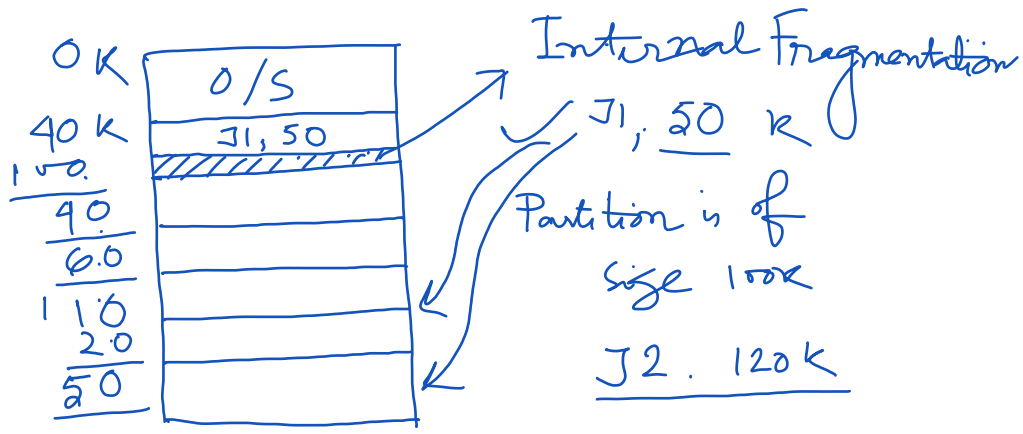
Tuesday, December 14, 2021 10:55 AM

Memory Management

MFT – Multiprogramming with Fixed no. of tasks



External fragmentation

1) First fit ✓
2) Best fit ✓
3) Worst fit ✓ X

MFT suffers from Internal as well as external fragmentation

MVT

Multiprogramming with Variable no. of Tasks

Memory holes will be created

Suffers from external fragmentation

Compaction

logical address

Physical " RAM Address

2 GB

Three problems of memory

— Not enough RAM

— Holes in our address space

— Program writing over each other

Prog 1

0x 1234

value = 3

Prog 2

0x 1234

Value = 7

* 0x 1234 = _____ ?

Virtual Memory

— level of indirection

— How does it solve problem?

— Translation / Page Table

Where does these page table reside?

