CYBER 180 Technical Introduction

A course in the CYBER 180 curriculum

Student Handout

CONTROL DATA
EDUCATION COMPANY
a service of
CONTROL DATA CORPORATION

CYBER 180 Technical Introduction

A course in the CYBER 180 curriculum

Student Handout

CONTROL DATA
EDUCATION COMPANY
a service of
CONTROL DATA CORPORATION

ASE Part No.: RW3020

1 2 3 4 5 6 7 8 9/86 85 84 83 82 81

REVISION RECORD	
REVISION	**DESCRIPTION**
A	Manual Release
(01-01-83)	
B	**Bring Manual up to date with Announced Product Line**
(7-30-84)	
Publication No. RW3020-1	

REVISION LETTERS I, O, Q AND X ARE NOT USED

CYBER 180 TECHNICAL INTRODUCTION

Address comments concerning this manual to:

CONTROL DATA CORPORATION
National Coordinator
5001 West 80th Street
Bloomington, Minnesota 55437
Attn: Curtis Vicha

or use Comment Sheet in the back of this manual.

CONTROL DATA PRIVATE

TABLE OF CONTENTS

COURSE DESCRIPTION

COURSE TITLE

CYBER 180 Technical Introduction

COURSE NUMBER

RW3020

COURSE LENGTH

Two days

DESCRIPTION

This course introduces the CYBER 180 hardware and the operating system (NOS/VE) designed to run on it. Both topics are covered in survey fashion. Hardware discussions include virtual memory protection, interrupts, and some instructions. Software discussions involve the implementation language, job flow and migration, and various features.

PREREQUISITES

There are no prerequisites for this course but the course is oriented to programmers. Some comparisons are made between NOS and NOS/VE and between CYBER 170 and CYBER 180. This course is a prerequisite for all the courses in the CYBER 180 software curriculum.

COURSE OBJECTIVES

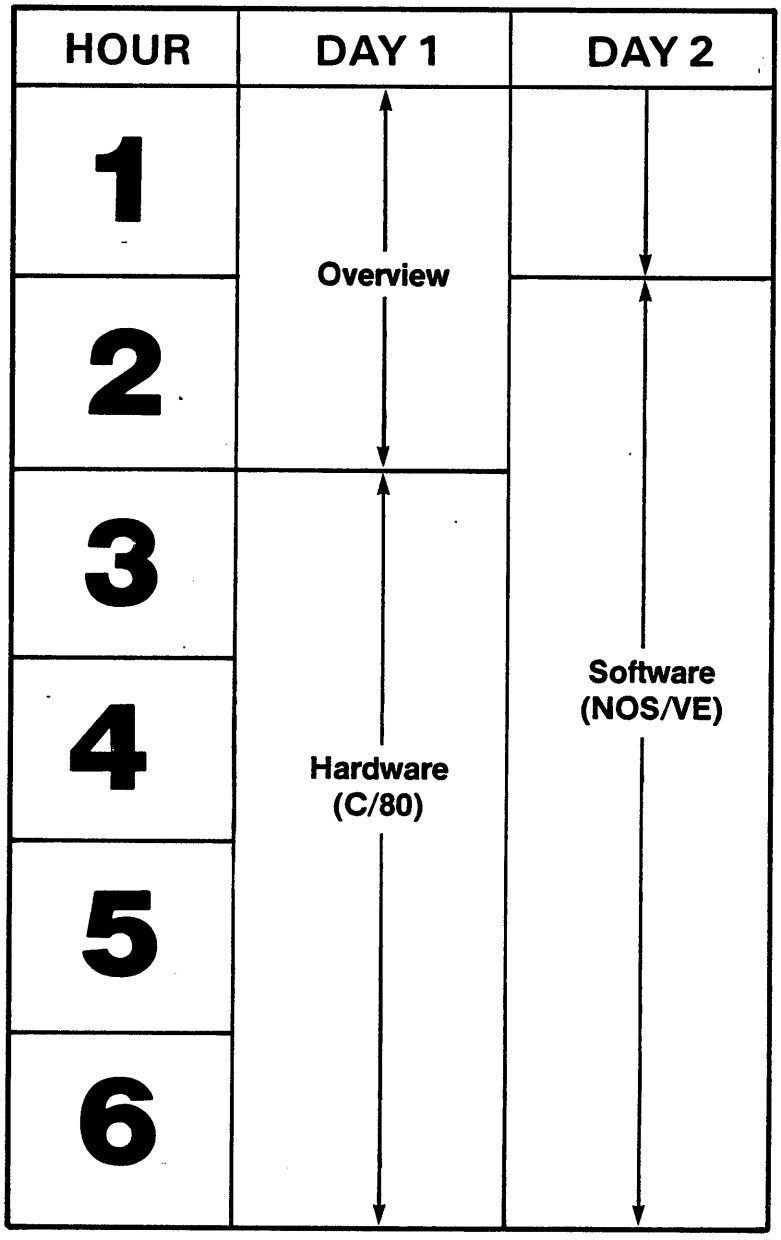
Generally, the student should become familiar with basic concepts of the C180 hardware and NOS/VE; he or she should be able to define or describe the most basic acronyms, phrases, "buzzwords," and the concepts behind them. Specific objectives are listed at the beginning of each chapter.

REFERENCE MATERIAL

The manuals currently available for general reference are the:

CDC Cyber 170 Models 815, 825, 835, 845 and 855 Computer Systems
CDC Cyber 180 Models 810, 830, 835, 845 and 855 Computer Systems
General Description CDC Pub. No. 60459960,
CDC Cyber 170 Computer Systems Models 815, 825, 835, 845 and 855
CDC Cyber 180 Computer Systems Models 810, 830, 835, 845 and 855
Hardware Reference Manual, Virtual State, Volume II
Instruction Descriptions and Programming Information
CDC Pub. No. 60458890

COURSE CHART



OUTLINE

I. Overview (2 hours)
- A. History
- B. Migration
- C. Models
- D. Architecture

II. Hardware Overview (5 hours)
- A. Block Diagram
- B. Buffer Memories
- C. Virtual Memory
  1. Access Privilege
  2. Rings
  3. Key/Lock
- D. CALL/RETURN and Interrupts
  1. Stack
  2. Registers
  3. Exchange Package
  4. Exchange and Trap
- E. Instructions and Data
  1. Instruction Formats
  2. BDP Instructions
  3 Floating Point Format

III. Software (5 hours)
- A. Concepts
  1. Objectives
  2. CYBIL
  3. OS Structure
- B. Job Flow
  1. Entry
  2. Initiation
  3. Command Processing
  4. Program Execution
  5. Termination
- C. Features
  1. Dual State
  2. Permanent Files
  3. Basic Access Methods
  4. Interactive Facility
  5. System Command Language
  6. Operator Facility
  7. Products

VI. Appendixes
1. Register Definition
2. Abbreviations
3. 170/180 Comparisons
4. Review Questions

CHAPTER 1: OVERVIEW

PREVIEW

History of NOS/VE Project
Migration Strategy
Product Line Models
Hardware Architecture

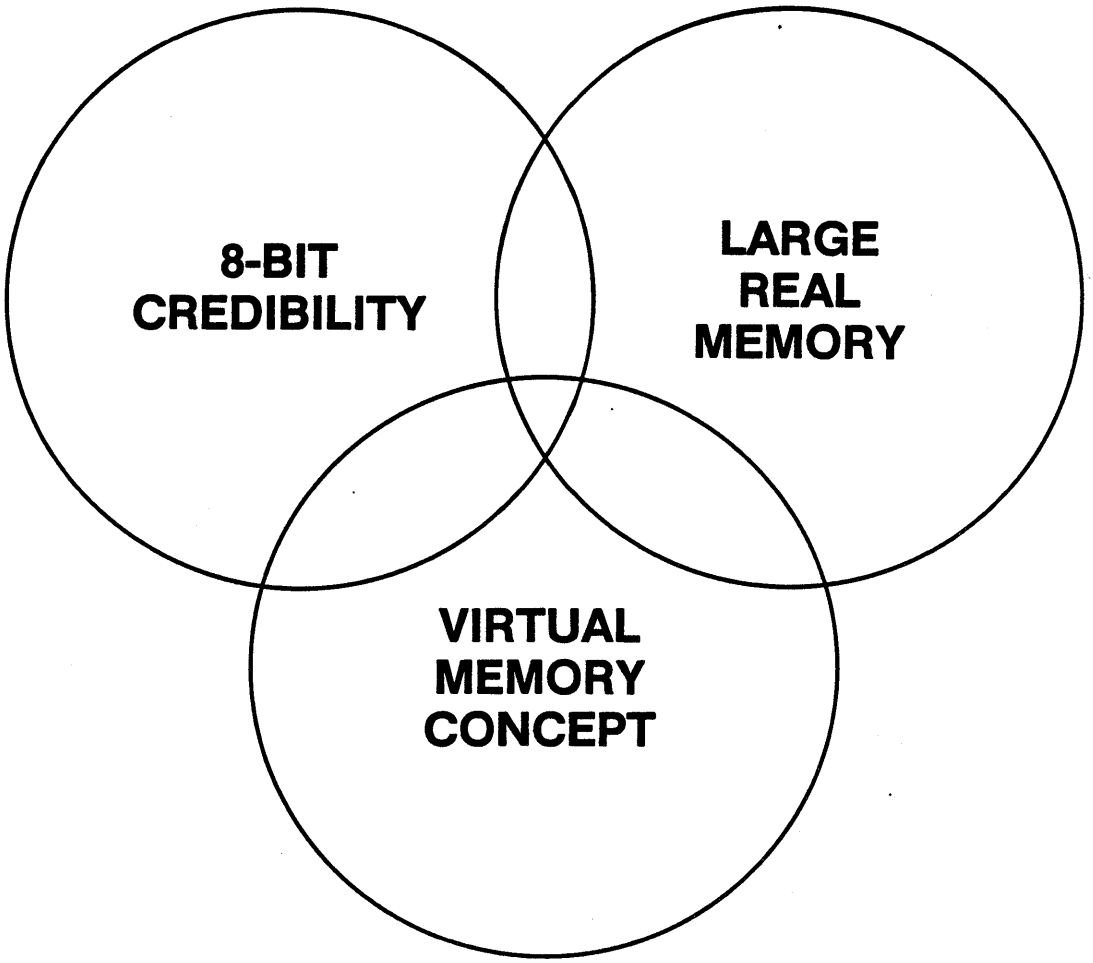
REFERENCES

An Introduction to CYBER 180, Chapter 1

OBJECTIVES

- State the basic requirements of the C180 hardware.
- Outline the strategy for migrating CDC's current customer base to C180.
- Distinguish between the models of the C180 product line which are currently defined.
- List the major features of the C180 architecture.
- List the kinds of protection available through the virtual memory protection mechanism.
- Explain the RAM acronyms.

REQUIREMENTS



Mission: A unique architecture for all new models to be produced and marketed by Control Data in the future.

CYBER 180 HISTORY

1973 -NCR/CDC Task Force
-Virtual Memory
-Large Memories
-BDP
-Single set of Software

1974 -Processors, Memories and I/O Subsystems Defined
S1, S2, S3

1975 -Work Continued on Memories and CPUs

1976 -Power on CPU
-IPL Named CYBER 180
-AD&C Formed/Joint Venture Dissolves
-IOU Defined

1977 -Power on Memory
-Work Started on Operating System

1978 -Power on IOU
-Work Started on Products

1979 -Power to First Model

1980 -Development Machine Available
-First Early Bird Customer

1981 -Operating System Release toTest

1982 -Products Release to Test
-800 Series Announced

1983 -NOSVE R1 Internal Release
-Product Introduction

1984 -CYBER 180, NOS/VE Product Announcement

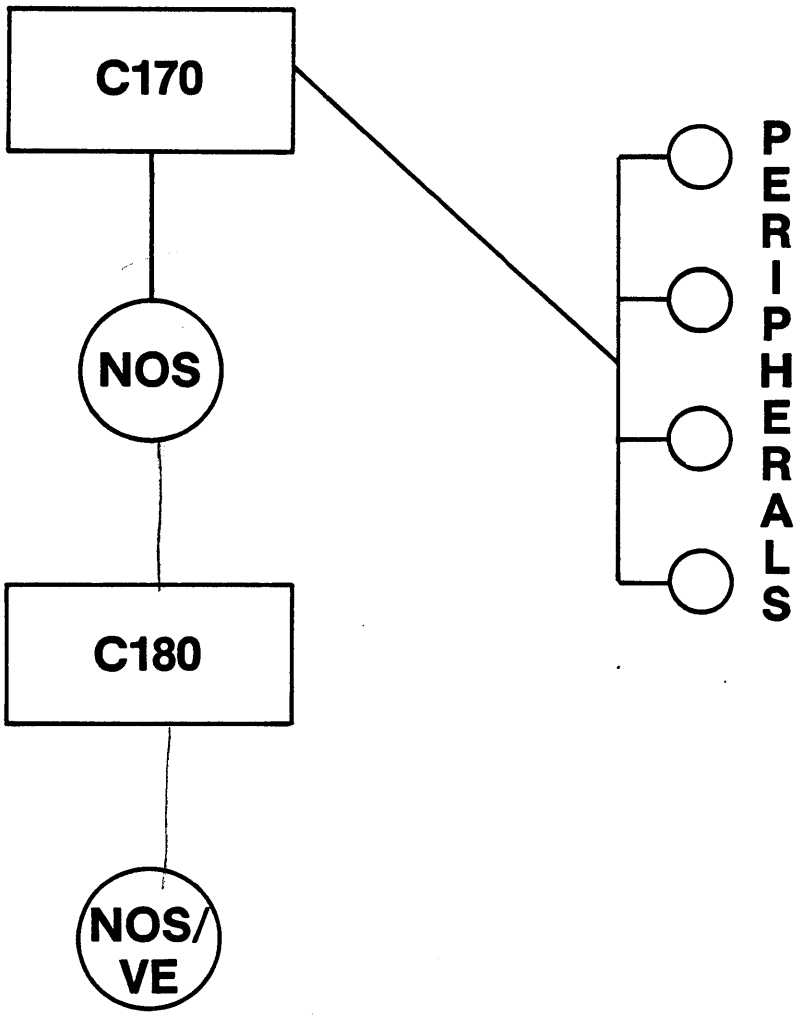
CYBER 180 STATUS

- o Specifications
- o Mainframes

C170	Models	C180
815 -	S1(-)/S1CR(-)	- 810
825 -	S1/S1CR	- 830
835 -	S2	- 835
845 -	S3(-)	- 845
855 -	S3	- 855
	THETA	- 990

- o Operating System NOS/VE
  - -Design
  - -Implementation
  - -Test
- o Products
  - -Compilers
  - -Tools and Utilities
  - -Documentation
  - -Courses
  - -Applications

MIGRATION STRATEGY



MIGRATION STEPS
CURRENT CYBER 170 SYSTEM

MAINFRAME REPLACED BY DUAL-STATE MAINFRAME

- Continue to Run 170 Software
- Retain Peripherals
- Price/Performance Improvement
- Memory Expansion
- No Conversion

Gradually Convert to New Software as Features of CYBER 180 Software Become Attractive

vs.

Suddenly Convert to New Software and Hardware

TECHNICAL MIGRATION FACILITIES

- Dual-State Mainframes
- Mixed-State Software Capability
- Source Language Compiler Compatibility
- Conversion Aids
- Some External Compatibility in Operating Systems

MODEL COMPARISONS

MODELS	815 (S1 MINUS)	825 (S1)	835 (S2)	845 (S3 MINUS)	855 (S3)
AVAILABLE	1983	NOW	NOW	1983	NOW
PERFOR- 170	0.6	1.0	2.5	4.5	7.5
MANCE 180	0.7	1.2	3.0	5.5	9.0
CPU CLOCK (nsec)	50 x 2 100	50 x 2 100	56 x 2 112	64	64
CACHE SPEED (ns) SIZE	NO	NO	YES 112 2K WORDS	YES 64 2K-4K WORD	YES 64 2K-4K WORD
INSTRUCTION LOOK AHEAD	2 WORD BUFFER	2 WORD BUFFER	3 WORD P 2 WORD B	12 INSTRUCT	12 INSTRUCT
MICROCODED	ALL	ALL	MOST	SOME	SOME
MEMORY SIZES (MB)	2, 4, 8	2, 4, 8	4, 8, 12, 16	4, 8, 12, 16 (32)	4, 8, 12, 16 (32)
MEMORY CYCLE-ns	400 SERIAL	400 SERIAL	448 OVERLAP	448 OVERLAP	448 OVERLAP
MEMORY BOARD	64K MOS	64K MOS	16K MOS	16K MOS	16K MOS
OTHER	-	-	STANDARD COMPONENTS	ADVANCED LSI	ADVANCED LSI

RELATIVE TO C170

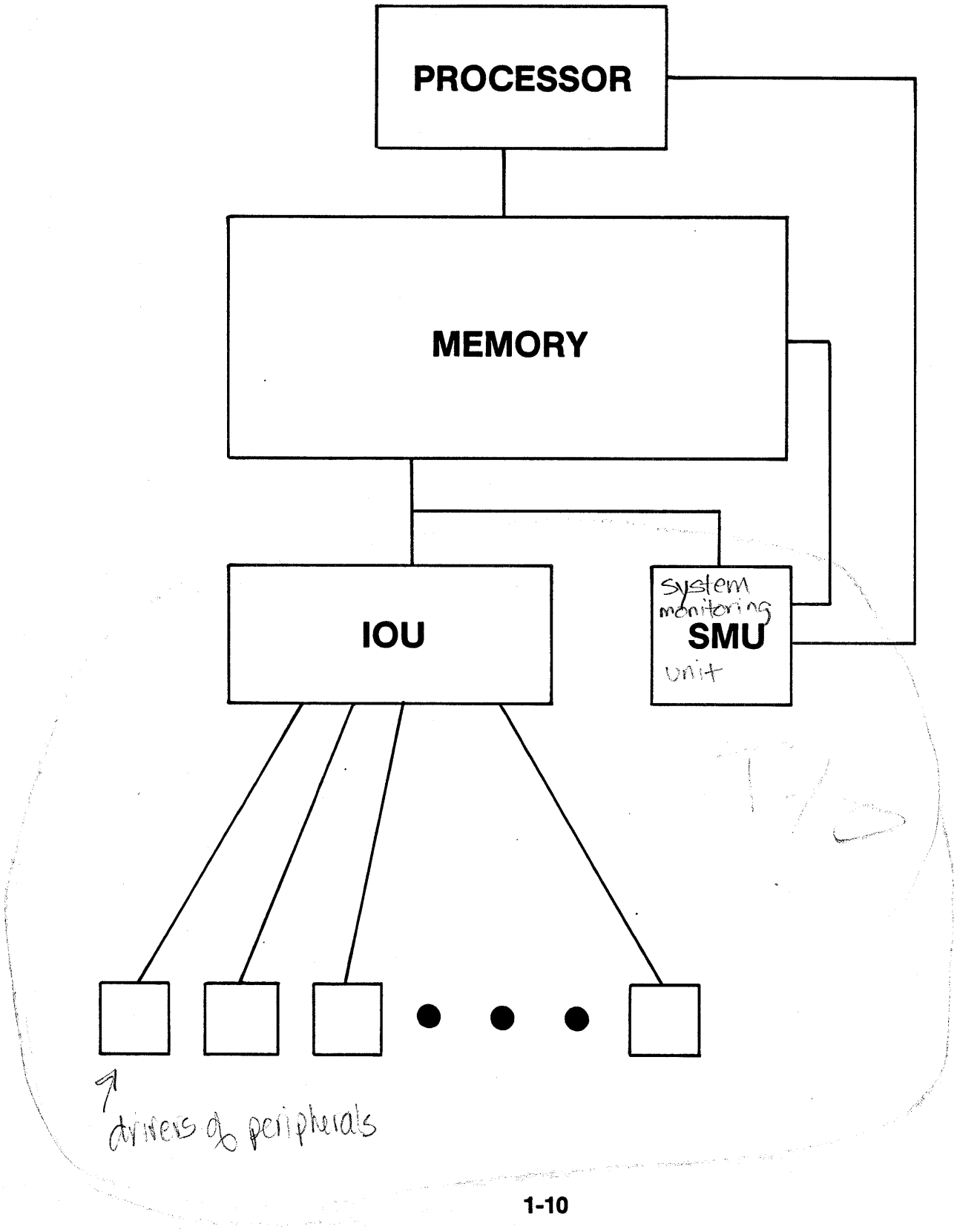
720 = 1
750 = 8-11
760 = 13-18

MODEL COMPARISONS

CYBER 180 170	810 815	830 825	835 835	845 845	855 855	990 -
	S1 MINUS S1CR MINUS	S1 S1CR	S2	S3 MINUS	S3	THETA
APPROX. -170 PERF.* -180	0.6 0.7	1.0 1.2	2.5 3.0	4.5 5.5	7.5 9.0	24 30
INSTRUCTIONS MICROCODED	ALL	ALL	MOST	SOME	SOME	LITTLE
VECTOR INSTRUCTIONS	N	N	N	N	N	Y
CACHE MEMORY SIZE	N -	N -	Y 2-4KW	Y 2-4KW	Y 2-4KW	Y 4KW
MAX XFER RATE 1W	200 NANO	100	56	64	64	16
MEMORY -MBYTES BANKS CYCLE	2,4,8,16 2,4 400 NS	2,4,8,16 2,4 400	4,8,16 8 448	4,8,16 8 448	4,8,16 8 448	8,16,24 32 16 64
DUAL PROCESSOR OPTION	N	Y	N	N	Y	Y
COOLED	AIR	AIR	LIQUID	LIQUID	LIQUID	LIQUID

*PERFORMANCE RELATIVE TO C170-720 = 1.0

CYBER 180 CONFIGURATION



HARDWARE FEATURES

CPU SUPPORT OF

- Scientific/Engineering
- Character Handling
- CYBER 170 State
- Virtual Memory
- Security

MEMORY TO SUPPORT

- 64-Bit Words
- Byte Addressing
- Cache
- SECDED
- Reconfiguring

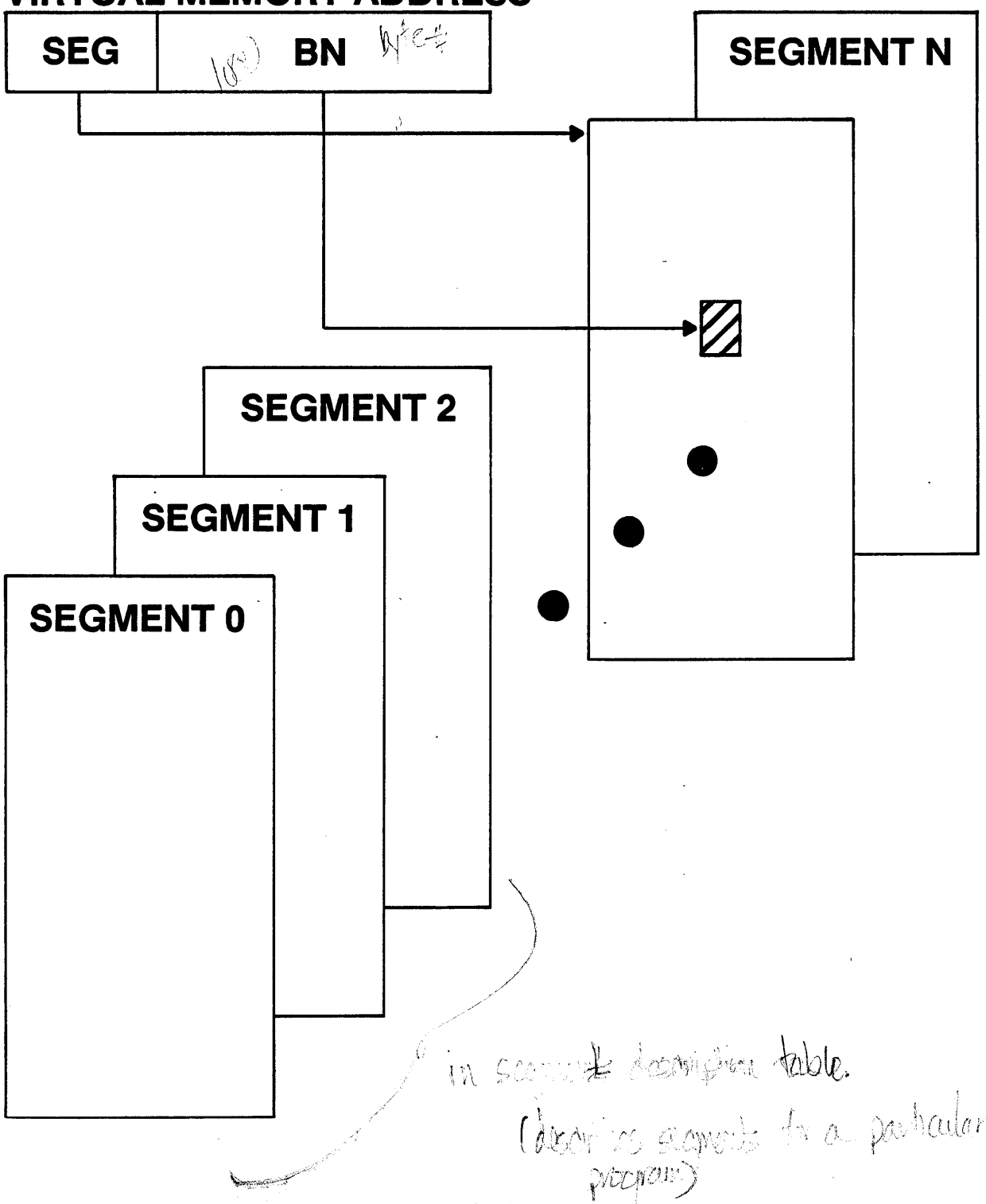
INPUT/OUTPUT UNIT TO SUPPORT

- Peripheral Processors
- 12- and 16-Bit Channels

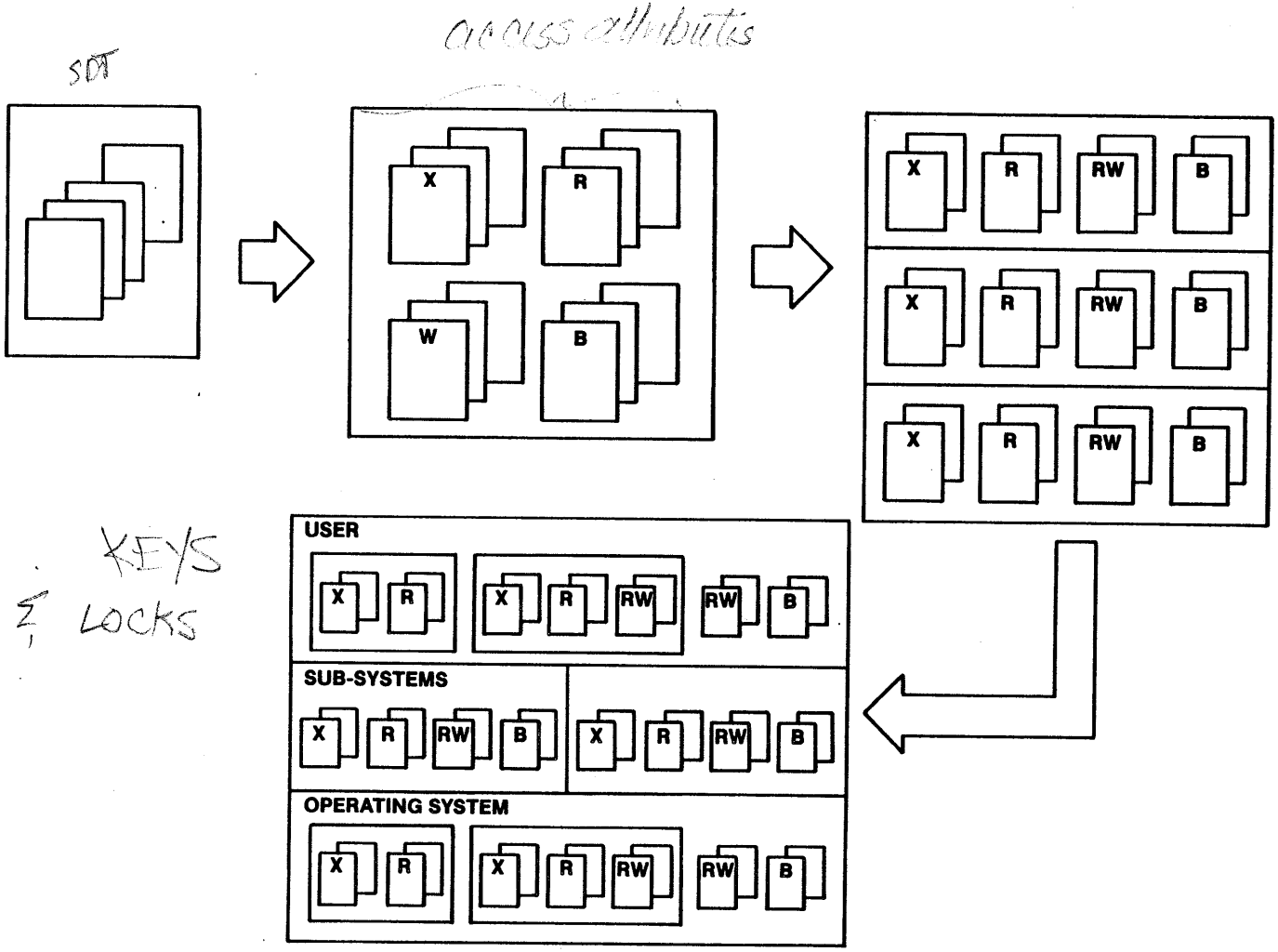
CYBER 180 VIRTUAL MEMORY

- Addressing
  - — Programmers Work Exclusively in Virtual Memory
- Protection
  - — System or User from Self
  - — System from User
  - — System from System (Compartments)
  - — Users from Other Users
  - — Users from System

