TECHNOLOGY EDUCATION PROGRAM

CYBER 180 OVERVIEW

1037

PRESENTED BY: JEOFF S. BARRETT, CONTROL DATA CORPORATION
J. A. "TONY" WILSON, CONTROL DATA CORPORATION

HUMAN RESOURCE DEVELOPMENT
P. O. BOX O
MINNEAPOLIS, MINNESOTA 55440

"AN INTRODUCTORY COURSE TO CYBER 180"

9/78

HISTORY

CYBER 180 HISTORY

APRIL 1973 — NCR/CDC TASK FORCES

JUNE 1973 — ASL FORMED TO DEFINE A NEW MACHINE LINE WITH:

- VIRTUAL MEMORY
- LARGE real MEMORIES
- BDP
- SINGLE SET OF SOFTWARE

1974 — PROCESSORS, MEMORIES AND I/O SUB-SYSTEMS DEFINED - IPL

— NOMENCLATURE ESTABLISHED:

- SYSTEMS: S1-S4
- PROCESSORS: P1-P4
- MEMORIES: M1-M4

— WORK STARTED ON P2 AND S1

1975 — WORK STARTED ON M2 AND P3

— WORK CONTINUED ON P2 AND S1

SEPT 1976 — POWER ON P2

— IPL NAMED CYBER 180 (= IPL Px + IPL M x + IOU)

— AD&C FORMED

— IOU DEFINED after NCR's 6150

FEB 1977 — POWER ON M2

FEB 1978 — POWER ON IOU

CYBER 180 MODELS

MODEL	ADV. 170 STATE		CYBER 180 STATE	
	PERF.	AVAIL.	PERF.	AVAIL.
S1	1.0	4Q80	1.2	1982
S2	2.5	1Q80	3.0	1982
S3	7.5	1Q81	9.0	1982
THETA	30.0	4Q83	34.0	1984

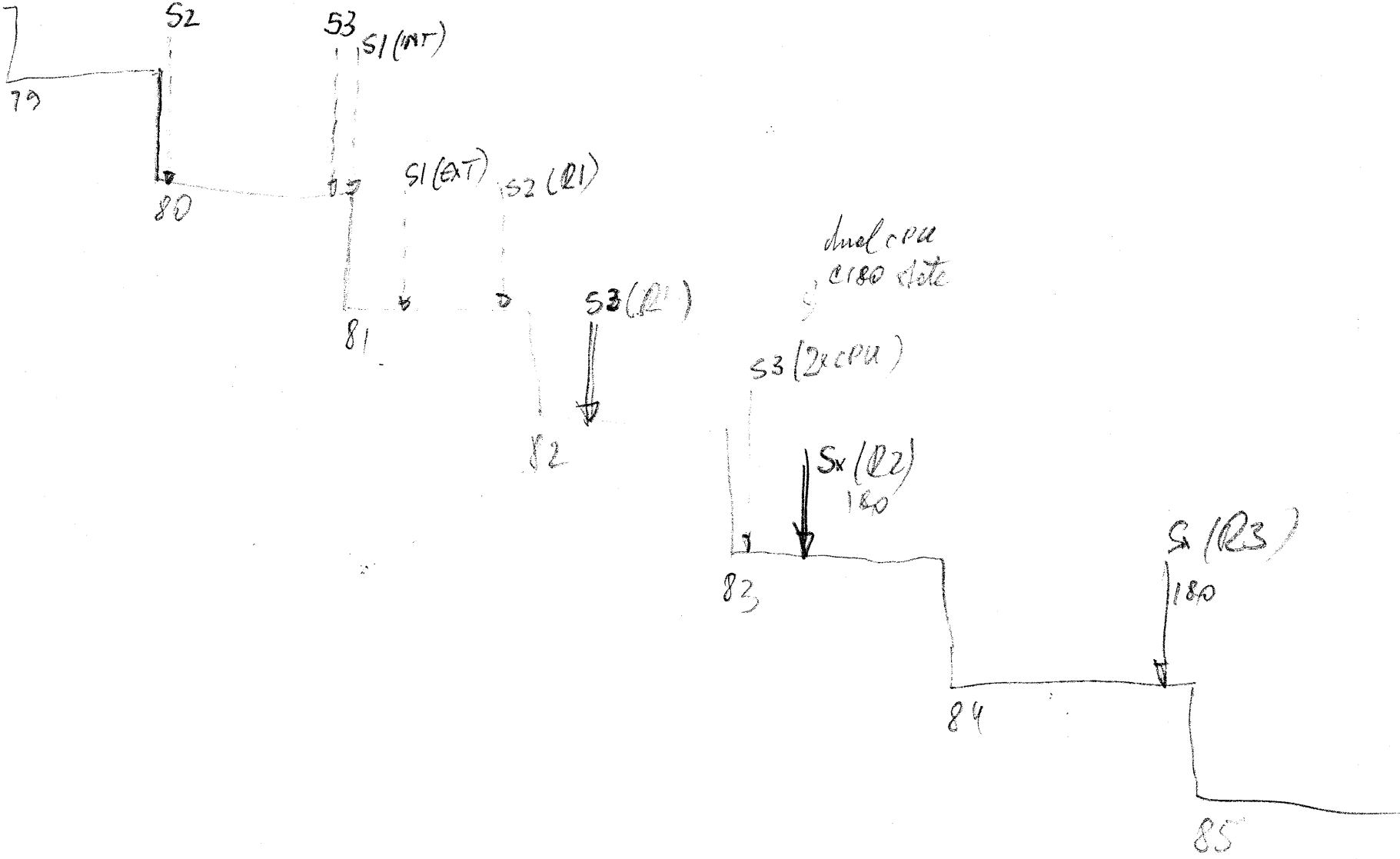
PERFORMANCE RELATIVE TO 172 = 1

175 = 8-11

176 = 13-18

CYBER 180 STATUS

- OBJECTIVES ESTABLISHED AND APPROVED
- PROCESSORS, MEMORIES AND I/O SYSTEM DEFINED - MIGDS
- S2 (P2, M2, INTERIM IOU) IN CHECKOUT PRIOR TO HANDOVER TO SOFTWARE
- P3 AND M3 IN DEVELOPMENT (POWER ON 4Q78)
- S1 IN DEVELOPMENT
- THETA IN DEVELOPMENT
- OPERATING SYSTEM DESIGN IN PROGRESS
  (PRELIMINARY GDS = ERS + GID AVAILABLE)
- IMPLEMENTATION LANGUAGE DEFINED - PASCAL-X
- INTERIM IMPLEMENTATION LANGUAGE, ASSEMBLERS, AND SIMULATORS (CP & PP) AVAILABLE
- WORK STARTED ON PRODUCT SET (FTN, COBOL, BASIC, S/M)


S2
S3
S1 (INT)
79
80
S1 (EXT)
S2 (R1)
81
S3 (R2)
82
dual cpu
c180 state
S3 (2x cpu)
Sx (R2)
180
83
Sx (R3)
180
84
85

NEW CYBER 180 FEATURES

HARDWARE FEATURES

- CPU SUPPORT OF:

  SCIENTIFIC/ENGINEERING
  CHARACTER HANDLING
  CYBER 170 STATE
  VIRTUAL MACHINE

- INSTRUCTION SET TO SUPPORT:

  FLOATING POINT EMPHASIZING EXECUTION SPEED
  LINKAGE BETWEEN CYBER 180 STATE AND CYBER 170 STATE
  CHARACTER MANIPULATION
  ADVANCED MEMORY MANAGEMENT
  PERFORMANCE MONITORING
  DEBUGGING

- MEMORY TO SUPPORT:

  CACHE
  SECDED
  RECONFIGURABILITY

- INPUT/OUTPUT UNIT TO SUPPORT:

  PERIPHERAL PROCESSORS
  50 MEGABIT/SEC CHANNEL
  24 MEGABIT/SEC CHANNEL
  2 PORT MUX

PHYSICAL CHARACTERISTICS

(TYPICAL SYSTEM)

S1 - SINGLE CABINET
AIR COOLED
50/60 Hz POWER
ECL 10K, ECL 10800 LOGIC TYPE
136 CIRCUITS PER PACKAGE
SEMICONDUCTOR MEMORY

S2 - THREE CONNECTED CABINETS
HYBRID COOLED
400 Hz POWER
ECL 10K LOGIC TYPE
136 CIRCUITS PER PACKAGE
SEMICONDUCTOR MEMORY (16K CIRCUIT)

S3 - THREE CONNECTED CABINETS
FREON COOLED P3, HYBRID COOLED M3 AND IOU
400 Hz POWER
LSI AND F100K LOGIC TYPE
150 ARRAY LSI BOARD
85 AND 98 F100K CIRCUITS PER PACKAGE
SEMICONDUCTOR MEMORY (16K CIRCUIT)

THETA - FOUR CONNECTED CABINETS
FREON COOLED PROCESSOR, HYBRID COOLED MEMORY AND IOU
400 Hz POWER
LSI AND F100K LOGIC TYPE
150 ARRAY LSI BOARD
85 AND 98 F100K CIRCUITS PER PACKAGE
BIPOLAR MEMORY (16K CIRCUIT)

RAM

RELIABILITY - AVAILABILITY - MAINTAINABILITY

BY DESIGN, 10 - 15% OF MANUFACTURING COST

FEATURES:

- DEDICATED MAINTENANCE CHANNEL
- CONCURRENT DIAGNOSTICS WITH SYSTEM OPERATIONS
- MAINTENANCE SOFTWARE SUPPORTS SYSTEM INTEGRITY
- ERROR LOGGING REGISTERS
- PARITY OR SECDED CODE ON ALL DATA PATHS
- DEGRADATION OF SYSTEM MEMORIES
- INSTRUCTION RETRY
- SYSTEM MAINTENANCE PANEL
- RIDE-THROUGH MOTOR/GENERATOR
- REMOTE MAINTENANCE
- FAULT ISOLATION DIAGNOSTICS
- CONFIGURATION ENVIRONMENT MONITOR (CEM)

Estimate
85% 180 user
came from 170

TECHNICAL MIGRATION FACILITIES

DUAL-STATE MAINFRAMES

MIXED-STATE SOFTWARE CAPABILITY 170 180 states

SOURCE LANGUAGE COMPILER

COMPATIBILITY

COMMON MAGNETIC TAPE FORMATS

CONVERSION AIDS

SOME EXTERNAL COMPATIBILITY IN OPERATING SYSTEMS

*MAJORITY OF LEVERAGE IS VIA FIRST THREE

MIGRATION STEPS FOR CUSTOMER

CURRENT CYBER 170 SYSTEM

MAINFRAME REPLACED BY DUAL-STATE MAINFRAME

- CONTINUE TO RUN 170 SOFTWARE
- RETAIN PERIPHERALS
- PRICE/PERFORMANCE IMPROVEMENT
- POSSIBLE MEMORY EXPANSION
- EASY DECISION–NO CONVERSION

AS FEATURES OF NOS/180 BECOME ATTRACTIVE AND ENHANCEMENT OF NOS/170 SLOWS

- GRADUALLY CONVERT TO NEW SOFTWARE WITH HARDWARE IN PLACE

VS.

- SUDDENLY CONVERT TO NEW SOFTWARE AND HARDWARE IF HE CHANGES VENDOR

PROCESSOR

- 1, 3, 9, 34 TIMES A CYBER 172
- REGISTER OPERATION:
  - -- 16 48 BIT A REGISTERS
  - -- 16 64 BIT X REGISTERS
- 128 TWO-REGISTER INSTRUCTIONS
  - -- 16 SCIENTIFIC
  - -- 18 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 16K – 32K CACHE MEMORY
- DUAL PROCESSOR OPTION

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 (5-7 MB)
- SUPPORTS MAINTENANCE SYSTEMS
  - -- DEDICATED MAINTENANCE CONTROL UNIT
  - -- SYSTEM CONSOLE - LOCAL/REMOTE
  - -- CONFIGURATION MONITOR
  - -- TWO PORT MULTIPLEXER

The PP doesn't know if it is a 6400 or 6600 with. It will just execute.

All dead start PPs are superseded in 6170 & 6150 PPs.

NOS/VE is CPU based i.e. PPs do not NOS/VE function, just I/O.

physically 1 → 20 this number def. by marketing.

for maintenance

PERIPHERALS

THREE CLASSES

- LOW PERFORMANCE -- COMMUNICATION LINES

  UNIT RECORD EQUIPMENT

  PRINTER
  CARD READER
  CRT TERMINALS

- INTERMEDIATE PERFORMANCE -- 6000 CHANNEL

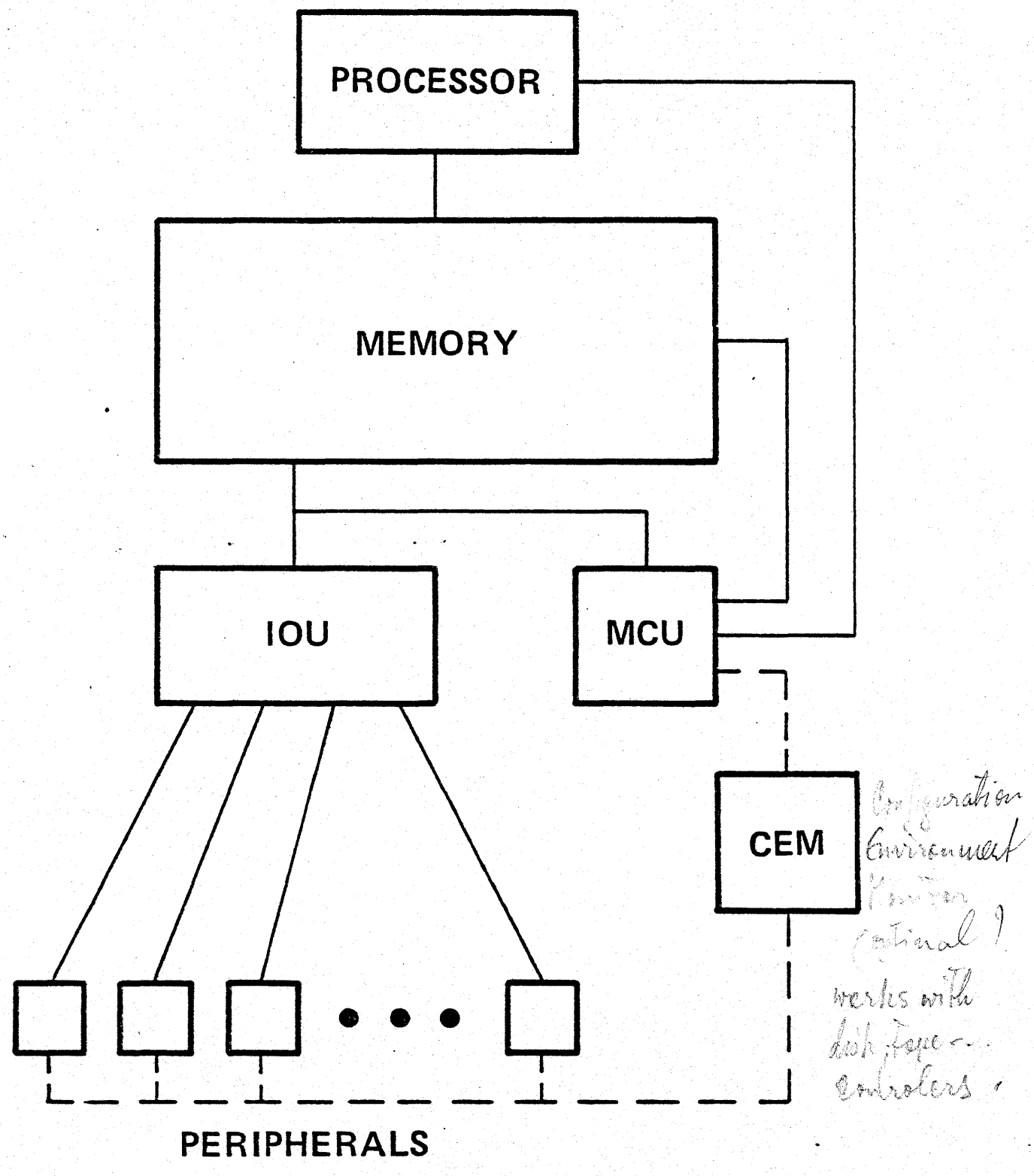
  MAGNETIC TAPE
  SERIAL HEAD DISK
  MASS STORAGE SYSTEM

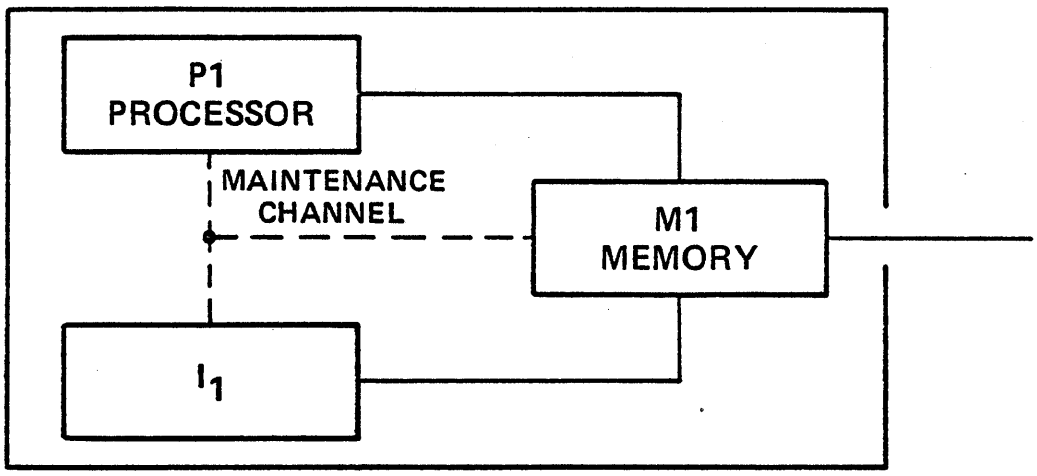
- HIGH PERFORMANCE -- 50 M BIT CHANNEL

  PARALLEL HEAD DISK
  SECONDARY STORAGE

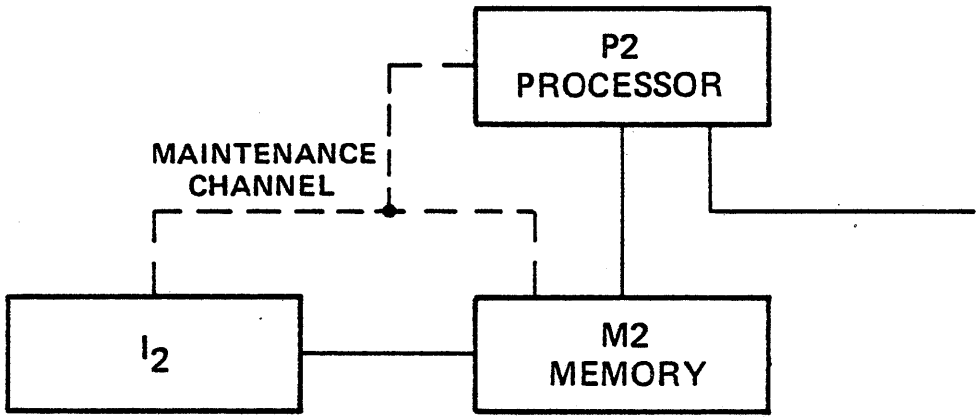
  CCD, BUBBLE OR EBAM

CYBER 180 CONFIGURATION

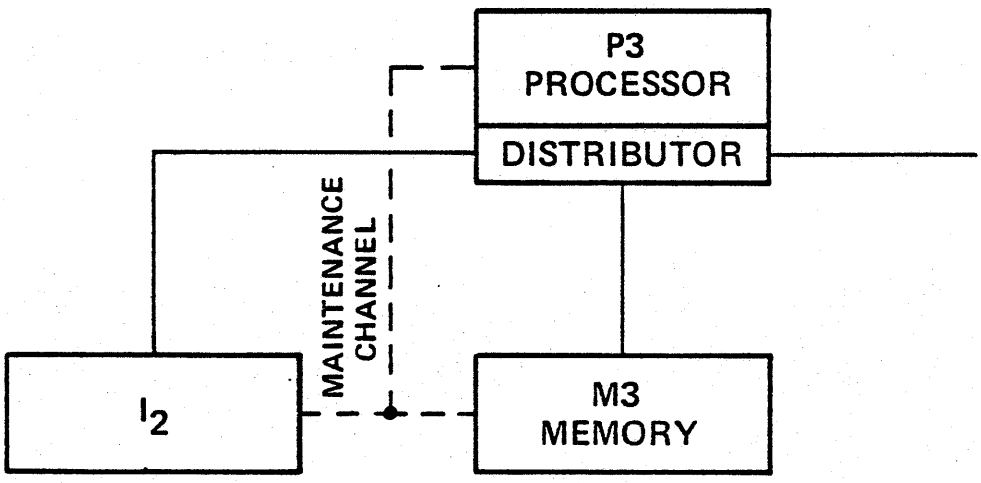




S1 SYSTEM



S2 SYSTEM

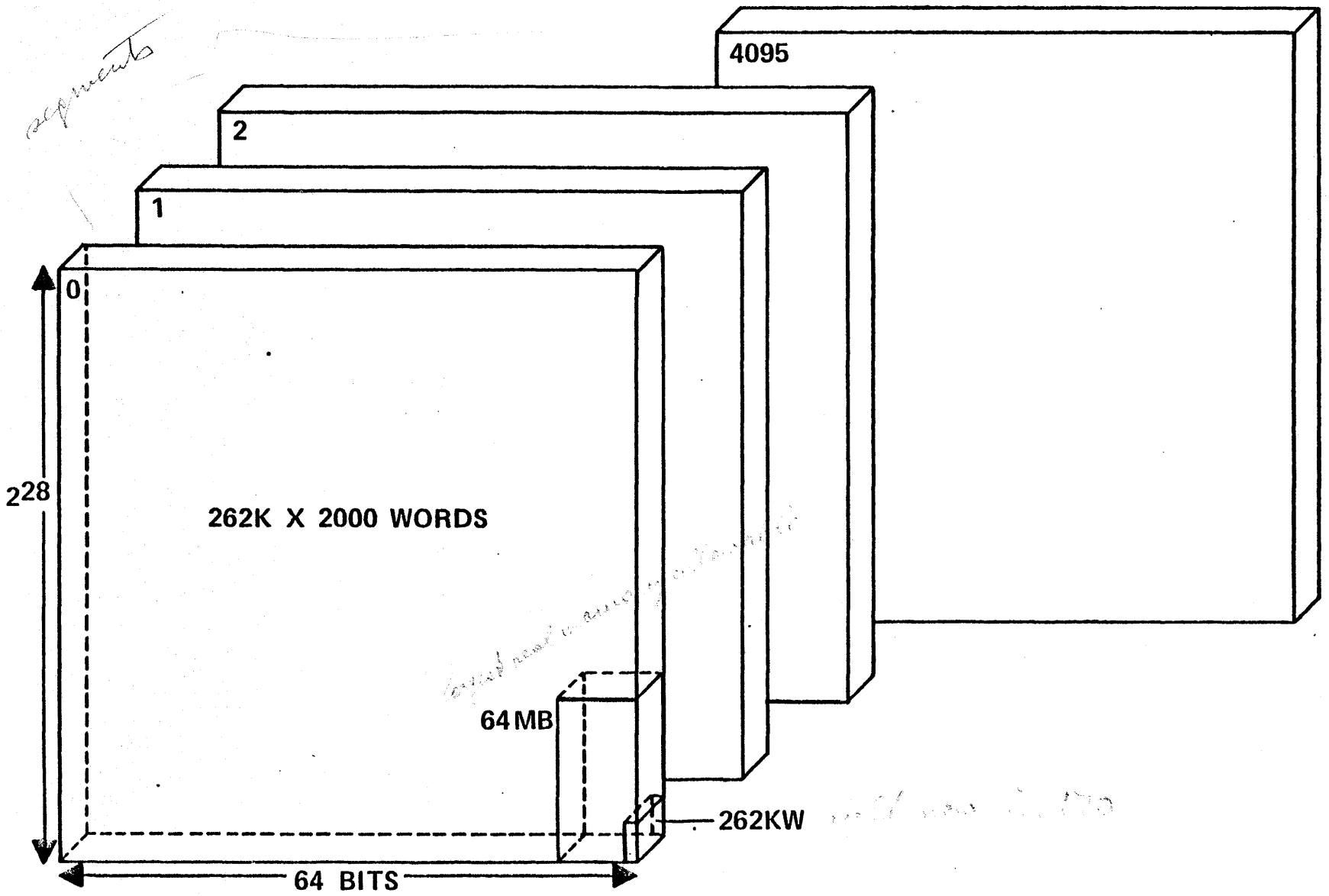


S3 SYSTEM

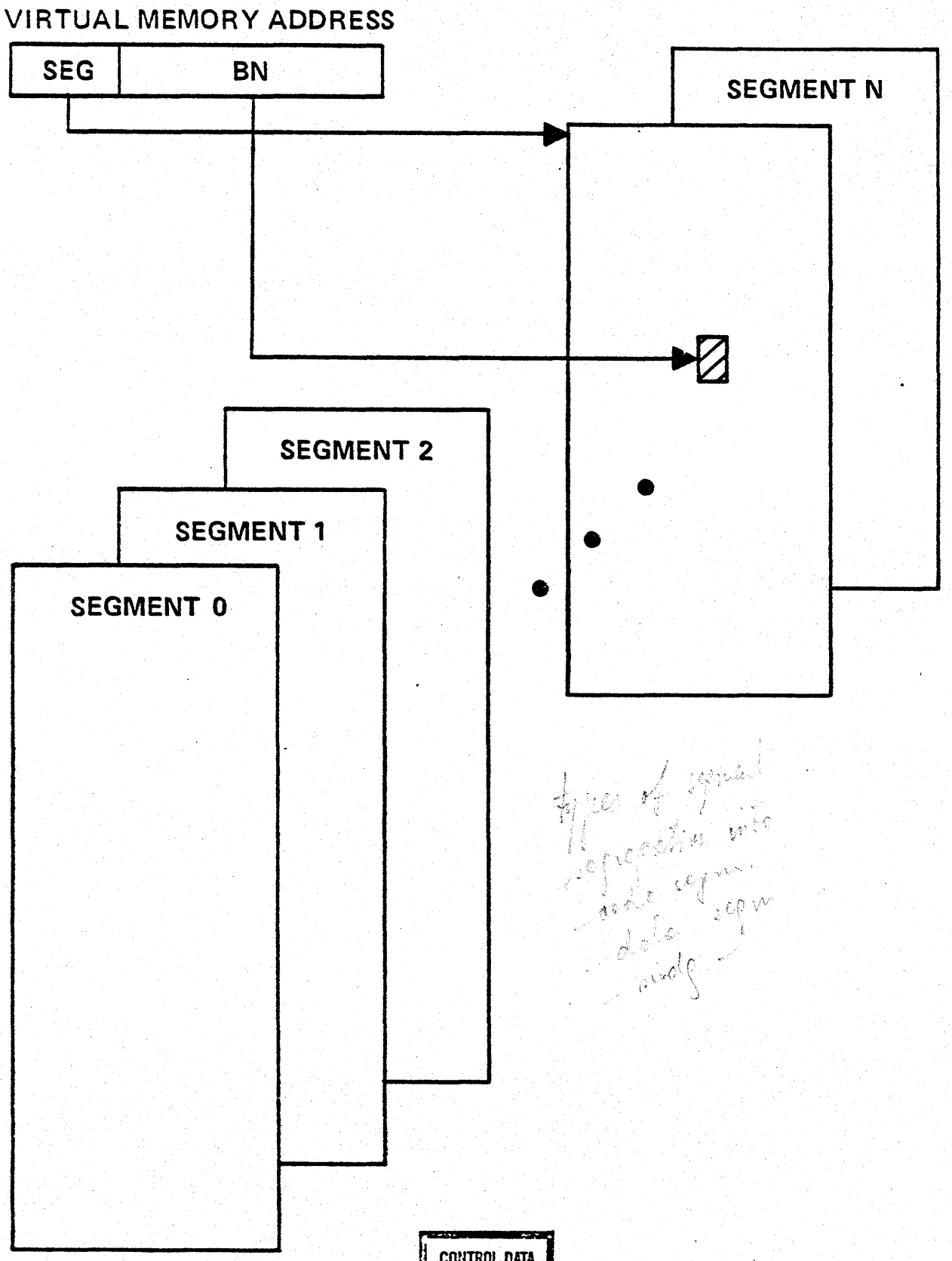
CYBER 180 VIRTUAL MEMORY

- ADDRESSING
  - PROGRAMMERS WORK EXCLUSIVELY IN VIRTUAL MEMORY
- PROTECTION
  - SYSTEM FROM USER
  - SYSTEM FROM SYSTEM (COMPARTMENTS)
  - USERS FROM OTHER USERS
  - USERS FROM SYSTEM

CYBER 180 VIRTUAL MEMORY ADDRESS SPACE



CYBER 180 VIRTUAL MEMORY USER ADDRESS SPACE



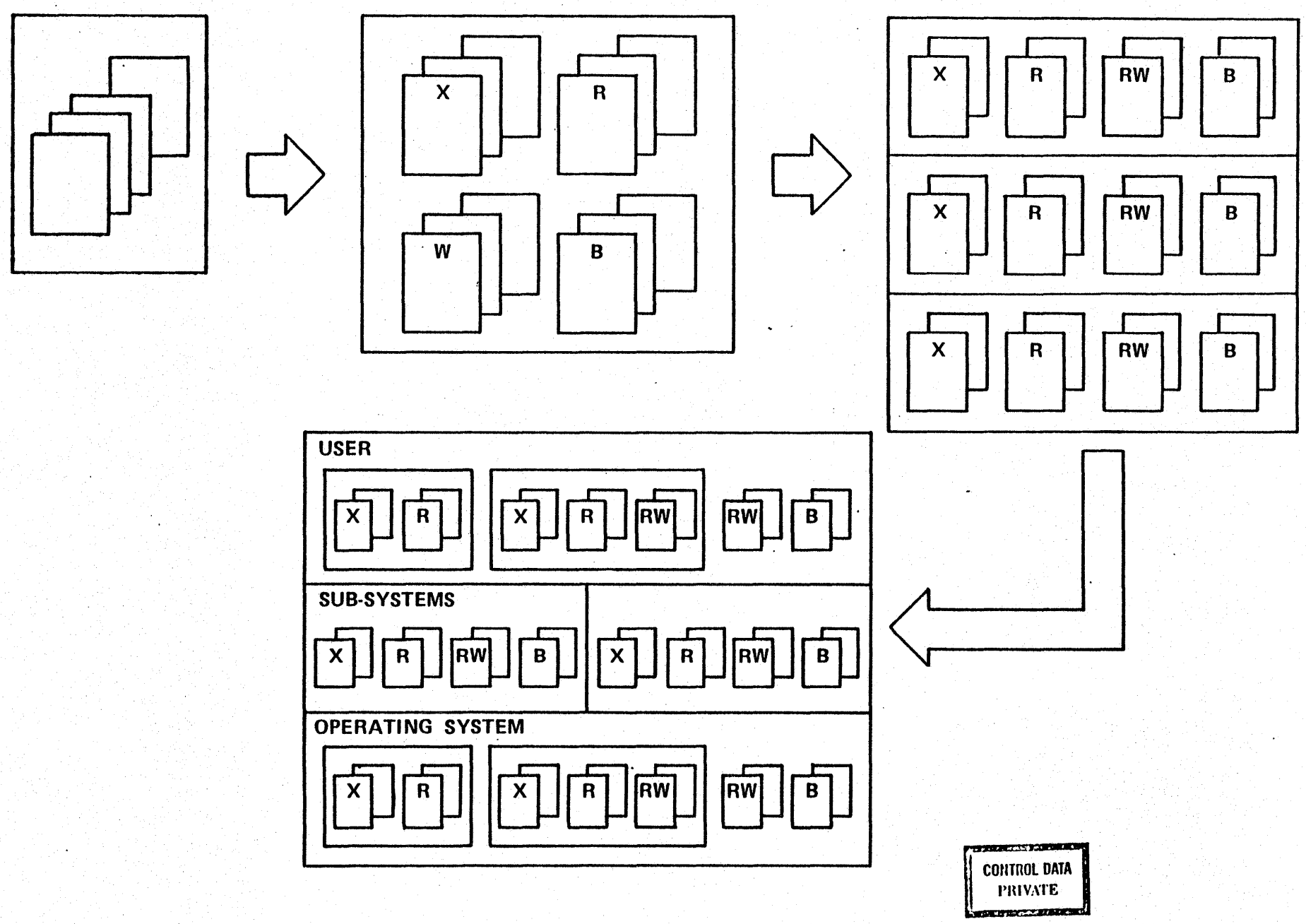
CYBER 180 VIRTUAL MEMORY

SEGMENT ATTRIBUTES

- BASIC PROTECTION ELEMENT
- ACCESS PRIVILEGES
  - READ
  - WRITE
  - EXECUTE
    - GLOBAL PRIVILEGE
    - LOCAL PRIVILEGE
- 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
  - GLOBAL LOCK FOR SEGMENT
  - LOCAL LOCK FOR SEGMENT

used in (b) the direction of indirect access by O.S

EXAMPLE OF PROTECTION WITHIN AN ADDRESS SPACE



CYBER 180 CALL/RETURN

- RECURSION
  - HARDWARE SUPPORT FOR STACK MANIPULATION
- PROTECTION SWITCHING
  - CONTROLLED MECHANISM FOR CROSSING RINGS
- STATE SWITCHING
  - TRANSFERRING CONTROL BETWEEN CYBER 170 AND CYBER 180 PROCESSES

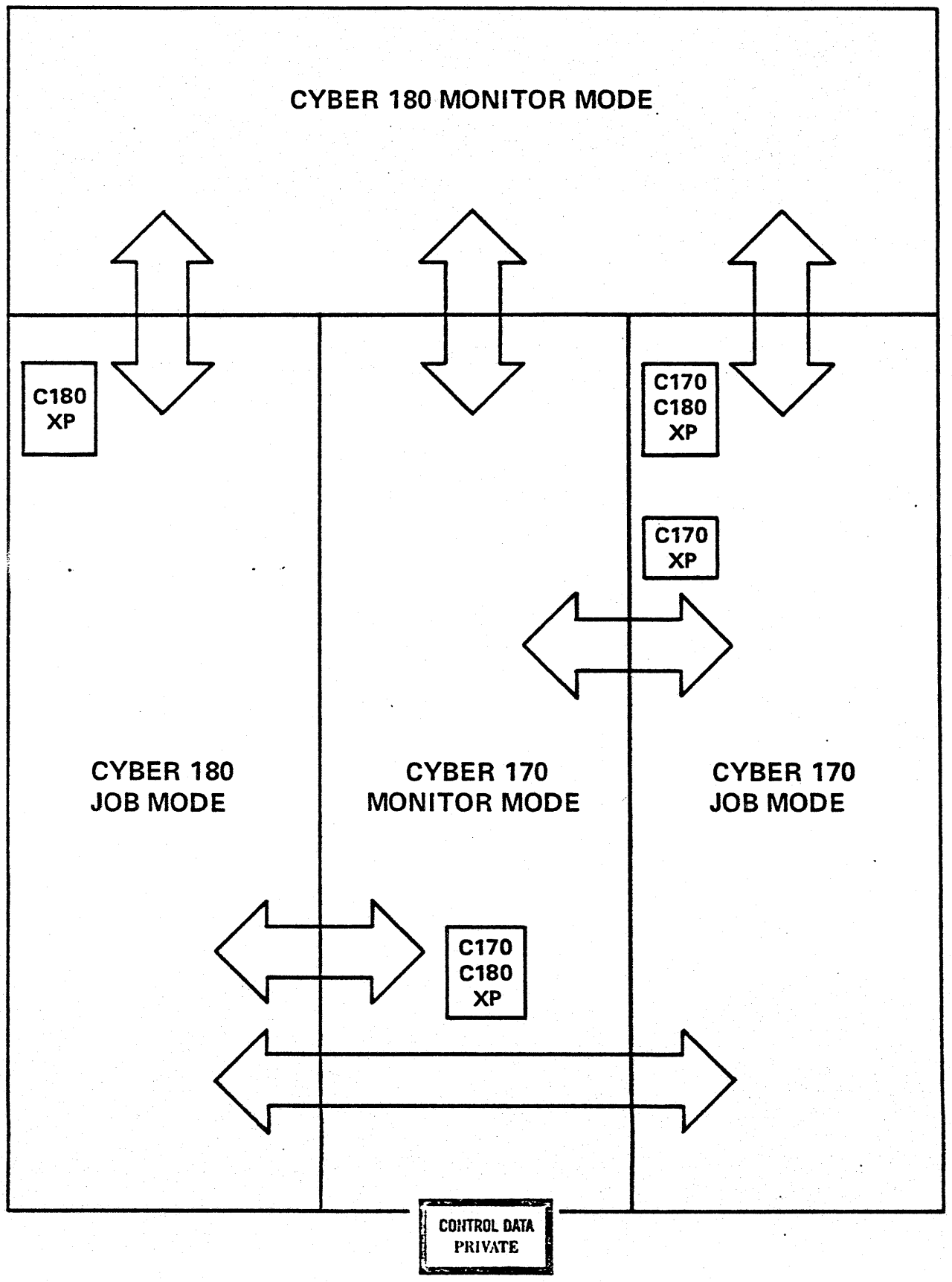
CYBER 180 INTERRUPTS

- EXCHANGE INTERRUPTS
  - FROM USER ADDRESS SPACE TO SYSTEM ADDRESS SPACE
    - SYSTEM CONDITIONS
    - MONITOR CONDITIONS
- TRAP INTERRUPTS
  - WITHIN USER ADDRESS SPACE
    - USER CONDITIONS
- CYBER 180 DESIGNED FOR O/S TO EXECUTE EXCLUSIVELY FROM CPU. PP'S ARE RESERVED STRICTLY FOR DEVICE DRIVERS AND SYSTEM MONITORING

CYBER 180 DUAL STATE

- EXECUTION OF CYBER 180 INSTRUCTIONS
- EXECUTION OF CYBER 170 INSTRUCTIONS
- SOFTWARE SUPPORT OF DUAL STATE JOB STREAMS
- ABILITY TO EXCHANGE BETWEEN STATES
- ABILITY TO CALL BETWEEN STATES
- ABILITY TO RUN DEDICATED SINGLE STATE

CYBER 180 DUAL STATE



CYBER 180 CHARACTER PROCESSING

- 8-BIT ASCII CHARACTER SET SUPPORTED
- 8-BIT EBCDIC CHARACTER SET SUPPORTED EXTERNALLY
- PACKED AND UNPACKED DECIMAL DATA TYPES SUPPORTED
- FULL SET OF CHARACTER HANDLING INSTRUCTIONS
  - ARITHMETIC (MIXED MODE)
  - MOVE
  - COMPARE
  - TRANSLATE
  - SCAN
  - EDIT
  - SUBSCRIPT CALCULATION AND RANGE CHECKING

CYBER 170/CYBER 180 DIFFERENCES

CYBER 170/CYBER 180 COMPARISON

CPU

CYBER 170	CYBER 180
60-BIT WORD	64-BIT WORD
WORD ADDRESSABLE	BYTE ADDRESSABLE
8 X-REGISTERS (60-BIT) 8 B-REGISTERS (18-BIT) 8 A-REGISTERS (18-BIT)	16 X-REGISTERS (64-BIT) 16 A-REGISTERS (48-BIT)
1'S COMPLEMENT ARITHMETIC	2'S COMPLEMENT ARITH. (CYBER 180) 1'S COMPLEMENT ARITH. (CYBER 170)
CYBER 170 INSTRUCTION SET	CYBER 180 INSTRUCTION SET AND CYBER 170 INSTRUCTION SET
THREE ADDRESS REGISTER TO REGISTER OPERATIONS	TWO ADDRESS REGISTER TO REGISTER OPERATIONS
SOME CHARACTER HANDLING INSTRUCTIONS (CMU)	FULL SET OF CHARACTER HANDLING INSTRUCTIONS

CYBER 170/CYBER 180 COMPARISON

MEMORY

CYBER 170	CYBER 180
MAX. 131K WORD USER ADDRESS SPACE	Concep. MAX. 4096 X 2^{31} BYTE USER ADDRESS SPACE
RA/FL RELOCATION	HARDWARE SEGMENTED MEMORY
17-BIT WORD ADDRESS WITHIN RA/FL ADDRESS SPACE	TWO PART ADDRESS: – SEGMENT NUMBER (12-BITS) – BYTE OFFSET WITHIN SEGMENT (32 BITS)
MAX. 262K WORD SYSTEM EXECUTABLE MEMORY	MAX. 64 MBYTE (POTENTIALLY 2^{31} BYTE) EXECUTABLE MEMORY
MEMORY MOVES AND SWAPPING TO MANAGE MEMORY	HARDWARE PAGING AND SWAPPING TO MANAGE MEMORY

CYBER 170/CYBER 180 COMPARISON

INPUT/OUTPUT

CYBER 170	CYBER 180
MAX. 2 X 10 12-BIT PPU'S	MAX. 4 X 5 16-BIT PP'S
MAX. 2 X 12 12-BIT CHANNELS	MAX. 6 X 4 12/16-BIT CHANNELS
EXECUTES 12-BIT PPU CODE	EXECUTES 12-BIT PPU CODE AND 16-BIT PP CODE
MEMORY SIZE: 4K X 12-BITS	MEMORY SIZE: 4K X 16-BITS
12-BIT WIDE DATA CHANNEL	12 OR 16-BIT WIDE DATA CHANNEL
60-BIT (5 X 12) ACCESS TO CENTRAL MEMORY	64-BIT (4 X 16) OR 60-BIT (5 X 12) ACCESS TO CENTRAL MEMORY
18-BIT CENTRAL MEMORY ADDRESS	28-BIT CENTRAL MEMORY ADDRESS
REAL CENTRAL MEMORY ADDRESSING	REAL CENTRAL MEMORY ADDRESSING
16-WORD DEADSTART PANEL	16-WORD DEADSTART PANEL PLUS 512-WORD ROM

CYBER 170/CYBER 180 COMPARISON

SYSTEM	
CYBER 170	CYBER 180
NO SHARED MEMORY AMONG USER ADDRESS SPACES	SEGMENTS SHARABLE AMONG USER ADDRESS SPACES
CODE/DATA COMBINED IN USER ADDRESS SPACE	SEGMENTS CAN BE READ, WRITE, EXECUTE
EXCHANGE OPERATION TO CALL CPU MONITOR	EXCHANGE OPERATION TO CALL CPU MONITOR
O/S RUNS AT SYSTEM CONTROL POINTS OR IN PPU'S	MOST SYSTEM CODE RUNS IN USER ADDRESS SPACE AND OBEYS SAME CALLING, LOADING/LINKING CONVENTIONS
SYSTEM AND USER CODE PROTECTED BY ADDRESS SPACE	SYSTEM AND USER CODE PROTECTED BY ADDRESS SPACE AND SYSTEM CODE IN USER ADDRESS SPACE PROTECTED BY RING MECHANISM AND KEY/LOCKS

VIRTUAL MEMORY & CALL/RETURN

CYBER 180 VIRTUAL MEMORY

- SEGMENTATION AND PAGING

- SEGMENT
  - COMPONENT OF "VIRTUAL ADDRESS SPACE"
  - POSSESSES ATTRIBUTES
    - LENGTH
    - READ, WRITE, EXECUTE
    - RINGS
    - GLOBAL AND LOCAL KEY/LOCKS

