TABLE OF CONTENTS

SOFTWARE FOR CUSTOMER ENGINEERS II - OPERATIONS

Intended Audience: Customer Engineers

Duration: 5 Days

Maximum Class Size: 10 Students

Prerequisites: Cray Employee
Knows Cray Architecture
Knows CPU and IOP Instruction Sets
Has worked with Cray Offline Diagnostics (DSS)
6 Months Site Experience

Course Description: An operations level course teaching you the skills to operate a Cray system from the IOP station. This course is centered around 4 nights of dedicated machine time to learn from experience how to install and deadstart a Cray system. Installing, generating and running online diagnostics is also done as an exercise. Kernel and Station commands, Diagnostics, COS and IOS debugger and operational aids and utilities are covered with hardware problems induced in a bug class.

Course Content:

1. IOP Station Operational Commands
2. Installing and Generating Diagnostics
3. COS Online Diagnostics
4. IOS System Diagnostics
5. Operational Aids and Utilities
6. Lab Exercises

Course Objectives:

1. Install, Deadstart and Restart a Cray System using deadstart parameter files.
2. Perform necessary system functions using Kernel, Station and Interactive Station commands.
3. Install and generate Diagnostics, making an FDUMP tape and listings.
4. Access and run COS online and IOS system diagnostics.
5. Use COS and IOS Debug Utilities to read and write memory or CPU registers.
6. Use Cray operational aids and utilities to maintain and troubleshoot online failures.
7. Read APML and IOS macro's in IOS code.

Motivation:

1. To communicate better with customers, operators, and analyst.
2. Improves your understanding of system operation.
3. Enables more efficient response to memory and disk errors.
4. Increased reliability by improving isolation of two time hits.
5. More time for analyst to spend on software problems.
6. Improves availability by reducing offline time used by CE.
7. Future Cray products require stronger software skills.

Software for Customer Engineers II

Monday	Tuesday	Wednesday	Thursday	Friday
Operations IOS Startup Kernel COS Startup Station STARTUP	Diagnostics Installation Generation Listing Tape	Diagnostics COS Batch MENU IOS System DOM MOSTEST HSPTEST	Operational Utilities EXTRACT HERG FDUMP	Exercises
Dedicated Lab Time IOS Tape Startup KERNEL Commands STATION Commands COS Startup Interactive Station	Dedicated Lab Time Parameter Files INSTALL DEADSTART File Utilities	Dedicated Lab Time COS Diagnostics IOS Diagnostics Disk Maintenence	Dedicated Lab Time COS Debug Read Memory Write Memory Breakpoint IOS Debug Read Write Breakpoint	Exercises

COURSE MATERIALS

Software for Customer Engineers II	Workbook
IOS Operators Guide	SG-0051
Operational Procedures	SM-0043
Operational Aids and Utilities	SM-0044 (optional)
COS release tapes and letters	
Diagnostic release tape and letters	
Sample install job outputs	
Listing of GENPL proc's	
APML Assembler Reference	SM-0036
Section 2 and 10 of SM-0046	

READING ASSIGNMENT

Monday Night:

SG-0051	Chapter 1		IOP Station
	Chapter 2		IOS Startup
	Chapter 3	pages 3-1 to 3-2	Kernel Commands
		3-10 to 3-11	Expander Commands
	Chapter 4	pages 4-1 to 4-10	Station Commands
COS Release Letter		skim through	COS Install
Diagnostic Release Letter			Diagnostic Install

Tuesday Night:

SM-0043	Chapter 5	pages 5-27 to 5-49	COS Startup
		pages 5-1 to 5-22	File Directives
SG-0051	Appendix F		File Utilities

Wednesday Night:

Diagnostics in SWCE II Workbook		
SM-0043	Chapter 6	COS Debug
SM-0043	Chapter 7	Dumping the Cray

Thursday Night:

SM-0044	Chapters 4 and 5	EXTRACT FDUMP
SM-0036	Chapters 1 and 4	APML

EVALUATION METHOD

EVALUATION OF YOUR PROGRESS IN GAINING EXPERTISE IN THESE SKILLS IS ACCOMPLISHED BY ASSIGNING A COMPETENCY LEVEL TO EACH SKILL.

Level

0 No knowledge and no experience.

1 Has some knowledge and limited experience with this skill, but not sufficient to contribute in a work environment.

2 Can perform some parts of this skill satisfactorily but requires instruction and supervision to perform the entire skill.

3 Can perform some parts of this skill satisfactorily but requires periodic supervision and/or assistance.

4 Can perform this skill satisfactorily without assistance and/or supervision.

5 Can perform this skill with proficiency in speed and quality without supervision or assistance.

6 Can perform this skill with initiative and adaptability to special situations without supervision or assistance.

7 Can perform this skill and can lead others in performing it.

Successfully completing this course should give you a competency level of three (3) for most skills. Experience on the job will continue to increase your competency level.

Software for Customer Engineers II

Date: ______________________________

Participant's Name:

____

Instructor's Name:

____

Region/Country:

____

Competency Levels

0 No experience and knowledge
1 Needs help with all parts of the task
2 Can do parts of a task requiring the skill
3 Can do the task with periodic assistance
4 Needs no assistance with the task
5 No assistance, fast and accurate
6 No assistance, fast and accurate under pressure
7 No assistance, can lead others

LEARNING LOG

CESW II									
Skills At the end of the course the learner is able to:									
Install, Deadstart, and Restart a Cray System.									
Use Kernel, Station and Interactive Station Commands.									
Generate an offline diagnostics tape.									
Access and run online diagnostics.									
Use IOS and COS Debug utilities.									
Use Cray Operational aids and utilities.									
Read AMPL and IOS macros.									
Levels	0	1	2	3	* 4	5	6	7	No Basis For Judgement

Sessions attended/held _____/_____

Exercises completed/assigned _____/_____

Labs attended/held _____/_____

This learning log is intended as an aid to the learner in establishing goals and plotting progress. It is not intended as an indicator of job performance and therefore should not be used in determining future job actions.

*Maximum level discernible by the instructor in an instructional environment.

INSTRUCTOR'S FEEDBACK

# Sessions attended/held ____/____
# Exercises completed/assigned ____/____
# Labs attended/held ____/____

MET THE PREREQUISITES OF THE COURSE

not at all — yes — was over qualified

Specifics:

SELF APPRAISAL

too high — is correct — too low

3 levels — 2 levels — 1 level — 1 level — 2 levels — 3 levels

Specifics:

WAS ACTIVE AND ATTENTIVE IN CLASS

not at all — to a normal degree — exceptionally so

Specifics:

MADE GOOD USE OF LAB TIME

not at all — to a normal degree — exceptionally so

Specifics:

MADE GOOD USE OF TERMINAL TIME

not at all — to a normal degree — exceptionally so

Specifics:

KEPT UP WITH THE REST OF THE CLASS

fell behind the class — yes — was ahead of the class

Specifics:

SHOWS A POSITIVE ATTITUDE ABOUT WORKING AT CRAY

not at all — to a normal degree — exceptionally so

Specifics:

Comments:

These are subjective appraisals based on the instructors brief and limited observations of the learners behavior during the class.

X

Software for Customer Engineers II

Date: ______________________

Participant's Name:

____

Instructor's Name:

____

Region/Country:

____

LEARNING LOG

CESW II									
Skills At the end of the course the learner is able to:									
Install, Deadstart, and Restart a Cray System.									
Use Kernel, Station and Interactive Station Commands.									
Generate an offline diagnostics tape.									
Access and run online diagnostics.									
Use IOS and COS Debug utilities.									
Use Cray Operational aids and utilities.									
Levels	0	1	2	3	* 4	5	6	7	No Basis For Judgement

Sessions attended/held _____/_____

Exercises completed/assigned _____/_____

Labs attended/held _____/_____

This learning log is intended as an aid to the learner in establishing goals and plotting progress. It is not intended as an indicator of job performance and therefore should not be used in determining future job actions.

*Maximum level discernible by the instructor in an instructional environment.

Software for Customer Engineers II

Date: ____________________

Participant's Name:

____

Instructor's Name:

____

Region/Country:

____

LEARNING LOG

CESW II									
Skills At the end of the course the learner is able to:									
Install, Deadstart, and Restart a Cray System.									
Use Kernel, Station and Interactive Station Commands.									
Generate an offline diagnostics tape.									
Access and run online diagnostics.									
Use IOS and COS Debug utilities.									
Use Cray Operational aids and utilities.									
Levels	0	1	2	3	* 4	5	6	7	No Basis For Judgement

Sessions attended/held ____/____

Exercises completed/assigned ____/____

Labs attended/held ____/____

This learning log is intended as an aid to the learner in establishing goals and plotting progress. It is not intended as an indicator of job performance and therefore should not be used in determining future job actions.

*Maximum level discernible by the instructor in an instructional environment.

IOP Station Operations

1

MODULE OBJECTIVES

With the aid of all furnished reference material, upon completion of this Cray System Operations module, the learner should be capable of:

1. Start IOS from Tape and Disk
2. Enter Kernel, Station, Interactive Station Commands
3. Respond to System Messages at MIOP and Station
4. Start COS and Install COS
5. Operate COS Debug and IOP Debug
6. Edit Startup Parameter Files
7. Dump System to Disk or Tape or Printer
8. Shutdown System
9. Back Drives In and Out

OPERATIONS DOCUMENTATION

System Startup	IOS Operators Guide	SG-51	Section 2
Kernel Commands	IOS Operators Guide	SG-51	Section 3
Station Commands	IOS Operators Guide	SG-51	Section 4
Deadstart Parameter Files	Operational Procedures	SM-43	Section 5
File Editor	IOS Operators Guide	SG-51	Appendix F
IOP Debug	IOS Internals	SM-46	Section 11
COS Debug	Operational Procedures	SM-43	Section 6
System Dumping	Operational Procedures	SM-43	Section 7
Sysdump	IOS Operators Guide IOS Internals	SG-51 SM-46	Appendix B Appendix E
COS Generation	Operational Procedures	SM-43	Section 1-4
Install	Release Letter		
IOS Diagnostics	IOS Internals	SM-46	Appendix D
COS Station Diagnostics	Release Letter		
Symbolic Interactive Debug		SG-56	

DEADSTART TAPE

@MTØ:Ø	$LOAD
@MTØ:1	$DISK
@MTØ:2	$DUMP
@MTØ:3	$DS
@MTØ:4	$OVL
@MTØ:5	$COS
@MTØ:6	RESTART
@MTØ:7	WARMSTART
@MTØ:8	DEADSTART
@MTØ:9	INSTALL

IOS TAPE DEADSTART

Procedure:

1. Mount the IOS deadstart tape on the IOS tape unit.
2. Push master clear and deadstart buttons.
3. Type "3" in response to the tapeload "from MTØ:" message at the MIOP kernel console.
4. If the kernel was assembled with the on-line debugger, type "X" when the ! prompt character appears.
5. When deadstart is complete, a system message will be posted at each kernel console.
6. Enter data and time when prompted to do so.

I/O SUBSYSTEM DEADSTART

MIOP is initially deadstarted from tape through the expander channel.

MIOP initializes the buffer memory configuration and writes a copy of the kernel to buffer memory.

MIOP then deadstarts the other IOPS in the configuration which causes the kernel to be read in from buffer memory.

These IOPs are then initialized by SYSS and BEGIN overlays.

The AMAP overlay is referenced at deadstart by all IOPs for configuration information.

MIOP INITIAL DEADSTART SEQUENCE

1. Operator pushes master clear button.

   This causes exit stack location zero to be set to zero.
   Clears channels' DN and BZ flags.

2. Operator pushes deadstart button.

   This causes first block of tape to be loaded into low memory.
   Interrupt occur when done.
   Hardware begins execution at address in exit stack location zero,
   which is zero

3. Tapeload routine (located in first block) loads rest of kernel from tape.

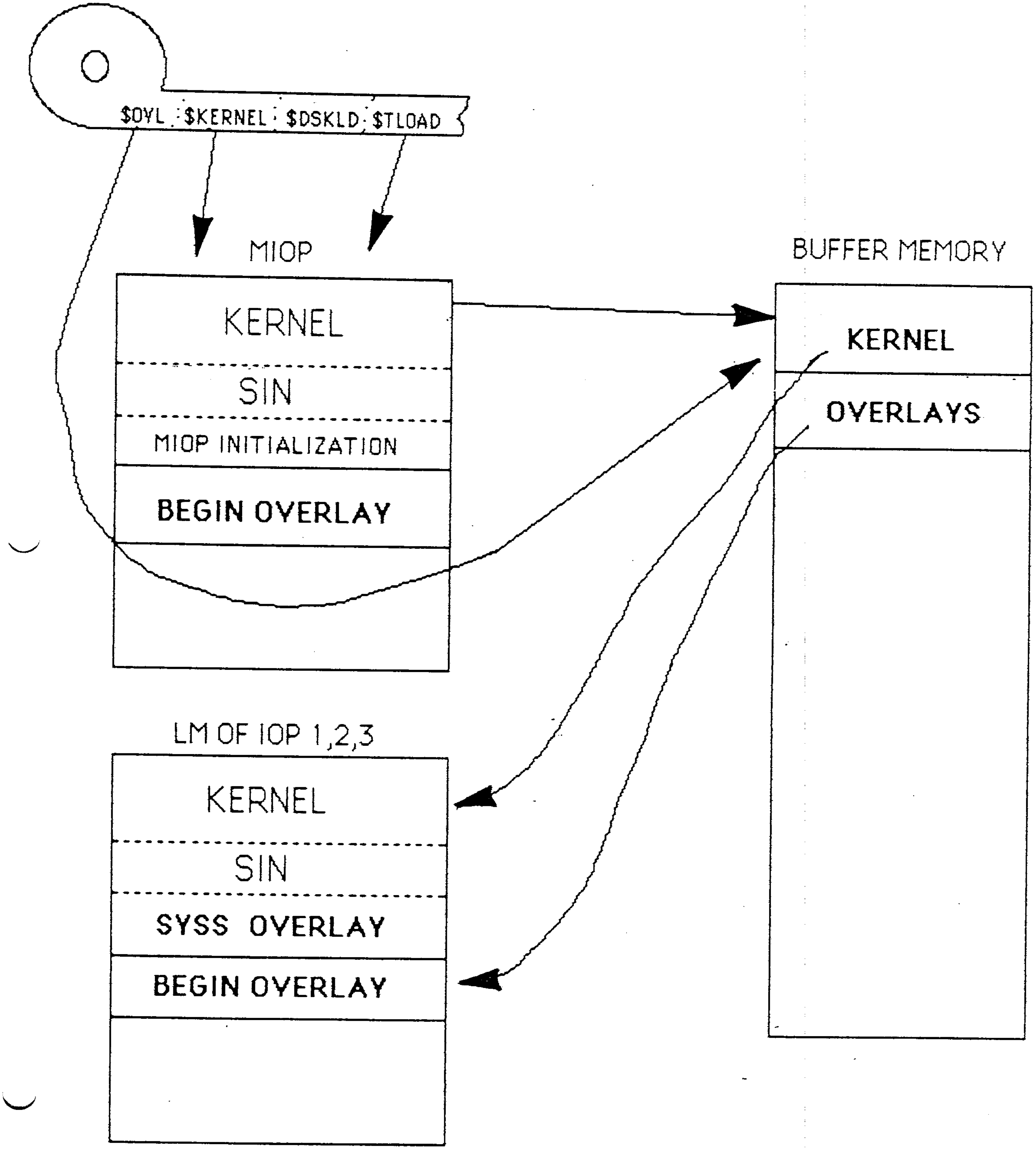
4. Disable interrupts on channels 3 to 47.

5. Perform a local memory and a buffer memory check.

6. Jump to system initialization routine.

7. Call BEGIN.

IOS INITIALIZATION



TYPES OF KERNEL COMMANDS

- Initialization commands
  - CRAY command
  - STATION command
  - MASTER command
  - CONFIG command
  - HELP command
- Concentrator commands
  - Communication with CRI front-end interface
    - CONC command
    - ENDCONC command
  - Communication with an NSC A130 adaptor
    - NSC command
    - NSCEND command
  - Interactive communication with COS
    - IAIOP command
    - IAIOP LOG command
    - IAIOP POLL command
    - IAIOP LOGOFF command
    - IAIOP END command
    - IACON command
- Device commands
- Peripheral Expander tape mount messages
- Peripheral Expander disk mount message
- Miscellaneous maintenance commands
  - LISTP command
  - LISTO command
  - UBTAPE command
  - PRTAPE command
  - ERRDMP command
  - ERROR command
  - TIME command
  - CLOCK command
- Deadstart Parameter File Utilities
  - COPY
  - EDIT
  - DUMP
  - FSTAT
  - DELETE
  - CLEAR
  - LOAD
  - DLOAD
  - DDUMP
  - DSTAT
  - FDUMP
  - FLOAD
  - PROC
  - DEF
- IOS Online Diagnostics
  - F80M
  - HSPTEST
  - MOSTEST
  - XMT
  - MPR
  - MDK
  - CPTEST
  - ECHOCP

IOP STATION DEVICE COMMANDS

Command	Function
ABORT	Terminates input or output
DISABLE	Places the device offline. A program using the device is allowed to perform I/O and terminate normally.
ENABLE	Places the device online
RESTART	Terminates input or output. If the station was performing output staging, the transfer is postponed and the dataset staging operation is reinitiated later.
RESUME	Resumes input or output on the designated device

A device command has the following general format:

```
command device
```

command One of the commands listed in table

device One of the following local device mnemonics:

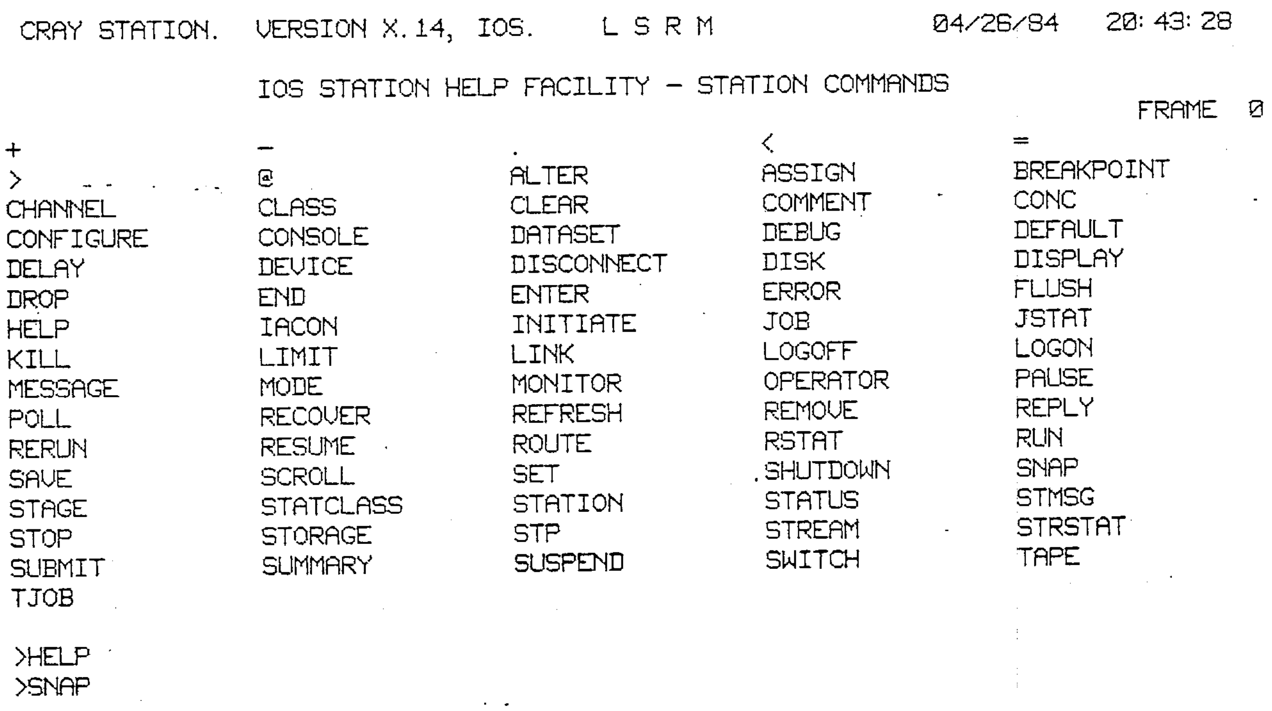
Mnemonic	Device
@MT0	Magnetic tape unit
@PR0	Printer/plotter
@DK0	Disk unit

TYPES OF OPERATOR STATION COMMANDS

Types of Operator Commands, Displays and Functions:

- Activation
- Deactivation
- Link Control
- Station Identification
- Peripherals Controls
- Dataset Staging
- Job Identification
- Job -Scheduling
- Job Execution
- Job Termination
- Job Commencement
- Station Messages
- Logfile Messages
- Display Format
- Link and Station Status
- Peripherals Status
- Job and Dataset Status
- Tape Configuration Display
- Tape Device Configuration
- Error Log Table Display

STATION COMMANDS



```
CRAY STATION.  VERSION X.14, IOS.   L S R M            04/26/84   20:43:28

               IOS STATION HELP FACILITY - STATION COMMANDS
                                                               FRAME  0
+            -            .            <            =
>            @            ALTER        ASSIGN       BREAKPOINT
CHANNEL      CLASS        CLEAR        COMMENT      CONC
CONFIGURE    CONSOLE      DATASET      DEBUG        DEFAULT
DELAY        DEVICE       DISCONNECT   DISK         DISPLAY
DROP         END          ENTER        ERROR        FLUSH
HELP         IACON        INITIATE     JOB          JSTAT
KILL         LIMIT        LINK         LOGOFF       LOGON
MESSAGE      MODE         MONITOR      OPERATOR     PAUSE
POLL         RECOVER      REFRESH      REMOVE       REPLY
RERUN        RESUME       ROUTE        RSTAT        RUN
SAVE         SCROLL       SET          SHUTDOWN     SNAP
STAGE        STATCLASS    STATION      STATUS       STMSG
STOP         STORAGE      STP          STREAM       STRSTAT
SUBMIT       SUMMARY      SUSPEND      SWITCH       TAPE
TJOB

>HELP
>SNAP
```

COS STARTUP

CPU deadstart requires a COS binary file and a parameter file.

Either of these can reside on tape or 80mB disk.

The parameter file may also be input from the console; or an existing one may be edited through the console.

The format of the start command, input at the MIOP kernel console, is:

```
START     COSFILE        PARFILE  ,ED

WHERE COSFILE IS:
     MTØ:n
          n  is tape file number.

PARFILE is:
     MTØ:n
          n  is tape file number.

     @TT - Parameter file is input from console.
```

ED indicates parameter file is to be edited first.

For 80MB Startup

The COS File is   directory name/file name

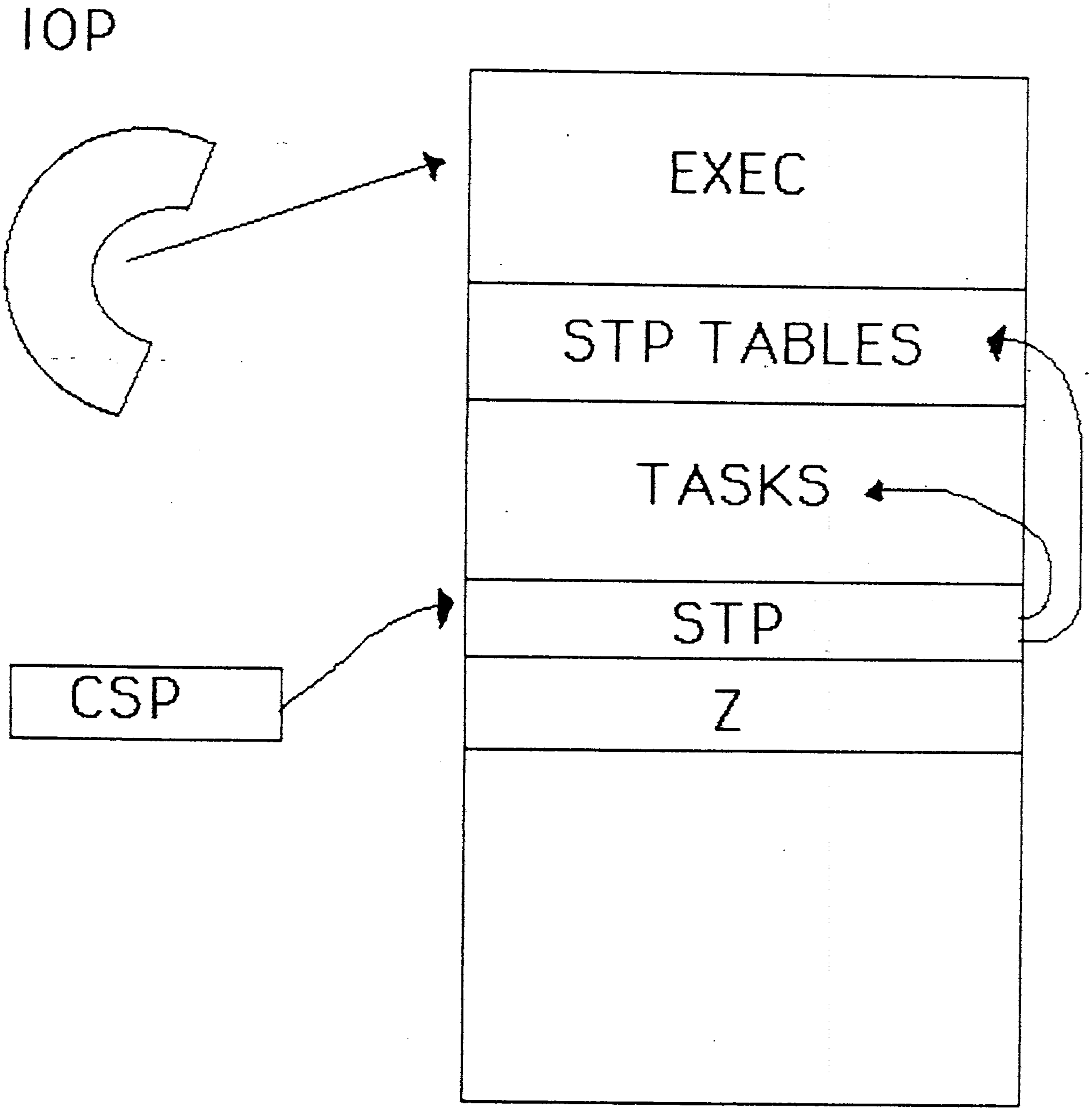
The PAR File is   directory name/file name

DSTAT gives a list of directories available

DEF DIR directory name sets up the default directory

FSTAT fives a list of the files

COS STARTUP



INSTALL

Loads the COS Binaries into CRAY memory.

CRAY mass storage is initialized for the very first time.

- A device label (DVL) is written on each disk unit.
- Space is zeroed and reserved on the master device sufficient to hold CRAY memory size.
- A roll job index dataset is initialized ($ROLL).

System tables are initialized for the very first time.

- Reflect how much useable disk space is available (DRT).
- Creates a disk dataset catalog (DSC) and writes the DSC to the master device.
- Makes entries in the DSC for $ROLL.

DEADSTART

Continues COS following a normal system shutdown.

Deletes DSC entires for input and output datasets (SDT).

Preserves DSC entries for permanent datasets.

Preserves disk space occupied by the system dump.

Copies system dump to another area if used and makes it a permanent dataset.

Rebuilds the system directory from disk if desired.

RESTART

Continues COS following an abnormal system interruption.

Preserves DSC entries for input and output datasets.

Preserves DSC entries for permanent datasets.

Preserves disk space occupied by the system dump.

Copies system dump to another area if used and makes it a permanent dataset.

Preserves rolled jobs and associated datasets if required.

Rebuilds the SDR from disk if desired.

DEADSTART PARAMETER FILE

```
*NOOP - SN27 STARTUP PARAMETER FILE
*NOOP - 02/09/84
*RESTART      CHANGE TO *DEADSTART FOR DEADSTART
*RRJ,1        CHANGE TO *RRJ,0 FOR WARMSTART
*NODUMP
*LOCK,2
*SKIPEFT
*CONFIG,DVN=DD-A1-22,NAVAIL
*CONFIG,DVN=DD-A1-23,NAVAIL
*CONFIG,DVN=DD-A1-26,NAVAIL
*CONFIG,DVN=DD-A2-26,NAVAIL
*CONFIG,DVN=DD-19-23,AVAIL,RDWRT
*CONFIG,DVN=BMR-0-20,AVAIL,RDWRT,RLS=Y,WDL=Y,SCR=Y
*FLAW,DD-A1-27         SN471     SPARE
C40-41                 CE FLAW
C51,T06                06/06/83  R DATA ERROR CH1
C335-337               CE FLAW
C631-632               CE FLAW
C1465-1466             CE FLAW
*ENDFLW
*FLAW,DD-A2-27         SN496     SPARE
C07,T02                05/11/83
C14,T11                06/06/83  R DATA ERROR CH3
C17,T11                06/06/83  R DATA ERROR CH3
C23,T02                05/11/83
C40-41                 CE FLAW
C230,T10               05/11/83
C320,T00               06/06/83  R DATA ERROR CH3
C317,T00               05/11/83
C335-337               CE FLAW
C631-632               CE FLAW
C1465-1466             CE FLAW
*ENDFLW
*FLAW,DD-19-23         SN110     SPARE DD19 with fault circuit
C40-41                 CE FLAW   was DD-19-22 on SN25
C77,T00                08/23/83
C171,T02               08/29/83
C335-337               CE FLAW
C631-632               CE FLAW
*ENDFLW
*END                   Remove this for install or bringing a drive online
*NOOP - BEGIN MASTER FLAW TABLE -
*FLAW,DD-A1-20         SN465     BATCH
C01-41                 CE FLAW + DIAGNOSTICS
C112,T10               06/24/82  RCV DATA ERROR
C120,T03
C122,T00               01/30/83  UNR
C335-337               CE FLAW
```

PARAMETER FILE EDITOR

Provides for creation and modification of parameter text files required for CPU deadstart.

The editor is run from the MIOP kernel console.

Each of the following will invoke the editor:

1. ED option on the start command.
2. Specifying TTI for PARFILE on the START command.
3. EDIT FN.

The editor operates in two modes:

1. Command input mode.
   This mode is recognized by a > in column 1.
2. Text input mode.
   Indicated by a line number in column 1.
   Input is accepted on a line-by-line basis.
   Terminates lines by carriage returns or line feeds.
   The ESC key returns control to command input mode.

There are seven commands available for editing parameter test files.

1. Insert LN

   Insert text following the specified line number.

2. Append

   Append text to the file.
   If file is empty, text will be accepted starting at line 1.

3. Delete LN1 LN2

   Delete lines LN1 to LN2 inclusive.

4. Replace LN1 LN2

   Replace lines LN1 to LN2, inclusive, with text to be input.

5. Type LN1 LN2

   Type lines LN1 to LN2 , inclusive, to the console.

6. Print LN1 LN2

   Print lines LN1 to LN2, inclusive, on the printer.

7. Bye

   Terminate the editor.
   The following message is displayed.

   "SAVE?"

   No - Edited version is discarded. If editor was called from start, edited version will be sent to CPU but not made permanent.

   Yes - "Enter file name:" Message is displayed. Edited version of the file will be saved in the default directory under the specified name.

PARAMETER FILE DIRECTIVES

*INSTALL

*DEADSTART

*RESTART

*OCTAL ADDRESS

*EBP

*DEBUG

*END

*MEMSIZ

*RESTORE

*CONFIG

*LCT

*FLAW

*ENDFLAW

*DEFLAW

*SKIPEFT

*DUMP

*NODUMP

*RRJ

*LOCK

*DSCERR

*DXTERR

*CLEANUP

*SDR

*JCLASS

*SYSTEM

*BOOT

*SUPSYS

*SYSLOG

*DXT

*HOLD

*IPARM

*TSCONC

DEADSTART FILE UTILITIES

Utilities have major changes in 1.14

Binaries are now on the 80MB
Disk is default device on IOP station
File utilities maintain startup binaries on the 80MB disk instead of the master device A1-20

Commands

EDIT FN
Invokes the parameter file editor

COPY FN1 FN2
Copy file FN1 to FN2
The copy is from tape to disk or disk to tape
FN2 cannot already be used
When copying to IOS directory the overlays must immediately follow the kernel file
When copying the other way, allow two consecutive files
Files for tape are labeled @MTO:n:NR

FSTAT @DKØ:dir/
Disply file status of one or more files
If not files

DELETE dir/FN,FN1
Delete the specified files from the specified directory

CLEAR dir:dir
Clear an entire directory

DUMP @MTØ:n dir/FN,FN
Execute a formatted dump of specified files FN to tape file n

LOAD @MTØ:n FN,FN
Load previously dumped tape into original directory
DUMP and LOAD are useful when directories get fragmented

DLOAD @MTØ: @DKØ:dir,dir
Load all files in named directories to expander disk
Tape must be created with DDUMP or FDUMP

DDUMP @DKØ:dir,dir
Will dump all named directores to expander tape

DSTAT @DKØ:dir
Displays the attributes of the name directories
Name
Creation date and time of directory
Size in words

```
FDUMP @DKØ:dir/FN1,FN2,FN3 @MTØ
     Dump all named files in requested directory to expander tape

FLOAD @MTØ: @DK0:dir/FN,FN
     Load all named files in requested directory to expander tape

PROC @DKØ:dir/FN
     Will cause a file of kernel commands created with IOS editor to be
          executed as if entered at a kernel console

RENAME @DKØ:dir/FN,FN,FN
     Renames files in requested directory

DEF
     Displays current default station values
     Independent of defaults for file utilities

DEF DEV
     Displays current station default device @DKØ @MTØ

DEF VOL vol
     Makes vol default VOLume in subsequent staging operations

DEF DIR dir
     Makes dir default Directory in subsequent staging operations
```

COS DEBUGGER

Allows online debugging of COS

Consists of IOP station overlays and executive requests

Allows setting of breakpoints and examination and modification of central memory and the CPU registers

Debugging commands entered at IOP station console

READING MEMORY

Letters A-Z examine memory

DEBUG command shows how each letter is set up to read central memory and the format

DISPLAY command changes letter's set up

+- scrolls left side of screen

⟨⟩scrolls right side of screen

WRITING MEMORY

ASSIGN to Exec, Task or Job (JSQ)

MODE to Exec, Task or Job (JSQ)

address=constant

reg=constant

BREAKPOINTS

EBP in Parameter File will Breakpoint Startup

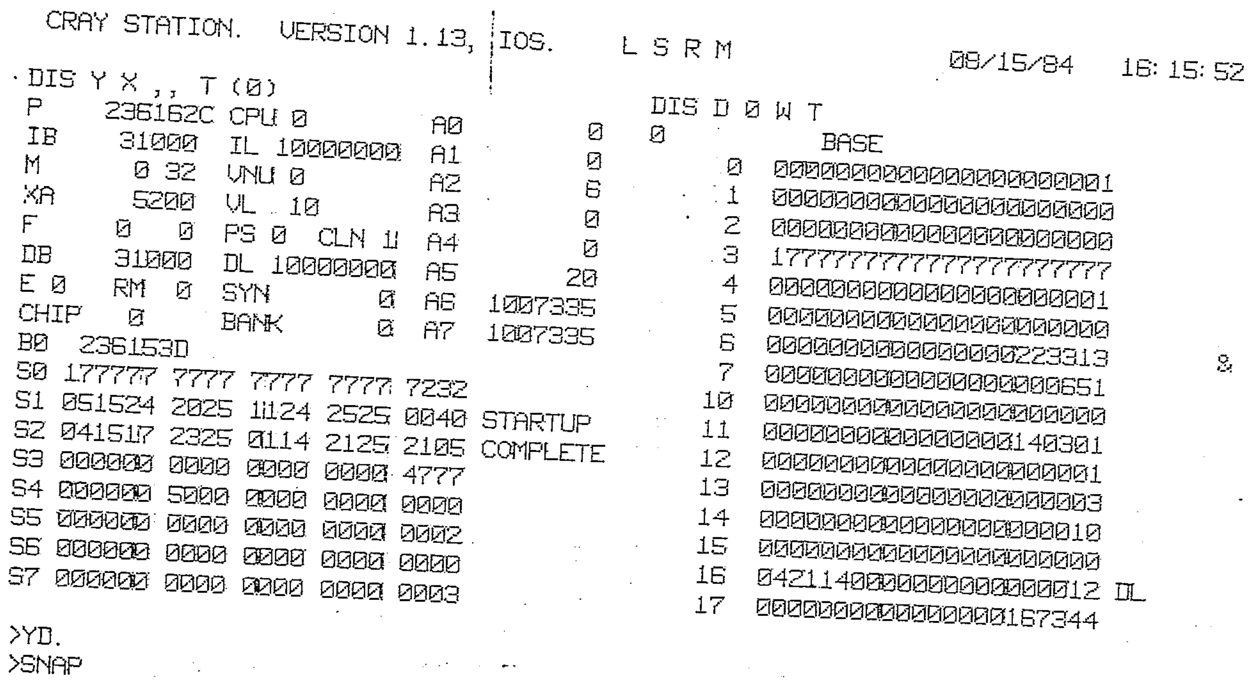
BREAKPOINT command will breakpoint Task or Job

8 breakpoints and double breakpoint

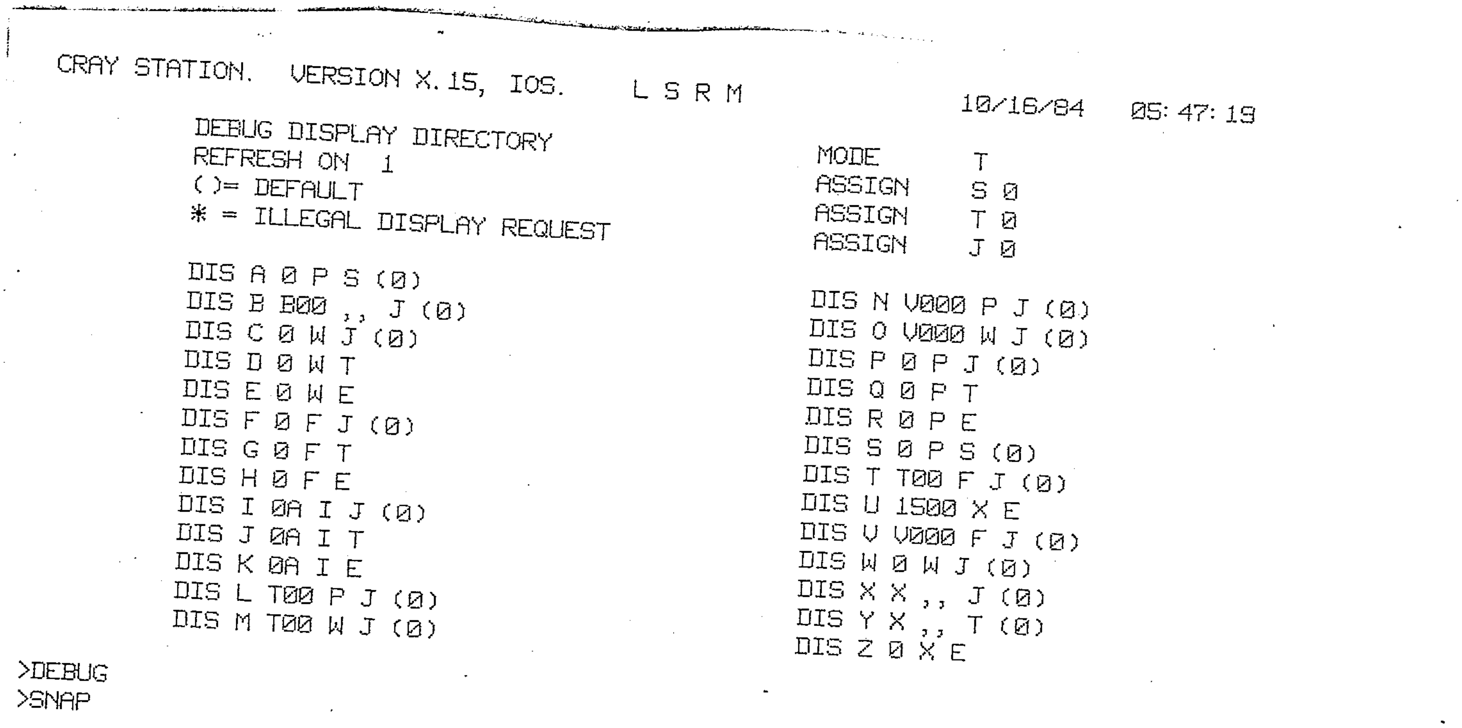
REMOVE removes the breakpoint number

RUN will continue execution to next BP

COS DEBUG



```
 CRAY STATION.  VERSION 1.13, IOS.     L S R M
                                                      08/15/84    16: 15: 52
·DIS Y X ,, T (0)
 P      236162C CPU 0                              DIS D 0 W T
 IB      31000  IL 10000000    A0           0    0         BASE
 M        0 32  VNU 0          A1           0        0   0000000000000000000001
 XA       5200  VL  10         A2           6        1   0000000000000000000000
 F       0    0 PS 0  CLN 11   A3           0        2   0000000000000000000000
 DB      31000  DL 10000000    A4           0        3   1777777777777777777777
 E 0    RM   0  SYN         0  A5          20        4   0000000000000000000001
 CHIP   0       BANK        0  A6     1007335        5   0000000000000000000000
 B0  236153D                   A7     1007335        6   0000000000000000223313      &
 S0 177777 7777 7777 7777 7232                       7   0000000000000000000651
 S1 051524 2025 1124 2525 0040 STARTUP              10   0000000000000000000000
 S2 041517 2325 0114 2125 2105 COMPLETE             11   0000000000000000140301
 S3 000000 0000 0000 0000 4777                      12   0000000000000000000001
 S4 000000 5000 0000 0000 0000                      13   0000000000000000000003
 S5 000000 0000 0000 0000 0002                      14   0000000000000000000010
 S6 000000 0000 0000 0000 0000                      15   0000000000000000000000
 S7 000000 0000 0000 0000 0003                      16   0421140000000000000012 DL
                                                    17   0000000000000000167344
 >YD.
 >SNAP
```



```
 CRAY STATION.  VERSION X.15,  IOS.     L S R M
                                                      10/16/84    05: 47: 19
         DEBUG DISPLAY DIRECTORY
         REFRESH ON  1                                MODE        T
         ()= DEFAULT                                  ASSIGN      S 0
         * = ILLEGAL DISPLAY REQUEST                  ASSIGN      T 0
                                                      ASSIGN      J 0
         DIS A 0 P S (0)
         DIS B B00 ,,  J (0)                          DIS N V000 P J (0)
         DIS C 0 W J (0)                              DIS O V000 W J (0)
         DIS D 0 W T                                  DIS P 0 P J (0)
         DIS E 0 W E                                  DIS Q 0 P T
         DIS F 0 F J (0)                              DIS R 0 P E
         DIS G 0 F T                                  DIS S 0 P S (0)
         DIS H 0 F E                                  DIS T T00 F J (0)
         DIS I 0A I J (0)                             DIS U 1500 X E
         DIS J 0A I T                                 DIS V V000 F J (0)
         DIS K 0A I E                                 DIS W 0 W J (0)
         DIS L T00 P J (0)                            DIS X X ,, J (0)
         DIS M T00 W J (0)                            DIS Y X ,, T (0)
                                                      DIS Z 0 X E
>DEBUG
>SNAP
```

IOS DEBUGGER

Allows on-line debugging of IOS.