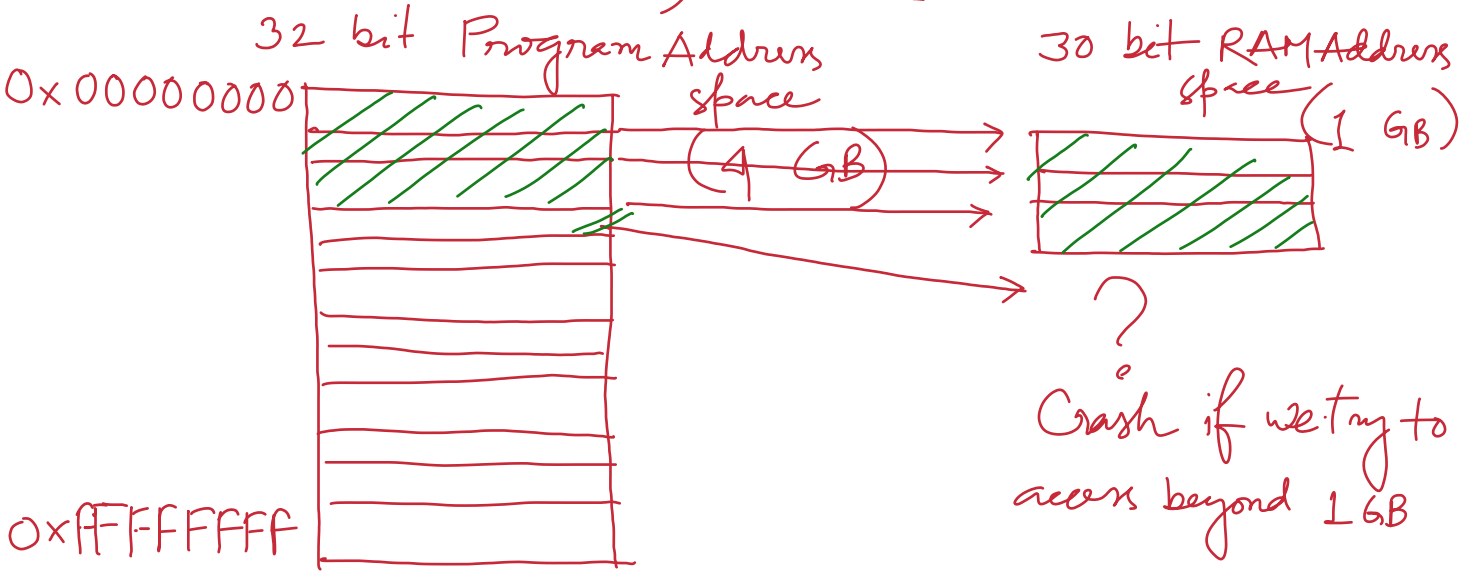
32 bit address

How much memory we can access

i) 2^{30} B

ii) 2^{32} B ✓

iii) 2^{32} words X



Key to problem

Same memory space

Every program has same memory space and actual RAM has same memory space

RAM ⟷ disk

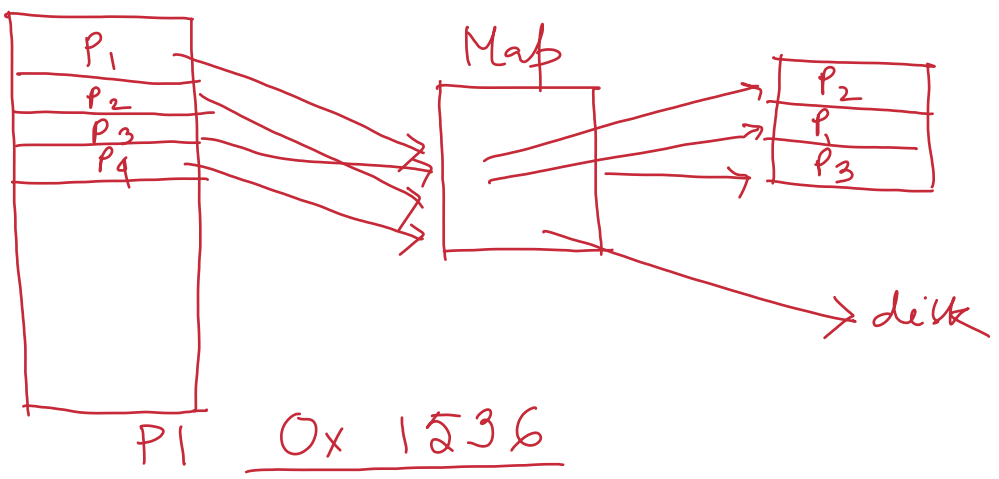
Virtual Memory

RAM — Memory

Disk — Storage

Without Virtual Memory

Program Address = RAM address



0x 1536

i) RAM address 0
ii) RAM " 1536
iii) Need more information

If a program uses more data that can fit into RAM, where does the data go?

Disk

32 bit 2^{32} B = 4 GB

MAP

VA	PA
.	
:	