CYBER 180 VIRTUAL MEMORY USER ADDRESS SPACE



EXAMPLE OF PROTECTION WITHIN AN ADDRESS SPACE

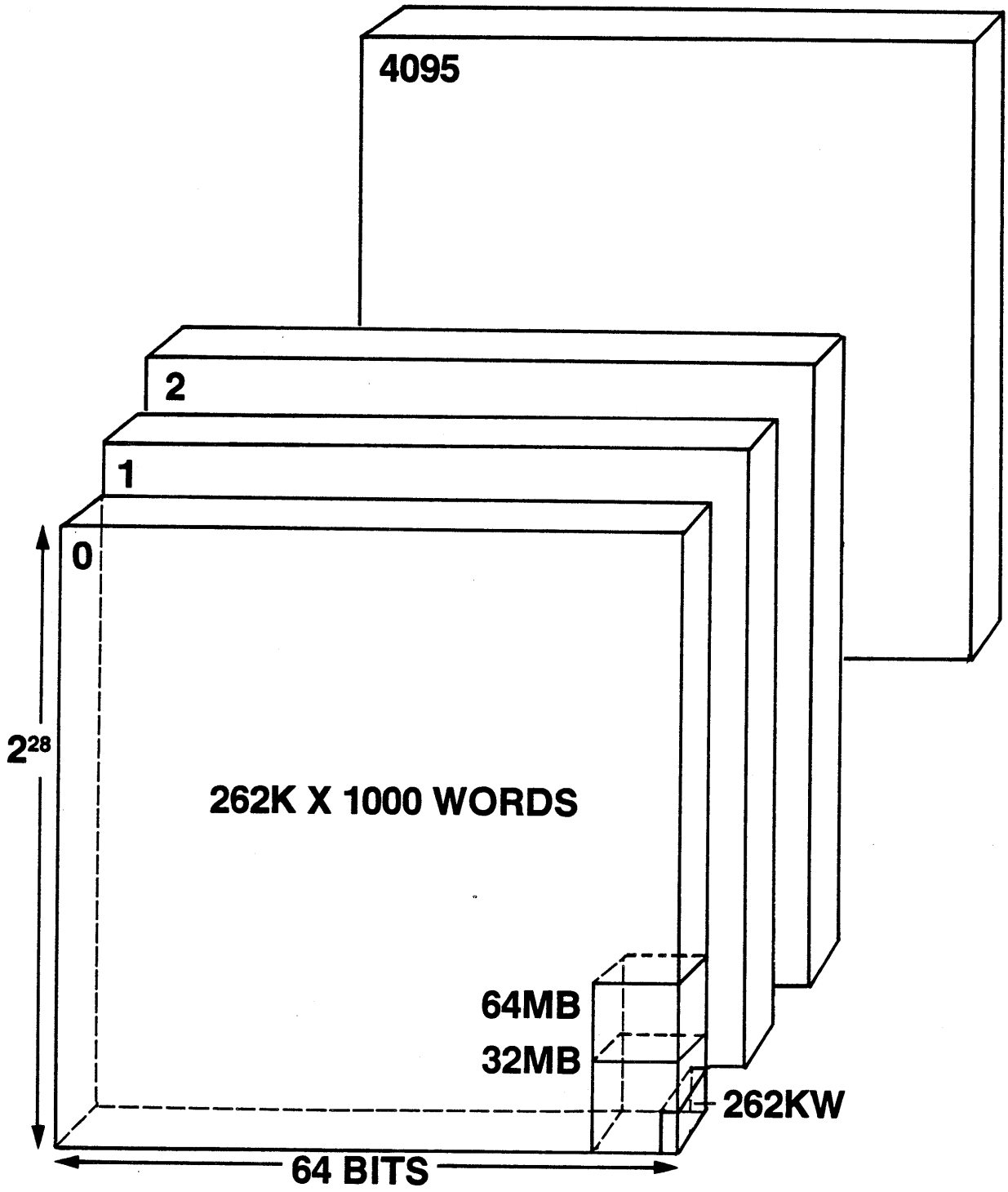


CYBER 180 VIRTUAL MEMORY

SEGMENT ATTRIBUTES

- Access Privileges
  - — Read
  - — Write
  - — Execute
- Ring Protection
  - — Ring(s) of Execution
  - — Ring(s) from which Segment may be Read/Written
  - — Ring(s) from which Segment may be Called
- Key/Lock Protection

MEMORY SIZE



MEMORY

- 64 Bit Word with SECDED
- Byte Addressable
- Very Large Real Memories
  - — 1MB to 64MB Configurations
  - — Physically Organized by Pages
- Virtual Memory Capability
  - — Logically Organized into Segments
  - — Sharable, Secure
  - — 4096 Segments of 2 Billion Bytes Each

INPUT OUTPUT UNIT

- 5 to 20 Peripheral Processors
  - — 170 Upward Compatible Instruction Set
  - — 16-Bit Memory, Instructions, Paths
- 8 to 24 Channels
  - — CYBER 170 Compatible
  - — 16-Bit Parallel High Speed Channel (future)
- Supports Maintenance Systems
  - — Dedicated System Monitor Unit
  - — Local/Remote Consoles

PERIPHERALS

THREE CLASSES

- Low Performance—Communication Lines
  - Unit Record Equipment
    - Printer
    - Card Reader
    - CRT Terminals
- Intermediate Performance—6000 Channel
  - Magnetic Tape
  - Serial Head Disk
- High Performance—50 M Bit Channel (future)
  - Parallel Head Disk
  - Secondary Storage (Future)

RELIABILITY—AVAILABILITY—MAINTAINABILITY (RAM)

R—DETECT ERRORS

- Computer-Aided Design
- Concurrent Diagnostics
- Error Logging Registers
- Parity/SECDED on Major Data Paths
- ~~Maintenance Control Unit~~

A—CONTAIN ERRORS

- Reconfiguration
- O/S in User Address Space
- Instruction Retry

M—FIX ERRORS

- Remote/On-Call Maintenance
- Fault Isolation Diagnostics
- Physical Environment Monitor Option

CHAPTER 2:
HARDWARE OVERVIEW

PREVIEW

Virtual Address Space
- Segmented
- Large
- Protected

Exchange vs. Trap

CALL/RETURN

Real Memory
- Large
- Paged

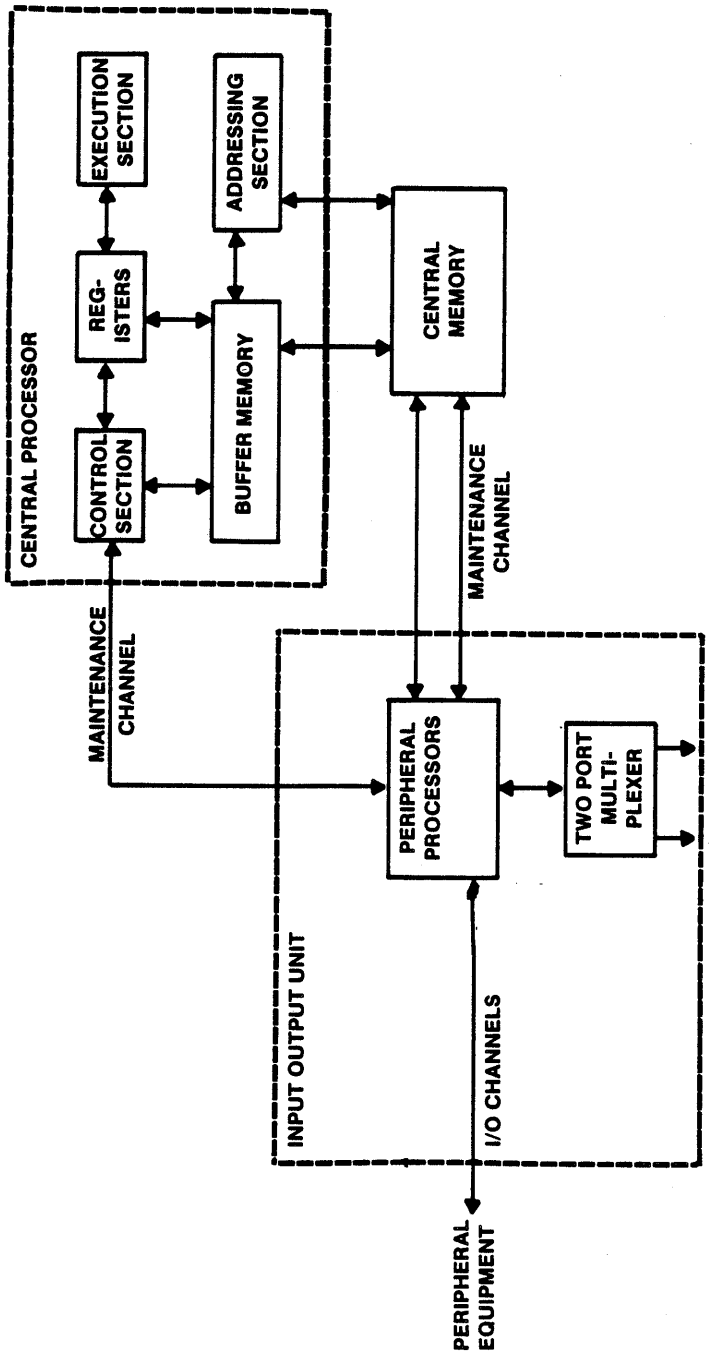
REFERENCES

General Description, Chapters 2-10

OBJECTIVES

- Explain relationship of SDT to memory protection.
- In general terms, explain the process of accessing a word in real memory.
- Describe how buffer memories improve access time.
- Describe the attributes which provide segment protection.
- Define Ring Brackets, Lock and Key, Access Attributes.
- Explain in general terms the CALL/RETURN process.
- Define Stack Frame Save Area.
- Identify types of registers.
- Explain how exchanges and traps are utilized.
- Identify types of instructions used on CYBER 180.

S2 BLOCK DIAGRAM



S2 BLOCK DIAGRAM

REGISTERS

- 16 X registers (64 bits)
- 16 A registers (48 bits)

MEMORY

- 64 Mbyte (potentially 2**31 byte)
- User virtual address space 4096 times 2**31 byte
- Hardware paged

PPs

- Memory size 4KX16 bits
- Up to 4X5 16-bit PPs
- Up to 6X4 12/16-bit channels
- 28-bit real memory address for PP read/write
- 250 ns major cycle time

MAINTENANCE CHANNEL

- Dedicated PP for SMU
- Loads processor micro code
- Read/write maintenance registers

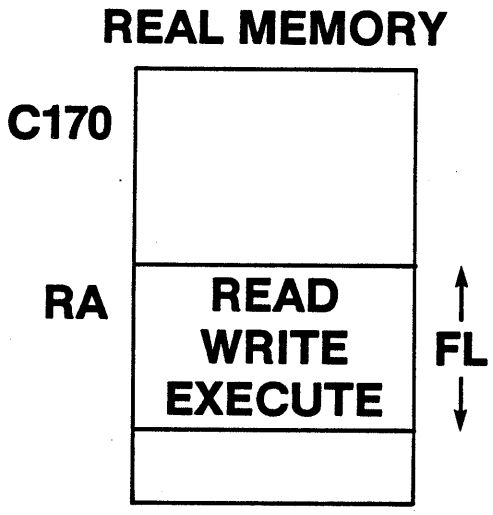
MORE CPU CAPABILITIES

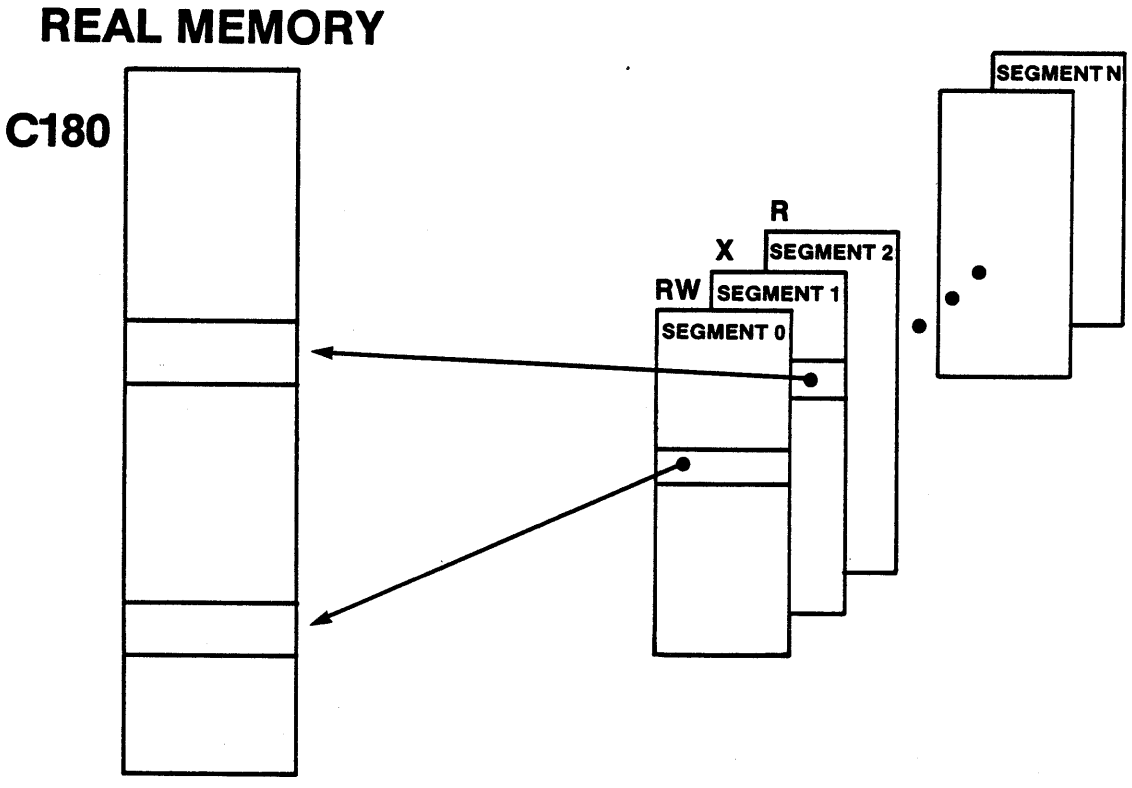
- Security Enforcement
- Paging
- Virtual Address Translation
- Performance Monitoring
- Debugging Aids
- Hardware Error Monitoring
- Timing
- Programmer Error Monitoring

PROCESSOR

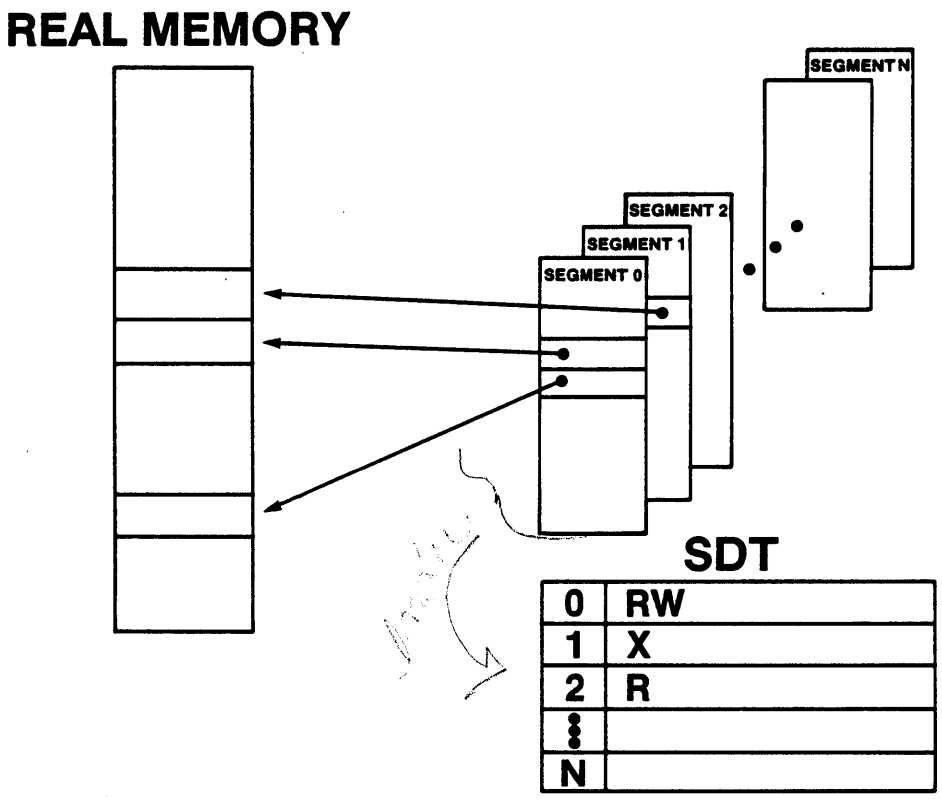
- Register Operation:
  - — 16 48 Bit A Registers
  - — 16 64 Bit X Registers
- 128 Two-Register Instructions
  - — 16 Scientific
  - — 17 BDP
  - — 76 General
  - — 18 System
  - — 16 and 32 Bit Formats
- Executes CYBER 170 Instruction Set Under Micro Program Control
- Security State Switching with 15 Levels
- Processor-Controlled 16KB–32KB Cache Memory
- Dual Processor Option (Selected Processors)

VIRTUAL MEMORY PROTECTION





PAGING



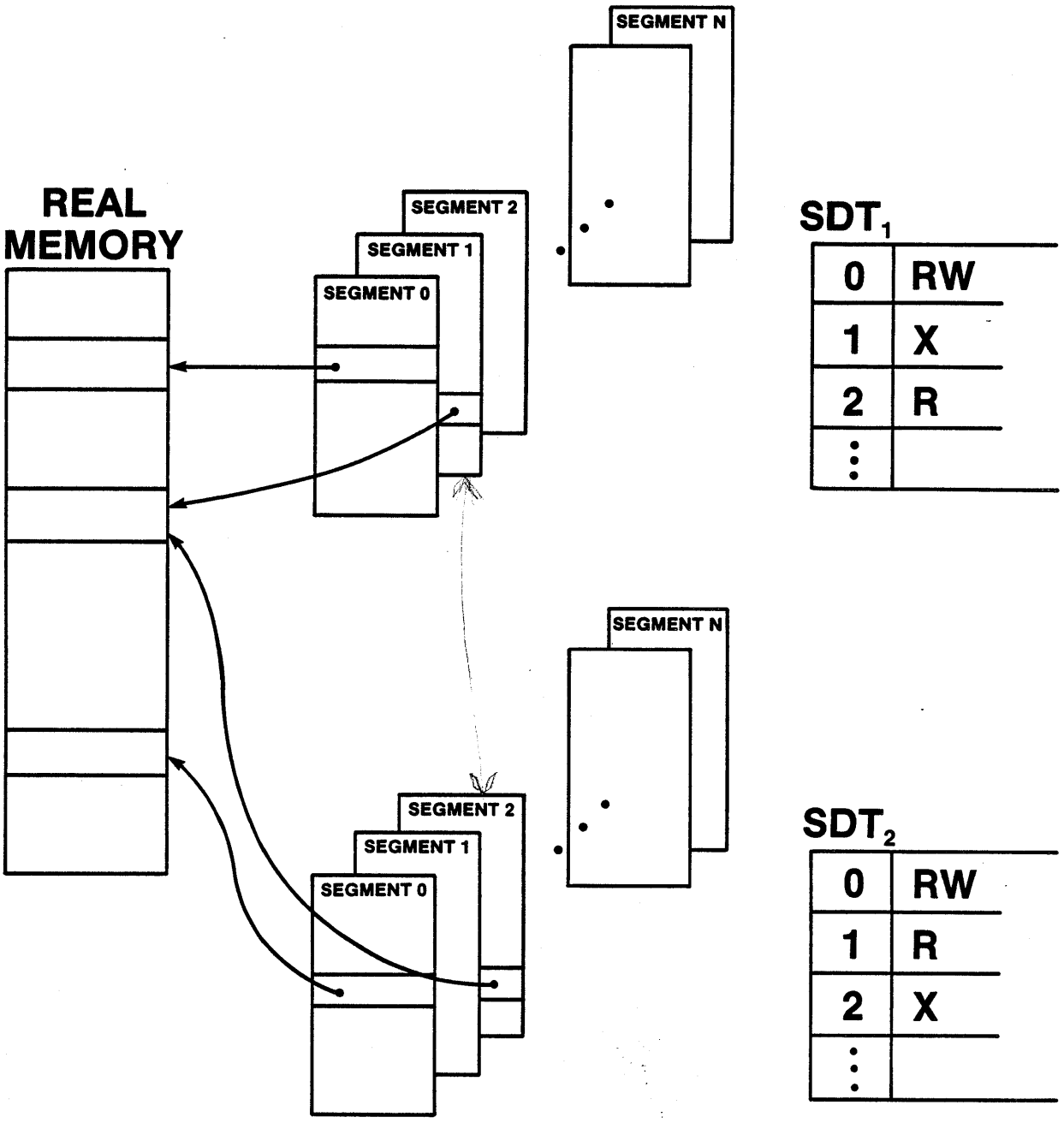
HARDWARE

- Page Fault if the Segment Portion is not in Real Memory
- Mark Modified Pages
- Note Most Recently Used Page

SOFTWARE

- Assign Pages
- Move Code and Data From Backing Store to Real Memory and Back if Modified

CODE SHARING



VIRTUAL MEMORY PROTECTION

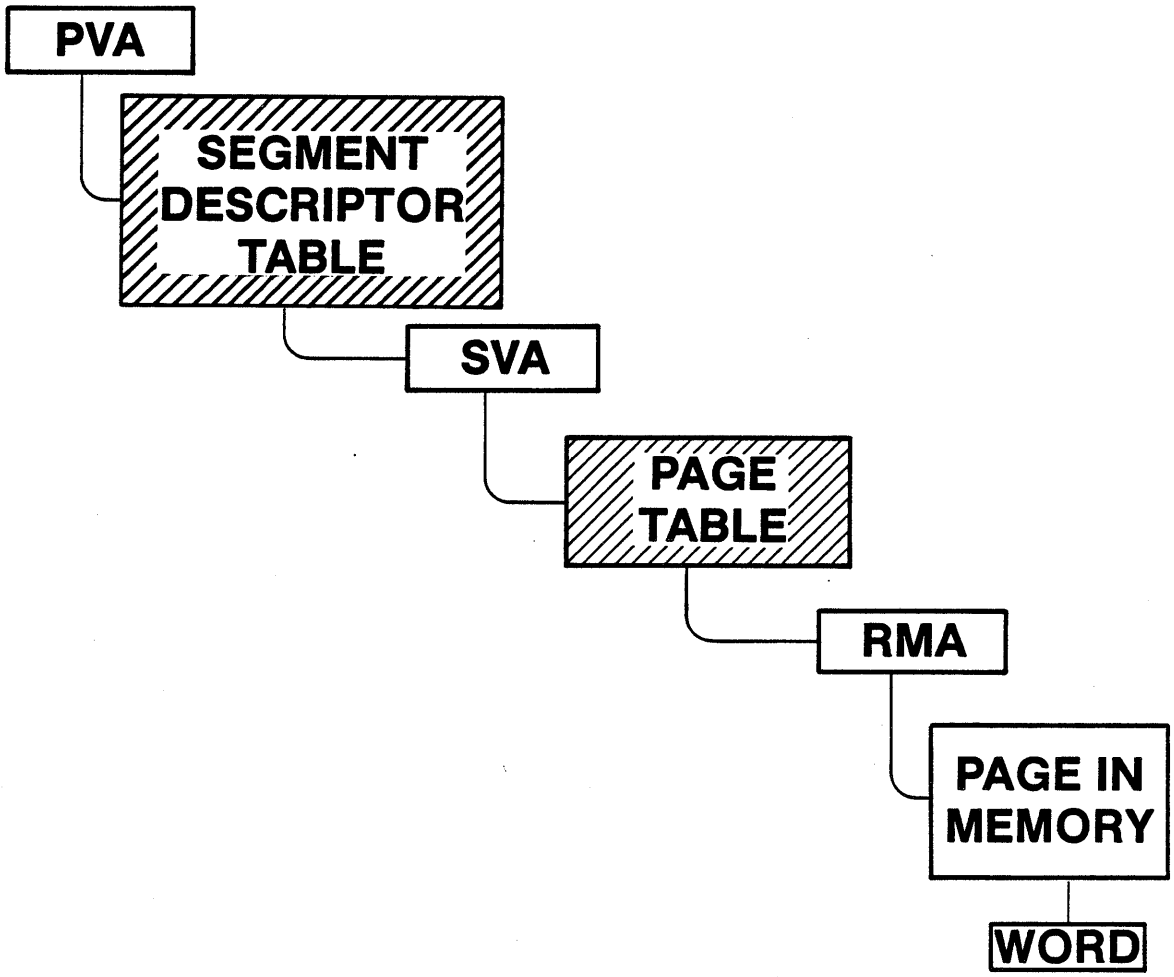
PROTECTION

- CYBER 170
  RA
  FL
- CYBER 180
  Segment Attributes
  Segment Descriptor Table (SDT)

REAL MEMORY RESIDENCE

- CYBER 170 Contiguous Space
- CYBER 180 Pages

OBTAINING
REAL MEMORY ADDRESS



OBTAINING REAL MEMORY ADDRESS

PVA—Process Virtual Address

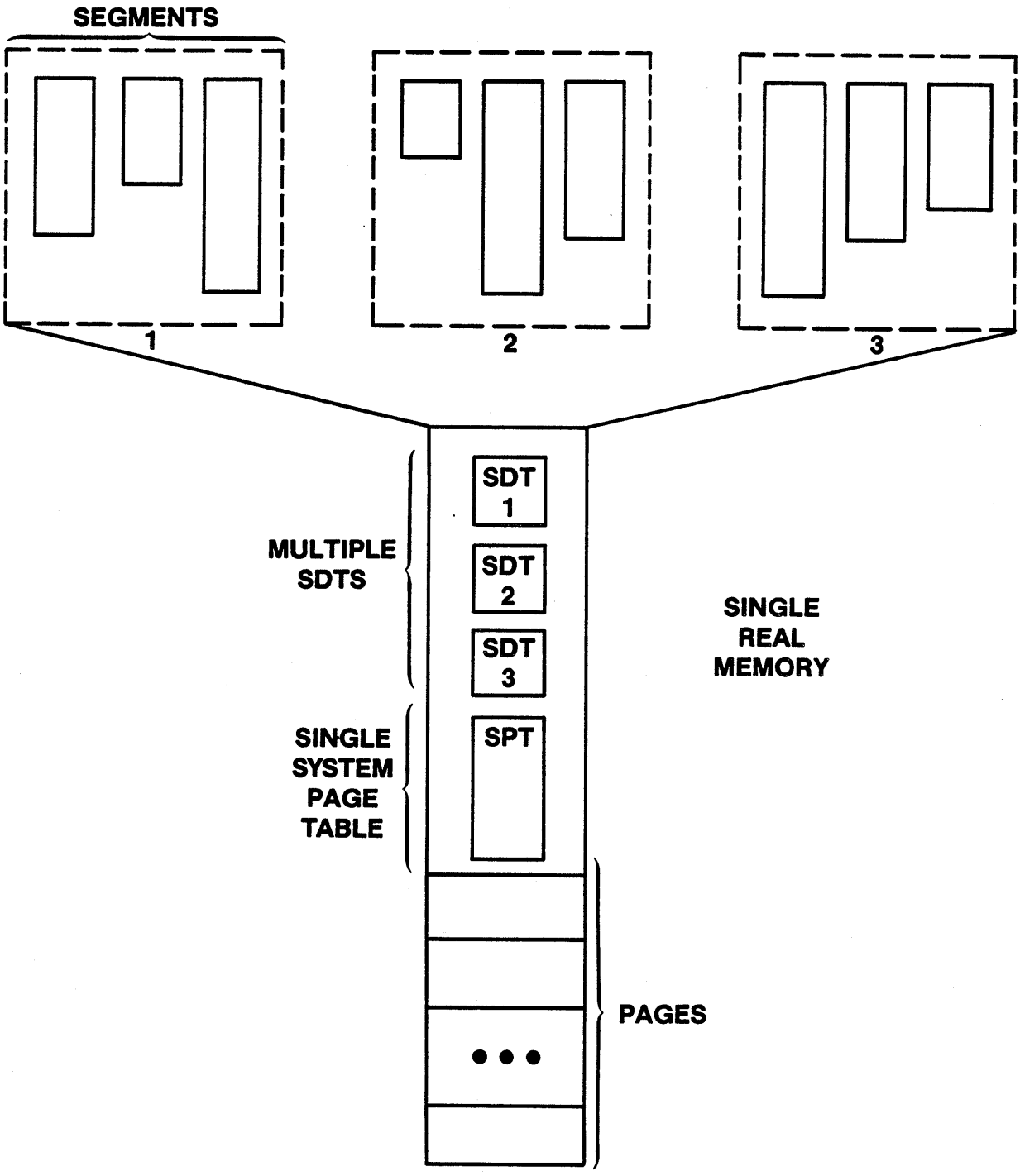
SDT—Translates PVA to SVA

SVA—System Virtual Address

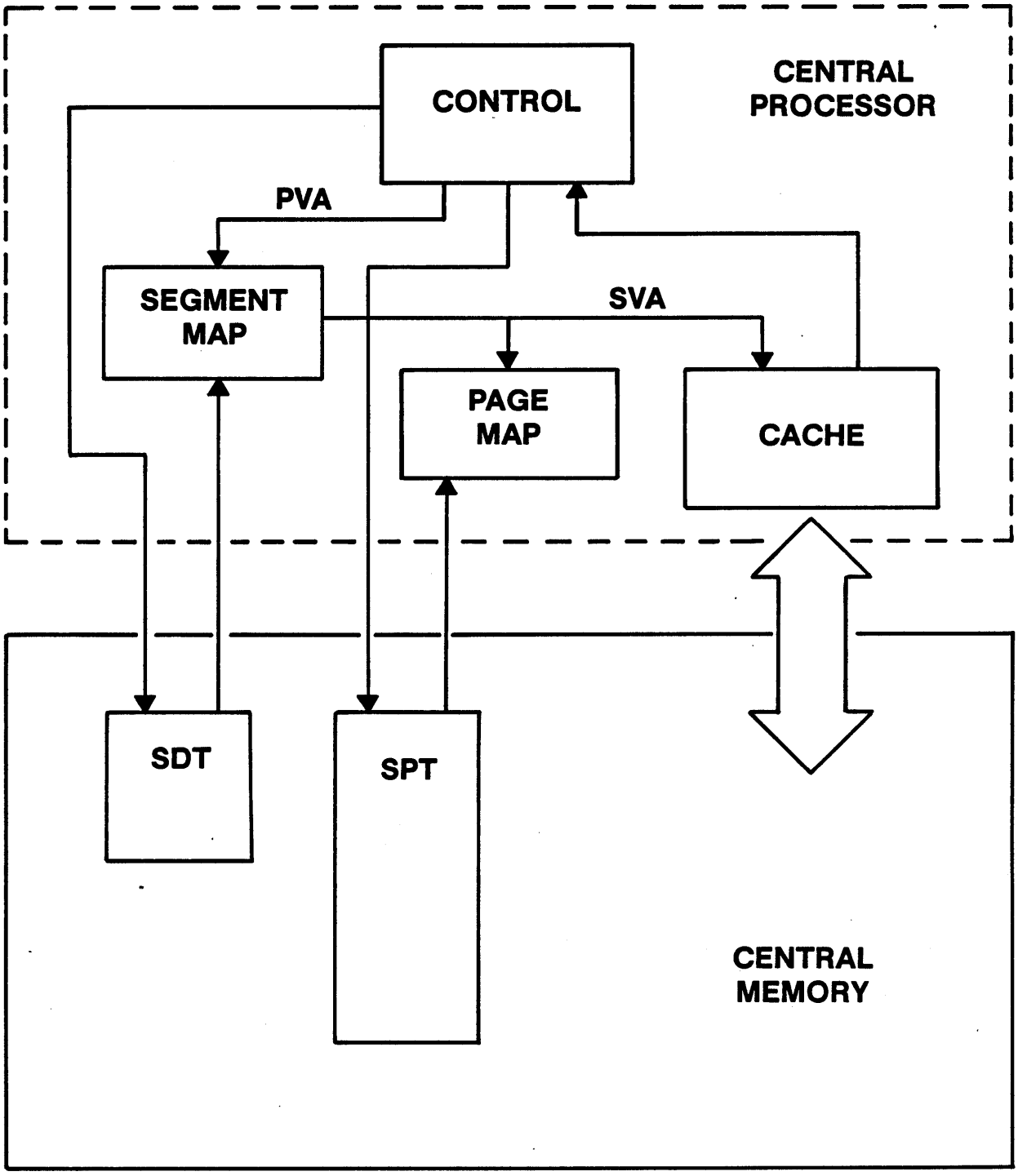
Page Table—Translates SVA to RMA

RMA—Real Memory Address

MULTIPLE VIRTUAL ADDRESS SPACE



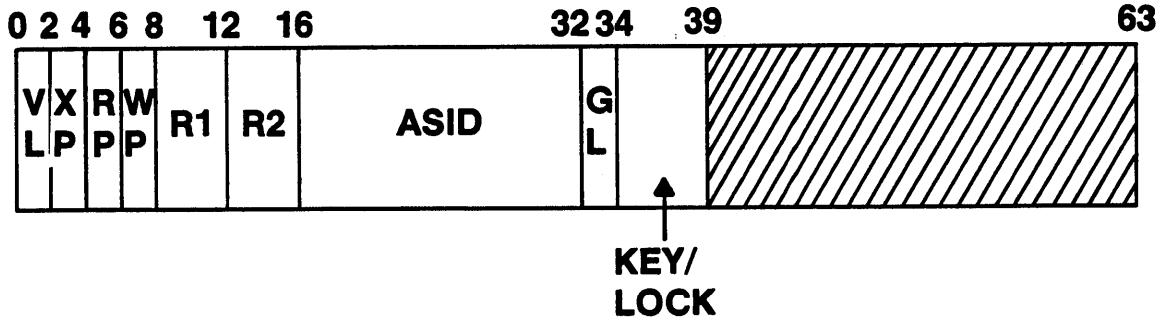
CYBER 180 BUFFER MEMORIES



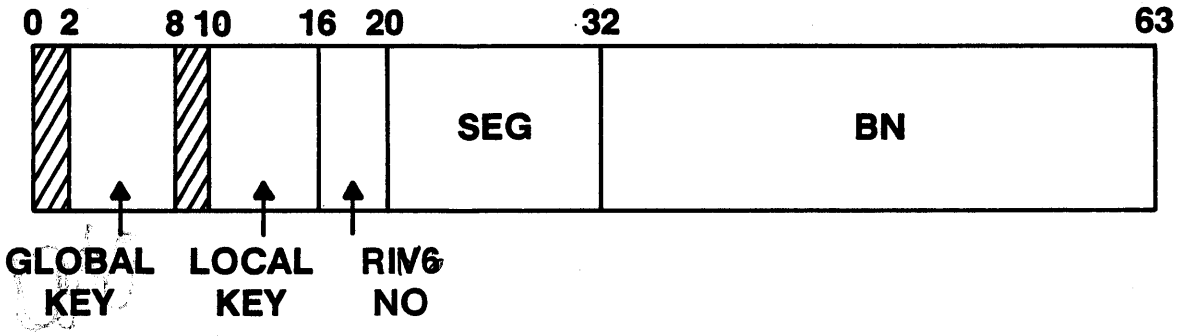
CYBER 180 BUFFER MEMORIES (MODEL DEPENDENT)