Assembled with the kernel and is MIOP resident at initialization.

Subsequent references to the debugger load it from buffer memory into an I/O buffer.

Allows setting of breakpoints and examination and modification of buffer memory and the I/O processor's registers and local memory.

Debugging commands entered at the kernel console.

Must have a kernel console on an IOP in order to debug it with the debugger.

The debugger may be entered several ways:

During system initialization

When a R=XFAR instruction is encountered in non-interruptible code

When an I/O processor halt occurs

When the debug command is entered at the kernel console

Debugger commands allow operator to display and modify the following:

A Register

B Register

C Register

P Register

E Register

Exit Stack

Operand Registers

Local Memory

Buffer Memory

Channel states may also be examined and channel functions issued with the debugger.

Up to 4 active breakpoints may be set in the code.

Double breakpoints may be specified.

DEBUG COMMANDS

A/	accumulator value	new value	
B/	B reg value	new value	
C/	carry bit value	new value	
Chan#I			Channel Status
Chan#I	Function		
EP#E/	stack value	new value	
#R/	register value	new value	
=	toggle between absolute and overlay relative		
LM address	overlay name /	address value	new value
P/	P reg value	new value	
addr	S (Non Int) T (Int)	overlay name	Breakpoints
S or T	Display Breakpoints		
X	Start execution at P		

SYSDUMP

Dumps selected resources to an area of disk pre-selected at install time, or specified during SYSDUMP.

This dump may then be formatted via FDUMP and disposed appropriately.

Restart may occur when the dump is complete.

The following memories and registers may be dumped:

- Central Memory
- Buffer Memory
- IOP Local Memory
- IOP Operand Registers
- IOP A, B, C, E Registers and Exit Stack
- IOP Channels' BZ and DN Flags
- CPU B, T, V and VM Registers

SYSDUMP is entered by typing CNTRL-D at the MIOP kernel console.

SYSDUMP

```
DMEM,TYPE=IOP1,FWA=0,LWA=177777,FORMAT=PARCEL,R.                               FDUMP 1.13           07/10/84     16:25:42
                                                                               SYSDUMP              05/04/84     16:12:25

000001050    030001 054000 060000 024010    020701 103002 070007 010002    0 X   ( ]
000001060    024700 020701 012006 024143    076143 020006 012012 024001    ) ]   (       (
000001070    020010 034001 020006 012014    024001 010000 034001 020006      8     (   8
000001100    012015 024001 010000 034001    020006 012016 024001 010000      (   8     (
000001110    034001 020006 012017 024001    010000 034001 056000 020007    8     (   8 \
000001120    012012 024001 030001 024151    002000 000000 010002 024703      ( 0 (       )
000001130    020010 024704 020151 024705    020007 024706 020006 024707      )   )   )   )
000001140    050000 024710 076702 020010    013055 100003 102002 076064    P )      -     4
000001150    000000 020010 016000 001176    024001 030001 024151 132064            ( 0 (  4
000001160    000000 020151 017000 024414    100003 102002 070002 070006          )
000001170    020107 103002 070003 076064    000000 074151 000000 001276     G     4
000001200    001330 004547 001254 002557    002534 001422 000000 001440             \
000001210    001460 000000 001555 001602    001745 002110 002172 002172     0         H
000001220    002342 024516 024623 002415    002507 002526 002616 002754      )N)    G V
000001230    003017 003043 000000 003051    003173 003272 000000 000000       #   )
000001240    004016 004016 024754 003512    004016 004016 004133 004157        )  J     [
000001250    004461 004315 004370 004377    020006 012011 024001 030001     1          ( 0
000001260    007001 005001 012000 006002    034001 020006 024035 077000            8   (
000001270    014154 014000 177777 024006    075000 012523 060000 024031          (    S  (
000001300    020006 024035 077000 014176    002000 000000 010005 024703      (           )
000001310    010001 024704 030031 024705    020031 012001 024001 030001      ) 0 )     ( 0
000001320    024706 020031 024707 010000    024710 076702 075000 012523    )   )   )      S
000001330    060000 024031 077000 014107    002000 000000 010005 024703      (    G      )
000001340    010002 024704 020035 024705    020031 024706 010000 024707      )   )   )   )
000001350    010000 024710 076702 020035    102002 070010 020007 012006      )
000001360    024001 010101 034001 075000    013414 020035 012012 024001    (  A8         (
000001370    030001 024151 020151 013012    132064 000000 056000 060000    0 (      4  \
000001400    024073 020035 012013 024001    020073 034001 077000 014154    (;    (  ;8
000001410    020151 013011 102002 070005    010014 025035 077000 014406              *
000001420    075000 013414 020006 012013    024001 010000 034001 020006            (   8
000001430    012014 024035 060000 024034    077000 014346 075000 012523      (   (        S
000001440    060000 024031 056000 020006    012013 024001 020031 034001      ( \     (   8
000001450    020006 012014 024035 060000    024034 077000 014346 071157        (   (
000001460    060000 024031 056000 060000    024036 056000 020006 012017      ( \   ( \
000001470    024001 020036 034001 077000    014107 002000 000000 010005    (   8    G
000001500    024703 010012 024704 020036    024705 020035 024706 020035    )   )   )   )
000001510    012017 024001 030001 024707    010000 024710 076702 020035      ( 0 )   )
000001520    106220 020035 012012 024001    030001 024151 020151 013012          ( 0 (
000001530    133064 000000 020035 012013    024001 020036 034001 020035     4      (   8
000001540    012017 024001 030001 024036    020006 012013 024001 020036      ( 0 (     (
000001550    034001 077000 014154 075000    013414 060000 024036 020036    8           (
000001560    012004 024031 077000 014107    020035 102002 070005 020036      (    G
000001570    012002 024031 071272 020006    012013 024001 020035 034001      (       (   8
000001600    075000 013414 060000 024036    041000 013003 100003 102002          ( B
000001610    070005 020036 023107 102002    070003 076064 000000 056000        &G     4  \
000001620    060000 024033 020033 017000    000566 131064 000012 020036      (        4
000001630    016000 005551 024001 030001    102002 070010 020006 012013        ( 0
000001640    024001 010000 034001 075000    013414 014000 005124 024210    (   8        T(
000001650    020210 024144 010010 024143    020143 126000 001664 010000      (   (
000001660    034144 027143 026144 071007    020210 012010 024001 010000    8 . ,       (
000001670    034001 077000 012402 002000    000000 010005 024703 010015    8           )
000001700    024704 020036 024705 020033    024706 020210 012002 024001    )   )   )     (
000001710    030001 024707 020210 012003    024001 030001 024710 076702    0 )     ( 0 )
000001720    020210 012001 024001 010020    034001 020210 012004 024001        (   8     (
000001730    020006 034001 020210 012006    024001 020033 034001 077000      8     (   8
000001740    012344 077000 003417 075000    012523 060000 024036 041000             S  ( B
000001750    013003 100003 102002 070005    020036 023107 102002 070003              &G
```

IOS STARTUP

Under certain conditions, the IOS may be restarted from the 80MB disk.

Prerequisite:

A file 'KERNEL' has previously been saved with the COPY file utility.

Procedure:

1. Type CNTRL-D at the MIOP kernel console. If "SYSDUMP?" appears, go to 5.

2. If no response, make sure there is no tape loaded on the tape drive and push master clear and and deadstart at the power unit.

3. If 2 results in entering the debugger. Type CNTRL-D to exit.

4. Type CNTRL-D again. If "SYSDUMP?" does not appear, a tape deadstart must be performed.

5. Type "Y" or "N" in response to "SYSDUMP?."

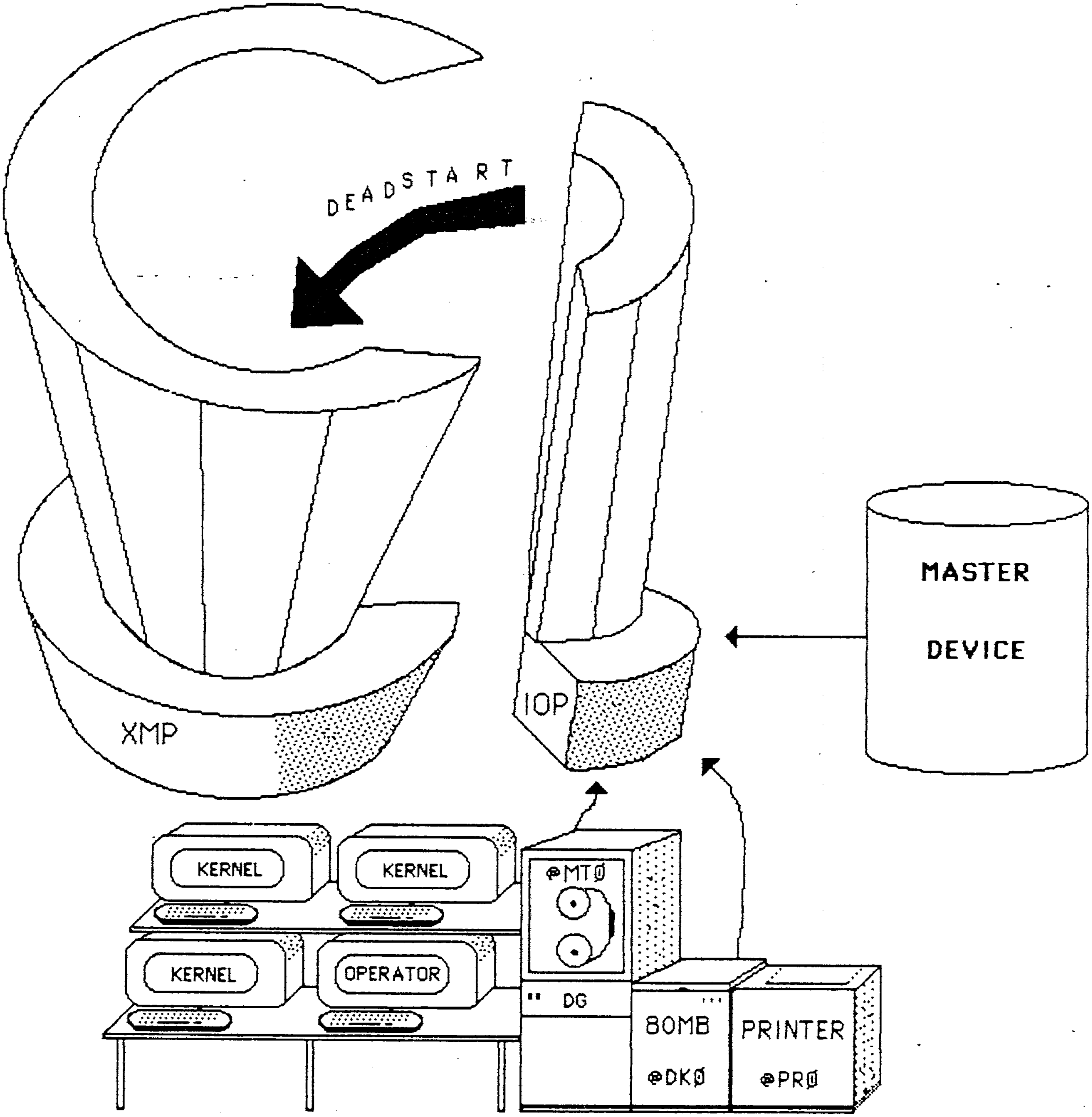
6. When dump complete (or immediately), "RESTART?" will be posted. TYPE "Y".

7. Enter dir/file in response to "ENTER RESTART FILE NAME:" message.

   Example: SN0101/KERNEL

8. If an error occurs, it may be necessary to deadstart from tape.

SYSTEM DEADSTART



DMP

Gives unformatted dump of different parts of the system as an aid in debugging.

Is a stand-alone program deadstarted into MIOP.

Prints out the following registers and memories:

Central Memory - 1500 to 16000 and 30000 to 33000

Buffer Memory -

IOP Local Memories - 1000 to 6000

IOP A, B, C, Operand Registers and Exit Stack

Local and Buffer Memory Trace Buffers

$DUMP

```
                                   IOP-0 LOCAL MEMORY

4400     077000 014642 020006 012003     024001 030001 024021 020006     ~... ...(.0.(. .
4410     012004 024001 030001 024022     077000 011065 014000 004761     ..(.0.(.~..5....
4420     024031 020031 012001 024001     030001 024032 020006 023032     (. ...(.0.(. .&.
4430     102007 020032 024031 020031     107013 076064 000000 020031     .. .(. ...¦4.. .
4440     012001 024001 020032 012001     024002 030002 034001 010024     ..(. ...(.0.8...
4450     024047 020006 024050 077000     014642 014000 005736 024001     (' .((~.......(.
4460     030001 024151 014000 004761     034151 075000 012351 000000     0.(i....8¦z.....
4470     025124 040502 046105 051452     030064 027461 031457 034064     *TABLES*04/13/84
4500     030071 035062 031072 031060     000000 000000 000000 004515     09:22:20.......M
4510     004615 004515 004515 000000     000000 000000 000000 000000     ...M.M..........
4520     000000 000000 000000 000000     000000 000000 000000 000000     ................
PREVIOUS LINE DUPLICATED
4610     000000 000000 000000 000000     000000 000000 000000 004625     ................
4620     004635 004625 004625 000000     000000 000000 000000 000000     ................
4630     000000 000000 000000 000000     000000 004643 004653 004650     ................
4640     004650 000000 000000 000000     000000 000000 000000 000000     ................
4650     000000 000000 000000 062770     000000 000000 000000 000000     ......e.........
4660     000000 000000 000000 000000     000000 000000 000000 000000     ................
4670     000000 000000 000000 000000     000000 000020 000000 011122     ...............R
4700     075400 100075 000025 000000     000000 000000 000000 000000     {..=............
4710     000000 000000 000000 000000     000000 000000 000000 000000     ................
PREVIOUS LINE DUPLICATED
4730     000000 000000 000000 000000     000047 070244 070334 070424     .........'p.p.q.
4740     070514 000000 000000 000000     000000 000000 000000 000000     qL..............
4750     000000 000000 000000 000000     000000 000000 000000 000000     ................
4760     000000 000000 075354 000000     000000 000000 000000 000000     ....z...........
4770     000000 000000 040000 000000     005037 000001 000000 000000     ....@...........
5000     005073 000010 000000 000000     005127 000022 000000 000000     .;.......W......
5010     005163 000002 000000 000000     005217 000132 000000 000000     .s........Z.....
5020     005253 000145 000000 000000     005307 000017 000000 000000     ...e............
5030     005343 000146 000000 000000     005406 000012 000000 000000     ...f............
5040     000000 000000 000000 000000     000000 000000 000000 005057     .............../
5050     040000 000000 000000 000000     000000 000000 000000 000000     @...............
5060     000000 000000 006140 000000     000000 000000 000000 000000     .....`..........
5070     000001 000006 000000 000000     000000 000000 000000 000000     ................
5100     000000 000000 000000 005113     040000 000027 000000 075140     ........K@....z`
5110     000000 000000 000000 000000     000000 000000 006174 000000     .............¦..
5120     000000 000000 000441 000000     000001 000126 000000 000000     .....!.....V....
5130     000000 000000 000000 000000     000000 000000 000000 005147     ...............g
5140     040000 000000 000000 000000     000000 000000 000000 000000     @...............
5150     000000 000000 000000 000000     000000 000000 000000 000000     ................
5160     000001 000000 000000 000000     000000 000000 000000 000000     ................
5170     000000 000000 000000 005203     040000 000000 000000 000000     .......@........
5200     000000 000000 000000 000000     000000 000000 006144 000000     .............d..
5210     000000 000000 000000 000000     000001 000006 000000 000000     ................
5220     000000 000002 000000 000000     000000 000000 000000 005237     ................
5230     040000 000046 000000 000000     000000 000000 000000 000000     @..&............
5240     000000 000000 006704 000000     000000 000000 000460 000000     .............0..
```

OPERATIONS QUIZ

1. What station command displays the jobs COS is handling ?

2. What kernel command starts the IOP station display ?

3. What do you type in and where when you here the console beeping ?

4. What command will change a jobs priority, time limit job class or ID ?

5. What two commands will shutdown the front end station and hyperchannels ?

6. What is the STATC command used for ?

7. Who uses the deadstart parameter file and what does it do ?

8. What are the four COS startup's and what is their key difference ?

9. DEBUG typed in at an IOP station does what ?

10. If *SDR is in the deadstart parameter file what happens and what is neccessary to use the system verbs again .

OPERATIONS QUIZ

1. What station command displays the jobs COS is handling ?

2. What kernel command starts the IOP station display ?

3. What do you type in and where when you here the console beeping ?

4. What command will change a jobs priority, time limit job class or ID ?

5. What two commands will shutdown the front end station and hyperchannels ?

6. What is the STATC command used for ?

7. Who uses the deadstart parameter file and what does it do ?

8. What are the four COS startup's and what is their key difference ?

9. DEBUG typed in at an IOP station does what ?

10. If *SDR is in the deadstart parameter file what happens and what is neccessary to use the system verbs again

Installing and Generating Diagnostics

2

MODULE OBJECTIVES

With the aid of all furnished materials, upon completion of this Offline Diagnostics module, the learner should be able to:

1. Install diagnostic libraries
2. Update the libraries
3. Use GENPL to generate binarys and listing files
4. Use GENPL to generate an FDMP tape and listings
5. Build a DSS system

GENERATION AND INSTALLATION PROCEDURES

A new set of utilities for Diagnostic Generation and installation have been developed to replace BLD and ECD. The new utilities, which reside in a program library called GENPL, were designed to:

1. Minimize the amount of time that is spent by the Software Test and Integration group (STI) in supporting the utilities.

2. To be user friendly by taking into account the limited experience many CE's have in editing and submitting jobs to the Cray.

3. To allow the user to do the generation and installation all at once (create binaries and listings) or to do the generation and installation in stages.

4. To allow the user to generate binaries and/or listings for any number and combination of diagnostics he chooses.

5. To support all Cray sites regardless of hardware configuration and operating system.

6. To reduce the number of tapes needed to support all the site hardware configurations.

The generation and installation utilities are contained in a program library named GENPL. The use of "procedures" (SR-0011 PART 3 4-1) has been used extensively throughout the GENPL. Each diagnostic has its own unique sequence of control statements known as a PROC. PROC defines the beginning of an inline procedure definition block. The prototype statement specifies the name of the procedure and identifies character strings within the procedure that are to be substituted when the procedure is called. COS uses values supplied with the procedure call and default parameter values from the prototype statement to replace these strings.

The procedure definition body is a sequence of COS control statements processed as part of the current statement file when the procedure is called.

It is the use of PROCS that enables the generation and installation process to be simplified. By having a variety of PROCs (that may call PROCs themselves) to choose from, a diagnostic release can be generated and installed with a minimum of effort.

The complete diagnostic generation and installation process can be completed with the submission of a single job to the Cray at sites with I/O Subsystems (extra steps have to be taken at a Data General MCU site to transfer the software from the Data General tape drive to Cray disk). This would include creating the binaries tape (creation of binaries are only relevant to IOP based sites) and printing the listings. Many options are available to the user including being able to choose the ID under which all files are saved on the Cray disk; generating either binaries and listings or both; deleting the binary and/or listings files from the Cray disk after they have been used; being able to do all the above options with either all diagnostics, just one or any number (maximum 20) in between.

INSTALL - Loads the Software from the Release Tape (1S Sites):

The INSTALL control statement does fetches from the expander chassis tape drive to load the released diagnostic software to Cray disk.

```
INSTALL,ID=uid.
```

Parameter:

ID=uid - User identification. 1 through 8 alphanumeric characters to be assigned to the datasets loaded out to the Cray disk. This uid has to be used throughout the generation and installation process. The default is DIAGSYS.

SETUP - Makes the Software Local to the Job:

The SETUP does several accesses to the software needed by the job that has been previously saved to Cray disk by the INSTALL control statement.

```
SETUP,ID=uid.
```

Parameter:

ID=uid - User identification. 1 through 8 alphanumeric characters to be assigned to the datasets loaded out to the Cray disk. This uid has to be used throughout the generation and installation process. The default is DIAGSYS.

GEN - Generates the Diagnostic Binary and Listing Files:

The GEN control statement calls numerous PROCs which assembles the diagnostic binaries and listings and saves them to Cray disk for later use.

```
GEN,ID-uid,I=idn,NOBIN,NOLST,LIST=name,MAC.
```

Parameter:

ID=uid - User identification. 1 through 8 alphanumeric characters to be assigned to the datasets loaded out to the Cray disk. This uid has to be used throughout the generation and installation process. The detault is DIAGSYS.

I-idn - Name of dataset that contains directives to UPDATE. 1 through 8 alphanumeric characters. This parameter is used by DTID only to create bugfix releases. The default is 0.

NOBIN - This parameter specified alone restricts the generation of diagnostic binary files. The default is generate binary files.

NOLST - This parameter specified alone restricts the generation of diagnostic listing files. The default is generate listing files.

LIST=name - Name of labeled LIST pseudo instructions to be processed. This parameter is passed to the CAL or APML control statements within GEN. The default is no LIST pseudo instructions processed. LIST=MONITOR will include the monitor listing with the diagnostic listing.

MAC - This parameter specified alone enables listing of macro expansion. This parameter is passed to the CAL or APML control statements within GEN. The default is no listing of macro extensions.

GENSOME - Generates Random Diagnostic Binaries and Listing Files:

The GENSOME control statement calls numerous PROCs (maximum of 20) which assembles the diagnostic binaries and listings and saves them to Cray disk for later use.

```
GENSOME,ID=uid,I=idn,NOBIN,NOLST,LIST=name
```

```
,MAC,1=diagproc,.....,20=diagproc.
```

Parameter:

ID=uid - User identification. 1 through 8 alphanumeric characters to be assigned to the datasets loaded out to the Cray disk. This uid has to be used throughout the generation and installation process. The detault is DIAGSYS.

I-idn - Name of dataset that contains directives to UPDATE. 1 through 8 alphanumeric characters. This parameter is used by DTID only to create bugfix releases. The default is 0.

NOBIN - This parameter specified alone restricts the generation of diagnostic binary files. The default is generate binary files.

NOLST - This parameter specified alone restricts the generation of diagnostic listing files. The default is generate listing files.

LIST=name - Name of labeled LIST pseudo instructions to be processed. This parameter is passed to the CAL or APML control statements within GENXMP. The default is no LIST pseudo instructoins processed. LIST=MONITOR will include the monitor listing with the diagnostic listing.

MAC - This parameter specified alone enables listing of macro expansion. This parameter is passed to the CAL or APML control statements within GENXMP. The default is no listing of macro extensions.

1=diagproc - These parameters (1 through 20) allow the user to list random diagnostics that he chooses to assemble. The minimum number of diagnostics that can be assembled using GENSOME is 2; the maximum is 20.

TAPE - Writes the FDMP Tape:

```
TAPE,ID=uid,D.
```

Parameter:

ID=uid - User identification. 1 through 8 alphanumeric characters to be assigned to the datasets loaded out to the Cray disk. This uid has to be used throughout the generation and installation process. The default is DIAGSYS.

D - This parameter specified alone will delete the binary files from off the Cray disk once they are disposed to the expander chassis. The default is not to delete the binary files from Cray disk.

LISTING - Prints the Diagnostic Listings:

The LISTING control statement disposes the diagnostic listings to a printer.

```
LISTING,ID-uid,D,F=printer.
```

Parameter:

ID=uid - User identification. 1 through 8 alphanumeric characters to be assigned to the datasets loaded out to the Cray disk. This uid has to be used throughout the generation and installation process. The default is DIAGSYS.

D - This parameter specified alone will delete the binary files from off the Cray disk once they are disposed to the expander chassis. The default is not to delete the binary files from Cray disk.

F=printer - Destination printer for the listings. This can either be EXPANDER or ECLIPSE. If ECLIPSE is used the listing files are disposed to the Data General station with the id of DI. The default is EXPANDER.

DIAGNOSTIC GENERATION AND INSTALLATION FOR A (COS) IOS BASED SYSTEM

This section describes how to load the 2.00 diagnostic software onto the Cray-1S/M or Cray X-MP system that has an I/O subsystem (IOS) as the Maintenance Control Unit (MCU). Also, this section describes the generation and installation steps.

The next steps are performed during batch while COS is running.

1. Mount the release tape on the expander chassis tape drive.

2. Submit the following job to COS:

   The following jobs examples are for an XMP system. Users with S/M systems have to replace, where appropriate, the control statements INSTLXMP, GENXMP and SETUPXMP with the following respective control statements: INSTL1S, GEN1S and SETPUP1S for CRAY-1S systems and INSTALL, GEN and SETUP for CRAY-1M systems.

   ```
   JOB,JN=jn.          Refer to SR-0011
   ACCOUNT.            Refer to SR-0011
   FETCH,DN=$PROC,MF=AP,TEXT=DSD:0,WAIT.
   INSTALL.
   SETUP.
   GEN.
   LISTING.
   ```

   NOTE: Refer to the section on control statements for choice of parameters available to you.

3. Reply to the tape mount request on the MIOP console by typing:

   RESUME @MTO

   This job fetches a file from off the tape which is a library ($PROC) of all the procedures needed to complete the generation and installation of the released diagnostics.

4. Reply to the second tape mount request on the MIOP console by typing:

   RESUME @MTO

   When the tape finishes rewinding remove it and mount a 1200' scratch tape in preparation for writing the binaries to tape.

5. Reply to the third tape mount request on the MIOP console by typing:

   RESUME @MTO

   The tape you are now writing on the expander chassis is the FDMP tape.

   Examine the output to ensure that the job completed without errors.

DIAGNOSTIC GENERATION AND INSTALLATION FOR A (COS) IOS BASED SYSTEM

This section describes how to load the 2.00 diagnostic software onto the Cray-1S/M or Cray X-MP system that has an I/O subsystem (IOS) as the Maintenance Control Unit (MCU). Also, this section describes the generation and installation steps.

The next steps are performed during batch while COS is running.

1. Mount the release tape on the expander chassis tape drive.

2. Submit the following job to COS:

The following jobs examples are for an XMP system. Users with S/M systems have to replace, where appropriate, the control statements INSTLXMP, GENXMP and SETUPXMP with the following respective control statements: INSTL1S, GEN1S and SETUP1S for CRAY-1S systems and INSTALL, GEN and SETUP for CRAY-1M systems.

```
JOB,JN=jn.        Refer to SR-0011
ACCOUNT.          Refer to SR-0011
FETCH,DN=$PROC,MF=AP,TEXT=OSD:0,WAIT.
INSTALL.
SETUP.
GEN.
LISTING.
```

NOTE: Refer to the section on control statements for choice of parameters available to you.

3. Reply to the tape mount request on the MIOP console by typing:

```
RESUME @MTO
```

This job fetches a file from off the tape which is a library ($PROC) of all the procedures needed to complete the generation and installation of the released diagnostics.

4. Reply to the second tape mount request on the MIOP console by typing:

```
RESUME @MTO
```

When the tape finishes rewinding remove it and mount a 1200' scratch tape in preparation for writing the binaries to tape.

5. Reply to the third tape mount request on the MIOP console by typing:

```
RESUME @MTO
```

The tape you are now writing on the expander chassis is the ROMP tape.

Examine the output to ensure that the job completed without errors.

GENSOME is another optional control statement that you can use in place of the GEN control statement. You may also choose to generate a single diagnostic binary and/or listing, in which case you replace the GEN control statement line in the job above with the diagnostics name prefixed by the letter G, for example:

```
JOB,JN=jn.                 Refer to SR-0011

ACCOUNT.                   Refer to SR-0011

ACCESS,DN=$PROC,PDN=PROCLIB,ID=uid.

SETUP.

GARA.

TAPE.

LISTING.
```

The above job will generate the binary and listing for the diagnostic ARA, dispose the binary to the expander chassis tape drive and dispose the listing to the expander chassis printer. The binary tape written can be loaded to DSS with the FLOAD @ command.

GENSOME is another optional control statement that you can use in place of the GEN control statement. You may also choose to generate a single diagnostic binary and/or listing, in which case you replace the GEN control statement line in the job above with the diagnostics name prefixed by the letter G, for example:

```
JOB,JN=jn.                  Refer to SR-0011
ACCOUNT.                    Refer to SR-0011
ACCESS,DN=$PROC,PDN=PROCLIB,ID=uid.
SETUP.
GARA.
TAPE.
LISTING.
```

The above job will generate the binary and listing for the diagnostic ARA, dispose the binary to the expander chassis tape drive and dispose the listing to the expander chassis printer. The binary tape written can be loaded to DSS with the FLOAD @ command.

OFFLINE DIAGNOSTIC GENERATION QUIZ

1. What is the input dataset to BLD ?

2. What determines the diagnostics you want on the DSS tape for ECD ?

3. What language is BLD and ECD written in ?

4. What is the output of BLD ?

5. Name two reasons for using the program ECD ?

6. What language is the GENPL written in ?

7. What would you use to change FLIST ?

8. What would you use to get a listing of GENPL ?

9. What would you use to modify a diagnostic ?

10. How do you install the diagnostic program libraries ?

COS Online Diagnostics

3

MODULE OBJECTIVES

With the aid of all furnished materials, upon completion of this module, the learner should be capable of:

1. Running CPU online diagnostics.
2. Running DDTEST on disk.
3. Running LADDER on tape.
4. Using MENU.
5. Running DSEQ.
6. Running IOP Online Diagnostics.

COS ONLINE DIAGNOSTICS

CPU Diagnostic Jobs

MENU	- Diagnostic Selection
PATH	- Flushes data through channels
SIGMAO	- Vectory Memory Stress
DSEQ	- CPU Diagnostics
DDTEST	- DISK
CMST	- DISK
LADDER	- TAPE

Various CPU Diagnostic Binaries

IOP System Diagnostics

- MOSTEST
- HSPTEST
- CHNTST
- XMT
- XDK
- XPR
- CPTEST
- ECHOCP
- DOM
    - F80M
    - IFP
    - BMOL

MENU

MENU FOR RUNNING ONLINE DIAGNOSTICS

This program is designed to make life easier for anyone who may want to run ONLINE DIAGNOSTICS. It is really self prompting but does assume a couple things.

It is intended to run on the Interactive Concentrator.
(on an IOP console.)
It assumes knowledge of a couple of test setup procedures, specifically the JCL for DDTEST and CMST.

This program is run by ACCESSing it and then entering its name, MENU

```
Example JCL -  ACCESS,DN=MENU,ID=DIAGSYS.
               MENU.
```

This program prompts you for an ACCOUNT card which is valid for your site. At this stage you have to type in the ACCOUNT card parameters.

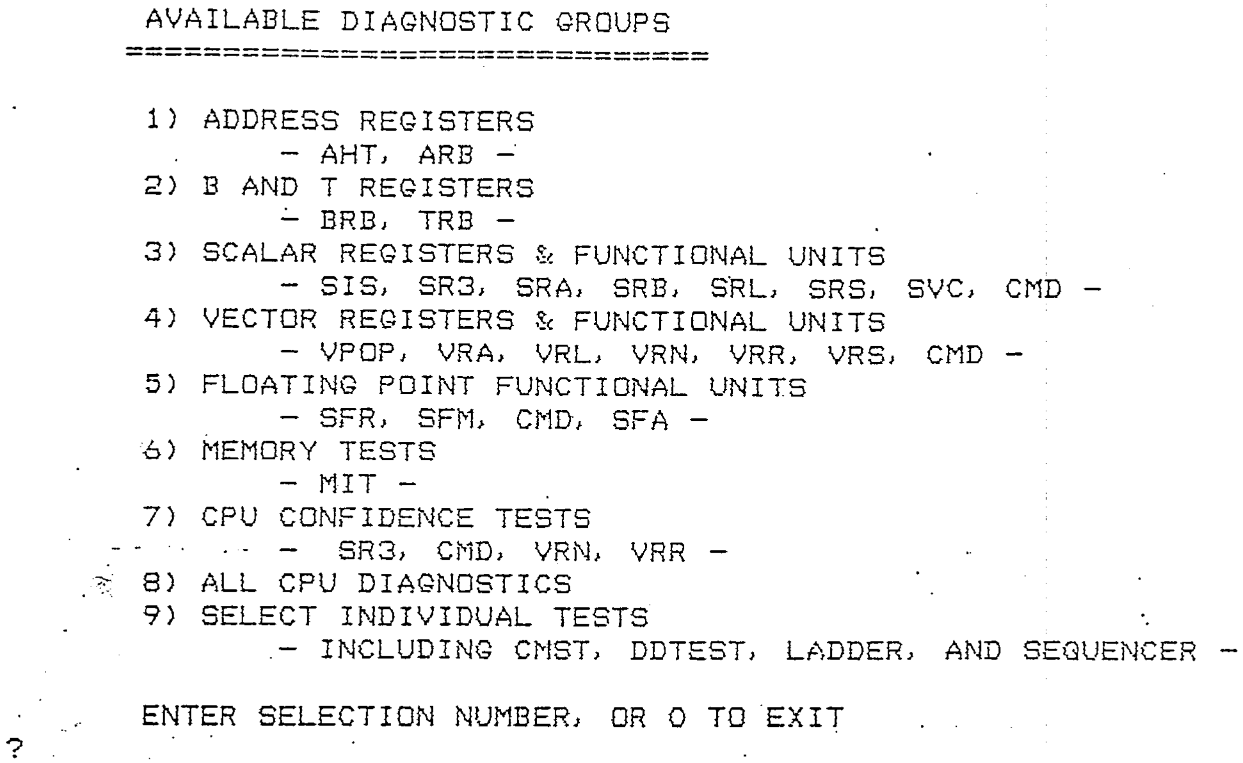
```
Example -      ACCOUNT,AC=nnnnnnn,US=nnnnnn,UPW=nnnnnn.
```

(Each site has its own set of parameters. This is only an example.)