- PAGE
  - UNIT OF REAL MEMORY MANAGEMENT
  - PARTIAL RESIDENCE OF SEGMENTS
  - DOES NOT POSSESS ATTRIBUTES

SEGMENT PROTECTION ATTRIBUTES

1. ACCESS ATTRIBUTES

   - READ
   - WRITE
   - EXECUTE

2. RINGS

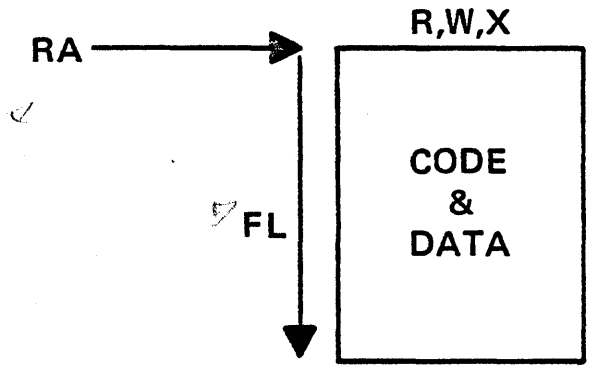
   - 15 HIERARCHICAL LEVELS OF PROTECTION
   - "ONE WAY" PROTECTION

3. GLOBAL AND LOCAL KEY/LOCK

   - 63 PROTECTION COMPARTMENTS
   - "TWO WAY" ISOLATION AND SEPARATION

ACCESS TO A SEGMENT IS ONLY PERMITTED WHEN ALL THREE PROTECTION TESTS ARE PASSED

CYBER 170 ADDRESS SPACE

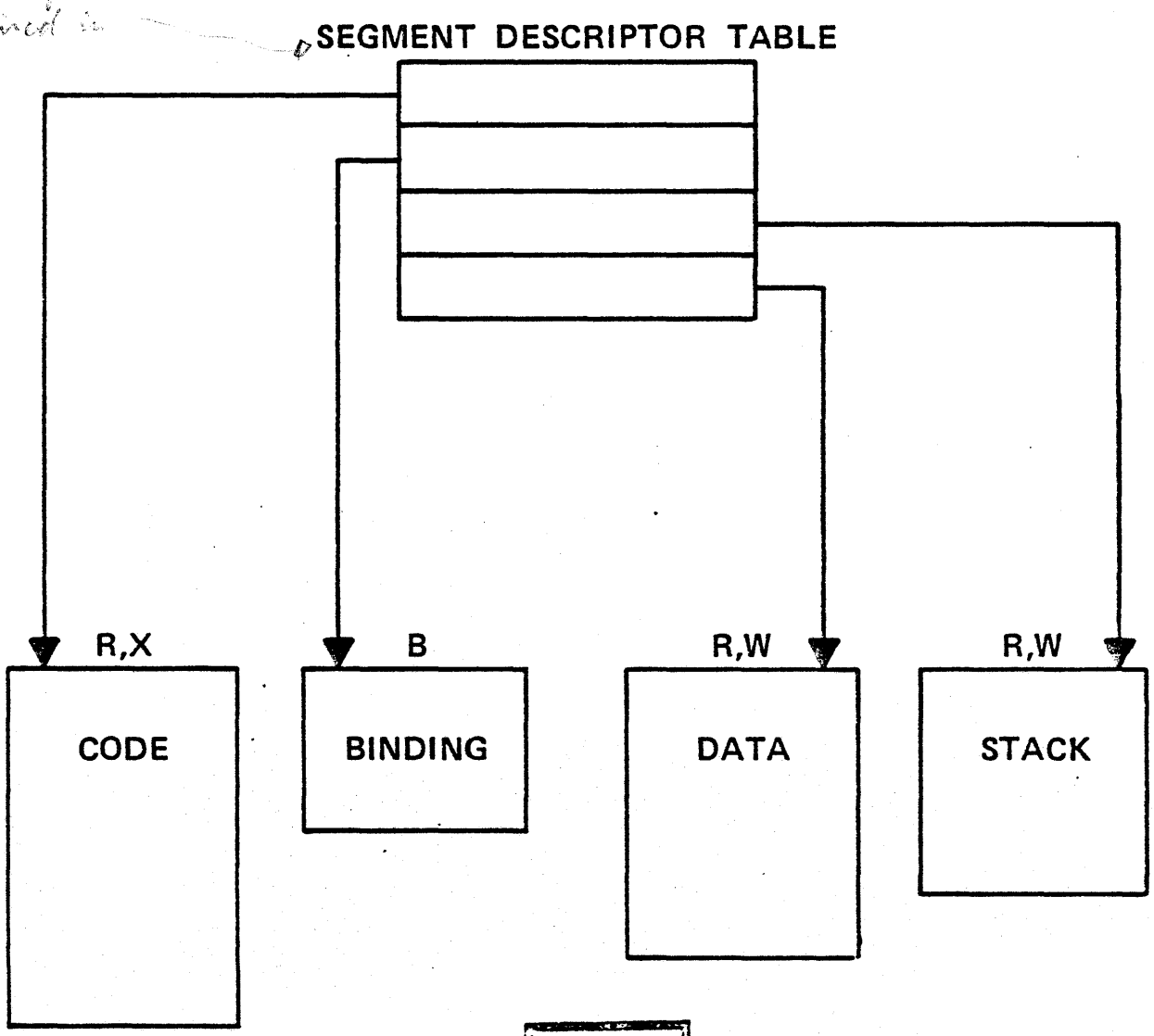


in 170

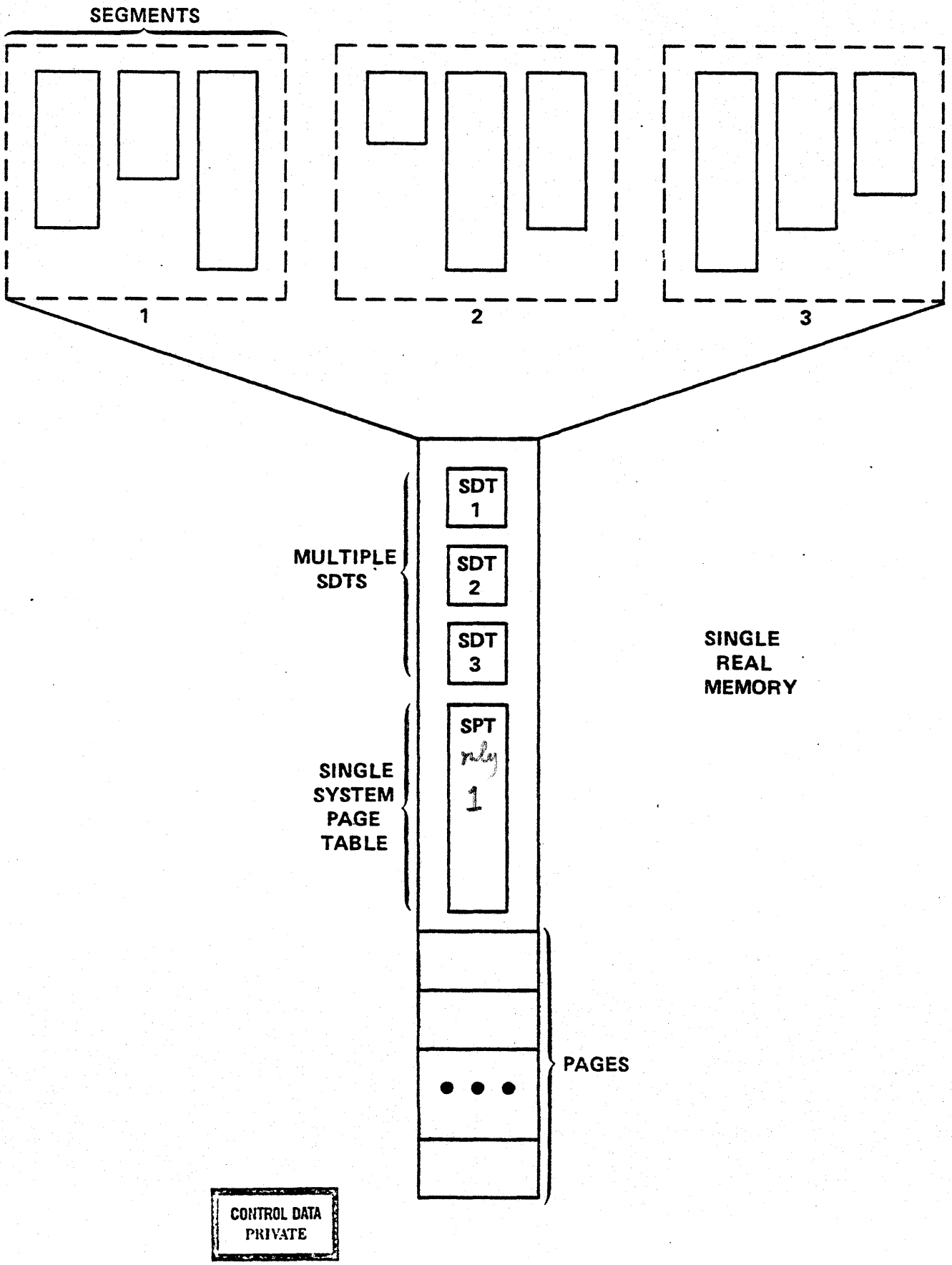
in 180

User space defined in

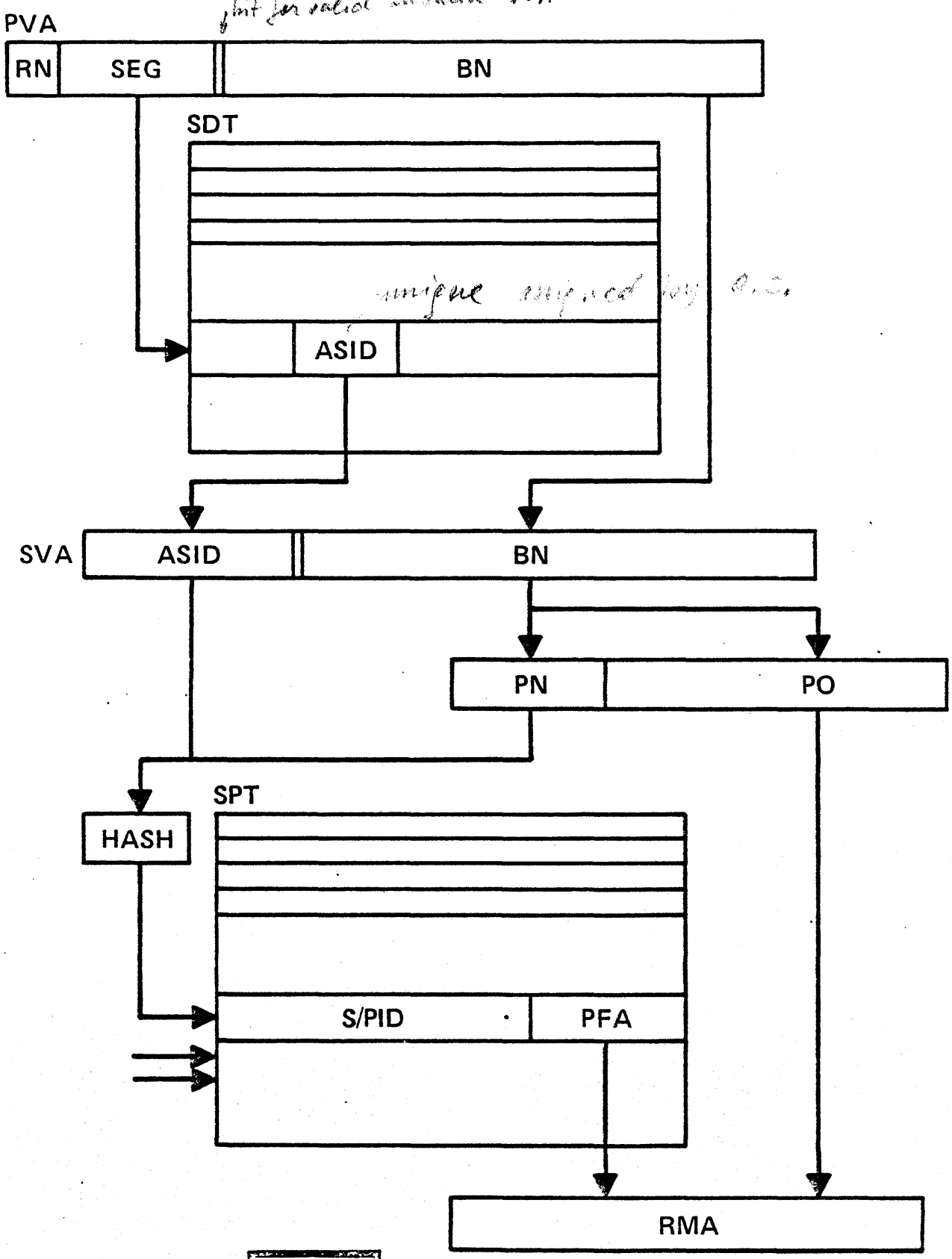
CYBER 180 ADDRESS SPACE



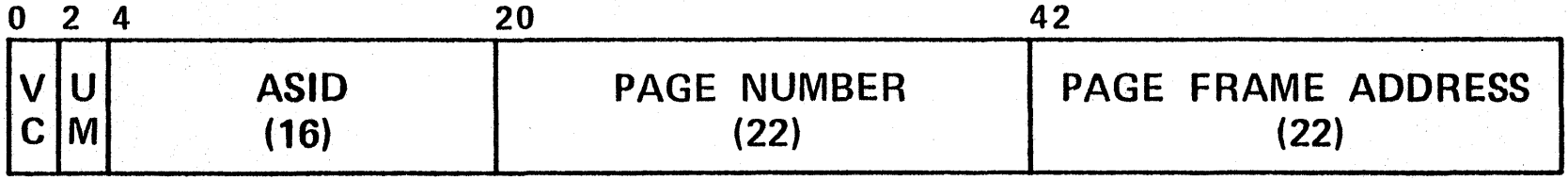
MULTIPLE VIRTUAL ADDRESS SPACE



ADDRESS TRANSLATION



PAGE DESCRIPTOR



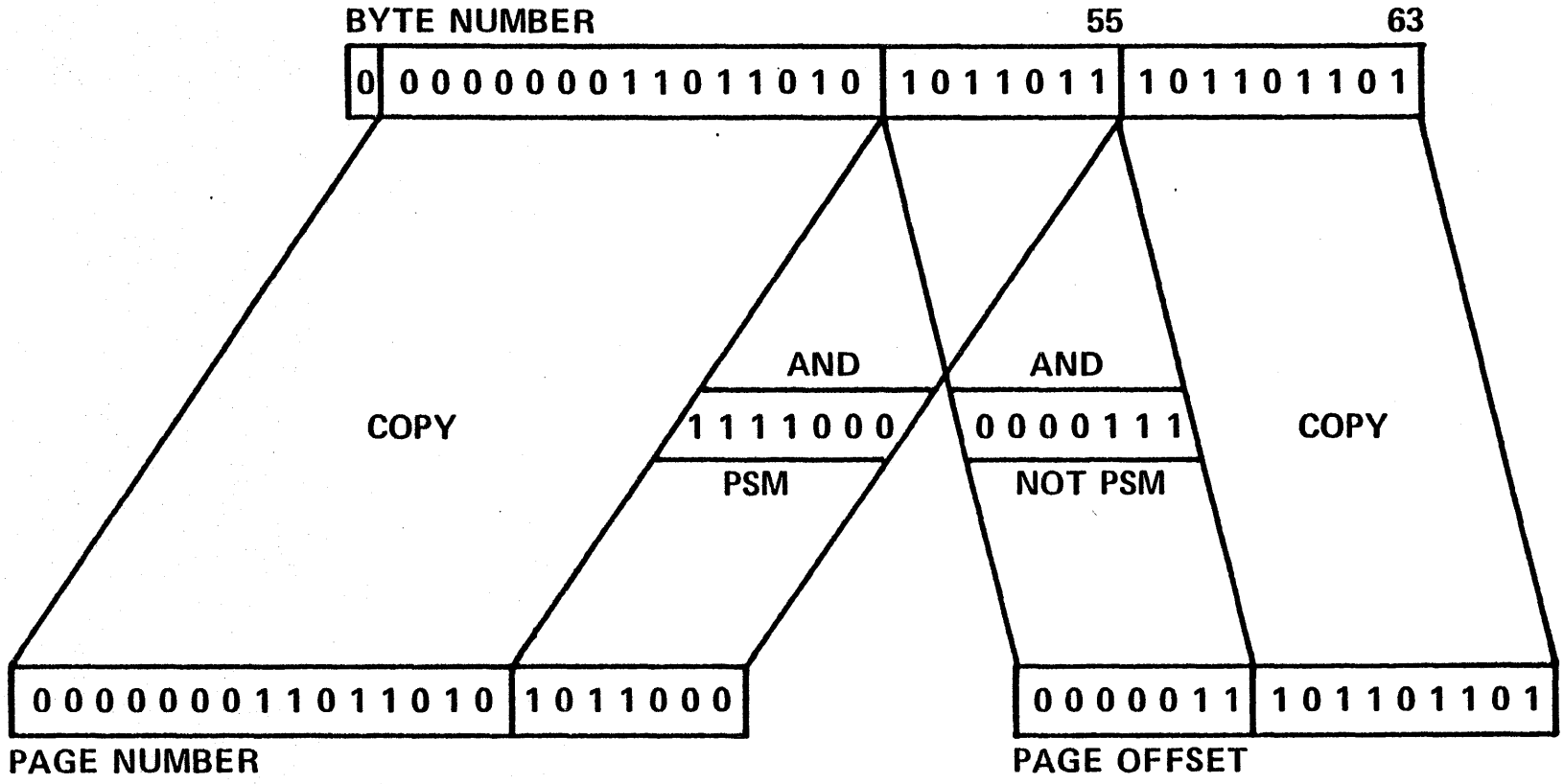
VC

00 INVALID ENTRY, STOP SEARCH
01 INVALID ENTRY, CONTINUE SEARCH
10 VALID ENTRY, STOP SEARCH
11 VALID ENTRY, CONTINUE SEARCH

UM

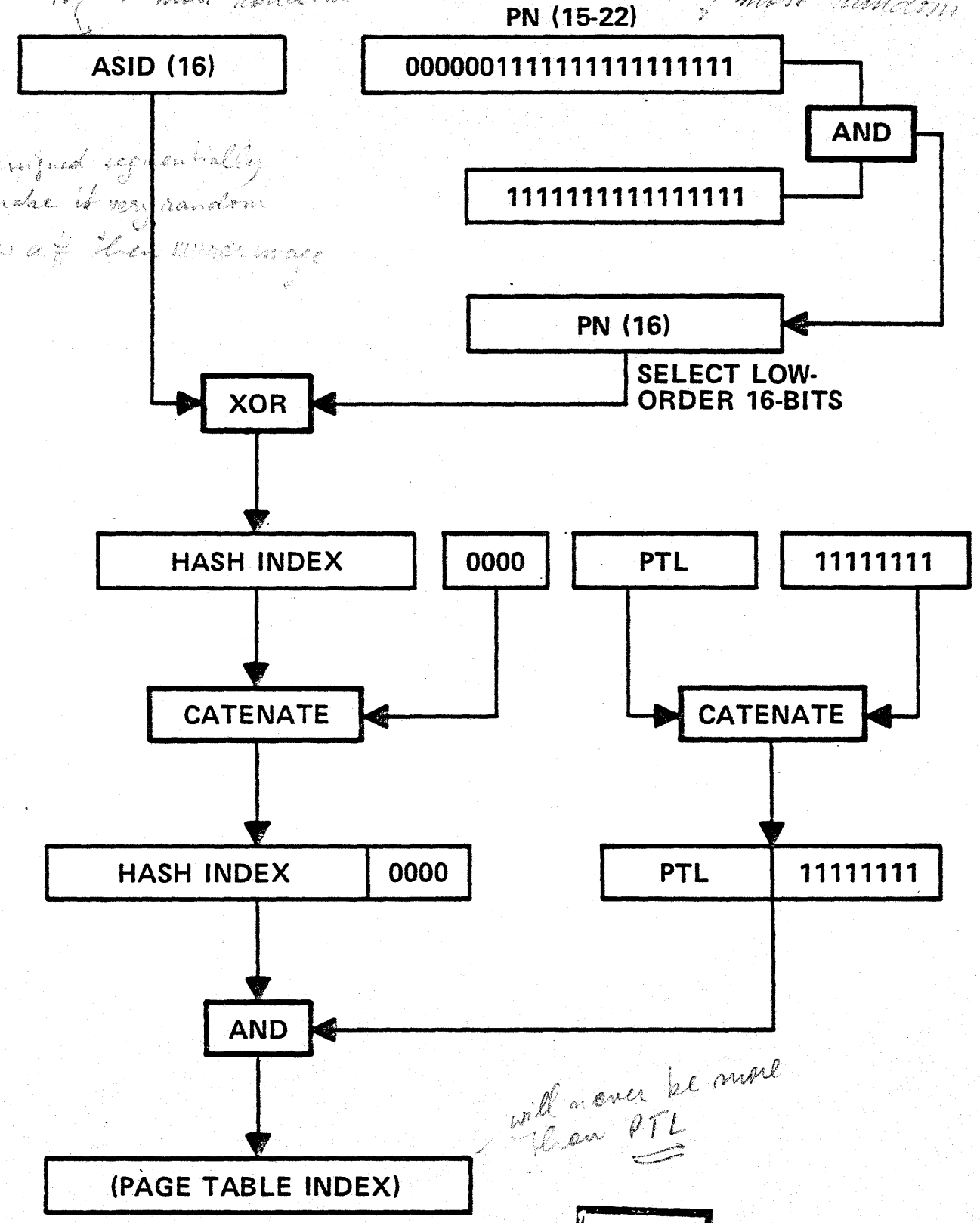
00 FRESH PAGE
01 (RESERVED)
10 USED BUT NOT MODIFIED
11 USED AND MODIFIED

FORMATION OF PAGE NUMBER AND PAGE OFFSET

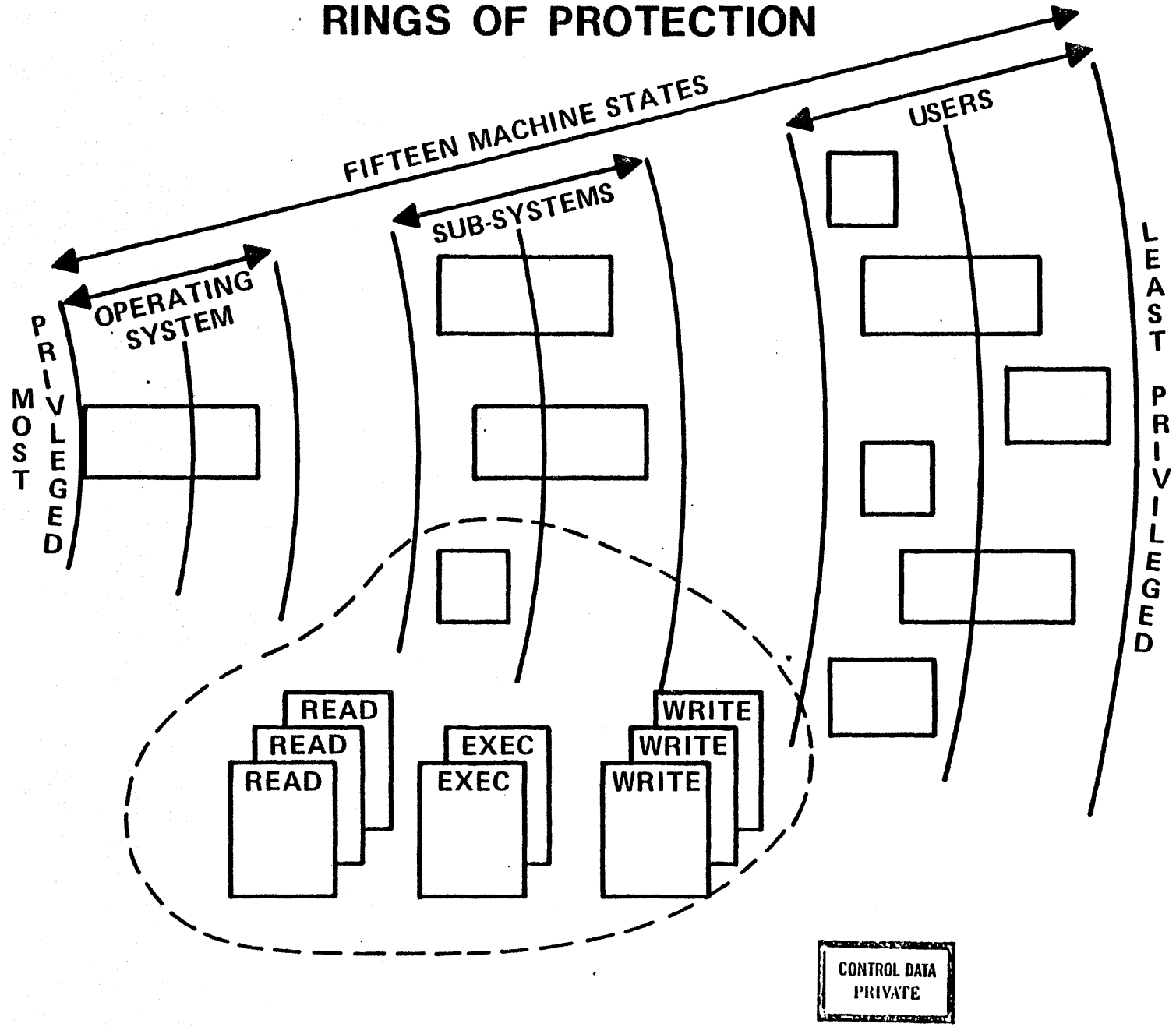


i.e. offset for page description
into page table

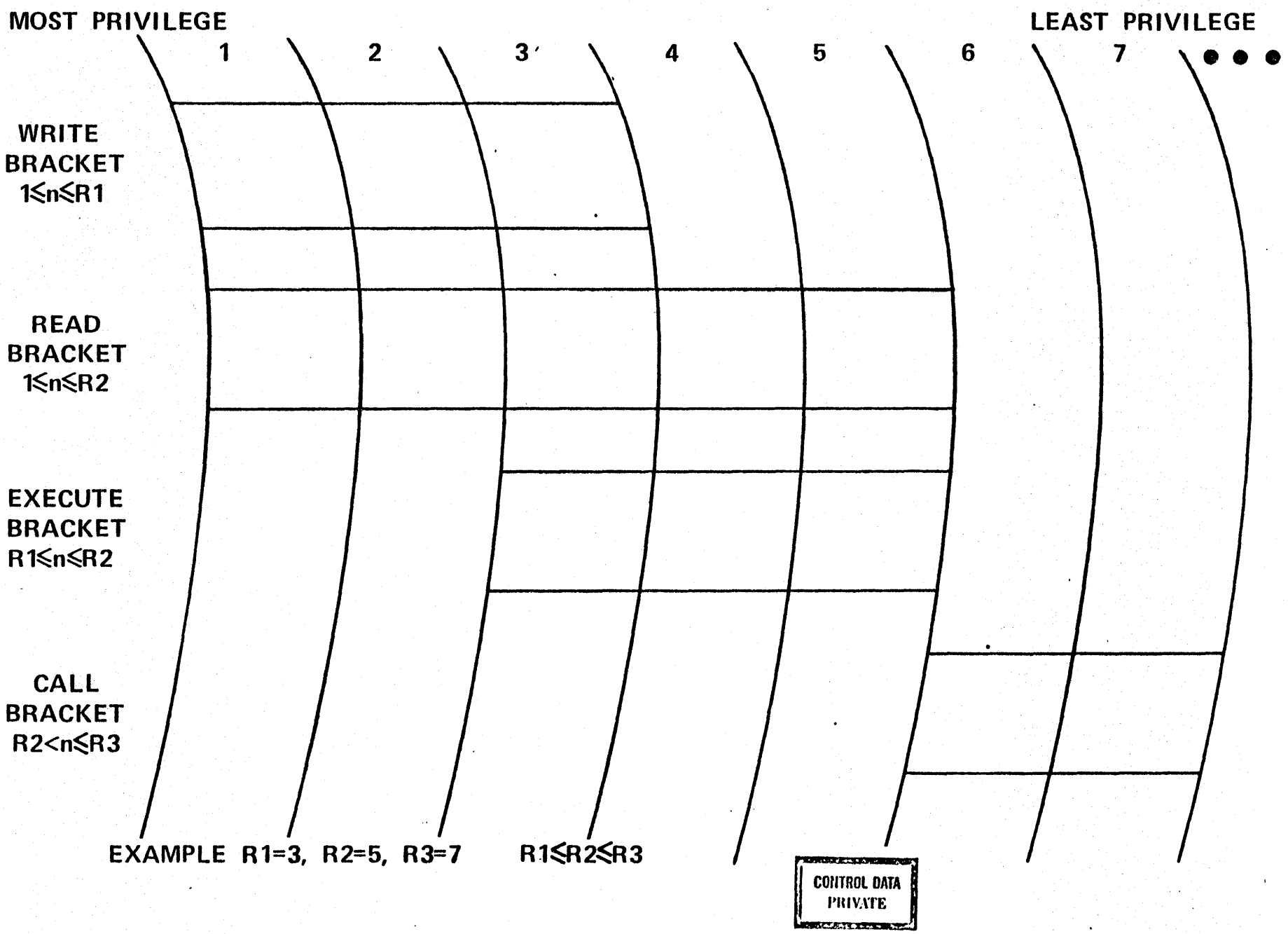
HASHING ALGORITHM



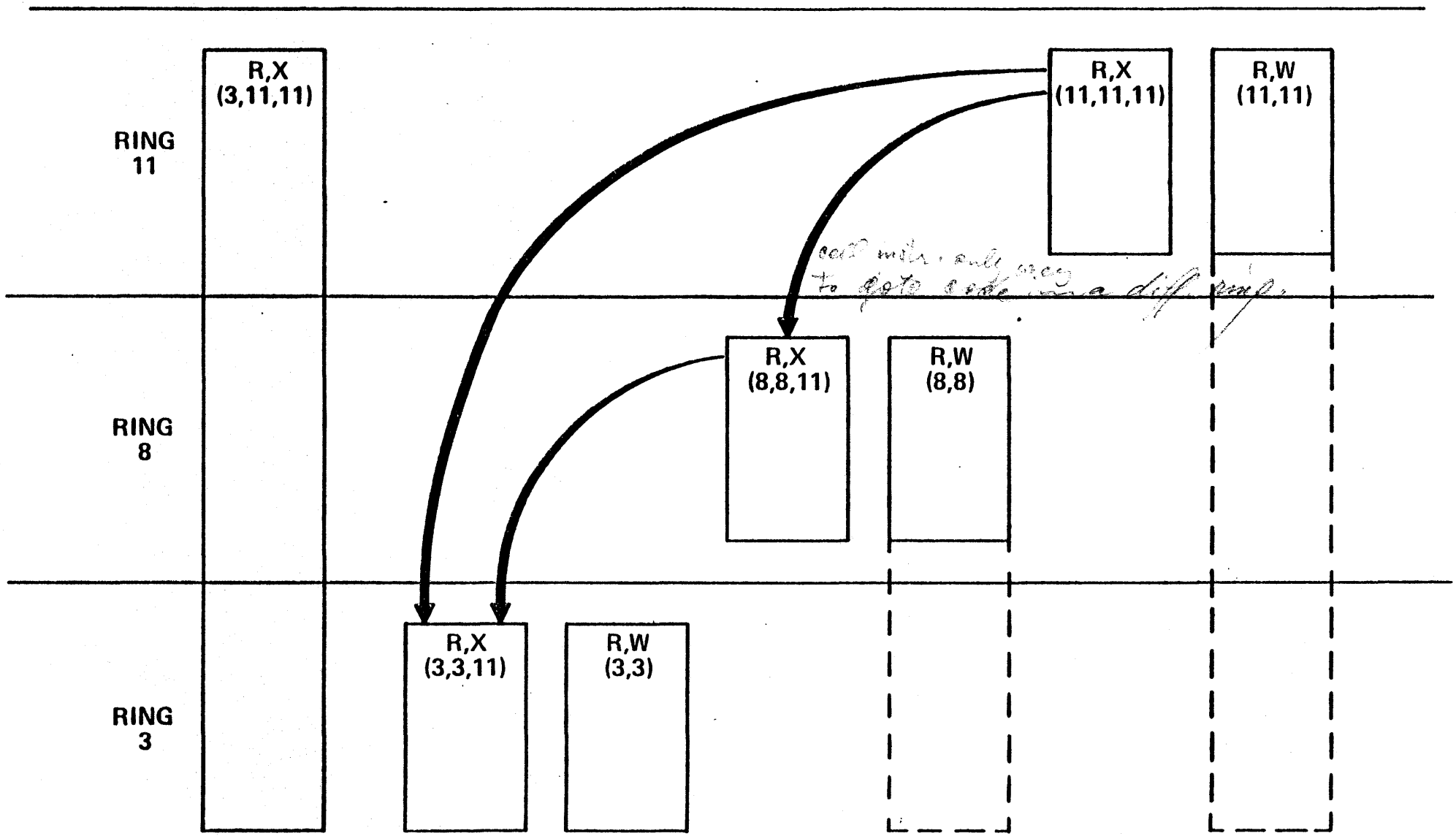
RINGS OF PROTECTION



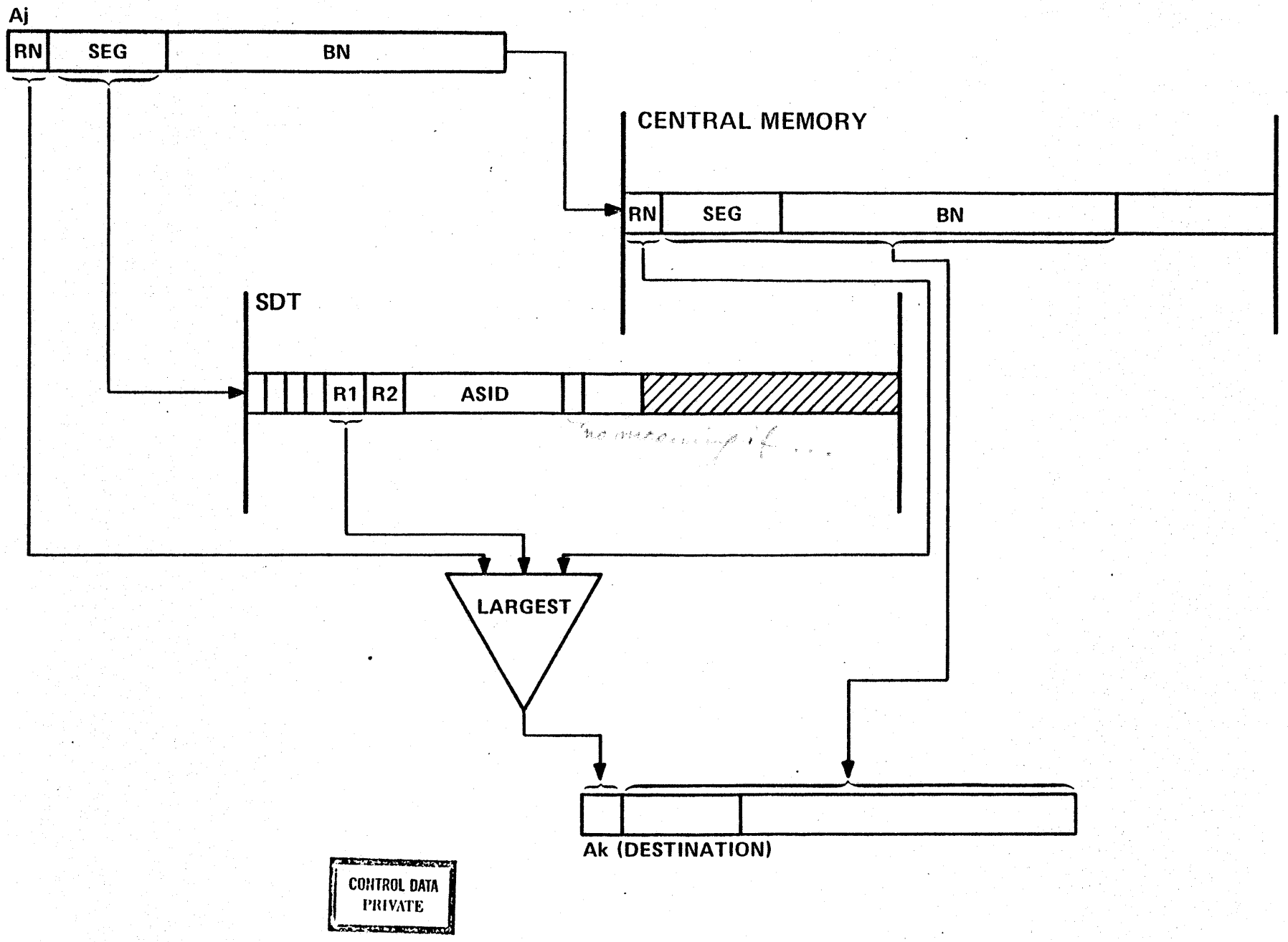
RING BRACKETS



EXAMPLE OF RING BRACKETS

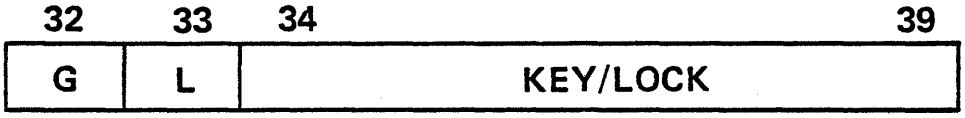


A-REGISTER RING VOTING



GLOBAL AND LOCAL KEY/LOCK

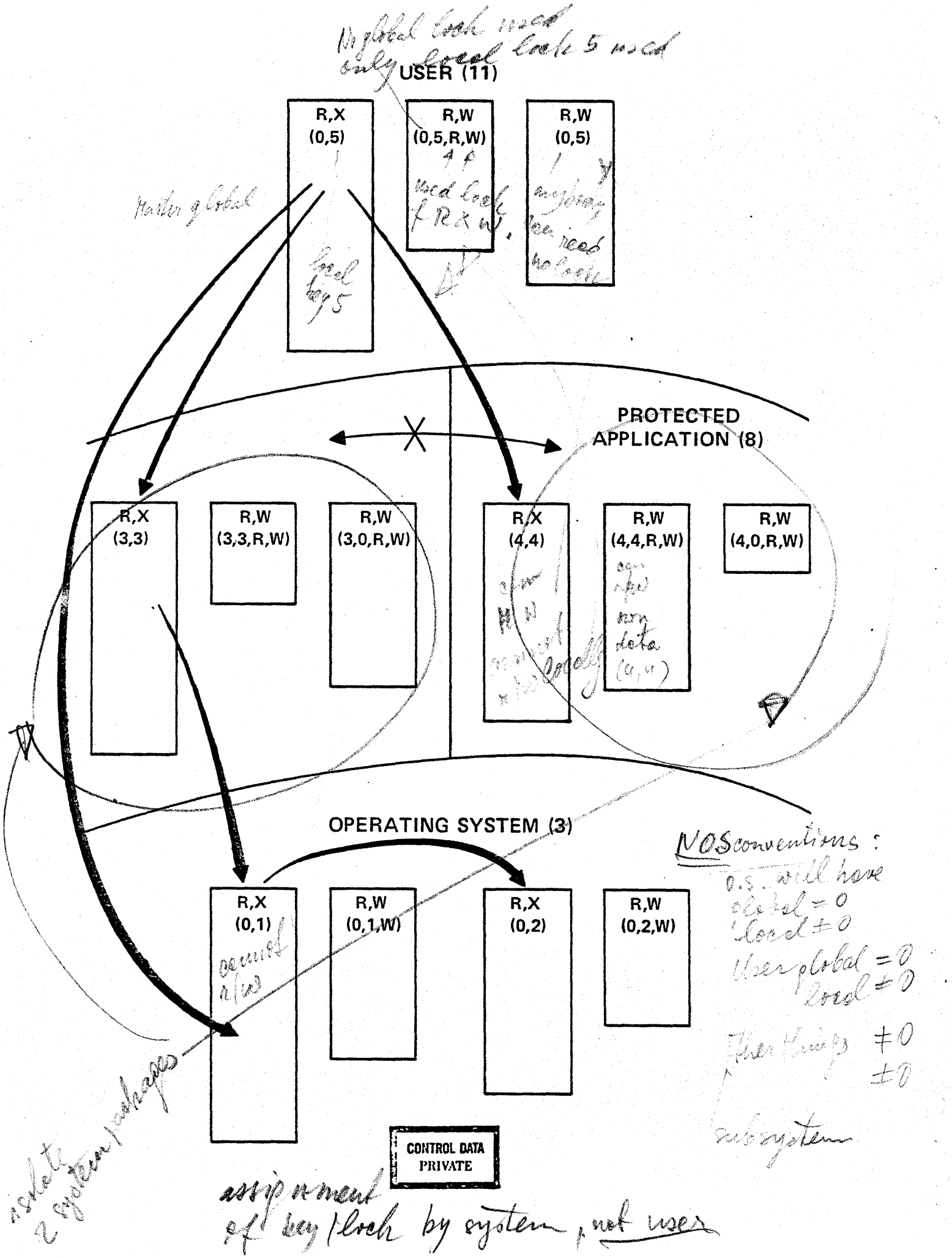
- SINGLE KEY OR LOCK PER SEGMENT
  - KEY WHEN SEGMENT IS EXECUTING
  - LOCK WHEN SEGMENT IS REFERENCED

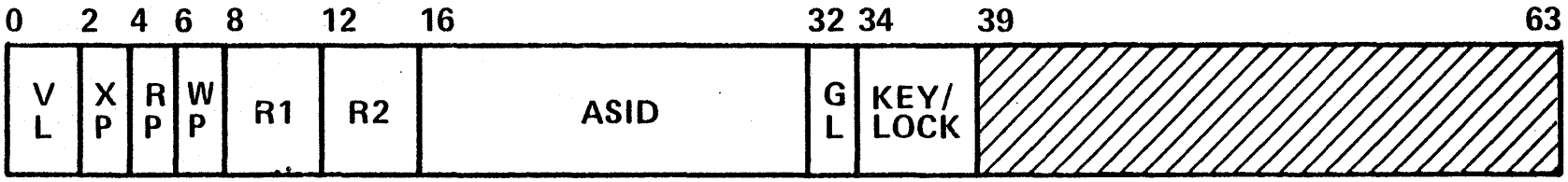


PROCEDURE G	L	GLOBAL	LOCAL
0	0	MASTER KEY	MASTER KEY
0	1	MASTER KEY	6 BIT KEY
1	0	6 BIT KEY	MASTER KEY
1	1	6 BIT KEY	6 BIT KEY

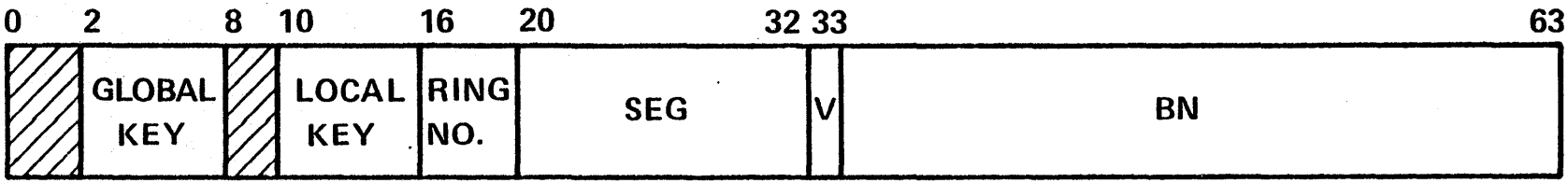
DATA G	L		
0	0	NO LOCK	NO LOCK
0	1	NO LOCK	6 BIT LOCK
1	0	6 BIT LOCK	NO LOCK
1	1	6 BIT LOCK	6 BIT LOCK

- KEY LOCK TEST PERFORMED FOR READ ACCESS IF READ CONTROLLED BY KEY/LOCK IN SDT
- KEY LOCK TEST PERFORMED FOR WRITE ACCESS IF WRITE CONTROLLED BY KEY/LOCK IN SDT


USER (11)
R,X
(0,5)
R,W
(0,5,R,W)
R,W
(0,5)
PROTECTED
APPLICATION (8)
R,X
(3,3)
R,W
(3,3,R,W)
R,W
(3,0,R,W)
R,X
(4,4)
R,W
(4,4,R,W)
R,W
(4,0,R,W)
OPERATING SYSTEM (3)
R,X
(0,1)
R,W
(0,1,W)
R,X
(0,2)
R,W
(0,2,W)
CONTROL DATA
PRIVATE



SEGMENT DESCRIPTOR TABLE ENTRY – SDE



PROGRAM ADDRESS REGISTER – P-REGISTER

SDT } in CM
PDT

CYBER 180 BUFFER MEMORIES (MODEL DEPENDENT)

in Processor

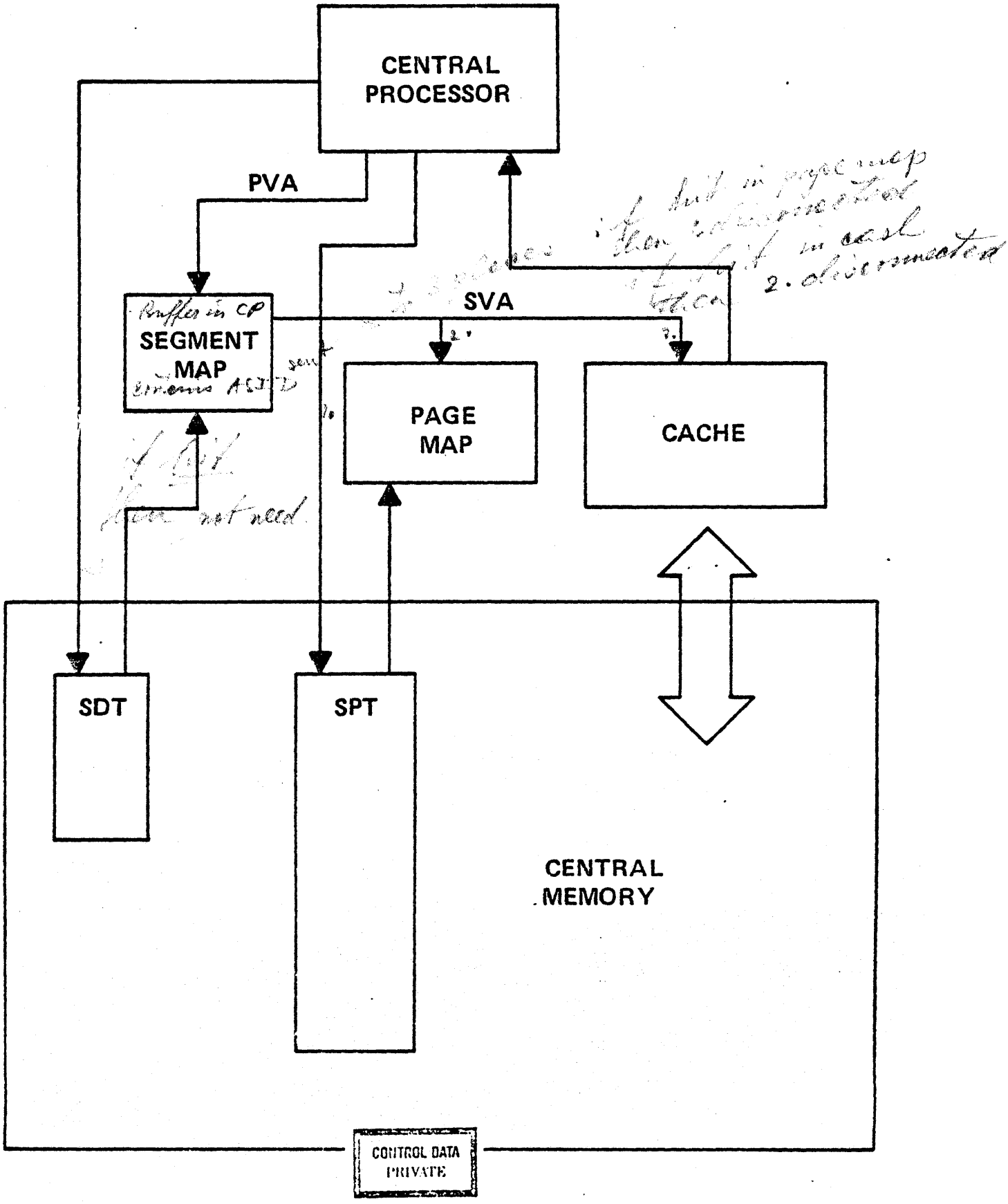
- ADDRESS TRANSLATION BUFFER MEMORIES
  - SEGMENT MAP — a memory to hold a copy of SDT part
  - PAGE MAP
- CENTRAL MEMORY BUFFER MEMORY
  - CACHE