- Address Translation Buffer Memories
  - — Segment Map
  - — Page Map
- Central Memory Buffer Memory
  - — Cache
- Predominantly Hardware Managed
- Software Responsible for Purging

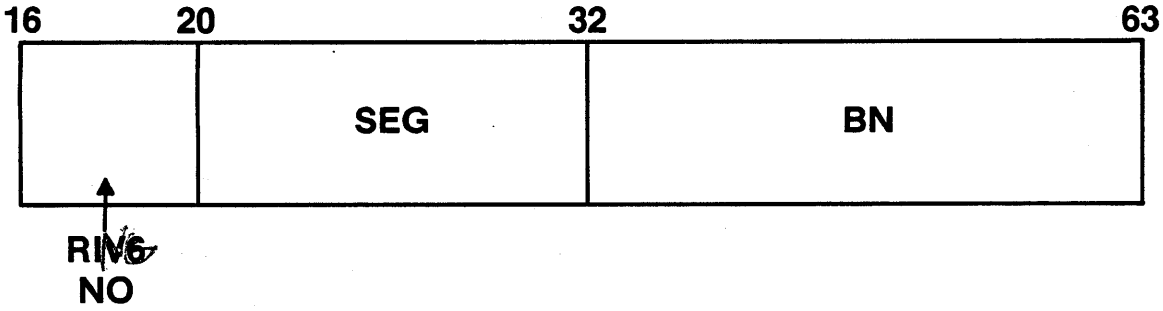
SEGMENT DESCRIPTOR TABLE ENTRY — SDE



PROGRAM ADDRESS REGISTER — P-REGISTER



PROCESS VIRTUAL ADDRESS



SEGMENT DISCRIPTOR TABLE ENTRY SDE

- BITS

0-1	VALID (VL) 00 INVALID 01 RESERVED 10 REGULAR SEGMENT 11 CACHE BYPASS SEGMENT
2-3	EXECUTE 00 NONEXECUTABLE 01 NONPRIVILEGED EXECUTABLE SEGMENT 10 LOCAL PRIVILEGED EXECUTABLE SEGMENT 11 GLOBAL PRIVILEGED EXECUTABLE SEGMENT
4-5	READ 00 NONREADABLE 01 READ UNDER CONTROL OF KEY/LOCK 10 READ NOT CONTROLLED BY KEY/LOCK 11 BINDING SECTION
6-7	WRITE 00 NONWRITABLE 01 WRITE CONTROLLED BY KEY/LOCK 10 WRITE NOT CONTROLLED BY KEY/LOCK 11 RESERVED
8-15	VALUES FOR RING BRACKETS
16-31	ASID—ACTIVE SEGMENT IDENTIFIER
32-33	LOCAL KEY/LOCK FLAG
34-39	VALUE OF KEY/LOCK

PROGRAM ADDRESS REGISTER P-REGISTER

- BITS

2-7	VALUE FOR LOCAL KEY
10-15	VALUE FOR GLOBAL KEY
16-19	PROCESS RING NUMBER
20-31	SEGMENT NUMBER
32	VALID ADDRESS BIT
33-63	BYTE NUMBER

PROCESS VIRTUAL ADDRESS

- BITS

16-19	PROCESS RING NUMBER
20-31	SEGMENT NUMBER
32	VALID ADDRESS BIT
33-63	BYTE NUMBER

SEGMENT PROTECTION ATTRIBUTES

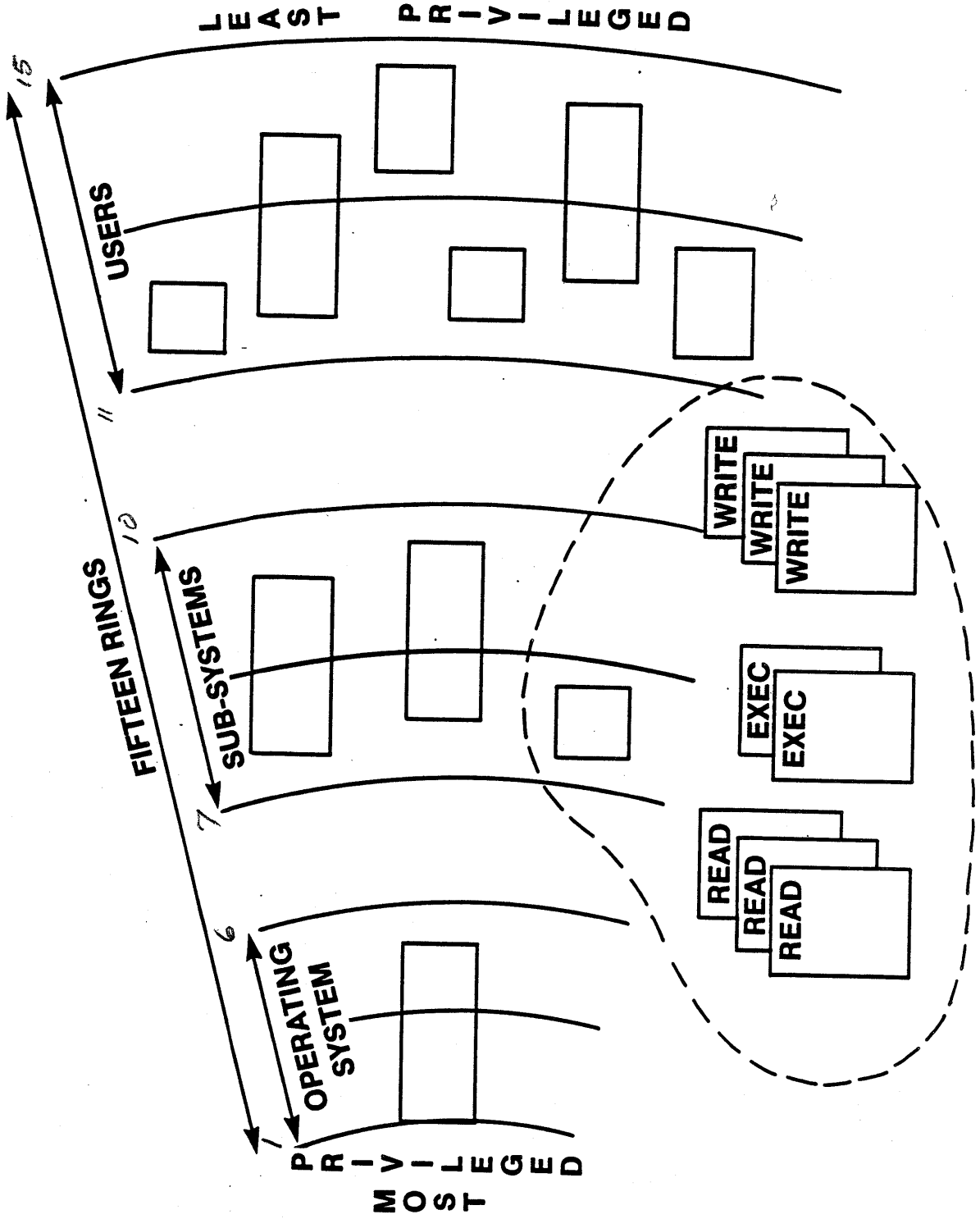
1. Access Attributes
- Read
- Write
- Execute

2. Rings
- 15 Hierarchical Levels of Protection
- "One Way" Protection

3. Local Key/Lock
- 64 Protection Codes
- "Local" Data Protection

Access to a Segment is Only Permitted When All Three Protection Tests are Passed.

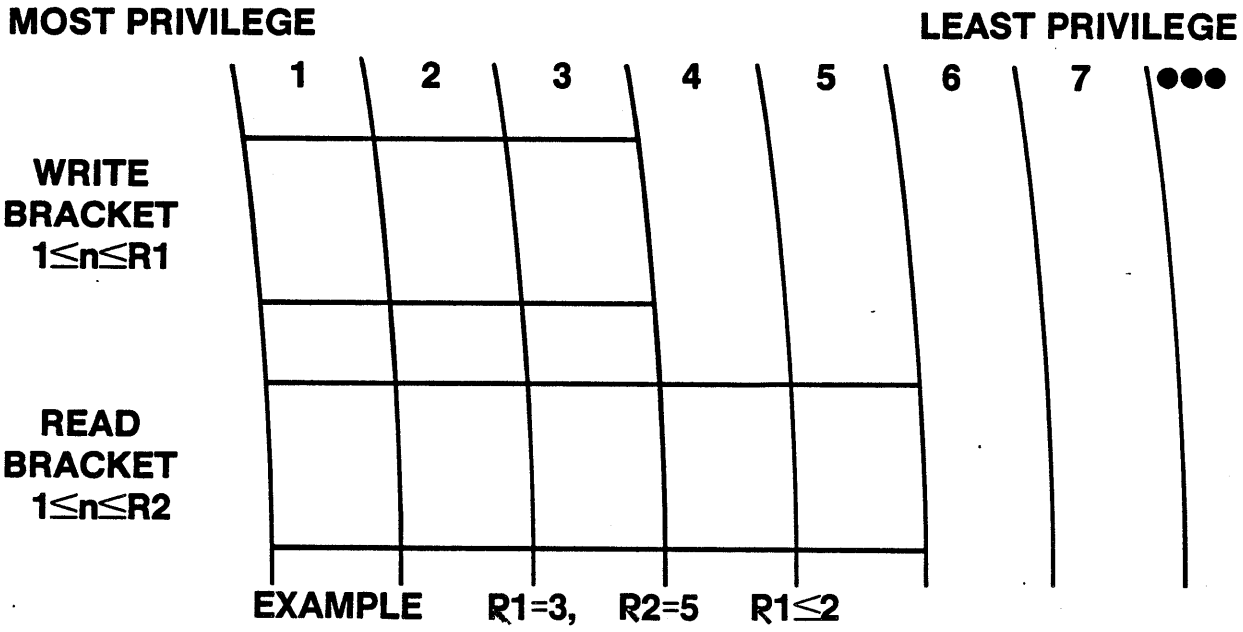
RINGS OF PROTECTION



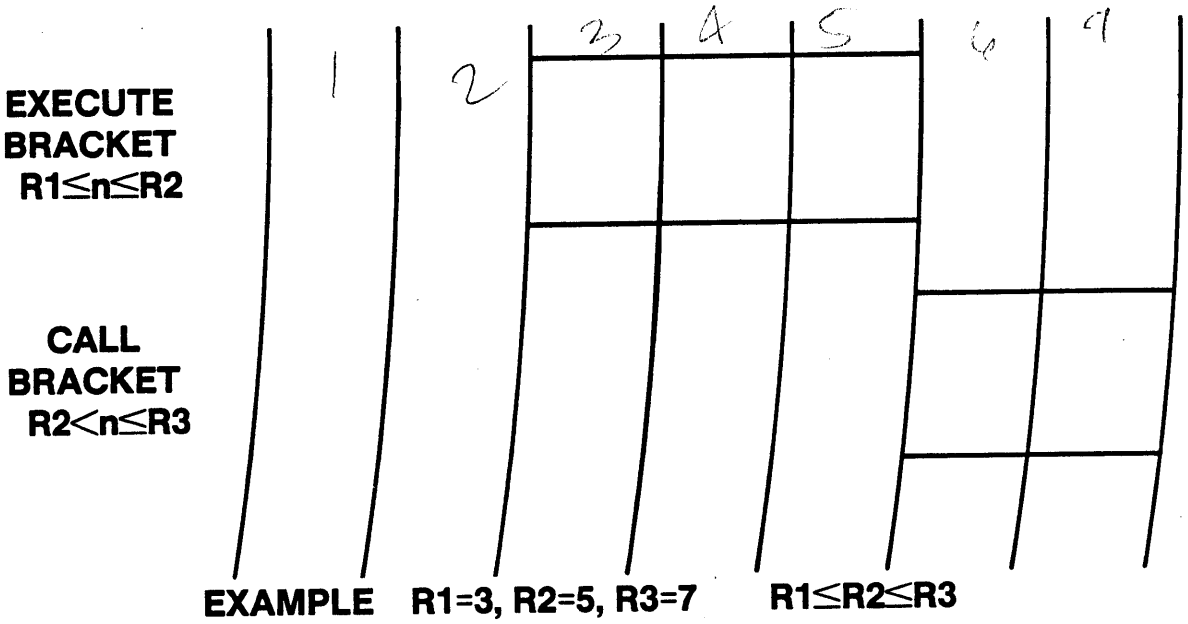
RINGS OF PROTECTION

- Operating System Rings 1–6
- Subsystems Rings 7–10
- Users' Rings 11–15
- Segment May Function in a Range of Rings (Ring Bracket)

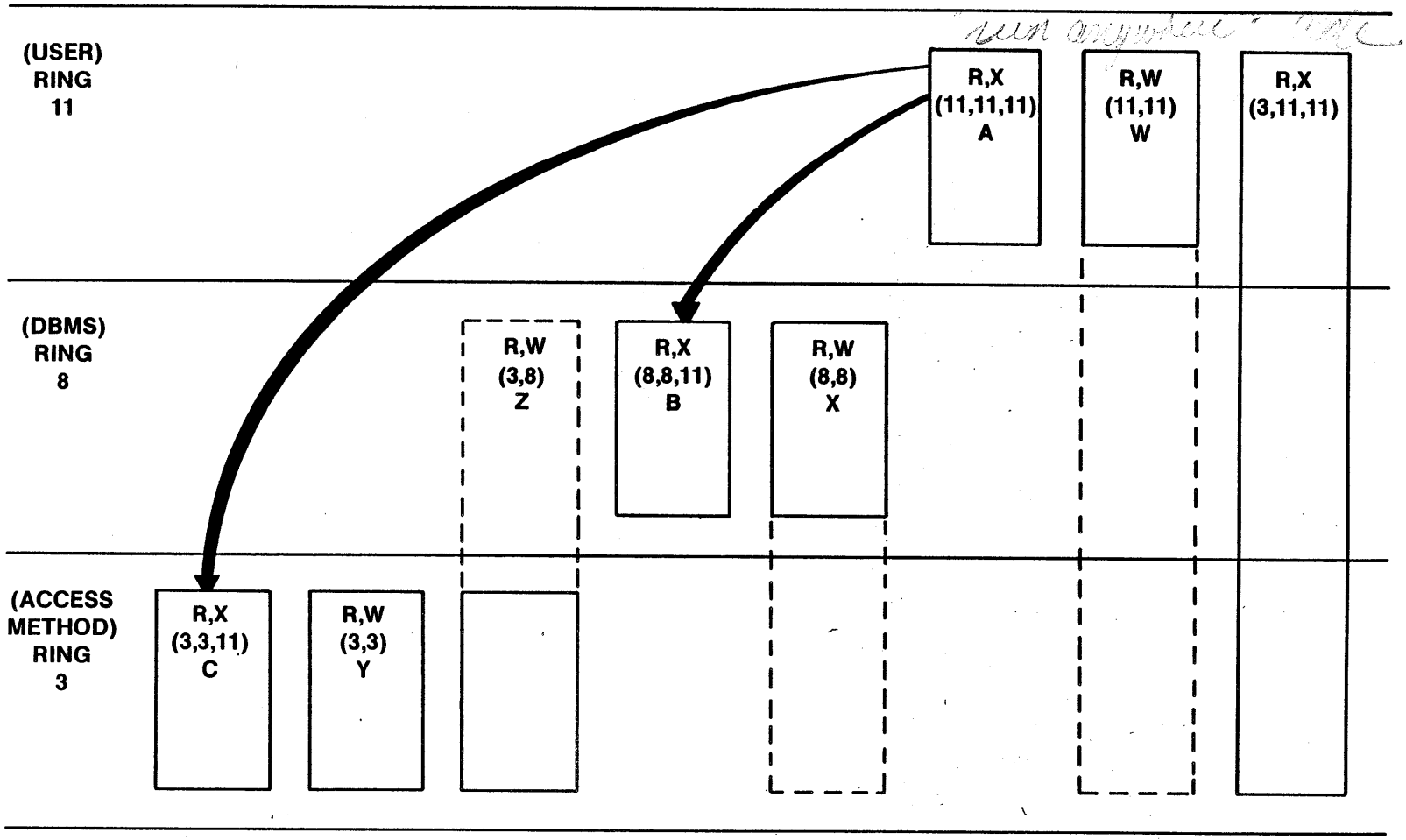
RING BRACKETS
DATA



RING BRACKETS
CODE



EXAMPLE RING BRACKETS



ACCESS PERMISSIONS
R: READ
W: WRITE
X: EXECUTE
RING ATTRIBUTES
(R1,R2,R3)

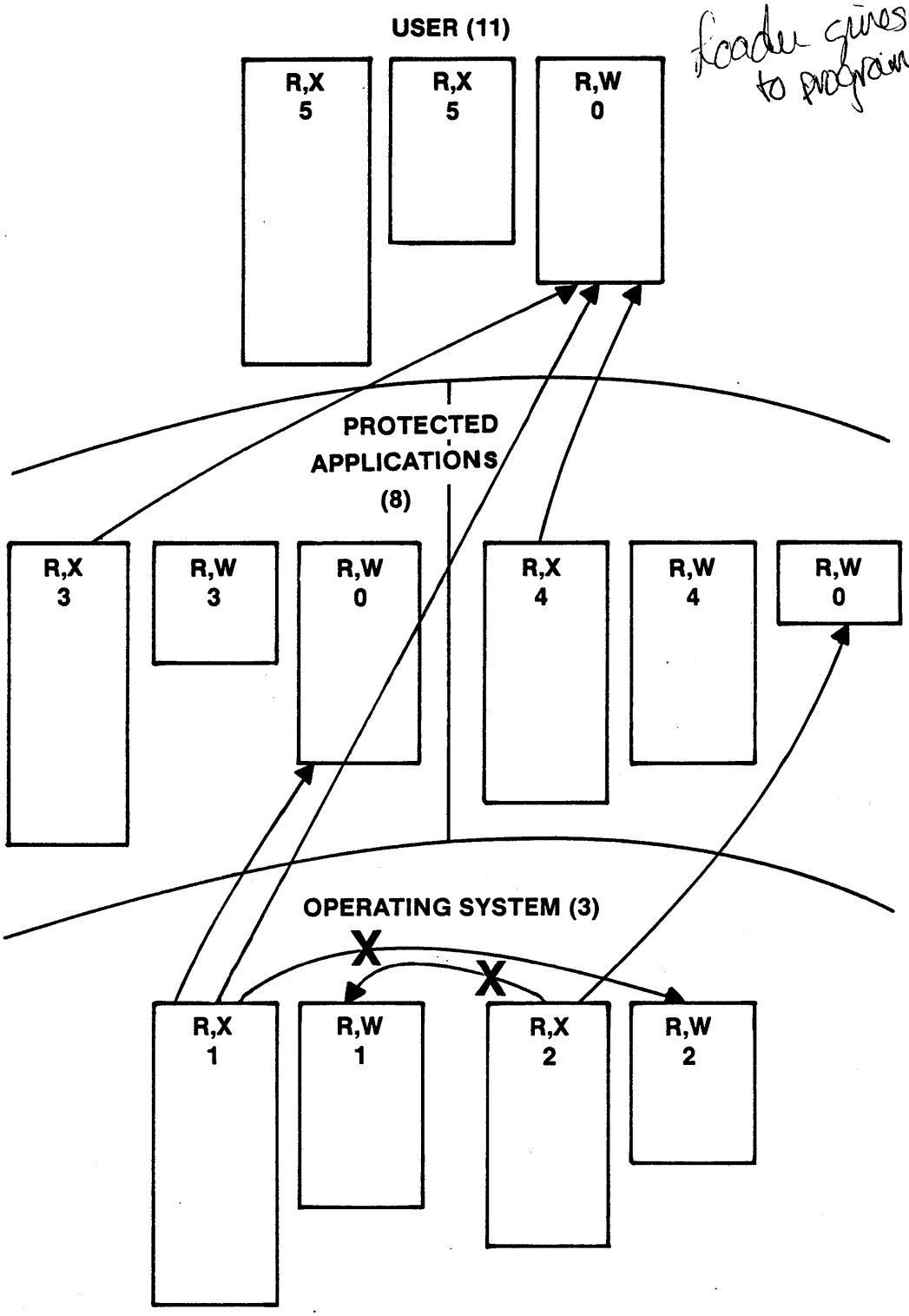
RING BRACKETS EXAMPLE

Segment	May Be Called By	Has Read Access To	Has Write Access To
A	A	W	W
B	A or B	W, X, or Z	W or X
C	A, B, or C	W, X, Y, or Z	W, X, Y, or Z

LOCK AND KEY PROTECTION

- Firewalls Subsystems in the Same Ring
- Isolates Data in Less Privileged Ring

LOCAL
LOCKS AND KEYS
DATA



GLOBAL AND LOCAL KEY/LOCK

- Single Key or Lock per Segment
  - — Key When Segment is Executing
  - — Lock When Segment is Referenced

PROCEDURE L	LOCAL
0	MASTER KEY
1	6 BIT KEY

DATA OR PROCEDURES L	
	NO LOCK
	6 BIT LOCK
0	
1	

- By using nonzero local locks, data can be restricted to be written or accessed by only "local" procedures. Normally, no procedure will have a master local key.
- User and system procedures are normally assigned a nonzero local key.

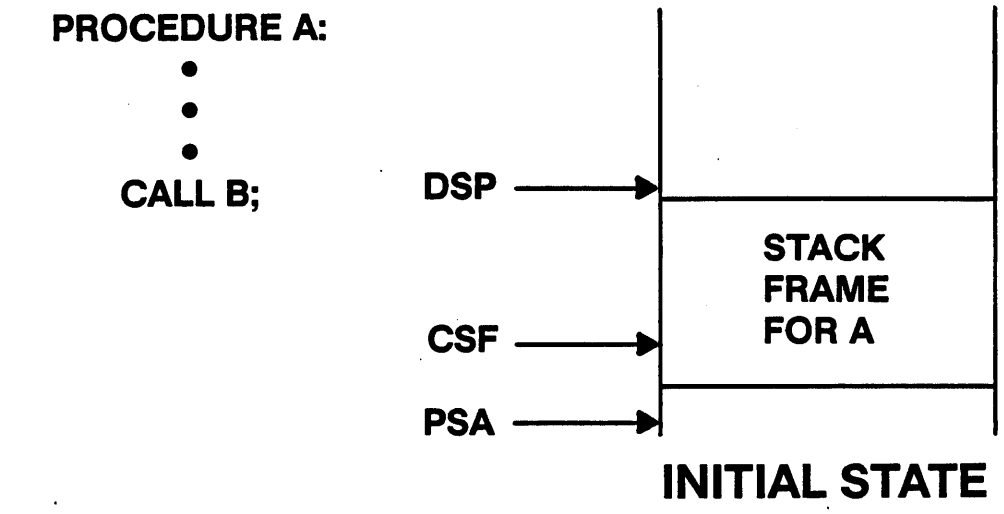
CALL/RETURN
INTERRUPTS

EXAMPLE OF BLOCK STRUCTURE

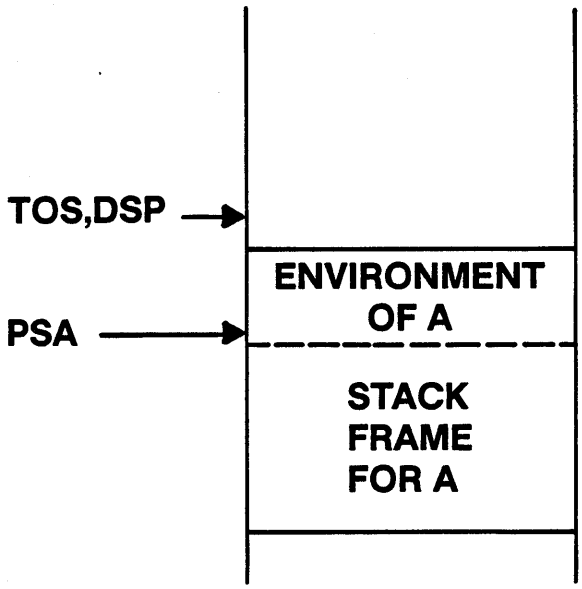
```
                6  procedure a;
                7      var
                8          i_a,
                9          j_a: integer;
               10      procedure b;
               11          var
               12              i_b: integer;
               13
               14          procedure c;
               15              var
               16                  i_c: integer,
               17                  j_a: boolean;
               18              i_c := i_b;
               19              if j_a then
               20                i_a := i_c;
               21              ifend;
               22              d; [Call procedure D]
               23          procend c;
               24          i_b := j_a;
*ERROR*        25          i_b := i_c;
               26          c; [Call procedure C]
               27      procend b;
               28
               29      procedure d;
               30          var
               31              i_d: integer;
               32
               33          procedure e;
               34              var
               35                  i_e: integer;
               36              i_e := i_d;
               37              i_e := i_a;
               38          procend e;
*ERROR*        39          c; [Call procedure C]
               40      procend d;
                       b; [Call procedure B]
                   procend a;
```

LINE NUMBER	SEVERITY LEVEL	ERROR MESSAGE
24	ERROR	Undeclared identifier - I_C.
37	ERROR	Undeclared identifier - C

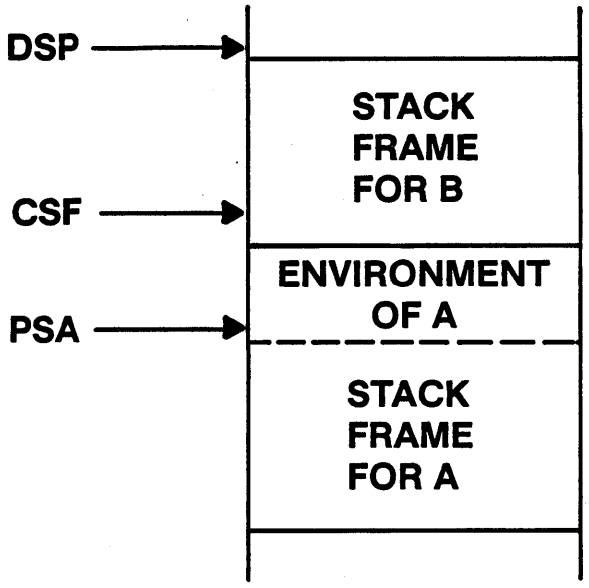
BASIC CALL MECHANISM



INITIAL STATE



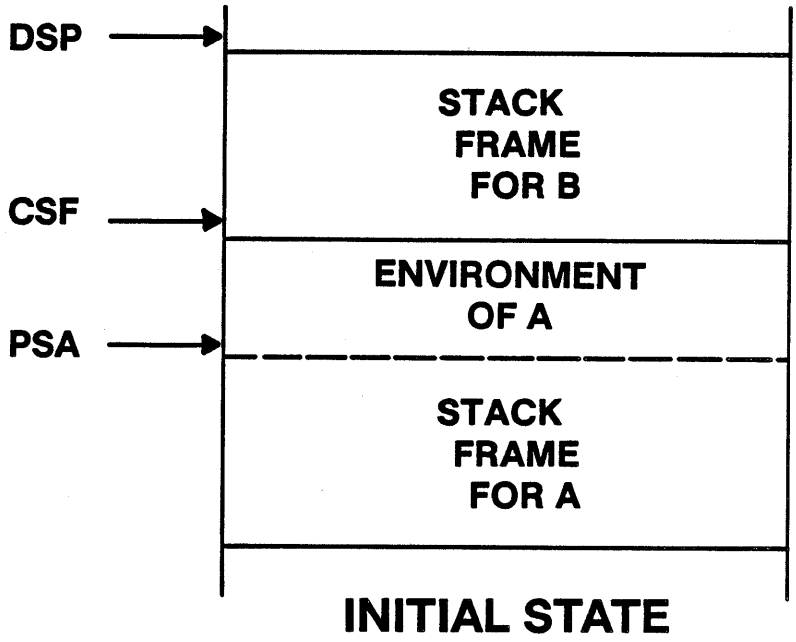
AFTER CALL ISSUED



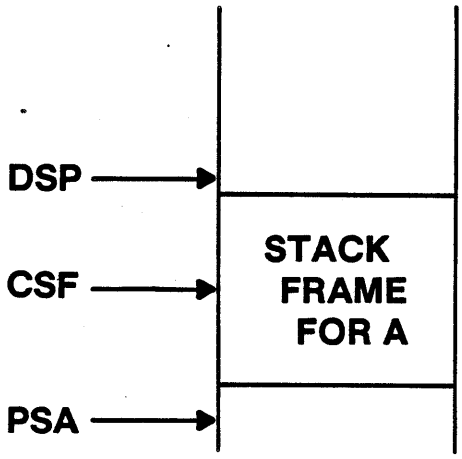
AFTER SOFTWARE CREATION OF STACK FRAME FOR B

