VM/1

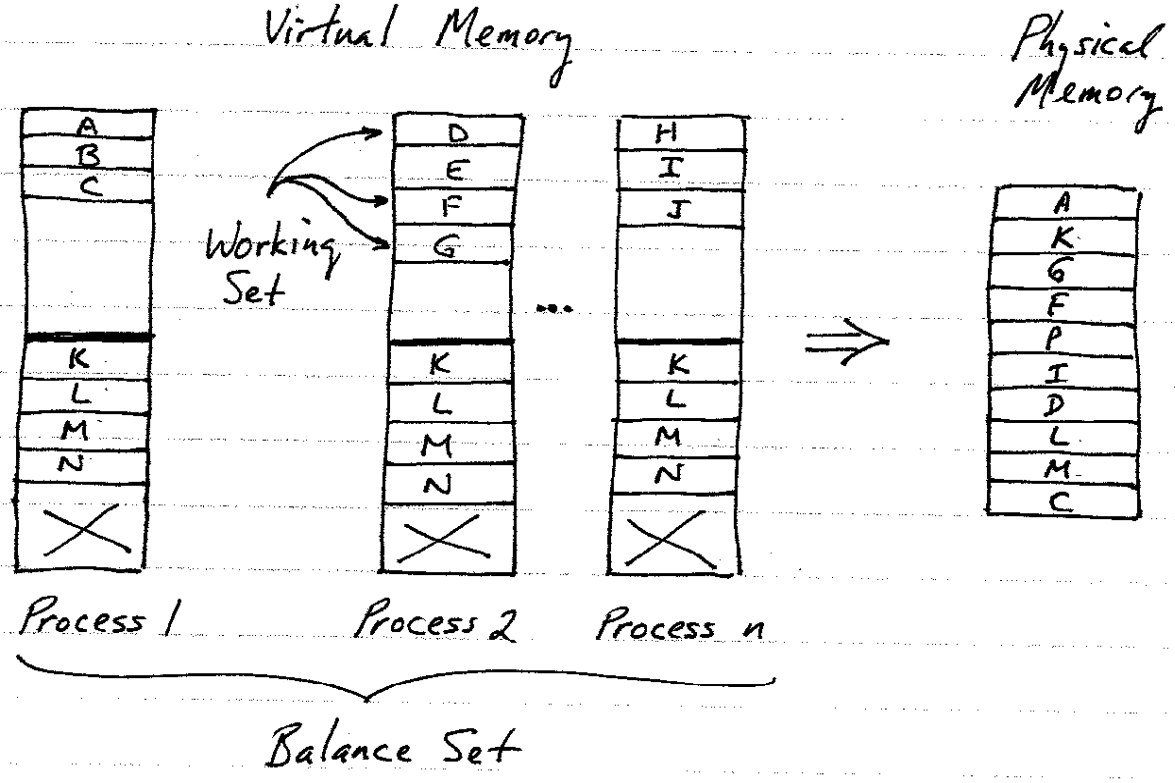
# *Virtual Memory*

# Virtual Memory

* ISA has large VA space.
  - Allows user to uniquely identify lots
* Physical Memory is smaller
  - Cost issue

