VERSIONS OF DOSVM PROVIDE ALL USERS WITH 64K VIRTUAL MEMORY SPACE. THE PAGE MAPS ARE NOT WRITTEN AS CODE BUT GENERATED BY A PROGRAM (MAKML6 OR MAKM32) IN MEMORY, SAVED AND THEN RESTORED ON TOP OF THE DOSVM MEMORY IMAGE FILE.

CUSTOM PAGE MAPS MAY BE GENERATED IN ORDER TO REDUCE PAGEING DISC SPACE REQUIREMENTS BY MODIFYING THE SOURCE CODE OF MAKM16 OR MAKM32 AND RE-EXECUTING IT TO GENERATE NEW PAGE MAPS. THE SOURCE CODE CONTAINS A TABLE OF WORDS, ONE FOR EACH VIRTUAL COMPUTER. EACH WORD HAS THE FOLLOWING FORMAT:-



```
                     (0001) *        GENERATE PAGE-MAPS
                     (0002) *
          000000     (0003) BREG    EQU     #101
          000000     (0004) USR31#  EQU     BREG.AND.8
          030000     (0005)         ORG     '30000
030000:   000000     (0006) COMMON  OCT     0                 COMMON DISK ADDR
030001:   000000     (0007) DSKBEG  OCT     0                 BEGINNING DISK ADDR FOR PAGES OTHER THAN COM
                     (0008) *
                     (0009) *  TABL DESCRIPTION:
                     (0010)      ONE WORD PER USER, -1 AFTER LAST USER
                     (0011) *    BIT 1:   FIRST PART IN-MEMORY
                     (0012) *    BIT 2:   SECOND PART NOT COMMON DISK
                     (0013) *    BIT 3:   PAGES ON ALTERNATE PAGING DEVICE
                     (0014) *    BITS 9-16: NO OF PAGES IN FIRST PART (ALLOCATED SPACE FOR USER)
                     (0015) *
          030002     (0016) TABL    EQU     *
030002:   000060     (0017)         OCT     000060            00 SPECIAL PAGE MAP FOR ASSOCIATIVE BUFFERS
                     (0018)         IFZ     USR31#
030003:   140076     (0019)         OCT     140076            01  IN-MEMORY, 32K-:2000 ALLOCATED
                     (0020)         ELSE
                     (0022)         ENDC
030004:   000200     (0023)         OCT     200               02
030005:   000200     (0024)         OCT     200               03
030006:   000200     (0025)         OCT     200               04
030007:   000200     (0026)         OCT     200               05
030010:   000200     (0027)         OCT     200               06
030011:   000200     (0028)         OCT     200               07
030012:   000200     (0029)         OCT     200               08
030013:   000200     (0030)         OCT     200               09
030014:   000200     (0031)         OCT     200               10
030015:   000200     (0032)         OCT     200               11
030016:   000200     (0033)         OCT     200               12
030017:   000200     (0034)         OCT     200               13  USE ALT PAGE DEVICE
030020:   000200     (0035)         OCT     200               14
030021:   000200     (0036)         OCT     200               15
030022:   000200     (0037)         OCT     200               16
                     (0038)         IFN     USR31#
                     (0039)         ENDC
```

DOSVM SUPERVISOR CALLS (SVC)

ALL DOSVM USERS RUN IN "VIRTUAL MODE". VIRTUAL MODE IS A COMBINATION OF:

A) PAGING MODE WHICH IMPLIES ADDRESS TRANSLATION AND THE POTENTIAL OF GENERATING PAGE-FAULT INTERRUPTS. IT ALSO MEANS THAT THE USER CANNOT JUMP OUT OF HIS VIRTUAL MEMORY.

B) RESTRICTED MODE - WHICH MEANS THAT CONTROL AND I/O INSTRUCTIONS WILL GENERATE A RESTRICTED MODE INTERRUPT.

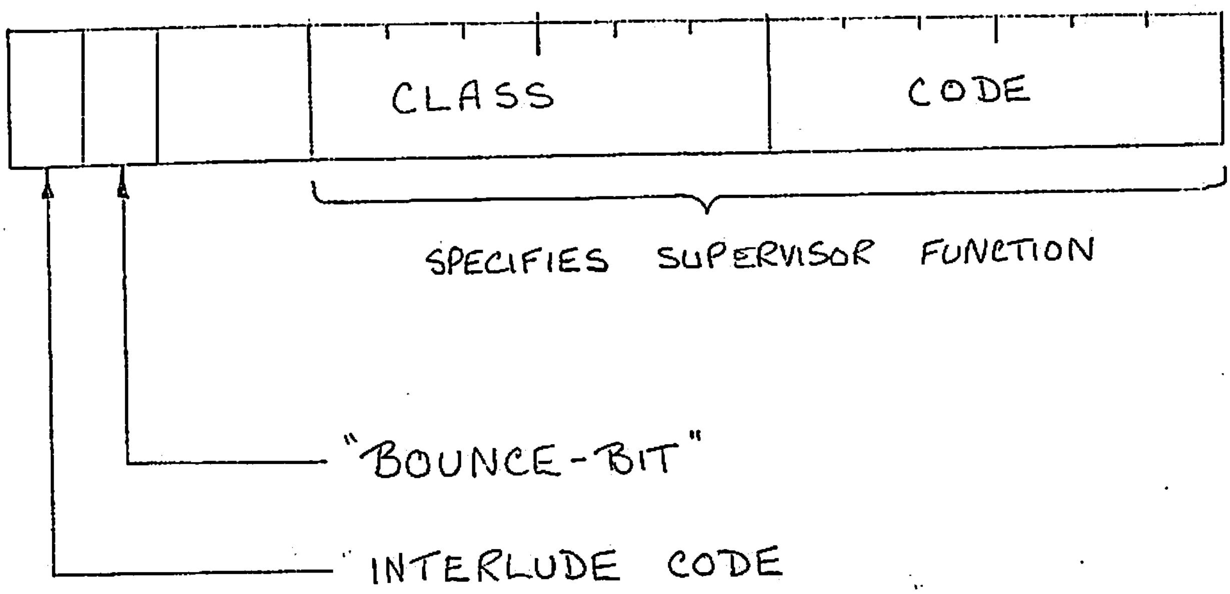
BUT WHAT IF THE USER WISHES TO CALL ON THE SUPERVISOR TO PERFORM A FUNCTION ON HIS BEHALF:

E.G. RETURN INFORMATION
PERFORM DEVICE I/O
ACCESS THE FILE SYSTEM
ETC

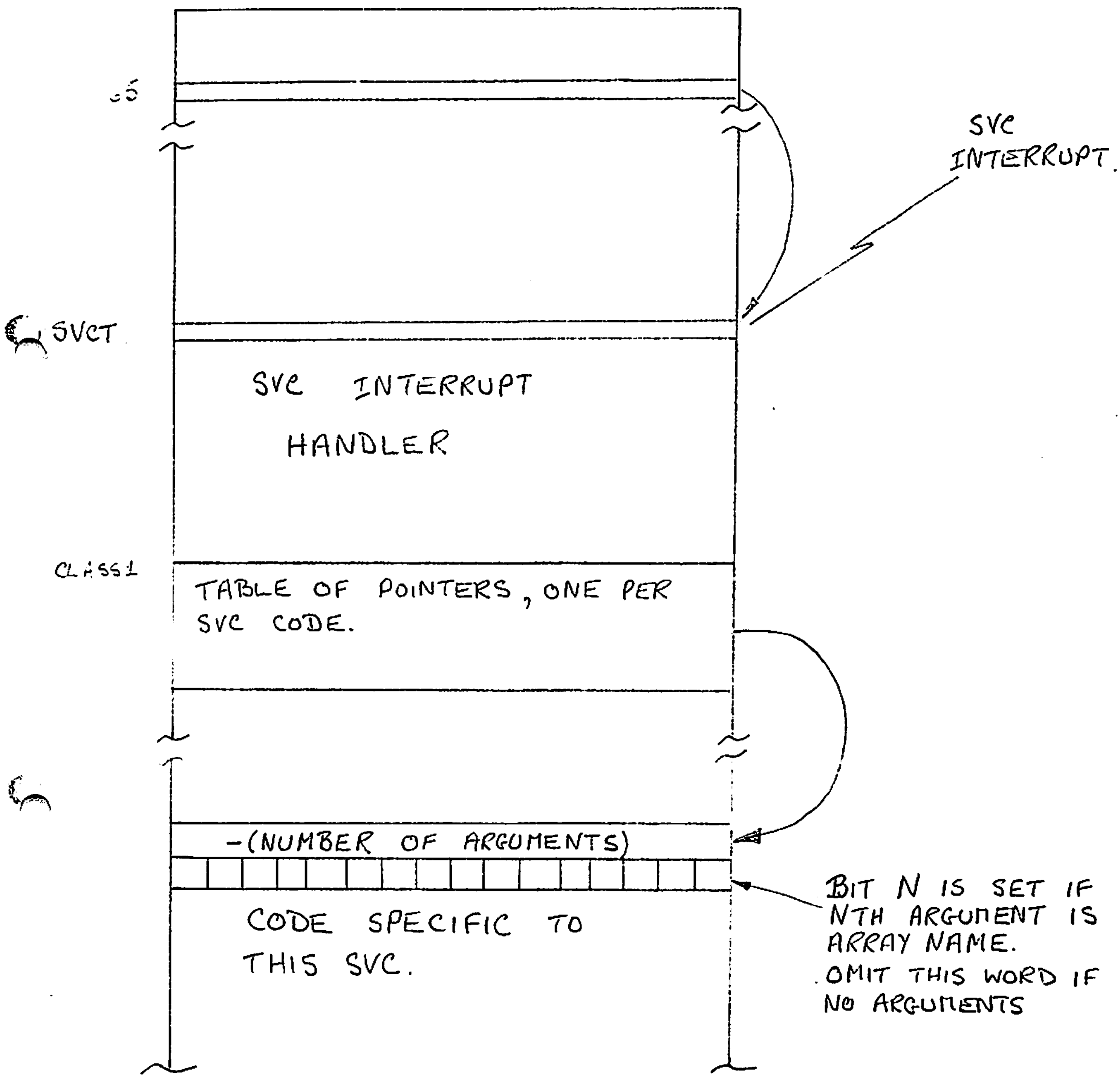
IN THIS CASE THE USER MUST ISSUE THE "SVC" INSTRUCTION, FOLLOWED BY A CODE TO TELL THE SUPERVISOR WHAT HE WANTS IT TO DO.

THE SVC INSTRUCTION IS NOT RESTRICTED, BUT GENERATES AN INTERRUPT USING LOCATION '65 AS THE VECTOR.

SVC CODE FORMAT



DOSVM SVC ENTRY HANDLER



DURING EXECUTION OF AN SVC:-

THE USER'S MACHINE STATE IS STORED IN RVEC (ONE 16-WORD BUFFER PER USER).

THE VALUES OF THE ARGUMENTS PASSED TO THE SVC ARE STORED IN RMVEC. IF AN ARGUMENT IS AN ARRAY NAME, THE ADDRESS OF THE ARRAY IS STORED.

RMVEC IS ONE ARRAY OF 40 WORDS AND IS USED AS WORK SPACE BY SUPERVISOR FUNCTIONS.

DOSVM REV. 8 SVC CODES

CLASS 1

CODE (OCTAL)	NAME	DESCRIPTION
100	ATTACH	ATTACH TO UFD
101	SEARCH	OPEN/CLOSE/DELETE ETC FILE
102	SAVE	SAVE MEMORY IMAGE
103	RESTOR	READ SAVED FILE INTO MEMORY
104	RESUME	EXECUTE SAVED FILE
105	EXIT	RETURN TO COMMAND LEVEL
106	ERRRTN	ERROR RETURN
107	-NOT USED-	
110	GETERR	GET ERROR VECTOR
111	PRERR	PRINT ERROR VECTOR
112	GINFO	GIVE STATUS INFORMATION
113	CNAME	CHANGE FILE NAME
114	ERRSET	SET ERROR VECTOR
115	FORCEW	FORCE BUFFER-WRITE TO DISC

DOSVM REV. 8 SVC CODES

CLASS 2

CODE (OCTAL)	NAME	DESCRIPTION
200	READ	READ FROM DISC
201	WRITE	WRITE TO DISC
202	RDLIN	READ LINE FROM A FILE
203	WTLIN	WRITE LINE TO A FILE

CLASS 3

300	PRWFIL	PSITION/READ/WRITE FILE
301	CNECT$	CONNECT SHARED PROCEDURE
302	ENTRY$	ENTER SHARED PROCEDURE
303	SEXIT$	EXIT FROM SHARED PROCEDURE

CLASS 4 - NONE

CLASS 5 (CANNOT REFLECT)

500	RREC	READ DISC RECORD
501	WREC	WRITE DISC RECORD
502	TIMDAT	GET TIME AND DATE
503	-RESERVED FOR DIGITAL INPUT-	
504	-RESERVED FOR GOULD PLOTTER-	
505	RECYCL	GIVE UP TIME SLICE
506	D$INIT	INITIALISE DISC
507	BREAK	ENABLE/INHIBIT QUIT
510	T$MT	MAG TAPE
511	TLMPC	MPC LINE PRINTER
512	TCMPC	MPC CARD READER
513	T$AMLC	ASSIGNED AMLC LINE HANDLER
514	T$VG	VERSATEC PRINTER/PLOTTER

DOSVM REV. 8 SVC CODES

CLASS 6

CODE (OCTAL)	NAME	DESCRIPTION
600	COMANL	INPUT COMMAND LINE
601	C1IN	INPUT COMMAND CHARACTER
602	CMREAD	
603	CMINP	SWITCH TO/FROM COMMAND FILE
604	CNIN$	

CLASS 7

700	-NOT USED-	
701	-NOT USED-	
702	TNOU	OUTPUT N CHARS TO TTY+NL
703	TNOUA	OUTPUT N CHARS TO TTY
704	-NOT USED-	

CLASS 10

1000	T$MT	MAG TAPE
1001	T$SLC	SMLC

CLASS 11

1100	TLMPC	MPC LINE PRINTER

CLASS 12

1200	TCMPC	MPC CARD READER

CLASS 13

1300, 1301, 1302	RESERVED FOR USERS CUSTOM SVC'S

SVC RETURN

AFTER THE SUPERVISOR HAS PERFORMED OR ATTEMPTED TO PERFORM THE REQUIRED FUNCTION IT GOES TO THE SVC RETURN LOGIC.

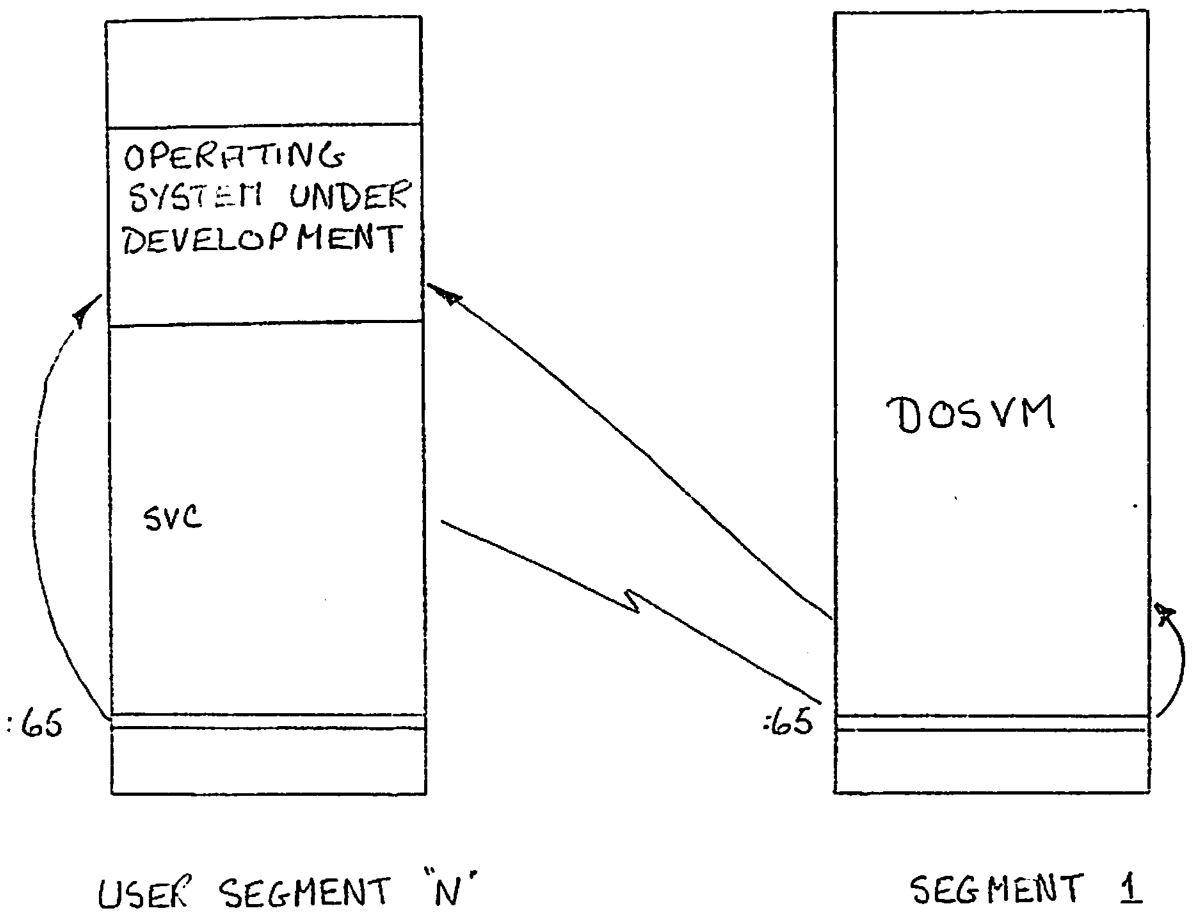
A) FUNCTION COMPLETE - SETS USERS P COUNT IN RVEC TO POINT TO USER'S NORMAL RETURN ADDRESS (VIRTUAL). GOES TO GENERAL TRAP RETURN LOGIC WHICH WOULD NORMALLY RESTORE THE USER USING RVEC.

B) FUNCTION NOT POSSIBLE - E.G. NO ROOM IN DEVICE BUFFER. SET THE USER'S STATE AS APPROPRIATE (E.G. OPWAIT OR INPUT WAIT). CALL THE DOSVM SCHEDULER (COMXIT).

C) ERROR RETURN - SETS USERS P COUNTER IN RVEC TO ERROR RETURN VALUE SPECIFIED AND GOES TO GENERAL TRAP RETURN LOGIC.

SVCSW 1 = ON

SVCSW 0 = OFF



NOTE: CLASS 5 SVC'S CANNOT BE VIRTUALISED

VIRTUALISATION OF PIO INSTRUCTIONS

AS DOSVM USER RUNS IN RESTRICTED MODE, ANY I/O INSTRUCTIONS WILL GENERATE AN INTERRUPT VIA LOCATION '62.

THE VIRTUAL MACHINE CONCEPT IMPLIES THAT A PROGRAM WHICH WILL RUN UNDER DOSVM WILL ALSO RUN UNDER DOS (AND VICE-VERSA) AND WILL ALSO RUN FREE STANDING (AND VICE VERSA).

IN ORDER TO ACHIEVE THIS, DOSVM WILL HANDLE THE RESTRICTED MODE INTERRUPT, INTERPRET THE I/O INSTRUCTION GENERATING THE INTERRUPT, AND IF POSSIBLE, PERFORM THE FUNCTION REQUESTED BY THAT INSTRUCTION.

INTERPRETATION OF THE FOLLOWING INSTRUCTIONS IS PROVIDED:-

ASR, CENTRONICS PRINTER, OR ANY SERIAL PORT ON OPTION A

OCP 4, OCP 104, INA 4, INA 1004, INA 1204, INS 1304, OTA 4, OTA 104, SKS XX04.

PTR

OCP XX01, SKS XX01, INA 1, INA 1001.

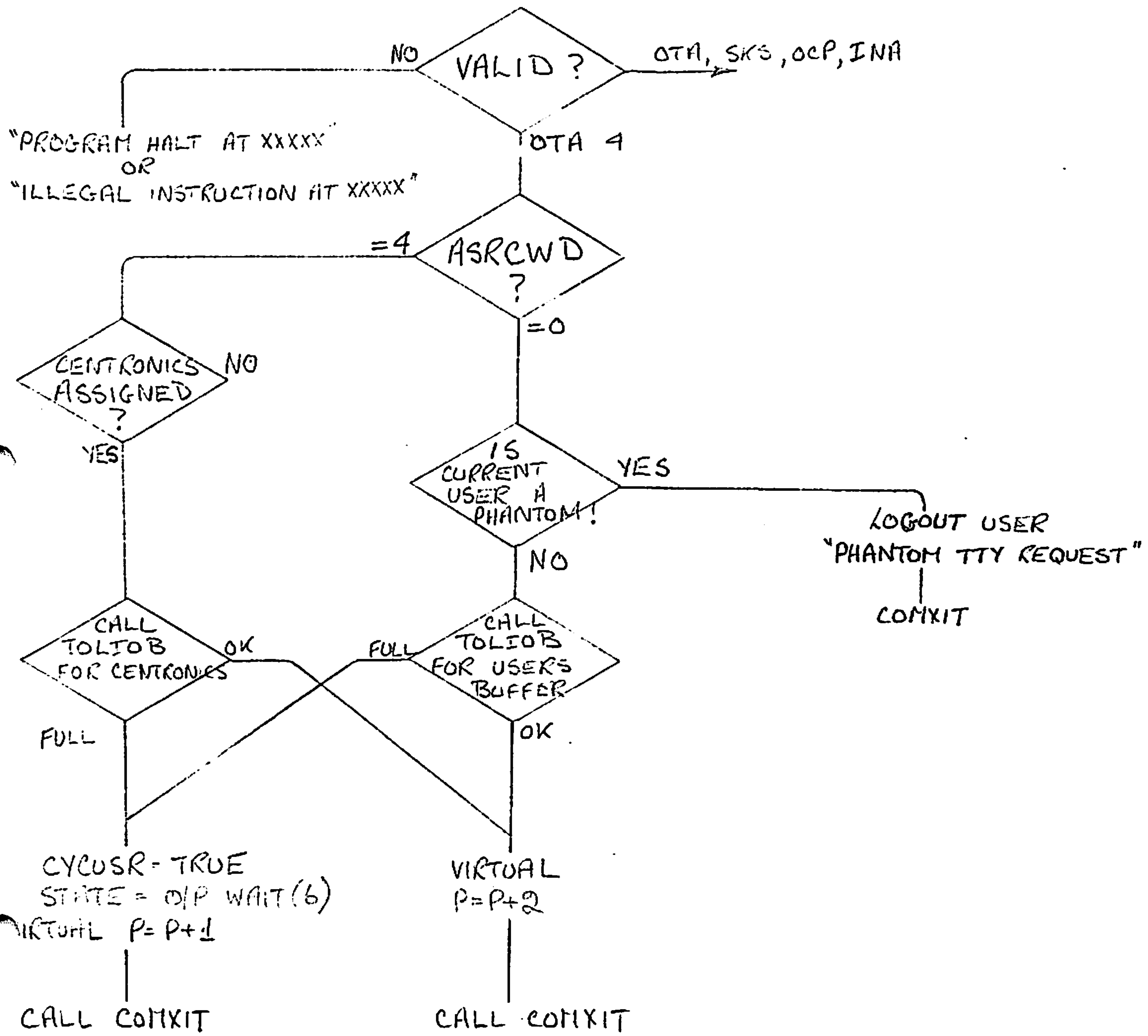
PTP

OCP XX02, SKS XX02, OTA XX02

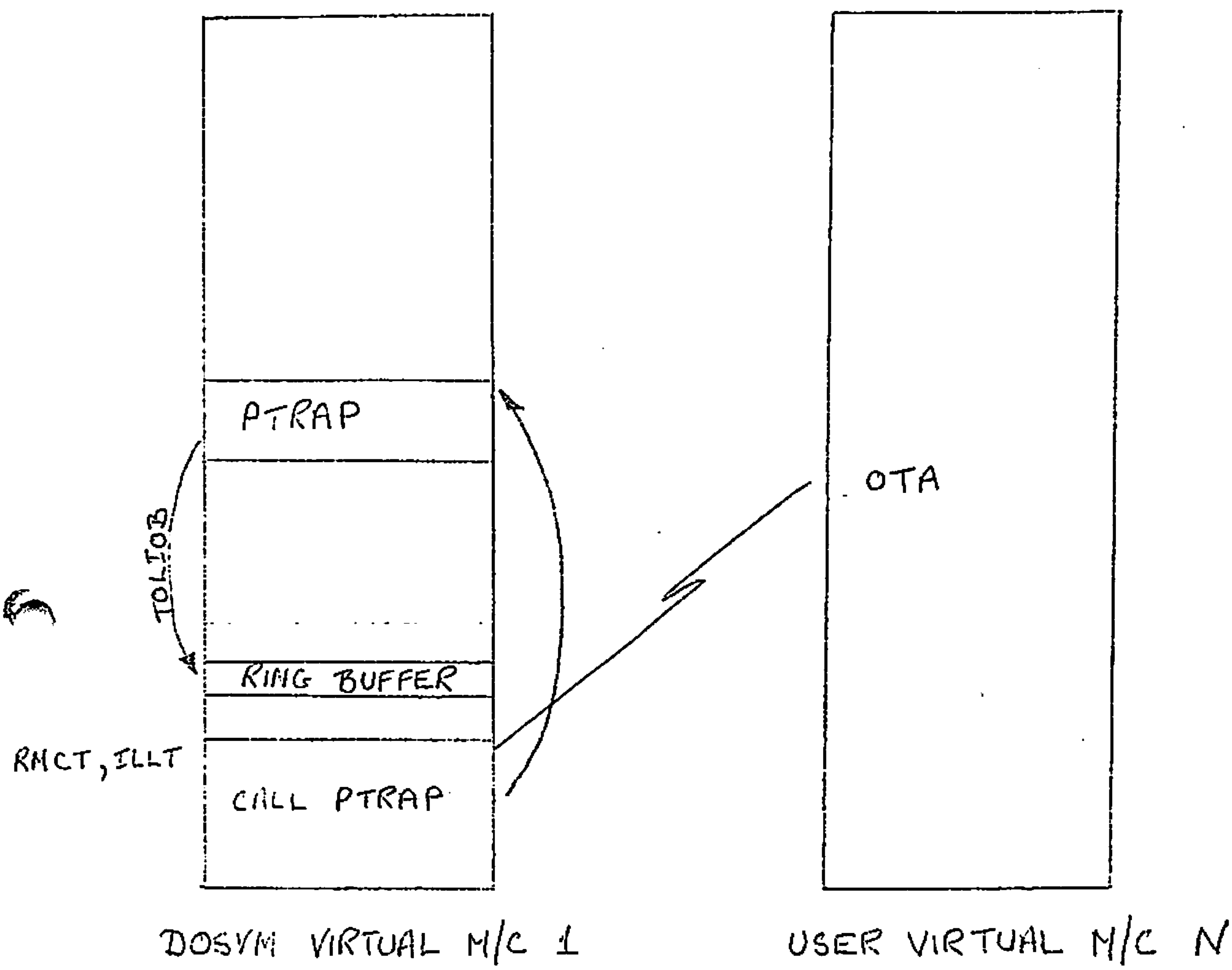
CONTROL PANEL

INA 1620, OTA 1720.

