PRIMOS CONCEPTS & TUNING
(CE1025)

PRIMOS PRINCIPLES & TUNING
(SAOS32)

PRIMOS Revision 20.2
Date: December 10, 1986

All correspondence on suggested changes to this document should be directed to:

Prime Technical Education Center
Prime Computer, Inc.
Prime Park
Natick, MA 01760

Contents

continued on next page

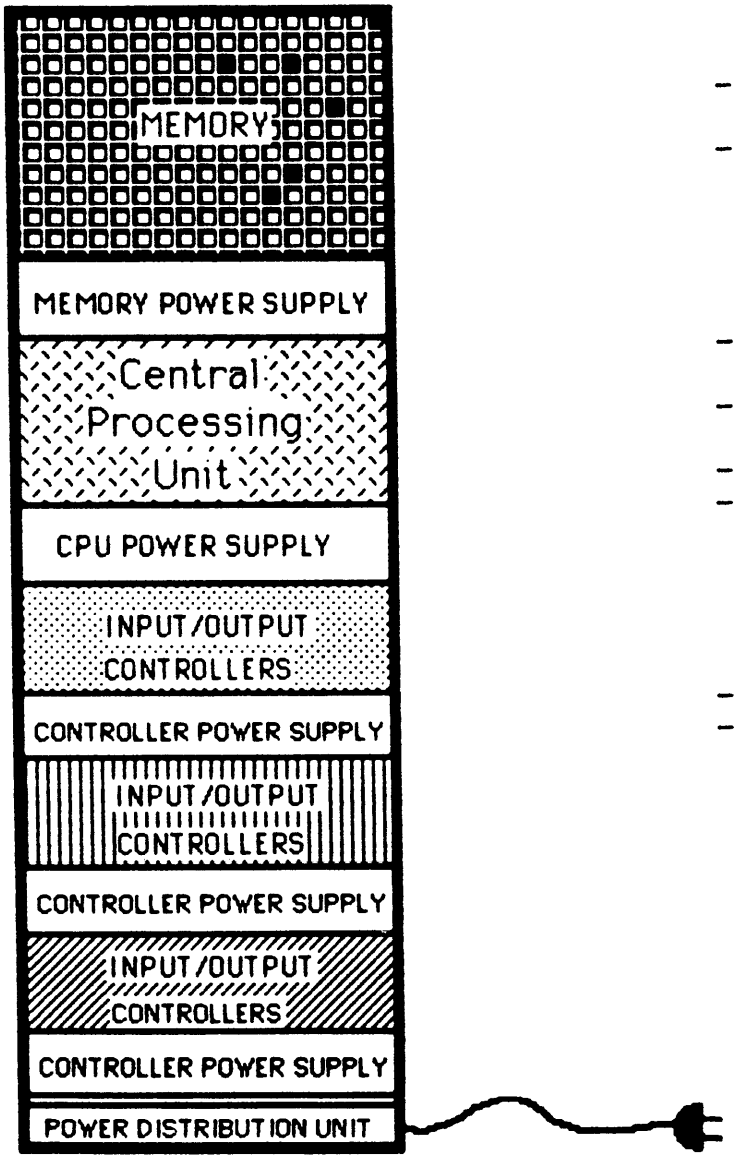
Lesson 1 - Computer Concepts

Objective: Upon successful completion of this lesson, students will be able to:

- Describe some main components of the CPU and define related terminology.
- Define what registers are and how they are organized and used on Prime computers.
- Describe the basic components of the operating system and how they relate to the hardware.

Computer Components
Hardware

50 Series Backplane



Main Memory

- Holds data and instructions
- 1/2, 1, 2, 4, 8 MB boards

CPU

- Performs logical operations
- Performs arithmetic operations
- TTL or ECL logic
- High speed memory components

I/O Controllers

- Aux storage devices
- Data input and output

BUS SIZES

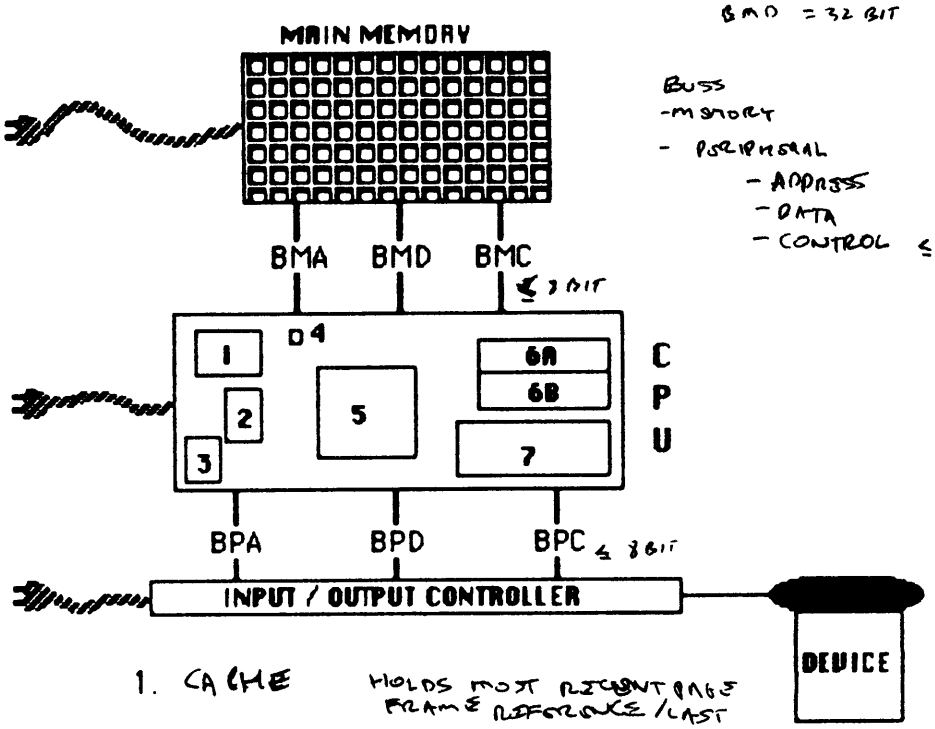
BPA+BPD = 16 BIT

TTL → BMA = 22 BIT
→ BMD = 16 BIT

ECL → BMA = 23 BIT
→ BMD = 32 BIT

9955 II → BMA = 24 BIT
BMD = 32 BIT

Some Components of the CPU



BUSS
- MEMORY
- PERIPHERAL
  - ADDRESS
  - DATA
  - CONTROL ≤ 8 BIT

16-64 KB

1. CACHE — HOLDS MOST RECENT PAGE FRAME REFERENCE / LAST INSTRUCTION EXECUTED AND MEMORY FETCH
2. STLB — VM → PHYSICAL MEM — 4E - WIDE WORD, INTERLEAVED LOADS 64 BITS TO CACHE
3. IOTLB — DISK RECORD → MEMORY ADDRESS
4. PROGRAM COUNTER
5. REGISTER FILE
6. A MICROCODE — INCLUDES CORE ROUTINES LIKE SYSCP...
   B DECODER
7. ALU

Registers and the PC

- Registers - High speed memory locations (on the CPU board) used as work areas for the CPU. Each register contains 32 bits. They are organized into Register Sets (RSx), each containing 32 registers. All of the register sets together constitute the register file.

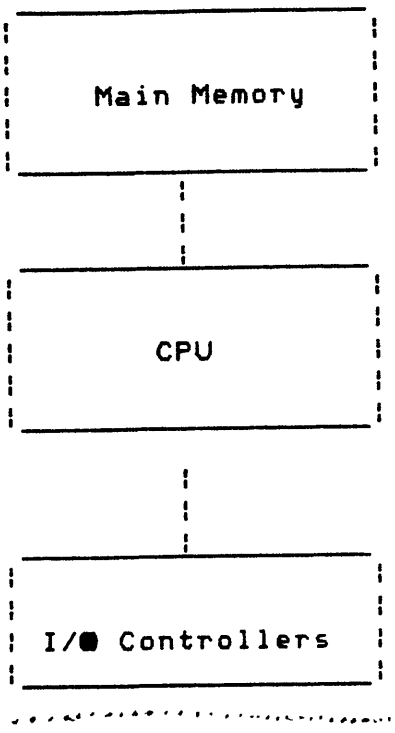
- The Register File:

	RS0	RS1	RS2	RS3
2250 750 850 (2) any older	Micro code scratch	DMA channels	Current Register Set #0 (CRS0)	CRS1

	RS4	RS5	RS6	RS7
9950 9750 9755 9955 9955-II				

	RS8	RS9	RS10	RS11
2350 2450 2550 2655 9650 9655				

Computer Components
Operating System



Memory Management
- Physical
- Virtual 512 MB

Process Exchange
- Scheduling

Program Environment
- PCL PROGRAM CALL LOAD
- Addressing modes
- Exception handling

MMx
Disk I/O
- Locate mechanism

Async Terminal I/O

Software Operation

o In order for a program to execute on a computer, all languages must be broken down into machine level (binary) information. This information can be divided into three main components.

- INSTRUCTIONS
- DATA
- ADDRESSES

Instructions - Instructions tell the machine to do something. They usually affect a register or a memory location.

Data - Data is information stored in memory for use by a program. It can be numeric (integer, floating point, etc) or character (ASCII, EBCDIC).

Addresses - An address points to either an instruction, data, or another address. Addresses are usually calculated by the CPU from information supplied by an instruction. The end result is called the Effective Address (EA).

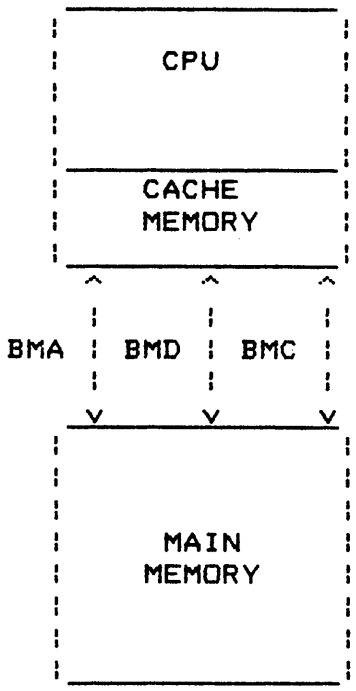
Instructions, data, and addresses are distinguished by the way in which they are used.

Lesson 2 - Memory Management

Objectives: Upon successful completion of this lesson, students will be able to:

- Explain how Cache Memory reduces the effective memory access time for memory reference instuctions.
- Describe how interleaving and wide-word memory fetches work, and the benefits of each.
- Explain how virtual memory is organized.
- Explain how a virtual address is translated into a physical address.
- Describe the function of the STLB.

Cache Functional Diagram



Cache Hit Rate

DETERMINED BY

1. SIZE OF CACHE
2. LOCALITY OF REFERENCE
3. FETCH SIZE

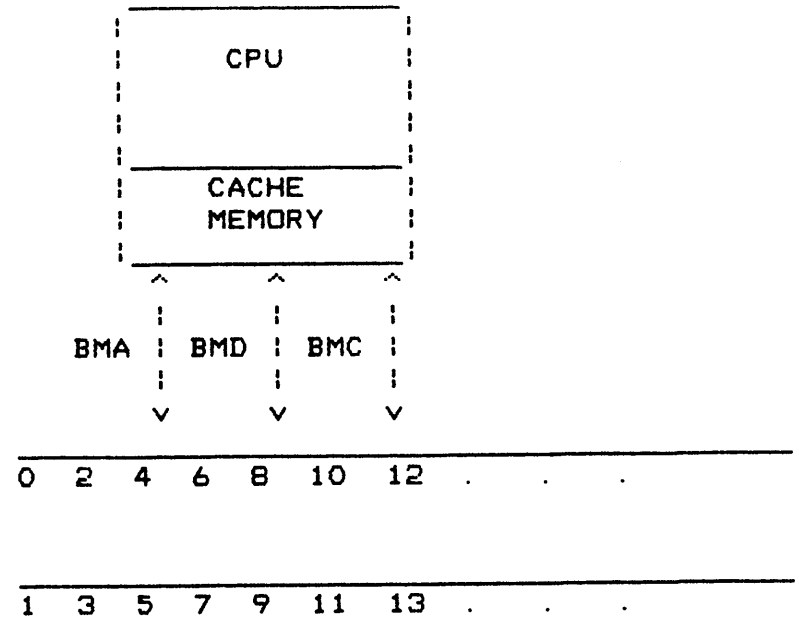
Effective Memory Access Times

Effective Memory Access Time:
- Cache Hit-rate
- Cache Access Time
- Main Memory Access Time

Assuming:
- same locality of reference on all systems.
- all memory boards are interleaved.
- main memory access times are the same on all systems.

	cache size	fetch size	hit rate	cache speed	effective memory access time
2250	2 KB	32	85%	80 ns.	230 ns.
2350	16 KB	64	95%	80 ns.	180 ns.
2450	16 KB	64	95%	80 ns.	132 ns.
2655	16 KB	64	95%	80 ns.	132 ns.
9655	16 KB	64	95%	80 ns.	132 ns.
9750	16 KB	64	95%	40 ns.	105 ns.
9755	16 KB	64	95%+	40 ns.	84 ns.
9955	64 KB	64	98%	40 ns.	58 ns.
9955-II	64 KB	64	98%+	32 ns.	46 ns.
9950	16	64	90-95	40 ns	

Interleaving

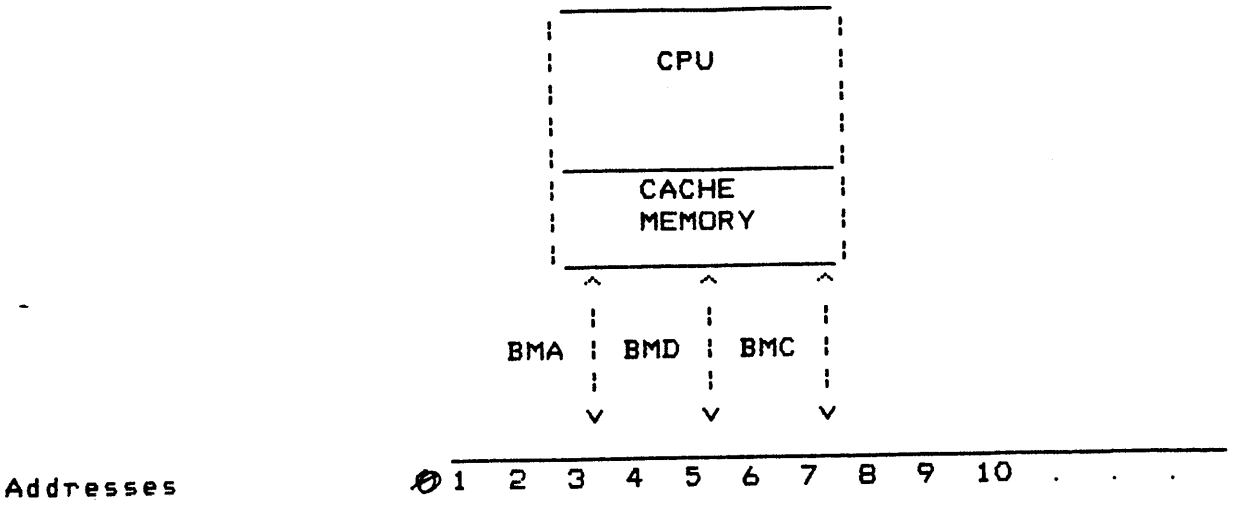


- Interleaving is implemented using two identical boards.

- The same location is fetched off of both boards resulting in 32 bits transfered to cache for one memory fetch.

- 1MB memory boards are self-interleaving. IF CONFIGURED CORRECTLY (1MB IS SWITCHABLE)

Wide-word Memory



WIDE WORD

- The word (16 bits) requested is sent to cache via the data bus.
- The next word (16 bits) is sent to cache via the address bus.
- The 9750, 9755, 9950, 9955, and 9955-II do not use wide-word. They all have a 32 bit data bus.
- Wide-word and interleaving result in 4 words (64 bits) in cache from a single fetch.

WIDE WORD / INTERLEAVE

BMA
2
3

BMD
0
1

0 2 4 6 8
1 3 5 7 9

Cache Benefit Example

Here is an example of a FTN program fragment:

```
      INTEGER*2 ARRAY(3), MAX, INDEX
      DATA ARRAY/10,5,15/                 /* INITIALIZE ARRAY VALUES
      DATA MAX/0/                         /* 0 IS SMALLEST NON-NEG INTEGER
      DO 100 INDEX = 1,3                  /* FOR ALL 3 ARRAY VALUES
  100 IF (ARRAY(INDEX).GT.MAX) MAX = ARRAY(INDEX)
      PRINT *,  MAX
```

The expanded generic assembly language might look like this:

```
      92
      93 A-register = 1
      94 INDEX = A-register
      95 Go To instruction at 98
      96 INDEX = INDEX + 1
      97 A-register = INDEX
      98 If A-register <= 3, Skip
      99 Go To instruction at 105
     100 X-register = A-register
     101 A-register = ARRAY-1 + X-register
     102 If A-register <= MAX, Skip
     103 MAX = A-register
     104 Go To instruction at 96
     105 Print MAX

     200   3            [constant 3]
     300   _____        [INDEX]
     400   0            [MAX]
     401   10           [ARRAY(1)]
     402   5            [ARRAY(2)]
     403   15           [ARRAY(3)]
```

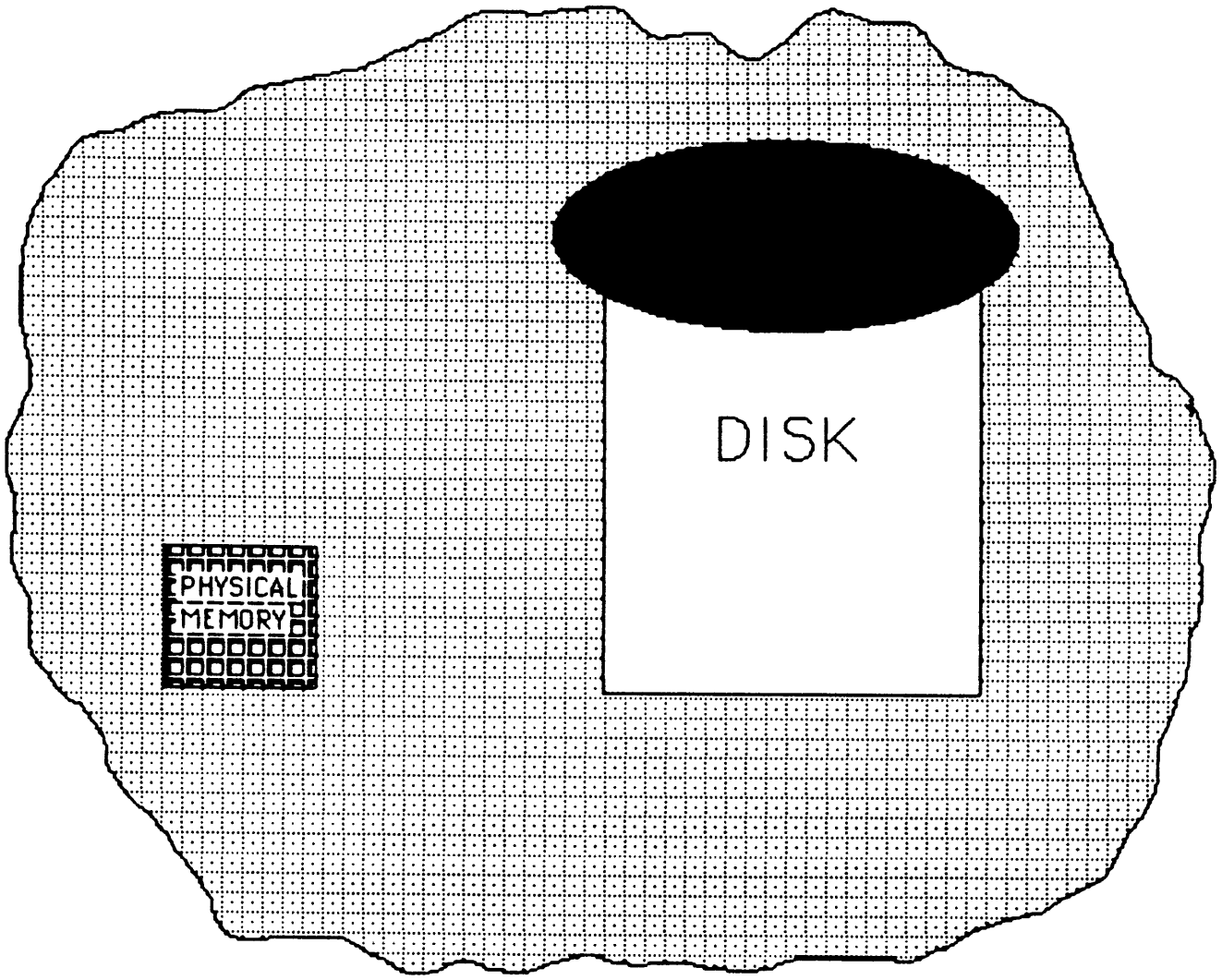
	NONE	YES-16 BIT FETCH	YES-32	YES-64
92			1	1
93	1	1	0	0
94	1	1	1	0
95	1	1	0	0
96	3	1	1	1
97	3	1	0	0
98	4	1	1	0
99	1	1	0	0
100	3	1	1	1
101	3	1	0	0
102	3	1	1	0
103	2	1	0	0
104	3	1	1	1
105	1	1	0	0
	–	–	–	–
200	4	1	1	1
300	10	4	4	4
400	6	3	3	3
401	1	1	0	0
402	1	1	1	0
403	1	1	0	0
MEMORY ACCESSES	52	24	16	12

Cache Memory Example - continued

R = memory reference Read W = memory reference Write
I = Instruction D = Data

1st time	2nd time	3rd time	4th time
R I 93			
R I 94			
W D 300 [INDEX]			
R I 95 (jump)			
	R I 96	R I 96	R I 96
	R D 300 [INDEX]	R D 300	R D 300
	W D 300 [INDEX]	W D 300	W D 300
	R I 97	R I 97	R I 97
	R D 300 [INDEX]	R D 300	R D 300
R I 98 (skip)	R I 98 (skip)	R I 98 (skip)	R I 98
R D 200 [3]	R D 200	R D 200	R D 200
			R I 99 (jump)
R I 100	R I 100	R I 100	
R I 101	R I 101	R I 101	
R D 401 [ARRAY(1)]	R D 402 [(2)]	R D 403 [(3)]	
R I 102	R I 102 (skip)	R I 102	
R D 400 [MAX]	R D 400	R D 400	
R I 103		R I 103	
W D 400 [MAX]		W D 400	
R I 104 (jump)	R I 104 (jump)	R I 104 (jump)	R I 105

Virtual Memory

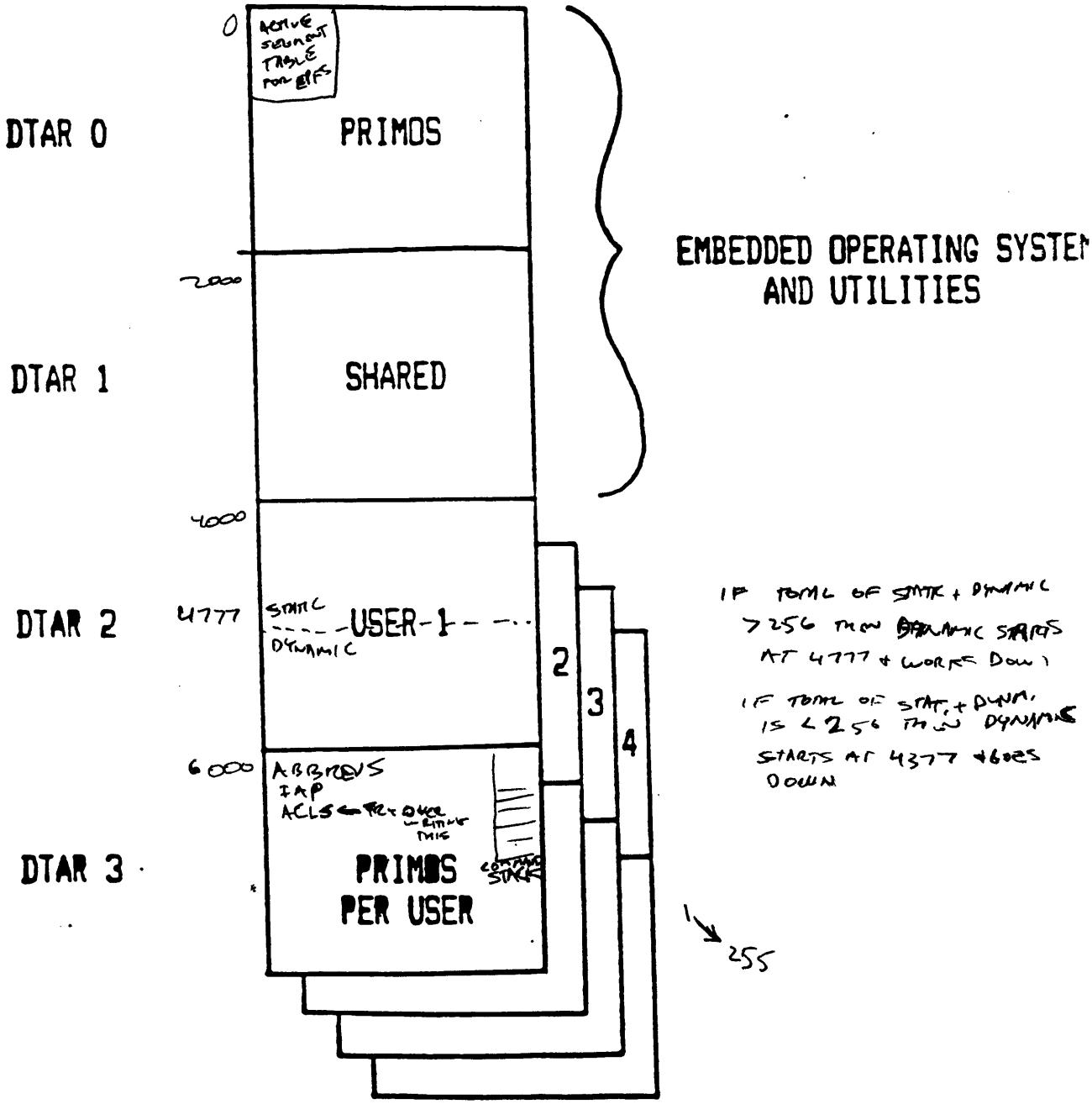


a PAGE is a manageable piece of data

PHYSICAL MEMORY PAGE = 1024 (16 BIT) WORDS
DISK RECORD (DATA) = 1024 (16 BIT) WORDS

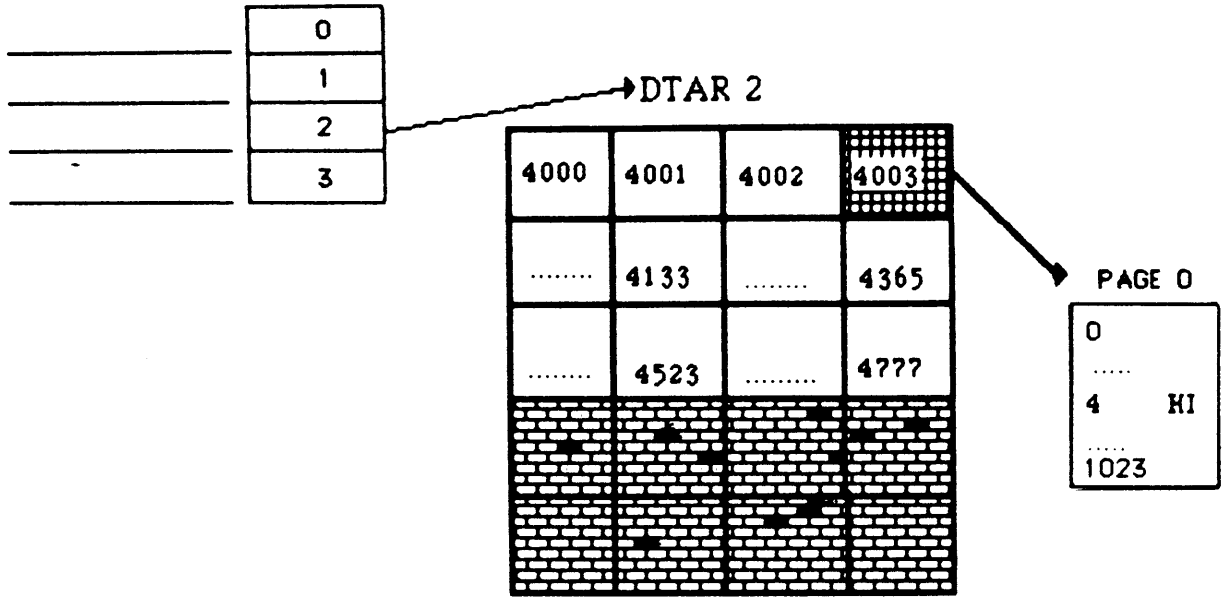
50-Series Virtual Memory Space

DESCRIPTOR TABLE ADDRESS REGISTER



Virtual Memory Delineation

SIZES IN 20.2	OCT	DEC	DME
	'717 = 464		0
	'765 = 502		1
	'777 = 512		2
	'14 = 13		3



VIRTUAL ADDRESS

USER "SEES": | SEGMENT 4003 WORD 4 |

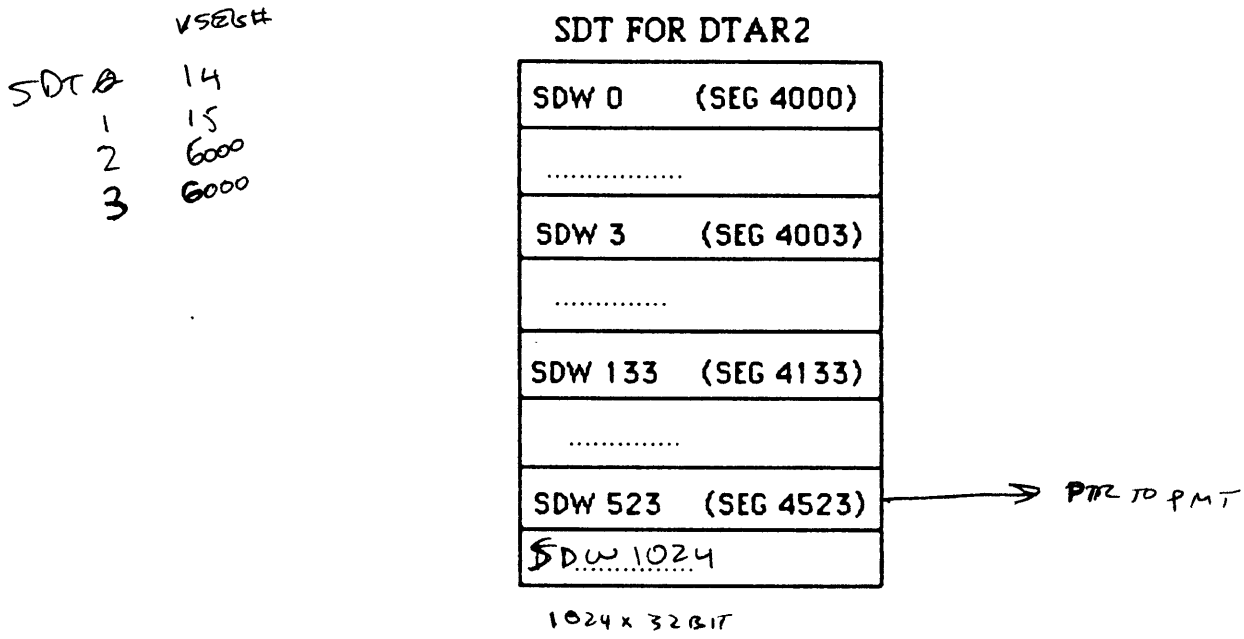
VIRTUAL ADDRESS

HARDWARE "SEES": | DTAR 2 | SEGMENT 0003 | PAGE 0 | WORD 4 |

This page for NOTES

Segment Descriptors

SDT - SEGMENT DESCRIPTOR TABLE: A ~~LIST~~ TABLE OF DESCRIPTORS OF ALL THE SEGMENTS IN A SPECIFIC DTAR. EVERY DTAR HAS AN SDT.



SDW - SEGMENT DESCRIPTOR WORD: INFORMATION ABOUT A PARTICULAR SEGMENT. IT SHOWS:

- IF THE SEGMENT IS USED OR UNUSED
- THE ACCESS RULES FOR THE SEGMENT
  - RING 0 OR 3
- POINTER TO A LIST OF THAT SEGMENT'S 64 PAGES

Page Map Table

PMT - PAGE MAP TABLE: A LIST OF ALL 64 PAGES IN A SPECIFIC SEGMENT. A PMT IS ALSO KNOWN AS AN HMAP.

HARDWARE MAP

32 BITS WIDE = 256 BYTES LONG

8 BYTES/PAGE x 64 PG/SEG x 16 SEGS = 8192 BYTES/SYS ALLOCATED FOR PMTS

PMT ENTRY - CONTAINS INFORMATION ABOUT A SPECIFIC VIRTUAL PAGE. IT SHOWS WHERE THE PAGE IS; PHYSICAL MEMORY AND/OR DISK

IF THE PAGE IS IN PHYSICAL MEMORY, THE PMT ENTRY ALSO SHOWS:

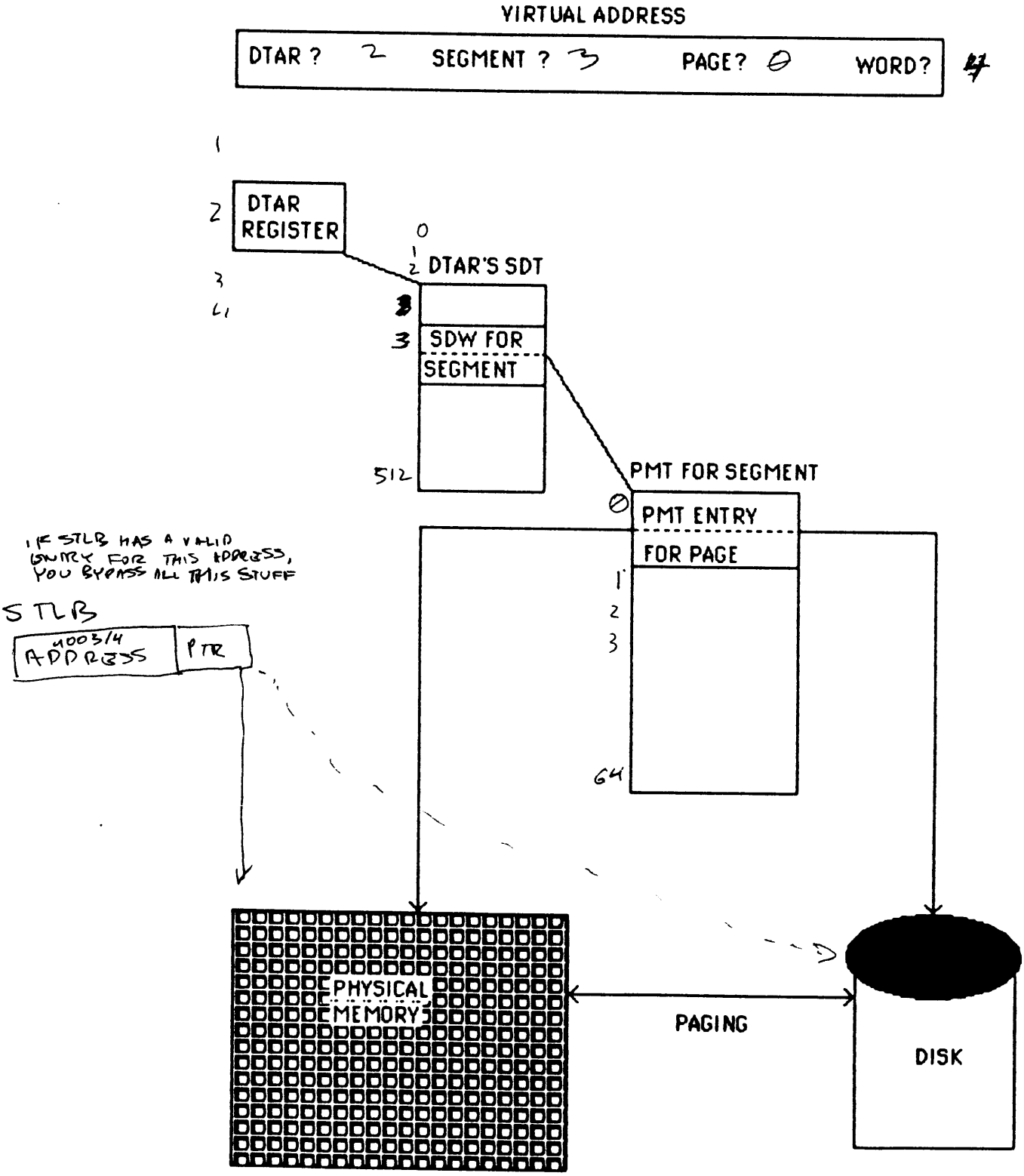
- HOW OFTEN THE PAGE IS USED — USED BIT
- WHETHER OR NOT IT IS PAGEABLE — WIRED BIT
- IF IT HAS BEEN MODIFIED — MODIFIED BIT
- THE PHYSICAL PAGE NUMBER

PMT FOR SEGMENT 4003

PMT ENTRY - PAGE 0
PMT ENTRY - PAGE 1
.....
PMT ENTRY - PAGE 17
.....
PMT ENTRY - PAGE 36
.....
PMT ENTRY - PAGE 55
.....
PMT ENTRY - PAGE 63

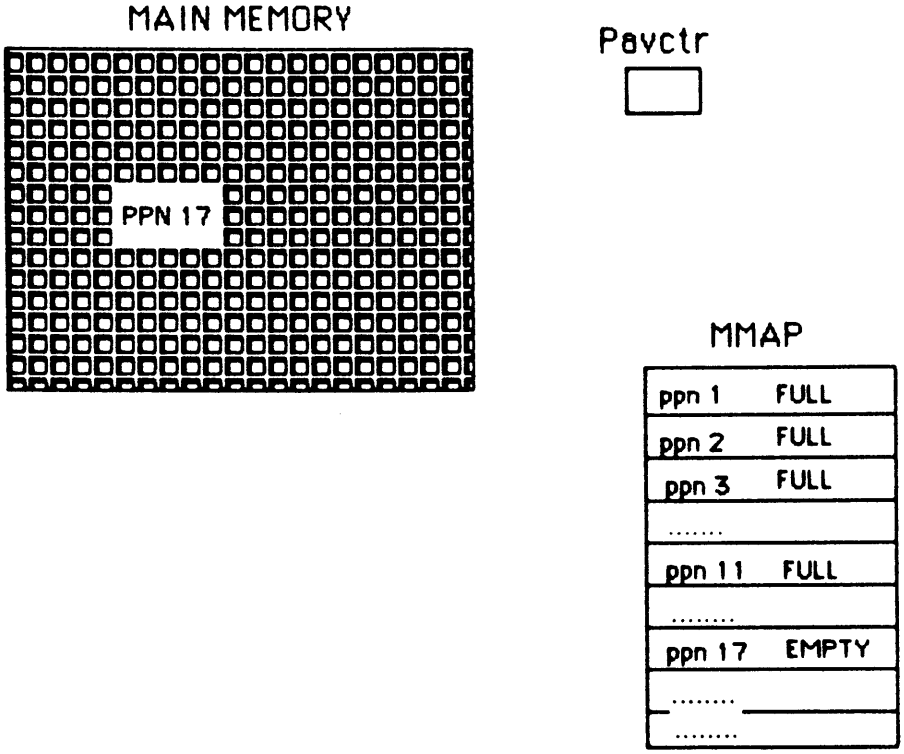
Address Translation

4003/4

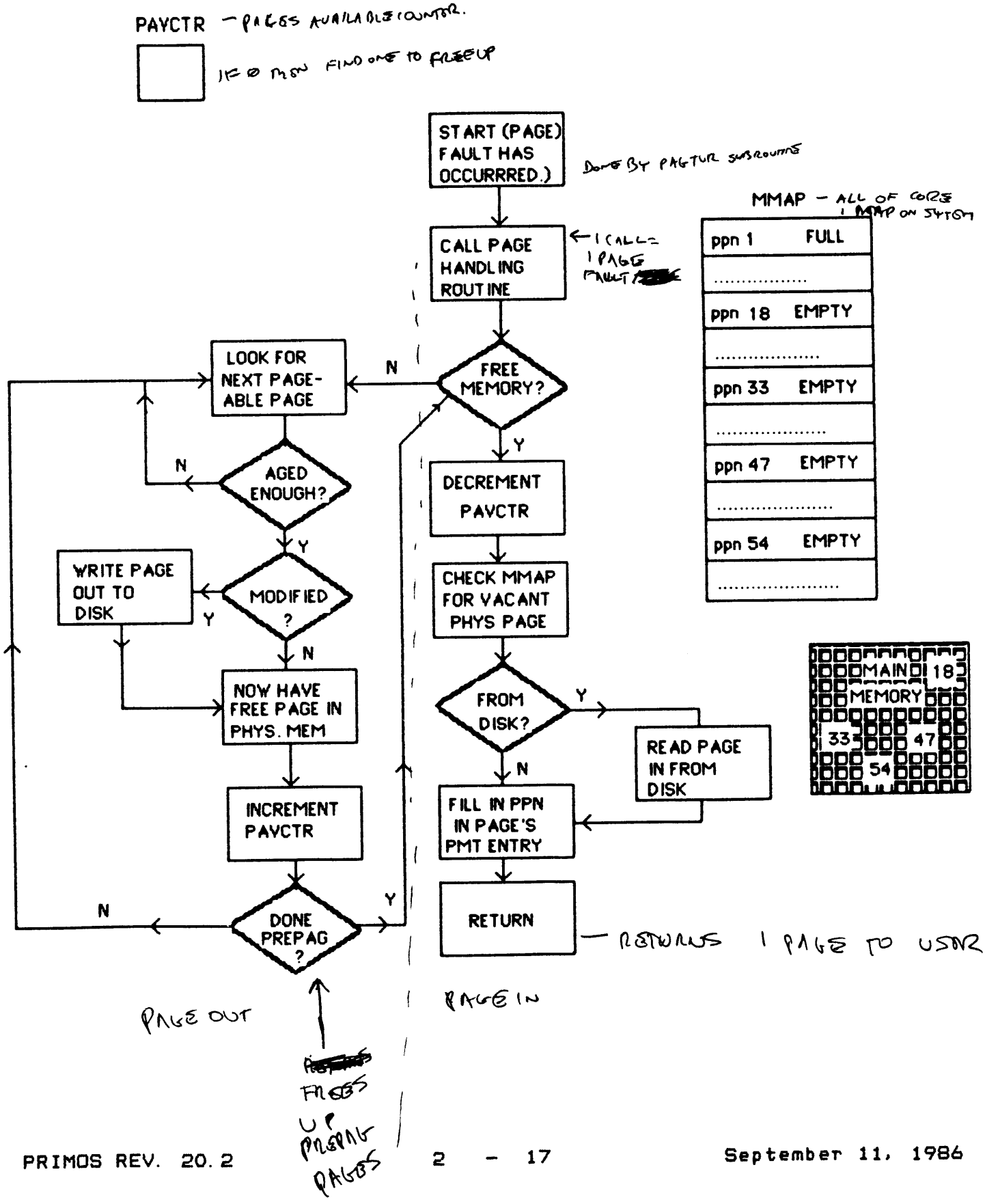


This Page for Notes

Memory Map


MAIN MEMORY
PPN 17
Pavctr
MMAP
ppn 1 FULL
ppn 2 FULL
ppn 3 FULL
.....
ppn 11 FULL
.....
ppn 17 EMPTY
.....
.....