MIPS — Word

1 Word = 4 B

Word = $\frac{2^{32}}{4} = 2^{30} \simeq$ 1 billion

Page Table

Fine grain : Maps each word address to the map

0 – 4095 words

4096 – 8191 words

4096 Words

Coarse grain Fewer mappings

1024 Words 1 Word = 4B

1024 × 4 = 4096 B

1 Page = 4096 B
= 4 KB

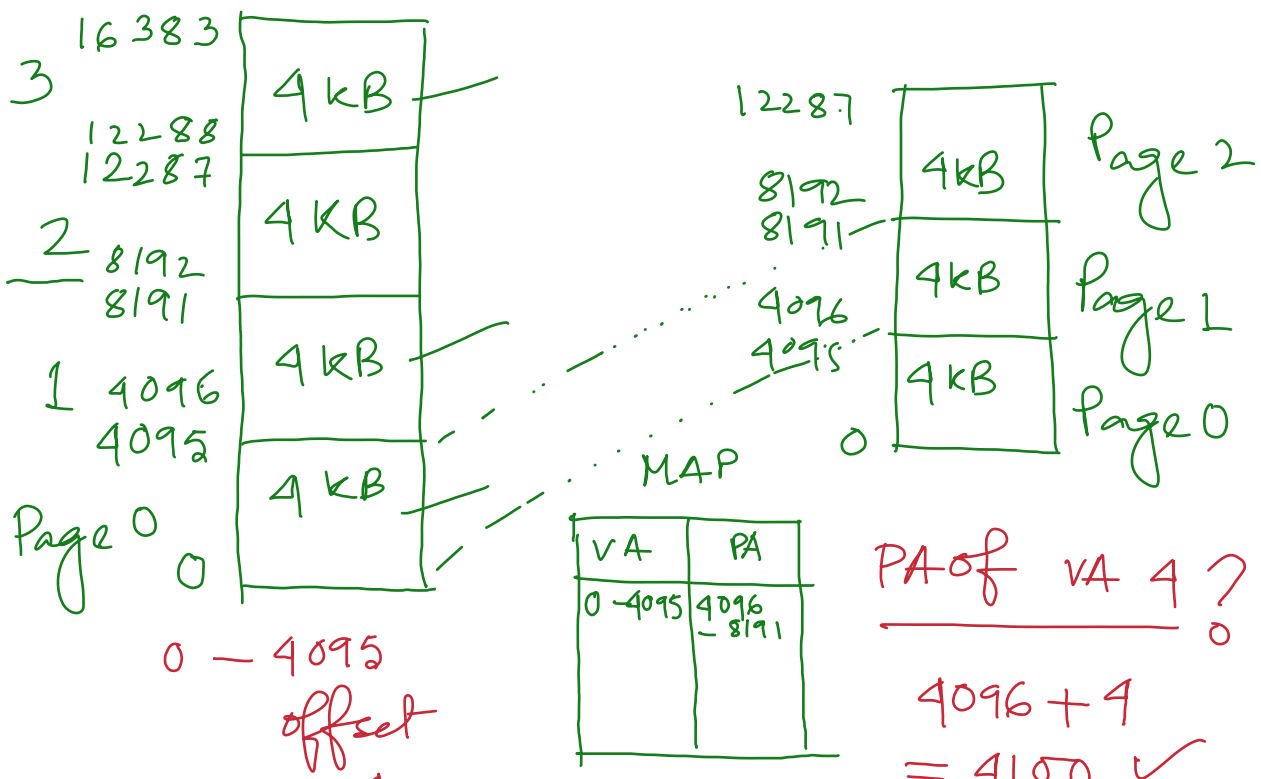
2^{30} entries $\simeq$ 1 word

$\frac{2^{30}}{1024} \simeq 2^{20}$ $\simeq$ 1024 words

1 million

VA

PA



0 — 4095

offset

Byte 4

PA of VA 4 ?

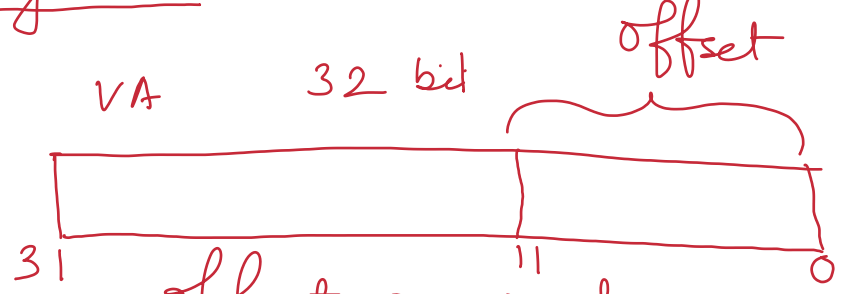
4096 + 4

= 4100 ✓

4kB

= 4096 B

$= 2^{12}$ B



offset never changes

Can we have a page size of 12 kB ?