P1 Mr Clark

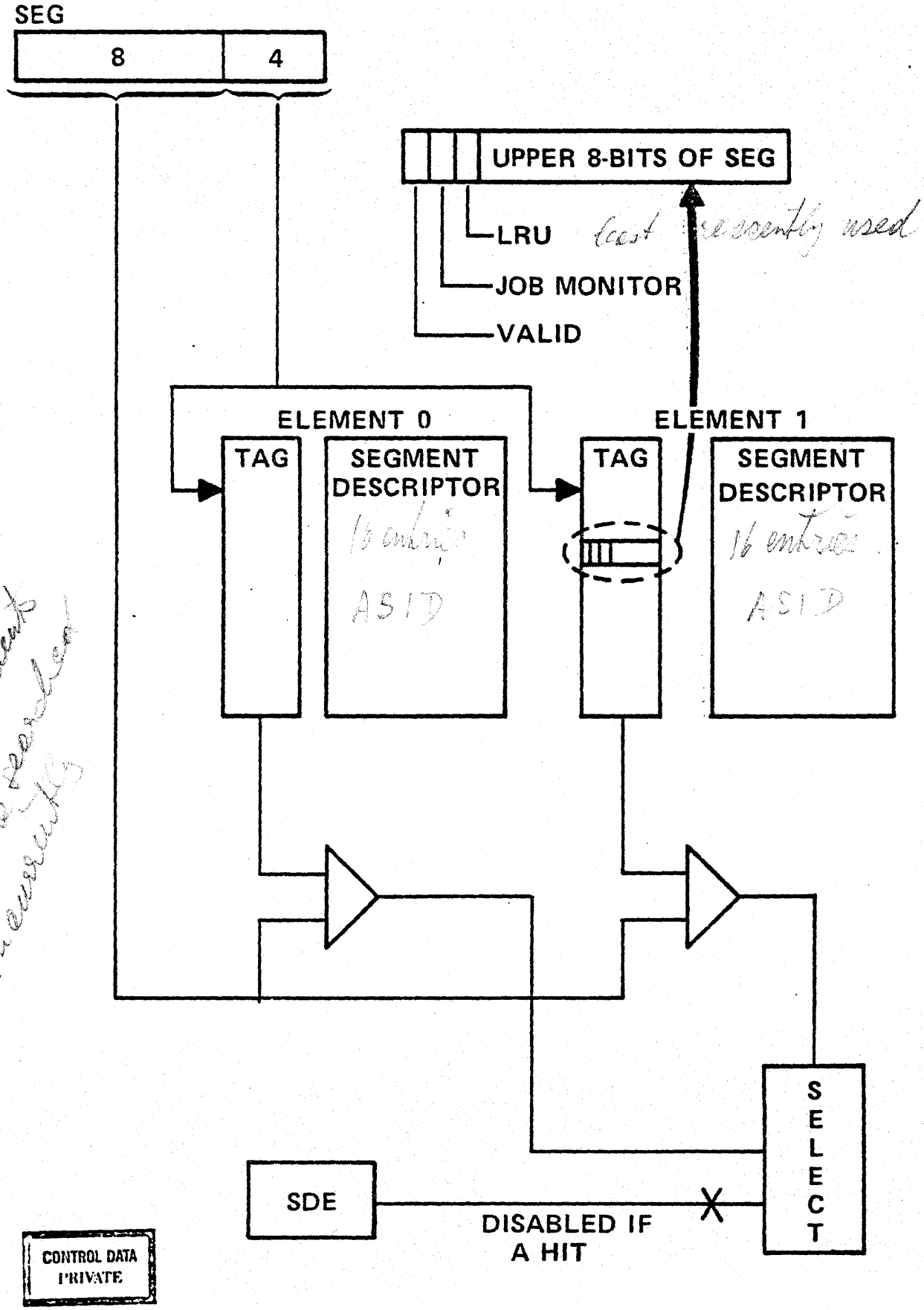
		Cache Instr hit rate (I)	Map hit rate (M)
P1	16 KB	0.92	0.98
P3	32 KB	0.95	0.99
Theta	32 KB	0.95	0.995

- PREDOMINANTLY HARDWARE MANAGED
- SOFTWARE RESPONSIBLE FOR PURGING

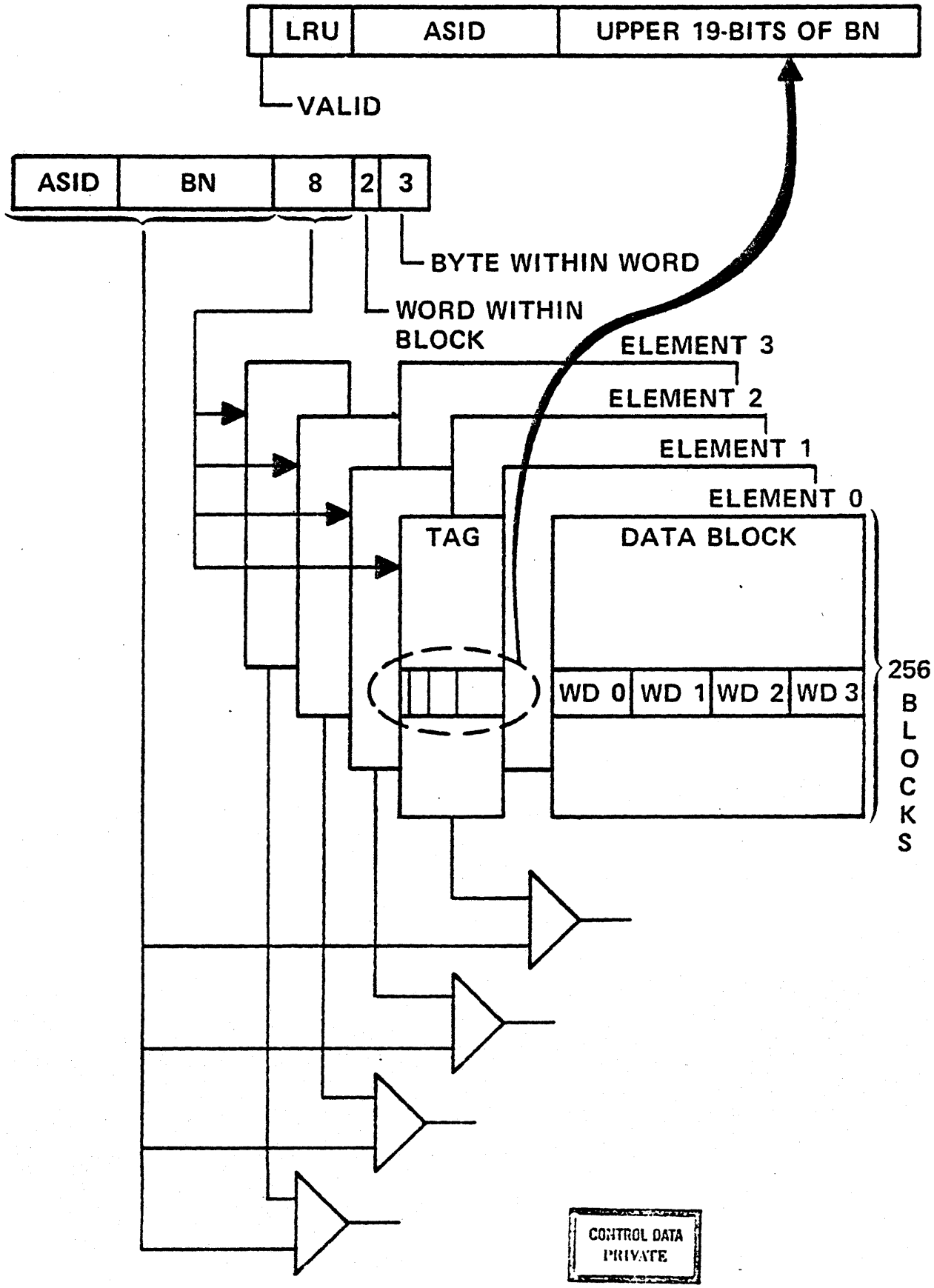
CYBER 180 BUFFER MEMORIES



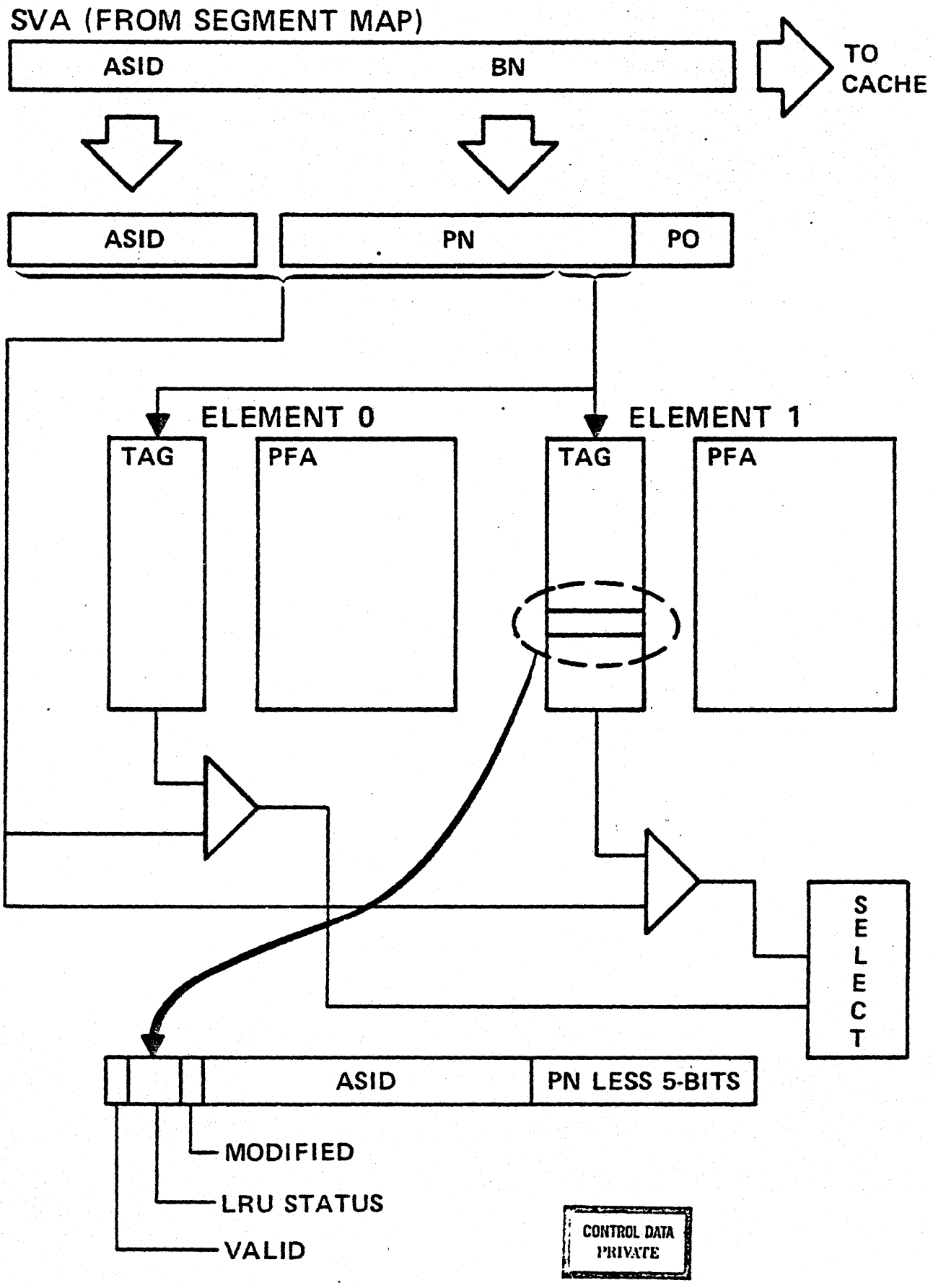
SEGMENT MAP OPERATION



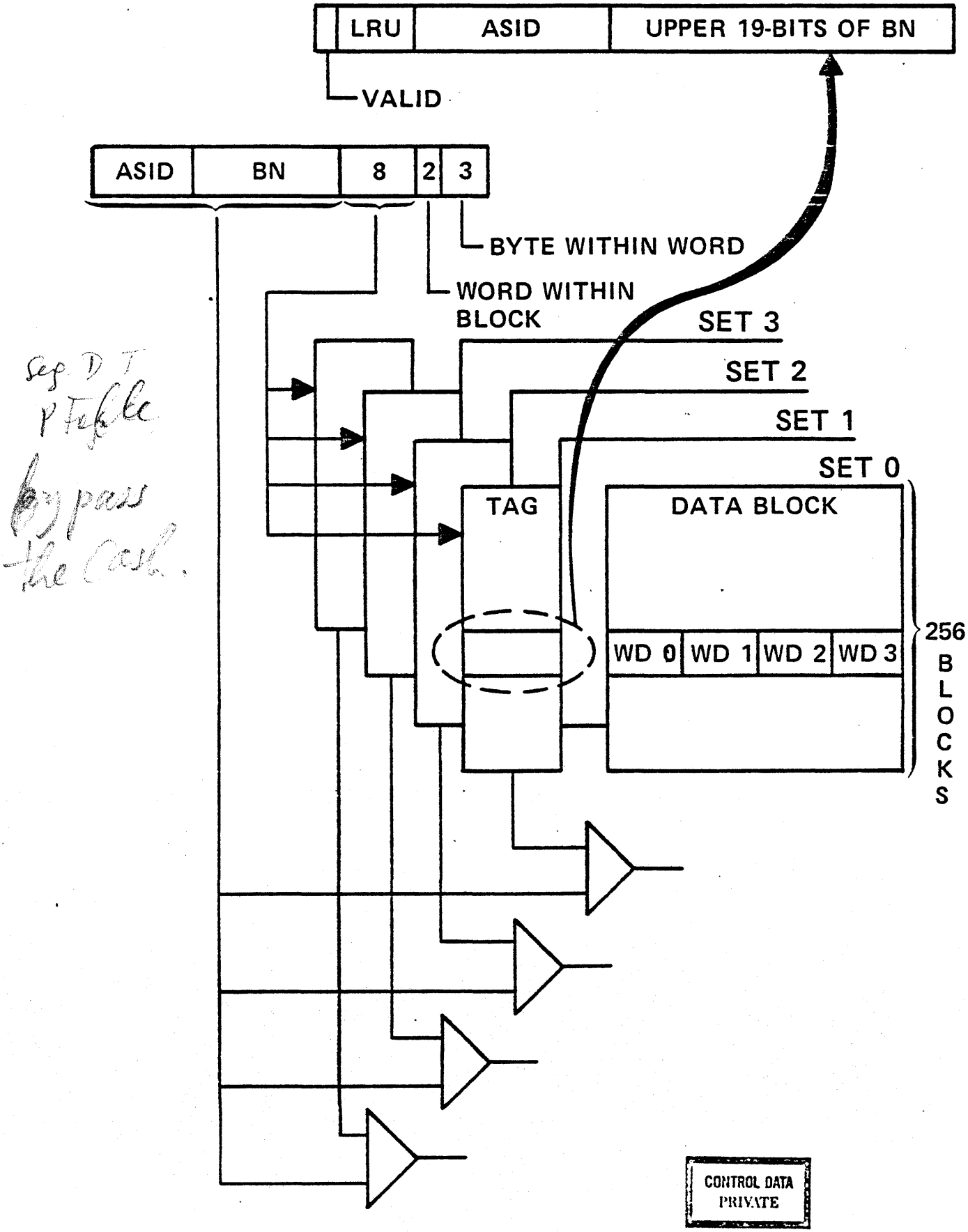
CACHE MEMORY OPERATION



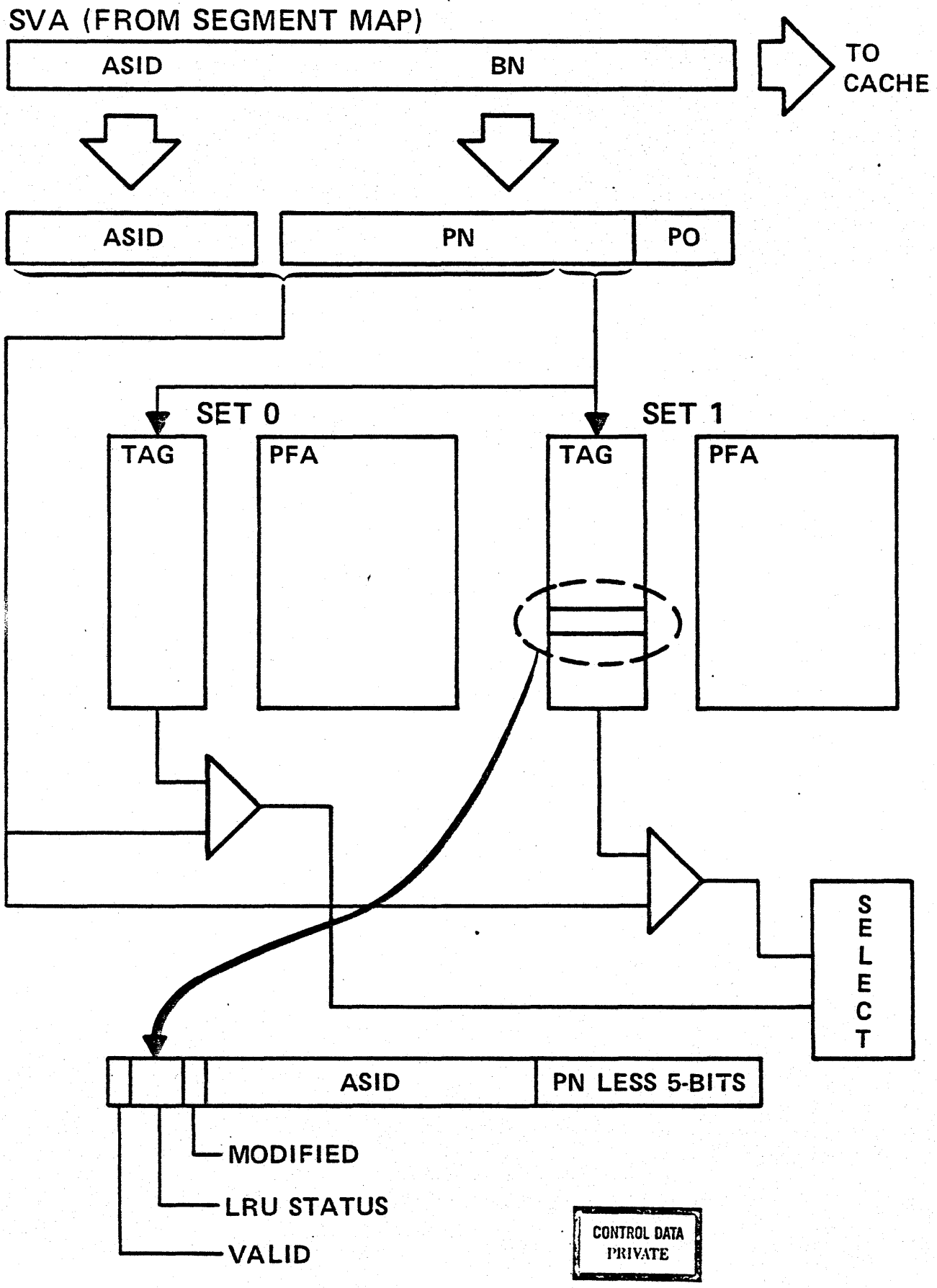
PAGE MAP OPERATION



CACHE MEMORY OPERATION



PAGE MAP OPERATION

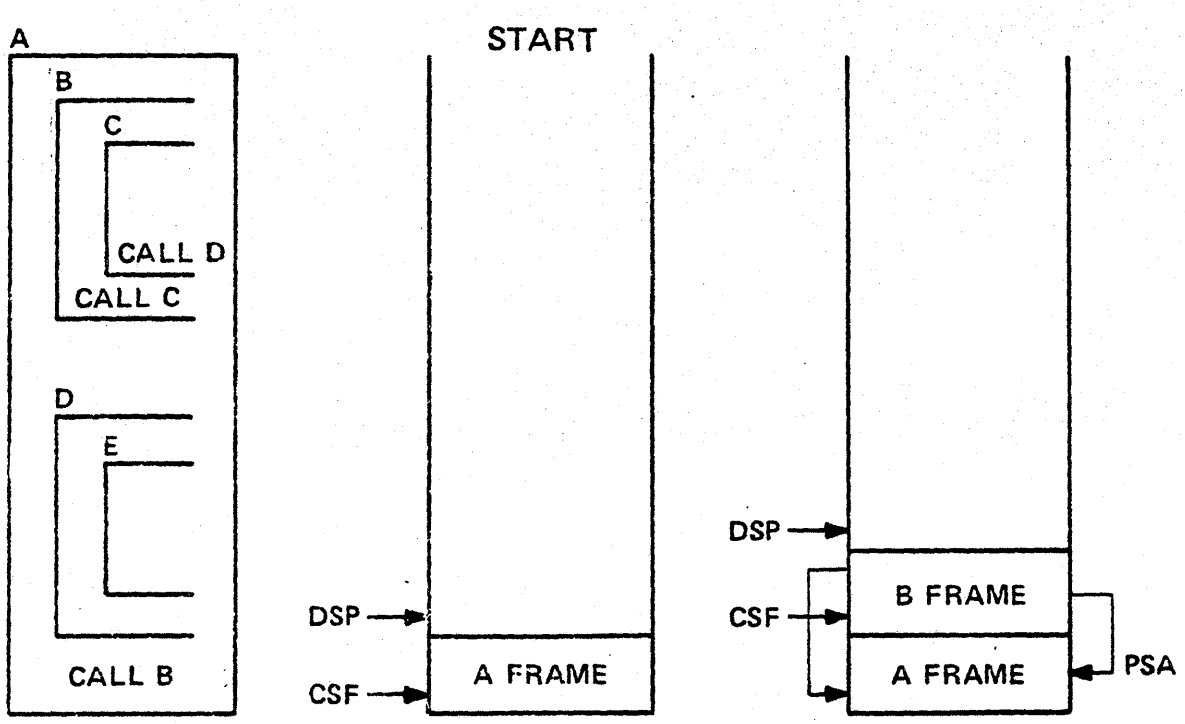


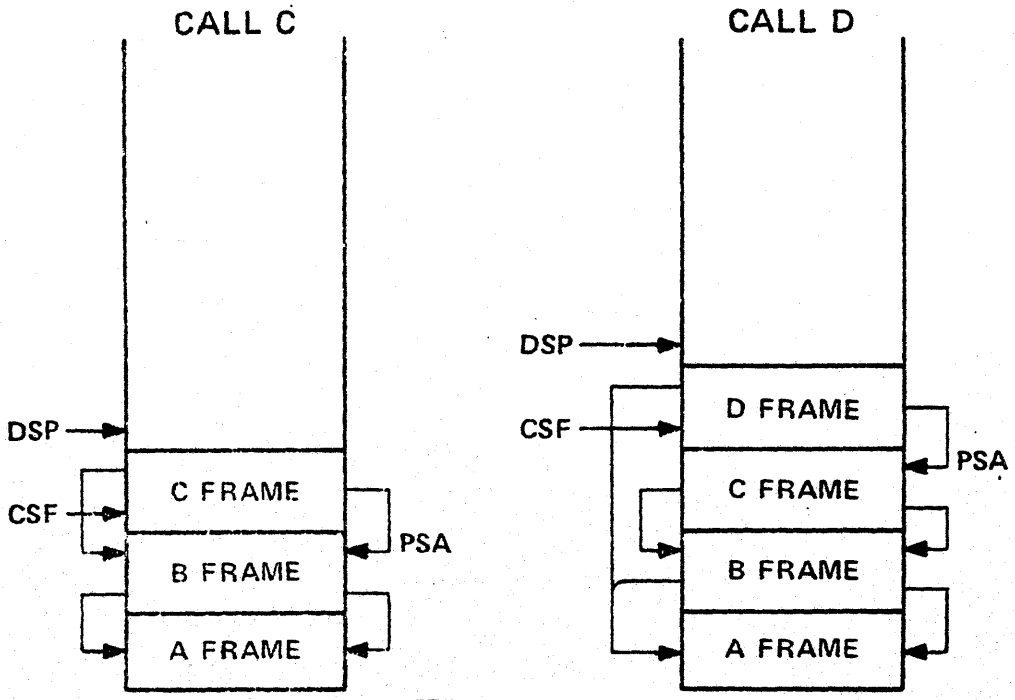
EXAMPLE OF BLOCK STRUCTURE

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

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

STACK FRAME MANIPULATION
BY CALL/RETURN



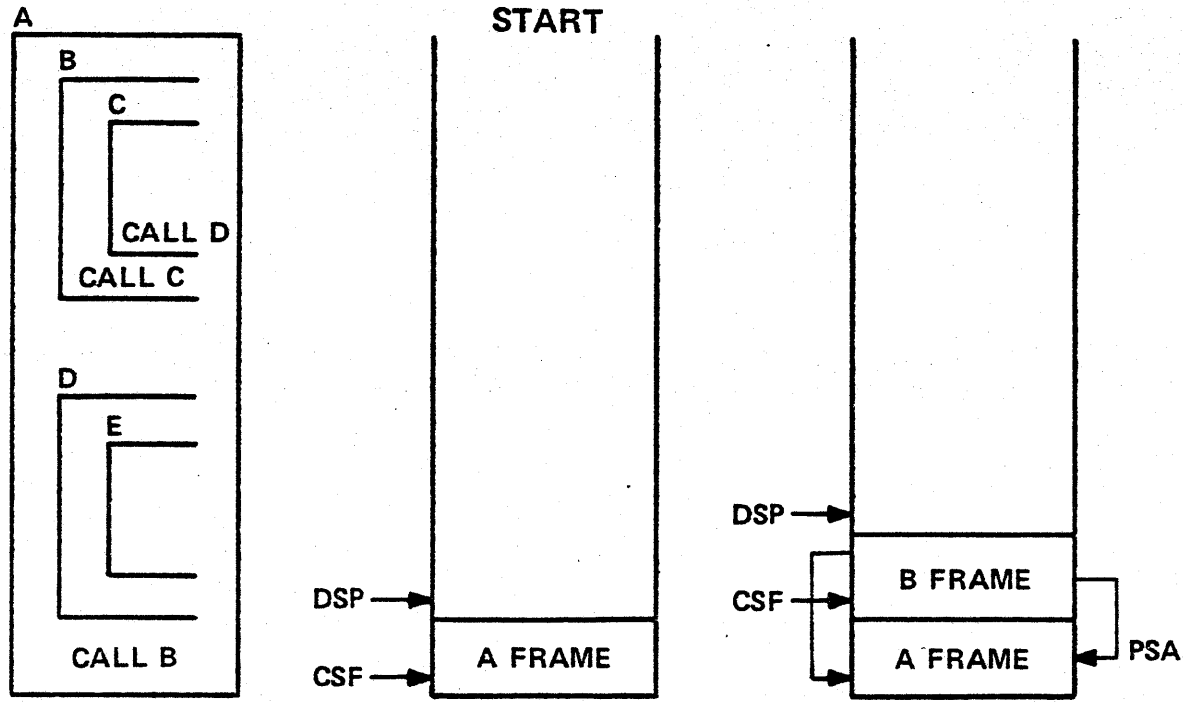


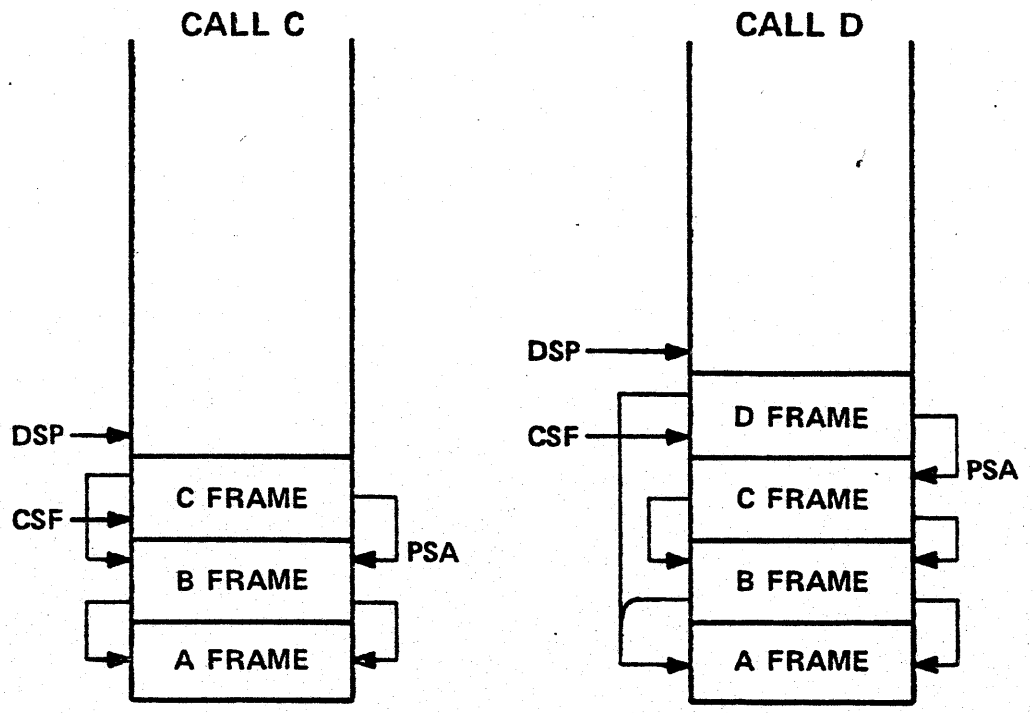
EXAMPLE OF BLOCK STRUCTURE

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

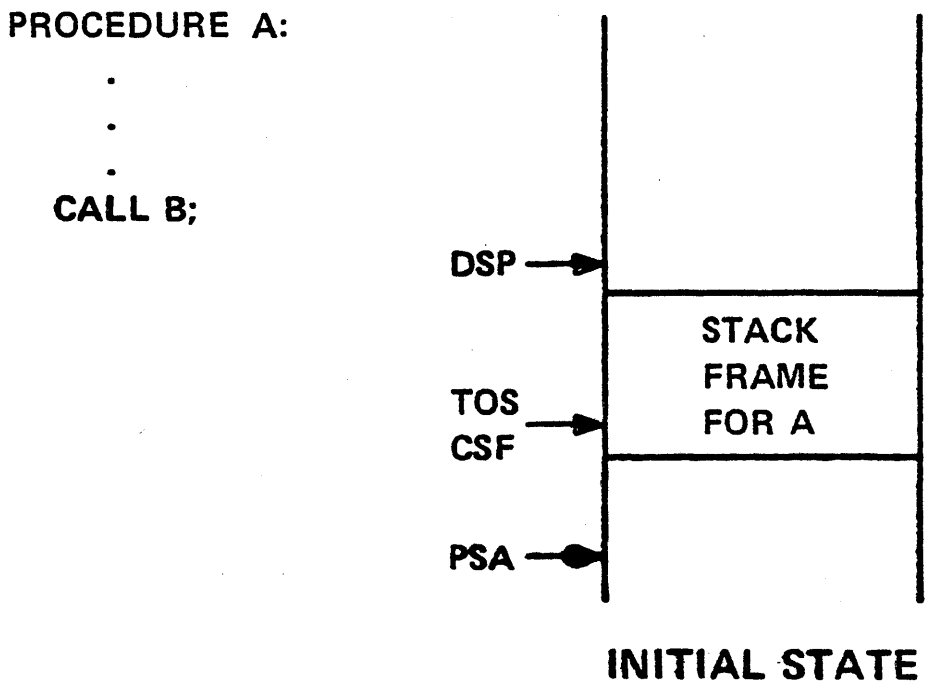
LINE NUMBER	SEVERITY LEVEL	ERROR MESSAGE
24	ERROR	Undeclared identifier - I_C.
37	ERROR	Undeclared identifier - C

STACK FRAME MANIPULATION BY CALL/RETURN

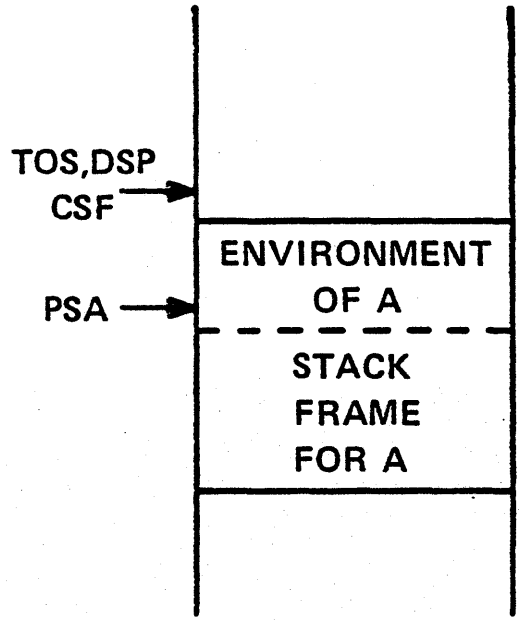




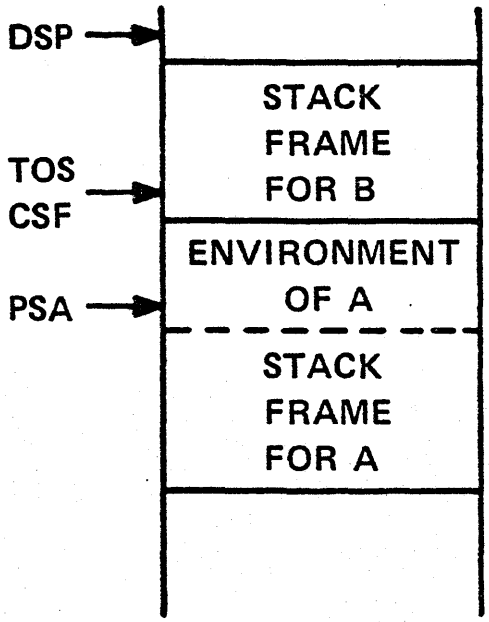
BASIC CALL MECHANISM



INITIAL STATE



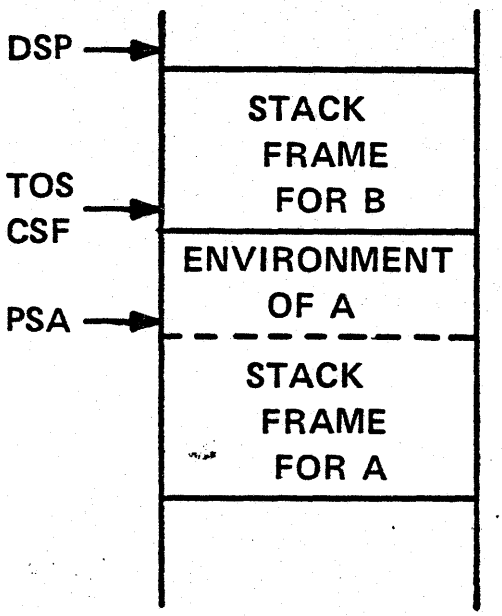
AFTER CALL ISSUED



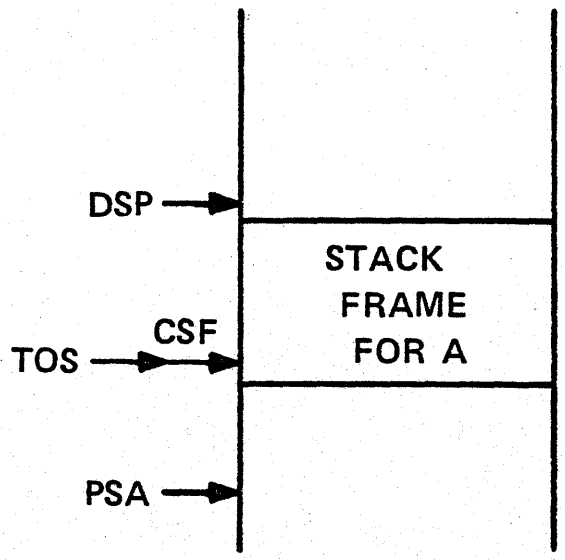
AFTER SOFTWARE CREATION OF STACK FRAME FOR B

BASIC RETURN MECHANISM

```
PROCEDURE B;
      .
      .
      .
RETURN
END
```



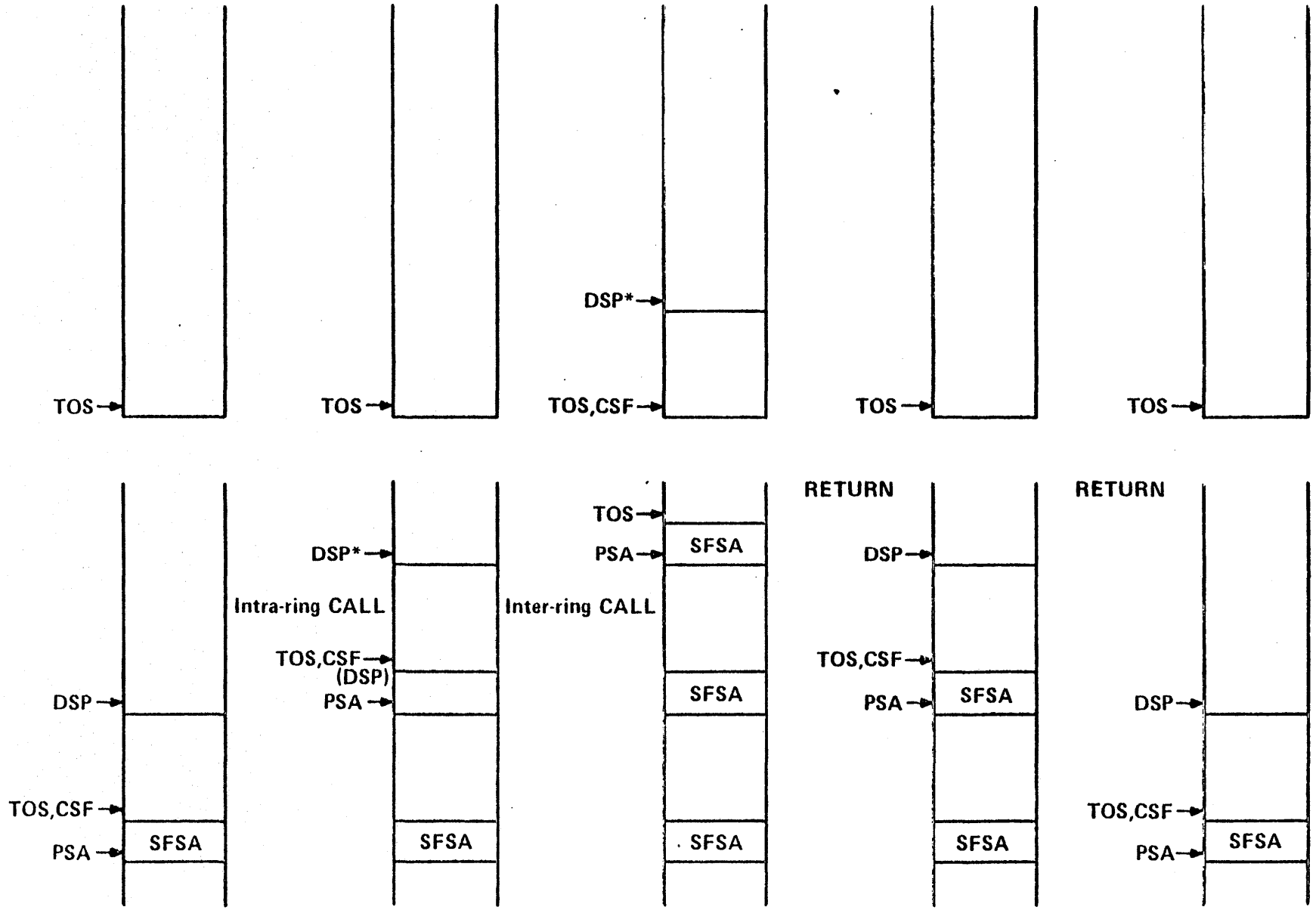
INITIAL STATE



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

TOS is reset from final value in A1 (CSF) by the RETURN mechanism.

CALL/RETURN



* Moved by software

CALL INSTRUCTION FORMATS

CALL INDIRECT

CALL PER (A_J) DISPLACED BY 8 * Q, ARGUMENTS PER (A_K)

CALL RELATIVE

CALL TO (P) DISPLACED BY 8 * Q, BINDING SECTION POINTER PER (A_J), ARGUMENTS PER (A_K)

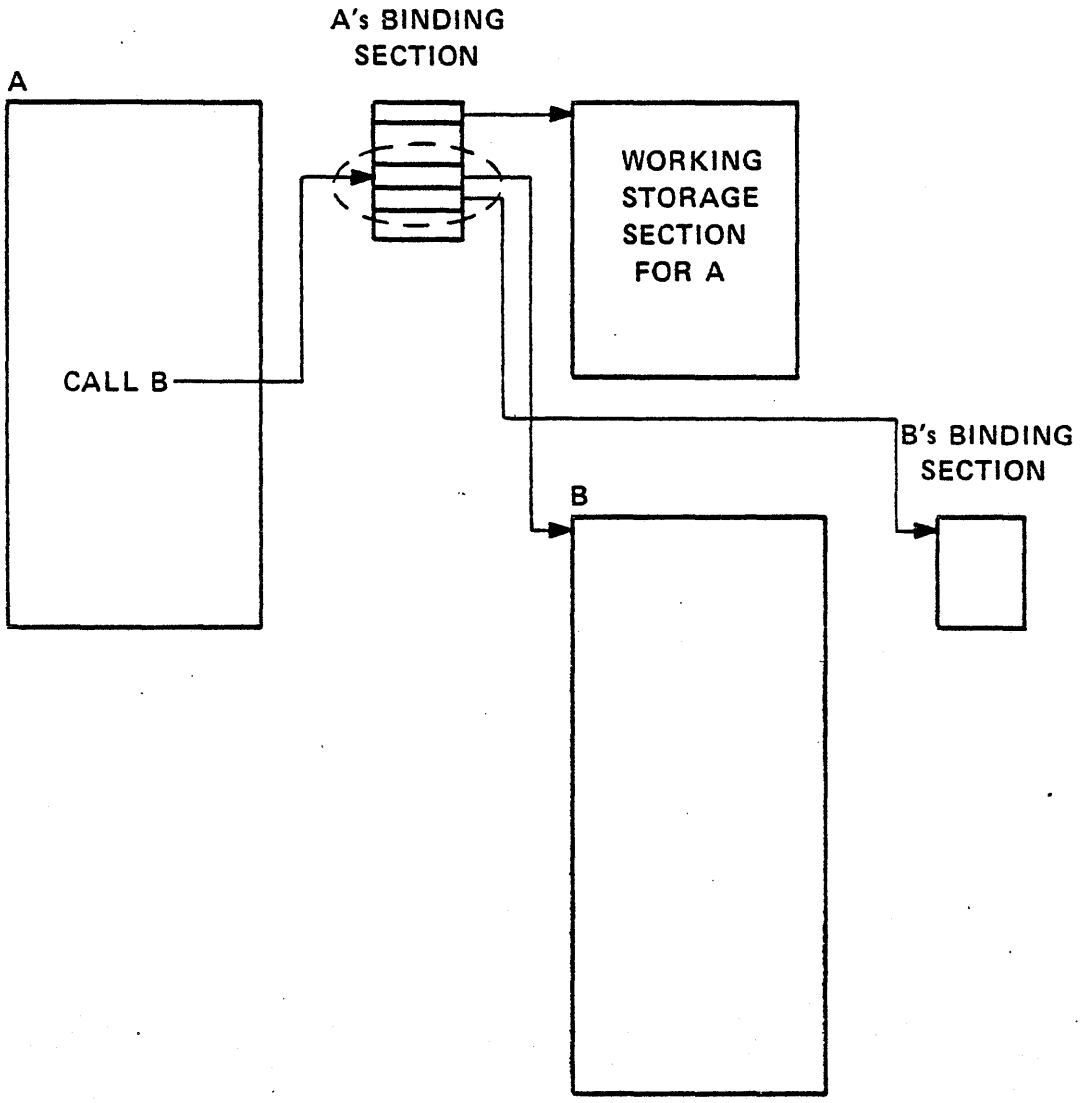
RETURN

RETURN

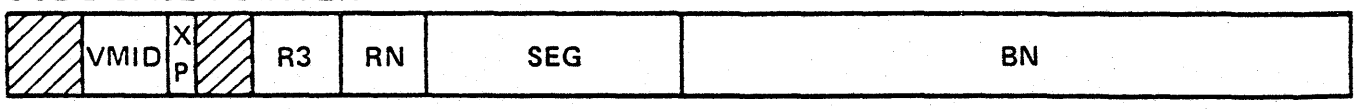
REGISTER USAGE

(A_0)	–	DYNAMIC SPACE POINTER
(A_1)	–	CURRENT STACK FRAME POINTER
(A_2)	–	PREVIOUS SAVE AREA POINTER
(A_3)	–	BINDING SECTION POINTER
(A_4)	–	ARGUMENT POINTER

CALL INDIRECT EXAMPLE

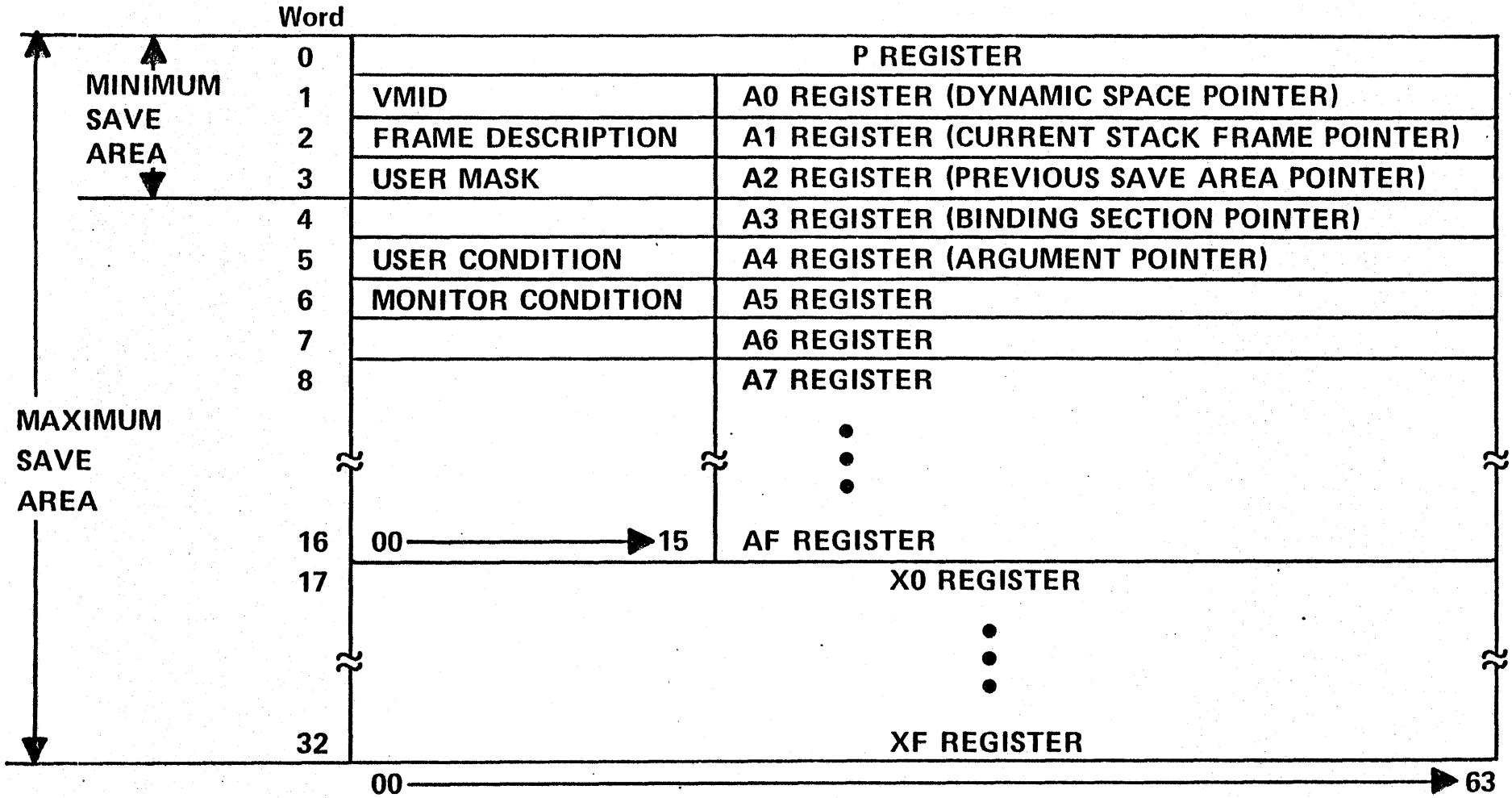


CODE BASE POINTER

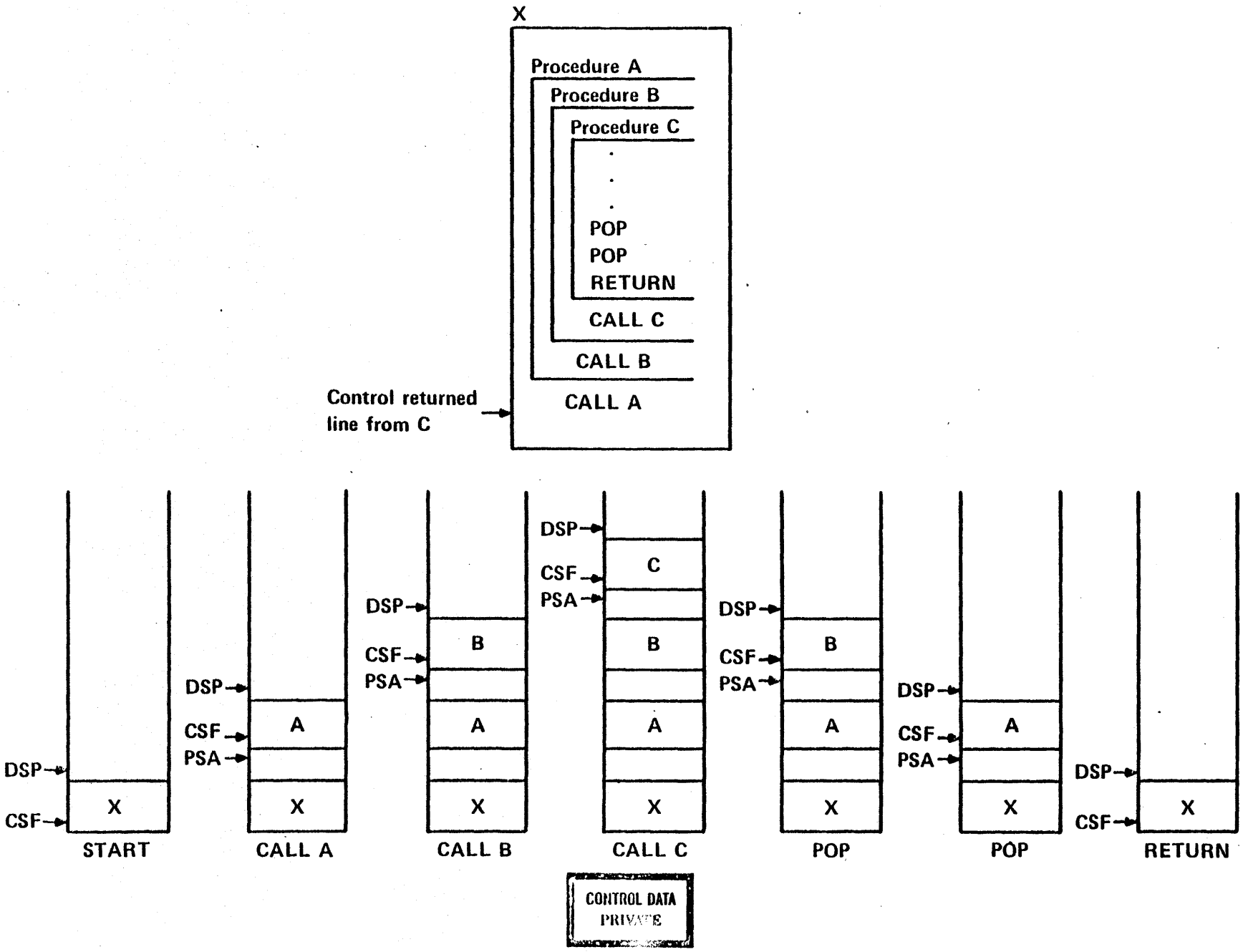


POINTER TO CALLEE'S BINDING SECTION

STACK FRAME SAVE AREA



EXAMPLE OF THE POP INSTRUCTION



INTERRUPTS & REGISTERS

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

critical frame flag: i.e. mid e.g don't leave procedure before some things have been done (what is up to me) if you return from prog. when this flag is on → set CFF in UCR.