BASIC RETURN MECHANISM

```
PROCEDURE B;
    •
    •
    •
RETURN
END
```

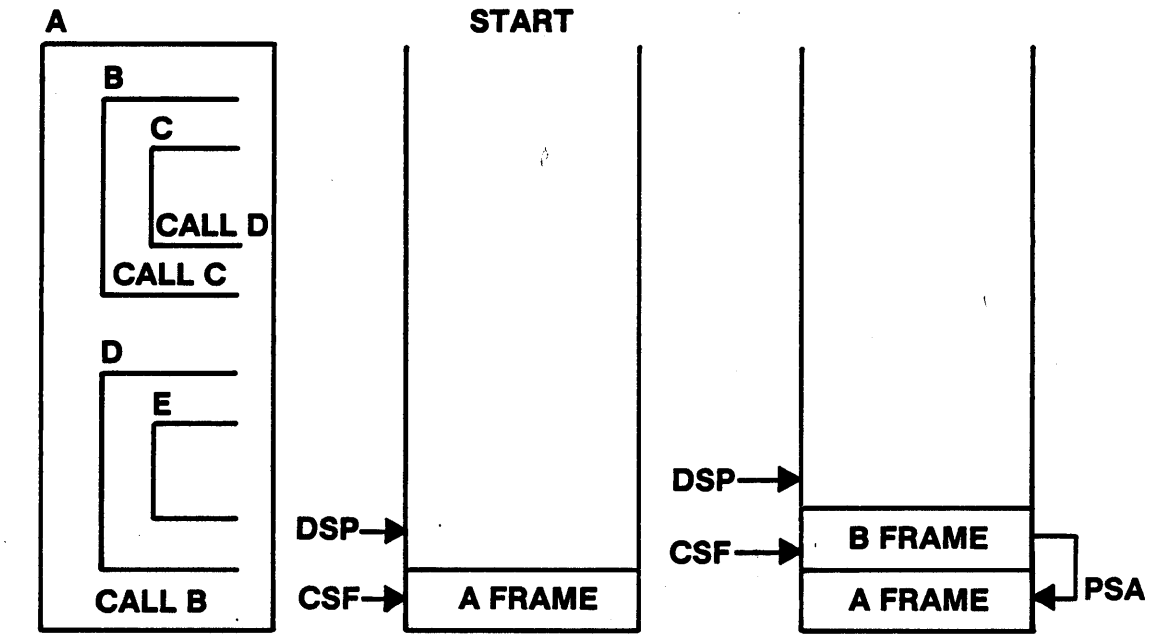


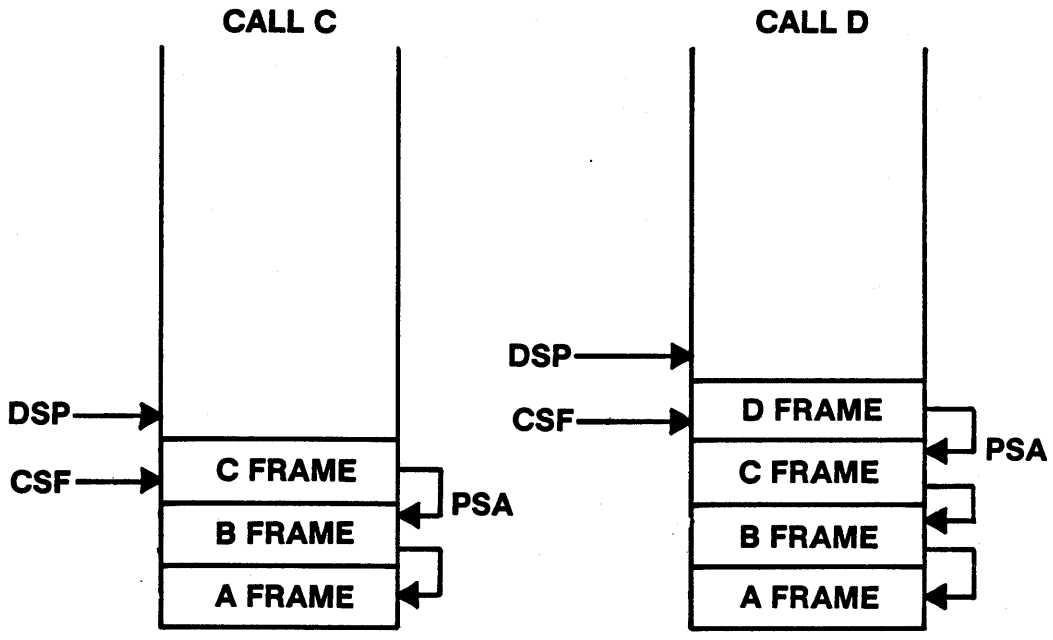
INITIAL STATE



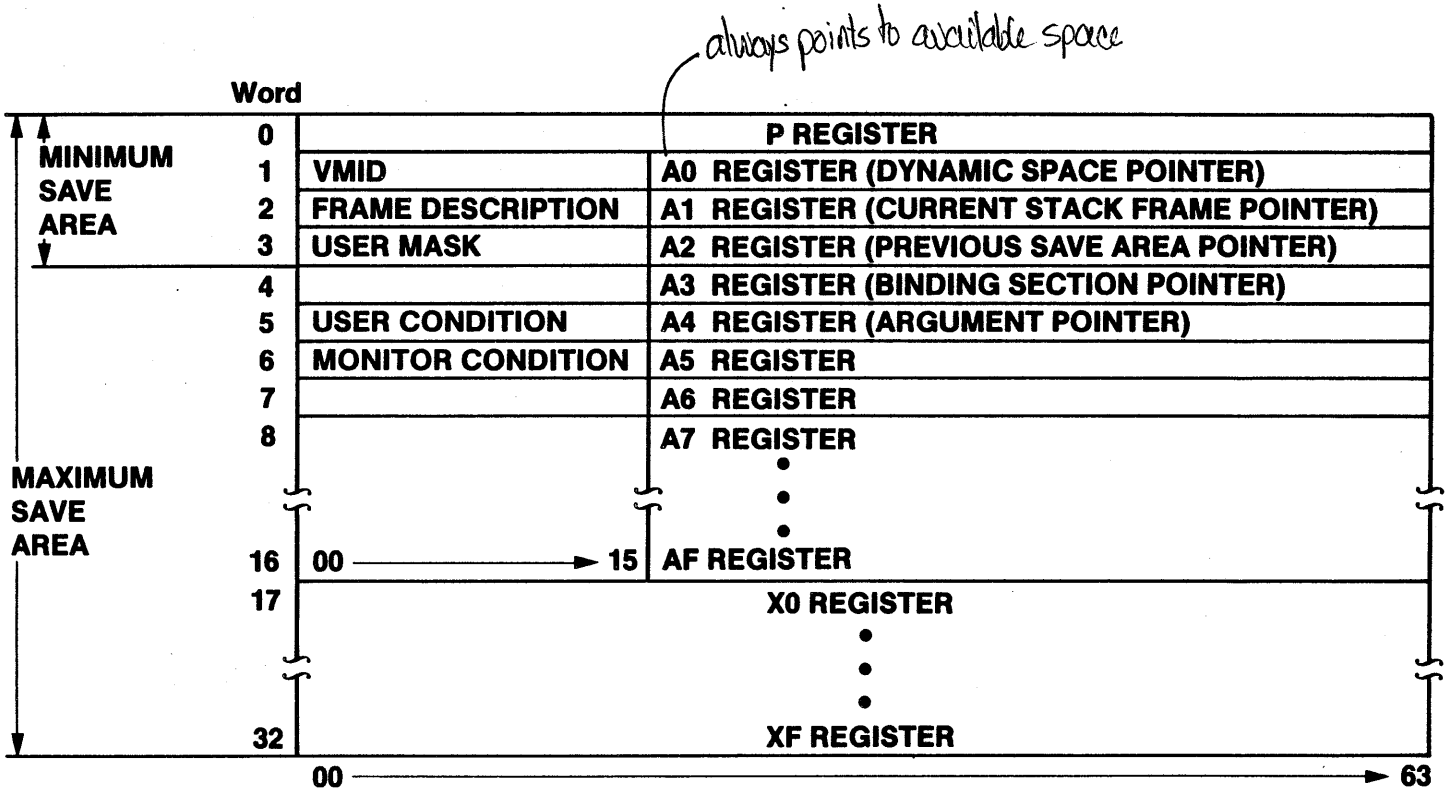
DSP, CSF and PSA are all reset from A's stack frame save area.

STACK FRAME MANIPULATION BY CALL/RETURN

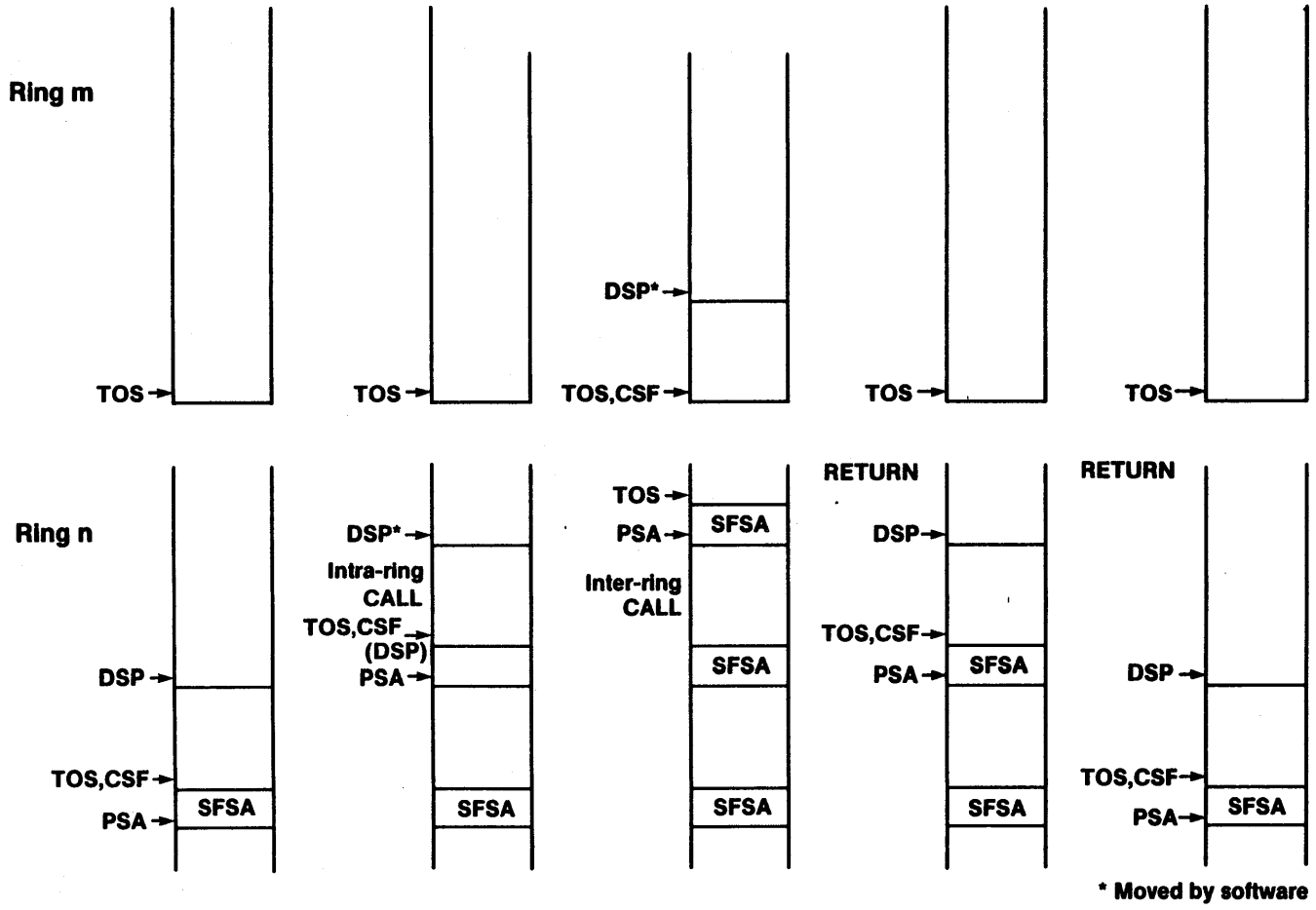




STACK FRAME SAVE AREA



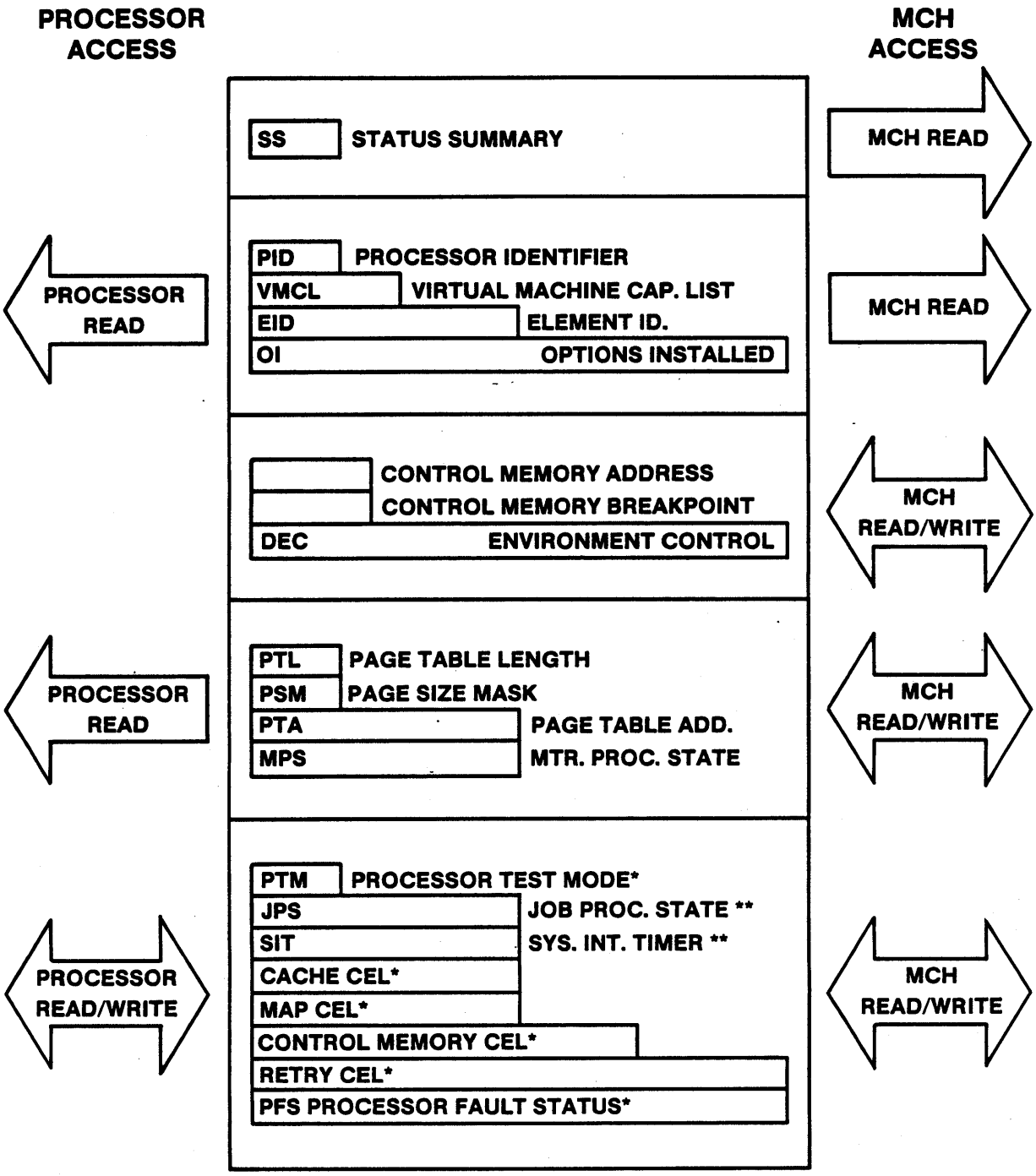
CALL/RETURN
Intra- and Inter-Ring



CYBER 180 REGISTERS

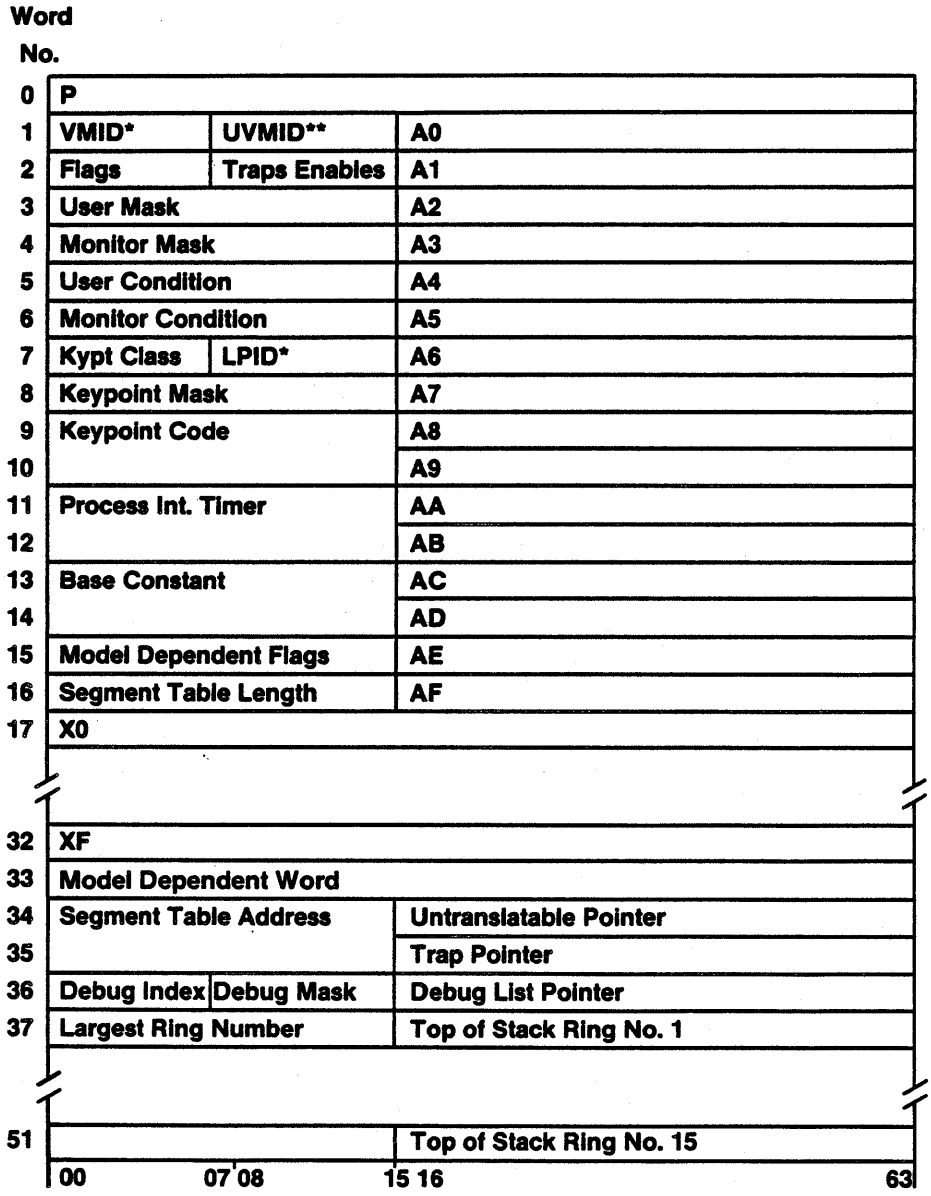
- Processor State Registers
  - — Either Hard-Wired, or
  - — Defined at Deadstart/Initialization
- Process State Registers
  - — Define an Exchange Interval
  - — Defined by an Exchange Package
  - — Either Live Registers, or
  - — Stored in Real Memory

PROCESSOR STATE REGISTERS



* WRITE IN GLOBAL PRIVILEGE MODE ONLY
** WRITE IN MONITOR MODE ONLY

EXCHANGE PACKAGE

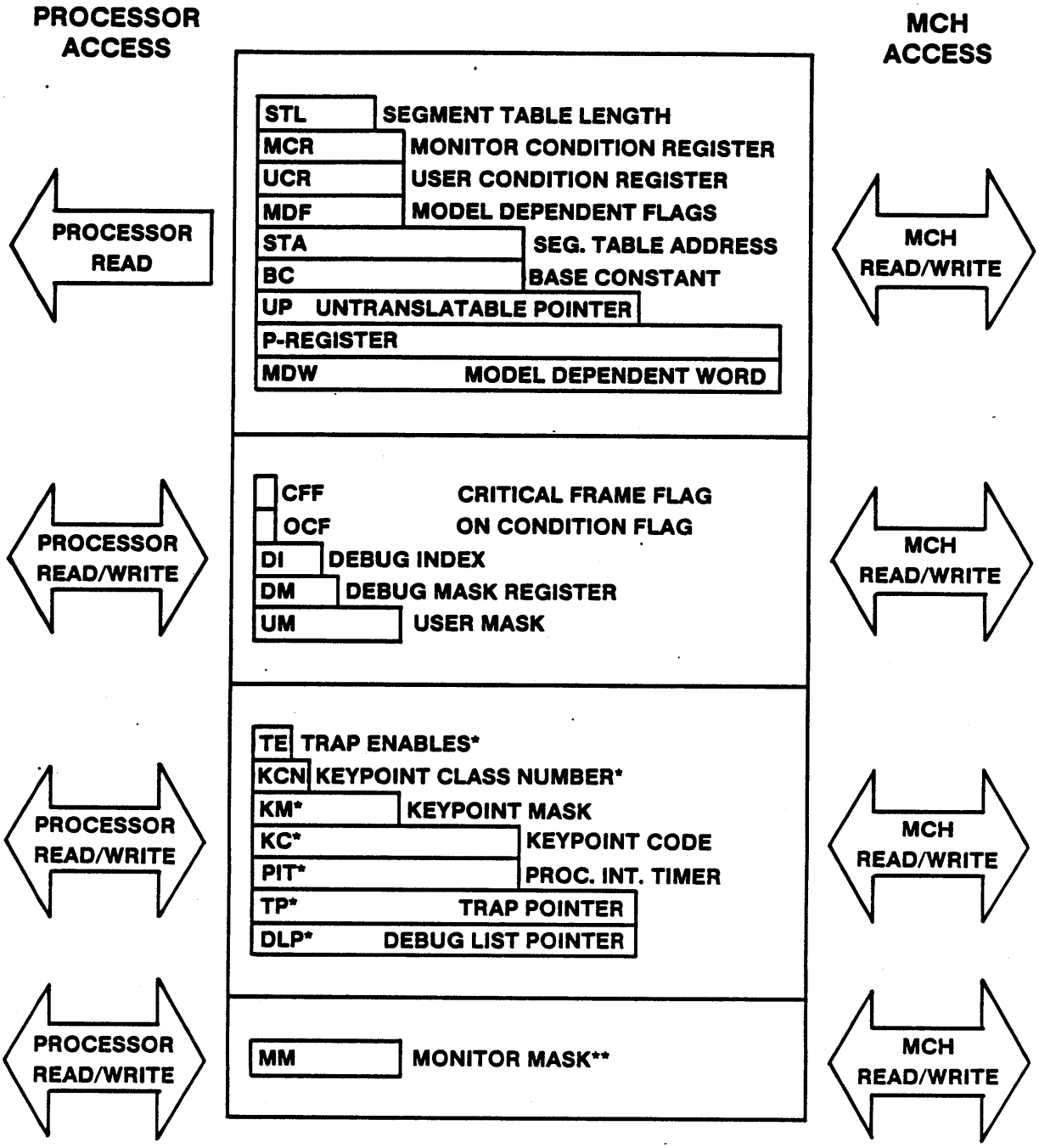


* Virtual Machine Identifier

** Untranslatable Virtual Machine Identifier

*** Last Processor Identification

PROCESS STATE REGISTERS



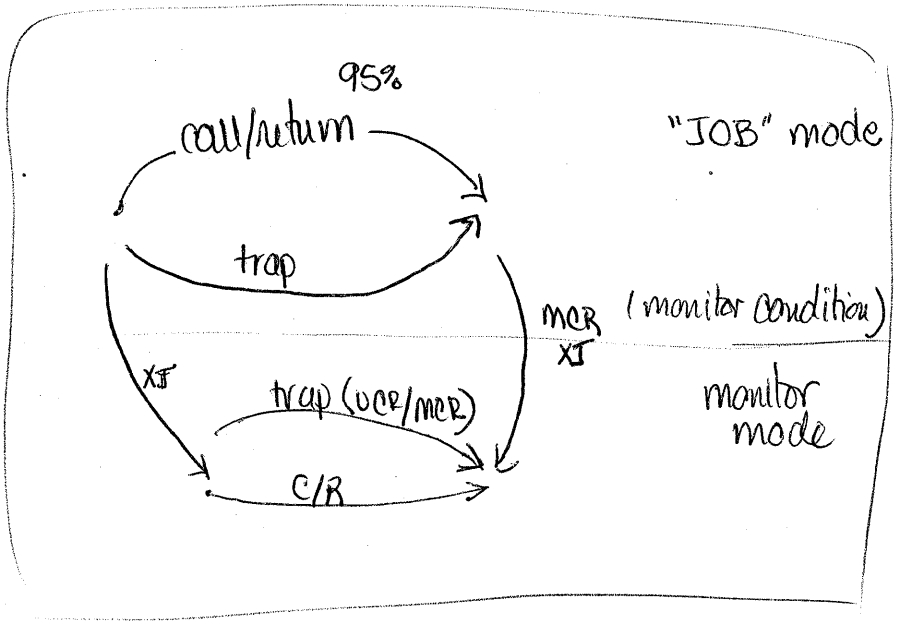
* WRITE IN LOCAL PRIVILEGED MODE ONLY

** WRITE IN MONITOR MODE ONLY

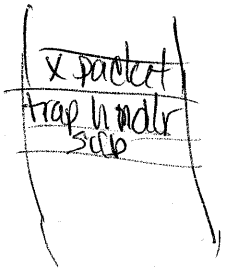
CONTROL TRANSFER

- Call/Return
- Exchange Jump—Change States
- Exchange Interrupt—Job State, Control Transfers to Monitor Address Space
- Trap Interrupt—Job/Monitor State, Processed Within Address Space of Current Process

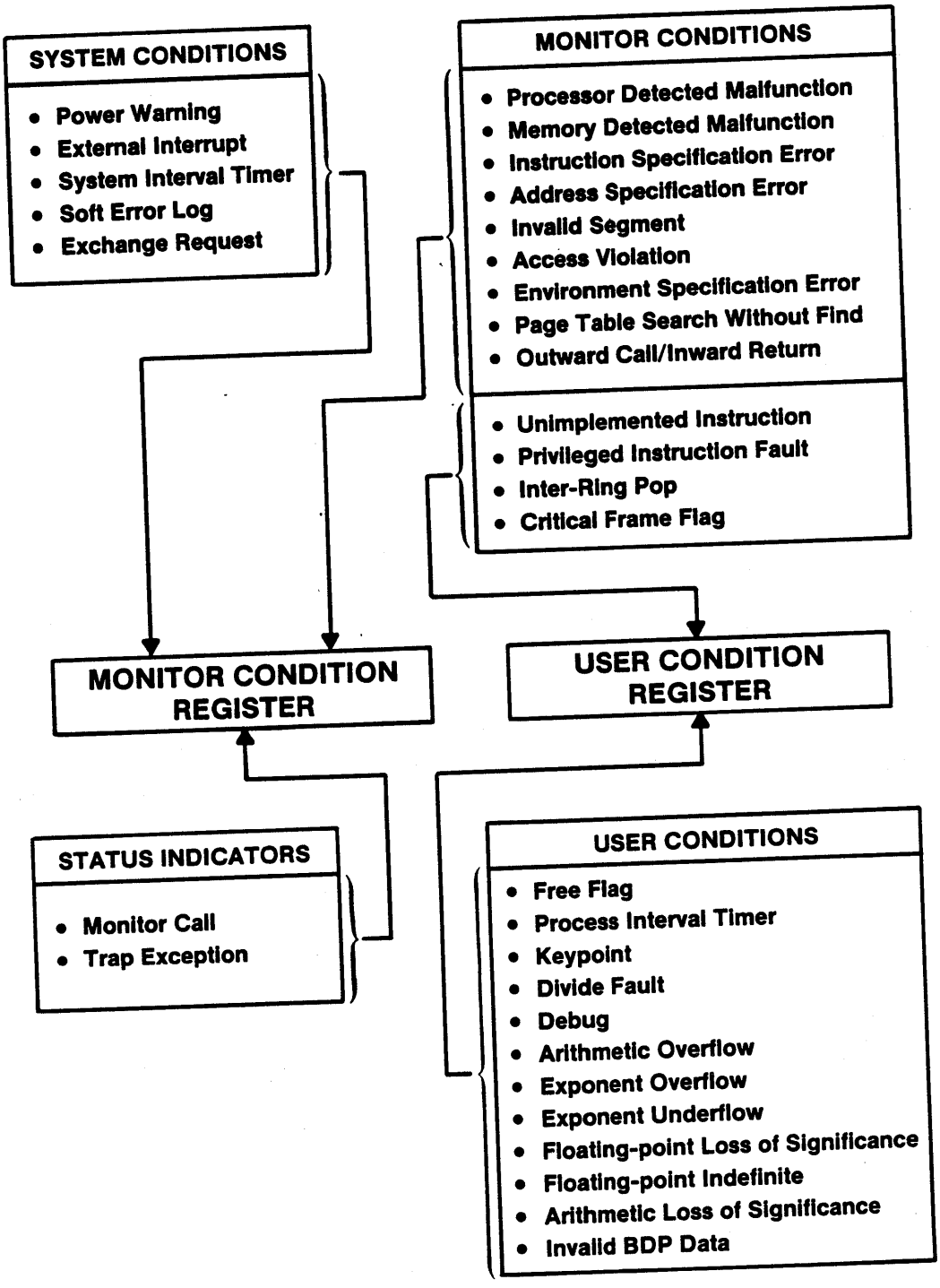
Communication on CIBO/ NOS/VE



trap - (call/return) the exchange packet is put to the stack. along with a trap handler.