Page Size is power of 2

Page Size → 256 kB

32 bit VA

31 bit ? PA

33 bit

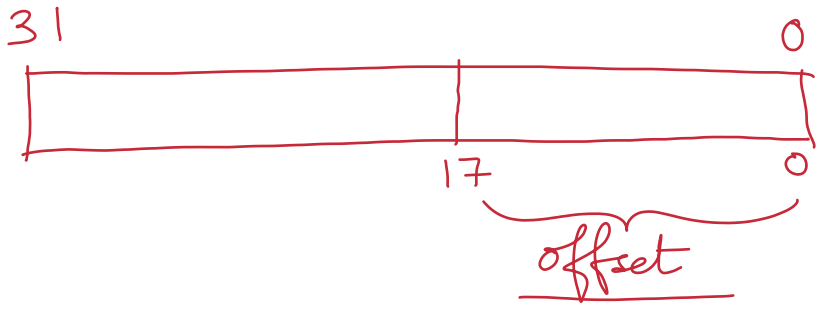
RAM 2 GB $= 2^{31}$ B

RAM 8 GB $= 2^{33}$ B

256 kB $= 2^8 \times 2^{10}$ B

$= 2^{18}$ B

18 bit offset



Address Translation

2^{30} B = 1 GB per process

Program Address Space

4 GB VA space

1024 words

4096 B

1 Page Size = 4096 B = 4 KB

Table size 2^{20} B ⇒ 1 MB

32 bit VA

256 MB RAM

28 bit PA

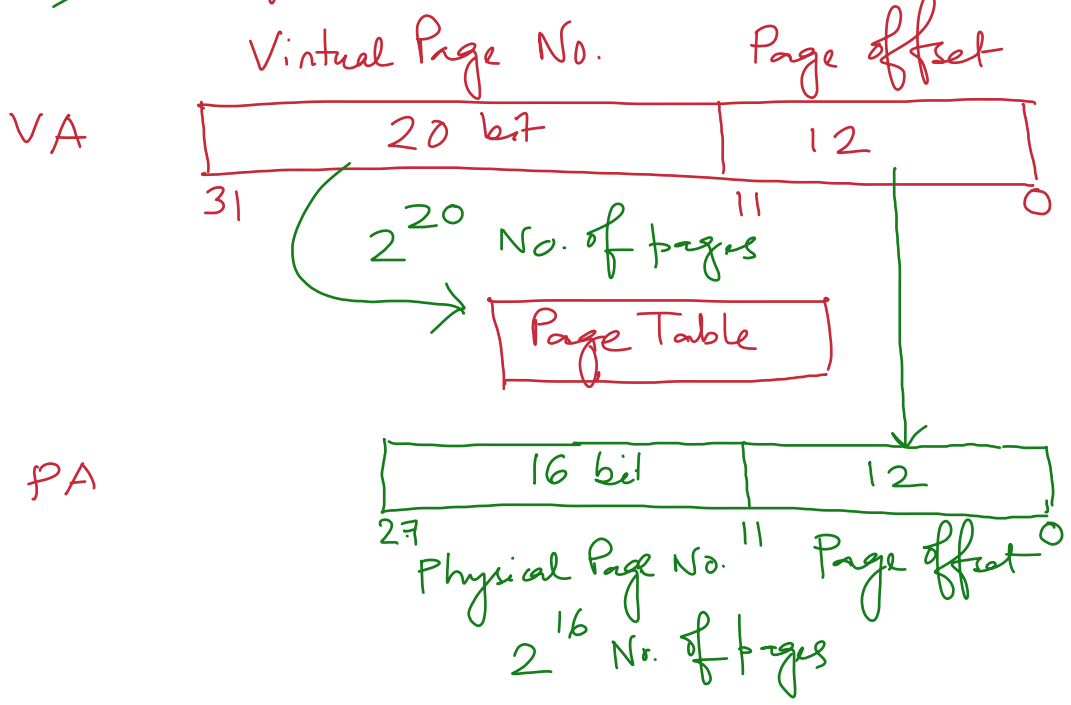
256 MB

$= 2^{8} \times 2^{20}$ B

$= 2^{28}$ B

4096 B → 4 KB pages

offset 12 bit

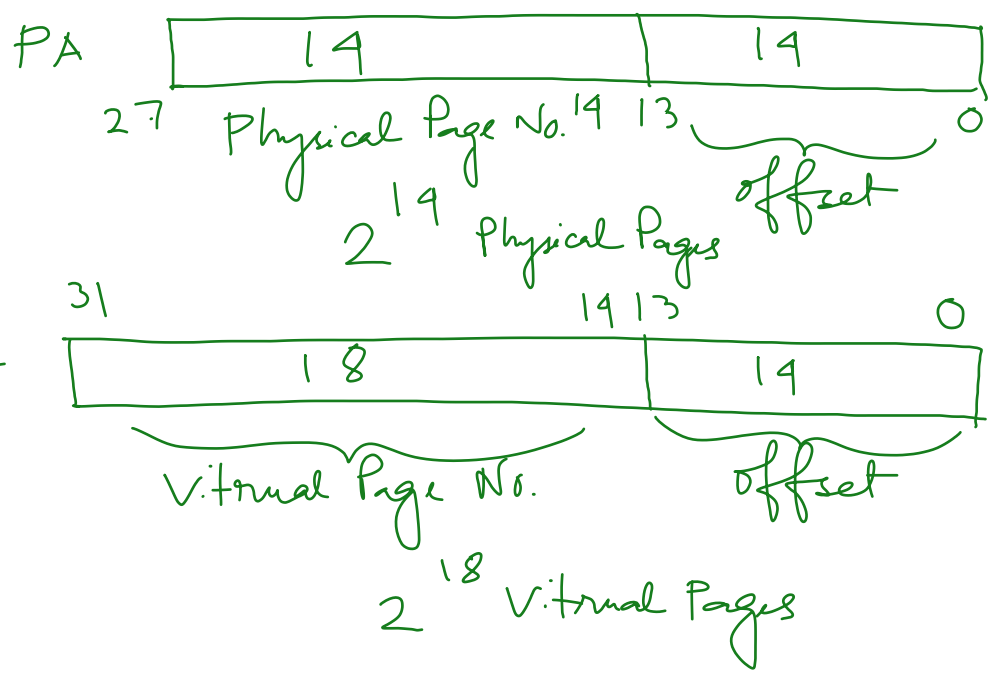


Page Size 16 KB

$= 2^{4} \times 2^{10}$ B

$= 2^{14}$ B

Page offset 14 bit