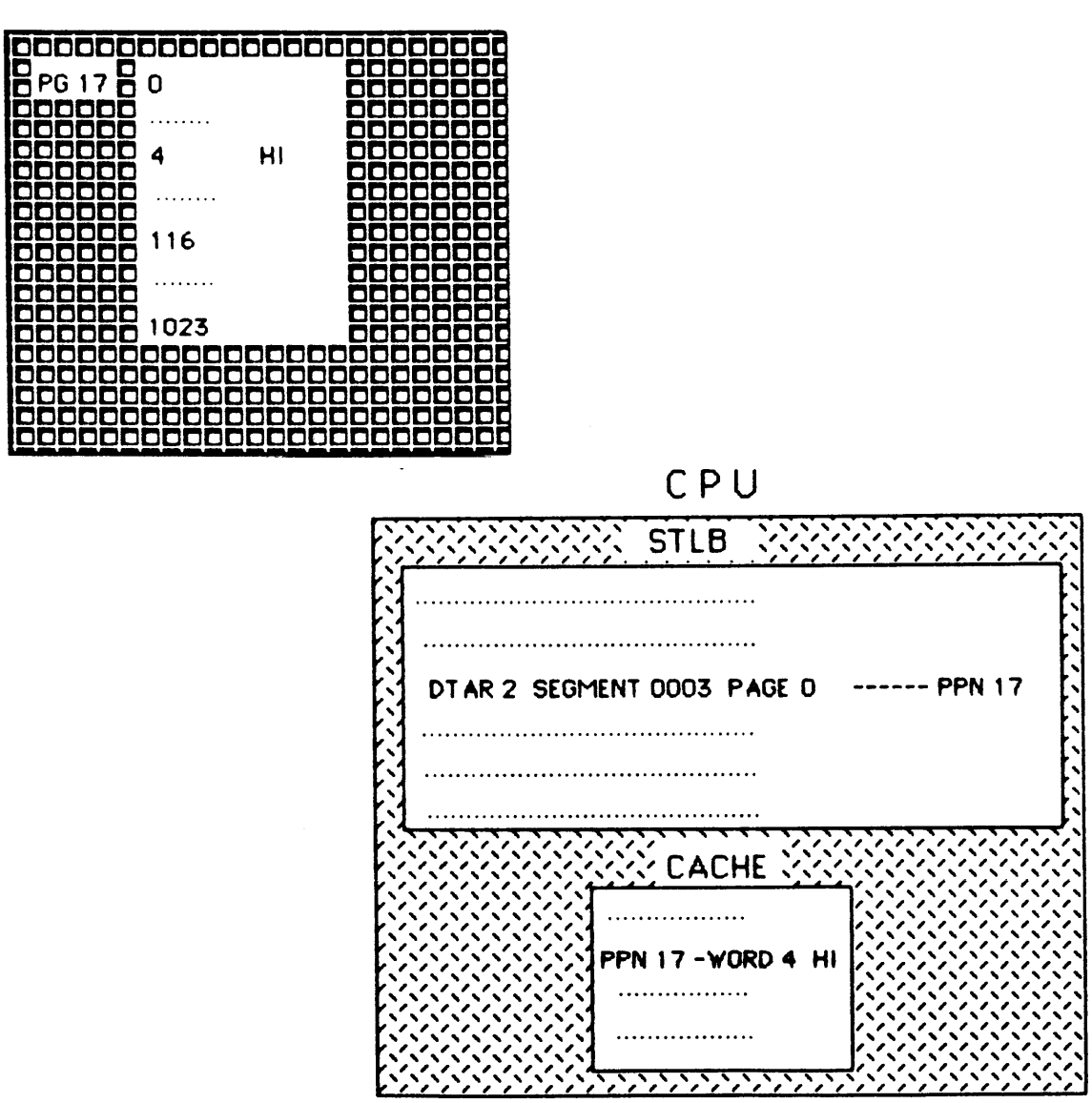
Paging


PAVCTR - PAGES AVAILABLE COUNTER.
IF 0 THEN FIND ONE TO FREE UP
START (PAGE) FAULT HAS OCCURRRED.)
DONE BY PAGTUR SUBROUTINE
MMAP - ALL OF CORE 1 MMAP ON SYSTEM
CALL PAGE HANDLING ROUTINE
1 CALL = 1 PAGE FAULT
ppn 1 FULL
ppn 18 EMPTY
ppn 33 EMPTY
ppn 47 EMPTY
ppn 54 EMPTY
FREE MEMORY?
N
Y
LOOK FOR NEXT PAGE-ABLE PAGE
AGED ENOUGH?
N
Y
DECREMENT PAVCTR
CHECK MMAP FOR VACANT PHYS PAGE
WRITE PAGE OUT TO DISK
MODIFIED ?
Y
N
FROM DISK?
Y
N
READ PAGE IN FROM DISK
MAIN MEMORY
18
33
47
54
NOW HAVE FREE PAGE IN PHYS. MEM
INCREMENT PAVCTR
FILL IN PPN IN PAGE'S PMT ENTRY
DONE PREPAG ?
N
Y
RETURN
RETURNS 1 PAGE TO USER
PAGE OUT
PAGE IN
FREES UP PREPAG PAGES

STLB & Cache Validation

4003/4

VA ≡ DTAR 2 SEGMENT 3 PAGE 0 WORD 4



This page for NOTES

Flushing the STLB

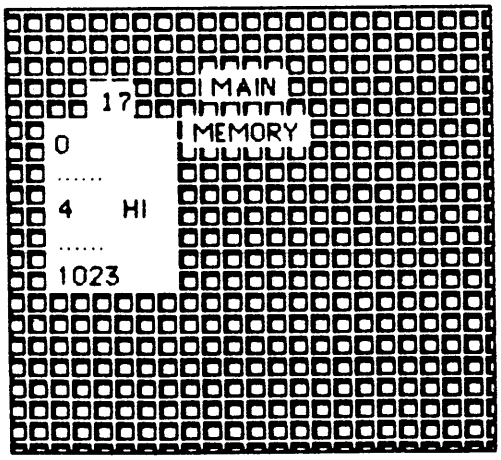
VIRTUAL ADDRESS

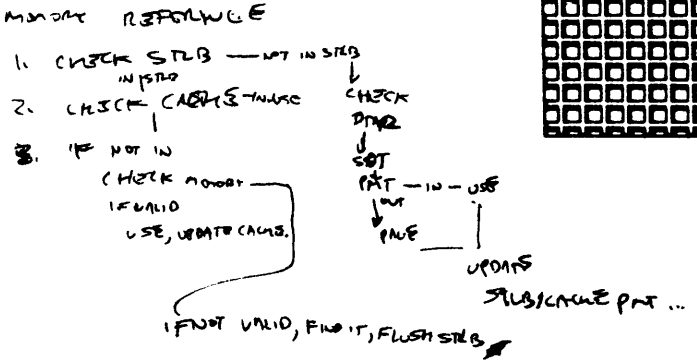
DTAR 2	SEGMENT 0003	PAGE 0	WORD 4

STLB

.....				
.....				
DTAR 2	SEGMENT 0003	PAGE 0	-----	PHYSICAL PAGE 17
.....				
.....				
.....				

CHECKS THAT THE POINTER IN STLB MATCHES THE REQUESTED PAGE IS THE ONE POINTED TO IF SO IT CAN BE USED, IF NOT, THE SYSTEM MUST GET THE CORRECT PAGE. IF THIS HAPPENS IT ALSO FLUSHES THE ENTIRE STLB





Memory Management Exercise

DIRECTIONS: Circle the best answer to each question.

1. Which factor does not affect the cache hit rate?
   A. Size of cache.
   (B) Number of users.
   C. "Locality of reference."
   D. Size of memory fetch.

2. The fast speed of cache and the cache hit rate improve performance by:
   A. Increasing the effective memory access time.
   (B) Reducing the effective memory access time.
   C. Increasing the address translation time.
   D. Reducing the address translation time.

3. The main reason for interleaved memory is:
   (A) To increase the size of a memory fetch. OR SPEED
   B. To pair up memory boards.
   C. To increase the locality of reference.
   D. To ship data up the address bus.
   E. None of the above.

4. Which of the following statements about virtual memory is NOT true?
   A. It is divided into segments.
   B. It is implemented using paging and address translation.
   C. It is the memory addressing range available to programmers.
   D. It allows the combined size of all executing programs to be larger than main memory.
   (E) It is entirely allocated at coldstart.

5. The maximum number of segments PRIMOS Rev 20.2 can support is:
   A. 128.
   B. 1022.
   C. 4096.
   (D) 8192.

6. Page faults are detected during:
   A. Cache hits.
   B. STLB hits.
   C. Process exchange
   D. Address translation
   E. All of the above

7. A page fault (with PREPAG = 3) can result in:
   A. 0 to 4 actual disk I/Os.
   B. 1 to 4 actual disk I/Os.
   C. Always 2 disk I/Os.
   D. Always 4 disk I/Os.

8. The CONFIG directive NSEG:
   A. Specifies how big the SDTs will be.
   B. Specifies the range of DTAR2 segments for all users.
   C. Will allocate NSEG number of total segments.
   D. Will put a limit on the number of PMTs which can be allocated on the system.

9. An SDT describes:
   A. All the enabled segments on the system.
   B. All the enabled segments within a DTAR.
   C. All the pages within a segment.
   D. None of the above.

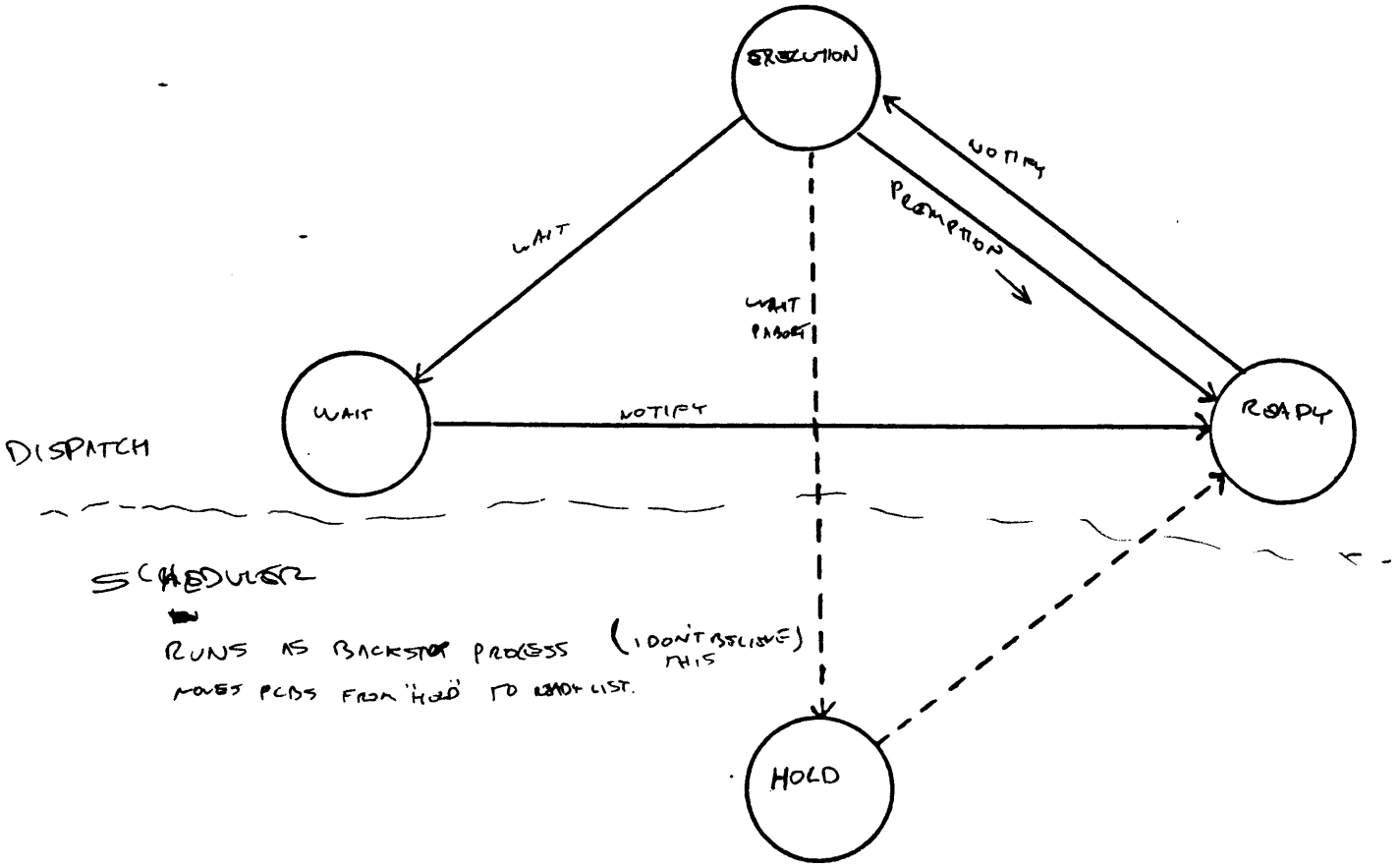
10. Wiring a page:
   A. Means it cannot be shared.
   B. Is accomplished in the STLB.
   C. Means it is never paged.
   D. Removes an entry from the MMAP.
   E. Both (B.) and (C.) are true.

Lesson 3 - Process Exchange and Scheduling.

Objectives: Upon successful completion of this lesson, students will be able to:

- Describe the basic states of a process.
- Describe the operation of the Dispatcher and process exchange.
- Describe how an external interrupt can put a process into operation.
- Describe the purpose of the HOLD queues and the operation of the Scheduler.

Process State Diagram



IF YOU ARE EXECUTING AND A HIGHER PRIORITY PROCESS ENTERS READY QUEUE, YOU ARE PREEMPTED & PUT BACK ON READY LIST.

PABORT = PROCESS ABORT - AFTER EXHAUSTING EITHER MAJOR OR MINOR TIME SLICE YOU ARE PUT "ON HOLD" MAJOR SLICE IS EVEN LOWER

Process Exchange

3 Data Structures

1) PCB PROCESS CONTROL BLOCK ALLOCATED AT LOGIN
2) RSNOW LIST
3) WAIT LIST
   - SEMAPHORES

2 Instructions

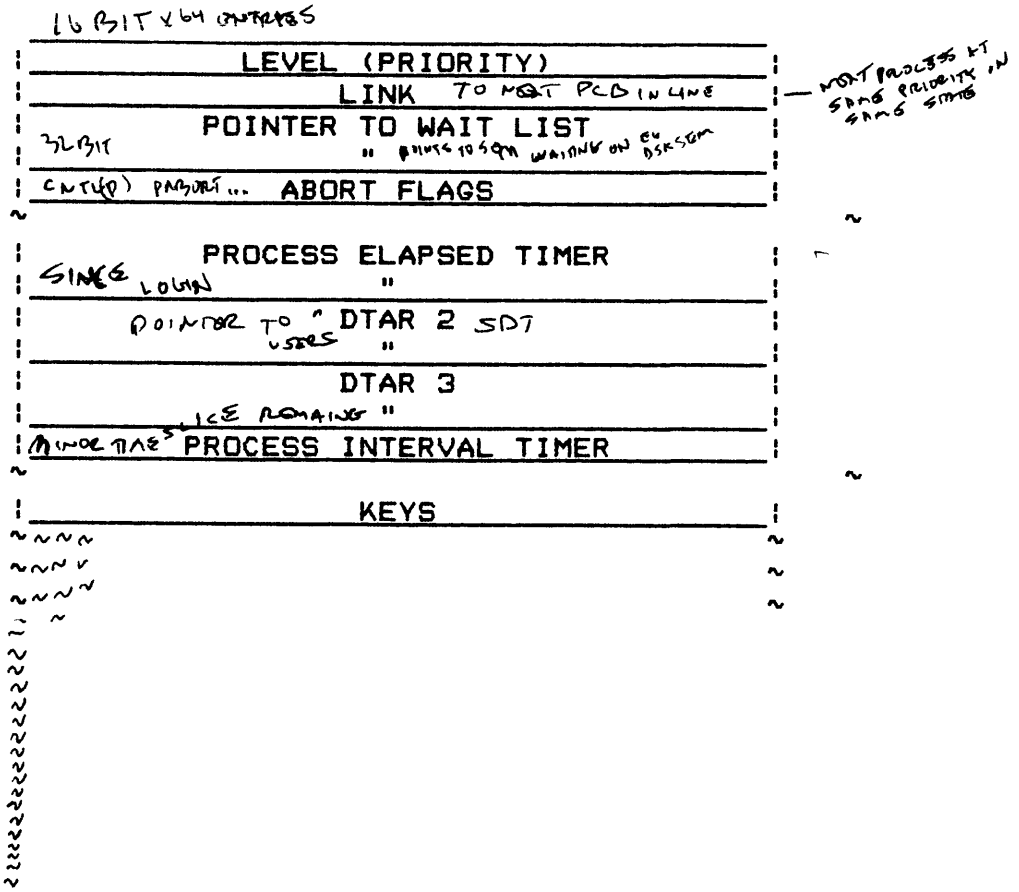
1) WAIT
2) NOTIFY

2 MECHANISMS

DISPATCHER
- HARDWARE - DOES PROCESS EXCHANGE SAVES FOUR REGISTERS - PB
  S B
  Z B
  KEYS

SCHEDULER
- SOFTWARE
- SCHED.PMA

Process Control Block (PCB)



CHAP TUNES MAJOR TIME SLICE & PRIORITY
(52 BITS TUNES MINOR TIME SLICE

PCBS ARE IN SEG 4 '1001000 + USER NUMBER * 100

SEG PCB FOR USER 2 = SEG 4/100200

READY LIST IS IN SEG 4

Ready List Priority Order

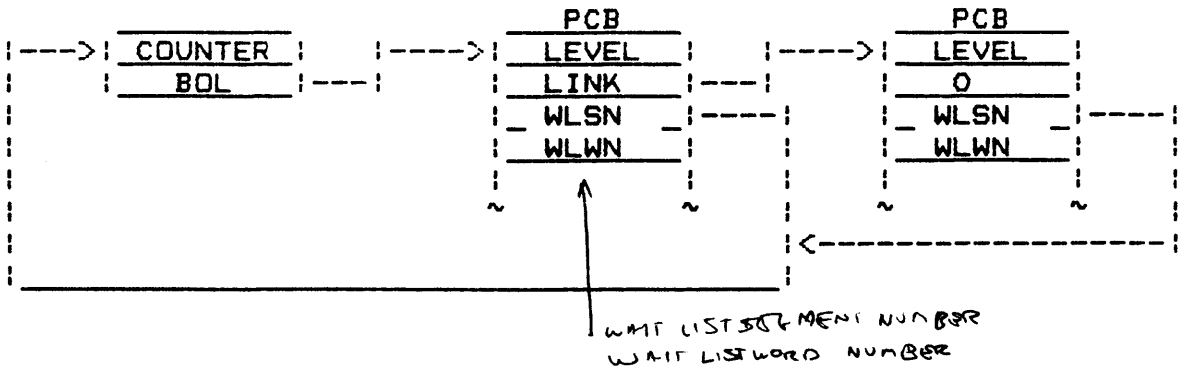
PRIORITY	16 BITS WIDE	
highest	Reserved for system use WAS NETMAN	
	CLOCK PROCESS / FNTSTOP - 850's NO BACKSTOP	%CLK %PNT
	AMLC PROCESS (AMLDIM, ASYNDM)	%AMLC %ASYNC
	SMLC PROCESS (SLCDIM, SLXDIM) SMLC/MLC	%SLC → I DON'T TRUST THIS
	IPQIPC, IPQBSP PROCESSES ROIMPQC - INTERPROCESS COMMUNICATIONS	%SYNC
	MPC PROCESSES (MPCDIM, MP2DIM)	%-MPC
	VERSATEC PROCESS, MPC-4	%VCPPI
	RING NET CONTROLLER PROCESS	%PNC
	DISK PROCESS DSKIO SMDIO	%DSK
-	NETMAN	
	SUPERVISOR PROCESS - CONSOLE	
	USER LEVEL 3	
	USER LEVEL 2	
	USER LEVEL 1 (DEFAULT LEVEL)	
	USER LEVEL 0	
	IDLE LEVEL	
	SUSPEND LEVEL	
lowest	BK1PCB (BACKSTOP 1) / BK2PCB (BACKSTOP 2)	%IDL1
	END OF READY LIST = 1	

DIM = DEVICE INTERRUPT MODULE

MPC = MULTIPURPOSE CONTROLLER = URC = UNIT RECORD CONTROLLER

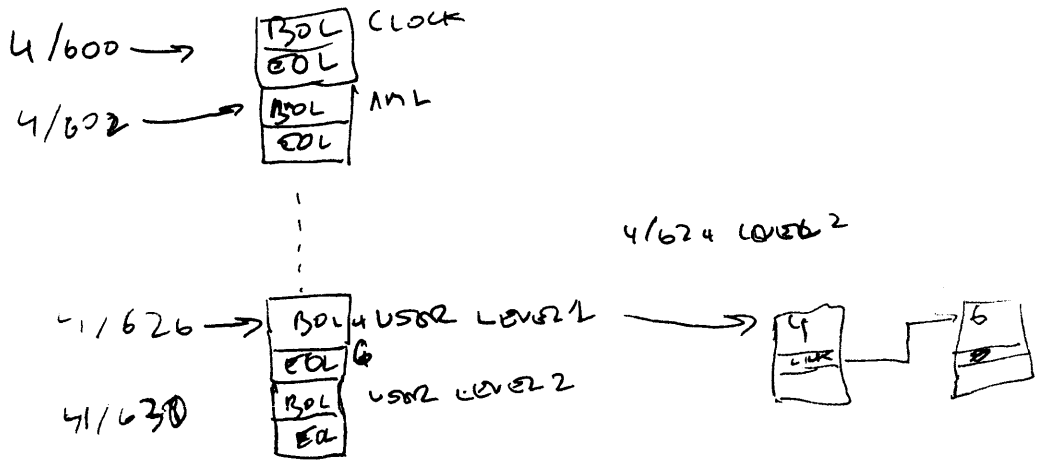
AMLC LASTLINE = CTI - CHARACTER TIME INTERRUPT 1/CHAR 9600 = 960 INT/SEC

Wait List (Semaphore)



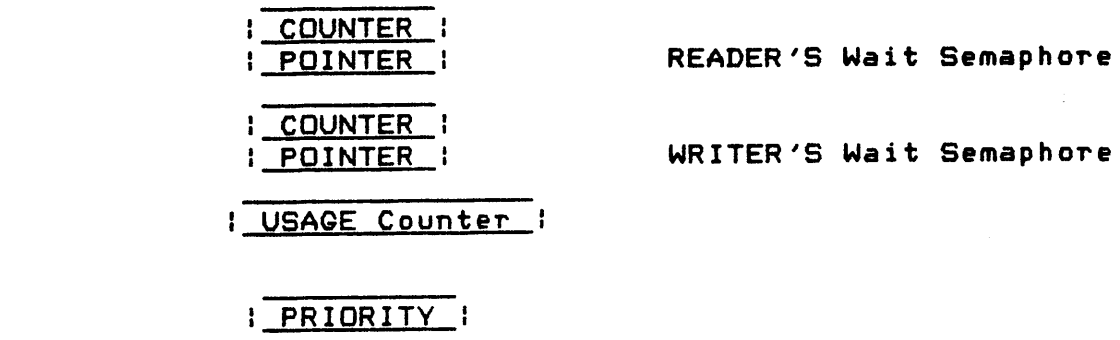
WAIT <semaphore name>
access semaphore
count = count + 1
if count > 0
then PCB --> Wait List
else process continues

NOTIFY <semaphore name>
access semaphore
count = count - 1
first PCB --> Ready List



System Locks

Each lock consists of the following data structure:



Locks will allow N readers or 1 writer.

A writer will wait on the writers semaphore if there are any active readers or an active writer.

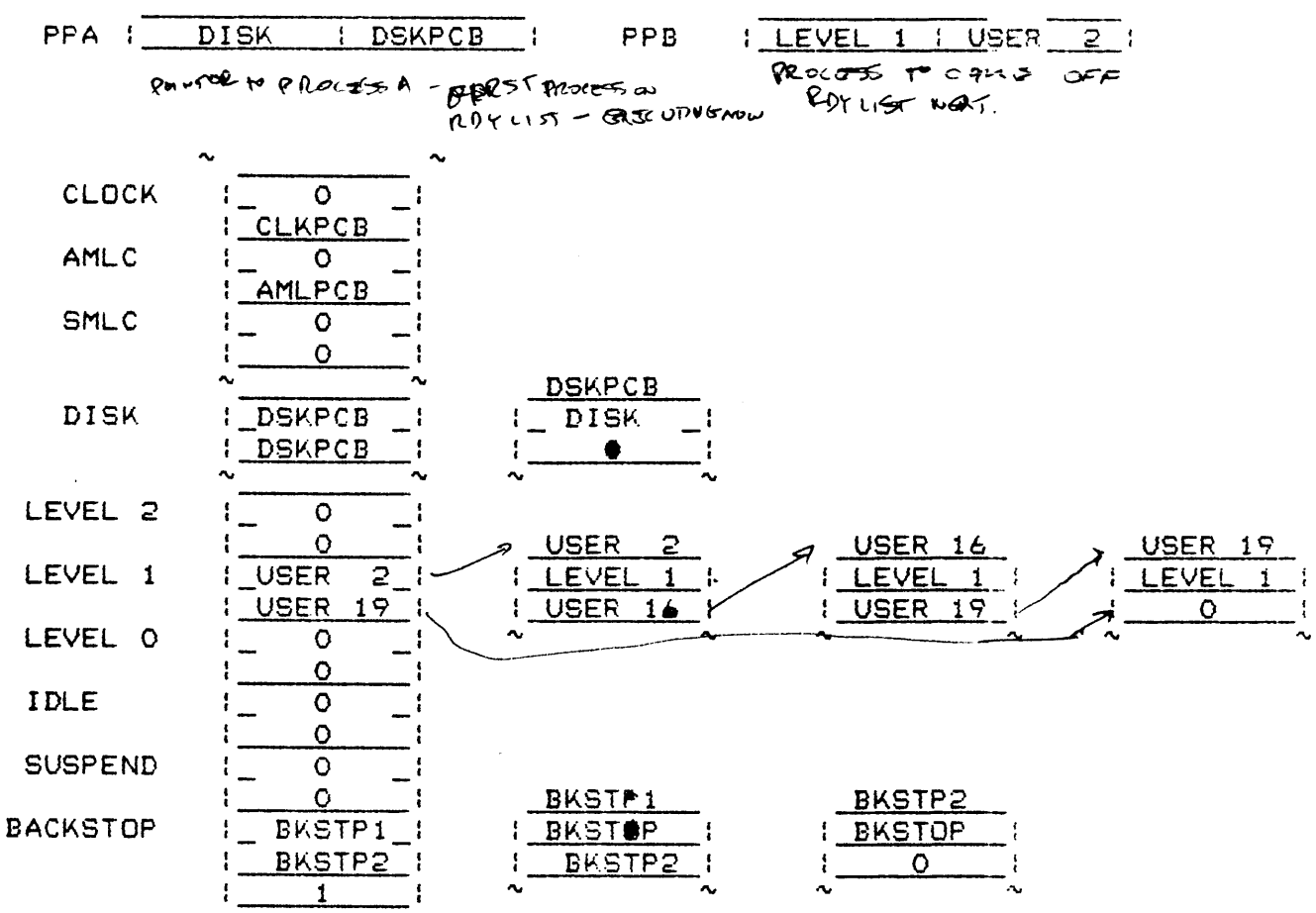
A reader will wait on the readers semaphore if there is an active writer or if a writer is waiting.

When the USAGE counter is equal to

0	the lock is free (available)
+N	there are N active readers
-1	there is one active writer

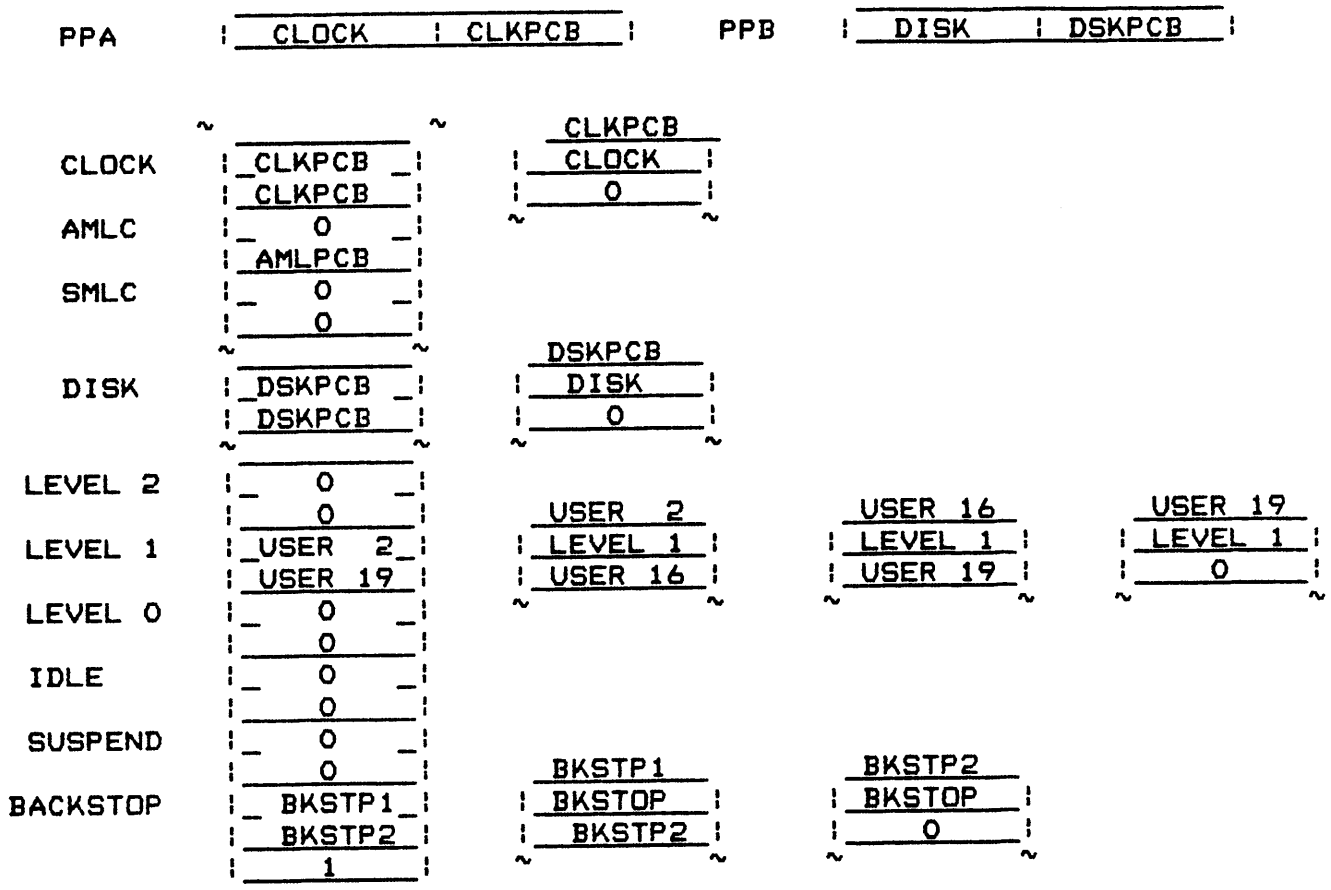
Priority is used to force an order to avoid deadly embrace situations.