You are then prompted for an ID. This ID is the one assigned to your on-line diagnostics. The MENU needs this info to access the diagnostics for running. DIAGSYS should be used as the ID.

Finally, a number of options are listed to allow for selecting different tests.

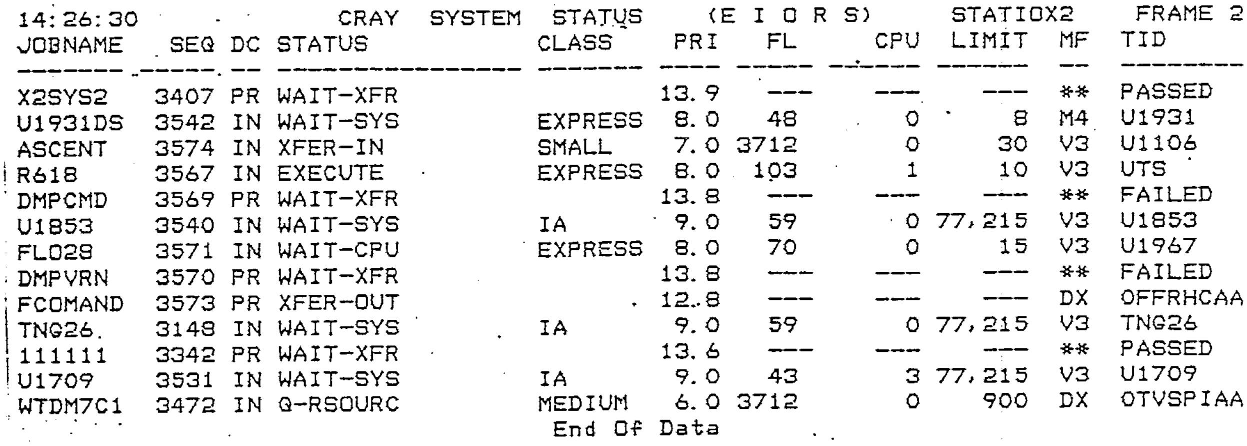
DIAGNOSTICS



```
 AVAILABLE DIAGNOSTIC GROUPS
==============================

 1) ADDRESS REGISTERS
        - AHT, ARB -
 2) B AND T REGISTERS
        - BRB, TRB -
 3) SCALAR REGISTERS & FUNCTIONAL UNITS
        - SIS, SR3, SRA, SRB, SRL, SRS, SVC, CMD -
 4) VECTOR REGISTERS & FUNCTIONAL UNITS
        - VPOP, VRA, VRL, VRN, VRR, VRS, CMD -
 5) FLOATING POINT FUNCTIONAL UNITS
        - SFR, SFM, CMD, SFA -
 6) MEMORY TESTS
        - MIT -
 7) CPU CONFIDENCE TESTS
        -  SR3, CMD, VRN, VRR -
 8) ALL CPU DIAGNOSTICS
 9) SELECT INDIVIDUAL TESTS
        - INCLUDING CMST, DDTEST, LADDER, AND SEQUENCER -

 ENTER SELECTION NUMBER, OR 0 TO EXIT
?
```



```
14:26:30              CRAY   SYSTEM  STATUS    (E I O R S)     STATIOX2    FRAME 2
JOBNAME   SEQ DC STATUS                CLASS     PRI   FL     CPU  LIMIT  MF  TID
------- ----- -- ---------------- ------- ---- ----- ----- ------ -- ---------
X2SYS2   3407 PR WAIT-XFR                       13.9  ---    ---     ---  **  PASSED
U1931DS  3542 IN WAIT-SYS          EXPRESS  8.0    48      0       8  M4  U1931
ASCENT   3574 IN XFER-IN           SMALL    7.0 3712      0      30  V3  U1106
R618     3567 IN EXECUTE           EXPRESS  8.0   103      1      10  V3  UTS
DMPCMD   3569 PR WAIT-XFR                       13.8  ---    ---     ---  **  FAILED
U1853    3540 IN WAIT-SYS          IA       9.0    59      0 77,215  V3  U1853
FL029    3571 IN WAIT-CPU          EXPRESS  8.0    70      0      15  V3  U1967
DMPVRN   3570 PR WAIT-XFR                       13.8  ---    ---     ---  **  FAILED
FCOMAND  3573 PR XFER-OUT                       12.8  ---    ---     ---  DX  OFFRHCAA
TNG26.   3148 IN WAIT-SYS          IA       9.0    59      0 77,215  V3  TNG26
111111   3342 PR WAIT-XFR                       13.6  ---    ---     ---  **  PASSED
U1709    3531 IN WAIT-SYS          IA       9.0    43      3 77,215  V3  U1709
WTDM7C1  3472 IN Q-RSOURC          MEDIUM   6.0 3712      0     900  DX  OTVSPIAA
                                    End Of Data
```

PROGRAM PATH

This program will run a series of diagnostic tests. The flow is as follows:

1. Attempt to move zeros through all the vector registers, then to memory, and then to disk. If any word in the memory image is not zero, the program will display:

   "ERROR ON ZERO VECTOR MOVE. INDEX = n",

   where n is the index of the first nonzero element. The program will then exit.

2. Attempt to move one-bits through all the vector registers, then to memory, and then to disk. If any word in the memory image is not all one-bits, the program will display:

   "ERROR ON ONES VECTOR MOVE. INDEX = n",

   where n is the index of the first element not containing all ones. The program will then exit.

3. Attempt to move a 2/5 pattern through all the vector registers, then to memory, and then to disk. If any word in the memory image does not contain the pattern, the program will display:

   "ERROR ON 2/5 VECTOR MOVE. INDEX = n",

   where n is the index of the first element in the image not containing the pattern. The program will then exit.

4. Attempt to move each of 1000 random numbers through all the vector registers, then to memory, and then to disk. If, for any of these random numbers, any word in the memory image does not contain that number, the program will display:

   "ERROR ON RANDOM VECTOR MOVE. INDEX = n",

   where n is the index of the first element not containing the current random number. The program will then exit.

5. Read the all-zero image from disk to memory. If any word of the memory image is not zero, the program will display:

   "ERROR ON DISK ZERO VECTOR. INDEX = n",

   where n is the index of the first nonzero element. The program will then exit.

6. Read the all-ones image from disk to memory. If any word of the memory image is not all ones, the program will display:

```
"ERROR ON DISK ONES MOVE.  INDEX = n",
```

where n is the index of the first element not containing all ones. The program will then exit.

7. Read the 2/5 image from disk to memory. If any word of the memory image does not contain the pattern, the program will display:

```
"ERROR ON 2/5 DISK MOVE.  INDEX = n",
```

where n is the index of the first element in the image not containing the pattern. The program will then exit.

8. Read each of the random number images from disk to memory. If any word of the image does not contain the current random number, the program will display:

```
"ERROR ON RANDOM DISK MOVE.  INDEX = n",
```

where n is the index of the first element not containing the current random number. The program will then exit.

9. Perform eight logical adds in each element of each vector register, and then store the final result in memory. If any word of the result in memory does not contain the correct value, the program will display:

```
"C  D  I  J  m1  m2  m3  m4"
"ERROR ON VECTOR LOGICAL SUM CALL.  INDEX = m3",
```

where m1 is the actual result and m2 is the correct result of of the m4'th set of logical sums, and m3 is the index of the first element containing an erroneous result. The program will then exit.

10. Perform ten additions in each element of vector registers 2 through 7, and then store the final result in memory. If any word of the result in memory does not contain the correct value, the program will display:

```
"C  D  I  J  m1  m2  m3  m4"
"ERROR ON VECTOR SUM CALL.  INDEX = m3",
```

where m1 is the actual result and m2 is the correct result of the m4'th set of additions, and m3 is the index of the first element containing an erroneous result. The program will then exit.

11. Perform ten multiplications in each element of vector registers 2 through 7, and then store the final result in memory. If any word of the result in memory does not contain the correct value, the program will display:

```
"C  D  I  J  m1  m2  m3  m4"
"ERROR ON VECTOR SUB CALL.  INDEX = m3",
```

where m1 is the actual result and m2 is the correct result of the m4'th set of multiplications and m3 is the index of the first element containing an erroneous result. The program will will then exit.

12. Perform ten multiplications in each element of vector registers 2 through 7, and then store the final result in memory. If any word of the result in memory does not contain the correct value, the program will display:

```
"C  D  I  J  m1  m2  m3  m4"
"ERROR ON VECTOR MULT CALL.  INDEX = m3"
```

where m1 is the actual result and m2 is the correct result of the m4'th set of multiplications and m3 is the index of the first element containing an erroneous result. The program will then exit.

PROGRAM SIGMAO

SIGMAO is a vector/memory stress test. It loads varying vector lengths with floating point values, does summations on the vector contents and compares to see that the results are correct. By varying the vector lengths, memory conflicts of up to sixteen banks are caused.

The following error messages are sent to the log file:

```
   ****** DETECTED HARDWARE ERROR ON SUMMATION ******

PASS COUNT = XXXXX

EXPECTED VALUE = XXXXXX

ACTUAL VALUE = YYYYY

STRIDE A = XXXX  STRIDE B = YYYY  TRIP = ZZZZ
```

PROGRAM DSEQ

DIAGNOSTIC SEQUENCER (DSEQ)

DSEQ is a program built from procedure files to act as a staging and monitoring program for all on-line diagnostics.

The program is built from four procedure files. These are:

DSEQ - This procedure file is the initial invoker to all other procedure files.

EXECUTE - This procedure file is the staging and monitoring program, allowing the diagnostic to pass/fail.

FAIL - This procedure file is the routine used on an error condition. The routine produces an error dump.

SUITE - This procedure file is used to invoke all diagnostics and to allow for setting of pass count/pass count multipliers in each test.

PROGRAM FLOW:

DSEQ is a monitor for use with the on-line diagnostics. The program is comprised of procedure files working in conjunction with each other.
DSEQ is the procedure that first initiates the on-line diagnostics. DSEQ sets up the pass count and pass count multiplier of the number of times the diagnostic is executed. The SUITE procedure is then initialized and is executed. This procedure allows for looping on all diagnostics the number of passes set in DSEQ. SUITE initiates another procedure called EXECUTE. EXECUTE is the monitor itself, allowing the diagnostic to pass the number of passes requested or on an error initiates the error routine. FAIL is the error routine that on an error creates a dump file for purpose of analysis. FAIL will allow the user to advance to the next diagnostic on failure without an abort.

DDTEST

PURPOSE:

TO TEST OUT THE SPARE DD-19/DD-29 DISK DRIVE ON-LINE.

CONTROL STATEMENT:

```
DDTEST,DV=DD-19-30,TEST=SR(:OR:RR),NTKS=4109,
       PERCENT=100,LOOP=9999999,(NOSAVE),(DELETE),
       (TRIAL),MSG=ALL,(NOENG),(NODELAY),(NOACC),
       DN='',PATTYPE=ALL,RANSEED=0,
       DELINT=1:0,DELLEN=1:0,DF=U,
       DT=(DEVICE-TYPE).
```

PARAMETERS

NAME — DESCRIPTION

DV

The 'LDV' of the disk drive(s) to be tested. The device type & unit is appended to the string 'ZZZ' to produce the name of the dataset which will be used to test the disk (e.g. ZZZ1930), provided that the 'DN' parameter is not present.

EXAMPLE: DV=DD-19-32:DD-29-62:DD-19-53
DEFAULT: DV=DD-19-30

TEST

The various tests to be performed.

'SR' - SEQUENTIAL READ.
'OR' - OSCILLATORY READ. (SEE METHOD)
'RR' - RANDOM READ.

EXAMPLE: TEST=SR:OR:RR
DEFAULT: TEST=SR

NTKS

The number of tracks to be tested for each disk. If there are multiple 'DV'S' then each one can have a different 'NTKS' value. If one is not specified for a certain 'DV' then the value previously specified is used.

EXAMPLE: NTKS=:2000
DEFAULT: A COMPLETE DISK. IF THE DISK IS A DD-19 DRIVE THEN THIS CORRESPONDS TO 4109 TRACKS, FOR A DD-29 IT IS 8219 TRACKS, HOWEVER THE NUMBER OF ENGINEER'S TRACKS IS SUBTRACTED FROM THIS VALUE UNLESS 'NOENG' (SEE BELOW) IS SPECIFIED. ONCE A VALUE IS SPECIFIED THEN THAT VALUE IS USED FOR ANY FOLLOWING DEVICES FOR WHICH A VALUE IS .NOT. SPECIFIED.

PERCENT

The percentage of 'NTKS' to be tested. In fact, the actual number of tracks tested is 'NTKS*PERCENT/100'. This allows one to specify the number as a percentage of a disk.
The value given applies to the following devices for which the percentage is not specified.
EXAMPLE: PERCENT=10:50
DEFAULT: PERCENT=100

LOOP

The number of iterations to perform. An iteration consists of writing a single pattern and performing the requested tests.
EXAMPLE: LOOP=11 (WILL RUN ALL 11 DATPAT PATTERNS)
DEFAULT: LOOP=9999999

NOSAVE

The program usually saves the 'ZZZDV' dataset after it has been written, unless it is already permanent. This parameter prevents this.
DEFAULT: PARAMETER IS NOT PRESENT

DELETE

The program usually accesses the 'ZZZDV' dataset prior to overwriting it. This parameter will delete and release the dataset first.
DEFAULT: PARAMETER IS NOT PRESENT

TRIAL

This allows one to test new features to the program easily. It causes the program to simulate both I/O errors and data validity errors.
DEFAULT: PARAMETER IS NOT PRESENT

MSG

This specifies what type of messages, if any are sent to the user & system logs.
'NONE' - ONLY ERROR INFORMATION MESSAGES.
'ALL' - PROGRESS MESSAGES & ERROR INFORMATION.
DEFAULT: MSG=ALL

NOENG

This specifies that the disk drive under test does not have any tracks reserved for the engineers.
This in practice will rarely be the case, there are always 7 cylinders reserved by COS for the engineers.
It would require modifying COS itself to free up the engineering tracks.
This parameter is available 'just in case'.
DEFAULT: PARAMETER IS NOT PRESENT

NODELAY

This prevents the program from going into 'delay' mode. It also causes the program to start executing without waiting for the operator to switch on the disk drive to be tested, i.e. it's effects are as if the operator had first switched sense-switch 2 on, then switched it off immediately, causing the job to come out of 'delay' state.
DEFAULT: PARAMETER IS NOT PRESENT

NOACC

This prevents the program from attempting to access the 'ZZZDV' dataset, it is useful for testing.
It is also useful for running several copies of 'DDTEST' on the same device, otherwise they would only run one at a time, having queued for access to the 'ZZZDV' dataset.
DEFAULT: PARAMETER IS NOT PRESENT

DN

This is the name of the file which is to be used for testing the device. If this parameter is not specified then the name will be generated internally according to the description given for the 'DV' parameter.
EXAMPLE: DN=TESTDS1:TESTDS2:TDS3
DEFAULT: DN='' I.E. NULL

PATTYPE

This is the number of the pattern to be used for testing the disk drive. Valid values for this are:

ALL - This means that all the assembled patterns plus the randomly generated ones will be used.

RANDOM - This means that only the patterns randomly generated will be used for testing.

1-9999 - This means that the pattern normally used on this iteration when 'ALL' is specified will be used. (e.g. 18 means that the 6th pattern MOD(18,12) will be used).

DEFAULT: PATTYPE=ALL

RANSEED

This defines the seed to be used for generating the sequence of random numbers which will be used for the random pattern. It is specified as an integer in the range 0-99999.
DEFAULT: RANSEED=0

DELINT

This is the amount of time that the program will execute before going into 'DELAY' state. This only takes effect if sense-switch 2 is set, The format of this parameter is variable, it can be any of the following:
'SS', 'MM:SS' OR 'HH:MM:SS'.

```
EXAMPLE: DELINT=30          -30 SECS.
         DELINT=1:30        -1MIN 30SECS.
         DELINT=1:1:30      -1HR 1MIN 30SECS.
```

MAXIMUM VALUE= 5:0:0.
DEFAULT: DELINT=1:0

DELLEN

This is the amount of time that the program will go into 'DELAY' state. This is only if sense-switch 2 is set. The format of this parameter is the same as that for parameter 'DELINT' above.
MAXIMUM VALUE= 5:0:0.
DEFAULT: DELLEN=1:0

DF

The data format of the 'ZZZDV' dataset.
DF=U Means that it is or will be an unblocked file.
DF=B Means that it is or will be a blocked file.
If an unblocked format is used, then the last word of every track will contain the track number so that we can check for what would be a 'BLOCK NUMBER ERROR' in a blocked dataset.
DEFAULT: DF=U

DT

The device-type of the device being tested.

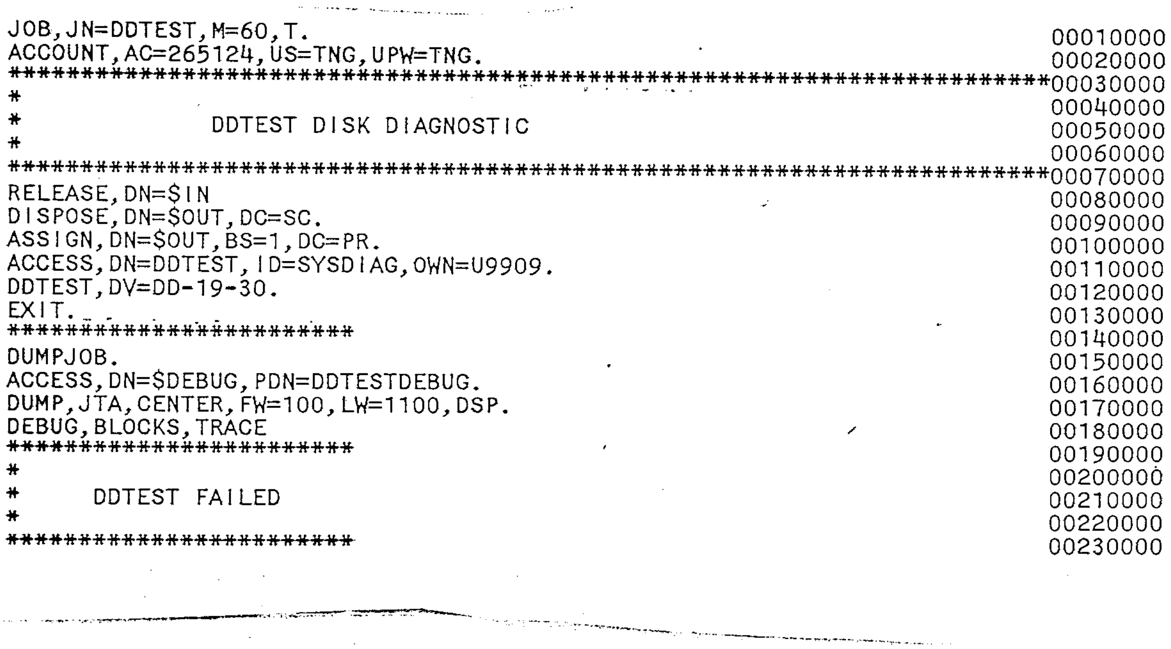
DT=DD19	Means	A DD-19 DISK-DRIVE.
DT=DD29	"	A DD-29 DISK-DRIVE.
DT=SSD	"	A 16MWORD SSD.
DT=SSD8	"	AN 8MWORD SSD.
DT=SSD16	"	A 16MWORD SSD.
DT=SSD32	"	A 32MWORD SSD.
DT=BMR	"	A 1MWORD BMR.
DT=BMR1	"	A 1MWORD BMR.

If no 'DT' parameter is specified then the 'DV' parameter is scrutinized, and if it starts 'DD-19-' or 'DD-29-' then the 'DT' parameter is assumed accordingly.
DEFAULT: DF=UNDEF:UNDEF:.....:UNDEF

METHOD

The program reads the parameters passed to it to decide what to do. In general, it:

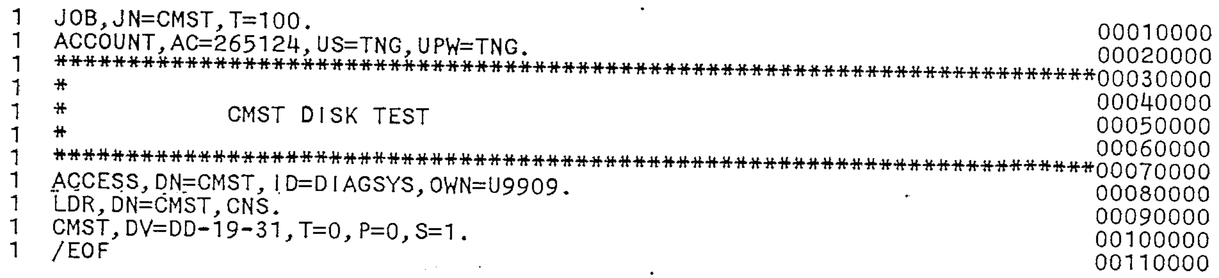
1) Takes the last 4 numeric characters of the 'DV' parameter & appends them to the string 'ZZZ' to produce a dataset name.
2) It accesses a dataset with this name or with the name specified by the 'DN' parameter in 'UQ' mode.
3) If the file exists and the 'DELETE' parameter is present, it deletes and releases the dataset.
4) If the file does not exist or was deleted, it assigns it to the specified device.
5) It then goes into 'WAIT' state, until sense-switch 2 is set. This enables the operators to switch on the disk being tested. (It would normally be switched off). If the 'TRIAL' parameter is specified then the program does not wait.
6) When sense-switch 2 is set by the operator, the program starts properly, it writes/overwrites the 'ZZZDV' file with the next test pattern, 1 track per record.
7) On the 1st iteration the program saves the 'ZZZDV' file, unless the 'NOSAVE' parameter was specified. If the dataset is already permanent it calls 'ADJUST' instead.
8) Depending upon which tests were requested, the program reads the dataset sequentially, in an oscillatory fashion, or in a random fashion. The oscillatory mode reads the 1st track - last track - 2nd track - etc. This maximizes disk head movement.
9) The process is then repeated until the required number of iterations (6 - 8) is exhausted, or until the operator sets sense-switch 1 to cause the program to terminate.
10) If any errors occur, they are reported in the logfile for data-check or 'BLOCK-NUMBER' errors on $OUT.When an error is detected the track on which the error occurred is flagged internally as bad. It is then re-tried. If, at any time, on the retry or on a different test, that same track gives a 2nd error, then the track is internally flawed and will not be used again during this run of the disk utility.
11) The user has the option of running tests on up to 8 disk drives. These tests are not performed in parallel, the iteration of tests are performed 1st on one drive, then on the next, etc.
12) The user has the option of using 'BLOCKED' or 'UNBLOCKED' I/O. The default is 'UNBLOCKED' as this cuts down on memory (no I/O buffers are required), also the use of 'SETPOS' is asynchronous, and since no 'READ-AHEAD' is performed, a bad status from 'IF(UNIT' indicates that the error is in the track being read, not the next one which will (possibly) be read.
    However, since there is no equivalent of a 'BLOCK NUMBER' check with an unblocked dataset, this is done internally by setting the last word of the track (word 22000B) to the value of the track number. This is only done for 'DF=U' and on a buffer-in this word is tested against the track number and any discrepancy causes a 'BLOCK-NUMBER' error to be reported.



```
JOB,JN=DDTEST,M=60,T.
ACCOUNT,AC=265124,US=TNG,UPW=TNG.
************************************************************************
*
*             DDTEST DISK DIAGNOSTIC
*
************************************************************************
RELEASE,DN=$IN
DISPOSE,DN=$OUT,DC=SC.
ASSIGN,DN=$OUT,BS=1,DC=PR.
ACCESS,DN=DDTEST,ID=SYSDIAG,OWN=U9909.
DDTEST,DV=DD-19-30.
EXIT.
************************
DUMPJOB.
ACCESS,DN=$DEBUG,PDN=DDTESTDEBUG.
DUMP,JTA,CENTER,FW=100,LW=1100,DSP.
DEBUG,BLOCKS,TRACE
************************
*
*     DDTEST FAILED
*
************************
```

CMST is an online disk test which runs on the COS operating system. It executes as a normal user job and requests its disk space from the operating system. A job consisting of COS control statements must be keypunched, or composed under a text editor, then submit via the front end computer (figure 5-2). Parameters on the CMST control statement can be set as follows:

DV	device - model - logical unit number
T	test section 0 = run all sections 1 = write sequential 2 = read sequential 3 = random read 4 = random write
P	test pattern 0 = all zeros 1 = all ones 2 = checkerboard 3 = word index and block index 4 = complement word index and block index
S	size of disk space in 512 word blocks



```
1   JOB,JN=CMST,T=100.
1   ACCOUNT,AC=265124,US=TNG,UPW=TNG.
1   ************************************************************************
1   *
1   *            CMST DISK TEST
1   *
1   ************************************************************************
1   ACCESS,DN=CMST,ID=DIAGSYS,OWN=U9909.
1   LDR,DN=CMST,CNS.
1   CMST,DV=DD-19-31,T=0,P=0,S=1.
1   /EOF
```

LADDER

LADDER is an online tape test which runs under control of the COS operating system. It executes as a normal user job and requests a tape drive from the operating system. A job consisting of COS control statements must be keypunched, or composed under a text editor and then submitted via the front end computer. A parameter on the ladder control statment gives the dataset name to be used by the program. Other parameters are read in by the program from $IN and must follow the /EOF control statement. Each parameter is on a separate record and starts in column one, and is right justified.

Parameter 1) Starting tape record length (5 digits).

Parameter 2) Ending tape record length (5 digits).

Parameter 3) Record length increment ('+' increasing, '-' decreasing).

Parameter 4) Read/write switch: 0=write/read 1=read 2=write.

Parameter 5) Pass count parameter: Default is 1.

```
JOB,JN=LADDER,T=100,M,*6250=1
ACCOUNT,AC=265124,US=TNG,UPW=TNG.
***********************************************************************
*
*                     ONLINE TAPE TEST
*
*
**********************************************************************
ACCESS,DN=LADDER,ID=SYSDIAG,OWN=U9909.
ACCESS,DN=TAPE,DT=*6250,VOL=LADSC1,NEW,DF=IC,MBS=32000.
LDR,DN=LADDER,CNS.
LADDER.
EXIT.
DUMPJOB.
DUMP,DSP,FL=405000,LW=420000.
/EOF
   1
3500
 100
   0
  50
   2
/EOF
```

ONLINE DIAGNOSTICS QUIZ

1. What is the program MENU used for ?

2. What language is MENU written in ?

3. If a diagnostic fails what are two options for action ?

4. What JCL statement is in the diagnostic $CS for memory dump ?

5.-How could you change the addresses and length of diagnostic ?

6. What does LADDER test ?

7. Where do you look to find if a diagnostic failed and which ones ?

IOS System Diagnostics

4

MODULE OBJECTIVES

With the aid of all furnished reference materials, following completion of this IOS Diagnostics Module, the learner should be capable of:

1. Starting and stoping the following diagnostics:

    - MOSTEST
    - HSPTEST
    - CHNTST
    - XDK
    - XMT
    - XPR
    - DOM

SYSTEM TESTS

System tests are released as part of the system software. The test name typed in at the MIOP kernel console followed by a carriage return initiates the diagnostic. All system tests run as overlays under control of the kernel.

When the test name is typed on the kernel console, the kernel checks the overlay table for the test name. If the test exists, the kernel places the diagnostic into execution.

The STOP command terminates MOSTEST, HSPTEST, and CHNTST. The engineer enters the STOP command in the following format at the Kernel console of each IOP in which the diagnostic is active.

```
     STOP test
```

test is the name of the online diagnostic test.

The ABORT command terminates XDK, XMT, and XPR. The engineer enters the ABORT command in the following formats at the IOPO Kernel console.

```
     ABORT DKØ    (Terminates XDK)
     ABORT MTØ    (Terminates XMT)
     ABORT PRØ    (Terminates XPR)
```

The test being halted displays an abort message.

MOSTEST

The MOSTEST diagnostic generates a high level of Buffer Memory I/O on all configured I/O Processors. MOSTEST allocates up to 256 512-word buffers. MOSTEST writes to and reads from each buffer using block sizes of from 1 to 512 words with varying data patterns.

The station error display lists any errors discovered. Enter the ERROR command at the station console to obtain the error display. The error display gives an address, data expected, and data received. MOSTEST displays a PASS COMPLETE message on the Kernel console or each IOP involved when all of the allocated Buffer Memory has been tested.

****

CAUTION

MOSTEST cannot be run concurrently with other software.

****

To load the diagnostic overlay, enter the MOSTEST command, followed by a carriage return, at the IOP0 Kernel console as follows:

```
MOSTEST
```

Because the test takes control of the I/O Subsystem while running, the diagnostic displays the following message on the screen:

```
ARE YOU SURE YOU WANT TO RUN THIS TEST?
```

MOSTEST begins if the response is YES.

The diagnostic runs until the STOP MOSTEST command is entered at the console of each I/O Processor in which MOSTEST is running.

HSPTEST

The HSPTEST diagnostic creates a high level of activity on all high-speed channels configured and tests Central Memory in the mainframe. HSPTEST writes to and reads from each 512-word block of Central Memory via the high-speed channel. The block sizes vary from 1 to 512 words, and varying data patterns are used.

The station error display lists any errors encountered. Enter the ERROR command at the station console to obtain the error display. The error display gives an address, data expected, and data received. HSPTEST displays a PASS COMPLETE message each time all of Central Memory has been tested.

****

CAUTION

HSPTEST cannot be run concurrently with other software.

****

To load the diagnostic overlay, enter the HSPTEST command, followed by a carriage return at the IOPO Kernel console as follows:

```
HSPTEST
```

The diagnostic then displays the following message on the screen:

```
THIS TEST WRITES OVER CPU MEMORY.
DO YOU REALLY WANT TO RUN IT?
```

HSPTEST begins if the response is YES.

The diagnostic runs until the STOP HSPTEST command is entered at the console of each IOP in which HSPTEST is running.

CHNTST

The CHNTST diagnostic is a very basic channel loop-back test. CHNTST verifies reliable data transfer on the CRAY asynchronous (low-speed) channel.

Before running CHNTST, connect together the input and output cables of the channel pair being tested by using a one foot cable assembly (part number 2203505) specially made for this purpose.

The station error display lists any errors encountered. Enter the ERROR command at the station console to obtain the display. The error display indicates an input or output channel error, status, data expected, data received, and an input or output channel time out.

****

CAUTION

CHNTST cannot be run concurrently with other software.

****

To load the diagnostic overlay, enter the CHNTST command, followed by a carriage return at the IOPO Kernel console as follows:

```
CHNTST
```

The diagnostic runs until an error is encountered or the STOP CHNTST is entered at the IOPO Kernel console.

XDK

The XDK diagnostic tests the expander chassis 80 MB disk drive. XDK establishes a 4000_8 parcel buffer of data and then writes the buffer to the entire disk, eight sectors at a time. The diagnostic then reads the disk into a second buffer and compares the two buffer for errors.

To load the diagnostic overlay, enter the XDK command, followed by a carriage return, at the IOPO Kernel console as follows:

```
XDK
```

The XDK diagnostic then displays the following message on the screen:

```
THIS TEST WRITES OVER THE ENTIRE DISK - CONTINUE?
TYPE ANY KEY TO CONTINUE
```

XDK begins when any key is typed.

The IOPO Kernel console displays the error messages. The error display gives the data expected and data received. XDK displays a PASS COMPLETE each time the disk is tested.

XDK runs until an error is encountered or the ABORT DKO command is entered at the IOPO Kernel console.

XDK

The XMT diagnostic tests the expander chassis tape drive. XMT writes several multi-block files to the tape drives, reads the files back, and compares the data for errors.

To load the diagnostic overlay, enter the XMT command, followed by a carriage return, at the IOPO Kernel console as follows:

```
XMT
```

The XMT diagnostic then displays the following message on the screen:

```
TYPE ANY KEY TO CONTINUE
```

XMT begins when any key is typed.

The IOPO Kernel console displays the error messages. The error display gives the data expected and the data received.

XMT runs until an error is encountered or the ABORT MTO command is entered at the IOPO Kernel console.

XPR

The XPR diagnostic exercises the expander chassis printer. The printer output consists of alternating pages of characters and plots. The operator must visually inspect the printer output to determine if an error occurred.

To load the diagnostic overlay, enter the XPR command, followed by a carriage return, at the IOPO Kernel console as follows:

```
XPR
```

XPR begins executing when the diagnostic is loaded.

XPR runs until the ABORT PRO command is entered at the IOPO kernel console.

DIAGNOSTIC ONLINE MONITOR (DOM)

DOM is a diagnostic online monitor which runs as an overlay under control of kernel.

DOM can be loaded into each I/O processor by typing in the name on the console attached to that IOP.

Example: DOM

When DOM has been loaded the following message is displayed:

```
DIAGNOSTIC ONLINE MONITOR ACTIVE
LOAD WHAT?
```

To load a diagnostic overlay, enter the diagnostic name.

Example: BMOL

DOM verifies that there is a diagnostic overlay of that name. It also verifies that the diagnostic can be run in the IOP where the name was entered. If there is no overlay of that name of if the overlay does exist but cannot be run in that particular IOP, DOM displays the following message:

```
*INVALID OVERLAY NAME*
```

DOM then starts over and again asks what to load.

If the diagnostic overlay does exist, and can be run in that IOP, DOM attempts to read in a parameter table. If a parameter table exists, DOM checks to see if a text display of unique keywords exist in the diagnostic. If the keywords exist, DOM reads in and displays this text as follows:

```
OPTIONS ARE:

ICHN = INPUT CHANNEL
OCHN = OUTPUT CHANNEL
ETC.
  .
  .
ETC.
```

DOM will then ask for parameters as follows:

```
ENTER PARAMETERS?
```

Parameters may be entered one at a time, or several on the same line separated by commas. Each entry is terminated by pressing the RETURN key.

A parameter table of standard keywords is contained in DOM. They are as follows:

```
ICHN = INPUT CHANNEL - FOR INTERFACES
OCHN = OUTPUT CHANNEL - FOR INTERFACES
ITYPE = INTERFACE TYPE
CMODE = INTERFACE MODE (MASTER, SLAVE, OR LOOP BACK)
CHAN = CHANNEL NUMBER - FOR PERIPHERALS
DEV = DEVICE ADDRESS - FOR PERIPHERALS
SECT = SECTION SELECTS - TO CHANGE DEFAULT SECTIONS SELECTED
```

Values for the above are entered in octal.

Stop conditions

```
SE - STOP ON ERROR
SSC - STOP AT END OF SUBCONDITION
SC - STOP AT END OF CONDITION
SSS - STOP AT END OF SUBSECTION
SS - STOP AT END OF SECTION
ST-- STOP AT END OF TEST.
```

The above stop conditions are turned on or off as follows:

```
ON=ST
  or
ON=ST,SS

OFF=ST
  or
OFF = ST,SS
```

Repeat conditions

```
CONT - CONTINUE FLAG
SCOP - SCOPE LOOP FLAG
LE - LOOP ON ERROR
RSC - REPEAT SUBCONDITION
RC - REPEAT CONDITION
RSS - REPEAT SUBSECTION
RS - REPEAT SECTION
RT - REPEAT TEST.
```

The above repeat conditions can be turned on or off as follows:

```
ON=LE
  or
ON=LE,CONT

OFF=LE
  or
OFF=LE,CONT
```

A parameter table of standard keywords is contained in DOM. They are as follows:

ICHN = INPUT CHANNEL - FOR INTERFACES
OCHN = OUTPUT CHANNEL - FOR INTERFACES
ITYPE = INTERFACE TYPE
CMODE = INTERFACE MODE (MASTER, SLAVE, OR LOOP BACK)
CHAN = CHANNEL NUMBER - FOR PERIPHERALS
DEV = DEVICE ADDRESS - FOR PERIPHERALS
SECT = SECTION SELECTS - TO CHANGE DEFAULT SECTIONS SELECTED

Values for the above are entered in octal.

Stop conditions

SE - STOP ON ERROR
SSC - STOP AT END OF SUBCONDITION
SC - STOP AT END OF CONDITION
SSS - STOP AT END OF SUBSECTION
SS - STOP AT END OF SECTION
ST - STOP AT END OF TEST.

The above stop conditions are turned on or off as follows:

ON=ST
or
ON=ST,SS

OFF=ST
or
OFF = ST,SS

Repeat conditions

CONT - CONTINUE FLAG
SCOP - SCOPE LOOP FLAG
LE - LOOP ON ERROR
RSC - REPEAT SUBCONDITION
RC - REPEAT CONDITION
RSS - REPEAT SUBSECTION
RS - REPEAT SECTION
RT - REPEAT TEST.