Q. How do we know if a page is not in RAM?

— No entry in the PT

— PT entry points to disk

— Address does not fit in the physical Address space

256 MB RAM

8 GB RAM 32 VA, 4kB Page Size

i) Page offset would be larger

ii) PA would be the same size as the VA

iii) You would not need VM

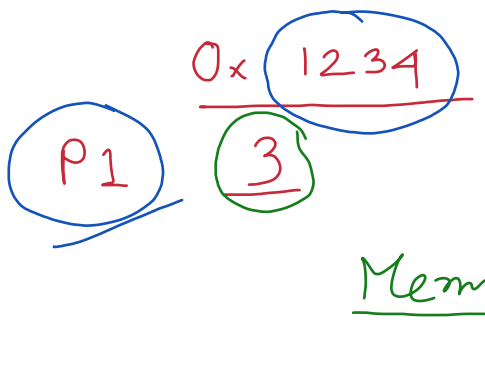
iv) PA would be larger than VA

8 GB RAM - 33 bit PA

Page offset 12 bit

VPN ⇒ 2^{20}

PPN ⇒ 2^{21}



Memory Isolation

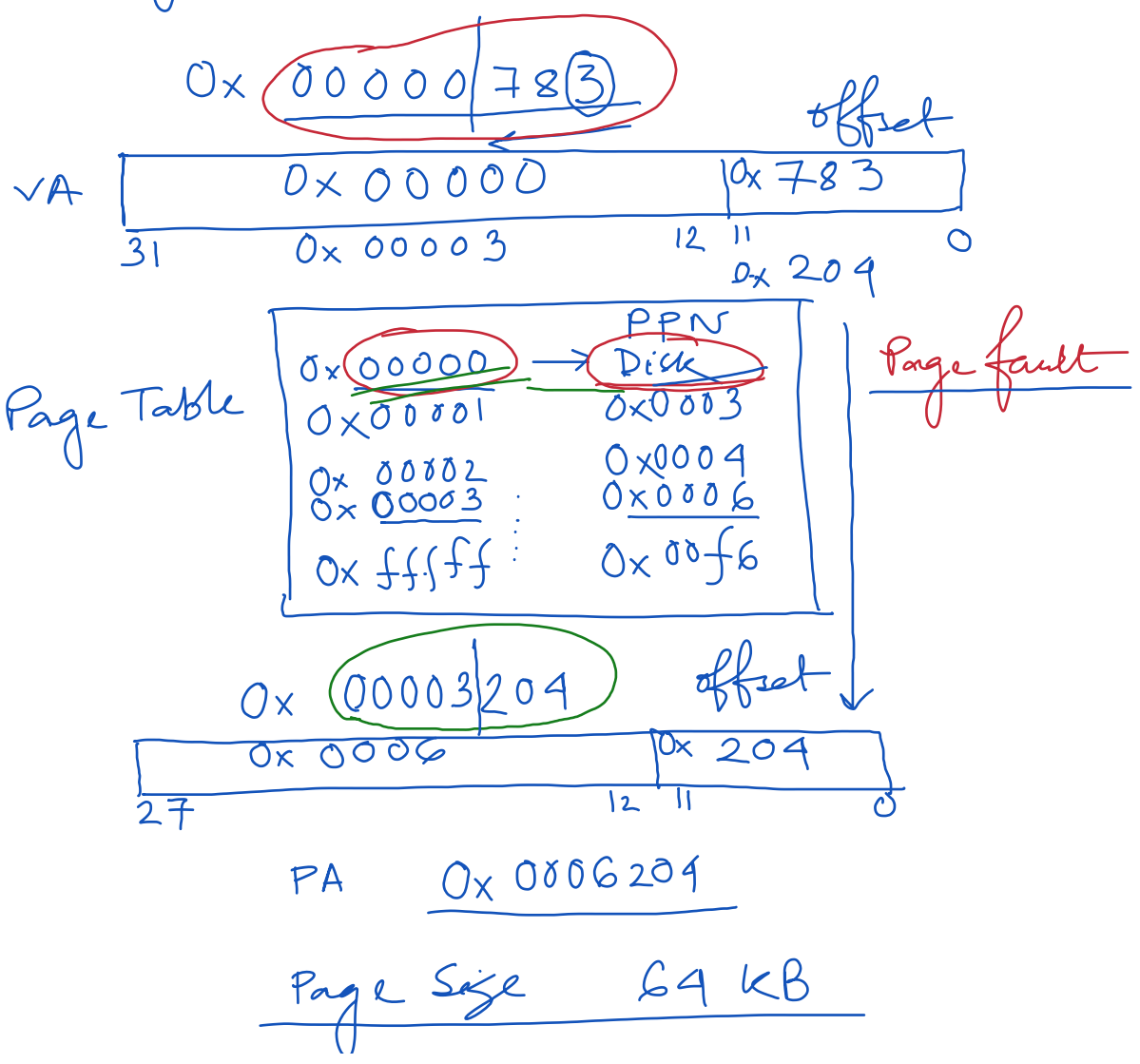
VA 32 bit

PA 28 bit

Page size 4 kB

0x 3

0011



PA 0x 0006204

Page Size 64 kB

- PTE says the page is on disk ~ 1 cycle
- H/w (CPU) generates a Page Fault Exception ~ 100 cycles
- The H/w jumps to OS page fault handler ~ 20000 cycles
  - The OS chooses a page to evict from RAM and write to disk ~ 40,000,000 cycles
  - If the page is dirty, its needs to be written back to disk first
  - OS reads the page from disk and it puts it in RAM ~ 40000 cycles
  - Os then changes the PT to map the new page ~ 1000 cycles
- The OS jumps back to the instruction that caused the page fault ~ 10000 cycles

Next time when this page is required
Will it be Page fault? No

How long does this take?

Short time

long time

incredibly long time 80 million cycles

OS X 10.9 the OS compresses page first.

How will you the victim page?