EXCHANGE PACKAGE

Word No.			
0	P		
1	VMID*	UVMID**	A0
2	Flags	Traps Enables	A1
3	User Mask		A2
4	Monitor Mask		A3
5	User Condition		A4
6	Monitor Condition		A5
7	Kypt Class	LPID***	A6
8	Keypoint Mask		A7
9	Keypoint Code		A8
10			A9
11	Process Int. Timer		AA
12			AB
13	Base Constant		AC
14			AD
15	Model Dependent Flags		AE
16	Segment Table Length		AF
17	X0		
≈			
32	XF		
33	Model Dependent Word		
34	Segment Table Address		Untranslatable Pointer
35			Trap Pointer
36	Debug Index	Debug Mask	Debug List Pointer
37	Largest Ring Number		Top of Stack Ring No. 1
≈			
51			Top of Stack Ring No. 15
	00	07 08 15 16	63

PMF in Theta but not S2, S3

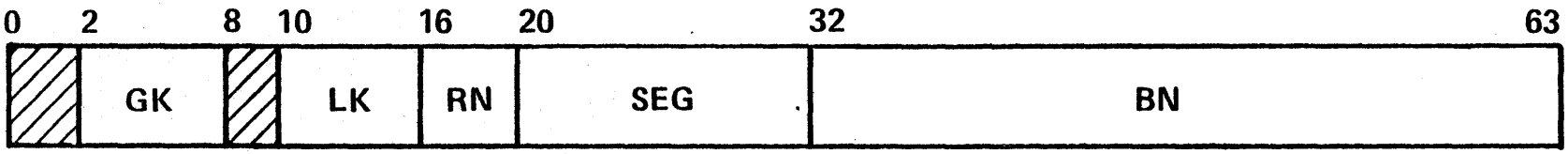
* Virtual Machine Identifier

** Untranslatable Virtual Machine Identifier

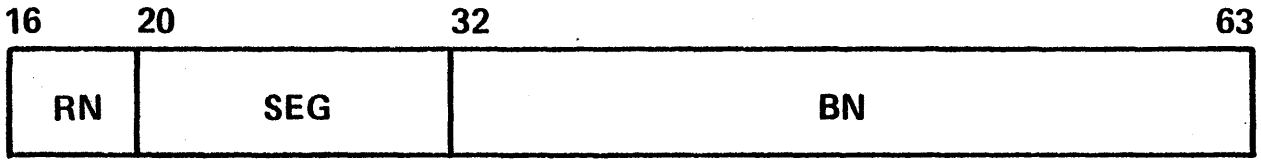
*** Last Processor Identification

CONTROL DATA PRIVATE

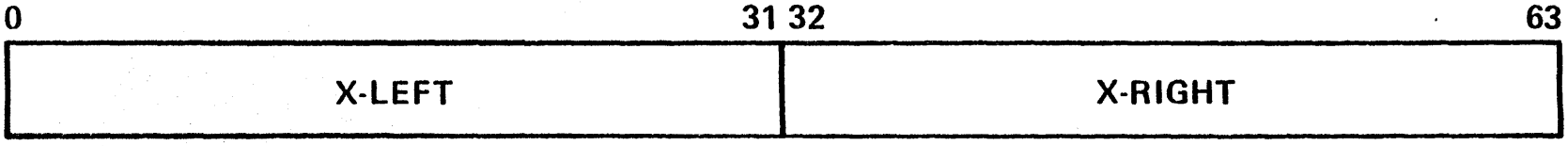
P-REGISTER FORMAT



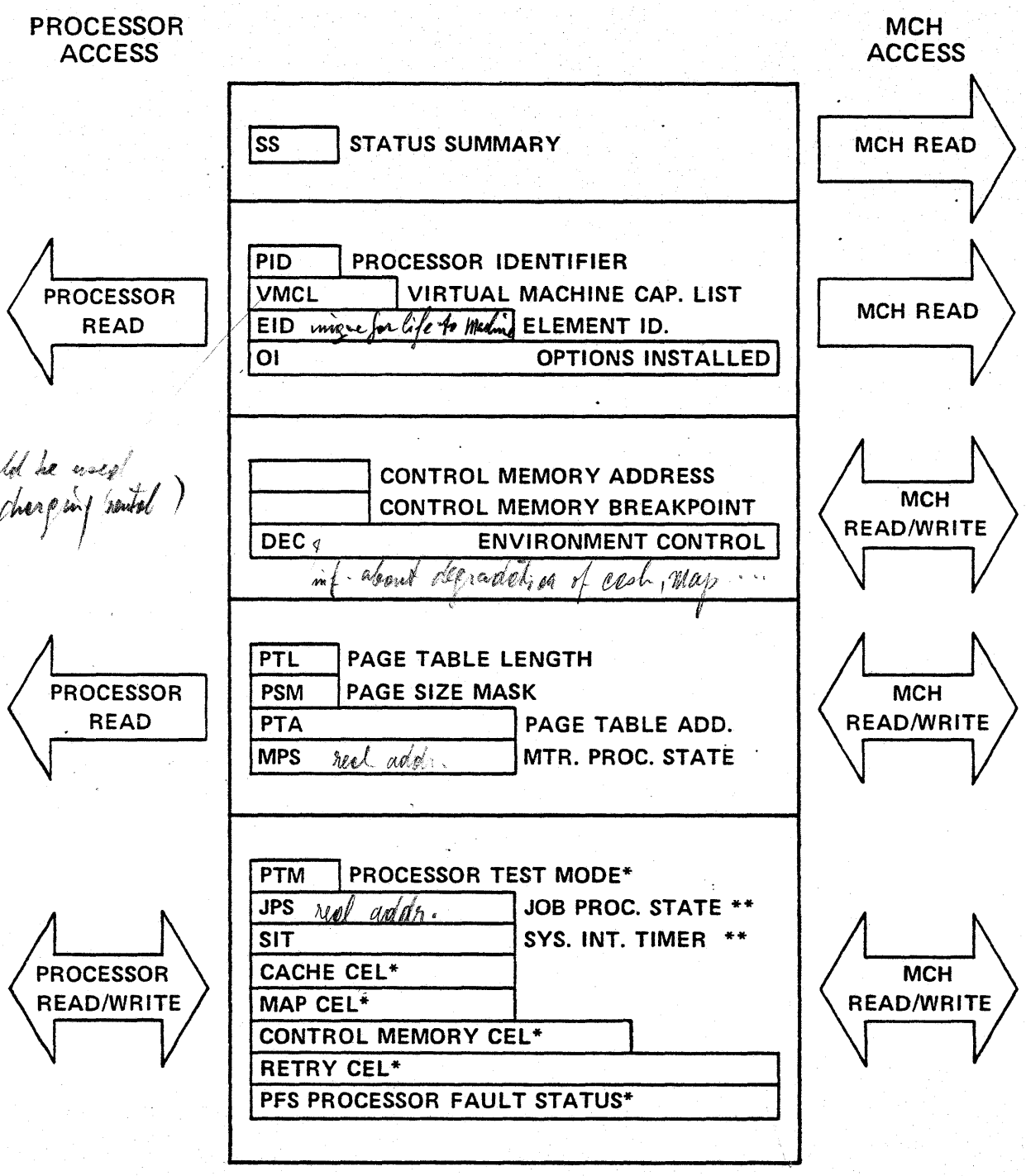
A-REGISTER FORMAT – PVA



X-REGISTER FORMAT



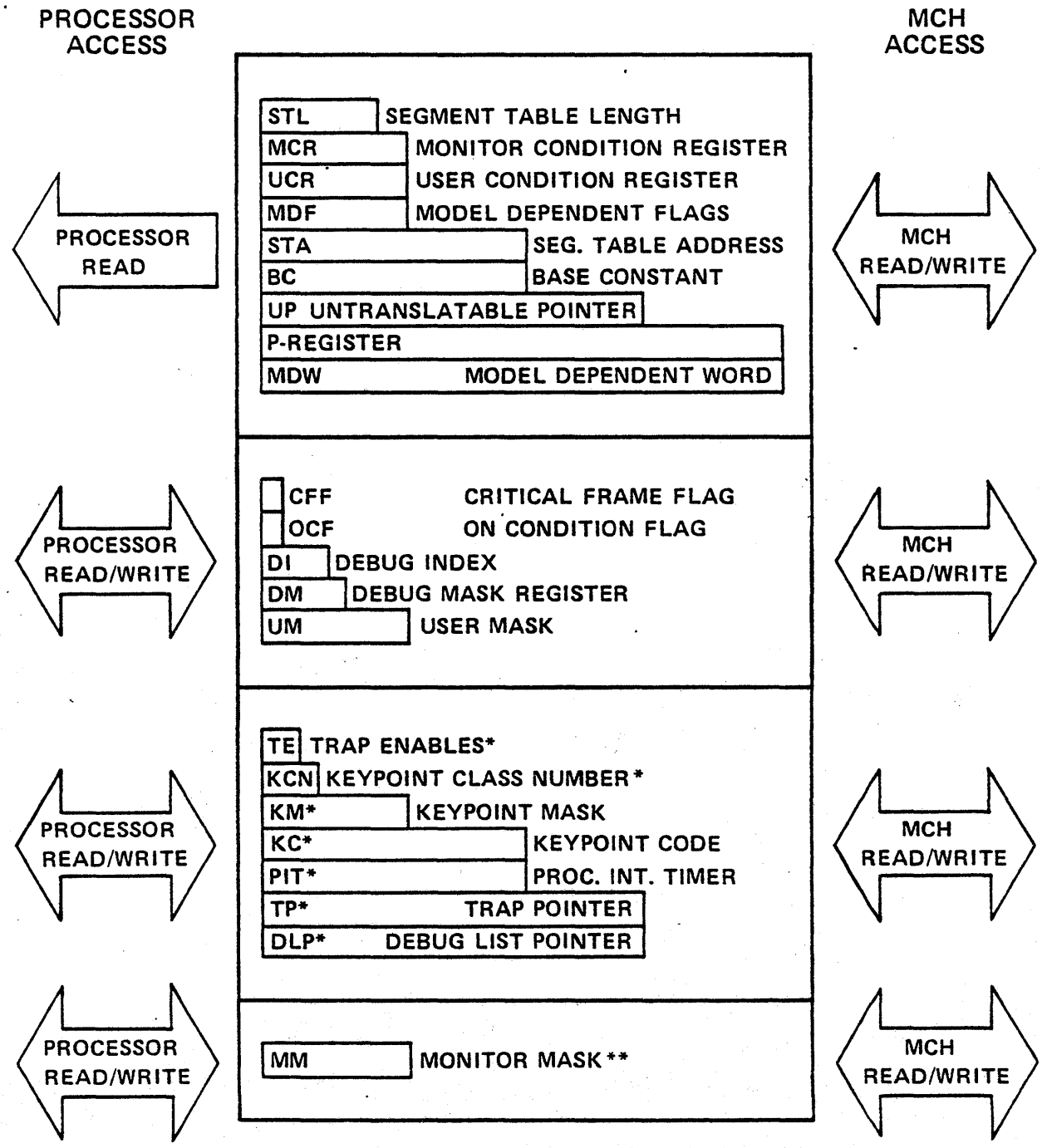
PROCESSOR STATE REGISTERS



* WRITE IN GLOBAL PRIVILEGE MODE ONLY

** WRITE IN MONITOR MODE ONLY

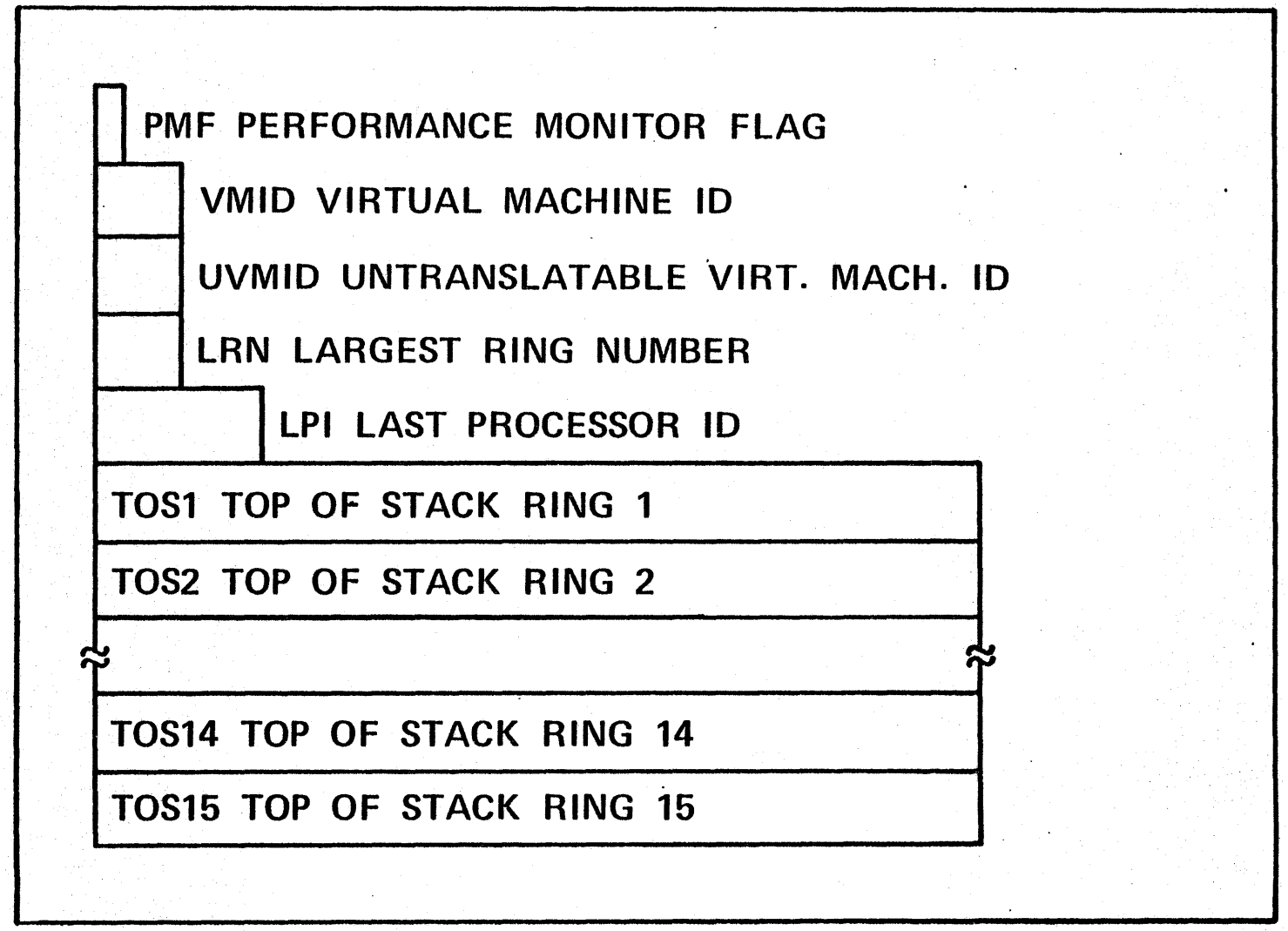
PROCESS STATE REGISTERS



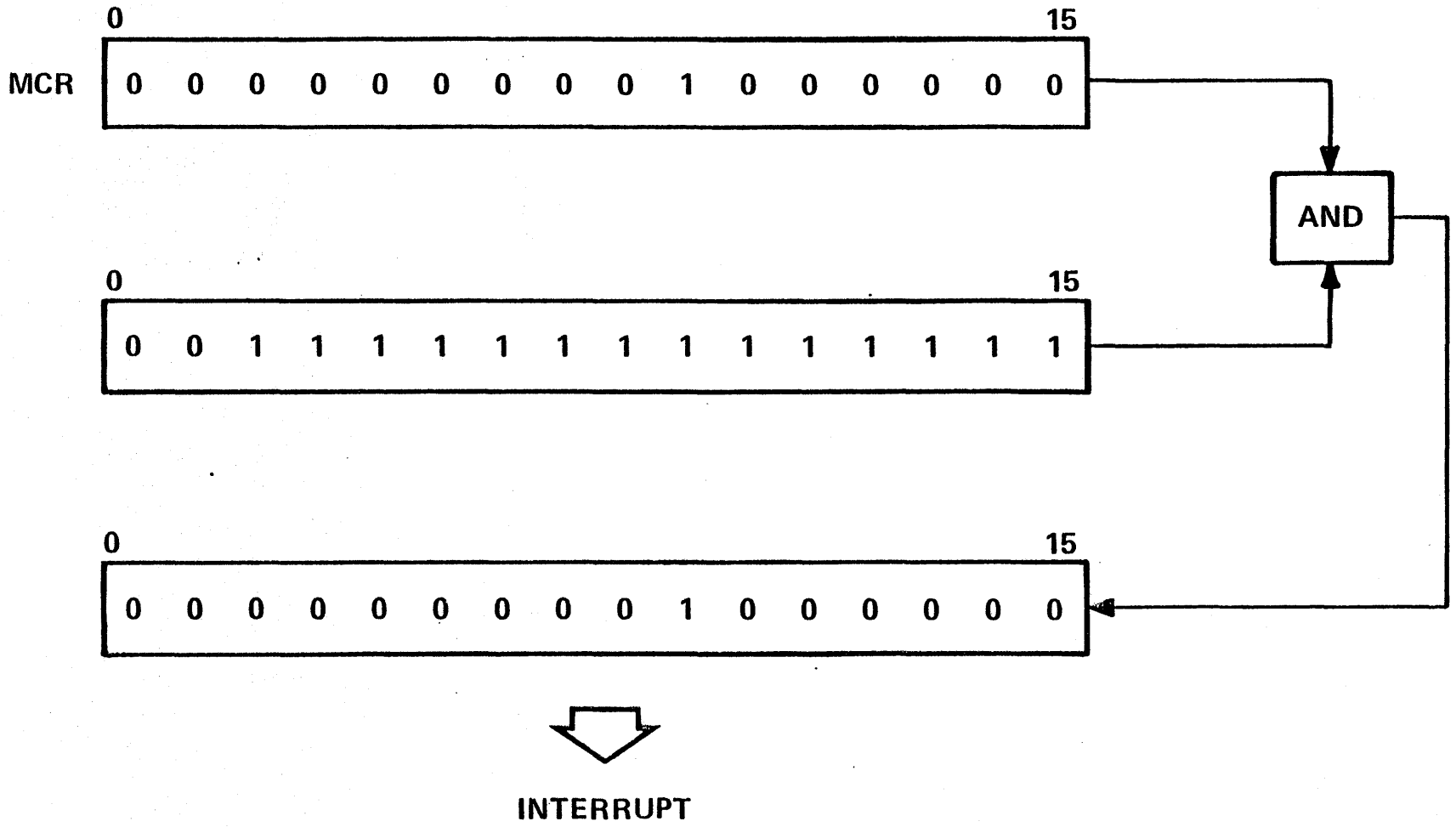
* WRITE IN LOCAL PRIVILEGED MODE ONLY

** WRITE IN MONITOR MODE ONLY

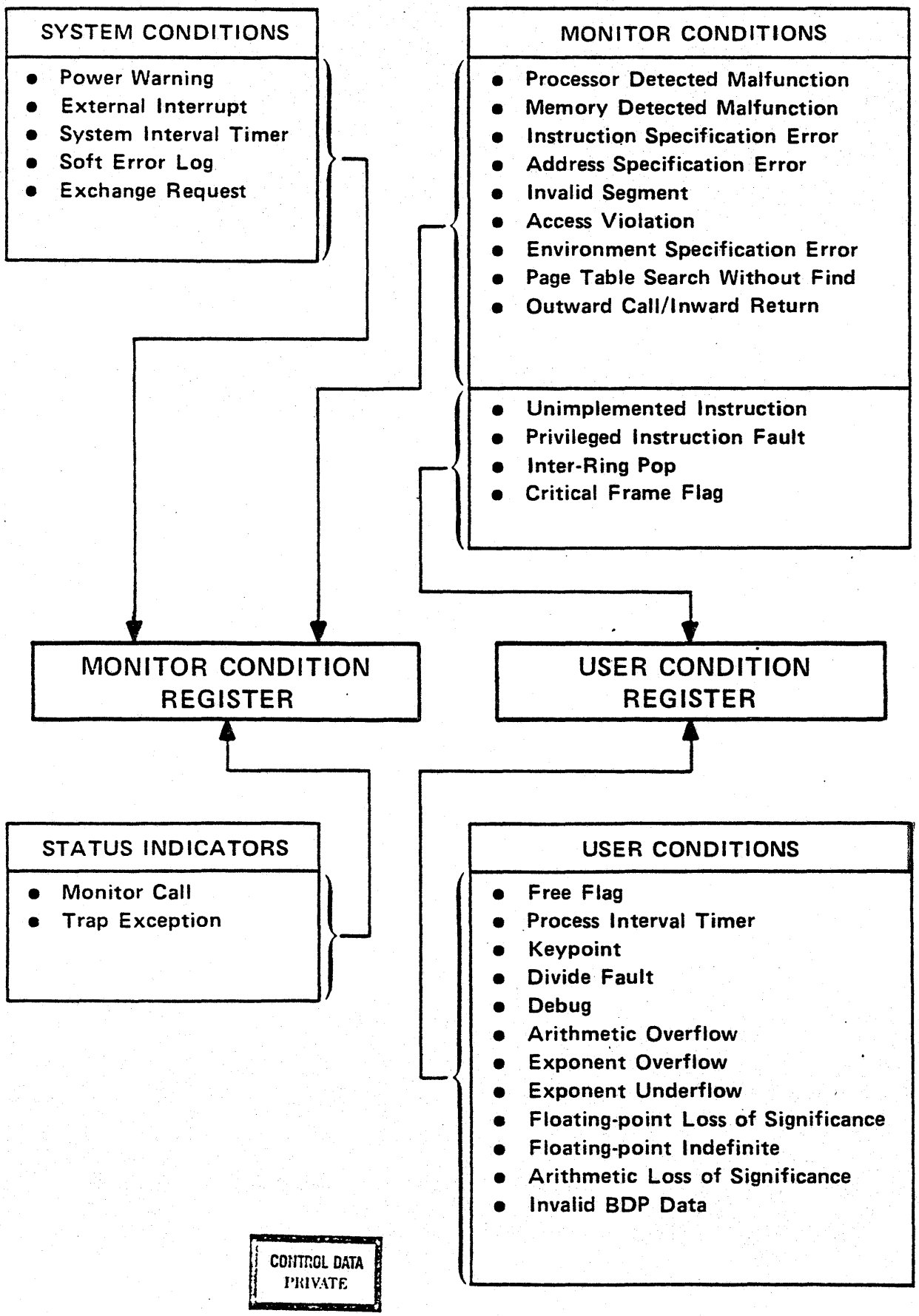
PROCESS STATE REGISTERS SET BY EXCHANGE JUMP



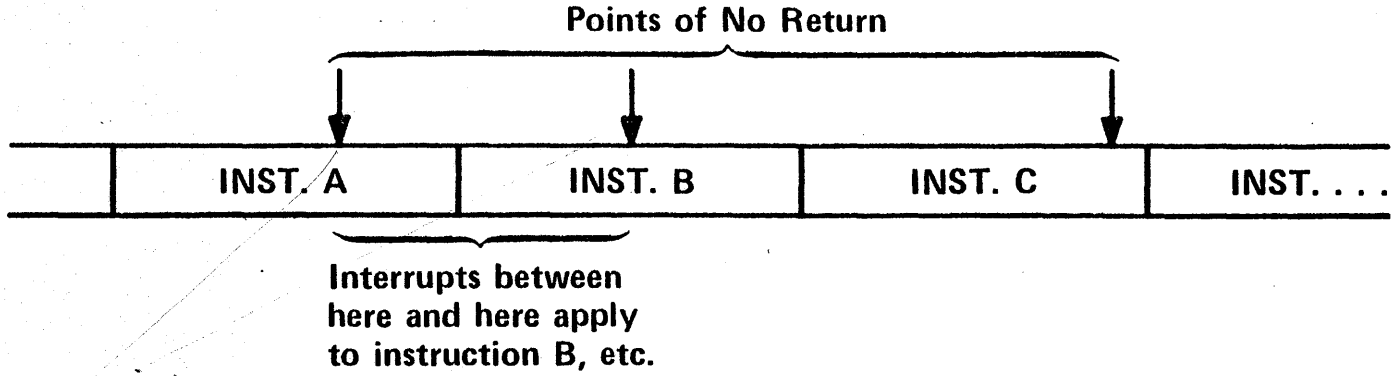
BASIC INTERRUPT MECHANISM



INTERRUPT CONDITIONS

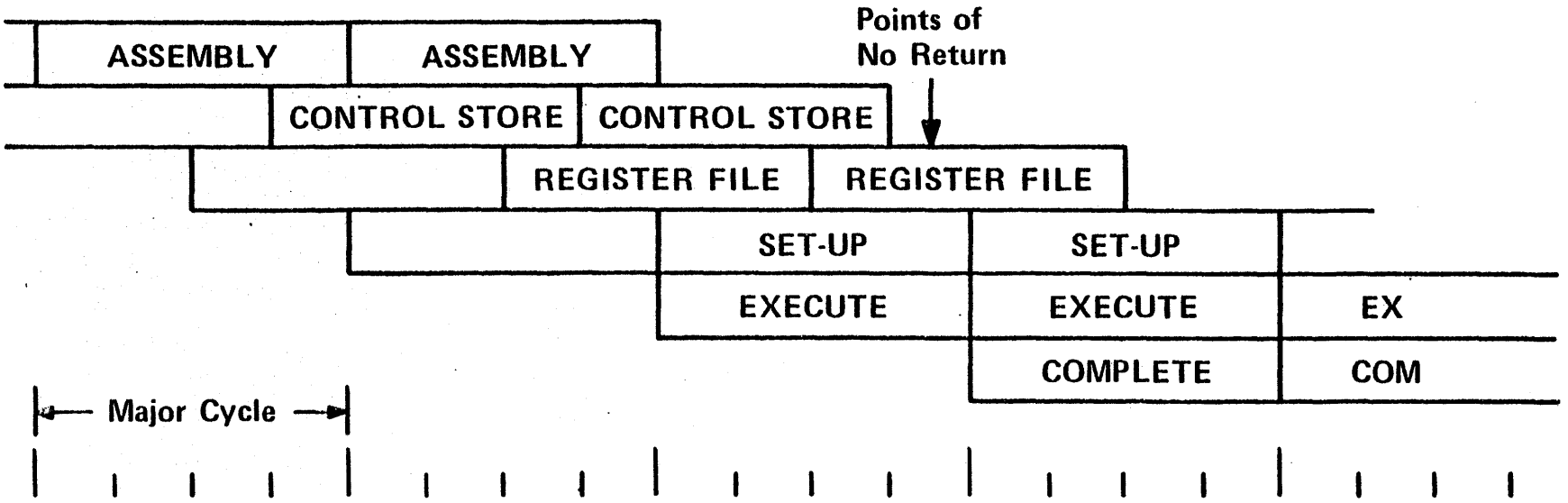


P3 PIPELINED INSTRUCTION STREAM



AN EXAMPLE OF AN INSTRUCTION STREAM

After this points committed to finish execution of instr. whatever happens, i.e. no cond. check.
In Theta there are several points of cond. check.



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.					

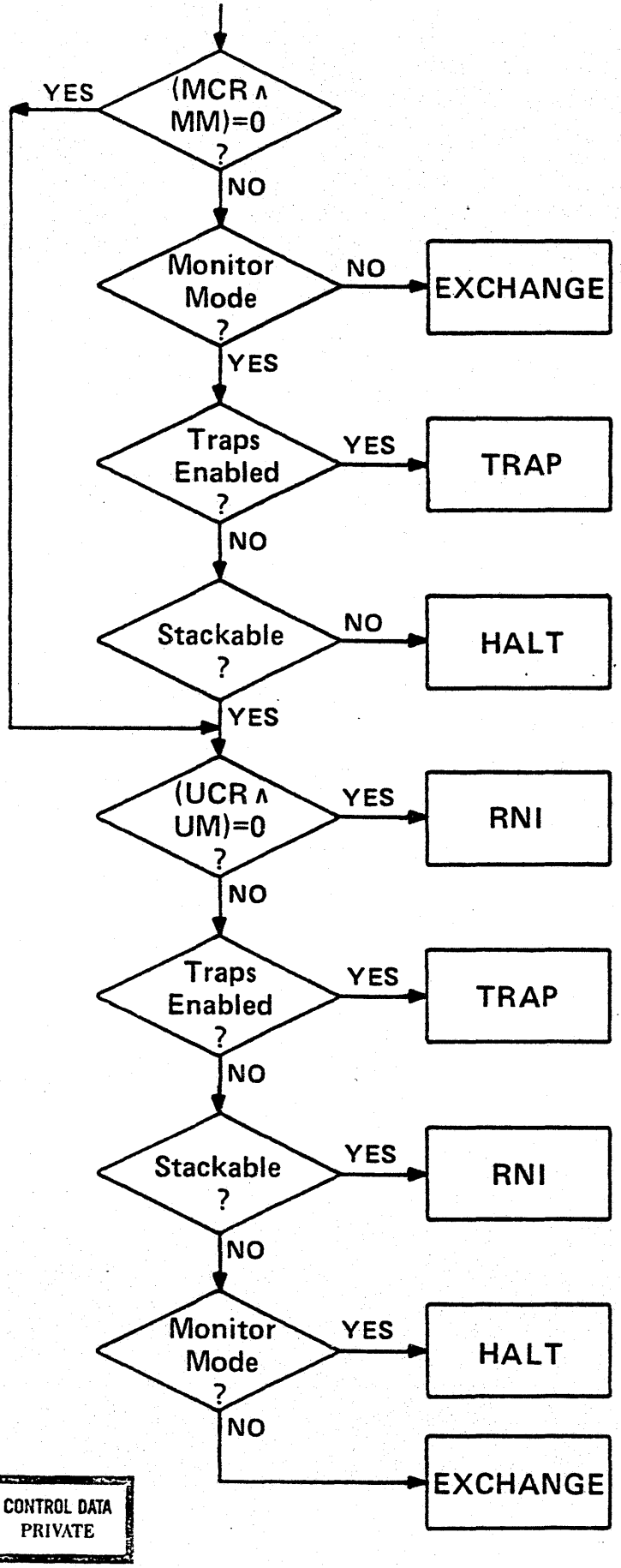
USER CONDITION REGISTER

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

problem forces imed. halt.

you work a little until finished

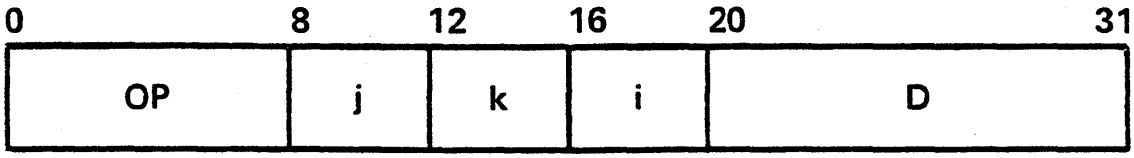
INTERRUPT FLOWCHART



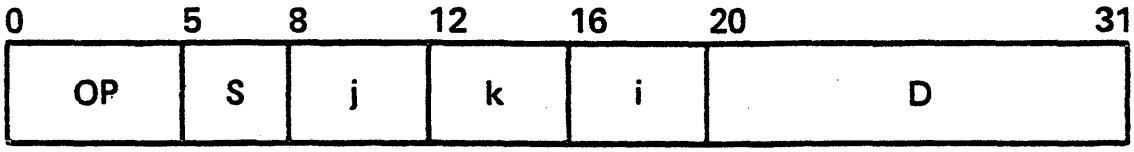
INSTRUCTIONS

OP CODE FORMATS

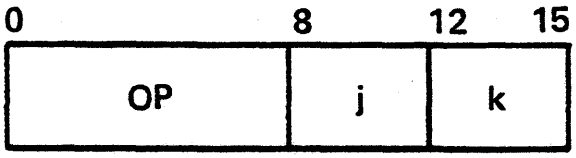
jkiD INSTRUCTION FORMAT



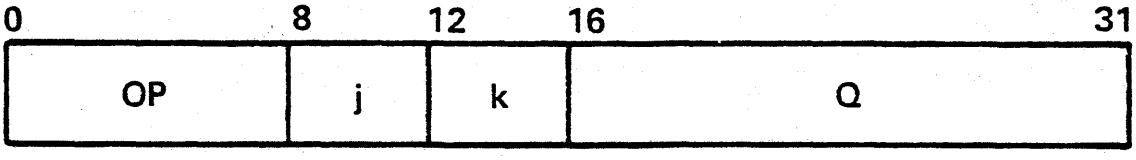
SjkiD INSTRUCTION FORMAT



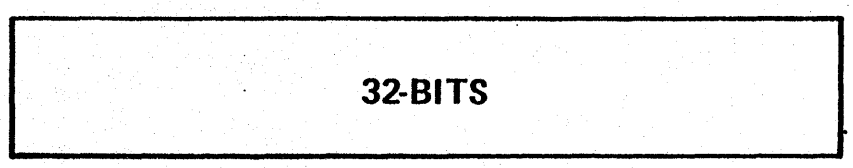
jk INSTRUCTION FORMAT



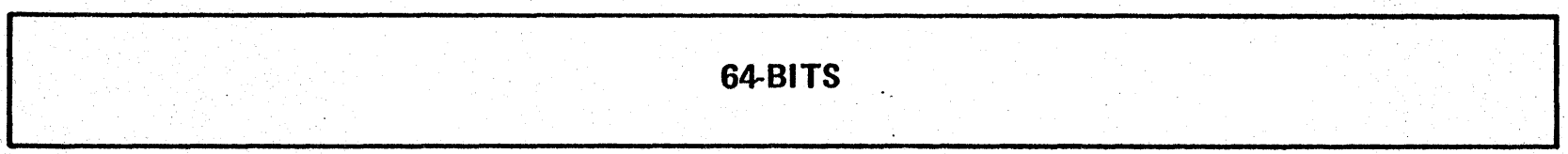
jkQ INSTRUCTION FORMAT



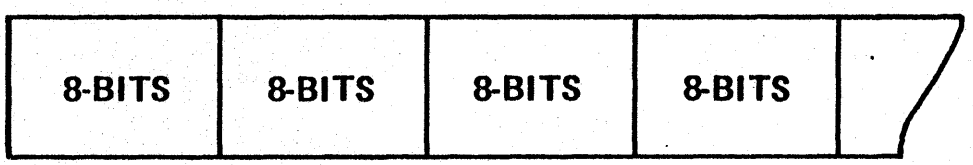
GENERAL DATA FORMATS



32-BIT, SIGNED, 2'S COMPLEMENT INTEGERS



64-BIT, SIGNED, 2'S COMPLEMENT INTEGERS



8-BIT ASCII CHARACTERS (BYTES)

GENERAL INSTRUCTIONS

REF NO.	OP CODE	FORMAT	MNEMONIC	NAME
005	A2	jkiD	LXI	LOAD WORD
006	82	jkQ	LX	LOAD WORD
007	A3	jkiD	SXI	STORE WORD
008	83	jkQ	SX	STORE WORD
001	DX	SjkiD	LBYTS,S	LOAD BYTES
003	DX	SjkiD	SBYTS,S	STORE BYTES
009	A4	jkiD	LBYT,X0	LOAD BYTES
011	A5	jkiD	SBYT,X0	STORE BYTES
013	86	jkQ	LBYTD,j	LOAD BYTES RELATIVE
014	88	jkQ	LBIT	LOAD BIT
015	89	jkQ	SBIT	STORE BIT
016	A0	jkiD	LAI	LOAD ADDRESS
017	84	jkQ	LA	LOAD ADDRESS
018	A1	jkiD	SAI	STORE ADDRESS
019	85	jkQ	SA	STORE ADDRESS
020	80	jkQ	LMULT	LOAD MULTIPLE
021	81	jkQ	SMULT	STORE MULTIPLE
022	24	jk	ADDX	INTEGER SUM
023	25	jk	SUBX	INTEGER DIFFERENCE
024	26	jk	MULX	INTEGER PRODUCT
025	27	jk	DIVX	INTEGER QUOTIENT
035	2D	jk	CMPX	INTEGER COMPARE
143	8B	jkQ	ADDXQ	INTEGER SUM SIGNED IMMEDIATE
166	10	jk	INCX	INTEGER SUM IMMEDIATE
167	11	jk	DECX	INTEGER DIFFERENCE IMMEDIATE
168	B2	jkQ	MULXQ	INTEGER PRODUCT SIGNED IMMEDIATE
037	94	jkQ	BRXEQ	BRANCH ON EQUAL
038	95	jkQ	BRXNE	BRANCH ON NOT EQUAL
039	96	jkQ	BRXGT	BRANCH ON GREATER THAN
040	97	jkQ	BRXGE	BRANCH ON GREATER THAN OR EQUAL
045	9C	jkQ	BRINC	BRANCH AND INCREMENT
027	20	jk	ADDR	HALF WORD INTEGER SUM
030	21	jk	SUBR	HALF WORD INTEGER DIFFERENCE
032	22	jk	MULR	HALF WORD INTEGER PRODUCT
034	23	jk	DIVR	HALF WORD INTEGER QUOTIENT
036	2C	jk	CMPR	HALF WORD INTEGER COMPARE

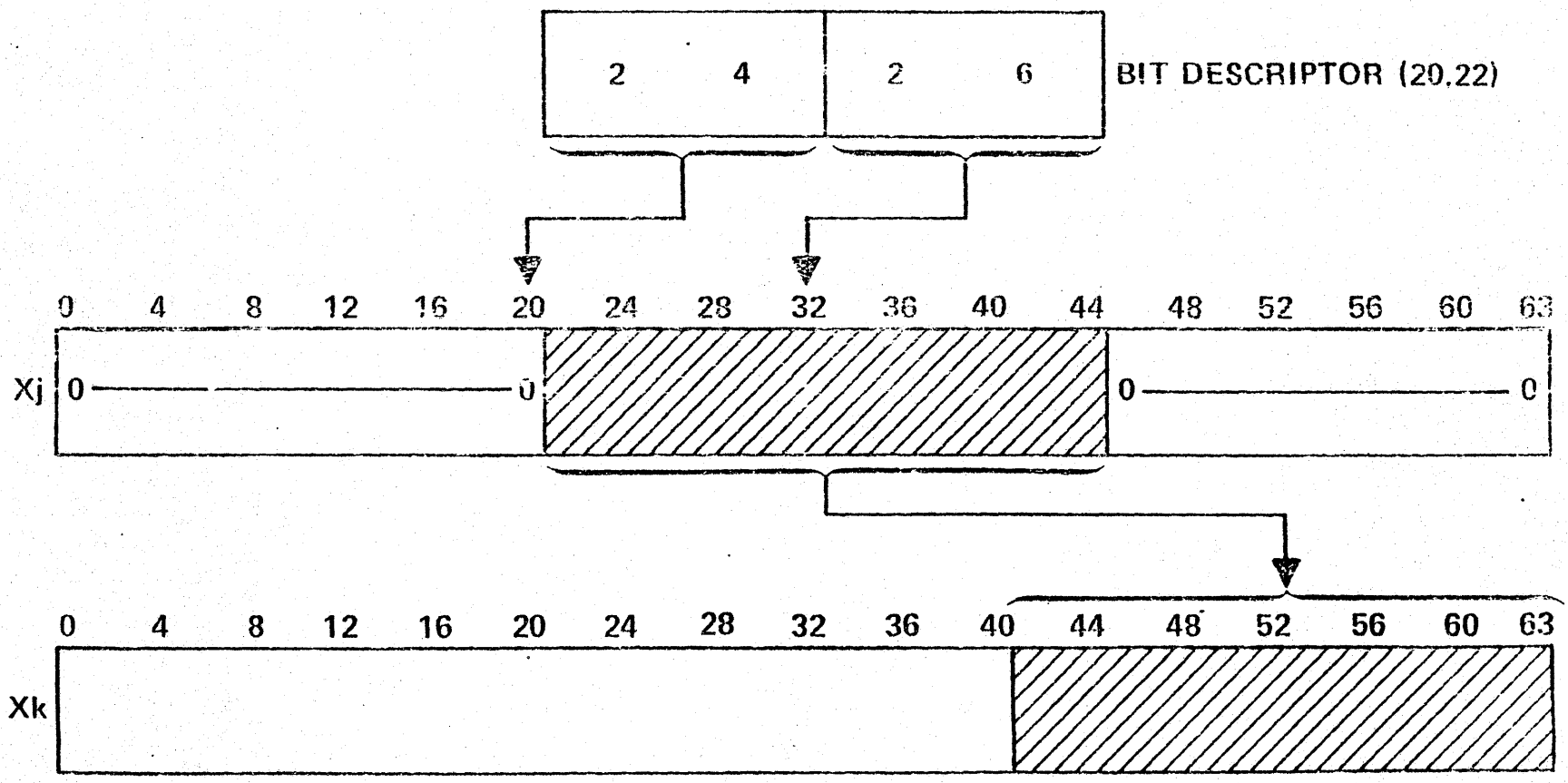
GENERAL INSTRUCTIONS

REF NO.	OP CODE	FORMAT	MNEMONIC	NAME
028	8A	jkQ	ADDRQ	HALF WORD INTEGER SUM SIGNED IMMEDIATE
029	28	jk	INCR	HALF WORD INTEGER SUM IMMEDIATE
031	29	jk	DECR	HALF WORD INTEGER DIFFERENCE IMMEDIATE
033	8C	jkQ	MULRQ	HALF WORD INTEGER PRODUCT SIGNED IMMEDIATE
041	90	jkQ	BRREQ	BRANCH ON HALF WORD EQUAL
042	91	jkQ	BRRNE	BRANCH ON HALF WORD NOT EQUAL
043	92	jkQ	BRRGT	BRANCH ON HALF WORD GREATER THAN
044	93	jkQ	BRRGE	BRANCH ON HALF WORD GREATER THAN OR EQUAL
049	0D	jk	CPYXX	COPY FULL WORD
050	0B	jk	CPYAX	COPY TO X FROM A
051	09	jk	CPYAA	COPY ADDRESS
052	0A	jk	CPYXA	COPY TO A FROM X
053	0C	jk	CPYRR	COPY HALF WORD
054	8E	jkQ	ADDAQ	ADDRESS INCREMENT SIGNED IMMEDIATE
056	2A	jk	ADDAX	ADDRESS INCREMENT
055	8F	jkQ	ADDPXQ	ADDRESS RELATIVE
161	A7	jkiD	ADDAD	ADDRESS INCREMENT
057	3D	jk	ENTP	ENTER IMMEDIATE POSITIVE
058	3E	jk	ENTN	ENTER IMMEDIATE NEGATIVE
059	8D	jkQ	ENTE	ENTER SIGNED IMMEDIATE
060	3F	jk	ENTL	ENTER X0 IMMEDIATE LOGICAL
061	1F	jk	ENTZ	ENTER ZEROS
			ENTO	ENTER ONES
			ENTS	ENTER SIGNS
164	39	jk	ENTX	ENTER X1 IMMEDIATE LOGICAL
165	87	jkQ	ENTC	ENTER X1 SIGNED IMMEDIATE
169	B3	jkQ	ENTA	ENTER X0 SIGNED IMMEDIATE
062	A8	jkiD	SHFC	SHIFT CIRCULAR
063	A9	jkiD	SHFX	SHIFT FULL WORD
064	AA	jkiD	SHFR	SHIFT HALF WORD