The above repeat conditions can be turned on or off as follows:

ON=LE
or
ON=LE,CONT

OFF=LE
or
OFF=LE,CONT

Further parameters unique to a particular diagnostic may be in each diagnostic. DOM provides a text display of these keywords and their description.

DOM will search for 'ON=' or 'OFF=' parameters only in its own table. Any other entry will cause a search of a parameter table in the diagnostic, then DOM's table. If a keyword cannot be found, the following message will be displayed:

```
INVALID PARAMETER
ENTER PARAMETERS?
```

When all parameters have been entered, to start execution of the diagnostic, type in GO .

To stop execution of the diagnostic, type in STOP DOM .

Each diagnostic has the option of terminating by use of a return call, or of calling the parameter handler, so that more parameters may be entered.

To restart a diagnostic with different parameters after a stop on error type in RESET and then enter new parameters.

To terminate after a stop on error, type in TERM .

Further parameters unique to a particular diagnostic may be in each diagnostic. DOM provides a text display of these keywords and their description.

DOM will search for 'ON=' or 'OFF=' parameters only in its own table. Any other entry will cause a search of a parameter table in the diagnostic, then DOM's table. If a keyword cannot be found, the following message will be displayed:

```
INVALID PARAMETER
ENTER PARAMETERS?
```

When all parameters have been entered, to start execution of the diagnostic, type in GO .

To stop execution of the diagnostic, type in STOP DOM .

Each diagnostic has the option of terminating by use of a return call, or of calling the parameter handler, so that more parameters may be entered.

To restart a diagnostic with different parameters after a stop on error type in RESET and then enter new parameters.

To terminate after a stop on error, type in TERM .

ONLINE BLOCK MUX/STC TAPE TEST (BMOL)

BMOL is loaded by the diagnostic monitor program (DOM) and its options are displayed on the console as follows:

```
    CHAN = BLOCK MUX CHANNEL TO BE TESTED.
    DEV = LOGICAL DEVICE ADDRESS (DEVICE ORDINAL) OR AN STC TAPE DRIVE.
```

BMOL consists of five test sections. Default is section 1 only.

SECTION 1

This section loads and reads back all accessible block mux registers using the following patterns:

```
    PATTERN 1  = 000000
    PATTERN 2 = 177777
    PATTERN 3 = 125252
    PATTERN 4 = 052525
```

If an error occurs, the pattern expected and the pattern received are displayed on the console along with the name of the failing register.

SECTION 2

Section 2 uses CPW lists which are executed by the block mux software. It first writes, then rewinds, then reads forward, and finally reads backward.

SECTION 3

Section 3 runs with interrupts locked out. It first writes using command chaining. Then writes two tapemarks and rewinds. It then reads what was just written using read commands without command chaining and verifies the data.

SECTION 4

Section 3 must have been run before running section 4. Section 3 first tests request in processing by causing busy conditions using forward space file, and backward space file commands. It then issues a command to an illegal divide address (HEX FF) and verifies that the control unit rejects it. It then rewinds the tape and reads to the end of file (two tape marks). And verifies the block number and word number in the first block read and the second block read are correct.

SECTION 5

Section 5 is a ladder test. It starts out writing 1 byte records and increments by 1 until a size of 256 is reached. It then increments by 64 until the maximum size (octal 1000000) is reached. It then writes two tapemarks and rewinds. The data just written is then read and verified.

All errors are displayed on the console in English. Note that the sense bytes displayed are in hexadecimal to facilitate communication with system test and check out (STCO) personel when problems occur.

IFP - INTERFACE TEST

IFP is loaded by and runs under the control of DOM. IFP checks the reliability of the IOP PF/PF type channels and their associated interfaces. The test runs in any one of five modes.

1. PF/PG loop-back
2. IA loop-back
3. Loop-back through interface
4. IOP master, front end slave
5. Front end master, IOP slave

In the first four modes, the diagnostic generates random data of a random number of parcels, outputs that data, and expects the same data in return. In the last mode, the IOP echoes data received from the front end master.

PARAMETERS

Set locations ICHAN and OCHAN (locations 210 and 211) to the desired input and output channels respectively.

Select interface type by setting location ITYPE (location 212) as follows:

212 = 0	PF/PG or IA loop-back, IBM, Honeywell, SEL, or TBM.
1	6600 interface
2	7600 interface

Select control mode by setting location CMODE (location 213) as follows:

213 = 0	Loop-back
1	CRAY master
2	CRAY slave

Set the test mode (location 26) to select the following mode:

26 = 0	Run continuously keeping pass and error count
1	Stop on error. Normal setting
2	Repeat the failing pattern and update the display
4	Repeat the failing pattern in a scope loop (don't update the display)

Set bit 2^{15} of location 26 to a 1 to force any of the above modes. By setting the desired parcel count and output buffer, this will allow you to send preset a transfer. The choices are:

26 = 100001	Does one transfer, updates the monitor and stops.
100002	Repeats a transfer, while update display.
100004	Repeats a transfer. Does not update display.

ERROR INFORMATION

On an error, the display gives the type of error that occurred, as listed below.

- OUTPUT CHANNEL CLEAR ERROR
- INPUT CHN CLEAR ERROR
- INPUT CHANNEL ERROR
- OUT CHN RES NOT ACTIVE
- OUTPUT CHANNEL TIME-OUT
- OUT CHN SEQUENCE ERROR
- OUT CHN ADDRESS ERROR
- INPUT CHANNEL TIME-OUT
- IN CHN PARITY/SEQ ERROR
- IN CHN ADDRESS ERROR
- DATA COMPARE ERROR
- SLAVE/SYNC START ERROR

Location CADE (216) is the channel address on error.

Location CEFE (217) are the channel error flags on error.

Location PCNTE (220) is the parcel count of failing transfer.

Location of WFE (221) indicates the address at which the diagnostic is waiting for a front end reply. This is only set when the IOP is unconditionally waiting for a front end response. Set during slave sync start-up and CHE start-up.

Location AAAA (215) is the contents of the accumulator on error.

Location 25 has the return address from error.

F80M

F80M is a formatter and diagnostic for the AMPEX 80 megabyte disk attached to the IOP - 0 expander chasis. It is loaded by the diagnostic monitor program :DOM". Parameters are requested and processed by "DOM" before giving control to "F80M". The options to be entered are as follows:

SECT = Enter the section numbers to be run by setting the correct bit (octal 177 would select all seven test sections).

LOCL = Beginning cylinder (0-1466) - default = 0.

LOHD = Beginning head (0-4) - default = 0.

HICL = High cylinder (0-1466) - default = 1466.

HIHD = High head (0-4) - default = 4.

NPAT = Number of data patterns (1-7) - default = 2.

DPAT = User data pattern (0-177777) - default = not used.

TO FORMAT AN 80 MB DISK

Deadstart the IOP with the special mini system supplied on tape.

Enter time and date when requested.

On the MIOP kernel console type in "DOM" (hit return).

The following message will be displayed:

```
DIAGNOSTIC ONLINE MONITOR ACTIVE

LOAD WHAT?
```

Type in "F80M" (hit return).

The above parameters will be displayed except for the SECT= parameter, followed by:

```
ENTER PARAMETERS:
```

Type in "SECT=16" (hit return) - this selects sections 2, 3, and 4.

The enter parameters message will be displayed again.

To start the formatting type in:

"GO" (hit return)

The following message will be displayed:

```
"WARNING - DATA WILL BE DESTROYED CONTINUE (Y OR N)"
```

If you are sure you have the right disk pack mounted type in:

"Y" (hit return)

Running messages will be displayed for each section as it starts. When formatting is complete, the following message will be displayed:

```
F80M STOPPED AT END OF SEC/TEXT
```

To terminate type in:

"TERM" (hit return)

NOTE: If errors are encountered and the program displays the message:

```
F80M STOPPED ON ERROR
```

The user can either enter "GO" (hit return) to continue or if you do not wish to stop on error:

"OFF=ST" (hit return) followed by:

"GO" (hit return) - the diagnostic will then continue displaying any errors but will not stop.

F80M CONSISTS OF SEVEN TEST SECTIONS

Section one is a DMA register test. It loads the DMA register with random values, then reads them back and compares them.

Section two is the disk formatter. It writes headers and blank data fields starting at low cylinder and low head, and ending at high cylinder high head. One track at a time is written followed by a check checksum on that track.

Section three is a surface analysis test. It writes from one to seven data patterns between low cylinder, low head and high cylinder, high head eight sectors at a time. It then reads back the data written and compare it to the data read. If an error is encountered, the following message is displayed on the kernel console:

```
CYL = XXX HD = Y SECT = ZZ
```

```
Where:  CYL = the cylinder number
        HD = the head address
        SECT = the sector number
```

Section four reads the bad sector table created by running sections two and three. If no bad sectors were encountered, the following message is displayed on the kernel console:

```
NO BAD SECTORS
```

If any bad sectors were encountered on cylinder zero the following message will be displayed on the kernel console:

```
WARNING - CYL 0 HEAD 0 BAD
```

NOTE: If cylinder zero head zero is bad the disk pack is unusable. If bad sectors were encountered on tracks other than cylinder zero head zero, section four will read in the ID's on that track, reposition then for writing, set the bad sector flag for bad sectors on that track and rewrite the format on that track.

Section five is a sector address test. It writes on all sectors of low cylinder and low head. The data portion of each sector is written with the sector number as the value of each parcel of data. Each sector is then read back and compared to be sure the correct number was written in the correct sector.

Section six is a ram buffer echo test and a disk echo test. It first writes two sectors of zero's and reads it back to be sure the buffer is working. It then writes two sectors on the disk at low cylinder low head using the standard data patterns or "DPAT" if specified. The number of patterns used is determined by "NPAT". The two sectors are read back and compared to be sure the data sent equals the data received.

Section seven is a seek test. It issues a seek to low cylinder, low head and makes sure an interrupts is generated when seek end is set.

CONSOLE MESSAGES

When F80M is first started a warning message is displayed on the kernel console to warn anyone running this test that data will be destroyed. The test will not run unless a response is received indicating that the user wishes to continue. The message displayed is as follows:

```
     WARNING - DATA WILL BE DESTROYED, CONTINUE (Y OR N)
```

Error messages:

```
     DMA REGISTER ERROR
          EXP=XXXXXX
          ACT=XXXXXX

     RAM BUFFER ERROR
          EXP=XXXXXX
          ACT=XXXXXX

     NOT READY

     DEVICE TIMEOUT

     DROPPED READY

     DRIVE FAULT

     DRIVE TIMEOUT

     SEEK ERROR

     SECTOR OVERFLOW

     ID HEAD ERROR

     ID CYLINDER ERROR
```

IOS SYSTEM DIAGNOSTICS QUIZ

1. What is the difference between COS online or IOS system diagnostics ?

2. What are the features of F80M ?

3. What monitor is used for MIOP system diagnostics ?

4. What command at what console is needed to bring up this monitor ?

5. What does HSPTEST check and how would you determine it's failure ?

6. What does MOSTEST check and how would you determine it's failure ?

7. Where would you find listings for IOS system diagnostics ?

8. What action would be best to take in a IOS system diagnostic failure ?

9. Should IOS system diagnostics be run during normal system operation ?

10. What language are IOS system diagnostics written in ?

IOS SYSTEM DIAGNOSTICS QUIZ

1. What is the difference between COS online or IOS system diagnostics ?

2. What are the features of FBOM ?

3. What monitor is used for MIOP system diagnostics ?

4. What command at what console is needed to bring up this monitor ?

5. What does HSPTEST check and how would you determine it's failure ?

6. What does MOSTEST check and how would you determine it's failure ?

7. Where would you find listings for IOS system diagnostics ?

8. What action would be best to take in a IOS system diagnostic failure ?

9. Should IOS system diagnostics be run during normal system operation ?

10. What language are IOS system diagnostics written in ?

Operational Aids and Utilities

5

MODULE OBJECTIVES

With the aid of all furnished materials, upon completion of this Operational Aids Module, the learner should be capable of:

1. Using EXTRACT to gather hardware information from the system log
2. Using HERG to create a hardware error report from the system log
3. Using FDUMP to format and print a system dump dataset

EXTRACT

4.1 INTRODUCTION

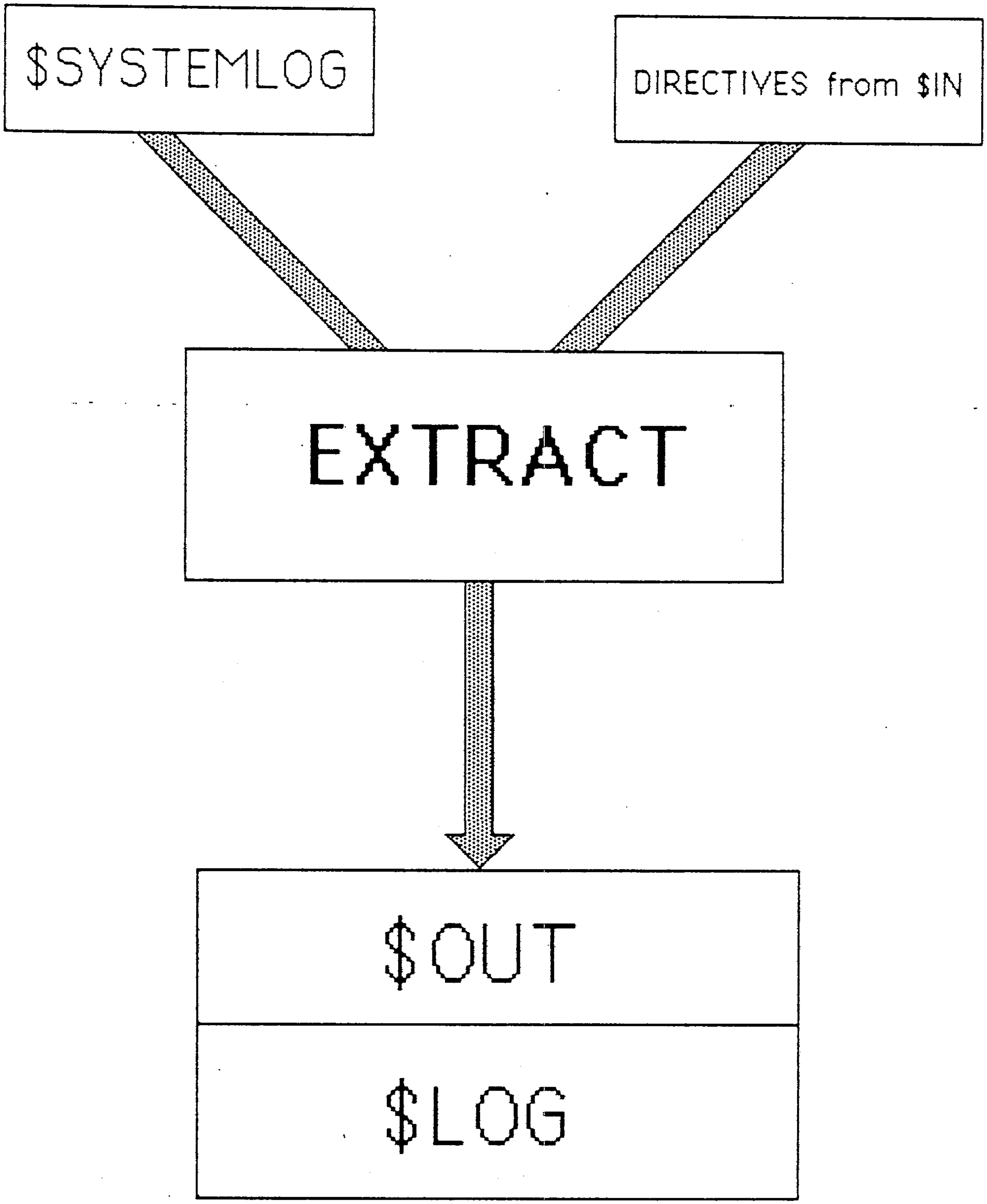
EXTRACT, an operating system utility program, selectively extracts and processes the contents of a system logfile containing messages issued by users and tasks during normal system operation. The system logfile is maintained by the Message Processor (MSG) task and contains a record for each message. Format and types of messages are described in the COS EXEC/STP/CSP Internal Reference Manual, publication SM-0040. EXTRACT diagnostic messages are described in the CRAY-OS Message Manual, publication SR-0039. Examples of EXTRACT directives are in Appendix A.

EXTRACT automatically accesses editions of $SYSTEMLOG to satisfy the user STARTIME and ENDTIME directives (see section 4.4.1) within the limitations imposed by the LE (lowest edition) and HE (highest edition) parameters of the EXTRACT control statement (see section 4.2). If the user does not specify the previous directives or parameters, the highest edition of $SYSTEMLOG is used. If the user supplies a local dataset $SYSLOG, only that dataset or a dataset specified as an input directive is processed.

EXTRACT is loaded into the user field and executed as a user job step when an EXTRACT control statement is encountered. The user controls the EXTRACT program through a set of directives. These directives are contained on the next file of the job's $IN dataset or on the dataset specified in the EXTRACT control statement. EXTRACT scans system logfiles looking for records satisfying user-specified criteria. When a record is found, EXTRACT processes it according to record type and subtype. For example, EXTRACT can calculate a CPU usage charge and write an entry on a report.

4.2 EXTRACT CONTROL STATEMENT

EXTRACT is loaded and executed using the following control statement.


$SYSTEMLOG
DIRECTIVES from $IN
EXTRACT
$OUT
$LOG

Format:

```
EXTRACT[,I=idn][,L=ldn][,PDN=pdn][,ID=id][,HE=he][,LE=le]
                                        [,ME=me][,ACCTLEN=al].
```

Parameters:

I=*idn* Name of dataset containing the EXTRACT directives. The default is $IN.

L=*ldn* Name of dataset to which the formatted listing is written. The default is $OUT. (This assignment can be overridden for one directive set by using the OUTPUT directive.) L=0 suppresses listing output.

PDN=*pdn* Permanent dataset name of the set of datasets automatically accessed. The default is $SYSTEMLOG. (PDN is provided as an aid for testing.)

ID=*id* ID of the set of datasets automatically accessed. The default is null. (ID is provided as an aid for testing.)

HE=*he* The highest edition of the set of datasets automatically accessed. (See section 4.3 for details on automatic dataset accessing.) The default is the highest existing edition.

LE=*le* The lowest edition of the set of datasets automatically accessed. (See section 4.3 for details on automatic dataset accessing.) The default is the lowest existing edition.

ME=*me* The ME parameter can be used to specify the number of successive missing editions that will be tolerated by the automatic accessing feature. By default, EXTRACT stops if it encounters five missing editions during the automatic scan (that is, ME=4).

ACCTLEN=*al*
The length, in Cray words, of the binary accounting record written to local dataset ACCOUNT. If omitted, the length of the record is appropriate for the current release. To read a system logfile written by COS release 1.11 and earlier releases, specify ACCTLEN=37.

```
    EXTRACT VERSION X.1.3     COMPILED [illegible]

17:38.795Ø  * RECOVERED DISK ERROR       READ IOP DEVICE: DD-A2-25    PERM. DATASET: CSIM      JSQ/JOBNAME:   1Ø82/SM
                   CYLINDER: ØØ36                   HEAD: ØØØØ              SECTOR: ØØØ1

17:38.7952   DISK ERROR PACKET     ERROR TYPE = Read data error           FINAL STATUS = Recovered
                 IOP/CHAN:      ØØ2/Ø25      FUNCTION:             READ
                 CYLINDER:      ØØØØ36       RETRY COUNT:              5
                 HEAD GROUP:    ØØØØØØ       FAULT STATUS:        Ø1ØØØØ
                 SECTOR:        ØØØØØ1       INTERLOCK STATUS:    ØØØØØØ
                 EARLY/LATE:    NORMAL       OFFSET/DIRECTION:    1Ø/EDGE
             Data error channel 3

22:35.8843   DISK ERROR PACKET                    ERROR TYPE:                 Seek error    FINAL STATUS:           Unrecovered
             IOP/CHAN:                  2/32      DEVICE TYPE:                      DD49    RETRY COUNT:                     1Ø
             EXPECTED CYLINDER:       ØØ1126      EXPECTED HEAD GROUP:            ØØØØØØ    EXPECTED SECTOR:             ØØØØ25
             CONTROLLER STATUS:       Ø53211      DRIVE GENERAL STATUS:           Ø4ØØ1Ø    FUNCTION:                      Seek
             STATUS  Ø:  ØØØ4Ø1      STATUS  1:  ØØØØØØ      STATUS  2:  177777      STATUS  3:  ØØØØØØ
             STATUS  4:  ØØØØ31      STATUS  5:  ØØØØØ5      STATUS  6:  ØØ1126      STATUS  7:  14Ø131
             STATUS  8:  ØØØØØØ      STATUS  9:  ØØØØØØ      STATUS 1Ø:  1ØØØ46      STATUS 11:  Ø1ØØØØ
             STATUS 12:  ØØØØØ1      STATUS 13:  ØØØØØØ      STATUS 14:  177777      STATUS 15:  ØØ1511
             STATUS 16:  ØØØØØØ      STATUS 17:  ØØØØØØ      STATUS 18:  15Ø32Ø      STATUS 19:  ØØØ4Ø3
             STATUS 2Ø:  ØØØØØØ      STATUS 21:  ØØØØØØ      STATUS 22:  ØØØØØØ      STATUS 23:  ØØØØØØ
             END CONTROLLER STATUS:  ØØØ2ØØ     END DRIVE GENERAL STATUS:     177777    END FUNCTION:                  Seek

22:35.9727   * UNRECOVERED DISK ERROR    WRITE IOP DEVICE: DD-A2-32    LOCAL DATASET: DDA232    JSQ/JOBNAME:   1Ø82/DD32A2E
                     CYLINDER: 1126                   HEAD: ØØØØ              SECTOR: ØØØØ

- - - - - END OF EXTRACT REPORT       7Ø423 RECORDS READ FROM $SYSLOG         4 RECORDS DISPLAYED ON $OUT
```

4.3 AUTOMATIC DATASET ACCESSING

If a local dataset $SYSLOG exists at the initiation of EXTRACT or $SYSLOG is specified as the operand of an INPUT directive, only that dataset is used. Otherwise, EXTRACT automatically accesses one or more editions of $SYSTEMLOG or editions of the dataset specified by the PDN and ID control statement parameters.

The automatically accessed set of editions is bounded by the values of the LE and HE control statement parameters and the set of permitted edition numbers ranging from 1 to 4095.

If the LE parameter is specified on the control statement, EXTRACT scans editions from the first existing edition with an edition number not less than the LE edition number and continuing with higher numbered editions.

If the LE parameter is not specified but the directive set contains a STARTIME directive, EXTRACT examines editions by decreasing the edition number until it finds one with a first record having a timestamp earlier than the specified STARTIME. If such an edition is found, EXTRACT scans forward from that edition. The first edition examined is the highest existing edition with an edition number no greater than the edition number specified by the HE control statement parameter.

If LE is not specified on the control statement and STARTIME is not specified in the directive set, EXTRACT processes the highest existing edition with an edition number no greater than the edition number specified by the HE control statement parameter.

When an automatically accessed dataset is exhausted, EXTRACT continues accessing successively higher editions until terminated by one of the following.

- LINES directive limit
- ENDTIME directive limit
- HE control statement parameter limit
- ME control statement parameter limit
- Edition number 4095

4.4 EXTRACT DIRECTIVES

EXTRACT directives permit extensive user selection of data extracted from the system logfile and user control over the format in which the data is processed.

```
1        EXTRACT VERSION X.13     COMPILED 03/12/84  RUN 03/20/84  14:39:06   ON SYSTEMLOG 03/20/84          PAGE    1

13:16:55.0814   * RECOVERED DISK ERROR        READ IOP DEVICE: DD-A2-22      PERM. DATASET: NASTRAN    JSQ/JOBNAME:    176/CRUSH
                        CYLINDER: 0175                     HEAD: 0007                 SECTOR: 0015

13:31:09.2660   * RECOVERED DISK ERROR        READ IOP DEVICE: DD-A2-22      PERM. DATASET: NASTRAN    JSQ/JOBNAME:    176/CRUSH
                        CYLINDER: 0175                     HEAD: 0007                 SECTOR: 0015

13:39:06.0926   * RECOVERED DISK ERROR       WRITE IOP DEVICE: DD-A1-21      LOCAL DATASET: ROLL DNT   JSQ/JOBNAME:      0/
                        CYLINDER: 1135                     HEAD: 0002                 SECTOR: 0007

13:45:47.0689   * RECOVERED DISK ERROR        READ IOP DEVICE: DD-A2-22      PERM. DATASET: NASTRAN    JSQ/JOBNAME:    176/CRUSH
                        CYLINDER: 0175                     HEAD: 0007                 SECTOR: 0015

13:45:48.9830   * RECOVERED DISK ERROR        READ IOP DEVICE: DD-A1-21      LOCAL DATASET: NAST29     JSQ/JOBNAME:    176/GNISS3
                        CYLINDER: 1220                     HEAD: 0000                 SECTOR: 0007

13:51:47.5770   * RECOVERED DISK ERROR        READ IOP DEVICE: DD-A1-21      LOCAL DATASET: NAST29     JSQ/JOBNAME:    176/GNISS3
                        CYLINDER: 1220                     HEAD: 0000                 SECTOR: 0000

13:51:56.9643   * RECOVERED DISK ERROR        READ IOP DEVICE: DD-A1-21      LOCAL DATASET: NAST29     JSQ/JOBNAME:    176/GNISS3
                        CYLINDER: 0652                     HEAD: 0007                 SECTOR: 0001

13:57:56.0994   * RECOVERED DISK ERROR        READ IOP DEVICE: DD-A2-20      LOCAL DATASET: ROLL DNT   JSQ/JOBNAME:      0/
                        CYLINDER: 1463                     HEAD: 0001                 SECTOR: 0005

14:00:18.4556   * RECOVERED DISK ERROR        READ IOP DEVICE: DD-A2-22      PERM. DATASET: NASTRAN    JSQ/JOBNAME:    176/CRUSH
                        CYLINDER: 0175                     HEAD: 0007                 SECTOR: 0015

14:21:57.6901   * UNRECOVERED DISK ERROR      READ IOP DEVICE: DD-A2-26      LOCAL DATASET: DDA226     JSQ/JOBNAME:    176/-XA226-
                        CYLINDER: 0261                     HEAD: 0005                 SECTOR: 0000

- - - - - - - END OF EXTRACT REPORT        53935 RECORDS READ FROM $SYSLOG           10 RECORDS DISPLAYED ON $OUT

EX001 - Unexpected token: 00110000
                 SELECT TYPE=HARDWARE,SUBTYPE=DISK.                                        00110000;

     - Discarded token:           ;
1
        14:38:48       0.0000   CSP    -------------------------------------------------------------------------------
        14:38:48       0.0000   CSP
        14:38:48       0.0000   CSP     ***  Use the    NEWS     control statement for general CRAY news. ***
        14:38:48       0.0000   CSP
        14:38:48       0.0000   CSP    03/14/84 - The amount of BUFFER MEMORY available to the user
        14:38:48       0.0000   CSP               as DVN=BMR-0-20 on the X-MP is now 368 tracks. It
        14:38:48       0.0000   CSP               was previously 383 tracks.
        14:38:48       0.0000   CSP
        14:38:48       0.0001   CSP             - An archive of datasets not accessed in 21 days was
        14:38:48       0.0001   CSP               run at 1230, 03/14.
        14:38:48       0.0001   CSP
        14:38:48       0.0001   CSP    03/13/84 - There will be no BATCH on the X-MP between the hours
        14:38:48       0.0001   CSP               of 0945 - 1115 on Wednesday, March 21, due to a demo.
        14:38:48       0.0001   CSP
        14:38:48       0.0001   CSP    -------------------------------------------------------------------------------
        14:38:48       0.0001   CSP            CRAY X-MP SERIAL-101/6     CRI - MENDOTA HEIGHTS, MINN. 03/20/84
        14:38:48       0.0001   CSP
        14:38:48       0.0001   CSP            CRAY OPERATING SYSTEM          COS X.13  ASSEMBLY DATE 03/14/84
        14:38:48       0.0001   CSP
        14:38:48       0.0001   CSP
        14:38:48       0.0001   CSP    JOB,JN=U1502A.                                                              00010000
        14:38:48       0.0010   CSP    ACCOUNT,AC=,US=,UPW=.                                              00020001
        14:38:54       0.0034   EXP    ***********************************************************************00030000
        14:38:54       0.0034   EXP    *                                                                           00040000
        14:38:54       0.0034   EXP    *     THIS JOB EXTRACTS DISK HARDWARE ERRORS FROM THE SYSTEMLOG            00050000
        14:38:54       0.0034   EXP    *          SWCE EX13A                                                       00060000
        14:38:54       0.0034   EXP    *                                                                           00070000
```

EXTRACT processes directives in a set until encountering an END statement or a period, then it processes the system logfile. A file of directives can contain more than one set of directives. The following example consists of two sets of directives. The first set uses the system logfile with the local dataset name LOGA and writes a report on FILEA. The second set reads the logfile with the local dataset name $SYSLOG (or if there is no local dataset $SYSLOG, datasets selected by the automatic accessing feature) and writes a report on FEFILE.

```
OUTPUT=FILEA, INPUT=LOGA,
SELECT TYPE=ASCII, SELECT SUBTYPE=PDM, END,
OUTPUT=FEFILE, TYPE=HARDWARE, SUBTYPE=DISK.
```

An EXTRACT directive can begin in any column of a record but must be completely contained on a given record; continuation of a directive to the next record is not permitted. A single record can contain multiple directives separated by commas or semicolons. The final directive on a record need not be terminated since the end of the record implies a comma or semicolon. Blanks used in directives are optional and are ignored when used. The following directive formats do not explicitly show the comma that can be required as a separator.

When a verb is omitted from an EXTRACT directive, it is assumed to be SELECT. Appropriate defaults are supplied for all directives. For example, the OUTPUT directive specifies the dataset on which the report is written. Its default is the dataset specified by the L control statement parameter. The OUTPUT directive is required only when a user wants to modify the listing dataset selected by the L control statement parameter.

A null or empty directive file causes EXTRACT to write all entries in the system logfile.

EXTRACT supports the following directive verbs.

- SELECT Specifies records selected for processing
- INPUT Specifies local dataset name of system logfile scanned
- OUTPUT Specifies dataset on which the EXTRACT report is written
- LINES Specifies maximum number of records (in decimal) EXTRACT writes on the output dataset
- FLUSH Requests EXTRACT to write 100 dummy messages in the system log
- NOHEADER Turns off page headings in the EXTRACT report
- DUMP Causes EXTRACT to write the report in octal, formatted into 64-bit words

- RAWDUMP Causes EXTRACT to write the raw source data from which the report was taken onto a dataset named RAWDUMP
- LEFT8 Causes addresses in memory failure records to be modified reflecting the design of the left eight banks of memory (that is, only the left half of memory is available for use at the time of failure)
- RIGHT8 Causes addresses in memory failure records to be modified reflecting the design of the right eight banks of memory (that is, only the right half of memory is available for use at the time of failure)
- END Permits the use of multiple sets of EXTRACT directives on a single file
- SUMMARY Suppresses the normal printing of TYPE=HARDWARE, SUBTYPE=SINGLE DOUBLE messages and instead, prints a summary report immediately before termination

4.4.1 SELECT DIRECTIVE

SELECT directives specify records selected for processing. Multiple values can be specified in the special cases of SUBTYPE and MSGID. A maximum of five subtypes can be specified on a single SELECT directive. Examples of uses of the SELECT directive are in Appendix A.

Because some fields used for selection are only in specific message types, not all selection criteria are relevant for all message types. The description of the system logfile given in the COS EXEC/STP/CSP Internal Reference Manual, publication SM-0040, indicates the relevant fields for specific types.

Formats:

```
SELECT parameter,

parameter,
```

Parameters:

JOBNAME=*jn*

The job name used in selecting records to be extracted. All messages associated with this job name are selected.

USER=*n*† User number; 1-15 alphanumeric characters. Only records pertaining to the specified user number are extracted.

DATASET=*dn*†

The dataset name used in selecting records to be extracted. All PDM messages associated with this dataset are selected. *dn* is 1-7 alphanumeric characters.

STARTIME=*hours:minutes:seconds month/day/year*

The time and day after which messages are scanned in the logfile. Each value must be given as two digits. The placement of time and date can be changed. Time of day is required. Date is optional; the default is today's date. For example,

```
STARTIME=14:28:00  03/14/80
```

If the STARTIME directive is not provided, EXTRACT scans the local dataset $SYSLOG, if $SYSLOG is provided; otherwise, EXTRACT scans for the highest edition number of $SYSTEMLOG (it cannot be higher than the edition number specified by the HE control statement parameter) of the set of datasets specified by the PDN and ID control statement parameters.

ENDTIME=*hours:minutes:seconds month/day/year*

The time and day when message scanning of the logfile terminates. Each value must be given as two digits. The placement of time and date can be changed. Time of day is required. The date is optional. If ENDTIME > STARTIME, the date defaults to the same date as STARTIME; otherwise, the date defaults to the next day.

If EXTRACT is scanning a user-provided $SYSLOG dataset, logfile scanning terminates at the end of the logfile. If the automatic access feature is used and the ENDTIME directive is not provided, scanning terminates by exceeding the highest edition permitted by the HE control statement parameter or by exceeding the number of successive missing editions permitted by the ME control statement parameter.

ENDTIME=NOW

A special form of the ENDTIME field causing the date and time to be set to the time of EXTRACT initiation. This form of the field is used when times later than the EXTRACT run initiation are in the system log.

† Deferred implementation

JOBSEQ
JSQ } =*n* Sequence number of user job. Indicates records of a user job to be processed.

SOURCE=*sid*†

Station ID of front-end computer. Indicates messages for jobs or datasets originating at the front-end computer identified by *sid*.

DEST=*sid*† Station ID. Selects messages for jobs or datasets destined for the front-end computer identified by *sid*.

$LOG= {YES
NO}

If YES, selects messages also written to a user log. If NO, user log messages are not selected. The default is all messages, if written to a user log or not.

TERMINAL=*tid*†

Terminal ID. Indicates terminal for which selected messages are destined or from which they originated.

MSGID=$msgid_1\ msgid_2 \ldots msgid_5$

5-alphanumeric character message identifier. The identifier is compared to the first five characters of an ASCII-type message to determine if it should be selected. A maximum of five identifiers can be specified on a single SELECT directive. For example,

```
SELECT MSGID=JS001 CL001 CL003
```

selects all ASCII-type messages beginning with the text JS001, CL001, or CL003.

TYPE=*type* Type of log entry. Types available are:

ACCOUNT	All records relating to machine resource usage by a user job
ASCII	ASCII character string messages, including all user-oriented messages
HARDWARE	All hardware errors detected by the operating system during normal operation
MISC†	Miscellaneous records such as attempted security violations
SPM	Performance and usage reports on COS, the CPU, and system I/O

† Deferred implementation

STARTUP	Messages issued by system Startup
STATION	Station-related messages
TRACE	System trace records

$\left.\begin{array}{l}\text{SUBTYPE}\\ \text{SUB}\end{array}\right\}=\mathit{subtype}_1\mathit{subtype}_2\ldots\mathit{subtype}_5$

Subtypes relevant to a particular major type. The TYPE parameter, if specified, must precede a SUBTYPE parameter. Examples of subtype use are found in Appendix A. If a TYPE directive is not specified, EXTRACT deduces the type from the subtype. For example, coding SUBTYPE=SINGLE allows EXTRACT to deduce TYPE=HARDWARE. However, if a subtype mnemonic is not unique (for example, SUBTYPE=DISK), a TYPE directive must be supplied resolving the ambiguity. All specified subtypes must belong to the same type if the type of a request is explicitly specified or deduced from the first subtype operand. The valid subtypes for each type follow.

ACCOUNT subtypes available are:

ACT	PDM accounting messages
TERM	Job termination messages
TQM	TQM tape accounting messages

ASCII subtypes available are:

ABORT	User abort messages
CSP	Control Statement Processor messages
DEC	Disk Error Correction messages
DQM	Disk Queue Manager messages
EXP	Exchange Package Processor messages
JCM	Job Class Manager messages
JSH	Job Scheduler messages
LOG	Message Processor messages
MEP	Memory Error Processor messages
MSG	Message Processor messages
PDM	Permanent Dataset Manager messages
SCP	Station Call Processor messages
SPM	System Performance Monitor messages
STP	Task initialization messages
TQM	Tape Queue Manager messages
USER	User-originated messages

HARDWARE subtypes available are:

CHANNEL	Channel errors
DISK	Disk errors
DOUBLE	Uncorrected memory errors
HWMSG	Miscellaneous hardware messages
IOPERR	IOP error messages
IOPDISK	IOP disk error messages
IOPTAPE	IOP tape error messages
SINGLE	Corrected memory errors

SPM subtypes available are:

CHANINT	Channel interrupt counts
CLASS	Job class statistics
CPU	CPU usage
DISK	Same as DISKUSE
DISKCHAN	Disk channel usage
DISKUSE	Disk usage
EXECCALL	EXEC call usage
EXECREQ	EXEC requests from each task
JSHSTAT	Job Scheduler statistics
LINK	Link usage
MEM	User memory usage
SYSBUF	System buffer memory usage
TASK	Task usage
USERCALL	User call usage

STARTUP subtypes available are:

HCR	Hardware characteristics record
PDR	Permanent dataset recovery messages
RRJ	Recovery of rolled job messages

STATION subtypes available are:

RECEIVED	Messages relating to staging in of datasets
TRANSMIT	Messages relating to staging out of datasets

NOTE

The two previous STATION subtypes will not be used after COS release version 1.12. The station will replace the RECEIVED subtype with the ACQUIRE subtype for user jobs. The message content of TRANSMIT will be retrieved using TYPE=ASCII,SUBTYPE=SCP but the content of the TRANSMIT remains the same.