Ready List Example 1



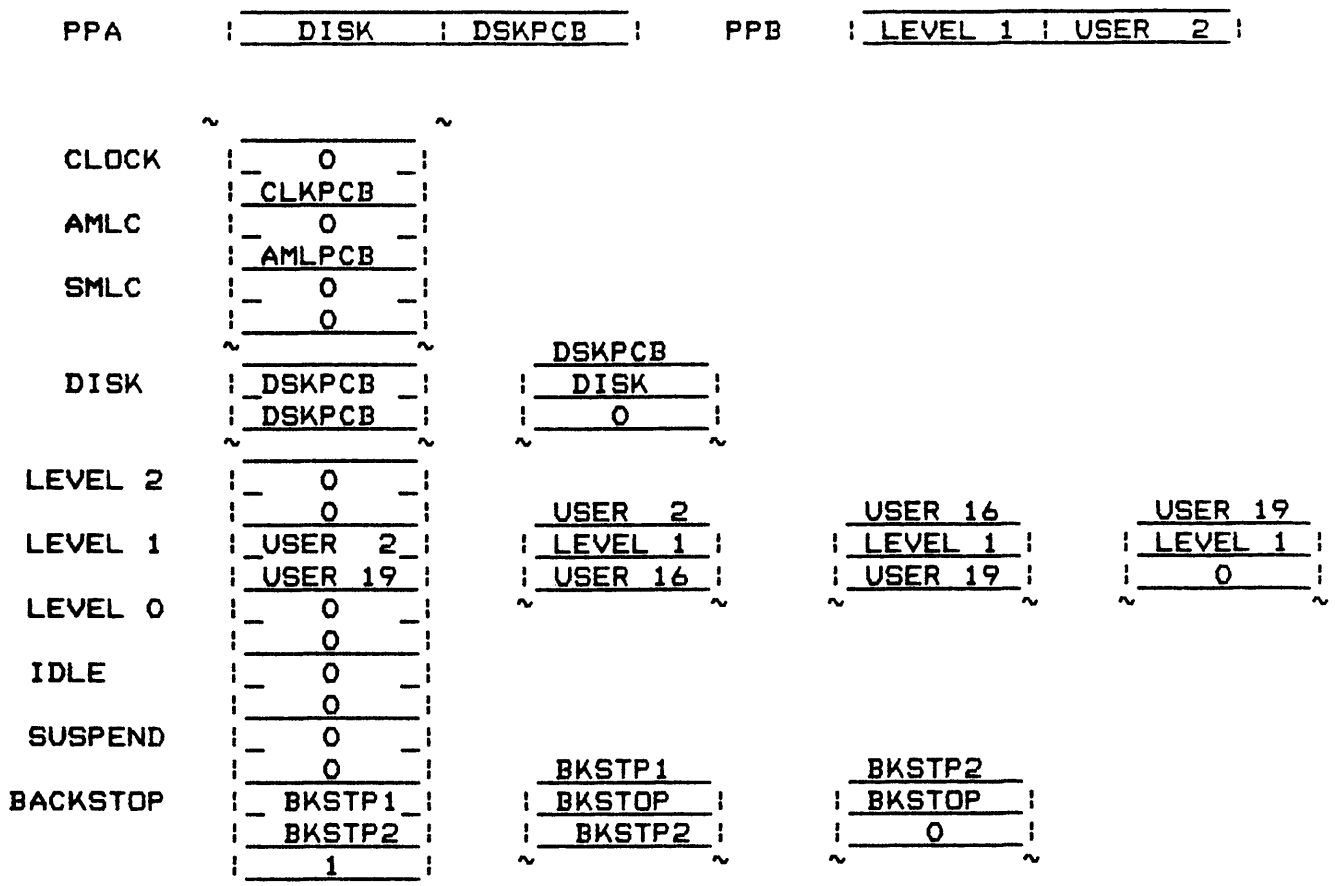
Ready List Example 2

THIS IS A PREEMPTION OF EXAMPLE 1

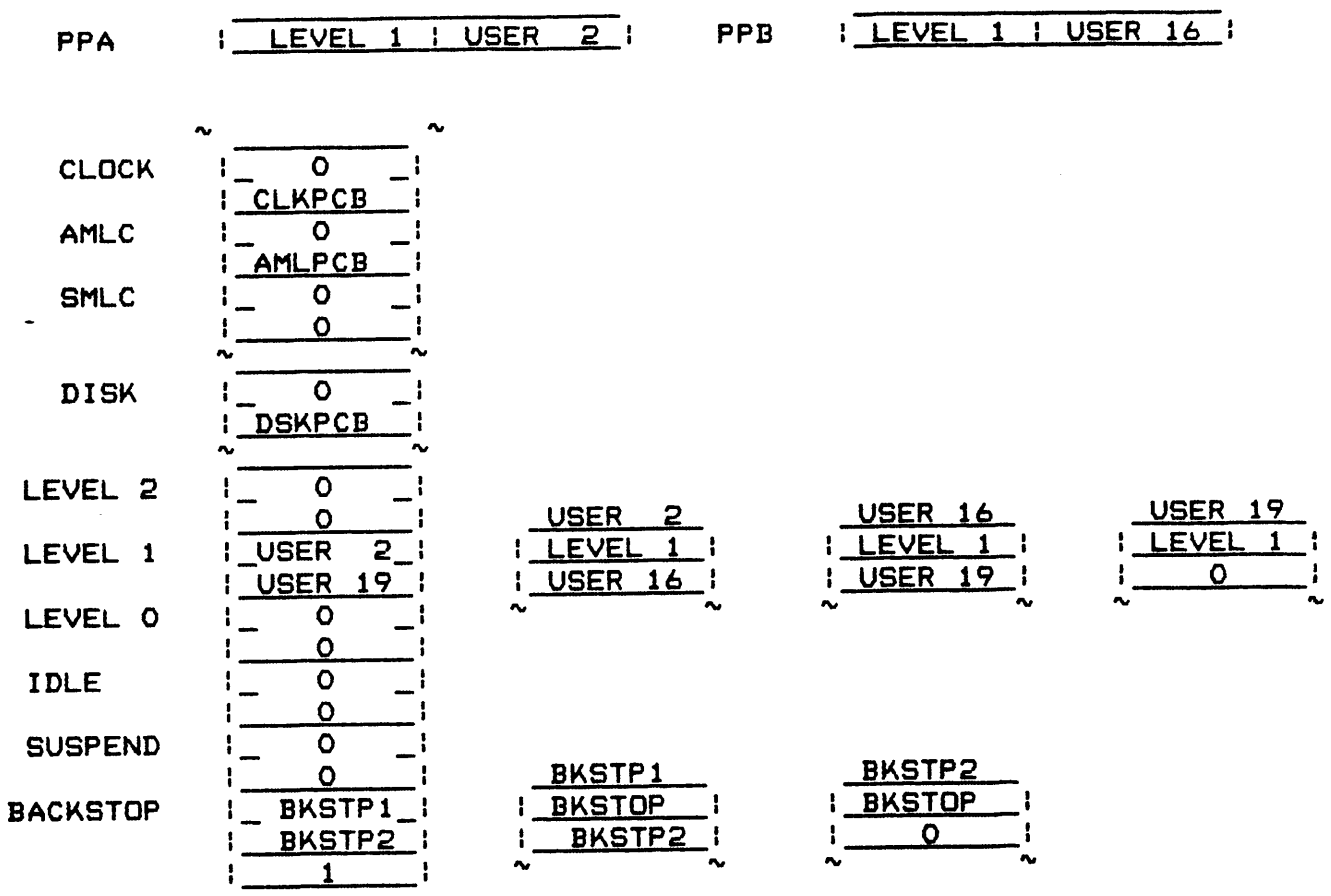


Ready List Example 3

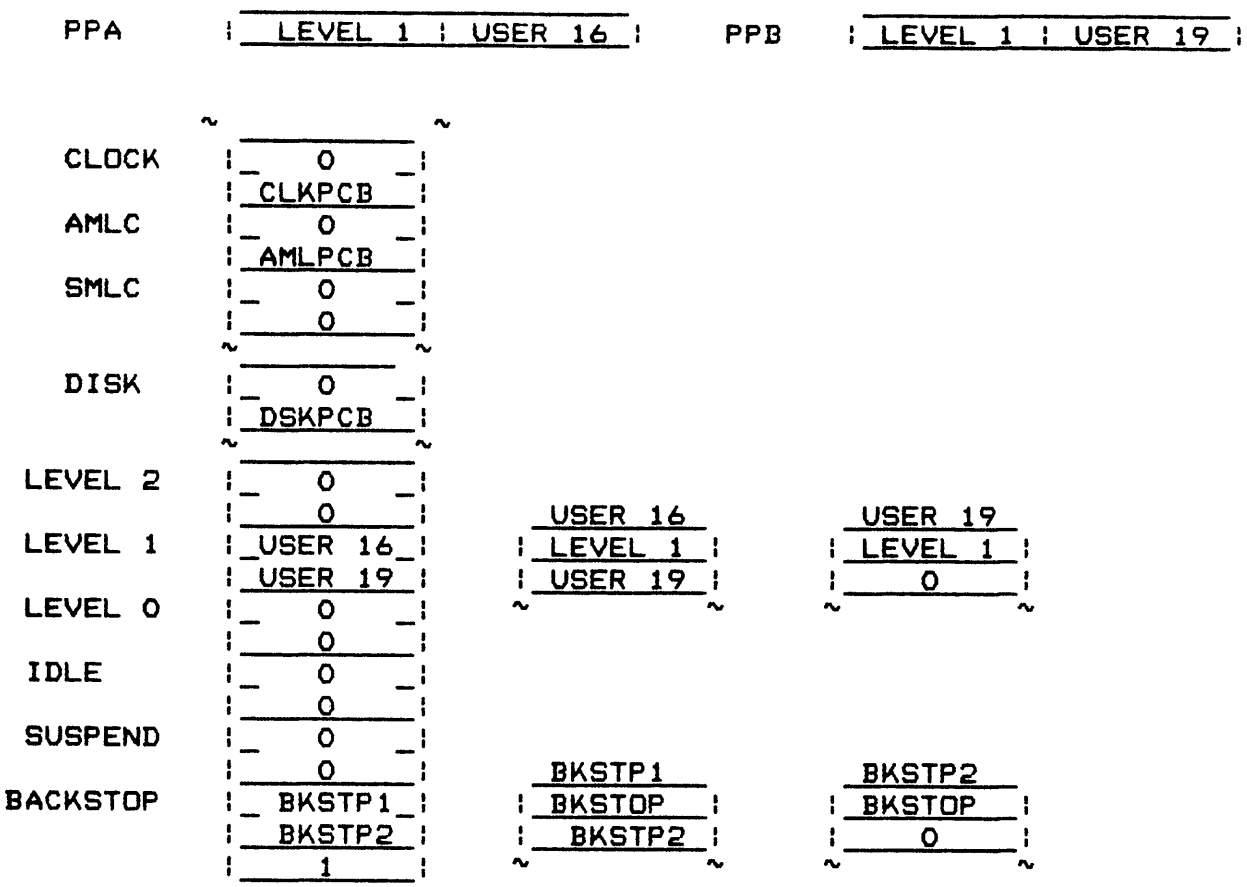
CLOCK FINISHED DISK RECORD TO ACTION



Ready List Example 4

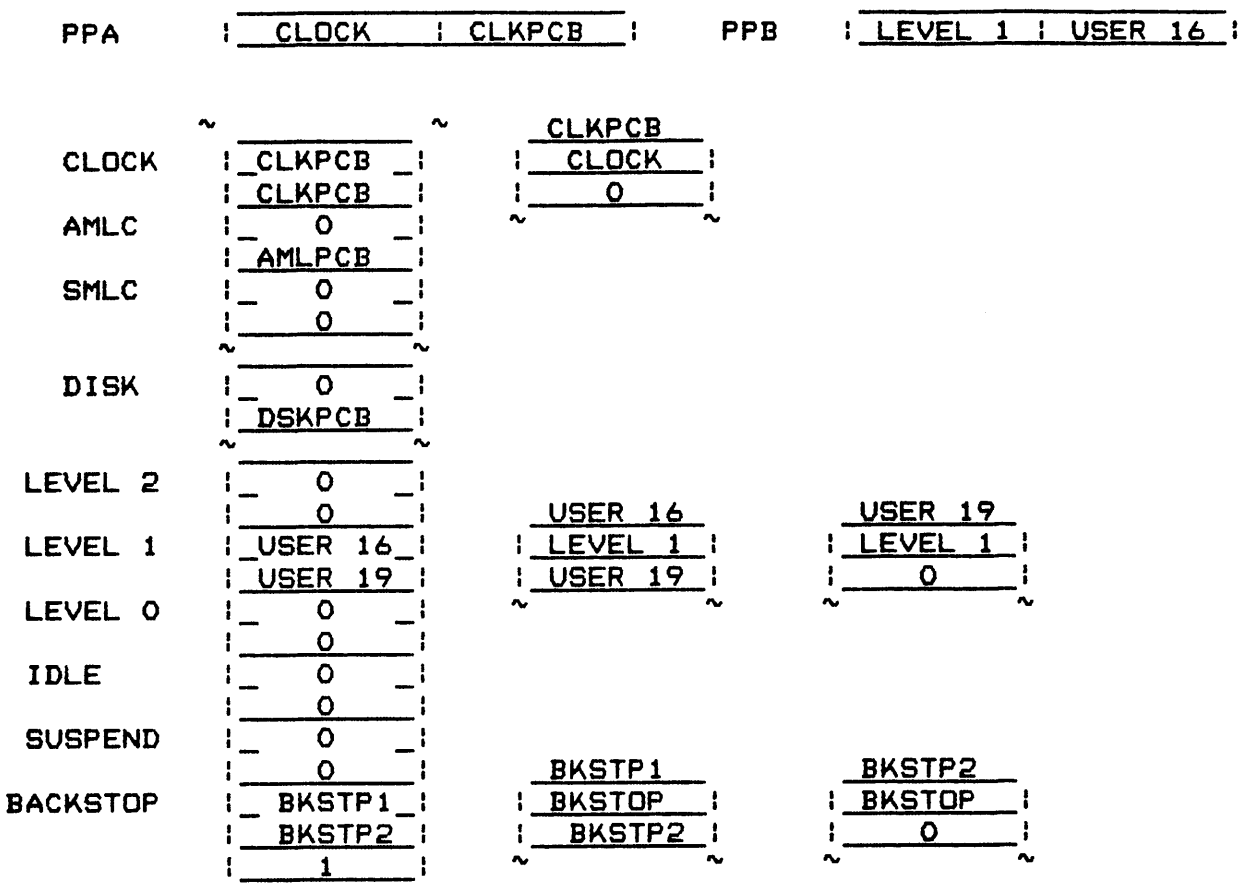


Ready List Example 5

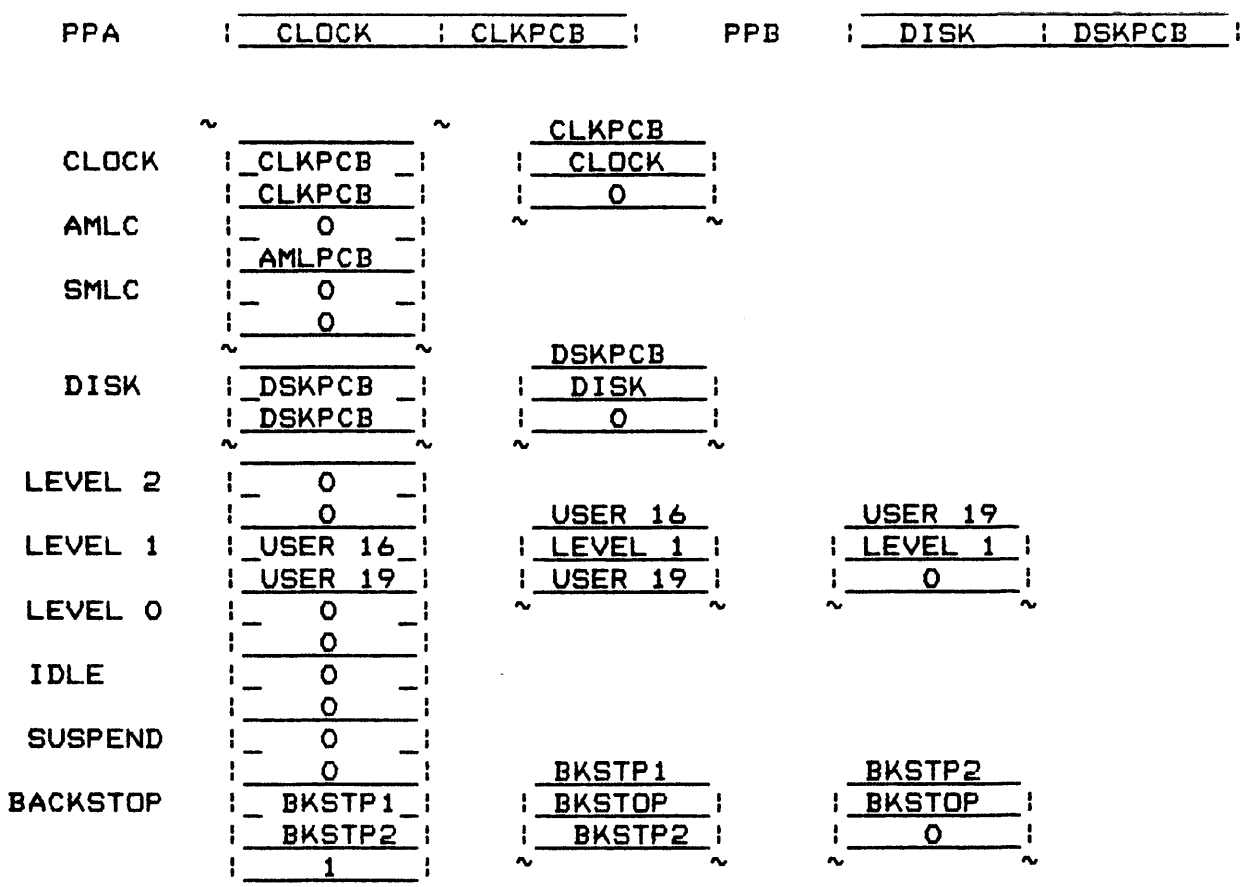


Ready List Example 6

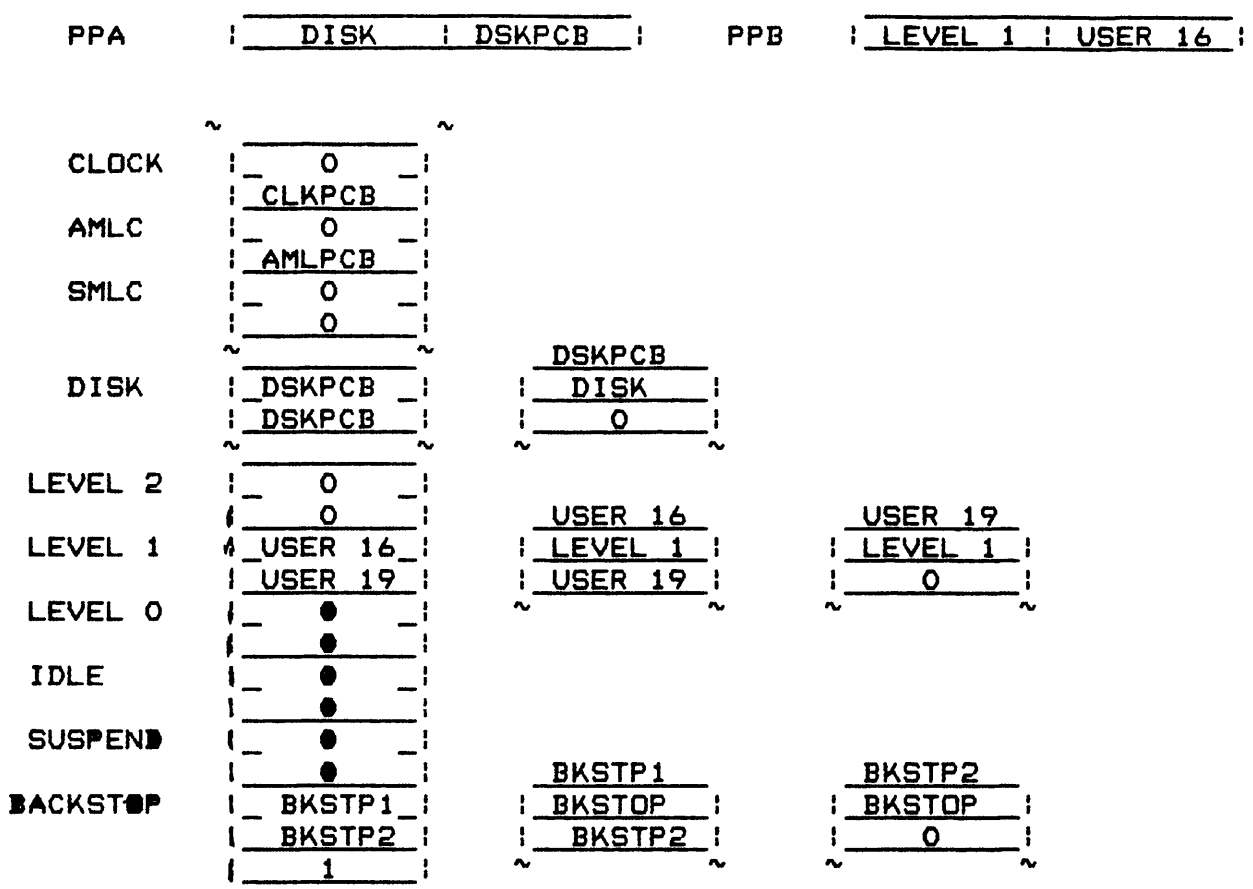
PREEMPTION



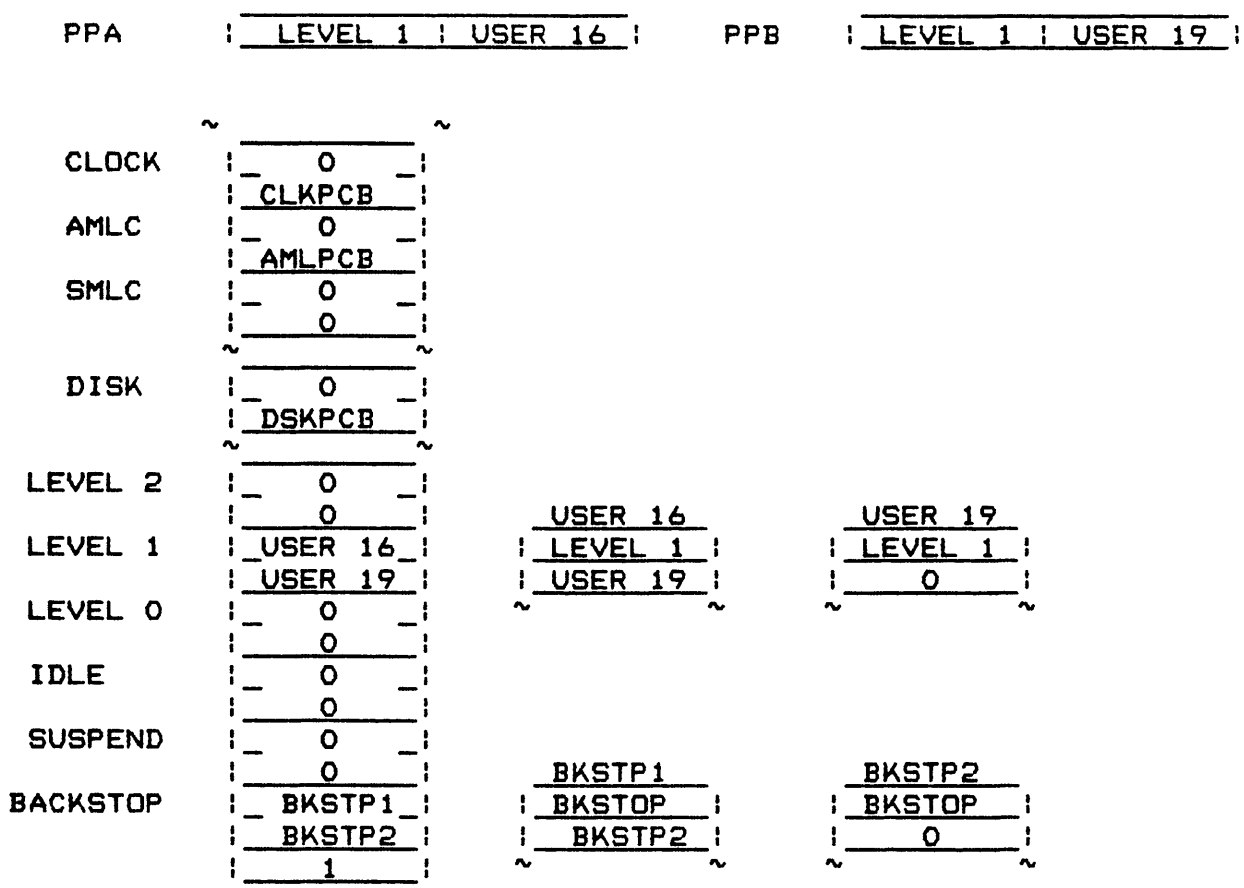
Ready List Example 7



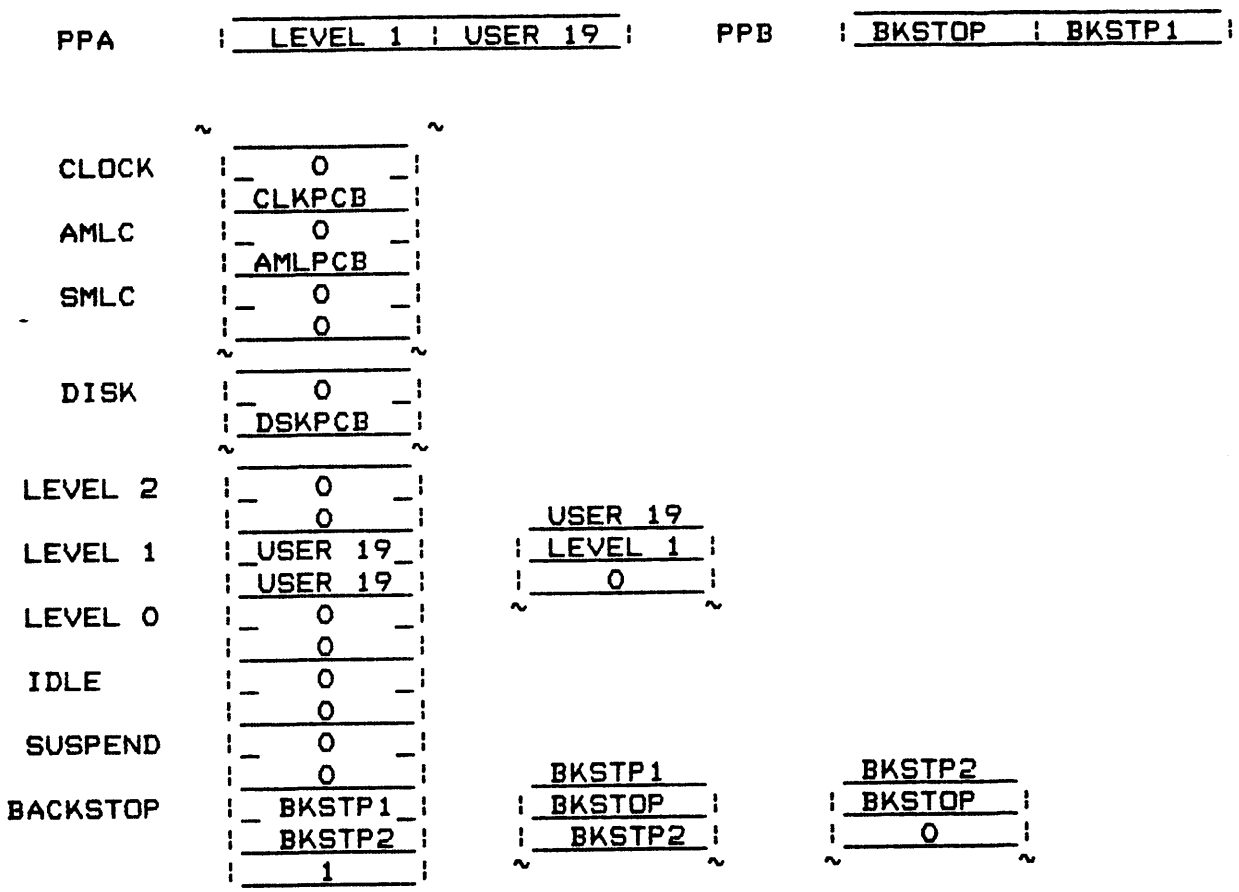
Ready List Example 8



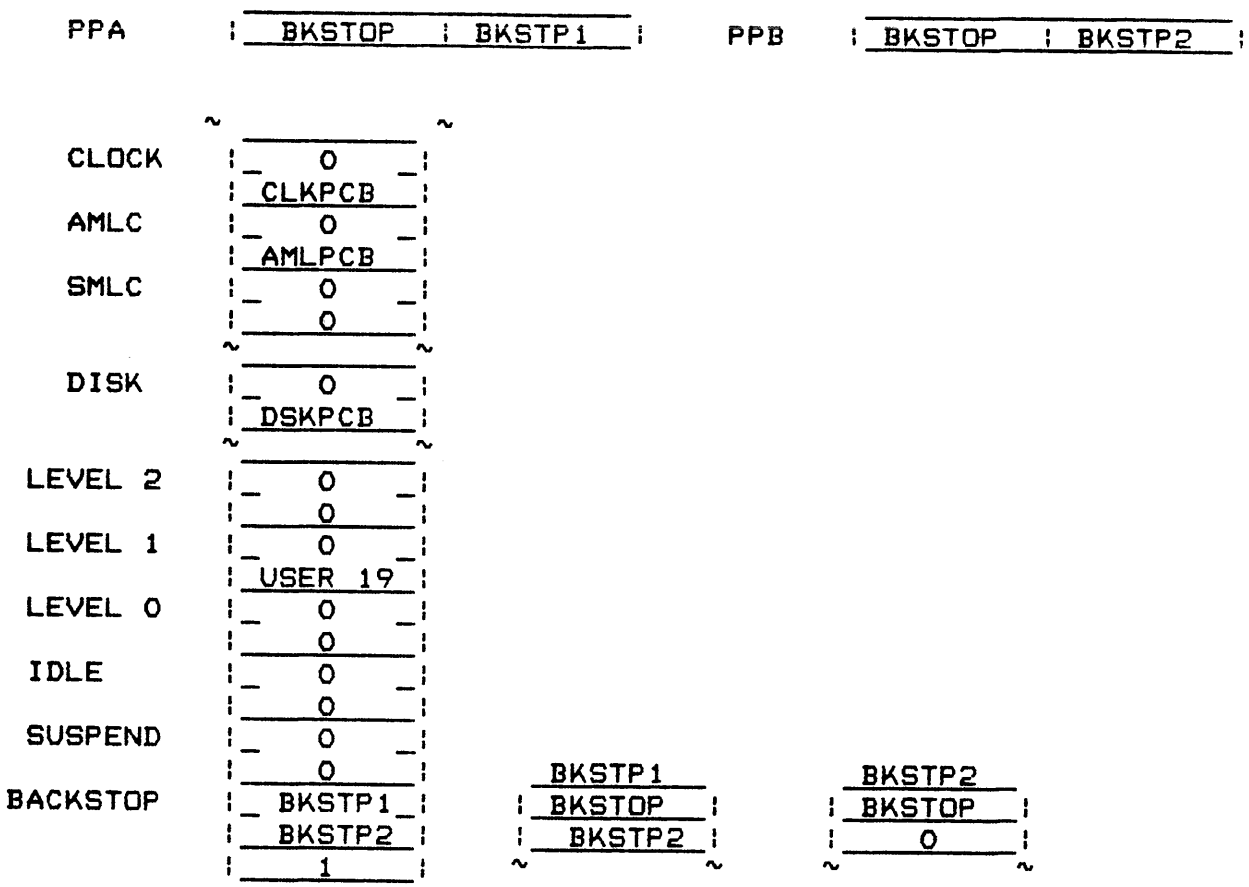
Ready List Example 9



Ready List Example 10



Ready List Example 11



External Interrupts

- There are three basic catagories of exceptions:

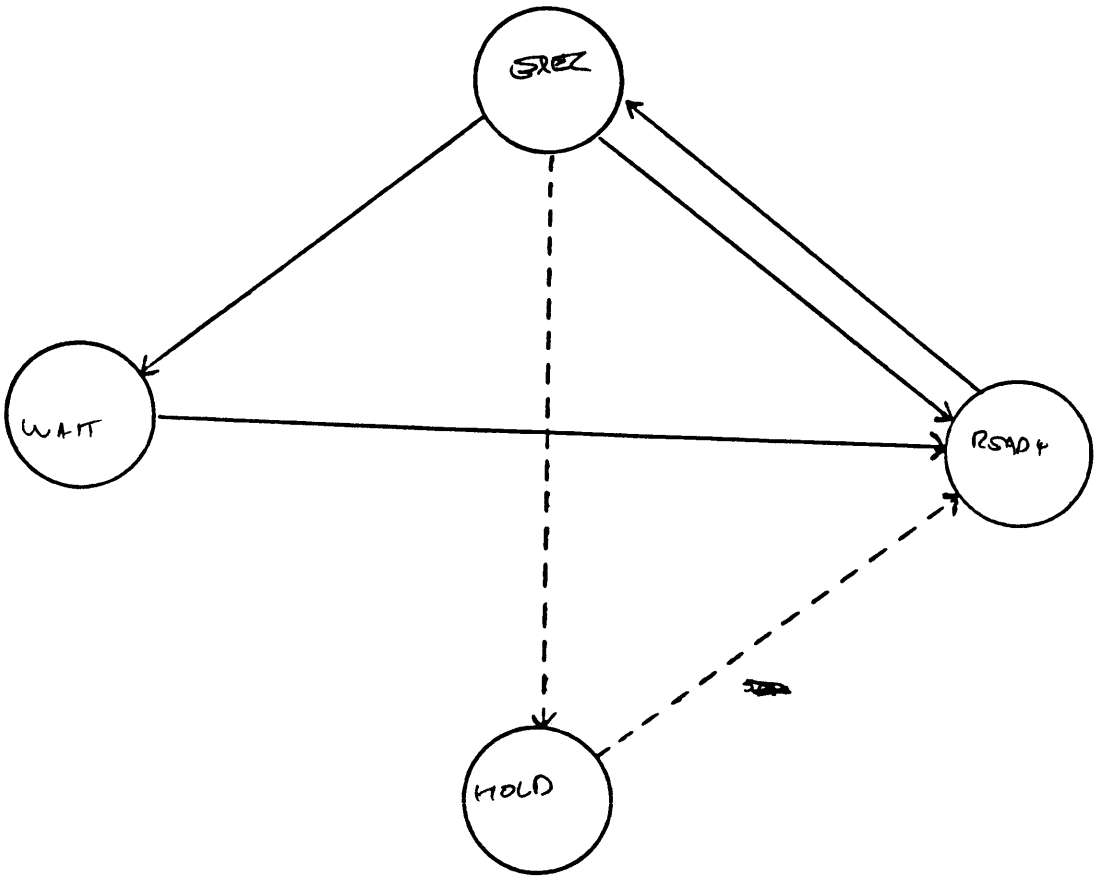
Exceptions
1) Interrupts.
2) Checks.
3) Faults.

- An external interrupt is a method by which a controller can notify a process to the ready list.

- Here is the basic sequence of events:
  1) Controller raises an interrupt request.
  2) CPU acknowledges interrupt.
  3) Controller ships CPU an address on BPA. IN MICROCODE SCRATCH
  4) CPU will save PC, PB, and KEYS in special registers and load address from controller into PC. ↓ SAVES PROCESS INTERVAL TIMER IN CORRECT PCB
  5) CPU will now execute Phantom Interrupt Code (PIC).
  6) PIC usually consists of an INEC instruction which will:
     a. tell controller that interrupt is being serviced.
     b. notify an interrupt process to the ready list.
     c. restore the PC, PB, and keys.

INEC = INTERRUPT NOTIFY EXTERNAL CALL

Process State Diagram
Interactive User Processes



~~SCHEDULER~~ MOVES PCBS FROM HOLD TO READY
BACKSTOP PROCESS ("THE SCHEDULER")

Scheduling Of Users

- PRIMOS scheduling is based on two criteria.
  - PROCESS EXCHANGE
  - SCHEDULER which consists of:
    a. Backstop process
    b. SCHED subroutine.

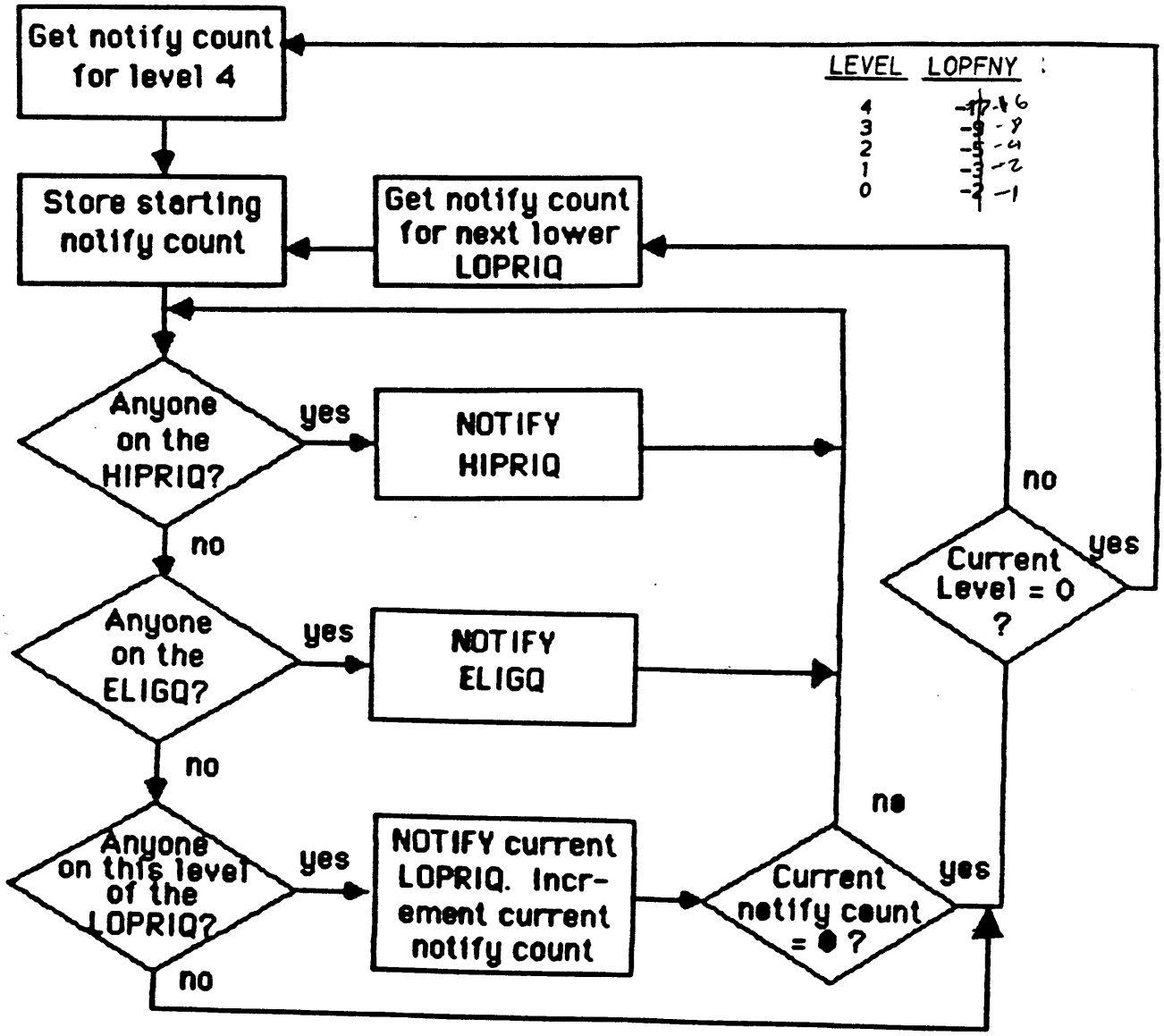
- o The Backstop process is responsible for maintaining the 9 HOLD queues. It will bring one process at a time to the ready list.
- o SCHED responds to a PABORT subroutine call to place a user PCB on one of 9 HOLD queues after it exhausts it's minor time-slice.
- o Here are the 9 HOLD queues:
  - HIPRIQ, high priority (interactive user finishes a command). HITS CR.
  - ELIGQ, eligibility (major time-slice remaining). BUT MINOR TIME SLICE EXPIRED
  - 5 LOPRIQs, low priority (major time-slice exhausted).
    - LOPRIQ 4, user level 4 (supervisor level).
    - LOPRIQ 3, user level 3
    - LOPRIQ 2, user level 2
    - LOPRIQ 1, user level 1
    - LOPRIQ 0, user level 0
  - IDLEQ, will be examined when no other process is holding.
  - SUSPQ, will not be examined.

USERS WAIT ON BUFSEM UNTIL CR.

HOW DOES USE OF * OTHER THAN CR FOR END OF TRAN AFFECT MOVETO HI PRIQ

Backstop Process

LOPRI NOTIFY COUNT



Scheduling Example

HIPRIQ: CPU

ELIGQ:

LOPRIQ 4:

3:

2:

1:

0:

IDLEQ:

Process Exchange and Scheduling Exercise

1. The backstop process:
   - A. Is the same as the dispatcher.
   - B. Is what controls paging on the system.
   - C. Interrupts the CPU when a WAIT instruction is detected.
   - D. Maintains the STLB.
   - (E) None of the above.

2. When in the "wait" circle on the process state diagram, you are waiting for:
   - A. The CPU.
   - B. A major time-slice end.
   - (C) Some system resource.
   - D. The backstop process.
   - E. None of the above.

3. When does the backstop process get executed?
   - A. Right after the clock.
   - B. When it issues a NOTIFY.
   - (C) When the Hold queues are empty.
   - (D) When it's the highest priority process on the ready list
   - E. Every Tuesday, whether we need to or not.

4. The NOTIFY instruction:
   - A. Initiates a DMx request.
   - B. Indicates a cache and STLB miss.
   - C. Notifies the scheduler that a process is in the Hold state.
   - D. Indicates a time-slice end.
   - (E) Can put a process back on the ready list.

5. A Deadly Embrace is:
   - A. More than one process waiting on a semaphore.
   - B. A semaphore with a negative counter.
   - (C) Two processes waiting for resources which the other owns.
   - D. A NOTIFY to an empty system lock.
   - E. None of the above

6. The dispatcher:
   - (A) Is responsible for saving and restoring register sets.
   - B. Issues NOTIFYs to processes in the "wait" state.
   - C. Maintains the Hold Queues.
   - D. Is a software process.
   - E. None of the above.

7. When a processes minor timeslice expires:
   - A. It is put on the ready list.
   - (B) It goes to the ELIGQ.
   - C. It goes to one of the LOPRIQs.
   - D. It goes to HIPRIQ.
   - (E) Either (B) or (C.), depending on the major timeslice.