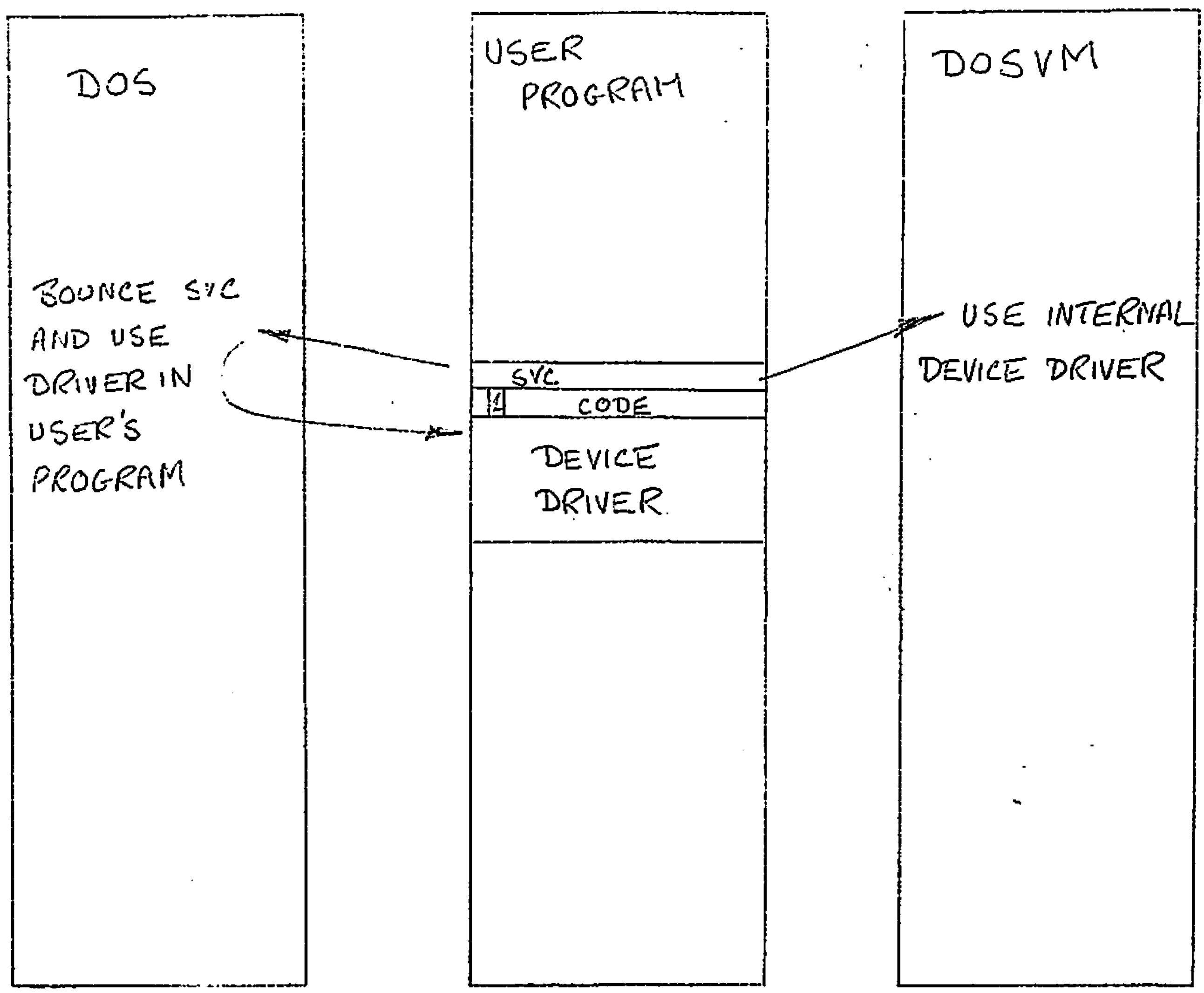
PTRAP



VALID ?
NO
OTA, SKS, OCP, INA
OTA 4
"PROGRAM HALT AT XXXXX"
OR
"ILLEGAL INSTRUCTION AT XXXXX"
ASRCWD ?
=4
=0
CENTRONICS ASSIGNED ?
NO
YES
IS CURRENT USER A PHANTOM ?
YES
NO
LOGOUT USER
"PHANTOM TTY REQUEST"
COMXIT
CALL TOLIOB FOR CENTRONICS
OK
FULL
CALL TOLIOB FOR USERS BUFFER
FULL
OK
CYCUSR = TRUE
STATE = O/P WAIT (6)
VIRTUAL P = P+1
CALL COMXIT
VIRTUAL
P=P+2
CALL COMXIT



EXAMPLE OF I/O VIRTUALISATION



VIRTUAL MACHINE CONCEPT - THE BOUNCE-BIT.

TRAP HANDLING IN DOSVM

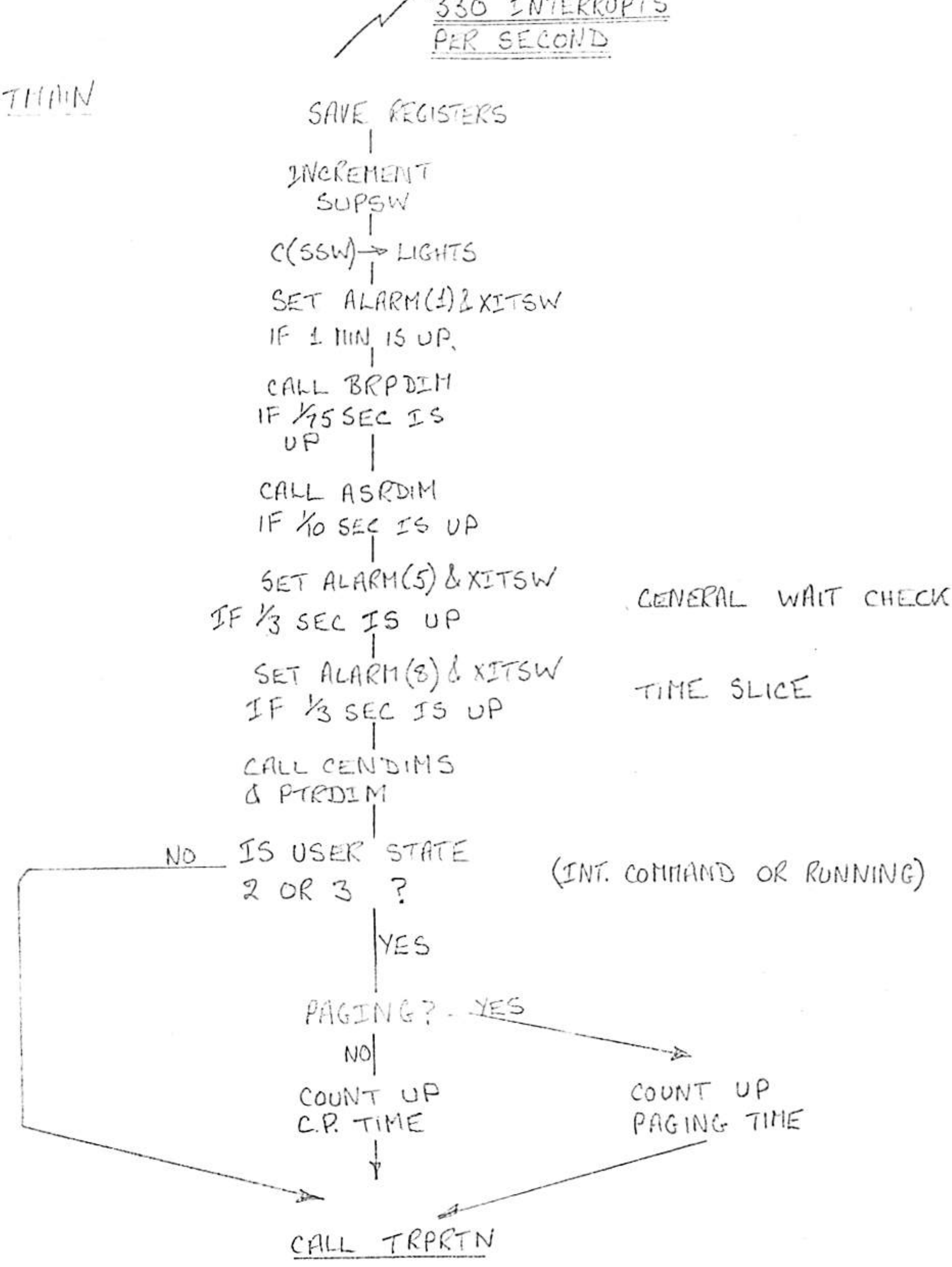
EVERY TYPE OF INTERRUPT IS ASSOCIATED WITH A BUFFER AREA IN WHICH THE MACHINE STATE OF THE INTERRUPTED PROGRAM IS STORED.

RVEC CONTAINS A COPY OF THE MACHINE STATE FOR EVERY USER WHO IS NOT ACTUALLY RUNNING.

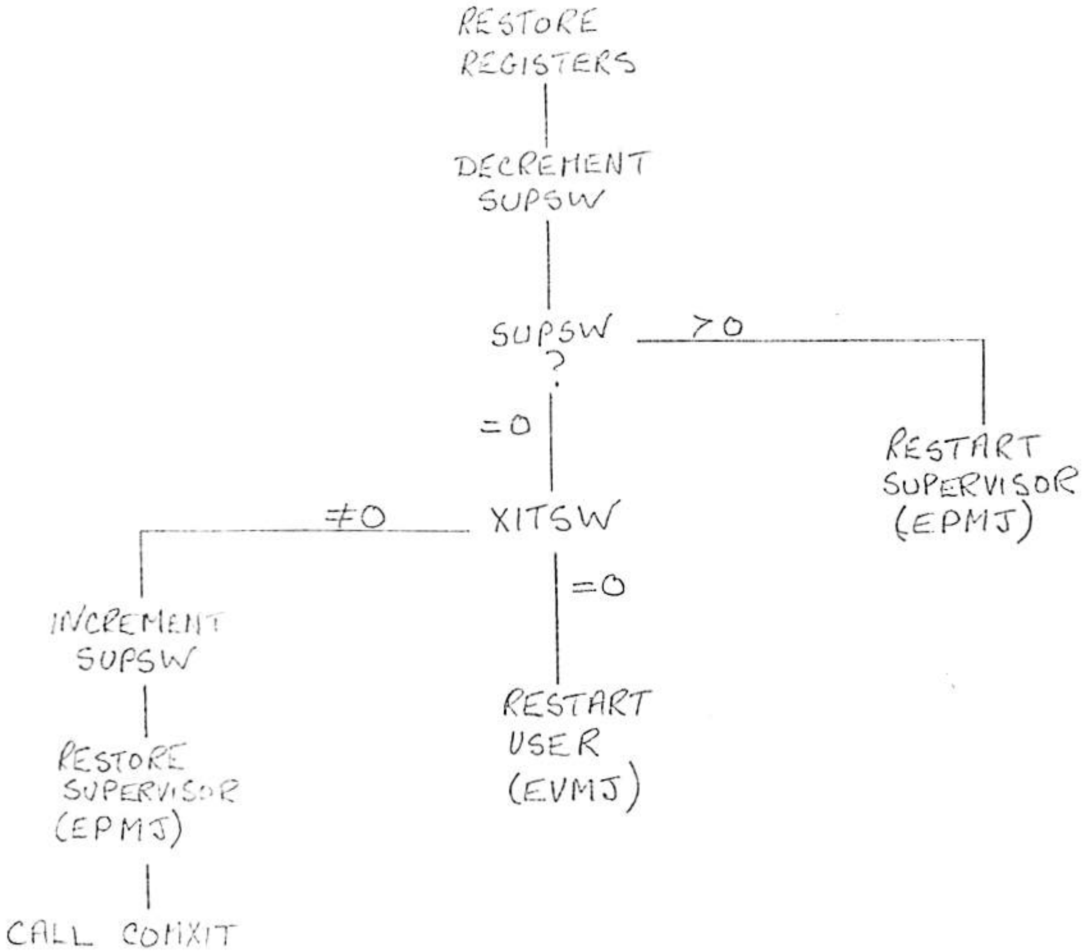
ON RETURNING FROM AN INTERRUPT, TRPRTN (TRAP RETURN) IS CALLED, GIVING THE ADDRESS OF THE MACHINE STATE BUFFER TO BE RESTORED. IF THE INTERRUPT HAS NOT CAUSED RESCHEDULING (XITSW = 0), TRPRTN WILL JUMP DIRECTLY BACK TO THE USER WITH THE MACHINE STATE SUPPLIED.

IF THE INTERRUPT HAS CAUSED RESCHEDULING, TRPRTN WILL COPY THE MACHINE STATE OF THE USER INTO THE USER'S OWN BUFFER IN RVEC AND CALL COMXIT TO GO ON TO THE NEXT USER.

BBCT IN TMAIN



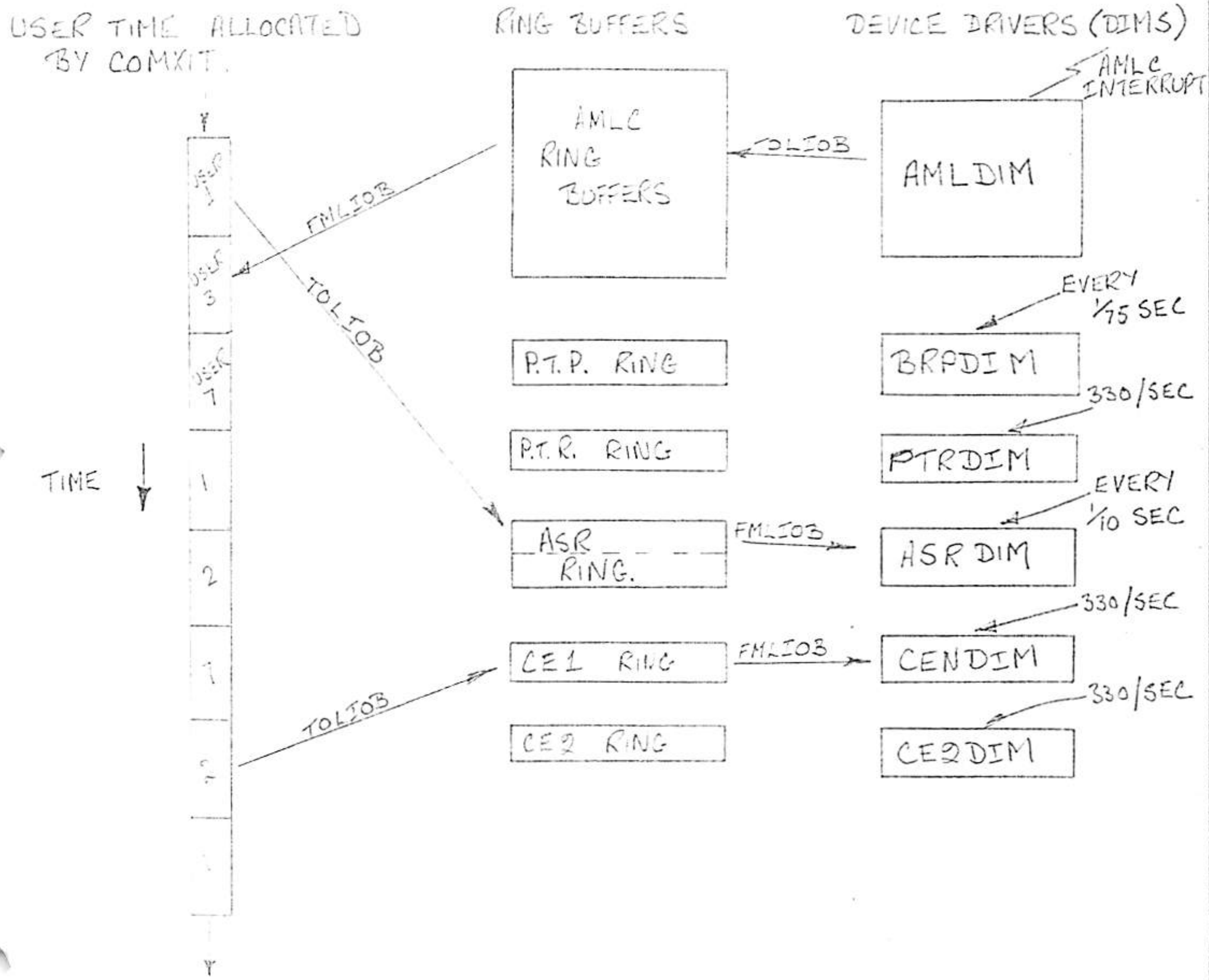
TRAP RETURN - TRPRTN



OPERATION OF LOW-SPEED CHARACTER DEVICES -

I.E. ASR, TERMINALS ON AMLC, PAPER TAPE

AND THE CENTRONICS PRINTERS



"FMLIOB" AND "TOLIOB" ARE CALLED BY THE SUPERVISOR ON BEHALF OF THE USER TO TRANSFER CHARACTER BETWEEN HIS VIRTUAL MEMORY AND THE RING BUFFER. THEY MAY BE ACTIVATED BY EITHER

A) THE USER DOING AN SVC WHICH CALLS "FMLIOB" OR "TOLIOB"

B) THE USER DOING A VALID INA OR OTA WHICH IS TRAPPED BY THE SUPERVISOR WHO CALLS FMLIOB OR TOLIOB

WHEN A USER TRIES TO DO AN OUTPUT TO A RING BUFFER WHICH IS FULL, OR AN INPUT FROM A RING BUFFER WHICH IS EMPTY, HE IS PUT IN TO A WAIT STATE, AND CONTROL OF THE C.P.U.'S IS PASSED TO THE NEXT USER BY COMXIT.

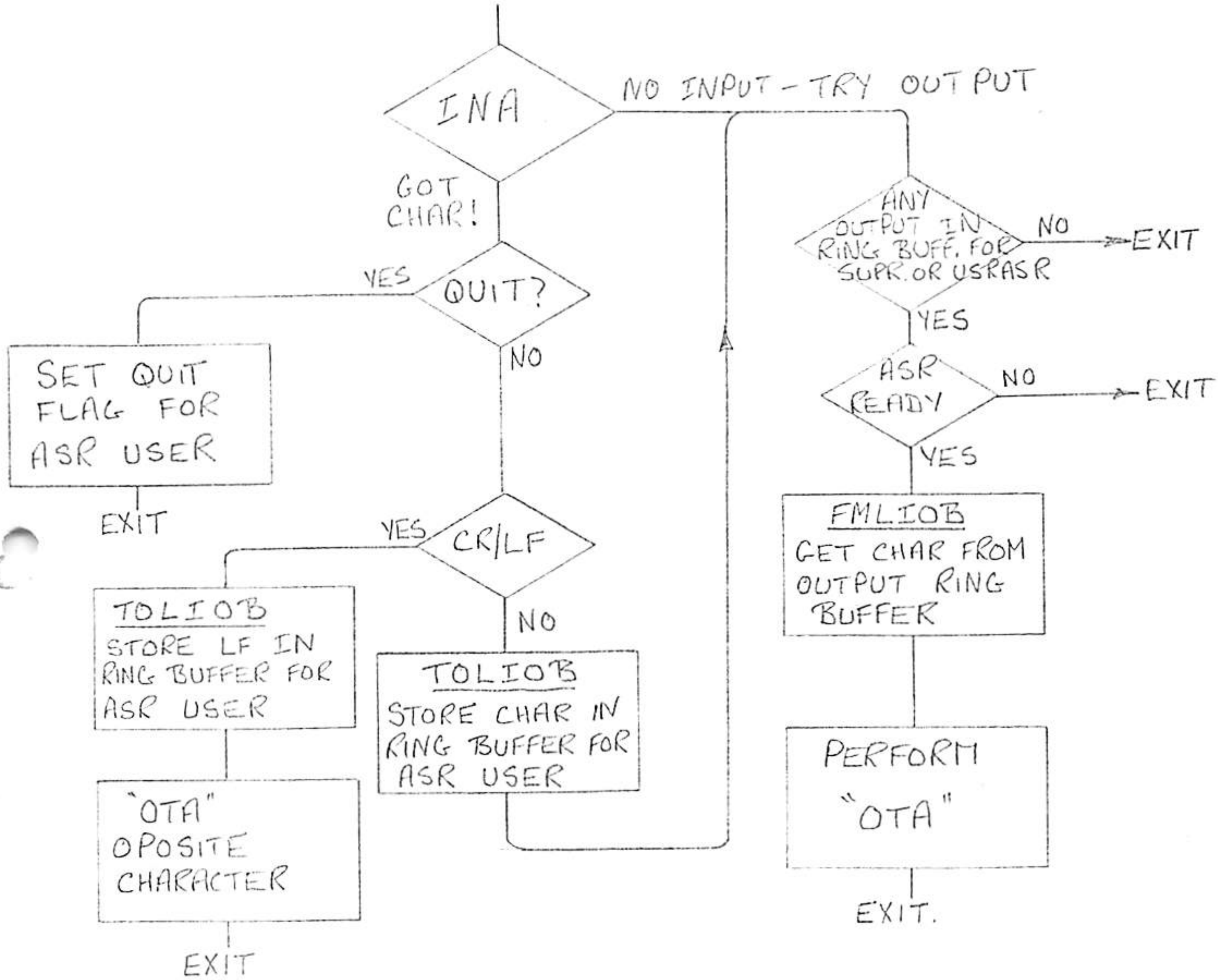
SUBROUTINE BUFCHK IS USED BY THE SUPERVISOR TO SEE IF A RING BUFFER IS FULL OR NOT.

CALLING SEQUENCE

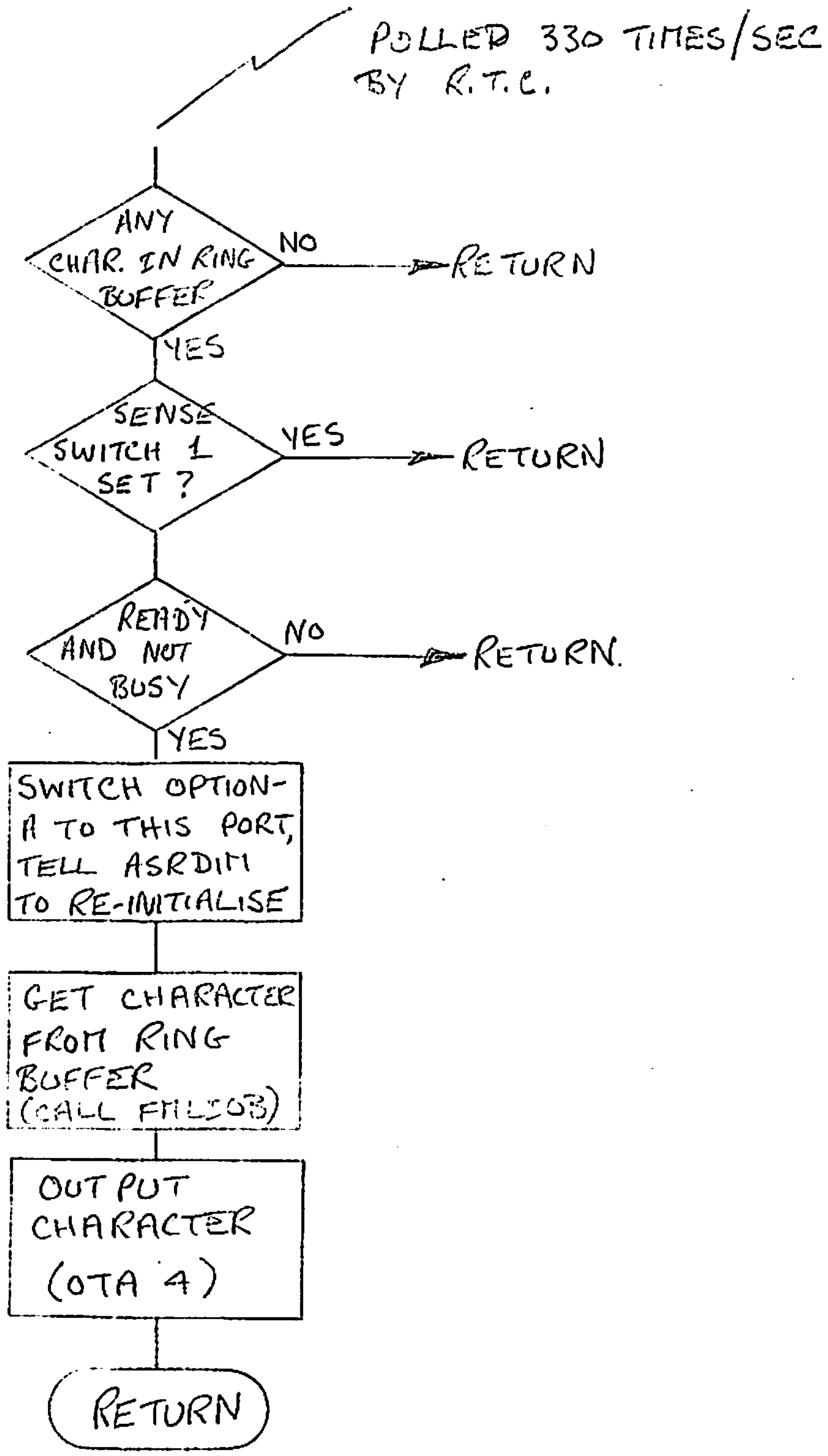
```
CALL FMLIOB(BUFFERINDEX,CHAR)RETURNS TRUE IF GOT A CHAR.
CALL TOLIOB(BUFFERINDEX,CHAR)RETURNS TRUE IF CHAR STORED.
CALL BUFCHK(BUFFER INDEX)     RETURNS TRUE IF ENOUGH ROOM
```

ASR-DRIVER - ASR DIM

POLLED 10 TIMES PER SECOND



CENTRONICS PRINTER DRIVER (CENDIM)



DOSVM AMLC DRIVER

(AMLDIM)

THE AMLC DRIVER WILL NOW CONFIGURE ITSELF TO USE EITHER AN 8 OR 16 LINE AMLC WITH A DEVICE ADDRESS OF 53 OR 54. THE 31 USER VERSION WILL MODIFY ITSELF TO USE A COMBINATION OF TWO 8 OR 16 AMLC BOARD SO LONG AS ONE HAS A DEVICE ADDRESS OF 53 AND THE OTHER AN ADDRESS OF 54. THE SYSTEM CONSOLE MAY BE USED TO RE-CONFIGURE AMLC LINES BY THE AMLC COMMAND.