* Virtual Memory Management
  - Access Control
  - Translation

* The VAX Model

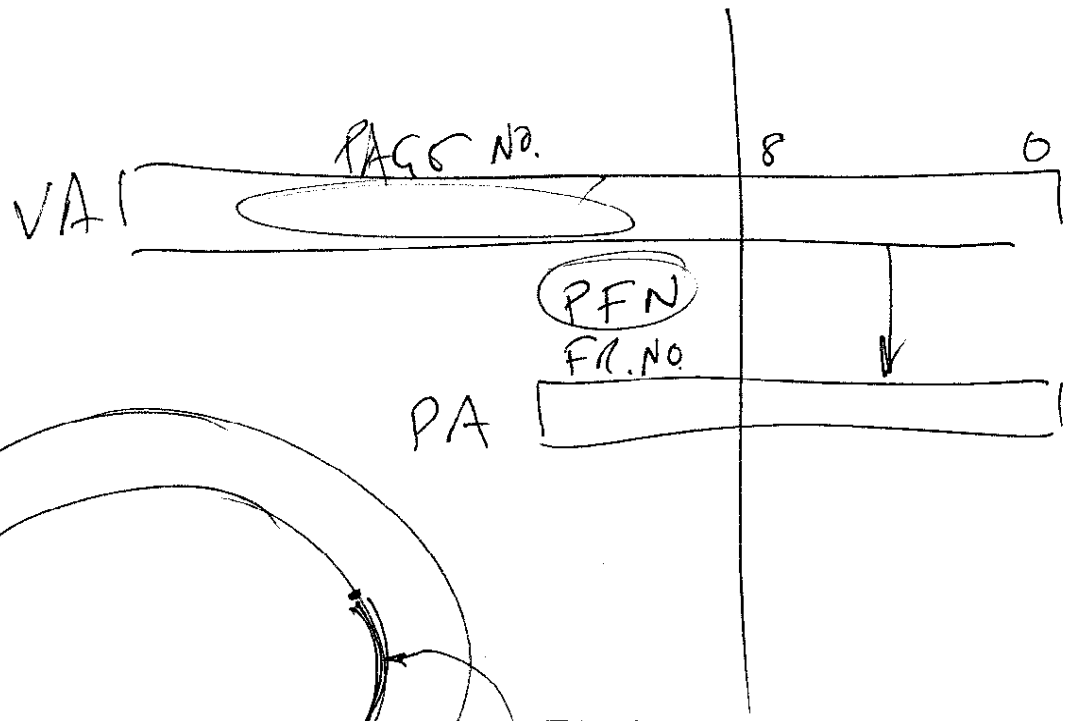


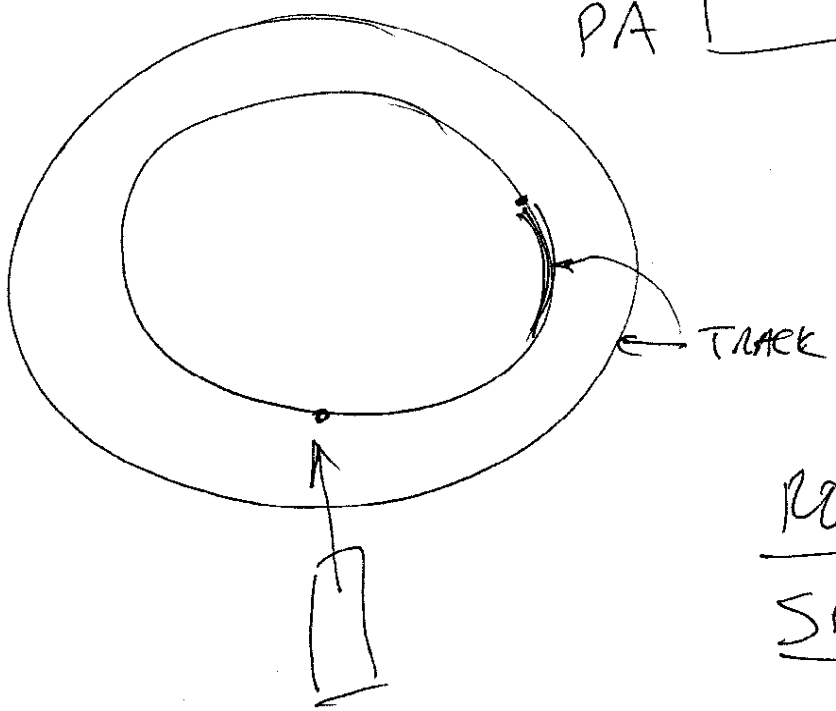
# FRAME

# PAGES

VAX: $2^9$ BYTES

x86: $2^{12}$ BYTES

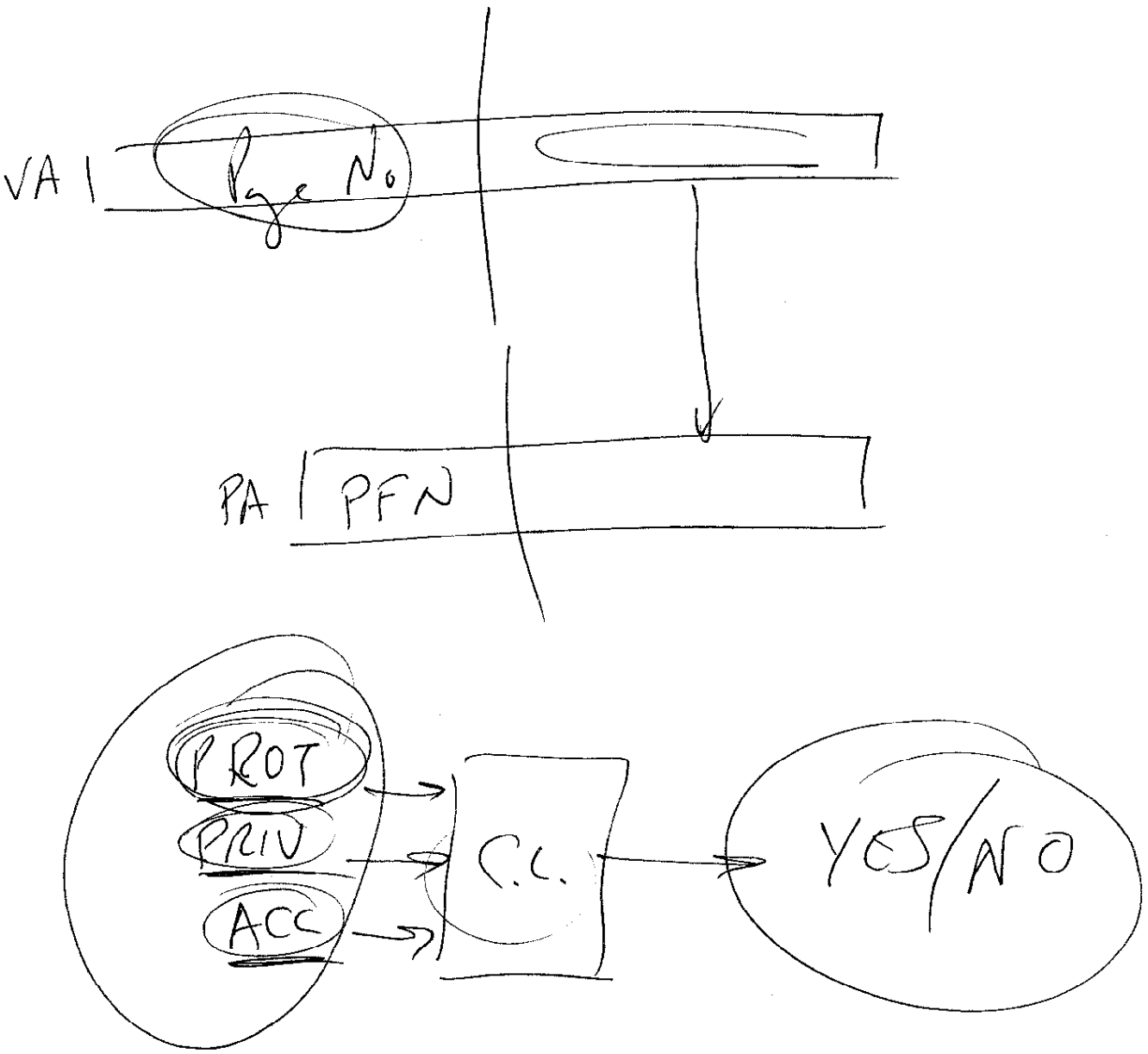



ROTATION

SEEK

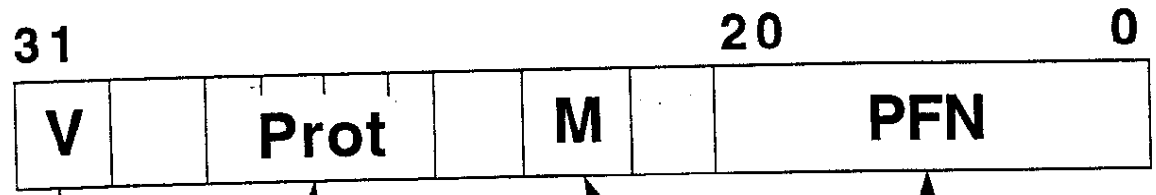
RESIDENT




VMS
WNT

HAL
IBM

# The PTE




Can I Believe The PFN

Has the Page Been Written

In which frame is the Page Stored

Do I have the Right to do this access

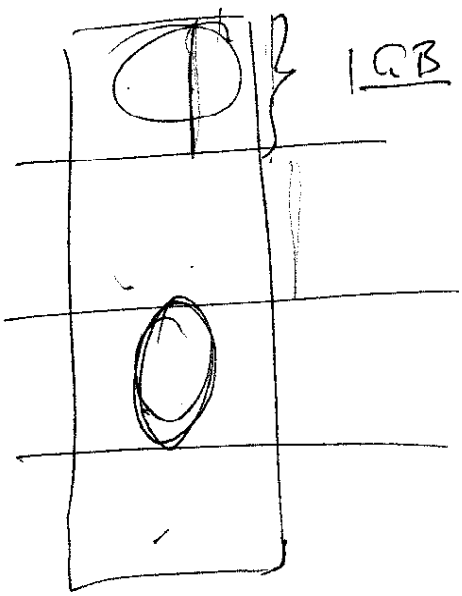
$\{NO,R,R/W\}^{\{K,E,S,U\}}$

81 Possibilities in 4 bits ?

**Note:**

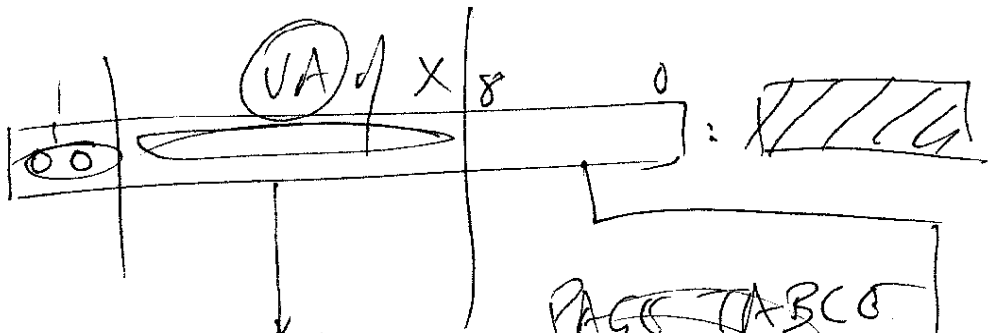
No Ref. Bit!