8. A writer to a system resource:
   - A. Always preempts active readers.
   - B. Always must wait.
   - (C) Waits if there are active readers.
   - D. Can NOTIFY the reader semaphore. HE DOESN'T DO IT.
   - (E) Both C & D are true

9. A PCB does not contain:
   - A. The current remaining minor timeslice.
   - B. The addressing mode that the process is running.
   - C. Flags which show the state of the process.
   - (D) A WAIT instruction.
   - E. A link word to other PCBs.

10. If we have 25 widgets available on the system, and we want to set up a wait list to guarantee that the 26th process to want a widget will wait, we would initialize the counter to a:
   - A. 0
   - (B) -25
   - C. 25
   - D. -1
   - E. none of the above.

Lesson 4 - Direct Memory Transfer Input/Output

Objectives: Upon successful completion of this lesson, students will be able to:

- Describe the Prime implementation of Direct Memory Transfer.
- Explain the concept of I/O bandwidth.
- Explain how burst-mode DMA transfers increase I/O bandwidth.
- Explain DMA overruns

Direct Memory Data Transfers

- On Prime machines, there are two methods employed to transfer data between I/O devices and main memory:

  1) PIO instructions
  2) DMx microcode

- PIO instructions are a group of assembly (PMA) level instructions which can transfer 1 16-bit word to or from controllers, plus perform control operations. These instructions are used primarily for control purposes.

- DMx microcode is used to do bulk data transfers. When a controller signals a DMx request, the CPU will execute a microcode trap. The trap will suspend the currently executing process and begin to execute DMx microcode. To do the transfer, the CPU must know two pieces of information; the location of the data buffer and the amount of data to transfer. This information is typically stored in a "channel".

- There are four types of DMx:

  1. DMA, Direct Memory Access DISK
  2. DMC, Direct Memory Channel
  3. DMT, Direct Memory Transfer
  4. DMQ, Direct Memory Queue

  Each method has advantages and disadvantages in terms of speed, volume and control features and so form a comprehensive range of methods.

I/OTLB Address Translation

- I/OTLB - Since DMx uses virtual addresses when addressing memory, we want to guarantee that the addresses will be pre-translated in order to avoid doing full address translations. The I/O Table Lookaside Buffer is specific for segment 0 and must be initialized before each DMx transfer is started.

DMA Transfer

1. SET UP FOR DMA

2. CONTROLLER READS FROM DEVICE INTO BUFFER

3. ~~WHEN BUFFER IS FULL~~, CONTROLLER RAISES DMx REQUEST
   WHEN THERE IS DATA TO TRANSFER

4. CPU (TRAPS TO DMx U-CODE) ACKNOWLEDGES CONTROLLER

5. CONTROLLER SHIPS CHANNEL LOCATION TO CPU

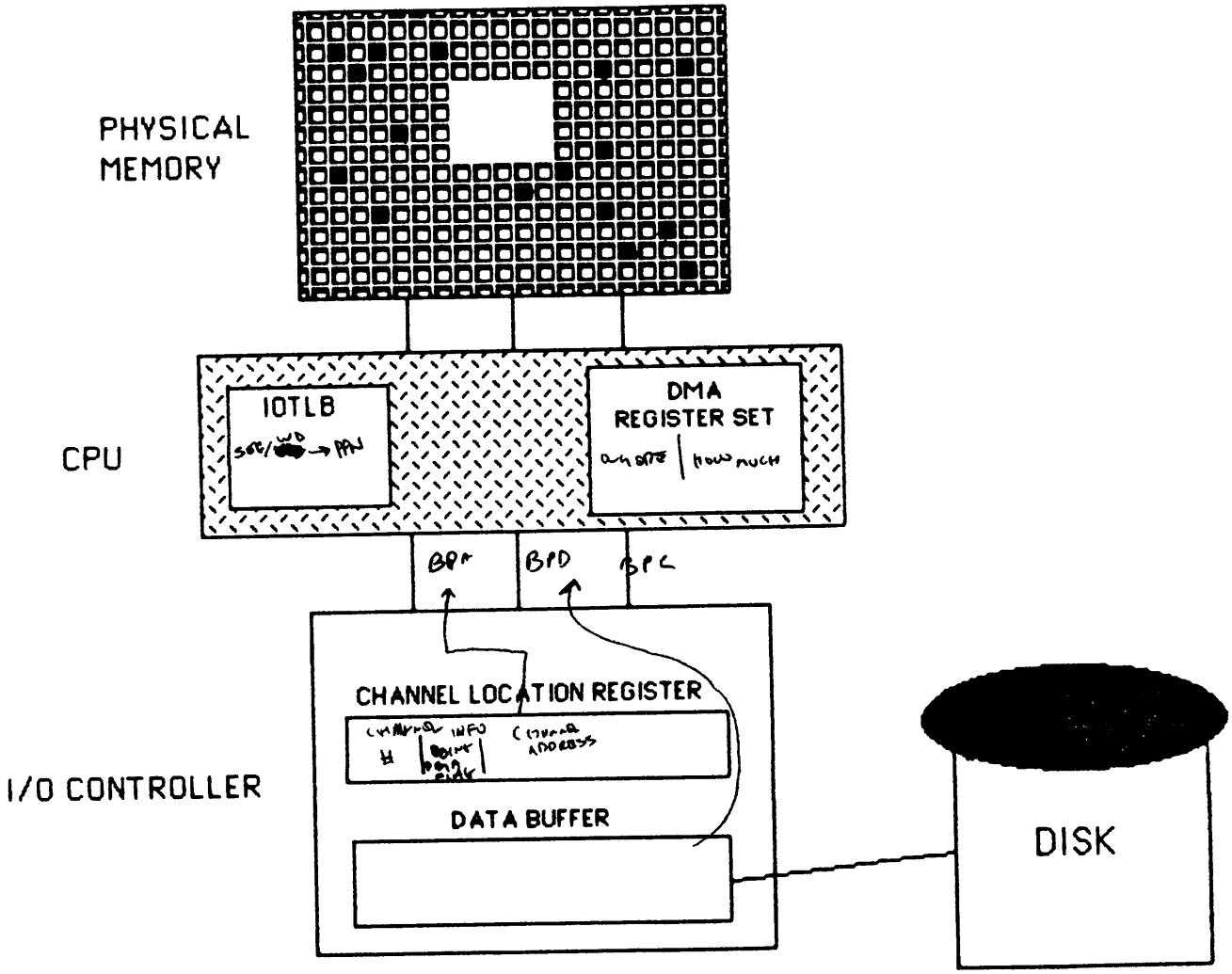
6. CONTROLLER SHIPS 16-BIT WORD OF DATA TO CPU

7. CPU SHIPS UP TO 'WHERE'; CHECKS 'HOW MUCH'

8. IF DONE, SENDS END OF RANGE SIGNAL TO CONTROLLER, IF NOT STEP 6

9. CONTROLLER SENDS EXTERNAL INTERRUPT SIGNAL TO CPU TO DO NOTIFY

DMA Flow


PHYSICAL MEMORY
CPU
IOTLB
DMA REGISTER SET
BPA
BPD
BPC
I/O CONTROLLER
CHANNEL LOCATION REGISTER
DATA BUFFER
DISK

DMx Type Information

DMA Transfers

Uses:
- disk data transfers
- tape transfers of less than 4096 16-bit words
- PNC controllers

Advantages:
- faster than DMC or DMQ
- you can chain channels together (scatter-gather)
- has the highest bandwidth (using burst mode)

Disadvantages:
- only 32 channels available
- maximum of 4096 16-bit words per channel
- data buffer must be in segment 0

DMC Transfers

Uses:
- MDLC and SMLC controllers
- AMLC and QAMLC for character input
- tape transfers of more than 4096 16-bit words
- MPC controllers (parallel printers)

Advantages:
- faster than DMQ
- large number of channels available
- you can chain channels together
- 64KW maximum transfer size (theoretical limit)

Disadvantages:
- slower than DMA and DMT
- data buffer must be in segment 0

DMx Type Information (cont.)

DMQ Transfers

Uses:
- QAMLC for character output
- ICS1, ICS2, and ICS3 for async character input and output

Advantages:
- can read and write from the data buffer simultaneously
- data buffer can be in any segment
- buffer can be up to 64KW in size

Disadvantages:
- slowest of all DMx methods
- data buffer must be a power of 2 in size

DMT Transfers

Uses:
- outputting disk channel programs to the controller
- AMLC for character output
- ICS1, ICS2, and ICS3 for downline loading microcode

Advantages:
- Fastest of the DMx methods

Disadvantages:
- no channel (controller must "control" transfer)
- data buffer must be in segment 0

Lesson 5 - Processor Features

Objective: Upon successful completion of this lesson, students will be able to:

- Describe the major features of the various processors and how they relate to overall system performance.

Common Processor Features

- Multi-user multi-function timesharing systems.
- Microprocessor control unit with process exchange.
- Multiple user register sets.
- 32 bit architecture.
- 255 user processes.
- 512 MB virtual address space.
- Segment Table Lookaside Buffer (STLB).
- Hardware integer arithmetic.
- Cache memory.

2250

- PRIME's entry level system.
- Designed for the office environment.
- Easy-to-use operator interface.
- Microcode implementation of floating point and decimal/character business instructions.
- Slow system clock rate.
- 20% slower disk transfer rate.
- Limited configurability
  - 10 total slots
    - 2 CPU boards
    - Maximum 4 MB Main Memory
    - Maximum 32 users
    - 1 disk/tape controller board
    - 1 ICS1 controller
    - 3 optional slots

2350, 2450, 2655 & 9655

- Custom gate array TTL logic.
- Larger 16 KB cache.
- Wide-word memory.
- Burst mode I/O (microcode based, 5.0 MB/sec. bandwidth).
- 8 user register sets.
- 512 STLB entries, 128 IOTLB entries.
- 2 stage instruction pipeline
- Decimal arithmetic hardware.
- Quad precision floating point hardware.
- 48 bit floating point ALU.

2350 & 2450 Configurability

2350
- 4 I/O controller maximum
- Maximum 8 MB main memory
- 2 board CPU (3 slots)
- Up to 16 terminal users
- Maximum 4 synchronous lines.
- Limited to 2 disk drives (240MB).

2450
- 4 I/O controller maximum
- Maximum 8 MB main memory
- 2 board CPU (3 slots)
- Up to 24 terminal users
- Maximum 4 synchronous lines.
- Limited to 2 disk drives (240MB).

2655 & 9655 Configurability

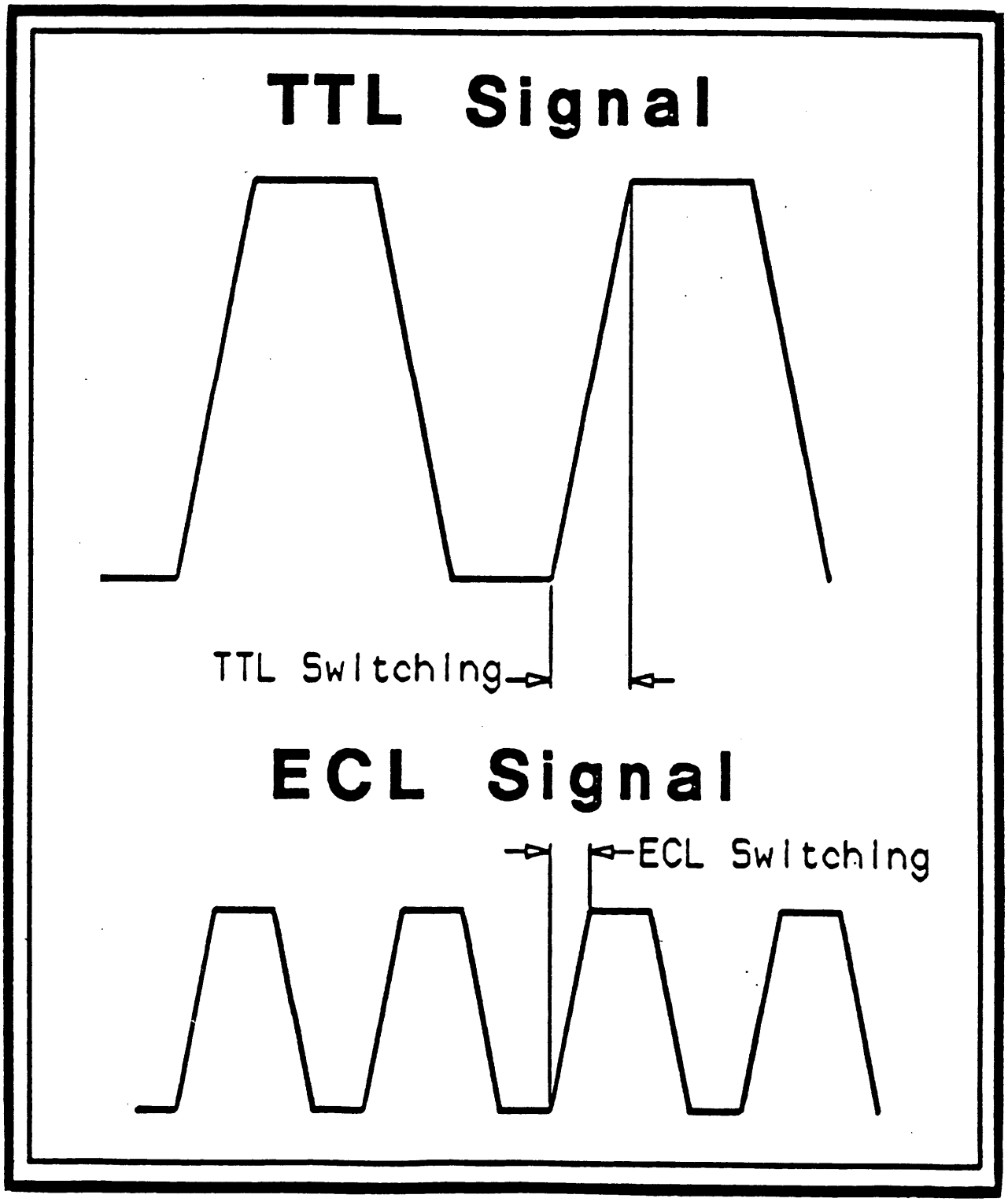
2655

- 7 I/O controller maximum
- Maximum 8 MB main memory
- 2 board CPU
- Up to 64 terminal users
- Maximum 8 synchronous lines.
- Limited by single 130 amp. power supply.
- Limited to 2 disk subsystems (4 drives, 1.2GB).

9655

- 10 I/O controller maximum.
- Maximum 8 MB main memory.
- 2 board CPU.
- Up to 128 terminal users.
- Maximum 8 synchronous lines.
- Limited by 130 amp. I/O power supply.
- Up to 4 disk controllers (16 drives, 10.5GB)

TTL vs ECL



ECL Processor Features

- Synchronous Pipeline
  - 10 stages.
  - every other stage is occupied.
  - each stage takes 40 nanoseconds to complete (beat rate).

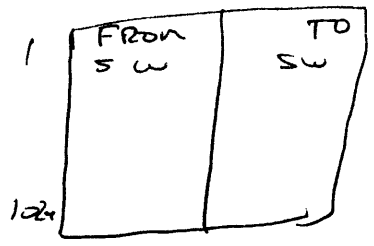
- Branch Cache
  - Record the target address for jump and branch instructions.
  - Predict the next instruction address for the pipeline.
  - 256 entries.

9750, 9755 & 9950

- CPUs uses ECL logic.
- Dedicated CPU backplane.
- 10 (5) stage synchronous pipeline.
- Branch cache (256 entries).
- Quad precision floating point hardware.
- Environmental sensors to detect, and take action, if overheating occurs.
- 4 user register sets.
- 40 ns. access time for cache, STLB and registers.
- 128 STLB and IOTLB entries.
- 9750 configurability
  - Maximum of 12 MB main memory
  - 10 I/O controller support
  - Maximum 192 terminal users
- 9755 configurability
  - Maximum of 16MB main memroy
  - 10 I/O controller support
  - Maximum 192 terminal users
- 9950 configurability
  - Maximum 16 MB main memory
  - 14 I/O controller support
  - Maximum 254 terminal users

9955 & 9955-II

- CPU uses ECL logic.
- 64 KB cache (98% hit rate).
- Branch cache (1024 entries).
- 512 STLB and IOTLB entries.
- Quad precision floating point hardware.
- Environmental sensors.
- Multiplier Array board.
- Soft Error Recovery (cache, lookaside buffers).
- 9955 configurability
  - Maximum of 16 MB main memory
  - 14 I/O controllers
  - 254 terminal users
- 9955-II configurability
  - Maximum of 32 MB main memory
  - 14 I/O controllers
  - 254 terminal users



Title: 50-Series Processor Features Homework

Objective: Upon successful completion of this lesson, students will be able to:

- Describe the features of the various processors.

Task: Fill in a table of all processor features

Fill in the correct value for each entry in the table.
Choose from the values listed in square brackets "[ ]."
You do not have to use all of the choices.

	2250	2550	9650	9750	9950	9955
n-bit architecture [32/16 bits]						
Simultaneous active processes [64, 128, 255]						
Direct connect terminal users [32, 48, 64, 96, 128, 175, 196, 254]						
Maximum main memory [2, 4, 6, 8, 12, 16 MB]						
STLB size [64, 128, 256, 512]						
Cache size [2, 4, 8, 16, 32, 64KB]						
I/O bandwidth [2, 2.5, 5, 5, 8, 9 MB/S]						
Burst mode I/O [yes/no]						
Wide-word memory [yes/no]						
branch cache [yes, no]						
circuit type [ECL, TTL, gate array]						
user register sets [1, 2, 3, 4, 8]						
pipeline stages [0, 1, 2, 3, 4, 5, 10]						
Integer arithmetic [Hardware/Firmware]						
Character/Decimal [Hardware/Firmware]						
Floating Point [Hardware/Firmware]						
Procedure Call [Hardware/Firmware]						
Process Exchange [H/F/Software]						
Quad precision [H/F/Software]						

EXTRA CREDIT

Do the same for the following old models:

550-II, 750, 850, 250-II, 550-I, 250-I, 500, 400

Lesson 6 - Disk Input/Output

Objectives: Upon successful completion of this lesson, students will be able to:

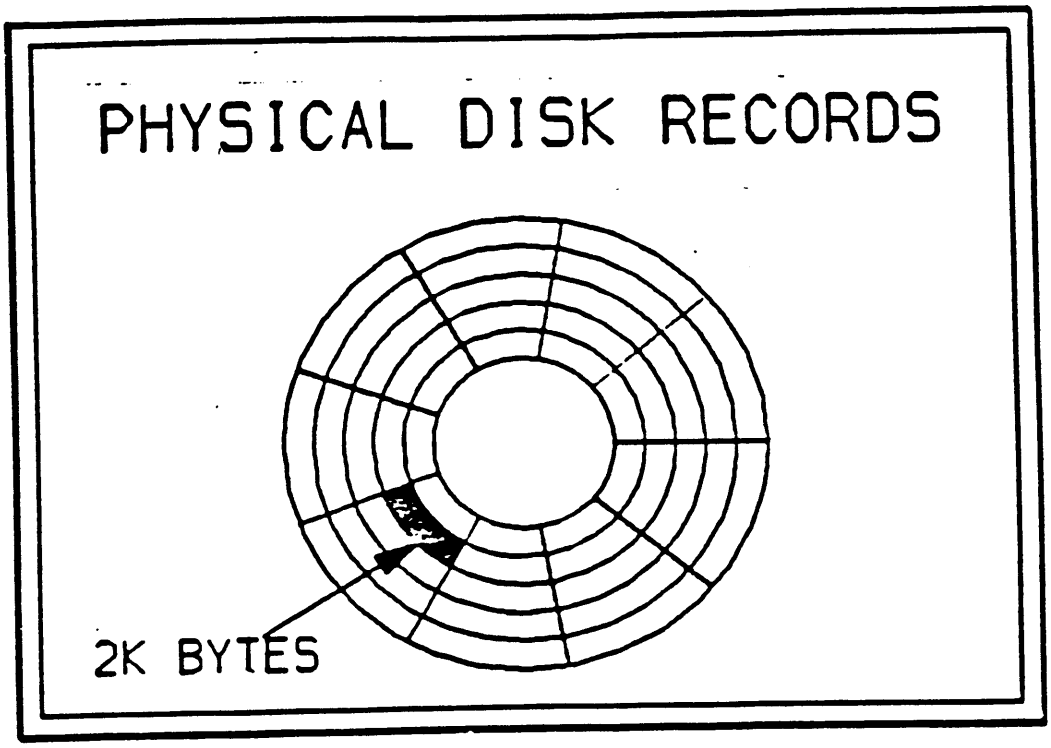
- Describe the basic layout of the disk subsystem hardware.
- Describe the various components of disk I/O time.

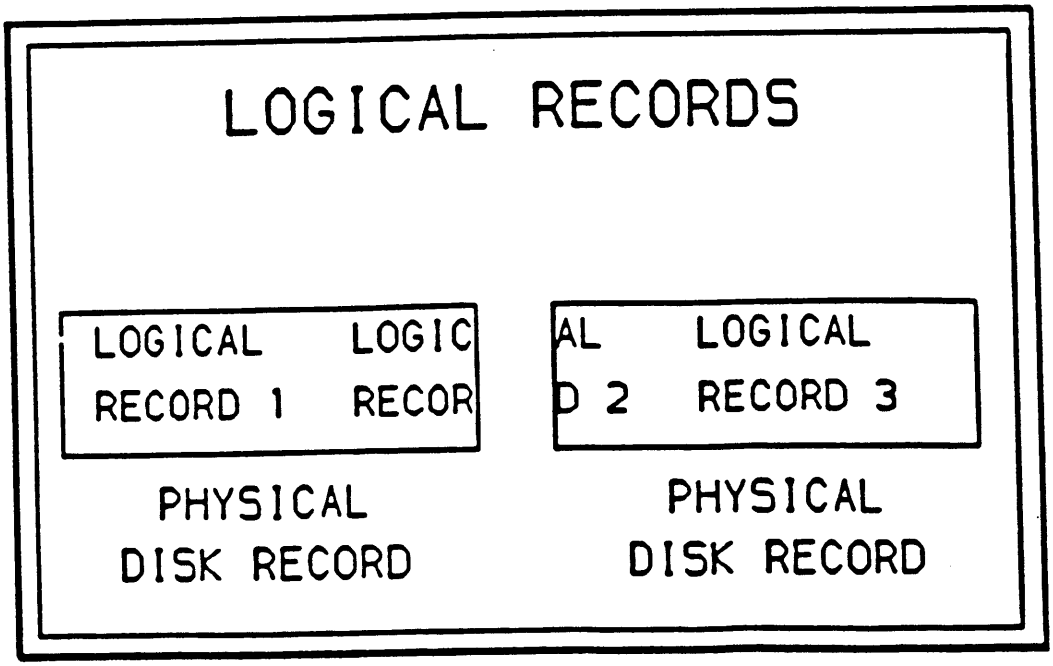
Disk Concepts

- The I/O bus connects the CPU to one to four disk controllers (maximum two prior to revision 19.3).
- Each disk controller controls one to four disk drives.
- Each disk drive has a disk data pack and heads.
- The disk data pack consists of a number of platters on a spindle.
- A logical disk contains a number of adjacent platters.
- There is at least one head for each platter surface.
- Each platter is divided into a number of concentric tracks.
- Each track is divided into 9 records.
- Each record magnetically encodes 1040 words of data. + 32 BYTE HDR
- The disk drive spins the disk pack at about 3600 rpm.
- All of the heads as a single unit mechanically seek a desired cylinder.
- The head at the desired record reads/writes the data.

Disk Concepts

2048
32





Disk I/O Time

- The total time it takes to process an I/O request is described by the following formula:

  Disk I/O time = wait time + seek time + latency time + transfer time

- Wait time is the amount of time it takes from when a process submits a request until it is acted upon by the disk controller. There are two major things you may have to wait for:

  1) To get a Queue Request Block (QRB). There are:
     - 7 QRBs at Rev 18
     - 17 QRBs at Rev 19.1
     - 32 QRBs at Rev 19.3 and on

  2) The other processes in the work list for a particular drive which are ahead of you.

- Seek time is the amount of time it takes once the controller gets a request for the heads to get "on cylinder". A random seek takes about 40-45 ms. The average seek time can be reduced from this amount by two methods:

  1) Ordering seeks. This is a method of ordering request in ascending order (track #).

  2) Overlapped seeks. A controller can have all drives simultaneously seeking.

Disk I/O Time con't

Disk I/O time = wait time + seek time + latency time + transfer time

- Latency time is the amount of time it takes the disk to rotate into position once the heads are on cylinder. This is strictly a function of the disk drive.

- Transfer time is the amount of time it takes (using DMx) to transfer one disk record into memory.

  It is possible to have more than one controller transfering a data record at the same time. This is called overlapped transfers.

Lesson 7 - The LOCATE Mechanism (Associative Buffers)

Objectives: Upon successful completion of this lesson, students will be able to:

- Describe the Locate mechanism and where it fits in to the scheme of the disk I/O mechanism.
- Describe the process of a disk request from start to finish.

Associative Buffers

- An associative (or LOCATE) buffer is a main memory copy of a disk record.

- Associative buffers are a means of reducing the number of disk accesses needed for logical file access.

- Multiple logical reads to one physical record may require only one disk access.

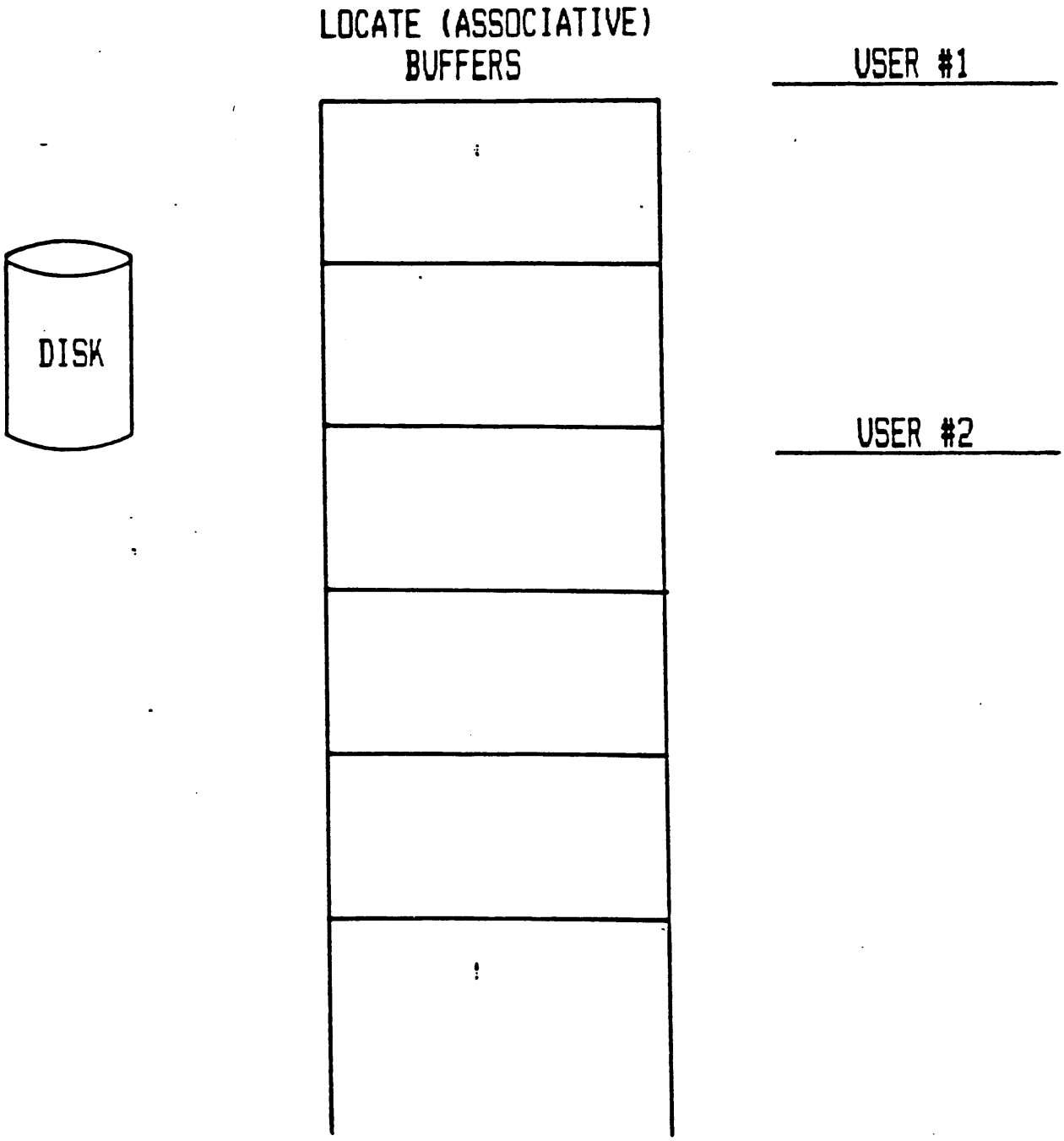
- Multiple logical writes to one physical record may require only one disk read and one disk write.

- Each user can own one locate buffer. An owned locate buffer is wired in memory.

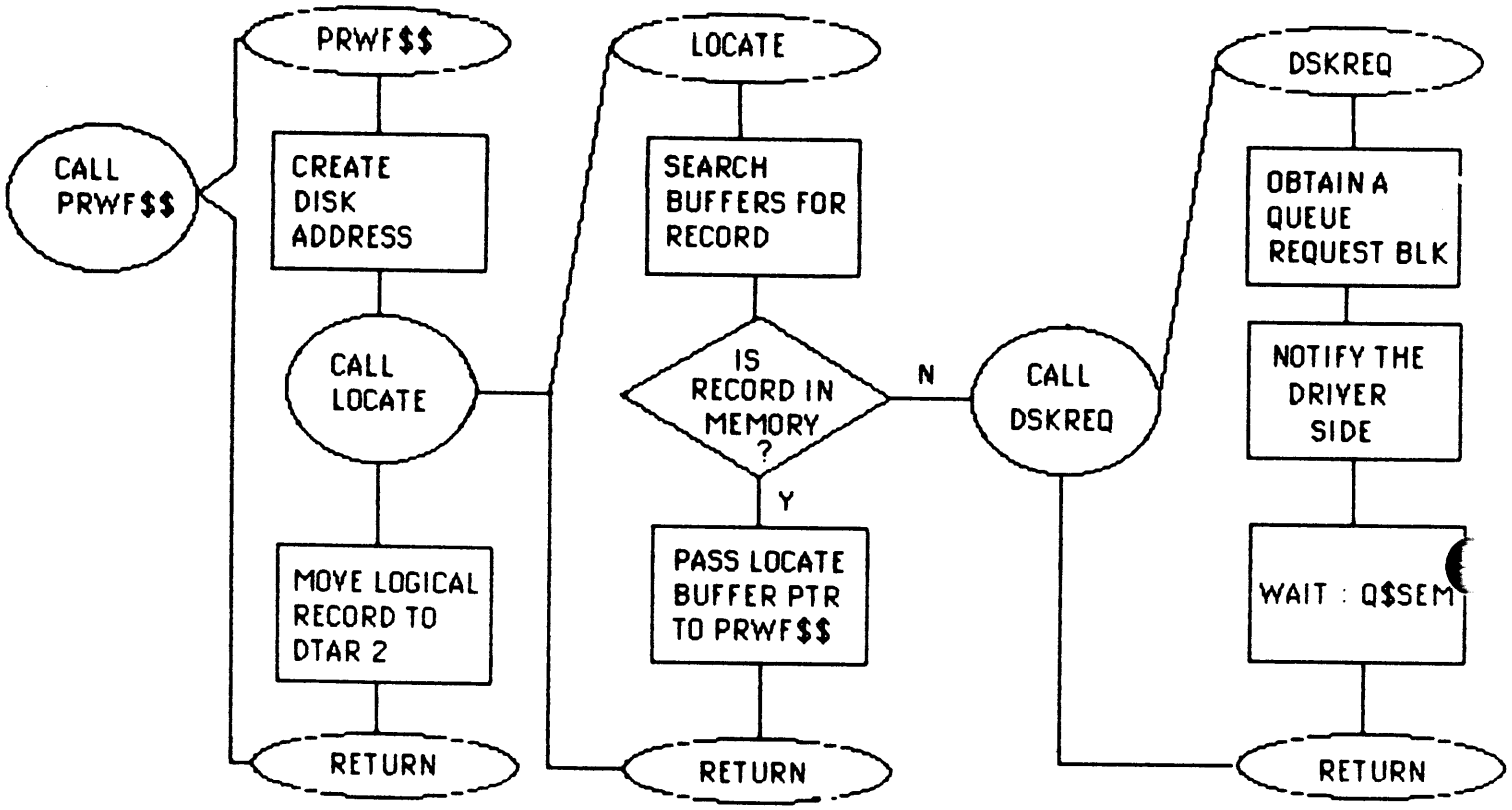
- Previously owned locate buffers remain in memory until they are again owned (wired), or deleted from memory.

- If a locate buffer has been modified, it is written back to the file system disk by user 1 and/or when it is deleted from memory. User 1 copies all modified locate buffers to the file system disk once a minute.

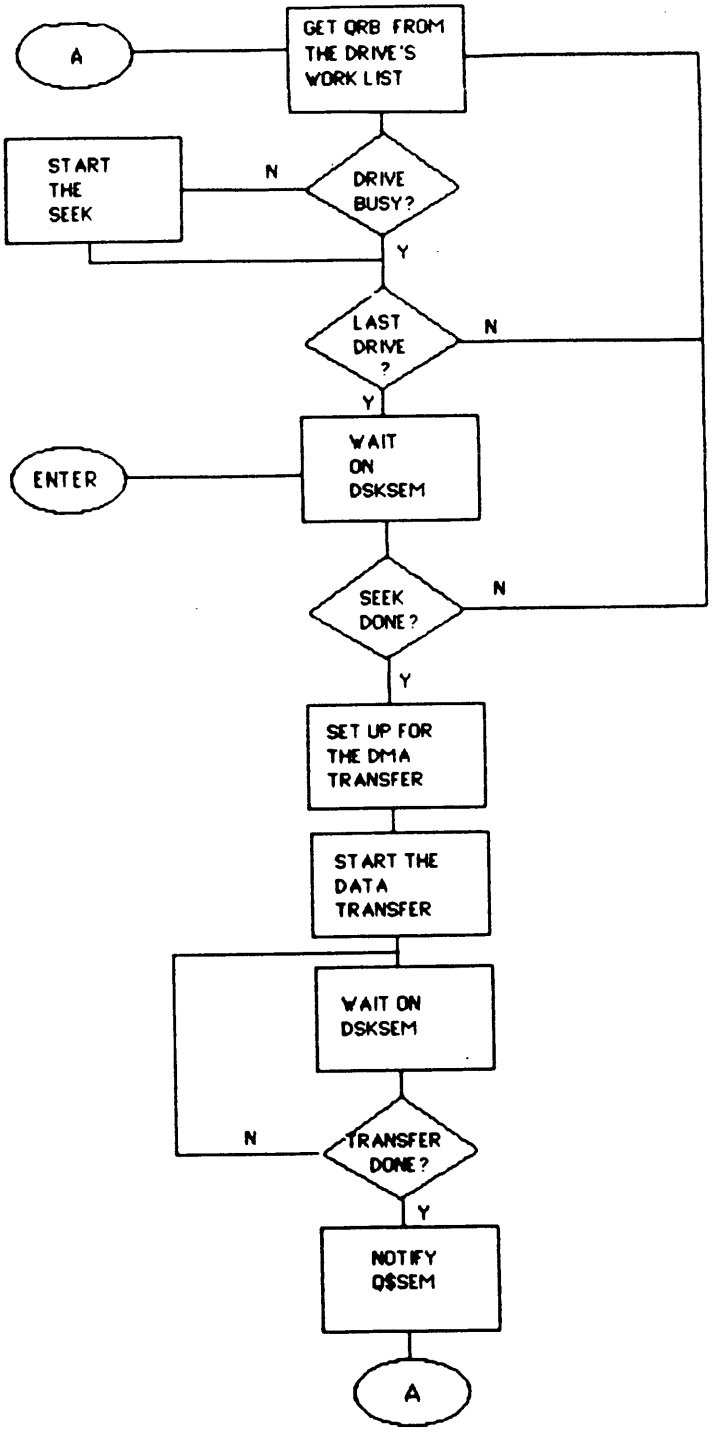
LOCATE Mechanism



File I/O


CALL PRWF$$
PRWF$$
CREATE DISK ADDRESS
CALL LOCATE
MOVE LOGICAL RECORD TO DTAR 2
RETURN
LOCATE
SEARCH BUFFERS FOR RECORD
IS RECORD IN MEMORY ?
N
Y
PASS LOCATE BUFFER PTR TO PRWF$$
RETURN
CALL DSKREQ
DSKREQ
OBTAIN A QUEUE REQUEST BLK
NOTIFY THE DRIVER SIDE
WAIT : Q$SEM
RETURN

The Driver


A
GET QRB FROM THE DRIVE'S WORK LIST
START THE SEEK
N
DRIVE BUSY?
Y
LAST DRIVE ?
N
Y
ENTER
WAIT ON DSKSEM
SEEK DONE?
N
Y
SET UP FOR THE DMA TRANSFER
START THE DATA TRANSFER
WAIT ON DSKSEM
N
TRANSFER DONE?
Y
NOTIFY Q$SEM
A

Lesson 8 - The File System

Objectives: Upon successful completion of this lesson, students will be able to:

- List the various types of data structures on disk.
- Describe what a directory looks like and how it works.
- Describe the various ways of organizing data files.
- Describe what unit tables are and how they are used.
- Describe how quotas are implemented.

Physical Disk Structures

A disk drive is divided into one or more partitions where a partition is one or more pairs of heads. Each partition must contain:

1). MFD (Master file directory)

2). DSKRAT (Disk record availability table)

3). BOOT (For initial loading)

4). UFD DOS (Initially empty - not actually required)

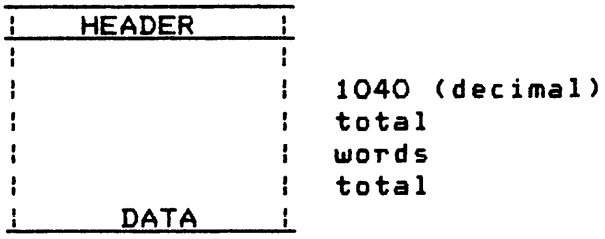
5). UFD CMDNCO (Initially empty)

6). BADSPT (If badspots on the disk)

Each partition is divided into a number of 1040 word records.

The record header is 16 words for storage module devices.

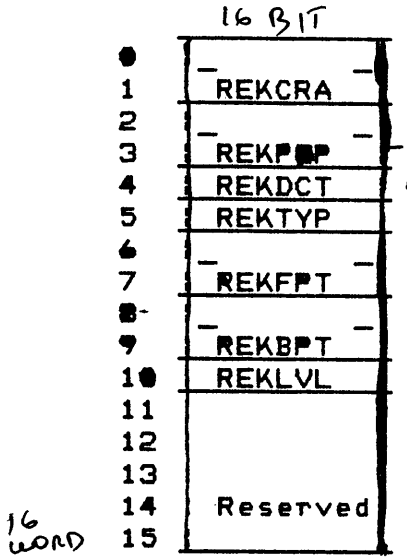
The remainder of the record holds data (1024 words).