AMLC (PROTOCOL) LINE (CONFIG)

PROTOCOL MAY BE	A)	TTY	- TERMINAL PROTOCOL
	B)	TTYHS	- HIGH SPEED TERMINAL
	C)	TRAN	- TRANSPARENT PROTOCOL
	D)	TRANHS	- HIGH SPEED TRANSPARENT
	E)	TTYNOP	- DO NOTHING

CONFIG IS THE LINE CONFIGURATION WORD

AMLC DRIVER TABLES

ONE WORD FOR EACH AMLC LINE

LWORD

LINE CONFIGURATION WORD

E.G.
- LINE NUMBER
- LINE SPEED
- STOP BITS
- PARITY
- CHARACTER LENGTH

IADR

- POINTER TO ADDRESS OF I/P PROTOCOL

OADR

- POINTER TO ADDRESS OF O/P PROTOCOL

a) Output Line Configuration OTA '0154

1	2 3 4	5 6 7	8 9 10	11 12 13	14 15 16

Line Number (Bit 4 is LSB)

Character Length

15	16		
0	0	-	5 bits
1	0	-	6 bits
0	1	-	7 bits
1	1	-	8 bits

Type of Parity

0 - odd parity
1 - even parity

Parity Disable

0 - Enable Parity
1 - Diable Parity

* Parity is generated on transmit and checked on receive.

Number of Stop Bits

0 - 1 stop bit
1 - 2 stop bits

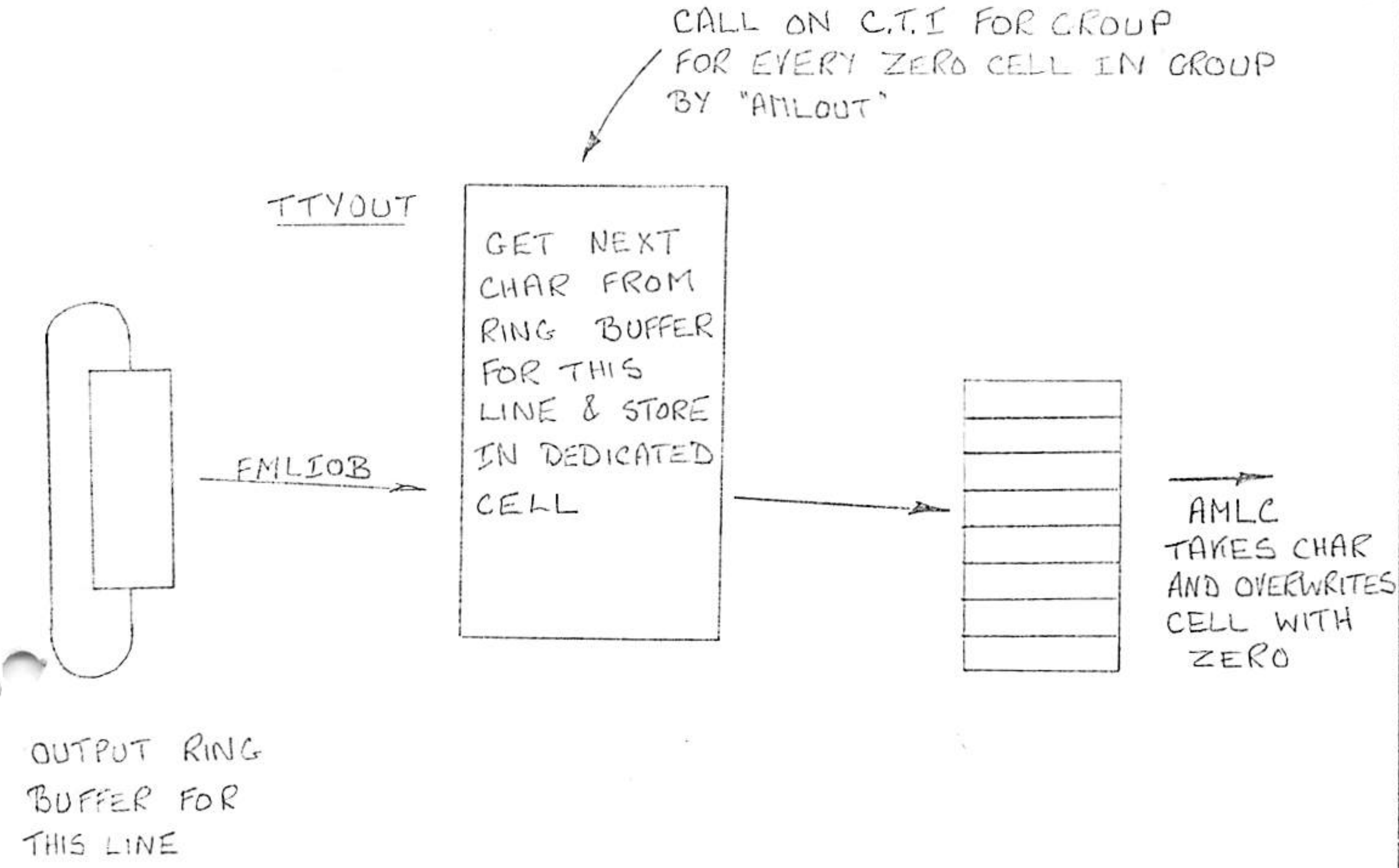
Line Speed (Data Rate)

	8	9	10		
built into controller	0	0	0	-	110 baud
	0	0	1	-	134.5 baud
	0	1	0	-	300 baud
	0	1	1	-	1200 baud
	1	0	0	-	Programmed Clock (9600)
set to any of list via jumpers on board	1	0	1	-	Assigned by user from the following 75,150,600, 1800,2400,480 9600 and 19, 2 baud (Default Selection is 101 - 75 baud, 110 - 150 baud, 111 - 1800 baud
	1	1	0	-	
	1	1	1	-	

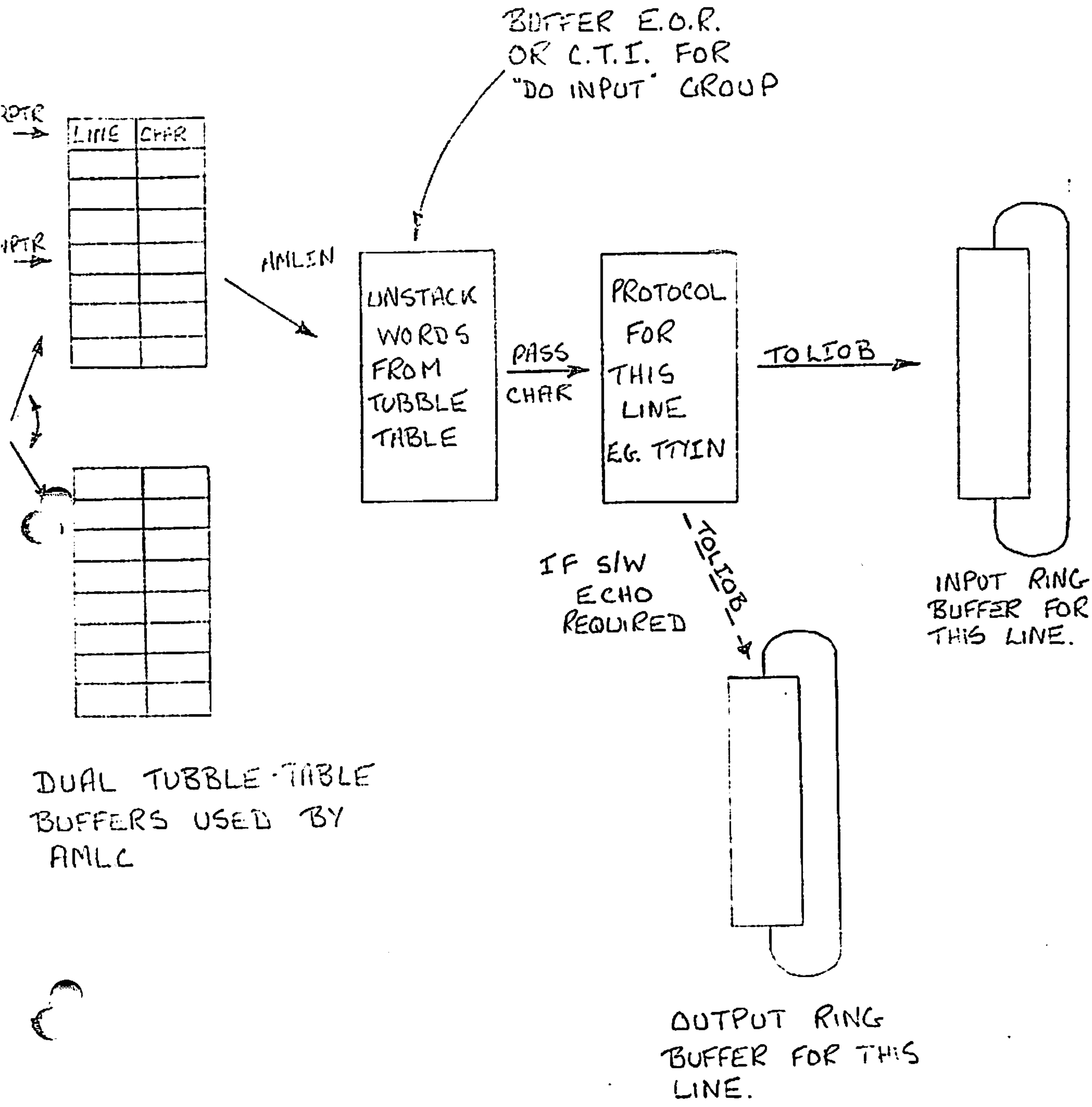
Loop Line (bit 7)

Data Set Control Bit (bit 6)

AMLC OUTPUT

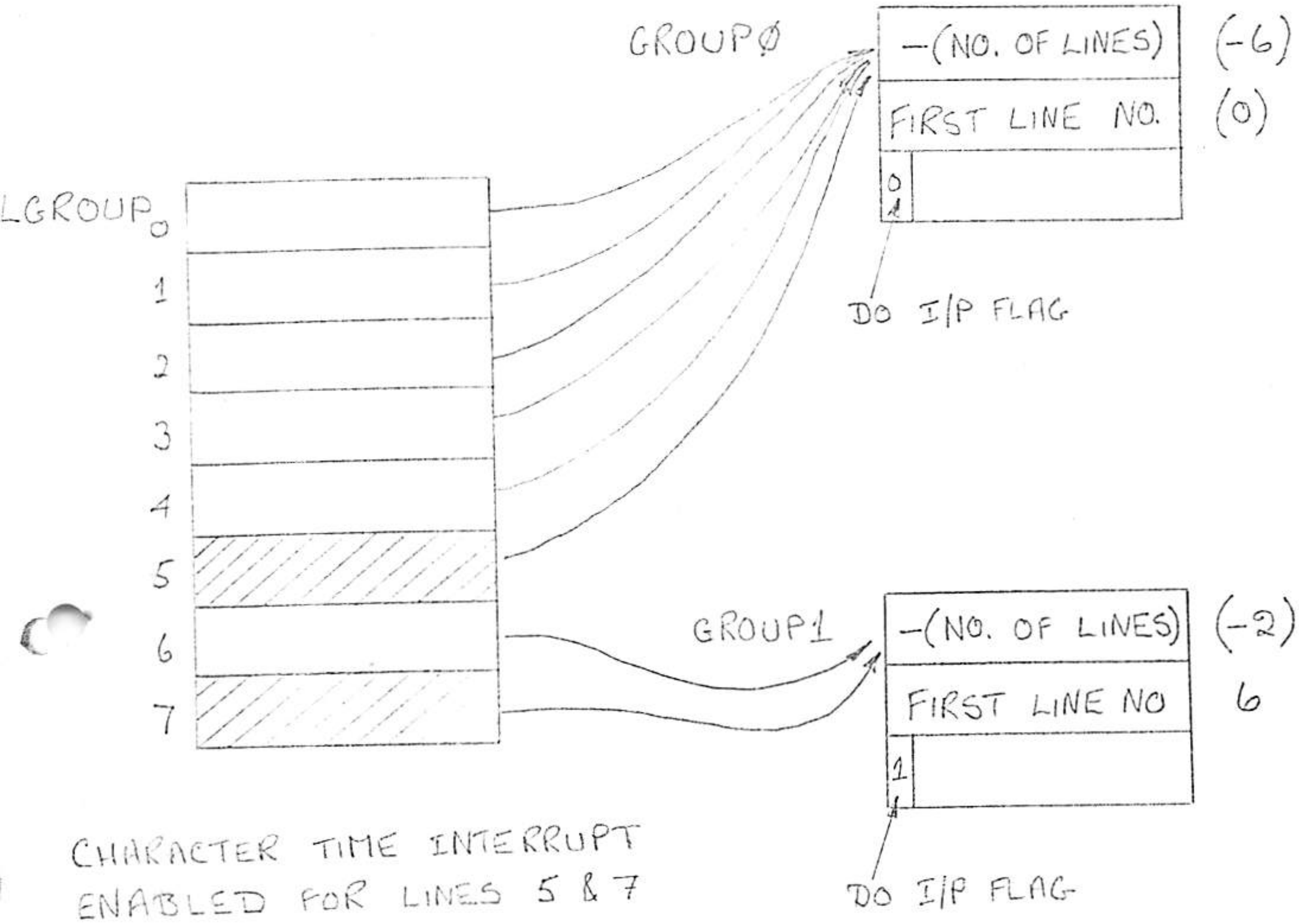

CALL ON C.T.I FOR GROUP
FOR EVERY ZERO CELL IN GROUP
BY "AMLOUT"
TTYOUT
GET NEXT CHAR FROM RING BUFFER FOR THIS LINE & STORE IN DEDICATED CELL
EMLIOB
AMLC TAKES CHAR AND OVERWRITES CELL WITH ZERO
OUTPUT RING BUFFER FOR THIS LINE

AMLC INPUT



BUFFER E.O.R. OR C.T.I. FOR "DO INPUT" GROUP
PTR
LINE
CHAR
PTR
AMLIN
UNSTACK WORDS FROM TUBBLE TABLE
PASS CHAR
PROTOCOL FOR THIS LINE E.G. TTYIN
TOLIOB
TOLIOB
IF S/W ECHO REQUIRED
INPUT RING BUFFER FOR THIS LINE.
DUAL TUBBLE TABLE BUFFERS USED BY AMLC
OUTPUT RING BUFFER FOR THIS LINE.

LINE GROUP CONFIGURATION



CHARACTER TIME INTERRUPT
ENABLED FOR LINES 5 & 7

AMLC LINE PROTOCOLS

TTY (TTYIN, TTYOUT) TERMINAL PROTOCOL

*ECHO CHARACTER IF IN FULL DUPLEX.

*FORCE BIT 8 OF CHAR. ON

*IGNORE LF

*STORE LF IN BUFFER WHEN CR RECEIVED AND ECHO CR LF

*DO NOT ECHO IF LINE BUFFER IS FULL

*CONTROL P AND BREAK CAUSE TERMINAL TO QUIT IF NOT AN ASSIGNED LINE

TRAN (TRNSIN,TRNOUT) TRANSPARENT PROTOCOL

*IGNORE STATUS AND BREAK

*NO CHARACTER MODIFICATION

*NO REFLECTION

TTYHS(TRHOUT) HIGH SPEED TERMINAL PROTOCOL

*IF MORE THAN 40 CHARS IN OUTPUT BUFFER, TURN ON CTI FOR THAT LINE, THEN USE TTYOUT PROTOCOL. IF LESS THAN 40, TURN OFF CTI.

TRANHS(TRHOUT) HIGH SPEED TRANSPARENT PROTOCOL

AS TTYHS, BUT THEN USES TRANSPARENT PROTOCOL

TTYNOP

NO ACTION

ASSIGNABLE AMLC LINES

SYSTEM CONSOLE

CONFIG NUSR PAGEDEV1 COMDEV AVALIM PAGEDEV2 NLINE

NLINE IS THE NUMBER OF ASSIGNABLE LINES

NUSR+NLINE MUST BE LESS THAN OR EQUAL

TO 16 (OR 32 FOR BIG DOSVM)

ASSIGNABLE LINES NORMALLY START AFTER THE TERMINAL

LINES.

USER TERMINAL

ASSIGN AMLC (PROTOCOL) LINE (CONFIG)

USE PROGRAM WHICH DOES:-

```
CALL T$AMLC(LINE,LOC(BUF),CCNT,KEY,STATV)

   KEY = 1 INPUT
       = 2 INPUT TILL NEW LINE
       = 3 OUTPUT
       = 4 GET NO. CHARS IN I/P BUFFER
       = 5 CHECK ENOUGH ROOM IN O/P BUFFER
```

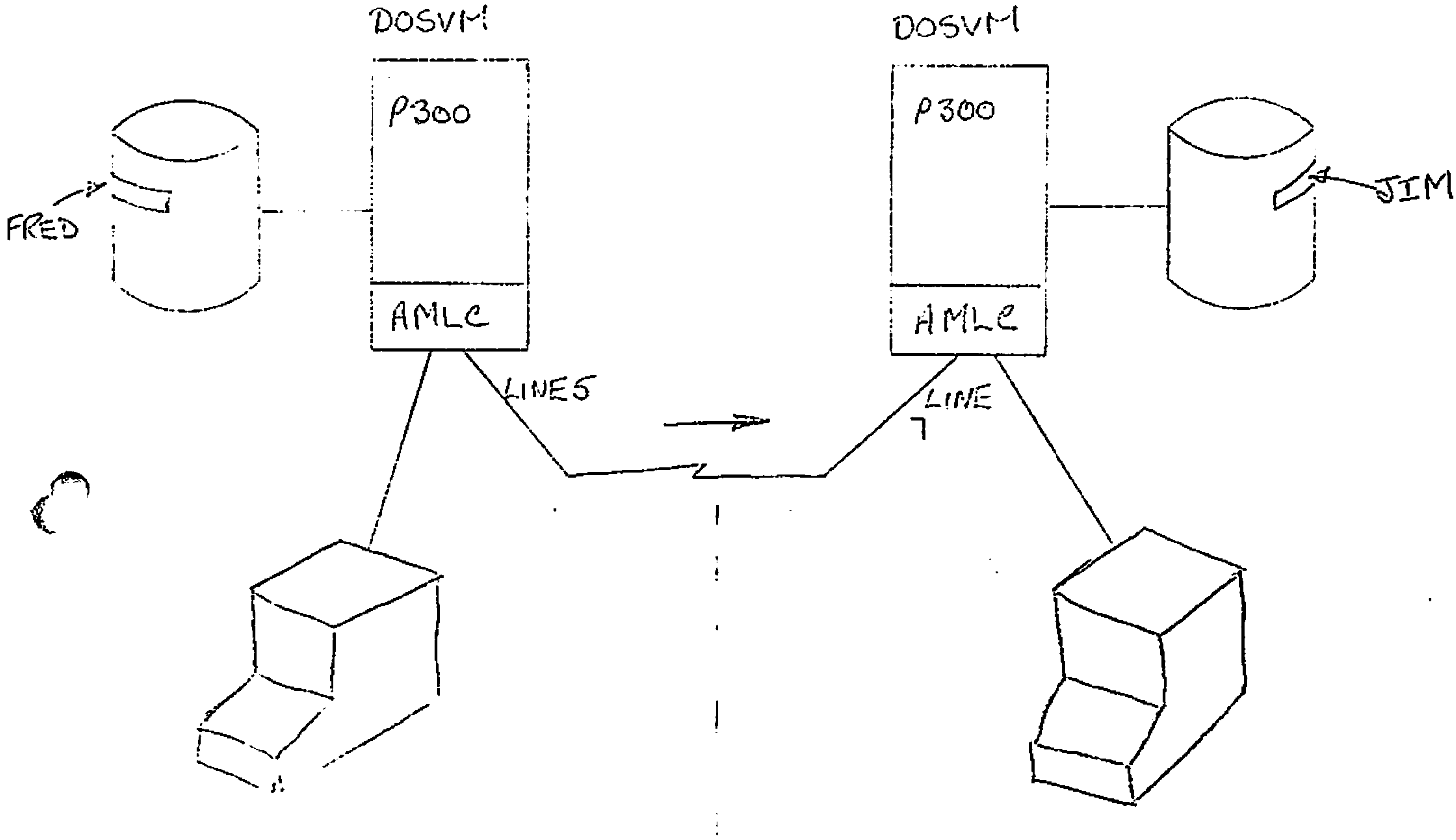
UNASSIGN AMLC LINE

NOTE

T$AMLC USES SVC 513

ASSIGNABLE LINES DO NOT INVOLVE EXTRA PAGING SPACE

COMMUNICATION BETWEEN DOSVM SYSTEMS



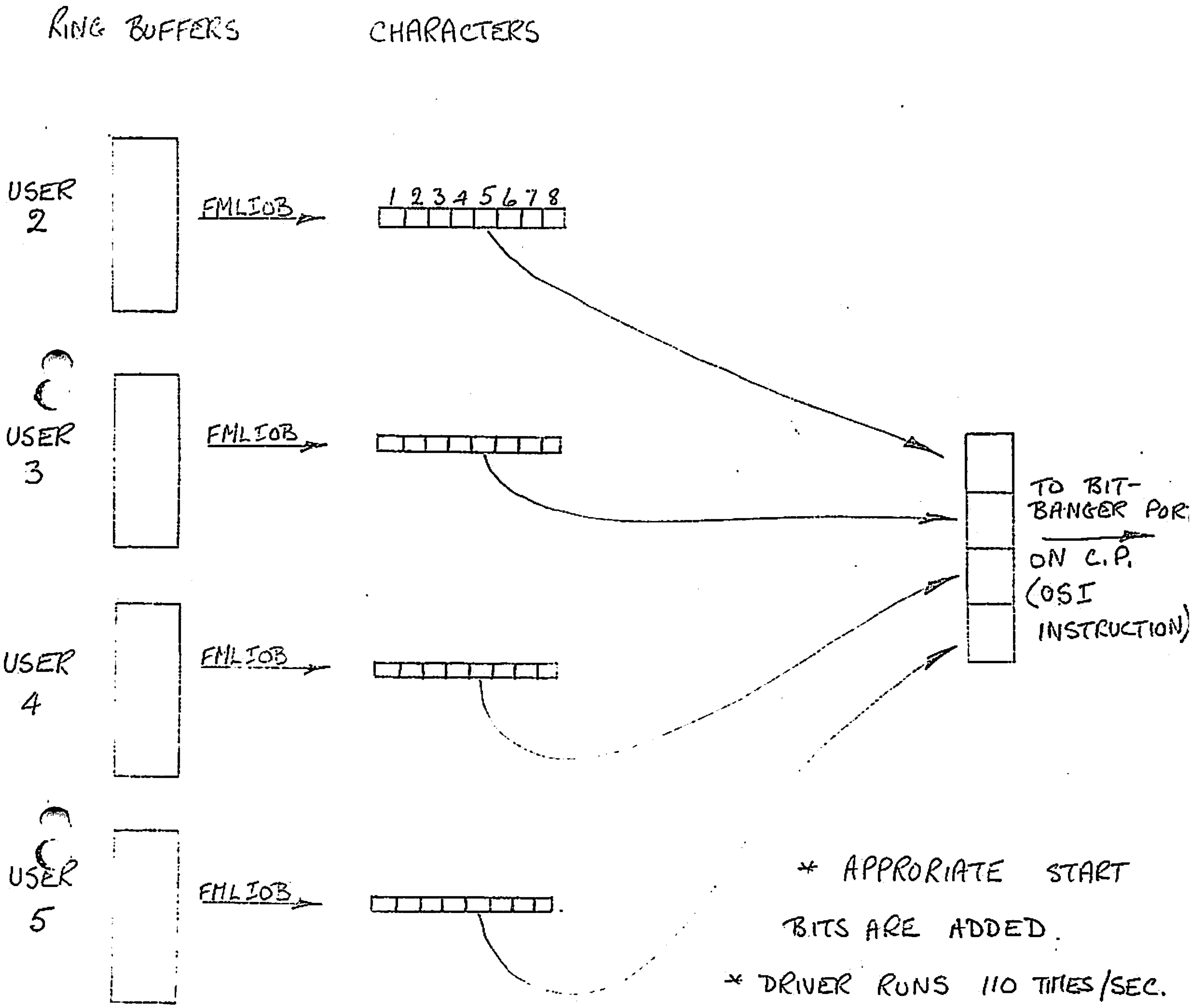
OK; AS AMLC TRANS 5	OK; AS AMLC TRANS 7
OK; TRAMLC	OK; TRAMLC
TRANSMIT FRED 5 (N)	RECEIVE JIM 7 (N)
OK;	OK;

DATA IS TRANSMITTED IN BLOCKS OF 64 WORDS

MILESTONES WILL BE OUTPUT EVERY N BLOCKS.