INTERRUPT CONDITIONS



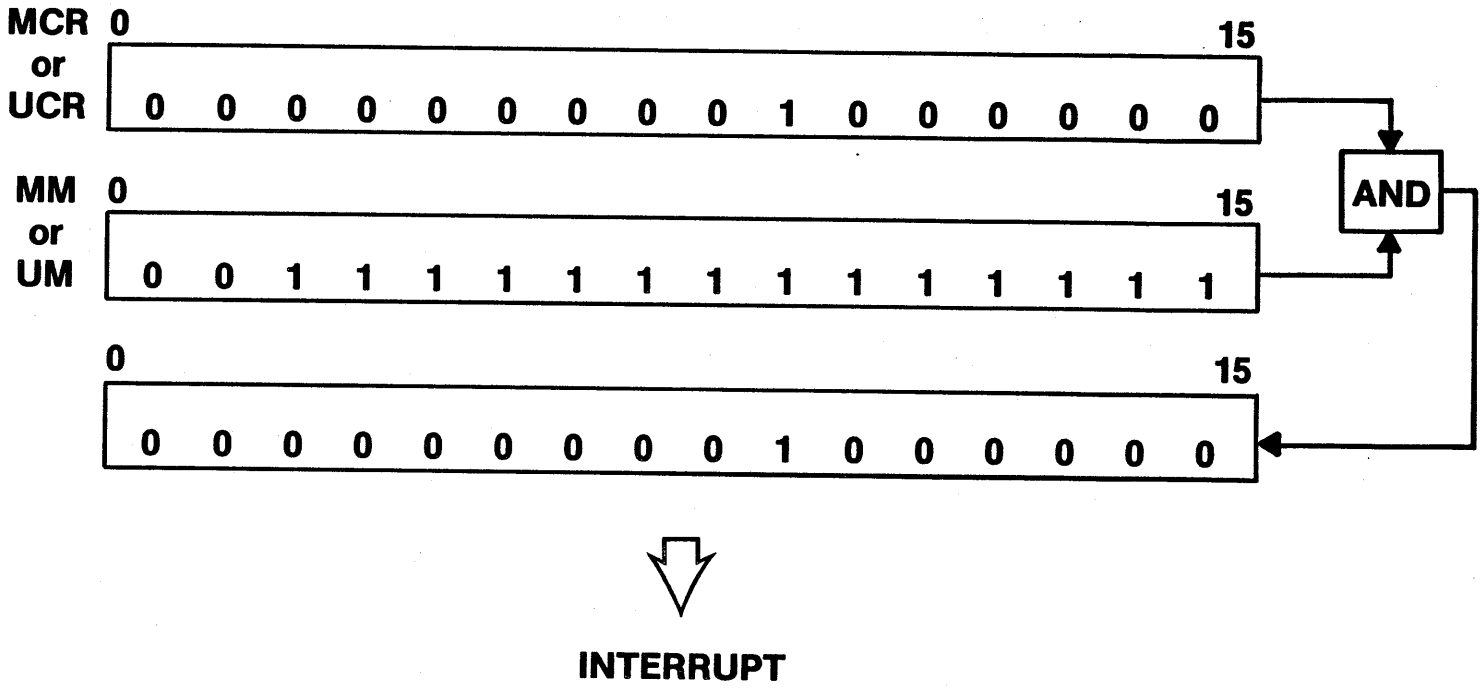
MONITOR CONDITION REGISTER

BIT NUMBER AND DEFINITION		TRAPS ENABLED		TRAPS DISABLED		
		TRAP ENABLE F/F SET AND TRAP ENABLE DELAY F/F CLEAR AND MASK BIT SET		TRAP ENABLE F/F CLEAR OR TRAP ENABLE DELAY F/F SET AND MASK BIT SET		MASK BIT CLEAR
		JOB MODE	MONITOR MODE	JOB MODE	MONITOR MODE	JOB OR MONITOR MODE
0 Processor Detected Malfunction	Mon	EXCH	TRAP	EXCH	HALT	HALT
1 Memory Detected Malfunction	Mon	EXCH	TRAP	EXCH	HALT	HALT
2 Power Warning	Sys	EXCH	TRAP	EXCH	STACK	STACK
3 Instruction Specification Error	Mon	EXCH	TRAP	EXCH	HALT	HALT
4 Address Specification Error	Mon	EXCH	TRAP	EXCH	HALT	HALT
5 Exchange Request	Sys	EXCH	TRAP	EXCH	STACK	STACK
6 Access Violation	Mon	EXCH	TRAP	EXCH	HALT	HALT
7 Environment Specification Error	Mon	EXCH	TRAP	EXCH	HALT	HALT
8 External Interrupt	Sys	EXCH	TRAP	EXCH	STACK	STACK
9 Page Table Search Without Find	Mon	EXCH	TRAP	EXCH	HALT	HALT
10 System Call	Status—This bit is a flag only and does not cause any hardware action.					
11 System Interval Timer	Sys	EXCH	TRAP	EXCH	STACK	STACK
12 Invalid Segment	Mon	EXCH	TRAP	EXCH	HALT	HALT
13 Outward Call/Inward Return	Mon	EXCH	TRAP	EXCH	HALT	HALT
14 Soft Error Log	Sys	EXCH	TRAP	EXCH	STACK	STACK
15 Trap Exception	Status—This bit is a flag only and does not cause any hardware action.					

USER CONDITION REGISTER

BIT NUMBER AND DEFINITION			TRAPS ENABLED		TRAPS DISABLED		
			TRAP ENABLE F/F SET AND TRAP ENABLE DELAY F/F CLEAR AND MASK BIT SET		TRAP ENABLE F/F CLEAR OR TRAP ENABLE DELAY F/F SET AND MASK BIT SET		MASK BIT CLEAR
			JOB MODE	MONITOR MODE	JOB MODE	MONITOR MODE	JOB OR MONITOR MODE
0	Privileged Instruction Fault	Mon	TRAP	TRAP	EXCH	HALT	These mask bits are permanently set.
1	Unimplemented Instruction	Mon	TRAP	TRAP	EXCH	HALT	
2	Free Flag	User	TRAP	TRAP	STACK	STACK	
3	Process Interval Timer	User	TRAP	TRAP	STACK	STACK	
4	Inter-ring Pop	Mon	TRAP	TRAP	EXCH	HALT	
5	Critical Frame Flag	Mon	TRAP	TRAP	EXCH	HALT	
6	Keypoint	User	TRAP	TRAP	STACK	STACK	
7	Divide Fault	User	TRAP	TRAP	STACK	STACK	STACK
8	Debug	User	TRAP	TRAP	Debug bit will not set when traps disabled.		
9	Arithmetic Overflow	User	TRAP	TRAP	STACK	STACK	STACK
10	Exponent Overflow	User	TRAP	TRAP	STACK	STACK	STACK
11	Exponent Underflow	User	TRAP	TRAP	STACK	STACK	STACK
12	F. P. Loss of Significance	User	TRAP	TRAP	STACK	STACK	STACK
13	F. P. Indefinite	User	TRAP	TRAP	STACK	STACK	STACK
14	Arithmetic Loss of Significance	User	TRAP	TRAP	STACK	STACK	STACK
15	Invalid BDP Data	User	TRAP	TRAP	STACK	STACK	STACK

INTERRUPT MECHANISM



INSTRUCTIONS

INSTRUCTIONS CLASSIFICATIONS

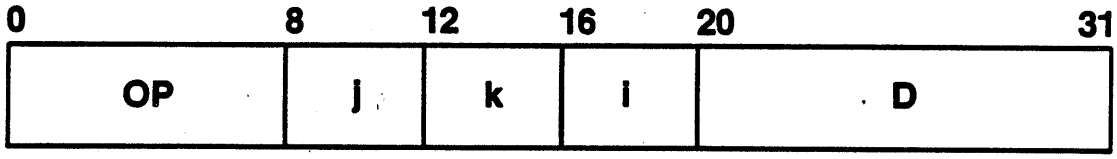
- General Instructions
  - — Load/Store
  - — Integer Arithmetic
  - — Branch
  - — Copy
  - — Shift
  - — Logical
- Business Data Processing
  - — BDP Numeric (Arithmetic, Scale, Decimal Compare, Numeric Move)
  - — Byte (Compare, Scan, Translate, Move, Edit)
  - — Calculate Subscript
  - — Immediate Data (Move, Compare, Add)
- Floating Point
  - — Conversion
  - — Arithmetic
  - — Branch
- System Instructions
  - — Call/Return
  - — Exchange
  - — Load Page Table Index
  - — Purge Buffers
  - — Copy State Registers

INSTRUCTION PRIVILEGE

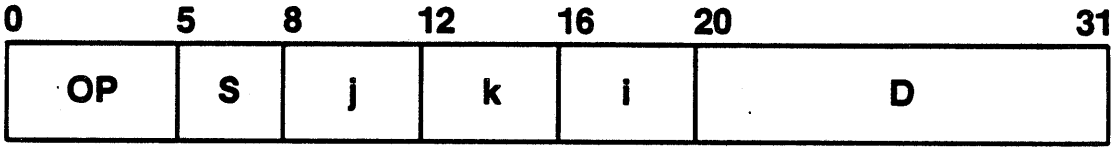
- Nonprivileged
- Local Privileged
- Global Privileged
- Monitor

OP CODE FORMATS

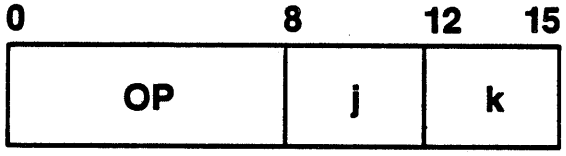
jkiD INSTRUCTION FORMAT



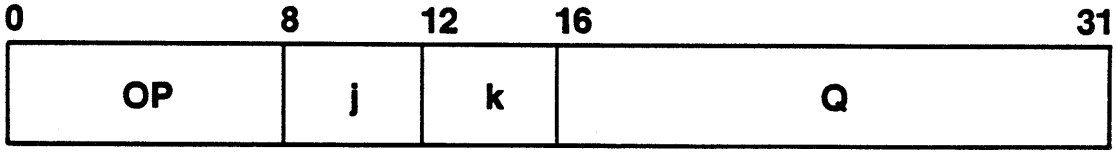
SjkiD INSTRUCTION FORMAT



jk INSTRUCTION FORMAT

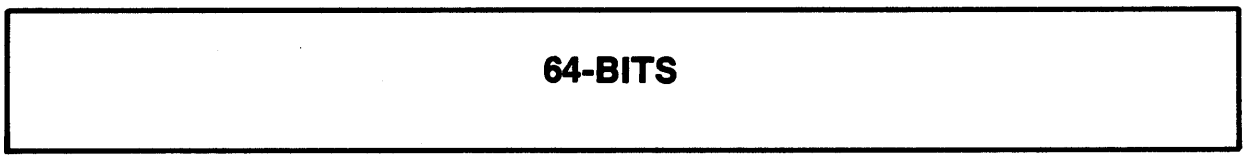


jkQ INSTRUCTION FORMAT

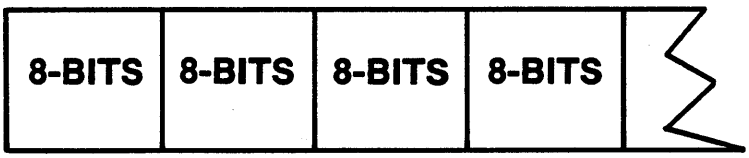


GENERAL DATA FORMATS

32-BIT, SIGNED, MAXIMUM VALUE 2,147,483,648

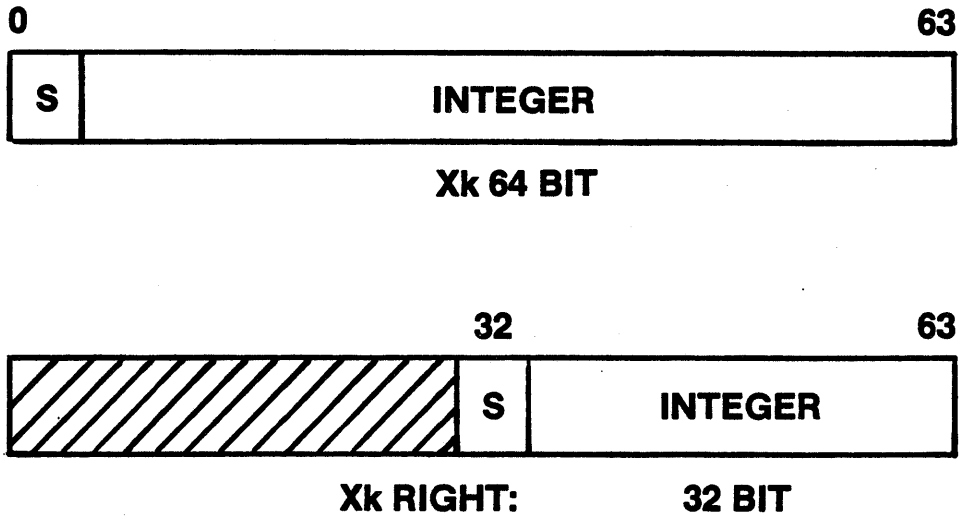


64-BIT, SIGNED, MAXIMUM VALUE 2.3 * 10 ** 18

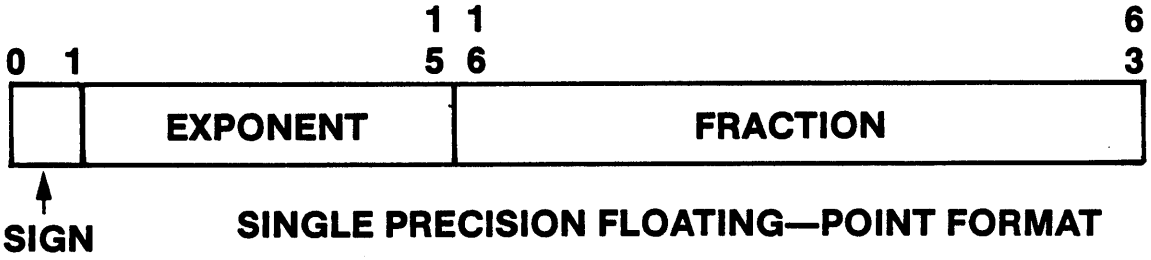


8-BIT ASCII CHARACTERS (BYTES)

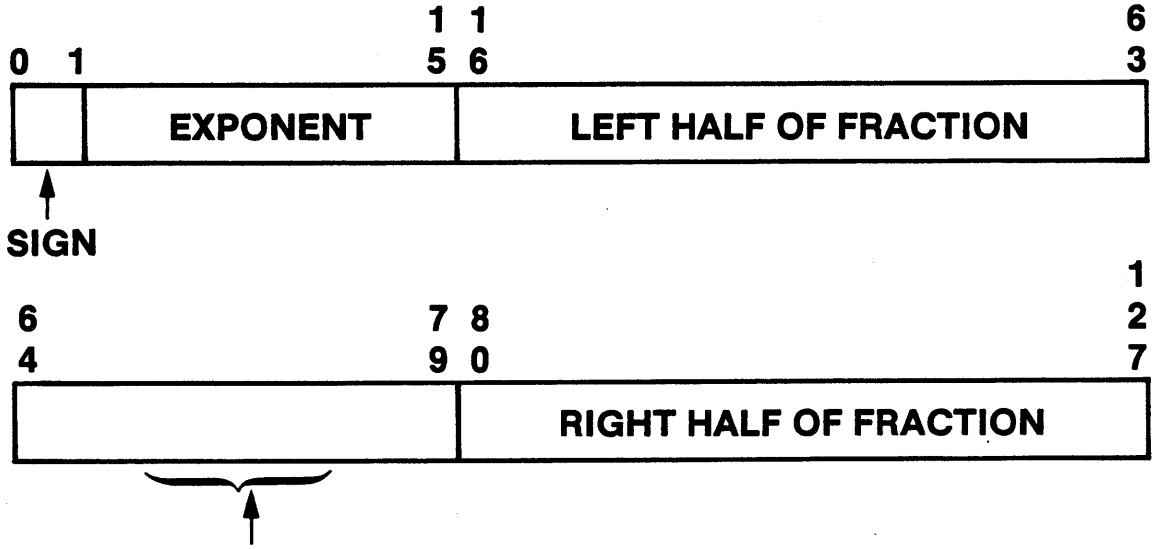
INTEGER ARITHMETIC INSTRUCTION



FLOATING POINT FORMATS



SINGLE PRECISION FLOATING—POINT FORMAT



NOT USED AS INPUT, BUT PROVIDED ON OUTPUT

DOUBLE PRECISION FLOATING-POINT FORMAT

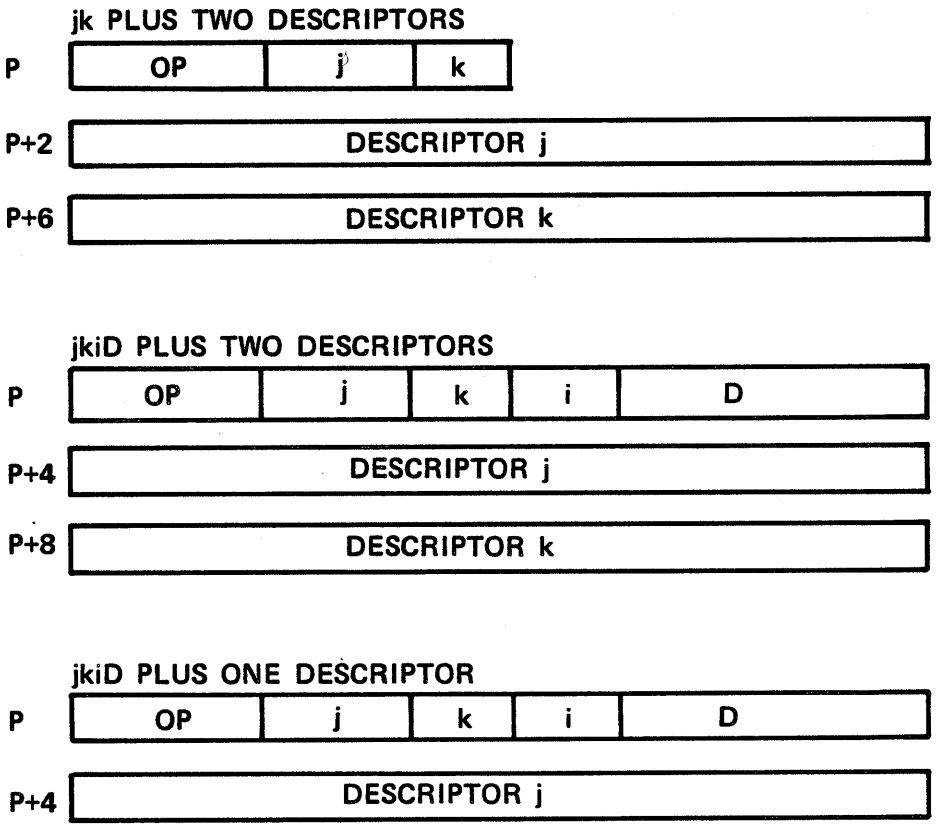
FLOATING POINT REPRESENTATION

	HEXADECIMAL EXPONENT INCLUDING COEFFICIENT SIGN	ACTUAL EXPONENT (TO THE BASE 2)	INPUT ARGUMENTS	RESULTS
	7XXX	- - - -	INDEFINITE	7000.0 ⟶ 0
COEFFICIENT SIGN EQUAL TO 0 (POSITIVE NUMBERS)	6FFF ↑ 5000	$2^{12,287}$ ↑ $2^{4,096}$	INFINITE	OVERFLOW MASK = 0 : 5000.0 ⟶ 0 OVERFLOW MASK = 1 : AS SHOWN
	4FFF ↑ 4000 3FFF ↓ 3000	2^{4095} ↑ 2^{0} 2^{-1} ↓ $2^{-4,096}$	STANDARD	AS SHOWN
	2FFF ↓ 1000	$2^{-4,097}$ ↓ $1^{-12,288}$	ZERO	UNDERFLOW MASK = 0 : 0000.0 ⟶ 0 UNDERFLOW MASK = 1 : AS SHOWN
	0XXX	— — —	ZERO	NOT APPLICABLE
	8XXX			
COEFFICIENT SIGN EQUAL TO 1 (NEGATIVE NUMBER)	9000 ↑ AFFF	$2^{-12,288}$ ↑ $1^{-4,097}$	ZERO	UNDERFLOW MASK = 0 : 0000.0 ⟶ 0 UNDERFLOW MASK = 1 : AS SHOWN
	B000 ↑ BFFF C000 ↓ CFFF	$2^{-4,096}$ ↑ 2^{-1} 2^{0} ↓ $2^{4,095}$	STANDARD	AS SHOWN
	D000 ↓ EFFF	2^{4096} ↓ $2^{12,287}$	INFINITE	OVERFLOW MASK = 0 : D000.0 0 OVERFLOW MASK = 1 : AS SHOWN
	FXXX	- - -	INDEFINITE	7000.0 ⟶ 0

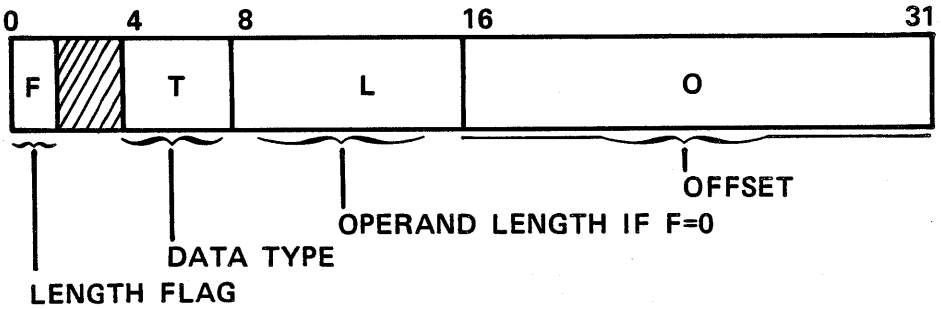
BDP INSTRUCTIONS

SUBGROUP	SHORT NAME
BDP NUMERIC	SUM DIFFERENCE PRODUCT QUOTIENT SCALE SCALE ROUNDED DECIMAL COMPARE NUMERIC MOVE
BYTE	COMPARE COMPARE COLLATED SCAN WHILE NON-MEMBER TRANSLATE MOVE BYTES EDIT
SUBSCRIPT	CALCULATE SUBSCRIPT
IMMEDIATE DATA	MOVE IMMEDIATE DATA COMPARE IMMEDIATE DATA ADD IMMEDIATE DATA

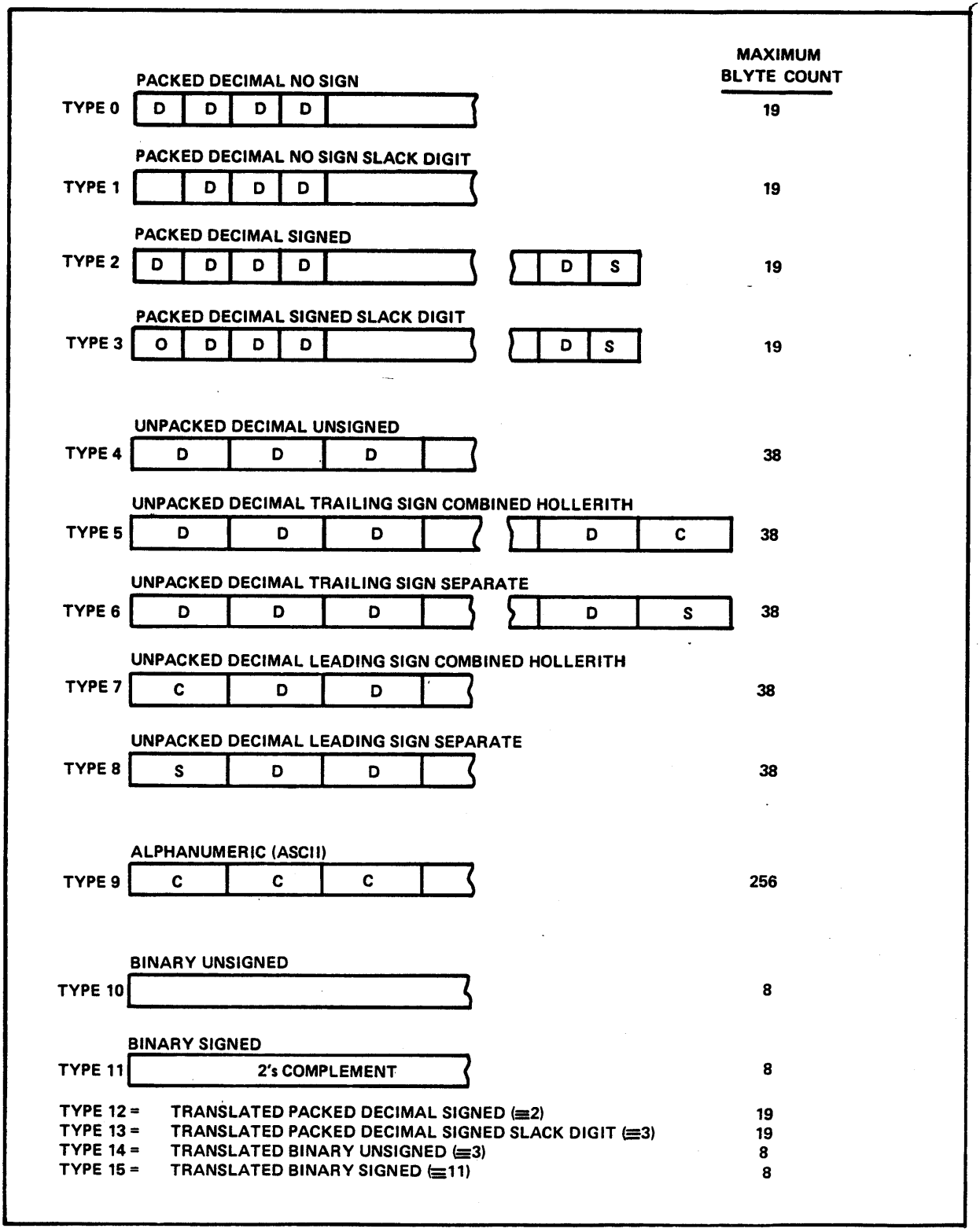
BDP OPERATION CODES



BDP DESCRIPTOR



BDP DATA TYPES



CHAPTER 3:
NOS/VE CONCEPTS

PREVIEW

CYBIL Introduction
NOS/VE Structure
Job Flow
NOS/VE Features
NOS/VE Products

REFERENCES

OBJECTIVES

- Explain the importance of using a high-level implementation language.
- Explain the relationship between job and task.
- Explain how access privilege and rings work together to protect code and data.
- Outline the flow of a job through the system.
- Explain how a user gets a new task started.
- List the functions that NOS performs for NOS/VE in a dual state system.
- List the features of the permanent file system.
- Summarize the features of the basic access method. State how a user can make requests to use these facilities.
- List the resources that the user can control.
- List the features of the interactive facility.
- State the uses of the system command language. List its characteristics.
- Distinguish between local and remote operators.
- List the products included in the current and future releases.

NOS/VE OBJECTIVES

Objectives	Strategies
RAM	Hardware
Configurability	SASD
Expandability	CYBIL
Usability	Standards
Consistency	Command Interface
Efficiency	Program Interface
Security	Dual State
Migration Ease	CP Operating System
	Code Isolation
	System Using Itself
	On-Line Development

IMPLEMENTATION LANGUAGE

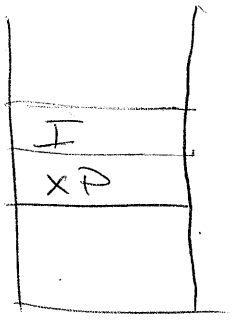
CYBIL

- Developed by Control Data
- Based on PASCAL
- Extensive Bounds Checking (type, data structures)
- Related to Hardware
- Communication Vehicle

- block structured, modern language.