Logical Disks

- Master File Directory (MFD)
  - is the top level directory.
- DiSK Record Availability Table (DSKRAT)
  - is created by MAKE, and patched (if necessary) by FIX_DISK.
  - has a bit to indicate the status of every record in the partition, in use, or free.
  - linked records are assigned on the same cylinder when possible.
  - should be as large a practical
- The BADSPoT file (BADSPT)
  - will be created by MAKE either by manual entry, test, or from a pre-existing BADSPT file.
  - will hold re-located records detected by COPY_DISK when a badspot is encountered.
  - holds all badspots for entire physical disk.
  - have MAKE conduct the most severe test (takes a long time).
  - FIX_DISK can be used to add badspots.

Record Header Format



Record address of this record.

RA of the directory entry for this record, or the first record in the file.

Number of data words in record.

Type of file (only on first record).

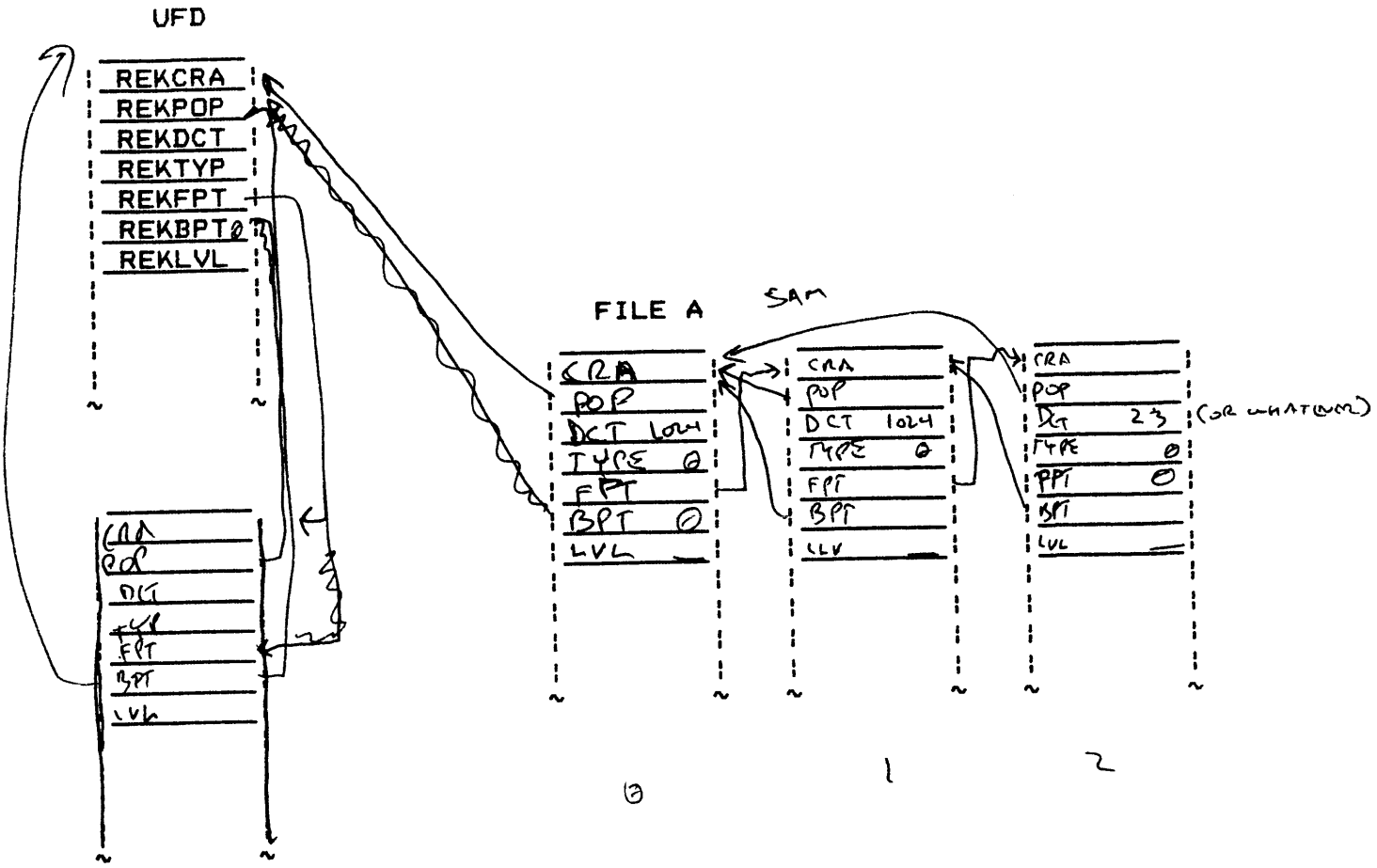
RA of the next sequential record, or a 0 if the last record.

RA of the previous record, or a 0 if the first record.

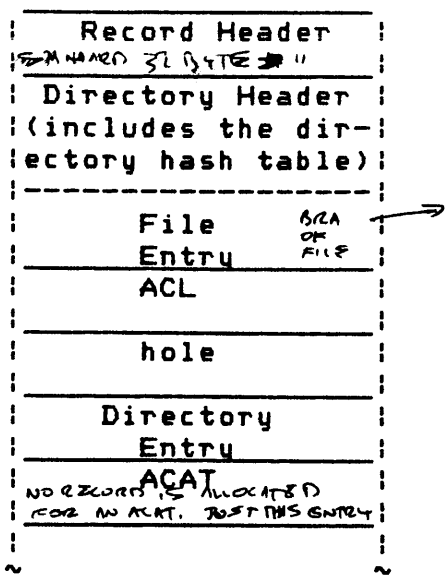
Index level for DAM files.

REKTYP 0 = SAM
1 = DAM
2 = SEGSAM
3 = SEGDAM
4 = UFD
5 = UFD (ACL)
6 = ACCESS CAT
7 = CAM FILES

Disk Record Logical Structure



Directory Structure



- DIRECTORY HEADER
  - Password.
  - Quota information.
  - Date/time stamp.
  - Directory hash table (127 16-bit words)

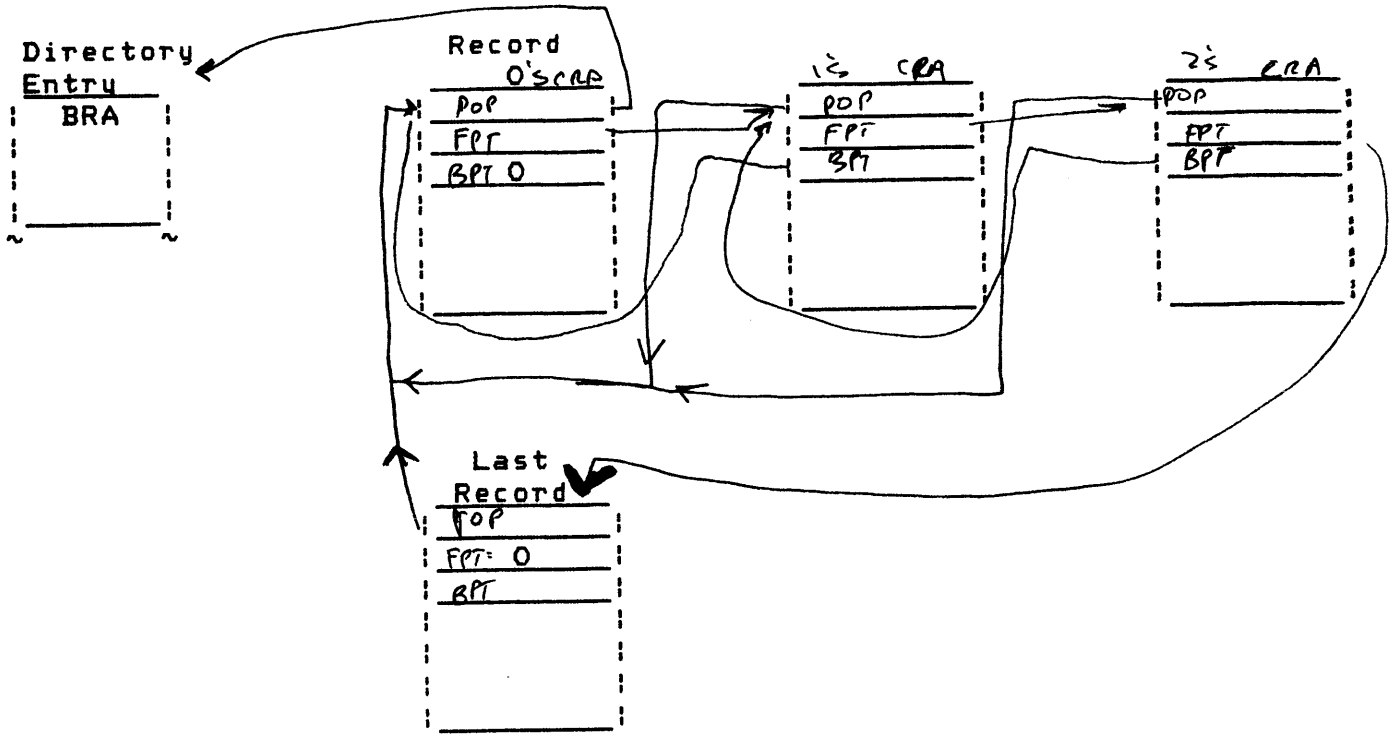
- FILE ENTRY
  - Pointer to first record of file (BRA).
  - Protection information (password protection keys, ACL position, RBF flag, etc).
  - Integrity information (date/time last saved, read/write locks, truncated flag, etc).
  - Type of file (SAM, DAM, SEGSAM, SEGDAM, SUBUFD)
  - File or directory name

- ACL ENTRY
  - Access pairs.

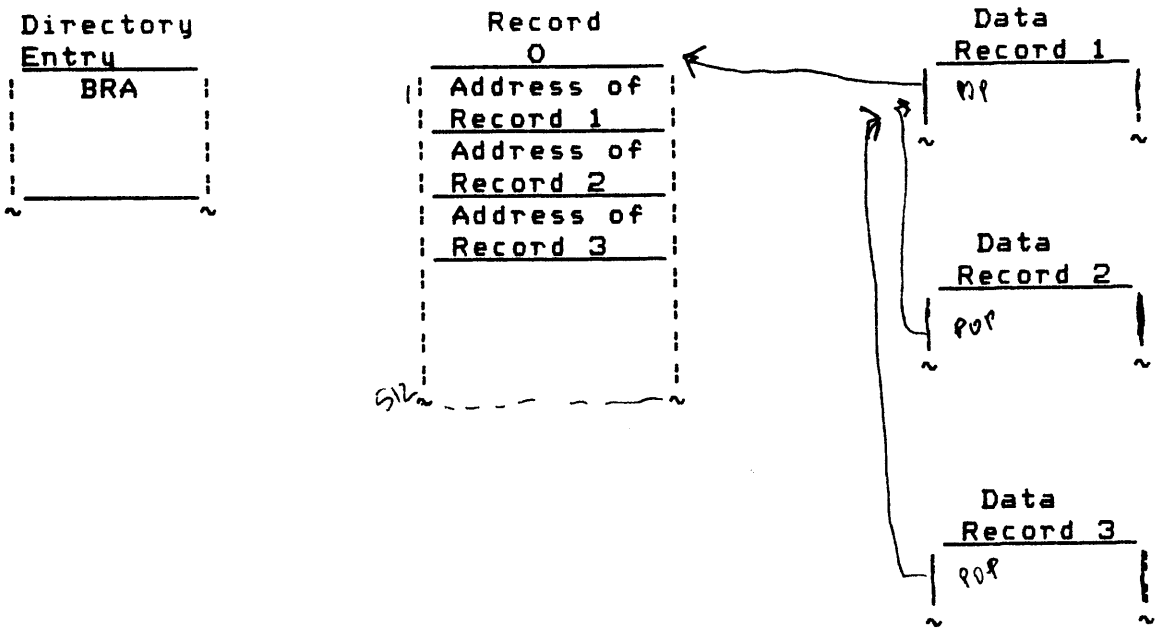
- ACAT ENTRY
  - Name of ACAT.
  - Pointer to ACL (within directory).

- HOLE (VACANT ENTRY).
  - Caused by deletion of file object.
  - Will be re-used if new entry fits.
  - Eliminated by FIX_DISK -UFD_COMPRESSION.

SAM Files

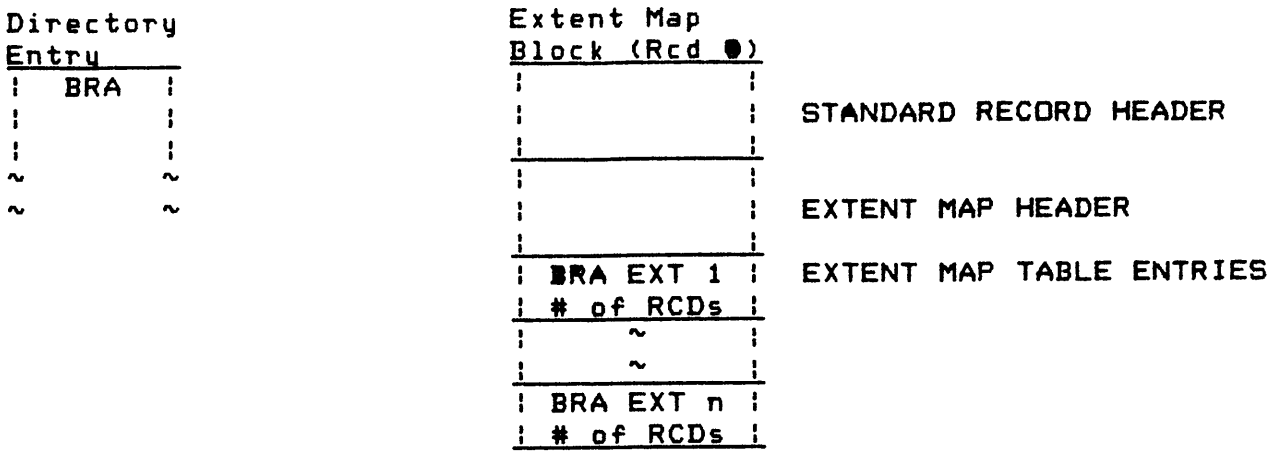

Directory
Entry
BRA
Record
0's CRA
POP
FPT
BPT 0
1's CRA
POP
FPT
BPT
2's CRA
POP
FPT
BPT
Last
Record
POP
FPT: 0
BPT

DAM Files



WHEN MORE THAN 512 DATA RECORDS ARE ALLOCATED, A SECOND PRIMARY RECORD IS ALLOCATED AND A NEW INDEX RECORD IS CREATED AT THE NEXT LEVEL UP, WHICH POINTS TO THE TWO INDEX RECORDS THAT POINT TO DATA RECORDS

CAM File

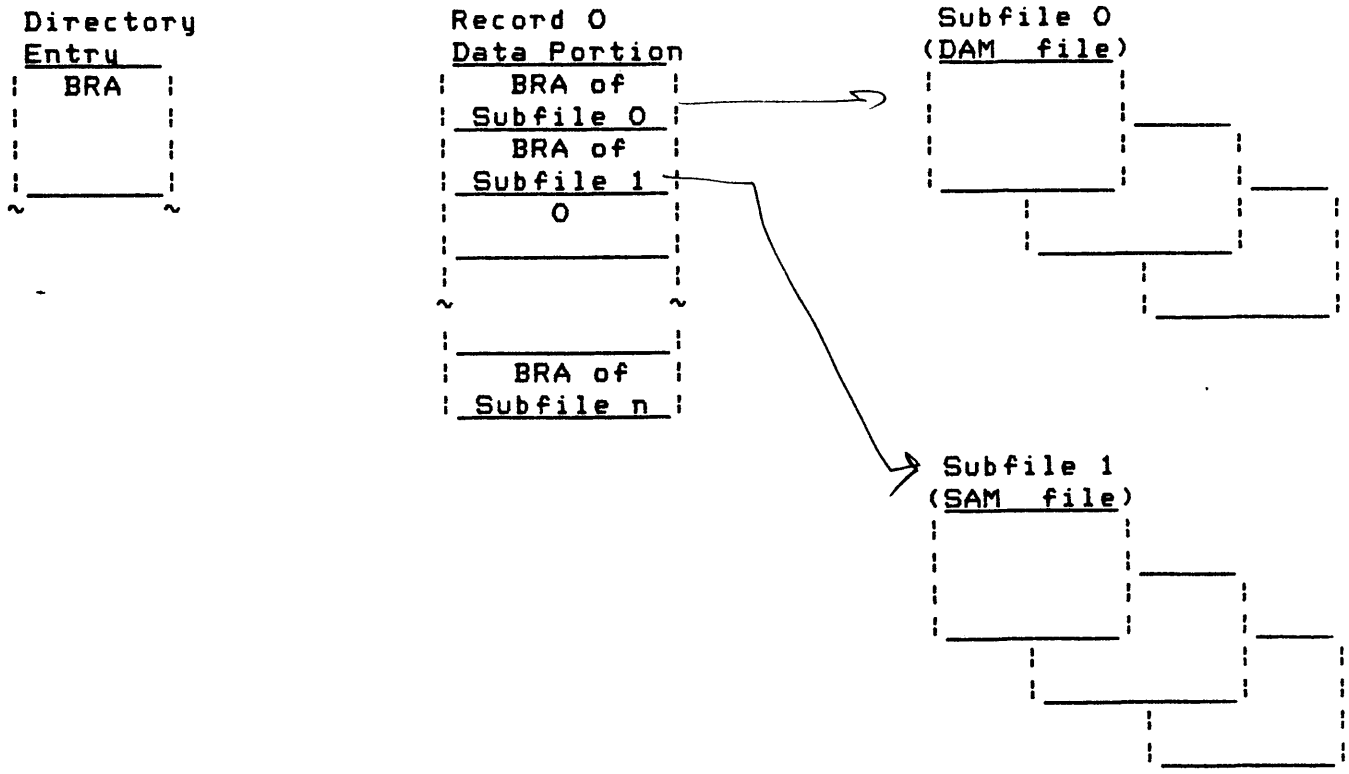


Ext 1	Ext 1	Ext 1	Ext n	Ext n	Ext n
Rcd 1	Rcd 2	Rcd 3	Rcd 1	Rcd 2	Rcd 3

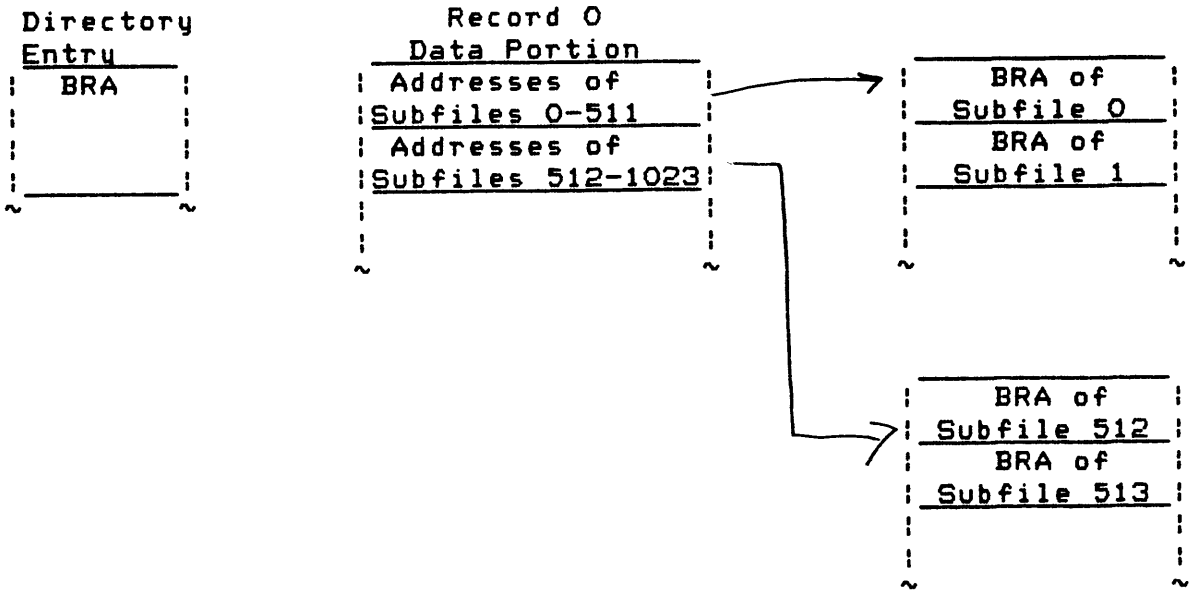
16 REC/EXTENT MAX

340 EXTENTS MAX PER EXTENT MAP BLOCK

SEGSAM Directory Format



SEGDAM Directory Format



Unit Tables - Definitions

- A unit table (ut) is a list of pointers to unit table entries.

- A unit table entry (ute) describes a file system object that is currently in use via the file system. It contains:
  - The current disk address of the record we last accessed.
  - The parent directory address.
  - Access rights.
  - Read/Write locks.
  - Current logical position in file.
  - Quota pointers.
  - Misl info.

- A file system object is a data file, directory or access category. These objects may reside on a local or a remote system.

Unit Tables - Rev 19.4

PRE-19.4 METHOD

- Per-User unit tables allocated/deallocated dynamically.
- Maximum of 131 units per user.
- 8 units guaranteed per user.
- Maximum of 3247 system units available.
- Unit table is same size no matter how many active units.
- At login, get 131 file units:

```
      0 system unit
1 - 127 available for user
    128 home
    129 current
    130 IAP
```

19.4 METHOD (and on)

- Per-user unit tables allocated/deallocated dynamically.
- Maximum of 32,768 units per user.
- Users are guaranteed all the units they want.
- Maximum of 256,000 system units available.
- Unit table dynamically grows as more file units are requested.
- Initially, (at login) get 38 file units:

```
   -5 temporary attach
   -4 como
   -3 IAP
   -2 home
   -1 current
    0 system
 1-32 available for user
```

Disk Quotas

- Quotas are implemented by the use of two data structures:

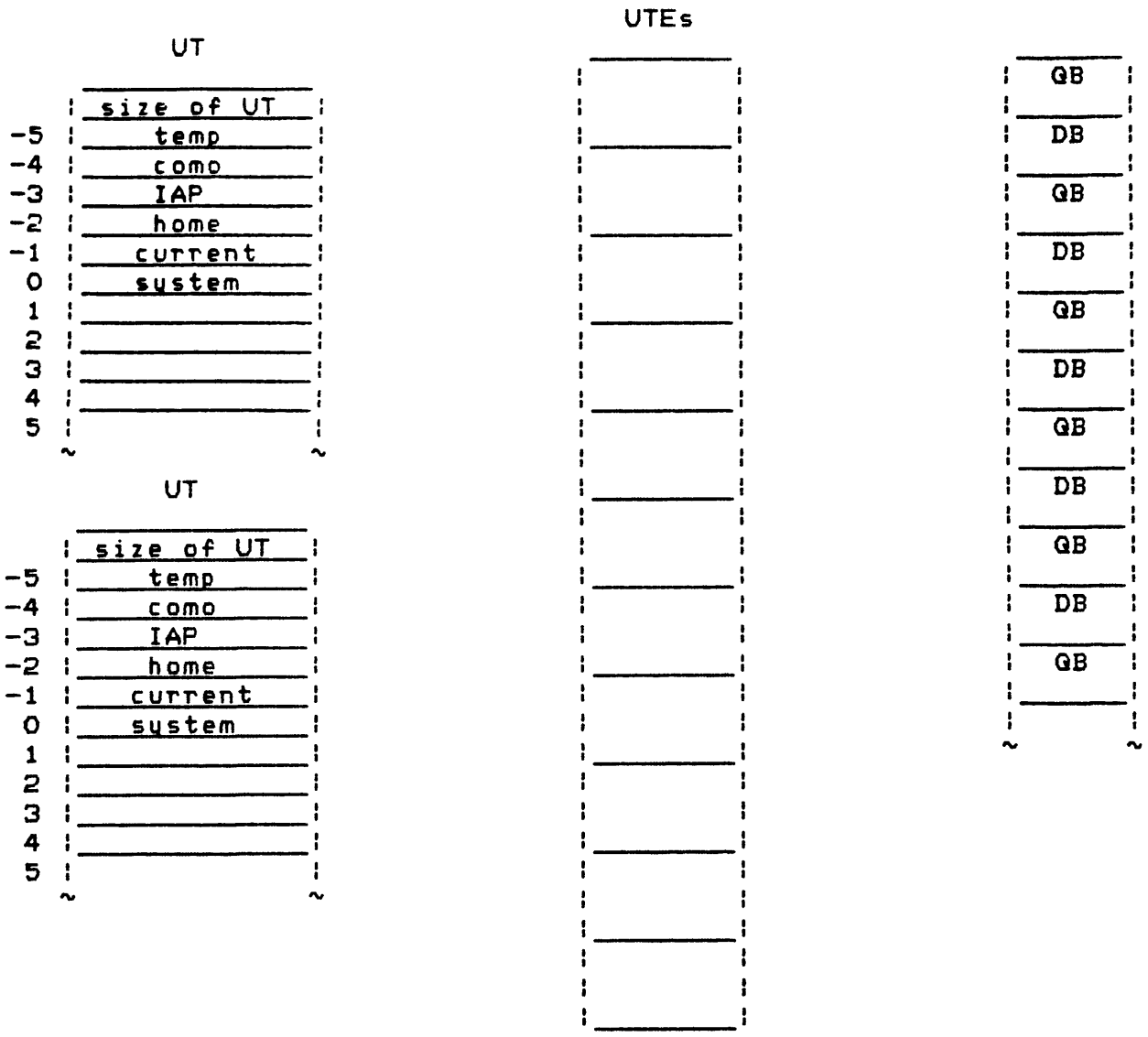
  DIRECTORY BLOCK (DB)
  - User count (how many people are using this directory).
  - BRA of directory.
  - Quota modified flag.
  - Number of records used in this directory.

  QUOTA BLOCK (QB)
  - User count.
  - BRA of directory.
  - Pointer to parent UFD's QB.
  - Quota left in tree.

  Quota information is stored in these two structures as long as anyone is accessing a particular directory. When the directory not in use, this information is stored in the directory header on disk.

- Quota information is ONLY updated when the last user leaves a directory. Thus overhead from quotas is very small.

Unit Table Allocation



Disk, File System and LOCATE Exercise

1) Logical disks should be as large as possible because:
   A. there are more records per cylinder in large partitions.
   B. there are more cylinders per surface in large partitions.
   C. there are more records per surface in large partitions.
   D. none of the above.

2) BADSPT (badspot) files:
   A. are not related to system performance.
   B. should contain every disk record that has ever had an error.
   C. need contain only bad disk records that are detected by MAKE.
   D. both (A) and (B) are true.

3) Programs will take the most advantage of the locate buffering mechanism if they
   A. have small sequentially processed logical records.
   B. process data in sequential disk records.
   C. only read from or only write to (not both) a disk record.
   D. all of the above.

4) A locate (associative) buffer is:
   A. a collection of pointers to a disk record.
   B. a main memory copy of a disk record.
   C. an area in cache set aside for disk I/Os.
   D. wired in memory until a user logs off.

5) UFDs:
   A. are strictly main memory structures.
   B. contain file entries, ACLs, ACATs, directory entries, and holes.
   C. are limited to one disk record in size.
   D. do not have a standard disk record header.

6) Unit tables:
   A. are accessed when opening a file.
   B. are always accessed through a hash.
   C. are pointed at by Quota Blocks and Directory Blocks.
   D. contain Unit Table Entries.

7) A physical disk record is:
   A. 1024 decimal 16-bit words.
   B. 1040 octal 16-bit words.
   C. 1024 octal 16-bit words.
   D. 1040 decimal 16-bit words.
   E. As long as the application requires it to be.

8) SMDIO:
   A. Is another name for the LOCATE mechanism.
   B. Works only with DAM files
   C. Sets up for DMA.
   D. Is the disk driver.
   E. Both C & D are true.

9) The only thing that actually resides in a directory record is:
   A. An ACL.
   B. A file
   C. A directory
   D. None of the above
   E. All of the above

10) The DSKRAT is:
   A. Another name for the I/O driver.
   B. A record header.
   C. A bit map of the data records on the partition.
   D. Located in every UFD on a partition

Lesson 9 - The Program Environment

Objectives: Upon successful completion of this lesson, students will be able to:

- Define the four basic addressing modes on Prime.
- Describe the PCL mechanism, including stacks, base registers, and ECBs.
- Describe how SEG loads programs in memory.
- Describe the differences between SEG and EPFs, plus describe the other advantages in the implementation of EPFs.

Addressing Modes

16S - 16 stands for 16KW maximum address space.
- S stands for Sector mode (current and sector zero).
- Uses absolute physical addresses.
- Honeywell compatable mode.
- Prime 200-9950.
- Store instructions automatically flush 9950 pipeline.

32S - 32 stands for 32KW maximum address space.
- Same as 16S, but only allow one level of indexing.

32R - 32 stands for 32KW maximum address space.
- R stands for Relative mode (relative to PC, sector zero).
- Prime 300-9950
- Store instructions automatically flush 9950 pipeline.

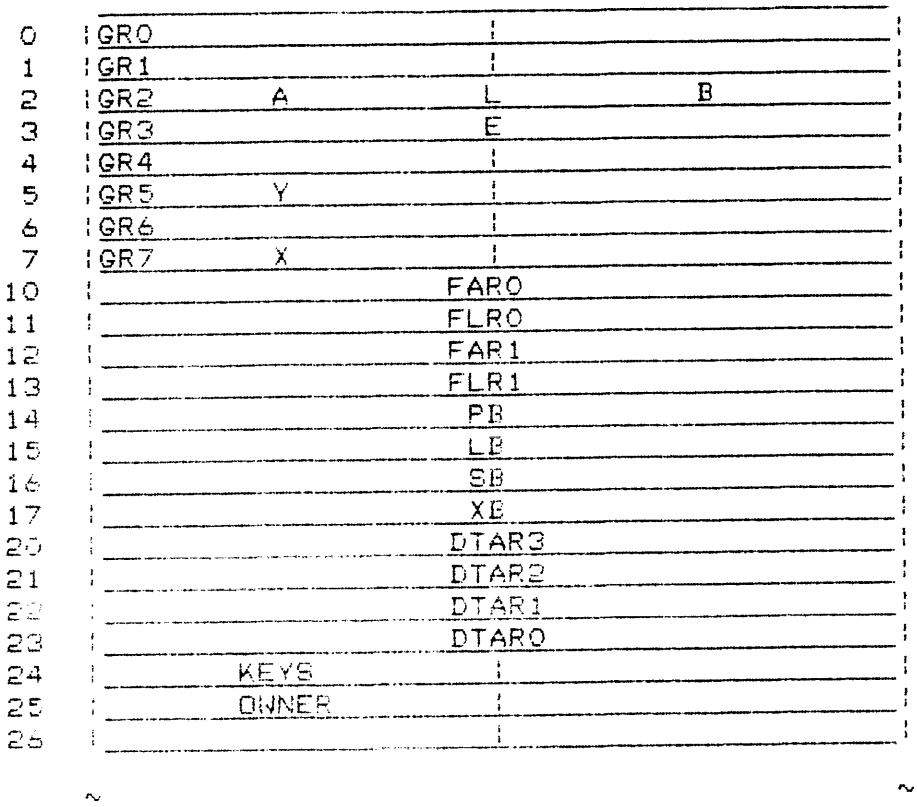
64R - 64 stands for 64KW maximum address space.
- Same as 32R, but only allow one level of indirection.

64V - 64 stands for 64KW address space per segment.
- V stands for Virtual mode.
- Uses base registers for segment number.
- Prime 400-9950.
- Pure procedure is assumed, no automatic pipeline flush.

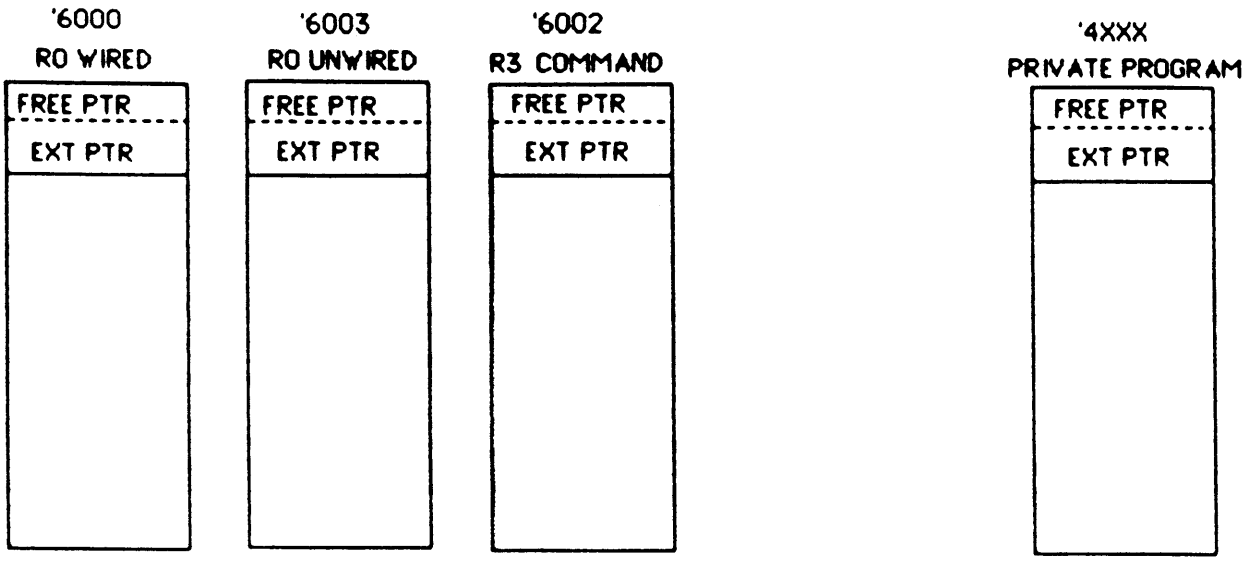
32I - 32 stands for 32 bit word length.
- I stands for Integer mode (or Immediate).
- Uses 8 general registers.
- Prime 500-9950

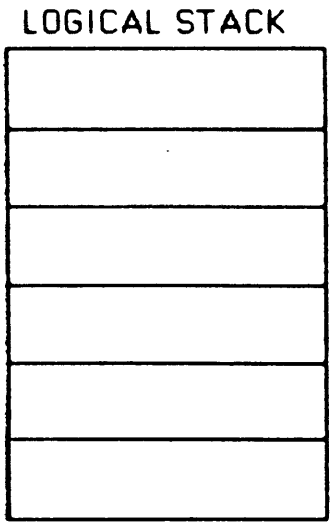
32IX - X stands for extended I-mode.
- Prime 2350-9950
- General purpose registers can be used like base registers.
- Pure procedure is assumed, no automatic pipeline flush.

Current User Registers



Stack Architecture





Stack Data Structures

STACK FRAME
- 1 per invocation.
- contains:
  - return pointer (caller's PB+PC).
  - caller's SB, LB and keys.
  - argument pointers.
  - dynamic data.
- pointed to by the Stack Base register (SB).

"STACKS" which are used on Prime:
- ring 0 wired stack (seg 6000).
- ring 0 unwired stack (seg 6003).
- ring 3 or "command" stack (seg 6002, DTAR2 as needed).
- Program stack (seg 4xxx as assigned).

STACK ROOT HEADER
- 1 per "stack".
- Free Pointer - where the new frame will go in this area.
- Extension Pointer - where to extend if necessary.

Procedure and Link Areas

- Assembly code is divided into two main parts:

PURE CODE (PROCEDURE AREA)
- Contains read-only parts of the program (usually instructions and constants).
- Pure code is shareable (only one copy per system is needed).
- The segment where the procedure area is located is contained in the PB.
- The PC keeps track of the current instruction which is being executed.

IMPURE CODE (LINKAGE AREA)
- Contains static data, address pointers, and the ECB (Entry Control Block).
- Every user must have their own copy of linkage when they execute.
- The beginning address (both segment and word number) is contained in the LB.

SEG

- Relocating loader. This means compilers produce addresses relative to beginning of the module. Thus a program will reference an address via a base register (i.e. LDA LB%+23).

- Linking loader (checks to see that all references are resolved).

- Maps program into SEGSAM directory, stores the notation of where in memory each part should be located.

- Creates "Static mode runfiles". This means that the program will execute using the same addresses every time it is executed.

- Is also used to invoke the programs, restoring program images into the appropriate locations in memory.

- Advantages of the default load (segment allocation)

  - programmer does not need to understand virtual memory in order to load programs

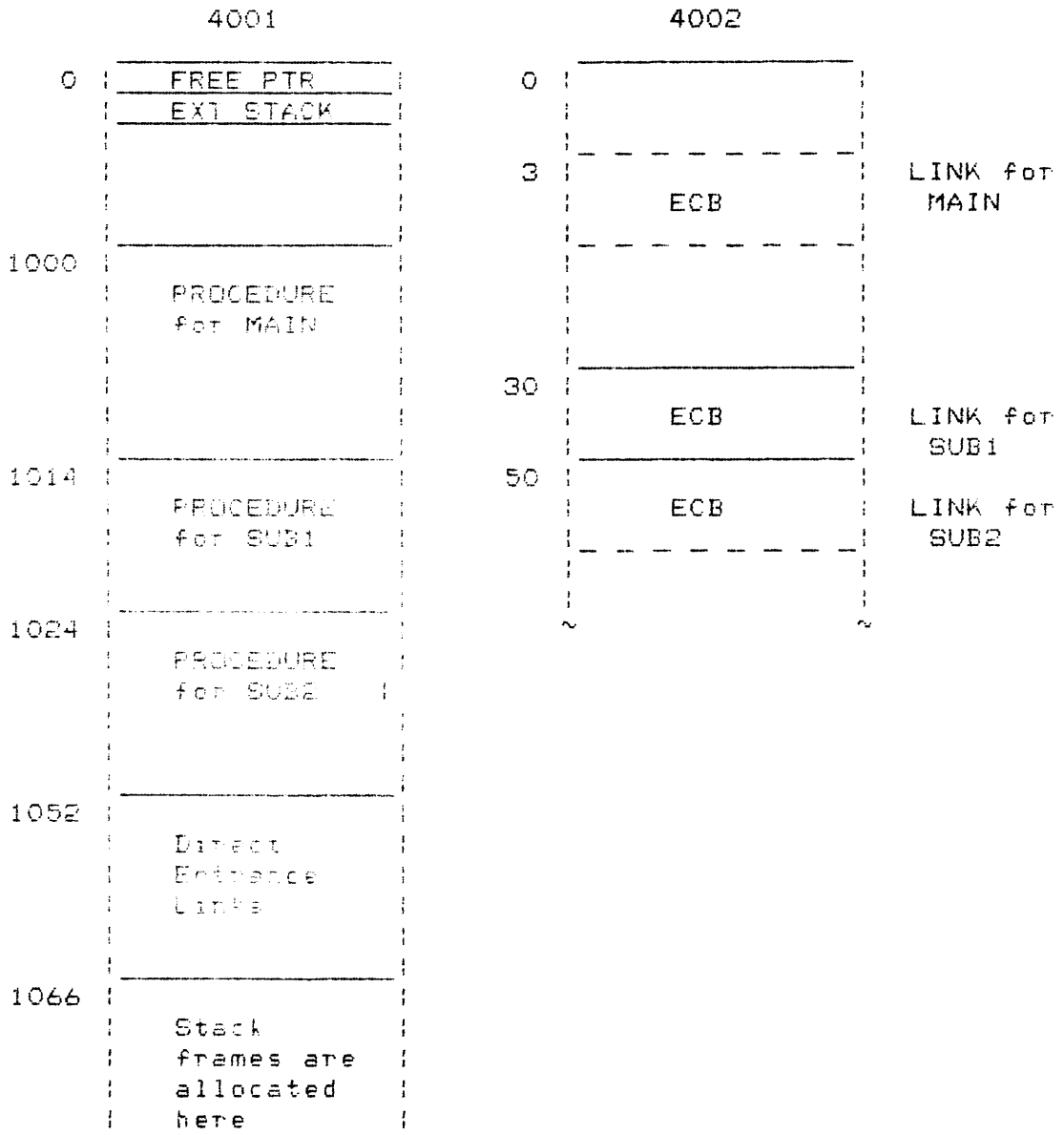
  - is needed in order to use DBG.

