Burroughs

B 7000

MCP

EP 4055

CLASS HANDOUT

9/79

Printed in U.S.A.

"The names used in this publication are not of individuals living or otherwise. Any similarity or likeness of the names used in this publication with the names of individuals, living or otherwise, is purely coincidental and not intentional."

Burroughs believes that the information described in this publication is accurate and reliable, and much care has been taken in its preparation. However, no responsibility, financial or otherwise, can be accepted for any consequences arising out of the use of this material, including loss of profit, indirect, special, or consequential damages. There are no warranties which extend beyond the program specification.

The Customer should exercise care to assure that use of the information in this publication will be in full compliance with laws, rules and regulations of the jurisdictions with respect to which it is used.

The information contained herein is subject to change. Revisions may be issued from time to time to advise of changes and/or additions.

Correspondence regarding this document should be forwarded directly to Customer Education, Burroughs Corporation, Burroughs Place, Detroit, Michigan 48232.

TABLE OF CONTENTS

COURSE OUTLINE

DAY	TOPIC	MATERIALS
1	Handouts	1. B7000 Advanced System Support Student Materials (EP 6132-1)
		2. B7700 Systems Reference Manual (1060233)
		3. B7000/B6000 Excerpts for Pocket Reference (1083136)
		4. B7000/B6000 ESPOL Information Manual (5000094)
		5. B7700 MCP Mark II.9 Student Text (EP 6132-3)
	B7700 Review	1. Start/Terminate I/O Flowchart (I01-I06)
	Homework Assignment	1. Hardware Quiz (HQ1-HQ3)
2	Review Hardware Quiz	- - - - - -
	ESPOL	1. OBJECT/BRUN (OBJ1-OBJ2)
		2. CODE/STUFF (CDS1-CDS2)
		3. BUZZ (BZ1)
	Homework Assignment	1. ESPOL Quiz (EQ1-EQ3)
3	Review ESPOL Quiz	- - - - - -
	BOOTSTRAP CODE	1. Disk Bootstrap Code (DBC1-DBC2)
		2. CODE (C1)
	Handouts	1. MCP Listings*
	GETITGOING	- - - - - -

*MCP listings include the following portions:

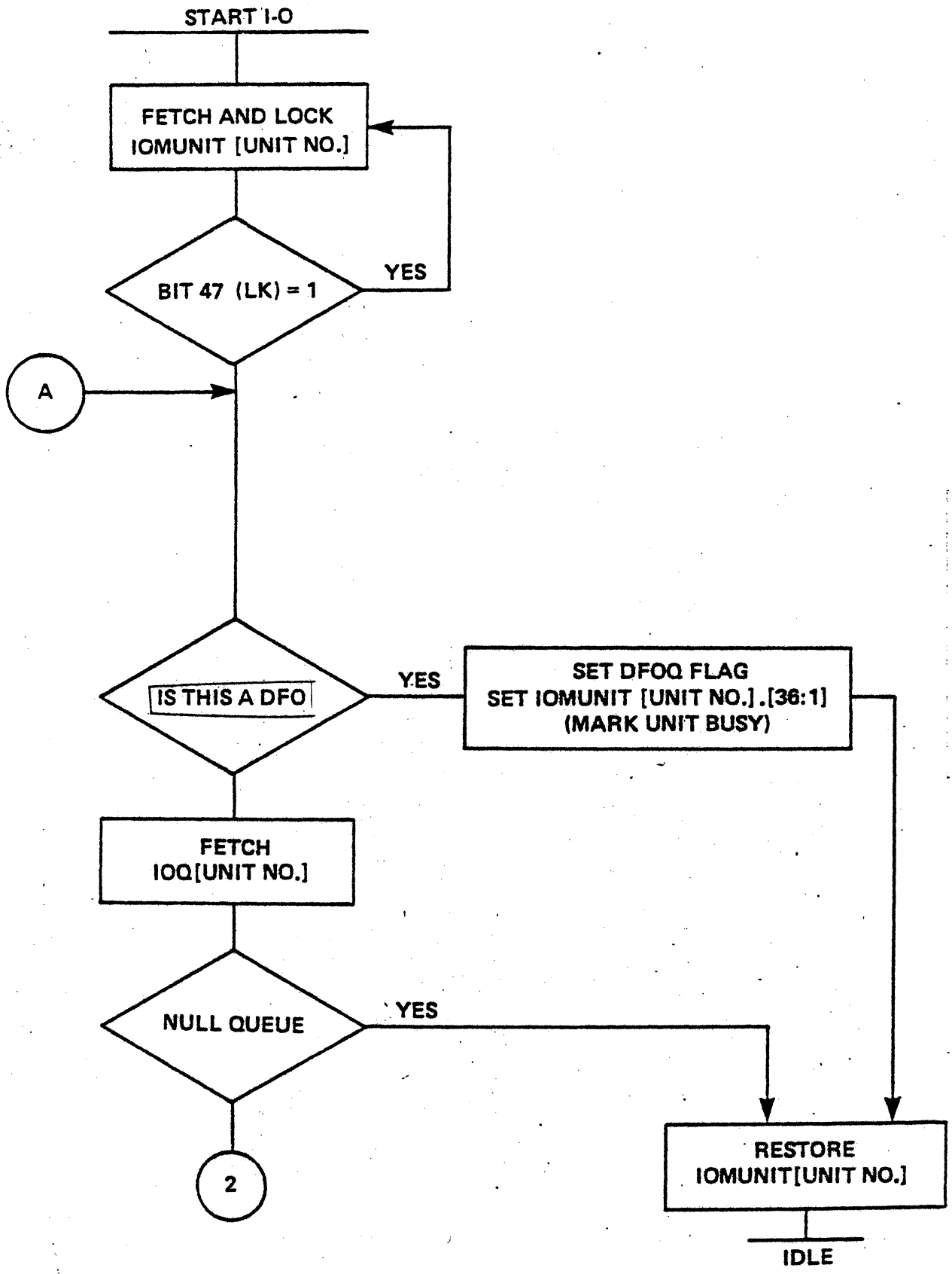
- GLOBALS
- FIBSTACK Procedures
- Process Control Procedures
- Event Oriented Procedures
- Process Communications Procedures
- SOPHIA
- Independent Runner Procedures
- Memory Management
- Initialization Code
- Overlay File Management
- KANGAROO, PROCESSKILL, INTERLO(
- BLOCKEXIT

COURSE OUTLINE (Cont.)

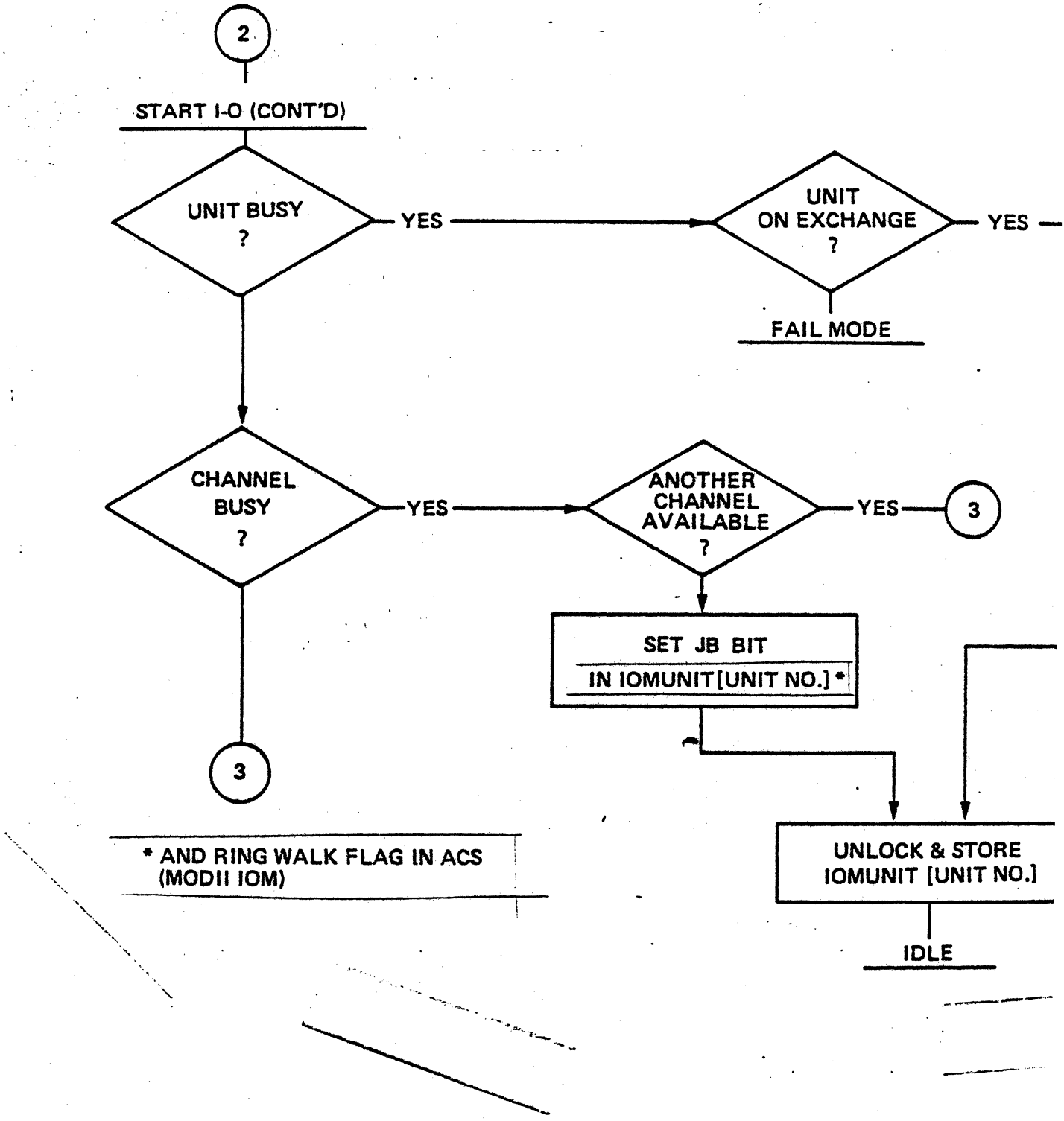
DAY	TOPIC	MATERIALS
4	PRIMARYINITIALIZE	1. H/L Stacks (HLS1-HLS3)
	SECONDARYINITIALIZE	
5	ANABOLISM	- - - - - -
	FORK	1. FORKER (FOR1-FOR2)
	WAIT	- - - - - -
	CAUSE	- - - - - -
	LIBERATE	- - - - - -
	PROCURE	- - - - - -
	GEORGE	- - - - - -
	STACKMOVER	- - - - - -
	DIAL	- - - - - -
	ANSWER	- - - - - -
	SPEAK	- - - - - -
	HANGUP	- - - - - -
	Homework Assignment	1. ESPOL/MCP Lab (EML1-EML3) 2. SYMBOL/RUNNER (RUN1-RUN3)
6	Review ESPOL/MCP Lab	- - - - - -
	Handout	1. Memory Mangement (MM1-MM9) 2. GETSPACE (GET1-GET2)
	GETSPACE	- - - - - -
	FORGETSPACE	- - - - - -
	GETAREA	1. Getarea Forgetarea Links (GFL1)
	FORGETAREA	- - - - - -
	BLOCKEXIT	- - - - - -
	Handout	1. Memory Dump (locally produced at site)

COURSE OUTLINE (Cont.)

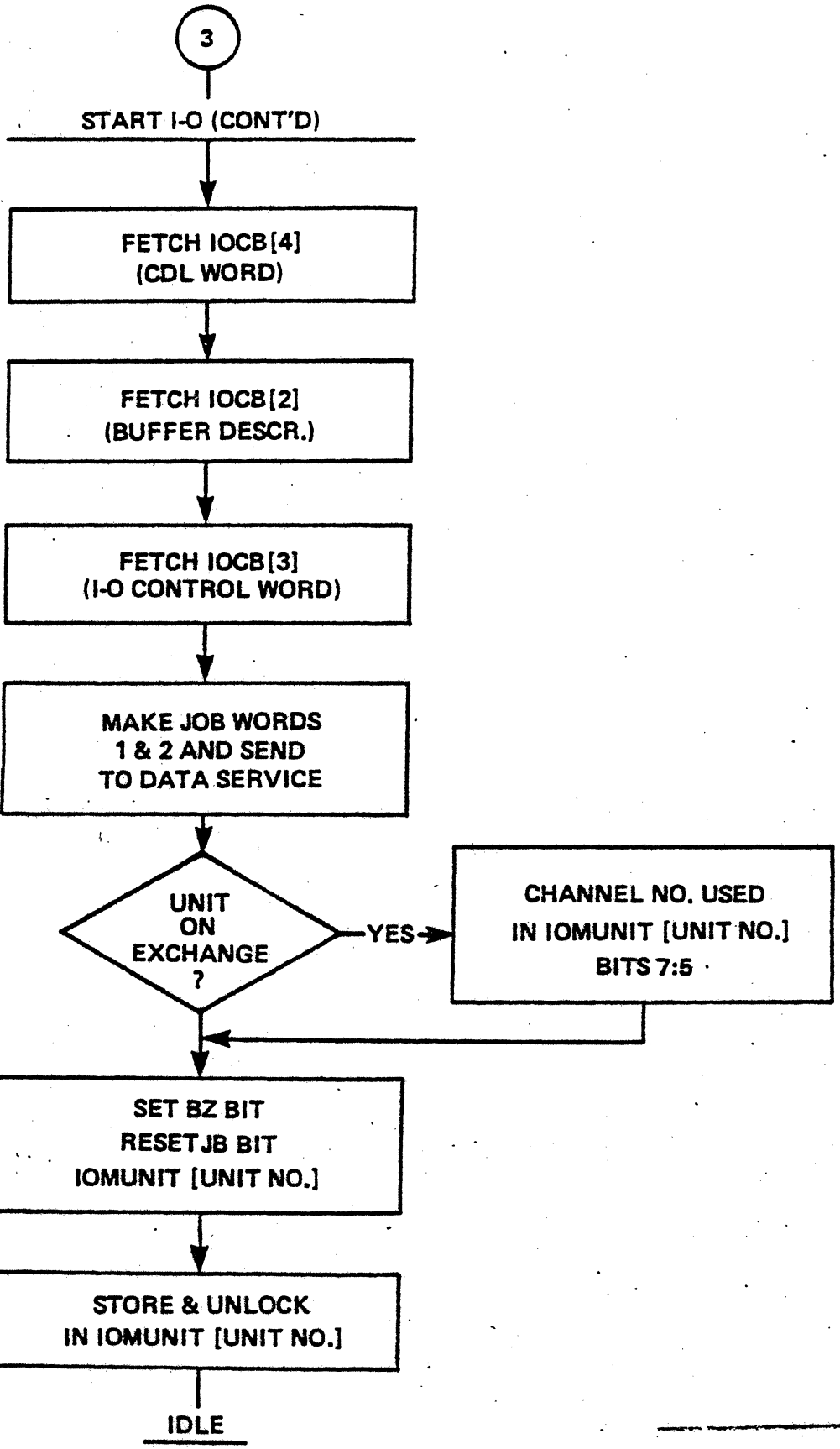
DAY	TOPIC	MATERIALS
7	Handout	1. IPC/STUFF (IPS1-IPS3)
	ALGOL PROCESS CODE	- - - - - -
	DOCTOR	- - - - - -
	INITIATE	- - - - - -
	Lab (4-5 hours)	- - - - - -
8	BOJ	- - - - - -
	EOJ	- - - - - -
	Lab (4-5 hours)	- - - - - -
9	KANGAROO	- - - - - -
	PROCESSKILL	- - - - - -
	INTERLOOPER	- - - - - -
	WSSHERRIFF	- - - - - -
	Lab (4-5 hours)	- - - - - -
10	Handout	1. FINDOLAYSPACE/ LOSEOLAYSPACE (FLS1)
	FINDOLAYSPACE	
	LOSEOLAYSPACE	
	FIBSTACK	1. FIBSTACK/INFO (FSI1-FSI2)
		2. FIBSTACK/INFO Program Dump (FID1-FID3)
	Handout	1. INTX/X (INT1-INT3)



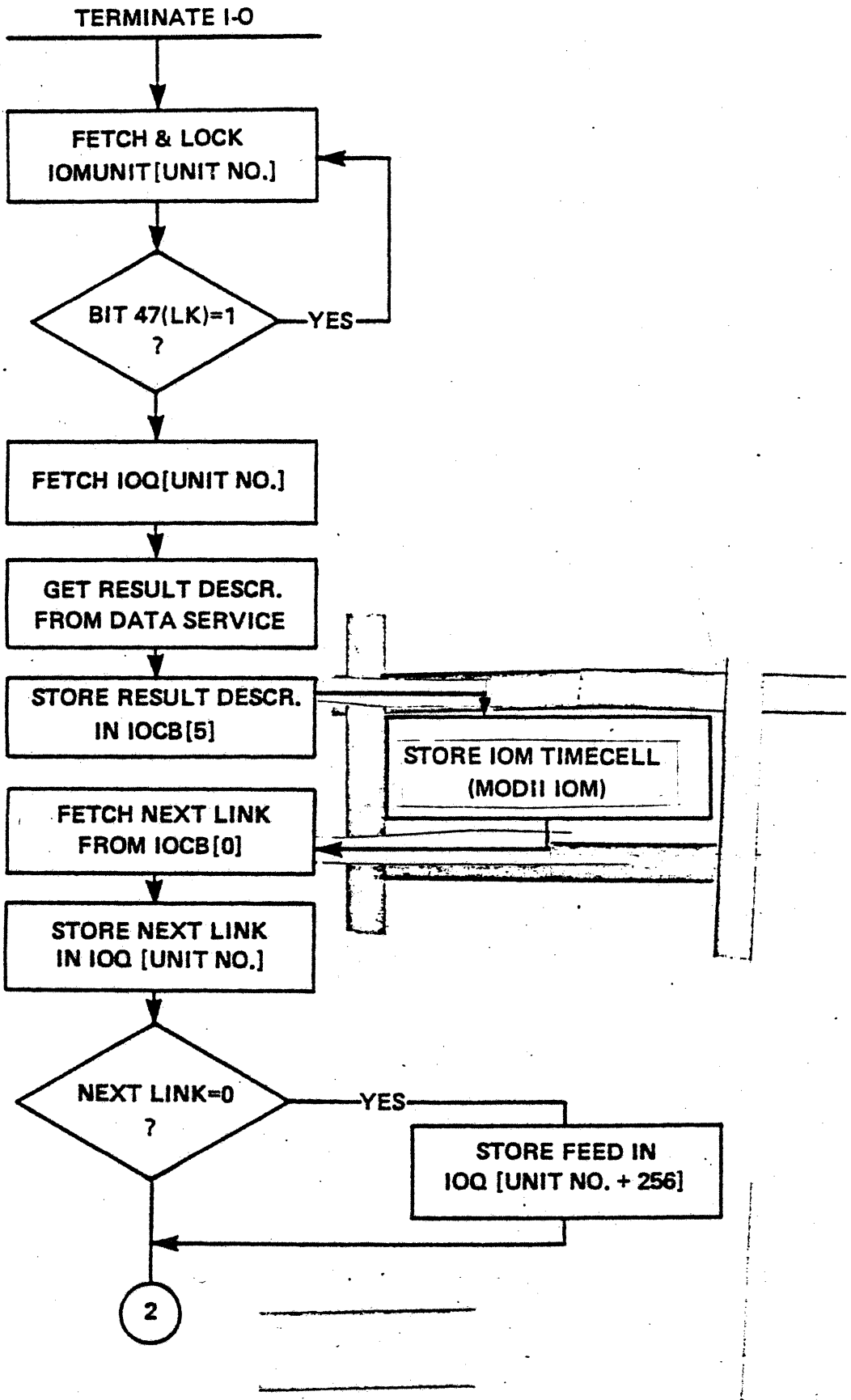
Start I/O Terminate I/O Flowchart (Sheet 1 of 6)



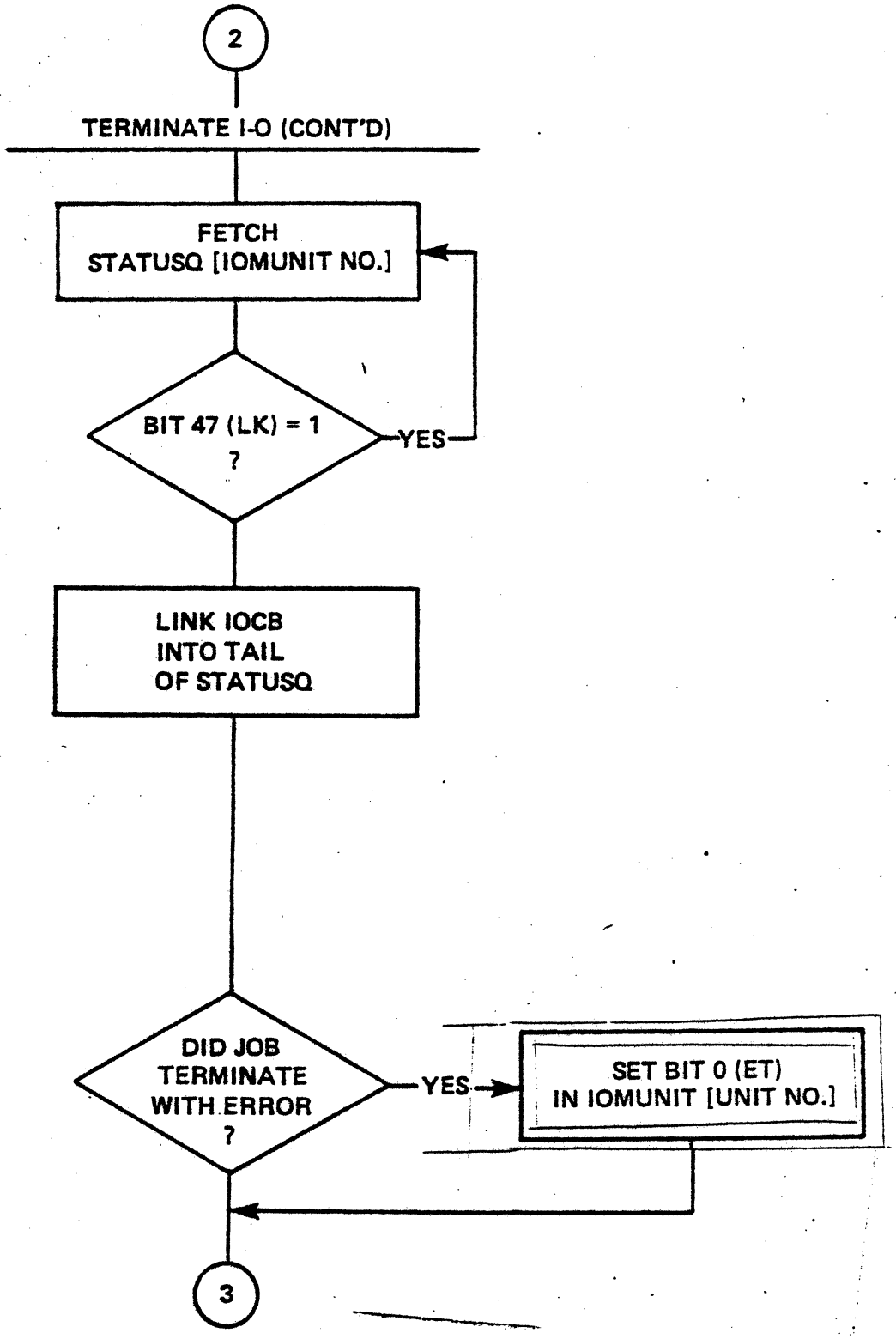
Start I/O Terminate I/O Flowchart (Sheet 2 of 6)



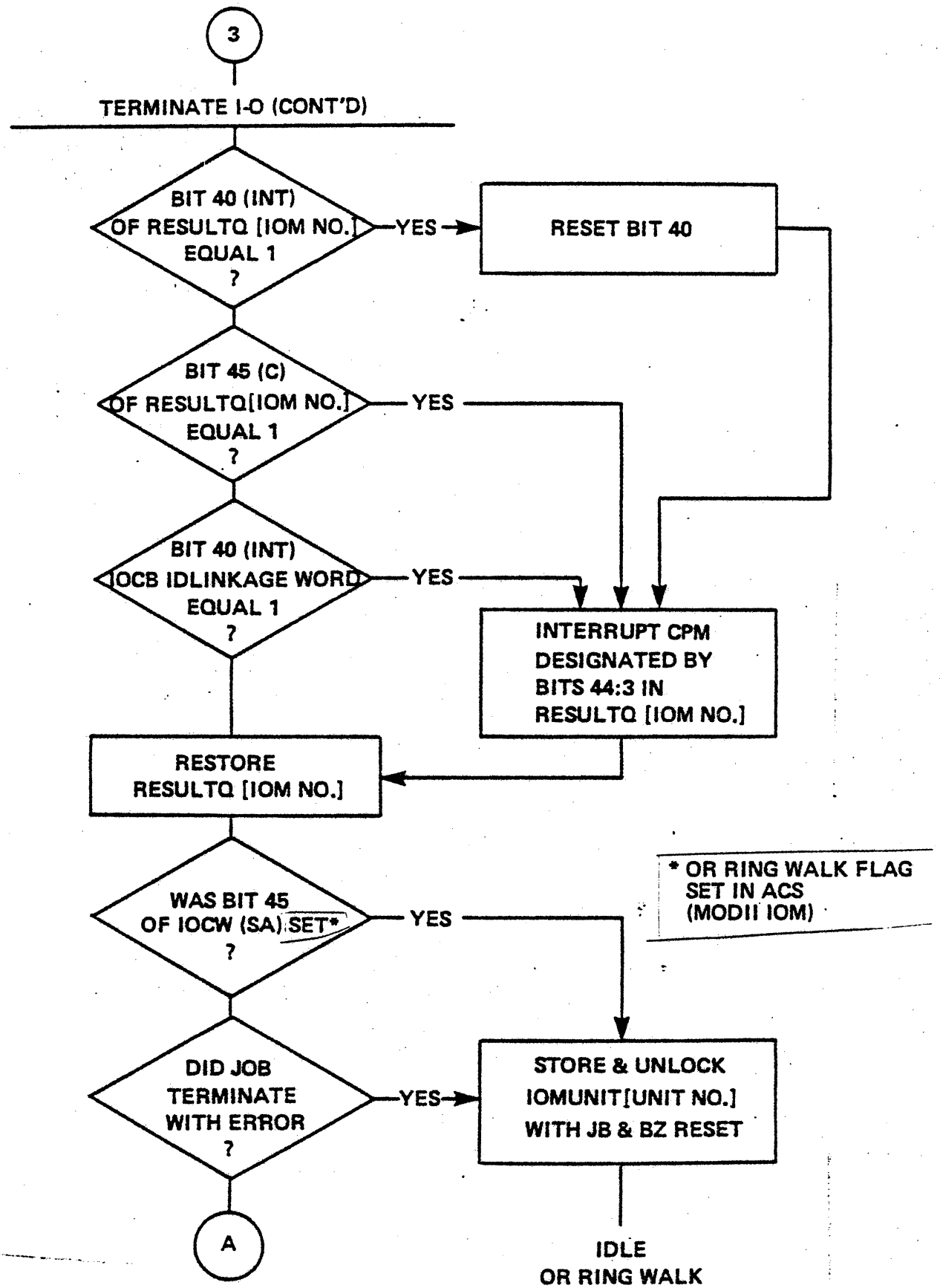
Start I/O Terminate I/O Flowchart (Sheet 3 of 6)



Start I/O Terminate I/O Flowchart (Sheet 4 of 6)



Start I/O Terminate I/O Flowchart (Sheet 5 of 6)



Start I/O Terminate I/O Flowchart (Sheet 6 of 6)

HARDWARE QUIZ

1. A B7700 System may include a maximum of _____ Central Processors and I/O Processors in any combination, and may include a maximum of _____ Memory Modules. The maximum addressable memory for a Logical system however is _____.