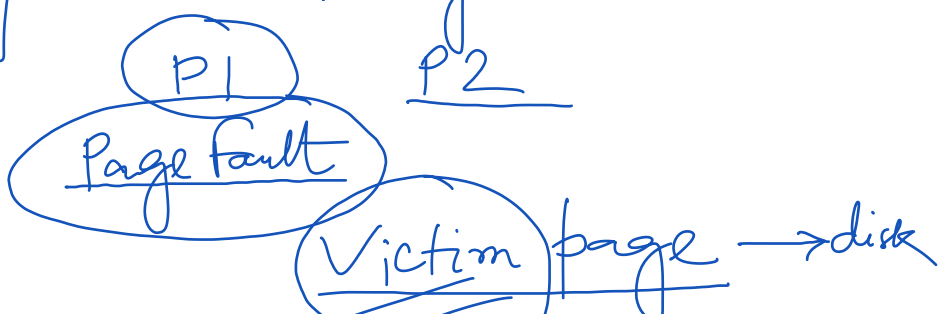
Page Replacement Algorithm

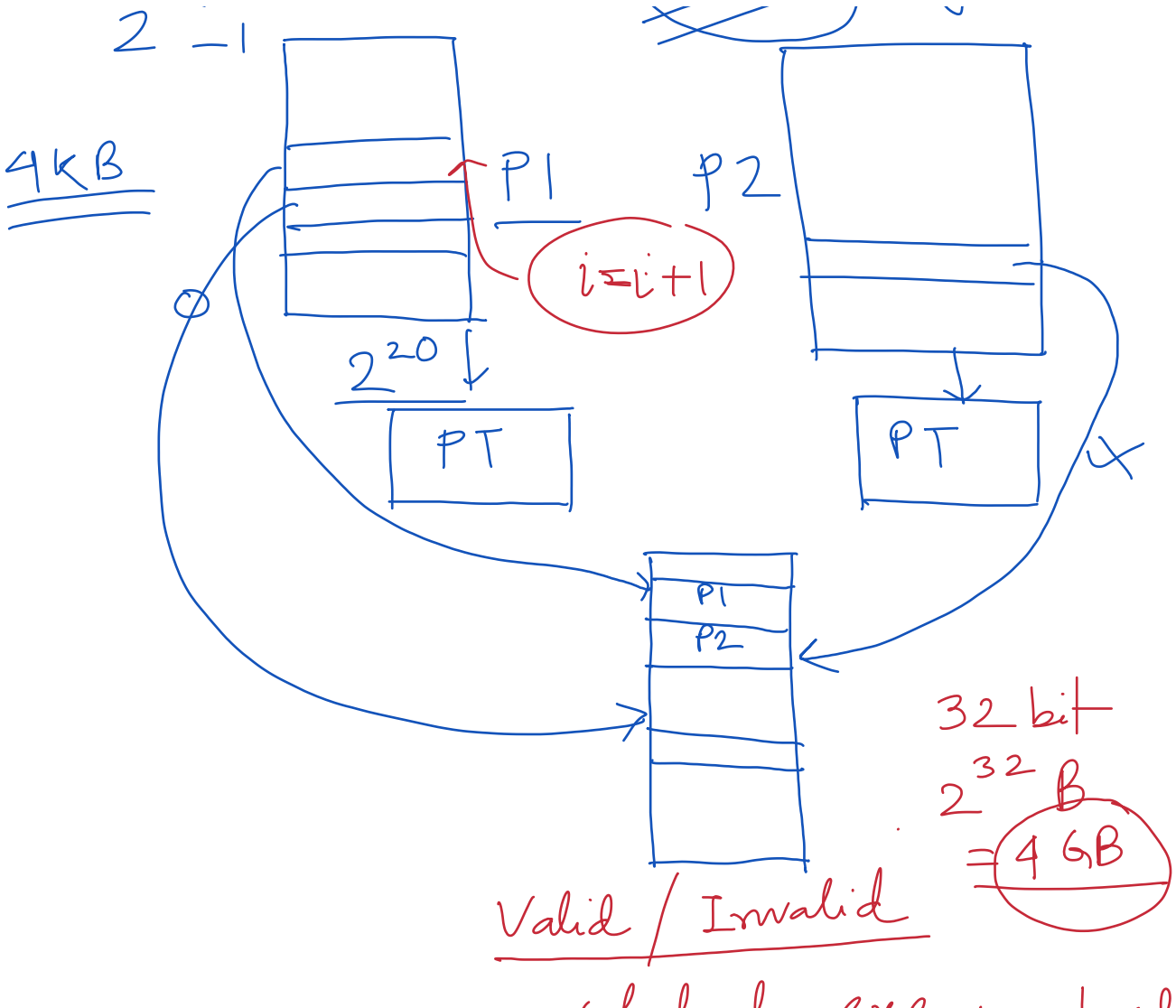
Paged Translation — Coarse grain

Victim page

Modify / Dirty bit — 0 Not modified
bit — 1 Modified

Page Table Entry





Notepad.exe userlevel

Kernel mode — 2 GB workspace
User mode — 2 GB system space

Address Sequence:

0100, 0432, 0101, 0612, 0102,
0103, 0104, 0101, 0611, 0102,
0103, 0104, 0101, 0610, 0102,
0103, 0104, 0101, 0609, 0102,