COMMANDS Messages relating to station operator commands
MESSAGES Messages relating to station messages received from front end or system task

TRACE subtype available is:

TQMTRACE Tape Queue Manager trace buffer

4.4.2 INPUT DIRECTIVE

The INPUT directive specifies the local dataset name of the system logfile to be scanned. This directive allows several logfiles to be processed in one job. The INPUT directive is not required if only one logfile dataset is to be processed and if that dataset has been accessed with the local dataset name of $SYSLOG. If the INPUT directive is not given and the local dataset name $SYSLOG does not exist at the time of program invocation, EXTRACT automatically accesses editions of the set of datasets specified or implied by the PDN and ID control statement parameters using the local dataset name $SYSLOG.

Format:

INPUT=*dn*,

Example:

```
INPUT=DAYLOG,
```

4.4.3 OUTPUT DIRECTIVE

The OUTPUT directive specifies the name of the dataset on which the EXTRACT report is written. The default dataset is the dataset specified by the L control statement parameter.

Format:

OUTPUT=*dn*,

Example:

```
OUTPUT=TEMP,
```

EXTRACT output is written on TEMP instead of $OUT.

4.4.4 LINES DIRECTIVE

The LINES directive specifies the maximum number of $SYSTEMLOG records (in decimal) EXTRACT writes on the output dataset. The default limit is 1000, which is the count of $SYSTEMLOG records and not the number of physical lines printed.

Format:

```
LINES=n,
```

Example:

```
LINES=500,
```

4.4.5 FLUSH DIRECTIVE

The FLUSH directive requests EXTRACT to write 100 dummy messages in the system log, ensuring that system log memory buffers are flushed to disk.

Format:

```
FLUSH,
```

4.4.6 NOHEADER DIRECTIVE

The NOHEADER directive turns off page headings in the EXTRACT report. This feature is useful when the report is processed by another program.

Format:

```
NOHEADER,
```

Example:

```
MSGID=FT001,NOHEADER,LINES=10000.
```

4.4.7 DUMP DIRECTIVE

DUMP causes EXTRACT to write the report in octal, formatted into 64-bit words. (The ASCII character representation appears in the right margin.)

Format:

```
DUMP,
```

4.4.8 RAWDUMP DIRECTIVE

The RAWDUMP directive causes EXTRACT to write the raw source data from which the report was taken onto a dataset named RAWDUMP.

Format:

```
RAWDUMP,
```

4.4.9 LEFT8 AND RIGHT8 DIRECTIVES

The LEFT8 and RIGHT8 directives are designed for installations with a CRAY-1 Computer System hardware configuration having only the left or the right eight banks of memory. These directives are used in directive sets specifying hardware-type messages with memory error subtypes. LEFT8 and RIGHT8 cause the addresses in the memory failure records to be modified reflecting a specific 8-bank configuration. If the SUMMARY directive obtains a memory error summary report, all possible configurations appear in the report. LEFT8 and RIGHT8 are not required when the SUMMARY directive is used.

Formats:

```
LEFT8,

RIGHT8,
```

Example:

```
TYPE=HARDWARE
SUBTYPE=SINGLE,LEFT8
```

4.4.10 END DIRECTIVE

The END directive or a period allows multiple directive sets to be processed in one EXTRACT run.

Formats:

```
END

.
```

Example:

```
SELECT TYPE=ASCII, LINES=10000,
OUTPUT=DAYLOG, END
SELECT TYPE=HARDWARE.
```

The END directive or period is not required to terminate the final directive set.

4.4.11 SUMMARY DIRECTIVE

The SUMMARY directive suppresses the normal printing of TYPE=HARDWARE, SUBTYPE=SINGLE DOUBLE messages. Instead, a summary report of memory errors is printed at the end of the report.

HERG

HERG (HARDWARE ERROR REPORT GENERATOR) takes the place of BREAKER and SORT. It sould be run online.

CONTROL STATEMENT:

```
HERG,SDATE=date,EDATE=date,STIME=time,ETIME=time,ED=xx,
     LASTED=xx,SUMMARY,CPU,DISK=xx,BMHSC,TAPE,LME,LEFT8,
     RIGHT8,CHIPTYP=xx,SN=xx,RELLEV=xx,IOP=xx,CHAN=xx,
     SORT=xxx,CSLEVEL=x,STATS,PDN=xxxxxxxxxxxxxxx,ID=xxxxxx.
```

SDATE - Date from which to start report.

EDATE - Date at which to end report.

STIME - Starting time.

ETIME - Ending time.

ED - First edition of $SYSTEMLOG to search.

LASTED - Last edition of $SYSTEMLOG to search.

SUMMARY - Produce summary of CPU memory errors.

CPU - List CPU memory errors.

DISK - List DISK errors for specified device (1.11 RELEASE); DISK alone indicates all devices.

IOP - IOP on which disk is attached (1.12+ RELEASE); CHAN alone indicates all IOPs.

CHAN - IOP CHANNEL (octal) on which disk is attached; CHAN alone indicates all CHANNELS.

SORT - SORT=DISK sorts all disk errors by device.
SORT=CPU sorts CPU memory errors by chip.
SORT alone does both.

BM - List BUFFER MEMORY errors.

HSC - List HIGH SPEED CHANNEL errors.

TAPE - List TAPE errors.

LME - List LOCAL MEMORY errors.

CSLEVEL - Sort level for CHIP summary.

STATS - Output STATISTICAL summary only.

SN - CPU SERIAL NUMBER.

RELLEV - Release level, ie. 11 (FOR 1.11), X12 (FOR X.12) etc.

PDN - PDN of file to scan, defaults to $SYSTEMLOG.

ID - ID for PDN to scan, defaults to zeroes.

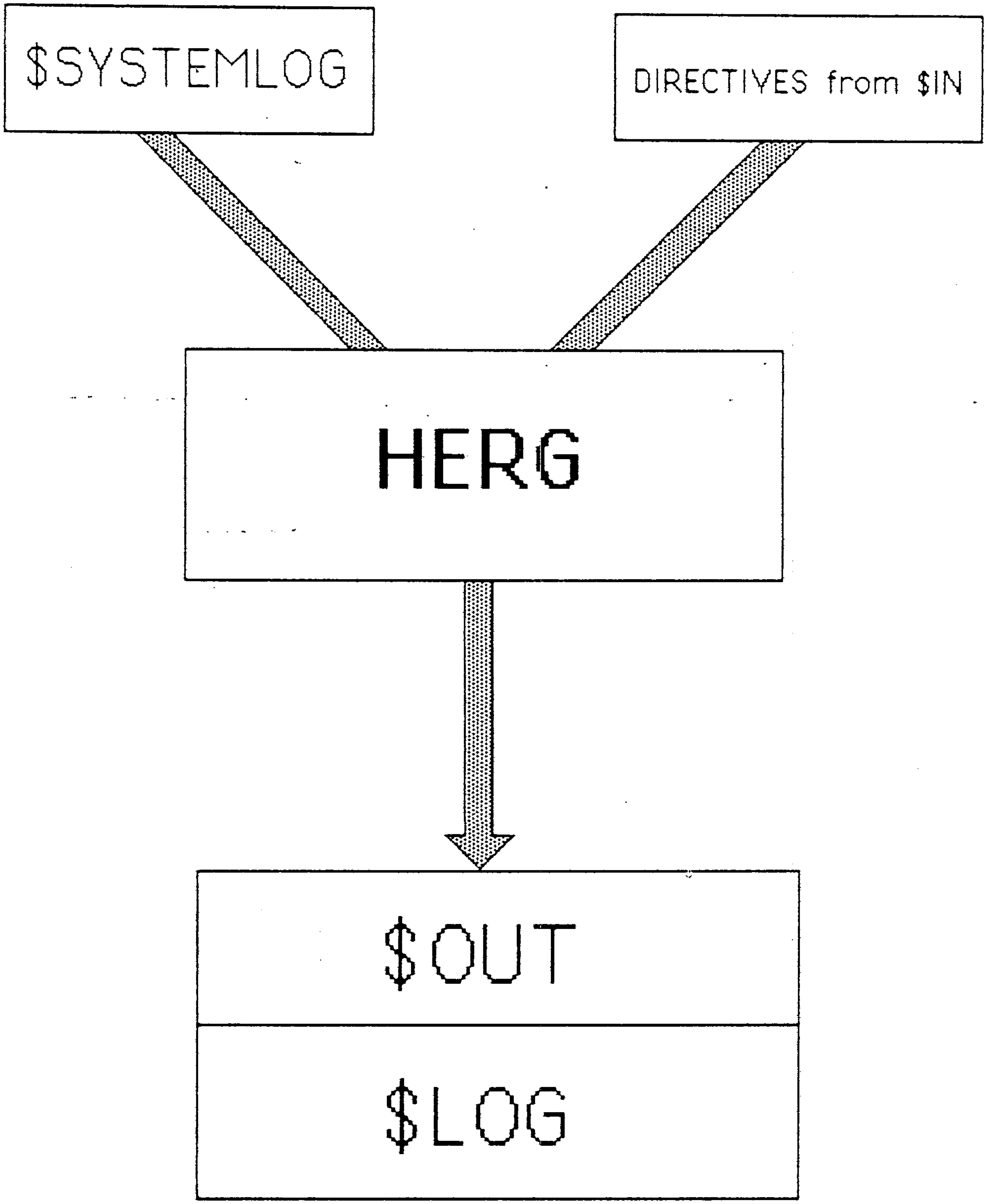
THE FOLLOWING UNIT NUMBERS ARE USED:

FT10 - FT25 Data for memory errors by column.

FT49 Burst data.

FT50 Contains memory error sort.

FT51 - FT82 Data for disk error by device.


$SYSTEMLOG
DIRECTIVES from $IN
HERG
$OUT
$LOG

```
                                                   CRAY-1 STATISTICAL REPORT
                                                      SERIAL NUMBER  201

                                                START DATE: 09/17/84      00:00:00
                                                 END DATE:  09/17/84      23:59:37
                                                 RUN DATE:  09/18/84      14:17:17

TOTAL CRAY MEMORY ERRORS:          4
      CORRECTABLE ERRORS:          4                  BURSTING MODULES
      UNCORRECTABLE ERRORS:        0                  MOD   FREQ   COUNT
      MODULE FALLOUT:              2                  ------------------
      BURST ERRORS:                2                  K03   00010000002
      BURST PERIODS:               1
      BURSTING MODULES:            1
      ADJUSTED ERROR COUNT:        2


TOTAL DISK ERRORS:                 9
      RECOVERABLE ERRORS:          4
      UNRECOVERABLE ERRORS:        0


TOTAL COMMON MEMORY ERRORS:        0
      CORRECTABLE ERRORS:          0
      UNCORRECTABLE ERRORS:        0


TOTAL LOCAL MEMORY ERRORS:         0
      IOP-0:                       0
      IOP-1:                       0
      IOP-2:                       0
      IOP-3:                       0


TOTAL HIGH SPEED ERRORS:           0
      INPUT CHAN A:                0
      OUTPUT CHAN A:               0
      INPUT CHAN B:                0
      OUTPUT CHAN B:               0
      INVALID ERRORS:              0


TOTAL TAPE ERRORS:                51
```

```
           ERROR REPORT RUN   09/18/84  14:11:50                SERIAL NUMBER 201                 PAGE   1

    09/17/84    MONDAY

09:34:36.2189  DISK ERROR PACKET                ERROR TYPE:     Read     FINAL STATUS:         Recovered
                REASON RECOVERY INVOKED:  DN AND BZ BOTH SET
                IOP/CHAN:               2/22     DEVICE TYPE:                   DD49      RETRY COUNT:                 1
                EXPECTED CYLINDER:    000104     EXPECTED HEAD GROUP:         000007      EXPECTED SECTOR:        000043
                CONTROLLER STATUS:    021611     DRIVE GENERAL STATUS:        000200      FUNCTION:                    Read
                STATUS  0: 000402       STATUS  1: 051123        STATUS  2: 000104        STATUS  3: 003400
                STATUS  4: 000046       STATUS  5: 000022        STATUS  6: 003443        STATUS  7: 140135
                STATUS  8: 000000       STATUS  9: 000000        STATUS 10: 000000        STATUS 11: 000000
                STATUS 12: 000025       STATUS 13: 000000        STATUS 14: 177777        STATUS 15: 001510
                STATUS 16: 000040       STATUS 17: 000000        STATUS 18: 070160        STATUS 19: 000401
                STATUS 20: 000000       STATUS 21: 000001        STATUS 22: 000000        STATUS 23: 000000
                A - OFFSET: Disabled                             B - OFFSET: Disabled
09:34:36.2376   * RECOVERED DISK ERROR  ,   READ IOP DEVICE = 49-A2-22,  LOCAL DATASET = A222    ,  JOBNAME = A222
                 CYLINDER = 0000,  HEAD = 00,  SECTOR = 00

09:36:37.7147  DISK ERROR PACKET                ERROR TYPE:     Read     FINAL STATUS:         Recovered
                REASON RECOVERY INVOKED:  DN AND BZ BOTH SET
                IOP/CHAN:               1/22     DEVICE TYPE:                   DD49      RETRY COUNT:                 1
                EXPECTED CYLINDER:    000744     EXPECTED HEAD GROUP:         000001      EXPECTED SECTOR:        000033
                CONTROLLER STATUS:    015611     DRIVE GENERAL STATUS:        000200      FUNCTION:                    Read
                STATUS  0: 000402       STATUS  1: 051123        STATUS  2: 000744        STATUS  3: 000400
                STATUS  4: 000036       STATUS  5: 000022        STATUS  6: 000433        STATUS  7: 140135
                STATUS  8: 000000       STATUS  9: 000000        STATUS 10: 000000        STATUS 11: 000000
                STATUS 12: 000025       STATUS 13: 000000        STATUS 14: 177777        STATUS 15: 001530
                STATUS 16: 000040       STATUS 17: 000000        STATUS 18: 070160        STATUS 19: 000401
                STATUS 20: 000000       STATUS 21: 000001        STATUS 22: 000000        STATUS 23: 000000
                A - OFFSET: Disabled                             B - OFFSET: Disabled
09:36:37.7371   * RECOVERED DISK ERROR  ,   READ IOP DEVICE = 49-A1-22,  LOCAL DATASET = A122    ,  JOBNAME = A122
                 CYLINDER = 0000,  HEAD = 00,  SECTOR = 00
09:38:00.9962   SINGLE  BIT MEMORY ERROR  EM = 1  RM = 2  SYN = 367  ADDR = 24000007  BIT = 39  LOCATION = N-14 C.S.05
                ERROR WORD = 0400020060036724000007  JOBNAME =  AC3DSS   BA = 26353000  P = 00043526D
09:38:29.4072   TAPE ERROR PACKET: DATA CHECK       ,  SOURCE ID =  C,  DEST ID = C1,  DENSITY = GCR (6250 BPI)
                TYPE: 0005,  BLOCK NUMBER = 005576,  RETRY COUNT = 001,  RECOVERED FLAG = RECOVERED
                                                 ORIGINAL                 LAST          SENSE BYTES
                             CHANNEL:              021                    021         08 C4 00 40 00 40 3D 00
                         DEVICE PATH:               13                     13         00 08 00 00 00 37 D1 7D
                       DEVICE STATUS:               0E                     0C         96 D2 00 00 04 00 10 80
                            FUNCTION:            WRITE                    WRITE
09:38:40.0775   TAPE ERROR PACKET: DATA CHECK       ,  SOURCE ID =  C,  DEST ID = C1,  DENSITY = GCR (6250 BPI)
                TYPE: 0005,  BLOCK NUMBER = 007555,  RETRY COUNT = 001,  RECOVERED FLAG = RECOVERED
                                                 ORIGINAL                 LAST          SENSE BYTES
                             CHANNEL:              021                    021         08 C4 00 40 00 40 3D 00
                         DEVICE PATH:               13                     13         00 88 00 00 00 37 D1 7D
                       DEVICE STATUS:               0E                     0C         96 D2 00 00 04 00 10 80
                            FUNCTION:            WRITE                    WRITE
```

COS OPERATIONAL AIDS MANUAL

5.1 INTRODUCTION

FDUMP is a utility program for formatting and printing the contents of a dataset containing an image of the CRAY-1 and CRAY X-MP Computer Systems Central Memory, I/O Subsystem (IOS) Local Memory, IOS Buffer Memory, and the Solid-state Storage Device, according to user-supplied directives. Normally, the dataset is created by system Startup following a system failure and operator-initiated memory dump. However, any dataset properly formatted is acceptable to FDUMP. See section 5.6 for a description of the required dataset format.

FDUMP executes as a job step within a user job during normal system operation. Ordinarily, the dump to be formatted is accessed by the job before calling FDUMP. However, the $DUMP dataset written by DUMPJOB can also be used as input to FDUMP. Options available in FDUMP include dumping absolute memory, dumping memory specified by symbolic constants, dumping symbolic values, dumping various preformatted dumps, copying the dump to another dataset in compressed format, and decompressing the data from a compressed dataset.

5.2 FDUMP CONTROL STATEMENT

FDUMP is loaded and executed using the following control statement.

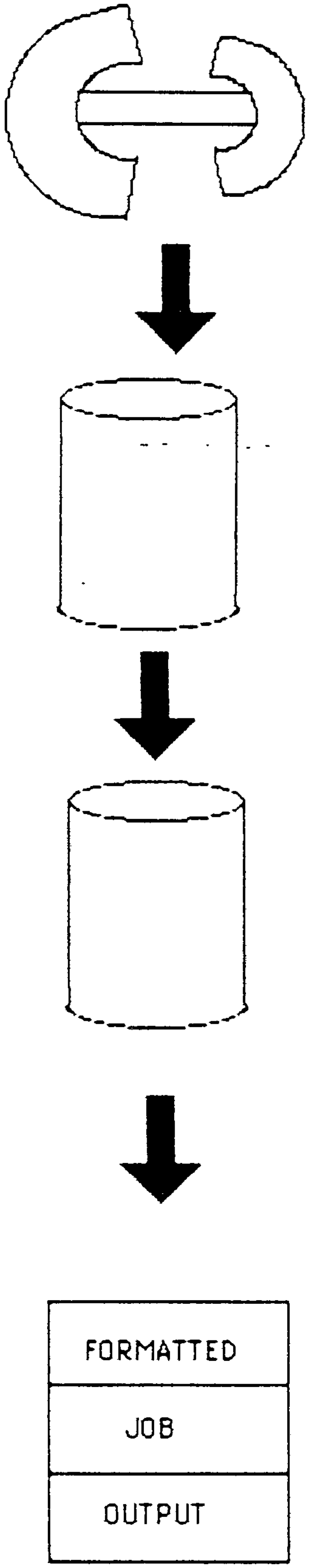
Format:

```
FDUMP[,I=idn][,L=odn][,TRANS=trans][,NPC=ch][,CPU=cpu][,LIMIT=limit].
```

Parameters:

I=*idn* Name of the dataset containing user directives. If I is omitted or specified without a value, the default is $IN. The dataset named by the I parameter, or implied by its defaults, cannot appear in the user directive file (as an operand of the FILES or AUTO directives).

FORMATTED SYSTEM DUMPS



CRAY SYSTEM CONTENTS

CNTL-D initiates SYSDUMP

DISK SCRATCH BUFFER

STARTUP processes buffer

CRAY1SYSTEMDUMP

PDN=
OWN=
ED=
R=

FORMATTED DUMP
IN $OUT

JOB with FDUMP directives

L=*odn* Name of the dataset to receive the listing output. If L is omitted or specified without a value, the default is $OUT. The dataset named by the L parameter or implied by its defaults cannot appear in the user directive file as an operand of the FILES or AUTO directives.

TRANS=*trans*

Translation option. If the TRANS parameter is omitted, lowercase characters are converted to the nonprinting character as specified or implied by the NPC parameter. If the parameter is specified without a value or as TRANS=UPC, lowercase characters are converted to uppercase. If TRANS is equated to NONE, no translation of printable characters occurs.

NPC=*ch* Nonprinting character option. If the NPC parameter is omitted, nonprinting characters are converted to blanks. If NPC is specified without a value, nonprinting characters are converted to periods. NPC can be equated to any single character; the specified character replaces nonprinting characters.

CPU=*cpu* CPU option. If omitted, and if the XP option on the DMEM control statement is not equated to anything, FDUMP attempts to format exchange packages from internal evidence. Since this is not always successful, an interpretation can be forced by specifying CPU=S for CRAY-1 machines or CPU=X for CRAY X-MP machines.

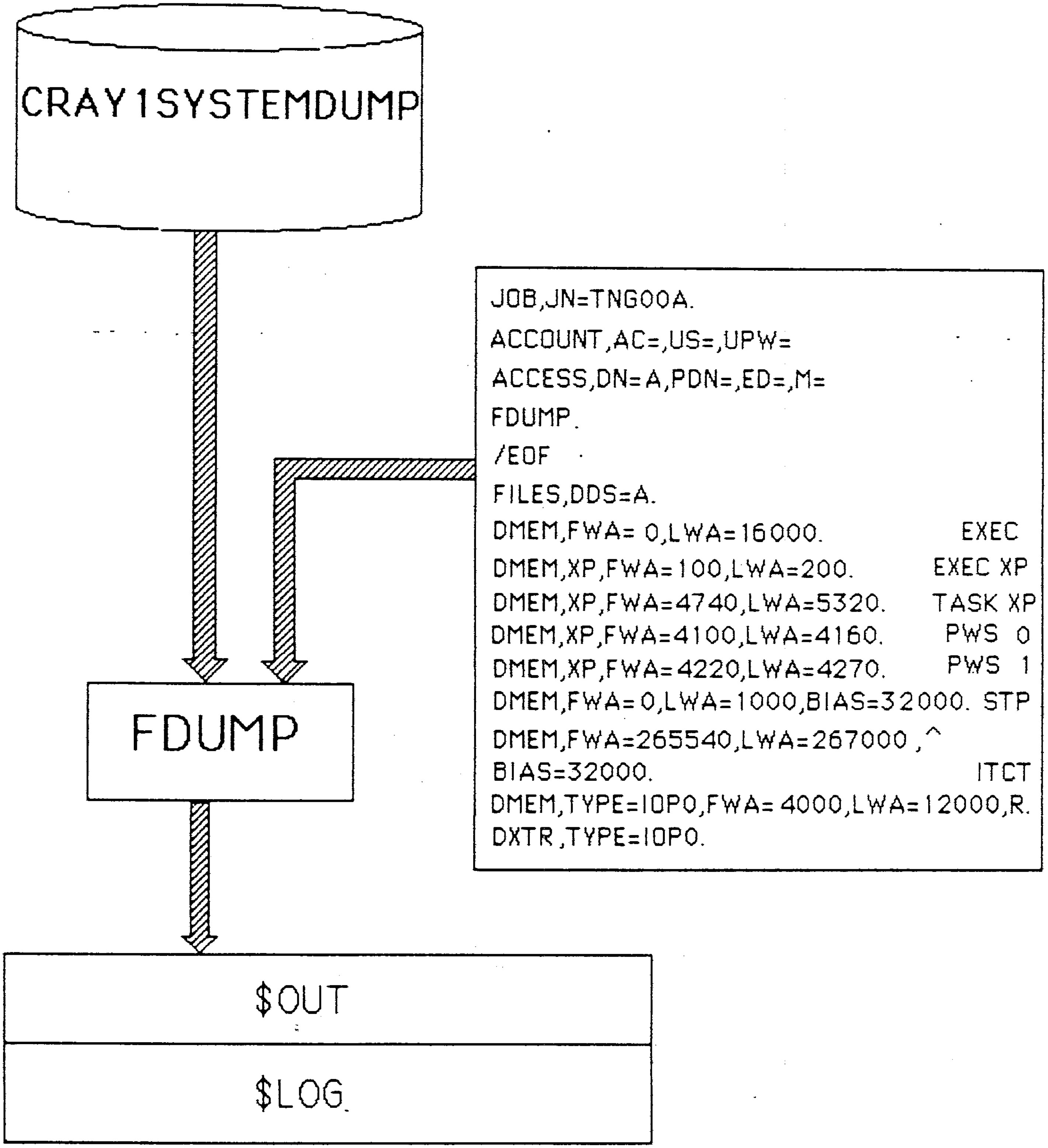
LIMIT=*limit*

Line limit option for the DXTR directive. If specified, LIMIT must be equated to a decimal number (applies to every DXTR directive). The LIMIT parameter can also be specified on the DXTR directive, but LIMIT only applies to the specific DXTR directive.

5.3 FDUMP DIRECTIVES

FDUMP interprets directives and performs the requested function. All FDUMP directives are processed in the order encountered. The format of an FDUMP directive is similar to a COS control statement. (See the CRAY-OS Version 1 Reference Manual, publication SR-0011, for a description of control statements.) The FDUMP directive is a free-form card image consisting of a verb with or without parameters. Some verbs require parameters. A parameter consists of a keyword, usually followed by a value. Parameters are order independent. A period terminates the directive. If there is no period before column 80, the end-of-card is treated as a period. Blanks are ignored. The listable output dataset

FDUMP



```
JOB,JN=TNG00A.
ACCOUNT,AC=,US=,UPW=
ACCESS,DN=A,PDN=,ED=,M=
FDUMP.
/EOF
FILES,DDS=A.
DMEM,FWA= 0,LWA=16000.                EXEC
DMEM,XP,FWA=100,LWA=200.           EXEC XP
DMEM,XP,FWA=4740,LWA=5320.       TASK XP
DMEM,XP,FWA=4100,LWA=4160.         PWS  0
DMEM,XP,FWA=4220,LWA=4270.         PWS  1
DMEM,FWA=0,LWA=1000,BIAS=32000. STP
DMEM,FWA=265540,LWA=267000 ,^
BIAS=32000.                            ITCT
DMEM,TYPE=IOP0,FWA= 4000,LWA=12000,R.
DXTR,TYPE=IOP0.
```

contains a copy of all directives encountered on the input dataset. (Directives read from the AUTO directive file are not copied onto the listable output dataset.)

FDUMP recognizes two classes of directives. The first class controls the memory to be dumped, specifying which datasets are involved and the memory dump limits. The second class controls the titles and spacing to produce a readable listing.

5.4 CLASS 1 DIRECTIVES

Class 1 directives control the memory to be dumped, specifying which datasets are involved and specifying the dump limits. Class 1 directives include the following verbs:

- FILES Identifies datasets containing the current dump and any symbol table datasets to be read in. Multiple dumps can be processed in a single execution by using multiple FILES directives. A FILES directive must precede all of the remaining directives, unless an AUTO directive specifies a directive dataset containing a FILES directive.
- DMEM Formats and prints memory range
- COMP Compresses a dump or portions of a dump onto an alternate dataset
- XCOMP Decompresses a dataset created by COMP onto an alternate dataset
- DSYM Prints the value of a symbol and the contents of the location addressed by the symbol
- AUTO Dumps memory according to a set of directives contained on an auxiliary dataset. A standard set of directives was contained within the program in releases before the 1.11 COS release. Beginning with the 1.11 COS release, the standard directive set is distributed as a separate dataset, AUTODIR.
- DSDT Formats and prints the System Dataset Table area from a COS system dump
- DXTR Formats and prints the EXEC trace area from a COS system dump. The trace can be from the Cray Central Memory (EXEC History Trace) or from IOP and Buffer Memory (IOP Kernel Trace).

- SETBIAS Defines an offset for subsequent addresses
- DSSD Formats and prints SSD sectors

5.4.1 FILES DIRECTIVE

The FILES directive identifies the datasets containing the dump to be processed and any symbol table datasets used for definition of symbolic address or length specifications. Multiple dumps can be processed in a single execution using multiple FILES directives.

Format:

FILES,DDS=*dn*[,SYM1=s_1][,SYM2=s_2][,SYM3=s_3].

Parameters:

DDS=*dn* Local dataset name of the current dump dataset. This parameter is required and must not name a dataset specified on the FDUMP control statement.

SYM$_n$=s_n The local dataset names of up to three datasets containing symbol tables written by the CAL assembler. FDUMP reads the first file from each dataset to construct its working symbol table.

The dump dataset must be unblocked and must contain a memory dump in the format written by Startup. It can be a permanent dataset previously accessed by the job. The symbol table datasets must be blocked.

Symbol table processing includes removal of duplicate symbols from the tables if the duplicated symbol contains the @ character in either the second or third character position. This process allows field definition symbols to be removed from the second and third symbol tables if their values are already known. Other duplicate symbols are retained.

Symbol table searching is ordinarily performed with the dataset identified by SYM1 searched before SYM2, and SYM2 searched before SYM3. On most directives supporting symbolic addresses, the symbol table search can be limited to those entries coming from one specific dataset. If FDUMP cannot find a symbolic name in the specified symbol tables, it writes an informative diagnostic to the list dataset and skips the directive.

Once a symbol table has been read in, all symbols defined in that table remain defined until a subsequent FILES directive is encountered, specifying a different dataset for the same SYM_n parameter. Subsequent FILES directives not specifying a particular SYM_n parameter do not affect the associated symbol table. A symbol table can be removed without replacement by specifying SYM_n=0.

Some examples of the FILES directive follow.

(1) FILES,DDS=DUMP1.

The current dump is contained in the local dataset with the name DUMP1; no symbolic information is used.

(2) FILES,DDS=DUMP2,SYM1=SYM.

The current dump is contained in the local dataset with the name DUMP2; symbol table information is contained in the local dataset with the name SYM.

(3) FILES,DDS=X,SYM1=SYM1,SYM2=SYM2.

The current dump is on dataset X; symbol tables are on datasets SYM1 and SYM2. The table from SYM1 is searched first. If no matching symbol is found, the table from SYM2 is searched.

(4) FILES,DDS=Y.

The current dump dataset is changed to Y. The directive does not affect the use of SYM1 and SYM2 for symbol definition.

(5) FILES,DDS=Y,SYM1=0.

Dataset SYM1 is deleted from consideration in symbol definition, leaving the current dump on Y and the symbols only on SYM2.

5.4.2 DMEM DIRECTIVE

The DMEM directive formats and prints a range of memory. In addition to dumping Cray Central Memory, the DMEM directive can be used to dump Buffer Memory, Local Memory with one to four IOPs (with or without IOP registers), and selected CSIM simulator tables.

The DMEM directive can be used with the AUTO directive to format memory AUTO does not ordinarily print. Addresses and lengths can be specified in a variety of ways if one or more symbol tables are provided. DMEM attempts to resolve parameters specified symbolically. At least one FILES directive specifying the current dump dataset must precede the first DMEM directive.

Memory limits must be specified when using DMEM to print IOP memory, if the user does not want only a dump of the registers from the specified IOP. Memory limits are supplied by specifying the first word address (FWA or FWA@) and the ending addresses specified in one of the following ways.

- LWA or LWA@ Specifies the last word address directly
- LWA1 or LWA@1 Specifies the last word address + 1
- L or L@ Specifies the number of words to dump
- LE or LE@ and NE or NE@
  Specifies the length of each entry in a table being dumped and the number of entries in the table, respectively

Several of the DMEM keywords have an alternate form with @ suffix signaling one level of indirect addressing. For example, FWA@=100 means the first word address dumped can be found as the low-order 22 bits of word 100 (subject to BIAS) in the dump. Indirect addressing is useful when dynamically allocated tables are dumped and the remaining dump contents become pointers for the tables. Keywords allowing the @ suffix are FWA, LWA, LWA1, L, NE, LH, LE, and BIAS. The @ suffix can be used only when dumping Central Memory.

Format for dumping Cray Central Memory:

```
DMEM[,TYPE=type],FWA=fwa{[,LWA=lwa][,LWA1=lwa1][,L=l]}[,LH=lh]

[,LE=le,NE=ne][,NOLE][,XP=xp][,FORMAT=format][,BIAS=bias][,SDN=sdn].
```

Format for dumping the I/O Processor registers and memory:

```
DMEM,TYPE=IOPn[,R][,FWA=fwa][,LWA=lwa][,LWA1=lwa1][,L=l].
```

Parameters:

TYPE=*type* Type of memory dumped. If unspecified, Cray Central Memory is dumped. The parameter cannot be specified without a value. The following keywords are recognized:

BMEM	Buffer Memory
BTVRG	CRAY-1 B, T, and V registers
BTVRG1	CRAY X-MP B, T, and V registers
CL01	Cluster registers for CRAY X-MP
CL02	Cluster registers for CRAY X-MP
CL03	Cluster registers for CRAY X-MP
CSIMDCU	CSIM disk control unit status
CSIMDIR	CSIM directive status
CSIMDSU	CSIM disk storage unit status
CSIMDSUC	CSIM disk storage unit contents
CSIMMISC	CSIM miscellaneous information
CSIMSTAS	CSIM station status
CSIMXP	CSIM current exchange package
CSIMXP1	CSIM CRAY X-MP exchange package
IOP0	IOP-0 registers and memory
IOP1	IOP-1 registers and memory
IOP2	IOP-2 registers and memory
IOP3	IOP-3 registers and memory
MEM	Cray Central Memory

Addresses of all IOP*n* types are interpreted as parcel addresses.

FWA=*fwa* Address of the first word dumped. If specified as FWA, the address is the symbol value or absolute octal number equated to it. If specified as FWA@, the address is the low-order 22-bit value of the word addressed by the symbol or the absolute octal number equated to it.

LWA=*lwa* Address of the last word dumped. If specified as LWA, the address is the symbol value or absolute octal number equated to it. If specified as LWA@, the address is the low-order 22-bit value of the word addressed by the symbol or the absolute octal number equated to it. LWA, LWA1, and L are mutually exclusive.

LWA1=*lwa1* If specified as LWA1, the address is the first word after the last word dumped. If specified as LWA1@, the address is the low-order 22-bit value of the first word after the last word dumped. LWA, LWA1, and L are mutually exclusive.

L=*l* Number of words (or parcels) dumped. If L, the parameter value is the symbol value or absolute octal number equated to it. If L@, the value of the parameter is the low-order 22-bit value of the word addressed by the symbol or absolute octal number equated to it. LWA, LWA1, and L are mutually exclusive.

LH=*lh* Header length at the beginning of a table being dumped. If specified as LH, the parameter value is the symbol value or absolute octal number equated to it. If specified as LH@, its value is the low-order 22-bit value of the word addressed by the symbol or the absolute octal number equated to it.

When LH is specified, the table header is individually dumped and separated by a blank line from the rest of the table. If LH is not specified, LH defaults to 0.

LE=*le* Length of each entry within a table being dumped. If LE is specified, NE must also be specified. If LE is specified, the parameter value is the symbol value or absolute octal number equated to it. If specified as LE@, its value is the low-order 22-bit value of the word addressed by the symbol or absolute octal number equated to it.

When LE is specified and NOLE is not specified, each table entry is individually dumped and separated by a blank line from the preceding table entry.

NE=*ne* Number of entries in a table being dumped. If NE is specified, LE is required. If specified as NE, the parameter value is the symbol value or absolute octal number equated to it. If specified as NE@, its value is the low-order 22-bit value of the word addressed by the symbol or absolute octal number equated to it.

NOLE Inhibits the dumping of each table entry separately when the LE keyword is used. NOLE is ignored unless LE or LE@ is also specified. NOLE is useful when using NE and LE together to specify a table length; it has no effect on the dumping of a table header with the LH parameter.

XP=*xp* Dumps the memory range in exchange package format. The last word address dumped is rounded up to an exchange package boundary if necessary. This parameter cannot be used when TYPE is specified as IOP*n*.

If XP is specified alone, the exchange package is dumped according to the CPU parameter on the control statement. If neither is specified, the exchange package is dumped in CRAY-1 format only if the fields used on the CRAY X-MP for the data base address and data limit address are both 0.

XP can be equated to S for the CRAY-1 or to X for the CRAY X-MP, forcing the appropriate exchange package format.

FORMAT=*format*
Dump format. Values are WORD or PARCEL. If FORMAT is omitted, the default is WORD.

BIAS=*bias* Bias value. *bias* is added to the first word address and last word address dumped. If specified as BIAS, the parameter value is the symbol value or absolute octal number equated to it. If specified as BIAS@, its value is the low-order 22-bit value of the word addressed by the symbol or absolute octal number equated to it. If specified with IOP*n*, this parameter is ignored.

bias applies only to the current directive and adds to the bias specified by a preceding SETBIAS directive (see section 5.4.9). The BIAS parameter is resolved before any other parameters and affects the evaluation of any other parameter keyword suffixed by @.

SDN=*sdn* Symbol table dataset used to define symbolic references in this directive. Ignored if TYPE=IOP*n* is specified. If *sdn* does not match one of the symbol table dataset names in effect from preceding FILES directives, FDUMP issues a warning and ignores the directive.

R Dumps registers of the specified IOP (register dump precedes memory dump if both are specified on the directive). If used with types other than IOP*n*, the parameter is ignored.

Examples using the DMEM directive follow.

Assume the following FILES directive is in effect:

```
FILES,DDS=DUMP1,SYM1=EXECSYM,SYM2=STPSYM.
```

EXECSYM contains the symbol table from assembly of COS EXEC, and STPSYM contains symbol table entries from the assembly of STP.

(1) DMEM,FWA=0,LWA=1000.

Dumps words 0_8 through 1000_8 of Cray Central Memory. The following forms also dump these words. (Assume Cray Central Memory location 100_8 contains 401_8.)

```
DMEM,FWA=0,LWA1=1001.
DMEM,FWA=0,L=1001.
DMEM,FWA=0,LH=401,LE=10,NE=40.
DMEM,FWA=0,LH@=100,LE=10,NE=40.
```

(2) DMEM,FWA=SXBF,L=20,XP.

Locates symbol SXBF in the supplied symbol tables and prints 20_8 words beginning at SXBF, formatted as an exchange package. SXBF is assumed to be relative to location 0 of the dump if a previous SETBIAS command was not specified.

(3) DMEM,FWA=SIM,L=1.

Dumps contents of location SIM.

(4) DMEM,FWA=SIM,L=1,SDN=STPSYM,BIAS=XEND.

Dumps contents of STP location SIM. The SDN parameter forces the STP symbol SIM to be used instead of the EXEC symbol. The BIAS parameter causes the base address of STP to be included when determining the address of the location dumped.