Rgn

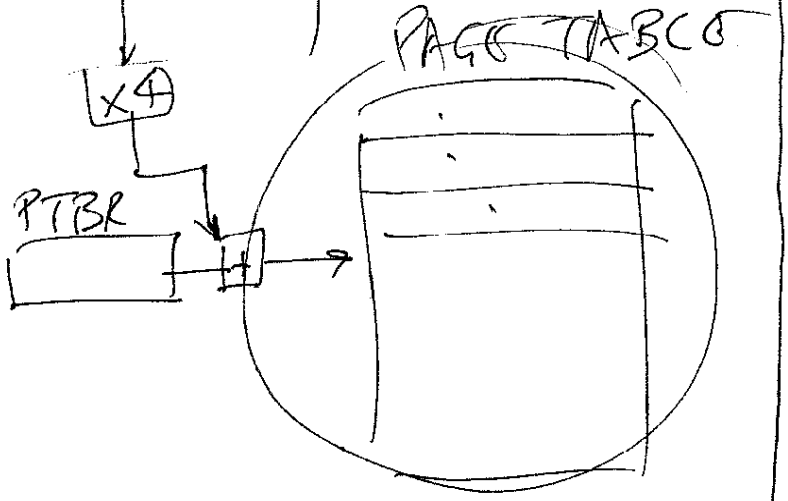


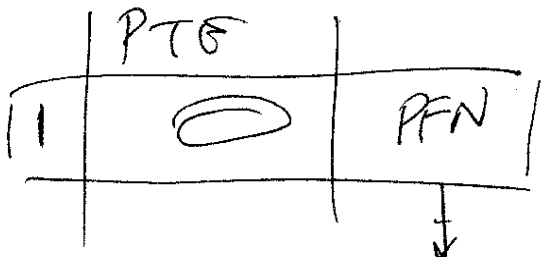

$\frac{2^{30}}{29} = 2^{21} \times 4 = 2^{23}$

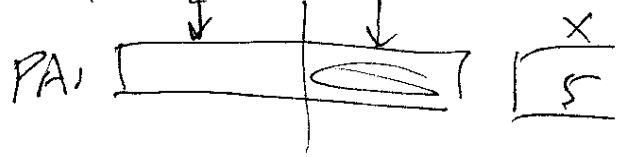
8148







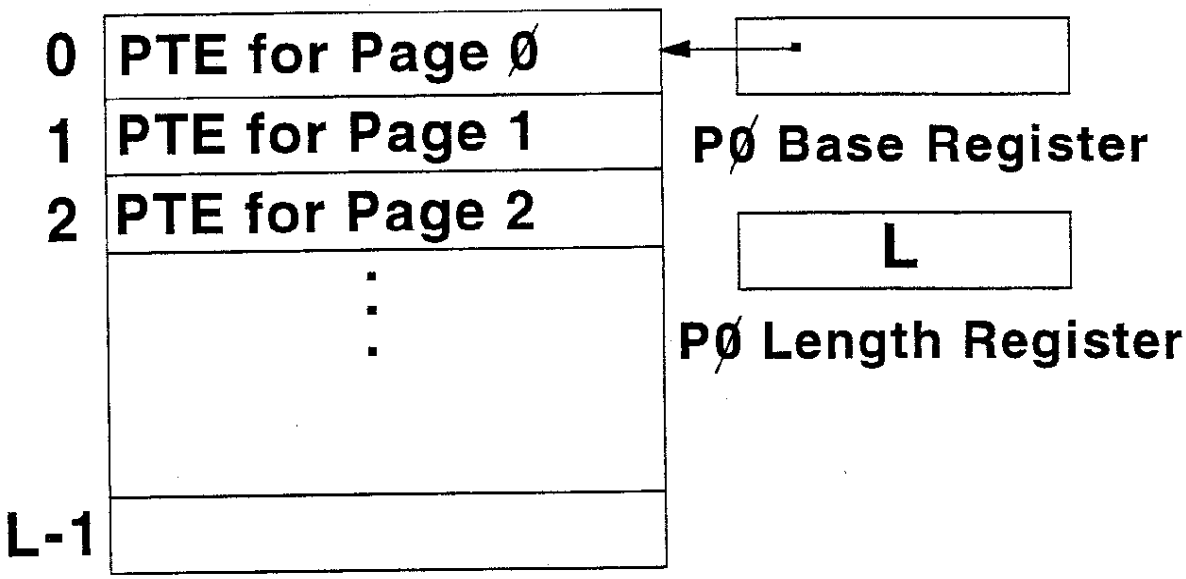



# *Page Tables*

* One for each region

* For example, the ~~P0~~ PROGRAM Page Table

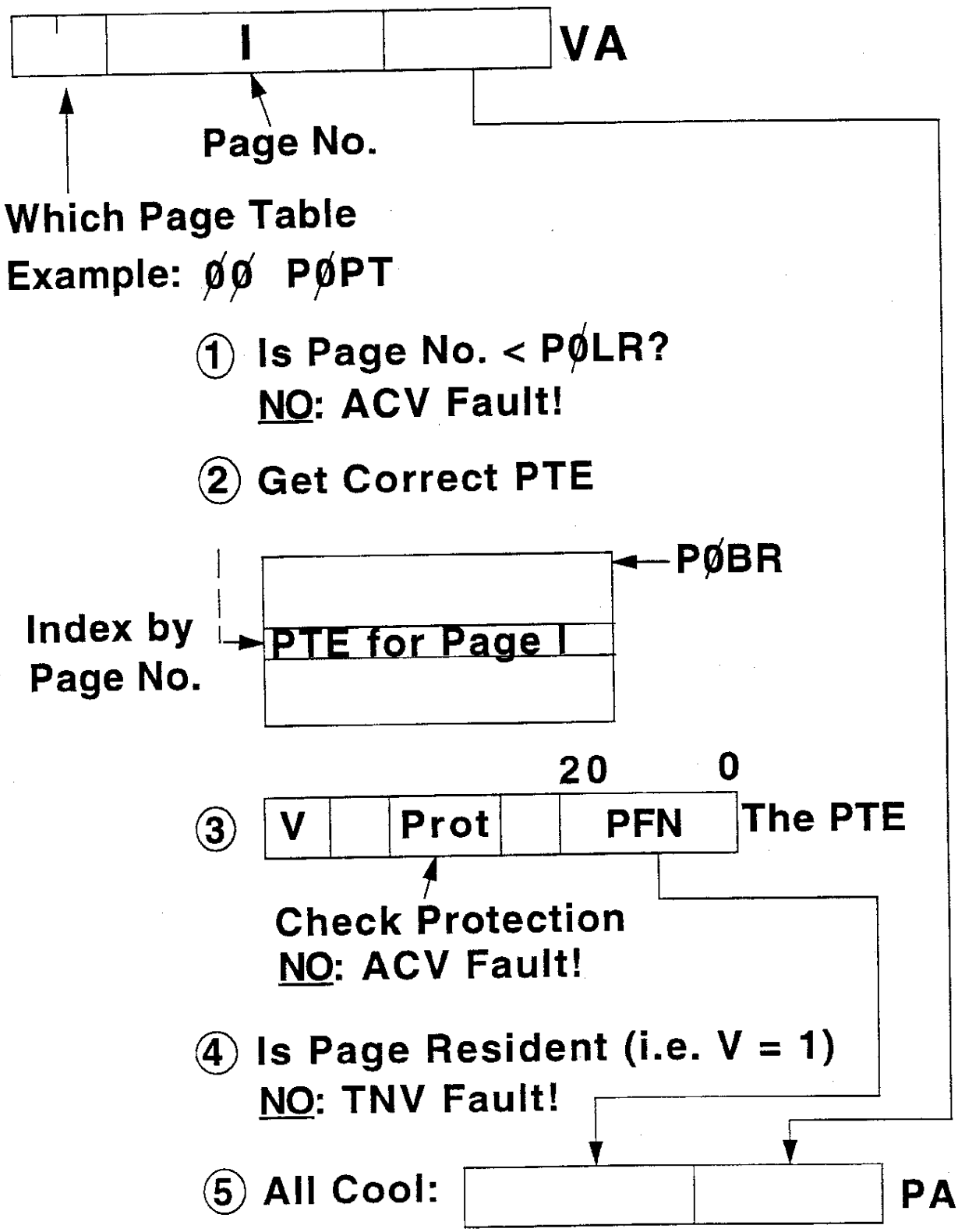


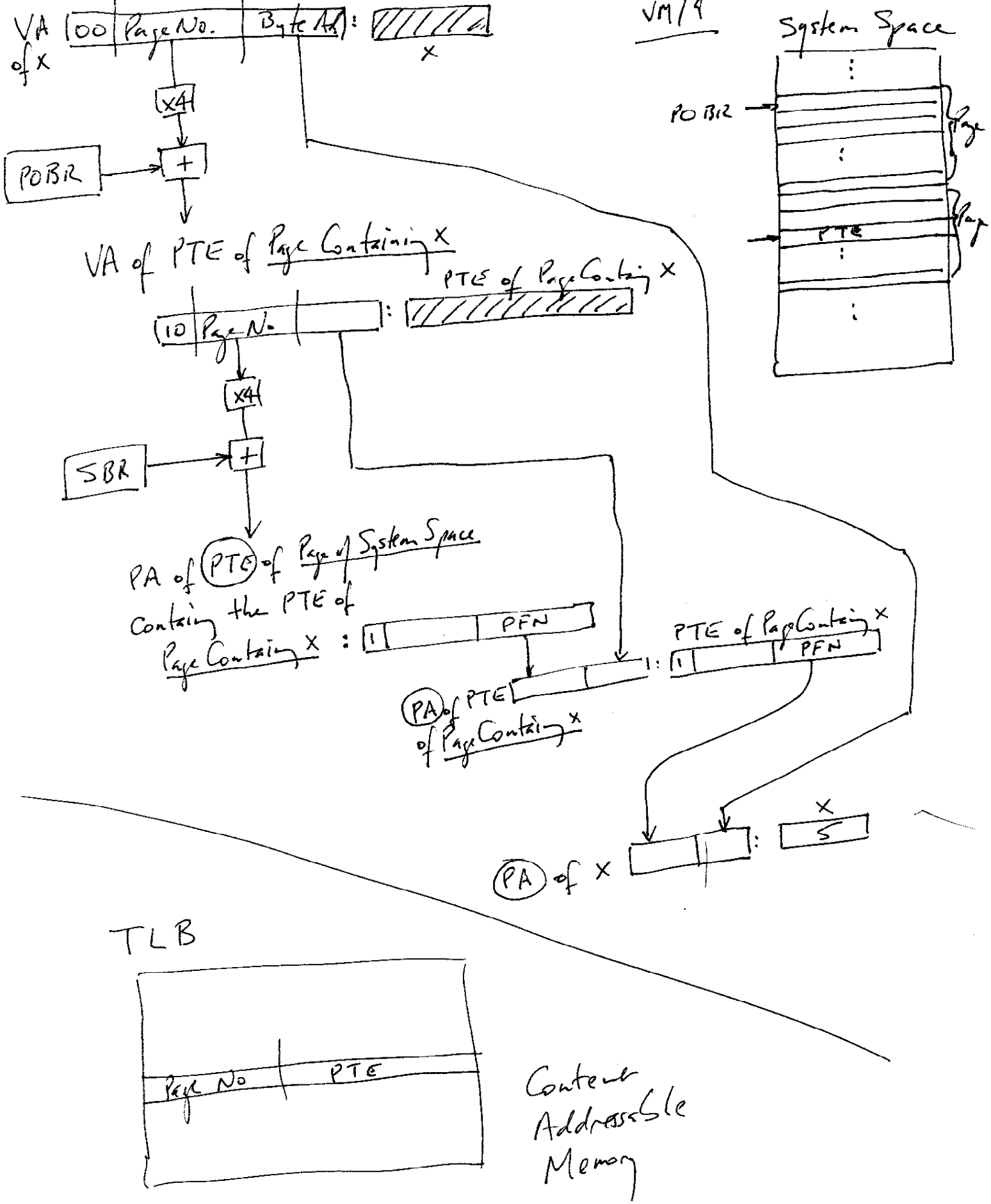

* Sequentially stored in System Virtual Space

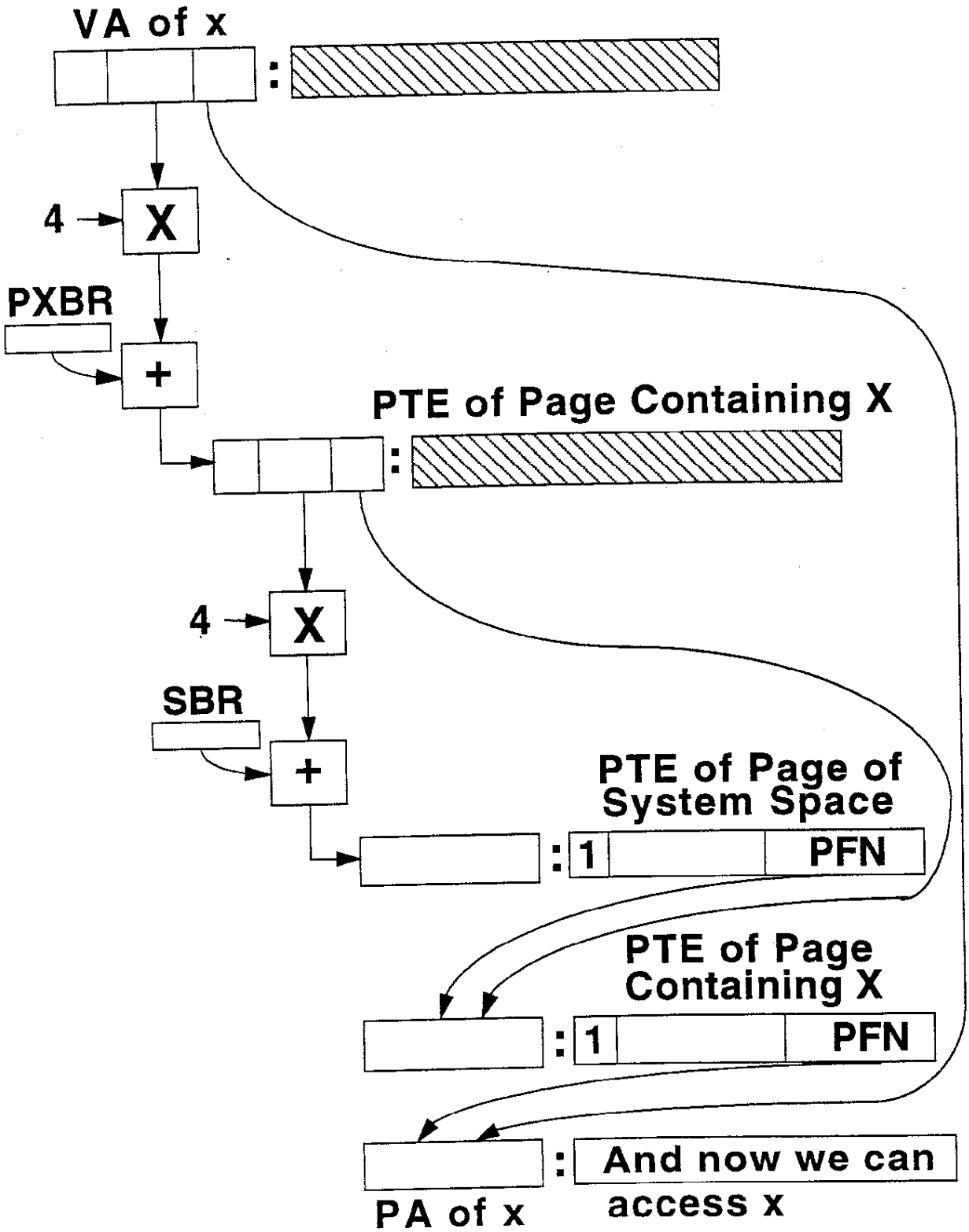
* P0LR used for ACV checks

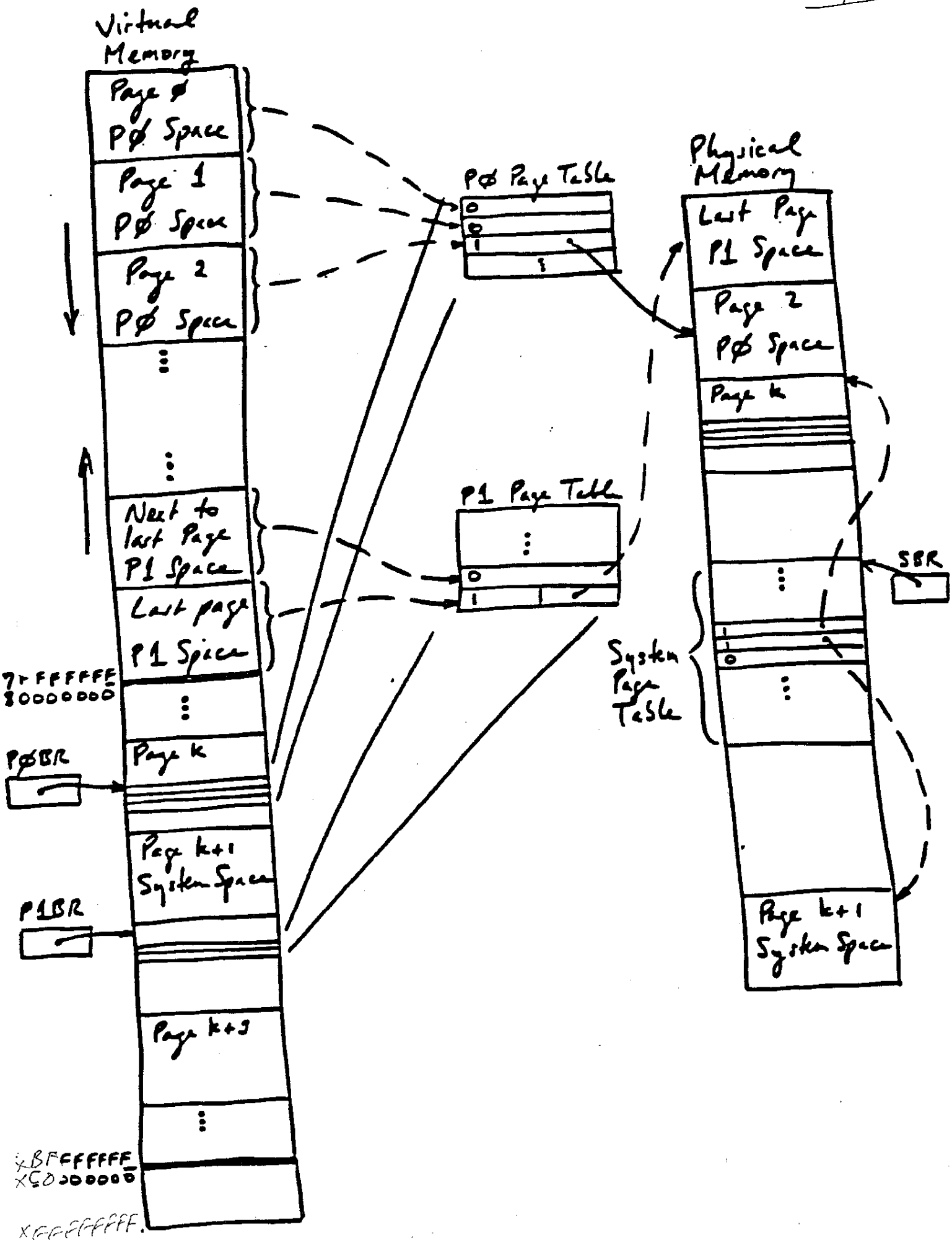
* PTE used for ACV, TNV checks

# The Abstraction




VM/9
System Space
VA of x
00
Page No.
Byte Adr
x
x4
PCBR
+
PTE
VA of PTE of Page Containing x
PTE of Page Containing x
10
Page No.
x4
SBR
+
PA of PTE of Page of System Space Containing the PTE of Page Containing x