2. (T) (F) The B7700 Central Processors operate at a basic clock frequency of 16 Mhz and 8 Mhz; the I/O Processors and Memory operate at 8 Mhz.

3. (T) (F) Differences between the B6700 and B7700 machine language instruction sets require that B6700 programs be re-compiled to run on the B7700.

4. (T) (F) The master system clock obtains its power from the power regulators or Memory Control Module zero.

5. (T) (F) Peripheral Controllers and Disk Controllers for the B7700 are identical to B6700 Peripheral Controllers. These components are housed in B6700 style Peripheral Control Cabinets.

6. (T) (F) The Memory Limits Registers prevent memory accesses by specified requestors.

7. (T) (F) "Phasing" allows 4 words to be transferred to or from memory in parallel (one clock time).

8. (T) (F) Error check bits allow correction of all one-bit and two-bit memory errors.

9. (T) (F) A memory requestor sends an access request to all Memory Controllers on the system. Only that controller which controls the address requested will respond to the request.

10. (T) (F) If a memory requestor makes a memory request for more words than a Memory Module can send (or receive) in one "burst" the requestor must keep count of the number of words it receives (or sends) and keep repeating the access request until the word count is satisfied.

11. What is the maximum number of Peripheral Control Cabinets that can be connected to a B7700 IOM? _____

12. What is the maximum number of peripheral units that can be connected to a B7700 system? _____

13. Are DCP's included in the above count? _____

14. Can two DFI data channels of an IOM be connected to one PC cabine and the other two channels of the same DFI be connected to anothe PC cabinet? _____

15. What is the maximum number of words that an IOM will request fron Memory? _____

16. Does "scan in peripheral status" use the scan bus? _____

17. When the MCP causes an IOM to interrogate peripheral status, what information is delivered to the MCP? _____

18. Where is the above information placed so that the MCP knows where it is? _____

19. When a B7700 system has more than one IOM, does the MCP have to give more than one IOM the "interrogate peripheral status" command in order to get the status of all units? _____

20. What is the significance of requestor number on the MCM? _____

21. Why is memory limited to one million words? _____

22. What is the maximum number of Peripheral Controls that can be connected to an IOM? _____

23. What is the maximum number of Peripheral Controls that can be placed in one PC cabinet? _____

24. Between what modules does the Scan Buss connect? _____

25. Does "interrogate peripheral status" make use of the "IOMUNIT" or "IOQ" arrays? _____

26. Name one thing that can cause a status change interrupt. _____

27. What prevents both the MCP and an IOM from changing an I/O queue linkage word at the same time? _____

28. What prevents both the MCP and an IOM from changing a status queu link word at the same time? _____

29. When two IOM's have paths to the same unit, which IOM will the MCP select to do an I/O operation on the unit? _____

30. When doing asynchronous I/O, how does an IOM know which CPM to interrupt? _____

31. What would the MCP get back from a "BUZZ47" on a unit table word if an IOM has just lock fetched that word? _____

_____

32. What prevents two CPM's from giving the same IOM different commands? _____

33. How does the MCP know whether or not an IOM has responded to the last HA command that was given to it? _____

_____

```
                                        B R U N
                                        = = = =

                        NEW SYMBOLIC:  NEWTAPE ON OLDSPACK.

                $SET NEW LIST STACK                                                                     000:0000:0
                $SET CODE                                                                               000:0000:0
                $SET SEQ                                                                    00001000    000:0000:0
                BEGIN                                                                       00002000    000:0000:0
(01,00002) = 8,0000
(01,00003) = SEGMENT DESCRIPTOR                                                                 8,0002) IS SEGMENT 00003
                                        003:0000:0  NVLD                FF
                REAL R;                                                                 1   00003000    003:0000:1
(02,00002) = R
                BOOLEAN B;                                                              1   00004000    003:0000:1
(02,00003) = B
                PROCEDURE P;                                                            1   00005000    003:0000:1
(02,00004) = P
                   PROGRAMDUMP;                                                         1   00006000    003:0000:1
                                        003:0000:1  MKST                AE
(01,00004) = SEPARATE INTRINSIC
                                        003:0000:2  NAMC  (01,00004)    6004
                                        003:0000:4  LIB                 9202
                                        003:0001:0  ENTR                A8
                                        003:0001:1  EXIT                A3
*************************************** STACK BUILDING CODE FOR LEVEL 03 ***************************************
                IF NOT B THEN                                                               00007000    003:0001:2
                                        003:0001:2  VALC  (02,00003)    1003
                                        003:0001:4  LNOT                92
                   P                                                                        00008000    003:0001:4
                                        003:0002:1  MKST                AE
                                        003:0002:2  NAMC  (02,00004)    5004
                ELSE                                                                        00009000    003:0002:4
                                        003:0002:4  ENTR                A8
                   R := R + 1;                                                              00010000    003:0002:5
                                        003:0001:4  BRTR-LINK 0000:0    A10000
                                        003:0002:5  BRUN-LINK 0000:0    A20000
                                        003:0003:2  VALC  (02,00002)    1002
                                        003:0003:4  ONE                 81
                                        003:0003:5  ADD                 80
                                        003:0004:0  NAMC  (02,00002)    5002
                                        003:0004:2  STOD                98
                                        003:0001:4  BRTR       0003:2   A14003
                R := R + 1;                                                                 00011000    003:0004:3
                                        003:0004:3  VALC  (02,00002)    1002
                                        003:0004:5  ONE                 81
                                        003:0005:0  ADD                 80
                                        003:0005:1  NAMC  (02,00002)    5002
                                        003:0005:3  STOD                98
                END.                                                                        00012000    003:0005:4
                                        003:0005:4  EXIT                A3
                                        003:0002:5  BRUN       0004:3   A26004
*************************************** STACK BUILDING CODE FOR LEVEL 02 ***************************************
                                        003:0005:5  ZERO                80
                                        003:0006:0  ZERO                90
(02,00004)    = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
                                        003:0006:1  MPCW                9F
                                        003:0007            003:0000:1  30020000E303
                                        003:0008:0  PUSH                84
                                        003:0008:1  BRUN       0001:2   A24001
                                        003:0008:4  NVLD                FF
                                        003:0008:5  NVLD                FF
                                                                                        8,0000(003) IS 0009 LONG
================================================================================================================
NUMBER OF ERRORS DETECTED = 0.
NUMBER OF SEGMENTS = 2.  TOTAL SEGMENT SIZE = 9 WORDS.  CORE ESTIMATE = 16 WORDS.  STACK ESTIMATE = 7
PROGRAM SIZE = 14 CARDS, 29 SYNTACTIC ITEMS, 7 DISK SEGMENTS.
PROGRAM FILE NAME:  BRUN. 87700 CODE GENERATED.
COMPILATION TIME = 2.549 SECONDS ELAPSED;  0.277 SECONDS PROCESSING;  0.539 SECONDS I/O.
================================================================================================================
```

OBJ1

```
B7700 PROGRAM DUMP FOR BRUN. (MIX-3944/3945, SIK-008)
MCP:  SYSTEM/MCP., MARK 30.071.755      INTRINSICS:  SYSTEM/INTRINSICS. (LOADED)          SYSTEM SERIAL: #124
CAUSE OF DUMP:  PROGRAM REQUESTED @  003:0001:1, 003:0004:3.
PROGRAMDUMP OPTIONS:  (DEFAULT)

029A = LOSR  (06A2D)

0037 (01,0002)  0 000000 000001  OP:   DCT:00000000 00000001, BCL:00000001, EBC:777777, DEC:1
0036            3 008200 10E003  RCW:  LL=03, NORML STATE                  (USER SEGMENT       @ 0003:0001:1)
                                 SEG DESC:  3 800000 97DF23
                                 CODE:  3 FFAE60 048202 >3 A8A310 03A140< 3 03AE50 04A8A2
0035 ----D[01]=>3 C1A001 C04002 *MSCW: PREVIOUS MSCW @ 0033; D[001]=001C IN STACK 01A

0034            3 018600 40A003  RCW:  LL=02, NORML STATE                  (USER SEGMENT       @ 0003:0004:3)
                                 SEG DESC:  3 800000 97DF23
                                 CODE:  3 A9A310 03A140  3 03AE50 04A8A2  3 600410 028180 >3 500280 100281< 3 805002 88A380
0033 ----D[03]=>3 4D8002 E0C005 *MSCW: PREVIOUS MSCW @ 002E; D[021]=002E

0032 (02,0004)  7 0D8200 00E003  PCW:  LL=03, D[1] SEGMENT @ 0003:0000:1, NORML STATE
0031 (02,0003)  0 000000 000000
0030 (02,0002)  0 000000 000000
002F            3 000000 088451  RCW:  LL=02, CNTRL STATE                  (MCP SEGMENT        @ 0451:0000:0 (25066220))
                                 SEG DESC:  3 80003E A38E24
                                 CODE: >3 823482 269588< 3 820280 879587
002E ----D[02]=>3 CD9001 608002 *MSCW: PREVIOUS MSCW @ 002C; D[011]=0016 IN STACK 0D9

002D            3 000000 002000  RCW:  DUMMY (RUN)
002C ----D[02]=>3 C08919 D08028 *MSCW: PREVIOUS MSCW @ 0001; D[011]=919D IN STACK 008

0000 = BOSR  (06793)
```

OBJ2

C O D E / S T U F F

```
VERSION:6.3.000 (OBJECT PROGRAM)

                BEGIN                                                                   005:0000:0
                  REAL A,B,C,D;                                                 1       005:0000:0
A=(0, D0)
B=(0, D1)
C=(0, D2)
D=(0, D3)
                  WORD W,WP;                                                            005:0001:0
W=(0, D4)
WP=(0, D5)
                  PROCEDURE P = WP (AA,BB);                                             005:0002:0
P=(0, D5)
                  VALUE AA,BB;                                                          005:0000:0
                  REAL AA,BB;                                                           005:0000:0
                  NULL;                                                                 005:0000:0
AA=(1, 2)
BB=(1, 3)
                  PROCEDURE (P1);                                                       005:0000:0
                  NULL;                                                                 005:0000:0
                  PROCEDURE (P2) (AA,BB);                                               005:0000:0
                  VALUE AA,BB;                                                          005:0000:0
                  REAL AA,BB;                                                           005:0000:0
                  NULL;                                                                 005:0000:0
AA=(1, 2)
BB=(1, 3)
                  P(A,B);                                                               005:0000:0
                          005:0000:0  MKST                AE
                          005:0000:1  NAMC  (00,00005)    4005
                          005:0000:3  VALC  (00,00000)    0000
                          005:0000:5  VALC  (00,00001)    0001
                          005:0001:1  ENTR                AB
                  P1 (WP);                                                              005:0001:2
                          005:0001:2  MKST                AE
                          005:0001:3  NAMC  (00,00005)    4005
                          005:0001:5  LODI                958C
                          005:0002:1  ENTR                AB
                  P1 (4"0023" & 1 TAG);                                                 005:0002:2
                          005:0002:2  MKST                AE
                          005:0002:3  NAMC  (00,00023)    4023
                          005:0002:5  ENTR                AB
                  P1 (4"4120030000020" & 1 TAG);                                        005:0003:0
                          005:0003:0  MKST                AE
                          005:0003:1  LIT8                8E
                          005:0004                        4120030000020 2022009300000040

                          005:0005:0  ONE                 81
                          005:0005:1  STAG                9584
                          005:0005:3  ENTR                AB
                  P2 (W) (C,D);                                                         005:0005:4
                          005:0005:4  MKST                AE
                          005:0005:5  NAMC  (00,00004)    4004
                          005:0006:1  LODI                958C
                          005:0006:3  VALC  (00,00002)    0002
                          005:0006:5  VALC  (03,00003)    0003
                          005:0007:1  ENTR                AB
                  P2 (WP) (C,D);                                                        005:0007:2
                          005:0007:2  MKST                AE
                          005:0007:3  NAMC  (00,00005)    4005
                          005:0007:5  LODI                958C
                          005:0008:1  VALC  (00,00002)    0002
                          005:0008:3  VALC  (00,00003)    0003
                          005:0008:5  ENTR                AB
                  P2 (4"0003" & 1 TAG) (0,D);                                           005:0009:0
                          005:0009:0  MKST                AE
                          005:0009:1  NAMC  (00,00003)    4003
                          005:0009:3  ZERO                80
                          005:0009:4  VALC  (00,00003)    0003
                          005:000A:0  ENTR                AB
                  (NAME(WP))':='W;                                                      005:0004:1
                          005:000A:1  NAMC  (00,00005)    4005
```

CDS1

```
                        005:000A:3  NAMC  (00.00004)   40D4
                        005:000A:5  LOUT               959C
                        005:000B:1  EXCH               86
                        005:000B:2  OVRD               BA
          [4"0003" & 1 TAG]:=WP;                                                005:0009:3
                        005:000B:3  NAMC  (00.00003)   4003
                        005:000B:5  NAMC  (00.000D5)   40D5
                        005:000C:1  LOUT               959C
                        005:000C:3  EXCH               86
                        005:000C:4  OVRD               BA
          [4"00C0" & 1 TAG]:=A;                                                 005:000C:5
                        005:000C:5  NAMC  (00.000C0)   40C0
                        005:000D:1  VALC  (00.000D0)   0000
                        005:000D:3  STOD               BE
     END.                                                                       005:000D:4
                        005:000D:4  NVLD               FF
                        005:000D:5  NVLD               FF
PCW = (0.0003) [7 00000000A0051
COMPILE 0. K.
CORE SIZE = 228 WORDS.
DO-STACK SIZE = (1 + 208) WORDS.
PROGRAM HAD 24 CARD IMAGES. WITH 156 SYNTACTIC ITEMS.
PROGRAM FILE NAME: CODE/STUFF COMPILED ON THE B6800 FOR THE B7800      (VERSION: 0.0.000)
NO CALLS ON BLOCKEXIT.
COMPILATION TIMES WERE:
                    00004.591 SECONDS ELAPSED
                    00000.402 SECONDS PROCESS
                    00000.671 SECONDS I-O
```

CDS2

```
BUZZ(X)
   DEXI
   NAMC  X
   ONE
   RDLK
   BRFL      OUT OF LOOP - LOCKED
   ONE
   CBON
   SCRF (2)
   BRFL
```

```
$ SET READLOCKTIMEOUT
BUZZ(X)
   DEXI
   ONE
   LT8 35 }  READ UP
   RPRR   }    SNR
   INSR     [29:10]
   NAMC   X
   STFF          | SIRW X
   DUPL          | SNR  1
   RSDN AFTER -> | SIRW X
   RDLK
   DUPL                    | FB
   BRFL                    | SIRW
   IMKS AFTER ------------> | MSCW
   NAMC   (0,C2) READLOCKTIMEOUT (26642)
   RSDN          AFTER
   ZERO
   ENTR          | 0
   DUPL          | FB
   DLET          | SIRW X
   DLET          | IRW 0,C2
                 | MSCW
                                | FB
                                | SIRW X
                                | IRW 0,C2
                                | MSCW
```

ESPOL QUIZ

1. Consider the following:

```
PROCEDURE X;
BEGIN
   ARRAY A[4];
   REAL I, J, K;
   FIELD PBITF = 47:1;
   A: = A & 1 PBITF;
   I: = 1;
   J: = 2;
   K: = 3;
   EXIT;
END;
```

(A) What does array A contain? _____

(B) What does DSCR for array A look like? _____
(Remember copy DSCR action)

2. What code is generated by the following?

(A) REAL X; _____

(B) X: = EXCHANGE (TOPOFSTACK); _____

(C) EXCHANGE (TOPOFSTACK); _____

(D) TOPOFSTACK: = EXCHANGE (TOPOFSTACK); _____

(E) TOPOFSTACK: = X: = 0; _____

(F) STOP(0,X); _____

(G) EXCHANGE (TOPOFSTACK,X); _____

(H) TOPOFSTACK: = EXCHANGE (TOPOFSTACK,X) +DUPLICATE; _____
_____

(I) TOUCH (EXCHANGE(DUPLICATE)); _____

(J) TOUCH (EVAL(TOPOFSTACK)); _____

(K) BOOLEAN B; _____

(L) X: = REAL (B); _____

(M) B: = BOOLEAN(X); _____

(N) WORD W; _____

(O) GO TO W; _____

3. Which of the following are syntactically illegal and why?

(A) SAVE ARRAY SA[*];

(B) LAYOUT L(3:1: = 1, FLD = 2:1, 1:1: = 1, BZERO = 0:1: = 1);

(C) FILE F(MYUSE = IN);

(D) WORD W; REAL R;

(E) GO TO W;

(F) F.OPEN: = TRUE;

(G) MYSELF.STATUS: = -1;

(H) W: = 1 & 5 TAG & L(4,,*);

(I) R: = 0 & 2 TAG;

(J) [4"80000FF000AC" & 5 TAG] : = W;

4. Consider the following:

```
PROCEDURE P(A,B);
VALUE A,B;
WORD A,B;
BEGIN
   WORD W;
   PROCEDURE (Q);
   BEGIN
      A: = B;
      EXIT;
   END;
   W: = MAKEPCW(Q);
   Q[IF M[0]THEN NAME(A]
      ELSE (IF M[1] THEN NAME (B)
      ELSE NAME(W))];
END;
```

(A) What is the effect of the above procedure?

5. What details must be taken care of by the stand-alone ESPOL programmer (EG.,BLOCKEXIT)?

6. What are the differences (in effect) between the following sets of code?

(A)
```
WORD W;
W:=0;
```
and
```
REAL R;
R:=0;
```

(B)
```
REPLACE P BY 0
FOR 10 WORDS;
```
and
```
REPLACE P BY 0
FOR 10 OVERWRITE;
```

(C)
```
PROCEDURE P(X);
REAL X;
X: = 0;
```
and
```
PROCEDURE P(X);
VALUE X; REAL X;
X: = 0;
```

7. Define the following:

(A) SAVE _____

(B) M _____

(C) FIELD _____

(D) SCANIN _____

(E) SCANOUT _____

(F) HEYOU _____

(G) SIRW _____

(H) PCW _____

(I) MSCW _____

(J) DATA DESCRIPTOR _____

(K) COPY DESCRIPTOR ACTION _____

(L) RCW _____

DISK BOOTSTRAP CODE

HARDLOAD RD - AT HA 4

16:17 - ERROR FIELD
24:8 - UNIT NUMBER
32:5 - CHANNEL NUMBER

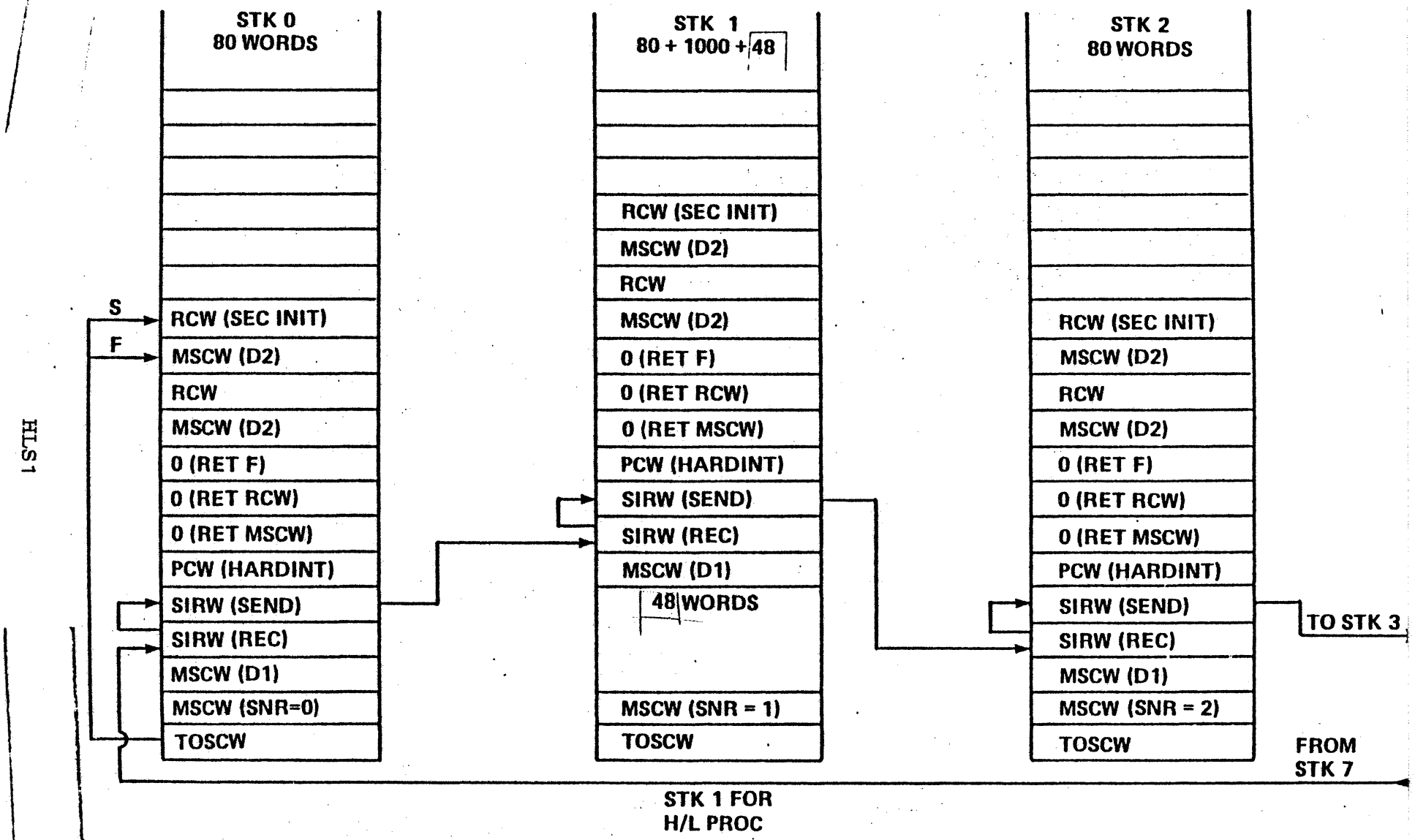
MEMORY ADDRESS	MEMORY CONTENTS
0	0 8A0000000003
1	3 800188B00049
2	0 A0008C30001C
3	7 000682584001
4	0 000000000040
5	0 000189400040
6	0 12200000134C
7	0 510100026372
8	3 B0B095B9B234
9	3 B31E0095B9B2
A	3 35B095B9B230
B	3 B095B9B225B0
C	3 95B9AE400300
D	3 000004981B20
E	3 054000BB0007
F	3 B00004982714
10	3 0491400795BA
11	3 B5B22895B895
12	3 8F95840005A1
13	3 6012B08EB203
14	3 95B44000BAAB
15	0 055660025680

DISK BOOTSTRAP CODE

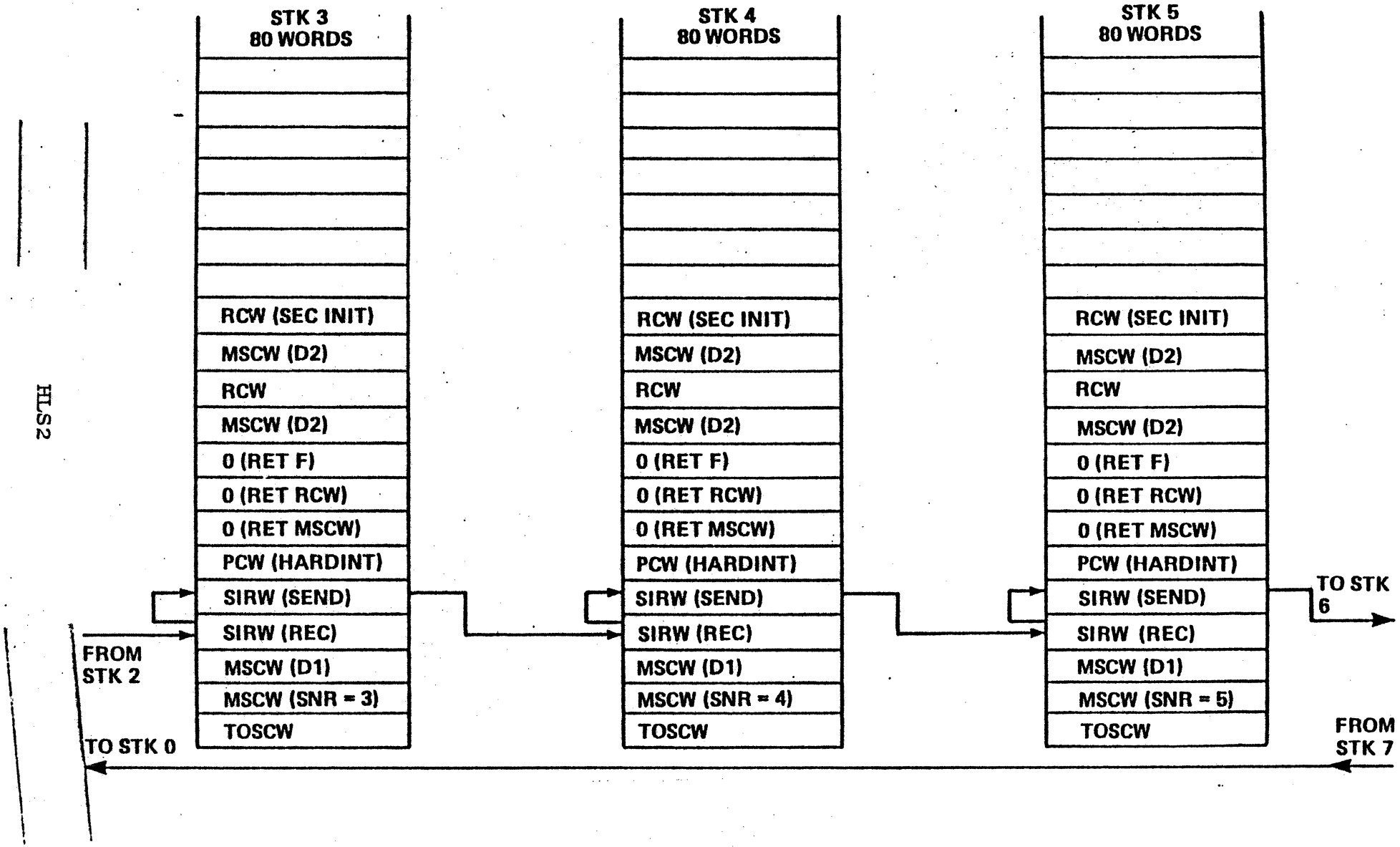
MEMORY ADDRESS	MEMORY CONTENT
0	HA WORD ZERO SYNC I/O
1	SEGMENT DESCRIPTOR ADDRESS 4"49" LENGTH 4"188B"
2	INFORMATION FOR MCP
3	PCW LL 1 CNTL 1:825:3
4	HARDLOAD R/D
5	BUFF DESCRIPTOR ADDRESS 4"40" LENGTH 4"1894"
6	IOCW
7	CDL
8	ZERO ZERO SPRR LT8 34
9	LT16 1E00 SPRR LT8 35
A	ZERO SPRR LT8 30
B	ZERO SPRR LT8 25 ZERO
C	SPRR MKST NAMC 0,3 VALC 0,0
D	VALC 0,4 FLTR 27:32:5
E	NAMC 0,0 OVRN VALC 0,7
F	ZERO VALC 0,4 FLTR 39:20:4
10	LOR NAMC 0,7 RDLK
11	DELT LT8 28 RPRR
12	INCN PAUS VALC 0,5
13	BRTR 12:3 ZERO CHSN LT8 3
14	STAG NAMC 0,0 OVRD ENTR
15	LENGTH OF SAVE STUFF AND ADDRESS OF SEGMENT DICTIONARY