(5) DMEM,FWA=B@SDT,L=SZ@SDT,LE=LE@SDT,BIAS=XEND.

Dumps the System Dataset Table area. Each System Dataset Table entry is separated from the others by a blank line. LH defaults to 0, so a table header is not printed.

5.4.3 COMP DIRECTIVE

The COMP directive compresses a dump, deleting portions containing sequences of identical words more than three words long. Each section of memory on the output dataset is preceded by a control word indicating if compression occurred. If compression did not occur, the control word is followed by the uncompressed data. When compression is possible, the control word specifies first and last word addresses represented by the data following.

Format:

COMP,IDN=*idn*,ODN=*odn*[,FWA=*fwa*,LWA=*lwa*].

Parameters:

IDN=*idn* Name of dataset containing the compressed dump

ODN=*odn* Name of dataset receiving the compressed dump image

(2) DMEM,FWA=SXBF,L=20,XP.

Locates symbol SXBF in the supplied symbol tables and prints 20_8 words beginning at SXBF, formatted as an exchange package. SXBF is assumed to be relative to location 0 of the dump if a previous SETBIAS command was not specified.

(3) DMEM,FWA=SIM,L=1.

Dumps contents of location SIM.

(4) DMEM,FWA=SIM,L=1,SDN=STPSYM,BIAS=XEND.

Dumps contents of STP location SIM. The SDN parameter forces the STP symbol SIM to be used instead of the EXEC symbol. The BIAS parameter causes the base address of STP to be included when determining the address of the location dumped.

(5) DMEM,FWA=B@SDT,L=SZ@SDT,LE=LE@SDT,BIAS=XEND.

Dumps the System Dataset Table area. Each System Dataset Table entry is separated from the others by a blank line. LH defaults to 0, so a table header is not printed.

5.4.3 COMP DIRECTIVE

The COMP directive compresses a dump, deleting portions containing sequences of identical words more than three words long. Each section of memory on the output dataset is preceded by a control word indicating if compression occurred. If compression did not occur, the control word is followed by the uncompressed data. When compression is possible, the control word specifies first and last word addresses represented by the data following.

Format:

```
COMP,IDN=idn,ODN=odn[,FWA=fwa,LWA=lwa].
```

Parameters:

IDN=idn Name of dataset containing the compressed dump

ODN=odn Name of dataset receiving the compressed dump image

OPERATIONAL AIDS AND UTILITIES QUIZ

1. Explain what EXTRACT does and with what dataset(s) ?

2. What other names has HERG had in the past releases ?

3. Explain what HERG does and with what dataset(s) ?

4. What Cray publication is EXTRACT and FDUMP described in ?

5. How would you use the information obtained from EXTRACT or HERG ?

6. What does FDUMP used for and what does it do ?

7. What dataset does FDUMP process ?

8. What are FDUMP's directives used for ?

9. Where is the typical place to put FDUMP directives ?

10. What does FDUMP need for automatic formatted dumps ?

OPERATIONAL AIDS AND UTILITIES QUIZ

1. Explain what EXTRACT does and with what dataset(s) ?

2. What other names has HERG had in the past releases ?

3. Explain what HERG does and with what dataset(s) ?

4. What Cray publication is EXTRACT and FDUMP described in ?

5. How would you use the information obtained from EXTRACT or HERG ?

6. What does FDUMP used for and what does it do ?

7. What dataset does FDUMP process ?

8. What are FDUMP's directives used for ?

9. Where is the typical place to put FDUMP directives ?

10. What does FDUMP need for automatic formatted dumps ?

APML

6

MODULE OBJECTIVES

With the aid of all furnished reference materials, upon completion of this APML Module, the learner should be capable of:

1. Read APML source programs

2. Breakdown complex assignment and conditional statements

3. Write and assemble without errors an APML program

4. Use basic APML Pseudo's in a program

5. Use $APTEXT Macros in a program

A PROCESSOR MACRO LANGUAGE

APML is a powerful translator with middle language features.

```
APML,CPU=type,I=idn,L=ldn,B=bdn,E=edn,
    ABORT,DEBUG,LIST=name,S=sdn,SYM=sym
                            T=bst,X=xdn
```

Source Statements can be:

Symbolic Machine Instructions - APML Card

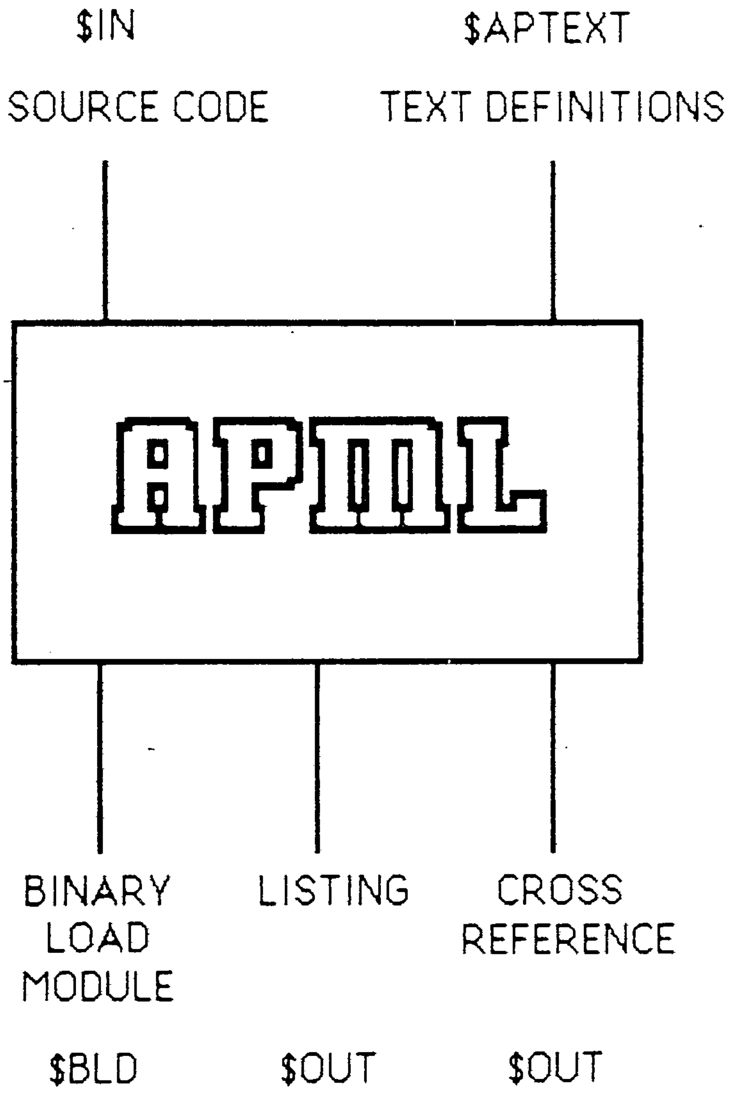
Complex Architecture Statements - Assignment Syntax

Pseudos - Controls Symbols and Assembler

Macros - Defined in $APTEXT

All CAL pseudo's available except COMMON and OPDEF/APML has unique Pseudos and Macros.

APML ASSEMBLER



$IN

SOURCE CODE

$APTEXT

TEXT DEFINITIONS

APML

BINARY LOAD MODULE

$BLD

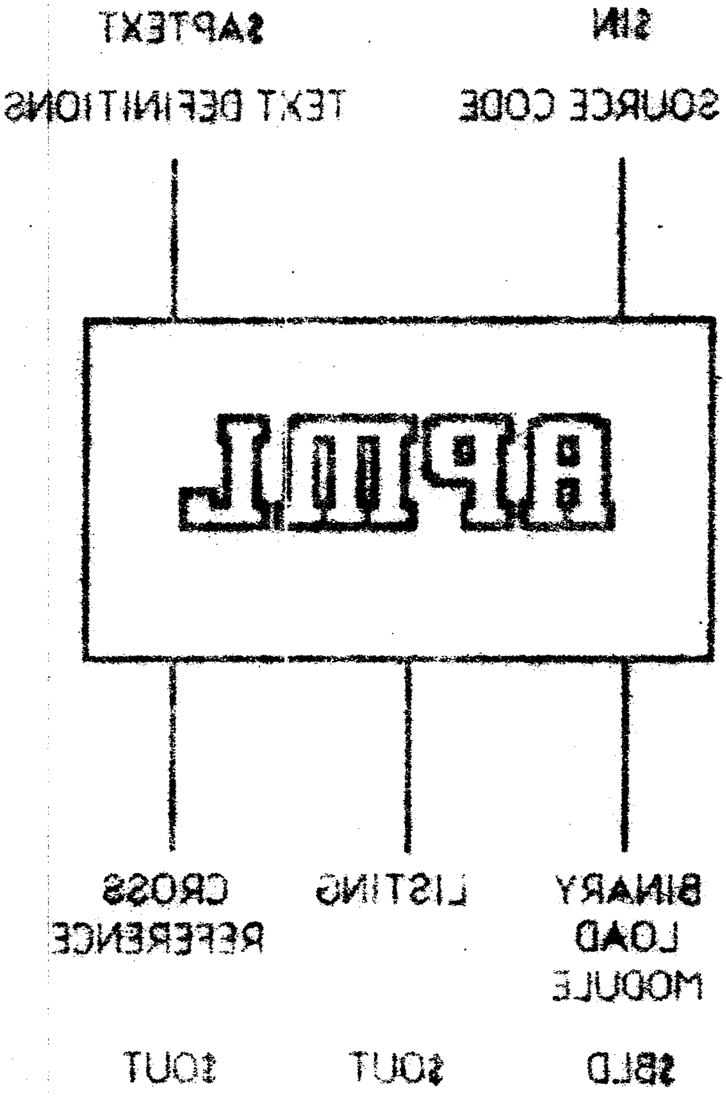
LISTING

$OUT

CROSS REFERENCE

$OUT

APML ASSEMBLER

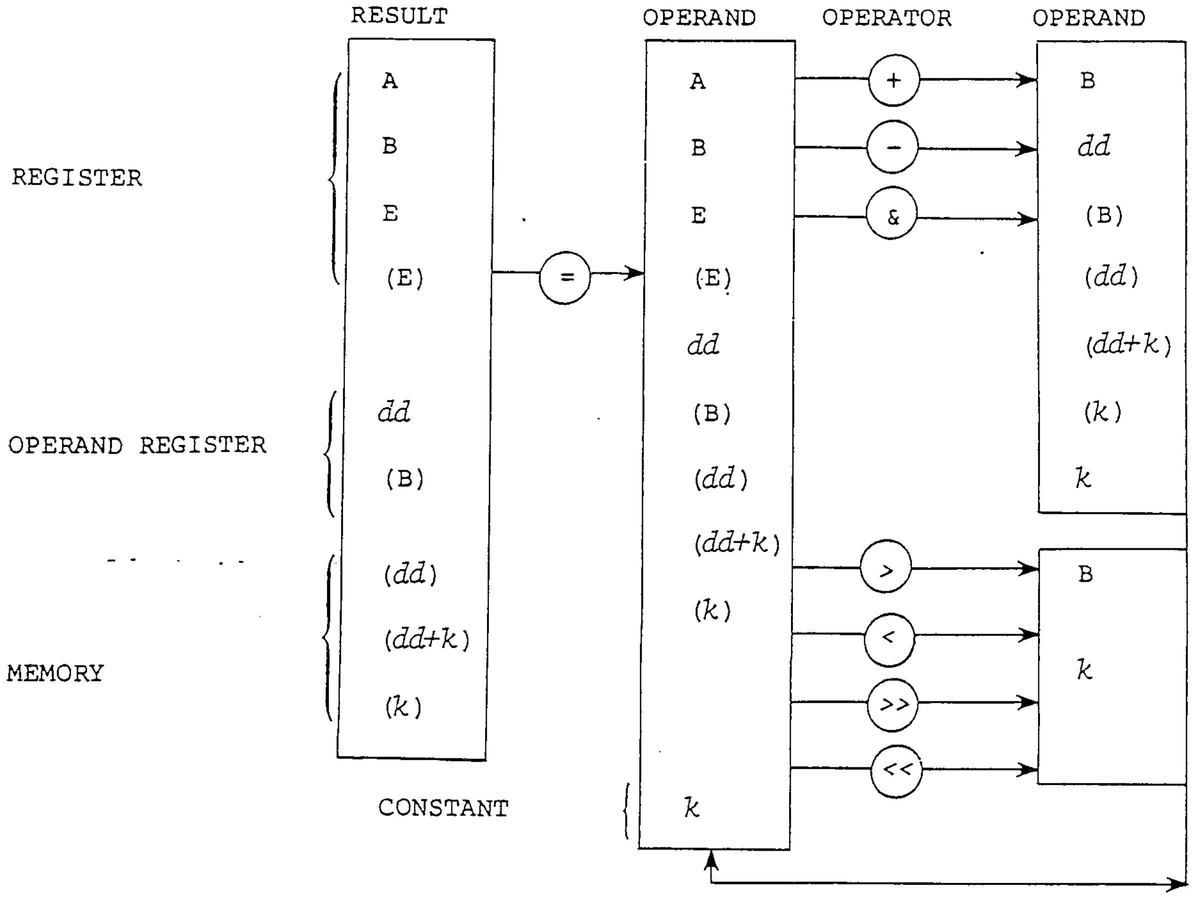


APML VS. CAL

```
                                      IDENT     CAL
                                      START     BEGIN
0   0000000000000000000012  NUM       CON       10
1                        1  SUM       BSS       1
                     2a+    BEGIN     =         *
2a  1001 00000000+                    A1        NUM,0
 c  <opdef>                           A2        1
 d  <opdef>                           A3        2
3a  <opdef>                           A4        0
 b  031110                  LOC       A1        A1-1
 c  030442                            A4        A4+A2
 d  030223                            A2        A2+A3
4a  030001                            A0        A1
 b  011 00000003b+                    JAN       LOC
 d  1104 00000001+                    SUM,0     A4
5b  <macro>                           ENDP
                                      END
```

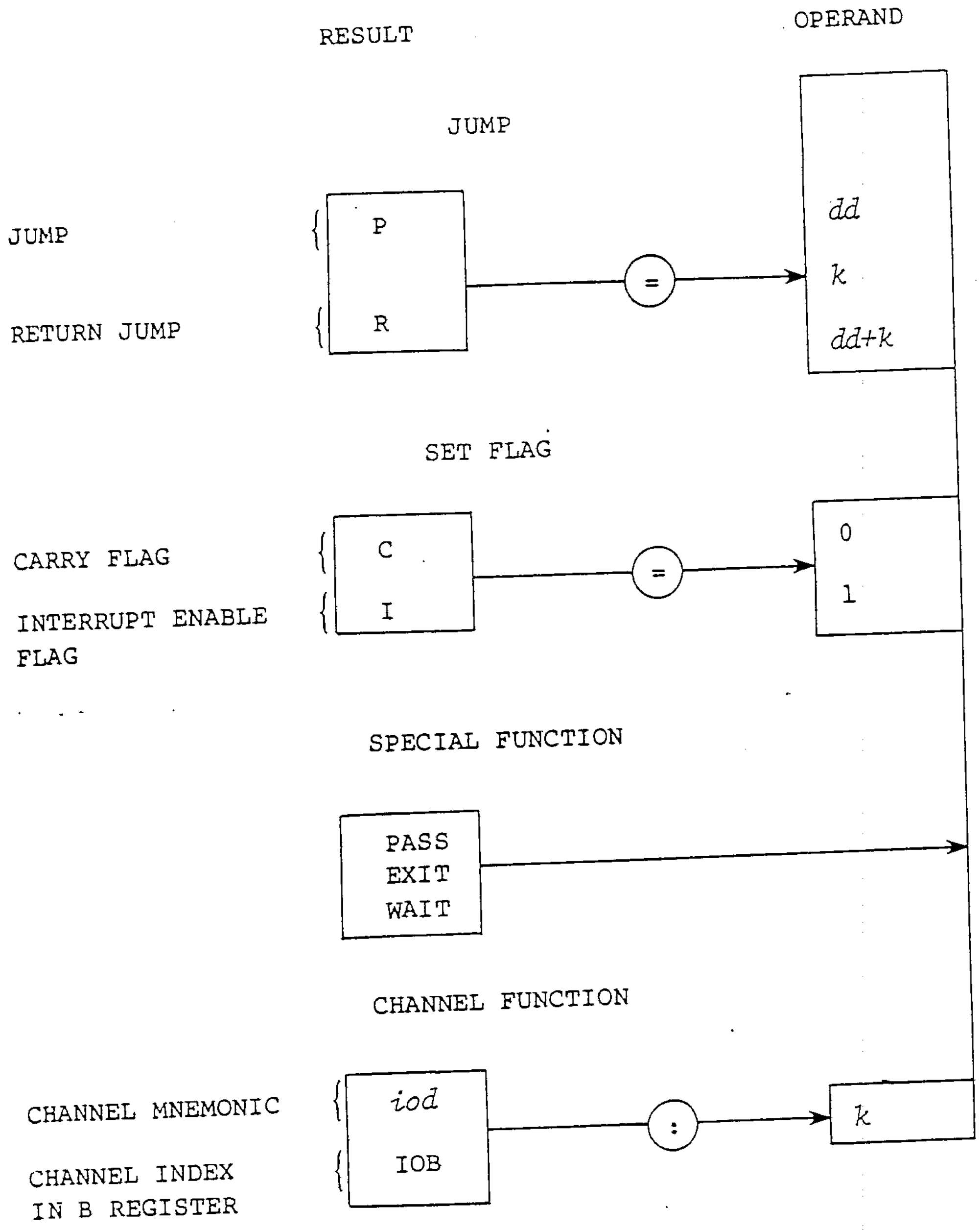
```
                                       IDENT     APML
                    0        SC        EQUALS    0
                    1        R1        EQUALS    1
                    2        R2        EQUALS    2
                    3        R3        EQUALS    3
                                       SCRATCH   SC
 0  010012  024001                     R1=12
 2  010001  024002                     R2=1
 4  010000  024003                     R3=0
 6  027001                   LOC       R1=R1-1
 7  020003  022002  024003             R3=R3+R2
12  010002  025002                     R2=R2+2
14  020001  107007                     P=LOC,R1#0
16  014000 /000023  024000             (SUM)=R3
    020003  034000
23                           SUM       <1>
                                       END
```

REPLACEMENT



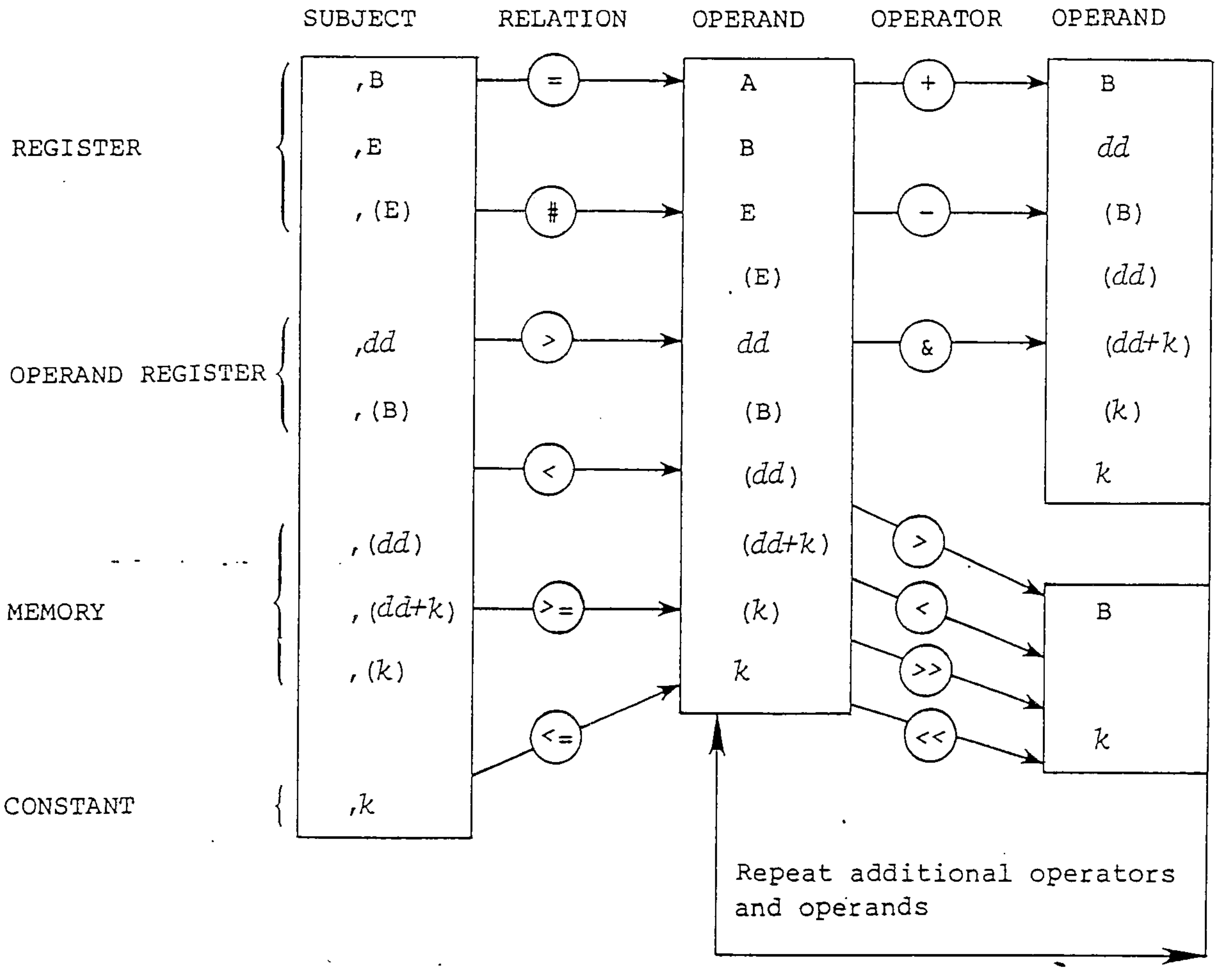
Repeat additional operators and operands

Assignment syntax

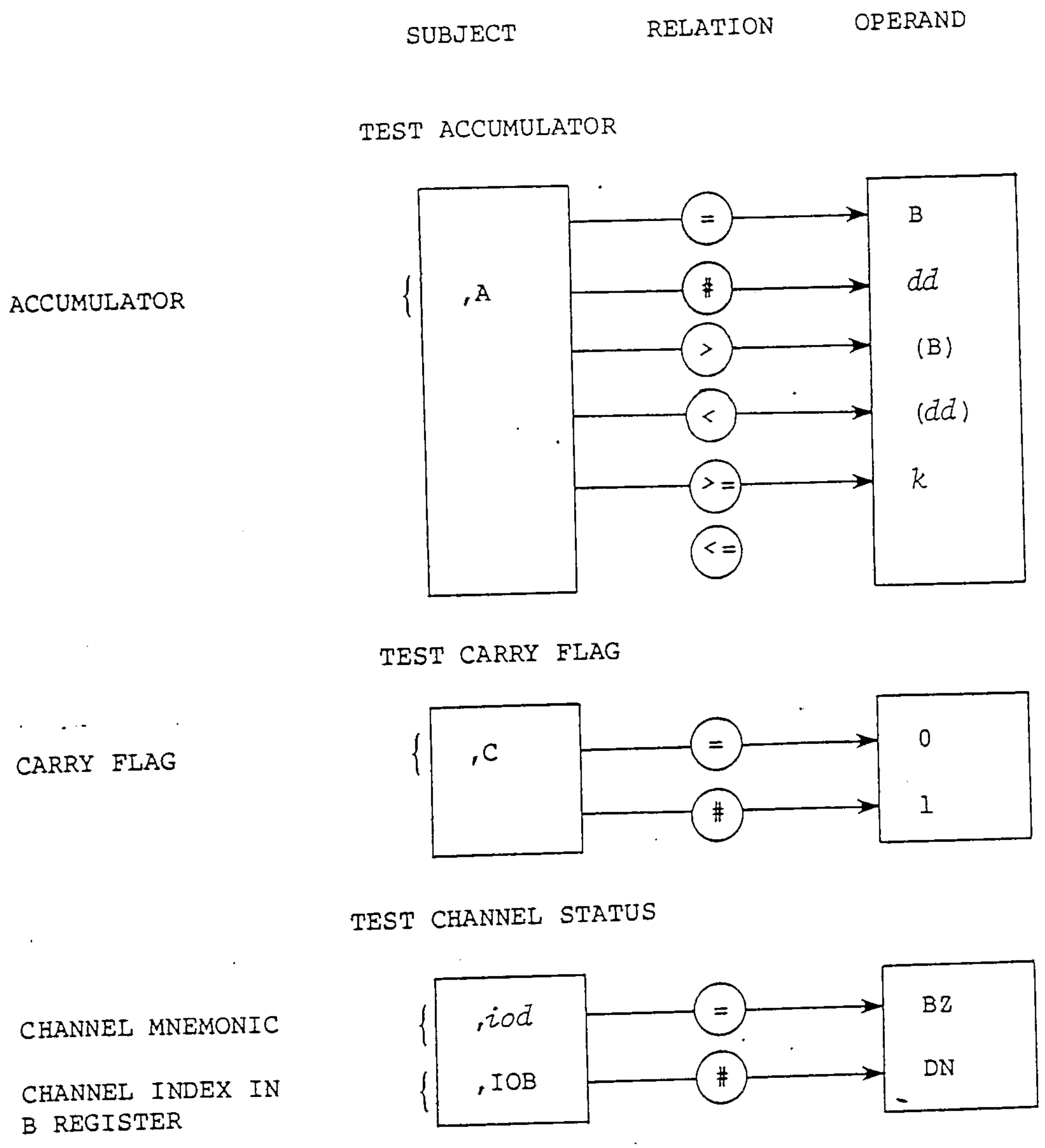


Assignment syntax (continued)

TEST REGISTER OR MEMORY



Condition syntax



Condition syntax (continued)

```
0       <macro>
0       000334
1       020432   062000   004010
        051000
5       074000   /000000  024410
        014000   /000023  024411
        150002    022430  033411
        034410
17      074431
20      040000
21      070000
22      174000
23      000223
```

```
IDENT           ASSIGN
REGDEFS         ,(R3,R4,R5)
334
A=R5+(B)>10&B

(LOC)=E+R3-(BOG)


P=R4
C=1
WAIT
IOB:14
223
END
```

```
0       <macro>
0       150002   052000   006010
        024410   020430   023410
        101002   050000
10      053000   103003   010007
        154002
14      101003   010000   054000
17      040005   100002   010000
```

```
IDENT           CONDIT
REGDEFS         ,(R3)
A=B,R3<E+B>>10


E=7,A=B

B=0,C=0
A=0,MOS=DN
END
```

```
                                    *
                                             IDENT     ADANGER
0     <macro>                                REGDEFS   ,(R1)
0    010007                                  A=7
1    020430   012025   024410                B=A+(R1+25)
     032410   054000
                                             END
```

```
                                    *
                                             IDENT     CDANGER
                           400      R3       EQUALS    400
0    010010                                  A=10
1    020400   013037   102002                B=A,R3#37
     054000
                                             END
```

IDENT AND END

Required

IDENT identifies program module.

IDENT is physically the first statement of each module.

END is physically the last statement of each module.

LOCATION	RESULT	OPERAND
IGNORED	IDENT	NAME
IGNORED	END	IGNORED

NAME - Name of Program Module.

Example:

```
                                  *
      0    050000                             IDENT      PSEUDO
                                              A=B
                                              END
```

EQUALS AND SET

Defines a symbol with the value and attributes determined by the expression.

Symbol is <u>not</u> redefinable for equals.

Symbol is redefinable for set.

LOCATION	RESULT	OPERAND
SYMBOL	EQUALS	EXP,ATTRIBUTE
SYMBOL	SET	EXP,ATTRIBUTE

SYMBOL - Unqualified Symbol

EXP - Any Expression

ATTRIBUTE - Optional, Overrides Attribute of EXP

P - PARCEL

W - WORD

V - VALUE

Example:

```
                                  *
                                           IDENT      EQUSET
                       3          R1       EQUALS     3
                                           BASEREG    R1
                    1024          GEORGE   EQUALS     1024
                      17          CAT      SET        17,P
   0    075003 /000017                     P=CAT
                    1031          CAT      SET        GEORGE+5
                                           END
```

BSS - BSSZ

Reserves 64 BIT words in local memory, starting at current location counter. Forces word boundary in doing so.

LOCATION	RESULT	OPERAND
SYMBOL SYMBOL	BSS BSSZ	COUNT COUNT

SYMBOL - Optional, is assigned word address of location counter

COUNT - Number of words

Example:

```
                                     *
                                             IDENT     BSSBSSZ
     0    050000                             A=B
      1W                       12    NON     BSS       12
     13W                        4    ZERO    BSSZ      4
     74                              HERE    *
                                             END
```

SCRATCH

Used to declare scratch registers for generating code from complex statements.

LOCATION	RESULT	OPERAND
IGNORED	SCRATCH	R_1,R_2,R_3,R_4,R_5

R_I Up to 5 previously defined or non-definable symbols. Symbols must be defined elsewhere.

Example:

```
                             *
                                       IDENT     SCRATCH
                      1      SHARK     EQUALS    1
                      6      DO        SET       6
                                       SCRATCH   SHARK,DO,DA
                      4      DA        EQUALS    4
0                            LOC       <1>
1   014000 /000000  024001             (LOC)=(1057)
    014000 /001057  024006
    030006  034001
                                       END
```

BASE

Allows specification of numeric data being octal, decimal, or mixed. Default is octal.

LOCATION	RESULT	OPERAND
IGNORED	BASE	DBASE

DBASE   Desired base. O-OCTAL, D-DECIMAL, M-MIXED
          *Reverts to previous base

Example:

```
                                  *
                                         IDENT     BASE
0    010012                              A=12
1                                 BASE   *
1    010012                              A=12
                                         BASE      D
2    010014                              A=12
                                         BASE      *
3    010012                              A=12
                                         END
```

REGISTER

LOCATION	RESULT	OPERAND
ORIGIN	REGISTER	(SYM_1,SYM_2,...)

ORIGIN Starting operand register number (octal)

SYM_I List of symbols to be assigned to operand register

Same as the following:

SYM_1 Equals ORIGIN

SYM_2 Equals ORIGIN + 1

.

.

.

SYM_I Equals ORIGIN + (I-1)

Example:

```
                                   #
                                              IDENT      REGISTER
      0    <macro>                 7          REGISTER   (R1,AA,CAT)
                                              SCRATCH    R1
      0   030011   024010                     AA=(R!CAT)
                                              END
```

BASEREG

A base register is required for two parcel jumps and for referencing data in a relocated piece of code (overlay).

Two parcel jumps ,DD+K, are generated by the assembler for branch points outside of the current 'page'.

A page is a block of source code within which all branches are relative, i.e., one parcel.

'Pages' are delimited by 'page boundaries' which are formed as follows:

1. IDENT Statement

2. At 512 Parcels

3. By a Pseudo Instruction which forces a Word Boundary

4. By a PDATA Pseudo Instruction with a Label

6. By a NEWPAGE Pseudo Instruction

The BASEREG pseudo is used to declare a base operand register.

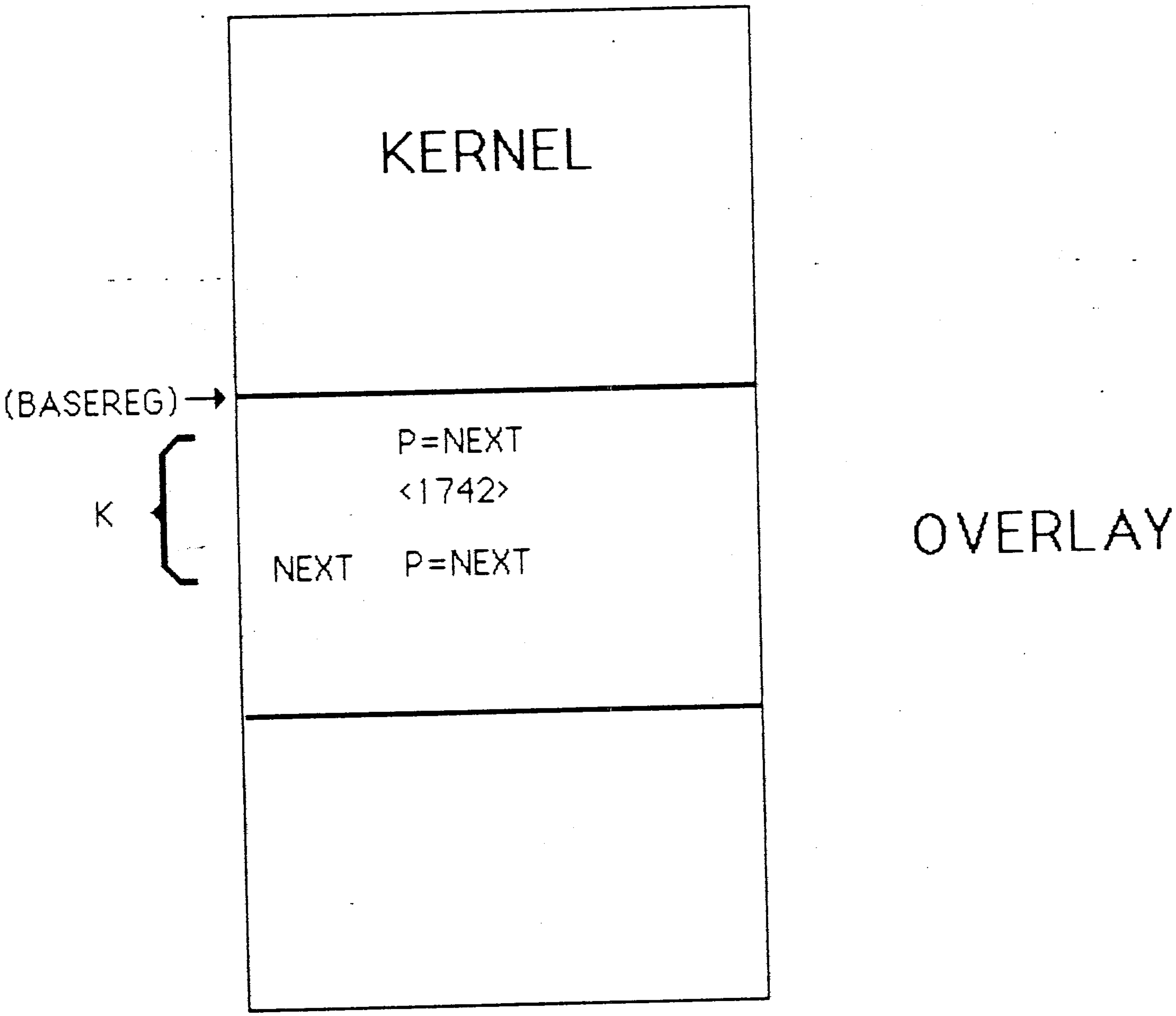
LOCATION	RESULT	OPERAND
IGNORED	BASEREG	R

R Symbol representing Base Register

Example:

```
                                  *
                                           IDENT     BASEREG
                           1      R1       EQUALS    1
                                           BASEREG   R1
      0   075001 /001744                   P=NEXT
      2                                    <1742>
   1744                           NEXT     <1>
                                           END
```

BASEREG



FIELD

LOCATION	RESULT	OPERAND
SYM	FIELD	P,S,W

SYM Field Symbol Name

P Parcel Offset

S Starting Bit (Default 0)

W Width of Field (Default 16)

The following parameters are generated:

- @P Parcel offset from beginning of table
- @S Starting bit of field (software numbered)
- @N Width of field
- @M Mask for field, right justified
- @X Complement of mask in proper position in field

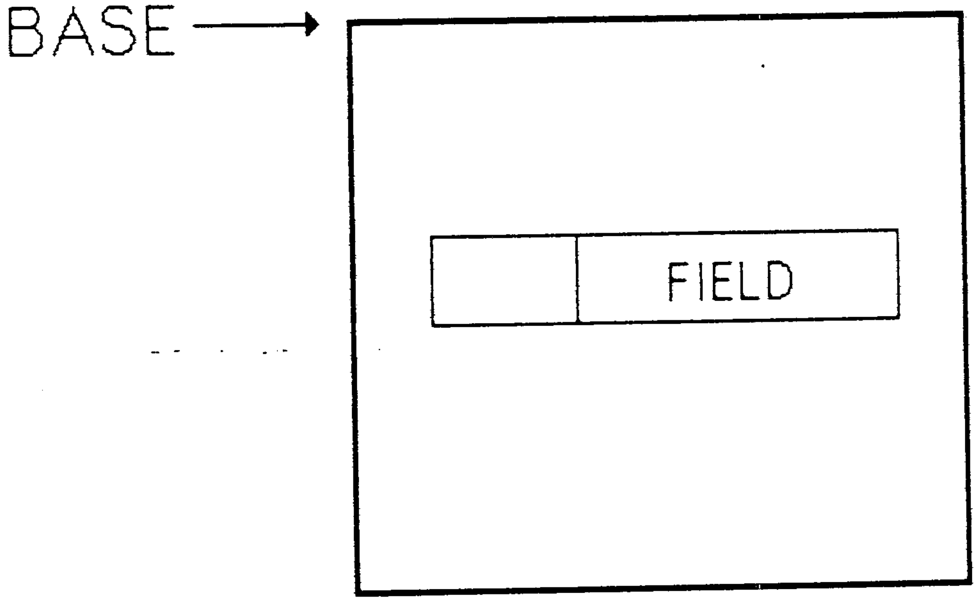
If P=* - @P is undefined

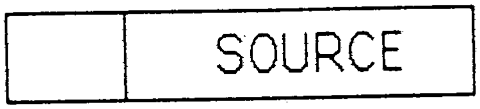
If S=* - @S,-@N,-@M,-@X are undefined

Example:

```
                                   *
                                          IDENT     FIELD
     0    <macro>                  RC@AGE FIELD     0,0,7              .AGE
     0    <macro>                  RC@WGT FIELD     0,7,9              .WEIGHT
     0    <macro>                  RC@SZ1 FIELD     1,,3               .FEET
     0    <macro>                  RC@SZ2 FIELD     1,3,4              .INCHES
     0    <macro>                  RC@YR  FIELD     2,,16              .BIRTH YEAR
                                          END
```