1
PFN
PTE of Page Containing x
1
PFN
PA of PTE of Page Containing x
PA of x
x
5
TLB
Page No
PTE
Content Addressable Memory


VA of x
4
X
PXBR
+
PTE of Page Containing X
4
X
SBR
+
PTE of Page of System Space
1
PFN
PTE of Page Containing X
1
PFN
And now we can access x
PA of x


Virtual Memory
Page 0 P0 Space
Page 1 P0 Space
Page 2 P0 Space
Next to last Page P1 Space
Last page P1 Space
7FFFFFFF
80000000
Page k
Page k+1 System Space
Page k+3
BFFFFFFF
C0200000
FFFFFFFF
P0BR
P1BR
P0 Page Table
P1 Page Table
Physical Memory
Last Page P1 Space
Page 2 P0 Space
Page k
Page k+1 System Space
SBR
System Page Table

# ONE FINAL EXAMPLE

LETS MODIFY THE VAX ISA TO MAKE IT EASIER TO SEE WHAT IS GOING ON. WE WILL RETAIN THE ESSENTIAL ELEMENTS, BUT WE WILL REDUCE ALL THE NUMBERS.

FOR EXAMPLE: VA WILL GO FROM 32 bits to 9 bits
PAGE SIZE WILL GO FROM 512 B to 16 B
PA WILL BE 7 bits
PTE WILL STILL BE 4 BYTES.

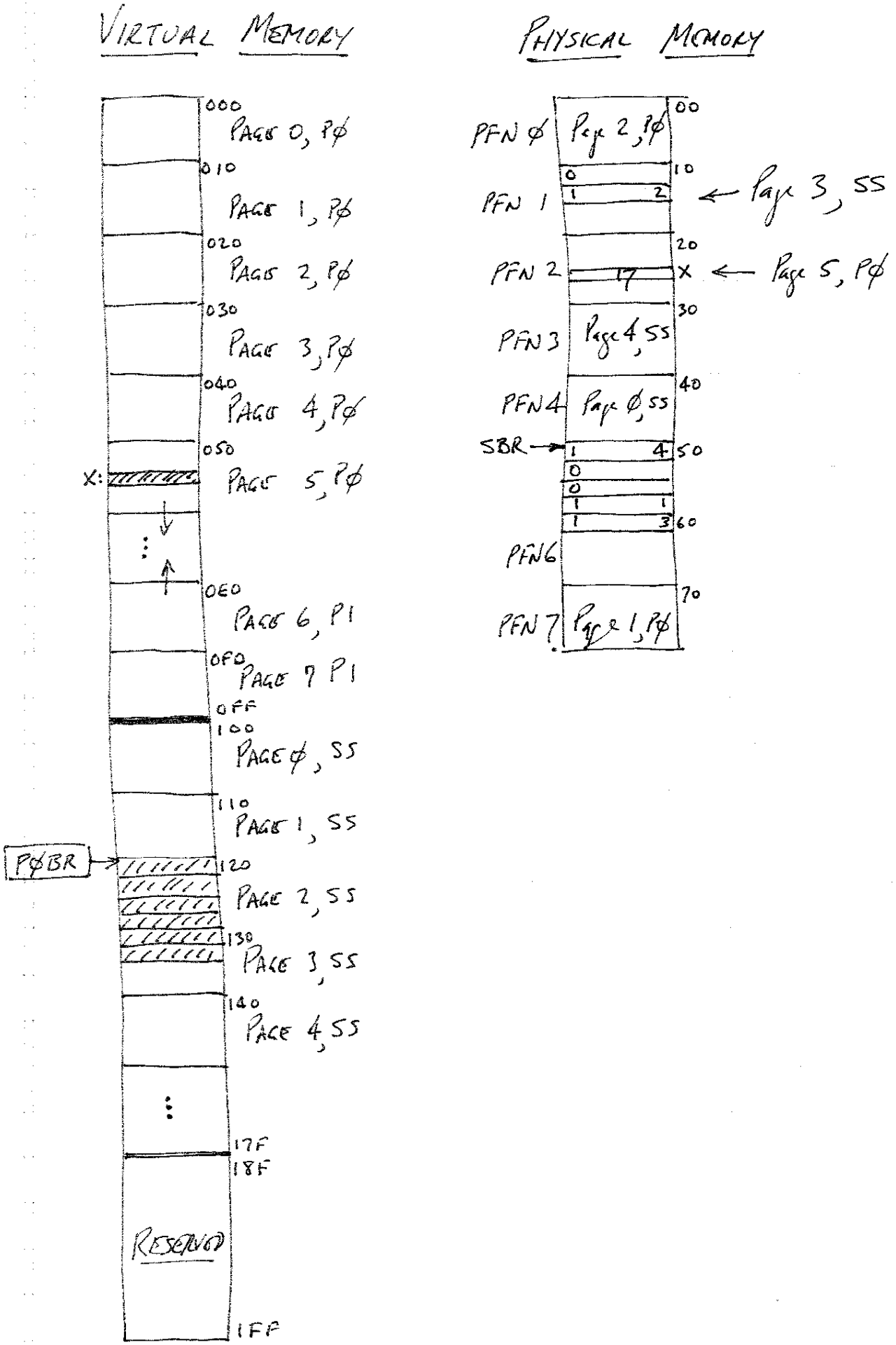
PAGE 2 SHOWS A SNAPSHOT OF VIRTUAL & PHYSICAL MEMORY SEVERAL THINGS ARE WORTH NOTING:

① VIRTUAL MEMORY = 512 BYTES. ∴ 32 PAGES POSSIBLE
P0 HAS A MAX OF 8. IN OUR EXAMPLE: 6 PAGES WERE NEEDED
P1 " " " " ". IN OUR EXAMPLE: 2 PAGES " "
S0 " " " " ". IN OUR EXAMPLE: 5 PAGES " "

② P0 PAGE TABLE STARTS (P0BR) AT VA = 120 (NOTE: 2, 0 ARE HEX DIGITS)
SINCE THERE MUST BE 6 ENTRIES, THE P0 PAGE TABLE CONSUMES ALL OF PAGE 2, HALF OF PAGE 3 OF S0.

③ SYSTEM PAGE TABLE STARTS (SBR) AT PA = 50 (NOTE: 5 IS OCTAL, 0 IS HEX)
SINCE THERE MUST 5 ENTRIES, SYSTEM PAGE TABLE CONSUMES ALL OF FRAME 5 AND 1/4 OF FRAME 6.