CODE

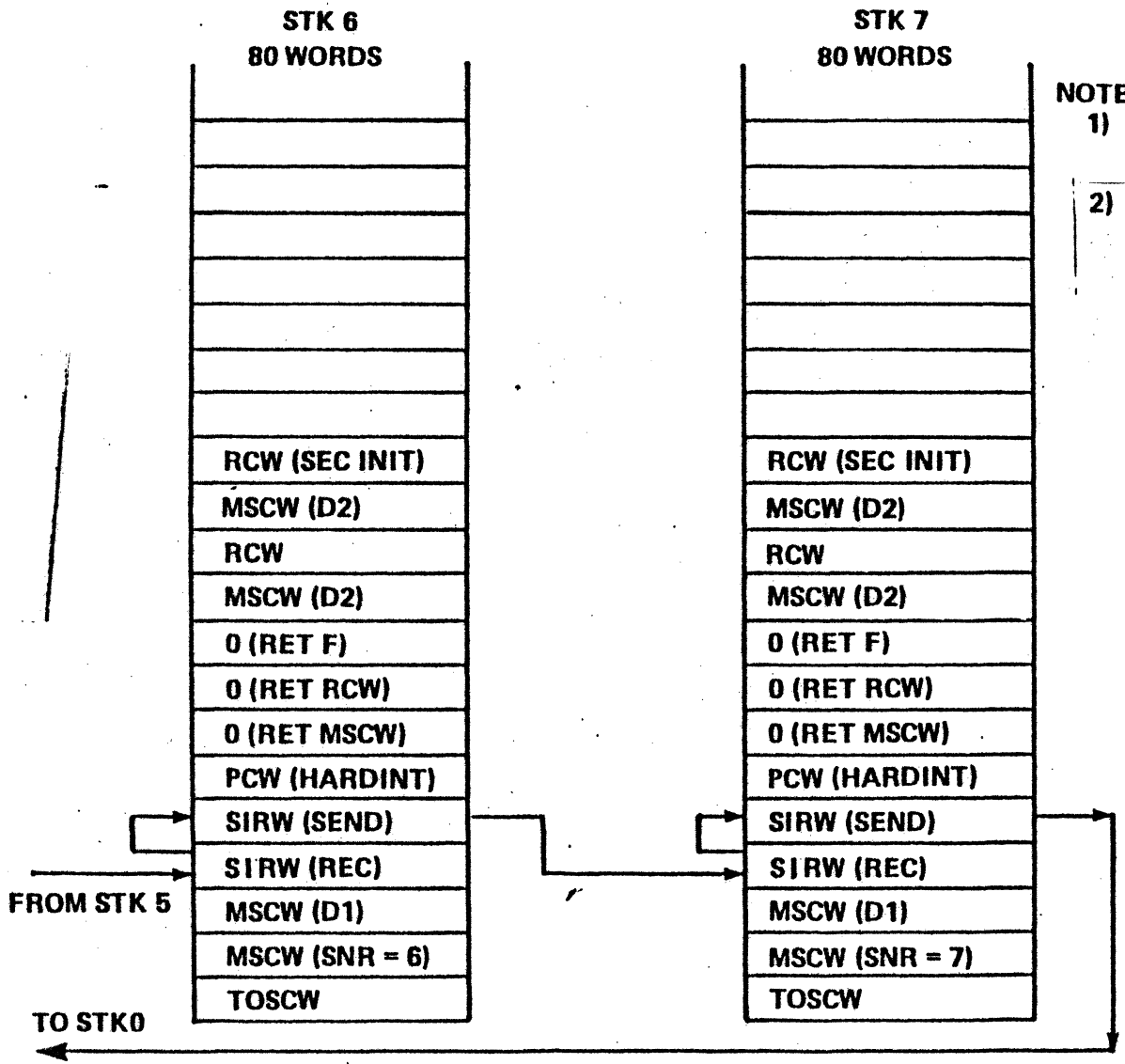
```
VERSION:0.0.000 (OBJECT PROGRAM)

                BEGIN                                                                   00001000  005:0000:0
                  REAL X;                                                           1   00002000  005:0000:0
X=(0, D0)
                  BOOLEAN B;                                                            00003000  005:0000:0
B=(0, D1)
                  WORD W;                                                               00004000  005:0000:0
W=(0, D2)
                  X:=EXCHANGE(TOPOFSTACK);                                              00005000  005:0000:0
                                005:0000:0   EXCH                   86
                                005:0000:1   NAMC  (00,000D0)       4000
                                005:0000:3   STOD                   8E
                  EXCHANGE(TOPOFSTACK);                                                 00006000  005:0000:4
                                005:0000:4   EXCH                   86
                                005:0000:5   DLET                   85
                  TOPOFSTACK:=EXCHANGE(TOPOFSTACK);                                     00007000  005:0001:0
                                005:0001:0   EXCH                   86
                  TOPOFSTACK:=X:=0;                                                     00008000  005:0001:1
                                005:0001:1   ZERO                   80
                                005:0001:2   NAMC  (00,000D0)       4000
                                005:0001:4   STON                   89
                  STOP(0,X);                                                            00009000  005:0001:5
                                005:0001:5   ZERO                   80
                                005:0002:0   VALC  (00,000D0)       0000
                                005:0002:2   EXCH                   86
                                005:0002:3   HALT                   DF
                                005:0002:4   DLET                   95
                                005:0002:5   DLET                   85
                  EXCHANGE(TOPOFSTACK,X);                                               00010000  005:0003:0
                                005:0003:0   NAMC  (00,000D0)       4000
                                005:0003:2   RDLK                   958A
                                005:0003:4   DLET                   85
                  TOPOFSTACK:=EXCHANGE(TOPOFSTACK,X)+DUPLICATE;                         00011000  005:0003:5
                                005:0003:5   NAMC  (00,000D0)       4000
                                005:0004:1   RDLK                   958A
                                005:0004:3   DUPL                   87
                                005:0004:4   ADD                    80
                  TOUCH(EXCHANGE(DUPLICATE));                                           00012000  005:0004:5
                                005:0004:5   DUPL                   87
                                005:0005:0   EXCH                   86
                                005:0005:1   DLET                   85
                  TOUCH(EVAL(TOPOFSTACK));                                              00013000  005:0005:2
                                005:0005:2   EVAL                   AC
                                005:0005:3   DLET                   85
                  X:=REAL(B);                                                           00014000  005:0005:4
                                005:0005:4   VALC  (00,000D1)       0001
                                005:0006:0   NAMC  (00,000D0)       4000
                                005:0006:2   STOD                   8E
                  B:=BOOLEAN(X);                                                        00015000  005:0006:3
                                005:0006:2   STON                   89
                                005:0006:3   NAMC  (00,000D1)       4001
                                005:0006:5   STOD                   8E
                  GO TO W;                                                              00016000  005:0007:0
                                005:0007:0   NAMC  (00,000D2)       4002
                                005:0007:2   DBUN                   AA
                END.                                                                    00017000  005:0007:3
                                005:0007:3   NVLD                   FF
                                005:0007:4   NVLD                   FF
                                005:0007:5   NVLD                   FF
PCW 3 (0,0003) [7 0000000840005]
COMPILE O. K.
CORE SIZE = 219 WORDS.
DO-STACK SIZE = (1 * 208) WORDS.
PROGRAM HAD 17 CARD IMAGES, WITH 96 SYNTACTIC ITEMS.
PROGRAM FILE NAME: CODE COMPILED ON THE B6800 FOR THE B7800      (VERSION: 0.0.000)
NO CALLS ON BLOCKEXIT.
COMPILATION TIMES WERE:
                        00003.967 SECONDS ELAPSED
                        00000.371 SECONDS PROCESS
                        00000.554 SECONDS I-O
```


STK 0
80 WORDS
STK 1
80 + 1000 + 48
STK 2
80 WORDS
S
F
RCW (SEC INIT)
MSCW (D2)
RCW
MSCW (D2)
0 (RET F)
0 (RET RCW)
0 (RET MSCW)
PCW (HARDINT)
SIRW (SEND)
SIRW (REC)
MSCW (D1)
MSCW (SNR=0)
TOSCW
RCW (SEC INIT)
MSCW (D2)
RCW
MSCW (D2)
0 (RET F)
0 (RET RCW)
0 (RET MSCW)
PCW (HARDINT)
SIRW (SEND)
SIRW (REC)
MSCW (D1)
48 WORDS
MSCW (SNR = 1)
TOSCW
RCW (SEC INIT)
MSCW (D2)
RCW
MSCW (D2)
0 (RET F)
0 (RET RCW)
0 (RET MSCW)
PCW (HARDINT)
SIRW (SEND)
SIRW (REC)
MSCW (D1)
MSCW (SNR = 2)
TOSCW
TO STK 3
FROM
STK 7
STK 1 FOR
H/L PROC

HLS1


STK 3
80 WORDS
STK 4
80 WORDS
STK 5
80 WORDS
RCW (SEC INIT)
MSCW (D2)
RCW
MSCW (D2)
0 (RET F)
0 (RET RCW)
0 (RET MSCW)
PCW (HARDINT)
SIRW (SEND)
SIRW (REC)
MSCW (D1)
MSCW (SNR = 3)
MSCW (SNR = 4)
MSCW (SNR = 5)
TOSCW
FROM STK 2
TO STK 0
TO STK 6
FROM STK 7

HLS2



NOTE:

1) ONLY H/L PROC HAS FIXED PART OF STACK AND A TASK (TASKDESC)

2) ALL TO SCW's SET UP SUCH THAT S →RCW SEC. INIT., F →D2 MSCW.

HLS3

```
VERSION:0.0.000 (OBJECT PROGRAM)

                BEGIN                                                                   000:0000:0
                  REAL A,B;                                                    1        005:0000:0
A=(0, 00)
B=(0, 01)
                  POINTER P;                                                            005:0000:0
P=(0, 02)
                  DEFINE                                                                005:0000:0
                     STACKSIZE = 500 #,                                                 005:0000:0
                     PRIORITY = 99 #,                                                   005:0000:0
                     VISIBLE = - #,                                                     005:0000:0
                     ONEONLY = - #;                                                     005:0000:0
                  PROCEDURE MYMCPIR(AA,BB);                                             005:0000:0
MYMCPIR=(0, 03)
                  VALUE AA,BB;                                                          005:0000:0
                  REAL AA,BB;                                                           005:0000:0
                  NULL;                                                                 005:0000:0
AA=(1, 2)
BB=(1, 3)
                  PROCEDURE MCPIRWITHLONGNAME;                                          005:0000:0
MCPIRWITHLONGNAME=(0, 04)
                  NULL;                                                                 005:0000:0
                  FORK MYMCPIR(A,B) [STACKSIZE,PRIORITY];                               005:0000:0
                                005:0000:0  MKST                  AE
                                005:0000:1  NAMC  (00,000D3)      40D3
                                005:0000:3  VALC  (00,00000)      00D0
                                005:0000:5  VALC  (00,00001)      00D1
                                005:0001:1  MKST                  AE
                                005:0001:2  NAMC  (00,000AF)      40AF
                                005:0001:4  LIT6                  B301F4
                                005:0002:1  LIT8                  8263
                                005:0002:3  LIT48                 BE
                                005:0003                          07D4E8D4C3D7 0175235065141727

                                005:0004:0  LIT6                  B3C9D9
                                005:0004:3  ISOL                  9A0F30
                                005:0005:0  JOIN                  9542
                                005:0005:2  ENTR                  AB
                                005:0005:3  ENTR                  AB
                  REPLACE P BY 48"11","THEMCPIRNAMEONSPO";                              005:0005:4
                                005:0005:4  NAMC  (00,00002)      40D2
                                005:0006:0  LOAD                  8D
                                005:0006:1  LIT8                  8288
                                005:0006:3  ISOL                  9A0A30
                                005:0007:0  ONE                   B1
                                005:0007:1  TUNU                  EE
                                005:0007:2  BLET                  B5
=(0, 05)
                                005:0007:3  ZERO                  B0
                                005:0007:4  NAMC  (00,00005)      40D5
                                005:0008:0  INDX                  A6
                                005:0008:1  LIT8                  8203
                                005:0008:3  INSR                  9C2A03
                  FORK MYMCPIR(A,B) [ONEONLY,STACKSIZE,PRIORITY,P];                     005:0009:0
                                005:0009:0  LIT8                  8211
                                005:0009:2  TUND                  E6
                                005:0009:3  MKST                  AE
                                005:0009:4  NAMC  (00,000D3)      40D3
                                005:000A:0  VALC  (00,00000)      00D0
                                005:000A:2  VALC  (00,00001)      00D1
                                005:000A:4  MKST                  AE
                                005:000A:5  NAMC  (00,000AF)      40AF
                                005:000B:1  LIT6                  B301F4
                                005:000B:4  CHSN                  BE
                                005:000B:5  LIT8                  8263
                                005:000C:1  NAMC  (00,00002)      40D2
                                005:000C:3  LOAD                  8D
                                005:000C:4  ZERO                  90
                                005:000C:5  ENTR                  AB
                                005:000D:1  ENTR                  AB

FORKED (Sheet 1 of 2)
```

FOR1

```
                REPLACE P BY 48"09","MCPIRNAME";                                                    005:0009:1
                                005:000D:1  NAMC  (00,00002)   40D2
                                005:000D:3  LOAD               8D
                                005:000D:4  LIT8               8290
                                005:000E:0  ISOL               9A0830
                                005:000E:3  ONE                81
                                005:000E:4  IUNU               EE
                                005:000E:5  DLET               85
                                005:000F:0  LIT8               8203
                                005:000F:2  NAMC  (00,00005)   40D5
                                005:000F:4  INDX               A6
                                005:000F:5  LIT8               8203
                                005:0010:1  INSR               9C2A03
                FORK MCPIRWITHLONGNAME [STACKSIZE,VISIBLE PRIORITY,P];                              005:0010:4
                                005:0010:4  LIT8               8209
                                005:0011:0  IUND               E6
                                005:0011:1  MKST               AE
                                005:0011:2  NAMC  (00,00004)   40D4
                                005:0011:4  MKST               AE
                                005:0011:5  NAMC  (00,000AF)   40AF
                                005:0012:1  LIT16              8301F4
                                005:0012:4  LIT8               8263
                                005:0013:0  CHSN               BE
                                005:0013:1  NAMC  (00,00002)   40D2
                                005:0013:3  LOAD               8D
                                005:0013:4  ZERO               80
                                005:0013:5  ENTR               A4
                                005:0014:0  ENTR               A8
                FORK MCPIRWITHLONGNAME [ONEONLY STACKSIZE,VISIBLE PRIORITY];                        005:0014:1
                                005:0014:1  MKST               AE
                                005:0014:2  NAMC  (00,00004)   40D4
                                005:0014:4  MKST               AE
                                005:0014:5  NAMC  (00,000AF)   40AF
                                005:0015:1  LIT16              8301F4
                                005:0015:4  CHSN               BE
                                005:0015:5  LIT8               8263
                                005:0016:1  CHSN               BE
                                005:0016:2  LIT8               8205
                                005:0016:4  NAMC  (00,00005)   40D5
                                005:0017:0  INDX               A6
                                005:0017:1  BSET               962A
                                005:0017:3  ZERO               80
                                005:0017:4  ENTR               A8
                                005:0017:5  ENTR               A8
                END.                                                                                005:001F:0
                                                                CONSTANT POOL 3(0,0005) IS 0009 WORDS LONG
PCW 2 (0,0003) [7 0000000040051
COMPILE 0. K.
CORE SIZE = 238 WORDS.
DO-STACK SIZE = (3 * 208) WORDS.
PROGRAM HAD 21 CARD IMAGES, WITH 113 SYNTACTIC ITEMS.
PROGRAM FILE NAME: FORKER COMPILED ON THE B6800 FOR THE B7800      (VERSION: 0.0.000)
NO CALLS ON BLOCKEXIT.
COMPILATION TIMES WERE:
                        00004.791 SECONDS ELAPSED
                        00000.508 SECONDS PROCESS
                        00000.686 SECONDS I-O
```

FOR2

ESPOL/MCP LAB

Write an ESPOL procedure to be bound to standard intrinsics. The procedure will take two parameters, a case number and an array. Thus the procedure will be declared as:

PROCEDURE ESPOLINT (CASENO, ARY)

A program has been written to drive your intrinsic. The name of this program is SYMBOL/RUNNER. A listing is attached. Change the comment line at the end of the program to call your intrinsic. Your intrinsi should do the following:

CASE NO.	ACTION
0	ARY[0] = D0 ARY[1] = D1 ARY[2] = D2 ARY[3] = F ARY[4] = S ARY[5] = LOSR ARY[6] = BOSR ARY[7] = SNR
1	ARY[0] = your stack number ARY[1] = your segment dictionary stack number ARY[2] = your intrinsics stack number ARY[3] = system intrinsics stack number
2	Your program-ID (name) in ARY. Standard form is good. You must use the word in your task to get your name.
3	RCW - MSCW chain.
4	Largest available memory area and address in MEM[0] list. Place answer in ARY[0] and ARY[1], largest available memory area and address in MEM[1] list. Place answer in ARY[2] and ARY[3]. In both cases, the address must point to the start of the area.

CASE NO.	ACTION
5	Build a SIRW to point to your tas that is, the memory area of your task. Strip tag and place in ARY [0] .
6	Invoke procedure P in SYMBOL/RUNI
7	The name of the intrinsics, stanc form is good. You may not use tl task of the intrinsics stack for this.
8	Do a disk read from the H/L unit address of 0 for 1 segment. Use disk wait.
9	Same as 8 except do a read with tags, then do a program dump. ZOT the array ARY to be sure the ALGOL program does not get tags.
10	Resize ARY to 60 words. You may not call resize and deallocate. Fix up MOM, links and copy descriptors.
11	ARY [0] = Number of tasks running. ARY [1] = Number of segment dictionaries in use.

INSTRUCTIONS

We will use CANDE. Use the following USERCODE/PASSWORD:

MCP/CLASS

Since all files will be placed under the usercode MCP, use unique names for your intrinsic and copy of RUNNER.

All files will be on a pack called MCPCLASS.

To make your instrinsic:

(1) Write the code. Use the following dollar options:

a. SINGLE STACK LIST INTRINSICS LOADINFO

b. The info file name is INFO.

c. Your procedure must end with an "END."

(2) Your procedure must be under the node INTX/=.

To bind your intrinsic:

(1) SO MSG. so you can see BIND.

(2) ST BINDINT

(3) BINDINT will do a BIND and CI.

(4) After CI get a copy of MYPART and fix line. Compile.

(5) Run with option fault arrays.

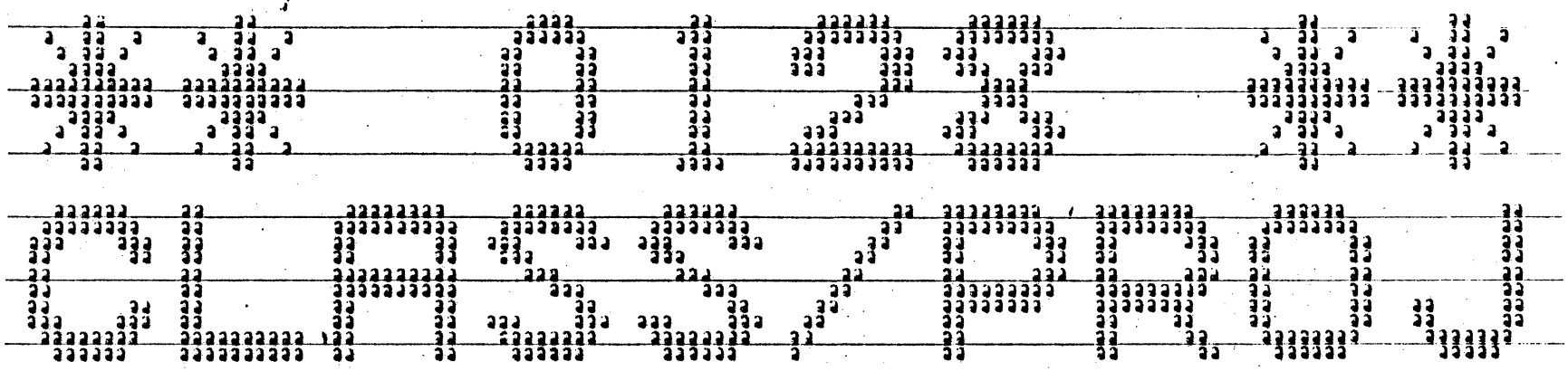
(6) To fix.

Make change.

Compile.

ST BINDINT

Run



DATE: 03:07:55,SYSTEM SERIAL: 1319, B MCP: SYSTEM/MCP30149. III.00.176

RUN 1

WORK FLOW STATEMENTS

```
00000100  ?JOB CLASS/PROJECT;
00000200    CLASS 3;
00000300  BEGIN
00000400  COMPILE MY/PART ALGOL LIBRARY;
00000500  ALGOL FILE NEWTAPE(TITLE=SYMBOL/RUNNER);
00000600  DATA CARD;
00006600  ?END JOB
```

JOB SUMMARY

```
03:06:33    BOJ  0128  CLASS/PROJECT.
                       JOB ENTERED SYSTEM:              03:06:33
                       QUEUE: 3
                       ORIGINATING UNIT: 26
                       PRIORITY: 50

03:06:34    BOT  0129  SYSTEM/ALGOL ON OLDSPACK.
                       TASK TYPE: COROUTINE
                       PRIORITY: 50
                       CODEFILE: MY/PART.
03:06:42         0129  MY/PART REMOVED ON OLDSPACK PK208 .
03:06:43         0129  SYMBOL/RUNNER REMOVED ON OLDSPACK PK208 .
03:06:43    EOT  0129  SYSTEM/ALGOL ON OLDSPACK.
                       1.594 SEC CPU, 2.487 SECS IO
                       59 CARDS READ  103 LINES PRINTED
                       MEM INTEGRAL: CODE=57.029, DATA=42.705
                       AVERAGE CORE USAGE: CODE=13975 DATA=10465
                       ELAPSED TIME: 00:00:09

03:06:44    EOJ  0128  CLASS/PROJECT.
                       0.105 SEC CPU, 0.254 SECS IO
                       MEM INTEGRAL: CODE=0.029, DATA=0.414
                       AVERAGE CORE USAGE: CODE=81 DATA=1152
                       ELAPSED TIME: 00:00:10
```

SYMBOL/RUNNER (C1...

MY/PART

NEW SYMBOLIC: SYMBOL/RUNNER ON OLDSPACK.

```
                $SET NEW STACK LIST                                                    00001000  000:0000:0
                % SET INSTALLATION                                                     00002000  000:0000:0
                BEGIN                                                                  00003000  000:0000:0
(01,00002) = B.0000
(01,00003) = SEGMENT DESCRIPTOR
                                                                                                  B.0003 IS SEGMENT 00003
                FILE LINE(KIND=PRINTER);                                          1    00004000  003:0000:1
(02,00002) = FUNNY STRM
(02,00003) = LINE
                                                                                                  DATA IS 0035 LONG
                DEFINE ASIZE = 30 #;                                                   00005000  003:0000:1
                ARRAY A[0:ASIZE-1];                                                    00006000  003:0000:1
(02,00004) = A
                REAL I;                                                                00007000  003:0000:1
(02,00005) = I
                PROCEDURE P;                                                           00008000  003:0000:1
(02,00006) = P
                BEGIN                                                                  00009000  003:0000:1
                  ARRAY STUFF[0:21];                                                   00010000  003:0000:1
(01,00004) = SEGMENT DESCRIPTOR
                                                                                                  P IS SEGMENT 00004
(03,00002) = STUFF
                  POINTER P;                                                      2    00011000  004:0000:1
(03,00003) = P
                  REPLACE P:=POINTER(STUFF) BY " " FOR 22 WORDS;                       00012000  004:0000:1
                  REPLACE P BY "YOU MADE IT.";                                         00013000  004:0003:3
                  WRITE(LINE,22,STUFF);                                                00014000  004:0008:5
(01,00005) = SEPARATE INTRINSIC
                  PROGRAMDUMP;                                                         00015000  004:000F:5
(01,00006) = SEPARATE INTRINSIC
                END;                                                                   00016000  004:0010:5
(01,00007) = SEPARATE INTRINSIC
                                                                                                  P(004) IS 0027 LONG
                PROCEDURE DUMPIT;                                                 2    00017000  003:0000:1
(02,00007) = DUMPIT
                BEGIN                                                                  00018000  003:0000:1
                  ARRAY OUTLINE[0:21];                                                 00019000  003:0000:1
(01,00008) = SEGMENT DESCRIPTOR
                                                                                                  DUMPIT IS SEGMENT 00008
(03,00002) = OUTLINE
                  POINTER PO;                                                     2    00020000  008:0000:1
(03,00003) = PO
                  REAL FULLTIMES,SZ,I,NUMLEFT,J,INX;                                   00021000  008:0000:1
(03,00004) = FULLTIMES
(03,00005) = SZ
(03,00006) = I
(03,00007) = NUMLEFT
(03,00008) = J
(03,00009) = INX
                  DEFINE BLANK = REPLACE PO:=POINTER(OUTLINE) BY " " FOR 22 WORDS #;   00022000  008:0000:1
                  FULLTIMES:=((SZ:=SIZE(A)) DIV 8)-1;                                  00023000  008:0002:1
                  FOR I:=0 STEP 1 UNTIL FULLTIMES DO                                   00024000  008:0033:0
                    BEGIN                                                              00025000  008:0097:2
                    BLANK;                                                         3   00026000  008:0097:2
                    FOR J:=0 STEP 1 UNTIL 7 DO                                         00027000  008:000A:3
                      BEGIN                                                            00028000  008:0009:1
                      REPLACE PO:PO BY POINTER(A[INX],4) FOR 12 WITH                4  00029000  008:0009:1
                       HEXTOEBCDIC;                                                    00030000  008:000D:1
(01,00009) = SEPARATE INTRINSIC
                      PO:=PO+3;                                                        00031000  008:000E:5
                      INX:=*+1;                                                        00032000  008:0010:0
                      END;                                                             00033000  008:0011:1
                    WRITE(LINE,22,OUTLINE);                                        4   00034000  008:0013:5
                    END;                                                               00035000  008:001A:5
                  NUMLEFT:=(FULLTIMES+1)*8;                                        3   00036000  008:0019:2
                  IF NUMLEFT:=SZ-NUMLEFT EQL 0 THEN                                    00037000  008:001D:0
                    WRITE(LINE [SPACE 3])                                              00038000  008:001E:4
                  ELSE                                                                 00039000  008:0021:0
(01,0000A) = SEPARATE INTRINSIC
                    BEGIN                                                              00040000  008:0023:2
```

```
                        BLANK;                                                  3   00041000   008:0023:5
                        FOR I:=0 STEP 1 UNTIL NUMLEFT-1 DO                          00042000   008:0027:3
                           BEGIN                                                    00043000   008:002C:1
                           REPLACE PO:PO BY POINTER(AE[INX],4) FOR 12 WITH      4   00044000   008:002C:1
                            HEXTOE8CD1C;                                            00045000   008:002E:1
                           PO:=PO+3;                                                00046000   008:002E:5
                           INX:=*+1;                                                00047000   008:0031:0
                           END;                                                     00048000   008:0032:1
                        WRITE(LINE [SPACE 3], 22, OUTLINE);                     4   00049000   008:0032:4
                        END                                                         00050000   008:0039:5
                     END;                                                       3   00051000   008:003B:5
                                                                                    DUMPIT(008) IS 0051 LONG
                     FOR I:=0 STEP 1 UNTIL 11 DO                                2   00052000   003:0000:1
                        BEGIN                                                       00053000   003:0000:5
                        COMMENT YOURINTNAME[I,A);                               2   00054000   003:0000:5
                        DUMPIT;                                                     00055000   003:0000:5
                        DEALLOCATE(A);                                              00056000   003:0001:3
(01,00008) = SEPARATE INTRINSIC
                        END;                                                        00057000   003:0003:0
                     END.                                                       2   00058000   003:0005:4
                                                                                    9.0000(003) IS 0012 LONG
```

RUN3

```
NUMBER OF ERRORS DETECTED = 0.
NUMBER OF SEGMENTS = 9.  TOTAL SEGMENT SIZE = 143 WORDS.  CORE ESTIMATE = 996 WORDS.  STACK ESTIMATE = 18
PROGRAM SIZE = 58 CARDS, 284 SYNTACTIC ITEMS, 10 DISK SEGMENTS.
PROGRAM FILE NAME:  MY/PART, 8700 CODE GENERATED.
COMPILATION TIME = 5.834 SECONDS ELAPSED;  1.087 SECONDS PROCESSING;  1.600 SECONDS I/O.
```

MEMORY MANAGEMENT

Available memory areas are linked to each other. There are two chains for memory areas: one list is searched if the memory area can be overlaid; the other list is searched if the area is a save area. Memory management will try to keep all save areas at one end of memory (low memory address). Overlay areas are at the other end of memory (high memory address). This is done to ease memory checkerboard.