FIELD MACRO





PUT *SOURCE,FIELD,BASE*

DESTINATION

PUT *DEST,FIELD,BASE*

FIELD GETS AND PUTS

LOCATION	RESULT	OPERAND
L	GET	DEST,FIELD,BASE
L	PUT	SOURCE,FIELD,BASE
L	RGET	DEST,FIELD,SOURCE
L	RPUT	SOURCE,FIELD,DEST

L Optional Statement Label

DEST Destination Operand Register or Memory Location containing Data to be stored

FIELD -- Field to be loaded, defined by Field Macro

BASE An Operand Register containing Table Base Address

GET Loads a Field from a Table into an Operand Register or Memory Location

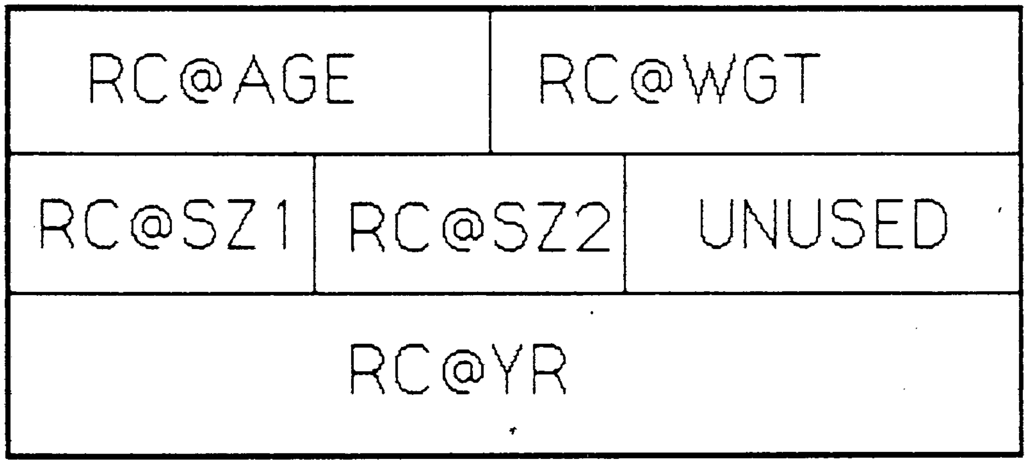
PUT Stores Data in a Field in a Table from an Operand Register or Memory Location

RGET Loads an Operand Register or Memory Location from a Field in an Operand Register or Memory Location

RPUT Loads a Field in an Operand Register or Memory Location from an Operand Register or Memory Location

```
                                         *
                                                   IDENT      FGETPUT
  0    <macro>                           0         REGISTER    (R1,R2,TABLE)
  0    <macro>                           RC@AGE    FIELD        0,0,7
  0    <macro>                           RC@WGT    FIELD        0,7,9
  0    <macro>                           RC@SZ1    FIELD        1,,3
  0    <macro>                           RC@SZ2    FIELD        1,3,4
  0    <macro>                           RC@YR     FIELD        2,,16
                                                   SCRATCH      R1
                                         *
  0   014000 /062340  024002                       R!TABLE=62340
  3   010032  024001                               R2=32
  5    <macro>                                     PUT          R2,RC@AGE,R!TABLE
 15   010245  024001                               R2=245
 17    <macro>                                     PUT          R2,RC@WGT,R!TABLE
 27   010005  024001                               R2=5
 31    <macro>                                     PUT          R2,RC@SZ1,R!TABLE
 44   010007  024001                               R2=7
 46    <macro>                                     PUT          R2,RC@SZ2,R!TABLE
 62    <macro>                                     GET          R2,RC@AGE,R!TABLE
 65   014000 /003676  023001                       R2=3676-R2
      024001
 71    <macro>                                     PUT          R2,RC@YR,R!TABLE
                                                   END
```

RPUT AND RGET

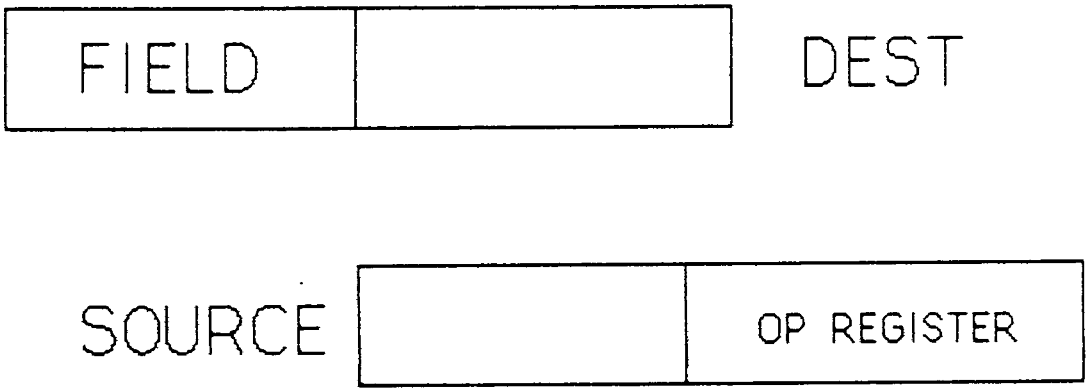


RC@AGE | RC@WGT

RC@SZ1 | RC@SZ2 | UNUSED

RC@YR

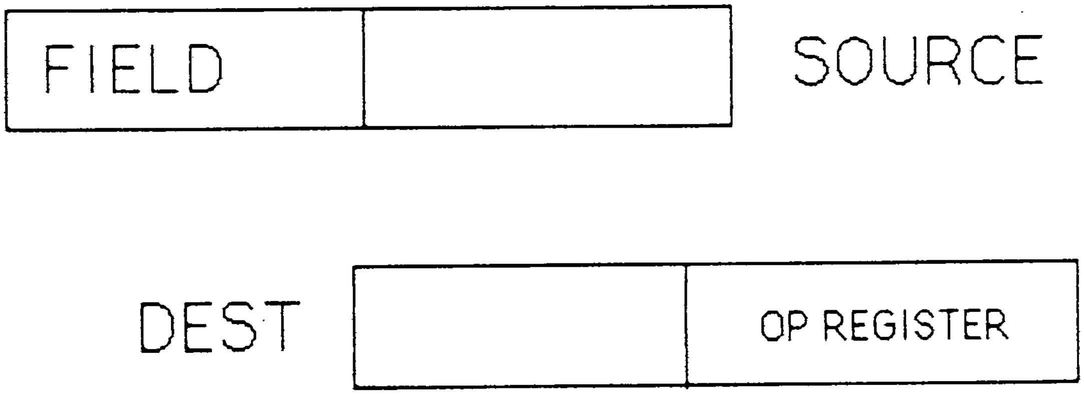
RPUT *SOURCE,FIELD,DEST*



FIELD DEST

SOURCE OP REGISTER

RGET *DEST,FIELD,SOURCE*



FIELD SOURCE

DEST OP REGISTER

KERNEL SERVICE REQUESTS

Control is passed to an overlay via the CALL and GOTO service requests.

CALL results in preserving the caller's SMOD in the software stack.

GOTO passes control directly to new overlay. Callers SMOD is not saved.

An overlay returns control to caller via the RETURN service request.

RETURN results in restoring the caller's SMOD from the software stack

OUTCALL calls an overlay in another IOP.

Parameters may be passed to a called overlay.

IO CALLS

IO is performed by the appropriate service request.

STATIO is a MIOP to BIOP request from/to buffer memory to/from central memory.

HSPR or HSPW moves data from/to Local memory to/from central memory.

MOSR or MOSW moves data from/to local memory to/from buffer memory.

TRANSFER moves data from/to buffer memory to/from central memory.

MSG or MSGR read and write the terminal.

A1300I performs I/O to the NSC Hyperchannel.

D4STIO and D4SEEK drive the DD49's.

TIME QUED ACTIVITY CONTROLS

PUSH and TPUSH puts the activity on a queue until "POP"ed.

POP reactivates "PUSH"ed activity.

TERM terminates and activity.

PAUSE suspends activity for specified time.

ALERT, AWAKE, ASLEEP, RESPOND control activities between IOP's

KERNEL REQUESTS

CODE	NAME	DESCRIPTION
1	PUSH	Put activity on a queue at priority.
2	POP	Remove activity from a queue and place it on CP queue at priority.
3	TERMINATE	Terminate an activity by releasing its' AD and SMOD areas.
4	GIVEUP	Reschedule an active task by priority.
5	D4STIO	Initiates a Read or Write to a DD49.
6	D4SEEK	Initiates a Seek on a DD49.
7	PAUSE	Suspend an activity for tenths of a second.
11	TPUSH	Put activity on a queue and on a timer queue for tenths of a second.
12	GMDAL	Allocates MOS DAL.
13	RMDAL	Releases MOS DAL.
14	ASLEEP	Returns next popcell dal. If none, push activity on popcell.
15	ALERT	Request another IOP to create an activity.
16	AWAKE	Request another IOP to activate an activity.
17	RESPOND	Send response to another IOP.
20	MSG	Send a message to a CRT.
21	MSGR	Send a message to a CRT and wait for response.
22	OUTPUT	Output a message to a CRT (station).
23	STATIO	Initiate I/O between a concentrator and a front end.
25	RECEIVE	Input one character from a console.
26	GDAL	Allocates Local Memory DAL.
27	RDAL	Release Local Memory DAL.
30	GETMEM	Allocate local memory.
31	RELMEM	Release local memory.
32	BGET	Allocate a 512 word (4000 parcel) local buffer.
33	BRET	Release a 512 word local buffer.
34	SEND	Sends message to mainframe.
35	MGET	Allocate a 512 word MOS buffer.
36	MPUT	Release a 512 word MOS buffer.
37	OUTCALL	Calls an overlay in another IOP to execute once.
42	HSPR	Initiates a read on High Speed Channel.
43	HSPW	Initiates a write on High Speed Channel
44	POLL	Send a message to the CPU.
45	TRANSFER	Move data between MOS and central memory.
46	MOSR	Read data from MOS to local memory.
47	MOSW	Write data from local to MOS memory.
50	CALL	Pass control to an overlay with return.
51	GOTO	Pass control to an overlay.
52	RETURN	Return control to an overlay.
53	FIND	Find MOS address and word length of an overlay.
54	FLUSH	Re-initialize overlay memory.
55	CREATE	Set up an independent activity and place it on a CPU queue.

OVERLAY DEFINITION

OVERLAY macro sets up parameters for an overlay.

LOCATION	RESULT	OPERAND
	OVERLAY	OVLNAME, TYPE=

OVLNAME	Name of this overlay
TYPE	If TYPE = DATA is specified then overlay is non-executable.

```
DON                                                           IOP  APML 1.14(12/27/84) 01/02/85 09:58:39  Page 3
                                                                                                            (1)
                                      *
          0     <macro>                          LISTOP
          0     <macro>                          OVERLAY      DON

                                                COMMENT   'Copyright (C) Cray Research, Inc., 1984'            OVERLAY  1
```

Overlay Format

Field	Parcel	Bits	Description
OV@NAM	0-3	0-15	Overlay name (up to eight ASCII characters)
OV@TYP	4	0	Type of overlay: 0 Executable 1 Data
OV@NUM	4	1-15	Overlay number
OV@PAR	5	0-15	Parameter information:
SM@NUM	–	0-6	Number of registers
SM@FST	–	7-15	First operand register
OV@ENT	6	–	Entry point (first executable statement)

```
                                                                     FDUMP 1.13          07/06/84      13:38:37
DMEM,TYPE=IOP1,FWA=0,LWA=177777,FORMAT=PARCEL,R.                     SYSDUMP             05/04/84      16:12:25

000024360    102006  020152  022154  024152    027155  071006  020152  022153      $ (  .     $
000024370    011177  024154  020151  012037    024001  030001  024152  020152     (      ( 0 (
000024400    012003  024001  030001  004011    024152  020152  023154  133064     ( 0    (    &  4
000024410    000026  001000  000000  000000    046504  024424  035434  000000            MD) ;
000024420    026654  024424  020003  016000    046504  026654  024414  000002    - )      MD- )
000024430    024414  027634  006264  026423    044123  050124  042523  000000    ) /    - HSPTES
000024440    000071  000430  010000  024430    024431  024432  024433  024434     9       ) ) ) ) )
000024450    024435  024445  024441  024460    014000  005575  024410  010000    ) )%)])0        )
000024460    034410  020107  102002  070032    010001  024452  020452  013003    9  G          )*]*
000024470    100003  102002  070023  010015    024464  020452  024465  014000               )4]*)5
000024500    000071  024466  010464  054000    014000  034430  076011  020006      9)6 4X    9
000024510    012013  024410  030410  026452    071026  020445  103033  010030        ) 1 -*   ]%
000024520    024464  010074  024465  010000    024466  010000  024467  010000    )4 <)5  )6  )7
000024530    024470  010445  024471  010464    054000  010000  076011  103002    )8 %)9 4X
000024540    070010  010004  024464  010464    054000  014000  034430  076011          )4 4X   9
000024550    071033  020445  024416  010074    024415  020415  102C06  010000       ]%)   <) ]
000024560    034416  027415  026416  071006    020445  012050  024446  020162    9 / -    ]% ()&
000024570    102002  070012  014000  002115    024444  077003  001341  077003            M)$
000024600    001633  075003  001273  020441    103023  010032  024464  010441              ]]      )4 ]
000024610    024465  010464  054000  010000    076011  103002  070010  010004    )5 4X
000024620    024464  010464  054000  014000    034430  076011  071023  020460    )4 4X    9      ]0
000024630    102002  070006  020441  024460    010000  024441  071033  077003          ]])0   )]
000024640    001633  014000  001730  024444    077003  001341  077003  001633              )$
000024650    077003  001633  014000  001744    024444  077003  001341  077003                )$
000024660    001633  014000  001757  024444    077003  001341  077003  001633              )$
000024670    014000  001767  024444  077003    001341  077003  001633  014000            )$
000024700    001774  024444  077003  001341    077003  001633  014000  002001         )$
000024710    024444  077003  001341  077003    001633  077003  001633  014000    )$
000024720    002006  024444  077003  001341    077003  001646  030446  004010      )$        1&
000024730    024423  020423  013061  102002    070010  010001  024430  010001    ) ]  1        )
000024740    024431  010001  024432  070040    020423  013062  102002  070006    )    )     ]  2
000024750    010001  024430  010001  024431    070027  020423  013063  102002      )     )     ]  3
000024760    070004  010001  024430  070020    020423  013064  102002  070004         )    ]  4
000024770    010001  024431  070011  020423    013065  102002  070004  075003      )    ]  5
000025000    001273  070002  071143  014000    002015  024444  077003  001341               )$
000025010    077003  001646  010000  024450    077003  001456  107013  030445           )(    .  1%
000025020    024453  020445  012001  024410    030410  024454  020453  013040    )+]%   )  1  ), ]+
000025030    100003  102002  071027  077003    001456  107032  030445  024455                  .   1%)-
000025040    020445  012001  024410  030410    024456  020455  013040  100003    ]%   )  ]  ).]-
000025050    102002  071046  020453  023455    102002  070006  020454  023456       &]+'-      ],'.
000025060    102002  070002  071057  020455    023453  024423  004020  024417          /]-'+)     )
000025070    020456  023454  024424  104002    027423  004020  021417  106074    ].',)    /    #   <
000025100    077003  001633  014000  002034    024444  077003  001341  077003              )$
000025110    001646  030446  004010  024423    020423  013114  102002  070004      1&   ) ]   L
000025120    010001  024433  070006  020423    013123  103002  070002  071027      )    ]   S
000025130    014000  002051  024444  077003    001341  077003  001646  030446        ))$          1&
000025140    004010  024423  020423  013102    102002  070004  010001  024434      ) ]   B        )
000025150    070006  020423  013127  103002    070002  071025  020432  102027      ]   W         ]
000025160    014000  002076  024444  077003    001341  077003  001646  030446        >)$           1&
000025170    004010  024423  020423  013131    102002  070004  010001  024435      ) ]   Y        )
000025200    070006  020423  013116  103002    070002  071027  020453  024437      ]   N       ]+)
000025210    020454  024440  077003  001703    020455  023437  024423  004020    ],)        ]-'  )
000025220    024417  020456  023440  024424    101002  027423  004020  021417    } ].'  )     /    #
000025230    020423  103006  020424  017000    001000  101002  070005  014000    ]    ]
000025240    001000  024457  070003  020424    024457  020434  103002  070004       )/    ] )/]
000025250    020457  024436  070003  010001    024436  020436  005002  024423    ]/)        ) ]     }
000025260    020457  005002  024424  020424    023423  024442  020457  023436    ]/    ) ]    )"]/
```

REGDEFS

Assigns Operand Registers to Register Symbols.

Allocates Scratch Registers.

Defines Temporary Registers for use by other Macros called within this Porgram Module.

LOCATION	RESULT	OPERAND
L	REGDEFS	GLOBAL,PARS,LOCAL,

L -- Optional Symbol or Constant between 0 and 777 octal specifies origin register

GLOBAL Up to 8 Register Symbols to be assigned to Registers 400_8 to 407_8.

PARS List of Symbols to be assigned to working Operand Registers.

LOCAL List of Symbols to be assigned to Local Registers.

The following registers are also defined:

%S1 to %S5	Scratch Registers (410-414)
%T1 to %T5	Macro Temporary Registers (415-422)
%W1 to %W5	Working Registers available to Overlay (423-427)

Example:

```
                              *
                                          IDENT      REGDEFS
0    <macro>                              REGDEFS    (G1,G2),(R1,R2,R3),(L1,L2)
0   020400                                A=G1
1   020415                                A=R!%T1
2   010006  024431                        R2=6
4   030432  034430                        (R1)=(R3)
6   020427                                A=R!%W5
                                          END
```

```
PETE
Overlay PETE

          6    <macro>                    REGDEFS     ,(AA,S1,S2,R1,R2,N1,N2)
          6   010030  024435              N1=30
         10   010012  024436              N2=12
         12   010027  024431              S1=27
         14   010010  024432              S2=10
         16    <macro>                    CALL        DON,(S1,S2,R0=R1,R0=R2),A1=R1,A2=N2
         40   020433  021434              A=R1&R2
         42    <macro>                    RETURN
                                          END
```

```
DON
Overlay DON

          6    <macro>                    REGDEFS     ,(P1,P2,P3,P4),(T0,T1)
          6   020430  023431  024434      T0=P1-P2
         11   020430  022431  024435      T1=P1+P2
         14    <macro>                    RETREG       T0,P3
         21    <macro>                    RETREG       T1,P4
         26    <macro>                    RETURN
                                          END
```

PARAMETER PASSING

The OVERLAY and REGDEF's macro work together to generate the overlay header.

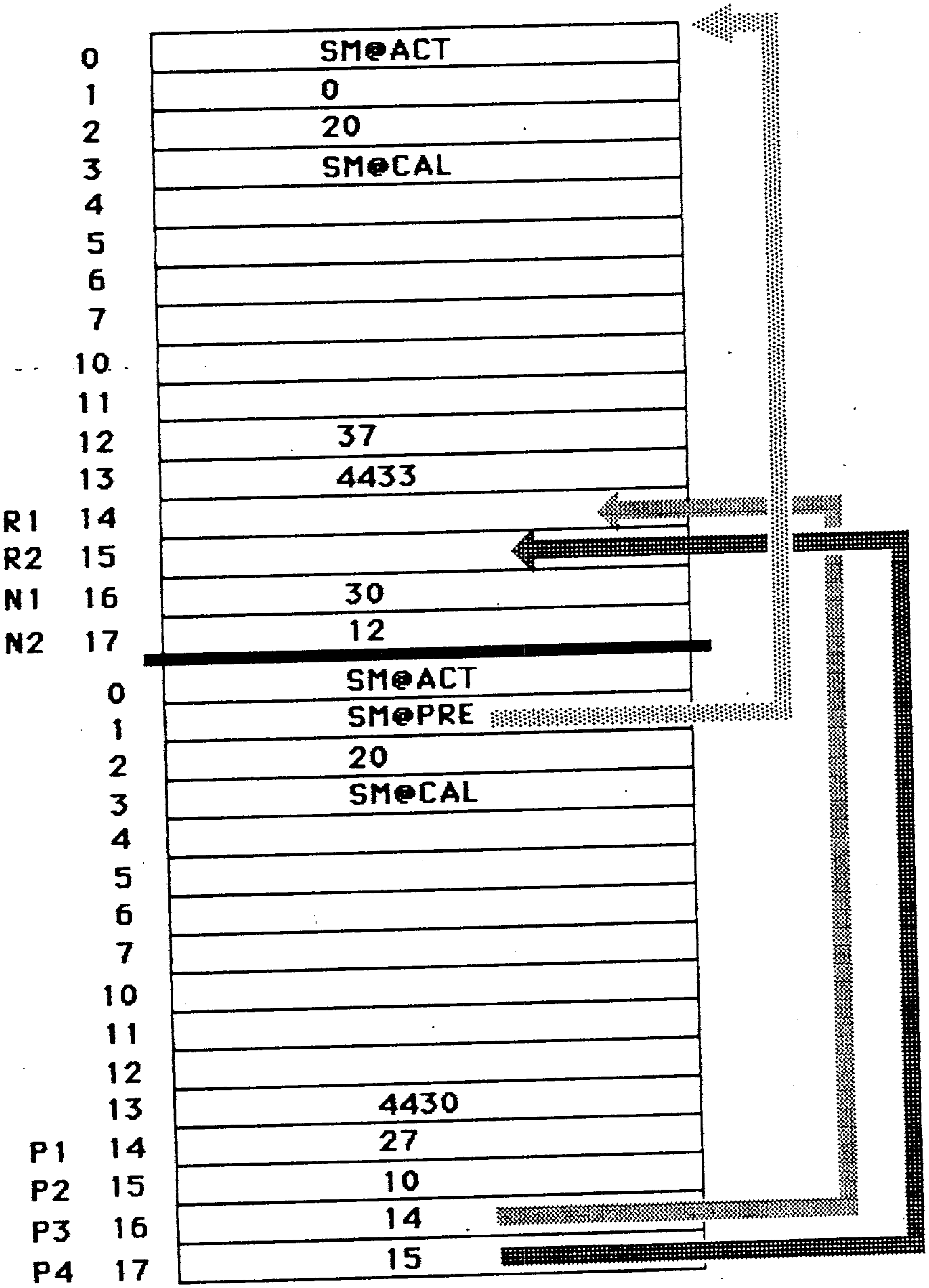
Header contains the starting parameter register number and number of parameter registers.

OVERLAY macro creates overlay header.

REGDEF defines the symbols to different registers.

Global Registers		#400-407
Parameter Registers		#430-437
Local		#400 on
Scratch for APML	%S1 - %S5	#410-414
Scratch Registers for Macros	%T1 - %T6	#415-422
Scratch Registers for Overlay	%W1 - %W5	#423-427

PETE AND DON'S SMOD'S



	0	SM@ACT
	1	0
	2	20
	3	SM@CAL
	4	
	5	
	6	
	7	
	10	
	11	
	12	37
	13	4433
R1	14	
R2	15	
N1	16	30
N2	17	12

	0	SM@ACT
	1	SM@PRE
	2	20
	3	SM@CAL
	4	
	5	
	6	
	7	
	10	
	11	
	12	
	13	4430
P1	14	27
P2	15	10
P3	16	14
P4	17	15

$APTEXT

$APTEXT is the system text input to the assembly of the entire IOS system.

$APTEXT contains

Macro definitions
Table and field dimensions
System symbols, codes and constants

Macros are defined in $APTEXT for sets of widely used functions

Exit stack address
Execution control
Table access
OVERLAY and REGDEFS definition

```
<prototype>                 PUTBYTE    BYTE,OFFSET,BASE,INC=                                AT.1789
                            LOCAL      XXX,YYY                                              AT.1790
<definition>                R!%T1 = BYTE&O'377                                              AT.1791
<definition>                R!%T2 = OFFSET>1+BASE                                           AT.1792
<definition>                P = XXX, 0 # OFFSET&1                                           AT.1793
<definition>                (R!%T2) = (R!%T2)<D'9+R!%T1>>D'9                                AT.1794
<definition>                P = YYY                                                         AT.1795
<definition>       XXX      *                                                               AT.1796
<definition>                (R!%T2) = (R!%T2)&O'177400+R!%T1                                AT.1797
<definition>       YYY      *                                                               AT.1798
<definition>                IFC        |_INC_|,NE,,1                                        AT.1799
<definition>                OFFSET = OFFSET+INC                                             AT.1800
                   PUTBYTE  ENDM                                                            AT.1801


                   ********************************************************************    AT.1803
                   *                                                                  *    AT.1804
                   *        REGDEFS  Define overlay registers                         *    AT.1805
                   *                                                                  *    AT.1806
                   * start  REGDEFS  global,pars,local,temp                           *    AT.1807
                   *                                                                  *    AT.1808
                   *        start    Optional; specifies starting register number.   *    AT.1809
                   *                 Default is %GBLREG.                              *    AT.1810
                   *        global   List of global registers.                        *    AT.1811
                   *        pars     List of parameter registers                      *    AT.1812
                   *        local    List of registers used locally                   *    AT.1813
                   *        temp     List of temporary registers                      *    AT.1814
                   *                                                                  *    AT.1815
                   ********************************************************************    AT.1816
                                                                                           AT.1817
                            MACRO                                                          AT.1818
<prototype>        START    REGDEFS  GLOBAL,PARS,LOCAL,TEMP                                AT.1819
                            LOCAL    $$S                                                   AT.1820
<definition>       $$S      SET      %GBLREG                                               AT.1821
<definition>                IFC      |_START_|,NE,,1                                       AT.1822
<definition>       $$S      SET      START                                                 AT.1823
```

```
<definition>       $$S      REGISTER (GLOBAL)                                              AT.1824
<definition>                IFE      $REGORG,GT,$$S+%GBLNUM,1                              AT.1825
<definition>                ERROR    .Too many global registers defined                    AT.1826
<definition>       $$S      SET      $$S+%GBLNUM                                           AT.1827
<definition>                SCRATCH  %S1,%S2,%S3,%S4,%S5                                   AT.1828
<definition>       $$S      REGISTER (%S1,%S2,%S3,%S4,%S5)                                 AT.1829
<definition>                REGISTER (%T1,%T2,%T3,%T4,%T5,%T6)                             AT.1830
<definition>                REGISTER (%W1,%W2,%W3,%W4,%W5)                                 AT.1831
<definition>       %P       EQUALS   $REGORG                                               AT.1832
<definition>                REGISTER (PARS)                                                AT.1833
<definition>       %NP      EQUALS   $REGORG-%P                                            AT.1834
<definition>                REGISTER (LOCAL)                                               AT.1835
<definition>                REGISTER (TEMP)                                                AT.1836
                   REGDEFS  ENDM                                                           AT.1837


                            MACRO                                                          AT.1839
<prototype>        REGORG   REGISTER REGLIST                                               AT.1840
                            LOCAL    $MSIZE                                                AT.1841
<definition>       $REG     IFC      |_REGORG_|,NE,''                                      AT.1842
<definition>       $REGORG  SET      REGORG                                                AT.1843
<definition>       $REG     ELSE                                                           AT.1844
<definition>                IFA      #DEF,$REGORG,1                                        AT.1845
<definition>                ERROR    .Register origin must be specified                    AT.1846
<definition>       $REG     ENDIF                                                          AT.1847
<definition>       $REG     ECHO     REG=(REGLIST)                                         AT.1848
<definition>                IFC      |_REG_|,NE,'',6                                       AT.1849
<definition>                IFC      |_REG_|,NE,'*',4                                      AT.1850
<definition>       $MSIZE   MICRO    'REG        ',D'8                                     AT.1851
<definition>       REGS     LIST     MAC                                                   AT.1852
<definition>       "$MSIZE" EQUALS   $REGORG                                               AT.1853
<definition>       REGS     LIST     *                                                     AT.1854
<definition>       $REGORG  SET      $REGORG+1                                             AT.1855
<definition>       $REG     ENDDUP                                                         AT.1856
                   REGISTER ENDM                                                           AT.1857
```

BIND

Resolves external symbol references among APML modules in a binary library.

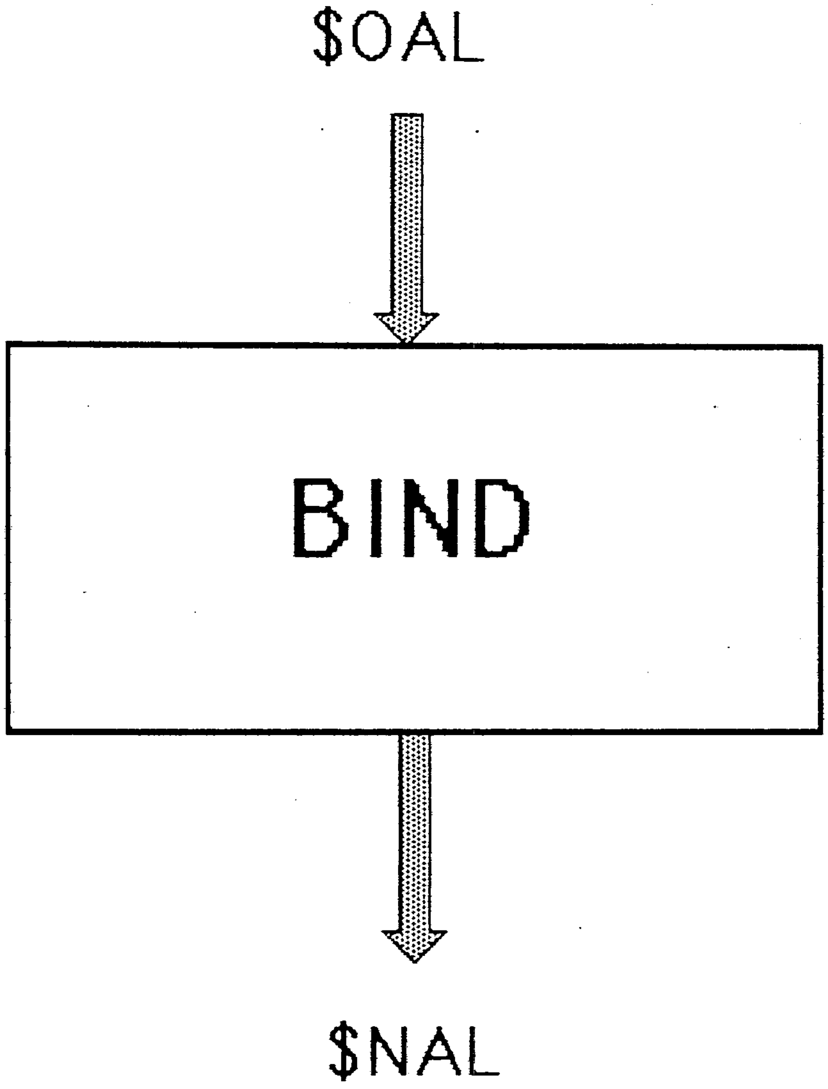
Bind is simular to the COS LDR but, unlike LDR, bind does not perform code relocation.

Bind is used during IOS system generation.

```
Bind,OAL=oaldn,NAL=naldn,L=ldn,Debug=,NA
```

Example:

```
JOB
ACCOUNT,
APML.
ACCESS,DN=IOSLIB,ID=V114.
BUILD,OBL=IOSLIB,I=0,REPLACE,NBL=$OAL.
BIND.
ADSTAPE,I=$NAL.
```



$OAL

BIND

$NAL

ADSTAPE

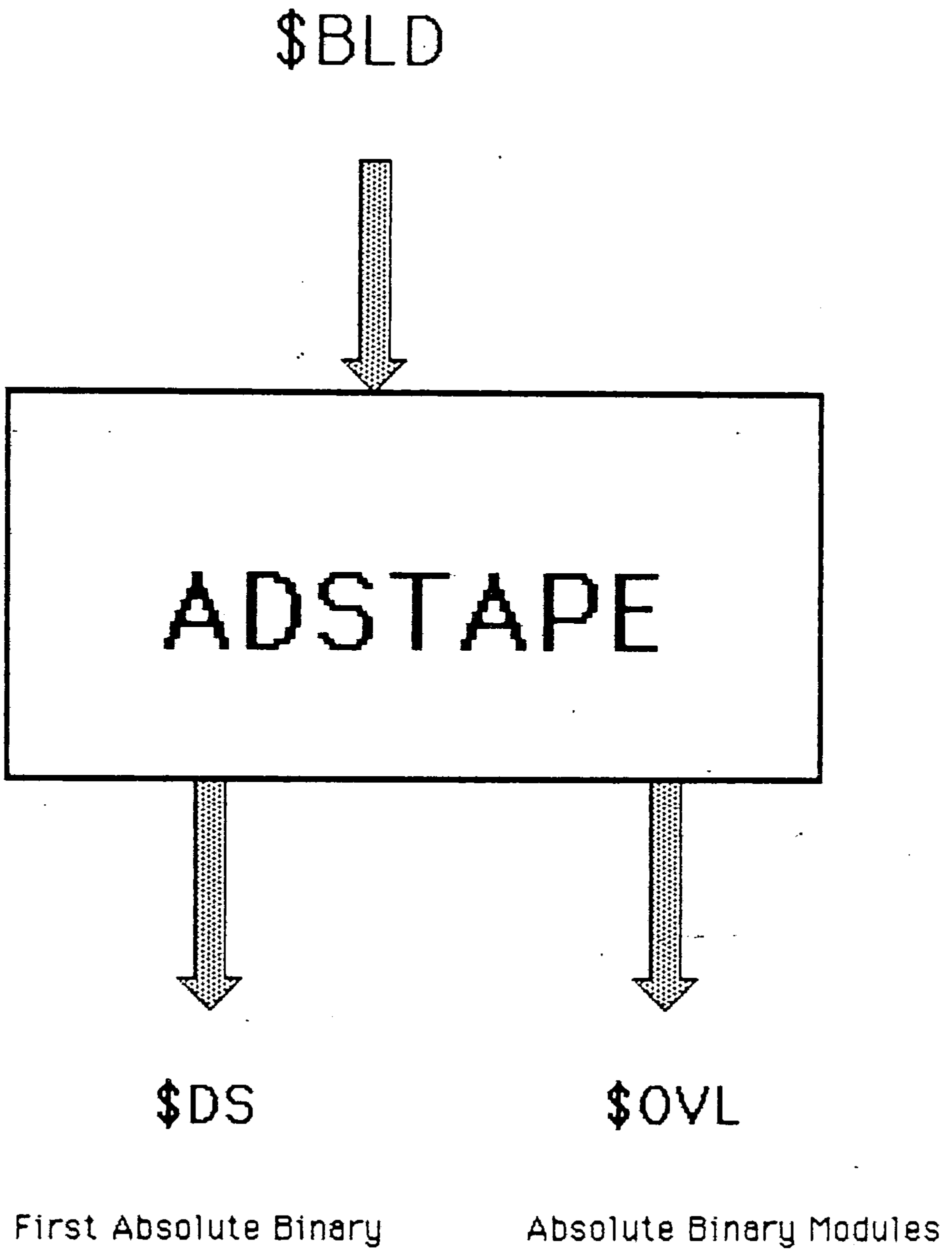
Builds deadstart datasets from absolute binary load modules generated by APML.

Generates unblocked datasets $DS and SOUL.

Control word precedes each absolute binary load module on $OUL.

```
ADSTAPE,I=idn,O=odn,OUL=ouldn
```

Example:


$BLD
ADSTAPE
$DS
$OVL
First Absolute Binary
Absolute Binary Modules

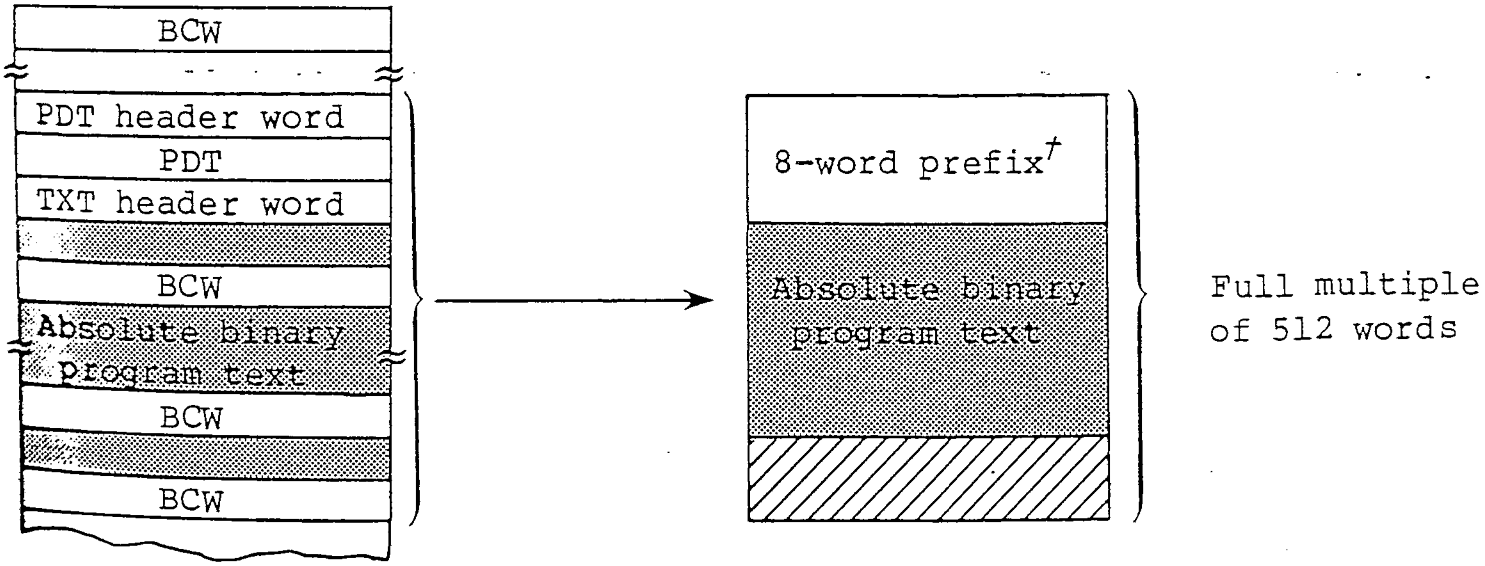
UNB

Converts an absolute binary load module from a COS blocked dataset. Unblocked format is required for COS startup and offline diagnostics.

```
UNB,I=idn,O=odn
```

Example:

```
JOB,JN=
ACCOUNT,AC=
APML.
UNB.
/EOF
```


BCW
PDT header word
PDT
TXT header word
BCW
Absolute binary program text
BCW
BCW
8-word prefix†
Absolute binary program text
Full multiple of 512 words

APML QUIZ

1. What does R! in front of a symbol denote ?

2. What dataset defines the APML macros ?

3. What language is APML written in and where does it execute ?

4. Is the LDR statement used with APML ?

5. What macro would you use to define registers ?

6. Why is the accumulator dangerous to use in a complex APML statement ?

7. What are kernel service requests ?

8. What symbol suffix's does the TABLE and FIELD macro generate and what is their meaning ?

9. What are the scratch registers defined with the REGDEF macro ?

10. How is the BASEREG used in a program ?

APML QUIZ

1. What does R in front of a symbol denote ?

2. What dataset defines the APML macros ?

3. What language is APML written in and where does it execute ?

4. Is the LDR statement used with APML ?

5. What macro would you use to define registers ?

6. Why is the accumulator dangerous to use in a complex APML statement ?

7. What are kernel service requests ?

8. What symbol suffix's does the TABLE and FIELD macro generate and what is their meaning ?

9. What are the scratch registers defined with the REGDEF macro ?

10. How is the BASEREG used in a program ?

PROGRAMMING EXERCISES

7

Exercise 1 Diagnostic Installation and Generation

Skill: Generate an Offline Diagnostics Tape

Tasks:

a. Write a batch job to install the diagnostic release tape to mass storage.

b. Write a batch job to assemble and generate the diagnostic release.

c. Write a batch job to make an fdump tape.

d. Write a batch job to get the listing for SFA.