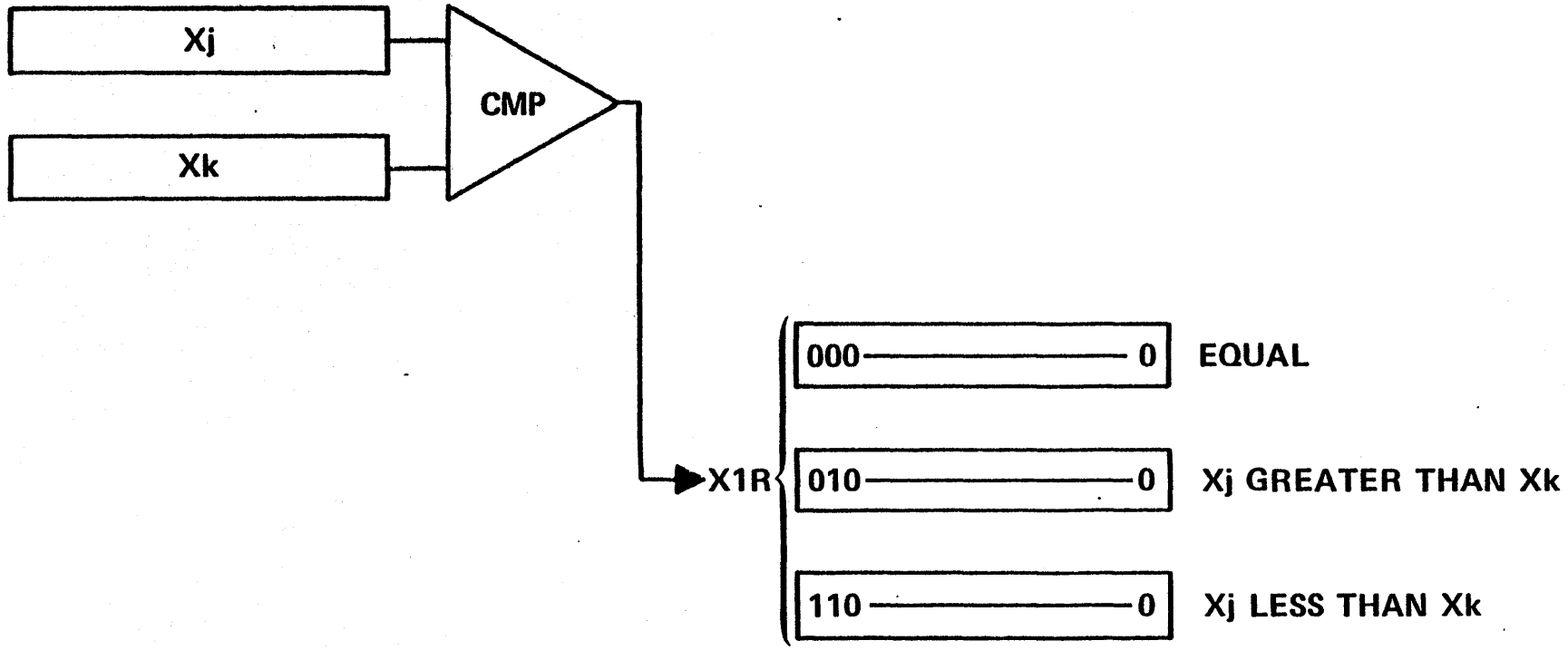
GENERAL INSTRUCTIONS

REF NO.	OP CODE	FORMAT	MNEMONIC	NAME
065	18	jk	IORX	LOGICAL SUM
066	19	jk	XORX	LOGICAL DIFFERENCE
067	1A	jk	ANDX	LOGICAL PRODUCT
068	1B	jk	NOTX	LOGICAL COMPLEMENT
069	1C	jk	INHX	LOGICAL INHIBIT
070	AC	jkiD	ISOM	ISOLATE BIT MASK
071	AD	jkiD	ISOB	ISOLATE
072	AE	jkiD	INSB	INSERT
145	1E	jk	MARK	MARK TO BOOLEAN
046	9D	jkQ	BRSEG	BRANCH ON SEGMENTS UNEQUAL
047	2E	jk	BRREL	BRANCH RELATIVE
048	2F	jk	BRDIR	INTER-SEGMENT BRANCH

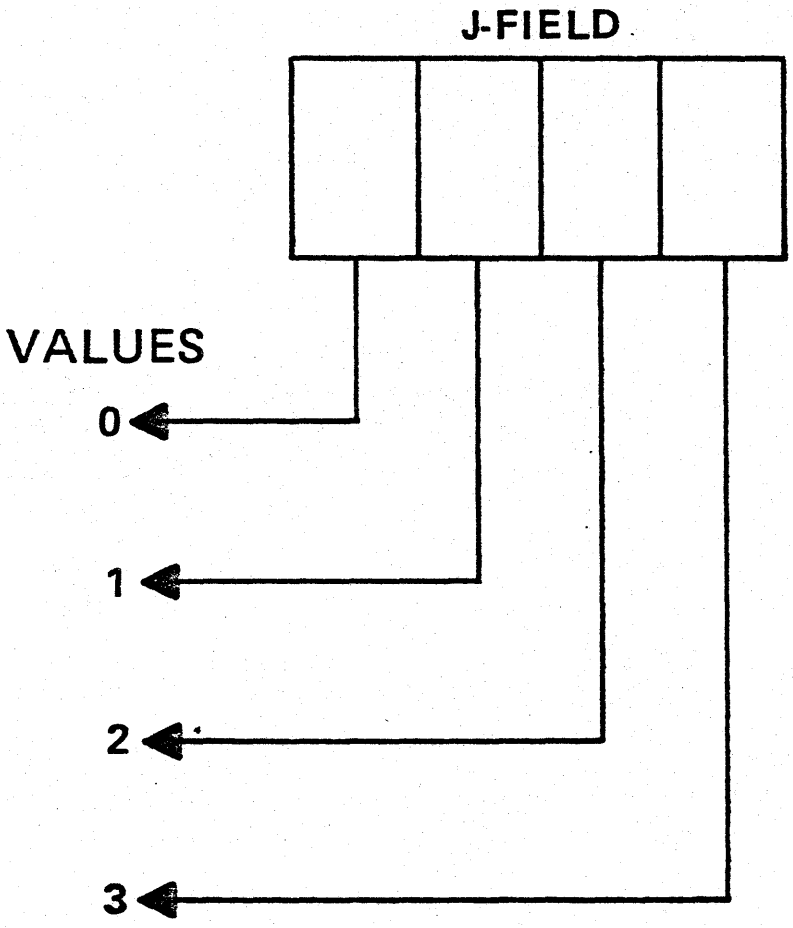
ISOLATE TO Xk



RESULTS OF INTEGER COMPARE



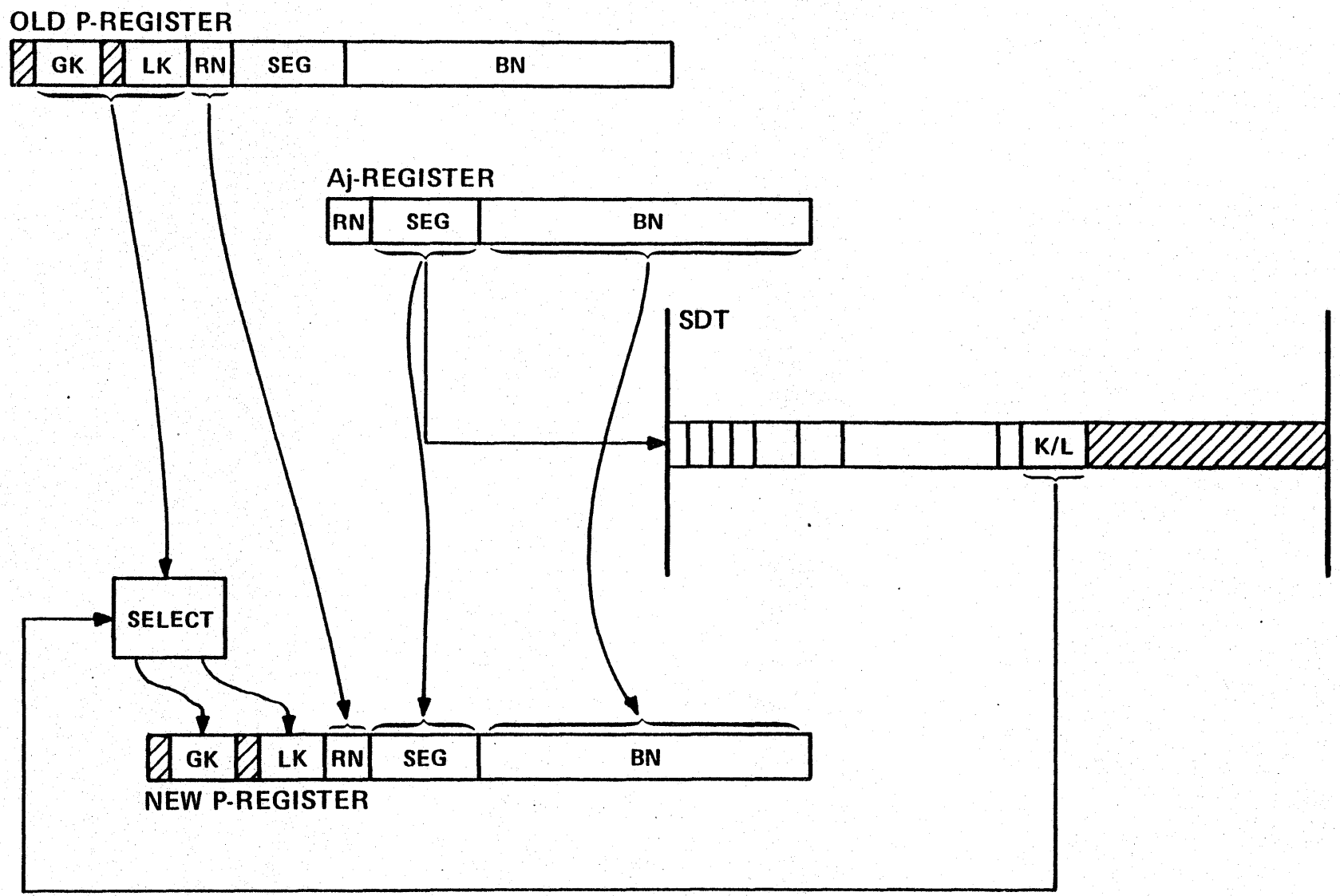
MARK TO BOOLEAN J-FIELD USAGE



MARK TO BOOLEAN TESTS

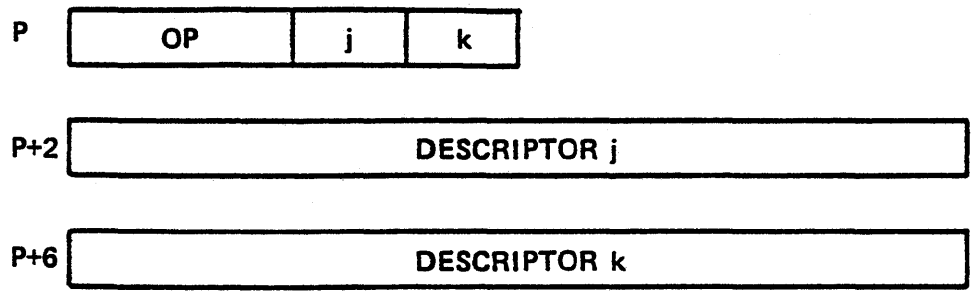
j-FIELD		TEST (LITERAL)	TEST ACTUAL	REQD
HEX	BINARY			
0	0000	NONE	NONE	NO
1	0001	<	<	YES
2	0010	NONE	NONE	NO
3	0011	<	<	NO
4	0100	>	>	YES
5	0101	> & <	≠	YES
6	0110	>	>	NO
7	0111	> & <	≠	NO
8	1000	=	=	YES
9	1001	< & =	≤	YES
A	1010	=	=	NO
B	1011	< & =	≤	NO
C	1100	> & =	≥	YES
D	1101	>, < & =	ALL	NO
E	1110	> & =	≥	NO
F	1111	>, < & =	ALL	NO

INTER-SEGMENT BRANCH

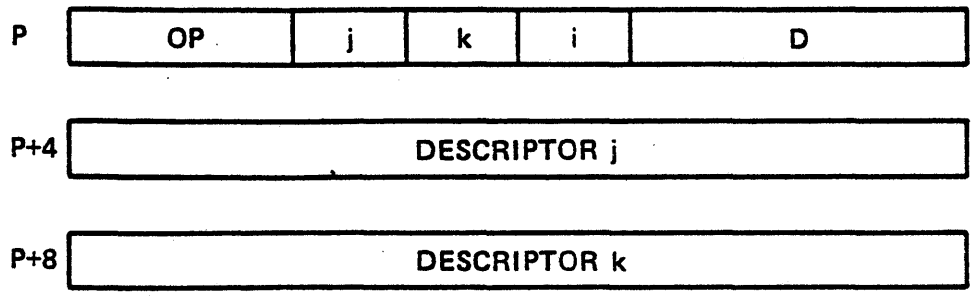


BDP OPERATION CODES

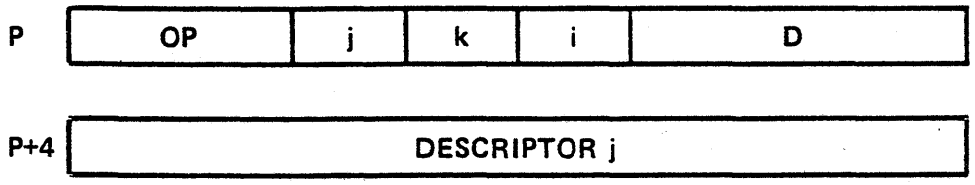
jk PLUS TWO DESCRIPTORS



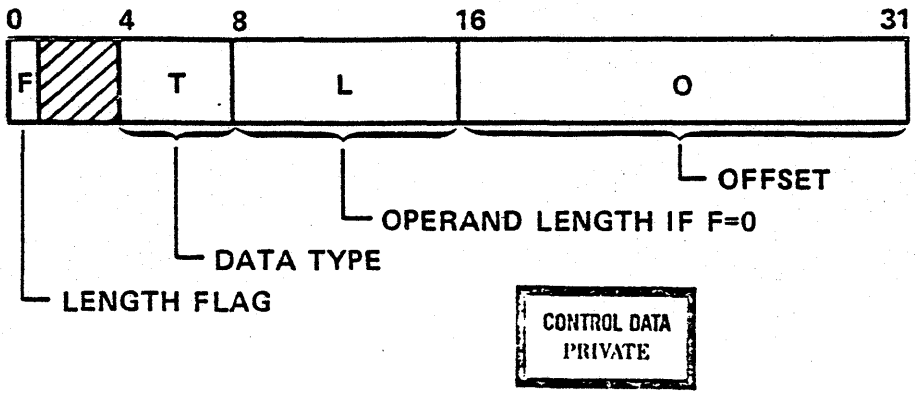
jkiD PLUS TWO DESCRIPTORS



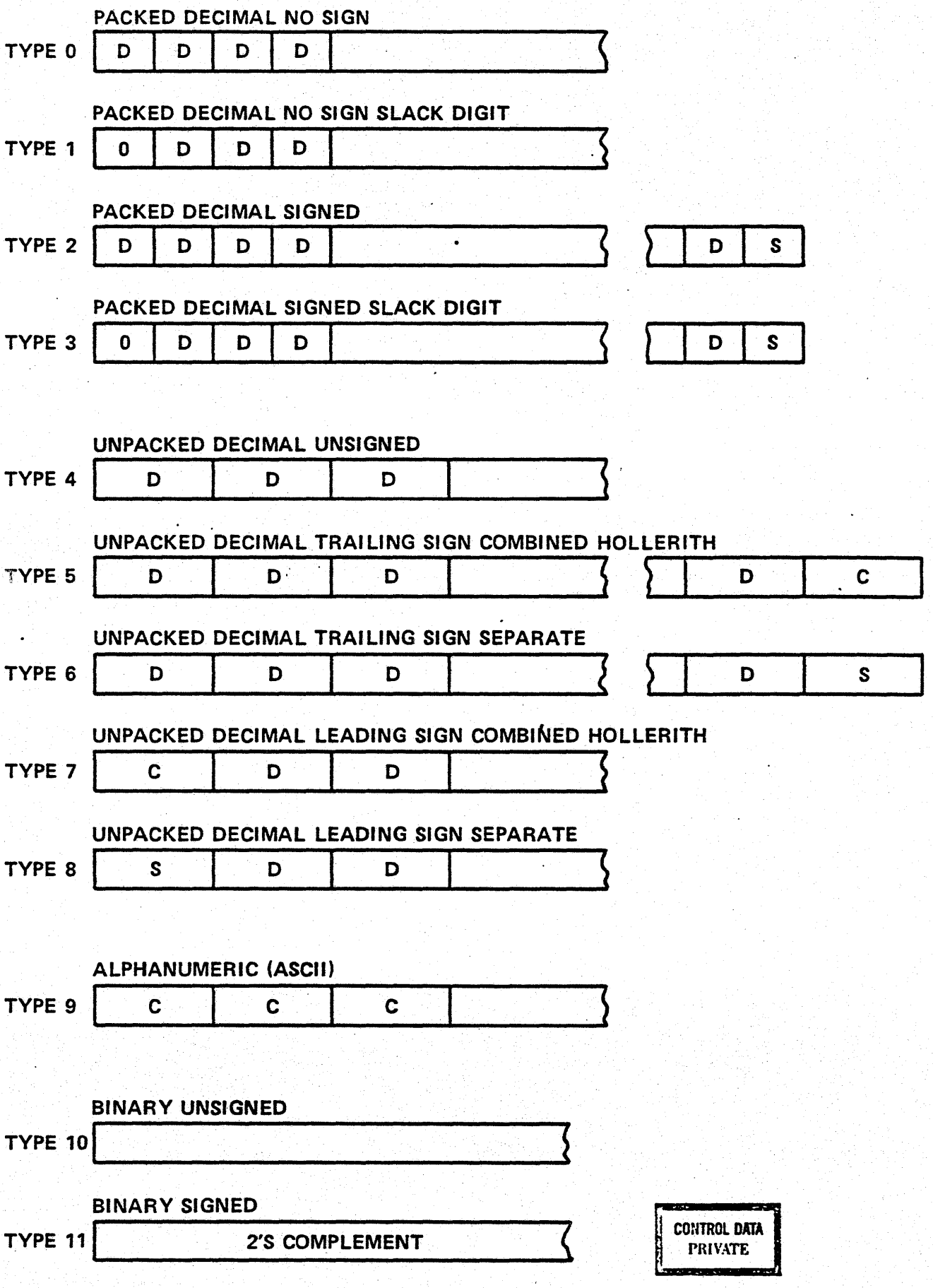
jkiD PLUS ONE DESCRIPTOR



BDP DESCRIPTOR

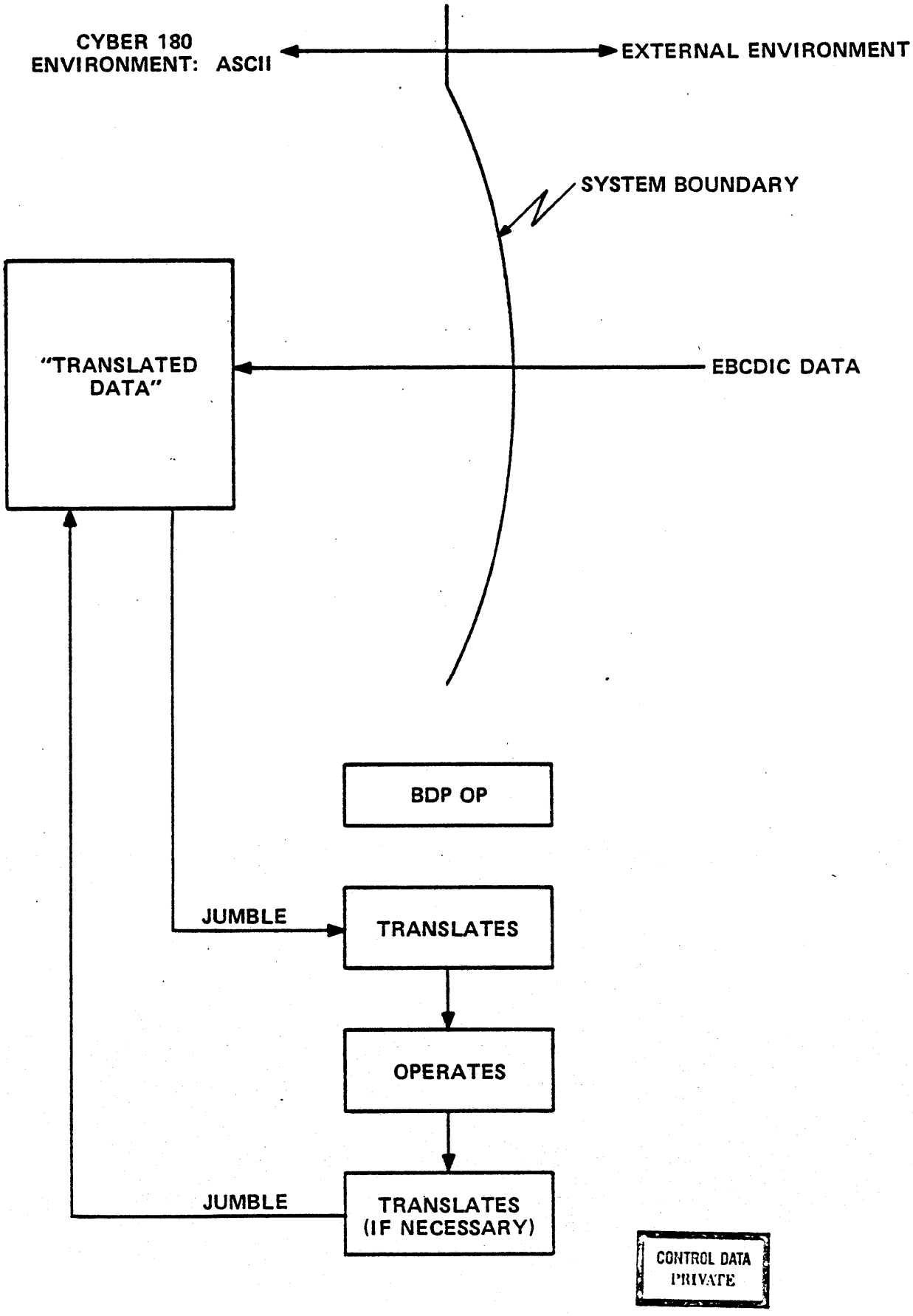


BDP DATA TYPES



CONTROL DATA
PRIVATE

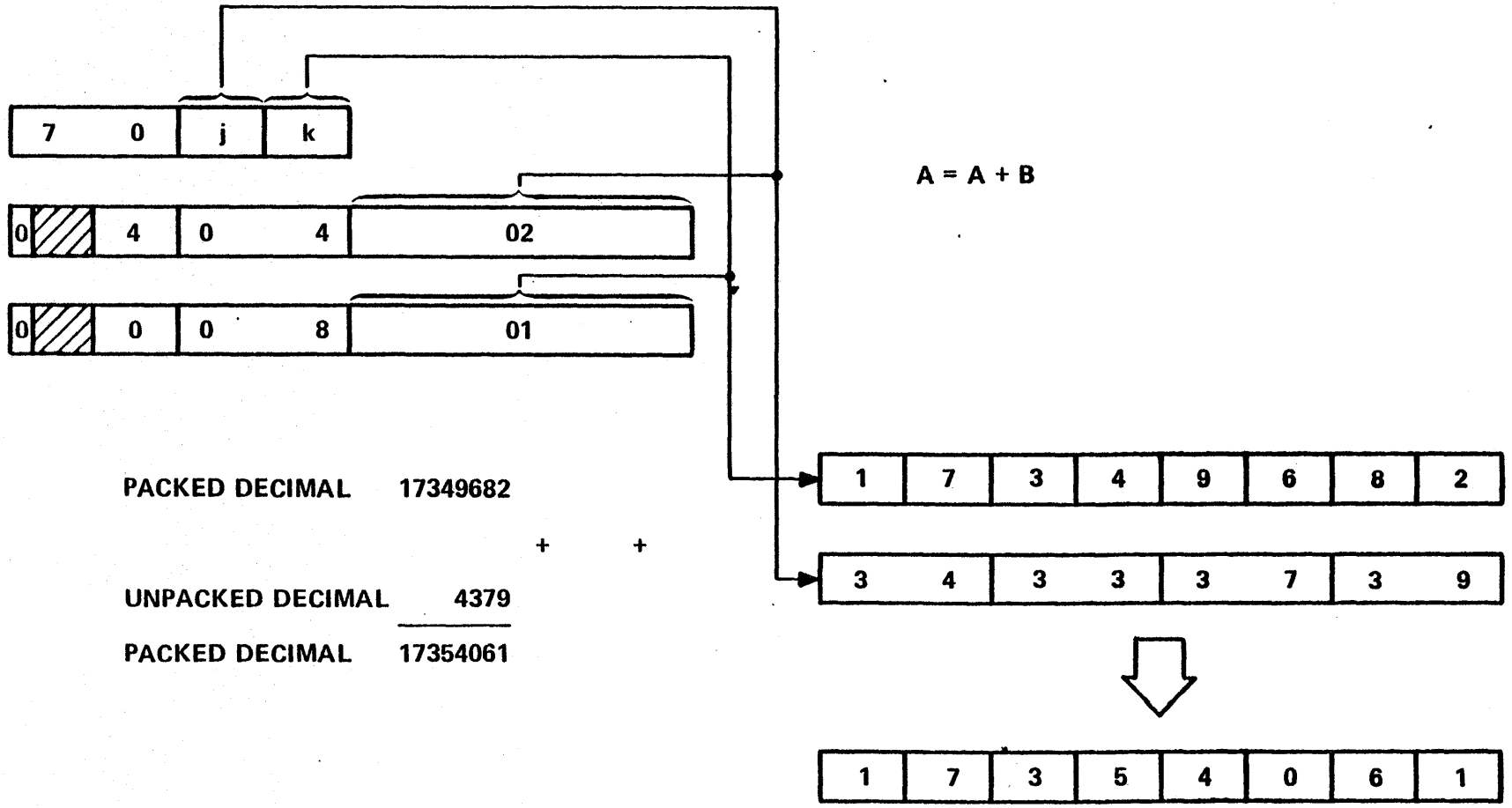
"TRANSLATED" DATA TYPES



BDP INSTRUCTIONS

REF NO.	OP CODE	FORMAT	MNEMONIC	NAME
074	70	jk	ADDN,Aj,X0	DECIMAL SUM
075	71	jk	SUBN,Aj,X0	DECIMAL DIFFERENCE
076	72	jk	MULN,Aj,X0	DECIMAL PRODUCT
077	73	jk	DIVN,Aj,X0	DECIMAL QUOTIENT
078	E4	jkiD(2)	SCLN,Aj,X0	DECIMAL SCALE
079	E5	jkiD(2)	SCLR,Aj,X0	DECIMAL SCALE ROUNDED
083	74	jk	CMPN,Aj,X0	DECIMAL COMPARE
092	75	jk	MOVN,Aj,X0	DECIMAL MOVE
084	77	jk	CMPB,Aj,X0	BYTE COMPARE
085	E9	jkiD(2)	CMPC,Aj,X0	BYTE COMPARE COLLATED
086	F3	jkiD(1)	SCNB,Aj,X0	BYTE SCAN WHILE NON-MEMBER
088	EB	jkiD(2)	TRANB,Aj,X0	BYTE TRANSLATE
089	76	jk	MOVB,Aj,X0	MOVE BYTES
091	ED	jkiD(2)	EDIT,Aj,X0	EDIT
154	F9	jkiD(1)	MOVI,Ai,D	MOVE IMMEDIATE DATA
155	FA	jkiD(1)	CMPI,Ai,D	COMPARE IMMEDIATE DATA
156	FB	jkiD(1)	ADDI,Ai,D	ADD IMMEDIATE DATA
096	F4	jkiD(1)	CALDF,Aj,X0	CALCULATE SUBSCRIPT AND ADD

EXAMPLE OF A DECIMAL ADD

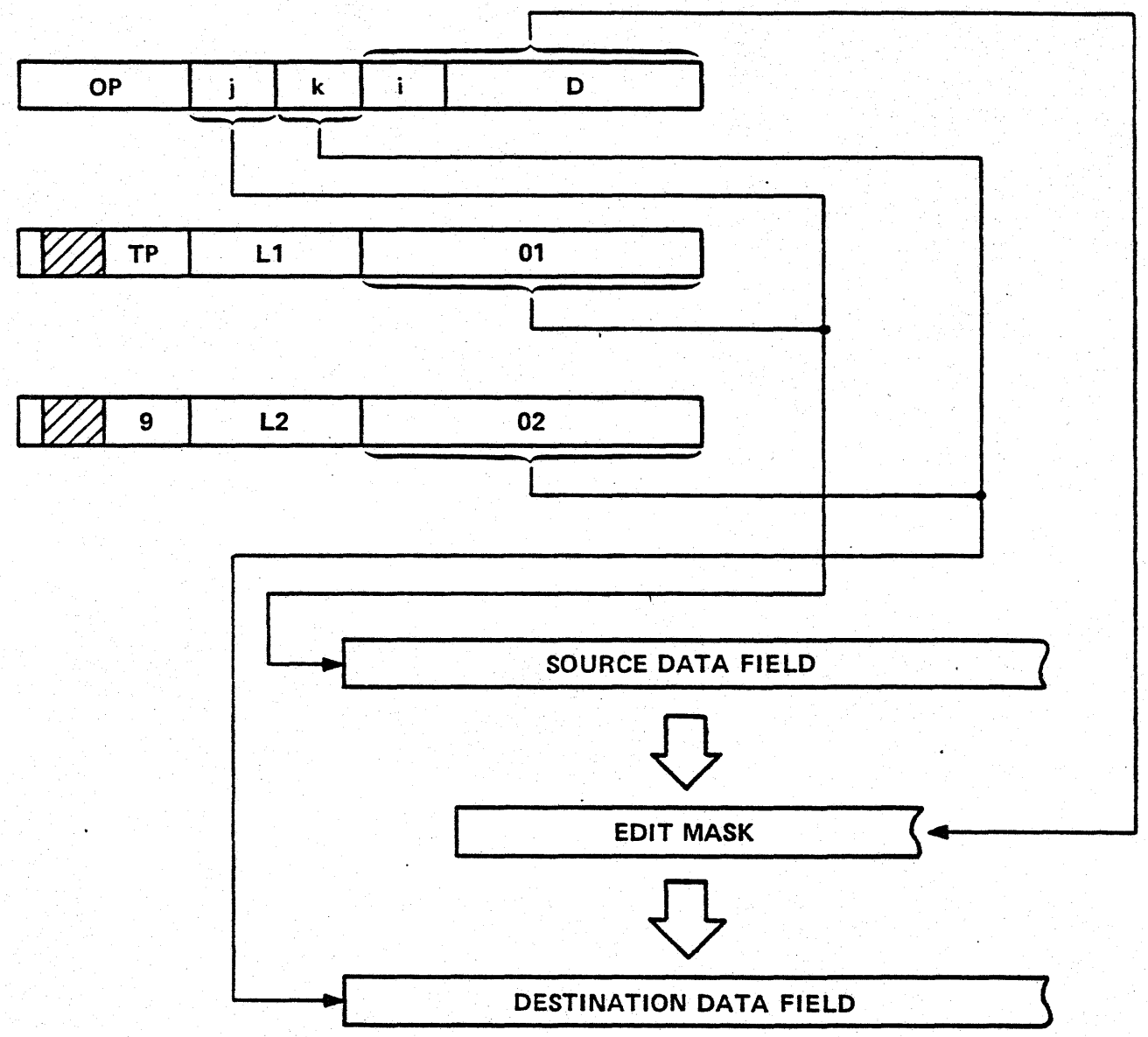


ADD $A_k + 0_1$ (TYPE 0 - PACKED DECIMAL UNSIGNED, LENGTH 8) TO

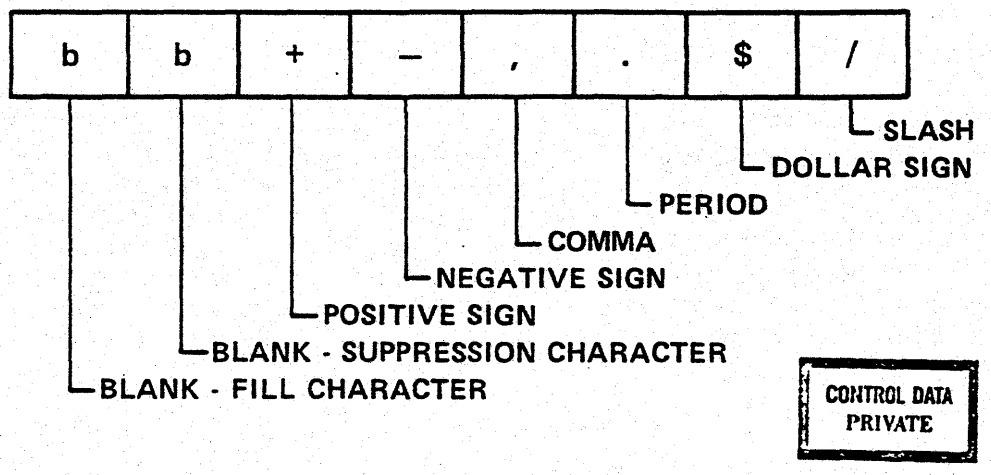
$A_j + 0_2$ (TYPE 4 - UNPACKED DECIMAL UNSIGNED, LENGTH 4) RESULT TO

$A_k + 0_2$ (SAME TYPE AS SECOND OPERAND)

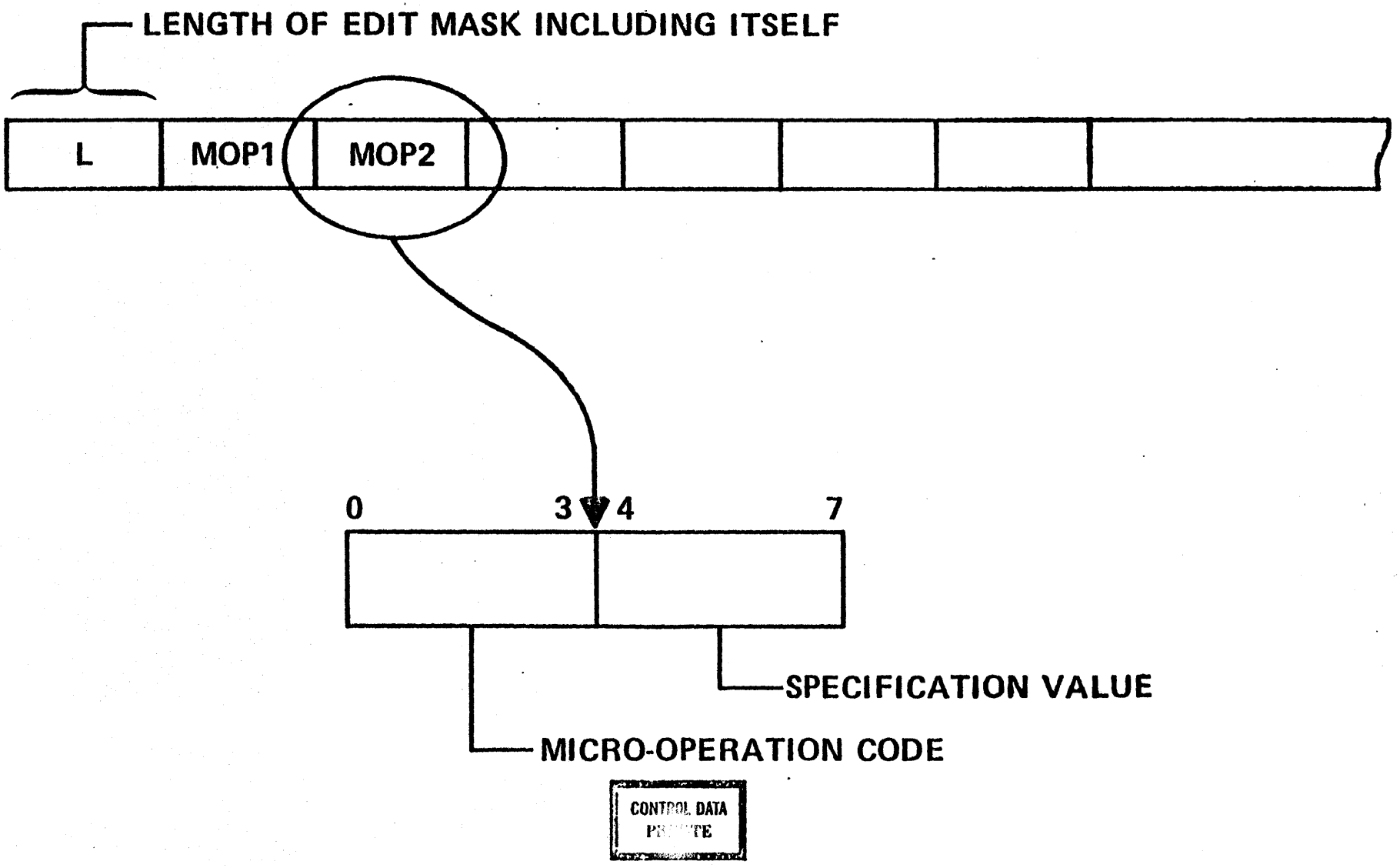
EDIT INSTRUCTION



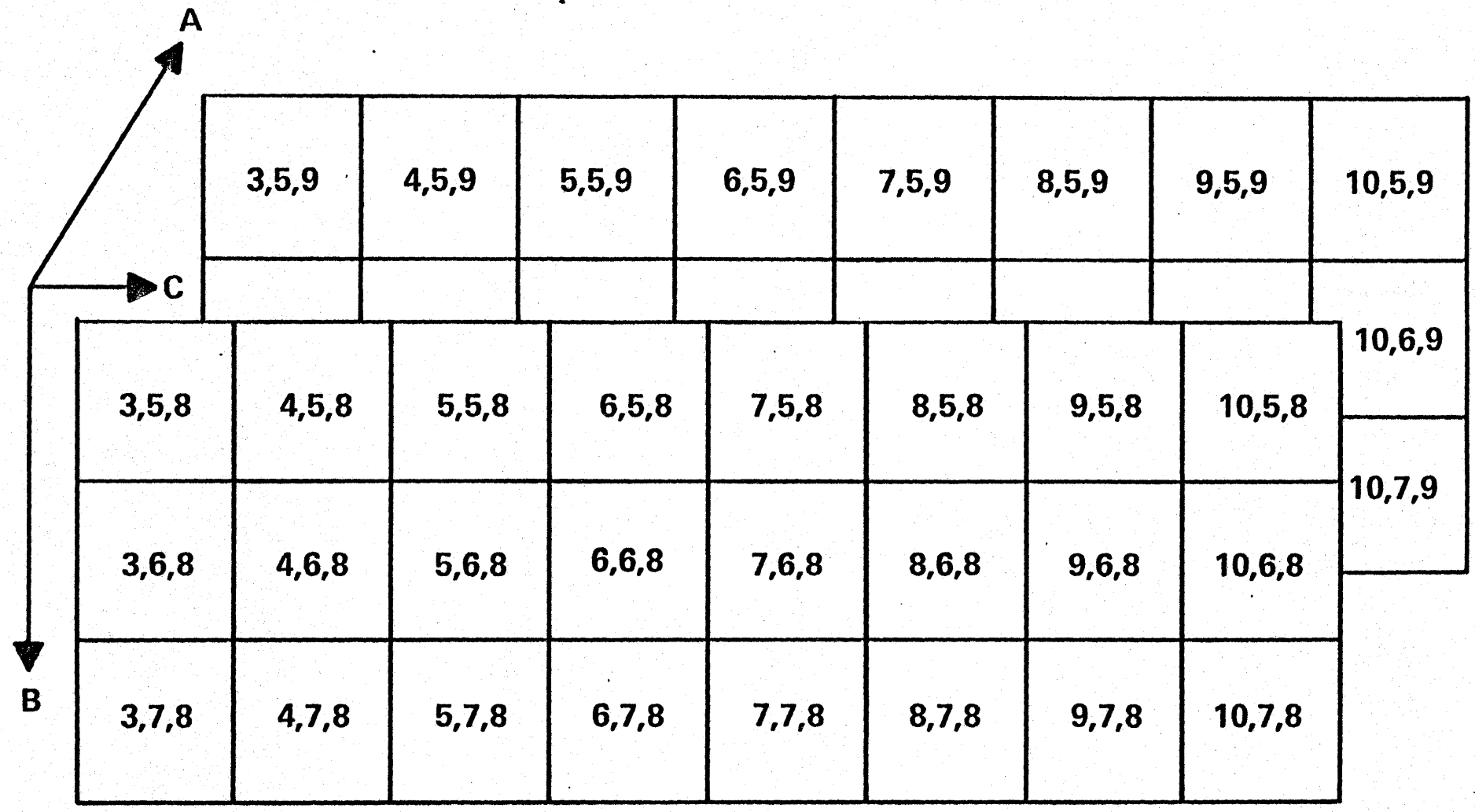
INITIAL VALUES OF THE SPECIAL CHARACTERS TABLE



THE EDIT MASK



ARRAY (3..10, 5..7, 8..9)



INDEX COMPUTATION

c b a		
3,5,8	00	0,0,0
4,5,8	1	1,0,0
5,5,8	2	2,0,0
6,5,8	3	3,0,0
7,5,8	4	4,0,0
8,5,8	5	5,0,0
9,5,8	6	6,0,0
10,5,8	7	7,0,0
3,6,8	8	0,1,0
4,6,8	9	1,1,0
5,6,8	10	2,1,0
6,6,8	11	3,1,0
7,6,8	12	4,1,0
8,6,8	13	5,1,0
9,6,8	14	6,1,0
10,6,8	15	7,1,0
3,7,8	16	0,2,0
4,7,8	17	1,2,0
5,7,8	18	2,2,0
6,7,8	19	3,2,0
7,7,8	20	4,2,0
8,7,8	21	5,2,0
9,7,8	22	6,2,0
10,7,8	23	7,2,0
3,5,9	24	0,0,1
4,5,9	25	1,0,1
5,5,9	26	2,0,1
6,5,9	27	3,0,1
7,5,9	28	4,0,1
8,5,9	29	5,0,1
9,5,9	30	6,0,1
10,5,9	31	7,0,1
3,6,9	32	0,1,1
4,6,9.	33	1,1,1
5,6,9	34	2,1,1
6,6,9	35	3,1,1
7,6,9	36	4,1,1
8,6,9	37	5,1,1
9,6,9	38	6,1,1
10,6,9	39	7,1,1
3,7,9	40	0,2,1
4,7,9	41	1,2,1
5,7,9	42	2,2,1
6,7,9	43	3,2,1
7,7,9	44	4,2,1
8,7,9	45	5,2,1
9,7,9	46	6,2,1
10,7,9	47	7,2,1

ARRAY [3. .10,5. .7,8. .9]

C(8) x B(3) x A(2)

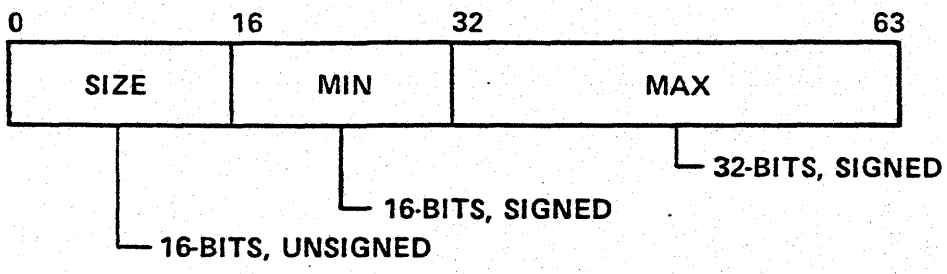
INDEX:

$[(c - \bar{c})\ \hat{b} + (b - \bar{b})]\ \hat{a} + (a - \bar{a})$

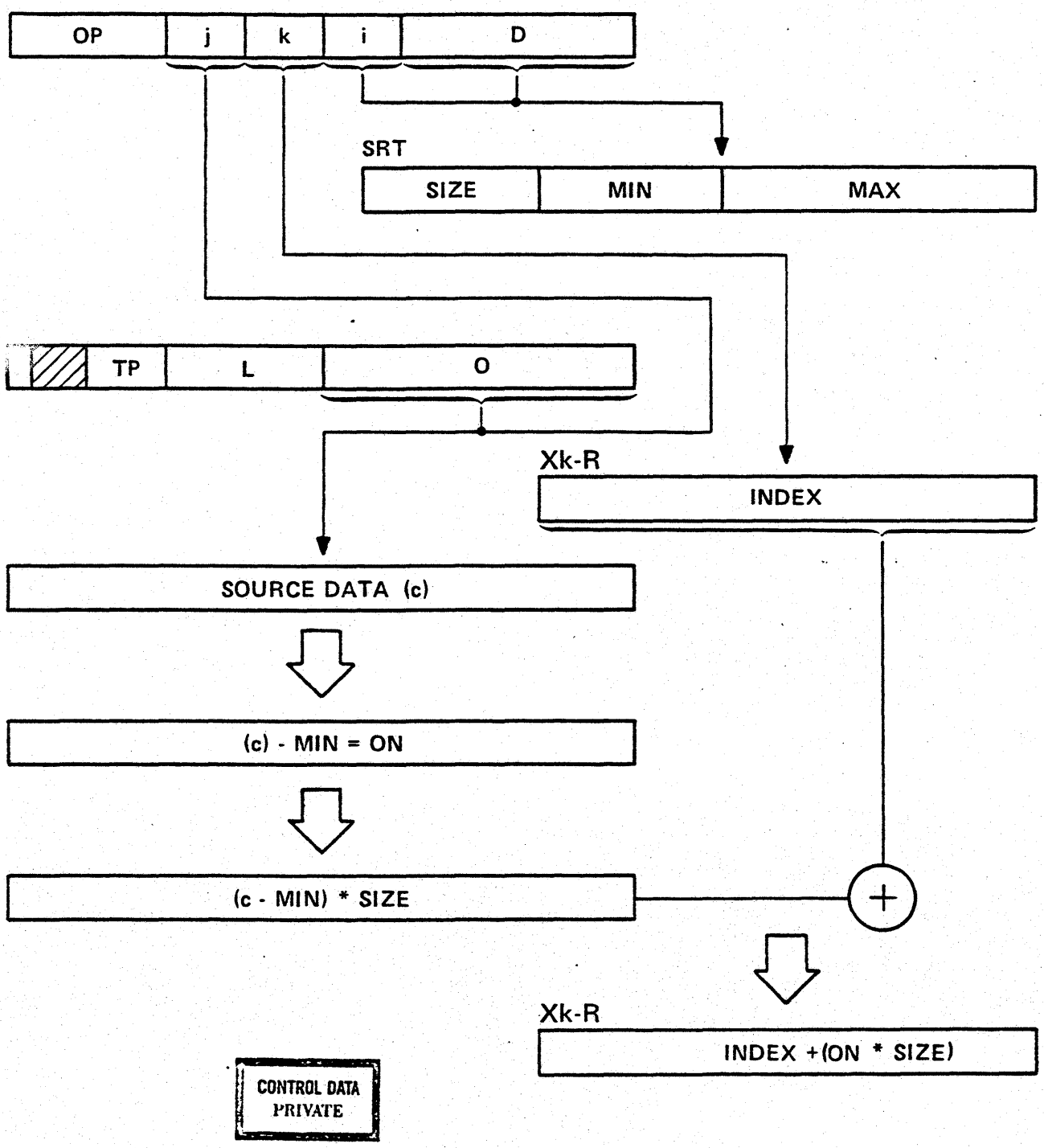
$\hat{b}$ = bmax - bmin + 1 = SIZE

$\bar{b}$ = bmin = MIN

SUBSCRIPT RANGE TABLE



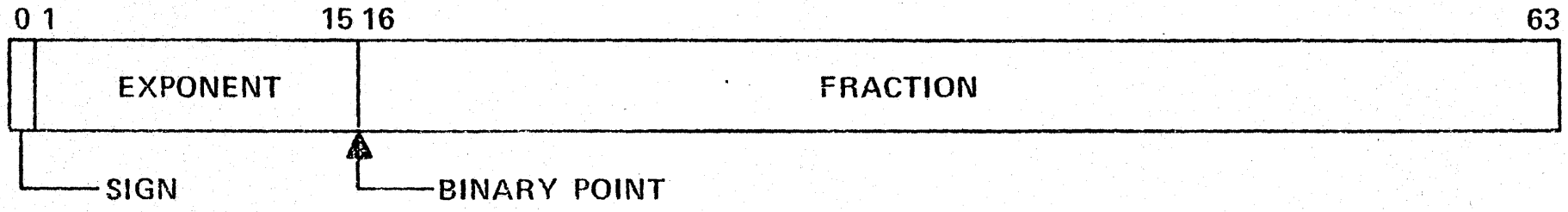
CALCULATE SUBSCRIPT OPERATION



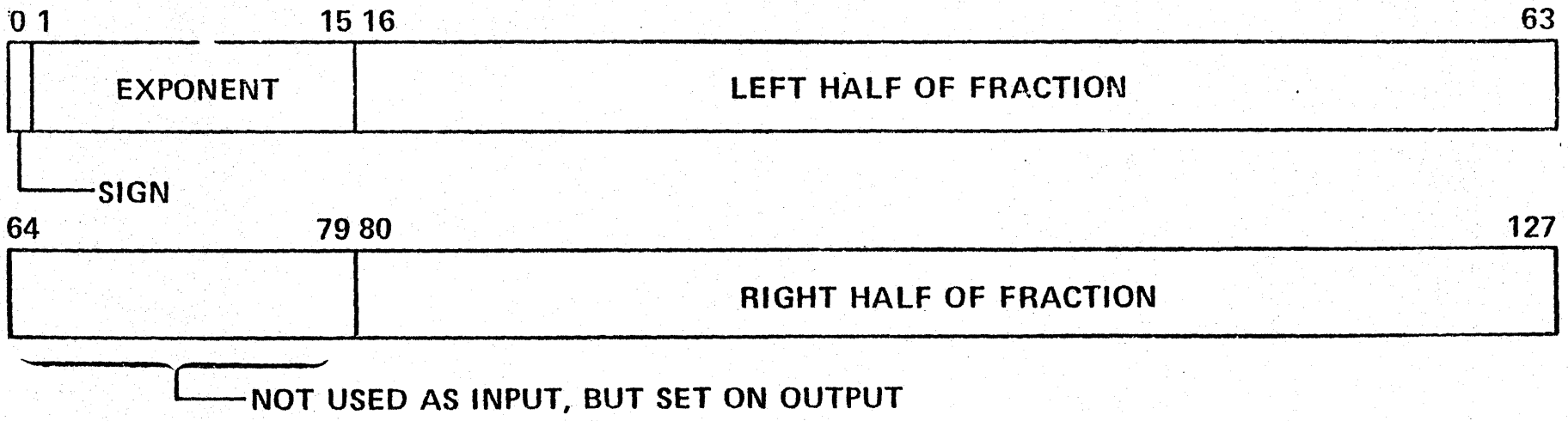
FLOATING POINT INSTRUCTIONS

REF NO.	OP CODE	FORMAT	MNEMONIC	NAME
097	3A	jk	CONI	FLOATING POINT CONVERT FROM INTEGER
098	3B	jk	CONF	FLOATING POINT CONVERT TO INTEGER
099	30	jk	ADDF	SINGLE PRECISION FLOATING POINT SUM
100	31	jk	SUBF	SINGLE PRECISION FLOATING POINT DIFFERENCE
103	32	jk	MULF	SINGLE PRECISION FLOATING POINT PRODUCT
104	33	jk	DIVF	SINGLE PRECISION FLOATING POINT QUOTIENT
109	98	jkQ	BRFEQ	BRANCH ON EQUAL
110	99	jkQ	BRFNE	BRANCH ON NOT EQUAL
111	9A	jkQ	BRFGT	BRANCH ON GREATER THAN
112	9B	jkQ	BRFGE	BRANCH ON GREATER THAN OR EQUAL
113	9E	jkQ	BROVR	BRANCH ON OVERFLOW
			BRUND	BRANCH ON UNDERFLOW
			BRINF	BRANCH ON INDEFINITE
114	3C	jk	CMPF	FLOATING POINT COMPARE
105	34	jk	ADDD	DOUBLE PRECISION FLOATING POINT SUM
106	35	jk	SUBD	DOUBLE PRECISION FLOATING POINT DIFFERENCE
107	36	jk	MULD	DOUBLE PRECISION FLOATING POINT PRODUCT
108	37	jk	DIVD	DOUBLE PRECISION FLOATING POINT QUOTIENT

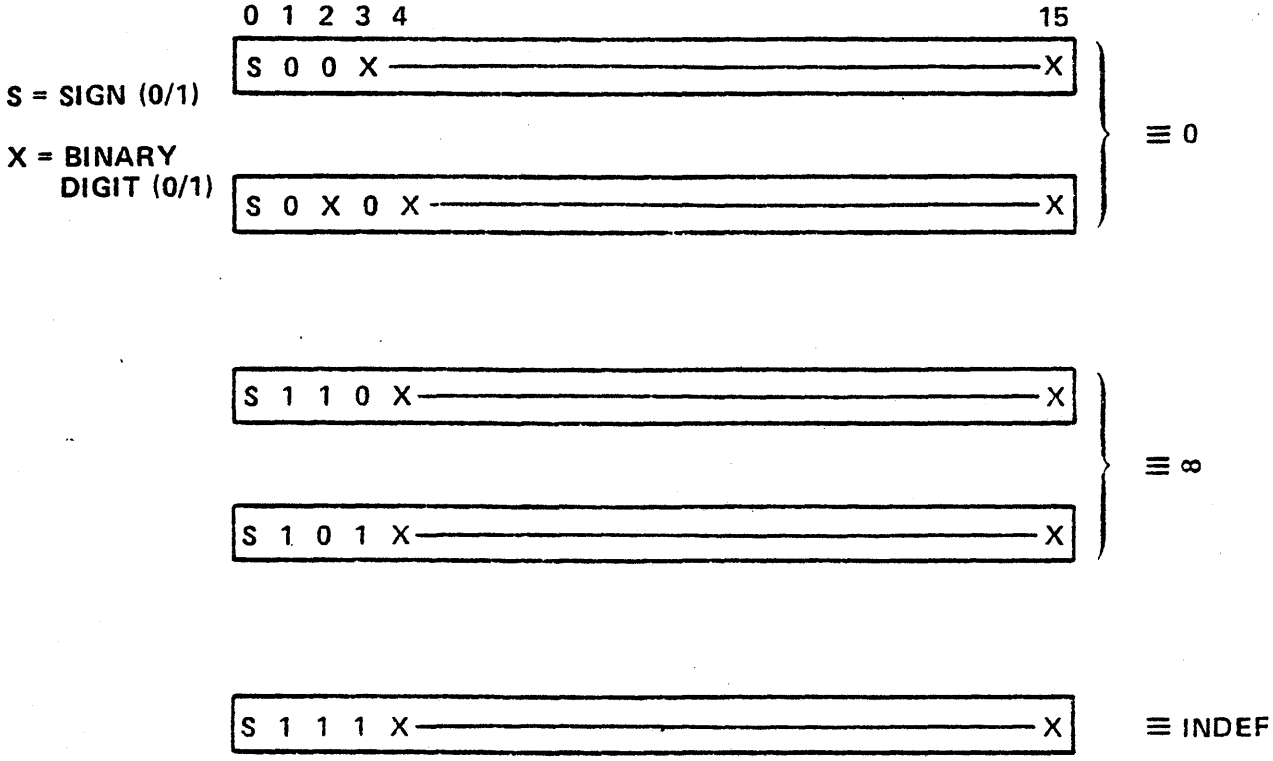
SINGLE PRECISION FLOATING-POINT FORMAT



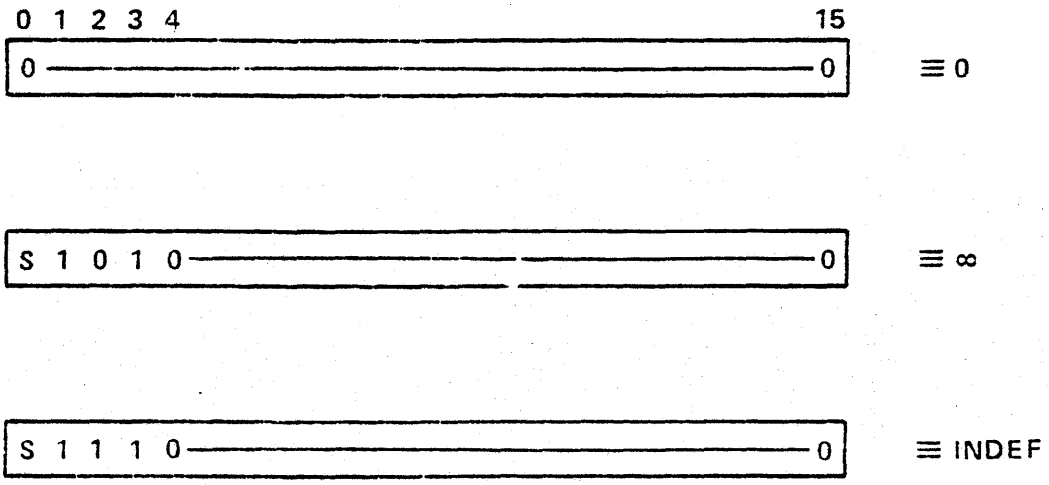
DOUBLE PRECISION FLOATING-POINT FORMAT



NON-STANDARD FLOATING-POINT NUMBERS – INPUT



NON-STANDARD FLOATING-POINT NUMBERS – OUTPUT



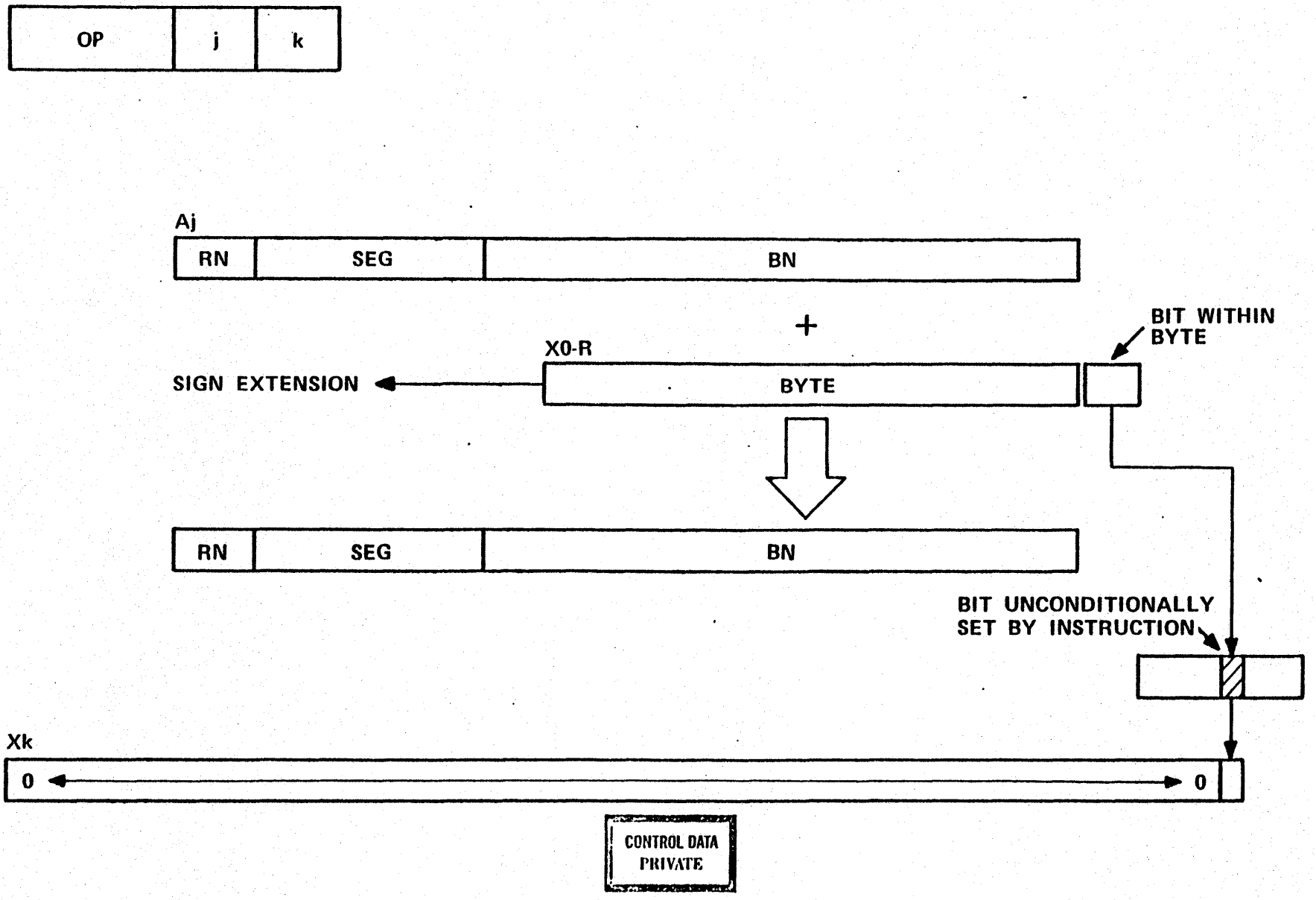
FLOATING POINT REPRESENTATION

	HEXADECIMAL EXPONENT INCLUDING CO-EFFICIENT SIGN	ACTUAL EXPONENT (TO THE BASE 2)	INPUT ARGUMENTS	RESULTS
COEFFICIENT SIGN EQUAL TO 0 (POSITIVE NUMBERS)	7XXX	– – – –	INDEFINITE	7000.0 ⟶ 0
	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	$2^{-4,097}$ ↓ $2^{-12,288}$	ZERO	UNDERFLOW MASK = 0 : 0000.0 ⟶ 0 UNDERFLOW MASK = 1 : AS SHOWN
	0XXX	– – –	ZERO	NOT APPLICABLE
COEFFICIENT SIGN EQUAL TO 1 (NEGATIVE NUMBERS)	8XXX			
	9000 ↑ AFFF	$2^{-12,288}$ ↑ $2^{-4,097}$	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

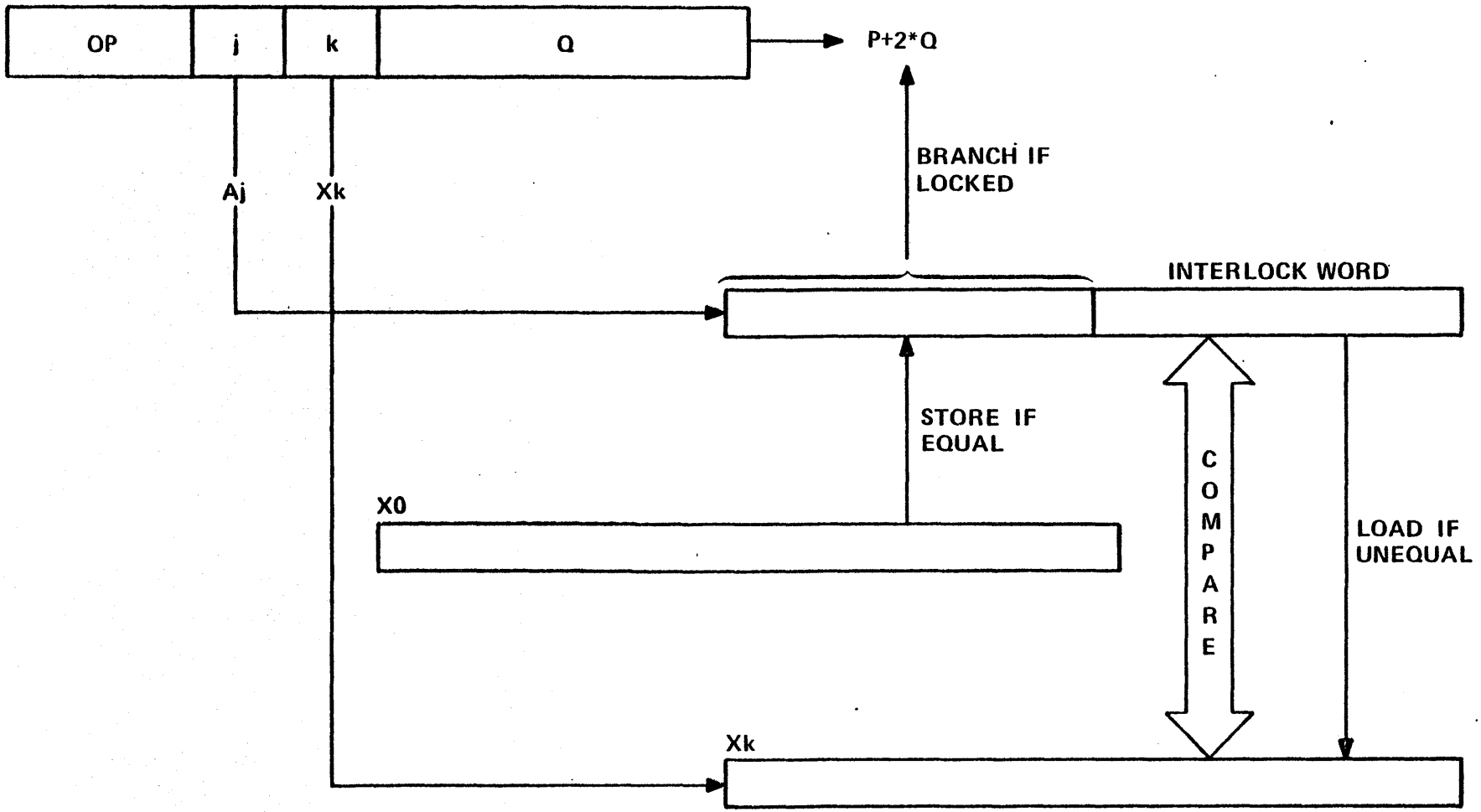
SYSTEM INSTRUCTIONS

REF NO.	OP CODE	FORMAT	MNEMONIC	NAME
115	B5	jkQ	CALLSEG	CALL INDIRECT
116	B0	jkQ	CALLREL	CALL RELATIVE
117	04	jk	RETURN	RETURN
118	06	jk	POP	POP
120	02	jk	EXCHANGE	EXCHANGE
122	03	jk	INTRUPT	PROCESSOR INTERRUPT
134	9F	jkQ	BRCR	BRANCH ON CONDITION REGISTER
124	14	jk	LBSET	TEST AND SET BIT
125	B4	jkQ	CMPXA	COMPARE SWAP
126	16	jk	TPAGE	TEST AND SET PAGE
127	17	jk	LPAGE	LOAD PAGE TABLE INDEX
130	0E	jk	CPYSX	COPY FROM STATE REGISTER
131	0F	jk	CPYXS	COPY TO STATE REGISTER
132	08	jk	CPYMX	COPY FREE RUNNING COUNTER
138	05	jk	PURGE	PURGE BUFFER
136	B1	jkQ	KEYPOINT	KEYPOINT
139	CX	SjkiD	EXECUTE,S	EXECUTE ALGORITHM
121	00	jk	HALT	PROGRAM ERROR

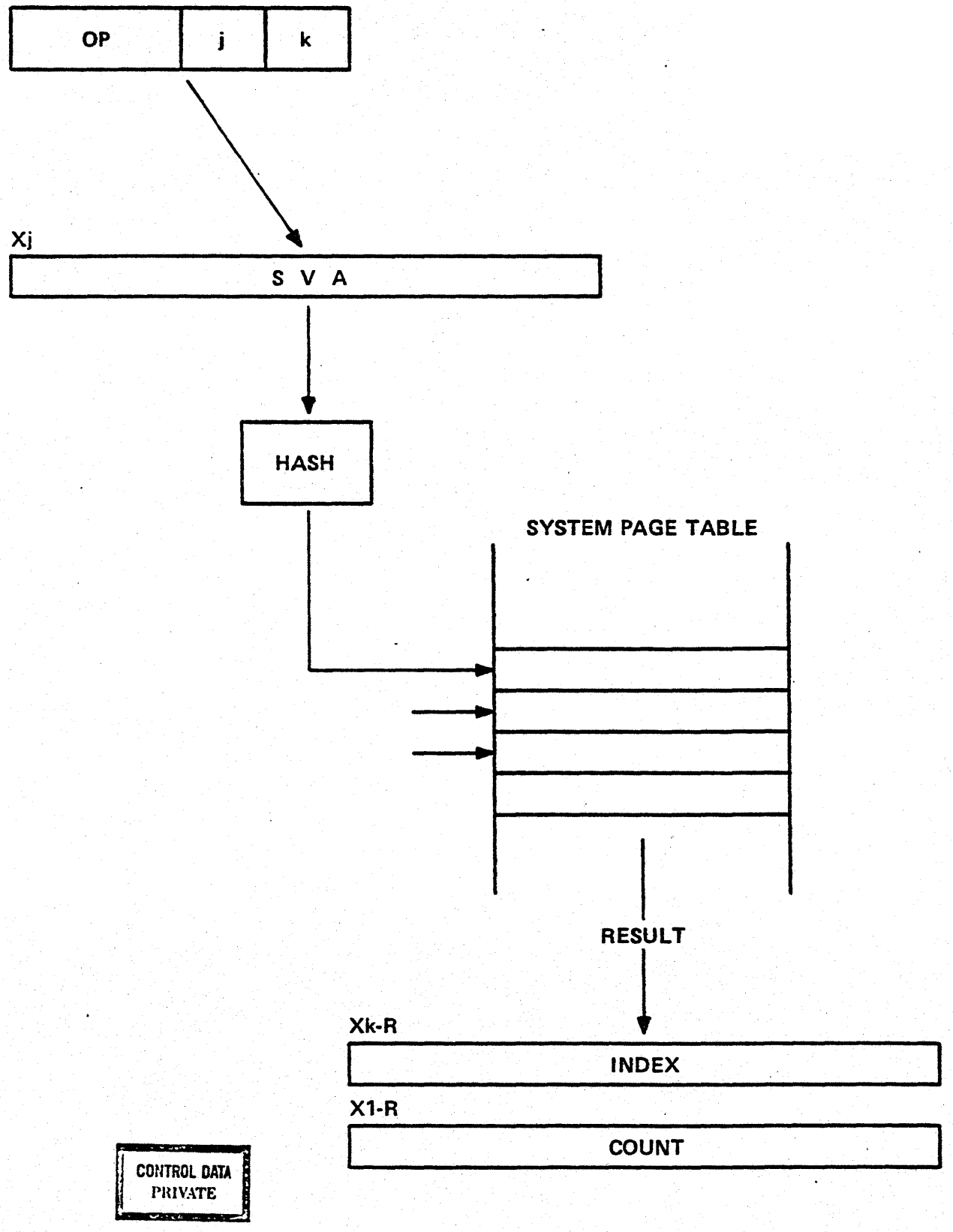
TEST AND SET BIT



COMPARE/SWAP OPERATION



LOAD PAGE TABLE INDEX



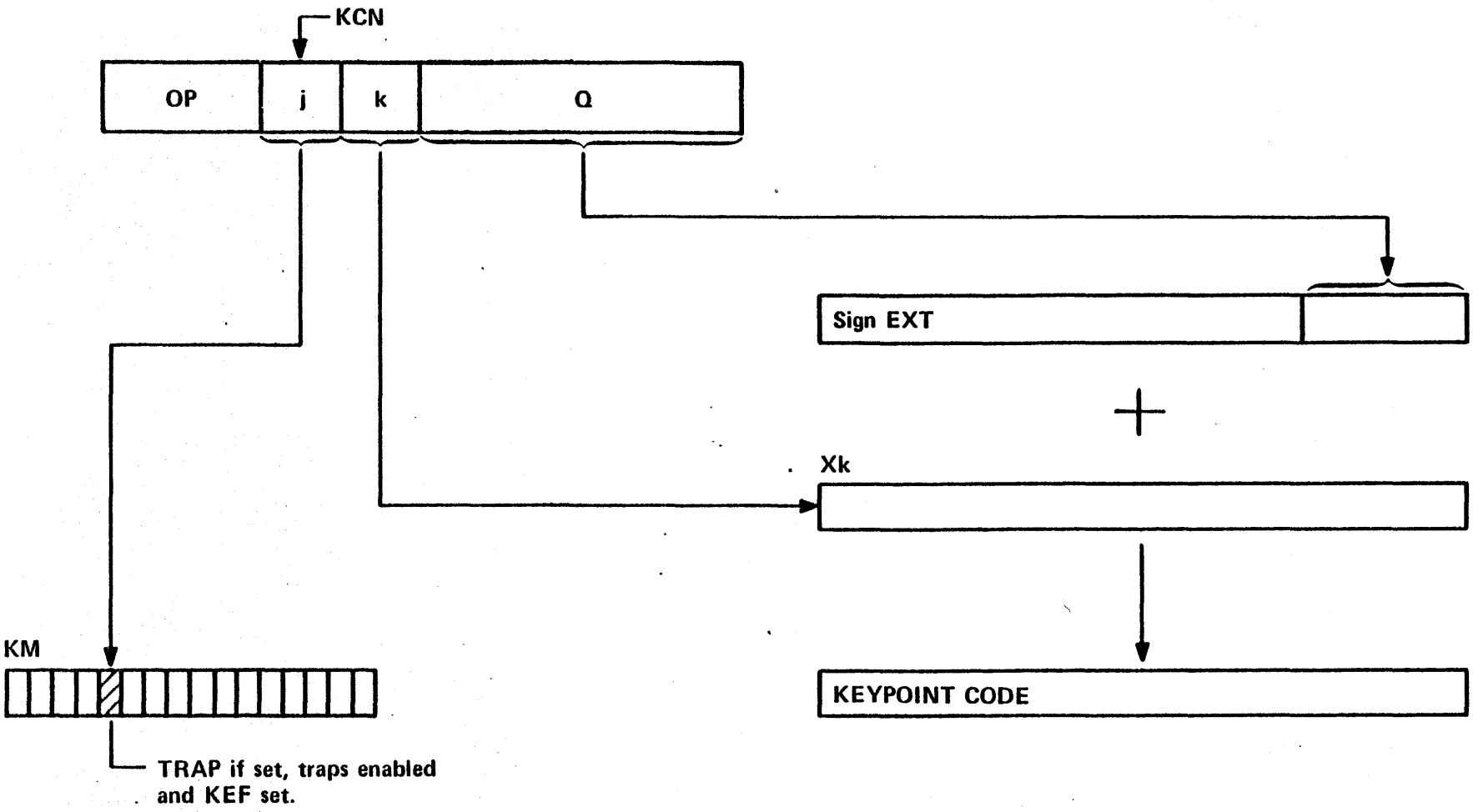
REGISTER DEFINITIONS AND ACCESSES

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			

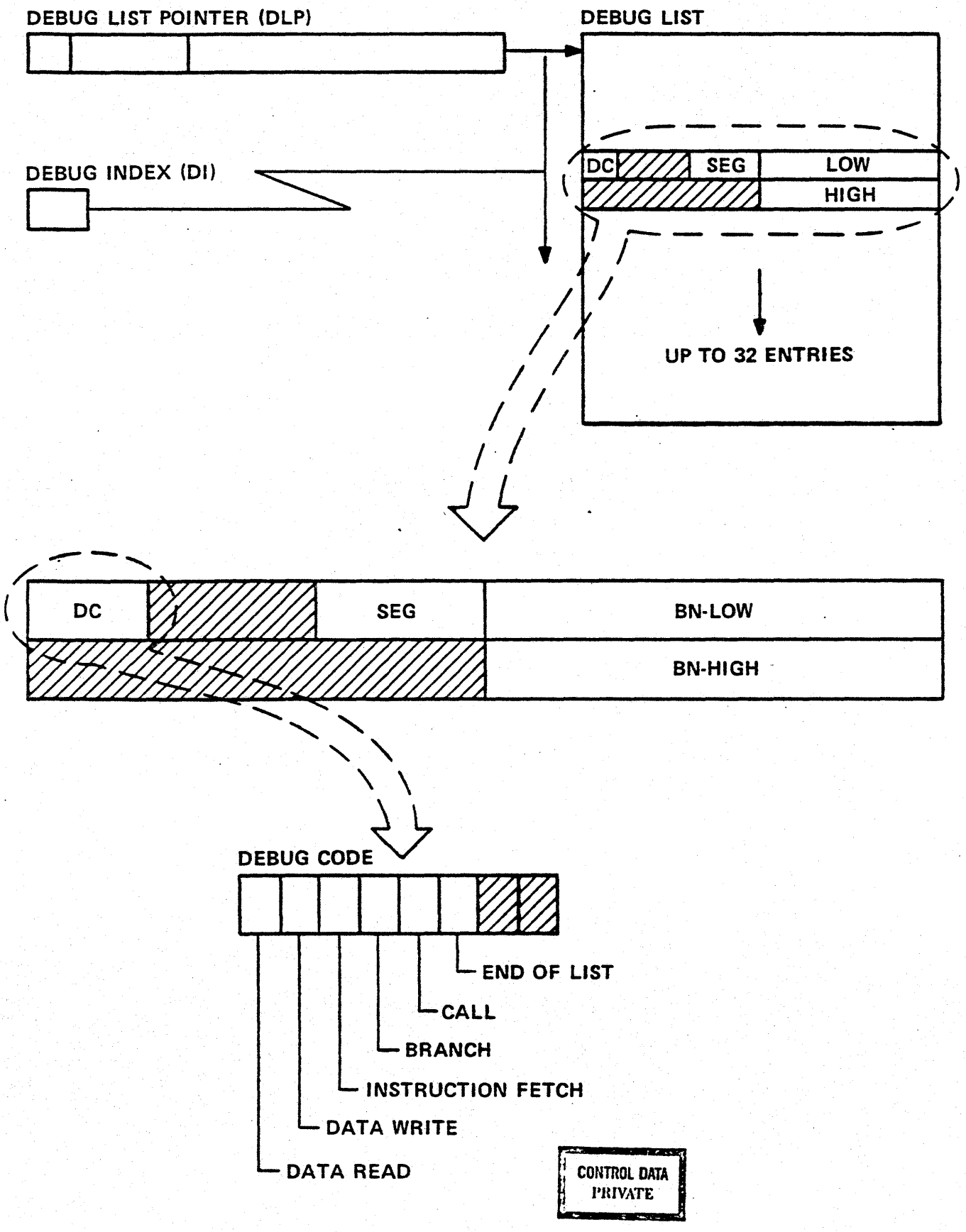
SYSTEM INSTRUCTIONS

REF NO.	OP CODE	FORMAT	MNEMONIC	NAME
115	B5	jkQ	CALLSEG	CALL INDIRECT
116	B0	jkQ	CALLREL	CALL RELATIVE
117	04	jk	RETURN	RETURN
118	06	jk	POP	POP
120	02	jk	EXCHANGE	EXCHANGE
122	03	jk	INTRUPT	PROCESSOR INTERRUPT
134	9F	jkQ	BRCR	BRANCH ON CONDITION REGISTER
124	14	jk	LBSET	TEST AND SET BIT
125	B4	jkQ	CMPXA	COMPARE SWAP
126	16	jk	TPAGE	TEST AND SET PAGE
127	17	jk	LPAGE	LOAD PAGE TABLE INDEX
130	0E	jk	CPYSX	COPY FROM STATE REGISTER
131	0F	jk	CPYXS	COPY TO STATE REGISTER
132	08	jk	CPYMX	COPY FREE RUNNING COUNTER
138	05	jk	PURGE	PURGE BUFFER
136	B1	jkQ	KEYPOINT	KEYPOINT
139	CX	SjkiD	EXECUTE,S	EXECUTE ALGORITHM
121	00	jk	HALT	PROGRAM ERROR

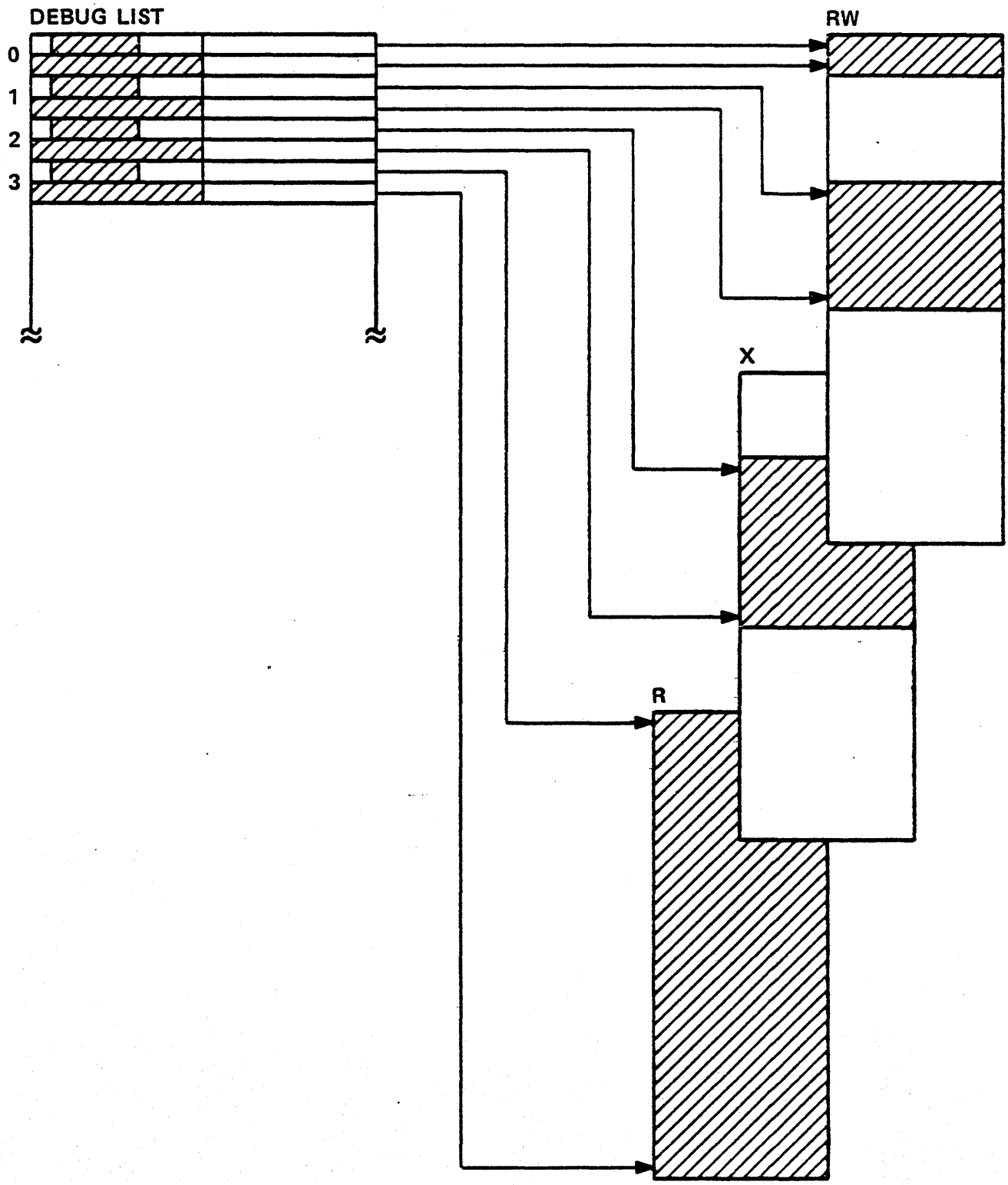
KEYPOINT OPERATION



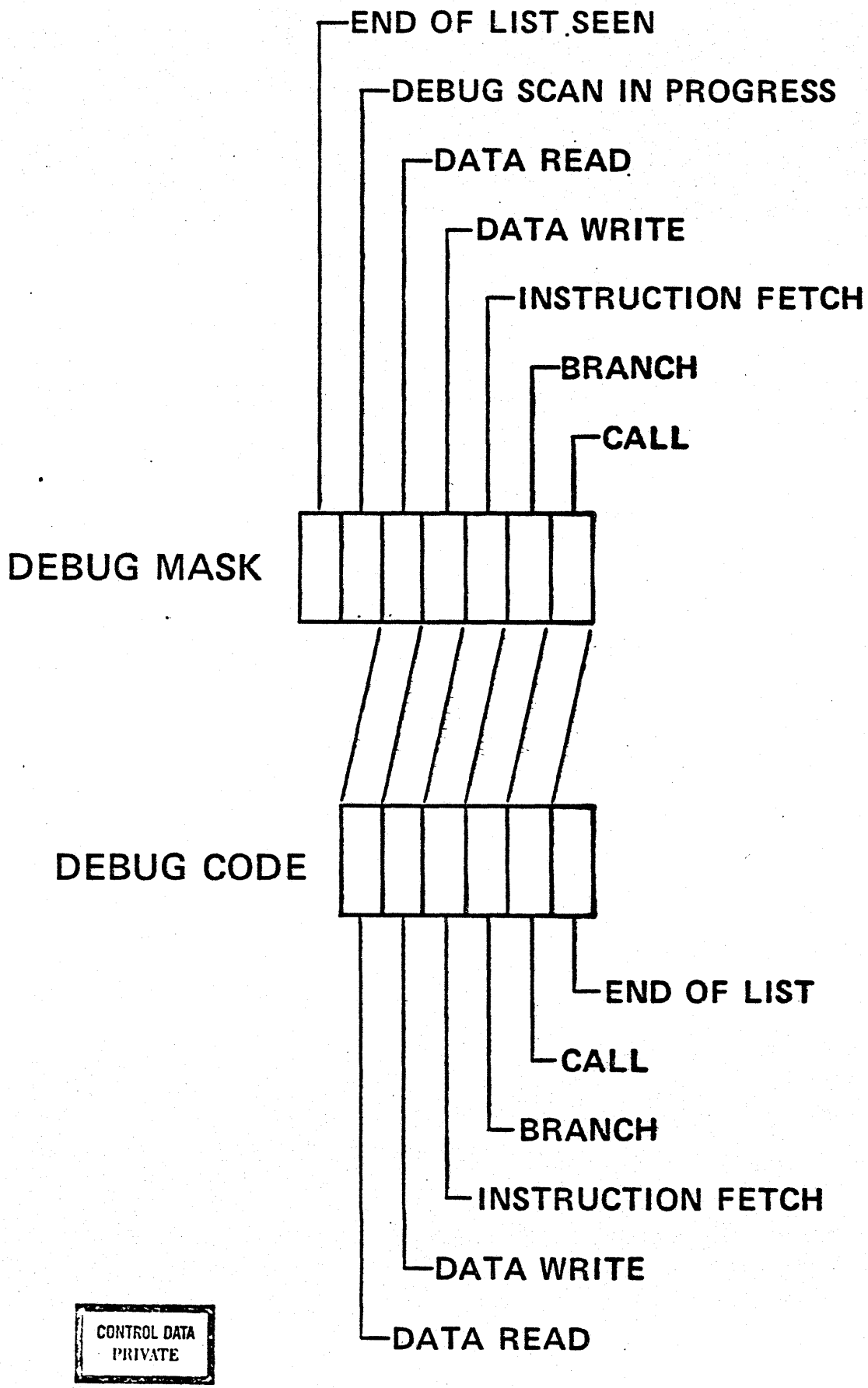
THE DEBUG LIST



DEBUG LIST ENTRIES

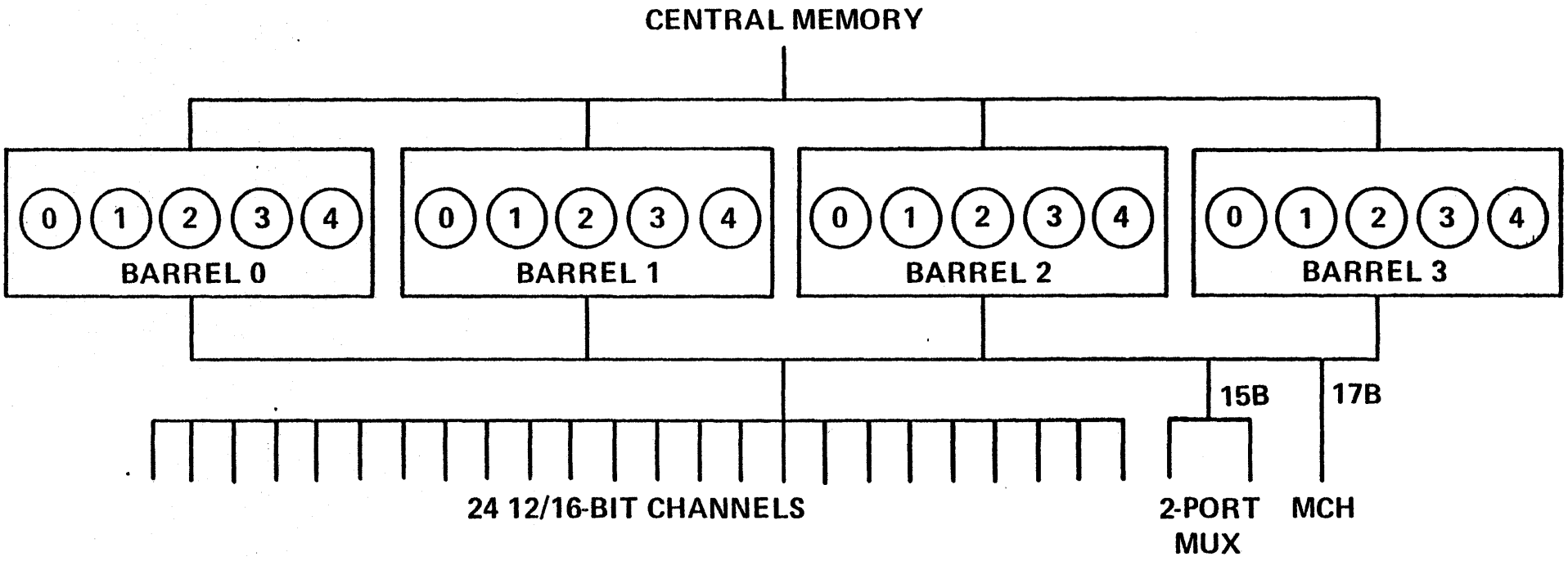


DEBUG CONDITION SELECT

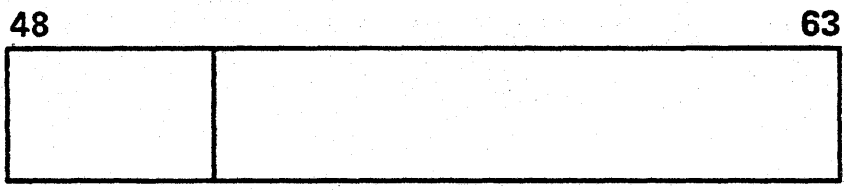


IOU

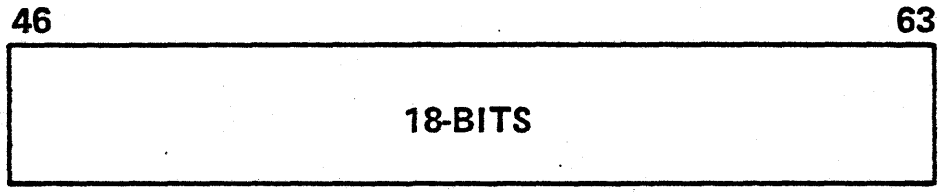
INPUT/OUTPUT UNIT (I2)