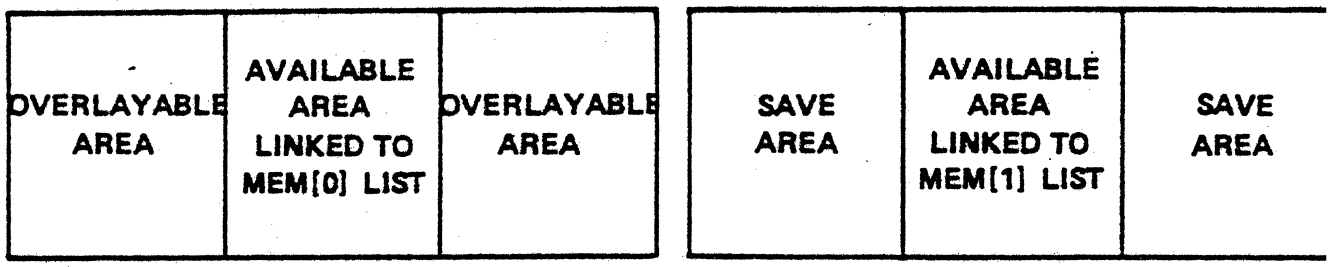
The start link for the overlay memory chain is MEM(0). The start link for the save memory chain is MEM(1). The MEM(0) list is in AGE order, i.e., last area forgotten is first in list. The MEM(1) is in size order. The order is smallest to largest. A memory area can be in one or both lists. If both areas (high area and low area) around an area to be placed in a memory chain are save, the area is linked into the save chain. If both areas are overlay, link new area into overlay chain. If low area is save and high area is overlay, link into both chains. If low area is overlay and high area is save, link into overlay (see Figure MM-1). Memory is always in the largest chunk; that is, there will never be two available areas back to back.

There will be 4 links around all memory areas. The link names for available areas are: AVAILA, AVAILB, AVAILY, and AVAILZ. The link names for in use areas are: LINKA, LINKB, LINKC, and LINKZ (see Figure MM-2 thru MM-7).

Consider the memory list in Figure MM-8. Assume we want 25 words of overlay memory. We will do a LLLU on MEM(0) looking for a size 25. We will stop at 30. This area is delinked from both memory chains.

The MCP will put a large number in the size field of the last link to memory. Thus, LLLU will never return A -1.

MEM[0] LIST FOR OVERLAYABLE MEMORY, ORDERED ON AGE (NEWEST FIRST).
MEM[1] LIST FOR SAVE MEMORY, ORDERED ON SIZE (SMALLEST FIRST).



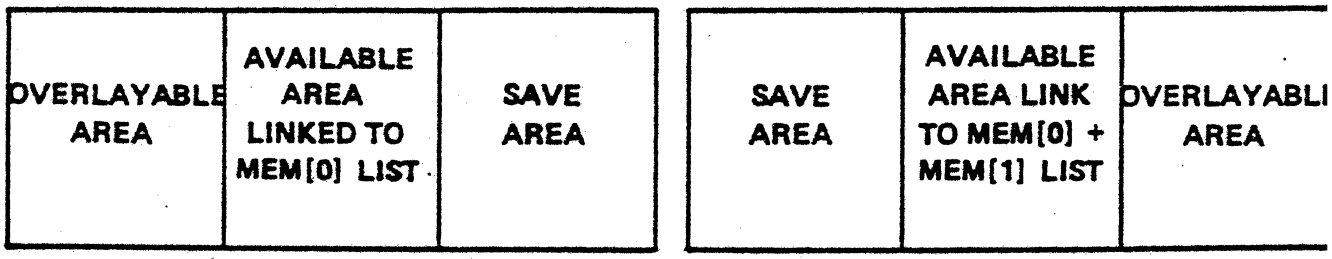
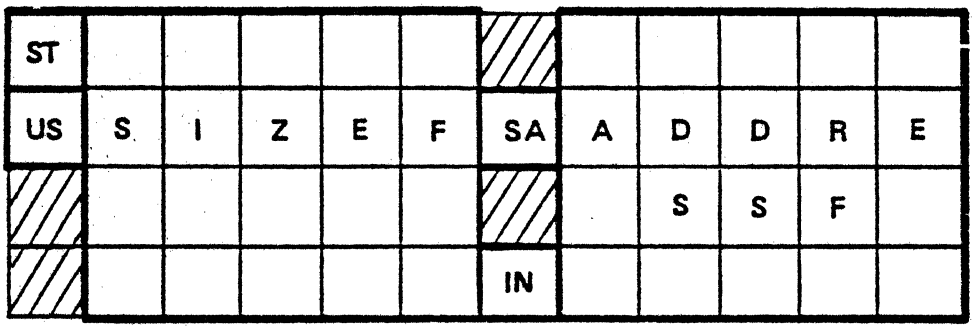


Figure MM-1. Available Memory Area Lists

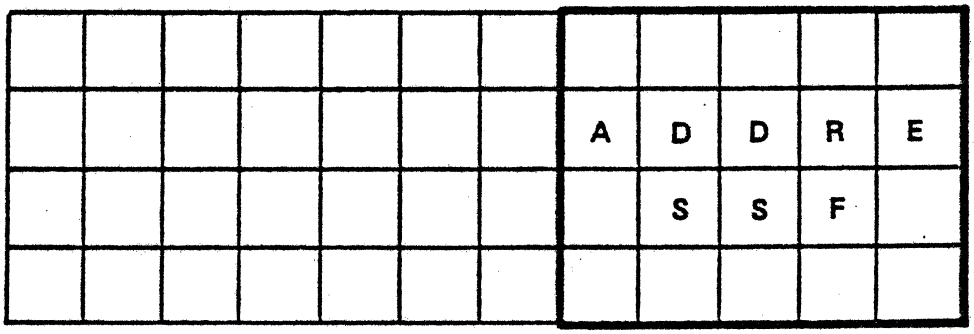
AVAILA TAG 1



[47:1] STOFFERF [22:1] SAVEDF
[46:1] USABLEF [20:1] INUSEF
[43:20] SIZEF [19:20] ADDRESSF

STOPPERF ON IF LAST AREA. STOPS LLLU
USABLEF OFF IN LAST AREA.
ADDRESSF LINK TO NEXT MEMORY AREA.
POINTS TO AVAILA. NOT IN SIZE ORDER.
SIZEF SIZE OF AREA INCLUDES LINKS

AVAILB TAG 0

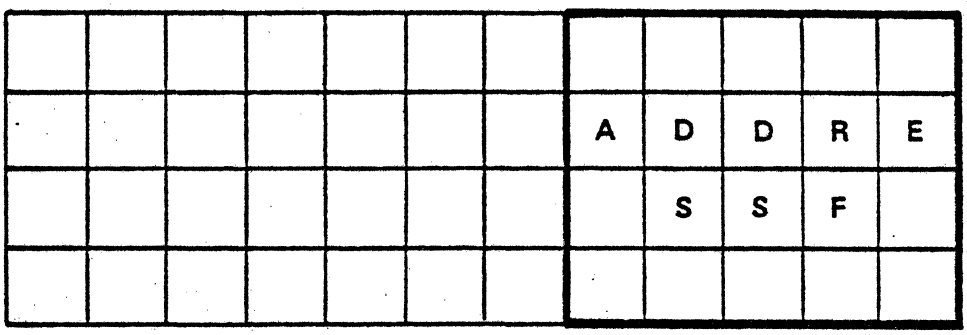


[19:20] ADDRESSF

ADDRESSF IS ADDRESS OF LINK WHICH
POINTED TO MY AVAILA WORD. THAT
IS, A BACK LINK.

Figure MM-2. Memory Links Around Available Areas Linked to MEM[0] (Sheet 1 of 2)

AVAILY TAG 0

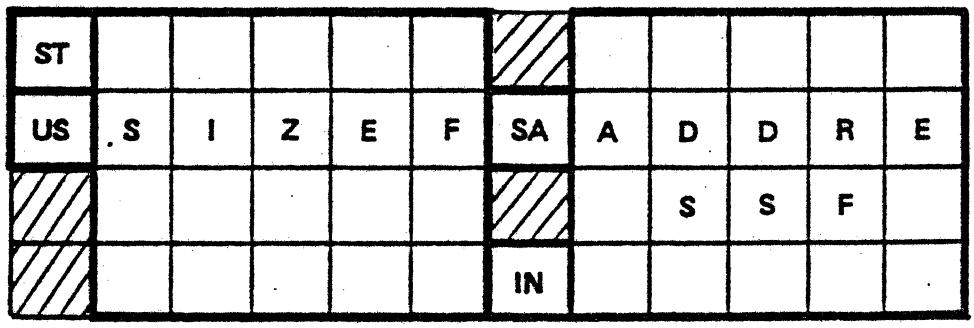


[19:20] ADDRESSF

ADDRESSF IS ADDRESS OF LINK WHICH POINTED TO MY AVAILZ WORD. THAT IS, A BACK LINK. ADDRESSF USED ONLY IF THIS AREA IS IN MEM[1] CHAIN ALSO.

NOTE: AVAILABLE MEMORY IS BETWEEN AVAILB AND AVAILY.

AVAILZ TAG 1



[47:1] STOPPERF [22:1] SAVEDF
[46:1] USABLEF [20:1] INUSEF
[43:20] SIZEF [19:20] ADDRESSF

SIZEF SIZE OF AREA INCLUDES LINKS
ADDRESSF LINK TO NEXT MEMORY AREA. POINTS TO AVAILZ. USED ONLY IF THIS AREA IS IN MEM[1] CHAIN ALSO.

Figure MM-3. Memory Links Around Available Areas Linked to MEM[0] (Sheet 2 of 2)

AVAILA TAG 1

ST											
US	S	I	Z	E	F	SA	A	D	D	R	E
								S	S	F	
						IN					

[47:1] STOPPERF
[46:1] USABLEF
[43:20] SIZEF
[22:1] SAVEOF
[20:1] INUSEF
[19:20] ADDRESSF

SIZEF INCLUDES LINKS
ADDRESSF LINK TO NEXT MEMORY AREA.
POINTS TO AVAILA. USED ONLY IF THIS
AREA IS IN MEM[0] CHAIN ALSO.

AVAILB TAG 0

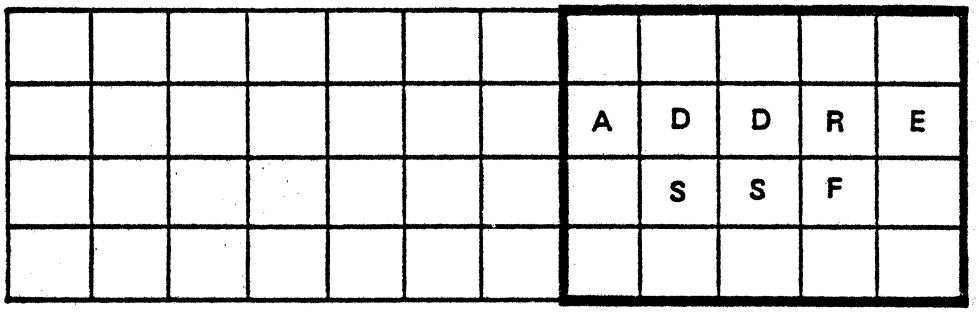
							A	D	D	R	E
								S	S	F	

[19:20] ADDRESSF

ADDRESSF IS ADDRESS OF LINK WHICH
POINTED TO MY AVAILA. THAT IS, A
BACK LINK. USED ONLY IF THIS AREA
IS IN MEM[0] CHAIN ALSO.

Figure MM-4. Memory Links Around Available Areas Linked to MEM [1] (Sheet 1 of 2)

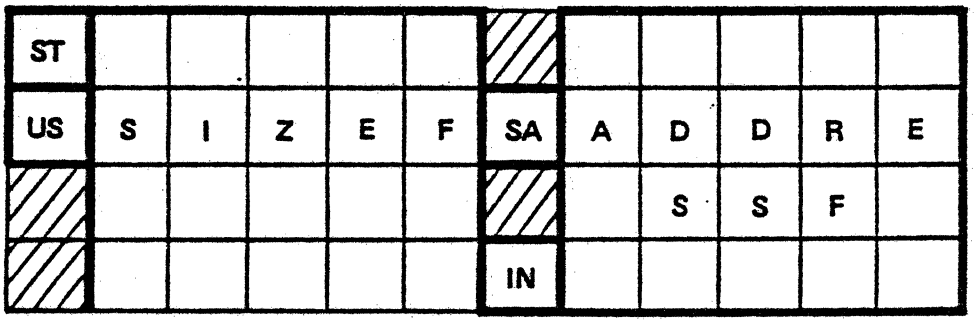
AVAILY TAG 0



[19:20] ADDRESSF
ADDRESSF IS ADDRESS OF LINK WHICH POINTED TO MY AVAILZ. THAT IS, A BACK LINK.

NOTE: AVAILABLE MEMORY IS BETWEEN AVAILB AND AVAILY.

AVAILZ TAG 1



[47:1] STOPPERF [22:1] SAVEOF
[46:1] USABLEF [20:1] INUSEF
[43:20] SIZEF [19:20] ADDRESSF

SIZEF INCLUDES LINKS
ADDRESSF LINK TO NEXT MEMORY AREA. POINTS TO AVAILZ WORD.
IN ORDER OF INCREASING SIZE.

Figure MM-5. Memory Links Around Available Areas Linked to MEM [1] (Sheet 2 of 2)

LINKA TAG 6

OL						ST					
CF	S	I	Z	E	F	CS	A	D	D	R	E
TS						SV		S	S	F	
PS						IN					

[47:2] OVERLAYCF
[45:1] TEMPSAVF
[44:1] PRECURSAVF
[43:20] SIZEF
[23:1] STICKYF
[22:1] CURSAVF
[21:1] SAVEF
[20:1] INUSEF
[19:20] ADDRESSF

OVERLAYCF WILL BE 3.
SIZEF INCLUDES LINKS.
ADDRESSF IS ADDRESS OF MOM.

LINKB TAG 7

OL						D					
CF			U	S	A	E	A	D	D	R	E
S	T	K	G	E	F	L		S	S	F	
N	R	F									

[47:2] OVERLAYCF
[45:10] STKNRF
[35:12] USAGEF
[23:4] DELTAF
[19:20] ADDRESSF

STKNRF STACK NUMBER OF MOM
USAGEF USAGE OF AREA (STACK, FIB, DUPEVEC)
DELTAF NUMBER OF SLOP (WASTE) WORDS
ADDRESSF DISK ADDRESS FOR OVERLAY

Figure MM-6. Memory Links Around Inuse Areas (Sheet 1 of 2)

LINKC TAG 3

USED TO STORE RCW OF WHERE
PBIT OCCURED OR
USED TO HOLD IOCW

NOTE: INUSE AREA IS BETWEEN
LINKC AND LINKZ

LINKZ TAG 3

SI						DP					
	S	I	Z	E	F	CS	A	D	D	R	E
						SV		S	S	F	
						IN					

[47:1] SAVINGF [21:1] SAVEF
[43:20] SIZEF [20:1] INUSEF
[23:1] DELTAPF [19:20] ADDRESSF
[22:1] CURSAVF

SAVINGF USED WHEN SAVING MODS
DELTAPF IF THERE ARE SLOP (WASTE) WORDS
ADDRESSF USED FOR MEMORY PRIORITY

Figure MM-7. Memory Links Around Inuse Areas (Sheet 2 of 2)

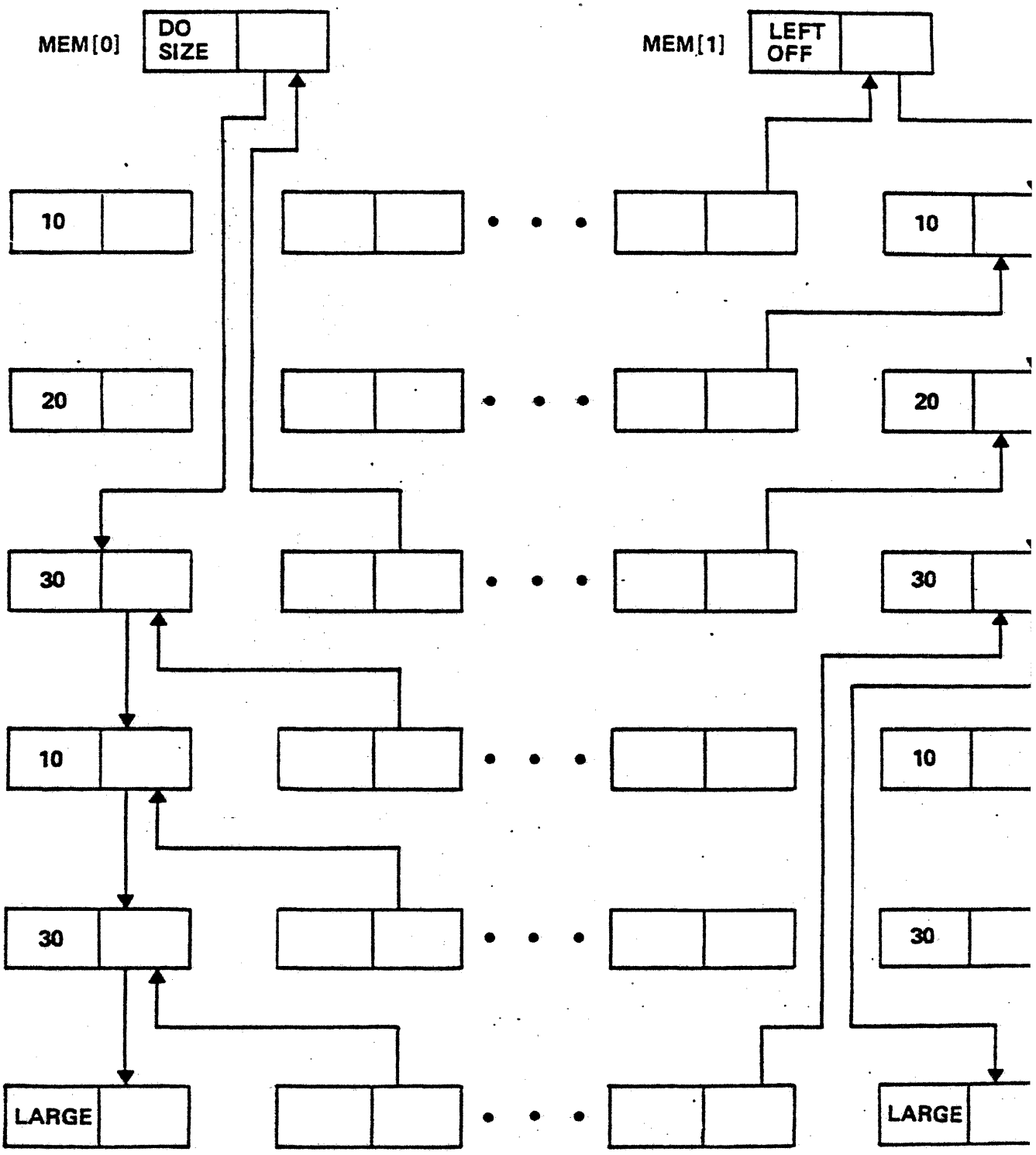


Figure MM-8. Memory Linked Lists (Available Lists)

GETSPACE(

NUMBER OF WORDS I WANT,
STACK NUMBER OF MOM,
USAGEL (See Below)
MEMORY ADDRESS OF MOM)

GETSPACE RETURNS MEMORY ADDRESS OF AREA. IF A 0 IS RETURNED, WE DID NOT GET ANY MEMORY.

USAGEL CONTAINS:
NOZEROXINGALLOWEDF = 28:1

THERE WILL BE NO COPY DESCRIPTOR TO THIS AREA. DO NOT NEED TO DO A STACK SEARCH WHEN YOU OVERLAY.

ODDBALLF = 27:4

THE USE OF THE AREA. SEE VALUES AT 03215000.

SAVEF = 21:1

GIVE ME SAVE MEMORY.

ADDRESSF = 19:20

DISK ADDRESS OF WHERE TO OVERLAY THIS AREA IF KNOWN.

OLAYEXITF = 47:1

USED AS LOCAL IN GETSPACE. SEE GETSPACE.

DONTLOSEMEF = 35:1

IF NO MEMORY AVAILABLE, DON'T OVERLAY TO GET ME SOME. I DO NOT WANT TO LOSE CONTROL OF PROCESSOR.

ICANDOWITHOUTITF = 34:1

IF NO MEMORY AVAILABLE EVEN AFTER YOU TRY OVERLAY, RETURN TO ME.

I WONTTELLIFYOUWONTTELL = 33:1

IF NO MEMORY AVAILABLE EVEN AFTER OVERLAY JUST WAIT IN GETSPACE. THAT IS, WHEN YOU RETURN I WILL HAVE MEMORY. IF THERE IS NO MEMORY DO NOT CALL DEFUNCT.

ITSNOTDOORDIEF = 32:1

NOT USED.

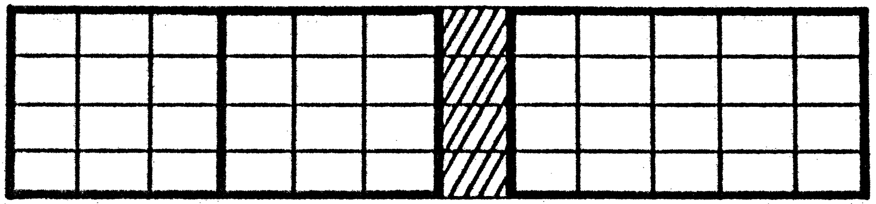
UNTOUCHABLEF = 31:1
UNTOUCHEDF = 30:1

I WILL NEVER WRITE IN THIS AREA... DO NOT NEED TO WRITE THIS AREA OUT TO DISK TO OVERLAY. EXAMPLE: CODE, VALUE ARRAY.

NOTAGSALLOWEDF = 29:1

NO MIXED TAGS IN THIS AREA. IT'S NOT A WORD ARRAY. WHEN YOU OVERLAY ARRAY DO NOT NEED TO DO A WRITE WITH TAGS.

GETAVAILA TAG 5

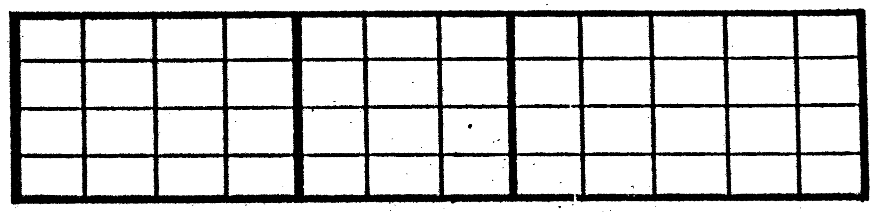


[47:12]
FOR LINKF
ROW
RELATIVE
ADDRESS.
AREA IN
FRONT OF
ME IN
THIS ROW

[35:12]
BACKLINKF
ROW
RELATIVE
ADDRESS.
AREA BEHIND
ME IN THIS
ROW

[19:20] ADDRESSF
BACKLINK.
TO SMALLER
AVAILB. MEM
ADDR.

GETAVAILB TAG 5



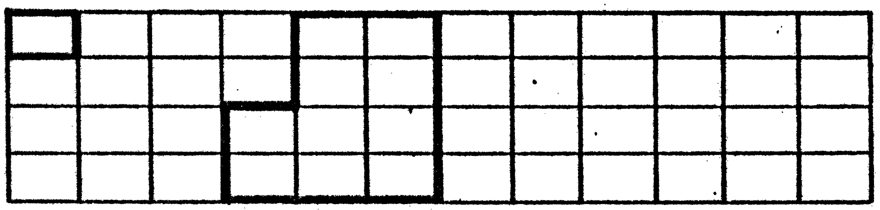
[47:16]
AREALENGTHF
LENGTH OF
AREA.
INCLUDE
LINKS

[31:12]
ROWNUMF
WHICH
ROW WE
ARE IN

[19:20] ADDRESSF
LINK TO NEXT
LARGER AREA.
TO AVAILB.