A CYBIL PROGRAM

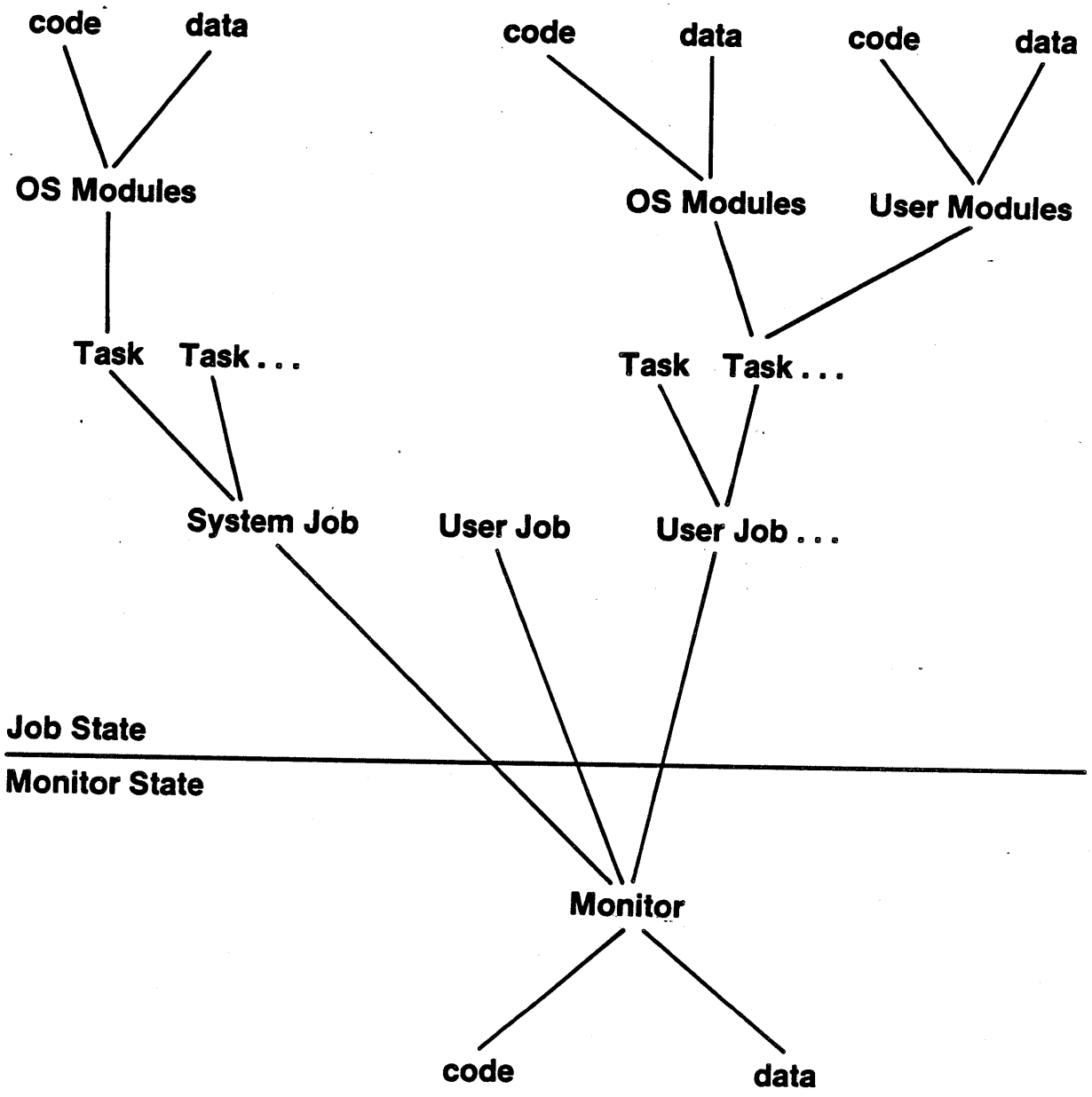
```
MODULE SAMPLE;
  TYPE RANGE=0..20;
  VAR  R:INTEGER,
    COUNT:RANGE;

  PROGRAM MAIN;
    VAR  A:INTEGER;
    PROCEDURE POWER;
      VAR  I:RANGE;
      R:=1;
      FOR I:=1 TO COUNT DO;
        R:=2*R;
      FOREND;
    PROCEND POWER;

    { EXECUTABLE STATEMENTS }
    COUNT:=5;
    POWER;
    A:=R;

  PROCEND MAIN;
MODEND
```



Stack

NOS/VE STRUCTURE

OS HIERARCHY



1. accounting
2. scheduling
3. recovery
4. tasks

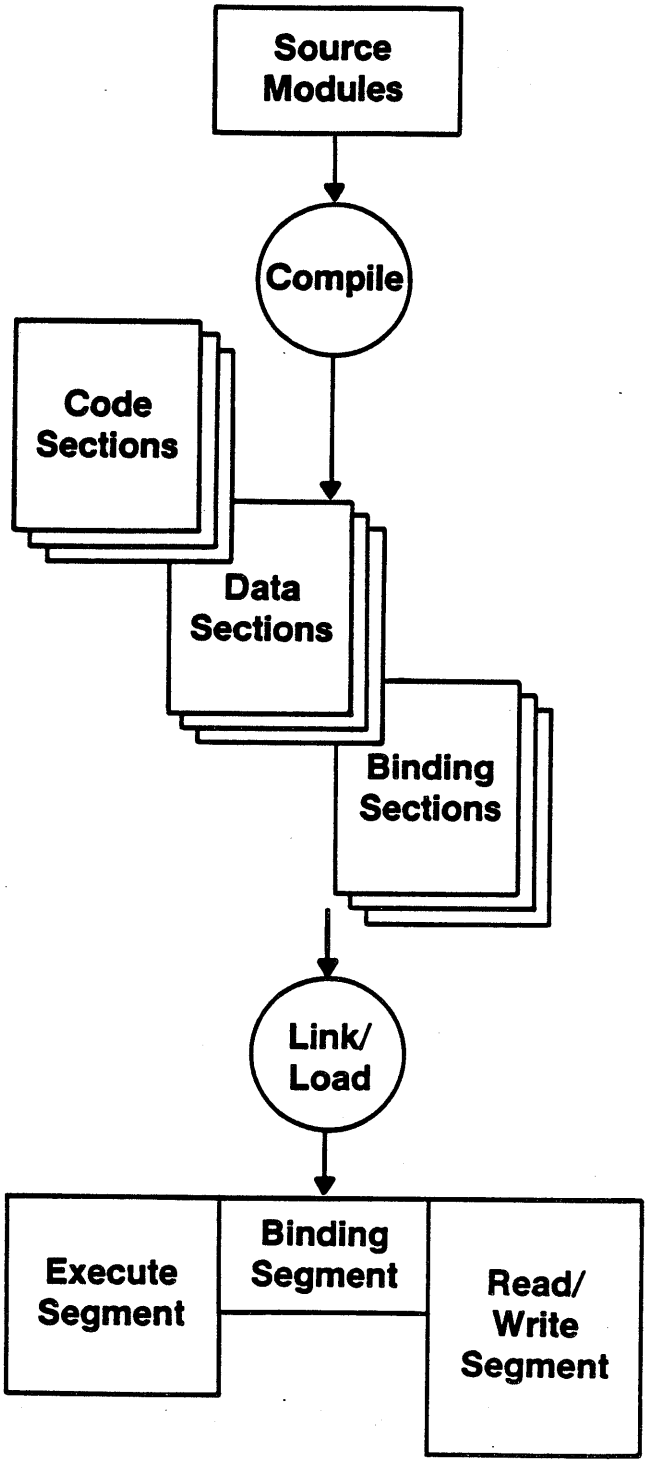
JOBS

- A job is the basic unit of work presented to NOS/VE.
- The job is the basic unit for:
  - — Validation
  - — Resource allocation
  - — Scheduling
  - — Activity logging and accounting
  - — Recovery
- Every job consists of multiple tasks for:
  - — Protection
  - — Serialization
  - — Multiprogramming

TASKS

- A program is a set of modules organized to perform a specific function
- A task is the execution of a program:
  - — Active tasks are protected from one another and compete for CPU resources
  - — A job may invoke multiple asynchronous tasks
  - — Tasks can communicate to synchronize their activities via Job local queues
- Each task has:
  - — An execution control block defining current execution status (includes exchange package)
  - — A segment descriptor table defining the task address space
  - — A priority

CREATE VM SEGMENTS



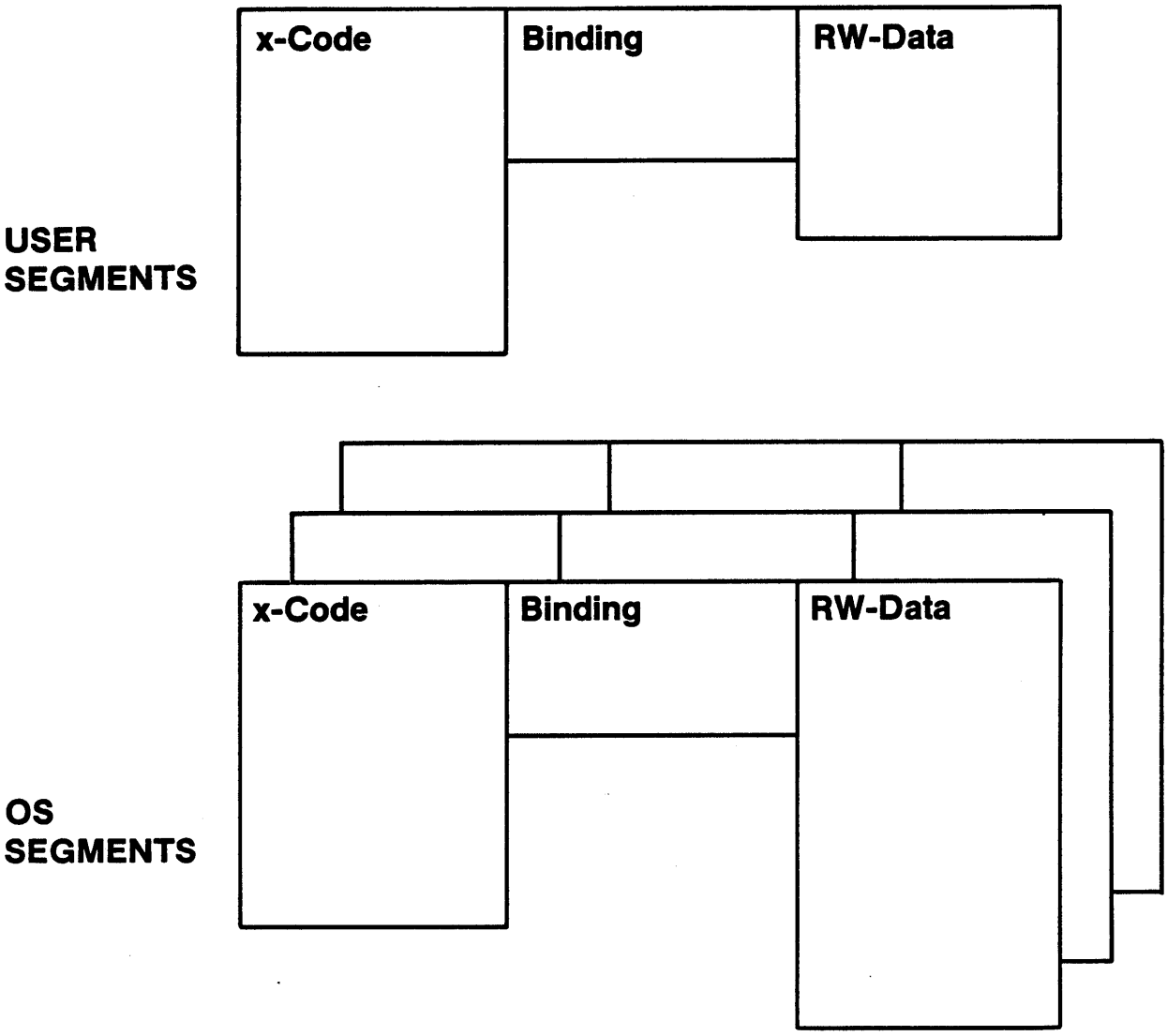
LINK PROGRAM

1. User prepares source program.
2. Compile the program. The compiler creates a set of sections for each module. Typically the sections are code, data, and binding.
3. Link the object modules (sections). The linker will collect all code sections into one execute segment, all data sections into one read/write segment, and so on. This will be done when an object library is generated. It will also be done when a task is executed, if necessary.
4. When a task is executed, a segment descriptor table (SDT) is built. It will contain an entry for all segments needed by the program—that is, user and system segments.

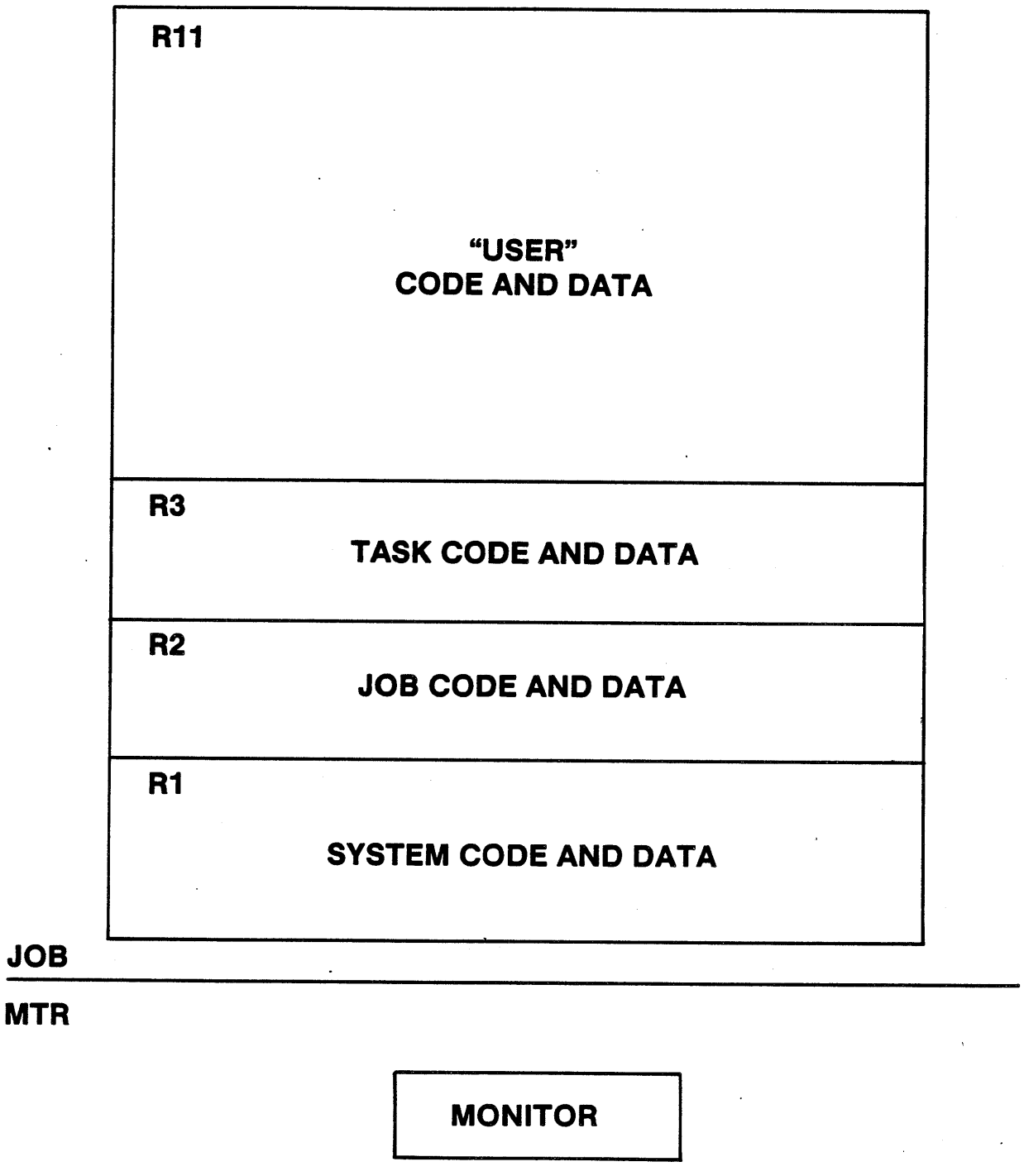
BINDING SEGMENT

- All linkage between segments uses binding:
  - — Read
  - — Write
  - — Call
- Purposes
  - — To enable code sharing, the re-entrant code must be able to access multiple data segments.
  - — To allow inward calls, the call bracket must be checked. The linker stores the R3 value in the binding segments.
  - — User programs must be linked into an existing linked operating system.

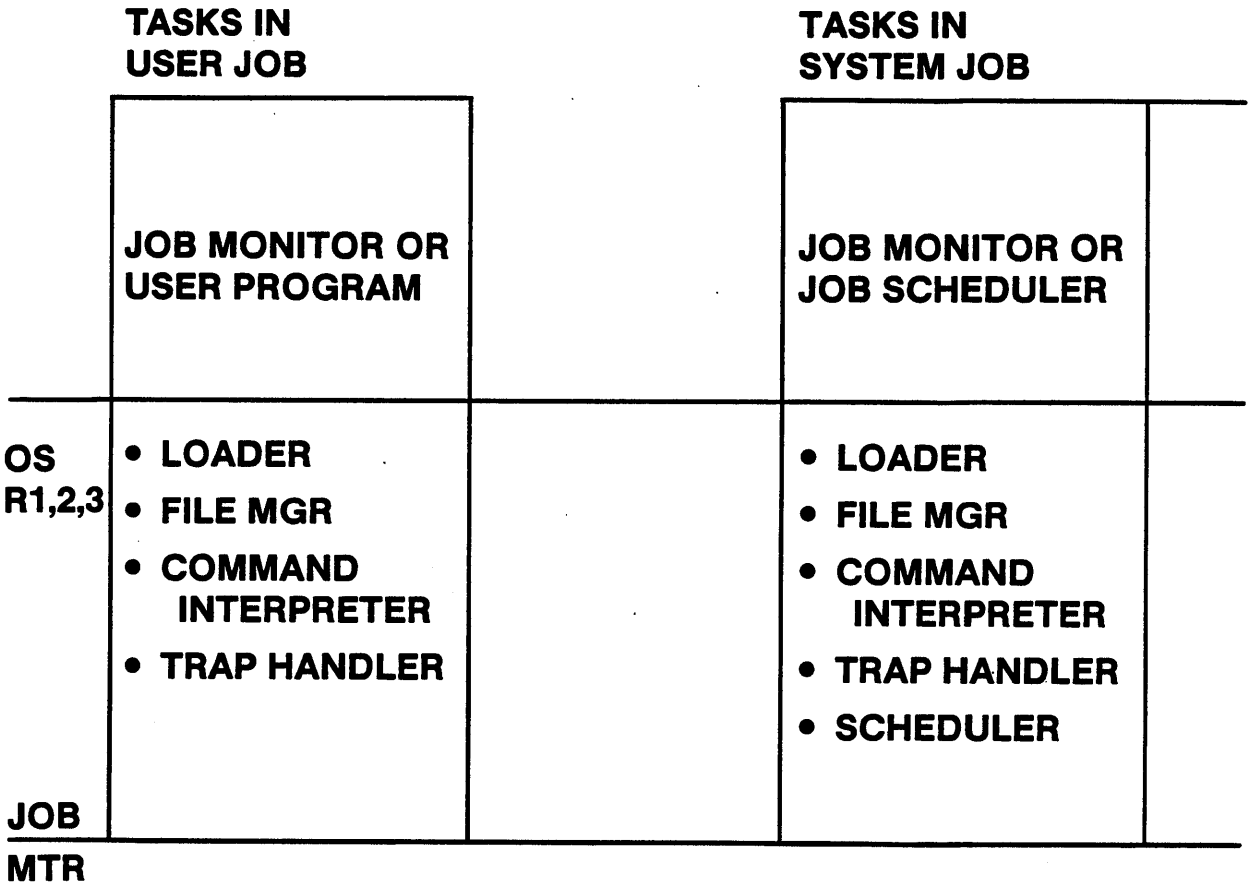
USER ADDRESS SPACE



PACKAGING


R11
"USER"
CODE AND DATA
R3
TASK CODE AND DATA
R2
JOB CODE AND DATA
R1
SYSTEM CODE AND DATA
JOB
MTR
MONITOR

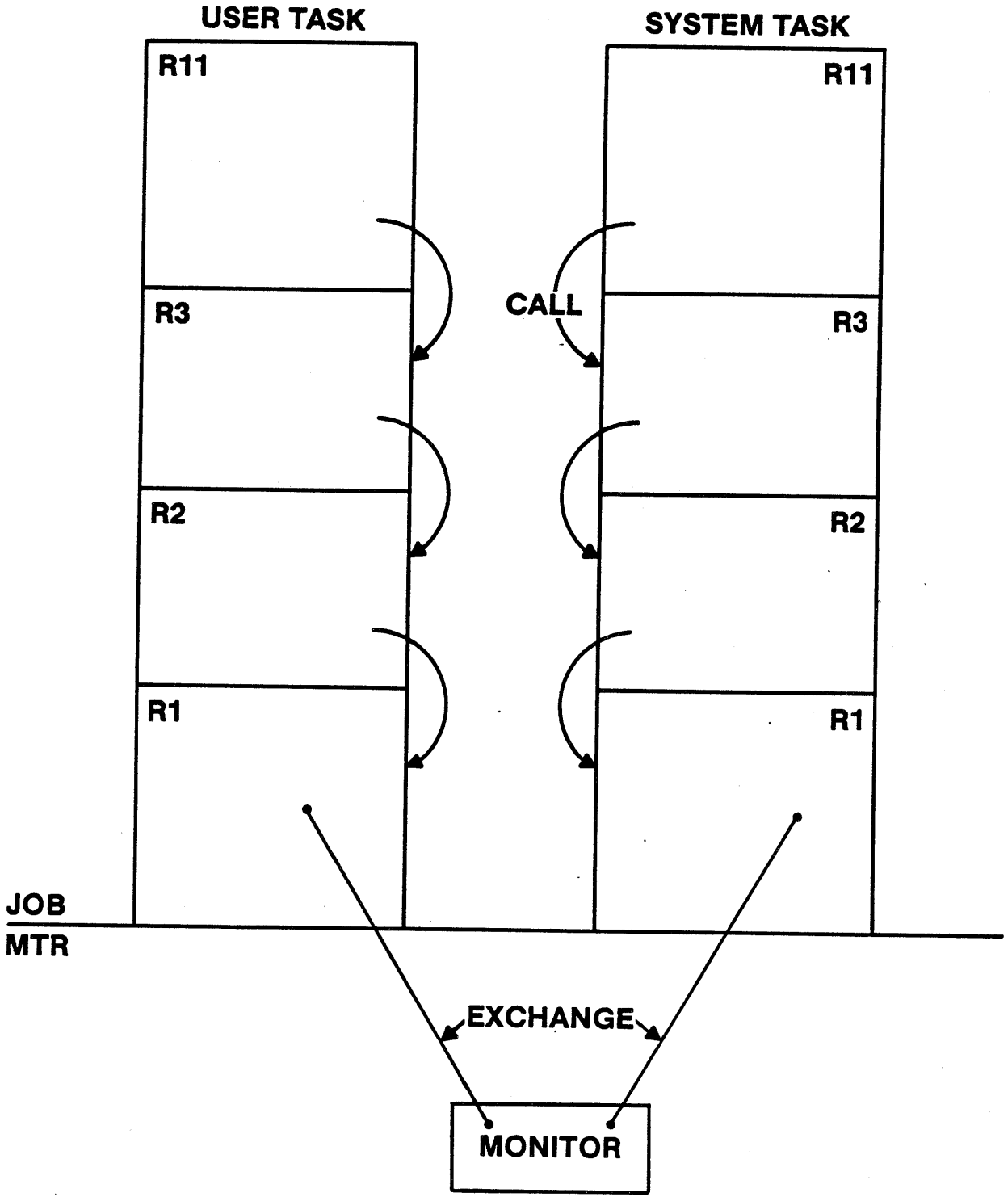
FUNCTIONS



MONITOR

- **Task Dispatcher**
- **Physical I/O** management of P.P. code
- **Page Manager**

COMMUNICATION (call/return)

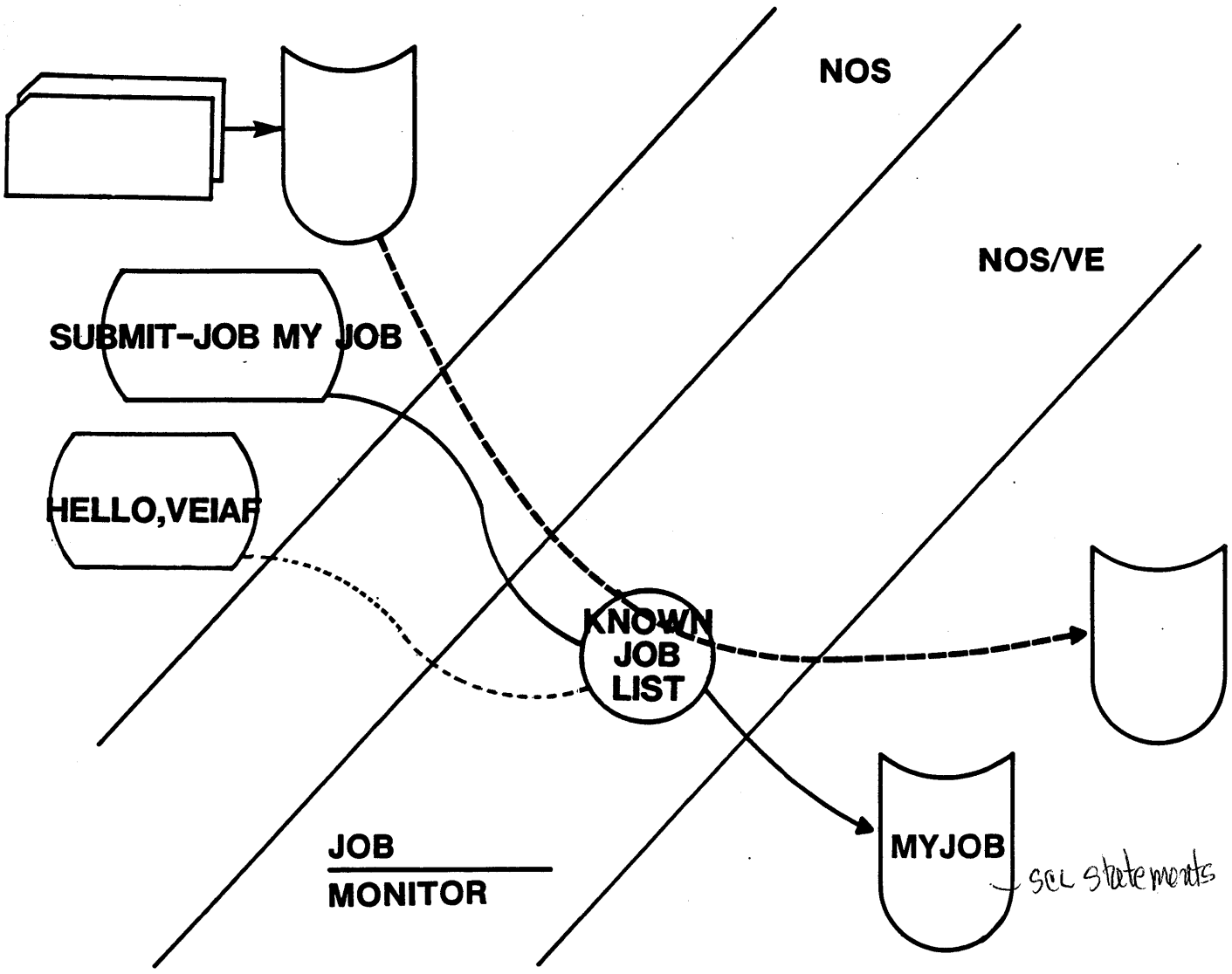


JOB FLOW

JOB FLOW

- Job Entry
- Job Initiation
- Command Processing
- Program Execution
- Job Termination

1
JOB ENTRY



- SUBMIT Command
- Interactive Login (via NOS)
- Remote Host Transfer (via NOS)

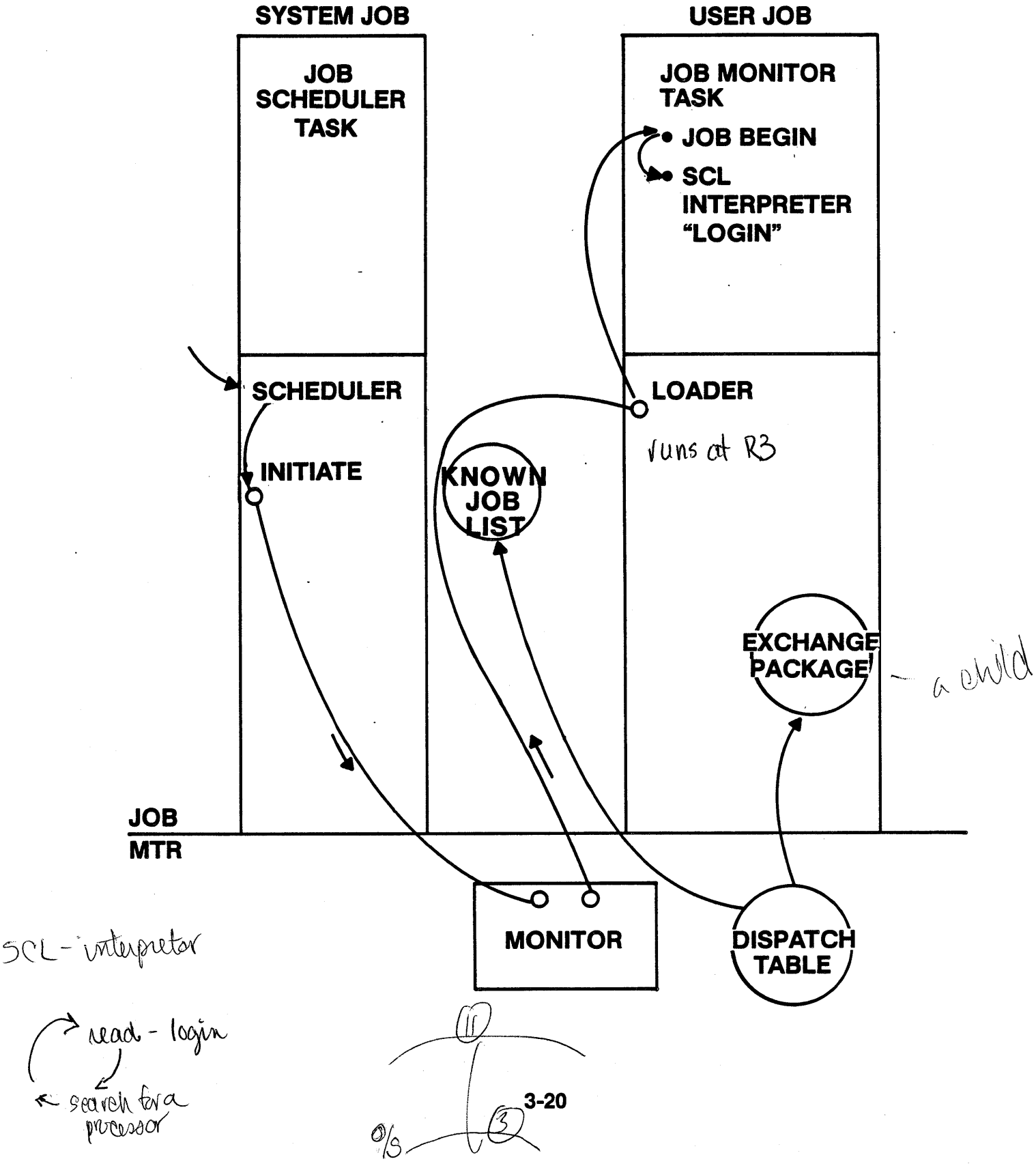
SUBMIT JOB

```
                .
                .
                .
/COLLECT_TEXT    MYJOB
CT/LOGIN   JHW12    JHWPW
CT/EXECUTE_TASK    MYBIN
CT/LOGOUT
CT/**
/SUBMIT_JOB    MY JOB    JOBNAME=JOB1
                .
                .
                .
```

JOB SCHEDULER

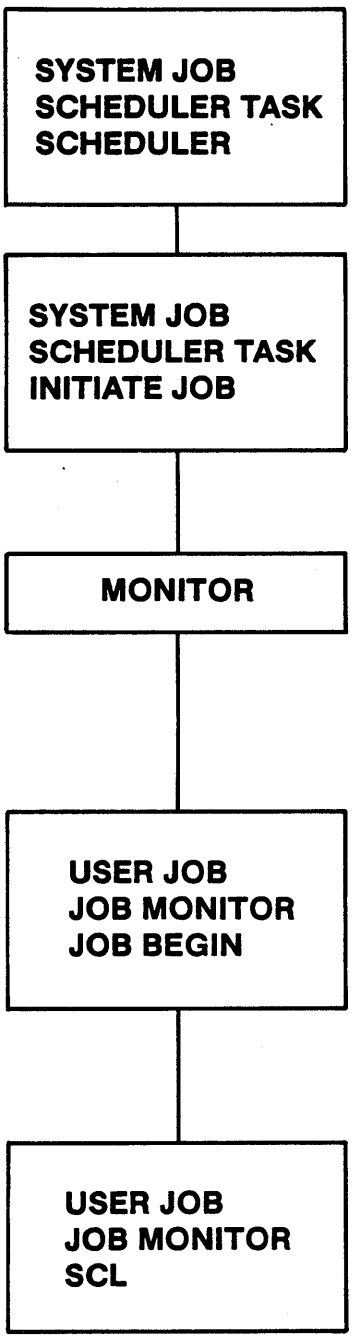
- Job scheduler executes as a system task
- Job scheduler determines:
  - — Order in which jobs in the input queue should be initiated
  - — When a job should be swapped into or out of memory
- Some examples of scheduling criteria are:
  - — Current priority within job class
  - — Job resource requirements
  - — Job class and status
  - — Current system resource availability
- Job scheduler monitors the available mix of queued, initiated, and swapped jobs and prioritizes them based on current system usage.