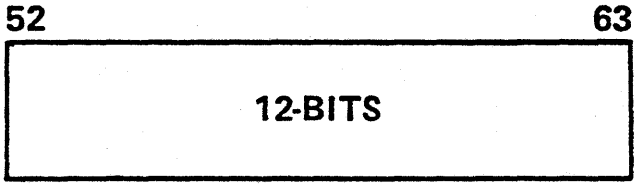
PP REGISTERS



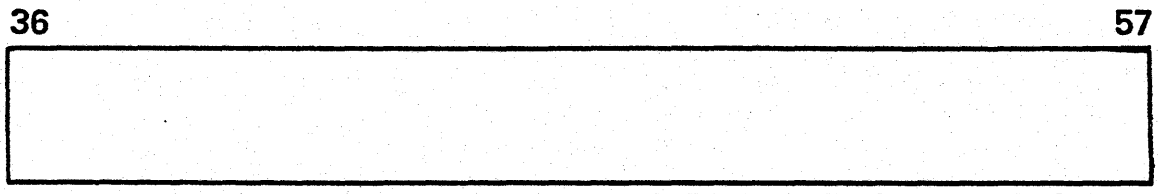
MEMORY WORD (12/16-BITS)



ARITHMETIC REGISTER

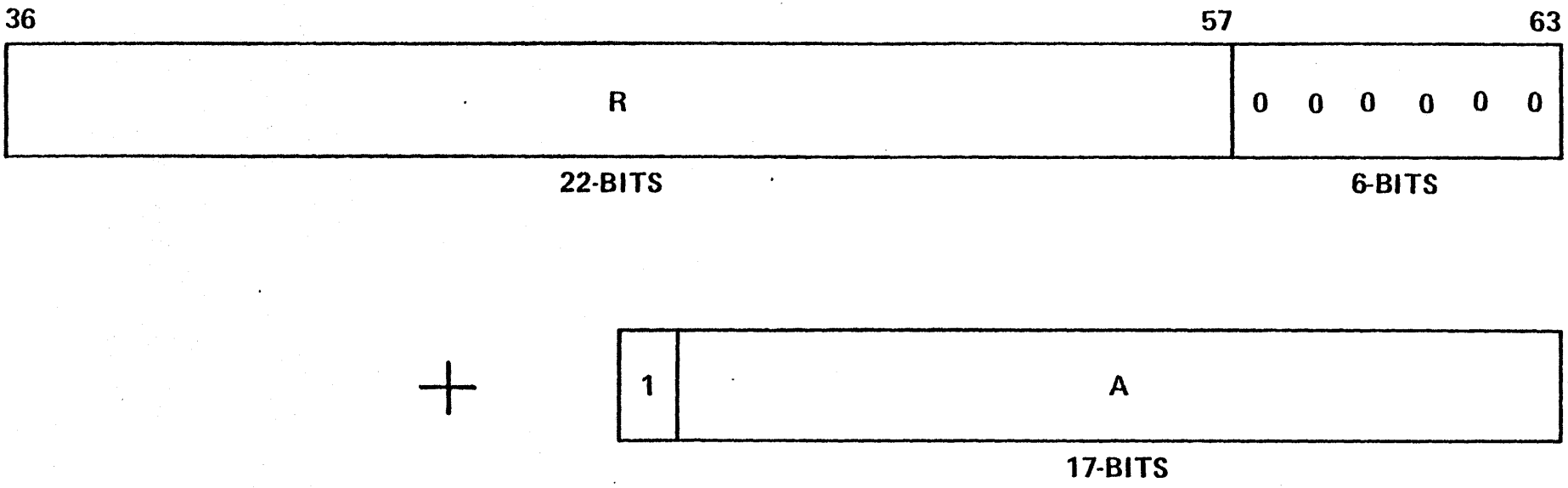


PROGRAM ADDRESS REGISTER

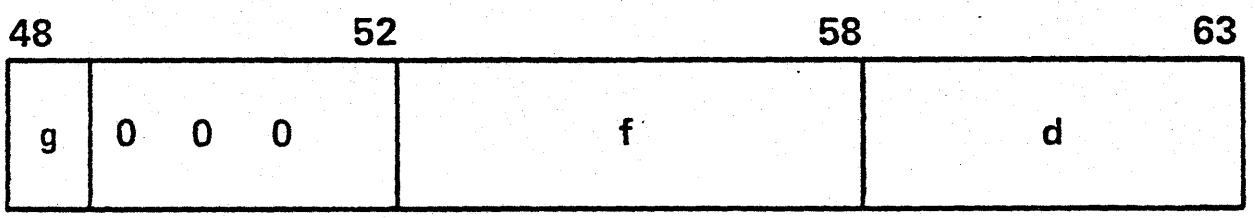


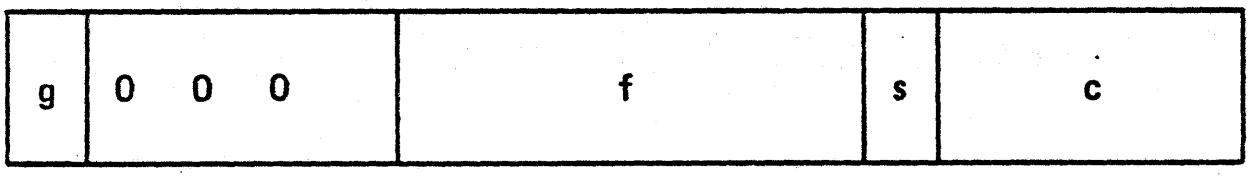
RELOCATION REGISTER

FORMATION OF CENTRAL MEMORY ADDRESSES

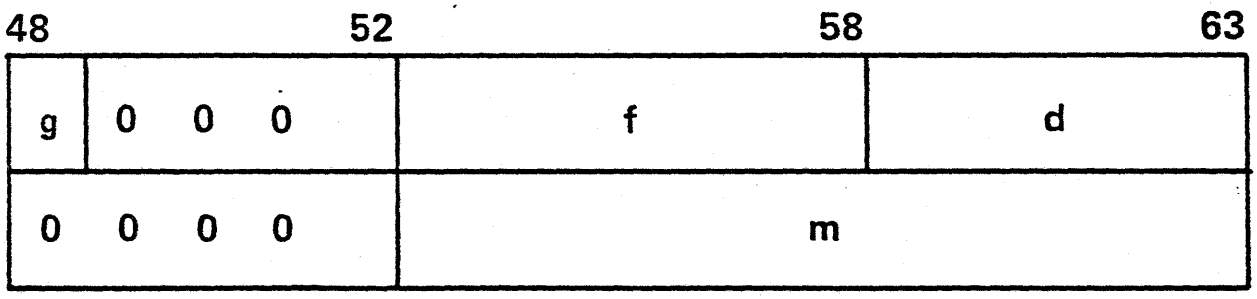


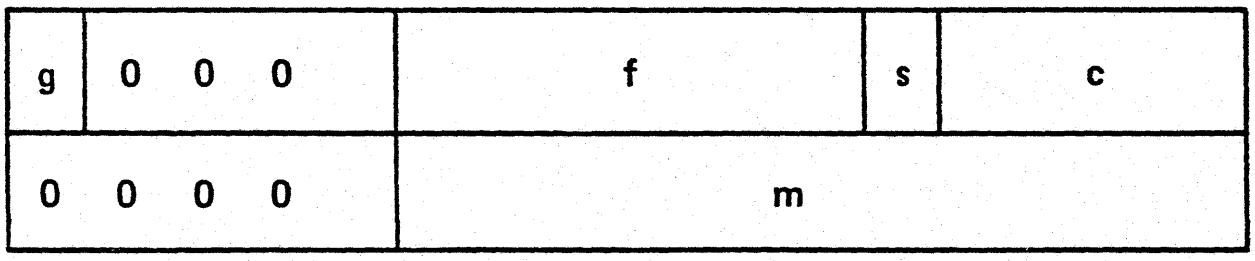
PP INSTRUCTION FORMATS





16-BIT FORMATS





32-BIT FORMATS

PP INSTRUCTIONS

OP CODE	MNEMONIC	NAME	OP CODE	MNEMONIC	NAME
0000	-	PASS	1000d	RDSL	CENTRAL READ AND SET LOCK
0001dm	LJM m,d	LONG JUMP	1001d	RDCL	CENTRAL READ AND CLEAR LOCK
0002dm	RJM m,d	RETURN JUMP	1002	-	PASS
0003d	UJN d	UNCONDITIONAL JUMP	1003	-	PASS
0004d	ZJN d	ZERO JUMP	1004	-	PASS
0005d	NJN d	NON-ZERO JUMP	1005	-	PASS
0006d	PJN d	PLUS JUMP	1006	-	PASS
0007d	MJN d	MINUS JUMP	1007	-	PASS
0010d	SHN d	SHIFT	1010	-	PASS
0011d	LMN d	LOGICAL DIFFERENCE	1011	-	PASS
0012d	LPN d	LOGICAL PRODUCT	1012	-	PASS
0013d	SCN d	SELECTIVE CLEAR	1013	-	PASS
0014d	LDN d	LOAD	1014	-	PASS
0015d	LCN d	LOAD COMPLEMENT	1015	-	PASS
0016d	ADN d	ADD	1016	-	PASS
0017d	SBN d	SUBTRACT	1017	-	PASS
0020dm	LDC m,d	LOAD	1020	-	PASS
0021dm	ADC m,d	ADD	1021	-	PASS
0022dm	LPC m,d	LOGICAL PRODUCT	1022 d	LPDL d	LOGICAL PRODUCT
0023dm	LMC m,d	LOGICAL DIFFERENCE	1023d	LPIL d	LOGICAL PRODUCT
002400	PSN	PASS	1024dm	LPML m,d	LOGICAL PRODUCT
0024d	LRN d	LOAD R			
002500	-	PASS	1025	-	PASS
0025d	SRD d	STORE R			
00260X	EXN	EXCHANGE JUMP	1026d	INPN	INTERRUPT PROCESSOR
00261X	MXN	MONITOR EXCHANGE JUMP			
00262X	MAN	MONITOR EXCHANGE JUMP MA			
0027X	KPT	KEYPOINT	1027	-	PASS

PP INSTRUCTIONS

OP CODE	MNEMONIC	NAME	OP CODE	MNEMONIC	NAME
0030d	LDD d	LOAD	1030d	LDDL d	LOAD
0031d	ADD d	ADD	1031d	ADDL d	ADD
0032d	SBD d	SUBTRACT	1032d	SBDL d	SUBTRACT
0033d	LMD d	LOGICAL DIFFERENCE	1033d	LMDL d	LOGICAL DIFFERENCE
0034d	STD d	STORE	1034d	STDL d	STORE
0035d	RAD d	REPLACE	1035d	RADL d	REPLACE
0036d	ADD d	REPLACE ADD ONE	1036d	ADDL d	REPLACE ADD ONE
0037d	SOD d	REPLACE SUBTRACT ONE	1037d	SODL d	REPLACE SUBTRACT ONE
0040d	LDI d	LOAD	1040d	LDIL d	LOAD
0041d	ADI d	ADD	1041d	ADIL d	ADD
0042d	SDI d	SUBTRACT	1042d	SBIL d	SUBTRACT
0043d	LMI d	LOGICAL DIFFERENCES	1043d	LMIL d	LOGICAL DIFFERENCE
0044d	STI d	STORE	1044d	STIL d	STORE
0045d	RAI d	REPLACE	1045d	RAIL d	REPLACE
0046d	AOI d	REPLACE ADD ONE	1046d	AOIL d	REPLACE ADD ONE
0047d	SOI d	REPLACE SUBTRACT ONE	1047d	SOIL d	REPLACE SUBTRACT ONE
0050dm	LDM m,d	LOAD	1050dm	LDML m,d	LOAD
0051dm	ADM m,d	ADD	1051dm	ADML m,d	ADD
0052dm	SBM m,d	SUBTRACT	1052dm	SBML m,d	SUBTRACT
0053dm	LMM m,d	LOGICAL DIFFERENCE	1053dm	LMML m,d	LOGICAL DIFFERENCE
0054dm	STM m,d	STORE	1054dm	STML m,d	STORE
0055dm	RAM m,d	REPLACE	1055dm	RAML m,d	REPLACE
0056dm	ADM m,d	REPLACE ADD ONE	1056dm	ADML m,d	REPLACE ADD ONE
0057dm	SOM m,d	REPLACE SUBTRACT ONE	1057dm	SOML m,d	REPLACE SUBTRACT ONE
0060d	CRD d	CENTRAL READ TO d	1060d	CRDL d	CENTRAL READ TO d
0061dm	CRM m,d	CENTRAL READ d WORDS	1061dm	CRML m,d	CENTRAL READ d WORDS
0062d	CWD d	CENTRAL WRITE FROM d	1062d	CWDL d	CENTRAL WRITE FROM d
0063dm	CWM m,d	CENTRAL WRITE d WORDS	1063dm	CWML m,d	CENTRAL WRITE d WORDS

PP INSTRUCTIONS

OP CODE	MNEMONIC	NAME	OP CODE	MNEMONIC	NAME
00640cm	AJM m,c	JUMP IF CH. C ACTIVE	1064cm	TSJM m,c	JUMP IF CH. C FLAG SET
00641cm	SCF m,40B+c	TEST AND SET CH. C FLAG			
00650cm	IJM m,c	JUMP IF CH. C INACTIVE	1065xcm	FCJM m,c	JUMP IF CH. C FLAG CLEAR
00651cm	CCF 40B+c	CLEAR CH. C FLAG			
00660cm	FJM m,c	JUMP IF CH. C FULL	1066	-	PASS
00661cm	SFM m,40B+c	JUMP IF CH. C ERROR FLAG SET			
00670cm	EJM m,c	JUMP IF CH. C EMPTY	1067	-	PASS
00671cm	CFM m,40B+c	JUMP IF CH. C ERROR FLAG CLEAR			
00700c	IAN c	INPUT TO A FROM CH. C	1070	-	PASS
00701c	IAN 40B+c	INPUT TO A FROM CH. C			
0071Xcm	IAM m,c	INPUT A WORDS FROM CH. C	1071xcm	IAPM m,c	INPUT A WORDS PACKED FROM CH. C
00720c	OAN	OUTPUT FROM A ON CH. C	1072	-	PASS
00721c	OAN 40B+c	OUTPUT FROM A ON CH. C			
0073xcm	OAM 40B+c	OUTPUT A WORDS ON CH. C	1073xcm	OAPM m,c	OUTPUT A WORDS PACKED ON CH. C
00740c	ACN c	ACTIVATE CH. C	1074	-	PASS
00741c	ACN 40B+c	ACTIVATE CH. C			
00750c	DCN c	DEACTIVATE CH. C	1075	-	PASS
00751c	DCN 40B+c	DEACTIVATE CH. C			
00760c	FAN c	FUNCTION A ON CH. C	1076	-	PASS
00761c	FAN 40B+c	FUNCTION A ON CH. C			
00770cm	FNC m,c	FUNCTION M ON CH. C	1077	-	PASS
00771cm	FNC m,40B+c	FUNCTION M ON CH. C			

SOFTWARE

CYBER 180 SOFTWARE OUTLINE

- ALL PRODUCTS CURRENTLY IN DESIGN OR DEFINITION PHASE

- GENERAL VIEW OF CAPABILITIES THE SYSTEM WILL PROVIDE

- MORE DETAILED VIEW OF THE MECHANISMS INVOLVED IN THE SUPPORT OF THE CYBER 180 VIRTUAL MEMORY ADDRESS MECHANISM

CYBER 180 SOFTWARE

- NOS/180 - NEW OPERATING SYSTEM FOR CYBER 180
- FORTRAN - REIMPLEMENTATION OF FTN 5
- COBOL - DUAL PRODUCT: CONVERTED COBOL 6
- BASIC - NEW DUAL PRODUCT
- SORT/MERGE - NEW DUAL PRODUCT
- DATA BASE - BASED ON EDMS

CYBER 180 SOFTWARE OVERVIEW

USER INTERFACE

COMMAND INTERFACE

- VIEW OF CYBER 180 AS SEEN BY ALL USERS FROM INTERACTIVE OR BATCH JOBS
- "CONTROL CARDS"

PROGRAM INTERFACE

- VIEW OF CYBER 180 AS SEEN BY ALL PROGRAMMERS: SYSTEM, APPLICATION, USER
- "MACROS"

CYBER 180 COMMAND INTERFACE

- BASED ON NOS/170 INTERFACE
- SYSTEM COMMAND LANGUAGE (SCL)
- COMMON SYNTAX, PARAMETERS AND PUNCTUATION RULES ACROSS ALL PROCESSING MODES

CYBER 180 COMMANDS

- SYSTEM ACCESS
  - LOGIN
  - LOGOUT
  - PASSWORD
  - CHARGE
- PERMANENT FILE MANAGEMENT
  - DEFINE
  - ATTACH
  - PURGE
  - GET
  - SAVE
  - REPLACE
  - PERMIT
- FILE MANAGEMENT
  - FILE
  - RETURN
  - UNLOAD
  - REWIND
- RESOURCE MANAGEMENT
  - REQUEST

CYBER 180 COMMANDS CONTINUED

- PROGRAM COMPILATION
  - FTN
  - COBOL
  - BASIC
- PROGRAM EXECUTION
  - LIBRARY
  - EXECUTE
  - "NAME CALL"
- JOB SUBMISSION
  - SUBMIT
  - DROP
- CONTROL COMMANDS
  - IF / IFEND
  - LOOP / LOOPEND
  - FOR / FOREND
- SCL PROCEDURES
  - PROC / PROCEND
  - PARAM
  - "PROCEDURE NAME CALL"
- UTILITIES
  - SOURCE CODE MAINTENANCE
  - OBJECT CODE MAINTENANCE
  - ACCOUNT, PROJECT, MEMBER AND USER ADMINISTRATION

CYBER 180 PROGRAM INTERFACE

- USER PROGRAM INTERFACE
  - FORTRAN
  - COBOL
- SYSTEM PROGRAM INTERFACE
  - PASCAL EXTENDED
  - NOS/180 ACCESSED VIA PASCAL EXTENDED PROCEDURE CALLS : PARAMETERS AND DATA STRUCTURES CONFORM TO PASCAL EXTENDED RULES FOR VARIABLES, CONSTANTS AND TYPES

PASCAL EXTENDED

- FOUR MAJOR DECLARATIONS
  - CONSTANTS
  - VARIABLES
  - PROCEDURES
  - TYPES
- DECLARATIONS MAY BE SPECIFIED GLOBALLY OR WITHIN A PROCEDURE
  - BLOCK STRUCTURE
- PROGRAMMER DEFINED TYPES
  - PERMISSIBLE VALUES

PASCAL EXTENDED BASIC SYNTAX

- COMPILATION UNIT

MODULE<module name>
<declaration>
<declaration>
<declaration>
• • •
MODEND<module name>;

<declaration>: : =<constant declaration>:<variable declaration>:
<procedure declaration>:<type declaration>

- PROCEDURE DECLARATION

PROCEDURE [<attributes>] <procedure name>(<parameter definition>)
<declaration>
<declaration>
<declaration>
• • •
<statement>
<statement>
<statement>
• • •
PROCEND <procedure name>;

<statement> : : =<assignment>:<IF THEN ELSE>:<FOR>:
<WHILE>:<procedure reference>: • • •

TYPE DECLARATIONS

- FIXED TYPES
  - INTEGER
  - CHARACTER
  - ORDINAL
  - BOOLEAN
  - SUBRANGE
  - POINTER TYPE
- STRUCTURED TYPES
  - SETS
  - STRINGS
  - ARRAY
  - RECORD
- STORAGE TYPES
  - SEQUENCES
  - HEAPS
- ADAPTABLE TYPES
  - ADAPTABLE STRING
  - ADAPTABLE ARRAY
  - ADAPTABLE RECORD
  - ADAPTABLE SEQUENCES
  - ADAPTABLE HEAPS

TYPE declarations

```
 5 module type_declarations_example;
 6
 7 type
 8    ordinal_example = (attached, opened, closed, detached);
 9 type
10    record_example = record
11       o:   ordinal_example,
12       i:   integer,
13       b:   boolean,
14    recend;
15 type
16    array_type_example = array [1 .. 10] of record_example;
17
18 {No memory allocated yet}
19
20 var
21    i1:   integer,
22    i2:   integer,
23    b1:   boolean,
24    b2:   boolean,
25    a1:   array_type_example,
26    a2:   array_type_example;
27
28 procedure example;
29    a1[3].o := opened;
30    a2[3].o := attached;
31 procend example;
32
33 modend type_declaration_example;
```

PASCAL_X Declarations

```
         5   module pascal_x_declarations_example;
         6
         7     const
         8       table_size = 100;
         9
        10     var
        11       table: array [1 . . table_size] of table_entry;
        12
        13     type
        14       table_entry = record
        15         file_name: string (10) of char,
        16         file_status: file_status_type,
        17       recend;
        18
        19     type
        20       file_status_type = (attached, opened, closed, detached);
        21
        22     procedure main_procedure;
        23
        24       var
        25         i: 1 . . table_size;
        26
        27       var
        28         k: 0 . . table_size;
        29
        30       procedure nested_procedure;
        31
        32         var
        33           k: 1 . . table_size;
        34
        35         k: = i;
        36       procend nested_procedure;
        37
        38       k : = 0;
        39       for i : = 1 to table_size do
        40         if table [i] . file_status = detached then
        41           table [i] . file_name : = '          ';
        42           k : = k + 1;
        43         ifend;
        44         table [i] . file_status : = opened;
*ERROR*  45         table [i] . file_status : = 1;
        46       forend;
        47       nested_procedure;
*ERROR*  48       k : = 500;
        49     procend main_procedure;
        50   modend pascal_x_declarations_example;
```

LINE NUMBER	SEVERITY LEVEL	ERROR MESSAGE
45	ERROR	Incompatible types are not assignable.
48	ERROR	Value out of range.

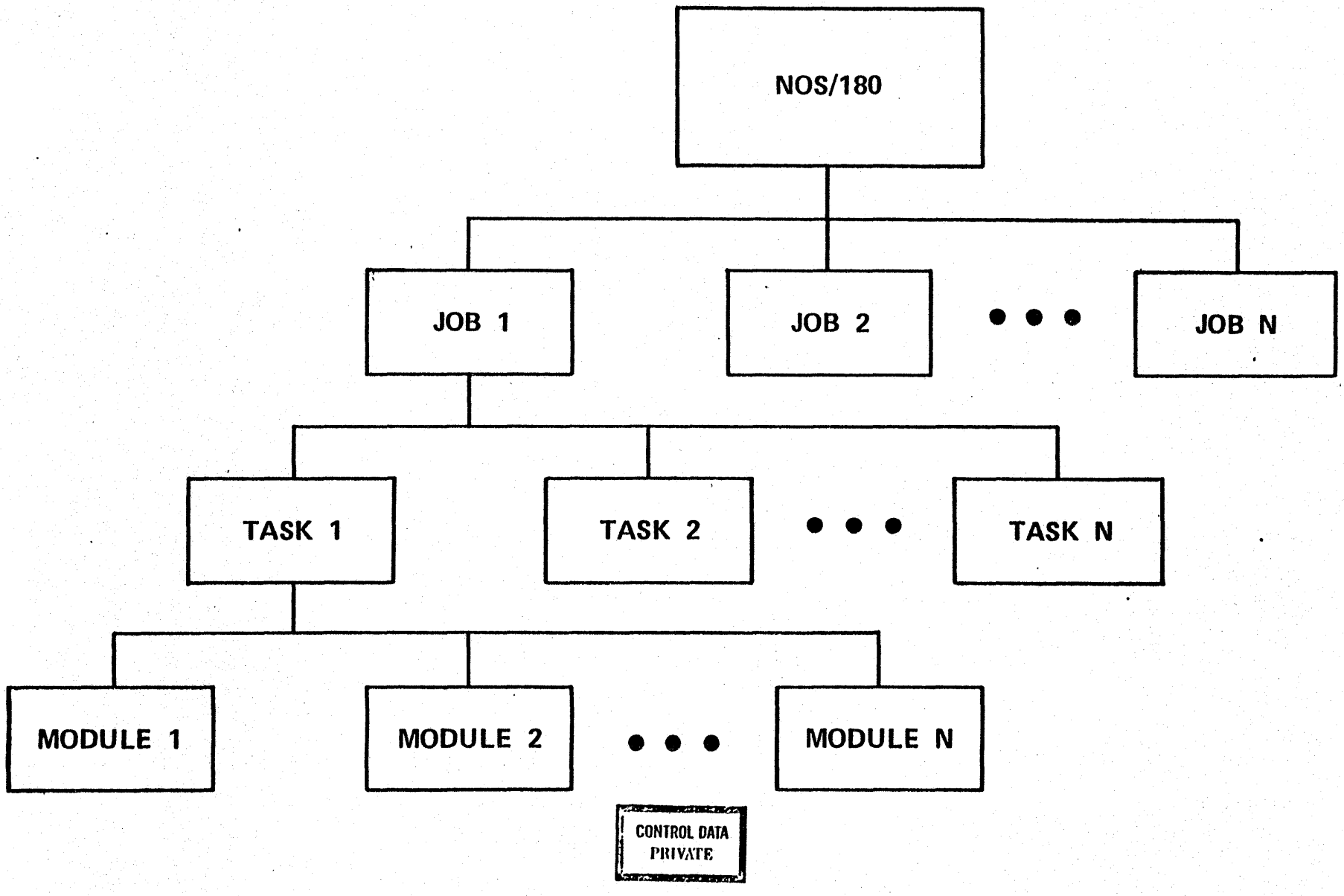
NOS/180 SYSTEM STRUCTURE

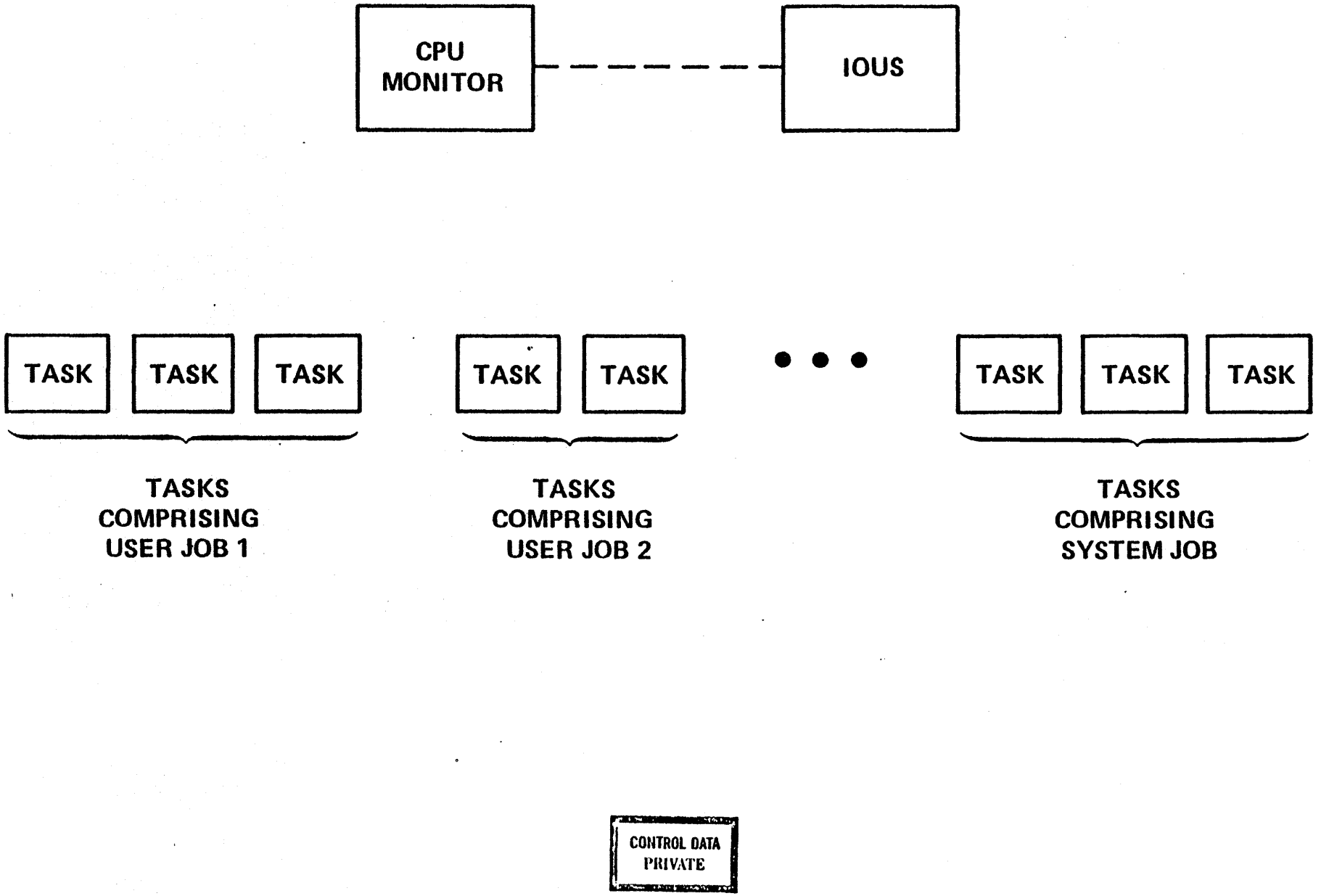
- we will not get Source
- aim: come up with single deadstart tape

- CPU BASED SYSTEM
  - PRODUCTIVITY
  - RELIABILITY
  - MAINTAINABILITY
  - IOU MIGHT NOT BE PRESENT IN SUCCESSOR
- VIRTUAL MEMORY
  - MOST SOFTWARE EXECUTES IN VIRTUAL MEMORY
  - SYSTEM USING ITSELF
- DUAL STATE
  - COEXIST WITH NOS OR NOS/BE
  - INITIAL VERSION ONLY RUNS IN DUAL STATE

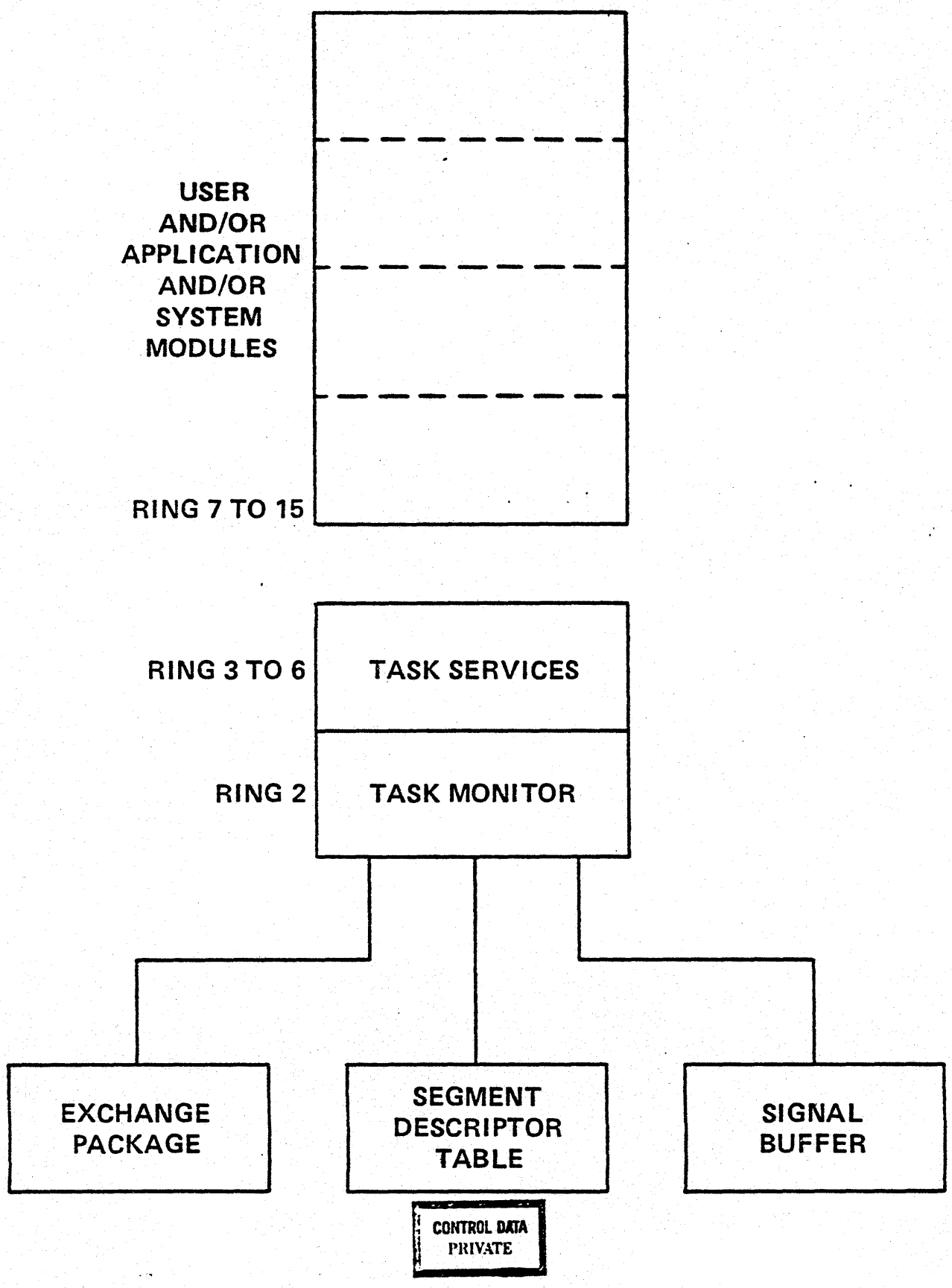
i.e. run NOS with a little bit in C180

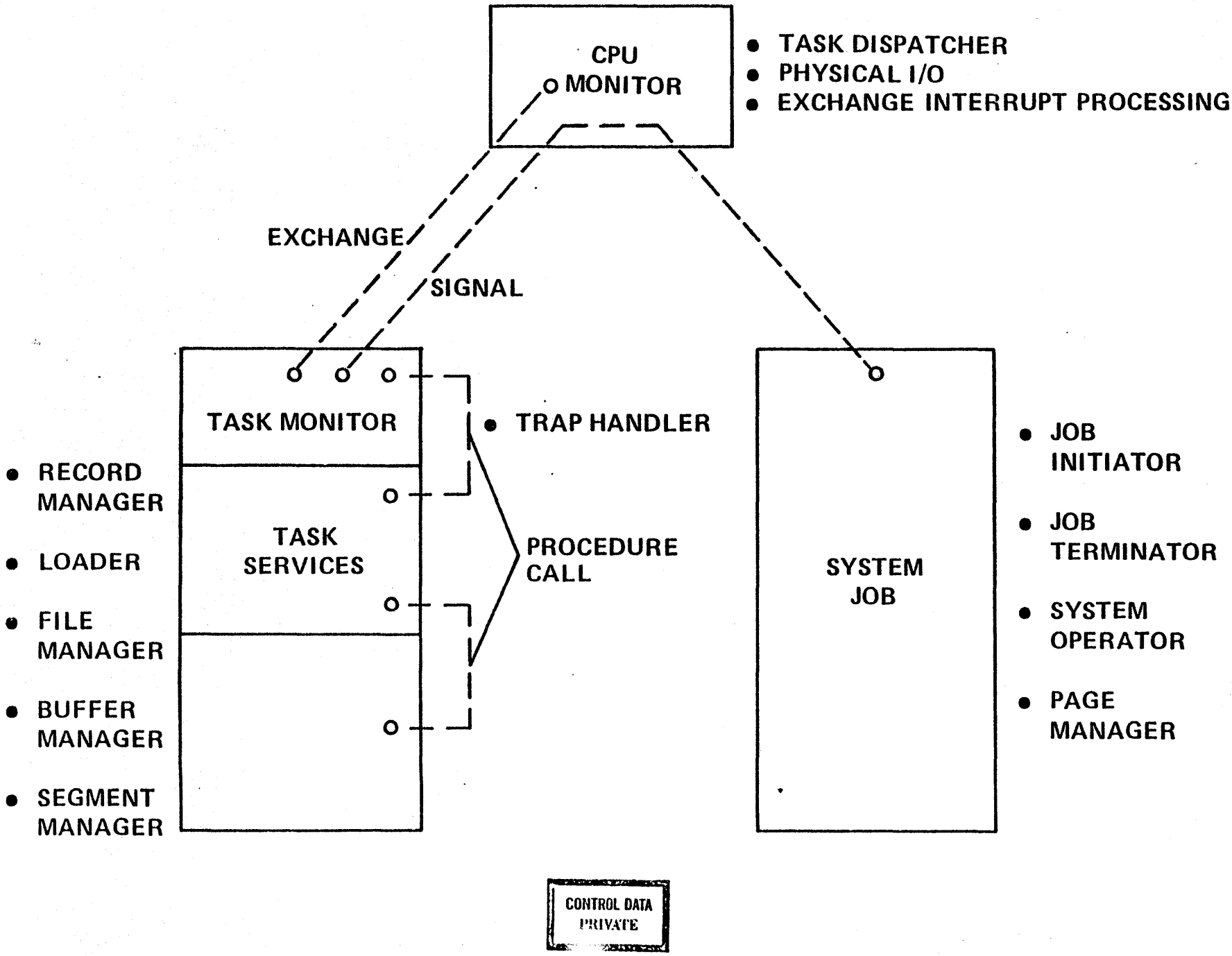
NOS/180 SYSTEM STRUCTURE





TASK ATTRIBUTES AND COMPONENTS




CPU
MONITOR
• TASK DISPATCHER
• PHYSICAL I/O
• EXCHANGE INTERRUPT PROCESSING
EXCHANGE
SIGNAL
TASK MONITOR
• TRAP HANDLER
• RECORD MANAGER
• LOADER
• FILE MANAGER
• BUFFER MANAGER
• SEGMENT MANAGER
TASK SERVICES
PROCEDURE CALL
SYSTEM JOB
• JOB INITIATOR
• JOB TERMINATOR
• SYSTEM OPERATOR
• PAGE MANAGER
CONTROL DATA
PRIVATE

NOS/180 MEMORY MANAGEMENT

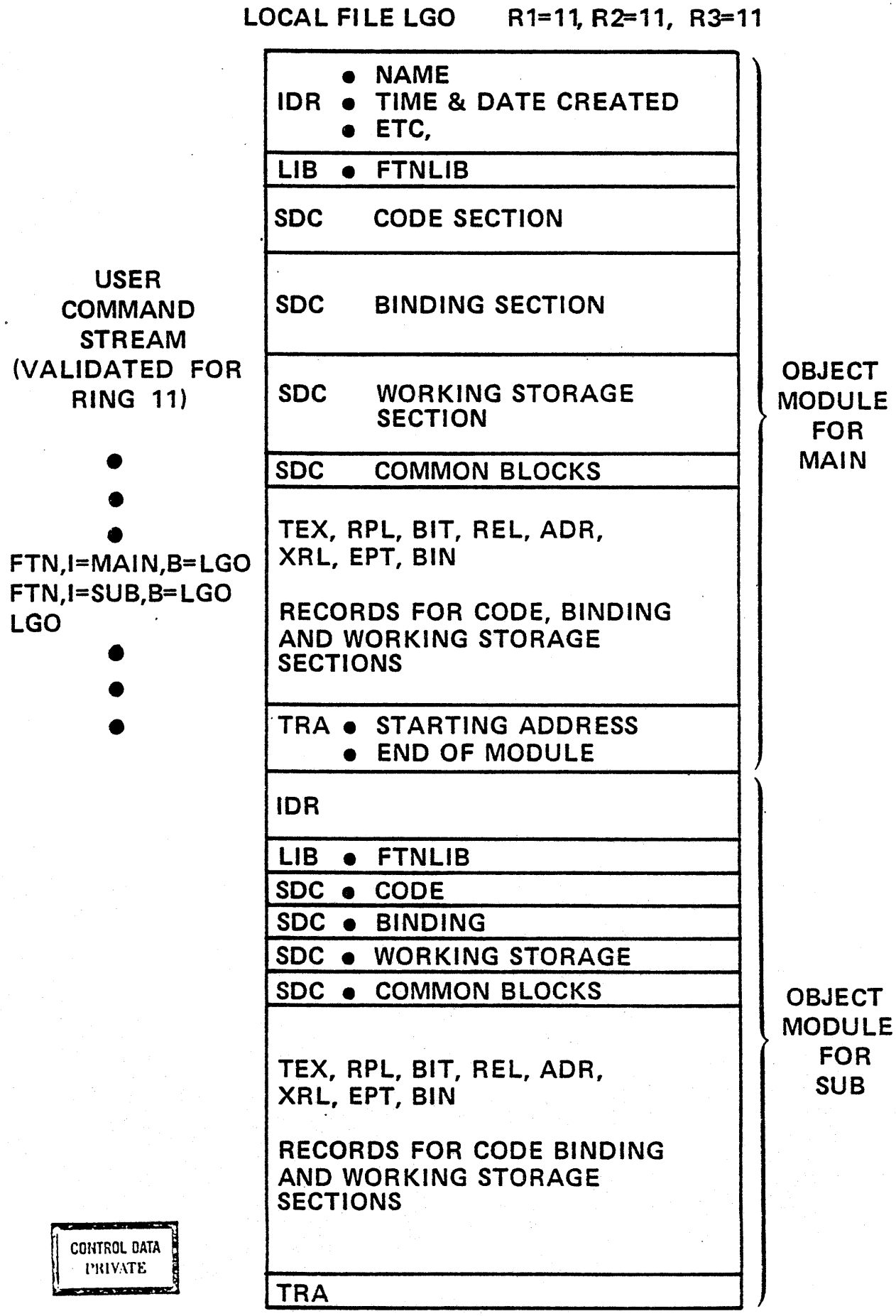
- VIRTUAL MEMORY (SEGMENTS)
  - ALL END USERS VIEW OF MEMORY
  - PRODUCT SET VIEW OF MEMORY
  - SIGNIFICANT PART OF OS VIEW OF MEMORY
- REAL MEMORY (PAGES)
  - LOW LEVEL OS VIEW OF MEMORY
  - LOCALITY OF REFERENCE