DOSVM BIT-BANGER DRIVER

110 BAUD VERSION



* APPRORIATE START BITS ARE ADDED.
* DRIVER RUNS 110 TIMES/SEC.

DOSVM - CLOCK VECTOR

1 -	1 MIN		UPDATE MINUTE COUNTER FOR USER, UPDATE ALL DISC BUFFERS, CHECK IF AUTO-LOGOUT REQUIRED.
2 -	1/75 SEC	-	PAPER TAPE PUNCH
3 -			NOT USED
4 -	1/10 SEC	-	ASR
5 -	1/3 SEC	-	GENERAL WAIT CHECK
6 -			ALARM VECTOR USED BY SMLC
7 -			ALARM VECTOR USED BY MAGNETIC TAPE
8 -	1/3 SEC		CYCLE CURRENT USER (TIME SLICE CLOCK)

VQUTM(8) - INITIALISED VALUE OF TIMER

VCLOCK(8)- VALUE COUNTED UP ON EVERY CLOCK INTERRUPT

VALARM(8)- FLAG (0 OR 1) USED BY COMXIT

COMXIT

IF XITSW IS TRUE, COMXIT CLEARS XITSW AND LOOKS AT ALL 8 WORDS IN THE ALARM VECTOR. IF XITSW IS FALSE, COMXIT GOES DIRECT TO THE SCHEDULER. THE FOLLOWING ACTION IS PERFORMED FOR EACH OF THE ALARMS:-

1.	-	60 SEC. ALARM - INCREMENT ELAPSED TIME AND DATE. UPDATE ALL ASSOCIATED DISC BUFFERS TO DISC. TICKLE MPC.
2,3 AND 4	-	ASR AND PTP - DO NOTHING.
5	-	1/3 SEC GENERAL WAIT CHECK - ALL USERS IN OUTPUT WAIT STATE ARE GIVEN A STATE OF "RUNNING". TICKLE GOULD PLOTTER.
6 - SMLC	-	CALL SMLCEX
7 - MT	-	CALL MTDONE
8	-	1/3 CYCLE CURRENT USER - SET CYCUSR TO TRUE

THEN WE GO TO THE SCHEDULER

DOSVM SCHEDULER - (PART OF COMXIT)

DECIDES WHICH SHOULD BE THE NEXT USER TO RUN AFTER THE CURRENT USER HAS COME TO THE END OF HIS TIME SLICE.

STARTS BY LOOKING FOR A USER IN ANY INPUT WAIT STATE (1, 5 OR 9).

STATE 1 - COMMAND I/P WAIT - CALL COMANL TO READ COMMAND FROM DEVICE OR FILE.

STATE 5 - INPUT WAIT - IF CHARACTERS ARE AVAILABLE FROM THE INPUT BUFFER, THE USER IS SET IN THE RUNNING STATE AND RE-STARTED.

STATE 9 - SUPERVISOR INPUT WAIT - SUPERVISOR PROCEDURE IS RE-STARTED.

IF NONE OF THESE STATES ARE FOUND, A NEW USER IS CHOSEN ON A RING BASIS, AND THE ACTION DEPENDS ON THE STATE OF THAT USER:-

USER STATE VARIABLE

1 - COMMAND I/P WAIT
2 - DOSSUB
3 - RUNNING
4 - DISABLED
5 - I/P WAIT
6 - GENERAL WAIT
7 - COMANL I/P WAIT
8 - SUPERVISOR WAIT
9 - SUPERVISOR I/P WAIT
10 - DISK I/O WAIT
11 - SUPERVISOR LOCKED
12 - FLEX TRAP WAIT
13 - NETWORK SYNC I/O
14 - GENERAL NETWORK WAIT
15 - FAM WAIT

DOSVM USER STATE VARIABLE

1 - COMMAND INPUT WAIT

2 - THE USER IS CURRENTLY RUNNING AN INTERNAL COMMAND IN THE SUPERVISOR (DOSSUB).

3 - RUNNING - I.E. THE USER IS CURRENTLY USING TIME ON THE C.P.U.

4 - NOT USED.

5 - INPUT WAIT. THE USER IS WAITING FOR INPUT FROM A DEVICE.

6 - GENERAL WAIT. (O/P) WAIT).

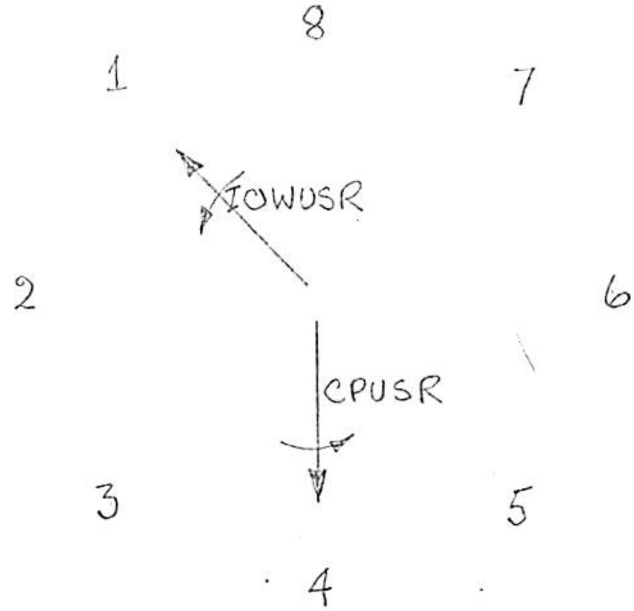
7 - COMANL INPUT WAIT

8 - SUPERVISER WAIT STATE

9 - SUPERVISOR INPUT WAIT STATE

CLOCK VECTORS & ALARMS USED BY PRIMOS

NO.	CLOCK	ALARM	COMMENTS
1	1 MIN	✓	UPDATE "TIME NOW". UPDATE ALL DISC BUFFERS. CHECK AUTO-LOGOUT CLOCK & LOGOUT IDLE USER IF NECESSARY. CANN MPCXEC
2	1/75 SEC		CALL BRPDIM IN TMAIN. NO ACTION IN COMXIT
3		✓	AMLC DROPPED CARRIER. FORCE LOGOUT LINE IF DLOGOT ≠ 0.
4	1/10 SEC		CALLS ASRDIM IN TMAIN.
5	1/3 SEC		GENERAL WAIT CHECK. RESET ANY USER IN O/P WAIT TO A "RUNNING" STATE. ALSO TICKLE THE GOULD PLOTTER!
6		✓	CALL SMLCEX - SMLC SERVICE ROUTINE
7		✓	CALL MTDONE - MAG TAPE CLEAN-UP ROUTINE.
8	1/3 SEC		TIME-SLICE CLOCK - CYCLE CURRENT USERS - SET CYCUSR - TRUE
9			USED BY NETWORK. PROCESS ALL NETWORK REQUESTS
10			USED BY FAM. IF FAM USER IS IN STATE 15 - CAUSE IT TO BE RUN



SCAN IOWUSR FIRST

-LOOK FOR ANY I/P WAIT STATE.

THEN SCAN CPUSR

- LOOK FOR ALL STATES & TAKE APPROPRIATE ACTION.

ACTIONS FOR CPUSR

COMMAND I/P WAIT - O/P MESSAGE IF ONE IS WAITING, LOOK FOR NEXT USER

DOSSUB - CALL DOSSUB (DOSSUB MAY CHANGE THE STATE & RECALL COMXIT, OR IF IT RETURNS, COMXIT SETS STATE TO C/I WAIT & CALL COMANL AGAIN.

RUNNING - CALL TRPRTN TO RUN USER

COMANL - CALL COMANL FOR USER & RUN IF COMMAND COMPLETE.

SUPERVISOR WAIT (8,9) - JUMP TO LOCATION IN DOSVM.

ACTIONS FOR CPUSR

COMMAND I/P WAIT - O/P MESSAGE IF ONE IS WAITING, LOOK FOR NEXT USER

DOSSUB - CALL DOSSUB (DOSSUB MAY CHANGE THE STATE & RECALL COMXIT, OR IF IT RETURNS, COMXIT SETS STATE TO C/I WAIT & CALL COMANL AGAIN.

RUNNING - CALL TRPRTN TO RUN USER

COMANL - CALL COMANL FOR USER & RUN IF COMMAND COMPLETE.

SUPERVISOR WAIT (8,9) - JUMP TO LOCATION IN DOSVM.

DOSVM INTERNAL COMMANDS

ALL INTERNAL COMMANDS HANDLED BY SUBROUTINE DOSSUB.

DOSSUB IS CALLED BY COMXIT WHEN STARTING A USER IN COMMAND INPUT WAIT STATE. USER STATE IN DOSSUB IS ALWAYS 2.

PRIMOS INITIALISATION - REV 11

SUBROUTINE AINIT (VERSION, MEMORY SIZE, LSVEC)

CALLED AFTER CONFIG COMMAND HAS BEEN READ

* CLEAR QUIT FLAG FOR ALL USERS
* TYPE CHEERY MESSAGE "PRIMOS III <REV 11.15>

  XX.XK MEMORY IN USE
  PLEASE ENTER DATE
* READ COMMAND USING DOS CMREAD
* IF COMDEV AND PAGDEV ARE EQUAL, SET UP PAGREL
* LOCK PAGE MAPS FOR NUMBER OF USERS CONFIGURED.
* LOCK SECTOR 0
* LOCK CODE BETWEEN MMAP AND LOCKIT
* IF CONFIG COMMAND REFERS TO VALID MODE NAME (SYSA OR SYSB), LOCK CODE BETWEEN LOCKFA AND LOCKTA. OTHERWISE DISABLE NETWORK.
* LOCK TTY INPUT AND OUTPUT BUFFERS.
* LOCK OTHER LOW SPEED BUFFERS.
* LOCK 1ST ASSOCIATIVE BUFFER.
* CLEAR USRCOM FOR SYSTEM
* ATTACH SYSTEM TO UFD "CMDNCO", USING A BLANK PASSWORD
* LOCK SMLC CODE BETWEEN SLCINI AND SLCTOP IF SMLC IS CONFIGURED IN BY "CONFIG" COMMAND.
* ALLOCATE ASSIGNABLE LINES.
* OPEN & SKIP FIRST LINE OF COMMAND FILE "C←PRMO"

CONFIG TRMUSR PAGDEV COMDEV AVALM PAGDEV2 NLINE PH

- ERROR IF COMMAND NOT GIVEN BY SYSTEM TERMINAL
- SET "CPTE" TO POINT TO END OF AVAILABLE MEMORY
- CHECKS (TRMUSR + NLINE) < 16 (OR 32)
- CHECKS (TRMUSR + PH) < 16 (OR 32)
- SETS ALL NON-TERMINAL USERS IN STATE 4.
- LOCKS PAGE MAPS OF TERMINAL USERS AND PHANTOM USERS. (INCLUDES SECTOR 1).
- LOCKS SECTOR 0.
- LOCKS DOSVM CODE BETWEEN "MMAP" AND "LOCKIT".
- LOCKS RING BUFFERS FOR TERMINALS AND ASSIGNED AMLC LINES.
- LOCKS RING BUFFERS FOR OTHER LOW SPEED DEVICES (PTR/P, CENTRONICS).
- REDUCE NUMBER OF ASSOCIATIVE BUFFERS FROM 48 TO 32 IF LESS THAN 16 USERS.
- LOCKS THE FIRST ASSOCIATIVE BUFFER IN MEMORY
- STARTS-UP THE COMMAND DEVICE (CALL TRWRAT)
- ATTACH THE SYSTEM TO CMDNCO (CALL ATTACH).
- CLEARS USRCOM FOR USERS.
- OUTPUTS "LOGIN PLEASE" ON ALL TERMINALS.

RESTORE

CALLS DOSVM SUBROUTINE "RESTOR".

READ THE "SAVE FILE VECTOR" INTO RVEC FOR CURRENT USER. MARKS THE USER'S PAGE MAP WITH "NO COPY ON DISC" FOR PAGES RESTORED PAGES.

READS SAVED FILE INTO SPECIFIED AREA OF USER'S VIRTUAL MEMORY.

OUTPUTS "OK", ON TERMINAL AND RETURNS FROM DOSSUB.

RESUME

AS FOR RESTORE, BUT INSTEAD OF OUTPUTING "OK" RETURNS FROM DOSSUB, SETS THE USER STATE TO 3 (RUNNING) AND CALLS COMXIT.

ASSIGN

SCAN VALID DEVICE NAME TABLE (DEVNAM) TO FIND DEVICE NAME SUPPLIED.

GIVE ERROR MESSAGE IF "DEVUSR" FOR DEVICE FOUND IS NOT ZERO OR EQUAL TO THE CURRENT USER ("CUSR").

SET DEVUSR FOR THIS DEVICE EQUAL TO CURRENT USER.

CALL "DEVONF" FOR THIS DEVICE TO PERFORM THE ASSIGN FUNCTION PARTICULAR TO THIS DEVICE - E.G. TO SET A FLAG TO BE CHECKED BY THE DEVICE DRIVER.

AMLC AND DISK ARE SPECIAL CASES FOR THE ASSIGN COMMAND.

COMINP (FILE NAME)

CALLS SUBROUTINE COMINP

COMINP OPENS THE COMMAND FILE ON THE SPECIFIED FILE UNIT (IF NOT SPECIFIED = 6) AND SETS COMSWI (CUSR) AND COMUNI (CUSR).

START

SETS UP RVEC FOR THE CURRENT USER TO EQUAL THE OCTAL PARAMETERS SUPPLIED. SETS THE USER STATE TO 3 (RUNNING) AND CALLS COMXIT.

CONFIG TRMUSR PAGDEV COMDEV AVALM PAGDEV2 NLINE PH

- ERROR IF COMMAND NOT GIVEN BY SYSTEM TERMINAL
- SET "CPTE" TO POINT TO END OF AVAILABLE MEMORY
- CHECKS (TRMUSR + NLINE) 16 (OR 32)
- CHECKS (TRMUSR + PH) 16 (OR 32)
- SETS ALL NON TERMINAL USERS IN STATE 4.
- LOCKS PAGE MAPS OF TERMINAL USERS AND PHANTOM USERS. (INCLUDES SECTOR 1).
- LOCKS SECTOR 0.
- LOCKS DOSVM CODE BETWEEN "MMAP" AND "LOCKIT".
- LOCKS RING BUFFERS FOR TERMINALS AND ASSIGNED AMLC LINES.
- LOCKS RING BUFFERS FOR OTHER LOW SPEED DEVICES (PTR/P, CENTRONICS).
- REDUCE NUMBER OF ASSOCIATIVE BUFFERS FROM 48 TO 32 IF LESS THAN 16 USERS.
- LOCKS THE FIRST ASSOCIATIVE BUFFER IN MEMORY
- STARTS-UP THE COMMAND DEVICE (CALL TRWRAT)
- ATTACH THE SYSTEM TO CMDNCO (CALL ATTACH).
- CLEARS USRCOM FOR USERS.
- OUTPUTS "LOGIN PLEASE" ON ALL TERMINALS.

RESTORE

CALLS DOSVM SUBROUTINE "RESTOR".

READ THE "SAVE FILE VECTOR" INTO RVEC FOR CURRENT USER. MARKS THE USER'S PAGE MAP WITH "NO COPY ON DISC" FOR PAGES RESTORED PAGES.

READS SAVED FILE INTO SPECIFIED AREA OF USER'S VIRTUAL MEMORY.

OUTPUTS "OK", ON TERMINAL AND RETURNS FROM DOSSUB.

RESUME

AS FOR RESTORE, BUT INSTEAD OF OUTPUTING "OK" RETURNS FROM DOSSUB, SETS THE USER STATE TO 3 (RUNNING) AND CALLS COMXIT.

ASSIGN

SCAN VALID DEVICE NAME TABLE (DEVNAM) TO FIND DEVICE NAME SUPPLIED.

GIVE ERROR MESSAGE IF "DEVUSR" FOR DEVICE FOUND IS NOT ZERO OR EQUAL TO THE CURRENT USER ("CUSR").

SET DEVUSR FOR THIS DEVICE EQUAL TO CURRENT USER.

CALL "DEVONF" FOR THIS DEVICE TO PERFORM THE ASSIGN FUNCTION PARTICULAR TO THIS DEVICE - E.G. TO SET A FLAG TO BE CHECKED BY THE DEVICE DRIVER.

AMLC AND DISK ARE SPECIAL CASES FOR THE ASSIGN COMMAND.

COMINP (FILE NAME)

CALLS SUBROUTINE COMINP

COMINP OPENS THE COMMAND FILE ON THE SPECIFIED FILE UNIT (IF NOT SPECIFIED = 6) AND SETS COMSWI (CUSR) AND COMUNI (CUSR).

START

SETS UP RVEC FOR THE CURRENT USER TO EQUAL THE OCTAL PARAMETERS SUPPLIED. SETS THE USER STATE TO 3 (RUNNING) AND CALLS COMXIT.

DOSVM INTERNAL COMMANDS

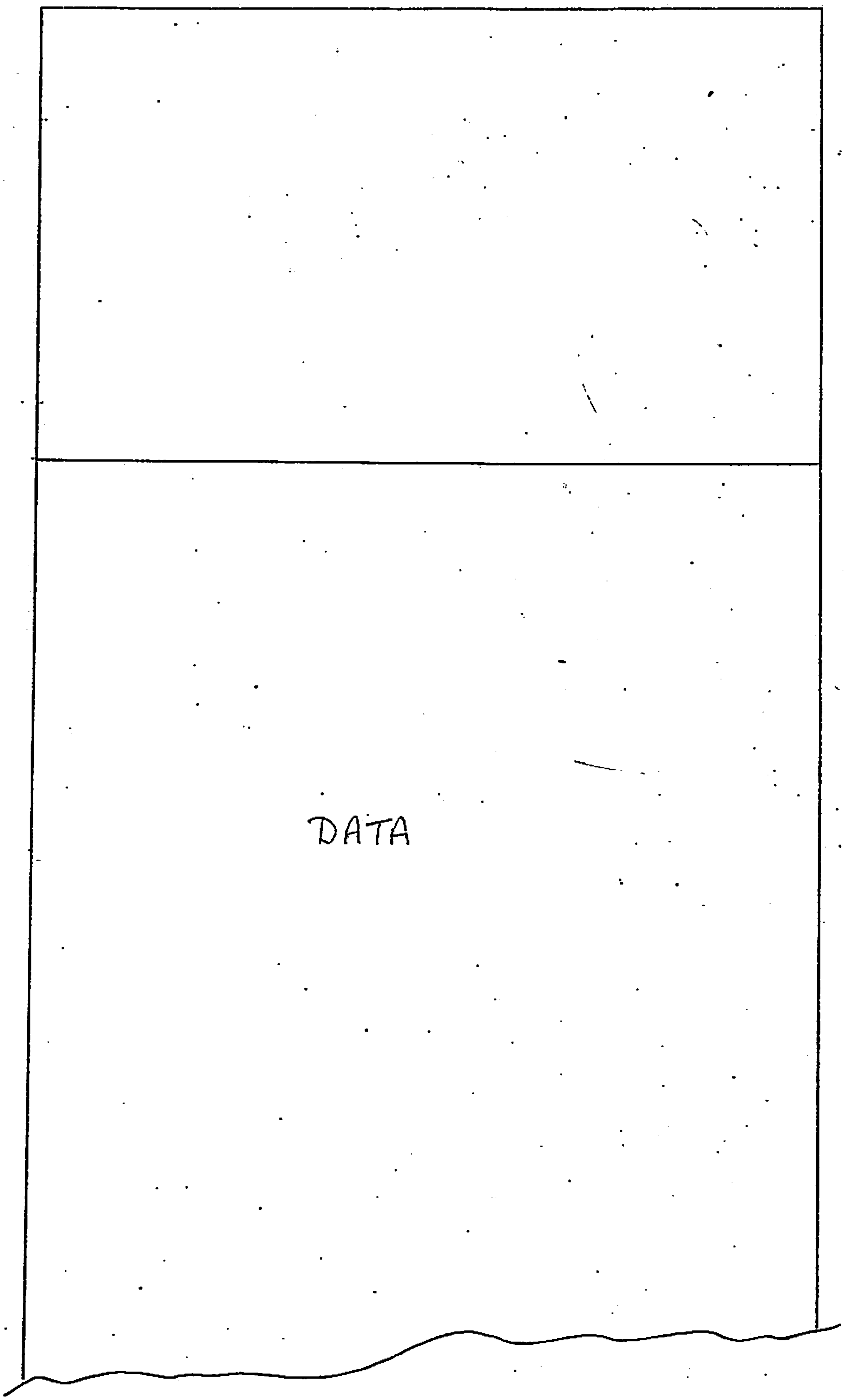
ALL INTERNAL COMMANDS HANDLED BY SUBROUTINE DOSSUB.

DOSSUB IS CALLED BY COMXIT WHEN STARTING A USER IN COMMAND INPUT WAIT STATE. USER STATE IN DOSSUB IS ALWAYS 2.

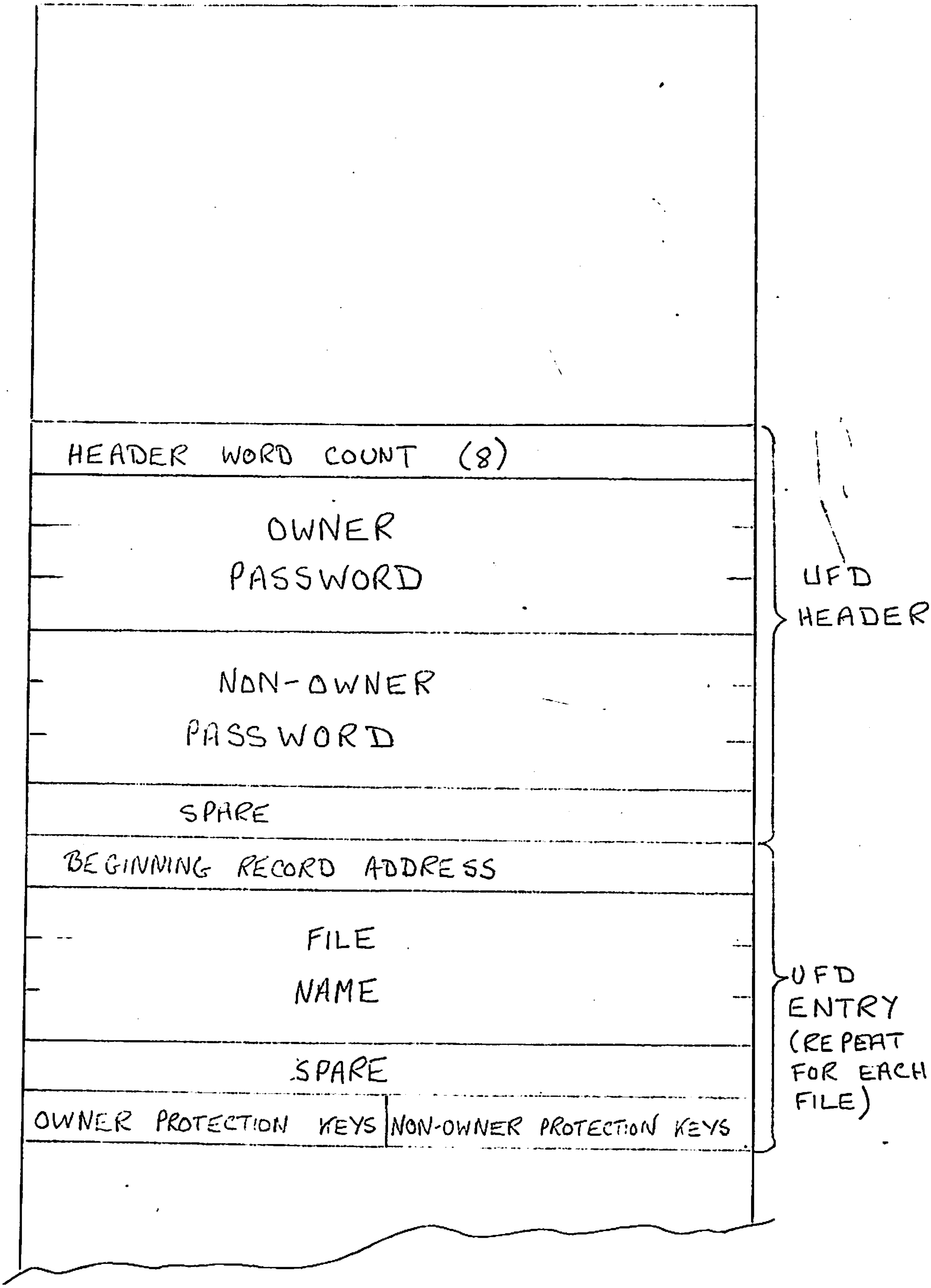
DISC RECORD FORMAT. -

REKCRA - THIS RECORD ADDRESS
REKPOP - FATHER RECORD ADDRESS (UFD OR SD)
REKFPT - FORWARD POINTER
REKBPT - BACKWARD POINTER
REKDCT - DATA COUNT IN RECORD
REKTYP - RECORD TYPE

DATA



UFD


HEADER WORD COUNT (8)
OWNER PASSWORD
NON-OWNER PASSWORD
SPARE
BEGINNING RECORD ADDRESS
FILE NAME
SPARE
OWNER PROTECTION KEYS
NON-OWNER PROTECTION KEYS
UFD HEADER
UFD ENTRY (REPEAT FOR EACH FILE)

- DSKRAT.