INITIATE JOB

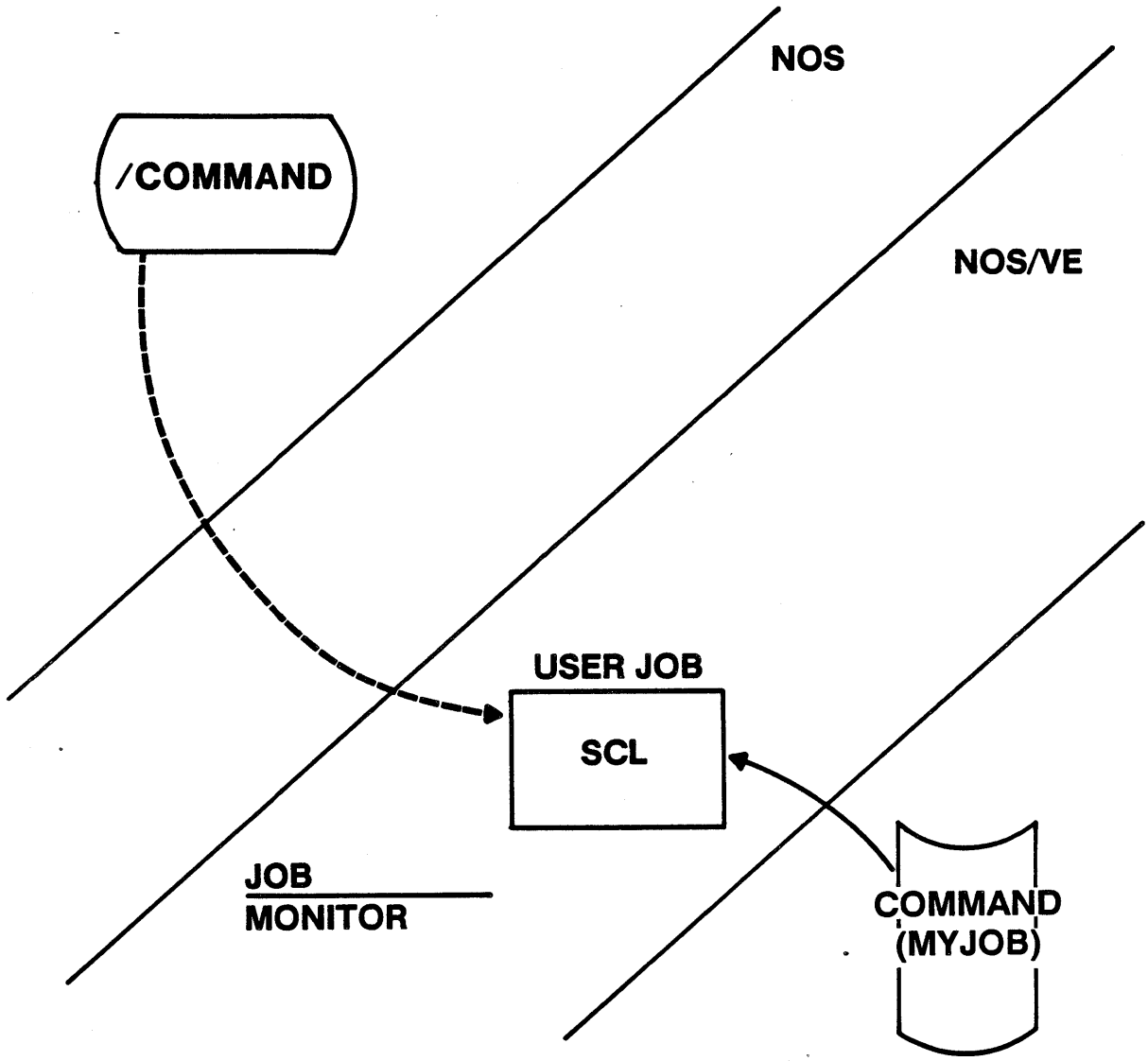


JOB INITIATION FLOW

- The scheduler selects the candidate from known job list
- Initiate job requests that monitor build the new job. This includes initializing some tables, including one that contains the exchange package.
- Monitor links the new task into the dispatch table. The task waits its turn. Eventually the dispatcher gives the new task its first time slice.
- The loader loads the first task which will be job monitor and starts it at the job begin entry point. Job begin initializes more tables and some files including the command file.
- The SCL Interpreter interprets the first command "LOGIN".



3
COMMAND PROCESSING

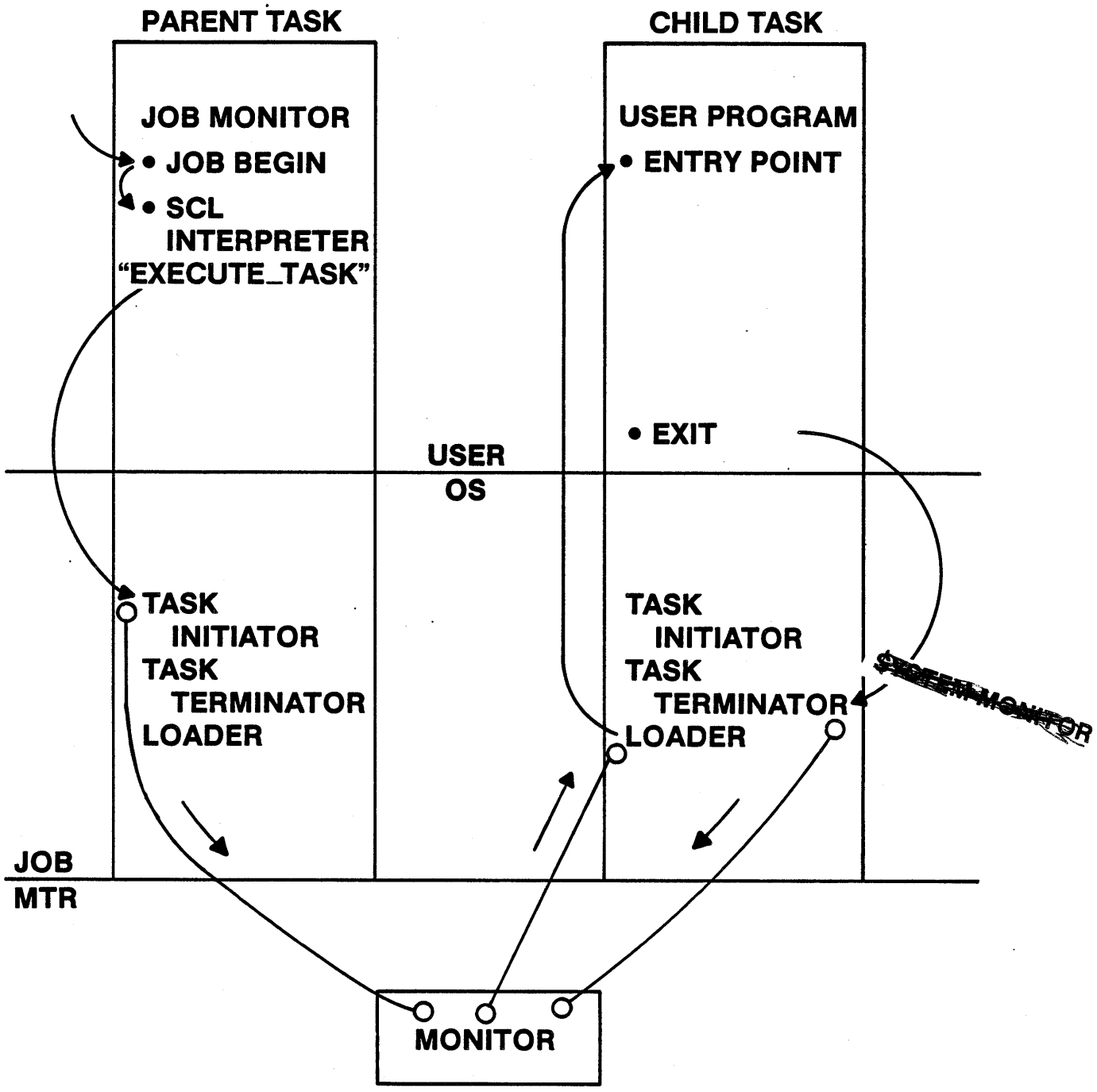


- **Command List** (local file catalog & ALLFILE catalog)
- **Name Call**
  - —**Program**
  - —**SCL Procedure**

SCL INTERPRETER

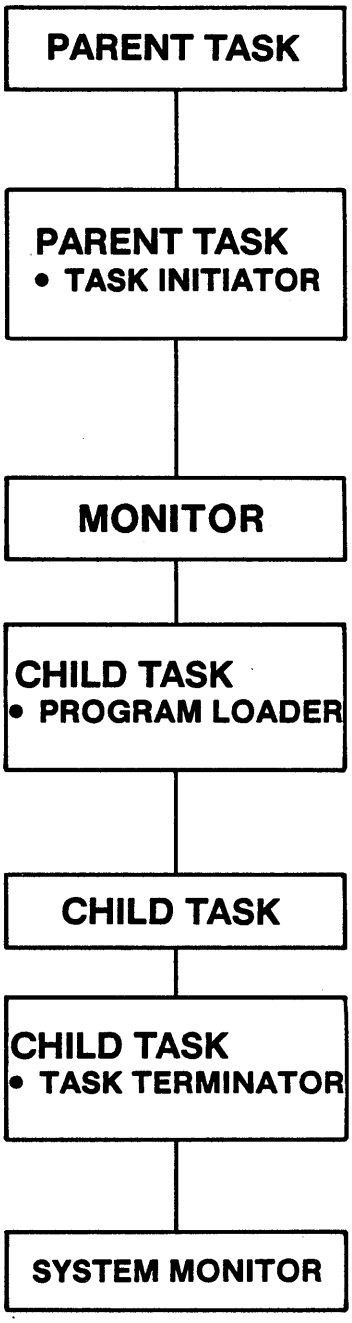
- SCL reads the command from the command file.
- SCL searches for the command.
  User Files
  System Files
- If the command is found , it may be a program or it may be an SCL procedure file.
  - — Commands on a procedure file will be executed.
    SCL interprets the first command.
  - — The program will be executed as a new task.
- In all cases, the SCL interpreter passes the command parameter list to a processor.

PROGRAM EXECUTION EXAMPLE

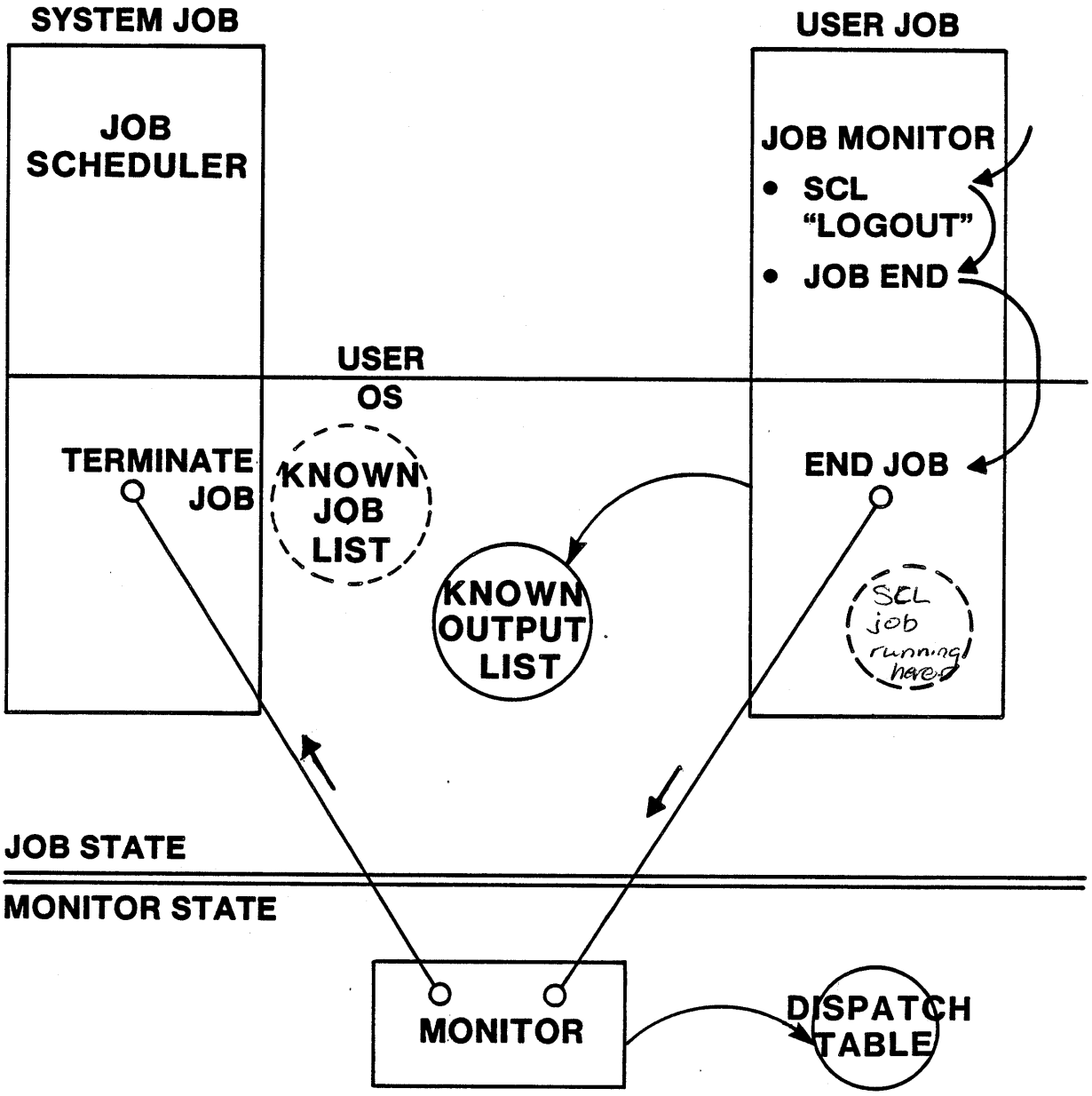


PROGRAM EXECUTION FLOW

- Program or command requests program execution
  Calls task initiator
- Builds tables for the new task
- Monitor links new task into CPU dispatch list.
  CPU is dispatched to new task
- Loader loads object modules
  Loader passes control to initial entry point
- The child task runs while the parent task waits
- Child task calls exit interface
- Cleans up task
  Exchanges to system monitor to request task termination
- Remove task entries from dispatch list
  Informs parent that child has terminated



4
TERMINATE JOB

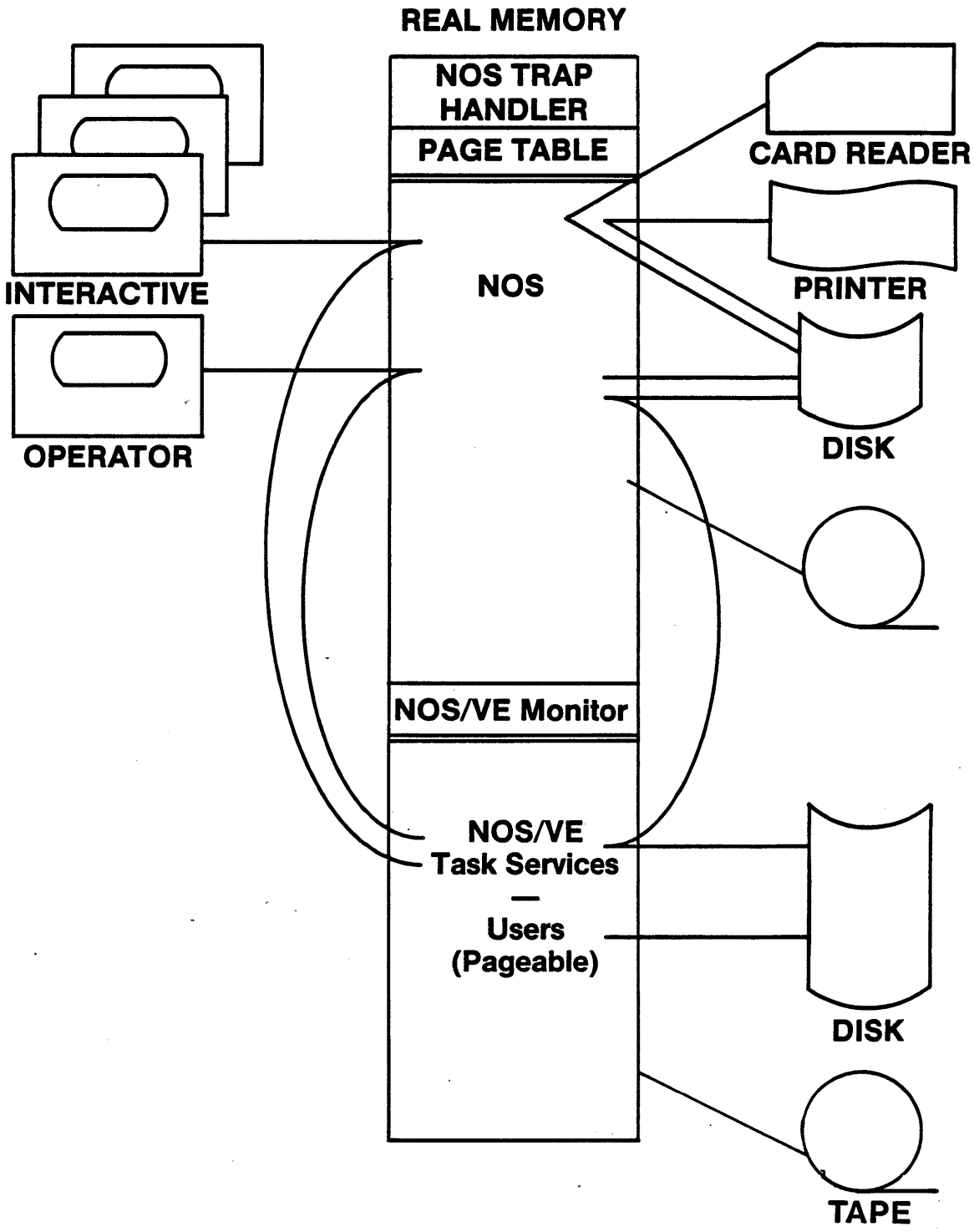


NOS/VE FEATURES

NOS/VE FEATURES

- Dual State
- Permanent Files
- Basic Access Methods
- Interactive Facility
- SCL
- Operator Facility
- Products

DUAL STATE MEMORY MAP



DUAL-STATE OVERVIEW

- User view is of two logically distinct machines in a multi-mainframe arrangement.
  - — Jobs and files belong to CYBER 170 or CYBER 180
  - — At deadstart time, memory, PPUs, channels and peripheral devices are partitioned between CYBER 170 and CYBER 180 logical machines
  - — Interface between CYBER 170 and CYBER 180 is through operating system services
- Access between CYBER 180 and CYBER 170 state is supported via central memory (memory link)
- Dual-state functions are not completely symmetric at release 1 but allow—
  - — Job submission to another logical machine
  - — The control and monitoring of a job executing in another logical machine
  - — File copying and routing between logical machines
  - — Inter-job message communication
- At release 1, some "front-end" functions are defined on CYBER 170 logical machine to serve—
  - — Input/output spooling
  - — Operator communication
  - — Network communication

NOS/VE PHASED RELEASES

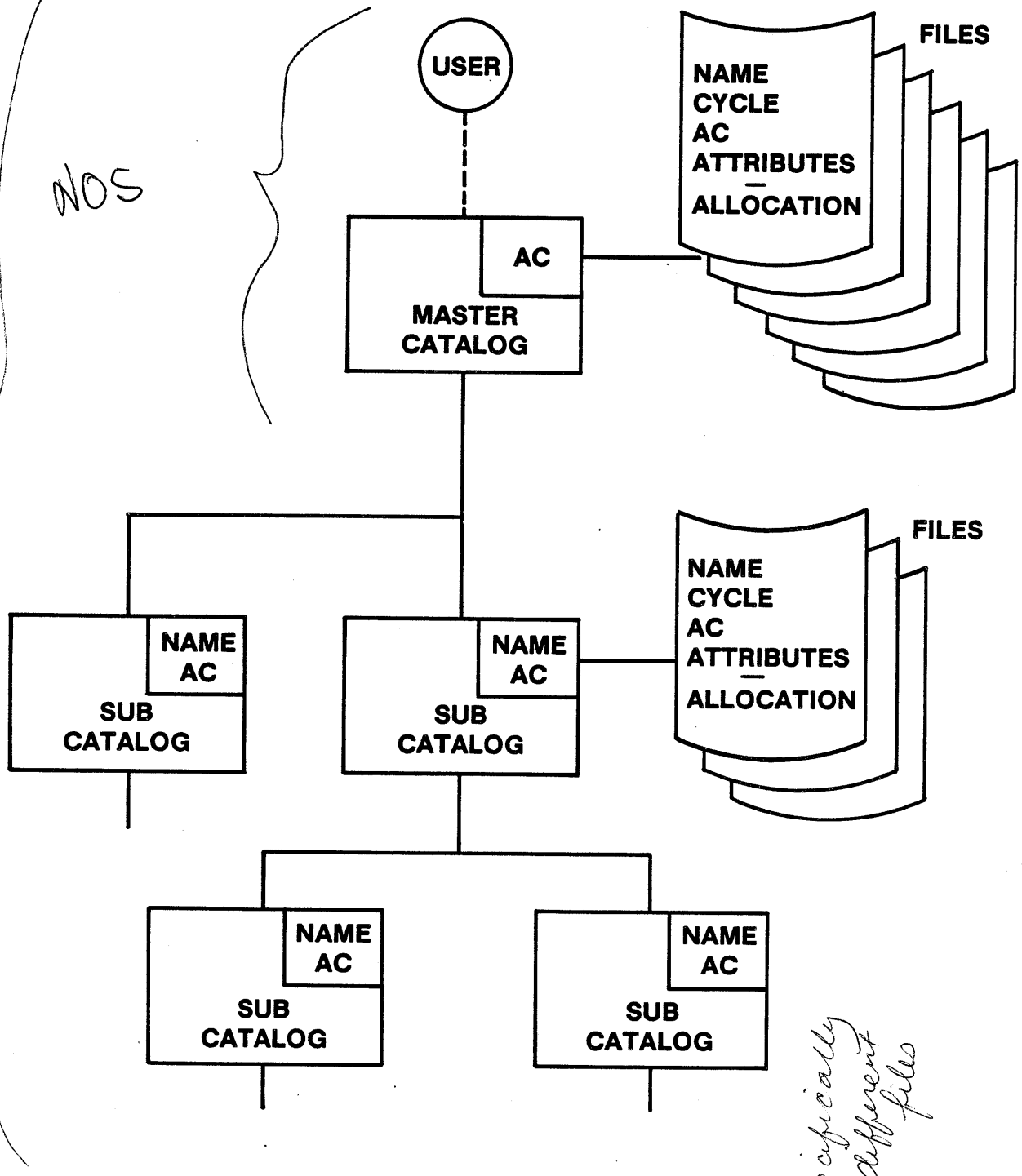
R1.0 - INTERNAL RELEASE
- BASIC OPERATING SYSTEM
- DUAL STATE W/NOS ONLY
- FORTRAN
- COBOL W/SINGLE KEY ISAM
- SORT/MERGE
- CONVERSION AIDS
- CYBIL

R1.1 - INITIAL EXTERNAL RELEASE (V1)
- OPERATING SYSTEM ENHANCEMENTS
  - JOB SCHEDULING
  - SWAPPING
- FULL SCREEN EDITOR
- COBOL W/ALTERNATE KEY ISAM
- LISP
- APL
- SUPPORT OF C170-815, 825, 835, 845, 855
  C180-810, 830, 835, 845, 855
- IM/VE
- UNIX/C
- DUAL STATE W/NOS/VE
- DUAL CPU SUPPORT (830, 855)

FUTURE RELEASES
- NAM/VE
- STAND ALONE NOS/VE

- SUPPORT OF C180-990
- BASIC
- PASCAL
- ADA - task orientated language

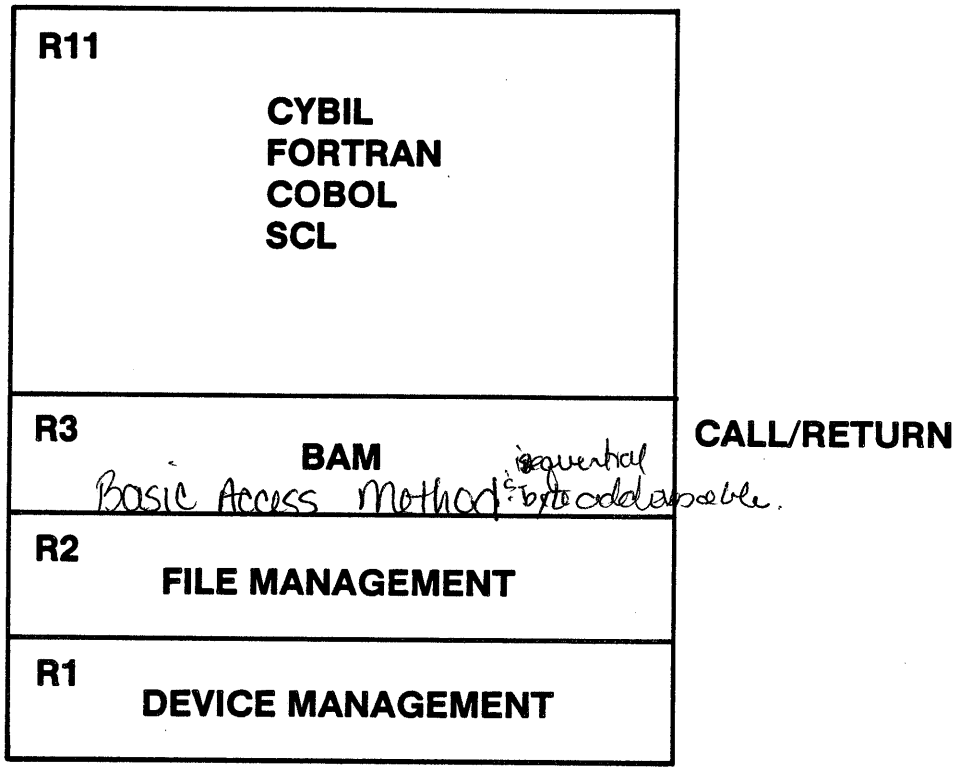
CATALOG/PERM FILE



PERMANENT FILE MANAGEMENT OVERVIEW

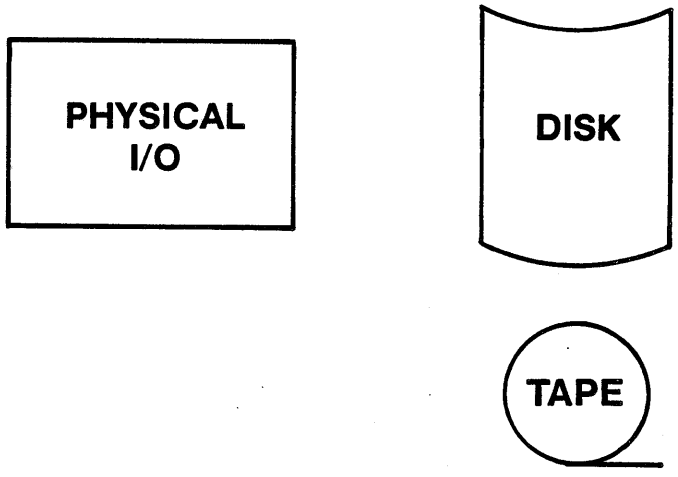
- Single permanent file mechanism which includes capabilities of NOS/170 direct and indirect files.
- Permanent file management functions are:
  - — Managing the registration of files in a catalog
  - — Establishing job access to a permanent file
  - — Access control mechanisms for limiting file access to permitted users according to authorized access modes
  - — Backup and recovery of permanent files
- Access Control (AC) to permanent files is a major element of NOS/VE security and protection.
  - — User/family/account/project qualification
  - — Access modes—read/write/append/execute/none
  - — Password
  - — Ring brackets
- A catalog is a system file containing information linking logical names of elements (e.g., permanent files, devices) with their description and access control lists. An entry in a catalog may be either a file name or the name of a subcatalog.
  - — Descriptor has identification and location of the element.
  - — Access control list has user names and their permitted access style.
  - — Access control list is maintained by the element owner.
- All permanent files are registered in a catalog.
- Each user is associated with a master catalog and may create subcatalogs.
- Each permanent file has a name up to 31 characters which is unique within the master catalog.
- Multiple versions of a file may be registered as cycles under a single name.
  - — Each cycle is a unique file.
  - — Access control list is common for all cycles.
- Users and their permanent files are logically grouped as families.
  - — Logical family name is used to route information within a complex of mainframes.

ACCESS METHOD ENVIRONMENT



JOB

MTR



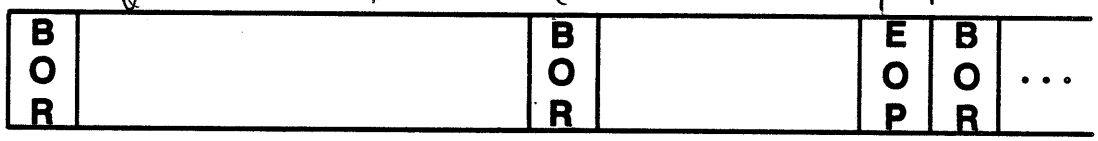
BASIC ACCESS METHOD

- Provides two access levels (record and segment)
- Provides security (all files have ring attributes)
- Provides full ANSI label processing*
- Provides error processing
- Supports five record types (W, S*, D*, F, U)
- Supports two block types (user/system specified)
- File attributes which describe the contents of the file are preserved with the file
- Provides device independence (MS, tape, terminal)
- Provides a single system default type for input job files, listings, etc. (V)
- Provides a basis for interchange of data with C170

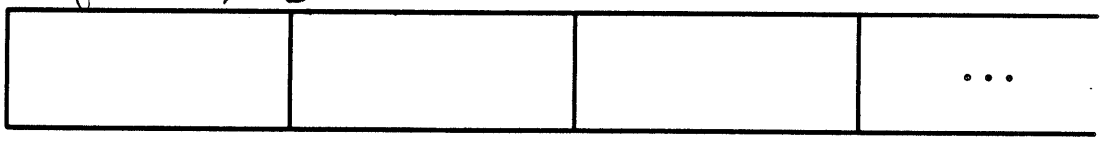
NOTE: *indicates features to be supported in future releases.