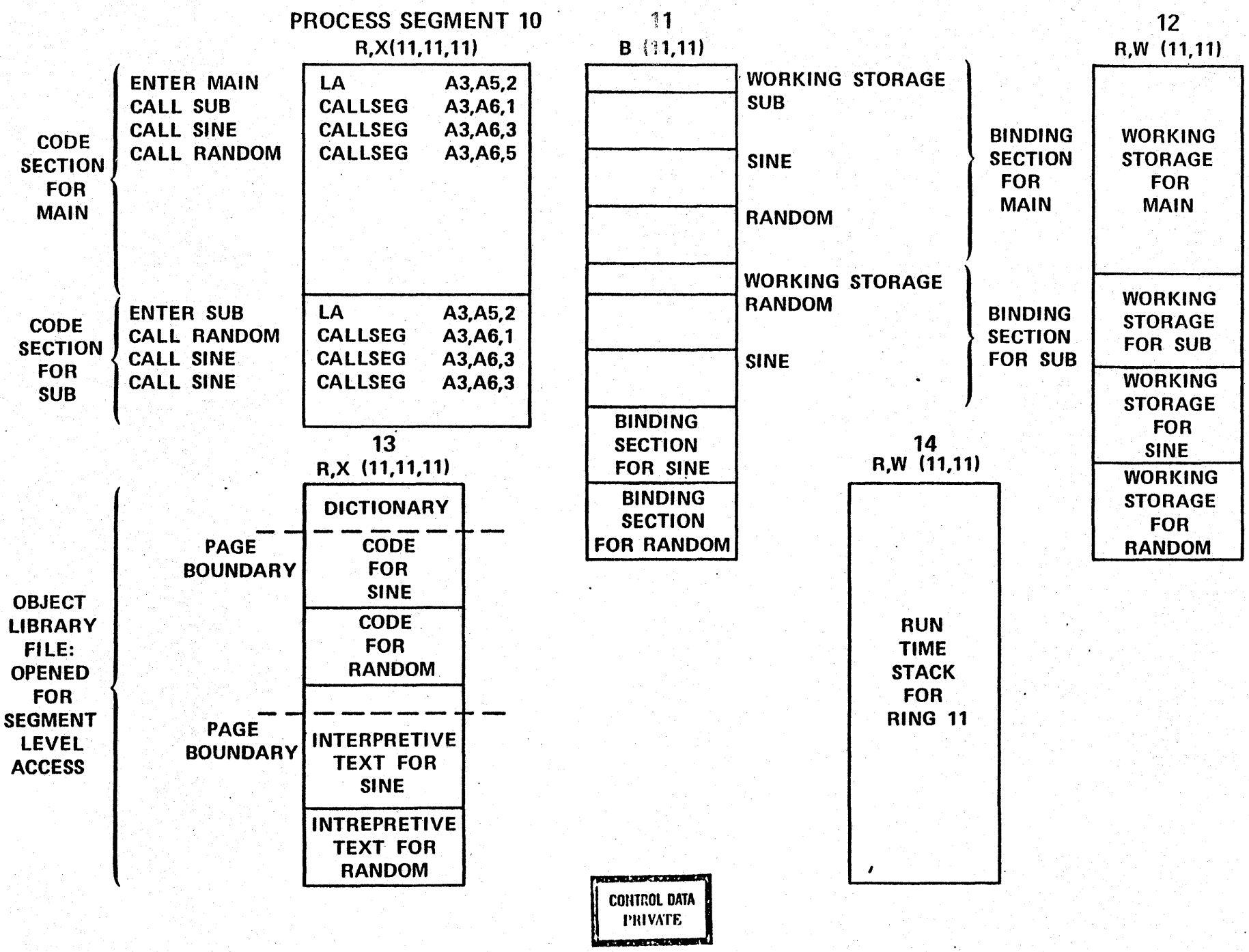
CYBER 180 VIRTUAL MEMORY MANAGEMENT RESPONSIBILITIES

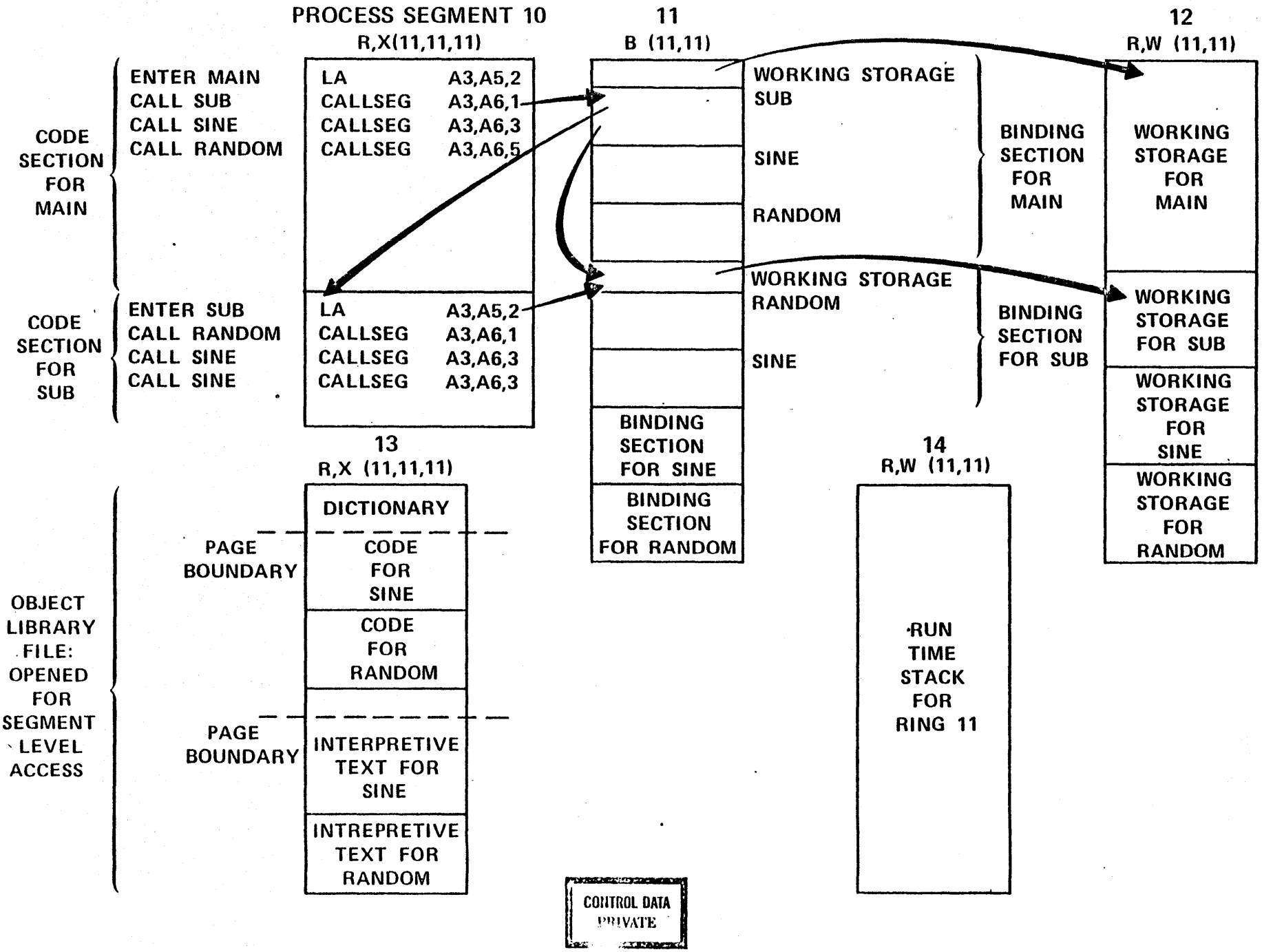
- FILE SYSTEM
  - MAINTAIN RING AND KEY ATTRIBUTES FOR ALL LOCAL AND PERMANENT FILES
  - PROVIDE "SEGMENT LEVEL ACCESS" TO FILES
- SEGMENT MANAGEMENT
  - ADD AND REMOVE SEGMENTS FROM TASK ADDRESS SPACE
  - ASSIGN ASIDS
- COMPILERS
  - GENERATE OBJECT MODULES
  - SEPARATE CODE, BINDING AND WORKING STORAGE
- LOADER
  - LINK MODULES IN VIRTUAL MEMORY
  - ASSIGN PVAS
  - ENFORCE BINDING SEGMENT CONVENTIONS
- OBJECT LIBRARY GENERATOR
  - REFORMAT OBJECT MODULES TO LOAD MODULES
  - CREATE OBJECT LIBRARIES
  - BIND MULTIPLE MODULES INTO A SINGLE MODULE

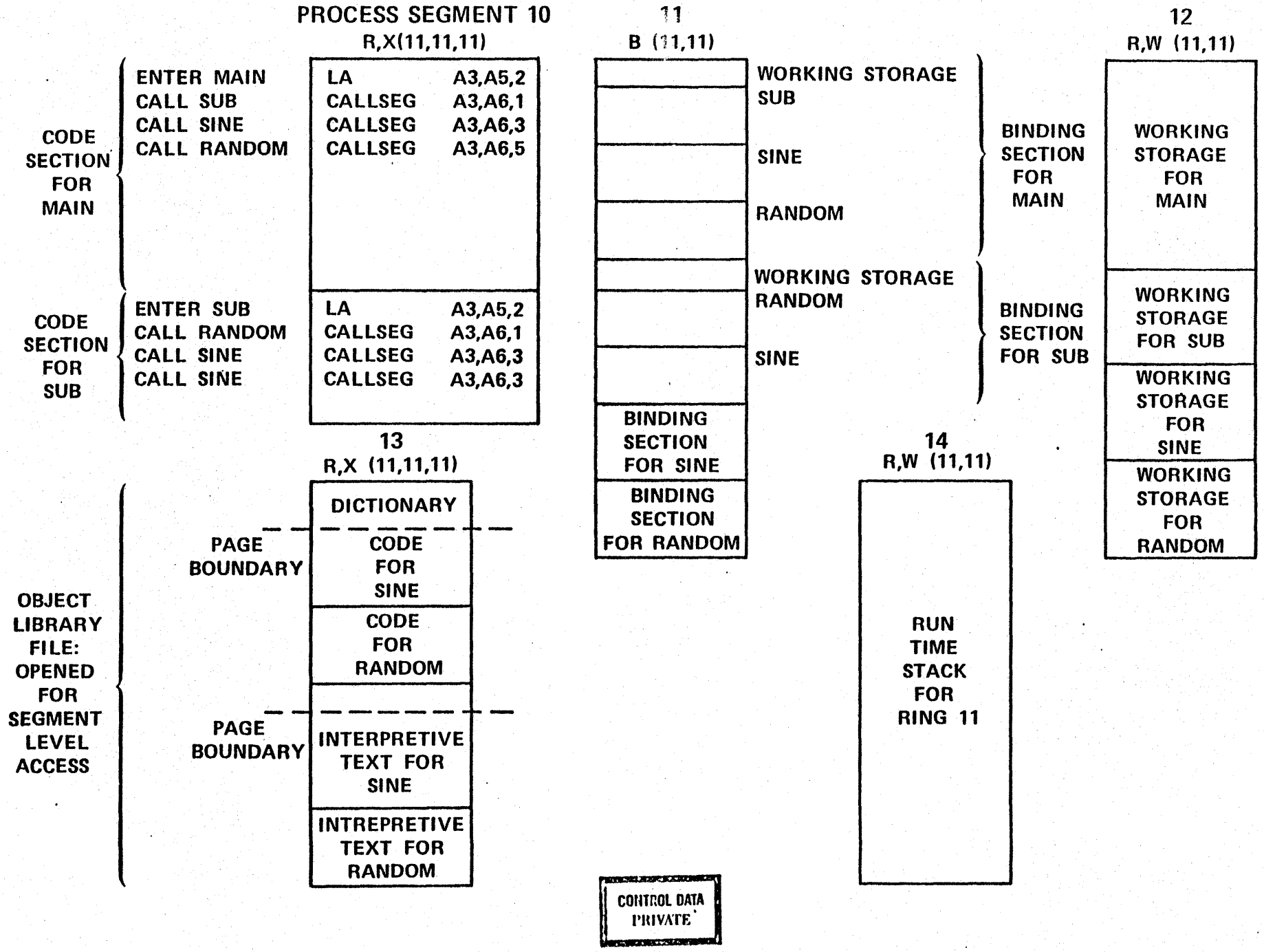
CYBER 180 REAL MEMORY MANAGEMENT RESPONSIBILITIES

- PAGE MANAGEMENT
  - MAINTAIN JOB WORKING SETS
- JOB SCHEDULER
  - SHARE SYSTEM REAL MEMORY AMONG ALL JOBS
- ALL VIRTUAL MEMORY SOFTWARE
  - EXHIBIT GOOD LOCALITY OF REFERENCE



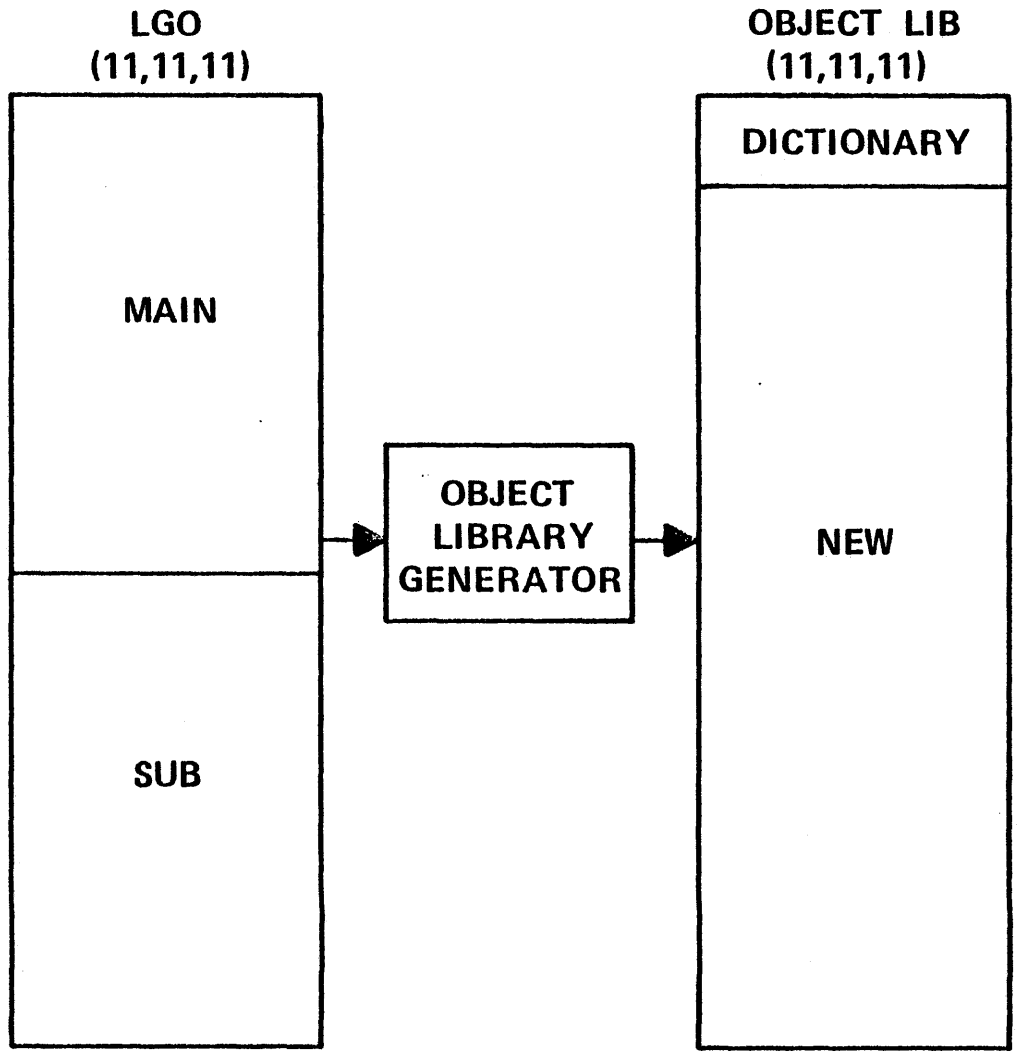

PROCESS SEGMENT 10
R,X(11,11,11)
CODE SECTION FOR MAIN
ENTER MAIN
CALL SUB
CALL SINE
CALL RANDOM
LA A3,A5,2
CALLSEG A3,A6,1
CALLSEG A3,A6,3
CALLSEG A3,A6,5
CODE SECTION FOR SUB
ENTER SUB
CALL RANDOM
CALL SINE
CALL SINE
LA A3,A5,2
CALLSEG A3,A6,1
CALLSEG A3,A6,3
CALLSEG A3,A6,3
OBJECT LIBRARY FILE: OPENED FOR SEGMENT LEVEL ACCESS
13
R,X (11,11,11)
DICTIONARY
PAGE BOUNDARY
CODE FOR SINE
CODE FOR RANDOM
PAGE BOUNDARY
INTERPRETIVE TEXT FOR SINE
INTREPRETIVE TEXT FOR RANDOM
11
B (11,11)
WORKING STORAGE SUB
SINE
RANDOM
WORKING STORAGE RANDOM
SINE
BINDING SECTION FOR SINE
BINDING SECTION FOR RANDOM
BINDING SECTION FOR MAIN
BINDING SECTION FOR SUB
14
R,W (11,11)
RUN TIME STACK FOR RING 11
12
R,W (11,11)
WORKING STORAGE FOR MAIN
WORKING STORAGE FOR SUB
WORKING STORAGE FOR SINE
WORKING STORAGE FOR RANDOM
CONTROL DATA PRIVATE


PROCESS SEGMENT 10
R,X(11,11,11)
11
B (11,11)
12
R,W (11,11)
CODE SECTION FOR MAIN
ENTER MAIN
CALL SUB
CALL SINE
CALL RANDOM
LA A3,A5,2
CALLSEG A3,A6,1
CALLSEG A3,A6,3
CALLSEG A3,A6,5
WORKING STORAGE SUB
SINE
RANDOM
BINDING SECTION FOR MAIN
WORKING STORAGE FOR MAIN
CODE SECTION FOR SUB
ENTER SUB
CALL RANDOM
CALL SINE
CALL SINE
LA A3,A5,2
CALLSEG A3,A6,1
CALLSEG A3,A6,3
CALLSEG A3,A6,3
WORKING STORAGE RANDOM
SINE
BINDING SECTION FOR SUB
WORKING STORAGE FOR SUB
BINDING SECTION FOR SINE
WORKING STORAGE FOR SINE
BINDING SECTION FOR RANDOM
WORKING STORAGE FOR RANDOM
13
R,X (11,11,11)
14
R,W (11,11)
OBJECT LIBRARY FILE: OPENED FOR SEGMENT LEVEL ACCESS
DICTIONARY
PAGE BOUNDARY
CODE FOR SINE
CODE FOR RANDOM
PAGE BOUNDARY
INTERPRETIVE TEXT FOR SINE
INTREPRETIVE TEXT FOR RANDOM
RUN TIME STACK FOR RING 11
CONTROL DATA PRIVATE

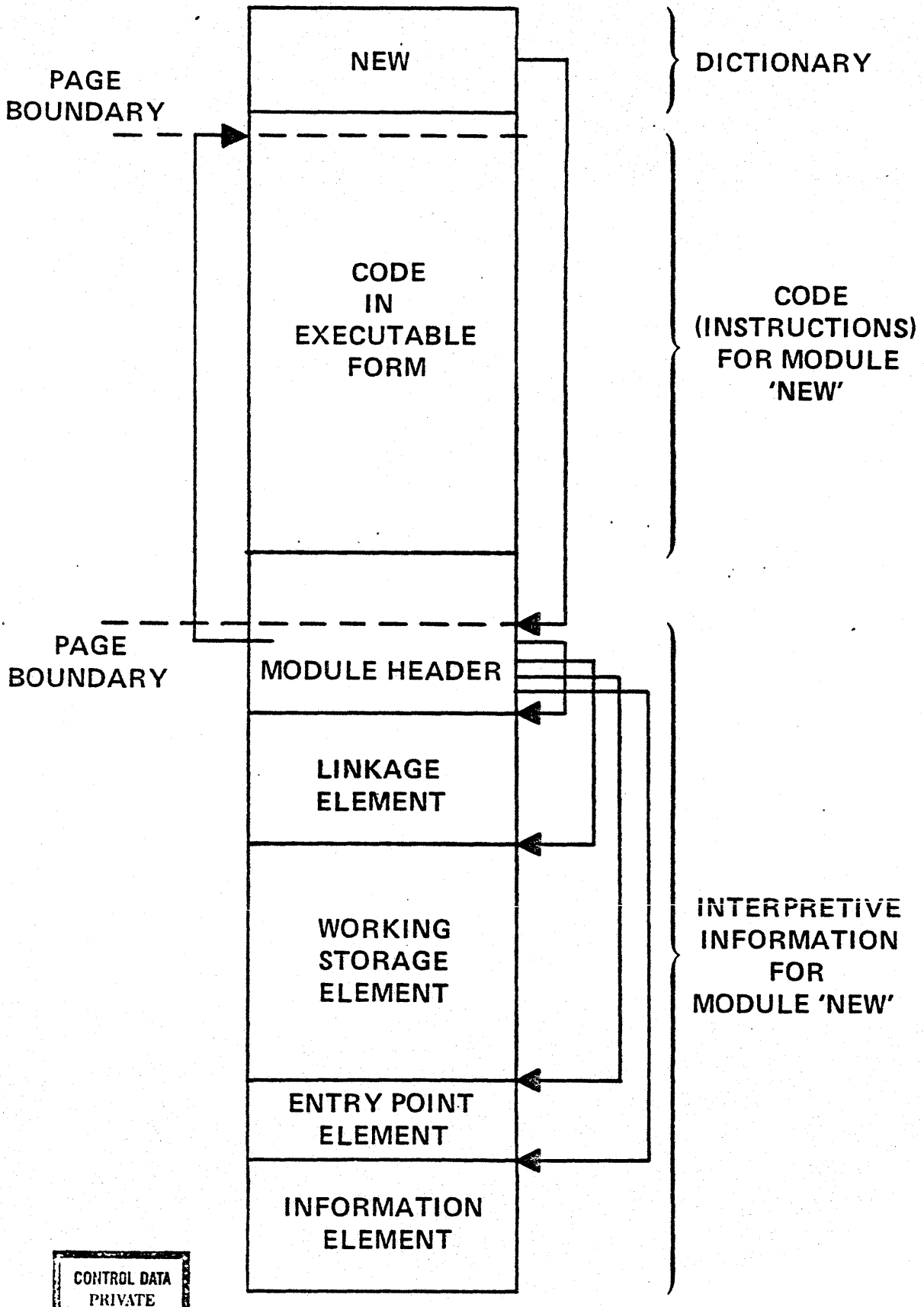

PROCESS SEGMENT 10
R,X(11,11,11)
CODE SECTION FOR MAIN
ENTER MAIN
CALL SUB
CALL SINE
CALL RANDOM
LA A3,A5,2
CALLSEG A3,A6,1
CALLSEG A3,A6,3
CALLSEG A3,A6,5
CODE SECTION FOR SUB
ENTER SUB
CALL RANDOM
CALL SINE
CALL SINE
LA A3,A5,2
CALLSEG A3,A6,1
CALLSEG A3,A6,3
CALLSEG A3,A6,3
OBJECT LIBRARY FILE: OPENED FOR SEGMENT LEVEL ACCESS
13
R,X (11,11,11)
DICTIONARY
PAGE BOUNDARY
CODE FOR SINE
CODE FOR RANDOM
PAGE BOUNDARY
INTERPRETIVE TEXT FOR SINE
INTREPRETIVE TEXT FOR RANDOM
11
B (11,11)
WORKING STORAGE SUB
SINE
RANDOM
WORKING STORAGE RANDOM
SINE
BINDING SECTION FOR SINE
BINDING SECTION FOR RANDOM
BINDING SECTION FOR MAIN
BINDING SECTION FOR SUB
12
R,W (11,11)
WORKING STORAGE FOR MAIN
WORKING STORAGE FOR SUB
WORKING STORAGE FOR SINE
WORKING STORAGE FOR RANDOM
14
R,W (11,11)
RUN TIME STACK FOR RING 11
CONTROL DATA PRIVATE

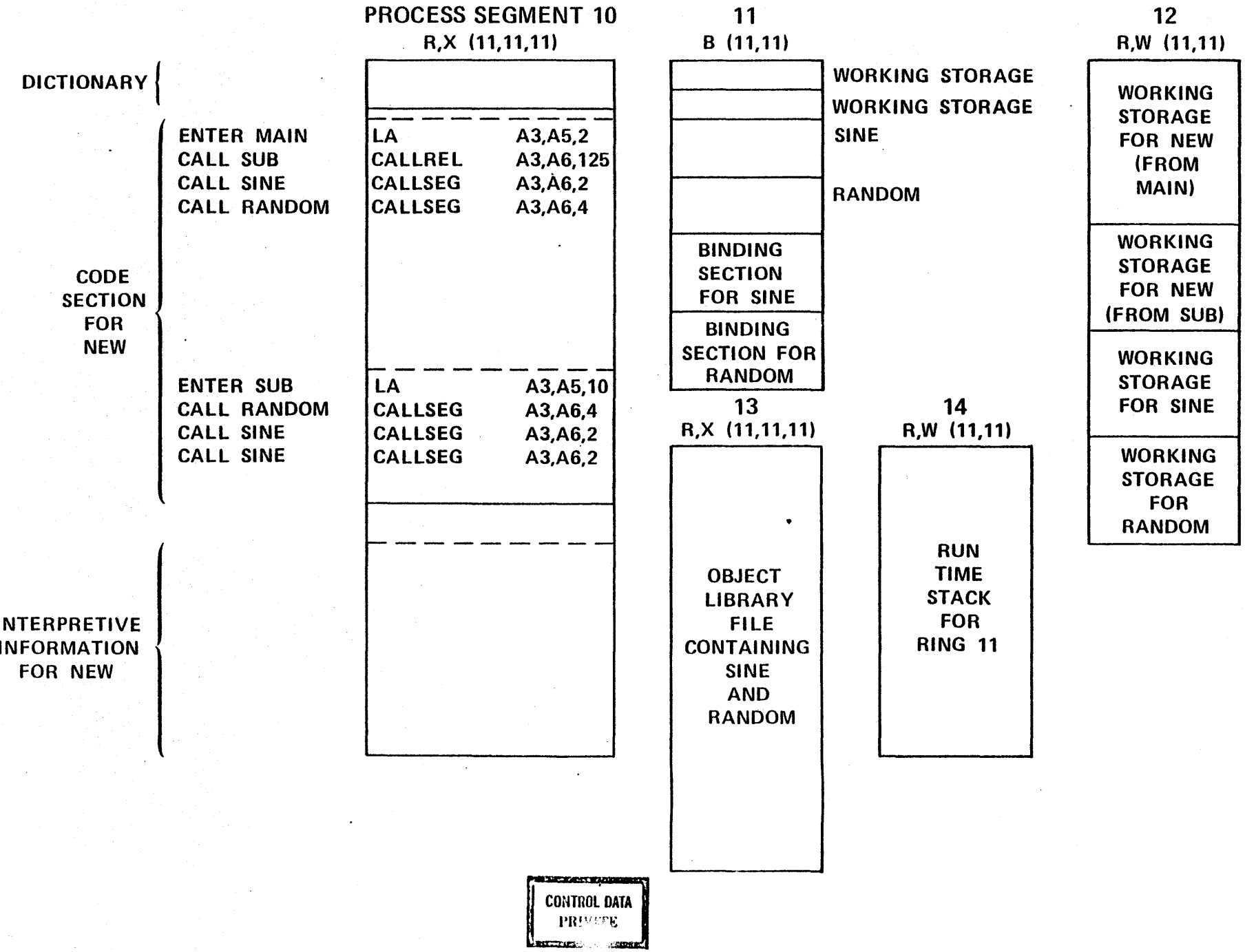
OBJECT LIBRARY GENERATION: MODULE BINDING

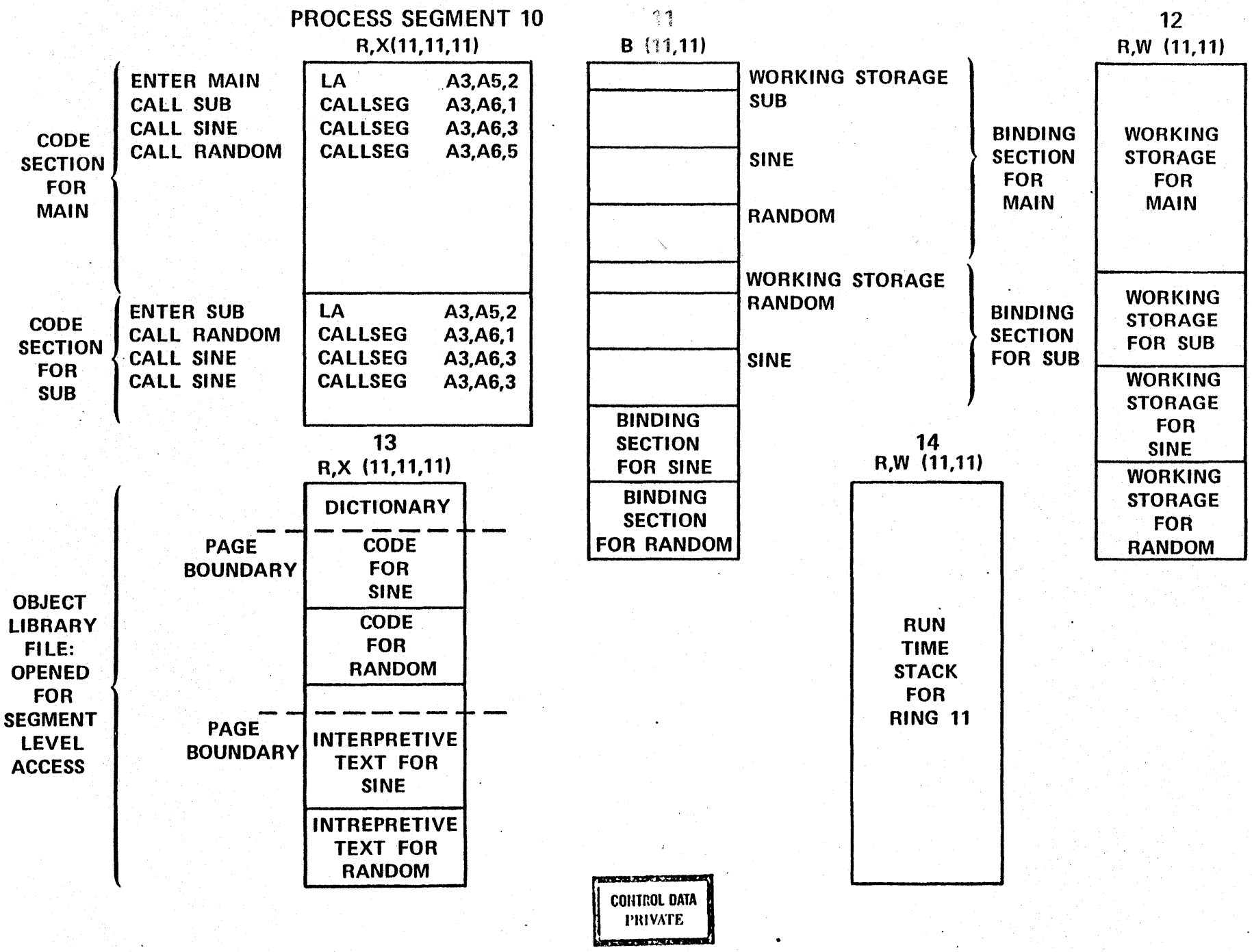
```
USER COMMANDS
        .
        .
        .
CREATE_OBJECT_LIBRARY
   CREATE_MODULE NEW (LGO,MAIN,SUB)
   GENERATE OBJECT_LIB
   END
DEFINE OBJECT_LIB
PERMIT OBJECT_LIB,USER=JAW,READ
```

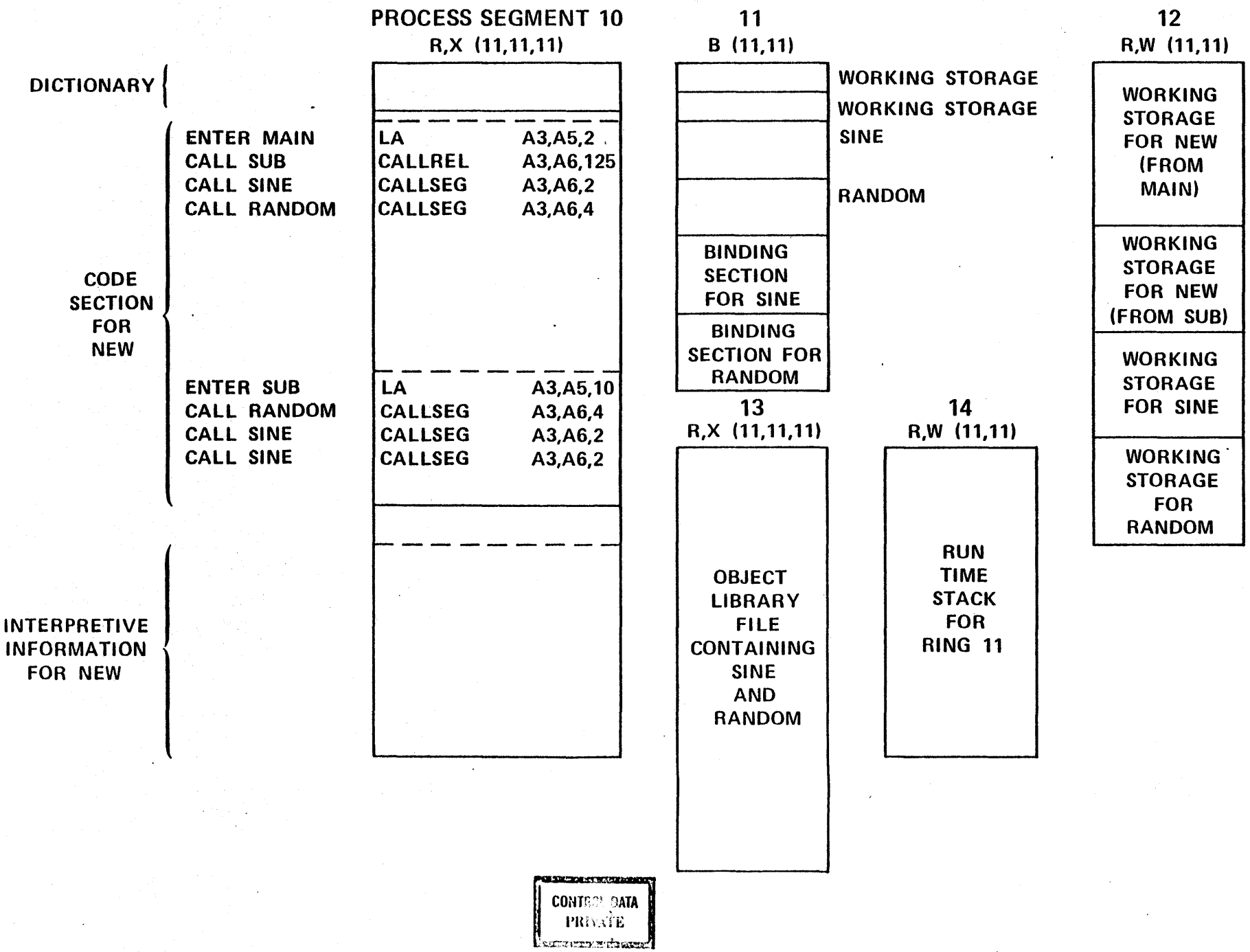


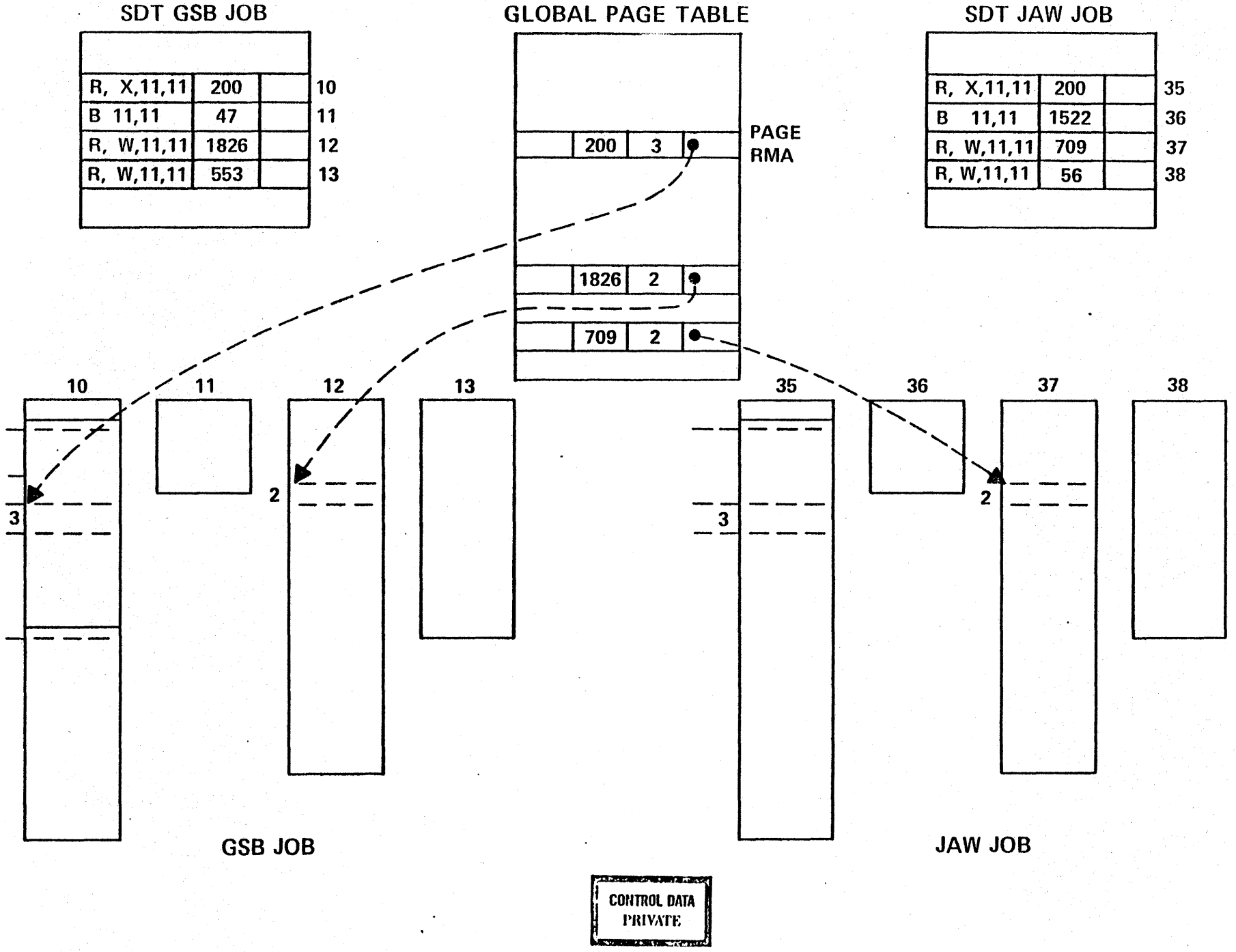
OBJECT LIBRARY FORMAT
"SEGMENT LEVEL ACCESS FILE"




PROCESS SEGMENT 10
R,X (11,11,11)
11
B (11,11)
12
R,W (11,11)
DICTIONARY
CODE SECTION FOR NEW
INTERPRETIVE INFORMATION FOR NEW
ENTER MAIN
CALL SUB
CALL SINE
CALL RANDOM
LA A3,A5,2
CALLREL A3,A6,125
CALLSEG A3,A6,2
CALLSEG A3,A6,4
ENTER SUB
CALL RANDOM
CALL SINE
CALL SINE
LA A3,A5,10
CALLSEG A3,A6,4
CALLSEG A3,A6,2
CALLSEG A3,A6,2
WORKING STORAGE
WORKING STORAGE
SINE
RANDOM
BINDING SECTION FOR SINE
BINDING SECTION FOR RANDOM
WORKING STORAGE FOR NEW (FROM MAIN)
WORKING STORAGE FOR NEW (FROM SUB)
WORKING STORAGE FOR SINE
WORKING STORAGE FOR RANDOM
13
R,X (11,11,11)
14
R,W (11,11)
OBJECT LIBRARY FILE CONTAINING SINE AND RANDOM
RUN TIME STACK FOR RING 11
CONTROL DATA
PRIVATE


PROCESS SEGMENT 10
R,X(11,11,11)
ENTER MAIN
CALL SUB
CALL SINE
CALL RANDOM
LA A3,A5,2
CALLSEG A3,A6,1
CALLSEG A3,A6,3
CALLSEG A3,A6,5
CODE SECTION FOR MAIN
ENTER SUB
CALL RANDOM
CALL SINE
CALL SINE
LA A3,A5,2
CALLSEG A3,A6,1
CALLSEG A3,A6,3
CALLSEG A3,A6,3
CODE SECTION FOR SUB
11
B (11,11)
WORKING STORAGE SUB
SINE
RANDOM
WORKING STORAGE RANDOM
SINE
BINDING SECTION FOR SINE
BINDING SECTION FOR RANDOM
BINDING SECTION FOR MAIN
BINDING SECTION FOR SUB
12
R,W (11,11)
WORKING STORAGE FOR MAIN
WORKING STORAGE FOR SUB
WORKING STORAGE FOR SINE
WORKING STORAGE FOR RANDOM
13
R,X (11,11,11)
DICTIONARY
PAGE BOUNDARY
CODE FOR SINE
CODE FOR RANDOM
PAGE BOUNDARY
INTERPRETIVE TEXT FOR SINE
INTREPRETIVE TEXT FOR RANDOM
OBJECT LIBRARY FILE: OPENED FOR SEGMENT LEVEL ACCESS
14
R,W (11,11)
RUN TIME STACK FOR RING 11
CONTROL DATA PRIVATE





NOS/180 ACCESS CONTROL

- ALL USERS OF THE SYSTEM MUST BE VALIDATED PRIOR TO GAINING ACCESS TO THE SYSTEM
- IDENTITY MUST BE VERIFIED BASED ON VALIDATION INFORMATION ON EVERY LOGIN OR BATCH JOB SUBMISSION
- ALL ACCESSES BY ALL USERS (SUBJECTS) TO ANY SYSTEM RESOURCE (OBJECTS) GOVERNED BY A "CONCEPTUAL ACCESS CONTROL MATRIX."

	SUBJECT A	SUBJECT B	FILE C	FILE D	TAPE DRIVE E
SUBJECT A		ADMIN.	OWNER R,W		OWNER USE
SUBJECT B			R	OWNER R,X	

- A VARIETY OF FEATURES OF THE SYSTEM ARCHITECTURE INTERACT TO IMPLEMENT THE CONCEPTUAL ACCESS CONTROL MATRIX

ACCESS CONTROL APPLIED TO CYBER 180 VIRTUAL MEMORY

- VALIDATION INFORMATION FOR EVERY USER INCLUDES LOWEST RING NUMBER OF EXECUTION
- FILE SYSTEM MAINTAINS FOUR RING BRACKETS FOR EVERY FILE: READ, WRITE, EXECUTE AND CALL
- ALL FILES CREATED BY THE USER ACQUIRE THE USER'S LEVEL OF PRIVILEGE (RING NUMBER)
- FILE SYSTEM ALLOWS "SEGMENT LEVEL" ACCESS TO FILES
- THE LOADER USES SEGMENT LEVEL ACCESS FOR OBJECT LIBRARIES
- THE LOADER USES SEGMENT MANAGEMENT TO ALLOCATE VIRTUAL MEMORY FOR DATA AREAS
- USERS CAN USE SEGMENT LEVEL ACCESS FOR THEIR OWN DATA FILES

78-06-28

CDC CYBER 180 ABBREVIATIONS

A register	Address register
AO/R	Architectural Objectives/Requirements
ASID	Active segment identifier
BC	Base constant
BCO	Branch on condition
BCR	Branch on condition register
BCT	Between command test
BDP	Business data processing
BN	Byte number
BS	Binding section
CBP	Code base pointer
CCD	Charge-coupled devices
CEL	Corrected error log
CF	Critical frame
CFF	Critical frame flag
CM	Central memory
CMA	Central memory access
CMU	Compare-move unit.
CPU	Central processing unit
CRT	Cathode ray tube
CSF	Current stack frame pointer
DAP	Design action paper
DC	Debug code
DEC	Model-dependent environment control (see EC)
DI	Debug index
DLP	Debug list pointer
DM	Debug mask
DMR	Debug mask register
DSP	Dynamic space pointer
EBAM	Electron beam accessed memory
DSP	Dynamic space pointer
EC	Environment control
ECC	Error correction code
ECL	Emitter-coupled logic
ECM	Extended central memory
ECS	Extended core storage
EDMS	European data managment system
EID	Element identifier
EM	Exit mode
EP	External procedure flag
ES	End suppression toggle
FL	Field length
FLC	Central memory field length register
FLE	Extended core storage field length register
FTN	Fortran
GK	Global key
GL	Global lock
ILH	Instruction look-ahead (P2)
I/O	Input/output
IOU	Input/output unit
JEP	Job mode exchange package

JPS	Job process state pointer
KC	Keypoint code
KCN	Keypoint class number
KEF	Keypoint enable flag
KM	Keypoint mask
LED	Light emitting diode
LK	Local key
LPID	Last processor identification
LRN	Largest ring number
LRU	Least recently used
LSI	Large-scale integration
MA	Monitor address
MAC	Maintenance access control
MCH	Maintenance channel
MCI	Maintenance channel interface
MCR	Monitor condition register
MCU	Maintenance control unit
MDF	Model-dependent flags
MDW	Model-dependent word
MEP	Monitor mode exchange package
MF	Monitor flag
MID	Maintenance identifier
MIGDS	Model-independent general design specification
MM	Monitor mask
MOP	Micro-operator
MOS	Metal-oxide semiconductor

78-06-28

MPS	Monitor process state pointer
MTR	Monitor
MUX	Multiplexer
M1...Mn	Central memory 1...central memory n
NOS/180	Network Operating System/CDC CYBER 180
NS	Negative sign toggle
OCF	On-condition flag
OI	Options installed
ON	Occurrence number
OP	Operation code
P register	Program address register
PCO	Printed circuit operation
PFA	Page frame address
PFS	Processor fault status
PID	Processor identifier
PIT	Process interval timer
PMF	Performance monitoring flag
PN	Page number
PO	Page offset
PP	Peripheral processor
PPM	Peripheral processor memory
PPS	Peripheral processor subsystem
PPU	Peripheral processor unit
PSA	Previous save area
PSF	Previous stack frame
PSM	Page size mask
PSR	Process state registers

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
RAM	Random access memory
RAM	Reliability, availability, maintainability
RMA	Real memory address
RMS	Rotating mass storage
RN	Ring number
ROM	Read-only memory
RP	Read access control (segment descriptor field)
SCL	System command language
SCT	Special characters table
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
SM	The symbol
SPID	Segment page identifier
SPT	System page table
SN	Negative sign
SRT	Subscript range table
SS	Status summary
SSP	Subsystem procedure
STA	Segment table address
STL	Segment table length
SV	Specification value
SVA	System virtual address
SWL	Software writers language
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
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 capibility list
VMID	Virtual machine identifier
WP	Write access control (segment descriptor field)
XP	Execute access control (segment descriptor field)
ZF	Zero field toggle

1, pgp for Debug

recover from 1 caller as much as possible but caller 2, 3 not interested.

2. dyn Ch_n

CYBER 180 OVERVIEW

COURSE OUTLINE

*****

I.CYBER 180 PRODUCT OVERVIEW

- A.HARDWARE
- B.SOFTWARE
- C.MARKETING

II.NEW CONCEPTS OF CYBER 180 HARDWARE

- A.VIRTUAL MEMORY
- B.BUFFER MEMORIES
- C.RING STRUCTURE
- D.LOCKS AND KEYS
- E.CALL & RETURN MECHANISM
- F.BINDING
- G.REGISTERS
- H.INTERRUPTS

III.INPUT OUTPUT UNIT

IV.MACHINE INSTRUCTIONS

V.SOFTWARE OVERVIEW

VI.CYBER 170 STATE

Instructor: NUREL BEYLERIAN

CDC: 730-7106 Silvia Clarke Course Coordinator

CDC CYBER 180 DOCUMENTS

*****

1. AN INTRODUCTION COURSE TO CYBER 180

   TEXT
   VIDEO COURSE

2. AN INTRODUCTION TO CYBER 180

3. MODEL INDEPENDENT GENERAL DESIGN SPECIFICATIONS

4. CYBER 180 ARCHITECTURAL OBJECTIVES/REQUIREMENTS

5. NOS/180 PRELIMINARY ERS

   COMMAND INTERFACE
   PROGRAM INTERFACE
   GENERAL INTERNAL DESIGN

6. PASCAL EXTENDED VIDEO COURSE

CYBER 180 Bibliography

Published Literature

1. Kathleen Jensen and Niklaus Wirth, "PASCAL User Manual and Report", Second Edition (New York, NY: Springer-Verlag, 1974)

2. Elliott I. Organick, "The Multics System: An Examination of Its Structure" (Cambridge, MA: M.I.T. Press, 1972)

Control Data Corporation Documentation

3. "CYBER 180 Architectural Objectives/Requirements (AO/R)", Doc. No. ARH1688

4. "CYBER 180 Configuration Notebook", Doc. No. unassigned*

5. "CYBER 180 Mainframe Model-Independent General Design Specification (MIGDS)", Doc. No. ARH1700

6. "NOS/180 External Reference Specification/General Internal Design (ERS/GID)", Doc. No. unassigned*

7. "CYBER 180 System Interface Standard (SIS)", Doc. No. S2196

8. "An Introduction to CYBER 180", Doc. No. unassigned*

9. "PASCAL Extended Language Specification", Doc. No. ARH2298

---

* available from Architectural Design & Control (AD&C), ARHOPS X2697