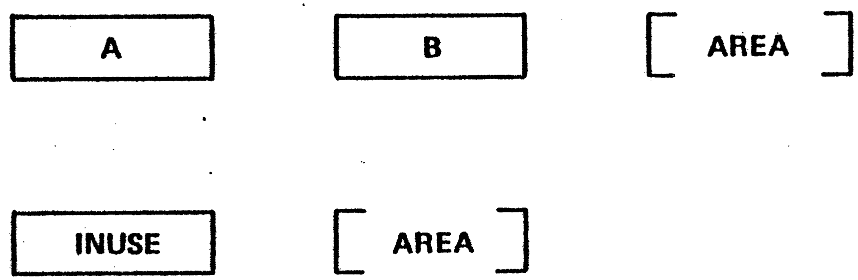
GFL 1

INUSE LINK TAG 7



[47:1]
PBITF
IF THERE
IS A SLOP
WORD

[33:10]
MSGLENGTHF
SIZE OF
AREA INCLUDE
THIS WORD



GETAREA FORGETAREA LINKS (P. S: [illegible])

I P C / S T U F F

```
                BEGIN                                                                                    00001000  003:0000:0
(01,00002) = 8.3000
(01,00003) = SEGMENT DESCRIPTOR
                                                                                                                8.3000 IS SEGMENT 00003
                                                003:0000:0  NVLD                FF
                TASK T;                                                                                1        00002000  003:0000:1
(02,00002) = T
                REAL X,Y;                                                                                        00003000  003:0000:1
(02,00003) = X
(02,00004) = Y
                PROCEDURE P;                                                                                     00004000  003:0000:1
(02,00005) = P
                    EXTERNAL;                                                                                    00005000  003:0000:1
(01,00004) = SEGMENT DESCRIPTOR
                                                                                                                   P IS SEGMENT 00004
                PROCEDURE P1;                                                                                    00006000  003:0000:1
(02,00006) = P1
                BEGIN                                                                                            00007000  003:0000:1
                    REAL A,B;                                                                                    00008000  003:0000:1
(01,00005) = SEGMENT DESCRIPTOR
                                                                                                                  P1 IS SEGMENT 00005
                                                005:0000:0  NVLD                FF
(03,00002) = A
(03,00003) = B
                    A:=A+1;                                                                            2        00009000  005:0000:1
                                                005:0000:1  NOOP                FE
                                                005:0001:2  BRUN      0002:4    A28002
                                                005:0002:4  VALC  (03,00002)    3002
                                                005:0003:0  ONE                 81
                                                005:0003:1  ADD                 80
                                                005:0003:2  NAMC  (03,00002)    7002
                                                005:0003:4  STOD                88
                    B:=B+1;                                                                                      00010000  005:0003:5
                                                005:0003:5  VALC  (03,00003)    3003
                                                005:0004:1  ONE                 81
                                                005:0004:2  ADD                 80
                                                005:0004:3  NAMC  (03,00003)    7003
                                                005:0004:5  STOD                88
                END;                                                                                         00011000  005:0005:0
                                                005:0005:0  EXIT                A3
**************************************  STACK BUILDING CODE FOR LEVEL 03  **************************************
                                                005:0002:1  ZERO                90
                                                005:0002:2  ZERO                80
                                                005:0002:3  PUSH                84
                                                005:0005:1  NVLD                FF
                                                005:0005:2  NVLD                FF
                                                005:0005:3  NVLD                FF
                                                005:0005:4  NVLD                FF
                                                005:0005:5  NVLD                FF
                                                                                                                 P1(005) IS 0006 LONG
                PROCEDURE P2(A,B);                                                                 2        00012000  003:0000:1
(02,00007) = P2
                VALUE A;                                                                                         00013000  003:0000:1
                REAL A,B;                                                                                        00014000  003:0000:1
                BEGIN                                                                                            00015000  003:0000:1
(03,00002) = A
(03,00003) = B
                    A:=A+1;                                                                                      00016000  003:0000:1
                                                003:0000:1  VALC  (03,00002)    3002
                                                003:0000:3  ONE                 81
                                                003:0000:4  ADD                 80
                                                003:0000:5  NAMC  (03,00002)    7002
                                                003:0001:1  STOD                88
                    B:=A+1;                                                                            2        00017000  003:0001:2
                                                003:0001:2  NAMC  (03,00003)    7003
                                                003:0001:4  EVAL                AC
                                                003:0001:5  VALC  (03,00002)    3002
                                                003:0002:1  ONE                 81
                                                003:0002:2  ADD                 80
                                                003:0002:3  STOD                88
                END;                                                                                         00018000  003:0002:4
                                                003:0002:4  EXIT                A3
**************************************  STACK BUILDING CODE FOR LEVEL 03  **************************************
```

IPS1

TPC/STUFF (Sheet 1 of 3)

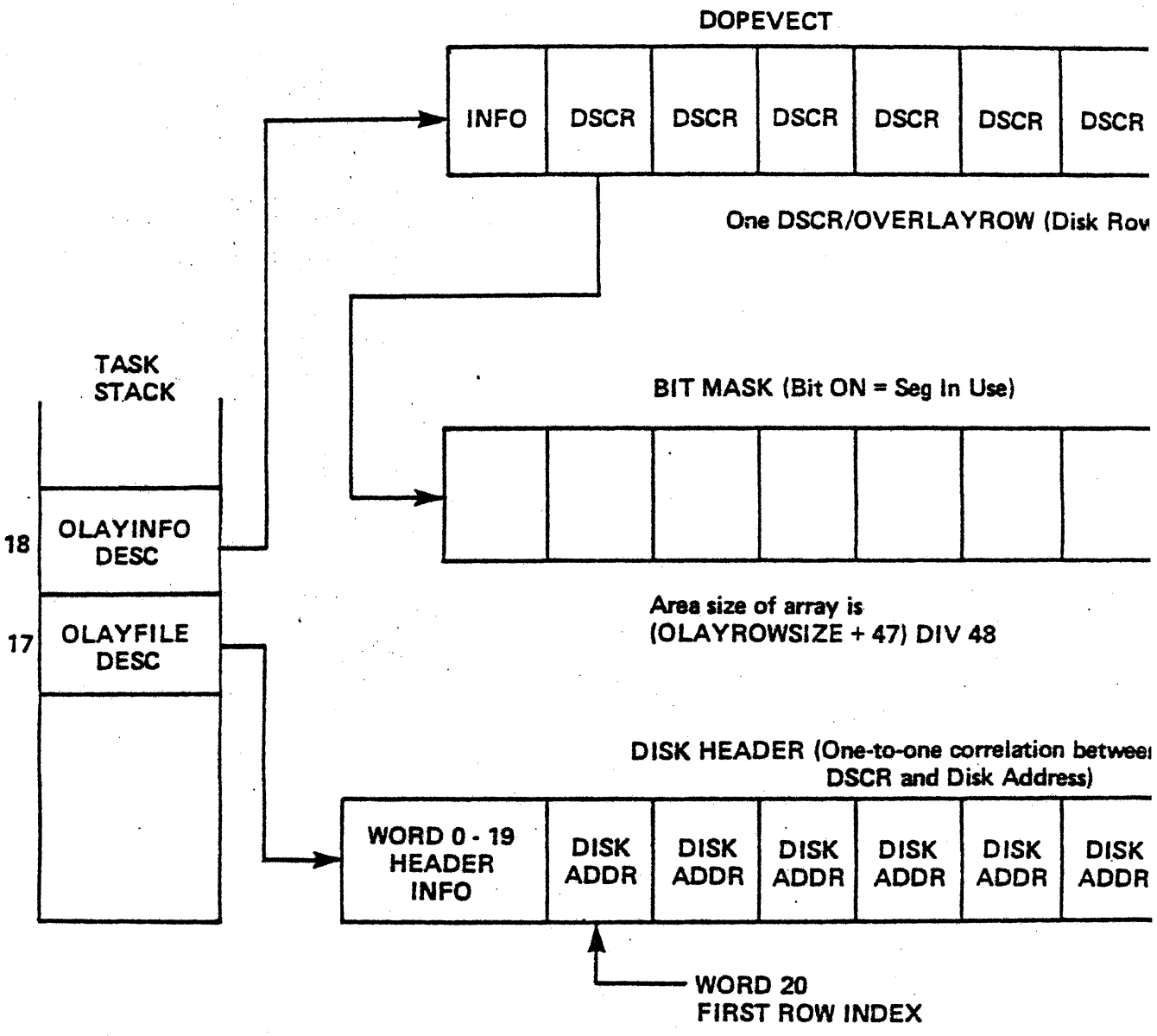
```
                REPLACE T.NAME BY "SYSTEM/PATCH.";                                  2   00019000  003:0002:5
                                    003:0002:5  MKST                AE
(01,00006) = SEPARATE INTRINSIC
                                    003:0003:0  NAMC  (01,00006)    6006
                                    003:0003:2  NAMC  (02,00002)    5002
                                    003:0003:4  STFF                AF
                                    003:0003:5  ZERO                90
(01,00007) = POOL DATA DESCRIPTOR
                                    003:0004:0  ZERO                90
                                    003:0004:1  NAMC  (01,00007)    6007
                                    003:0004:3  INDX                A6
                                    003:0004:4  BSET                962A
                                    003:0005:0  ENTR                AB
                RUN P [T];                                                          00021000  003:0005:1
                                    003:0005:1  MKST                AE
(01,00008) = SEPARATE INTRINSIC
                                    003:0005:2  NAMC  (01,00008)    6008
                                    003:0005:4  LIB                 8202
                                    003:0006:0  NAMC  (02,00005)    5005
                                    003:0006:2  STFF                AF
                                    003:0006:3  NAMC  (02,00002)    5002
                                    003:0006:5  STFF                AF
                                    003:0007:0  NAMC  (02,00000)    5000
                                    003:0007:2  STFF                AF
                                    003:0007:3  ENTR                AB
                CALL P1 [T];                                                        00021000  003:0007:4
                                    003:0007:4  MKST                AE
                                    003:0007:5  NAMC  (01,00008)    6008
                                    003:0008:1  ONE                 81
                                    003:0008:2  NAMC  (02,00006)    5006
                                    003:0008:4  STFF                AF
                                    003:0008:5  NAMC  (02,00002)    5002
                                    003:0009:1  STFF                AF
                                    003:0009:2  NAMC  (02,00000)    5000
                                    003:0009:4  STFF                AF
                                    003:0009:5  ENTR                AB
                PROCESS P2(X,Y) [T];                                                00022000  003:000A:0
                                    003:000A:0  MKST                AE
                                    003:000A:1  NAMC  (01,00008)    6008
                                    003:000A:3  ZERO                90
                                    003:000A:4  NAMC  (02,00007)    5007
                                    003:000B:0  STFF                AF
                                    003:000B:1  VALC  (02,00003)    1003
                                    003:000B:3  NAMC  (02,00004)    5004
                                    003:000B:5  STFF                AF
                                    003:000C:0  NAMC  (02,00002)    5002
                                    003:000C:2  STFF                AF
                                    003:000C:3  NAMC  (02,00000)    5000
                                    003:000C:5  STFF                AF
                                    003:000D:0  ENTR                AB
                END.                                                                00023000  003:000D:1
                                    003:000D:1  MKST                AE
(01,00009) = SEPARATE INTRINSIC
                                    003:000D:2  NAMC  (01,00009)    6009
                                    003:000D:4  ENTR                AB
                                    003:000D:5  EXIT                A3
```

IPS2

IPC/STUFF (Chapter 2, Page 2)

```
********************************  STACK BUILDING CODE FOR LEVEL 02  ********************************
(02,00002)  = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
                                    003:000E:0  LT8                 82F0
                                    003:000E:2  ISOL                9A1E30
                                    003:000E:5  LT8                 8205
                                    003:000F:1  STAG                9584
                                    003:000F:3  ZERO                30
                                    003:000F:4  ZERO                30
(02,00005)  = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
                                    003:000F:5  LT48                8F
                                    003:0010                        400000000004 (3"2000000000000004")
(02,00006)  = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
                                    003:0011:0  MPCW                9F
                                    003:0012                 005:000021:1   00020020E005
(02,00007)  = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
                                    003:0013:0  MPCW                9F
                                    003:0014                 003:0000:1     00020000E003
                                    003:0015:0  LT16                838008
                                    003:0015:3  ISOL                9A0330
                                    003:0016:0  LT8                 8206
                                    003:0016:2  STAG                9584
                                    003:0016:4  BRUN        000215  A2A002
                                    003:0017:1  NVLD                FF
                                    003:0017:2  NVLD                FF
                                    003:0017:3  NVLD                FF
                                    003:0017:4  NVLD                  FF
                                    003:0017:5  NVLD                  FF
                                                                    8.0000(003) IS 0018 LONG
                                                                          DATA IS 0003 LONG
================================================================================================================
NUMBER OF ERRORS DETECTED = 0.
NUMBER OF SEGMENTS = 7.  TOTAL SEGMENT SIZE = 33 WORDS.  CORE ESTIMATE = 45 WORDS.  STACK ESTIMATE = 12
PROGRAM SIZE = 23 CARDS, 91 SYNTACTIC ITEMS, 14 DISK SEGMENTS.
PROGRAM FILE NAME:  IPC/STUFF. 87700 CODE GENERATED.
COMPILATION TIME = 4.104 SECONDS ELAPSED;  0.420 SECONDS PROCESSING;  0.683 SECONDS I/O.
================================================================================================================
```

IPS3



INFO (03111)
INUSEROWS
ROWSIZE

DISK ADDR (070281)
UNIT NUMBER
SEG NUMBER

HEADER INFO (07)
ROWSIZE
NUMBER OF ROWS

```
           BEGIN                                                                          00001000   000:0000:0
 (01,00002) = 0.0000
 (01,00003) = SEGMENT DESCRIPTOR
                                                                                           0.0000 IS SEGMENT 00003
                                        003:0000:0  NVLD                  FF
              ARRAY A[0:29];                                                            1  00002000   003:0000:1
 (02,00002) = A
              FILE DSK(KIND=DISK,MYUSE=OUT,AREAS=20,AREASIZE=100,MAXRECSIZE=30,          00003000   003:0000:1
 (02,00003) = FUNNY STRW
 (02,00004) = DSK
              BLOCKSIZE=60);                                                             00004000   003:0000:1
                                            DATA POOL AT (02,00004):
                                            0000  010000000000
                                            0001  0Z0101030C4E2
                                            0002  020000000000
                                            0003  031004030801
                                            0004  031402031214
                                            0005  031164030E1E
                                            0006  030E3C000000
                                                                                            DATA IS 0007 LONG
              REPLACE POINTER(A) BY "HI THERE";                                          00005000   003:0000:1
                                        003:0000:1  ZERO                  80
                                        003:0000:2  NAMC   (02,00002)     5002
                                        003:0000:4  INDX                  A5
                                        003:0000:5  BSET                  962A
                                        003:0001:1  LT48                  BE
                                        003:0002                          C8C940E3C8C5 (3"62144500707443O5")
                                        003:0003:0  LT16                  B3D9C5
                                        003:0003:3  ISOL                  9A0F30
                                        003:0004:0  JOIN                  9542
                                        003:0004:2  LT8                   B20E
                                        003:0004:4  TUNO                  E6
              WRITE(DSK,30,A);                                                           00006000   003:0004:5
                                        003:0004:5  MKST                  AE
                                        003:0005:0  ZERO                  80
                                        003:0005:1  NAMC   (02,00004)     5004
                                        003:0005:3  STFF                  AF
                                        003:0005:3  INDX                  A6
                                        003:0005:4  LOAD                  30
                                        003:0005:5  ZERO                  80
                                        003:0006:0  LT8                   B21E
                                        003:0006:2  ZERO                  80
                                        003:0006:3  NAMC   (02,00002)     5002
                                        003:0006:5  INDX                  A6
                                        003:0007:0  LT48                  BE
                                        003:0008                          080000000680 (3"0200000003000200")
                                        003:0009:0  ENTR                  AB
                                        003:0009:1  DUPL                  B7
                                        003:0009:2  BRTR          000A:2  A1400A
 (01,00004) = SEPARATE INTRINSIC
                                        003:000A:2  NAMC   (01,00004)     6004
                                        003:000A:4  EXCH                  B6
                                        003:000A:5  IMKS                  CF
                                        003:000B:0  NAMC   (02,00004)     5004
                                        003:000B:2  STFF                  AF
                                        003:000B:3  ENTR                  AB
                                        003:0009:5  BRUN          000B:4  A28008
                                        003:000B:4  DLET                  B5
              PROGRAMDUMP(ARRAYS,FILES);                                                 00007000   003:000B:5
                                        003:000B:5  MKST                  AE
 (01,00005) = SEPARATE INTRINSIC
                                        003:000C:0  NAMC   (01,00005)     6005
                                        003:000C:2  LT16                  B30500
                                        003:000C:5  CHSN                  8E
                                        003:000D:0  ENTR                  AB
           END.                                                                          00008000   003:000D:1
                                        003:000D:1  MKST                  AE
 (01,00006) = SEPARATE INTRINSIC
                                        003:000D:2  NAMC   (01,00006)     6006
                                        003:000D:4  ENTR                  AB
                                        003:000D:5  EXIT                  A3
*************************************  STACK BUILDING CODE FOR LEVEL 02  *********************************************
 (02,00002)  = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
```

FSI1

FIBSTACK/INFO (Sheet 1 of 2)

```
B7700 PROGRAM DUMP FOR FIBSTACK/INFO. (MIX-3951/3952, STK-0EC)

MCP: SYSTEM/MCP., MARK 30.071.755     INTRINSICS: SYSTEM/INTRINSICS. (LOADED)          SYSTEM SERIAL: #124

CAUSE OF DUMP: PROGRAM REQUESTED @ 003:0000:1 (00008000).

PROGRAMDUMP OPTIONS:   ARRAY(S), FILE(S)

V29? = LOSR  (06A29)

0036 (01,0002)  0 400060 000501  OP:   DCT:20000000 00002401, BCL:+00000D01, EBC: ?????, DEC:-1281
0035            3 208200 D0A003  RCW:  LL=32, NORML STATE                    (USER SEGMENT       @ 0003:000D:1 (00008000))
                                 SEG DESC:  3 800001 6224E7
                                 CODE:  3 000860 0486CF   3 5004AF A985AE  3 600583 050.BE >3 A9AE60 .6A9A3< 3 B2F39A 1E3C92
0034 ----DE011=>3 C1A001 C04006 *MSCW: PREVIOUS MSCW @ 002E: DE011=001C IN STACK 01A

0033 (02,0005)  6 800000 002800  SCW: (BLOCK BELOW DECLARED 1 OR MORE FILES, SNGL-DIM ARRAYS)
0032 (02,0004)  5 800003 70740F  DESC (PRESENT-MOM): DATA, LENGTH=     55 (FILE: INTNAME=DSK., TITLE=DSK.)
                0(00) 1 408741 000010 SELECTOR
                1(01) 3 C1A001 C04002 FIBMSCW
                2(02) 0 000000 000000 FIBLOCK
                3(03) 0 0001E0 000300 RECORDSTATUS
                4(04) 0 010000 466024 FILESTATUS  (LEVEL=1,ALGOL, BUFFERS=002, WRITESERIAL STATE)
                5(05) 0 C80000 000000 TANKDATA1   (FILETYPE=0, UNITS=0, EXTMODE=EBCDIC, INTMODE=EBCDIC, BLOCKED)
                6(06) 0 003C00 000016 TANKDATA2   (PHYSICAL: BLOCKSIZE=00060, MINRECSIZE=00000, MAXRECSIZE=00030)
                7(07) 0 002002 000000 DISKBLOCK   (SEGPERBLK=002, RECPERBLK=002)
                8(08) 0 000800 000000 PAGESPEC
                9(09) 0 030000 000351 IOINFO   (KIND=DISKPACK)
               10(0A) 5 800001 405EEE LEB
                         0(0000) 0 C60030 000011 0 000001 000001 0 000000 000000 0 020001 400064 0 400057 CE36A3
                         5(0005) 0 003C00 00001E 0 000000 000000 0 000000 000000 0 020000 000000 0 000065 001000
                        10(000A) 0 000000 000000 THRU 11(000B)
                        12(000C) 0 000000 000010 0 000000 000012 0 070201 03C4E2 0 020000 000000 0 070101 03C4E2
                        17(0011) 0 020000 000000 0 040102 010000 0 021100 000000
               11(0B) 5 C00005 304599 IOAREA
                      (BUFFER #1) ----------------------------------------------------------------
                         NEXTLINK: 0 000000 000000
                         SIDELINK: 0 000000 000000
                      IOMAREADESC: 0 000000 000000
                         IOMIOCW: 0 000000 000000
                          IOMCOL: 0 000000 000000
                            IORD: 0 000000 000000
                      IOMTIMECELL: 0 000000 000000
                        LASTIOCW: 0 000000 000000
                            USER: 3 380000 000041
                          IOMASK: 0 000000 000000
                        AREADESC: 5 800003 C045AF
                        TIMECELL: 0 000000 000000
                            EVNT: 5 E10000 E04599
                         FIBDESC: 5 C00003 70740F
                       BFFREVENT: 0 000000 000001
                      BFFREVENT2: 0 000000 000000
                            IOAL: 0 0006F0 C06F0C
                            IOAW: 0 000000 000000
                     ACTUALBLOCK: 0 400000 000001
                      ACTUALKEY2: 0 000000 000000
                        DUPLCOPY: 0 000000 000000
                           JUNK1: 0 000000 000000
                           IOCWP: 0 030000 000351
                        USER DATA: (SHOWN BELOW IN HEX AND CORRESPONDING GRAPHICS)
                0(0000) C8C940 E3C8C5  D9C500 000000  000000 000000  000000 000000  000000 000000  /HI THERE??????????????????????/
                5(0005) 000000 000000  000000 000000  000000 000000  000000 000000  000000 000000  /????????????????????????????/
               10(000A) 000000 000000  000000 000000  000000 000000  000000 000000  000000 000000  /????????????????????????????/
               15(000F) 000000 000000  000000 000000  000000 000000  000000 000000  000000 000000  /????????????????????????????/
               20(0014) 000000 000000  000000 000000  000000 000000  000000 000000  000000 000000  /????????????????????????????/
               25(0019) 000000 000000  000000 000000  000000 000000  000000 000000  000000 000000  /????????????????????????????/
               30(001E) 000000 000000  000000 000000  000000 000000  000000 000000  000000 000000  /????????????????????????????/
               35(0023) 000000 000000  000000 000000  000000 000000  000000 000000  000000 000000  /????????????????????????????/
               40(0028) 000000 000000  000000 000000  000000 000000  000000 000000  000000 000000  /????????????????????????????/
               45(002D) 000000 000000  000000 000000  000000 000000  000000 000000  000000 000000  /????????????????????????????/
               50(0032) 000000 000000  000000 000000  000000 000000  000000 000000  000000 000000  /????????????????????????????/
               55(0037) 000000 000000  000000 000000  000000 000000  000000 000000  000000 000000  /????????????????????????????/
                      (BUFFER #2) ----------------------------------------------------------------
                         NEXTLINK: 0 000000 000000
                         SIDELINK: 0 000000 000000
                      IOMAREADESC: 0 000000 000000
                         IOMIOCW: 0 000000 000000
                          IOMCOL: 0 000000 000000
                            IORD: 0 000000 000000
                      IOMTIMECELL: 0 000000 000000
                        LASTIOCW: 0 000000 000000
                            USER: 0 380000 000041
```

FIB1

FIBSTACK/INFO Program Dump (Sheet 1 of 2)

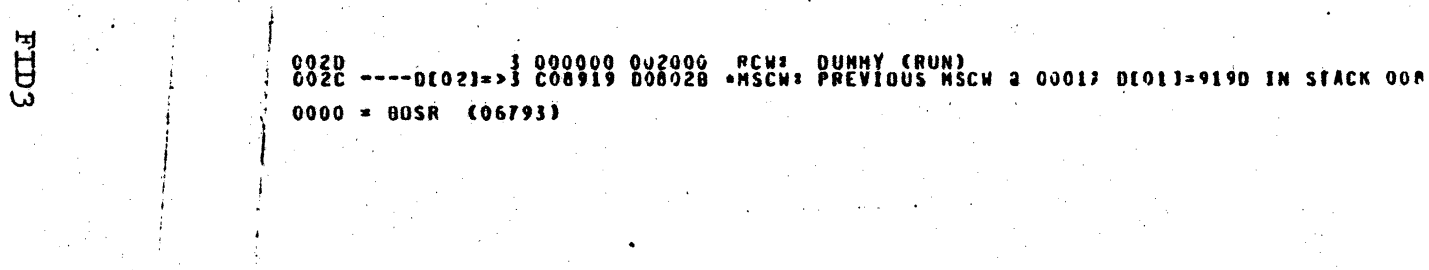
```
                                    003:000E:0  LT8                 92F9
                                    003:000E:2  TSOL                9A1E30
                                    003:000E:5  LT8                 9205
                                    003:000F:1  STAG                9534
(02,00003)  = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
                                    003:000F:3  NAMC  (01,00000)   6000
                                    003:000F:5  STFF                AF
                                    003:0010:0  BSET                960D
(02,00004)  = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
                                    003:0010:2  LT4B                BE
                                    003:0011                        270000740001 (3"116000003500000A")
                                    003:0012:0  LT8                 9205
                                    003:0012:2  STAG                9594
                                    003:0012:4  LT4B                BE
                                    003:0013                        800000002800 (3"400000000024000")
                                    003:0014:0  LT8                 9206
                                    003:0014:2  STAG                9594
                                    003:0014:4  BRUN      000011    A22000
                                    003:0015:1  NVLD                FF
                                    003:0015:2  NVLD                FF
                                    003:0015:3  NVLD                FF
                                    003:0015:4  NVLD                FF
                                    003:0015:5  NVLD                FF
                                                                                                    8.0000(003) IS 0016 LONG
*****************************************************************************************************************************
NUMBER OF ERRORS DETECTED = 0.
NUMBER OF SEGMENTS = 4.  TOTAL SEGMENT SIZE = 29 WORDS.  CORE ESTIMATE = 871 WORDS.  STACK ESTIMATE = 5
PROGRAM SIZE = 8 CARDS, 58 SYNTACTIC ITEMS, 11 DISK SEGMENTS.
PROGRAM FILE NAME:  FIBSTACK/INFO. 87700 CODE GENERATED.
COMPILATION TIME = 1.427 SECONDS ELAPSED:  0.347 SECONDS PROCESSING:  0.547 SECONDS I/O.
*****************************************************************************************************************************
```