DSKRAT HEADER:

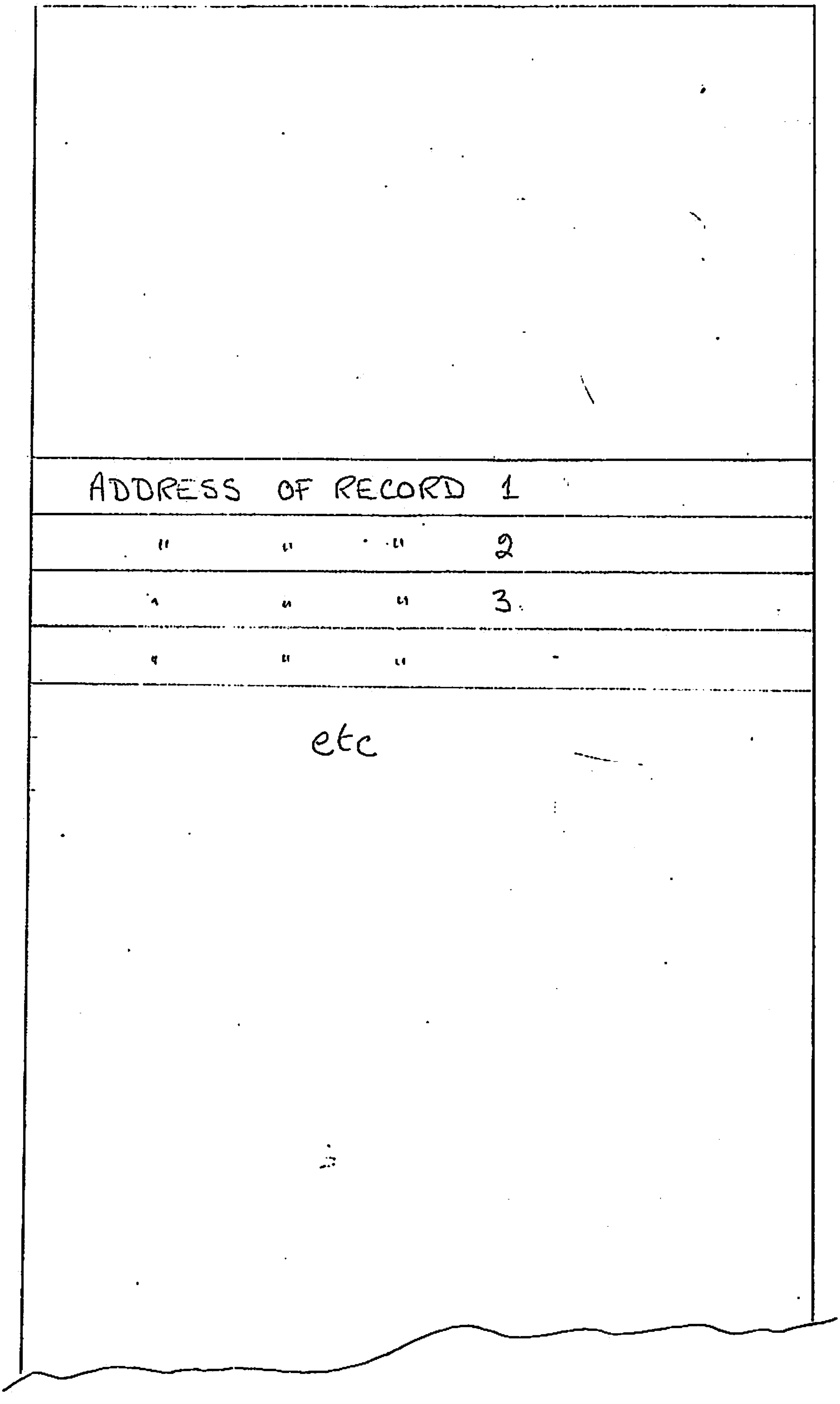
WORDS IN HEADER (5)
DISC RECORD SIZE (NORMALLY 448)
NUMBER OF RECORDS FOR FILE SYSTEM
CYLINDER COUNT
NUMBER OF HEADS IN PARTITION

NREC/16 WORDS:

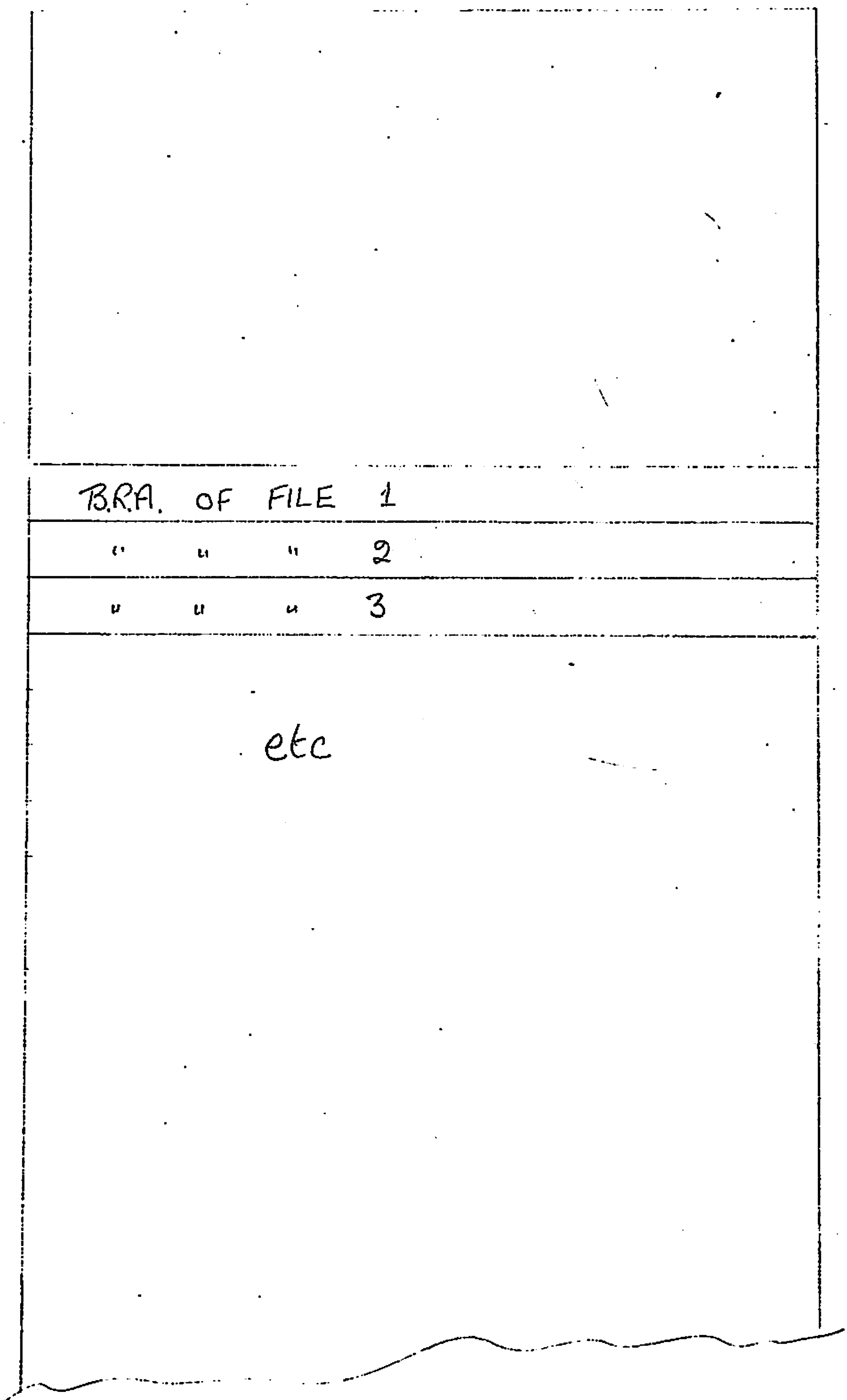
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	RECORDS 0-15
																RECORDS 16-31

0 = USED
1 = FREE

RECORD Ø OF DAM FILE



SEGMENT DIRECTORY

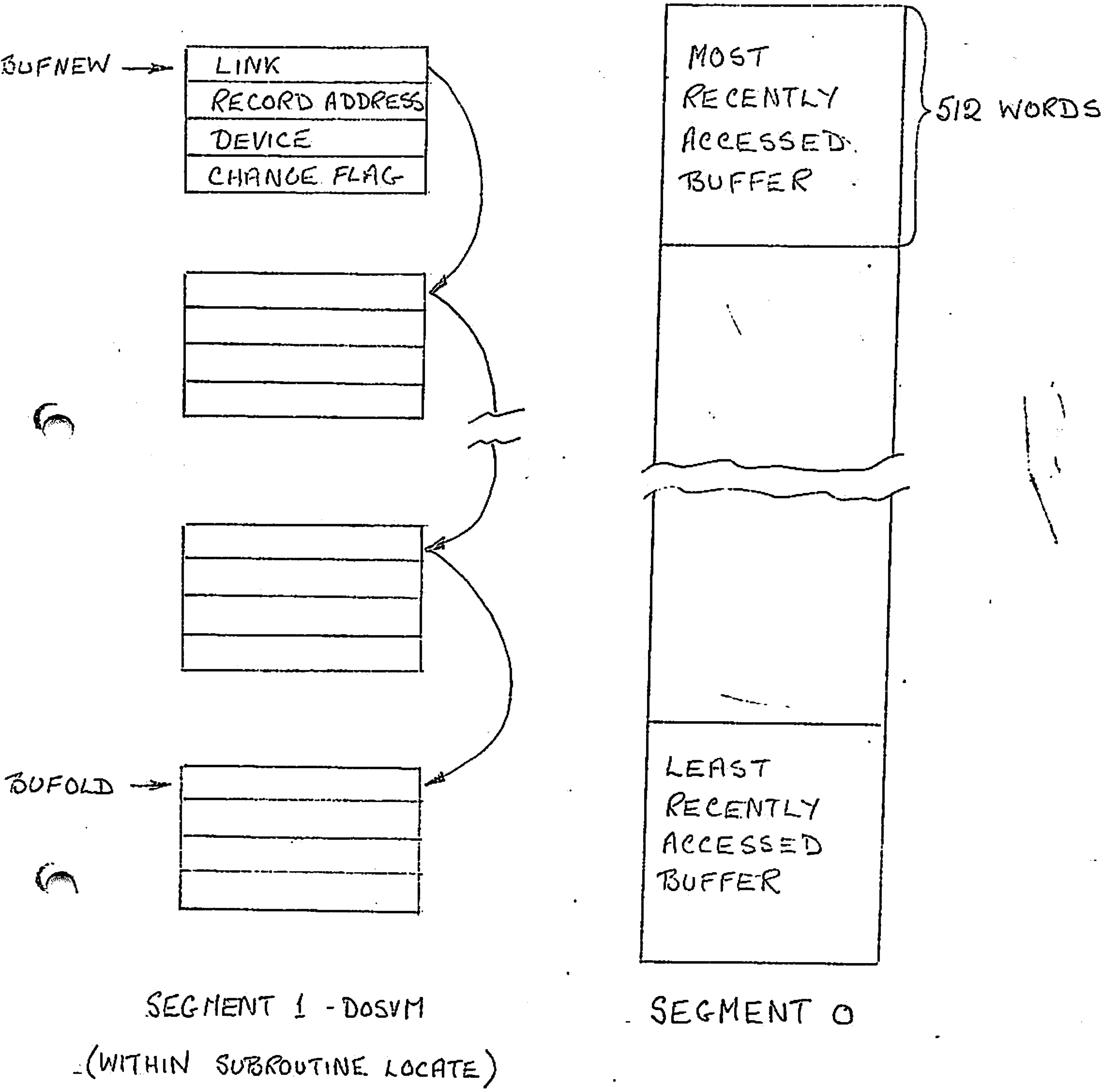


ASSOCIATE BUFFERS

WHEN ATTACH, SEARCH, PRWFIL, GETREC OR RTNREC WISH TO READ OR WRITE A RECORD THEY DO IT USING ASSOCIATIVE BUFFERS.

SAM FILES, DAM FILES, MFD'S UFD'S SEGMENT DIRECTORIES AND RAT'S ARE ALL ACCESSED VIA ASSOCIATIVE BUFFERS.

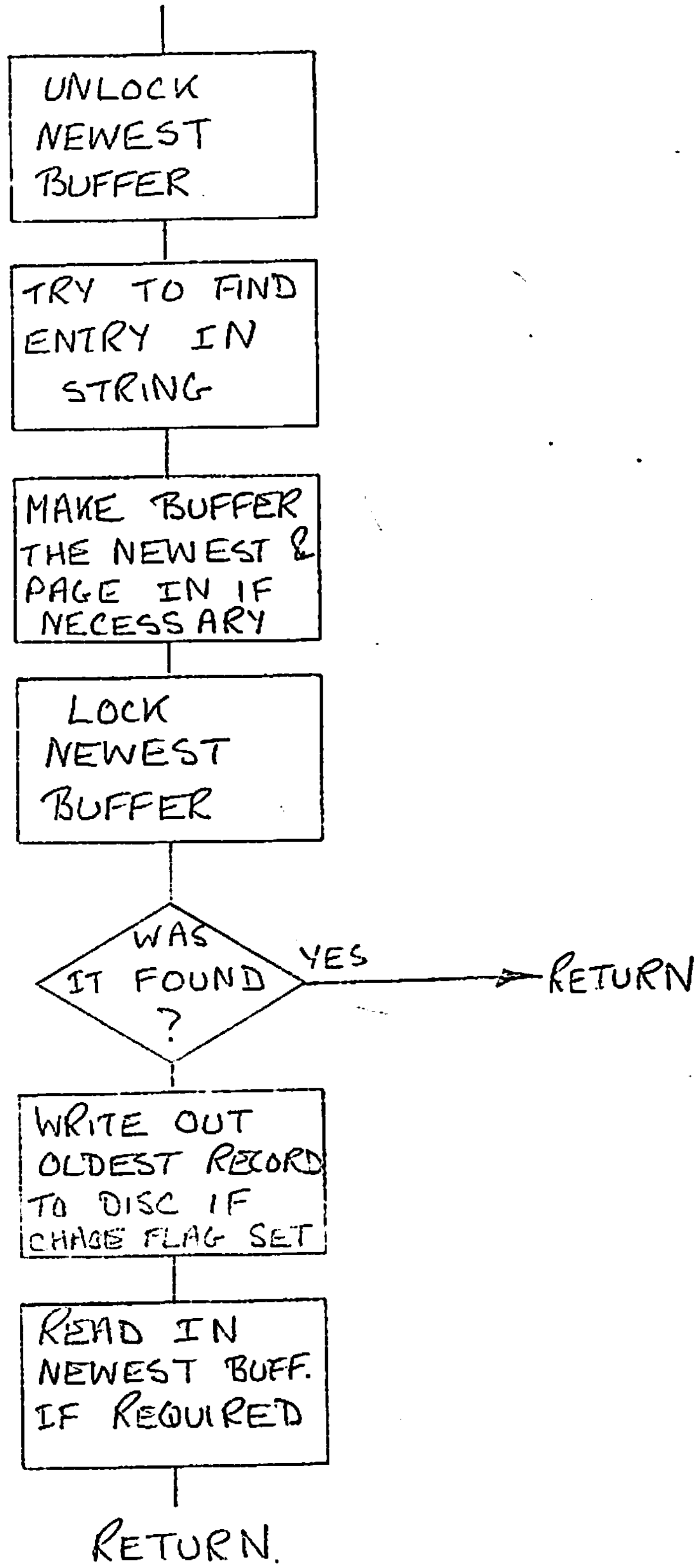
ASSOCIATIVE BUFFERS ARE HELD IN PAGED MEMORY IN A SPECIAL SUPERVISOR SEGMENT - SEGMENT 0.



BUFNEW
LINK
RECORD ADDRESS
DEVICE
CHANGE FLAG
BUFOLD
MOST RECENTLY ACCESSED BUFFER
512 WORDS
LEAST RECENTLY ACCESSED BUFFER
SEGMENT 1 - DOSVM
(WITHIN SUBROUTINE LOCATE)
SEGMENT 0

LOCATE (KEY RA LDEV)

KEY BIT 16 - CHANGE
BIT 15 - BYPASS READ



DYNAMIC DISC RECORD ALLOCATION

* WHEN THE FILE SYSTEM NEEDS MORE SPACE TO WRITE NEW DATA ON THE DISC IT USES THE FUNCTION GETREC.

NEW REC. ADD = GETREC (RECORD ADDRESS, DEVICE NO., ALTRET)

GETREC LOOKS AT THE BIT PATTERN IN DSKRAT AND TRIES TO FIND A SPARE RECORD AS NEAR AS POSSIBLE TO THE CURRENT RECORD

* WHEN THE FILE SYSTEM DELETES OR TRUNCATES A FILE, GIVING UP DISC SPACE, IT USES SUBROUTINE RTNREC

CALL RTNREC (RECORD ADDRESS, DEVICE NO.).

RTNREC UPDATES DSKRAT TO INDICATE THAT THAT RECORD IS NOW AVAILABLE.

USER COMRON (USRCOM) (PAGED)

STARTS AT :75000 IN DOSVM SEGMENT 1.
REPEATED FOR EVERY USER (256 WORDS PER USER).

REPEAT 18 TIMES:

VSTAT - STATUS E.G. OPEN FOR READING, WRITING, BOTH
VBRA - BEGINNING RECORD ADDRESS
VDVNO - DEVICE NUMBER
VDCRA - CURRENT RECORD ADDRESS } DAM FILES
VDRWP - READ/WRITE POINTER } DAM FILES
VCRA - CURRENT RECORD ADDRESS
VRWP - READ/WRITE POINTER
VRPIV - ACCESS PRIVILEDGES

CUFD:

1
2 CURRENT UFD NAME
3
4 CFDBRA BEGINNING RECORD ADDRESS
5 CFDDEV DEVICE NUMBER
6 CFDPOP RECORD ADDRESS OF FATHER UFD OR SD
7 CFDOWN = 1 IF OWNER = 0 IF NON OWNER

HOMUFD:

1
2 HOME UFD NAME
3
4 HOMBRA }
5 HOMDEV } AS FOR CUFD
6 HOMPOP }
7 HOMOWN }

LOGNAM:

1 }
2 } LOGIN NAME
3 }

COMPAR (40 WORDS) COMMAND LINE BUFFER
ERRVEC (40 WORDS) ERROR VECTOR
VCNECT (1 WORD) POINTER TO CONNECTED PROCEDURE
SCRAT (4 WORDS) SCRATCH FOR CNN

```
PASSWD     XXXXXX     ZZZZZZ
```

WHERE XXXXXX IS THE OWNED PASSWORD
AND ZZZZZZ IS THE NON-OWNER PASSWORD

```
PROTECT FILENAME  N1  N2
```

WHERE N1 IS THE OWNER PROTECTION KEYS
AND N2 IS THE NON-OWNER PROTECTION KEYS

VALID KEYS ARE:-

- 0 NO ACCESS
- 1 READ ACCESS
- 2 WRITE ACCESS
- 3 READ AND WRITE ACCESS
- 4 DELETE#TRUNCATE
- 5 DELETE, TRUNCATE AND READ
- 6 DELETE, TRUNCATE AND WRITE
- 7 ALL ACCESS

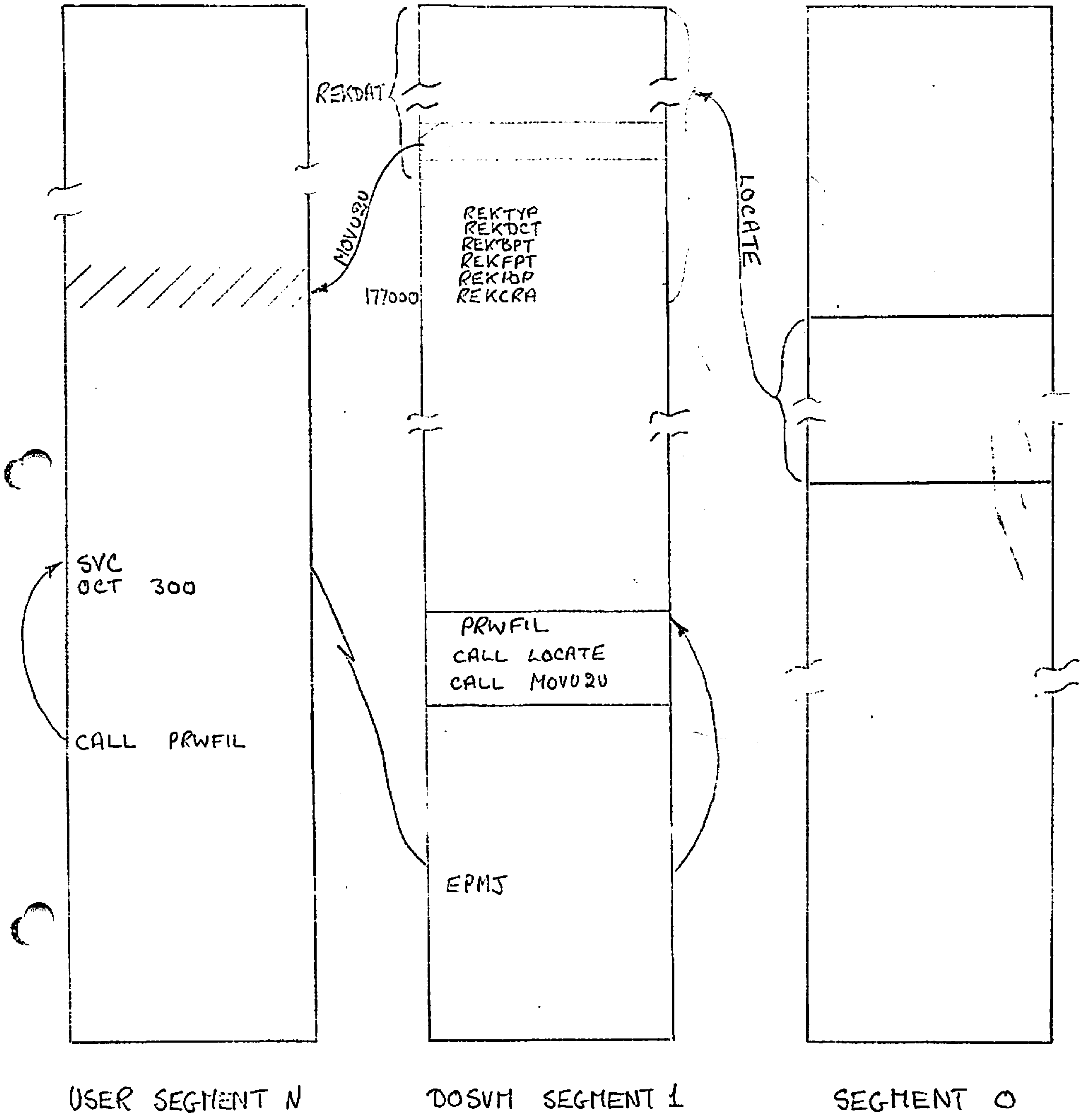
FORCE WRITE

```
CALL  FORCEW (0, FUNIT)
```

RWLOCK .