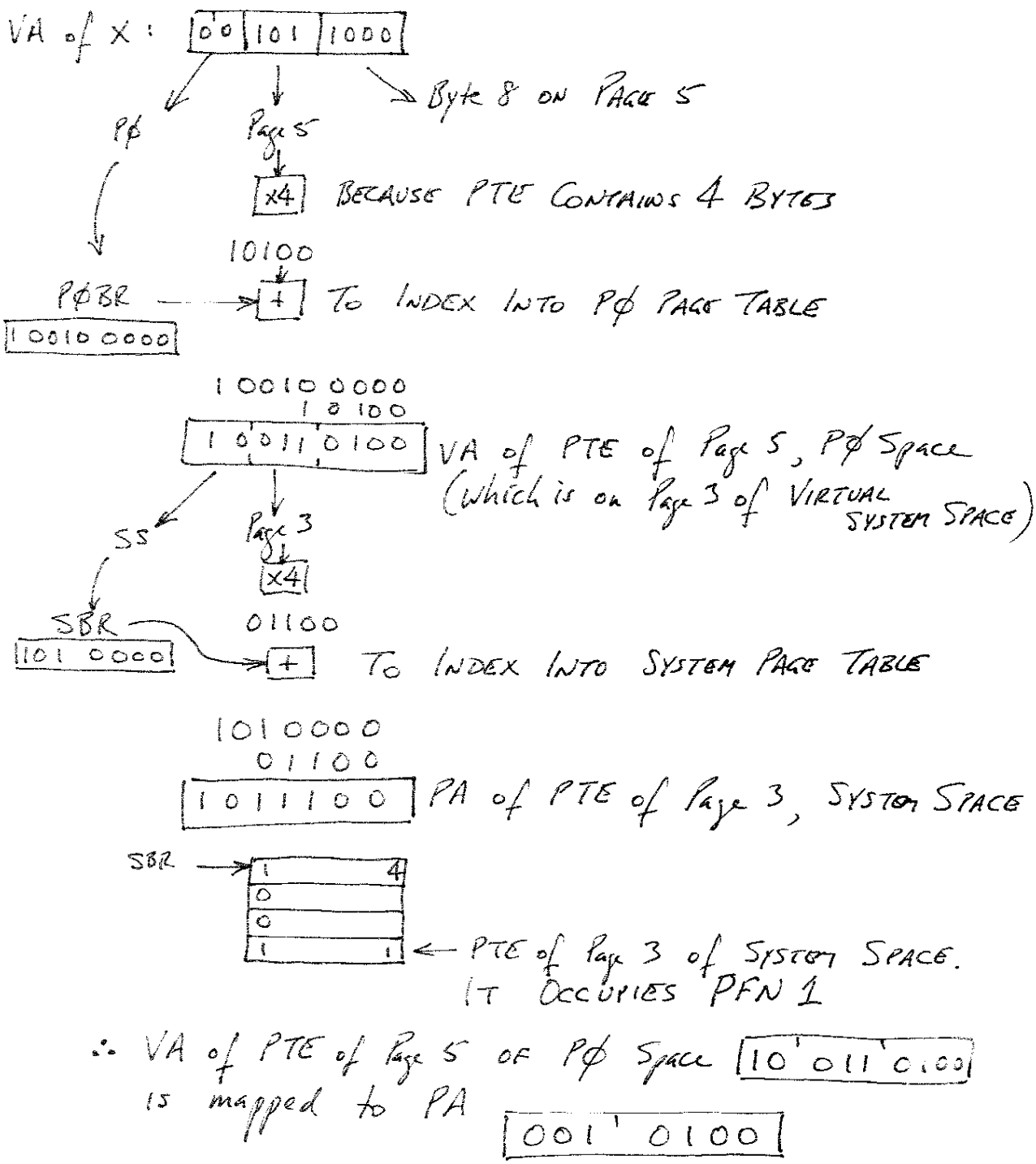
④ NOTE THAT SYS. PAGE TABLE INDICATES 3 PAGES RESIDENT (PAGES 0, 3, 4)
NOTE THAT PART OF P0 PAGE TABLE INDICATES PAGE 5 IN, PAGE 4 OUT


VIRTUAL MEMORY
PHYSICAL MEMORY
000
PAGE 0, PØ
010
PAGE 1, PØ
020
PAGE 2, PØ
030
PAGE 3, PØ
040
PAGE 4, PØ
050
X:
PAGE 5, PØ
0E0
PAGE 6, P1
0F0
PAGE 7 P1
0FF
100
PAGE Ø, S5
110
PAGE 1, S5
PØBR
120
PAGE 2, S5
130
PAGE 3, S5
140
PAGE 4, S5
17F
18F
RESERVED
1FF
PFN Ø
Page 2, PØ
00
10
PFN 1
0
1
2
← Page 3, S5
20
PFN 2
17
X
← Page 5, PØ
30
PFN 3
Page 4, S5
40
PFN 4
Page Ø, S5
SBR
1
4
50
0
0
1
1
1
3
60
PFN 6
70
PFN 7
Page 1, PØ

# VAX VM (CONT.)

FINALLY, A PAGE TABLE COMPUTATION:

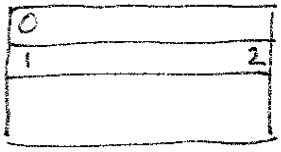
- WE WISH TO EXECUTE: LD R1, X
  WHERE X HAS VA: 00101 1000




∴ VA of PTE of Page 5 of P0 Space 10'011'0100
is mapped to PA
001'0100

# VAX VM EXAMPLE (CONT.)

SO, IF WE LOOK AT PFN 1



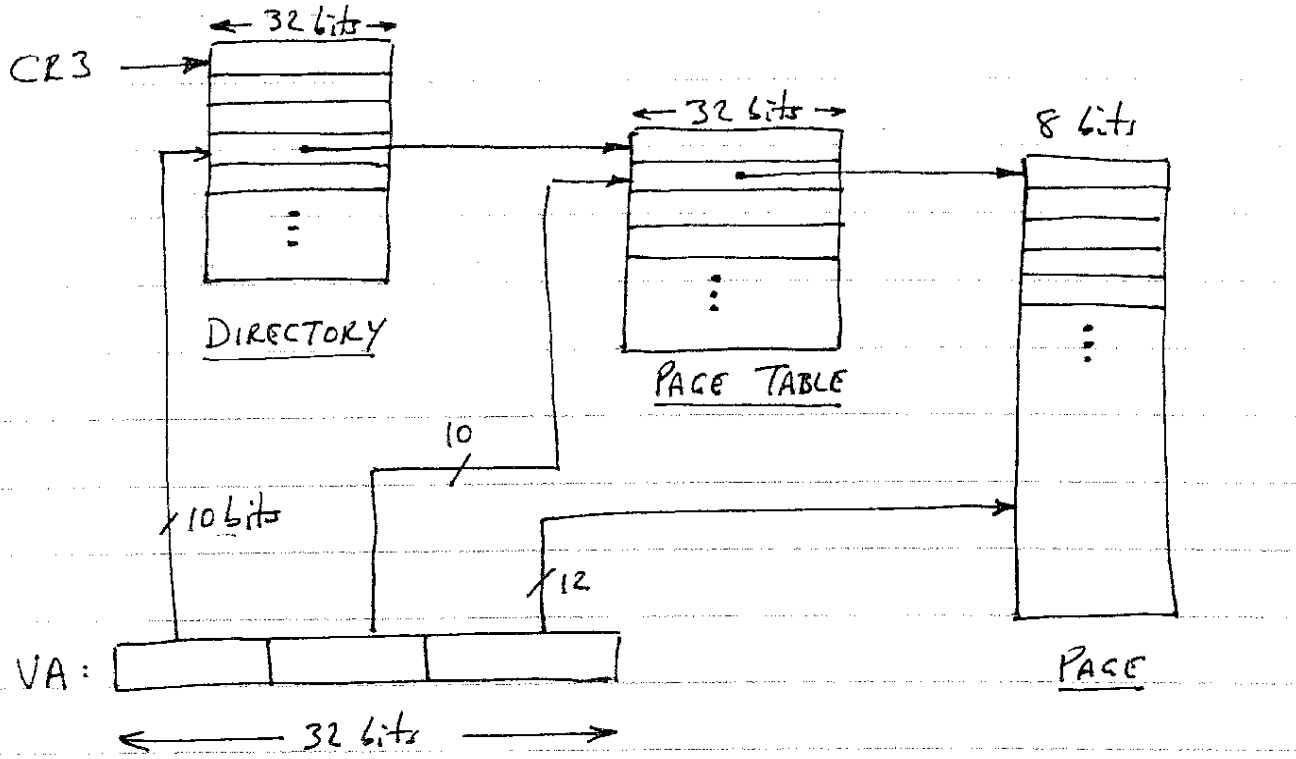

WE CAN READ THE PTE of Page 5, P0 SPACE.
WE SEE PAGE 5, P0 SPACE OCCUPIES PFN 2

THE VA of X [00 101 1000] THEREFORE MAPS TO

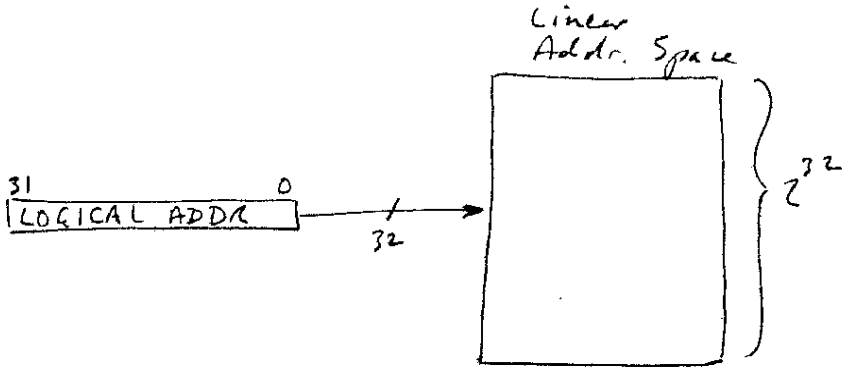
PA [010 1000]

WE LOOK IN THAT LOCATION AND FIND 17, WHICH WE LOAD INTO R1.