  - nothing will get overwritten (i.e loader will always allocated enough space, whereas a programmer may accidently overwrite portions).

  - can detect stack overflow, which is not always possible in non-default loads.

SEG Address Assignments

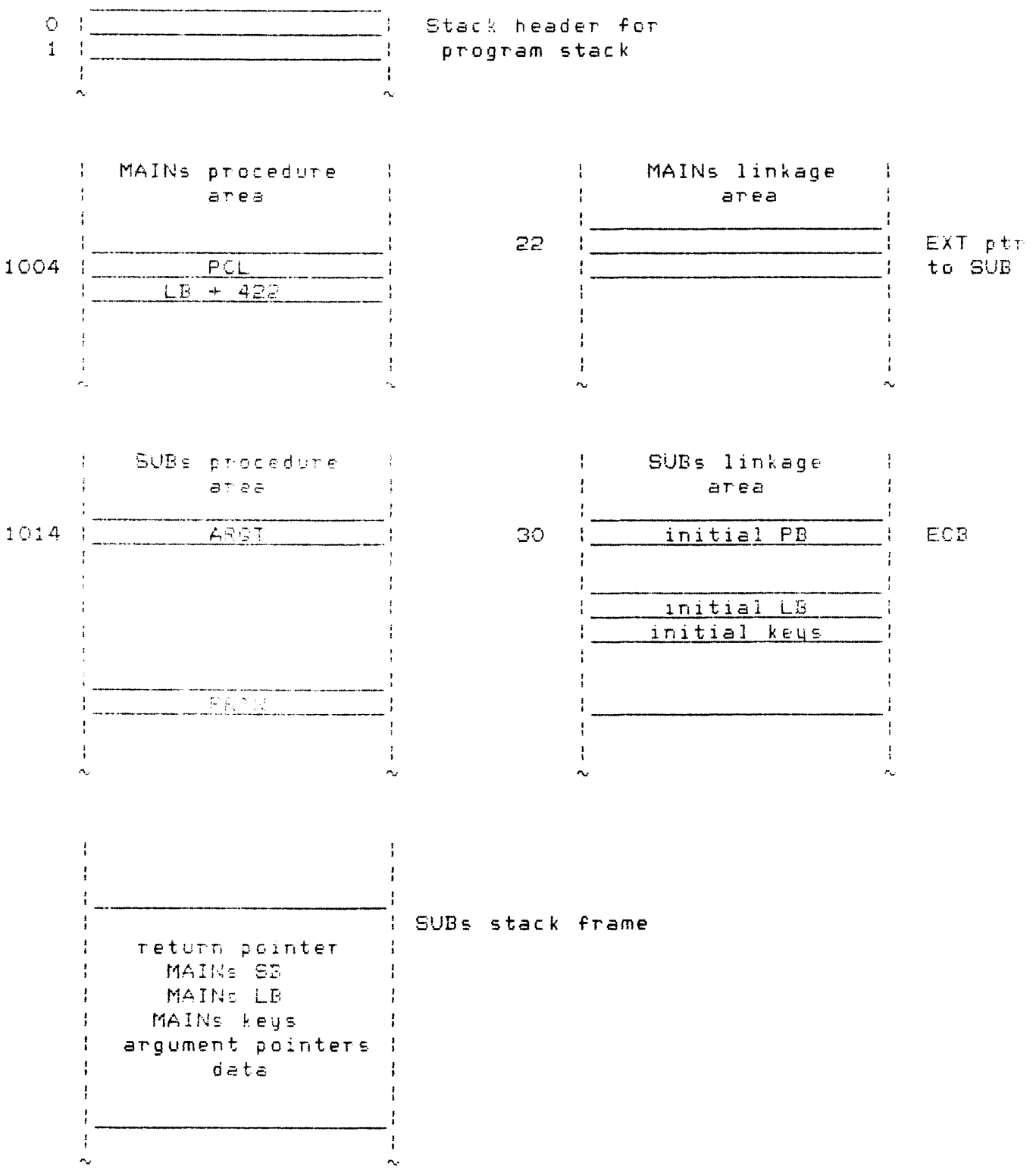
- When SEG is used to "load" a program, here are the default address assignments used:



SEG Maps

```
*START   4002  000003  *STACK   4001  001066  *SYM     000023

SEG. #     TYPE          LOW        HIGH       TOP
  4001     PROC##      001000     001065     001065
  4002     DATA        000000     000075     000075


ROUTINE       ECB              PROCEDURE      ST. SIZE  LINK FR.
  ####     4002  000003     4001  001000     000012  000030  4002  177400
  SUB1     4002  000030     4001  001014     000024  000020  4002  177430
  SUB2     4002  000050     4001  001024     000020  000026  4002  177450

DIRECT ENTRY LINKS
  EXIT    4001  001052     TNOU    4001  001056     TNOUA  4001  001062

COMMON BLOCKS

OTHER SYMBOLS
  F1920FP7  4001  001024
```

PCL Mechanism



PCL Related Instructions

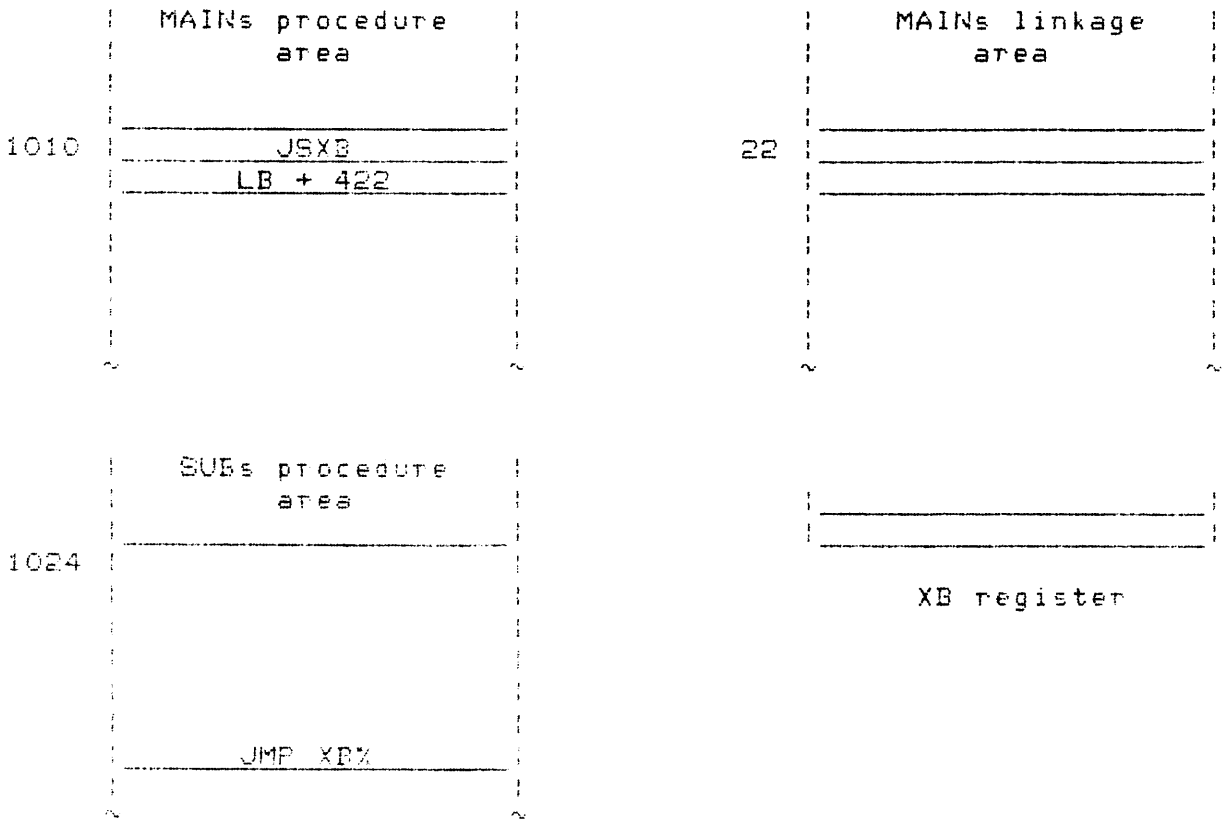
Entry Control Block (ECB)
- State of called procedure:
  - first executable statement.
  - size of stack frame.
  - displacement of first argument.
  - number of arguments.
  - LB of called procedure.
  - initial value for keys.
- Usually in the Link frame of the called procedure.

PCL - Microcoded instruction for fast and powerful processing.
1. Verify access to ECB.
   - if none, then ACCESS_VIOLATION$.
   - if pointer fault, try to link dynamically.
2. Create a new stack frame, at the top of the stack.
3. Save the caller's state (PB, LB, SB, keys) in the new stack frame from the user register set.
4. Load the callee's state (PB, LB, keys) into the register set from the ECB.
5. Calculate and store the indirect argument pointers.

ARGT - Argument Transfer instruction.
- Will finish an interrupted PCL instruction.
- Must be first instruction in any routine which accepts arguments.

PRTN - Procedure Return instruction.
1. Erase the old stack frame by reseting the top of stack to the callee's SB.
2. Restore caller's state (PB, LB, SB, keys) from the stack frame to the register file.

SHORTCALL Operation



SHORTCALL Instruction

- A shortcall operation is used for two pruposes: First, to avoid the overhead of a PCL when calling a very simple, small routine. Second, to do various tasks with registers that a PCL would destroy (i.e. changing the value of the keys).

  Shortcalls are usually based on the JSXB instruction. the JSXB will:
  1. Verify access to subroutine, if none, then ACCESS_VIOLATION$.
  2. Save the caller's PB in the XB register.
  3. Transfer control to the procedure (new PB).

  The JSXB instruction is still "pure" since it stores the return information in the XB (index base) register rather than memory.

- JSXB is faster than PCL because:
  - No new stack frame is allocated.
  - The LB, SB and keys do not have to be switched.
  - No return information has to be inserted into the stack frame.
  - No arguments are transfered

- Shortcalled routines are limited because:
  - They have no stack frame to help them return, or for data storage
  - They have no link area for data storage.
  - The base registers are still filled in for the calling program, and therefore cannot be used.
  - They must be written in PMA

- The Short Call statement is a Prime extension to standard Fortran and is also used by PLP (example programs are SHORT.FTN and SHORT.PLP in CLASS directory).

STATIC VS DYNAMIC RUNFILES

STATIC	DYNAMIC
.SEG, .SAVE	.RUN
SEG or LOAD loaders	BIND loader
Uses the same static segments for every invocation as assigned by SEG/LOAD	Uses available dynamic segments for every invocation as assigned by PRIMOS
Contains virtual addresses	Contain EPF Relocatable Pointers ERPs
Contains procedure and linkage images	Contains procedure image and a description of the linkage area(s)
Entire runfile is read into memory and paging space allocated	Procedure images mapped to memory via VMFA, required linkage is built, and paging space allocated for linkage; procedure read into memory as needed
User manages address space	PRIMOS manages address space
Limited restartability of command environment	Full restartability of command environment
Uses private stack (PSTK)	Uses command processor stack

BIND Load Map

```
Map of FACTORIAL

  START ECB: -0002/000002

  Segment Type      Low    High    Top
   -0002  DATA    000002 000153 000154
   +0000  PROC    001000 001375 001376


PROCEDURES:
  Name          ECB address     Initial PB%   Stack size  Link  size  Initial
 LB%
                -0002/000002   +0000/001000     000012       000056   -0002/17
7400
  FACT          -0002/000056   +0000/001046     000022       000022   -0002/17
7456
  TIDEC         -0002/000100   +0000/001074     000032       000024   -0002/17
7500
  TBUFIN        -0002/000126   +0000/001174     000020       000030   -0002/17
7524

DYNAMIC LINKS:
  C1IN          +0000/001356
  ERKL$$        +0000/001362
  T1OB          +0000/001346
  TNOU          +0000/001366
  TNOUA         +0000/001372
  TONL          +0000/001342
  TOVFD$        +0000/001352

COMMON AREAS:

OTHER SYMBOLS:

UNDEFINED SYMBOLS:
```

VMFA

- VMFA (Virtual Memory file Access) is a method of paging from the file system disks rather than the paging disks.

- A program which is to use VMFA must be stored as a "memory image" on disk. With EPFs, the procedure code is stored in this way.

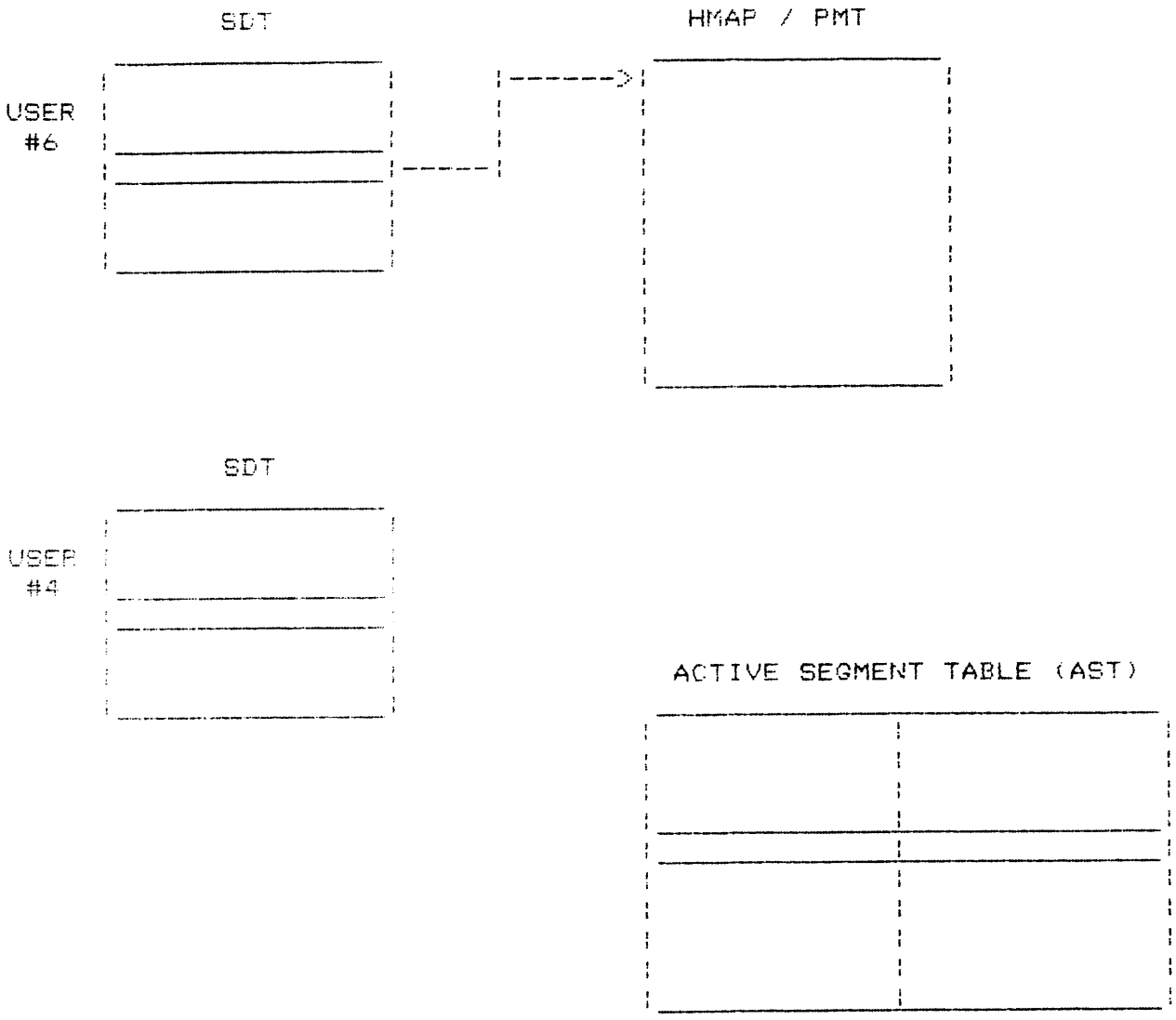
- When a program is "loaded" into memory, only the initial pages are brought into memory...the rest are brought in as needed by the normal paging algorithym when the page is first "touched".

- Since with EPFs the procedure code MUST be pure, when a page of memory from a procedure page must be paged out, there is always an accurate copy on disk and therefore no I/Os to disk are required.

- When EPFs are widely used, paging space requirements will be substantially less. Also, the amount of paging on the system should also decrease.

- With a program which has a large amount of procedure and a small amount of linkage, the execution startup time will be substantially less.

Dynamic Sharing (EPFs)



Caching EPFs

The Segment Mapping Table - SMT

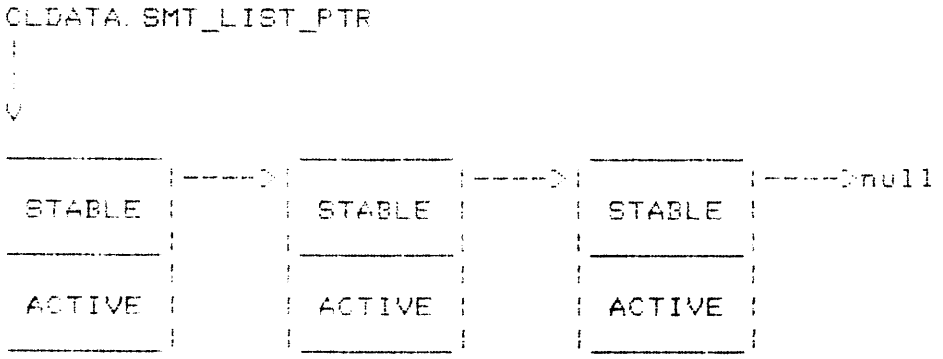
- Each process using an EPF must keep track of the status and virtual mapping for its use of that EPF. The table dynamically created at invocation time is called a Segment Mapping Table (SMT). There is one SMT for each EPF that a process has mapped into memory, and they are linked together into a list. The SMT contains the following type of information:

  - Stable information about the EPF that will not change regardless of the number of invocations, such as the number of procedure segments and linkage segments required.

  - Active information that could change from invocation to invocation, such as the command level.

  - An address table which keeps track of the virtual addresses being used for the current invocation of the EPF.

  - The full pathname of the EPF

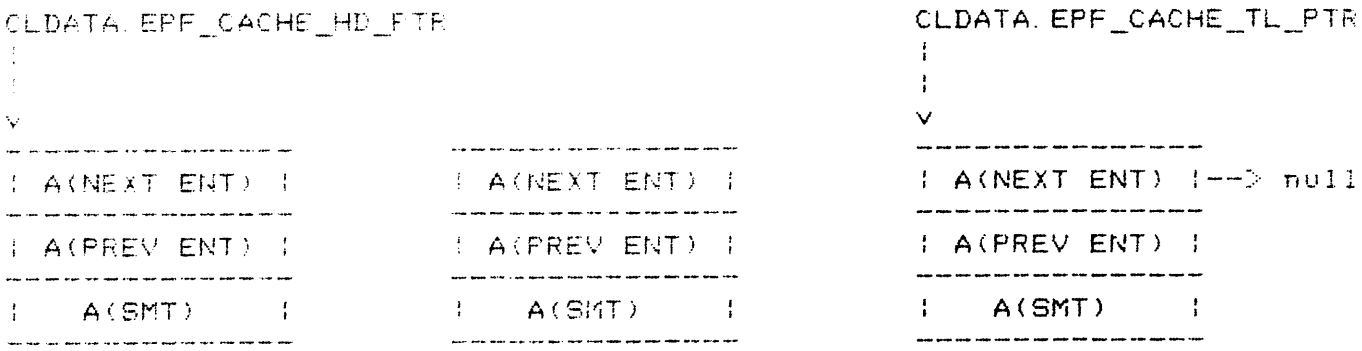


Invoking EPFs

When an EPF is invoked, a cache entry is threaded onto the head of the process' cache list, and then calls the EPF. When the EPF returns, its cache entry is left threaded onto the cache list, but its SMT is marked as being inactive. Another invocation of the EPF, while its cache entry is still threaded on the cache list, will only have to go through a partial initialization (i.e., static data and faulted IPs) of the linkage area.

An EPF's cache entry will remain on the cache list until it is removed because:

(1) the cache list has become full, and it is the least recently used entry,
(2) it has been explicitly removed with the Remove_Epf command,
(3) the user's ring 3 environment has been reinitialized, or
(4) a new command level is pushed.



Library Classes

- There are two main classes of EPF Libraries:
  - Program class
  - Process class

  The two library types are differentiated by their initialization requirements.

- A program class library runfile is given a new linkage area (re-initialized) for every program which calls it.

- A process class library runfile has its linkage allocated and initialized <u>once</u> upon initial execution by any program running within a process. This linkage area will be maintained for any other programs using this routine (at any command level). The linkage will be maintained until the user logs out, re-initializes his command environment, or explicitly removes the library.

Lesson 10 - Exception Handling

Objectives: Upon successful completion of this lesson, students will be able to:

- Describe the three types of exceptions which the system will handle.
- Describe what the different checks are, how they are caused, and how they are handled.
- Describe the difference between the fault mechanism, fault handlers, and the condition mechanism.
- Describe how dynamic linking is accomplished.
- Describe what constitutes "command depth"

Exceptions

o There are three types of exceptions recognized by the micro-code:

1) External Interrupts - A controller is signaling that it needs some work done by a software process.

2) Checks - A hardware malfunction has occured which was NOT caused by the currently executing process.

3) Faults - A software event has occured which WAS caused by the currently executing software.

Exception Handling Mechanism

- The microcode will handle all three exceptions using the same basic steps:

  1) Microcode detects the exception.

  2) The program counter (PB) and mode (keys) of the executing process are saved.

  3) The address of the exception handler (vector) is obtained from the appropriate source (controller or process' PCB).

  4) The addressing mode is set to 64V.

  5) The exception handling code is executed.

- The only difference between how the various exceptions are handled is where the PB and keys are saved, where the vector is obtained from, and the complexity of the handler.

Checks and Check Handling

o There are five types of checks on Prime system:

- Power fail (also environmental sensor checks).

  The check handler will check to see what caused the error and shut down the CPU with various degrees of speed and gracefulness.

- Memory parity errors (ECC).

  Parity errors in the data stored in main memory are detected before reaching the CPU. If a 1 bit error is detected, it will be corrected before being shipped. An Error Correct and Check Corrected (ECCC) signal is then sent and a check will occur. The event will be logged and things will continue. If a two or more bit error is detected, a ECC Uncorrected (ECCU) will be signaled. The check handler will map out the page with the bad memory, log out the user, and halt the CPU. The CPU will then be automatically warmstarted if the directive MEMHLT is set to NO.

Check Handlers con't.

- Machine Checks.

  Machine checks are parity errors which occur anywhere else aside from main memory data errors. They are handled by halting the system.

- Missing Memory.

  Missing memory errors are caused by accessing memory which does not exist. This will cause the machine to halt.

- Correctable parity (soft error recovery - 9955, 9955-II only).

  If a parity error is encountered in the STLB or cache, check handler will cause the entry to re-loaded.

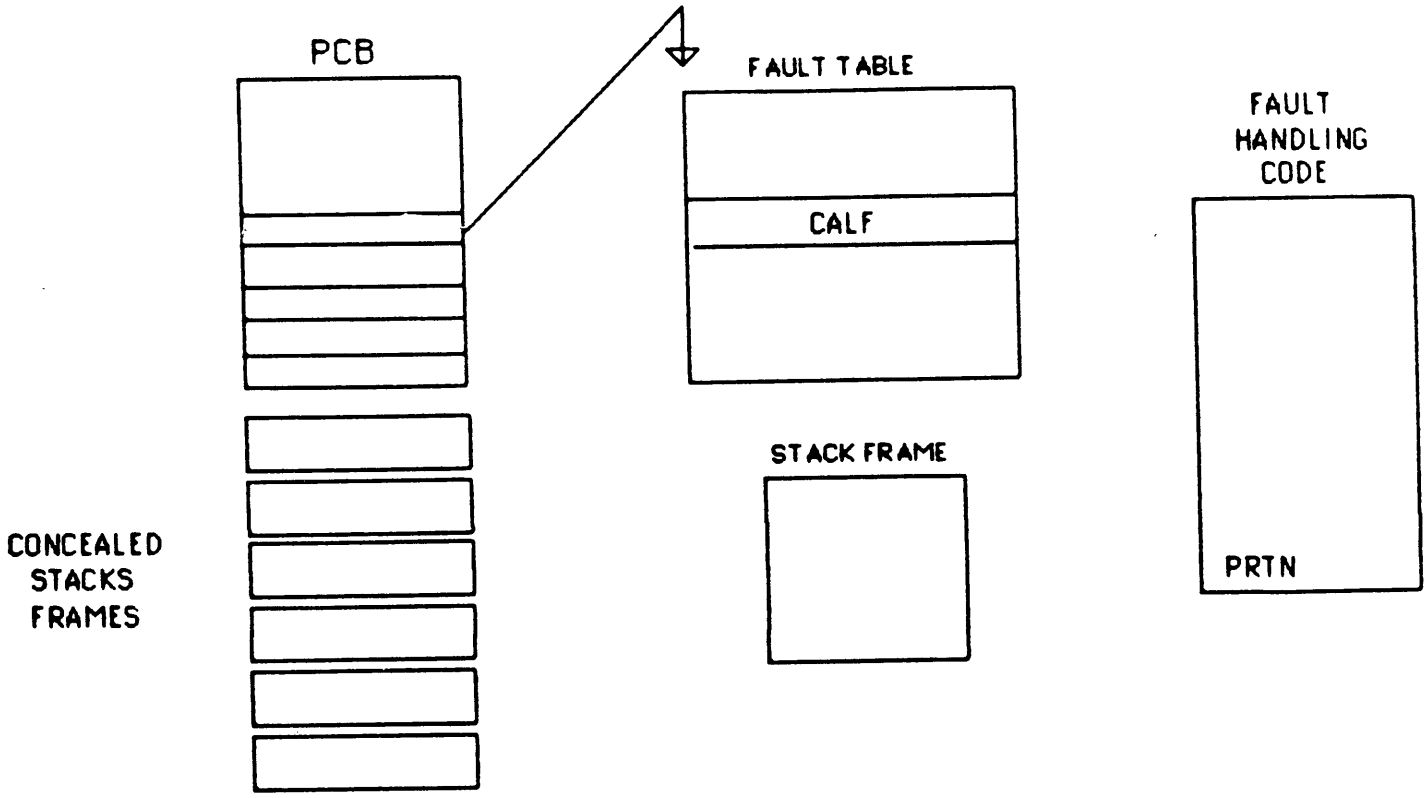
Fault Mechanism

- A FAULT is an unexpected event which has been detected as a result of the currently running software. The fault mechanism calls a software fault handler on behalf of the running software to process the event. The hardware detects a fault.

- The Fault mechanism microcode:

  1) Saves the PB and keys in the concealed stack from the register file.
  2) Transfers control (sets a new PB) to a CALF entry in the appropriate Fault table.

- The Call Fault Handler (CALF) instruction emulates a PCL by:
  1) Create a new stack frame, at the top of the stack.
  2) Save the caller's state (PB, keys) in the new stack frame from the concealed stack.
  3) Save the caller's state (LB, SB) in the new stack frame from the register file.
  4) Load the callee's state (PB, LB, keys) into the register file from the ECB.



Fault Handling

- Unimplemented Instruction (UII)
  - Processor tries to execute an instruction that is not implemented on this machine.
  - Emulate the hardware instruction with software.
  - If missing or error in the software routine signal the condition UII$.

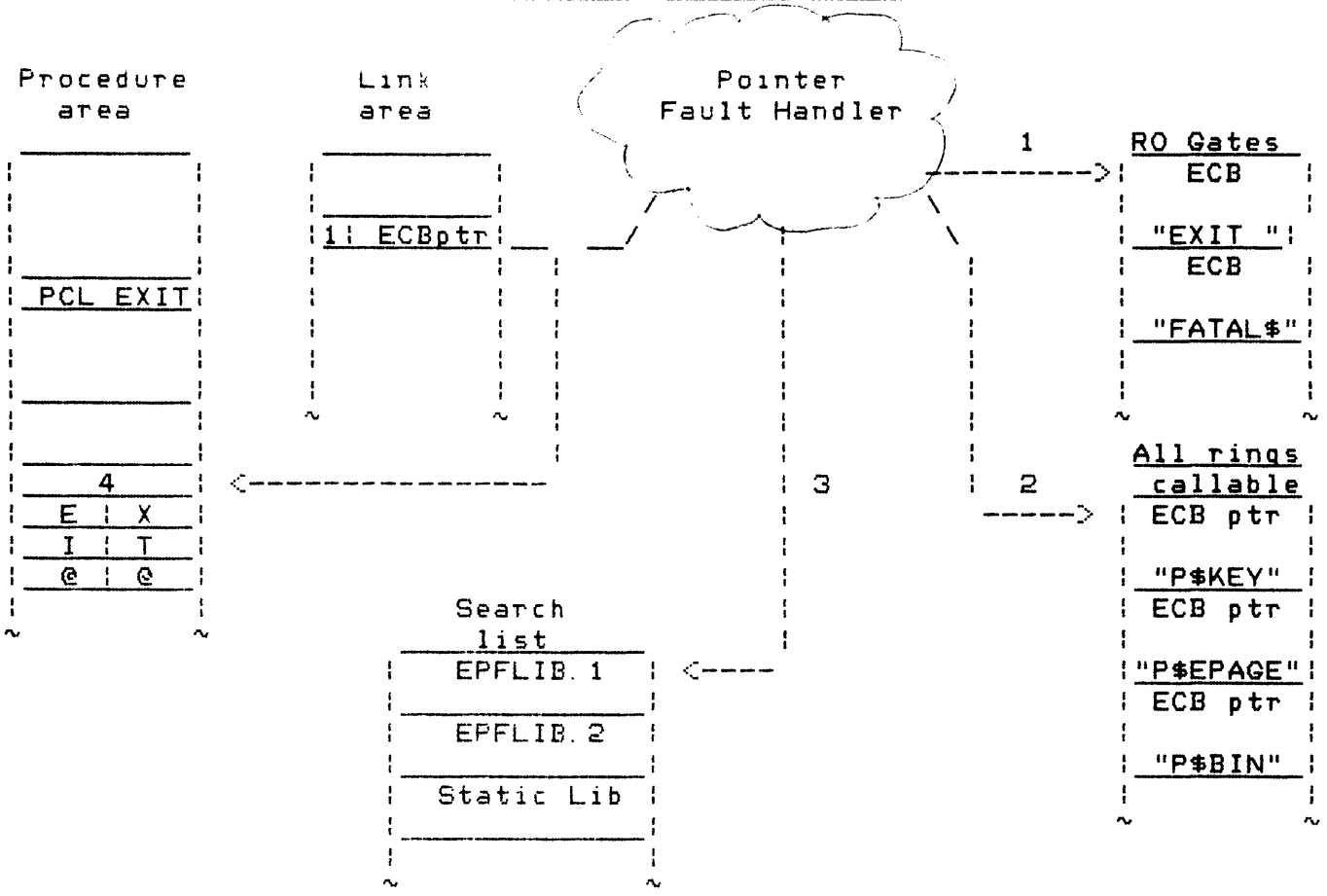
- Restricted Instruction
  - A process operating in ring 3 tried to execute a restricted instruction opcode.
  - The condition RESTRICTED_INSTRUCTION$ will be signalled.

- Access Violation
  - A process operating in a ring other than tried to access virtual memory which is not set up for that ring.
  - The condition ACCESS_VIOLATION$ will be signalled.

- Stack Overflow
  - PCL, or CALF instruction does not have enough room between the top of the stack and the end of the segment for the new frame.
  - If a ring 3 stack overflow try to allocate a stack extension segment (Primos revision 19).
  - If a ring 0 stack segment, no dynamic segments available, or no extension segment provided, signal the condition STACK_OVF$.

- Process Abort
  - The processes abort flags are non-zero when the process is dispatched.
  - The process abort handler (PABORT) will look at the abort flags and decide what abort occurred, and call the appropriate routine.
  - The various process aborts are:
    - Timeslice end
    - Forced logout (AMLC disconnect).
    - Inactivity timeout.
    - Software Interrupts (^P).

Dynamic Linking - Rev 19.4



-LOAD example, and this time discuss how

Rev 19.4 Dynamic Linking Operation

o Here is a step-by-step description of dynamic linking.

1) A PCL instruction executes. It accesses a pointer created in the link area which should point at the ECB of the called routine.

2) The ECB pointer has bit #1 set on. This triggers a pointer fault. A CALF instruction is executed and the pointer fault handler (PFH) is called.

3) The PFH examines the faulted pointer to see if it contains a valid address (not Os). If it does, the PFH strips the fault bit and accesses that address.

4) The address should point at a data structure called a DYNT. DYNTs are data structures which contain the name of the called routine plus a character count.

5) The PFH now calls LN_SLIB to check through the Ring 0 library, the Ring 3 library (All Rings Callable) and any of the users own libraries (whether EPF libraries or static libraries) for a match on the name contained in the DYNT.

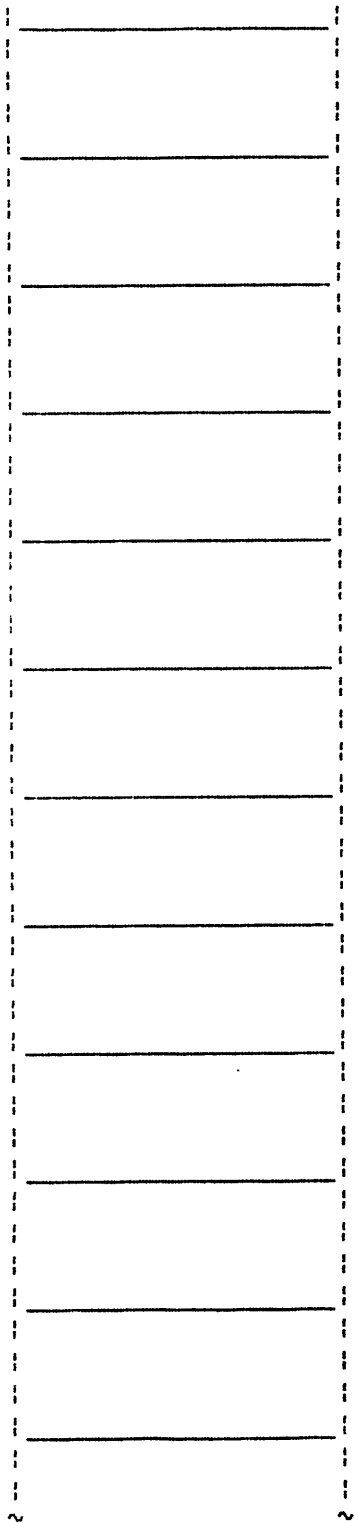
6) When a match is found, the correct address of the ECB will be filled into the original faulted pointer. Now the PCL will be re-executed and this time it will call the routine.

7) If a match is NOT found, the PFH will signal a LINKAGE_FAULT$ condition.

Condition Mechanism

- The Condition Mechanism is a method of suppling event handlers on a process by process basis. Some features are:
  - Strictly a software mechanism (not related to faults).
  - Can be used and modified by ring 3 users.
  - Used by fault handlers to bring a fault to the attention of user software.
- The condition mechanism is implemented via a collection of subroutines. Some of the key routines are:
  - SIGNL$ - records information about the condition and the state of the process at the time the condition was signalled.
  - ON-UNIT - a subroutine designed to handle a specific condition.
  - RAISE_ - searches the stack frames for an on-unit to handle the condition.

Condition Mechanism Example



Lesson 11 - Asynchronous and Terminal Input/Output

Objectives: Upon successful completion of this lesson, students will be able to:

- Describe the asynchronous character I/O process used with AMLC and ICS controllers.

- Correct asynchronous and terminal data loss with the AMLBUF and AMLIBL configuration directives.

The QAMLC/ICS Driver (AMLDIM/ASYNDM)

- default setting all AMLC lines to 1200 baud, TTY protocol, except the last line which defaults to 110 baud.
  [Defaults can wire memory that is never used.]

- AMLC [PROTOCOL] LINE [CONFIG] [LWORD], operator command

  ASSIGN AMLC [PROTOCOL] LINE [CONFIG] [LWORD], user command

  PROTOCOL

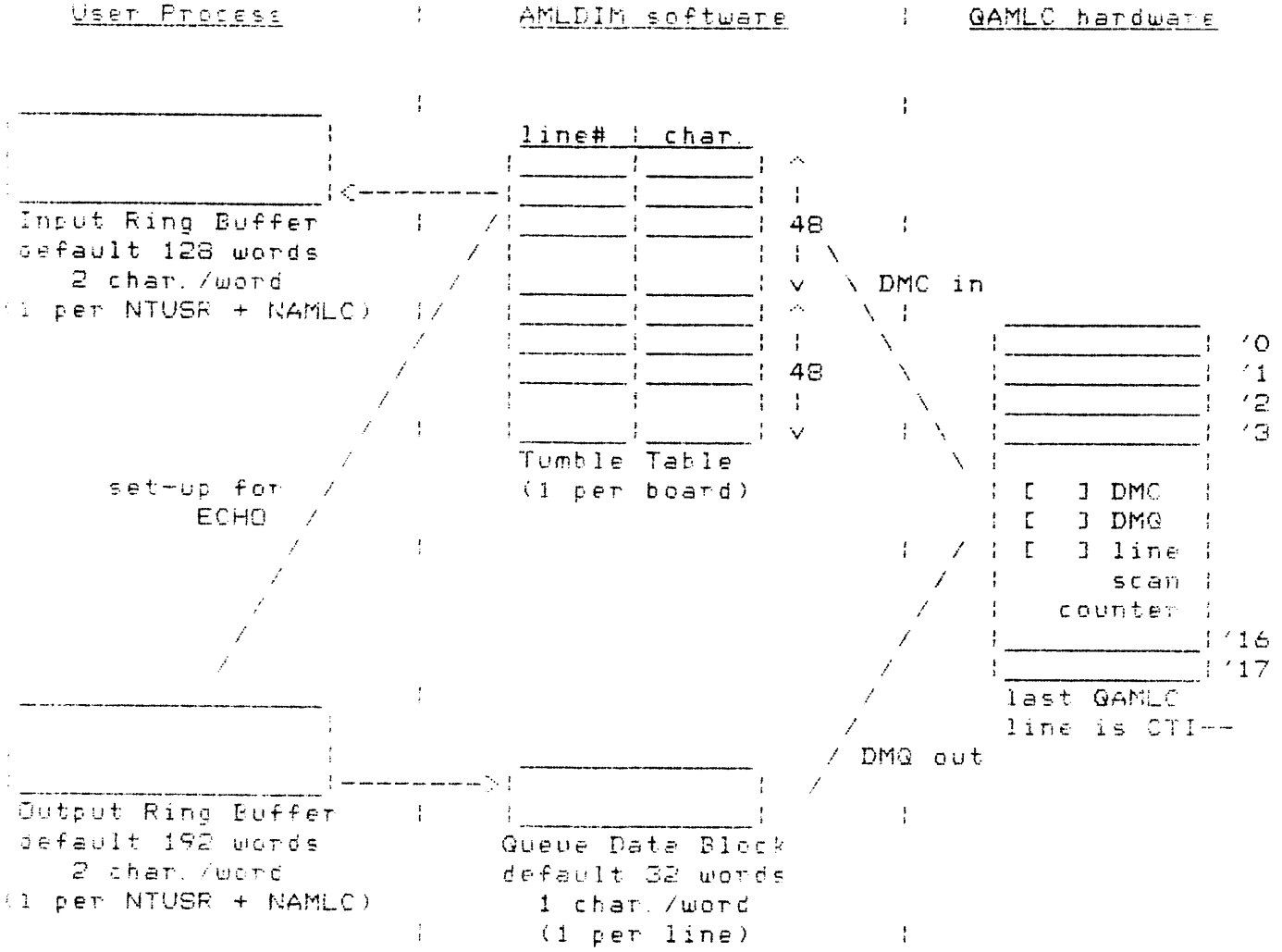
  - TTY, TTYUPC terminal protocol [Operator should only set for terminal users.]
  - TRAN transparent protocol
  - TTYNOP ignore this line [Operator should set for all unused and assigned lines]
  - ASP auto speed detect
  - TTBPIT 8-bit protocol

  LINE physical line number (octal)

  CONFIG data set control, baud rate, bit pattern, parity, reverse flow control

  LWORD terminal characteristics, user number assignment

Asynchronous Input/Output



1 MLC → TUMBLE TABLE HAS 2 SIDES W/ 8 CHAR EACH. (SEE R+ AMLIBL)
WHEN ONE IS FULL [redacted] BOARD SWITCHS +
SIGNALS FOR SYSTEM TO DUMP FIRST SIDE INTO USER
IRBS.