RECORD TYPES

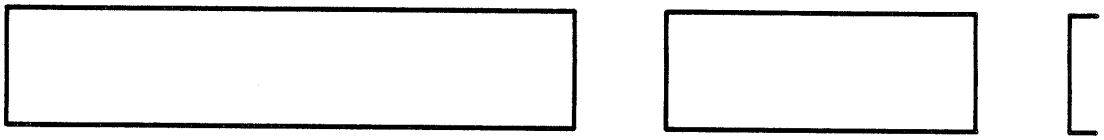
V – default record type (header info. for each record; end of partition)



F – fixed length (all records are same size)



U

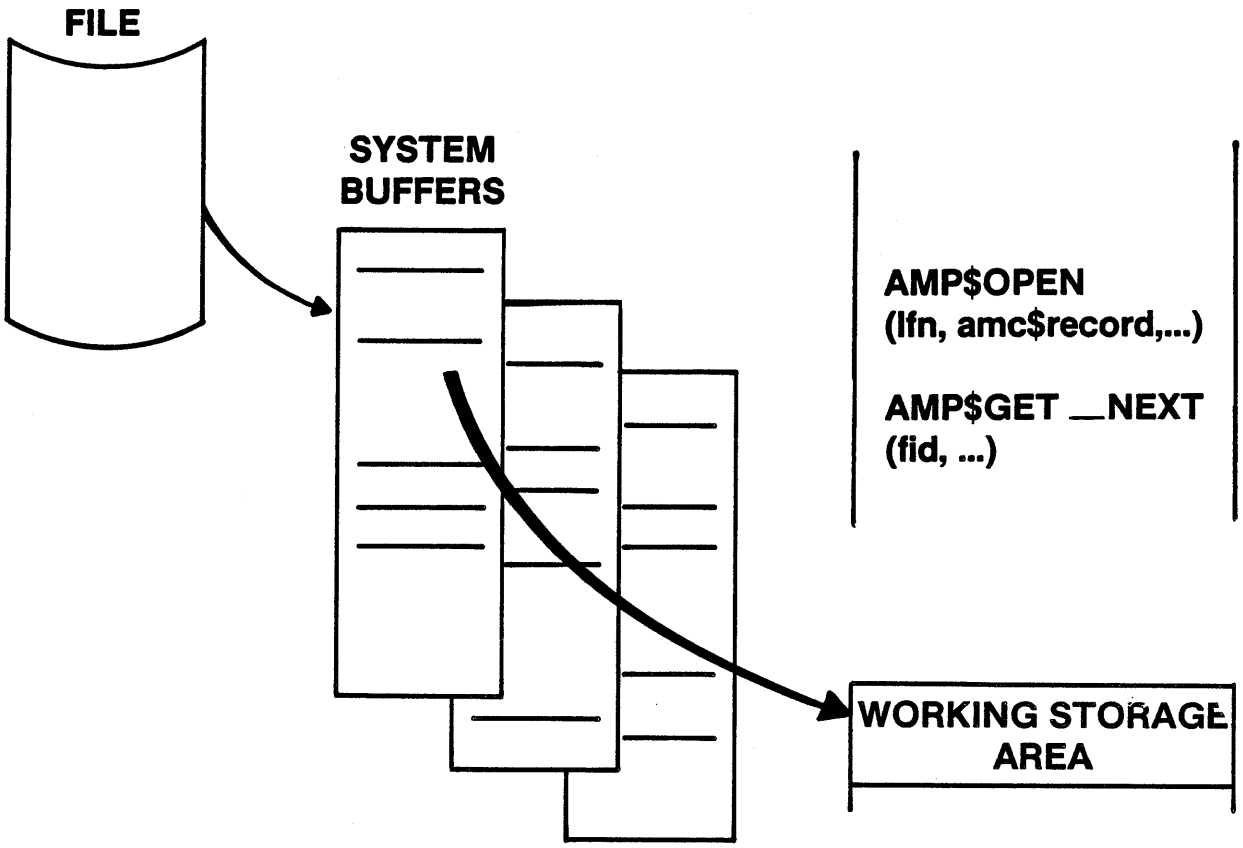


SUMMARY OF RECORD FEATURES

Feature	V	F	U
Self-defining Record Length	X		
Variable Length Record	X		X
Fixed Length Record	X	X	X
Partition Capability	X		
Deleted Record Capability	X		
Backspace Record Capability	X	X	X
Record Header	X		
Record Header Contains Reliability Information	X		
Support Foreign Data Structure			X
ANSI Standard Interchange Records Type		X	
Default NOS/VE Record	X		

X Indicates the feature is present

RECORD-LEVEL ACCESS



- SEQUENTIAL
- BYTE ADDRESSABLE

ACCESS LEVELS
RECORD- VS. SEGMENT-LEVEL ACCESS

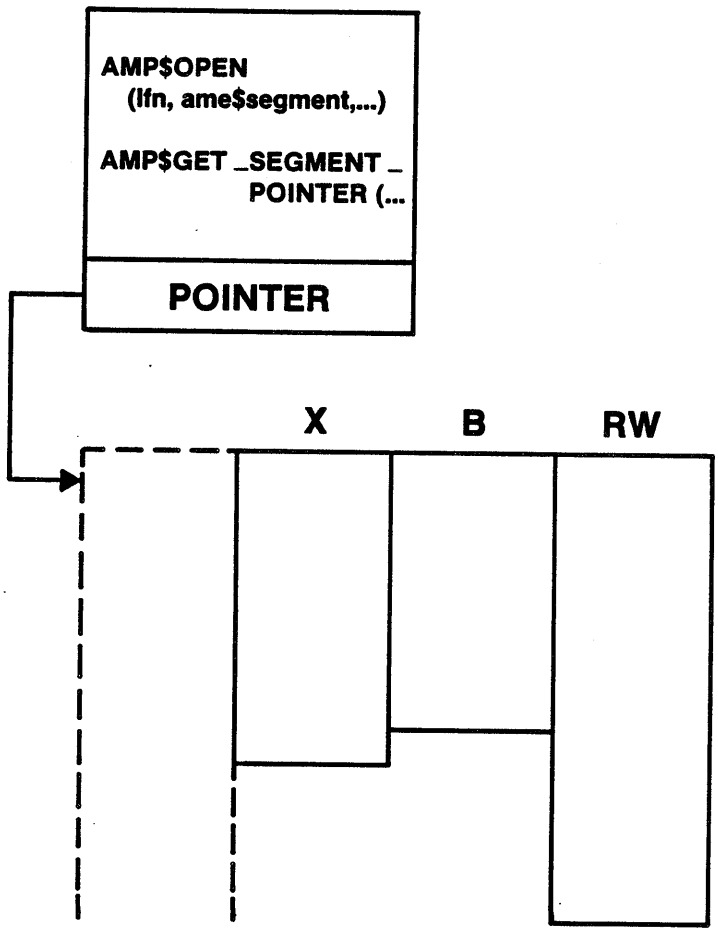
Record-level access

- The user must open the file for record-level access. The system provides buffers for the user.
- On a GET, the system will deliver one record as described in the file attributes into the user's buffer. Control information is stripped. On a PUT, the system will copy the record from the user's buffer. Control information is added.

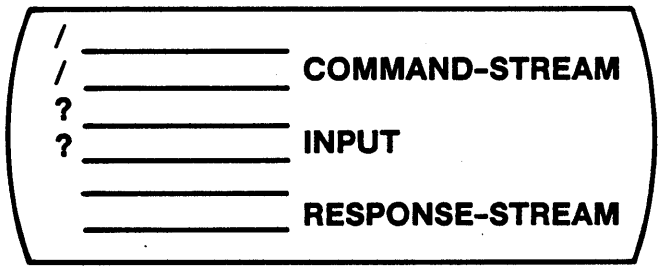
Segment-level access - faster

- The user must open the file for segment-level access. The system will add an entry to the SDT and return the PVA of the new segment to the requestor.
- The program accesses the new segment using CYBIL memory management instructions.

SEGMENT ACCESS



INTERACTIVE FACILITY



NOS/VE interactive terminal access is performed through the Network Access Method (NAM). Its interactive facilities are a superset of those of NOS/170. The terminal user may:

- Enter commands
- Enter data to programs
- Interrupt the execution of interactive jobs
- Receive command status messages
- Receive program output data
- Disconnect a terminal from a running interactive job, thus freeing the terminal for other work*
- Recover an interactive job that was disconnected from its terminal*
- Define terminal attributes

ATTRIBUTES
ACTION CHARACTERS
ABORT LINE
BACKSPACE
DELIMIT COLLECTOR
EOI
ECHO
OUTPUT DEVICE
PAGE CHARACTERISTICS
PROMPT
TERMINAL CLASS

NOTE: *indicates R2 feature

TERMINAL SESSION

```
                .
                .
                .
/HELLO, VEIAF
WELCOME TO NOS/VE...
PROCESSING SYSTEM PROLOG
PROCESSING USER PROLOG
/GETF FTPROG
/GETF FTDATA
/FTN I=FTPROG L=LIST B=LGO
/LGO
/COPF FTNRPT
REPORT...
                .
                .
                .
/SETWC $USER
/CRESL L=FTN_LIBRARY R=$LOCAL. FTN_LIBRARY
/SCU B=$LOCAL.FTN_LIBRARY R=FORTRAN_LIBRARY
SC/CRED D=FTPROG M=MODNAME S=$LOCAL.FTPROG
SC/EXPD D=FTPROG C=COMPILE   create compile file
SC/FTN I=COMPILE L=LIST B=$USER.LGO
SC/COPF $LOCAL.FTDATA FTDATA
SC/SETCL A=$USER      set command list
SC/LGO
SC/COPF FTNRPT
REPORT...
                .
                .
                .
SC/END WL=TRUE
/CREOL
COL/ADDM L=LGO    BIN
COL/GENL FORTRAN_LIBRARY
COL/QUIT
                  BIN
/EXET F=FORTRAN_LIBRARY
/COPF FTNRPT.2
REPORT...
                .
                .
                .
/LOGOUT
PROCESSING USER EPILOG
PROCESSING SYSTEM EPILOG
...APPLICATION: IAF
/BYE
                .
                .
                .
```

SYSTEM COMMAND LANGUAGE

- System Command Language (SCL) is the primary user interface to NOS/VE. Its features include:
  - — Single interface for both batch and interactive usage.
  - — English-like commands and responses.
  - — Consistent syntax and parameter naming conventions.
  - — Command lines of up to 256 characters (may be continued across several physical lines).
  - — Local and global variables.
  - — Standard built-in functions.
  - — Extensive conditional and unconditional branching and looping capabilities.
  - — Automatic condition processing.
  - — String and list processing facilities.
  - — Standard command parsing facilities.
  - — Multiple command lists may be stacked under user and system control (allows subcommand capability).

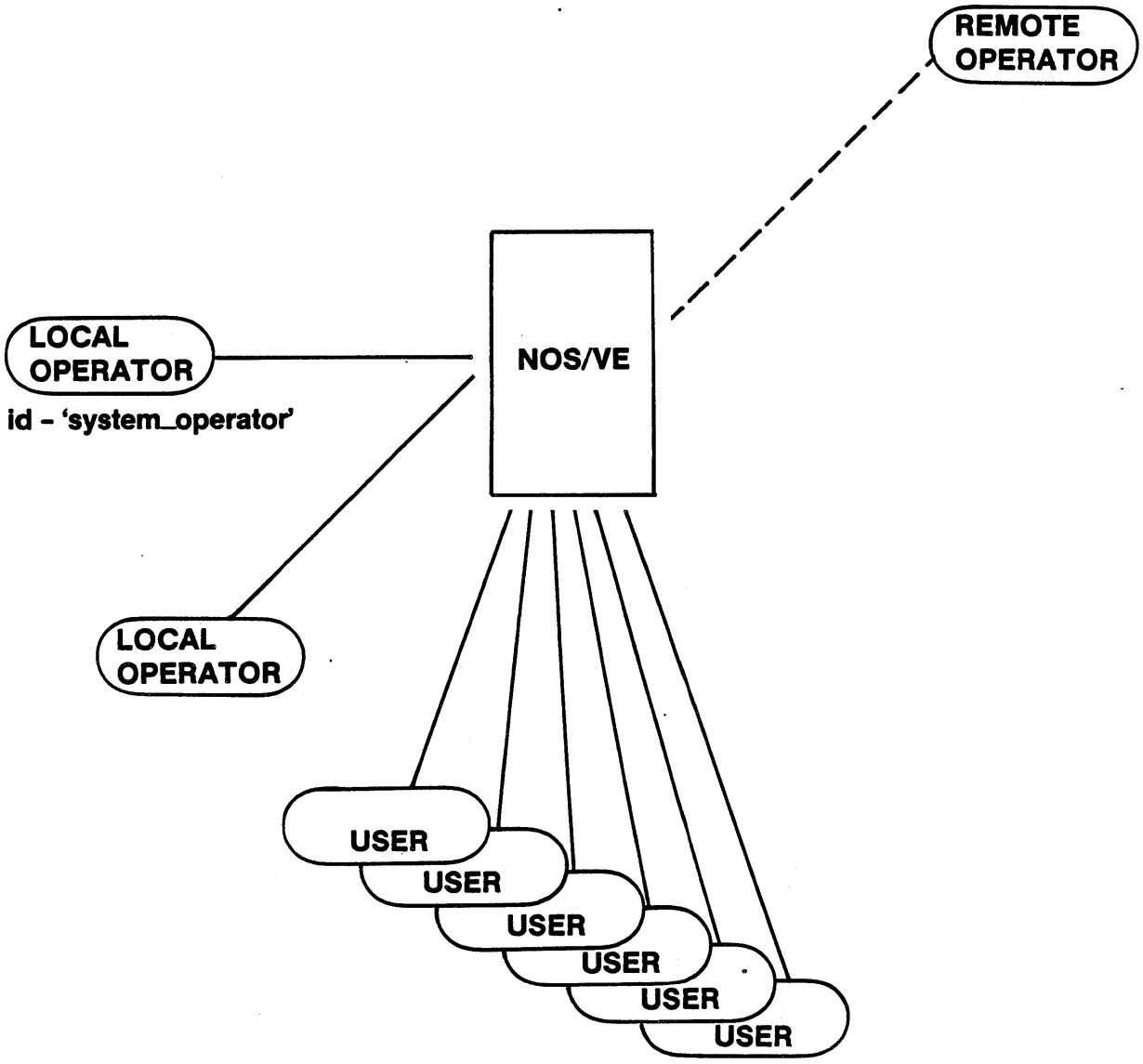
NOS/VE COMMANDS

- ACCESS AND CONTROL
  LOGIN
  SUBMIT_JOB
  TERMINATE_JOB
  HELP
  EXPLAIN
  DISPLAY_JOB_STATUS
  DISPLAY_LOG
  LOGOUT
- PERMANENT FILES
  CREATE_FILE
  DELETE_FILE
  CREATE_FILE_PERMIT
  CREATE_CATALOG
  CREATE_CATALOG_PERMIT
  SET_WORKING_CATALOG
- RESOURCE MANAGEMENT
  REQUEST_MAGNETIC_TAPE
  RESERVE_RESOURCE
  REQUEST_TERMINAL

NOS/VE COMMANDS
(Continued)

- FILE MANAGEMENT
  SET_FILE_ATTRIBUTES
  DETACH_FILE
  COPY_FILE
  COMPARE_FILE
  GET_FILE
  PRINT_FILE
- PROGRAM EXECUTION
  file name call
  EXECUTE_TASK
  DISPLAY_PROGRAM_ATTRIBUTES
- TOOLS
  CYBIL
  FORTRAN
  COBOL
  SOURCE_CODE_UTILITY
  CREATE_OBJECT_LIBRARY
- CONTROL
  FOR
  WHEN
  WHILE
  IF

OPERATOR FACILITY



OPERATOR COMMUNICATION OVERVIEW

- Supports communication between:
  - — NOS/VE and system operators
  - — User jobs and system operators
- NOS/VE operator consoles may be:
  - — The standard system console (75X terminal) connected through the two port MUX (supported at release 2)
  - — CC545 170 console may be used via dual-state at release 1
  - — Any interactive terminal (connected via CYBER 170 at release 1)
- An operator console is a terminal "logged-in" with system operator privileges.
  - — Operator commands and displays are processed by NOS/VE interactive jobs having system operator privileges granted by the NOS/VE user validation.
  - — The installation may distribute access privileges between users.
- On-line equipment configuration commands
- Tape drive assignment commands
- Job queue access and control
- Status and control of system jobs

PRODUCTS

- LANGUAGES
- CONVERSION AIDS
- SYSTEM TOOLS
- APPLICATIONS

LANGUAGES

AVAILABLE WITH INITIAL RELEASE - R1.1

- FORTRAN (ANSI 77)*
- COBOL (ANSI 74)*
- CYBIL*
- APL*
- SORT/MERGE

AVAILABLE WITH FUTURE RELEASES

- PASCAL
- LISP*
- BASIC (COMPILER)
- ADA
- C (UNDER THE VX/VE (UNIX) SUBSYSTEM OR NATIVE NOS/VE)

*COMMON SPECIFICATIONS USED FOR THESE COMPILERS FOR NOS/VE, NOS, AND NOS/BE

CONVERSION AIDS

- Command Sequences
  CCL ⟶ SCL*
  Tutor*
- Programs
  Common Specifications
  Dependency Flags
- Homogeneous Files
  Character Data
  Numeric
- Nonhomogeneous Files
  CRM Files
  FMU (File Maintenance Utility)
  File Access Procedures
  Language Oriented Converters*

*R2 Feature

SYSTEM TOOLS

- ON-LINE INTERACTIVE DEBUG
- ON-LINE MANUAL ACCESS AND GENERATION
- FULL SCREEN EDITOR
- SOURCE CODE UTILITY (SCU)
- OBJECT CODE UTILITY (OCM)
- ANALYZE PROGRAM DYNAMICS UTILITY

FMA: file management aids
(helps in NOS ⇄ NOS/VE conversion)

APPLICATIONS THAT WILL BE AVAILABLE FOR NOS/VE

- VX/VE (UNIX* SYSTEM) INCLUDING THE C COMPILER AVAILABLE AS A NOS/VE SUBSYSTEM
- IM-VE (INFORMATION MANAGEMENT)

  RELATIONAL DATA BASE MGMT SYSTEM FROM BATTELLE MEMORIAL INSTITUTE
- DI-3000 (GRAPHICS)

  USES THE THREE DIMENSIONAL STANDARD CORE FROM PRECISION VISUALS

  CATALOGS LISTING C180 APPLICATIONS WILL BE ISSUED PERIODICALLY

*TRADEMARK OF BELL LABORATORIES

APPENDIX

REGISTER DEFINITIONS AND ACCESSES

REGISTER GROUP	REGISTER NUMBER (S)	REGISTER NAME	COPY ACCESS PRIVILEGES		MCU ACCESS
			COPY FROM STATE REGISTER	COPY TO STATE REGISTER	
			READ	WRITE	
00-0F	00	STATUS SUMMARY	NO ACCESS		
	10	ELEMENT ID			
10-1F	11	PROCESSOR ID	UNPRIV.	NO ACCESS	R
	12	OPTIONS INSTALLED			
	13	VIRTUAL MACHINE CAPABILITY LIST			
	20	ENVIRONMENT CONTROL			
20-2F	31	CONTROL STORE ADDRESS	NO ACCESS	NO ACCESS	R/W
30-3F	32	CONTROL STORE BREAKPOINT			
	40	P REGISTER			
	41	MONITOR PROCESS STATE POINTER			
	42	MONITOR CONDITION REGISTER			
	43	USER CONDITION REGISTER			
	44	UNTRANSLATABLE POINTER			
40-4F	45	SEGMENT TABLE LENGTH			
	46	SEGMENT TABLE ADDRESS	UNPRIV.	NO ACCESS	R/W
50-5F	47	BASE CONSTANT			
	48	PAGE TABLE ADDRESS			
	49	PAGE TABLE LENGTH			
	4A	PAGE SIZE MASK			
	50	MODEL DEPENDENT FLAGS			
	51	MODEL DEPENDENT WORD			
	60	MONITOR MASK REGISTER			
60-6F	61	JOB PROCESS STATE POINTER	UNPRIV.	MONITOR	R/W
70-7F	62	SYSTEM INTERVAL TIMER			
	80-8F	PROCESSOR FAULT STATUS			
80-8F	90	RETRY CORRECTED ERROR LOG			
90-9F	91	CONTROL STORE CORRECTED ERROR LOG	UNPRIV.	GLOBAL	R/W
A0-AF	92	CACHE CORRECTED ERROR LOG			
B0-BF	93	MAP CORRECTED ERROR LOG			
	A0	PROCESSOR TEST MODE			
	C0-C3	TRAP ENABLES			
	C4	TRAP POINTER			
C0-CF	C5	DEBUG POINTER			
	C6	KEYPOINT MASK	UNPRIV.	LOCAL	R/W
D0-DF	C7	KEYPOINT CODE			
	C8	KEYPOINT CLASS NUMBER			
	C9	PROCESS INTERVAL TIMER			
	E0-E1	CRITICAL FRAME FLAG			
E0-EF	E2-E3	ON CONDITION FLAG			
	E4	DEBUG INDEX	UNPRIV.	UNPRIV.	R/W
F0-FF	E5	DEBUG MASK REGISTER			
	E6	USER MASK REGISTER			

CDC CYBER 180 ABBREVIATIONS

A REGISTER	ADDRESS REGISTER
ASID	ACTIVE SEGMENT IDENTIFIER
BC	BASE CONSTANT
BDP	BUSINESS DATA PROCESSING
BN	BYTE NUMBER
BS	BINDING SECTION
CBP	CODE BASE POINTER
CEL	CORRECTED ERROR LOG
CF	CRITICAL FRAME
CFF	CRITICAL FRAME FLAG
CM	CENTRAL MEMORY
CMU	COMPARE-MOVE UNIT
CPU	CENTRAL PROCESSING UNIT
CSF	CURRENT STACK FRAME POINTER
DC	DEBUG CODE
DEC	MODEL-DEPENDENT ENVIRONMENT CONTROL
DI	DEBUG INDEX
DLP	DEBUG LIST POINTER
DM	DEBUG MASK
DMR	DEBUG MASK REGISTER
DSP	DYNAMIC SPACE POINTER
EID	ELEMENT IDENTIFIER
FL	FIELD LENGTH
FLC	CENTRAL MEMORY FIELD LENGTH REGISTER
FLE	EXTENDED CORE STORAGE FIELD LENGTH REGISTER
GK	GLOBAL KEY
GL	GLOBAL LOCK
I/O	INPUT/OUTPUT
IOU	INPUT/OUTPUT UNIT
JEP	JOB MODE EXCHANGE PACKAGE
PTA	PAGE TABLE ADDRESS
PTE	PAGE TABLE ENTRY
PTL	PAGE TABLE LENGTH
PTM	PROCESSOR TEST MODE
PVA	PROCESS VIRTUAL ADDRESS
P1...Pn	CENTRAL PROCESSOR 1...CENTRAL PROCESSOR n
RA	REFERENCE ADDRESS
RA/FL	REFERENCE ADDRESS/FIELD LENGTH
RAC	CENTRAL MEMORY REFERENCE ADDRESS REGISTER
RAE	EXTENDED CORE STORAGE REFERENCE ADDRESS REGISTER

CDC CYBER 180 ABBREVIATIONS

(Continued)

RAM	RANDOM ACCESS MEMORY
RAM	RELIABILITY, AVAILABILITY, MAINTAINABILITY
RMA	REAL MEMORY ADDRESS
RN	RING NUMBER
ROM	READ-ONLY MEMORY
RP	READ ACCESS CONTROL (SEGMENT DESCRIPTOR FIELD)
SCL	SYSTEM COMMAND LANGUAGE
SDE	SEGMENT DESCRIPTOR TABLE ENTRIES
SDT	SEGMENT DESCRIPTOR TABLE
SECDED	SINGLE ERROR CORRECTION/DOUBLE ERROR DETECTION
SEG	PROCESS SEGMENT NUMBER
SFSA	STACK FRAME SAVE AREA
SIT	SYSTEM INTERVAL TIMER
SPT	SYSTEM PAGE TABLE
SS	STATUS SUMMARY
STA	SEGMENT TABLE ADDRESS
STL	SEGMENT TABLE LENGTH
SVA	SYSTEM VIRTUAL ADDRESS
S1...Sn	SYSTEM 1...SYSTEM n
TE	TRAP ENABLE
TED	TRAP ENABLE DELAY
TEF	TRAP ENABLE FLIP-FLOP
TOS	TOP OF STACK
TP	TRAP POINTER
UCR	USER CONDITION REGISTER
UEM	USER EXTENDED MEMORY
UM	USER MASK
UTP	UNTRANSLATABLE POINTER
UVMID	UNTRANSLATABLE VIRTUAL MACHINE IDENTIFIER
VC	SEARCH CONTROL CODE (PAGE DESCRIPTOR FIELD)
VL	SEGMENT VALIDATION (SEGMENT DESCRIPTOR FIELD)
VMCL	VIRTUAL MACHINE CAPABILITY LIST
VMID	VIRTUAL MACHINE IDENTIFIER
WP	WRITE ACCESS CONTROL (SEGMENT DESCRIPTOR FIELD)
XP	EXECUTE ACCESS CONTROL (SEGMENT DESCRIPTOR FIELD)

CYBER 170/180 SIMILARITIES/DIFFERENCES SUMMARY

CYBER 170	CYBER 180
CPU	
• 60-bit word	• 64-bit word
• word addressing	• byte/word addressing (8 bytes/word, byte = 8 bits)
• 8 X registers (60 bits) 8 B registers (18 bits) 8 A registers	• 16 X registers (64 bits) no B registers 16 A registers (48 bits) (store/load instructions)
• 1's complement arithmetic	• 2's complement arithmetic in CYBER 180 State, 1's complement arithmetic in CYBER 170 State
• CYBER 170 instruction set	• CYBER 180 + CYBER 170 instruction set mode bits (VMID for Virtual Machine Identifier) in CYBER 180 exchange package enables CYBER 180 or CYBER 170 State Instructions
• register-to-register operations • some character-handling instructions	• register-to-register operations • full set of character-handling instructions • vector instructions (memory-to-memory) on high-performance models
MEMORY	
• max 131K word user address space	• 4096 times 2**31 byte user virtual address space
• RA/FL relocation	• hardware segmented memory (max 4096 segments per user address space)

• 17 bit-word address within RA/FL defined address space	• two-part virtual address — segment number (12 bits) — signed byte offset into segment (32 bits) space
• 262K max system executable memory	• 64 Mbyte (potentially 2**31 byte) executable real memory
• memory moves + swapping to manage memory	• hardware paged + swapping to manage memory
PPs	
• up to 2×10 12-bit PPU's	• up to 4×5 16-bit PPs
• up to 2×12 12-bit channels	• up to 6×4 12/16-bit channels
• executes 12-bit PPU code	• executes 12-bit, 16-bit, or mixture, PP code (upward compatible with CYBER 170)
• memory size is 4Kx12 bits	• memory size is 4Kx16 bits
• 12-bit wide data channels	• 12-and/or 16-bit wide data channels
• 60-bit access to central memory	• 64 (4×16 bits)- and 60 (5×12 bits)- bit access to central memory
• 18-bit central memory address for PPU read/write	• 28-bit central memory addressed for PP read/write
• real central memory addressing	• real central memory addressing
• 500 ns major cycle time	• 250 ns major cycle time
• 16 words long Deadstart Panel	• 16 words long Deadstart Panel + 512-word read-only memory usable at deadstart
SYSTEM	
• no shared memory among user address spaces (defined by its RA/FL)	• segments sharable among user address spaces (code and data sharing possible)

• code/data mixed within user's address space	• segments can be read/write/execute, or combination—globally sharable code
• exchange operation to go CPU Monitor	• exchange operation to go to CPU Monitor
• CPU supports CYBER 170 instruction set	• CPU supports coexisting CYBER 180 and CYBER 170 instruction sets. VMID (Virtual Machine Indentifier) field, within CYBER 180 exchange package, is used to switch between CYBER 170/180 State instruction sets. The CYBER 170 environment for CYBER 170 NOS or CYBER 170 NOS/BE is established within the CYBER 180 job space and then state switching may be accomplished by an exchange or trap operation and Call or Return instructions. CYBER 170 external interrupts are supported and handled within the CYBER 170 environment.
• System runs at System Control Points or in PPU's CPU Monitor routes RA+1 requests	• Most CYBER 180 system code runs within user address space and obeys the same calling, loading/linking conventions. — levels of system code are protected by a hardware supported hierarchical 'ring' mechanism from less capable code modules. (Fifteen ring levels are provided.) — System code can be directly "called" by "RETURN Jump" similar to CALL instruction without software assist.
• Subsystems (i.e., Telex, Magnet, Data Manager, etc.) are protected by RA/FL mechanisms from each other or user, can be called via CPU Monitor	• Subsystems are hardware-supported (key/lock) mechanisms from each other, directly callable by user code without software assist.

REVIEW QUESTIONS
HARDWARE

1) What are the migration implications of dual state?

2) Why does the hardware have buffer memories?

3) What is the relationship between virtual memory and paged real memory?

4) What are the three kinds of virtual memory protection? What is protected in each case?

5) Code segments have three ring numbers. Data segments have two ring numbers. What do the ring numbers mean?

6) What is a PVA? An SDT?

7) In the phrase, "address translation," what gets translated to what?

8) Construct a call stack for the following sequence. Assume all calls are legal.
   A calls B
   B calls B
   B calls C
   C returns
   B calls B

9) How is an exchange interrupt caused? What happens when it occurs? How is a trap interrupt caused? What happens when it occurs?

10) What are the capabilities of the MCU?

REVIEW QUESTIONS
SOFTWARE

1) Who uses the command interface? The program interface?

2) List the reasons for a binding segment.

3) List the five facilities that use the memory link on NOS/VE release.

4) What is CYBIL?

5) Distinguish between job and task.

6) Distinguish between sections and segments.

7) List three file attributes.

8) Distinguish between record and segment level access.

9) Why are terminal attributes stored by NOS/VE?

10) What course in the sequence should you take next?

COMMENT SHEET

MANUAL TITLE: CYBER 180 TECHNICAL INTRODUCTION

PUBLICATION NO.: RW3020-1 **REVISION:** B

NAME: _____

COMPANY: _____

STREET ADDRESS: _____

CITY: _____ **STATE:** _____ **ZIP CODE:** _____

This form is not intended to be used as an order blank. Control Data Corporation welcomes your evaluation of this manual. Please indicate any errors, suggested additions or deletions, or general comments below (please include page number references).

CUT ALONG LINE

AA3419 REV. 4/79 PRINTED IN U.S.A.

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

FOLD ON DOTTED LINES AND STAPLE

STAPLE

STAPLE

FOLD

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 8241 MINNEAPOLIS, MINN.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION
National Coordinator
Bloomington Facility (MNA02B)
5001 West 80th Street
Bloomington, Minnesota 55437
Attn: Curtis Vicha

CUT ALO NE

FOLD

FOLD