RWLOCK MAY BE CHANGED TO ALLOW MULTI-ACCESS TO ONE FILE:-

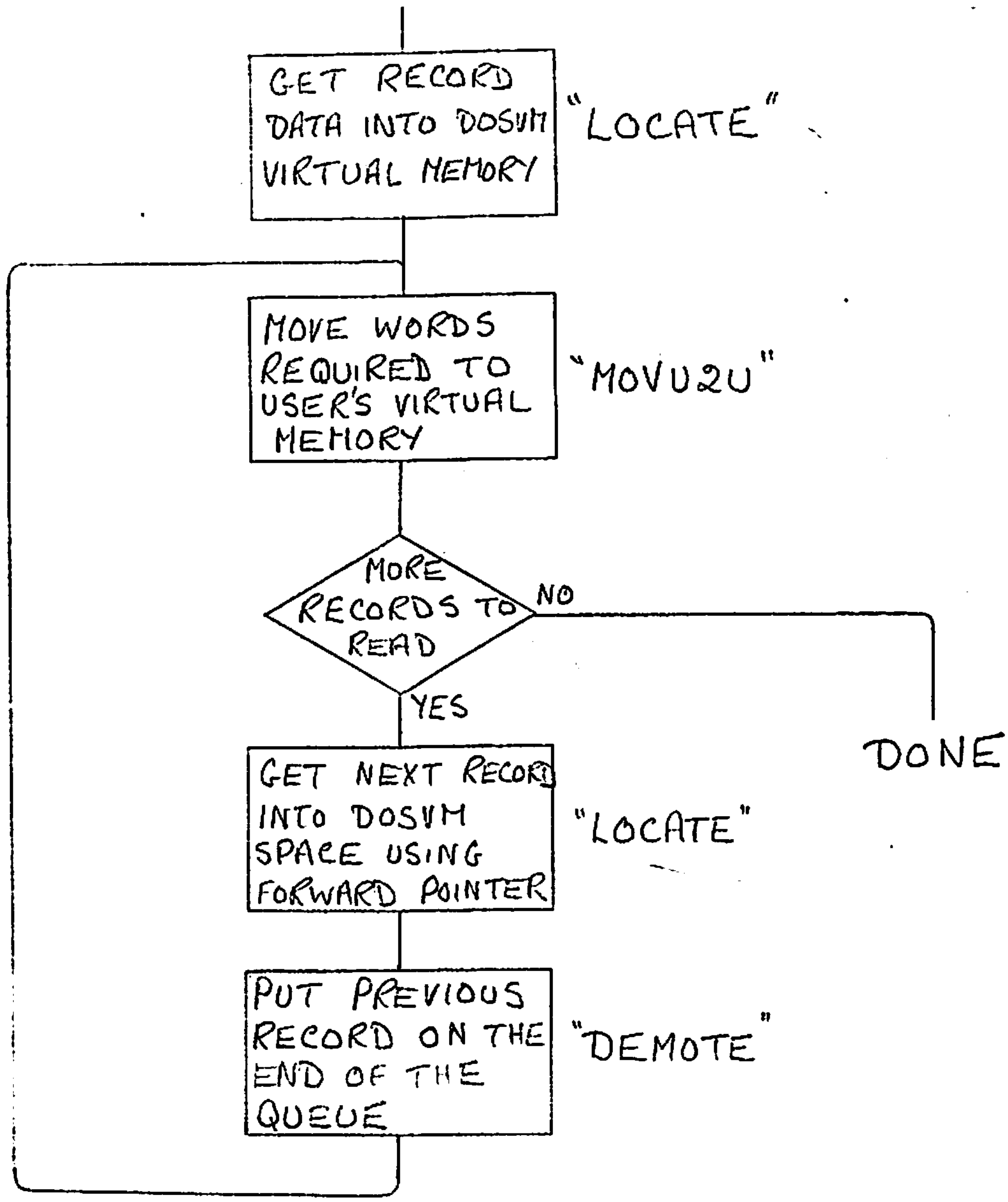
- 0 1 READER OR 1 WRITER
- 1 N READERS OR 1 WRITER (DEFAULT VALUE)
- 2 N READERS AND 1 WRITER
- 3 N READERS AND N WRITERS



ACTION OF PRWFIL ON READING

EXAMPLE OF USE OF LOCATE AND DEMOTE

FOR A SAM FILE IN DOSVM



MAG TAPE HANDLING UNDER DOSVM

CALL T$MT(UNIT, PBUFF, NW, INST, STATUS).

UNIT - MAG TAPE DRIVE - 0, 1, 2 OR 3.

PBUFF - ADDRESS OF DATA BUFFER

NW - NUMBER OF WORDS TO TRANSFER

INST - INSTRUCTION TO MAG TAPE - E.G. READ, WRITE, REWIND.

STATUS - STATUS VECTOR 3 WORDS:-

1 - STATUS FLAG (= 0 IF OPERATION COMPLETE,
=1 IF OPERATION STARTED.)

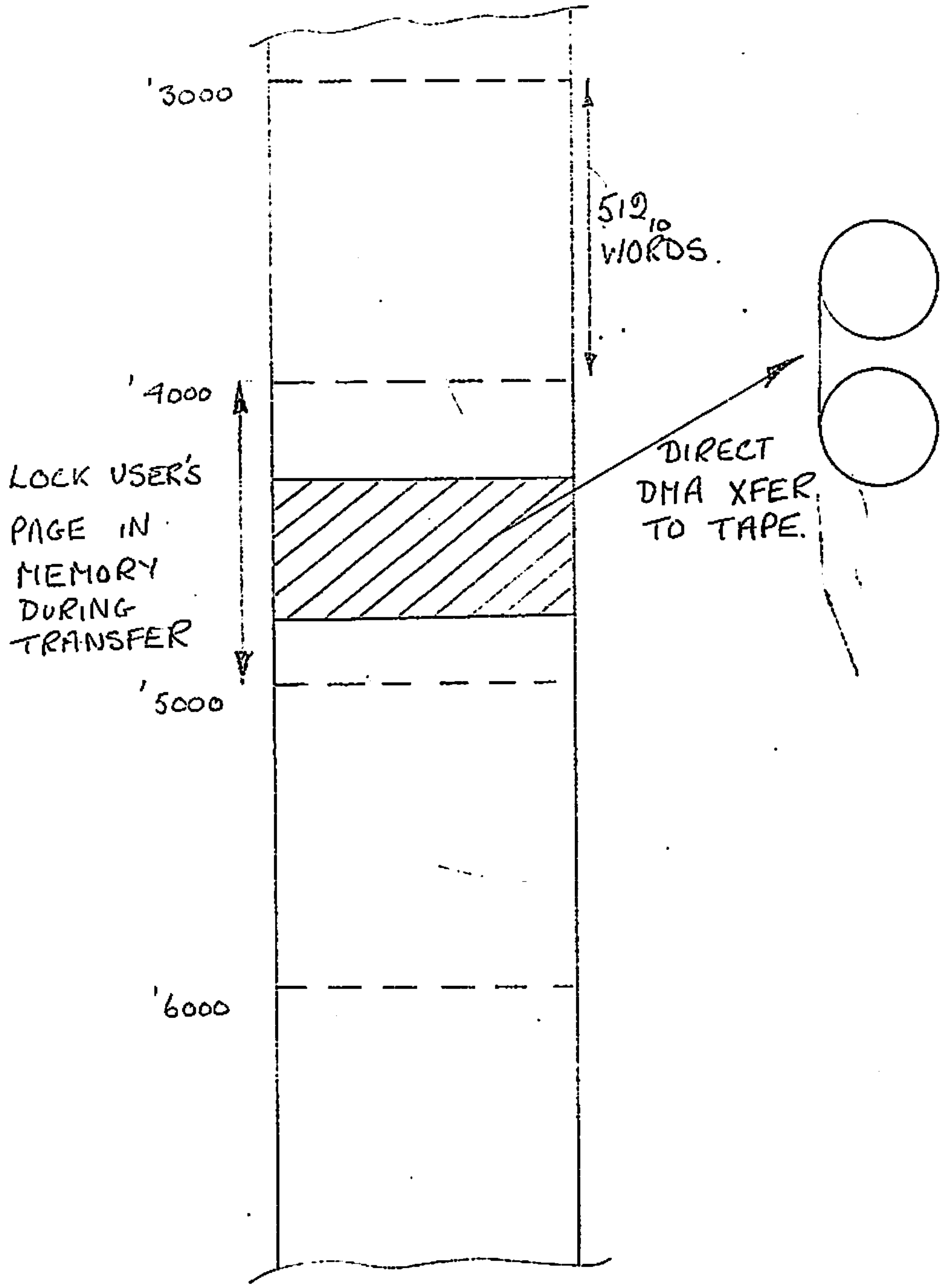
2 - HARDWARE STATUS

3 - NUMBER OF WORDS ACTUALLY TRANSFERRED

T$MT USES SVC 510

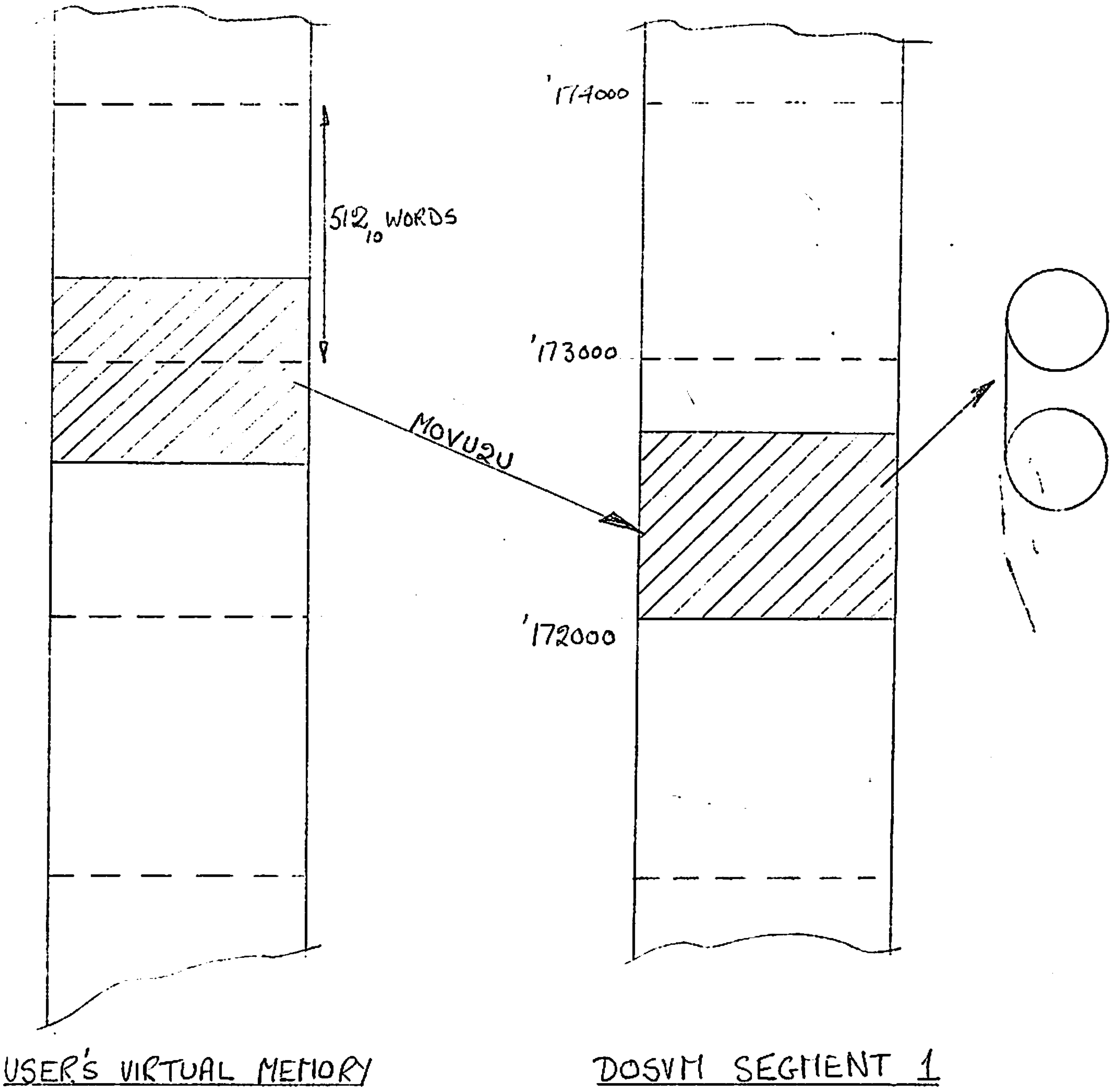
MAG TAPE IS NOT BUFFERED UNDER DOSVM

MAG TAPE TRANSFER WITHIN PAGE.

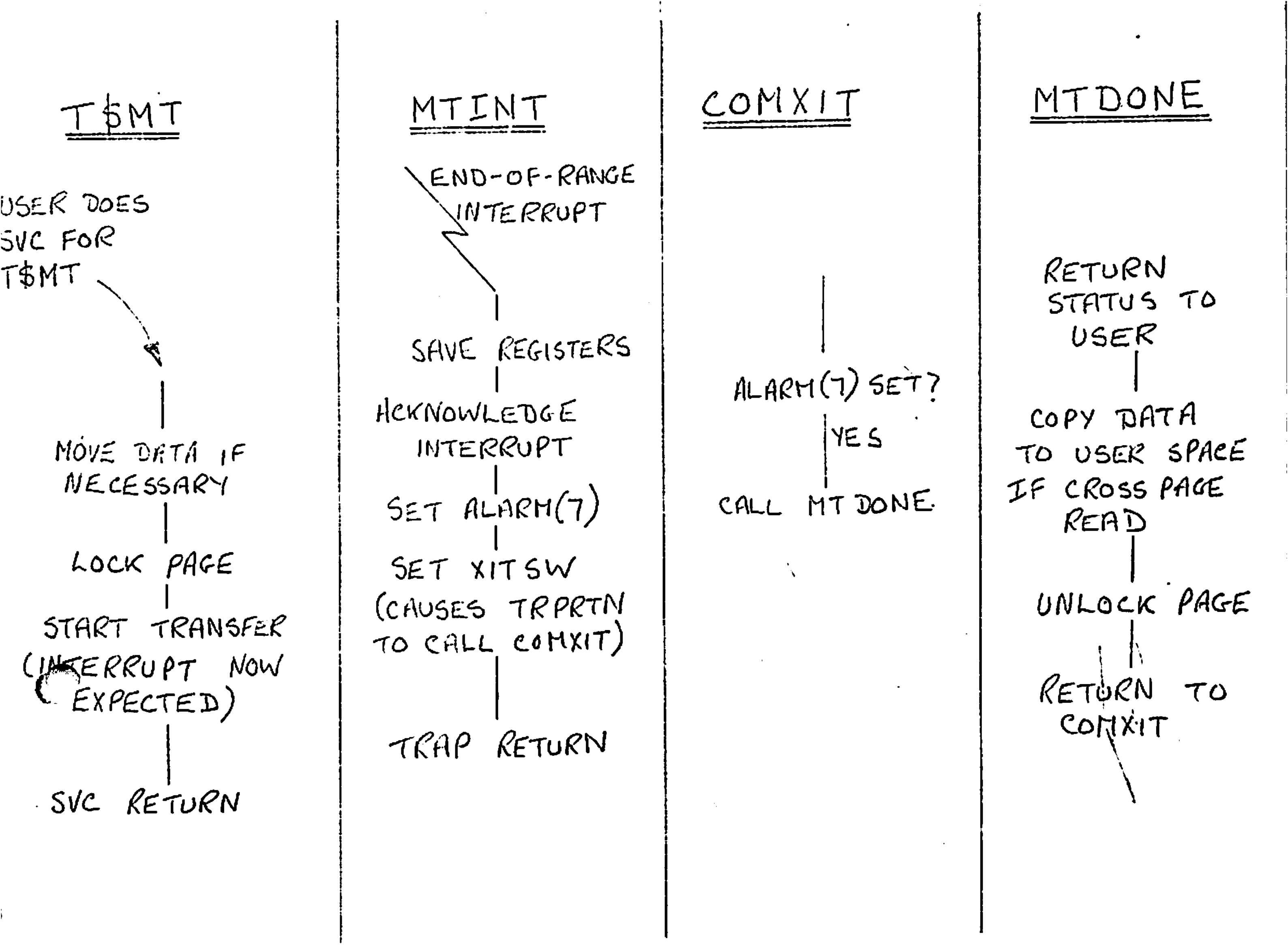


USER'S VIRTUAL MEMORY

MAG TAPE TRANSFER ACCROSS PAGE



AS I/O WORKS ON PHYSICAL ADDRESSES, AND BOTH USER SPACE AND THIS PART OF DOSVM ARE PAGED, 512 WORDS IS THE MAXIMUM BLOCK SIZE.


T$MT
USER DOES SVC FOR T$MT
MOVE DATA IF NECESSARY
LOCK PAGE
START TRANSFER (INTERRUPT NOW EXPECTED)
SVC RETURN
MTINT
END-OF-RANGE INTERRUPT
SAVE REGISTERS
ACKNOWLEDGE INTERRUPT
SET ALARM(7)
SET XITSW (CAUSES TRPRTN TO CALL COMXIT)
TRAP RETURN
COMXIT
ALARM(7) SET?
YES
CALL MTDONE
MTDONE
RETURN STATUS TO USER
COPY DATA TO USER SPACE IF CROSS PAGE READ
UNLOCK PAGE
RETURN TO COMXIT

DOSVM HANDLER FOR DATA PRODUCTS LINE PRINTER

CALL T$LMPC (UNIT, PBUF, NW, INST, STATUS).

UNIT	-	LINE PRINTER UNIT	
PBUF	-	POINTER DATA BUFFER (2 CHARACTERS/WORD)	
NW	-	NUMBER OF WORDS TO PRINT ON LINE	
INST	-	100000	READ STATUS
		40000	PRINT A LINE
		20012	SKIP A LINE
		20014	SKIP TO TOP OF PAGE
		200xx	SKIP ON CONTROL TAPE CHANNEL

STATUS - IS A 3 WORD ARRAY, STATUS IS RETURNED IN THE SECOND WORD:-

200 - ON-LINE

100 - NOT BUSY - I.E. THER IS ROOM IN THE PRINTER BUFFER QUEUE, AND A CALL TO WRITE A LINE WILL RETURN IMMEDIATELY.

300 - BOTH ON-LINE AND NOT BUSY.

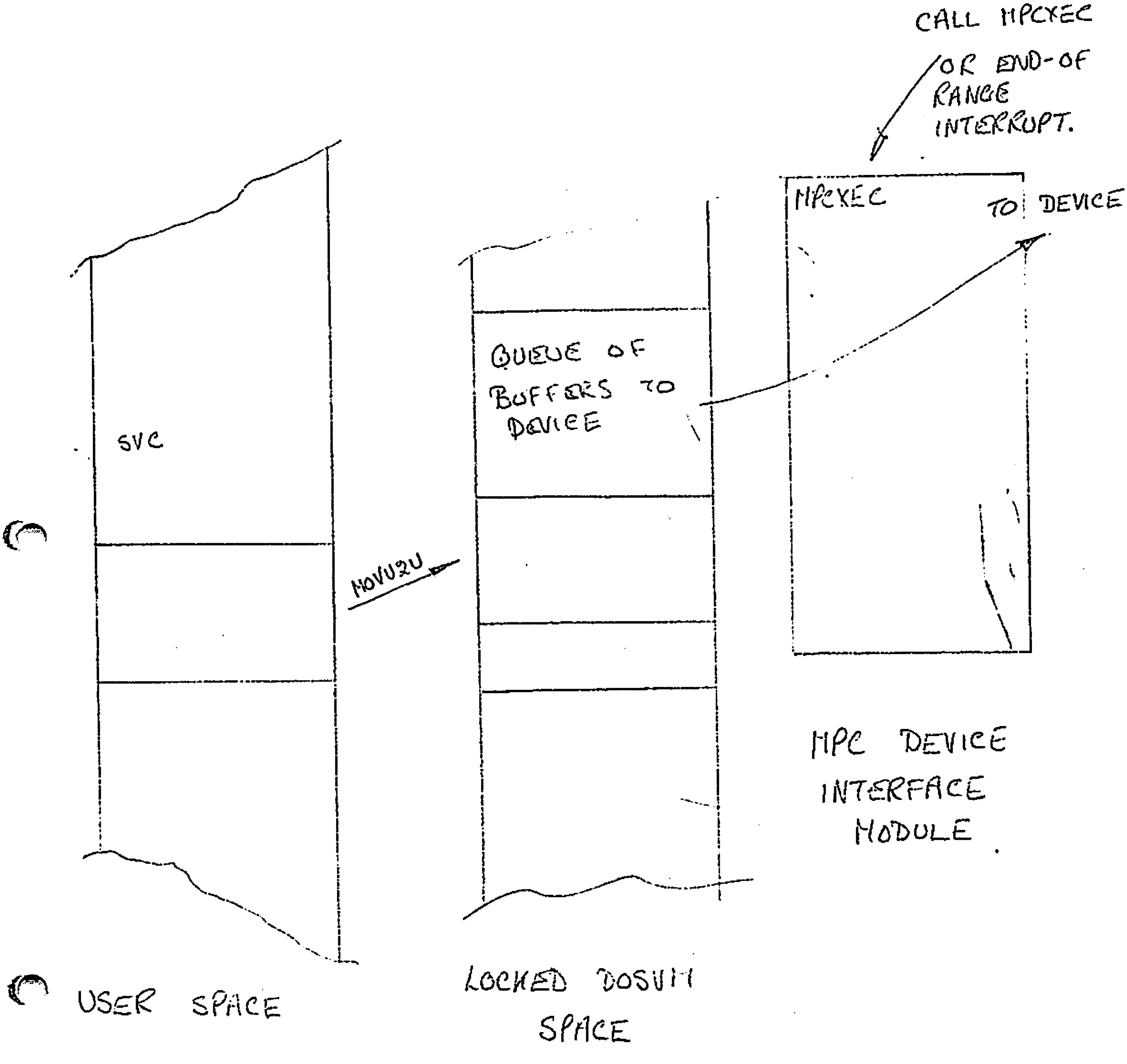
T$LMPC USES SVC 511

CARD READER

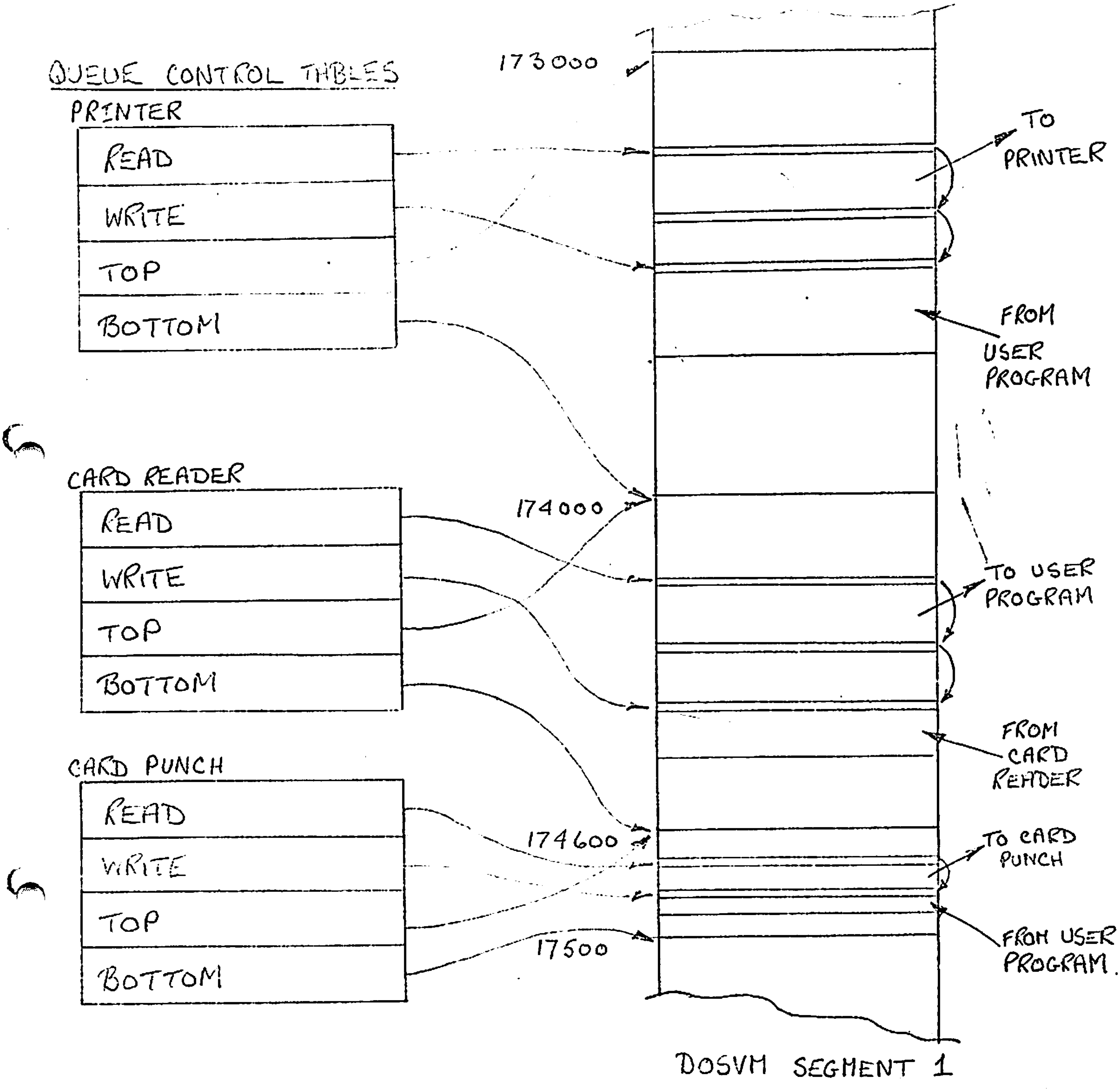
CALL T$CMPC(UNIT, PBUF, NW, INST, STATUS)

```
UNIT   -  UNIT NUMBER
PBUF   -  POINTER TO DATA BUFFER
NW     -  NUMBER OF WORDS TO READ
INST   -  100000 READ STATUS
       -   40000 READ CARD (ASCII)
       -   60000 READ CARD IN BINARY FORMAT
STATUS -   3 WORD VECTOR
       -   WORD 1 - DEVICE CODE
       -   WORD 2 - 200 - ON-LINE
                     40 - ILLEGAL ASCII
                     20 - DMX OVERRUN
                      4 - HOPPER EMPTY
                      2 - MOTION CHECK
                      1 - READ CHECK
```