FSI2

```
                    IOMASK: 0 000000 000000
                  AREADESC: 5 000003 C06F22
                  TIMECELL: 0 000000 000000
                      EVNT: 5 E10000 E06F0C
                   FIBDESC: 5 C00003 70740F
                BFFREVENT1: 0 200J00 000001
               BFFREVENT2: 0 000000 000000
                      IOAL: 0 000459 004599
                      IOAW: 0 000000 000000
              ACTUALBLOCK: 0 400000 000001
                ACTUALKEY2: 0 000000 000000
                  DUPLCOPY: 0 300000 000000
                     JUNK1: 0 000000 000000
                      IOCWP: 0 030000 000351
                 USER DATA: (SHOWN BELOW IN HEX AND CORRESPONDING GRAPHICS)
  0(0000) 000000 000000  000000 000000  000000 000000  000000 000000  000000 000000  //////////////////////////////
  5(0005) 000000 000000  000000 000000  000000 000000  000000 000000  000000 000000  //////////////////////////////
 10(000A) 000000 000000  000000 000000  000000 000000  000000 000000  000000 000000  //////////////////////////////
 15(000F) 000000 000000  000000 000000  000000 000000  000000 000000  000000 000000  //////////////////////////////
 20(0014) 000000 000000  000000 000000  000000 000000  000000 000000  000000 000000  //////////////////////////////
 25(0019) 000000 000000  000000 000000  000000 000000  000000 000000  000000 000000  //////////////////////////////
 30(001E) 000000 000000  000000 000000  000000 000000  000000 000000  000000 000000  //////////////////////////////
 35(0023) 000000 000000  000000 000000  000000 000000  000000 000000  000000 000000  //////////////////////////////
 40(0028) 000000 000000  000000 000000  000000 000000  000000 000000  000000 000000  //////////////////////////////
 45(002D) 000000 000000  000000 000000  000000 000000  000000 000000  000000 000000  //////////////////////////////
 50(0032) 000000 000000  000000 000000  000000 000000  000000 000000  000000 000000  //////////////////////////////
 55(0037) 000000 000000  000000 000000  000000 000000  000000 000000  000100 000000  //////////////////////////////
  12(0C) 5 C00003 C04580 BUFFDESC
  13(0D) 0 000000 000000 STDAREA
  14(0E) 5 000001 120001 FMTBUFFDESC
  15(0F) 0 000000 000000 FMTLOCK
  16(10) 0 000000 000000 FILIOTIME
  17(11) 7 408000 008693 PWRITES      (66703500)
  18(12) 7 408000 008768 PREADS       (70039500)
  19(13) 7 408000 00876C PWRITEN      (70102000)
  20(14) 7 408000 008760 PREADN       (70158500)
  21(15) 7 408000 00876E PSEEK        (70198200)
  22(16) 7 408000 008723 PREADR       (67735100)
  23(17) 7 408000 008739 PSEARCH      (68333000)
  24(18) 7 408000 0086AB PLOCKER      (66625400)
  25(19) 7 408000 008735 PRELEASE     (68133000)
  26(1A) 7 408000 0086DF PMOVEOUT     (67195500)
  27(1B) 7 408000 0086E0 PMOVEIN      (67199000)
  28(1C) 7 408000 008746 PWAIT        (68799500)
  29(1D) 7 408000 008779 PFLOAT       (71611230)
  30(1E) 0 00F900 330038 PCWCONTROL   (66703500) (66643500) (67578000)
  31(1F) 0 000000 00001E OFFSET
  32(20) 0 000000 000000 RECORDCOUNT
  33(21) 0 000000 000000 BLOCKCOUNT
  34(22) 0 400000 000001 LOWER
  35(23) 0 400000 000001 UPPER
  36(24) 0 000000 00001E MINRECSZ
  37(25) 0 000000 00001E RECSIZE
  38(26) 0 000000 000011 I
  39(27) 0 000000 000000 I
  40(28) 0 000000 000000 AEXP
  41(29) 1 419000 000000 DHEADER
                0(0000) 0 000035 600000 0 0010C0 002800 0 040000 000000 0 003C00 00001E 0 241817 B00A21
                5(0005) 0 300014 000064 0 C01358 80EC00 0 0C0000 013580 0 000000 000000 0 000000 000000
               10(000A) 0 000001 000000 0 000000 000000 0 000000 000000 0 000000 600000 0 000000 000000
               15(000F) 0 000000 CC0000 THRU 39(0027)
               40(0028) 0 070201 03C4E2 0 D20000 000000 0 000000 000000
  42(2A) 0 000000 000000 FIBEOF
  43(2B) 0 000000 000000 ACTNUM
  44(2C) 0 000000 000000 SBLOCKING
  45(2D) 0 000000 000000 SIOINFO
  46(2E) 0 000000 000000 SOFFSET
  47(2F) 0 000000 00003C CURRENTBLOCK
  48(30) 2 000000 000000 FILEEVENT1
  49(31) 2 000000 000000 FILEEVENT2
  50(32) 0 000000 000000 OUTPUTTRANSLATION
  51(33) 0 000000 000000 INPUTTRANSLATION
  52(34) 0 000000 000000 USEROUTINES
  53(35) 0 000003 000000
  54(36) 0 000000 000000 FIBLOCKSNR
0031 (02,0003)  1 4EE001 602000  SIRW: OFFSET=0016 (0016+0000) IN STACK 0EE
0030 (02,0002)  5 800001 E2265C  DESC (PPRESENT-MOM): DATA, LENGTH=     30
        0(0000) 0 CAC940 E3C8C5 0 09C500 000000 0 000C00 C00C00 0 000000 000000 0 000000 000000 0 000000 000000
        6(0006) 0 000000 000000 THRU 29(001D)
002F            3 000006 088451  RCW:  LL=02, CNTRL STATE                  (MCP SEGMENT       ) 0451:000010 (2506622C)]
                                 SEG DESC:  3 80003E A38E24
                                 CODE: >3 823482 269588< 3 820280 879587
002E ----D(02)=>3 CEE001 600002 *MSCW: PREVIOUS MSCW 2 002C? D(01)=0016 IN STACK 0EE
```

FID2

FIBSTACK/INFO Program Dump (Sheet 2 of 2)



FLD3

FIRSTACK/INFO Program Dump (Cl... ... ...)

```
    1     $SET INTRINSICS LIST STACK LINEINFO
    5     $LOADINFO
10000     PROCEDURE FCKINT(CASENO,ARY);
10100     VALUE CASENO,ARY;
10200     REAL CASENO;
10300     ARRAY ARY[*];
10400     BEGIN
10500       PROCEDURE P;
10600         NULL;
10700       WORD WP = P,4;
10800       WORD ARRAY STK[*];
10900       REAL INX,I,J,K;
11000       CASE CASENO OF
11100       BEGIN
11200         BEGIN % CASE 0 - THE REGISTERS
11300         ARY[0]:=D0;
11400         ARY[1]:=D1;
11500         ARY[2]:=D2;
11600         ARY[3]:=F;
11700         ARY[4]:=S;
11800         ARY[5]:=LOSR;
11900         ARY[6]:=BOSR;
12000         ARY[7]:=SNR;
12100         END;
12200         BEGIN % CASE 1 - STACK NUMBERS
12300         ARY[0]:=SNR;
12400         ARY[1]:=(W:=WORDSTACK[SNR,STUFFITMSCW]).SEGDICTF;
12500         ARY[2]:=M[F].STKNRF;
12600         ARY[3]:=INTRINSICSTKNUM;
12700         END;
12800         BEGIN % CASE 2 - PROGRAM ID
12900         STK:=WORDSTACK[SNR,TASKDESC];
13000         I:=STK[MYNAME].[15:16];
13100         REPLACE POINTER(ARY[0],8) BY POINTER(STK[I],8)
13200          FOR STK[I].TOTALCHRSF;
13300         END;
13400         BEGIN % CASE 3 - RCW MSCW CHAIN
13500         STK:=STACKVECTOR[SNR];
13600         INX:=F-BOSR; % MY MSCW
13700         I:=-1;
13800         DO
13900            BEGIN
14000            ARY[I:=I+1]:=STK[INX+1] & SETTAG(0); % RCW
14100            ARY[I:=I+1]:=STK[INX] & SETTAG(0); % MSCW
14200            END
14300         UNTIL INX:=INX-STK[INX].DFF LSS MSCWOFEOJ;
14400         END;
14500         BEGIN % CASE 4 - MEMORY AREAS
14600         DISALLOW;
14700         TOUCH(ARY[0]);
14800         SUZZ(MEMLOCK);
14900         W:=M[I:=MEM[0].ADDRESSF];
15000         DO
15100            BEGIN
15200            IF W.SIZEF GTR INX THEN
15300               BEGIN
15400               J:=I;
15500               INX:=W.SIZEF;
15600               END;
15700            END
15800         UNTIL BOOLEAN((W:=M[W.ADDRESSF]).STOPPERF);
15900         ARY[0]:=INX-4; % SUBTRACT OFF THE LINKS
16000         ARY[1]:=J+2; % THE START OF THE AREA
16100         ARY[3]:=M[I:=LISTLOOKUP(4"7FFFFFF",M,MEMBASE+1)].SIZEF-4;
16200         ARY[2]:=I-ARY[3]-1; % TO MEMORY AREA;
16400         UNLOCK(MEMLOCK);
16500         END;
16600         BEGIN % CASE 5 - STFW TO TASK AREA
16700         ARY[0]:=0 & STUFFEDIRW(0,,(W:=WORDSTACK[SNR,TASKDESC]).[19:4]+
16800          PSEUDOSTACKBASE,W);
16900         END;
17000         BEGIN % CASE 6 - PROCEDURE CALL
17100         WP:=6 & STUFFEDIRW(,,SNR,F-M[F].OFF-BOSR);
17200         P;
17300         END;
17400         BEGIN % CASE 7 - INTRINSICS NAME
17500         PROCURE(MISSINGINTRINSICEVENT);
17600         REPLACE POINTER(ARY[0],8) BY POINTER(MCPINFO[105],8) FOR
17700          MCPINFO[105].TOTALCHRSF;
17710         LIBERATE(MISSINGINTRINSICEVENT);
```

```
17800          END;
17900          BEGIN % CASE 8 - DISK READ
18000          DISKWAIT(
18100           ARY, % BUFFER
18200           -1, % INX TO IOCW (LINKC)
18300           30, % WORD COUNT
18400           0 & DISKADDRESSL(HLUNIT), % ADDRESS AND UNIT NUMBER
18500           DISKREAD); % HIGH ORDER BITS OF IOCW
18600          END;
18700          BEGIN % CASE 9 - DISK READ WITH TAGS
18800          DISKWAIT(ARY,-1,30,0 & DISKADDRESSL(HLUNIT),DISKREADWTAGS);
18900          PROGRAMDUMP(MCPREQUEST ARRAYS);
19000          REPLACE POINTER(ARY) BY 0 FOR ARY.LENGTHF OVERWRITE;
19100          END;
19200          BEGIN % CASE 10 - RESIZE ARY TO 60 WORDS
19300          MAKEPRESENTANDSAVE(ARY); % THE EASY WAY OUT
19400          I:=ARY.ADDRESSF; % THE AREA - DO NOT USE ARY AGAIN
19500          INX:=M[I-3].ADDRESSF; % MOM ADDRESS
19600          STK:=STACKVECTOR[SNR];
19700          J:=K:=F-BOSR; % SEARCH BELOW ME ONLY
19800          W:=I & 1 PBITF & 5 TAG;
19900          WHILE K:=MASKSEARCH(W,4"FFFFF" & 1 PBITF & 7 TAG,STK[K]) GTR
20000           MSCWOFEOJ DO
20100             BEGIN
20200             IF STK[K].IBITF THEN
20300                STK[K]:=* & 0 PBITF & INX ADDRESSF
20400             ELSE
20500                STK[K]:=* & 0 PBITF & 60 LENGTHF & INX ADDRESSF;
20600             K:=*-1;
20700             END;
20800          K:=J;
20900          W:=INX & 5 TAG;
21000          WHILE K:=MASKSEARCH(W,4"FFFFF" & 1 PBITF & 7 TAG,STK[K]) GTR
21100           MSCWOFEOJ DO
21200             BEGIN
21300             IF NOT STK[K].IBITF THEN
21400                STK[K]:=* & 60 LENGTHF;
21500             K:=*-1;
21600             END;
21700          FORGETSPACE(I);
21710          DISALLOW;
21800          I:=GETSPACE(60,SNR,0 & USAGEL(,,1),INX);
21900          M[INX]:=ARY:=I & 1 PBITF & 60 LENGTHF & 5 TAG;
21910          ARY:=*;
21920          REPLACE POINTER(ARY) BY 0 FOR 60 OVERWRITE;
22000          END;
22100          BEGIN % CASE 11 - STACK INFORMATION
22200          I:=MAXSTACKS-1;
22300          PROCURE(PROCESSCHANGELOCK);
22400          WHILE I:=MASKSEARCH(0 & 1 STACKKINDF,DUPLICATE,STACKINFO[I])
22500           GEQ MCPSTACK DO
22600             BEGIN
22700             IF J:=STACKKIND(I) EQL TASKSTACK THEN
22800                INX:=*+1
22900             ELSE IF J EQL SEGDICTSTACK THEN
23000                K:=*+1;
23100             I:=*-1;
23200             END;
23300          LIBERATE(PROCESSCHANGELOCK);
23400          ARY[0]:=INX;
23500          ARY[1]:=K;
23600          END;
23610          BEGIN % CASE 12 - CODE FILE NAME
23620          STK:=STACK[WORDSTACK[SNR,STUFFITMSCW].SEGDICTF,*]; % D1 STACK
23630          STK:=M[STK[CODEFILEDESC]]; % HEADER
23640          REPLACE POINTER(ARY[0],8) BY HEADERTITLE(STK) FOR
23650           STK[HEADERSIZE(STK)].TOTALCHRSF;
23660          END;
23700       END;
23800       EXIT; % DONT LET BLOCKEXIT SEE ANY STRANGE MOMS IN OUR STACK
23900    END.
```

TNTX/X (Sheet 2 of 2)

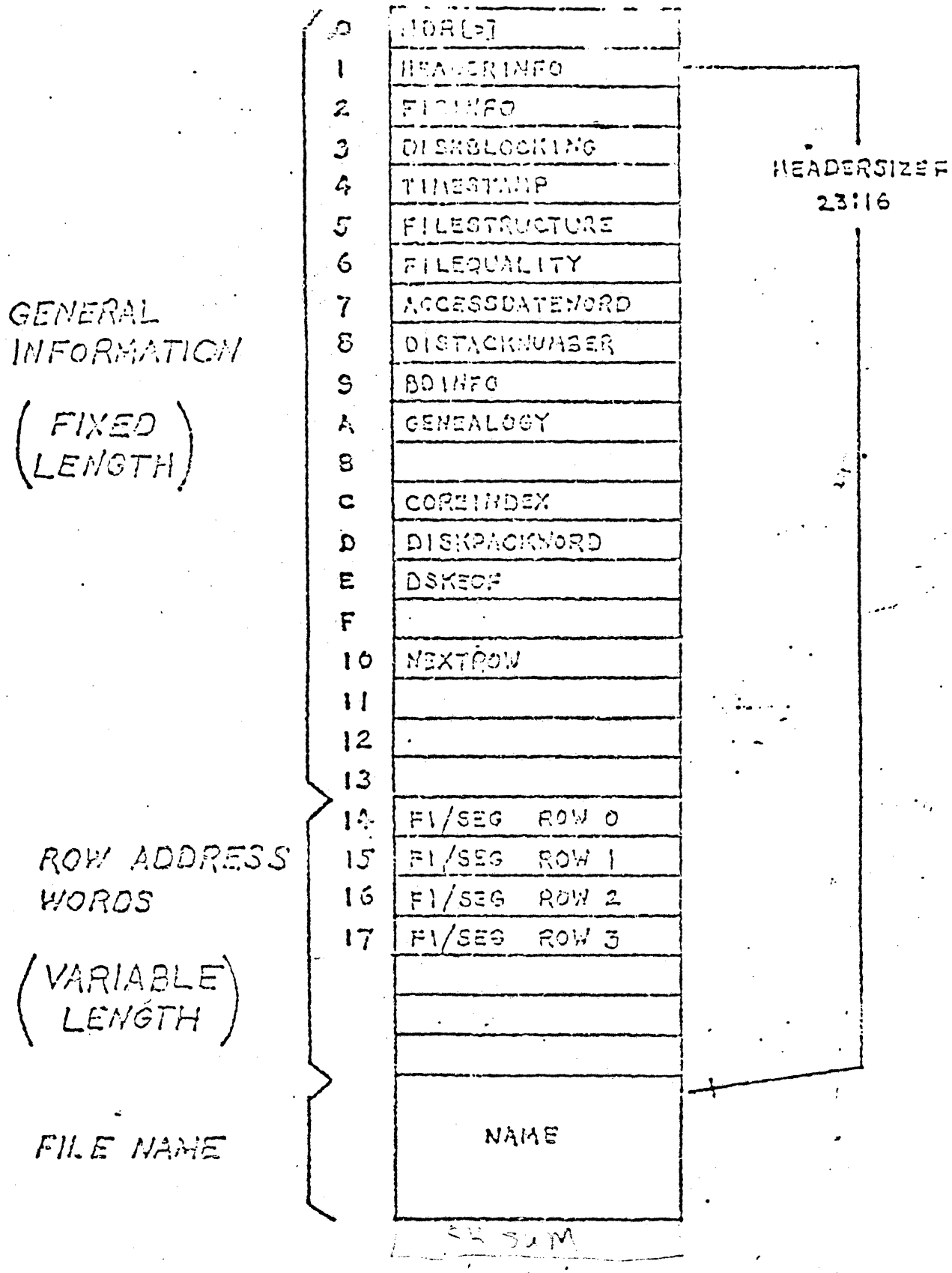


FIGURE 3. HEADER-NAME BLOCK.

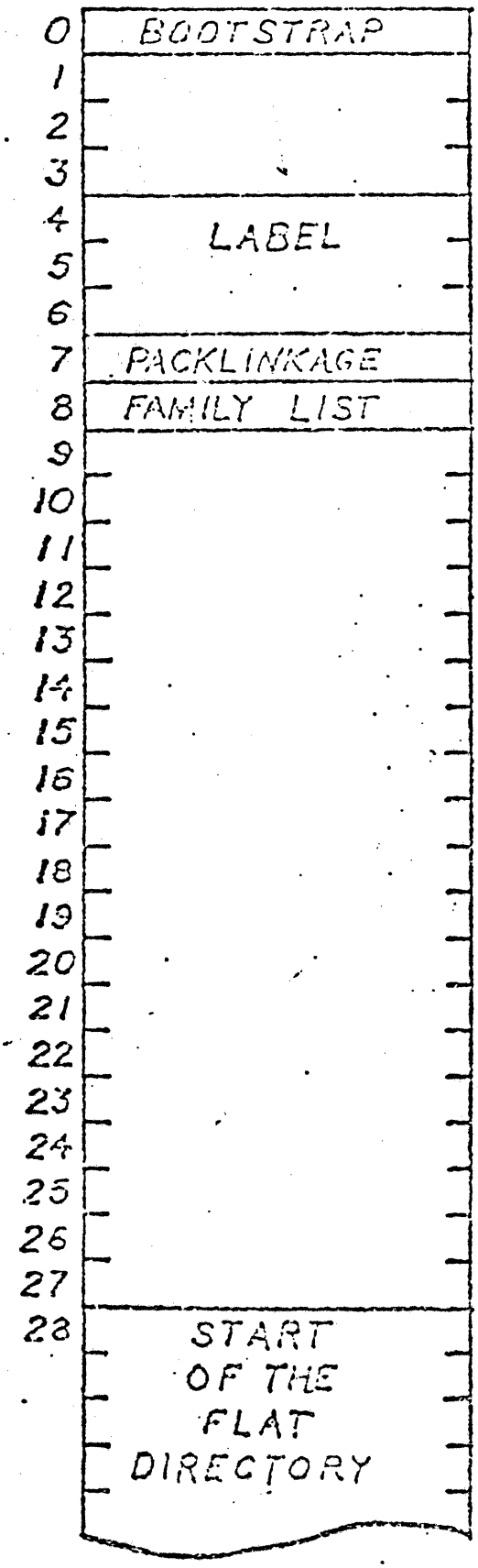


FIGURE 1 HALT/LOAD DISK LAYOUT

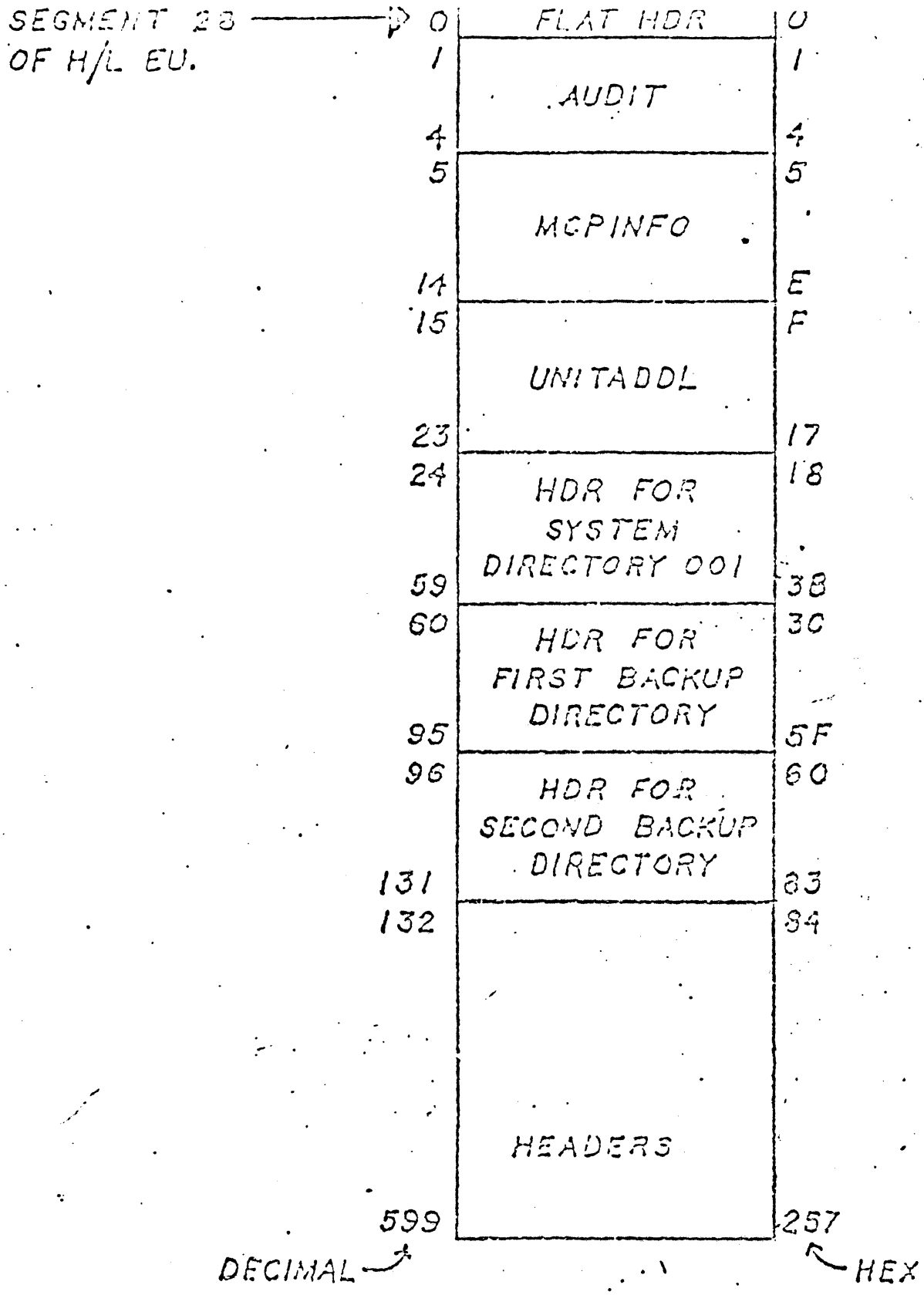


FIGURE 2 FIRST ROW OF THE FLAT DIRECTORY

THE PAST HAS A FIXED SIZE OF 75 BLOCKS, EACH OF 8 DISK SEGMENTS. TO FIND AN ENTRY IN THE PAST FOR THE FAMILY, A BLOCK IS CALCULATED BASED ON AN ARITHMETIC MANIPULATION OF THE CHARACTERS IN THE FAMILY NAME (IE. HASHED). THIS ENTRY CONTAINS POINTERS IDENTIFYING THE SECTION OF THE FAST WHICH IS APPLICABLE TO THE FAMILY. A SIMPLE SEARCH IS THEN PERFORMED ON THE PAST BLOCK UNTIL THE ENTRY WITH THE GIVEN FAMILY NAME IS FOUND. GIVEN BELOW IS AN EXAMPLE PAST ENTRY. NOTE THAT ALL PAST, AS WELL AS FAST BLOCKS, ARE CHARACTER ORIENTED (1440 CHARACTERS PER BLOCK).

CHARACTER	CONTENT	COMMENT
00	01	PNUM- NUMBER OF FAMILY MEMBERS.
01	04	PNAME- FAMILY NAME. FIRST CHARACTER IS THE NUMBER OF CHARACTERS IN THE NAME.
02	C4	
03	C9	
04	E2	
05	D2	
06	00	PSERIAL- SERIAL NUMBER IF PACK AND UNIT NUMBER IF HPT.
07	00	
08	20	
09	18	PTIMESTAMP- TIME THIS ENTRY WAS MADE.
10	BB	
11	5B	
12	DB	
13	70	
14	34	
15	00	PFASTIX- BEGINNING AND ENDING FAST BLOCKS FOR THIS FAMILY (BEGINNING OF FILE RELATIVE).
16	09	
17	40	
18	00	
19	95	

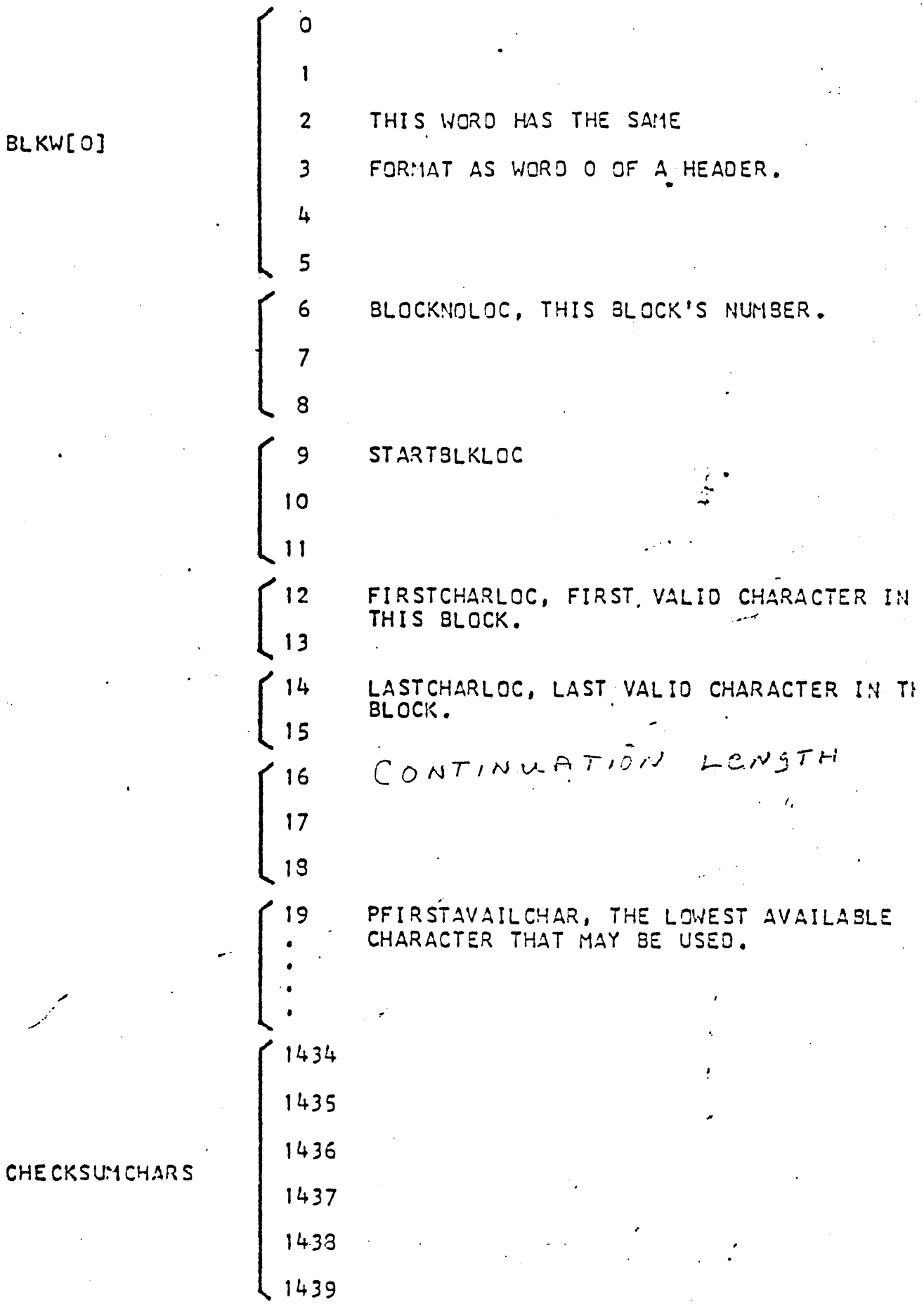


FIGURE 4 PACK ACCESS STRUCTURE (PAST) BLOCK.