Setting Buffer Sizes

- DMQ Size = characters per second / CTI rate
  (round up to nearest power of two)

  Exceptions:

  - If the device has a smaller buffer than the DMQ, the DMQ may have to be configured to a smaller size to prevent device buffer overflow.

- ORB Size = (characters per second / 2) / 2

  Exceptions:

  - If the system is very memory bound, you can safely reduce the size of the output buffer. Lower effective character throughput may result.

  - For applications with large working sets, the 1/2 sec wait may be more damaging to paging than having a large buffer. Therefore the buffer should be large enough to accomodate the whole output flow.

- IRB Size = amount a characters which can be input before the user process can empty the buffer. This is typically determined by three factors:

  1) Type ahead
  2) Block mode input
  3) Your processes response time (CPU speed plus number of users)

- Tumble tables Size = number of characters input before AMLDIM can empty 1/2 of buffer

Buffer Overflow Conditions

Tumble Table Overflow

symptom: losing data on multiple lines on the same AMLC board.
cause: one or more devices transmiting input faster than AMLDIM can empty half of the tumble table.
scenario: block mode terminals, computer links (including microcomputers).
solution: increase tumble table size with AMLIBL or ICS INPQSZ directive; move fast input transmit devices to ICS board, balance these devices among boards.

Input Ring Buffer Overflow - Rev 20 and before

symptom: losing data on a line, block mode terminal locks up
cause: device transmitting input faster than program software (not AMLDIM) processes the input ring buffer. Block mode lock up caused by missing EOT echo.
scenario: block mode terminals, computer links.
solution: increase input ring buffer size with AMLBUF directive.
NOTE: At Rev 20.2 AMLC controllers are capable of reverse flow control at the Input Ring Buffer, making IRB overflows obsolete.

Output Ring Buffer Full

symptom: none, slower program performance.
cause: attempt to put character in output buffer when full, causes one half second pause before trying again.
scenario: serial graphics output, computer links.
solution: increase output ring buffer size with AMLBUF directive.

Device Buffer Overflow

symptom: missing output with no input ring buffer overflow.
cause: device buffer or buffer window is smaller than QDB buffer.
scenario: NEC printers, devices that send XOFF too late.
solution: decrease QDB buffer size (minimum '20).

ICS Differences

- The ICS boards (ICS1, 2, & 3) are down line loaded during PRIMOS cold and warm start.

- The ICS boards wake up a process called ASYNDM.

- The ICS boards use DMQ for input and therefore can handle fast input transmitting devices better.

- The ICS boards have a default CTI of 1/10 second, which can be configured with the ICS INTRPT directive (PRIMOS 19.2.7).

- The ICS controllers can use reverse flow control. If configured, the controller will send an XOFF to a device if it gets 2 successive EOR signals when trying to transfer a character into memory. ICS1 boards (and some ICS2) do reverse flow control on the Input Ring Buffer only. ICS2 controllers (and some ICS2) can reverse flow control on the IRB and the DMQ buffers.

Configuring User Buffers

- There are three numbers associated with asynchronous input:

  Line # _____

  User # _____

  Buffer # _____

- The format of the AMLBUF directive is as follows:

  AMLBUF # IRB ORB DMQ

  This directive really has two functions:

  1) AMLBUF _____ _____ _____ _____

  2) AMLBUF _____ _____ _____ _____

How to Set Up AMLC

```
AMLC  00  TTY   2413  02000      /BUFFER # ____
AMLC  01  TTY   2413  02000      /BUFFER # ____
AMLC  02  TTY   2413  02000      /BUFFER # ____
AMLC  03  TTYNOP 0    02000      /BUFFER # ____
AMLC  04  TTY   2413  02000      /BUFFER # ____
AMLC  05  TTY   2413  02000      /BUFFER # ____
```

How NOT to Set Up AMLC

```
AMLC  00  TTY   2413  02000      /BUFFER # ____
AMLC  01  TTY   2413  02000      /BUFFER # ____
AMLC  02  TTY   2413  02000      /BUFFER # ____
AMLC  03  TTYNOP 0    02000      /BUFFER # ____
AMLC  04  TTY   2413  02000      /BUFFER # ____
AMLC  05  TTY   2413  02000      /BUFFER # ____
```

Title: Asynchronous Buffer Configuration.

Objectives: Upon sucessful completion of this lesson, students will be able to:

- Set the AMLBUF configuration directives to minimize wired memory.

Task:
In groups of four, given a description of a system's configuration, define the CONFIG file directive AMLBUF.

Conditions: Using any available course documentation.

Evaluation Standard:
Completion of the entire exercise.
Class review of the exercise will ensure correct answers.

The system:

- 750 processor
- 8 MB memory
- one 16 line QAMLC board
- one ICS1 board
- the terminal lines are as follows

00	9600 baud PT45, FORMS application
01	9600 baud PT45, FORMS application
02	9600 baud PST100, FORMS application
03	9600 baud PST100, FORMS application
04	unused
05	1200 baud modem line
06	1200 baud modem line
07	300 baud modem line
10	9600 baud PST100
11	300 baud QUEM letter quality printer
12	9600 baud PST100 terminal
13	9600 baud PST100 terminal
14	9600 baud PST100 terminal
15	9600 baud PST100 terminal
16	9600 baud PRINTRONIX printer
17	unused
20	9600 baud PST100 terminal
21	9600 baud PST100 terminal
22	9600 baud PST100 terminal
23	9600 baud PST100 terminal
24	1200 baud NEC letter quality printer
25	1200 baud hardcopy terminal
26	9600 baud PRINTRONIX printer
27	unused

Here are the completed AMLC commands for all 24 asynchronous lines. The NTUSR and NAMLC arguments are specified.

```
AMLC 00 TTY 2413  020002      / 9600 BAUD - FORMS APPLICATION
AMLC 01 TTY 2413  020003      / 9600 BAUD - FORMS APPLICATION
AMLC 02 TTY 2413  020004      / 9600 BAUD - FORMS APPLICATION
AMLC 03 TTY 2413  020005      / 9600 BAUD - FORMS APPLICATION
AMLC 04 TTYNOP 0  020000      / UNUSED LINE
AMLC 05 TTY 2313  020006      / 1200 BAUD MODEM
AMLC 06 TTY 2313  020007      / 1200 BAUD MODEM
AMLC 07 TTY 2313  020010      /  300 BAUD MODEM
AMLC 10 TTY 2413  020011      / 9600 BAUD
AMLC 11 TTYNOP 2213 020000    /  300 BAUD QEM PRINTER
AMLC 12 TTY 2413  020012      / 9600 BAUD
AMLC 13 TTY 2413  020013      / 9600 BAUD
AMLC 14 TTY 2413  020014      / 9600 BAUD
AMLC 15 TTY 2413  020015      / 9600 BAUD
AMLC 16 TTYNOP 2413 020000    / 9600 BAUD PRINTRONIX PRINTER
AMLC 17 TTYNOP 2213 020000    /UNUSED, BAUD RATE 300
AMLC 20 TTY 2413  020016      / 9600 BAUD
AMLC 21 TTY 2413  020017      / 9600 BAUD
AMLC 22 TTY 2413  020020      / 9600 BAUD
AMLC 23 TTY 2413  020021      / 9600 BAUD
AMLC 24 TTYNOP 2313 020000    / 1200 BAUD NEC PRINTER
AMLC 25 TTY 2313  020022      / 1200 BAUD HARDCOPY TERMINAL
AMLC 26 TTYNOP 2413 020000    / 9600 BAUD PRINTRONIX PRINTER
AMLC 27 TTYNOP 0    020000    / UNUSED

   NTUSR 22 (Octal)

   NAMLC  4 (Octal)
```

1) The %CPU averages 90.00% and the PF/S averages 6.00.
SET THE DMQ BUFFER SIZES ACCORDINGLY.

AMLC Baud Rate should be _____, making the CTI _____.
The ICS INTRPT should also be set. See the Sys Admin Guide for the appropriate value.

/*	line_number	default	default	dmq_buffer_size
AMLBUF	00	0	0	
AMLBUF	01	0	0	
AMLBUF	02	0	0	
AMLBUF	03	0	0	
AMLBUF	04	0	0	
AMLBUF	05	0	0	
AMLBUF	06	0	0	
AMLBUF	07	0	0	
AMLBUF	10	0	0	
AMLBUF	11	0	0	
AMLBUF	12	0	0	
AMLBUF	13	0	0	
AMLBUF	14	0	0	
AMLBUF	15	0	0	
AMLBUF	16	0	0	
AMLBUF	17	0	0	
AMLBUF	20	0	0	
AMLBUF	21	0	0	
AMLBUF	22	0	0	
AMLBUF	23	0	0	
AMLBUF	24	0	0	
AMLBUF	25	0	0	
AMLBUF	26	0	0	
AMLBUF	27	0	0	

2) Specify the AMLBUF commands for the input and output terminal buffers.

```
/* AMLBUF commands setting terminal buffers
/* buffer_number = user_number - 2
/*      buffer_number   input_buffer    output_buffer
AMLBUF      00

AMLBUF      01

AMLBUF      02

AMLBUF      03

AMLBUF      04

AMLBUF      05

AMLBUF      06

AMLBUF      07

AMLBUF      10

AMLBUF      11

AMLBUF      12

AMLBUF      13

AMLBUF      14

AMLBUF      15

AMLBUF      16

AMLBUF      17

AMLBUF      20
```

3) Specify the AMLBUF commands for the assigned line input and output buffers.

```
/* AMLBUF assignments for assigned lines
/* The first assigned buffer number = NTUSR - 1     (NRUSR = 0)
/* There is a pool of NAMLC lines
```

4) The %CPU averages 60.00% and the PF/S averages 15.00.
SET THE DMQ BUFFER SIZES ACCORDINGLY.

AMLC Baud Rate should be _____, making the CTI _____.
The ICS INTRPT should also be set. See the Sys Admin Guide for the appropriate value.

```
/*      line_number   default   default   dmq_buffer_size
AMLBUF      00          0         0
AMLBUF      01          0         0
AMLBUF      02          0         0
AMLBUF      03          0         0
AMLBUF      04          0         0
AMLBUF      05          0         0
AMLBUF      06          0         0
AMLBUF      07          0         0
AMLBUF      10          0         0
AMLBUF      11          0         0
AMLBUF      12          0         0
AMLBUF      13          0         0
AMLBUF      14          0         0
AMLBUF      15          0         0
AMLBUF      16          0         0
AMLBUF      17          0         0
AMLBUF      20          0         0
AMLBUF      21          0         0
AMLBUF      22          0         0
AMLBUF      23          0         0
AMLBUF      24          0         0
AMLBUF      25          0         0
AMLBUF      26          0         0
AMLBUF      27          0         0
```

Lesson 12 - Tuning the Scheduler

Objectives: Upon successful completion of this section, students will be able to:

- Be able to use CHAP to effectively reward or punish a process relative to the remaining processes on the system.
- Set ELIGTS to optimize throughput vs. response time appropriate to a systems's application mix.

Tuning the Scheduler

- The basic objectives of tuning the scheduler are as follows:
  - Punishing or rewarding a process or group of processes in relation to other processes on the system. The CHAP command allows this.
  - Setting an execution environment to favor either more interactive or more compute bound processes. ELIGTS is used for this purpose.

The CHAP Command

o CHAP CHAP is used to change the priority level and major time-slice of a process. CHAP has two versions, one must be issued at the system console, and the other is a user version. Here is the system console version:

```
CHAP    {-userno} {priority [time-slice]}
        {ALL}     {-IDLE}
                  {-SUSPEND}
```

userno	Is in the form -nn or ALL.
priority	Integer 0 to 3 (default = 1).
-IDLE	Put process(es) into the IDLE state. This argument will only work on phantom processes.
-SUSPEND	Put process(es) into the SUSPEND state.
time-slice	Length of major time-slice in tenths of seconds. 0 means reset to the system default (2 sec.) If omitted the time-slice is unchanged.

If both priority and timeslice are omitted, then priority and time-slice are set to the system default values.

Here is the user version:

```
CHAP    {UP}
        {DOWN}
        {LOWER nn [time-slice]}
        {IDLE}
```

UP	Sets user level to the default level.
DOWN	Sets user level to 0.
LOWER	Sets user level down by nn.
time-slice	Sets major t/s to the value specified. This will only set the value lower than the current value.
IDLE	Sets user level to IDLE. Can only be issued from a phantom.

Rewarding and Punishing Processes

- PRIORITY - When changing a process's priority, you should consider:
  - Priority determines which user level on the Ready List the process will go on when NOTIFYed. This is important because a process with a higher priority will ALWAYS pre-empt a lower priority process IMMEDIATELY.
  - Most semaphores are threaded in priority order. This means if two processes are waiting for the same resource, the higher priority process will be given access to that resource first.
  - Priority determines which LOPRIQ the process goes on when the major timeslice expires. This affects the time it takes for the process to be NOTIFYed back to the ready list.

- MAJOR TIMESLICE - When changing a process's major timeslice, you should consider:
  - Altering major timeslice has the most effect on compute bound processes (those processes using more than 4 seconds of CPU). It has little if any effect on interactive processes (although it can have a large effect if made short enough).
  - Altering the major timeslice has the most effect in compute environments. It has little if any effect when there are very few compute bound processes.
  - Every time you reach a point in a program which asks for character input, you have both your major and minor timeslices reset.

NOTE: If the major timeslice is set to 177777, this will cause SCHED to execute a return when called. Thus, when a minor timeslice end occurs, SCHED will not execute a WAIT on any of the hold queues. This will give unlimited access to the CPU. To prevent the system console from being trapped on a hold queue by a process with a 177777 timeslice, user #1 is also given the same timeslice.

Rewarding or Punishing Processes - con't

- To punish a process:

		SYSTEM: compute bound	SYSTEM: interactive
PROCESS	compute bound	lower priority lower major t/s	lower priority lower minor t/s
	interactive	lower priority	lower priority

- To reward a process:

		SYSTEM: compute bound	SYSTEM: interactive
PROCESS	compute bound	raise priority raise major t/s	raise priority raise minor t/s
	interactive	raise priority	raise priority

Tuning for Response Time vs. Throughput

- ELIGTS - ELIGTS is used to modify the minor time-slice from the system console. This will effect all users equally.

  ELIGTS {minor_timeslice} (default = 3/10 sec.)

- MINOR TIMESLICE - When changing the minor timeslice, you should consider:

  - Changing the minor timeslice will allow you to improve either response time or throughput, but at the expense of the other. Lowering the timeslice will cycle more processes through the CPU in a given amount of time, and each process will have to wait less time to execute. However, process exchange overhead will increase, and each process will be able to do less of it's work before being put on ELIGQ. Obviously, the reverse is also true.

  - Decreasing the minor timeslice can be useful in CPU bound systems to give better response time to interactive users.

  - When you lower the minor timeslice, you are changing the system's definition of an interactive user. Some tasks which used to finish in one shot at the CPU may take two or three. Therefore some "interactive" processes may actually have worse response time.

  - A very memory intensive application may find that it cannot get it's working set in memory before it runs out of minor timeslice, if paging is heavy and minor timeslice is set low enough. This process will drive system paging higher and will have very bad throughput.

Tuning for Response Time vs. Throughput - con't

- Interactive environments which may benefit from a decrease in ELIGTS:
  - word processing
  - data entry
  - EMACS, editing
  - transaction processing
- Compute bound environments which may benefit from an increase in ELIGTS:
  - CAD (3D modeling)
  - array processing
  - advanced math (number crunching)
  - program compilations
  - report processing
  - graphics

Lesson 13 - USAGE and Related Tuning Topics

Objectives: Upon successful completion of this section, students will be able to:

- Using USAGE, effectively monitor the system such that a valid sample of system performance is obtained.
- Be able to describe what any given field on a standard USAGE report is measuring.
- Determine, from information given in USAGE, when a hardware upgrade is needed.
- Identify a memory bottleneck using information given in USAGE, and suggest methods of eliminating it.
- Given a list of CONFIG directives, identify those that affect wired memory.
- Identify a CPU bottleneck on a system, identify the cause of the bottleneck, and suggest ways of eliminating the bottleneck.
- Determine the optimal value of MAXSCH for a given system, based on information given in USAGE, such that the CPU will be fully utilized, and yet page thrashing will be effectivly throttled.
- Identify an I/O bottleneck on a system, identify the cause of the bottleneck, and suggest ways of eliminating it.
- Set the NLBUF directive to the optimal value for a system, based on information given in USAGE.
- Optimize disk seek time, through eliminating fragmentation, optimizing the use of overlapped seeks, and large partitions.
- Correctly configure disks and controllers for maximum I/O efficiency.

System Tuning Overview

o In order to successfully tune a system, you must do the following:

1) Determine the configuration and workload on the system.

2) Gather information which will indicate how the system is performing.

3) Analyze the information, and determine what, if any, problems exist on the system.

4) Recommend possible solutions to the identified problems.

1 - Identifying Workload

o Information which should be gathered prior to the monitoring:

I. Hardware configuration.
   A. CPU
   B. Memory
   C. Controllers
   D. I/O devices

II. Software configuration.
   A. CONFIG file
   B. PRIMOS.COMI file
   C. PRIMOS Rev.

III. Workload.
   A. Summary of applications
   B. Operating shifts of users/applications
   C. Observed response times
   D. Observed throughput times
   E. Any observed bottlenecks

2 - Monitoring the System

- USAGE is a system metering tool. It can be used by any user at any terminal. The information it generates describes the status and performance of the CPU, main memory, disk subsystems, and other system internals. A sequence of one or more USAGE samples can be generated automatically or manually.

- The format of the USAGE command is:

  USAGE [options]

  Some of the more useful options are:

  -ALL Display all information, including system, user, and disk
  -FREQ n Will generate a sample every n seconds
  -TIMES n Will take n samples

USAGE

05 Aug 85 13:29:51.50 dTIME= 59.87 CPU= 47.00 I/O= 11.67
Up since 05 Aug 85 07:33:04 Monday CPUtot= 6216.94 I/Otot= 4472.34

%CPU	%Idl1	%Idl2	%Error	%I/O	%Ovlp	IO/S	PF/S
78.50	14.85	0.00	2.65	3.25	23.85	10.09	3.01

%Clock	%FNT	%MPC	%PNC	%SLC	%GPPI	%DSK
1.26	0.00	0.00	0.33	0.51	0.00	0.15

%AMLC	%Async	%Sync	%ICS	Segs	Used	Pages	Used	Wired
1.55	0.00	0.00	0.00	2816	1622	8192	8190	480

Locate	%Miss	%Found	%Same	%Share	Loc/S	LM/S
15748	2.07	75.19	22.66	0.08	263.05	5.45

Disk	Qwaits	%Qwait	DMAovr	%DMAovr	Hangs	%Hang
604	0	0.00	0	0.00	0	0.00

Usr	UserID	Mem	Wire	Segs	CPUtime	dCPU	%CPU	I/Otime	dI/O	%I/O
1	SYSTEM	3455	401	209	97.800	0.086	0.144	255.736	0.420	0.702
14	SGW	62	1	23	55.254	0.286	0.477	43.664	2.004	3.347
29	AMS	608	1	43	587.578	36.885	61.611	105.832	3.352	5.599
41	JOHANNA.C	58	1	18	31.930	1.463	2.444	7.812	0.000	0.000
	DONALD	14	1	18	14.145	0.309	0.517	6.708	0.000	0.000
66	BUD.K	30	1	16	56.106	0.233	0.390	44.984	1.148	1.918
79	LARRY.G	35	1	14	11.623	0.206	0.344	13.872	0.112	0.187
87	MARIA.C	40	1	19	16.788	1.852	3.094	16.412	2.464	4.116
93	RICH.D	114	1	22	6.865	0.317	0.530	6.352	0.016	0.027
102	SLAVE$	11	0	3	4.014	0.088	0.147	8.384	0.084	0.140
107	SYSTEM	100	1	11	49.269	0.333	0.556	60.148	0.620	1.036
109	SYSTEM	100	1	11	49.907	0.095	0.159	59.392	0.164	0.274
111	SYSTEM	100	1	11	116.022	0.937	1.565	67.212	0.088	0.147
113	NETMAN	24	1	4	234.703	1.015	1.695	7.488	0.036	0.060
114	RT_SERVER	48	1	9	5.318	0.016	0.027	0.824	0.000	0.000
116	FTP	41	1	12	174.847	0.499	0.833	172.044	0.588	0.982
117	FTPX	25	1	12	277.076	0.698	1.167	405.304	0.000	0.000
122	DAVE.G	53	1	8	6.686	0.530	0.886	2.476	0.156	0.261

Disk	Count	%Count	Time	%Util	Total %Count	Total %Util	Avg time (msec)
'26	261	43.21	5.34	4.46	79.84		
0	205	33.94	4.35	7.27	70.17	6.42	18.99
1	56	9.27	0.99	1.66	9.67	0.88	18.82
'27	343	56.79	6.33	2.64	20.16		
0	189	31.29	3.96	6.61	16.29	1.50	19.05
1	152	25.17	2.35	3.92	3.78	0.40	22.00
2	1	0.17	0.01	0.01	0.04	0.00	11.18
3	1	0.17	0.02	0.03	0.04	0.00	9.96

Monitoring the System - con't

- How long should the samples be?

  How long a sample to take depends entirely on what you will do with it. Here are some examples:

30 secs.	Good for taking a quick look at what's going on. Although this is the shortest recommended length, if you are interested in CPU, you probably could use 15 secs. If you are monitoring disk, 60 to 120 secs would be better.
60 secs.	Good for a quick look at disk utilization.
5 min.	For monitoring a longer period of time (i.e. a day), this would be about the maximum granularity you would want. This is also good for monitoring a particular application.
15 min.	For monitoring long periods, this is a very good granularity.
60 min.	For monitoring over days or weeks, this is probably adequate, although you may want to monitor certain times more closely.

- How long should you run samples?

  To do a full monitoring, you should run samples long enough to cover a full "cycle of activity". This would include a representitive sample of EVERYTHING that is done on your computer.

3 - Analyzing Data

o Here are the steps involved in analyzing data:

1) Identify average values for all the major performance meters.

2) Identify any peaks or valleys.

3) Identify any indicators which exceed certain "critical" values.

4) Identify any major changes from data collected previously (if available).

5) Identify problems or situations which could account for the collected data.

4 - Recommending Solutions

- The two main problems which need to be addressed:

  I. Response time.

  II. System throughput.

  These are the two most noticed by users, and therefore should be used as the base indicators of system performance. First, they should be defined:

  Response Time - The amount of time it takes once a command is issued for the computer to respond.

  System Thoughput - The number of specific tasks which the computer can do in a given amount of time.

- The person performing the tuning should decide which of these is most important to overall performance, as improving one may sometimes degrade the other. All solutions should contain a prediction as to the impact on these two parameters.

- A method of presenting solutions should be picked which will result in easy to read, statistically backed recommendations.

- Recommendations can be:

  - Hardware reconfiguration
  - Software reconfiguration
  - Administrative changes
  - Reprogramming the software

- With all recommendations, you should include a prediction of the effect which the change will have. For your own protection, be CONSERVATIVE.

This Page for Notes

USAGE - CPU Meters

05 Aug 85 13:29:51.50 dTIME= 59.87 CPU= 47.00 I/O= 11.67
Up since 05 Aug 85 07:33:04 Monday CPUtot= 6216.94 I/Otot= 4472.34

%CPU	%Idl1	%Idl2	%Error	%I/O	%Ovlp	IO/S	PF/S
78.50	14.85	0.00	2.65	3.25	23.85	10.09	3.01

%Clock	%FNT	%MPC	%PNC	%SLC	%GPPI	%DSK
1.26	0.00	0.00	0.33	0.51	0.00	0.15

%AMLC	%Async	%Sync	%ICS	Segs	Used	Pages	Used	Wired
1.55	0.00	0.00	0.00	2816	1622	8192	8190	480

Locate	%Miss	%Found	%Same	%Share	Loc/S	LM/S
15748	2.07	75.19	22.66	0.08	263.05	5.45

Disk	Qwaits	%Qwait	DMAovr	%DMAovr	Hangs	%Hang
604	0	0.00	0	0.00	0	0.00

Usr	UserID	Mem	Wire	Segs	CPUtime	dCPU	%CPU	I/Otime	dI/O	%I/O
1	SYSTEM	3455	401	209	97.800	0.086	0.144	255.736	0.420	0.702
14	SGW	62	1	23	55.254	0.286	0.477	43.664	2.004	3.347
29	AMS	608	1	43	587.578	36.885	61.611	105.832	3.352	5.599
41	JOHANNA.C	58	1	18	31.930	1.463	2.444	7.812	0.000	0.000
55	DONALD	14	1	18	14.145	0.309	0.517	6.708	0.000	0.000
68	BUD.K	30	1	16	56.106	0.233	0.390	44.984	1.148	1.918
79	LARRY.G	35	1	14	11.623	0.206	0.344	13.872	0.112	0.187
87	MARIA.C	40	1	19	16.788	1.852	3.094	16.412	2.464	4.116
93	RICH.O	114	1	22	6.865	0.317	0.530	6.352	0.016	0.027
102	SLAVE$	11	0	3	4.014	0.088	0.147	8.384	0.084	0.140
107	SYSTEM	100	1	11	49.269	0.333	0.556	60.148	0.620	1.036
109	SYSTEM	100	1	11	49.907	0.095	0.159	59.392	0.164	0.274
111	SYSTEM	100	1	11	116.022	0.937	1.565	67.212	0.088	0.147
113	NETMAN	24	1	4	234.703	1.015	1.695	7.488	0.036	0.060
114	RT_SERVER	48	1	9	5.318	0.016	0.027	0.824	0.000	0.000
116	FTP	41	1	12	174.847	0.499	0.833	172.044	0.588	0.982
117	FTPX	25	1	12	277.076	0.698	1.167	405.304	0.000	0.000
122	DAVE.G	53	1	8	6.686	0.530	0.886	2.476	0.156	0.261

Disk	Count	%Count	Time	%Util	Total %Count	Total %Util	Avg time (msec)
'26	261	43.21	5.34	4.46	79.84		
0	205	33.94	4.35	7.27	70.17	6.42	18.99
1	56	9.27	0.99	1.66	9.67	0.88	18.82
'27	343	56.79	6.33	2.64	20.16		
0	189	31.29	3.96	6.61	16.29	1.50	19.05
1	152	25.17	2.35	3.92	3.78	0.40	22.00
2	1	0.17	0.01	0.01	0.04	0.00	11.18
3	1	0.17	0.02	0.03	0.04	0.00	9.96

CPU Meters - con't

- %Idl1, %Idl2

  %Idl is the best overall indicator of how busy your CPU is. It measures the amount of CPU used by the backstop process. Since the Backstop only runs when the ready list is empty, any amount of time here is unused CPU.

- %CPU, CPU

  This is the amount of CPU used by all USER processes.

- CPUtime, dCPU, %CPU

  These meters tell for each user the amount of CPU used since login, during the sample, and as a percentage of the delta time.

- %Clock - %ICS

  These meters report the amount of CPU used by the various interrupt processes.

  %Clock The clock process is usually the highest priority process on the Ready List. This means that it will always execute when put on the ready list. It is NOTIFYed via hardware at a very regular interval (usually 1/330 sec). It's only purpose is to increment a number of timers, and NOTIFY other processes if work needs to be done.

  It should take about .25 - 2% on large systems, and 1 - 6% on smaller systems.

- %AMLC
  %Async These processes are the asynchronous driver processes. %AMLC is the for the AMLC boards and %Async is for the ICS boards. The sum of these two should be no more than 10%.

- %Error This is the amount of error in the sample. Error can be accounted for by the fact that certain events such as process exchange, DMx, interrupts, etc cannot be charged to any process. Also, CPU meters are kept in micro-seconds whereas the USAGE meters are in milli-seconds, so there is time lost due to rounding errors.

  This value may be positive or negative. The range should be about -1 - 3%, perhaps higher on smaller machines.

CPU Bottleneck

SYMPTOMS:

- %Idl1 is low (< 5%)
- %CPU is less than 70%
- High % for a system process (%AMLC, %Async, %DSK, etc)
- Response time is very poor
- Throughput is poor, especially CPU intensive jobs

SUSPECTED PROBLEM:

- Improper configuration.
- Bad controller (spurious interrupts).

SOLUTIONS:

- If %AMLC or %Async too high:
  1) Check baud rate of last line on last AMLC board.
  2) Check ICS INTRPT directive for ICS boards.
  3) NOP unused and non-assigned lines.
  4) Attempted high speed input.
  5) Check all devices for garbage characters.
- Check controller with high % for problems.

CPU Bottleneck - con't

SYMPTOMS:

- %IDL1 is low (< 5%)
- %CPU is high (> 90%)
- Response time is very poor
- Throughput is poor, especially CPU intensive jobs

SUSPECTED PROBLEM:

- CPU saturated

SOLUTIONS:

- On older machines, make sure interleaving and wide-word is turned on (if possible).
- Using the %CPU figure for each user, figure out which processes are using the most CPU.
  - If feasible, rewrite CPU intensive software to be more efficient.
  - If feasible, reschedule CPU intensive processes to a non-busy time.
- Use CHAP or ELIGTS command to reorder process execution to favor either throughput or response time.
- CONFIG changes:
  - Set ICS INTRPT to '12.
  - Set last line on AMLC to 110 baud.
- Upgrade CPU.

USAGE - Virtual Memory Meters

05 Aug 85 13:29:51.50 dTIME= 59.87 CPU= 47.00 I/O= 11.67
Up since 05 Aug 85 07:33:04 Monday CPUtot= 6216.94 I/Otot= 4472.34

%CPU	%Idl1	%Idl2	%Error	%I/O	%Ovlp	IO/S	PF/S
78.50	14.85	0.00	2.65	3.25	23.85	10.09	3.01

%Clock	%FNT	%MPC	%PNC	%SLC	%GPPI	%DSK
1.26	0.00	0.00	0.33	0.51	0.00	0.15

%AMLC	%Async	%Sync	%ICS	Segs	Used	Pages	Used	Wired
1.55	0.00	0.00	0.00	2816	1622	8192	8190	480

Locate	%Miss	%Found	%Same	%Share	Loc/S	LM/S
15748	2.07	75.19	22.66	0.08	263.05	5.45

Disk	Qwaits	%Qwait	DMAovr	%DMAovr	Hangs	%Hang
604	0	0.00	0	0.00	0	0.00

Usr	UserID	Mem	Wire	Segs	CPUtime	dCPU	%CPU	I/Otime	dI/O	%I/O
1	SYSTEM	3455	401	209	97.800	0.086	0.144	255.736	0.420	0.702
14	SGW	62	1	23	55.254	0.286	0.477	43.664	2.004	3.347
29	AMS	608	1	43	587.578	36.885	61.611	105.832	3.352	5.599
41	JOHANNA.C	58	1	18	31.930	1.463	2.444	7.812	0.000	0.000
55	DONALD	14	1	18	14.145	0.309	0.517	6.708	0.000	0.000
68	BUD.K	30	1	16	56.106	0.233	0.390	44.984	1.148	1.918
79	LARRY.G	35	1	14	11.623	0.206	0.344	13.872	0.112	0.187
87	MARIA.C	40	1	19	16.788	1.852	3.094	16.412	2.464	4.116
93	RICH.O	114	1	22	6.865	0.317	0.530	6.352	0.016	0.027
102	SLAVE$	11	0	3	4.014	0.088	0.147	8.384	0.084	0.140
107	SYSTEM	100	1	11	49.269	0.333	0.556	60.148	0.620	1.036
109	SYSTEM	100	1	11	49.907	0.095	0.159	59.392	0.164	0.274
111	SYSTEM	100	1	11	116.022	0.937	1.565	67.212	0.088	0.147
113	NETMAN	24	1	4	234.703	1.015	1.695	7.488	0.036	0.060
114	RT_SERVER	48	1	9	5.318	0.016	0.027	0.824	0.000	0.000
116	FTP	41	1	12	174.847	0.499	0.833	172.044	0.588	0.982
117	FTPX	25	1	12	277.076	0.698	1.167	405.304	0.000	0.000
122	DAVE.G	53	1	8	6.686	0.530	0.886	2.476	0.156	0.261

Disk	Count	%Count	Time	%Util	Total %Count	Total %Util	Avg time (msec)
'26	261	43.21	5.34	4.46	79.84		
0	205	33.94	4.35	7.27	70.17	6.42	18.99
1	56	9.27	0.99	1.66	9.67	0.88	18.82
'27	343	56.79	6.33	2.64	20.16		
0	189	31.29	3.96	6.61	16.29	1.50	19.05
1	152	25.17	2.35	3.92	3.78	0.40	22.00
2	1	0.17	0.01	0.01	0.04	0.00	11.18
3	1	0.17	0.02	0.03	0.04	0.00	9.96

Virtual Memory Meters - con't

- Segs, Used — VIRTUAL
  (MAX possible) (current)
  These two fields tell how many segments, or more specifically, Page Map Tables (PMTs) can be allocated, and how many PMTs are actually in use. Segs is NSEG + NVMFS, except that it cannot be greater than 8192.

  If Used approaches Segs, users will shortly be getting

  NO_AVAIL_SEG$ raised at location .....

  If this occurs, you must do at least one of the following:

  - increase NSEG.
  - have users logoff, ICE, or DELSEG.
  - decrease the command breadth for all users.
  - find out (from USAGE) who is using all the segments, and log that user off.

- Pages, Used, Wired — PHYSICAL

  These three fields tell how many pages of physical memory are in your system, how many are being currently used, and how many have been wired (i.e. these pages cannot be paged out to disk).

  Used will often be very close to Pages. Wired should be roughly about 8 - 15% of the total.

  If you attempt to reduce wired memory, the wired field is a very good way to measure your success.

- Mem, Wire, Segs (per user)

  Mem tells the number of physical pages each user owns at the end of a sample. If the system is paging fairly heavily, Mem can approximate the working set required for an application.

  Wire tells how many physical pages this user has wired in memory. This is fairly useless as it only records those pages in DTARs 2 & 3 (segments between 4000 and 7777). Each user will have 1 page wired in segment 6000 for their wired stack. All other pages wired in DTARs 0 & 1 are charged to SYSTEM.

  Segs reports the number of PMTs belonging to each user at the end of the sample. If you are running out of segments, this field can pin-point the user(s) who are using the virtual address space.

Memory Bottleneck

SYMPTOMS:

- %Idl1 is high (> 15%)
- PF/S are high (6 -- 15)
- %Util on paging disks is high (> 60%)
- %Miss is average or low
- Response time is fair to poor, especially when invoking programs
- Throughput is poor, especially programs which require large amounts of memory.

SUSPECTED PROBLEM:

- Page thrashing.

SOLUTIONS:

- Reduce PRIMOS working set (adjust CONFIG, PRIMOS.COMI).
- Redistribute paging (using PAGDEV, ALTDEV, and PRATIO) to non-busy controllers / drives to reduce I/O bottleneck.
- Reduce MAXSCH (covered later in chapter).
- Determine memory hogs and modify to reduce working set.
- Convert static programs to EPFs.
- Add memory.

Reducing Wired Memory

- Wired memory is main memory which cannot be paged to disk by the paging software. The more memory that is wired on the system, the less is available for the working set of the applications on your system. Thus, if your system is paging heavily, any and all pages which you can make available will help the situation.

  1 page of memory is 2048 bytes or 1024 ('2000) words (16 bits).

  NOTE: Typically, if a system is correctly configured, fine tuning with regard to wired memory will usually have a negligible effect. In some borderline cases however, the 10 - 20 pages you save may make the difference between thrashing or not thrashing.

- Why does memory get wired?

  1) Wired memory is required by the hardware configuration of the system. For example:

     - Each async line requires a wired DMQ buffer.
     - Each ICS board requires buffers for ROIPQNM.
     - Each controller present requires a corresponding process (DIMs), and the code for these must be wired.

  2) A certain amount of wired memory is required by PRIMOS, and this amount varies depending on the Rev. For example:

     - Many PRIMOS routines (such as PAGTUR) must be wired.
     - The interrupt handling code (Phantom Interrupt Code) must be wired.
     - The Ready List must be wired.
     - A certain number of Disk Request Blocks must be wired (this is Rev dependent).

Wired Memory - CONFIG Directives

3) Wired memory is required depending on how the system is configured. This applies both to the CONFIG directives, plus the different software products installed.

Here is a list of all the CONFIG directives that wire memory, and what that memory is used for:

NTUSR, NRUSR, NPUSR, NSLUSR

These four directives create processes. Each process will have a '100 word Process Control Block wired.

SMLC ON, SMLC CNTRLR, SMLC (or SYNC)

Each of these directives will wire down a DIM process, plus the minimum buffers needed to communicate with the devices.

NVMFS

NVMFS creates a table in memory called the Active Segment Table. Each entry in this table is 28 words in size. Thus, making the table 1024 entries will wire 28 pages.

VPSD

This directive wires the V mode Prime Symbolic Debugger in memory, so that it can be used. This wires about 4 pages of memory.

NLBUF

This directive allocates the number of LOCATE buffers. It will immediately wire a 22 word Buffer Control Block for each buffer. In addition, although LOCATE buffers are allocated dynamically, they tend to be virtually wired, and thus unavailable for program working sets.

Wired Memory - CONFIG Directives - con't

AMLBUF, REMBUF, ASRBUF — SINCE WE ARE SITTING AROUND MY BOX ANYWAY MAYBE MEMORY CAN BE SHORTENED

These three directives wire terminal buffers. They have the largest potential for wasting wired memory. To configure these buffers correctly, you should refer to the documentation, or take one of the courses offered on system administration.

AMLIBL, ICS INPQSZ

AMLIBL configures the size of the tumble table buffers, which are used for character input from the AMLC controllers. There is one tumble table per AMLC board. ICS INPQSZ performs the same function for ICS boards. There are two tables per eight lines.

LOUTQM

These directives affect wired memory in that they will force logout processes which exceed inactive or elapsed time limits. This is important since logged-in processes do require wired memory for their SDT, RO wired stack, PMTs, etc.