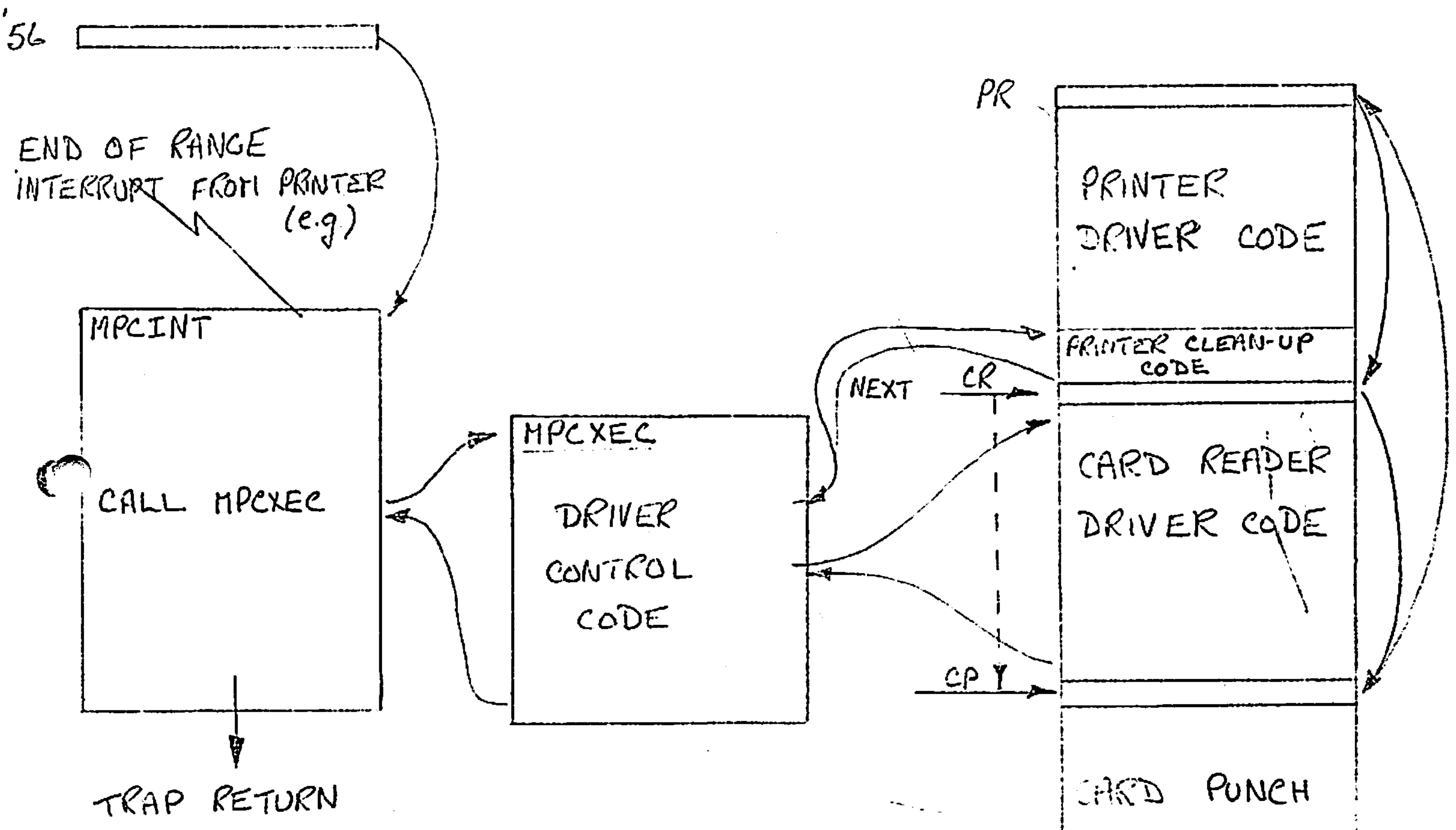
T$CMPC USES SVC 512


CALL MPCXEC OR END-OF RANGE INTERRUPT.
MPCXEC
TO DEVICE
QUEUE OF BUFFERS TO DEVICE
SVC
MOVU2U
MPC DEVICE INTERFACE MODULE
USER SPACE
LOCKED DOSVM SPACE

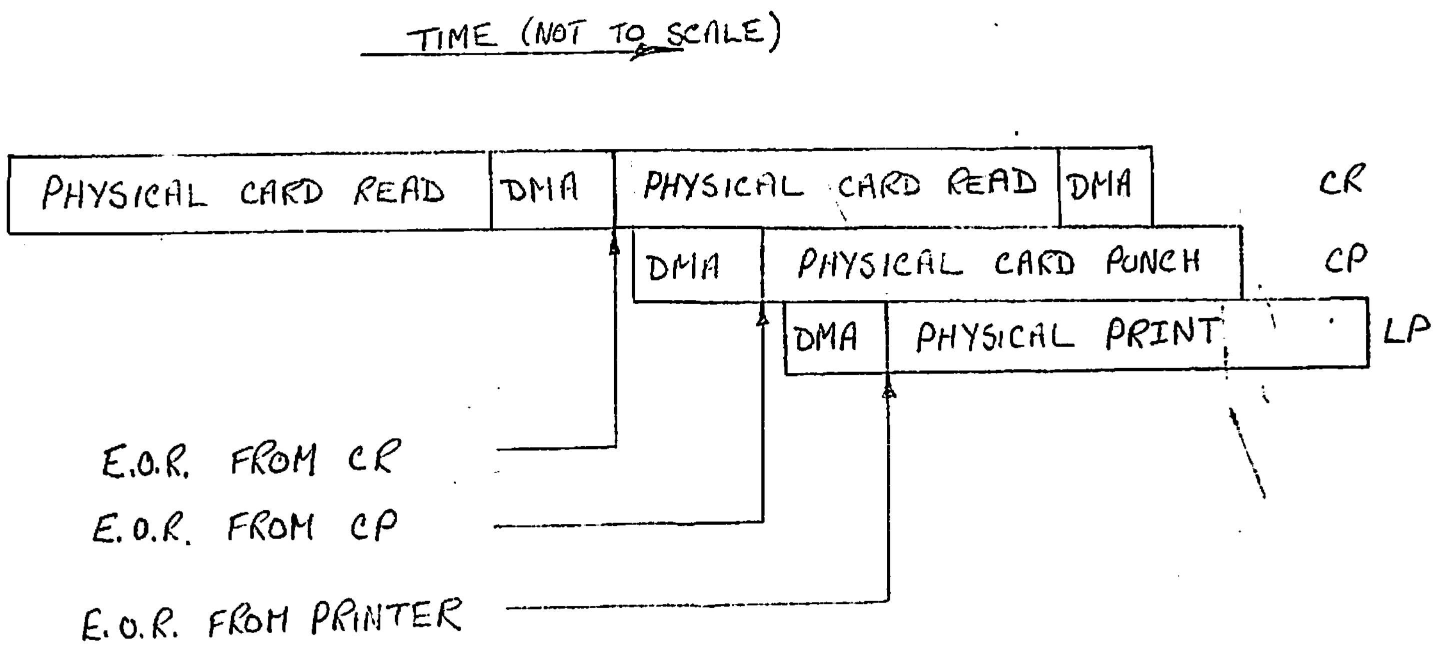
BUFFER CONTROL FOR PRINTER, CR AND CP ON MPC



INTERRUPT-LEVEL SOFTWARE FOR LP, CR AND CP ON MPC



CR IS OMITTED FROM STRING IF NOT ASSIGNED.



THE EFFECT IS THAT ALL THREE DEVICES CAN BE DRIVEN AT FULL SPEED, WITH EQUAL PRIORITY

MPC$NT - MPC $NITIALISATION

- SETS UP INTERRUPT VECTOR ('56) TO POINT TO "MPCINT"
- LOCKS MPC DRIVERS IN MEMORY

MPC$NT - MPC INTERRUPT HANDLER

- SAVES MACHINE STATE, ACKNOWLEDGES INTERRUPT
- CALLS MPCXEC TO CLEAN UP AND TRY NEXT DEVICE
- TRAP RETURN

USER SUPERVISOR CALLS FOR MPC (T$CMPC, T$LMPC)

- TESTS IF DEVICE ASSIGNED
- INITIALISES MPC (MPC$NI) IF NOT ALREADY DONE
- MOVES USERS DATA TO/FROM BUFFER QUEUE FOR DEVICE.
- IF NO ROOM IN BUFFER, OR NO INPUT DATA AVAILABLE, PUTS USER IN WAIT STATE AND CALLS COMXIT.

MPC CEVICE ASSIGNMENT

ASSIGNMENT OF MPC DEVICES CAUSE THE DOSVM VIRTUAL SECTOR 173 (PR) OR 174 (CR/CP) TO BE LOCKED IN MEMORY. IN THE CASE OF THE CARD READER, THE INPUT BUFFERS ARE CLEARED.

clIN

CALL clIN

GETS CHARACTER FROM USER TERMINAL OR COMMAND FILE.
RESULT IS RETURNED IN COMMON VARIABLES:-

"FLAG" - TRUE IF CHARACTER GOT

"clCHAR" - CHARACTER

COMMAND PROCESSING SUBROUTINES (INTERNAL TO DOSVM)

CNIN$

CALL CNIN$ (PBUFFER, CHAR-CHOUNT, ACTUAL COUNT)

- READS SPECIFIED NUMBER OF CHARACTERS FROM TERMINAL OR COMMAND FILE (DEPENDING ON THE COMMAND SWITCH (COMSWI) INTO USERS OWN BUFFER.

COMANL

CALL COMANL

READS COMMAND LINE FROM TERMINAL OR FILE OF CURRENT USERS INTO SUPERVISOR'S INTERNAL BUFFER FOR THAT USER IN USRCOM.

CMREAD

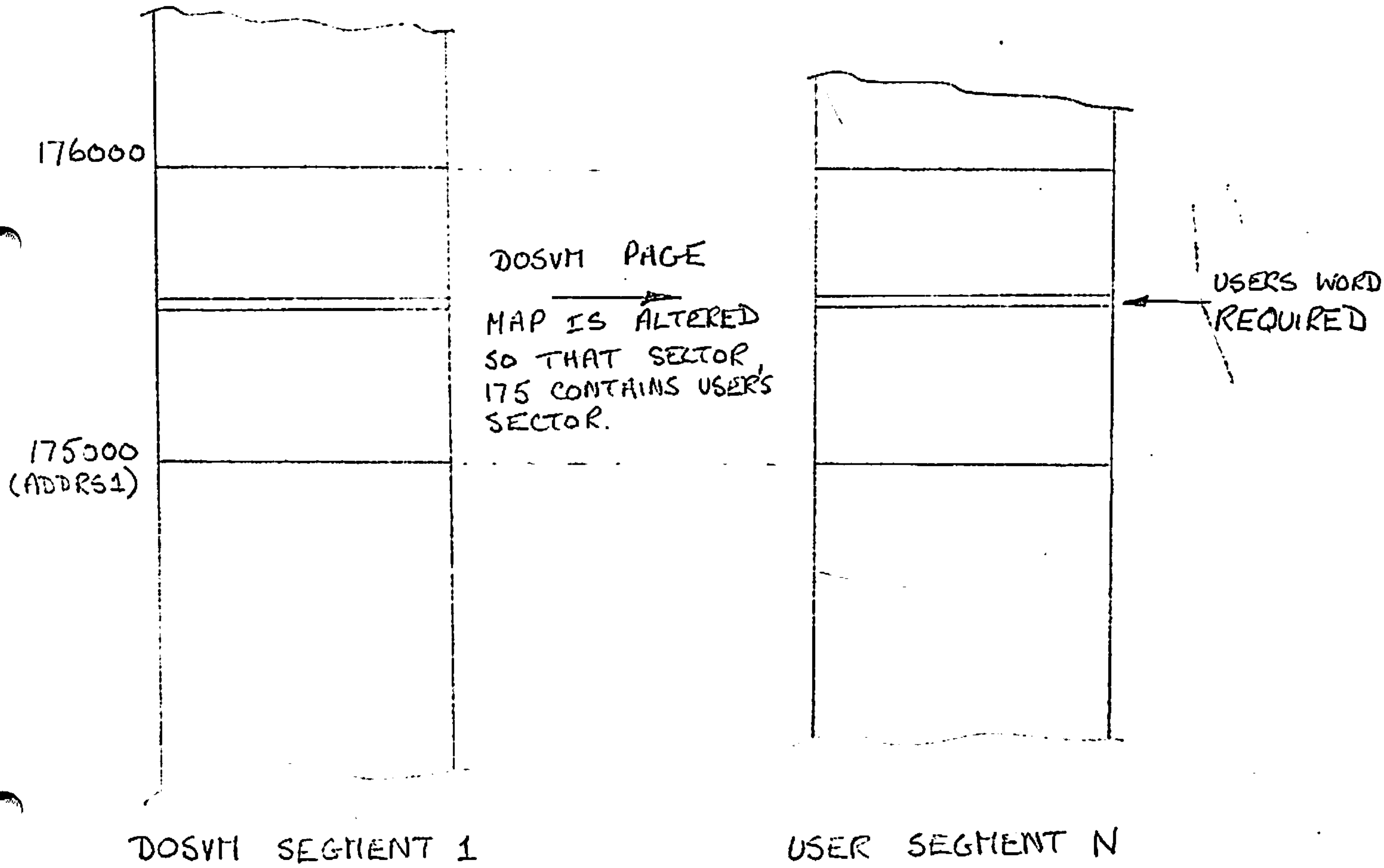
CALL CMREAD (ARRAY)

TRANSFERS ASCII FORMAT COMMAND LINE (IN USRCOM) TO ASCII#OCTAL ARRAY.

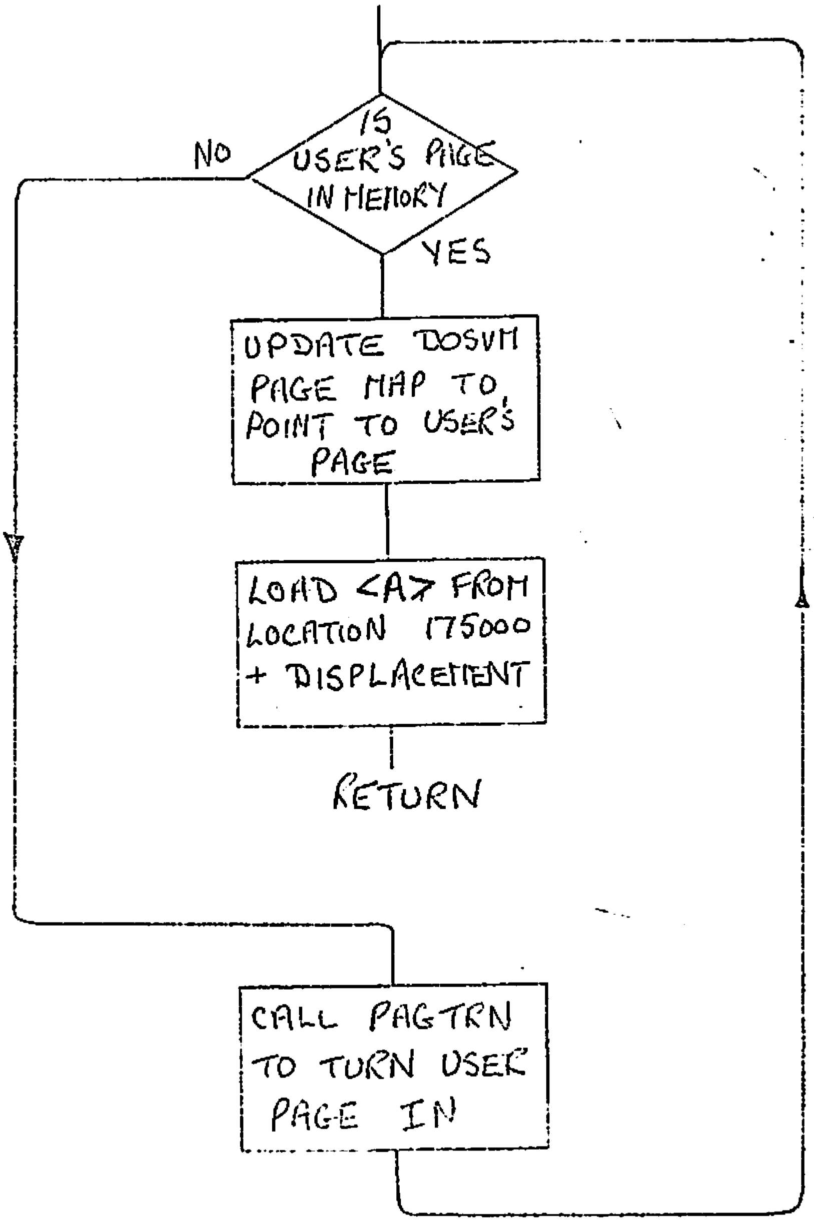
GETWRD - GETS A WORD FROM THE CURRENT USER'S VIRTUAL MEMORY

```
LDA     USER'S VIRTUAL ADDRESS
CALL    GETWRD
```

A = CONTENTS OF USERS VIRTUAL ADDRESS.
X = PAGE-DISPLACEMENT.



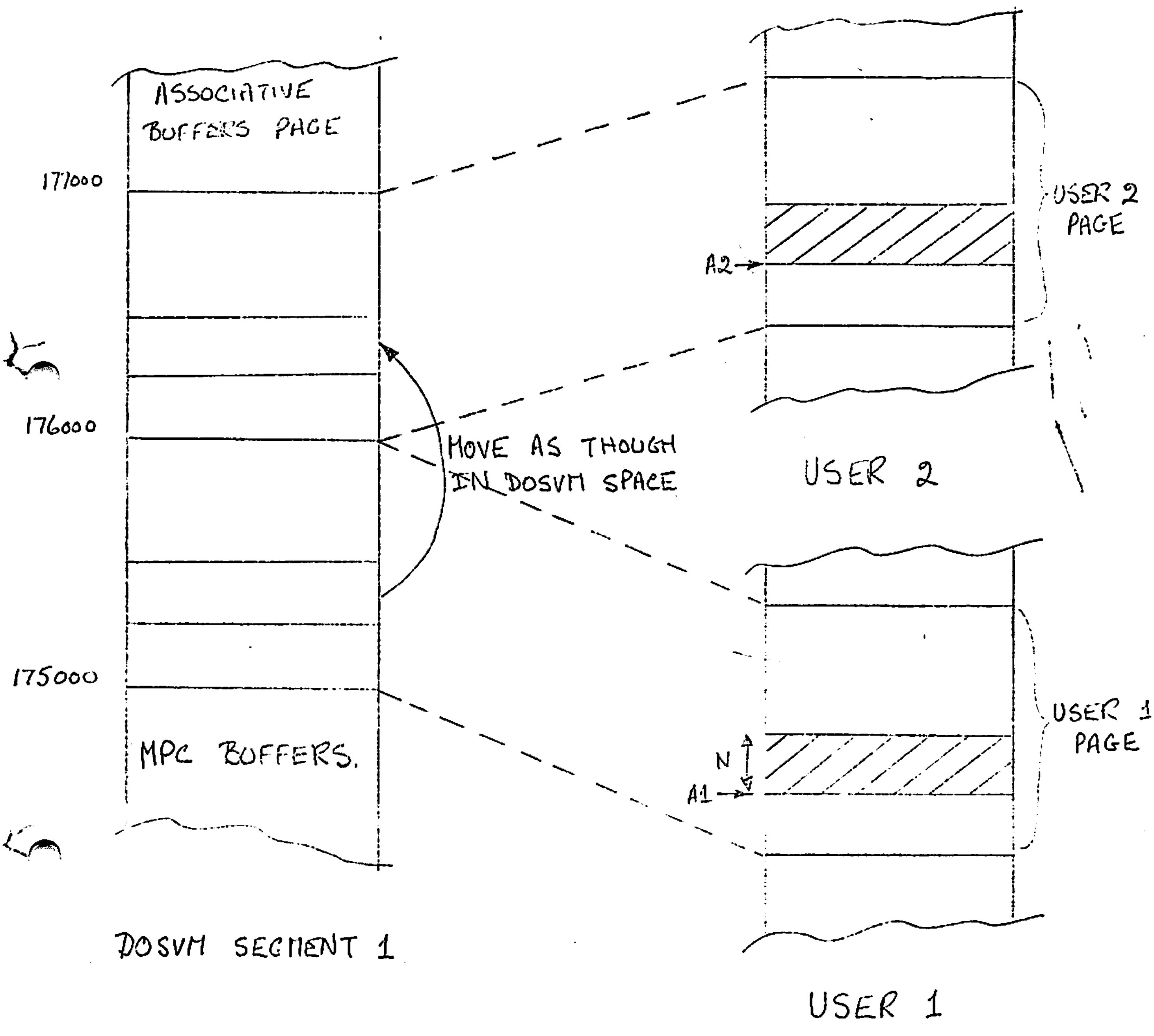
GETWRD



IS USER'S PAGE IN MEMORY
NO
YES
UPDATE DOSVM PAGE MAP TO POINT TO USER'S PAGE
LOAD <A> FROM LOCATION 175000 + DISPLACEMENT
RETURN
CALL PAGTRN TO TURN USER PAGE IN

MOVU2U

CALL MOVU2U (U1, A1, U2, A2, N)



NOTE: ACTUAL MOVE TAKES APPROXIMATELY 6 μsec PER WORD
REPEATED RE-MAPPING TAKES PLACE IS BUFFER CROSSES
A SECTOR BOUNDARY.

Bob Slater.

SUBJECT : CX MONITOR

A Background Monitor and a Command file submission program (CX USER) are now available on the Production time-share system. They allow any user to submit a command file into the queue and have it processed when all previously submitted files have been completed. In addition, the user may query the system to check the status of the Command file and its position in the FIFO queue.

I. Users View of the CX USER Program

1. File submission to the queue is easy. Simply type

CX file name

The file submission program will enter the command file at the bottom of the queue. In addition, it will be assigned a CO file number, from 1 to 71. All future references to this job will be made to the CO file, not to the original file name.

2. The following commands are recognised by the CX USER Program

CX /S## - Print the status of file ##
The job may exist in one of four states
WAITING : in queue, waiting to be run
EXECUTING : Command file running
ABORTED : job did not run to completion
COMPLETED : job terminated with CX /E (see below)

CX /D## - Drop file ## from WAIT queue

CX /Q - Print job Queue. This will list the CO file names and owners thereof, of jobs in the wait queue. Jobs executing at the time will have an asterisk before the file name.

CX /A - List the entire Job Activity File (71 entries). This will list all the entries and the status of each job held on the file.

CX /P - List all the users Personnel entries in the Job Activity File.

CX /E - End command file - can only be issued from a running command file (see description of command file)

II Description of Command File

Except for some minor changes, the CX Monitor command file has exactly the same format as a phantom command file.

The first line of this file should be :

**XXXXXX (where XXXXXX is the job identifier)

This is used to identify the job to the user and may be omitted. The next line must be an attach to the appropriate UFD and finally, the last line should contain

CX /E

This informs the Background Monitor that your job has completed correctly. A COMPLETE status is flagged in the activity file and the user is ~~flagged~~ logged out. Note that if the command file is terminated, for any reason, before this statement is encountered, an ABORT state is flagged in the activity file.

III CX Monitor Initialisation

1. To initialise the activity file

   A CXUFD
   R *CXINIT

2. To start the Background Monitor

   A CXUFD
   PH PH←GO

3. To run jobs see Users View of CXUSER

IV Description of Files required

UFD - CXUFD

CXUSER - -) C ← USER - command to create CX File Submission Program
CXSUB)
CXMSTR) C ← RUN - command to create Background Monitor
CXSLAV)
PH-GO Phantom to initialise the Background Monitor
CXINIT
*CXINIT Program to initialise the activity file

UFD - //XEQ

C←SCAN Should contain PH P← SCAN
S

P÷SCAN	should contain	A //XEQ R *SLAVE
*SLAVE	copied from CXUFD	
JOBS*	activity file created be * CXINIT	

V Activity File Format

For entries 1 - 71, user job entries, file format is

WORDS	DESCRIPTION
1	Job State Word 1 - in wait queue 2 - executing 3 - aborted 4 - job complete
2 - 4	Owners login name
5 - 8	Time entered queue
9 -12	Time started executing
13	Reserved
14-17	Time aborted / completed
18-20	Job Identifier

Entry 72 is the activity file in the batch control entry. Only two of the 20 elements are used as follows

(19.72) Next CO file # to be assigned

(20.72) Current job

VI Example CX Command File

The following is an example of a CX Command file :

```
** TESTID
A JIMW
A DEVLOP 1/2
FTN DEMO 1/777 30
LOAD
LO B<DEMO
LIB
SA *DEMO
EX
CX /E
```

STRUCTURE OF PAGED MEMORY SYSTEMS

by John William Poduska

Paging is a method of separating the virtual addresses generated by a program from the physical addresses of a memory system. But paging is much more flexible than simple relocation methods because it allows a nearly arbitrary mapping of virtual to physical address space.

Page-Turning is a separate notion and implies the use of a two-level storage system (high-speed memory and a drum system, for example) and provides a method for effectively decoupling the size of virtual address space from high-speed memory space; e.g., it is possible to run 64K programs in an 8K memory system.

The following describes the notion of paging, implementation of paging, associative paging, and page-turning. In addition, the relation of paging to effective high-speed memory allocation, reliability, and graceful degredation of computing systems is considered.

Notion of Paging

Paging is a method of distributing the memory requirements of an operational program over whatever HSM (High-Speed Memory) is available; i.e., a new process in a multifunctional computing system can be run in whatever HSM space is available. This problem of allocating HSM can be very serious in multiprogramming systems (especially in manned, multi-access systems), mainly because of the unpredictable requirements for capacity. In addition, paging allows memory reconfiguration to be done with software only (no hardware switching) by a supervisory program in such a way that user processes are unaffected by the change. Memory Paging or, more precisely, Physical Memory Paging not only solves the problem of allocation and reconfiguration, but in addition provides many other valuable features, such as memory protection, two-level storage, memory segmentation, etc.

This paging of memory is accomplished by considering the address as presented by a computer (the virtual address) and the address referenced by the memory system (the physical address) as being separate. For concreteness, let us consider the operation of a PRIME computer system. The virtual address of the PRIME is 16 bits wide and provides for 2^{16} or 64K words. However, the amount of physical HSM attached to the PRIME can vary from 4K to 64K words, so there is already apparent a distinction between virtual address space and physical address space.

Let us now go one step further and divide both the virtual memory and physical memory into equal-length blocks (we shall call them pages) of words, say $2^9=512$ words per block (actually any power of 2 would do, but 512 seems most reasonable). Then (as illustrated in figure 1) there are $2^7=128$ pages in the virtual address space, and 2^k pages in the physical HSM system.