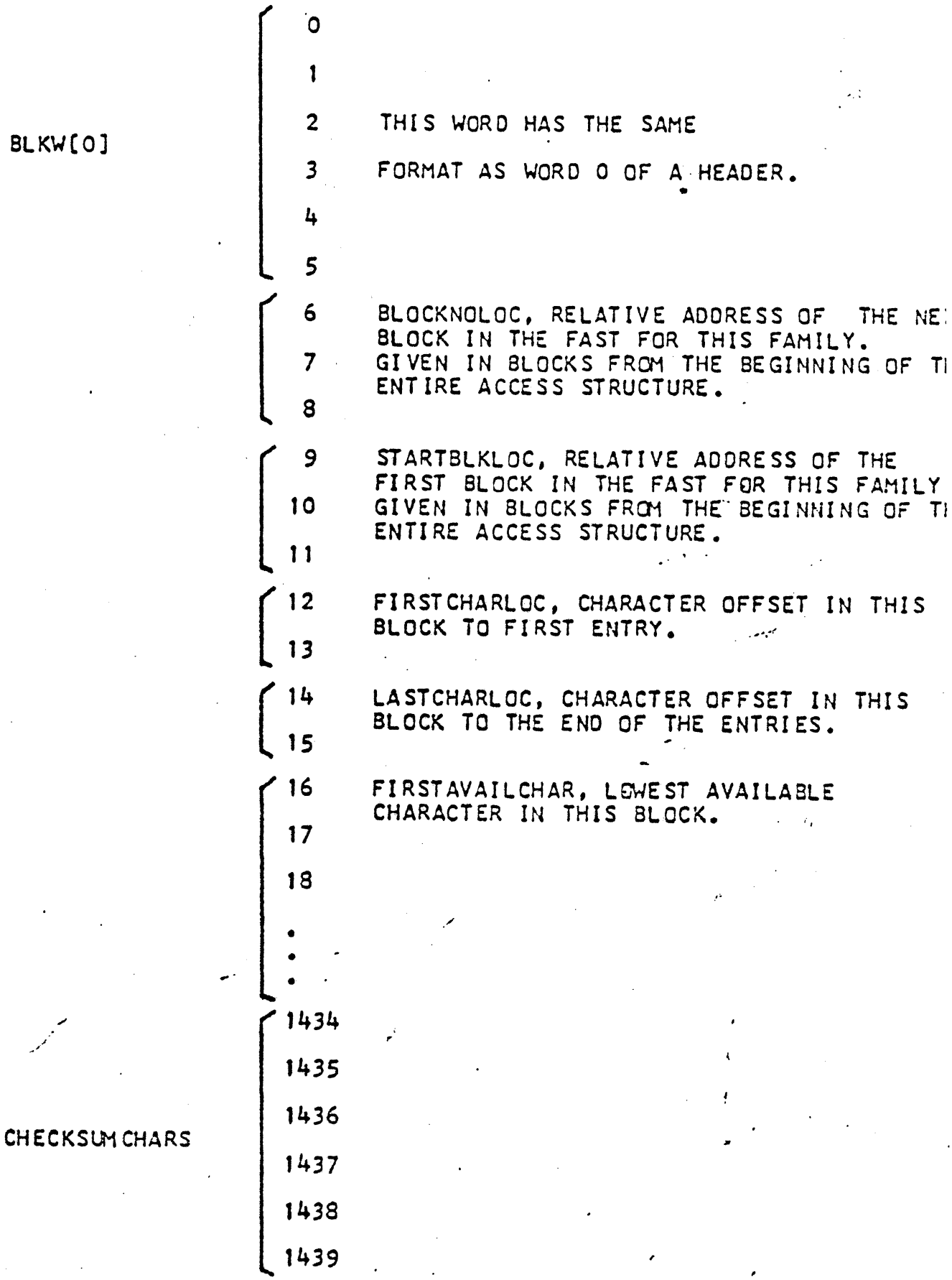


FIGURE 5 FILE ACCESS STRUCTURE (FAST).

AN EXAMPLE FAST ENTRY FOR FILE C HAS BEEN GIVEN BELOW:

CHARACTER	CONTENT	COMMENT
00	27	POINTS TO A FILE AND HAS NO BROTHER.
01	01	THERE IS 1 CHARACTER IN THE FILE'S NAME.
02	C3	NAME OF THE FILE IS 8"C"
03	00	HEADER IS AT OFFSET 4"F00" IN THE FLAT DIRECTORY AND IS 32 WORDS LONG.
04	40	
05	0F	
06	00	

THE WAY THE ABOVE INFORMATION WAS OBTAINED WAS BY USE OF THE INFORMATION PROVIDED BELOW:

CHARACTER 0 IN A FAST ENTRY IS CALLED THE "ID" CHARACTER AND HAS THE FOLLOWING FORMAT:

7:1 BROTHERF, ON IF THIS FILE HAS A BROTHER.

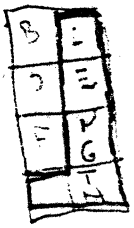
6:2 IDTYPEF, BROKEN DOWN AS FOLLOWS:

6:1 DIR, ON IF THIS ENTRY POINTS TO A DIRECTORY ENTRY.

5:1 FYLE, ON IF THIS ENTRY POINTS TO A FILE.

4:5 ENTRYL, LENGTH OF THIS ENTIRE ENTRY IN CHARACTERS. INCLUSIVE!

CHARACTER 1 IN A FAST ENTRY IS THE LENGTH OF THE NAME IN CHARACTERS. (NOT SELF INCLUSIVE)



CHARACTERS 2 THROUGH N ARE THE NAME. "N" REPRESENTS THE LAST CHARACTER IN THE NAME.

IF THE FYLE BIT IS ON, THE FOUR CHARACTERS FOLLOWING THE NAME ARE IN THE FORMAT GIVEN BELOW:

31:11 FILELENGTHF, LENGTH OF THE HEADER IN WORDS.

20:21 FILELOCF, POINTS TO THE HEADER IN THIS FAMILY'S FLAT DIRECTORY AND IS GIVEN IN SEGMENTS RELATIVE TO THE BEGINNING OF THE FLAT DIRECTORY.

IF THE THE FYLE BIT AND THE DIR BIT HAD ALSO BEEN ON THEN THE FOUR CHARACTERS DESCRIBED IMMEDIATELY ABOVE WOULD HAVE BEEN FOLLOWED BY TWO CHARACTERS OF THE FOLLOWING FORMAT (NOTE THAT THIS IS NOT THE CASE IN THE EXAMPLE ABOVE):

15:16 DIRDELF, POINTER TO THE NEXT LEVEL AND IS GIVEN IN CHARACTERS RELATIVE TO THE BEGINNING OF ~~THE BLOCK.~~ THIS ENTRY

IF THE DIR BIT HAD BEEN ON AND THE FYLE BIT HAD NOT BEEN ON THEN THE "4" CHARACTERS DESCRIBED ABOVE WOULD HAVE NOT EXISTED IN THE ENTRY AND IN THIS SPECIFIC ENTRY, THERE WOULD HAVE BEEN ONLY 5 CHARACTERS.

FIGURE 5 SHOWS HOW AN ENTIRE FAST BLOCK IS LAYED OUT. NOTICE ITS SIMILARITY TO THE PAST BLOCK LAYOUT OF FIGURE

ABSOLUTE SEGMENTS 4 AND 5 CONTAIN THE LABEL ON DISK UNITS. THIS LABEL IS ORIGINALLY SET UP BY THE SYSTEM LOADER IN THE PROCEDURE "WRITEHPTLBL" AND CONTAINS THE FOLLOWING INFORMATION:

```
8"VOL1"                                 % VOLUME 1 LABEL
UNIT FOR 6 DIGITS                       % SERIAL NUMBER IF PACK AND
                                        % UNIT NUMBER IF DISK
8" "
8"DISK          "                       % FAMILY NAME
8"67"                                   % INTERCHANGE CODE
8"0" FOR 1                              % USAGE CODE
8" " FOR 6                              % RESERVED
8" " FOR 14                             % OWNER FIELD
8" " FOR 28                             % RESERVED
8" "                                    % INDICATES THAT THIS IS THE
                                        % BASE UNIT
8"VOL2"                                 % VOLUME 2 LABEL
MCPINFO[1] FOR 5 DIGITS                 % INITIALIZATION DATE
8"67MC"                                 % INITIALIZATING THE SYSTEM
MARK
LEVEL
MCPINFO[7].SEGADDRESSF FOR 8 DIGITS     % ABSOLUTE SEGMENT OF THE
                                        % START OF THE FLAT DIRECTORY.
8"00000002"                             % ABSOLUTE STARTING SEGMENT OF
                                        % THE MASTER AVAILABLE TABLE.
8"00000000"
8"0" FOR 20                             % RESERVED
8"1" FOR 1                              % IN USE FLAG
8"0" FOR 70                             % RESERVED
```

MISCELLANEOUS INFORMATION

ABSOLUTE SEGMENT 7 ON DISK IS ORIGINALLY BUILT BY THE SYSTEM LOADER IN THE PROCEDURE "WRITEHPTLBL." ITS WORDS CONTAIN THE FOLLOWI INFORMATION

WORD	CONTENTS
0	BASE UNIT
1	TIME OF DAY
2	DATE (FROM MCPINFO[1])
3	SITE NUMBER
4	INITIALIZED TO A VALUE OF 1 BY THE LOADER
28	INITIALIZED TO A VALUE OF 4A3 (DEFAULTRESERVEDHLDISKSIZE)?B THE LOADER
29	INITIALIZED TO 5 (CURLABLEVEL BY THE LOADER

WORD 0

47:16 MARKERF — Records state of this reader. States are:

HEADERMARK — 4"3F3F" header of file

INUSEMARK

AVAILMARK — 4"3C3C" not in use area within header block. Will be used if system needs to enter file header.

BADAREAMARK — 4"3A3A" not in use area that the MCP will not use as a fault was detected in directory header that used to reside at that disk address.

MCPUSEMARK — 4"3939" in use file by the MCP and it is not to be removed. What would happen if MCP file header is removed?

Note that the MARKF is so arranged that it easily can be searched for with a SCAN pointer expression UNTIL construct. Also note that single bit error can be detected.

31:11 HDRBLØCKLENGTHF — Size of this header block. Points to next header block.

20:21 HDRLØCATIØNF — Link back to the Access Structure for a redundancy check.

Available areas are specified as a length of 1.
Individual Job description headers not in system director nor access structure.

THE MCP INFORMATION TABLE, MCPINFO, IS ORGINALLY CREATED BY THE SYSTEM LOADER AND MAINTAINED BY THE MCP. THIS TABLE IS LOCATED IN A FIXED LOCATION IN THE FIRST ROW OF THE FLAT DIRECTORY AND ITS CONTENTS ARE SHOWN BELOW. IN THE LOADER, THE FIXED LOCATION OF MCPINFO IS REFERRED TO AS "MCPINFOLOC."

WORD	INFORMATION
0	HEADER[0] WORD FORMAT. WILL CONTAIN THE MCP USE MARK IN THE MARKERF, THE BLOCK LENGTH IN WORDS IN THE HDRBLOCKLENGTHF AND SEGMENT INDEX INTO THE FIRST ROW OF THE FLAT DIRECTORY WHERE THIS TABLE I FOUND (A SELF POINTER).
1	DATE
2	NOT USED. USE TO BE THE OLD CORE FACTOR USED IN MEMORY MANAGEMENT.
3	SIZE OF THE MCP'S NAME.
4	SIZE OF THE INTRINSIC FILE'S NAME.
5	CREATION DATE OF THE MCP FILE.
6	NOT USED. USE TO BE THE NUMBER OF PSEUDO-READERS.
7	DISK ADDRESS OF THE DIRECTORY HEADER. THIS LOCATION IS LEFT NEGATIVE BY THE SYSTEM LOADER AS NOTE TO THE MCP IF A COLD START IS BEING PERFORMED
8	DIRECTORY ROW SIZE. 600 SEGMENTS BY DEFAULT.
9	6"MCPINFO" USED BY THE LOADER OR THE MCP AS A PARTIAL WAY OF VALIDATING THE EXISTANCE OF THE MCPINFO TABLE.
10	NUMBER OF AUTO PRINTERS PRESENTLY RUNNING ON THE SYSTEM.
11	ENDING ADDRESS OF THE FIRST OVERLAY ROW.
12	HEADER SIZE OF THE DISK DIRECTORY.

14	SYSTEM SERIAL NUMBER.
15	THE CURRENT SUMLOG NUMBER.
16	OVERLAY ROW SIZE.
17	MEMORY DUMP DISK SPACE.
18	LAST MIX NUMBER USED.
19	SETTING OF THE PRESENT SYSTEM RUN TIME OPTIONS.
20	OLAYGOAL. MAY BE CHANGED WITH THE "SF" ODT MESSA
21	AVAILMIN. MAY BE CHANGED WITH THE "SF" ODT MESSA
22	FACTOR. MAY BE CHANGED WITH THE "SF" ODT MESSAG
24-27	DATACOM FILES PREFIX ID. "SYSTEM" BY DEFAULT BU MAY BE CHANGED WITH THE DC <PREFIX> <ETX> ODT MESSAGE.
29-31	AS FILES PREFIX
32	RSPINFO
37	DCPINFO.
38	MCPINFOLEVEL.
39	MEMORY PRIORITY FACTOR. MAY BE CHANGED WITH THE "SF" ODT MESSAGE.
40	RESERVING RESTART MASK.
41	SYSTEM/SUMLOG IAD ADDRESS.
42	SYSTEM/SUMLOG IAD ADDRESS.
43	SYSTEM/SUMLOG ROW LENGTH.
44	LOAD LEVEL FOR MULTIPLEXORS
45-52	RESERVED FOR B7700

53-58	SUBSTITUTE BACKUP INFO FROM THE "SB" MESSAGE.
60-104	BASE MCP CODE FILE NAME IN STANDARD FORM.
101-104	DM MONITOR ID.
105-147	SYSTEM/INTRINSICS ID.
148-179	SYSTEM SUPERVISOR ID.
180	CATALOG FAMILY SERIAL NUMBER.
181-184	CATALOG FAMILY TITLE.
272-280	MPx3 RESERVED PATH BITS
281-283	SYSTEM/SWAPDISK PACKNAME.
284-286	MEMORY DUMP DISK ALLOCATIONS.
287-289	CM # (MEMONLY) MCP CODE FILE HEADER DISK ADDRESSES
290-292	MCP CODE FILE HEADER DISK ADDRESSES.
293-295	OVERLAY ROW ADDRESSES.
300	CHECKSUM.

CREATION OF A LARGE SYSTEMS HALT LOAD PACK WITH THE CM METHOD

1. Mount the pack that is to become a H/L pack on a drive.

2. RC the pack with a name of DSK.

   ```
   RC PKNN NAME=DSK
   ```

3. Copy in all SYSTEM files.

   ```
   COPY & COMPARE = FROM SYSTEM TO DSK(PACK)
   ```

4. CM to the MCP. This will write the BOOTSTRAP and transfer some MCP structures from the current H/L unit to the new H/L unit. These structures include MCPINFO and unit table.

   ```
   CM SYSTEM/MCP ON DSK
   ```

5. Change the name of the pack to DISK.

   ```
   LB PKNN NAME=DISK
   ```

6. After the label change, halt system or remove pack.

7. To bring up system on new H/L unit, power off all packs except the new H/L unit.

8. Change H/L "POINTER" (BLOCK (B6700,B6800), IOM CARD (B7000), BOOTSTRAP via SOFTCON (B5900,B6900)) to point to new H/L unit.

9. Load the system. At some point STARTSYSTEM will hang in the waiting entries. It will be looking for the CATALOG FAMILY. STARTSYSTEM should be IL'ed to the new H/L unit.

   ```
   <STARTSYSTEM MIX NUMBER>ILPK NN
   ```

10. Another CM should be done to the currently running MCP. This will allow DUMPANALYZER to see the MCP properly.

    ```
    CM SYSTEM/MCP
    ```

11. To allow a USERDATAFILE (USERCODE file) to be created one must force the MCP forget the serial number of the cld H/L unit. Failure to do this step will result in the MCP trying to place the USERDATAFILE on the old H/L unit. In the following procedure it is assumed the site does not have a packed named "QQQSSSJJJ":

    ```
    DL USERDATA ON QQQSSSJJJ
    DL USERDATA ON DISK
    ```

12. One can now make a USERDATAFILE.

B O O T S T R A P C O D E

ADDRESS MEMORY		CONTENTS MEMORY	COMMENTS
000	0	8A0000000003	HOME ADDRESS WORD. SYNC I/O.
001	3	8000FEC00040	SEGMENT DESCRIPTOR.
002	0	A800B000001C	INFORMATION FOR MCP.
003	7	000000084001	PCW: LL 1,CNTL,1:000:0
004	0	000000000040	FOR HARDLOAD RESULT DESCRIPTOR.
005	0	0000FEC00040	BUFFER DESCRIPTOR.
006	0	121000000000	IOCW. READ FORCE TAG.
007	0	510100382875	CDL.
008	3	B0B095B9B234	ZERO,ZERO,SPRR,LT8 34
009	3	B31E0095B9B2	LT16 1E00,SPRR,LT8 35
00A	3	35B095B9B230	ZERO,SPRR,LT8 30
00B	3	B095B9B225B0	ZERO,SPRR,LT8 25,ZERO
00C	3	95B9AE400300	SPRR,MKST,NAMC 0,3
00D	3	000004981B20	VALC 0,0,VALC 0,4,FLTR 27:32:5
00E	3	054000BB0007	NAMC 0,0,OVRN,VALC 0,7
00F	3	B00004982714	ZERO,VALC 0,4,FLTR 39:20:4
010	3	0491400795BA	LOR,NAMC 0,7,RDLK
011	3	B5B22895B895	DELT,LT8 28,RPRR
012	3	8F95840005A1	INCN,PAUS,VALC 0,5
013	3	6012B08EB203	BRTR 12:3,ZERO,CHSN,LT8 3
014	3	95B44000BAAB	STAG,NAMC 0,0,OVRD,ENTR

HARDLOAD RESULT DESCRIPTOR AT HOME ADDRESS[4]

[16:17] ERROR FIELD
[24:08] UNIT NUMBER
[32:05] CHANNEL NUMBER

REVIEW QUESTIONS

SUBROUTINE EXECUTION

A PROCEDURE IS DECLARED AT LEXICOGRAPHIC LEVEL 3 AND CALLED AT LEXICOGRAPHIC LEVEL 5. WHAT DISPLAY REGISTER POINTS TO THE MSCW FOR THIS PROCEDURE. D3

IN A MARK STACK CONTROL WORD, THE DISPLACEMENT FIELD (BITS [35:16]) IS RELATIVE TO SOSR .

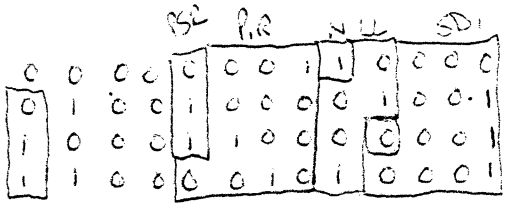
WHAT PROCESSOR REGISTER CONTAINS THE MEMORY ADDRESS OF THE MOST RECENTLY BUILT MARK STACK CONTROL WORD. F

THE "MOVE TO STACK" MACHINE INSTRUCTION CAUSES THE CENTRAL PROCESSOR TO ACCESS A DATA DESCRIPTOR AT 0[0] + 2. TO WHAT DOES THIS DATA DESCRIPTOR POINT. STACK VECTOR AREA

SEGMENT DESCRIPTORS CONTAINED IN THE USER PROGRAM SEGMENT DICTIONARY ARE ADDRESSED RELATIVE TO WHAT PROCESSOR REGISTER. D1

GIVEN THE FOLLOWING RETURN CONTROL WORD: 3 500621 894000

WHERE WILL THE PROCEDURE EXIT TO.



SEGMENT NUMBER 4

PIR 215

PSR 3

LEXICOGRAPHIC LEVEL 5

NORMAL OR CONTROL CONTROL

Identify What is Located $D_0 + 3$. Hardware Interrupt

REVIEW QUESTIONS
MACHINE LANGUAGE

(T) (F) MORE THAN ONE PROCESS CAN USE THE SAME SEGMENT DICTIONARY.

(T) (F) ONE PROCESS MAY ACCESS MORE THAN ONE SEGMENT DICTIONARY.
INTRINSICS, libraries, MCP

(T) (F) EACH CODE SEGMENT OF A PROGRAM MAY BE DESCRIBED BY SEVERAL SEGMENT DESCRIPTORS.

(T) (F) A CODE SEGMENT MAY BE REFERENCED BY SEVERAL PROGRAM CONTROL WORDS, EACH OF WHICH GIVE AN ENTRY POINT INTO THE CODE.

(T) (F) THE LEXICOGRAPHIC LEVEL AT WHICH A PROCEDURE WILL EXECUTE DEPENDS ON THE LEXICOGRAPHIC LEVEL OF THE CALLING ROUTINE.

(T) (F) A PROCEDURE RUNNING AT LEXICOGRAPHIC LEVEL THREE MAY DECLARE A PROCEDURE WHICH WILL RUN AT LEXICOGRAPHIC LEVEL THREE. EXPLAIN.

EXPLAIN THE FOLLOWING PROC REGISTERS

Stack overflow if J exceeds LOSR

S_r - Points to the top of stack in memory
F_r - Points to current logical bottom of stack, highest MSCW
D_0 - Points to the bottom of MCP, MSCW in it
D_1 - Points to segment dictionary of MCP
D_2 - Points to global variables
BOSR - hardware register,
LOSR
PSR - offset in current code word
PIR - offset to current word.
SDI -

A PROGRAM ABORTED AT 28 : 260 : 3

JUST BEFORE THE PROGRAM ABORTED, D[1] WAS SET TO 113D7, AND D[2] WAS SET TO 110BF.

THE CONTENTS OF SEVERAL MEMORY LOCATIONS ARE GIVEN BELOW:

ADDRESS			
01843	0 E53100 652780	0 1EF50B 01FA31	0 DABDAB DABD
113E3	5 080000 2407EB	3 800000 F15522	3 000000 0005
113F5	3 800001 122FA1	5 080018 A40870	3 800002 22150
113F8	5 880008 70C441	3 000001 0006A1	3 800000 016A4
113FE	3 800000 F1420E	3 800002 B15FE2	3 800001 315F3
11401	3 800003 013F	3 800028 A13C8A	3 800001 115CF
11404	3 000000 0005C3	5 080001 740382	3 000000 0006A
110C4	5 800001 011DA4	0 000000 000213	0 528000 040DC
13DB9	3 41EDBD 953E60	3 04B99A 2E01A0	3 218B20 029A1
13EF6	3 38A610 0388110	3 075005 A6700A	3 B8A262 40B38
14210	3 CB3453 8809B5	3 BFFFFF FFFFFF	3 5695A3 986C3
16A4A	3 B20380 958C95	3 B9AE60 2E3208	3 ABA3A3 423F2

DETERMINE WHY THE PROGRAM ABORTED

D1 113D7
+ SDT 28
Seg DESC. 11402

13C8A ADDRESS of SEG DESC.
260 P.R
13EF7 SEG DESC

A0 is INDEX + LOAD VALUE

1) Describe the function of each of the following words:

IRW (Indirect Reference word):
SIW (Step Index word):
Data Desc:
SCW (Software Control word):
PCW (Program control word):
MSCW (Mark Stack Control word):
RCW (Return Control word):
TOSCW (Top OF Stack Control word).
Tagged Desc:
SIRW (Stuffed Indirect Reference word).

2) Co[illegible]ate machine operator mnemonics for the following algol constructs:

A = (2, 1); B = (2, 3); C = (2, 4); D = (2, 5);
P = (2, 6).

A := B + 3 + 1;
A[A-2, B] := 5
C[1] := D[1, 1];

what algol constructs correspond to these code strings:

a) NAMC 2,2; VALC 2,3; NAMC 2,4; ONE; NXLV; ADD; NAMC 2,5; ONE; NXLV; ONE; NXLV; ADD; STOD.

b) NEST; NAMC 2,6; VALC 2,2; VALC 2,3; ENTR 2; 2; NEXT NAMC 2; 1; VALC 2,3; VALC 2,3; ENTR; STOD

3) DØ= 22ØØ ; SNR= 2 ; F = 4Ø2Ø

22ØØ = 3 4ØØ ØØ ØØ ØØ ØØ

22Ø2 = 5 6ØØ Ø2Ø ØØ1 ØØ Ø

1ØØØ = 5 8Ø Ø Ø4Ø Ø Ø2 ØØØ

1Ø1Ø = 5 8Ø Ø Ø4Ø Ø Ø3 Ø Ø Ø

1Ø22 = 5 8Ø ØØ4 ØØ Ø4 Ø ØØ

1Ø23 = 5 8Ø Ø Ø4 Ø Ø Ø5 Ø Ø Ø

1Ø24 = 5 3Ø1Ø4 Ø Ø Ø 6Ø Ø Ø

3Ø1Ø = 3 C Ø ØØ 2Ø Ø Ø 4 Ø Ø 9

4Ø1Ø = 3 C Ø 1Ø Ø1Ø Ø 8Ø Ø 9

42Ø = 3 C Ø 4Ø Ø1Ø Ø C Ø 1Ø

5Ø1Ø = 3 C Ø Ø Ø 2Ø Ø Ø 4 Ø Ø 9

6Ø1Ø = 3 C Ø 3Ø Ø 1Ø Ø 8 Ø Ø 9

The processor executes an exit operator. What values are contained in D2, D1, AND DØ after the operator completes?