Wired Memory - Product Requirements

4) Wired memory required by system use. Every product which is installed, and every user which logs onto the system, will cause memory to be wired down. Here are some instances:

- Every product which is installed has the potential of using wired memory. In particular, every product which is installed using the SHARE command will allocate and wire down memory for the PMTs. Therefore, do not SHARE products unless you intend on using them.

  Examples:

```
MIDASPLUS      - 4 pages
ROAM           - 2 pages
PRISAM         - 1 page
INFORMATION    - 6 pages
OAS            - 3 pages
EMACS          - 2 pages
CBL            - 3 pages
PRIMENET       - 43 pages  /* with 30 nodes configured
```

  Here are some other products with wired memory requirements:

- Networks

  Depending on how many nodes are configured, and whether you have full duplex, ring net, etc, you will wire approx 20 - 100 pages of memory for the data base. A smaller configuration will help keep this at a minimum.

- Data Base Products

  Both ROAM and MIDASPLUS now have their own disk block buffering system. Although this will not cause memory to be wired directly, these buffers will be virtually wired when the products are used, and thus unavailable for application working sets.

- PRIMIX

  Due to the way PRIMIX creates child processes, PRIMIX tends to have a large impact on virtual memory requirements, especially PMTs.

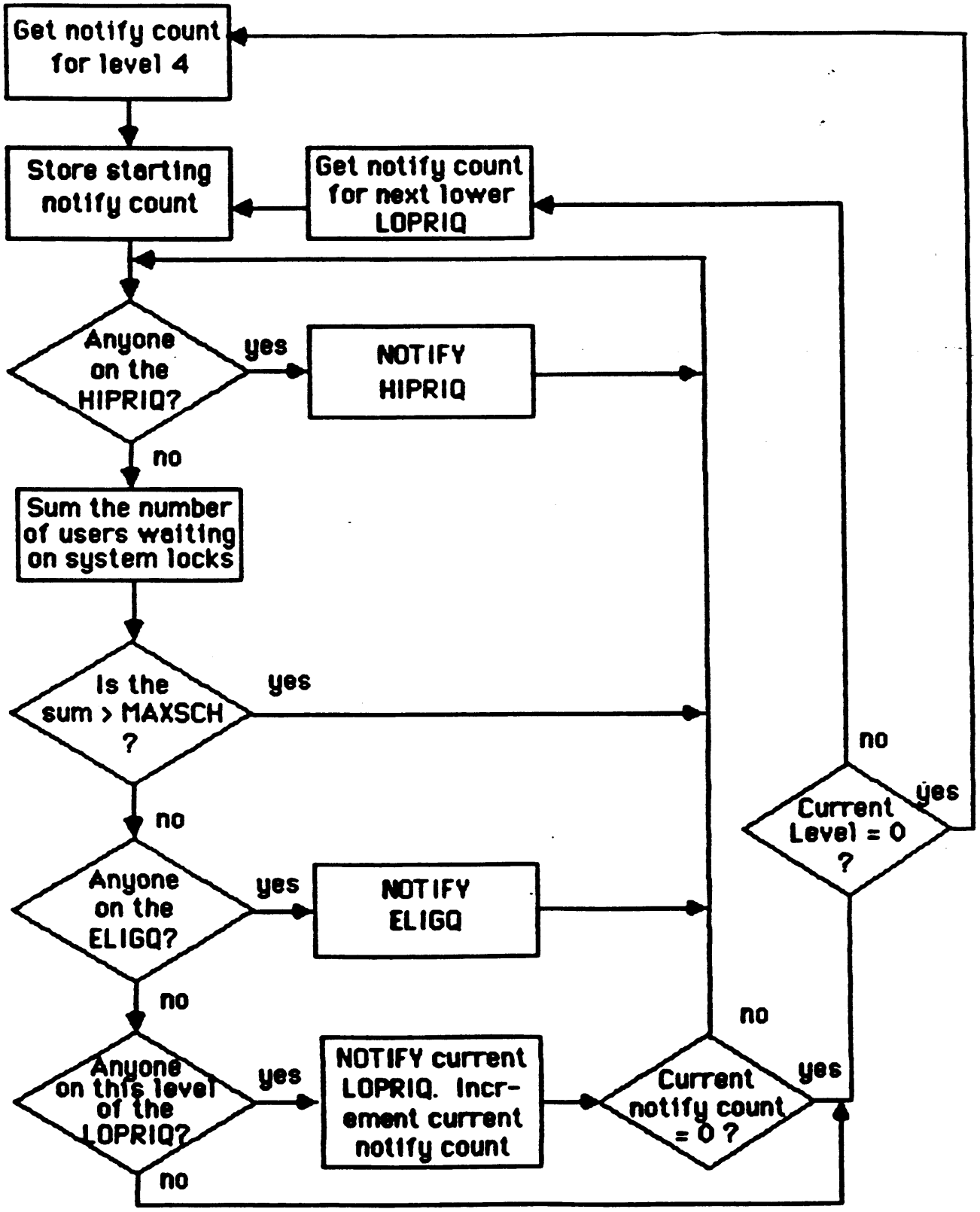
Wired Memory - System Usage

- For each user who logs onto the system, the following memory in wired:
  - SDTs
  - 1 page in segment 6000
  - PMTs for each segment used

    PMTs represent most of the wired memory requirements. Therefore, some things to know about PMTs:
    - DELSEG, ICE, and LO deletes PMTs and releases all pages associated with the segment.
    - EPFs will release PMTs when they are removed.
    - EPFs share PMTs for procedure, and therefore require less system memory.

- On the average, each user who logs in will require 3 - 6 wired pages.

BACKSTOP Flowchart - Re-visited



MAXSCH

- MAXSCH is a throttle on the number of processes competing for disk resources. The reason is to prevent page thrashing (where all processes are paging out each others working set). It does this by summing the following semaphores:

  PAGSEM - # of processes waiting for pages in transition.
  LOCSEM - # of processes waiting for a BCB (LOCATE).
  (DSKBLK - DSKQCT) - # of processes waiting for a disk I/O.
  UFDLOC - # of processes accessing a UFD.
  UTLOC - # of processes accessing a unit table.
  RATLOC - # of processes accessing a DSKRAT.

  The sum of these semaphores is then compared to the value MAXSCH. If the sum is greater, the backstop process will not service the ELIGQ, LOPRIQs, or the IDLEQ.

- Starting at Primos revision 19.0 MAXSCH is calculated as follows:

  MAXSCH = (megabytes_of_memory + 3) * x + y

  where, x is 1.2 if there exists an alternate device on a different controller than the primary device, otherwise it is 1.
  y is 1 if CPU is a P850, otherwise it is 0.

- The optimal value of MAXSCH is application dependent, hence there is no hard and fast formula to determine its value. Therefore, it is a operator command.

  rule of thumb:

$$\text{MAXSCH} = \frac{\text{Physical-Memory-Size} - \text{PRIMOS-locked-memory}}{\text{average-job-size}}$$

Tuning MAXSCH

- Symptoms of MAXSCH being set too low:
  - %Idl1 is high (> 15%)
  - PF/S are avg or low for your system (< 5 on small systems, < 10 on large systems)

- By increasing MAXSCH, you should see %Idl1 decrease (you are making it easier for processes to get out of the hold queues and back on the Ready List). As long is %Idl1 decreases, you are doing some good.

  PF/S should increase slightly if at all. At some point, PF/S may increase dramatically, and %Idl1 will also increase. You are now thrashing, and you should back MAXSCH off this mark.

- If %Idl1 begins to increase, you should try decreasing MAXSCH until %Idl1 gets to it's lowest point. %Idl1 is always the indicator.

- Remember to look at long samples. The "optimal" value of MAXSCH will change constantly through the day as the application mix changes. You are trying to find the best overall value for your average application mix.

This Page for NOTES

USAGE - Disk Information

05 Aug 85 13:29:51.50 dTIME= 59.87 CPU= 47.00 I/O= 11.67
Up since 05 Aug 85 07:33:04 Monday CPUtot= 6216.94 I/Otot= 4472.34

%CPU	%Idl1	%Idl2	%Error	%I/O	%Ovlp	IO/S	PF/S
78.50	14.85	0.00	2.65	3.25	23.85	10.09	3.01

%Clock	%FNT	%MPC	%PNC	%SLC	%GPPI	%DSK
1.26	0.00	0.00	0.33	0.51	0.00	0.15

%AMLC	%Async	%Sync	%ICS	Segs	Used	Pages	Used	Wired
1.55	0.00	0.00	0.00	2816	1622	8192	8190	480

Locate	%Miss	%Found	%Same	%Share	Loc/S	LM/S
15748	2.07	75.19	22.66	0.08	263.05	5.45

Disk	Qwaits	%Qwait	DMAovr	%DMAovr	Hangs	%Hang
604	0	0.00	0	0.00	0	0.00

Usr	UserID	Mem	Wire	Segs	CPUtime	dCPU	%CPU	I/Otime	dI/O	%I/O
1	SYSTEM	3455	401	209	97.800	0.086	0.144	255.736	0.420	0.702
14	SGW	62	1	23	55.254	0.286	0.477	43.664	2.004	3.347
29	AMS	608	1	43	587.578	36.885	61.611	105.832	3.352	5.599
41	JOHANNA.C	58	1	18	31.930	1.463	2.444	7.812	0.000	0.000
55	DONALD	14	1	18	14.145	0.309	0.517	6.708	0.000	0.000
68	BUD.K	30	1	16	56.106	0.233	0.390	44.984	1.148	1.918
79	LARRY.G	35	1	14	11.623	0.206	0.344	13.872	0.112	0.187
87	MARIA.C	40	1	19	16.788	1.852	3.094	16.412	2.464	4.116
93	RICH.O	114	1	22	6.865	0.317	0.530	6.352	0.016	0.027
102	SLAVE$	11	0	3	4.014	0.088	0.147	8.384	0.084	0.140
107	SYSTEM	100	1	11	49.269	0.333	0.556	60.148	0.620	1.036
109	SYSTEM	100	1	11	49.907	0.095	0.159	59.392	0.164	0.274
111	SYSTEM	100	1	11	116.022	0.937	1.565	67.212	0.088	0.147
113	NETMAN	24	1	4	234.703	1.015	1.695	7.488	0.036	0.060
114	RT_SERVER	48	1	9	5.318	0.016	0.027	0.824	0.000	0.000
116	FTP	41	1	12	174.847	0.499	0.833	172.044	0.588	0.982
117	FTPX	25	1	12	277.076	0.698	1.167	405.304	0.000	0.000
122	DAVE.G	53	1	8	6.686	0.530	0.886	2.476	0.156	0.261

Disk	Count	%Count	Time	%Util	Total %Count	Total %Util	Avg time (msec)
'26	261	43.21	5.34	4.46	79.84		
0	205	33.94	4.35	7.27	70.17	6.42	18.99
1	56	9.27	0.99	1.66	9.67	0.88	18.82
'27	343	56.79	6.33	2.64	20.16		
0	189	31.29	3.96	6.61	16.29	1.50	19.05
1	152	25.17	2.35	3.92	3.78	0.40	22.00
2	1	0.17	0.01	0.01	0.04	0.00	11.18
3	1	0.17	0.02	0.03	0.04	0.00	9.96

USAGE - Disk Information con't

- I/O counters:

Disk	- the number of actual I/Os during the sample period
Count	- the number of actual I/Os charged to each controller and drive
%Count	- Count / Disk
IO/S	- Disk / dTIME

- Error indicators:

Qwaits	- the number of times a process waited for a QRB
DMAovr	- the number of DMA overruns
Hangs	- the number of times a drive did not finish a seek

- I/O time used:

I/O=	- the total time spent on disk I/Os during this dTIME
Time	- the total time spent on disk I/Os for each controller and each drive
I/Otime	- the total time spent on disk I/Os for each user since login
dI/O	- the total time spent on disk I/Os for each user during the dTIME
%I/O	- dI/O / dTIME

- I/O utilization:

%Util (drive)	- Time / dTIME
%Util (controller)	- Time / dTIME / # drives on controller
%I/O	- I/O / dTIME / total # of drives

- Since coldstart:

Total %Count	- cumulative Count / cumulative total Count
Total %Util	- cumulative Time / cumulative dTIME
Avg Time	- cumulative Time / cumulative Count

IO/S - LM/S = PHSICAL I/O FOR PAGING

PF/S = CALLS TO PAGFLT

1 CALL TO PAGFLT RETURNS

1 PAGE TO USER IF PRIMOS IS MEMORY AVAIL

AND PROCESSOR PREPAGE IF NO MEMORY IS AVAILABLE.

Tuning Disk I/O

- Wait time can be reduced in the following ways:

  - Supply an adequate number of QRBs. If you are seeing Qwaits and you are on a pre-Rev 19.3 system, you will get an advantage by going up Rev.

  - Balance disk usage. Since a controller can overlap seeks, the more balanced the disk usage is, the more concurrent I/Os a controller can handle without long waits. Here are some strategies:

    - For short term balancing (< 6 hours), use %Util for the drives. Idealistically, you would want %Util to be equal on all drives.

    - For long term balancing (> 6 hours), use Total %Count. This will tell you overall what drives are being the most accessed. This will include ALL I/Os since coldstart.

    To balance disk usage, you must move accounts or files to non-busy disks. One of the busiest disks on the system is the paging disk. Using ALDEV and PRATIO, you can balance two disks without moving data by putting more pressure on either PAGDEV or ALTDEV according to use.

  - Buy additional disk drives / controllers. If all of your drives are being heavily utilized (%I/O > 70% and %Idl1 > 20%), your wait times can be very long per request. The only solutions are to reduce the number of I/O requests or buy additional drives so that you can balance the I/O over more drives.

Tuning Disk I/O - con't

o Seek time can be reduced in the following ways:

- Reduce fragmentation. Normally, a new file will have its records layed down cylinder by cylinder so that sequential access will have minimal seek time. Over time, with additions and deletions of records, a file will get fragmented around the disk. This can be identified by an increase in Avg Time for a disk. Usually, the average seek time should be about 10 - 20 ms. The solution is to backup the disk logically (using MAGSAV, BRMS, or COPY), MAKE the disk, and copy the data back.

- Large logical partitions. Unfortunatly, logical partitioning is done by surface rather than by cylinder. This means a small partition has small cylinders (eg. a 2 surface partition has 18 records / cylinder whereas a 10 surface partition has 90 records / cylinder). Therefore, the larger a partition is, the lower the amount of seeking that will be necessary during sequential access.

- Have one controller per drive (pre-Rev 19.3). Before Rev 19.3, a controller with more than one drive seeking would frequently wait for the wrong drive to finish. This problem can be solved by having less drives per controller, or by upgrading to Rev 19.3 where the problem has be fixed.

- Keep PAGDEV and ALTDEV off your major application disks if possible. Paging tends to get hit often and randomly, and will therefore interfere with sequential file operations.

Disk Tuning - Bottlenecked Disk

SYMPTOMS:

- %Idl1 is high (> 20%).
- Throughput is poor, especially for certain applications accessing one particular disk.
- %Util for some drives are much higher than on others.
- %I/O may be low, average, or high, depending on how many drives you have.

SUSPECTED PROBLEM:

- Bottlenecked disk drive.
- Bottlenecked disk controller (before Rev 19.3).

SOLUTIONS:

- Move heavily used directories and/or files to other drives / controllers.
- Change paging load using PAGDEV, ALTDEV, and PRATIO.
- If %I/O is also high, buy more disks / controllers so as to balance your I/O.
- Decrease number of applications using a particular file or account.
- Reprogram applications to use less I/O.
- Minimize seek time on the heavily used disks.

Disk Tuning - Inefficient Seeking

SYMPTOMS:

- %Idl1 is high (> 20%)
- %Util is much higher that %Count for a particular drive.
- Average seek time high (> 20ms).
- Throughput is poor, especially on I/O intensive applications.
- Throughput on I/O intensive jobs is degrading.

SUSPECTED PROBLEM:

- Disk fragmentation.
- Small partitions.
- Inefficient I/O in applications.

SOLUTIONS:

- Remake disk after logical backup.
- Make partitions as large as possible.
- Put paging and application files on different disks.
- Rewrite applications to be more I/O efficient.
- If %I/O is also high, buy additional drives so as to balance I/O between more drives.
- If pre Rev 19.3, buy additional controllers or upgrade Rev.

USAGE - LOCATE Information

05 Aug 85 13:29:51.50 dTIME= 59.87 CPU= 47.00 I/O= 11.67
Up since 05 Aug 85 07:33:04 Monday CPUtot= 6216.94 I/Otot= 4472.34

%CPU	%Idl1	%Idl2	%Error	%I/O	%Ovlp	IO/S	PF/S
78.50	14.85	0.00	2.65	3.25	23.85	10.09	3.01

%Clock	%FNT	%MPC	%PNC	%SLC	%GPPI	%DSK
1.26	0.00	0.00	0.33	0.51	0.00	0.15

%AMLC	%Async	%Sync	%ICS	Segs	Used	Pages	Used	Wired
1.55	0.00	0.00	0.00	2816	1622	8192	8190	480

Locate	%Miss	%Found	%Same	%Share	Loc/S	LM/S
15748	2.07	75.19	22.66	0.08	263.05	5.45

Disk	Qwaits	%Qwait	DMAovr	%DMAovr	Hangs	%Hang
604	0	0.00	0	0.00	0	0.00

Usr	UserID	Mem	Wire	Segs	CPUtime	dCPU	%CPU	I/Otime	dI/O	%I/O
1	SYSTEM	3455	401	209	97.800	0.086	0.144	255.736	0.420	0.702
14	SGW	62	1	23	55.254	0.286	0.477	43.664	2.004	3.347
29	AMS	608	1	43	587.578	36.885	61.611	105.832	3.352	5.599
41	JOHANNA.C	58	1	18	31.930	1.463	2.444	7.812	0.000	0.000
55	DONALD	14	1	18	14.145	0.309	0.517	6.708	0.000	0.000
68	BUD.K	30	1	16	56.106	0.233	0.390	44.984	1.148	1.918
79	LARRY.G	35	1	14	11.623	0.206	0.344	13.872	0.112	0.187
87	MARIA.C	40	1	19	16.788	1.852	3.094	16.412	2.464	4.116
93	RICH.O	114	1	22	6.865	0.317	0.530	6.352	0.016	0.027
102	SLAVE$	11	0	3	4.014	0.088	0.147	8.384	0.084	0.140
107	SYSTEM	100	1	11	49.269	0.333	0.556	60.148	0.620	1.036
109	SYSTEM	100	1	11	49.907	0.095	0.159	59.392	0.164	0.274
111	SYSTEM	100	1	11	116.022	0.937	1.565	67.212	0.088	0.147
113	NETMAN	24	1	4	234.703	1.015	1.695	7.488	0.036	0.060
114	RT_SERVER	48	1	9	5.318	0.016	0.027	0.824	0.000	0.000
116	FTP	41	1	12	174.847	0.499	0.833	172.044	0.588	0.982
117	FTPX	25	1	12	277.076	0.698	1.167	405.304	0.000	0.000
122	DAVE.G	53	1	8	6.686	0.530	0.886	2.476	0.156	0.261

Disk	Count	%Count	Time	%Util	Total %Count	Total %Util	Avg time (msec)
'26	261	43.21	5.34	4.46	79.84		
0	205	33.94	4.35	7.27	70.17	6.42	18.99
1	56	9.27	0.99	1.66	9.67	0.88	18.82
'27	343	56.79	6.33	2.64	20.16		
0	189	31.29	3.96	6.61	16.29	1.50	19.05
1	152	25.17	2.35	3.92	3.78	0.40	22.00
2	1	0.17	0.01	0.01	0.04	0.00	11.18
3	1	0.17	0.02	0.03	0.04	0.00	9.96

This Page for NOTES:

PRIMOS LOCATE Mechanism

- An LOCATE (or associative) buffer is a main memory copy of a disk record. LOCATE buffers are a means of reducing the number of disk accesses needed for logical file access.

- Multiple logical reads to one physical record may require only one disk access. Multiple logical writes to one physical record may require only one disk read and one disk write.

- Each user can own one LOCATE buffer. An owned LOCATE buffer is wired in memory. Previously owned LOCATE buffers remain in memory until they are again owned (wired), or deleted from memory.

- If a LOCATE buffer has been modified, it is written back to the file system disk by user 1 and/or when it is deleted from memory. User 1 copies all modified LOCATE buffers to the file system disk once a minute.

- USAGE LOCATE buffering information:

  Locate - the number of calls to the LOCATE mechanism. This is roughly the number of I/O requests made by applications.

  %Miss - the percentage of calls to LOCATE in which the requested disk record was not in memory.

  %Found - the percentage of calls to LOCATE in which the requested disk record was found in memory, but was unowned.

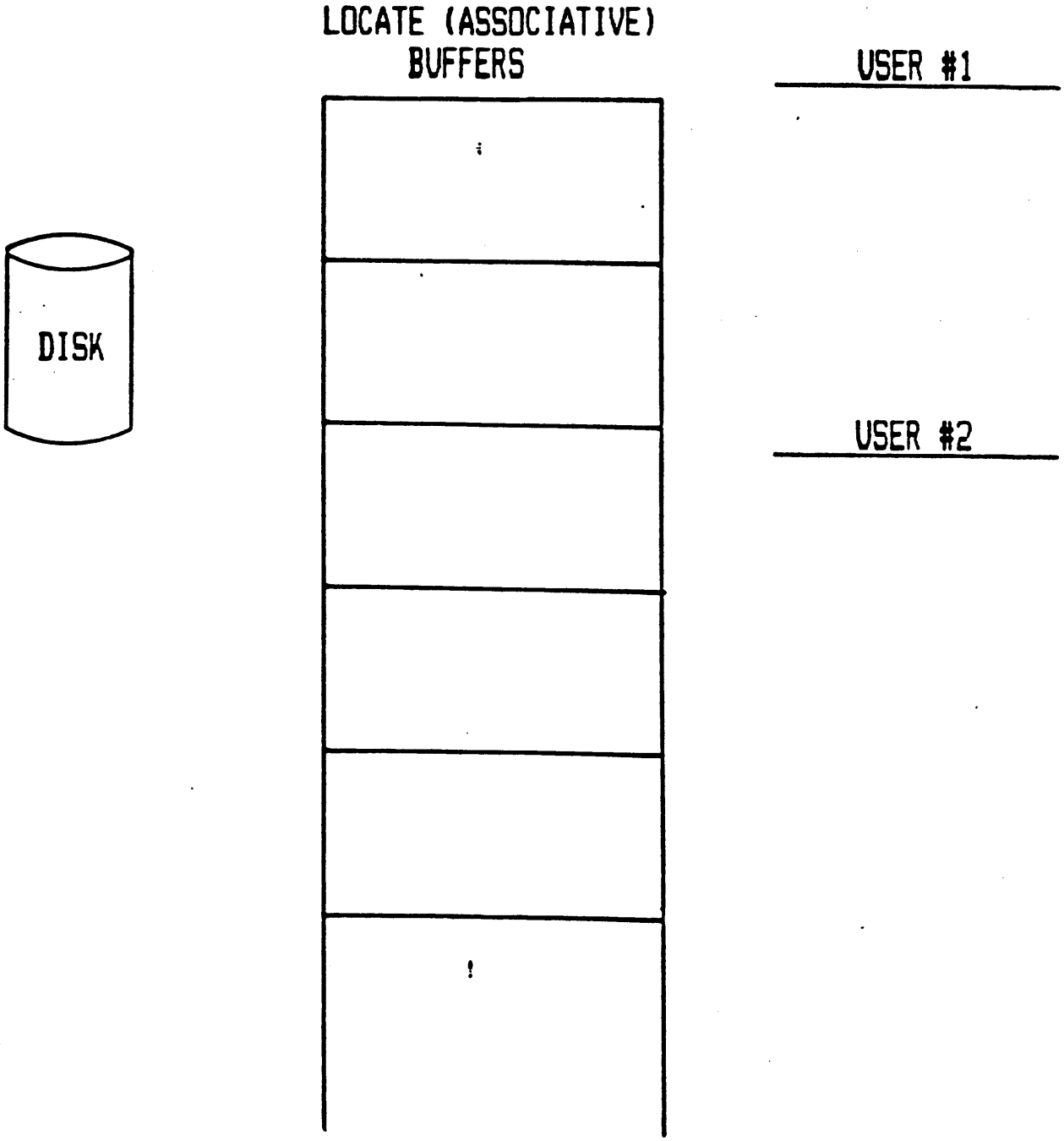
  %Same - the percentage of calls to LOCATE in which the requested disk record was found in memory, and the requesting process already owned it.

  %Share - the percentage of calls to LOCATE in which the requested disk record was found in memory, and the buffer was owned by another process.

  Loc/S - The average number of calls to LOCATE each second during the sample.

  LM/S - The average number of LOCATE misses per second during the sample period. It should be remembered that each LOCATE miss can result in 1 OR 2 actual I/Os.

LOCATE Mechanism



Using NLBUF

- The number of LOCATE buffers configured on the system can impact I/O performance. As of Rev 19.1, the number of buffers is configurable.

  - NLBUF CONFIG directive, range 8 - 256, default 64.
  - 22 word Buffer Control Block (BCB) wired at cold start for each buffer.

  Even though LOCATE buffers are only wired when owned, most LOCATE buffers are "virtually" wired into memory. Thus, it is a good idea to think of ALL configured buffers as wired memory.

- Changing NLBUF will affect both PF/S and %Miss. Since both of these directly relate to disk I/O, the idea is to decrease one without adversely affecting the other. Therefore, there are two major symptoms which can dictate a change in NLBUF:

  High %Miss and low or avg PF/S ..... increase NLBUF

  Low %Miss and high PF/S ..... decrease NLBUF

  NOTE: LM/S is a very important indicator in terms of validating the percentages. %Miss does not tell you anything without LM/S.

  Example #1

%Miss	Loc/S	LM/S
15.09	127.90	19.30

  Example #2

%Miss	Loc/S	LM/S
15.09	22.87	3.45

  Obviously, the 15% miss rate in example #1 is affecting the system much more than the 15% miss rate in example #2.

- Applications can take advantage of LOCATE by having small, sequentially accessed logical records. %Miss is most often a reflection of the application.

LOCATE Tuning

SYMPTOMS:

- %Idl1 is low (< 10%).
- %I/O is high (> 60%).
- %Util for a drive is high (> 80%).
- LO/S is average or high (> 30).
- %MISS is high (> 20%).
- Throughput is poor, especially on I/O intensive jobs.

SUSPECTED PROBLEM:

- LOCATE buffer thrashing.
- Inefficient LOCATE usage by applications.

SOLUTIONS:

- Increase NLBUF.
- Reprogram applications to take advantage of LOCATE mechanism.
- Reschedule I/O intensive applications to a less busy time.
- Decrease paging requirements if PF/S are average or high.
- Put paging on different disk from main applications.
- Balance I/O between all drives.
- Buy additional drives so I/O can balanced between more drives.

USAGE - ROAM Buffer Information

05 Aug 85 13:29:51.50 dTIME= 59.87 CPU= 47.00 I/O= 11.67
Up since 05 Aug 85 07:33:04 Monday CPUtot= 6216.94 I/Otot= 4472.34

%CPU	%Idl1	%Idl2	%Error	%I/O	%Ovlp	IO/S	PF/S
78.50	14.85	0.00	2.65	3.25	23.85	10.09	3.01

%Clock	%FNT	%MPC	%PNC	%SLC	%GPPI	%DSK
1.26	0.00	0.00	0.33	0.51	0.00	0.15

%AMLC	%Async	%Sync	%ICS	Segs	Used	Pages	Used	Wired
1.55	0.00	0.00	0.00	2816	1622	8192	8190	480

Locate	%Miss	%Found	%Same	%Share	Loc/S	LM/S
15748	2.07	75.19	22.66	0.08	263.05	5.45

Blki/o	Read	%Read	Write	%Write	Awrite	%Awrite	Blk/S
10	5	50.00	0	0.00	5	50.00	4.54

Disk	Qwaits	%Qwait	DMAovr	%DMAovr	Hangs	%Hang	Asyio	%Asyio
604	0	0.00	0	0.00	0	0.00	0	0.00

Usr	UserID	Mem	Wire	Segs	CPUtime	dCPU	%CPU	I/Otime	dI/O	%I/O
1	SYSTEM	3455	401	209	97.800	0.086	0.144	255.736	0.420	0.702
14	SGW	62	1	23	55.254	0.286	0.477	43.664	2.004	3.347
29	AMS	608	1	43	587.578	36.885	61.611	105.832	3.352	5.599
41	JOHANNA.C	58	1	18	31.930	1.463	2.444	7.812	0.000	0.000
55	DONALD	14	1	18	14.145	0.309	0.517	6.708	0.000	0.000
68	BUD.K	30	1	16	56.106	0.233	0.390	44.984	1.148	1.918
79	LARRY.G	35	1	14	11.623	0.206	0.344	13.872	0.112	0.187
87	MARIA.C	40	1	19	16.788	1.852	3.094	16.412	2.464	4.116
93	RICH.D	114	1	22	6.865	0.317	0.530	6.352	0.016	0.027
102	SLAVE$	11	0	3	4.014	0.088	0.147	8.384	0.084	0.140
107	SYSTEM	100	1	11	49.269	0.333	0.556	60.148	0.620	1.036
109	SYSTEM	100	1	11	49.907	0.095	0.159	59.392	0.164	0.274
111	SYSTEM	100	1	11	116.022	0.937	1.565	67.212	0.088	0.147
113	NETMAN	24	1	4	234.703	1.015	1.695	7.488	0.036	0.060
114	RT_SERVER	48	1	9	5.318	0.016	0.027	0.824	0.000	0.000
116	FTP	41	1	12	174.847	0.499	0.833	172.044	0.588	0.982
117	FTPX	25	1	12	277.076	0.698	1.167	405.304	0.000	0.000
122	DAVE.G	53	1	8	6.686	0.530	0.886	2.476	0.156	0.261

ROAM Buffer Manager

- At Rev 20.0, the ROAM Buffer manager was added. The motivation for this mechanism is:
  - The PRWF$$ / LOCATE mechanism which it replaces is very generalized and not very efficient for ROAM and the data base products.
  - The data base products need to be able to directly manipulate the disk buffers in order to do prioritizing.
- All ROAM buffers are 2kb in size.
- ROAM buffers are only wired during an I/O operation.

USAGE Statistics for ROAM Buffers

- A new line of statistics for ROAM buffer pool access has been added to USAGE output. These statistics will give you some idea of how much the ROAM buffers are being accessed vs the LOCATE buffers.

Blki/o	Read	%Read	Write	%Write	Awrite	%Awrite	Blk/S
10	5	50.00	0	0.00	5	50.00	4.54

asyio	%asynio
0	0.00

Stats	Meaning
Blki/o	Total number of logical block I/O operations in the sampling period
Read	Total number of read block I/O operations in the sampling period
%Read	The percentage of the total number of block I/O operations that were read operations
Write	Total number of synchronous write block I/O operations in the sampling period
%Write	The percentage of the total number of block I/O operations that were synchronous write operations
Awrite	Total number of asynchronous write operations in the sampling period
%Awrite	The percentage of the total number of block I/O operations that were asynchronous write operations
Blk/s	The average number of block I/Os per second during the sampling period.
asyio	The number of async write requests that DISKIO handled.
%asyio	The percentage of the total number of async write requests in the sampling period

Title: Disk I/O Balancing.

Objectives: Upon sucessful completion of this lesson, students will be able to:

- Shift the layout of directories and files across disks and controllers to increase overall disk input/output throughput.
- Set the PAGDEV, ALTDEV, and PRATIO configuration directives to reduce the average page fault time and increase overall disk input/output throughput.

Task: Given a description of a system's logical disk structure answer questions on improving disk throughput.

Conditions: Using any available course documentation.

A system administrator at a software applications shop has been having real I/O problems. The problem for you is to configure her system to maximize I/O efficiency.

- The system is a 9755, running PRIMOS revision 20.2
- There are 3MB of main memory.
- There are two disk controllers.
  - drive 0 on the first controller is a 80MB disk.
  - drive 0 on the second controller is a 80MB disk.
  - drive 1 on the second controller is a 300MB disk.
- All backup is done to tape.
- There are 64 lines configured.

PAGDEV is DISK10, 2nd controller, drive 1, 3 heads
COMDEV is DISK00, 1st controller, drive 0, 2 heads

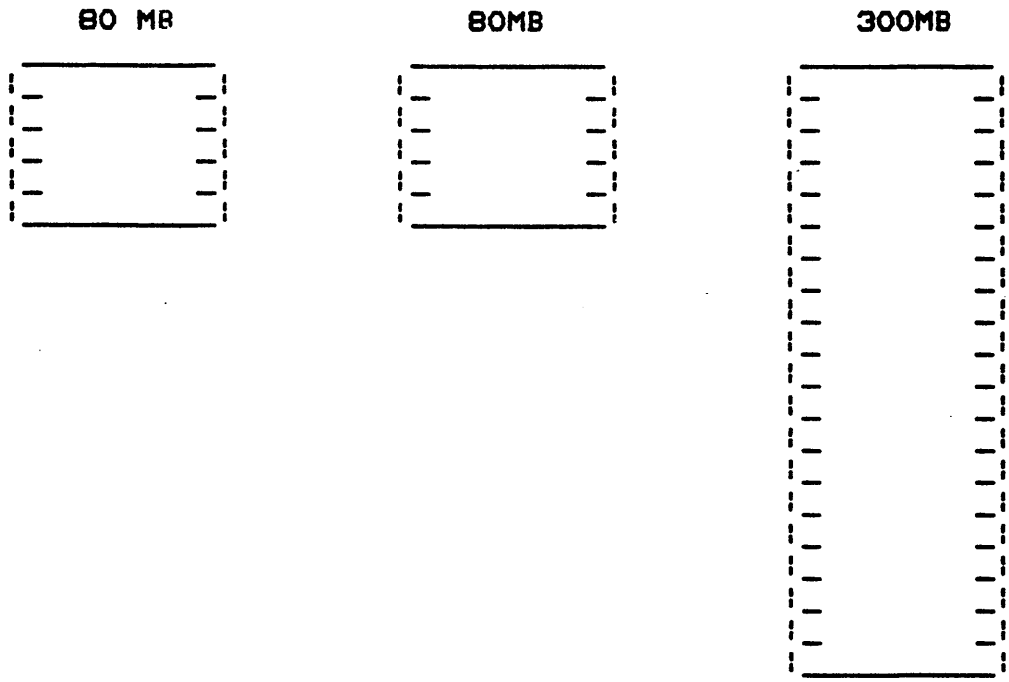
Here is the current logical disk layout:

VOLUME ID	TOTAL RECS	FREE RECS	% FULL	COMMENTS			
DISK00	14814	741	95.0	CONTROLLER 1	DRIVE 0	2 heads	
DISK01	14814	741	95.0			2 heads	
DISK02	7407	741	90.0			1 head	
DISK03	14814	741	95.0	CONTROLLER 2	DRIVE 0	2 heads	
DISK04	14814	1482	90.0			2 heads	
DISK05	7407	741	90.0			1 head	
DISK06	29628	1482	95.0		DRIVE 1	4 heads	
DISK07	29628	2963	90.0			4 heads	
DISK08	29628	1482	95.0			4 heads	
DISK09	29628	26665	10.0			4 heads	

Here is the approximate MFD usage given you by the System Administrator:

DISK	DESCRIPTION	USAGE
DISK00	Command device	Heavy
DISK01	Library of old software revs	Light
DISK02	R&D programmer accounts	Heavy
DISK03	R&D software source library	Moderate
DISK04	R&D misl	Moderate
DISK05	Administrative misl	Light
DISK06	Secretarial accounts using word proccessing	Heavy
DISK07	Payroll software - used only on Thurs and Fri	Heavy/light
DISK08	Accounting software	Heavy
DISK09	Executive accounts	Light

1) Draw in lines dividing the physical disk pack into logical partitions. Indicate where the data from the original partitions would be copied by writing in the old partition name(s) in the new partitions you have drawn in. Make sure to indicate which disks should go on which controllers.



PRATIO (if decided to use) =

Appendix A - Acronyms

ACAT	Access Catagory
ACL	Access Control List
ALU	Arithmetic Logic Unit
AMLC	Asynchronous Multi-Line Controller
AMLDIM	AMLC Device Interface Manager
AP	Arguement Pointer
ARGT	Arguement Transfer
ASD	Auto Speed Detect
AST	Active Segment Table
BADSPT	Badspot
BCB	Buffer Control Block
BMA	Bus Memory Address
BMC	Bus Memory Control
BMD	Bus Memory Data
BOL	Beginning of List Pointer
BPA	Bus Peripheral Address
BPC	Bus Peripheral Control
BPD	Bus Peripheral Data
BRA	Beginning Record Address
CALF	Call Fault Handler
CAM	Contiguous Access Method
CHAP	Change Priority
CPU	Central Processing Unit
CRSx	Current Register Set
CTI	Character Timed Interrupt
DAM	Direct Access Method
DB	Directory Block
DMA	Direct Memory Access
DMC	Direct Memory Channel (or Control)
DMQ	Direct Memory Queue
DMT	Direct Memory Transfer
DMx	Direct Memory Transfer
DSKRAT	Disk Record Availability Table
DTAR	Descriptor Table Address Register
DYNT	Dynamic
ECB	Entry Control Block
ECL	Emitter Coupled Logic
ELIGQ	Eligibility Queue
ELIGTS	Eligibility Timeslice (the minor timeslice)
EOL	End of List Pointer
EOR	End of Range
EPF	Executable Program Format
ERP	EPF Relocatable Pointer
GPPI	General Purpose Peripheral Interface
HIPRIQ	High Priority Queue
HMAP	Hardware Map
I/O	Input/Output
ICS	Intelligent Controller Subsystem
IDLEQ	Idle Queue
IOTLB	Input/Output Table Lookaside Buffer
IRB	Input Ring Buffer
LB	Link Base
LOPRIQ	Low Priority Queue
MAXSCH	Maximum Scheduled (Processes)

MDLC	Multi Line Data Link Controller
MFD	Master File Directory
MMAP	Memory Map
NLBUF	Number Locate Buffers
ORB	Output Ring Buffer
P-CTR	Program Counter
PABORT	Process Abort Handler
PAGTUR	Page Turner
PAVCTR	Page Available Counter
PB	Procedure Base
PCB	Process Control Block
PCL	Procedure Call
PDU	Power Distribution Unit
PIC	Phantom Interrupt Code
PIO	Programmed Input/Output
PMA	Prime Macro Assembler
PMT	Page Map Table
PNC	Prime Node Controller
PPA	Pointer to Process A
PPB	Pointer to Process B
PPN	Physical Page Number
PRTN	Procedure Return
PUDCOM	Per User Data Common
QAMLC	AMLC board using DMQ for character output
QB	Quota Block
QRB	Queue Request Block
ROIPQNM	Ring 0 Interprocess Queueing and Notification Mechanism
RSx	Register Set
SAM	Sequential Access Method
SB	Stack Base
SDT	Segment Descriptor Table
SDW	Segment Descriptor Word
SEGDAM	Segmented Direct Access Method
SEGSAM	Segmented Sequential Access Method
SMLC	Synchronous Multi Line Controller
SMT	Segment Mapping Table
STLB	Segment Table Lookaside Buffer
TTL	Transistor to Transistor Logic
UART	Universal Asynchronous Receive and Transmit Buffer
UII	Unimplemented Instruction
UT	Unit Table
UTE	Unit Table Entry
VMFA	Virtual Memory File Access
VPSD	Virtual Prime Symbolic Debugger
WLSN	Wait List Segment Number
WLWN	Wait List Word Number
XB	Extra Base