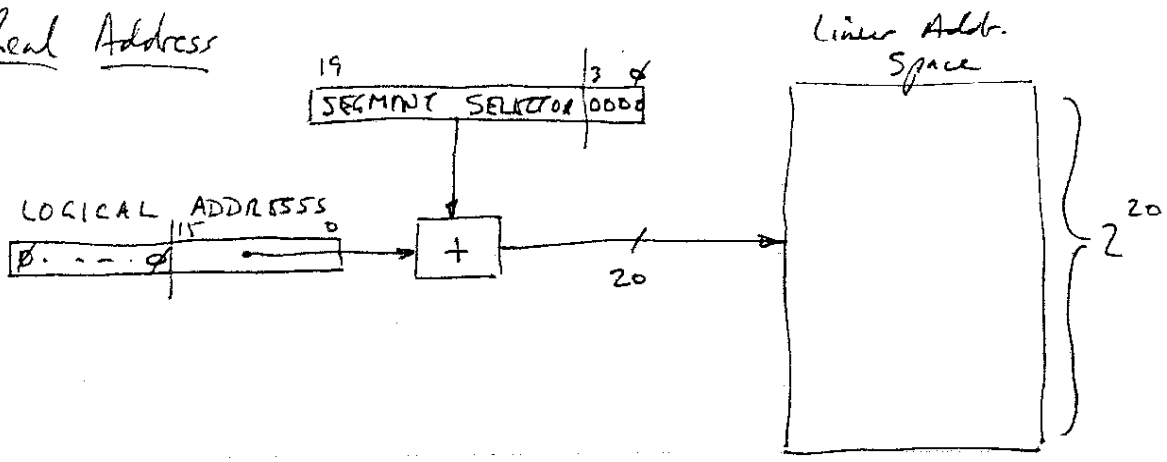
# IA-32


32 bits
CR3
DIRECTORY
32 bits
PAGE TABLE
8 bits
PAGE
10
10 bits
12
VA:
32 bits

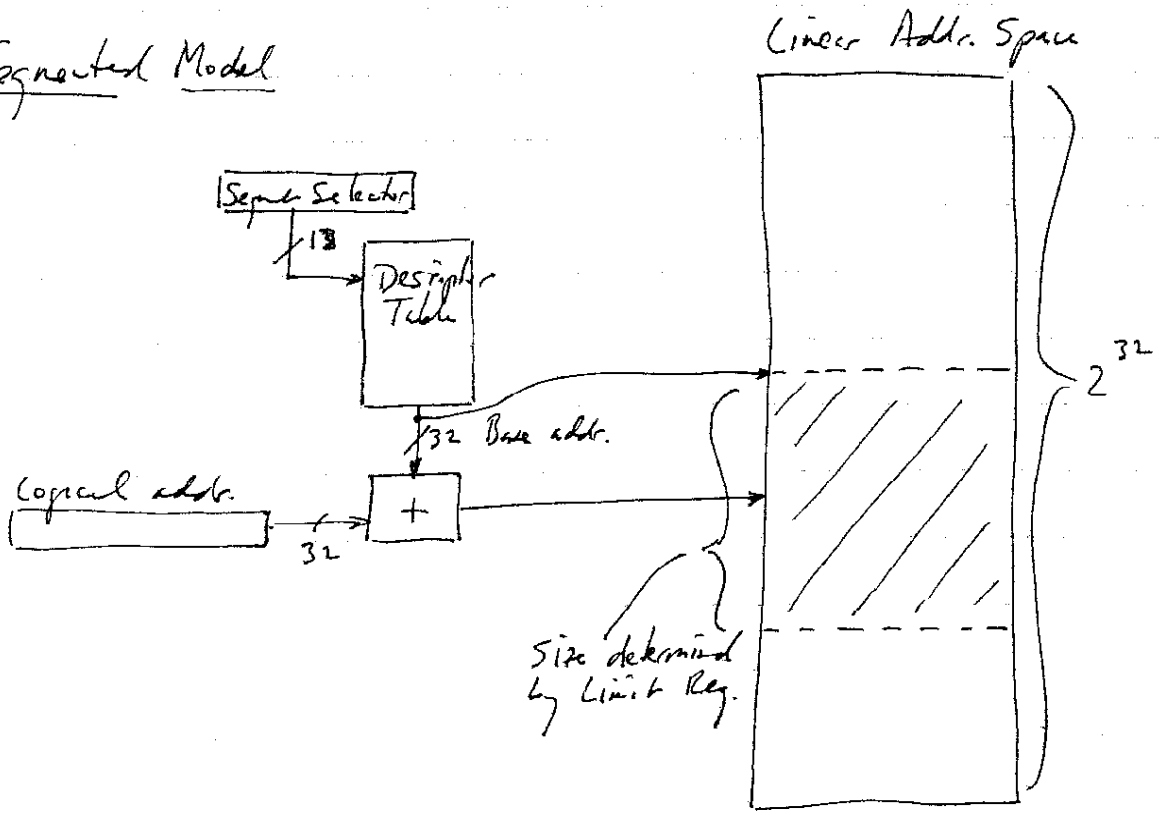
## Flat Model



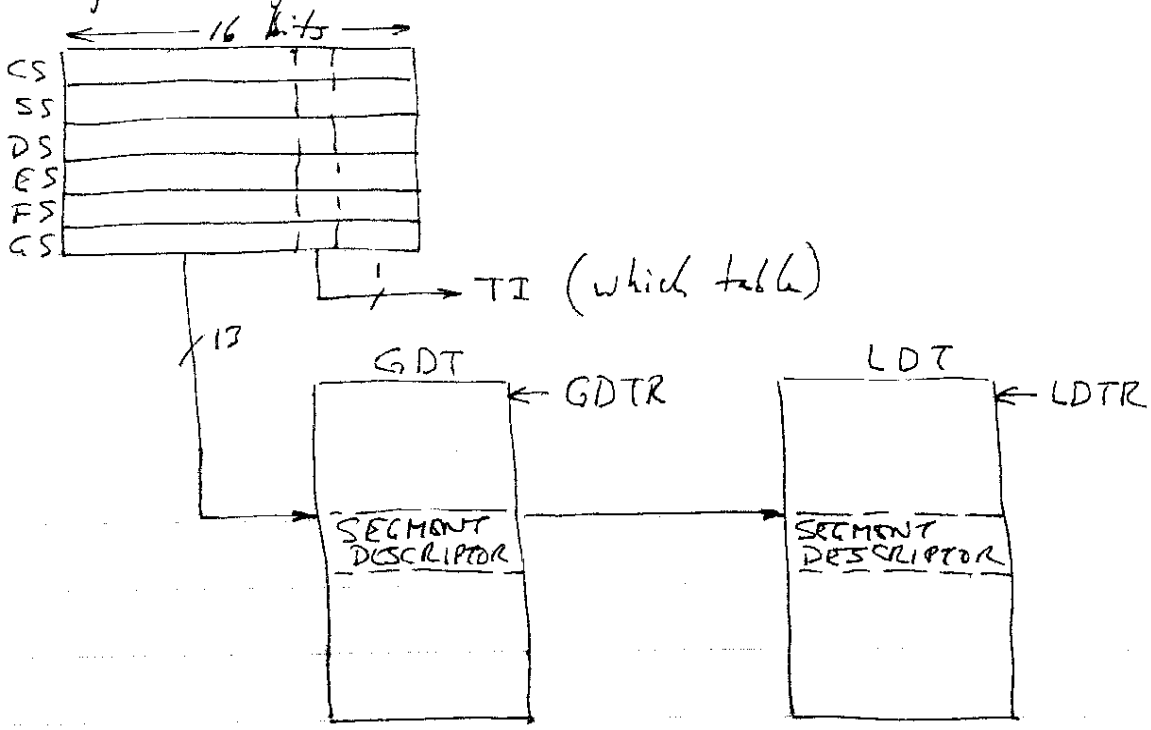

## Real Address




## Segmented Model

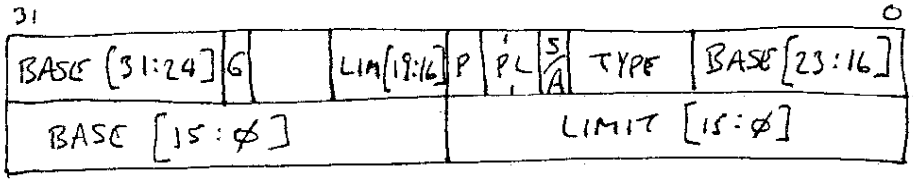