Now in a normal PRIME, one would associate virtual page 1 with physical page 1, virtual page 17 with physical page 17, etc. But it is perfectly possible to associate virtual page 1 with physical page 13, and virtual page 14 with physical page 7, etc., in any ordering desired as long as the mapping of virtual pages into physical pages is unique and complete – after all, any one physical page is just as good as any other physical page.

An example of mapping of virtual addresses into physical addresses is shown in figure 2. The mapping is as follows:

Virtual page	Physical page
00	12
01	21
02	13
03	01
04	03
05	00

Thus the virtual address '01065 is virtual page '01 and word number '065 and might be expressed as 01/065, since virtual page '01 maps into physical page '21, the corresponding physical address is 21/065.

All that is required to perform this association in the computer is a map of virtual page numbers into physical page numbers; i.e., a table is required which will yield the physical page number when the virtual page number is searched for. This map or table could be in a separate high-speed store, or, more reasonably (as shall be shown), it could be in normally addressable HSM. In either case, the hardware is required to perform a sort of indirect address cycle on every memory reference.

An alternate and perhaps instructive way to view the notion of paging developed here is to consider that the leftmost seven bits of the virtual address (i.e., the virtual page number) specify a base register; that the rightmost eight bits specify a relative displacement; and that the base register can contain only numbers (base addresses) which are multiples of '1000.

Implementation of Paging Schemes

The notion of paging is an important one because of the many systems advantages provided, but there is also one serious difficulty to circumvent: since every memory reference requires an indirection through the page map, the effective memory speed may be cut in half; i.e., every memory reference requires two memory cycles, one for mapping and one for datum. There are a number of ways of getting around this problem of multiple memory access, the most promising of which is associative paging (discussed later), but it is important to consider first just how a paging scheme might be implemented.

The central-processor/high-speed-memory (CP-HSM) interface may most reasonably be considered to be contained in four functional boxes as follows (see figure 3): the CP proper which given virtual addresses and receives (or stores) data, the MRC (Memory Reference Controller) which converts virtual addresses into physical addresses and attempts to access the proper memory bank, the ADU (Access Distribution Unit) which resolves conflicts between competing requirements for reference to a given memory bank, and the HSM bank proper which accepts physical addresses and yields (or stores) data. Computing systems contructed of these four functional boxes can be considered truly modular because additional CPs or HSMs can be attached to the system by nominal expansion of ADUs and MRCs.

The hardware equipment required for implementing a paging system fits most reasonably into the MRC; i.e., the functional unit which converts virtual addresses into physical addresses. In the case of a PRIME, the virtual address arriving at the MRC is 16 bits long (7 bits for virtual page number and 9 bits for word number). This division of virtual page and word number can usefully be represented by "vp|wn," meaning the concatenation of virtual page and word number. The MRC is then responsible for transforming the virtual page number into a physical page number (vp→pp), forming the resultant physical address (pp|wn) and then causing that memory reference to be made.

The process of looking up a page number in a map requires, of course, that the hardware know where the map is. The map might be very small in length and reside in live registers (as in the SDS-940), or it might be of moderate (say less than 64 entries) in length and reside in a small scratch pad memory (as in the IBM 360/44), or it might be of arbitrary length and reside in regular HSM (as in the IBM-360/67 or the GE-645). The latter scheme is by far the most flexible (and least costly) and will be considered here.

The residence of the page map in HSM implies that the address of the map be available to the MRC. While this address could be fixed (wired-in), a far more flexible scheme is obtained if the page map address can be set under program control. A page map address register (PMAR) is provided for this purpose and it defines the origin of the page map. The page map most conveniently begins at a location which is a multiple of the map length; e.g., for the PRIME with 7 bits for the virtual page number, the map is 128 words long and, most conveniently, begins at a location which is a multiple of 128. In addition, there must be a register to hold the resultant physical page number (PPR), as illustrated in figure 4.

With the MRC constructed as shown schematically in figure 4, the fetching operation takes part in two steps as follows:

1. Step 1, Mapping Fetch

   address: (PMAR)|vp
   data: ((PMAR)|vp)⟶(PPR)

2. Step 2, Data Fetch

   address: (PPR)|wn
   data: ((PPR)|wn)⟶CP

where parenthesis are used to denote contents. The potential inefficiency of paging shows clearly here because of the extra memory reference required for the mapping fetch.

Associative Paging

Paging implementation schemes which require a specially programmed store or special registers are less effective than those having page maps in HSM for a number of reasons, including: (1) page maps in HSM allow operating programs to alter the map, (2) several maps may be in HSM at one time, (3) the complete virtual memory space can be changed by simply reloading the PMAR, and (4) a given process may consist of many virtual core images. The primary disadvantage of having the page map in HSM is simply that effective memory speed is cut in half.

The advantages of the HSM map can be retained while increasing memory speed to nearly normal by employing a small associative store, and in fact, this is what is done on the Atlas, the GE-645, and the IBM 360/67. The idea is simply this (see figure 5): Suppose a few (say 4) holding registers (an associative store) are incorporated into the MRC to save the last few distinct mapping references; and suppose further that the associative store is consulted first before a mapping reference is made. If the associative store is very fast and causes no decrease in speed, one can expect very good results because (a) instructions are normally executed sequentially, and (b) data tends to be very well clustered. It turns out, in fact, that the statistics are better than one might expect; e.g., for a 64K virtual memory, 128 each 512 word pages, and a 4-word associative store, one can expect that fewer than 10% of the memory references of an "average" program require a mapping reference.

With such an associative store, the operation of the MRC would be something as follows (see figure 5):

Start: Compare vp with all vp's held in the associative store.

A. If vp matches one in store:
   1. form data address pp|wn
   2. read data and transmit to CP

B. If vp matches no vp in store:
   1. form map address (pmar)| vp
   2. read map and put result in associative store
   3. continue as in A above

Thus the MRC makes use of a simultaneous comparator circuit to perform the association function and performs a mapping reference to memory only as required.

All associative mapping schemes operate on much the same principles, and the only significant difference among the various systems are the number of registers in the associative store and the manner in which a new map reference replaces an existing one (step B.2). The number of registers (sometimes called "sticky" registers) in the associative store is clearly a variable subject to economic consideration, but the replacement scheme implemented can cause considerable variations in the efficiency of the system.

One simple replacement scheme is the "round robin" method in which the register replaced is choosen cyclically from the registers in the associative store. Such a replacement scheme makes no use of any knowledge gained from the usage history of the information, but even so it is a simple scheme and can be very effective. A random replacement scheme has very similar properties.

A more effective replacement scheme can be devised by ordering the registers in the associative store in order of time since last usage. Whenever an associative register is used, it is "promoted" to the top of the list and all others are "pushed down" a notch. Whenever a mapping reference is required, the least used register is replaced and promoted. Such a priority replacement scheme takes some account of history and on the average performs much better than a round-robin scheme.

Implementing a priority replacement scheme can be somewhat complicated. Theoretically, the information required to specify the priority sequence of N register is log2(N!) bits (flip-flops), but the logic requirements for changing the priority sequence are very large. One can also attach a binary register of length about log2(N) to each sticky register and use it to hold a priority number. This scheme requires a total of about N*log2(N) bits of storage but considerably less logic than the scheme above. A final scheme of implementing the priority mechanism is to have a separate bit of information for expressing the truth of p_{ij} (register i has priority over register j. Since p_{ij} is the negative of p_{ji}, and p_{kk} is not required, this scheme requires N*(N-1)/2 bits of storage. While the storage requirements of this method rise eventually as N*N compared with the N*log 2(N) of the above two methods, the logic requirements are very modest, and this latter method is invariably the least expensive implementation.

Page-Turning

The notion of paging so far developed consists of a mapping of virtual addresses into physical addresses through a map and perhaps aided by an associative store. However, it is quite possible that physical memory and virtual memory are different sizes, or that for some other reason it is necessary to avoid having all virtual pages in physical memory.

Suppose, for concreteness, that a PRIME was implemented with only 8K of physical HSM but that a mass storage device were connected. With but 8K of HSM, only 16 of the 128 possible virtual pages can reside in HSM at a given instant, and the rest must either be lost or placed in mass storage. Unless some modification is made to the paging system, the program is still constrained to 8K of HSM.

Suppose, however, that the excess portions of a 128-page program are kept in mass storage and that the paging system is modified as follows: Every entry in the page map points either to a physical page in HSM or to a place in mass store where the page contents are stored, each entry also has a bit to indicate whether the page is in HSM or not, and the MRC modified to perform an interrupt sequence if reference is made to a page not in HSM. Such a scheme extends the function of the page map from a simple memory mapping to include other information about interruption and page location. It will later prove useful to place even more information in the page map.

The interrupt caused by a page fault (the interrupt occurring when a page is not in HSM) can be used effectively by an interrupt routine to dump out an old page, bring in the page required, and restart the interrupted program. This notion is called Page-Turning, and it allows the size and availability of HSM to be completely decoupled from the size and number of processes partially loaded into HSM.

A paging and page-turning scheme can be viewed as causing HSM to be treated as an associative look-aside store operating in conjunction with master copies of information residing in mass storage. The efficiency of such a system depends on the availability of HSM, the speed of mass storage, and the strategy of the interrupt program (the "cartographer"). Usually, the secondary storage mechanism is a small fast drum and the strategy employed is a random or round-robin replacement scheme.

Page-turning is an important notion because it allows a process of nearly unbounded size to run in whatever HSM space is available. Furthermore, any number of processes (dependent or independent) can be partially loaded and execution can be switched among these processes at the will of the scheduling mechanism without regard to swapping. Finally, the extension of the MRC and memory map function opens the door to protection schemes for pure procedures, etc.

Two pitfalls do arise in the use of page-turning schemes and are worthy of some note. First, the interrupt sequence must cause the PMAR to be reloaded and the associative store to be cleared, otherwise the MRC might make memory references for the interrupt routine in the memory of original process. Second, the page-turning program must never allow itself to be swapped out, otherwise the program would get into an interrupt loop.

Summary

The notions of paging and page-turning are extremely important to the effective use of modern multifunctional computers because these schemes provide effective answers to the three most serious problems in the use of HSM, to wit:

1. HSM allocation, protection, and relocation
2. Modular expandability
3. Reliability and graceful degradation

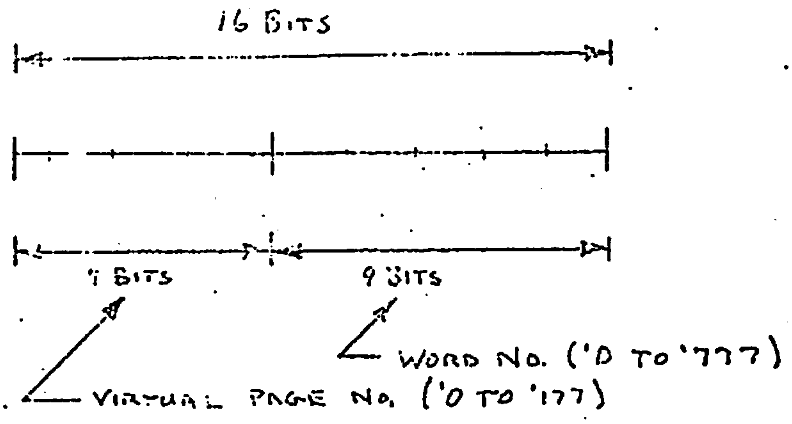
Paging itself provides a method of protecting and relocating memory in an effective way. Paging also allows the easy allocation of pages of HSM located anywhere, even in memory modules of different speeds. It avoids the problem of physically moving program and data to close up gaps in memory. Page-turning complements paging and eliminates the need for any relation between virtual memory size and physical memory size (except for efficiency consideration).

Paging and page-turning also provide a simple way of allowing and implementing a modular expansion capability for HSM. Memory map entries invariably provide room for many more pages than implemented in HSM; e.g., the PRIME paging mechanism would reasonably allow for 4,096 pages of 512 work for a total of over two million words. If the map generation routines choose page locations from a list held as data, then physical memory can be expanded almost without limit by the simple modification of a list – a trivial program change.

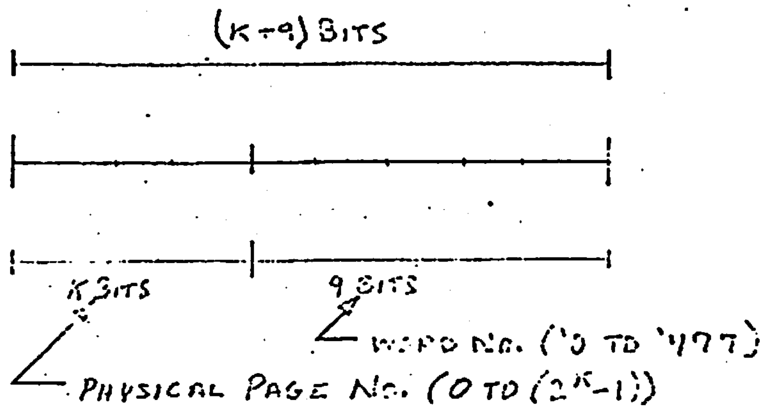
In a similar way, if a diagnostic routine (or a human maintainer) should discover a faulty memory module, a simple programmed change to the list of available pages would cause the map-maintaining routines to completely ignore that region of HSM. Thus, memory failures can be accommodated with software only – there is no need for hardware switching or shutdown.

The notions of paging and page-turning may be extended in several directions. For example, the map entries might very well have extra bits for indicating that a certain page is to be protected for read-only, or execute-only, or write-only, etc.

Memory allocation schemes have also been extended to include the notion of segmentation. This is a scheme for providing the user process with as many segments of decoupled memory as are necessary. Segmenting on the one hand, provides the user with the illusion of a two-dimensional memory organized as a few columns of many words each. Paging, on the other hand, is a tool provided for the efficient use of a supervisory routine and is undetectable by the user process.



LOGICAL ADDRESS SPACE



PHYSICAL ADDRESS SPACE

FIGURE 2

DIVISION OF ADDRESS SPACES

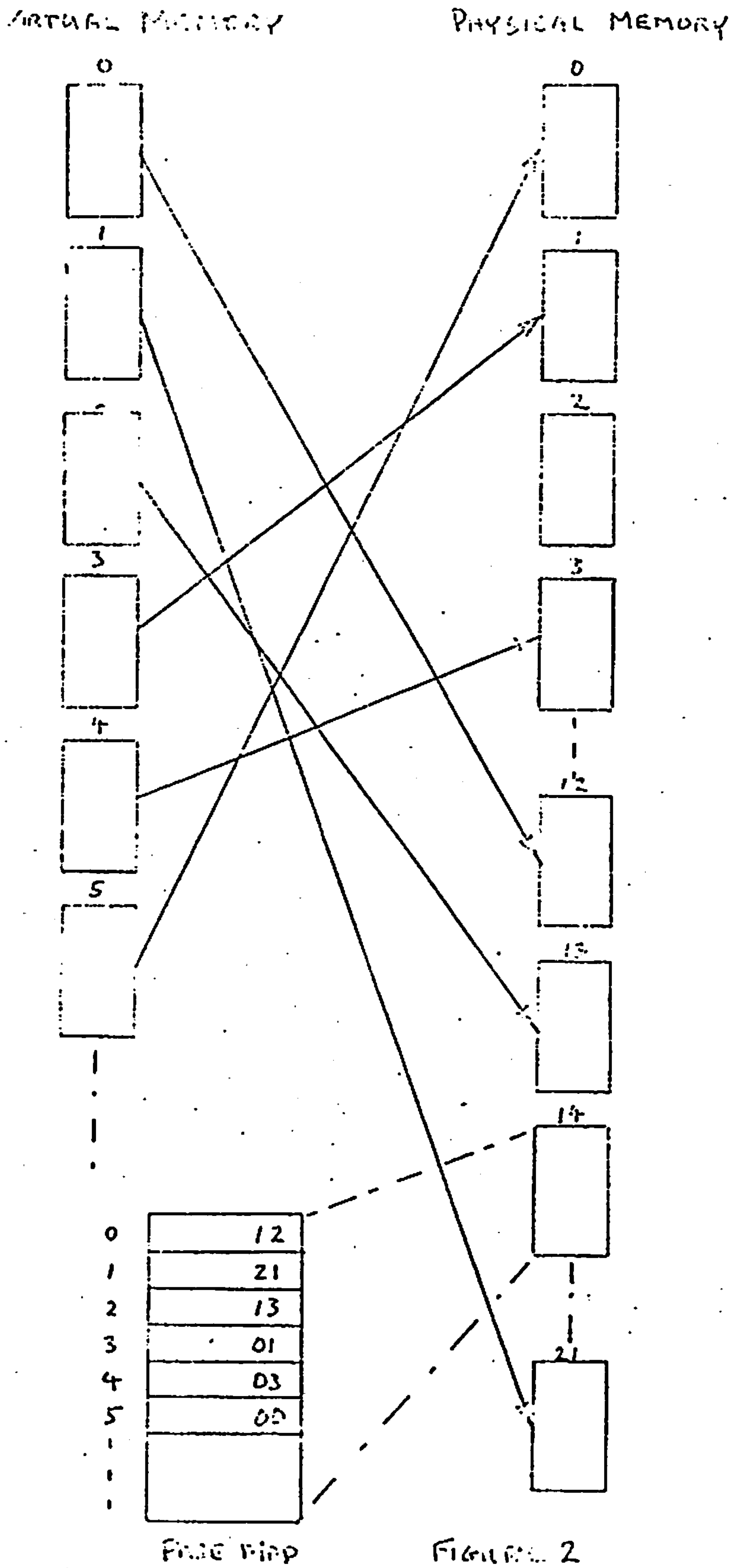


FIGURE 2
EXAMPLE OF PAGE MAPPING

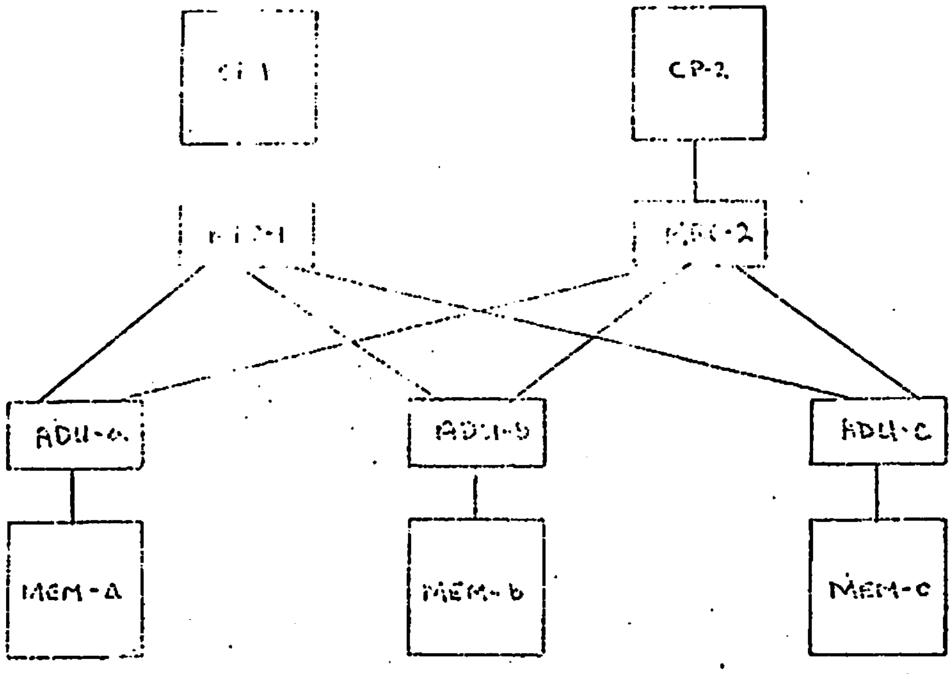


FIGURE 3
CONTROL PROCESSOR
HIGH SPEED MEMORY INTERFACE

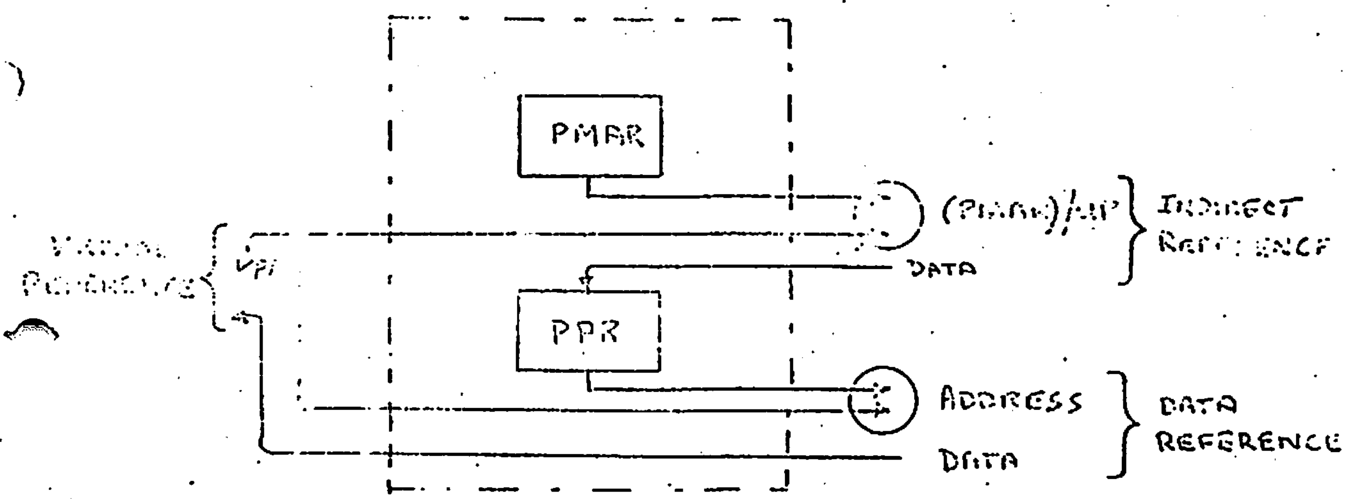


FIGURE 4
DATA AND ADDRESS
FLOW THROUGH MRC

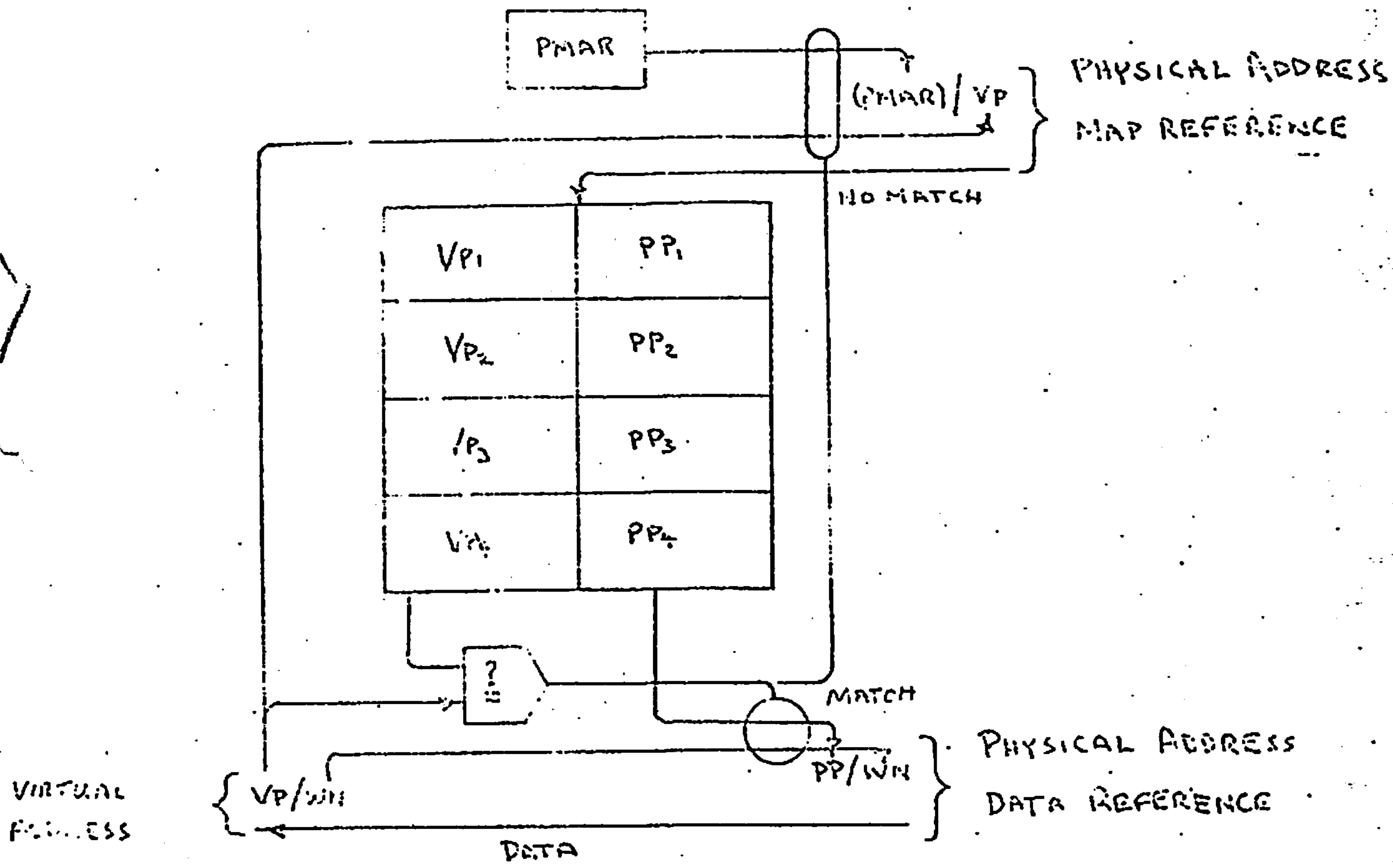


FIGURE 5
DATA AND ADDRESS FLOW
WITH
PAGING AND ASSOCIATIVE
STORE