4) what do you want to learn as a result of attending this class?

How to ANSWER all these questions correctly. I want to be able to ~~test~~ understand dumps better.

REVIEW QUESTIONS

SUBROUTINE EXECUTION

(T)(F) DISPLAY REGISTERS CONTAIN AN INDEX VALUE WHICH, WHEN ADDED TO THE CONTENTS OF THE "BOSR" REGISTER, LOCATES A MARK STACK CONTROL WORD.

(T)(F) THE D[2] REGISTER POINTS TO THE SAME LOCATION AS DOES THE BOSR REGISTER.

(T)(F) THE TOP-OF-STACK CONTROL WORD IS BUILT BY THE "MOVE TO STACK" MACHINE INSTRUCTION.

(T)(F) A STUFFED INDIRECT REFERENCE WORD MAKES IT POSSIBLE TO ADDRESS MEMORY LOCATIONS WHICH ARE NOT WITHIN ANY STACK.

True if being → PROCESS

(T)(F) MARK STACK CONTROL WORDS AND RETURN CONTROL WORDS ALWAYS OCCUR IN PAIRS. EXPLAIN.

generally they do, but not always

(T)(F) WHEN EXITING A PROCEDURE, IF BIT 13 OF THE RETURN CONTROL WORD IS A ONE, D[1] IS SELECTED AS THE SEGMENT DICTIONARY BASE REGISTER. OTHERWISE, D[0] IS SELECTED.

(T)(F) DISPLAY REGISTER ONE ALWAYS POINTS TO A PROGRAM SEGMENT DICTIONARY. EXPLAIN.

(T)(F) THE "STACK VECTOR" IS AN ARRAY OF DATA DESCRIPTORS, EACH OF WHICH DESCRIBES AN AREA OF MEMORY ALLOCATED AS A STACK OR SEGMENT DICTIONARY. ALL STACKS AND SEGMENT DICTIONARIES, INCLUDING THE MCP SEGMENT DICTIONARY, ARE THUS DESCRIBED.

(T)(F) THE MEMORY ADDRESS OF THE STACK VECTOR DESCRIPTOR DEPENDS ON THE SETTING OF THE D[0] REGISTER.

(T)( ) THE "F" REGISTER OF A CENTRAL PROCESSOR POINTS TO THE LAST MARK STACK CONTROL WORD IN THE STACK WHICH IS ACTIVE ON THAT PROCESSOR.

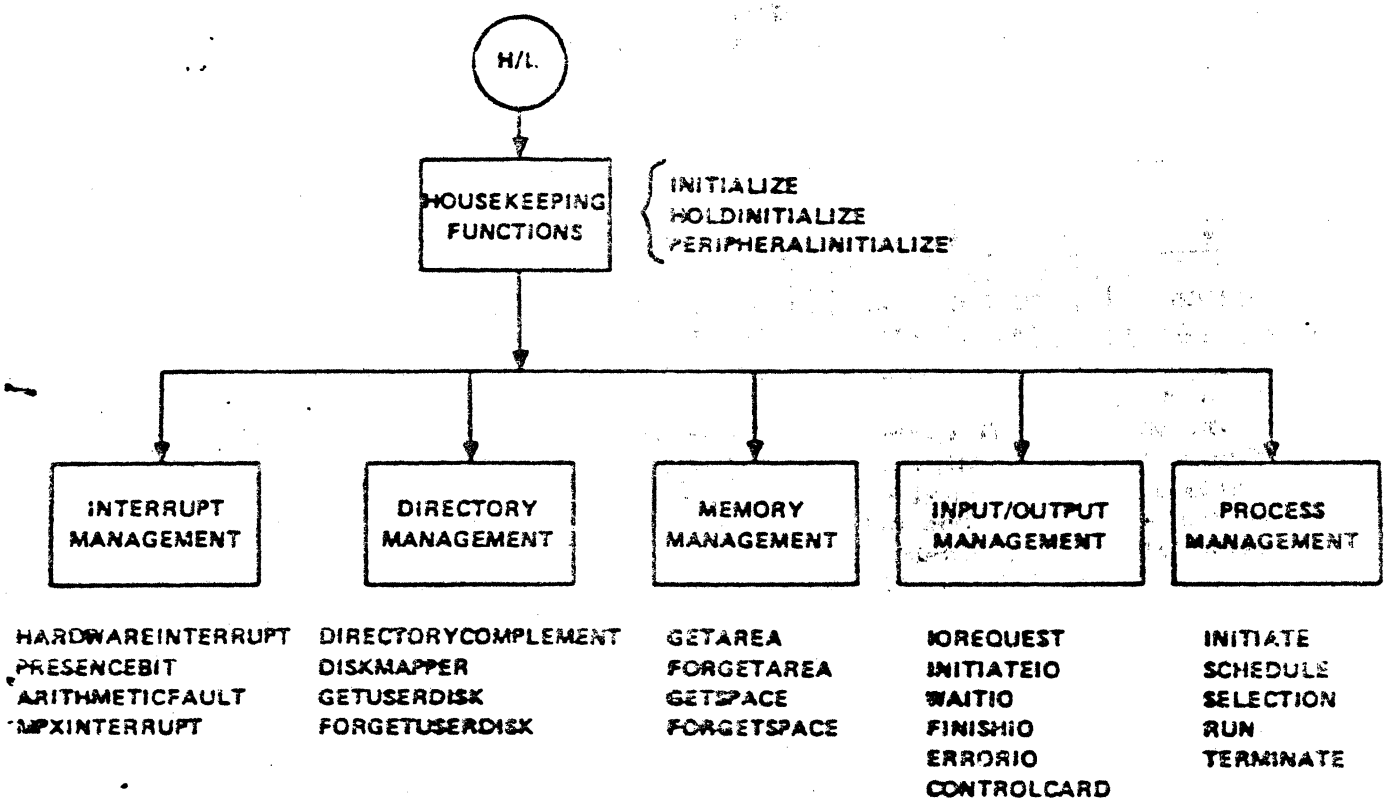
THE FIRST WORD (WORD ZERO) OF A PROCESS STACK SHOULD CONTAINS EITHER

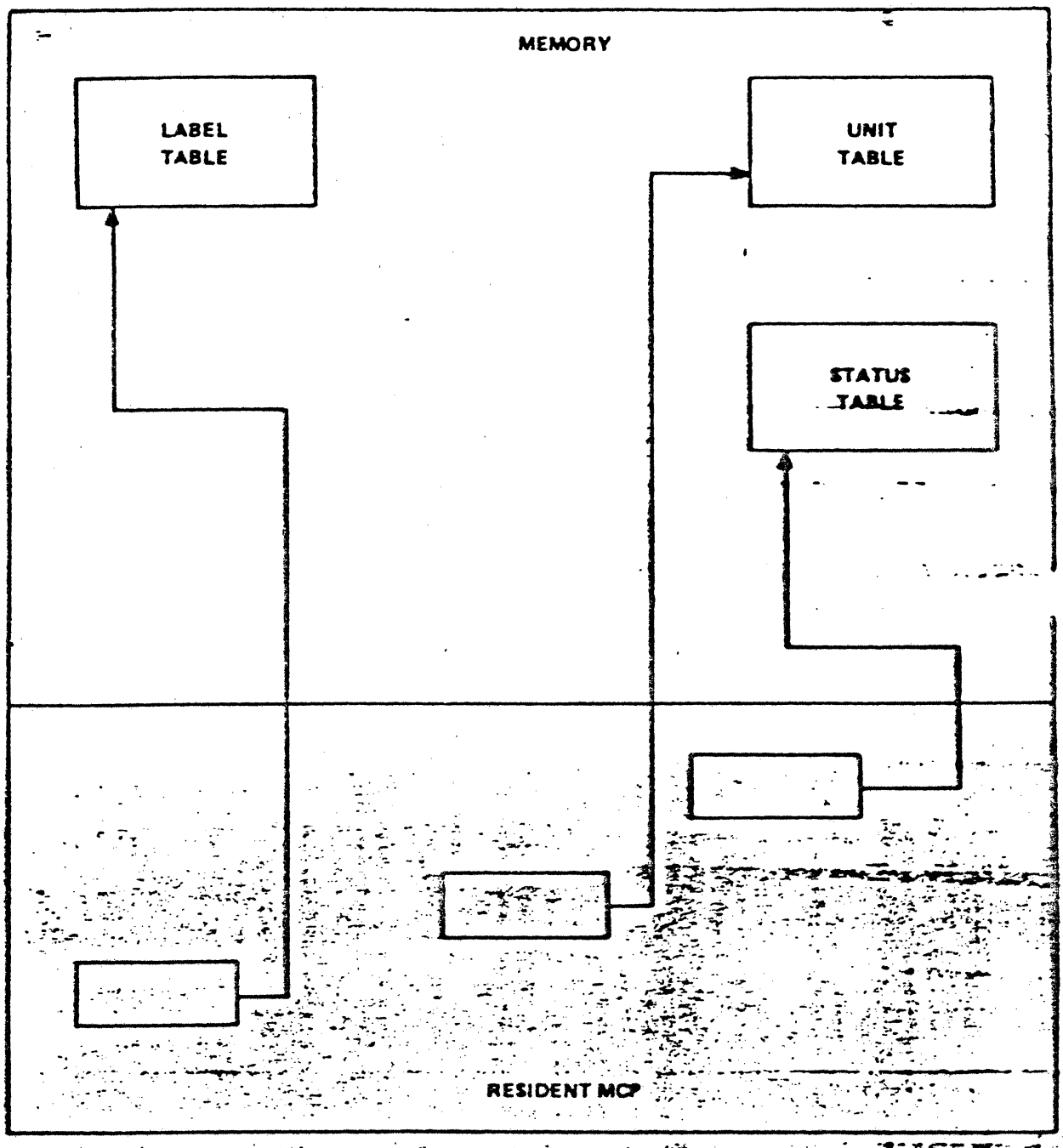
TOSCW NOT ACTIVE OR Processor ID

THE SECOND WORD (WORD ONE) CONTAINS MSCW

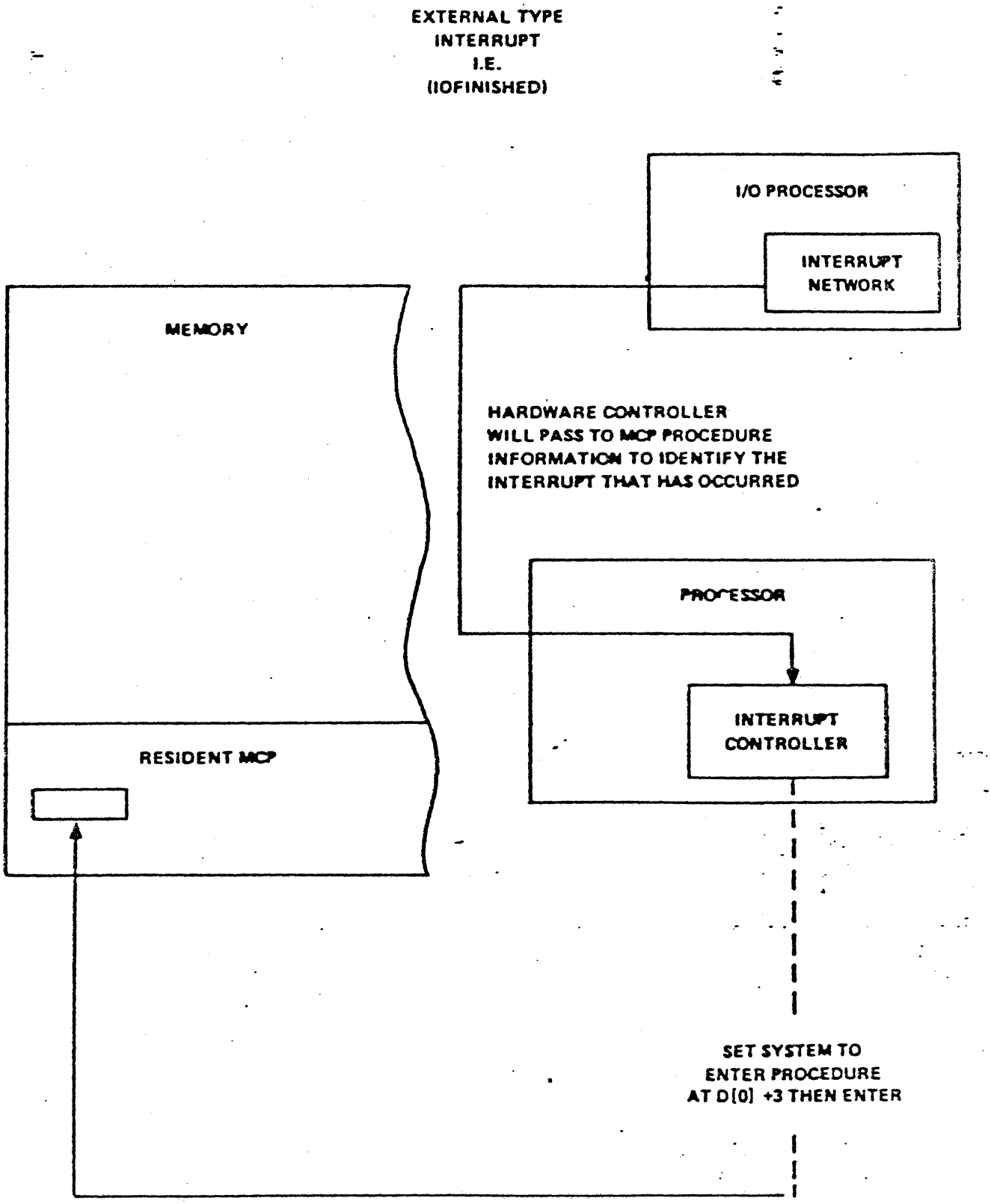
EXPLAIN.

MCP ORGANIZATION

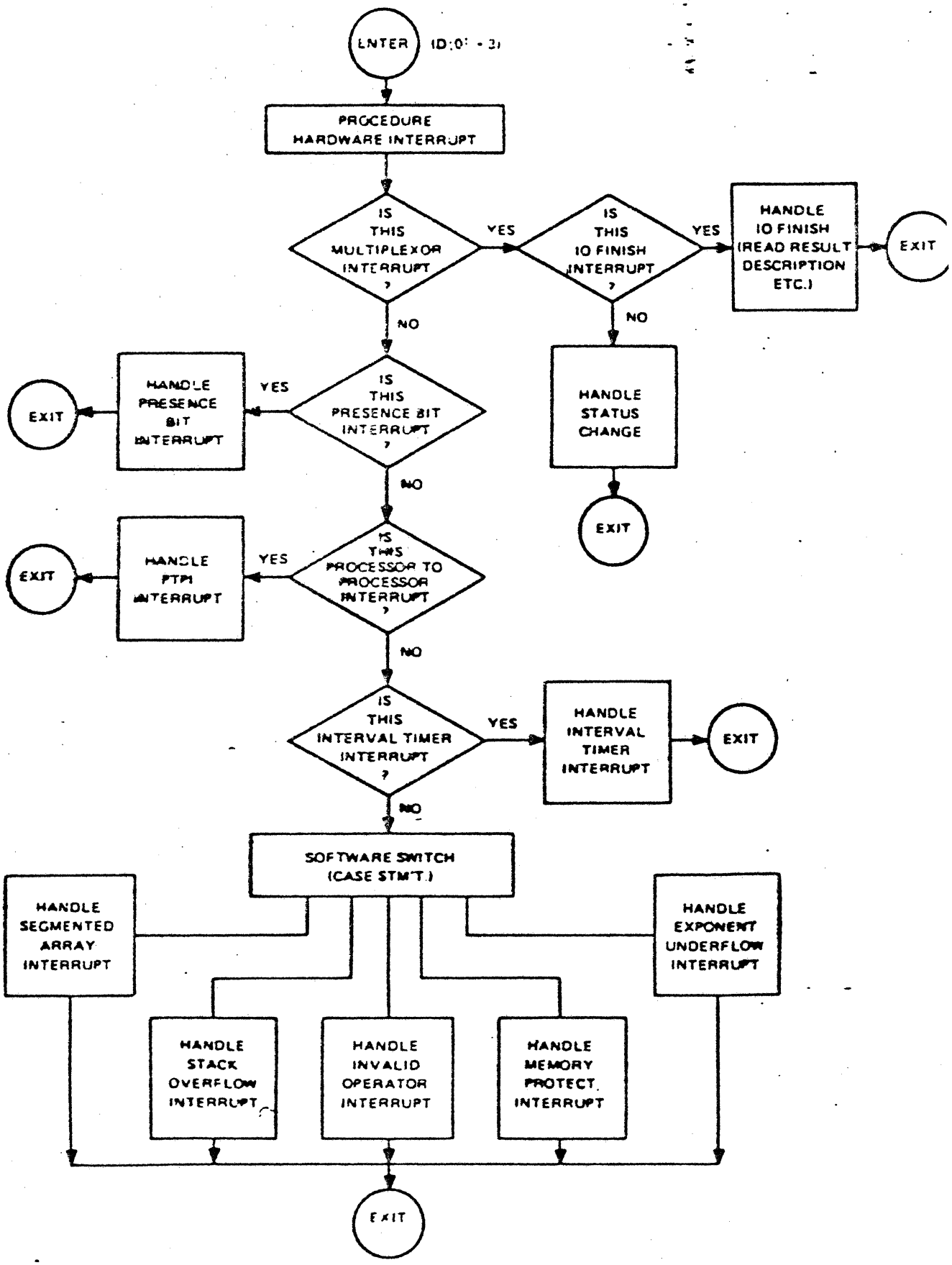




SOME TABLES USED BY MCP FOR RESOURCE ALLOCATION


EXTERNAL TYPE
INTERRUPT
I.E.
(IOFINISHED)
I/O PROCESSOR
INTERRUPT
NETWORK
MEMORY
HARDWARE CONTROLLER
WILL PASS TO MCP PROCEDURE
INFORMATION TO IDENTIFY THE
INTERRUPT THAT HAS OCCURRED
PROCESSOR
INTERRUPT
CONTROLLER
RESIDENT MCP
SET SYSTEM TO
ENTER PROCEDURE
AT D[0] +3 THEN ENTER

HARDWARE INTERRUPT FLOW DIAGRAM

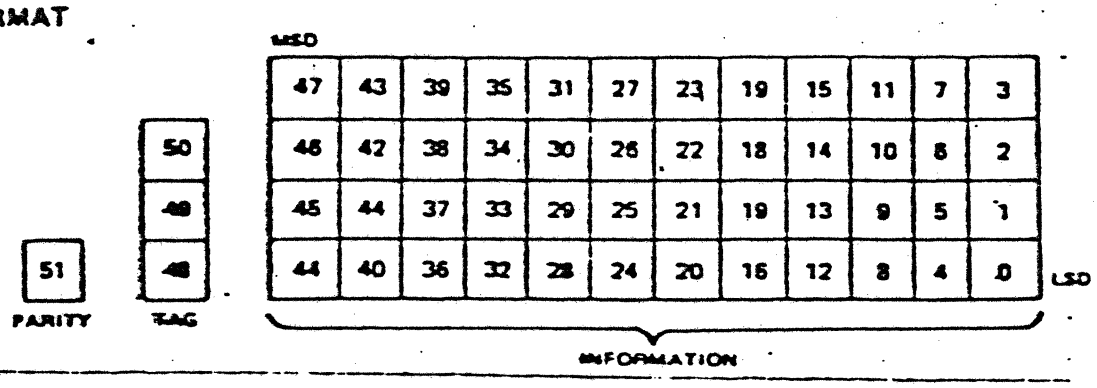


PRIORITY

F Field	What		
47:1	Invisable Independent Runner		
46:2	Special Independent Runner (Autobackup,DCC=3,SWAPPER=2)		
44:1	MCP code owns a soft lock		
42:2	MCS or user special (MCS=2, CP=1)		
39:1	Task being DSed		
38:1	Interactive Swap task		
37:1	WFL JOB		
36:7	Declared Priority		
29:6	Fine Priority		
23:4	Stack Status	Hex: F	Alive
		E	Ready
		D	Ready Swapped
		C	Asleep
		9	Holding
		8	Waiting
		7	Frozen
		6	To be continued
		5	Unemployed
		4	Selected
		3	Scheduled
		2	Dying
		1	Being set up
19:20	Link to next element in the READY Queue (MCP stack ends the chain of "ready words")		

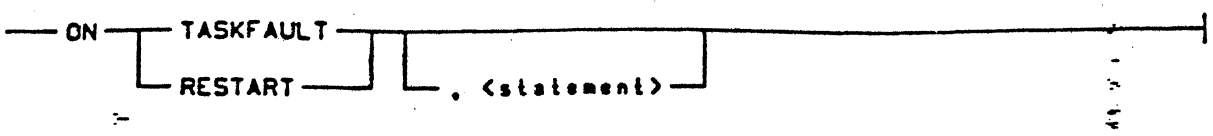
Fine priority::= 63*(MYPROCESSTIME + INTEGER(.075 sec)) DIV
(MYPROCESSTIME + MYIOTIME + INTEGER(.150 sec))

HEXADECIMAL FORMAT



ON Statement

Syntax



Example

```
<1> BEGIN JOB X;
        SUBROUTINE SUB;
        BEGIN
          ON TASKFAULT,
            BEGIN
              DISPLAY "SUB TASKFAULT TAKEN";
              GO ERR;
            END; % end of ON statement
          RUN X;% if X aborts, "SUB TASKFAULT TAKEN" is
                % displayed by the ON TASKFAULT;
          ON TASKFAULT;
          RUN Y;% if Y aborts, "JOB TASKFAULT" is displayed
        END; % end of SUB
        RUN A; % if A aborts, no TASKFAULT is executed
        ON TASKFAULT, DISPLAY "JOB TASKFAULT";
        RUN B; % if B aborts, "JOB TASKFAULT" is displayed
        SUB;
      ERR:
<1>   END JOB
```

Semantics

The ON statement allows the job task to take user specified action when (1) a task is abnormally terminated (ON TASKFAULT), or (2) the system is Halt/Loaded and the job must be restarted (ON RESTART).

Only one statement for each condition can be enabled at one time. Thus, any ON statement disables any previous ON statements for the same condition. If a subroutine executes an ON statement while running in the job, it will disable any previous ON statements until the subroutine is finished or the ON condition is disabled.

The statement ON TASKFAULT; enables the condition TASKFAULT. If any task subsequently is terminated abnormally, or a compilation is terminated for syntax errors, the statement will be executed. This includes operator or programmatic discontinuation as well as arithmetic faults. The statement ON TASKFAULT; disables the condition. An abnormal task termination will not have any effect on the job.

Similarly, the RESTART condition can be enabled and disabled by the statements:

```
ON RESTART, statement;

ON RESTART;
```

Subroutine exit or an ON RESTART; will return the restart condition to what it was before the subroutine was called.

If the enabled statement does not execute a GO statement, the job will resume

System Initialization

There are three distinct types of initialization available on the B 6800. They are the Cold Start, the Cool Start and the Halt/Load. Each has a special function.

The Cold Start is used when the system is first installed. It assumes that nothing is on the system. The Cold Start requires:

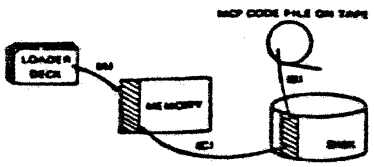
MCP object file on a library tape,
System loader object deck.

By setting the "card load select" button on the console and then pressing the "load" button, the hardware will execute a hardwired series of bootstrap instructions to read a card into memory and then transfer control to the machine coded instructions. This loaded bootstrap program reads in an object deck (about 50 cards produced by the ESPOL compiler) and then transfers control to this program. This program, known as the utility loader (UTILLOADER), reads a parameter card containing a tape name and a file name. UTILLOADER then searches the system to find which units are tape units and reads the tape labels. On finding a matching file name, it then determines if it is a library tape and if so, searches the directory on the tape. Finding the designated file, it loads the program LOADER into memory and transfers control to the program. (UTILLOADER also can load other ESPOL programs used for testing and maintenance.)

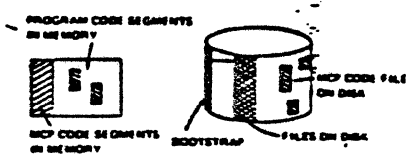
The LOADER program can perform many functions, the most important of which is loading the MCP from disk, tape, or disk pack. Its functions are specified by parameter cards.

After loading the MCP, the LOADER program initializes a disk directory and loads the primary initialized portion of the MCP into memory. Then it transfers control to this new program, i.e., the MCP. The MCP then begins its own initialization.

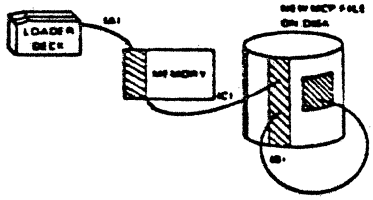
A Cold Start forces loss of any files on the disk family where the MCP is loaded.



The Cool Start operation on the B 6800 reloads the MCP code file on disk but leaves the remaining files intact. Bootstrap information, including the address of the MCP code file, is contained in the lowest address disk segment.



If the MCP code file becomes corrupted or overwritten, the same loader deck can reload the MCP from tape to disk, and if necessary, move the location of the code file on disk. A Cool Start can load a new MCP code file from disk by specifying the name of the file without disturbing the disk directory.



A Cool Start from disk can be done through the MCP by a CM (change MCP) message. In both cases, the MCP performs the initialization routine. No disk files are lost during this procedure; only MCP reinitialization of memory tables occurs. It is often used just to change MCPs.

The Halt/Load is the most common initialization, accomplished by:

a. Pressing the HALT button on the console.
b. Pressing the LOAD button, with the "card load select" button off.

The hardware reads the bootstrap segment at the lowest disk address, fetches the current MCP code file and the MCP reinitializes its tables. This process takes only seconds to perform. All of Main Memory is tested in the process, and any modules failing the test are deleted by the MCP from the configuration.

Recovery Aspects

HALT/LOAD RECOVERY

All tasks are aborted during a Halt/Load condition and memory is reinitialized. Jobs and job queues reside on disk and therefore are still present. By default the job is restarted at the last point where no task was running. Let us examine a job/task structure which is interrupted by a Halt/Load.

During normal operation the system uses the MCP code file on A. Code moves from A into memory and the bootstrap for a Halt/Load points to A. If disk pack A fails, however, the bootstrap is altered to Halt/Load to disk pack B; the system automatically uses the MCP code file on B. Using the same principle, one can duplicate the directory and the access structure on the same families.