## Segment Registers



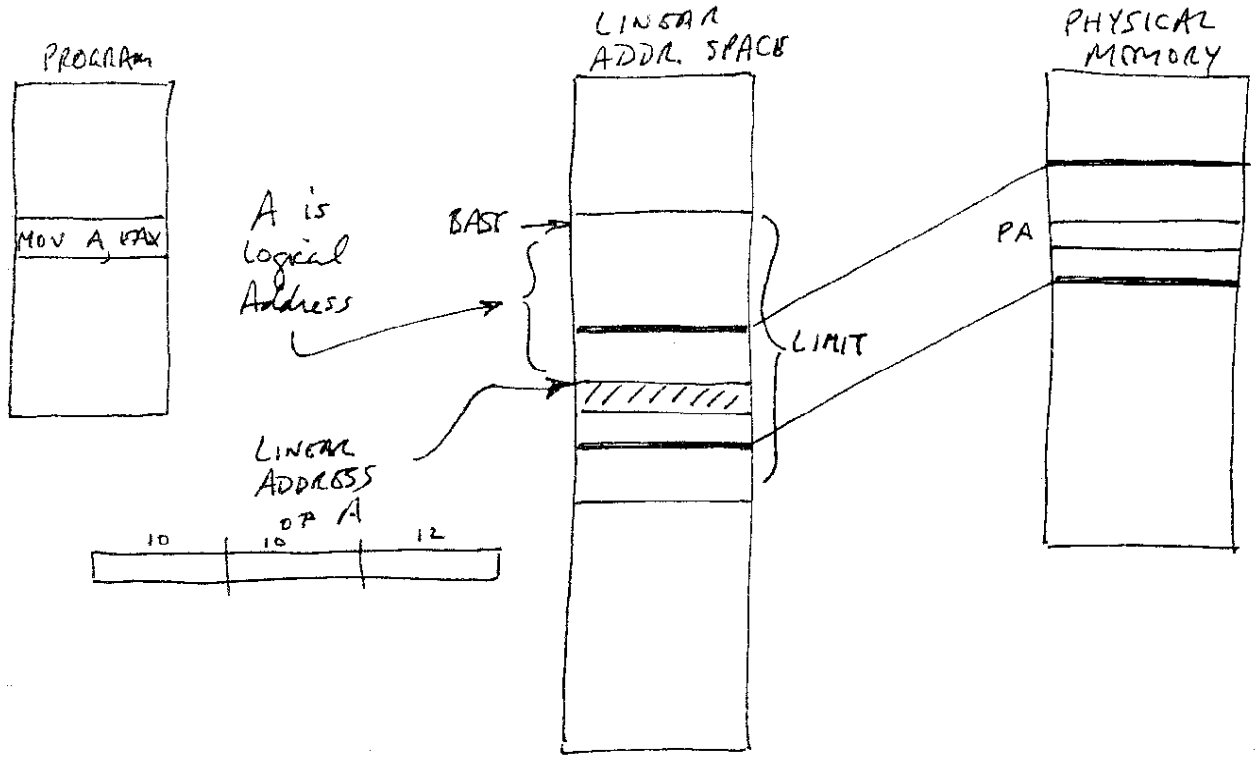

$2^{13}$ x 8 bytes each

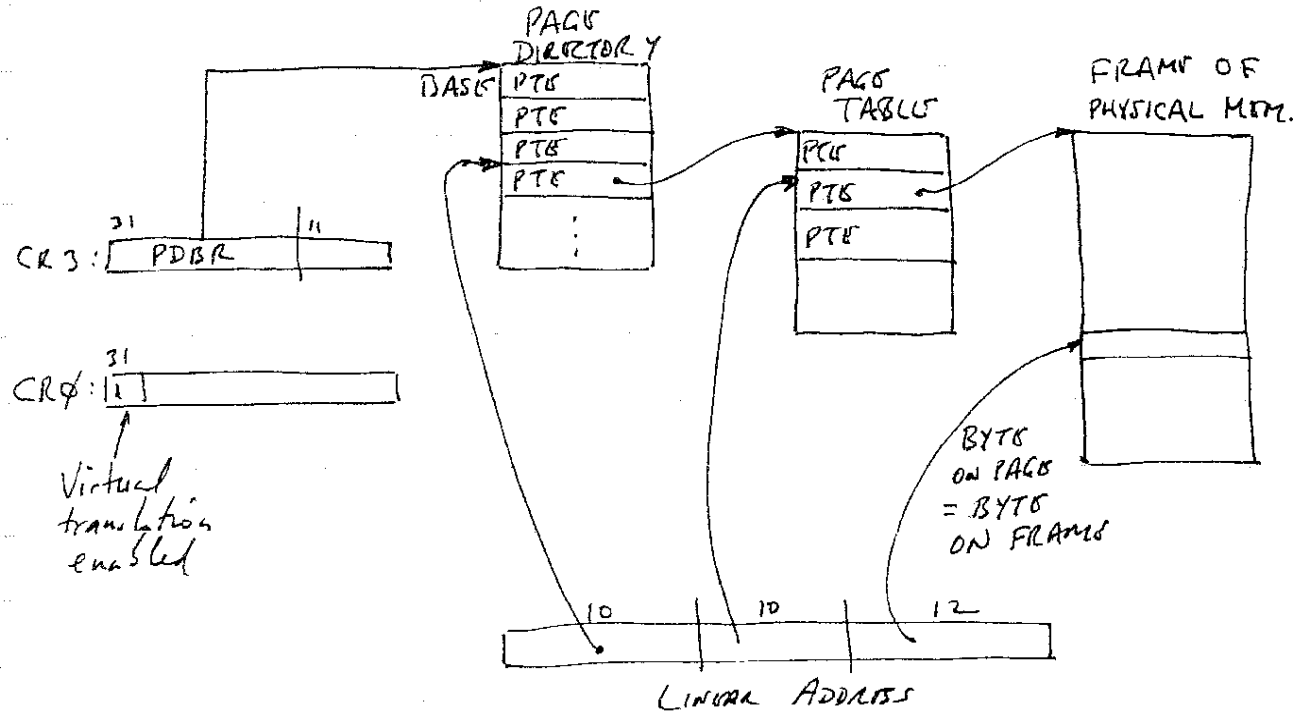
## Segment Descriptor

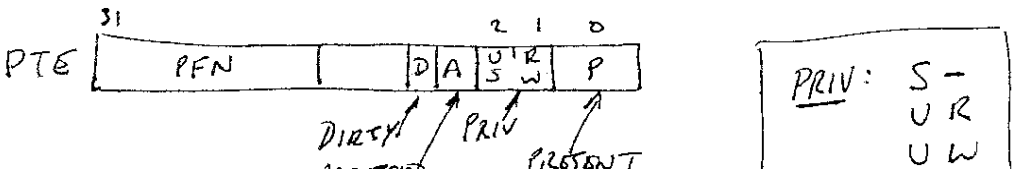



G: Granularity 1 BYTE / 4KBYTES
P: Segment Present
PL: Privilege Level
TYPE: Segment type

# Segmentation AND Paging



# Task State Segment (TSS)