Resources:

Front End Editor
SWCE II Workbook
Diagnostic Release Tapes
Diagnostic Release Letter

Tools: GENPL ECD BLD

Related Reading:

Diagnostic Release Letter
SWCE Workbook

Intended Lesson Results: To be able to take a diagnostic release and generate the necessary binaries and listings online and then generate an fdump tape to be loaded to 80MB DSS disk pack.

Exercise 1 Diagnostic Installation and Generation

Skill: Generate an Offline Diagnostics Tape

Tasks:

a. Write a batch job to install the diagnostic release tape to mass storage.

b. Write a batch job to assemble and generate the diagnostic release.

c. Write a batch job to make an fdump tape

d. Write a batch job to get the listing for SFA.

Resources:

Front End Editor
SWCE II Workbook
Diagnostic Release Tapes
Diagnostic Release Letter

Tools: GENPL ECD BLD

Related Reading:

Diagnostic Release Letter
SWCE Workbook

Intended Lesson Results: To be able to take a diagnostic release and generate the necessary binaries and listings online and then generate an fdump tape to be loaded to 80MB DSS disk pack

Exercise 2 Operational Aids and Utilities

Skill: Use Cray Operational Aids and Utilities

Tasks:

a. Write a batch job to archive all datasets in the TNG ownership with an ID=TNG___, to the IBM tape drive using PDSDUMP.

b. Write a batch job to restore the datasets that were archived by the previous job.

c. Write a batch job that runs EXTRACT to search the $SYSTEMLOG about Disk errors in the past week

d. Write a batch job that runs EXTRACT to search the $SYSTEMLOG for any memory errors in the past week.

e. Write a batch job that runs EXTRACT to search the $SYSTEMLOG for the $LOG messages from your last job.

f. Write a batch job that runs HERG and get error information on disk, tape and memory.

g. Write an FDUMP job that dumps the datasets CRAY1SYSTEMDUMP with an ID=TNGSWCE and ED=______ and ED=_______ One is a COS hang and the other is an IOS hang.

Resources:

SM-0044	Sections 4 and 5
SWCE II Workbook	

Intended Lesson Result: To be able to run jobs that maintain permanent datasets on disk or jobs that gather statistics from the $SYSTEMLOG to evaluate the system performance online.

Exercise 3 APML Assembly

Skill: Use IOP instructions
Read and write APML code
Use system Macros

Tasks:

a. Write and assemble without errors, a program which loads and then adds two operand registers, storing the results in a local memory location.

b. Write and assemble without errors, a program which sums up the valid exit stack entries (0-9) and stores the result in a local memory location.

c. Write and assemble without errors, a program which disables interrupts, stores the exit stack in local memory, sums up the interrupting channel numbers in an operand register, and restores the exit stack when all interrupts have been "handled" in this way.

Resources:

APML Reference	SM-0036
IOS Internals	SM-0046
IOP Hardware Ref	HR-0030
$APTEXT	(optional)

Related Reading:

SM-0046 pages 10-4 to 10-24

Intended Lesson Results: Read IOS diagnostic code such as MOSTEST or HSPTEST and be able to follow the IOS macro's and Kernel service requests

Exercise 3 APML Assembly

Skill: Use IOP instructions
Read and write APML code
Use system Macros

Tasks:

a. Write and assemble without errors, a program which loads and then adds two operand registers, storing the results in a local memory location.

b. Write and assemble without errors, a program which sums up the valid exit stack entries (0-9) and stores the result in a local memory location.

c. Write and assemble without errors, a program which disables interrupts, stores the exit stack in local memory, sums up the interrupting channel numbers in an operand register, and restores the exit stack when all interrupts have been "handled" in this way.

Resources:

APML Reference	SM-0036
IOS Internals	SM-0046
IOP Hardware Ref	HR-0030
$APTEXT	(optional)

Related Reading:

SM-0046 pages 10-4 to 10-24

Intended Lesson Results: Read IOS diagnostic code such as MOSTEST or HSPTEST and be able to follow the IOS macro's and Kernel service requests

Exercise 4 TNG Overlay Integration

Skills: Use Kernel Service Requests
Use $APTEXT macro's

Tasks: Write the Following three overlays

TNG1

- Display a message on the Kernel console
  (Similar to TNG1 in the class presentation)
- CALL TNG2
- RETURN

TNG2

- Allocate some local memory for a message
- Input the message from the console
- Allocate some buffer memory for the message
- Write the message to buffer memory
- OUTCALL TNG3 in all the processors except the IOP TNG1 & TNG2 are executing in (pass the buffer memory addressto TNG3 as a parameter)
- Deallocate any local and buffer memory you have used
- RETURN

TNG3

- Allocate some local memory for the message
- Read in the message from buffer memory
- Display the message on the console
- Deallocate any memory you have used
- RETURN

Resources: SM-0046 Chapter 2 and 10

Intended Lesson Results: To be able to add and integrate an activity from the kernel console such as in a diagnostic and be able to follow IOS macros and kernel service requests

Exercise 4   TNG Overlay Integration

Skills: Use Kernel Service Requests
Use $APTEXT macro's

Tasks: Write the following three overlays

TNG1

- Display a message on the Kernel console (Similar to TNG1 in the class presentation)
- CALL TNG2
- RETURN

TNG2

- Allocate some local memory for a message
- Input the message from the console
- Allocate some buffer memory for the message
- Write the message to buffer memory
- OUTCALL TNG3 in all the processors except the IOP TNG1 & TNG2 are executing in (pass the buffer memory addressto TNG3 as a parameter)
- Deallocate any local and buffer memory you have used
- RETURN

TNG3

- Allocate some local memory for the message
- Read in the message from buffer memory
- Display the message on the console
- Deallocate any memory you have used
- RETURN

Resources: SM-0046 Chapter 2 and 10

Intended Lesson Results: To be able to add and integrate an activity from the kernel console such as in a diagnostic and be able to follow IOS macros and kernel service requests

LAB EXERCISES

8

Kernel and Station Commands

With the aid of all furnished reference materials complete the following tasks:

1. Start IOS from the binaries on Disk

2. View the deadstart parameter file through the IOS editor and verify that you are RESTARTing COS

3. Initialize the IOP station

4. Go through the startup process until you have the STARTUP COMPLETE visable with the Y. command.

5. Shutdown the concentrators and network channels.

6. Initialize an Interactive station.

7. Using TEDI write the needed JCL and CAL that loops like a pass counter adding S1+1 and submit it as a job. Set the T=20 on the JOB statement.

8. Change it's priority, time limit and station ID

9. Turn off it's job class and the turn it back on.

10. Change the limit of the number of jobs to 1.

11. Display the LOOP jobs last $LOG

12. Change all ** ID's to AP

13. Suspend the job.

14. Rerun the job.

15. How much is the system using the CPU over the user

16. How much STP activity is going on

17. Display how the disk drives are configured

18. Display disk drive activity and error information.

19. How many front end stations are logged on

Kernel and Station Commands

With the aid of all furnished reference materials complete the following tasks:

1. Start IOS from the binaries on Disk

2. View the deadstart parameter file through the IOS editor and verify that you are RESTARTing COS

3. Initialize the IOP station

4. Go through the startup process until you have the STARTUP COMPLETE visable with the Y. command.

5. Shutdown the concentrators and network channels.

6. Initialize an interactive station.

7. Using TEDI write the needed JCL and CAL that loops like a pass counter adding S1+1 and submit it as a job. Set the T=20 on the JOB statement

8. Change it's priority, time limit and station ID

9. Turn off it's job class and the turn it back on.

10. Change the limit of the number of jobs to 1.

11. Display the LOOP jobs last $LOG

12. Change all ** ID's to AP

13. Suspend the job.

14. Rerun the job.

15. How much is the system using the CPU over the user

16. How much STP activity is going on

17. Display how the disk drives are configured

18. Display disk drive activity and error information.

19. How many front and stations are logged on

COS Release Installation

With the aid of all furnished reference materials and the COS release letters and tapes, complete the following.

1. Start IOS from tape.

2. Start COS from tape. View the deadstart parameter file to:

   a. Verify that A1-23 is the master device
   b. All other disk drives are offline
   c. All other disk drives are configured down
   d. You are installing on the scratch drive only

3. Submit the BIN114 job to load the Product Set binaries

4. Submit the PL114 job to load the Program libraries

5. Run JCSDEF, PRVDEF, and ACCTDEF to setup the user enviroment

Diagnostics

With the aid of all furnished reference materials, and the Online diagnostics release letter and tape, complete the following.

1. Run the batch job you have prepared to install the release tape

2. Run the batch job to generate the diagnostics

3. Run the batch job to write the fdump tape

4. Run the batch job to print the listing for SFA

5. Using an interactive IOP station, Access and execute MENU

6. Analyize a harware failure introduced into the system

7. Archive the datasets on a disk drive and configure the dive down so that it may be serviced and then restore the datasets

8. Restore the drive to the system.

9. Run HSPTEST

10. Run MOSTEST

11. Run DOM and F80M to format an 80mb disk pack

12. Go offline to DSS and verify the FDUMP worked and the diagnostics on it work.

Diagnostics

With the aid of all furnished reference materials, and the Online diagnostics release letter and tape, complete the following

1. Run the batch job you have prepared to install the release tape

2. Run the batch job to generate the diagnostics

3. Run the batch job to write the fdump tape

4. Run the batch job to print the listing for SFA

5. Using an interactive IOP station, Access and execute MENU

6. Analyize a harware failure introduced into the system.

7. Archive the datasets on a disk drive and configure the dive down so that it may be serviced and then restore the datasets

8. Restore the drive to the system.

9. Run HSPTEST

10. Run MOSTEST

11. Run DOM and FBOM to format an 80mb disk pack

12. Go offline to DSS and verify the FDUMP worked and the diagnostics on it work

COS and IOS Debug Utilities

With the aid of all furnished reference materials, complete the following tasks:

1. Hit CNTL-D

2. Startup IOS

3. Startup COS
    a. Modify the deadstart parameter file to breakpoint in STARTUP at address ______________ , which is at the message prompt beeps

4. Initialize the IOP station

5. Observe that the breakpoint suspended the system
    a. Remove the breakpoint

6. Use COS debug to read any location in central memory in the necessary format, use DISPLAY to change the defaults

7. Use COS debug to write into these locations

    a. EXEC's XFT for events
    b. STP table address 15
    c. A Job's A and S registers

8. Use IOS debug to read the following
    a. IOP1's Local memory address ___________.
    b. IOP1's Operand registers
    c. IOP1's main registers
    d. Buffer memory address ___________.

9. Use IOS debug to write into the following
    a. IOP0's local memory address _________
    b. IOP3's Operand register 500
    c. IOP3's acculmulator
    d. Buffer memory address __________

10. Let the instructor crash the system

11. Restart IOS and COS processing the dump just created

12. PDSDUMP the CRAY1SYSTEMDUMP to tape.

COS and IOS Debug Utilities

With the aid of all furnished reference materials, complete the following tasks:

1. Hit CNTL-D

2. Startup IOS

3. Startup COS
   a. Modify the deadstart parameter file to breakpoint in STARTUP at address ____________ , which is at the message prompt beeps

4. Initialise the IOP station

5. Observe that the breakpoint suspended the system
   a. Remove the breakpoint

6. Use COS debug to read any location in central memory in the necessary format, use DISPLAY to change the defaults

7. Use COS debug to write into these locations

   a. EXEC's XFT for events
   b. STP table address 15
   c. A Job's A and S registers

8. Use IOS debug to read the following
   a. IOP1's Local memory address ____________
   b. IOP1's Operand registers
   c. IOP1's main registers
   d. Buffer memory address ____________

9. Use IOS debug to write into the following
   a. IOP0's local memory address ____________
   b. IOP3's Operand register 500
   c. IOP3's accumulator
   d. Buffer memory address ____________

10. Let the instructor crash the system

11. Restart IOS and COS processing the dump just created

12. POSDUMP the CRAY-1SYSTEMDUMP to tape.

QUIZ 2

ANSWERS

OPERATIONS QUIZ

1. What station command displays the jobs COS is handling ?

    STATus

2. What kernel command starts the IOP station display ?

    STATION

3. What do you type in and where when you here the console beeping ?

    STMSG at a station console

4. What command will change a jobs priority, time limit job class or ID ?

    ENTer *jsq*

5. What two commands will shutdown the front end station and hyperchannels ?

    ENDCONC NSCEND

6. What is the STATC command used for ?

    To see the Job Class structure and which are on and how many jobs are active or executing

7. Who uses the deadstart parameter file and what does it do ?

    COS STARTUP Task to control or modify COS on initialization

8. What are the four COS startup's and what is their key difference ?
    INSTALL Rewrites the DSC and starts system with no datasets
    DEADSTART Clears all rolled and spooled datasets from disk
    RESTART Reruns jobs from beginning again
    WARMSTART Normally done, to continue jobs from where they were

9. DEBUG typed in at an IOP station does what ?

    Invokes the IOP Debugger - not to be done with system up

10. If *SDR is in the deadstart parameter file what happens and what is neccessary to use the system verbs again

    The system directory is cleared out and most JCL statements are not recognized by COS so you have to run a job SDR114 to ACCESS all verb datasets with the ENTER

OFFLINE DIAGNOSTIC GENERATION QUIZ

1. What is the input dataset to BLD ?

   FLIST and the program libraries (default)

2. What determines the diagnostics you want on the DSS tape for ECD ?

   BLIST generated by BLD (default)

3. What language is BLD and ECD written in ?

   FORTRAN

4. What is the output of BLD ?

   A dataset named JCL wich is the JCL stream for FLIST assembly

5. Name two reasons for using the program ECD ?

   To write an FDMP tape
   To print some listings

6. What language is the GENPL written in ?

   COS Job COntrol Language in PROC'S

7. What would you use to change FLIST ?

   TEDI is one good choice

8. What would you use to get a listing of GENPL ?

   UPDATE,P=GENPL,ID,N=Ø.

9. What would you use to modify a diagnostic ?

   UPDATE and it's directives contained in modifications (MODS)

10. How do you install the diagnostic program libraries ?

    FETCH or ACQUIRE from the IOP Tape drive
    FETCH,DN=LOAD,AC=MT,TEXT=MTØ:1.
    SUBMIT,DN=LOAD.

ONLINE DIAGNOSTICS QUIZ

1. What is the program MENU used for ?

    EASY LOADING OF ONLINE DIAGNOSTICS

2. What language is MENU written in ?

    FORTRAN

3. If a diagnostic fails what are two options for action ?

    GO OFFLINE AND RUN DIAGNOSTICS
    ISOLATE ONLINE WITH COS DEBUGGER OR MEMORY DUMP

4. What JCL statement is in the diagnostic $CS for memory dump ?

    DUMP -THIS MAY NEED TO BE CHANGED TO GET MORE OF THE DIAGNOSTIC

5. How could you change the addresses and length of diagnostic dump?

    Edit the job before submitting with TEDI and modify the DUMP statement

6. What does LADDER test ?

    ONLINE Tape Test

7. Where do you look to find if a diagnostic failed and which ones ?

    The job status Queue STAT
    ID field will contain **FAILED**

IOS SYSTEM DIAGNOSTICS QUIZ

1. What is the difference between COS online or IOS system diagnostics ?

ONLINE ARE NOT TO BE RUN WITH JOBS EXECUTING
IOS DIAGNOSTICS WRITE TO SYTEM MEMORY LOCATIONS

2. What are the features of F80M ?

TEST THE AMPEX 80MB DISK DRIVE
SURFACE ANALYSIS; FORMATTING, TESTING LOOPS

3. What monitor is used for MIOP system diagnostics ?

DOM DIAGNOSTIC ONLINE MONITOR

4. What command at what console is needed to bring up this monitor ?

DOM AT A KERNEL CONSOLE

5. What does HSPTEST check and how would you determine it's failure ?

HIGH SPEEDS ATTACHED TO THE IOP
TO ANALIZE YOU NEED TO DUMP WITH FILE 2 $DUMP

6. What does MOSTEST check and how would you determine it's failure ?

BUFFER MEMORY
TO ANALIZE YOU NEED TO DUMP WITH FILE 2 $DUMP

7. Where would you find listings for IOS system diagnostics ?

IOS SYSTEM KERNEL OVERLAY PROGRAMS

8. What action would be best to take in a IOS system diagnostic failure ?

GO OFFLINE AND TEST THE APROPRIATE IOP AND CHANNEL TYPE

9. Should IOS system diagnostics be run during normal system operation ?

NO ABSOLUTLY NOT AS THEY WRITE SYSTEM MEMORY LOCATIONS
OR CAN LOCK OUT NORMAL SYSTEM FUNCTIONS

10. What language are IOS system diagnostics written in ?

APML

OPERATIONAL AIDS AND UTILITIES QUIZ

1. Explain what EXTRACT does and with what dataset(s) ?

   SEARCHES THE SYSTEM LOG FOR SPECIFIED ENTRY TYPES
   $SYSLOG $SYSTEMLOG

2. What other names has HERG had in the past releases ?

   BREAKER OR SORT

3. Explain what HERG does and with what dataset(s) ?

   SORTS THROUGH THE SYSTEM LOG FOR HARDWARE ERROR ENTRIES

4. What Cray publication is EXTRACT and FDUMP described in ?

   SM-44 OPERATIONAL AIDS AND UTILITIES REFERENCE

5. How would you use the information obtained from EXTRACT or HERG ?

   IDENTIFY MEMORY BANK CHIP FAILURES
   IDENTIFY DISK AND TAPE DRIVE FAULTS AND ERRORS

6. What does FDUMP used for and what does it do ?

   SYSTEMDUMP FORMATTING TO CONVERT A CRAY BINARY DUMP
   TO A FORMATTED ASCII OCTAL MEMORY DUMP

7. What dataset does FDUMP process ?

   $CRAY1SYSTEMLOG

8. What are FDUMP's directives used for ?

   TO CONTROL WHICH DATASET TO DUMP AND WHAT ADDRESS RANGES

9. Where is the typical place to put FDUMP directives ?

   $IN IS THE EASIEST

10. What does FDUMP need for automatic formatted dumps ?

    THE CORRECT SYMBOL TABLES - EXECSYM AND STPSYM

APML QUIZ

1. What does R! in front of a symbol denote ?

AN OPERAND REGISTER

2. What dataset defines the APML macros ?

$APTEXT

3. What language is APML written in and where does it execute ?

CAL AND EXECUTES IN THE USER JOB AREA IN CENTRAL MEMORY

4. Is-the LDR statement used with APML ?

NO BIND AND ADSTAPE REPLACE IT

5. What macro would you use to define registers ?

REGDEFS (ZZ),(AA),(BB) OR 100 REGISTER (AA)

6. Why is the accumulator dangerous to use in a complex APML statement ?

VERY DYNAMIC AND COMPLEX STATEMENTS WILL USE IT
EVERYBODY USES THE ACCULMULATOR

7. What are kernel service requests ?

SPECIAL SYSTEM FUNCTIONS PERFORMED BY THE MONITOR (KERNEL)

8. What symbol suffix's does the TABLE and FIELD macro generate and what is their meaning ?

@LH	HEADER LENGTH	@LE	ENTRY LENGTH
@NE	NUMBER OF ENTRIES	@SZ	SIZE OF TABLE
@P	PARCEL OFFSET	@S	STARTING BIT NUMBER
@N	BIT WIDTH	@M	MASK RIGHT JUSTIFIED

9. What are the scratch registers defined with the REGDEF macro ?

%S1 TO %S5	TEMPORARY FOR COMPLEX APML
%T1 TO %T6	SCRATCH FOR MACRO'S
%W1 TO %W5	SCRATCH REGISTERS FOR OVERLAY

10. How is the BASEREG used in a program ?

TO DECLARE THE OPERAND REGISTER THAT THE OVERLAY IS

EXERCISE 2

ANSWERS

Exercise 1 Diagnostic Installation and Generation

Skill: Generate an Offline Diagnostics Tape

Tasks:

a. Write a batch job to install the diagnostic release tape to mass storage.

```
JOB,JN=TNG__A.
ACCOUNT,AC=265124,US=TNG,UPW=TNG.
FETCH,DN=$PROC,MF=AP,AC=MT,TEXT=DSD:Ø,WAIT.
INSTALL.
```

b. Write a batch job to assemble and generate the diagnostic release.

```
JOB,JN=TNG__A.
ACCOUNT,AC=265124,US=TNG,UPW=TNG.
ACCESS,DN=$PROC,PDN=PROCLIB,ID=DIAGSYS.
SETUP.
GEN.
```

c. Write a batch job to make an fdump tape.

```
JOB,JN=TNG__A.
ACCOUNT,AC=265124,US=TNG,UPW=TNG.
ACCESS,DN=$PROC,PDN=PROCLIB,ID=DIAGSYS.
SETUP.
TAPE.
```

d. Write a batch job to get the listing for SFA.

```
JOB,JN=TNG__A.
ACCOUNT,AC=265124,US=TNG,UPW=TNG.
ACCESS,DN=$PROC,PDN=PROCLIB,ID=DIAGSYS.
SETUP.
GSFA.
LISTING.
```

Intended Lesson Results: To be able to take a diagnostic release and generate the necessary binaries and listings online and then

Exercise 2 Operational Aids and Utilities

Skill: Use Cray Operational Aids and Utilities

Tasks:

a. Write a batch job to archive all datasets in the TNG ownership with an ID=TNG___, to the IBM tape drive using PDSDUMP.

```
JOB,JN=TNG__A.
ACCOUNT,AC=265124,US=TNG,UPW=TNG.
ACCESS,DN=$PDS,DT=*6250,VOL=ARCHIVE,NEW.
PDSDUMP,ID=TNG___
```

b. Write a batch job to restore the datasets that were archived by the previous job.

```
JOB,JN=TNG__A.
ACCOUNT,AC=265124,US=TNG,UPW=TNG.
ACCESS,DN=$PDS,DT=*6250,VOL=ARCHIVE.
PDSLOAD,ID=TNG___
```

c. Write a batch job that runs EXTRACT to search the $SYSTEMLOG about Disk errors in the past week

```
JOB,JN=TNG__A.
ACCOUNT,AC=265124,US=TNG,UPW=TNG.
EXTRACT.
/EOF
SELECT TYPE=HARDWARE,SUBTYPE=DISK.
```

d. Write a batch job that runs EXTRACT to search the $SYSTEMLOG for any memory errors in the past week.

```
JOB,JN=TNG__A.
ACCOUNT,AC=265124,US=TNG,UPW=TNG.
EXTRACT.
/EOF
SELECT TYPE=HARDWARE,SUBTYPE=SINGLE DOUBLE,SUMMARY.
```

e. Write a batch job that runs EXTRACT to search the $SYSTEMLOG for the $LOG messages from your last job.

```
JOB,JN=TNG__A.
ACCOUNT,AC=265124,US=TNG,UPW=TNG.
EXTRACT.
/EOF
USER=TNG__A.
TYPE=ASCII
```

f. Write a batch job that runs HERG and get error information on disk, tape and memory.

```
JOB,JN=TNG__A.
ACCOUNT,AC=265124,US=TNG,UPW=TNG.
ACCESS,DN=HERG,ID=DIAGSYS,OWN=U9909.
HERG,SN=201,RELLEV=X14.
```

g. Write an FDUMP job that dumps the datasets CRAY1SYSTEMDUMP with an ID=TNGSWCE and ED=______ and ED=_______ One is a COS hang and the other is an IOS hang.

```
JOB,JN=TNG__A.
ACCOUNT,AC=265124,US=TNG,UPW=TNG.
ACCESS,DN=A,PDN=CRAY1SYSTEMDUMP,R=READDUMP,ID=SWCE.
FDUMP.
/EOF
FILES,DDS=A.
DMEM,FWA=1400,LWA=16000.
FILES,DDS=B.
DMEM,TYPE=IOPØ,FWA=0,LWA=177777,R.
DXTR,LIMIT=200,TYPE=IOPØ.
```

Intended Lesson Result: To be able to run jobs that maintain permanent datasets on disk or jobs that gather statistics from the $SYSTEMLOG to evaluate the system performance online.

Exercise 3 APML Assembly

Skill: Use IOP instructions
Read and write APML code
Use system Macros

Tasks:

a. Write and assemble without errors, a program which loads and then adds two operand registers, storing the results in a local memory location.

```
JOB,JN=TNG__A.
ACCOUNT,AC=265124,US=TNG,UPW=TNG.
APML.
/EOF
          IDENT    APML1
          ABS
          REGDEFS  ,,(OP1,OP2)
*
          R!OP1=(DATA1)
          R!OP2=(DATA2)
          (RESULT)=R!OP1+R!OP2
          EXIT
DATA1     PDATA  12
DATA2     PDATA   10
RESULT    <1>
          END
```

b. Write and assemble without errors, a program which sums up the valid exit stack entries (0-9) and stores the result in a local memory location.

```
JOB,JN=TNG__A.
ACCOUNT,AC=265124,US=TNG,UPW=TNG.
APML.
/EOF
         IDENT     APML2
         START     SETUP
         ABS
         REGDEFS   ,,(OP,ACC,EP)
*
SETUP    R!ACC=Ø
         EGET      R!EP
         EPUT      Ø
LOOP    EQUALS     *
         EXSGET    R!OP
         R!ACC=R!ACC+R!OP
         EINCR
        P=LOOP,A<=R!EP
         EPUT      R!EP
         EXIT
         END
```

c. Write and assemble without errors, a program which disables interrupts, stores the exit stack in local memory, sums up the interrupting channel numbers in an operand register, and restores the exit stack when all interrupts have been "handled" in this way

```
JOB,JN=TNG__A.
ACCOUNT,AC=265124,US=TNG,UPW=TNG.
APML.
/EOF
          IDENT     APML3
          START     HERE
          ABS
          REGDEFS   ,,(PXSA,EP,INDEX,SUM)
*
HERE      I=Ø
          R!SUM=Ø
          R!INDEX=EXSAVE
          EGET          R!EP
          EPUT          Ø
LOOP      EQUALS        *
          EXSGET        R!PXSA
          (R!INDEX)=R!PXSA
          R!INDEX=R!INDEX+1
          EINCR
          P=LOOP,A<=R!EP
*
ADD       IOR:1
          R!SUM=A+R!SUM
          P=ADD,A<=Ø
          EPUT          R!EP
          R!INDEX=EXSAVE
*
RESTORE   R!PXSA=(R!INDEX)
          EXSPUT        R!PXSA
          R!INDEX=R!INDEX+1
          EDECR
          P=RESTORE,E#Ø
          EXIT
EXSAVE    <20>
          END
```

Intended Lesson Results: Read IOS diagnostic code such as MOSTEST or HSPTEST and be able to follow the IOS macro's and Kernel

```
       6    <macro>                                REGDEFS     ,,(FMADR,MSGLH)
                                         *
       6   014000 /000326  017000                  R!MSGLH=MSGEND-MESSAGE
          /000143  024431
      13                                 GET1      *
      13    <macro>                                GETMEM     R!MSGLH,R!FMADR
      33    <macro>                                  $IF  (A#0)
      35    <macro>                                    PAUSE 1
      46   071033                                      P=GET1
      47    <macro>                                  $ENDIF
      47    <macro>                                CLEAR       START=R!FMADR,COUNT=R!MSGLH
      62    <macro>                                COPY        %B+MESSAGE,R!FMADR,R!MSGLH
                                         *
     102    <macro>                                MSG         R!FMADR                         .DISPLAY MESSAGE ON KERNEL CON
                                         *
     113    <macro>                                CALL        TNG2                            .CALL TNG2
                                         *
     125   -<macro>   - -                          RELMEM      R!FMADR
     135    <macro>                                RETURN                                      .RETURN
                                         *
     143    005015                       MESSAGE  '*******************************************************'H,5015
     200    005015                                '*                                                     *'H,5015
     235    005015                                '*                TNG1  WILL CALL TNG2                 *'H,5015
     272                                          '*                                                     *'
     326                                 MSGEND    *
                                                   END
```

```
    6   <macro>                           REGDEFS    ,(BMUP,BMLO),(RSPADR,ID)
                                          EXT        %MYID
                                *
    6                           GET1      *
    6   <macro>                           GETMEM     200,R!RSPADR ·                .ALLOCATE LM BUFFER FOR MSGR
   26   <macro>                             $IF  (A#0)
   30   <macro>                               PAUSE 1
   41  071033                                 P=GET1
   42   <macro>                             $ENDIF
   42   <macro>                           CLEAR      START=R!RSPADR,COUNT=200
   55   <macro>                           MSGR       %B+MESSAGE,200,R!RSPADR        .INPUT MSG FROM CONSOLE
   74                           GET2      *
   74   <macro>                           MGET       R!BMUP,R!BMLO                  .ALLOCATE BM FOR MSG
  122   <macro>                             $IF  (A#0)
  124   <macro>                               PAUSE 1
  135  071041                                 P=GET2
  136   <macro>                             $ENDIF
  136   <macro>                           MOSW       R!BMUP,R!BMLO,R!RSPADR,200     .WRITE MSG TO BM
                                *
  156  020000  024433                     ID=R!%MYID
  160  020433  012001  011003   NXTIOP    ID=ID+1&3
       024433
  164  020433  023000  102026                  P=DONE,ID=R!%MYID
  167   <macro>                                OUTCALL     ID,TNG3,(R!BMUP,R!BMLO)
  213  071033                             P=NXTIOP
  214                           DONE      *
  214   <macro>                           RELMEM     R!RSPADR
  224   <macro>                           MPUT       R!BMUP,R!BMLO                 .DEALLOCATE BM MSG BUFFER
  240   <macro>                           RETURN                                   .RETURN TO TNG1
                                *
  246  005015                   MESSAGE  '*                TNG2 OVERLAY                        *'H,5015
  303  005015                            '*               ENTER A MESSAGE                      *'H,5015
  340                                    '*******************************************************'
  374                           MSGEND    *
                                          END
```

```
      6    <macro>                          REGDEFS     ,(BMUP,BMLO),(MSGADR)
                                  *
      6                           GET1      *
      6    <macro>                          GETMEM    200,R!MSGADR
     26    <macro>                            $IF (A#0)
     30    <macro>                              PAUSE 1
     41   071033                                P=GET1
     42    <macro>                            $ENDIF
     42    <macro>                          CLEAR     START=R!MSGADR,COUNT=200
     55    <macro>                          MOSR      R!BMUP,R!BMLO,R!MSGADR,200
     75    <macro>                          MSG       R!MSGADR
    106    <macro>                          RELMEM    R!MSGADR
    116    <macro>                          RETURN
                                            END
```

LAB 2

Kernel and Station Commands

With the aid of all furnished reference materials complete the following tasks:

1. Start IOS from the binaries on Disk
   set DA to 60
   Push MC/DS IOS/$KERNE

2. View the deadstart parameter file through the IOS editor and verify that you are RESTARTing COS
   START COS RESTART,
   E

3. Initialize the IOP station
   StaTION /LogON

4. Go through the startup process until you have the STARTUP COMPLETE visable with the Y. command.
   STMSG
   REP

5. Shutdown the concentrators and network channels.
   ENDCONC NSCEND

6. Initialize an Interactive station.
   IA IOP LOG IAC

7. Using TEDI write the needed JCL and CAL that loops like a pass counter adding S1+1 and submit it as a job. Set the T=20 on the JOB statement.

8. Change it's priority, time limit and station ID
   ENT JSφ P 15

9. Turn off it's job class and the turn it back on.
   ClASS MED OFF

10. Change the limit of the number of jobs to 1.
    LIM 1

11. Display the LOOP jobs last $LOG
    JSTAT JSφ

12. Change all ** ID's to AP
    RouTE ** AP

13. Suspend the job.
    SuS JSφ

14. Rerun the job.
    RuN JSφ

15. How much is the system using the CPU over the user
    MON CPU

16. How much STP activity is going on
    STP

17. Display how the disk drives are configured
    STOR DISK

18. Display disk drive activity and error information.
    DISK STOR

COS Release Installation

With the aid of all furnished reference materials and the COS release letters and tapes, complete the following.

1. Start IOS from tape. Set DA to 22
   Push MC/DS 3

2. Start COS from tape. View the deadstart parameter file to:
   START ~~COS~~
   @MTO:5 @MTO:
   ,ED

   a. Verify that A1-23 is the master device
   b. All other disk drives are offline
   c. All other disk drives are configured down
   d. You are installing on the scratch drive only

3. Submit the BIN114 job to load the Product Set binaries
   SUBMIT @MTO:Ø

4. Submit the PL114 job to load the Program libraries
   SUBMIT @MTO:Ø

5. Run JCSDEF, PRVDEF, and ACCTDEF to setup the user enviroment
   SUBMIT @MTO:59

Diagnostics

With the aid of all furnished reference materials, and the Online diagnostics release letter and tape, complete the following.

1. Run the batch job you have prepared to install the release tape

2. Run the batch job to generate the diagnostics

3. Run the batch job to write the fdump tape

4. Run the batch job to print the listing for SFA

5. Using an interactive IOP station, Access and execute MENU

6. Analyize a harware failure introduced into the system

7. Archive the datasets on a disk drive and configure the dive down so that it may be serviced and then restore the datasets
PDSDUMP CONFIG

8. Restore the drive to the system.
CONFIG

9. Run HSPTEST

10. Run MOSTEST

11. Run DOM and F80M to format an 80mb disk pack

12. Go offline to DSS and verify the FDUMP worked and the diagnostics on it work.

COS and IOS Debug Utilities

With the aid of all furnished reference materials, complete the following tasks:

1. Hit CNTL-D

2. Startup IOS IOS/$KERNEL

3. Startup COS
    a. Modify the deadstart parameter file to breakpoint in STARTUP at address ____________ , which is at the message prompt beeps

4. Initialize the IOP station STATION /LOGON

5. Observe that the breakpoint suspended the system
    a. Remove the breakpoint REM Ø

6. Use COS debug to read any location in central memory in the necessary format, use DISPLAY to change the defaults A-Z

7. Use COS debug to write into these locations

    a. EXEC's XFT for events
    b. STP table address 15
    c. A Job's A and S registers

8. Use IOS debug to read the following
    a. IOP1's Local memory address ____________
    b. IOP1's Operand registers
    c. IOP1's main registers
    d. Buffer memory address ____________

9. Use IOS debug to write into the following
    a. IOP0's local memory address ____________
    b. IOP3's Operand register 500
    c. IOP3's acculmulator
    d. Buffer memory address ____________

10. Let the instructor crash the system

11. Restart IOS and COS processing the dump just created

12. PDSDUMP the CRAY1SYSTEMDUMP to tape.

COS and IOS Debug Utilities

With the aid of all furnished reference materials, complete the following tasks:

1. Hit CNTL-D

2. Startup IOS   IOS/#KERNEL

3. Startup COS
   a. Modify the deadstart parameter file to breakpoint in STARTUP at address ________, which is at the message prompt beeps

4. Initialize the IOP station   STATION   'LOSCN'

5. Observe that the breakpoint suspended the system
   a. Remove the breakpoint   REM 0

6. Use COS debug to read any location in central memory in the necessary format, use DISPLAY to change the defaults   A-E

7. Use COS debug to write into these locations

   a. EXEC's XFT for events
   b. STP table address 15
   c. A Job's A and S registers

8. Use IOS debug to read the following
   a. IOP1's Local memory address ________
   b. IOP1's Operand registers
   c. IOP1's main registers
   d. Buffer memory address ________

9. Use IOS debug to write into the following
   a. IOP0's local memory address ________
   b. IOP3's Operand register 500
   c. IOP3's accumulator
   d. Buffer memory address ________

10. Let the instructor crash the system

11. Restart IOS and COS processing the dump just created

12. PDSDUMP the CRAY1SYSTEMDUMP to tape.