100 Bytes Per Page

Page size = 100 B

1, 4, 1, 6, 1, 1, 1, 1, 6, 1

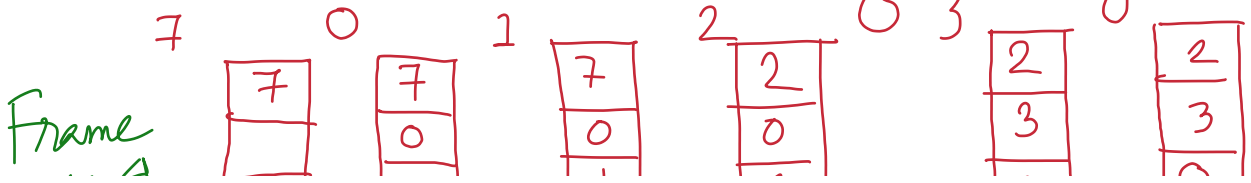
FIFO — FIRST IN FIRST OUT

Demand Paging

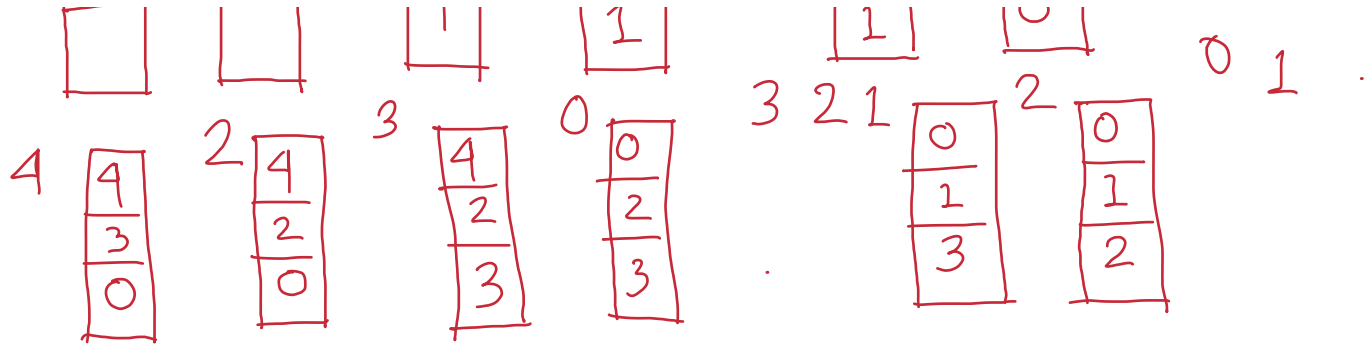
Page Fault = 15

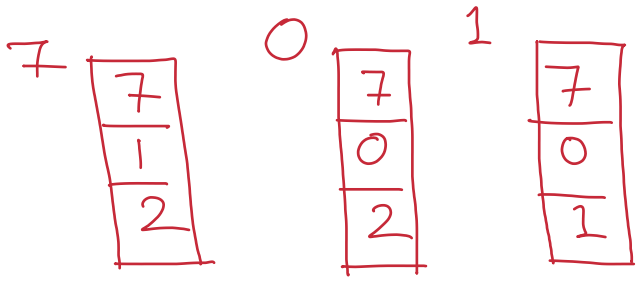
Reference string

7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1



size = 1 and 5





Hit rate

$= \frac{5}{20} \times 100$

Page Hit = 5

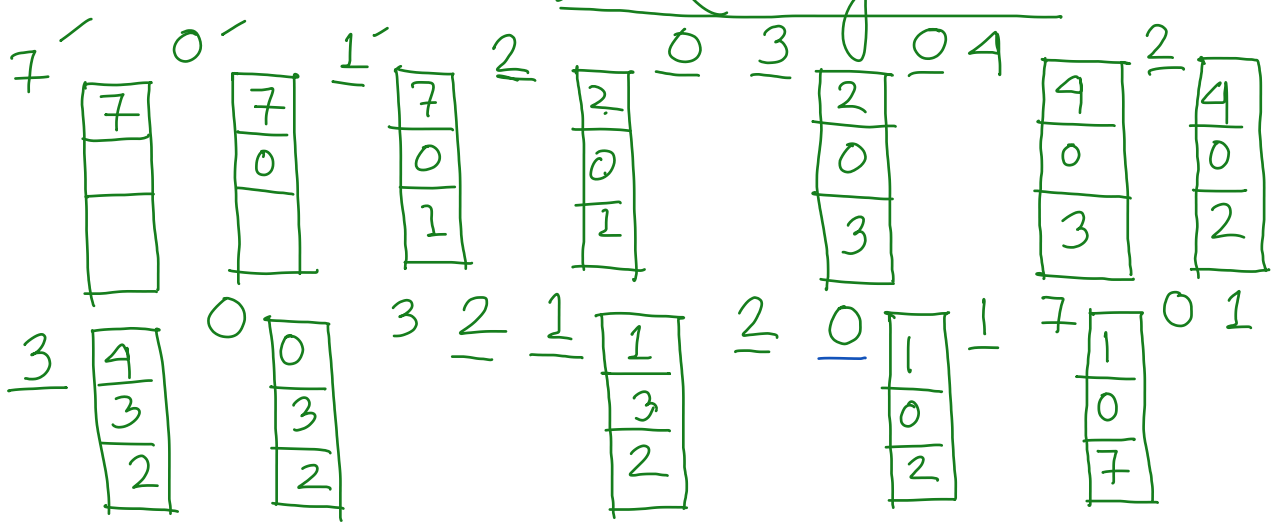
$= 25\%$

Miss Rate = 75%

By increasing frame no. for certain reference string, it has been observed that the no. of page fault increases

Belady's Anamoly

LRU — Least Recently Used



Page Fault = 12

Hit = 8

Hit Ratio $= \frac{8}{20} \times 100$

$= 40\%$

Miss Ratio $= \frac{12}{20} \times 100 = 60\%$

0 — 7

8 = 2^3 binary bit 3

0 — 2

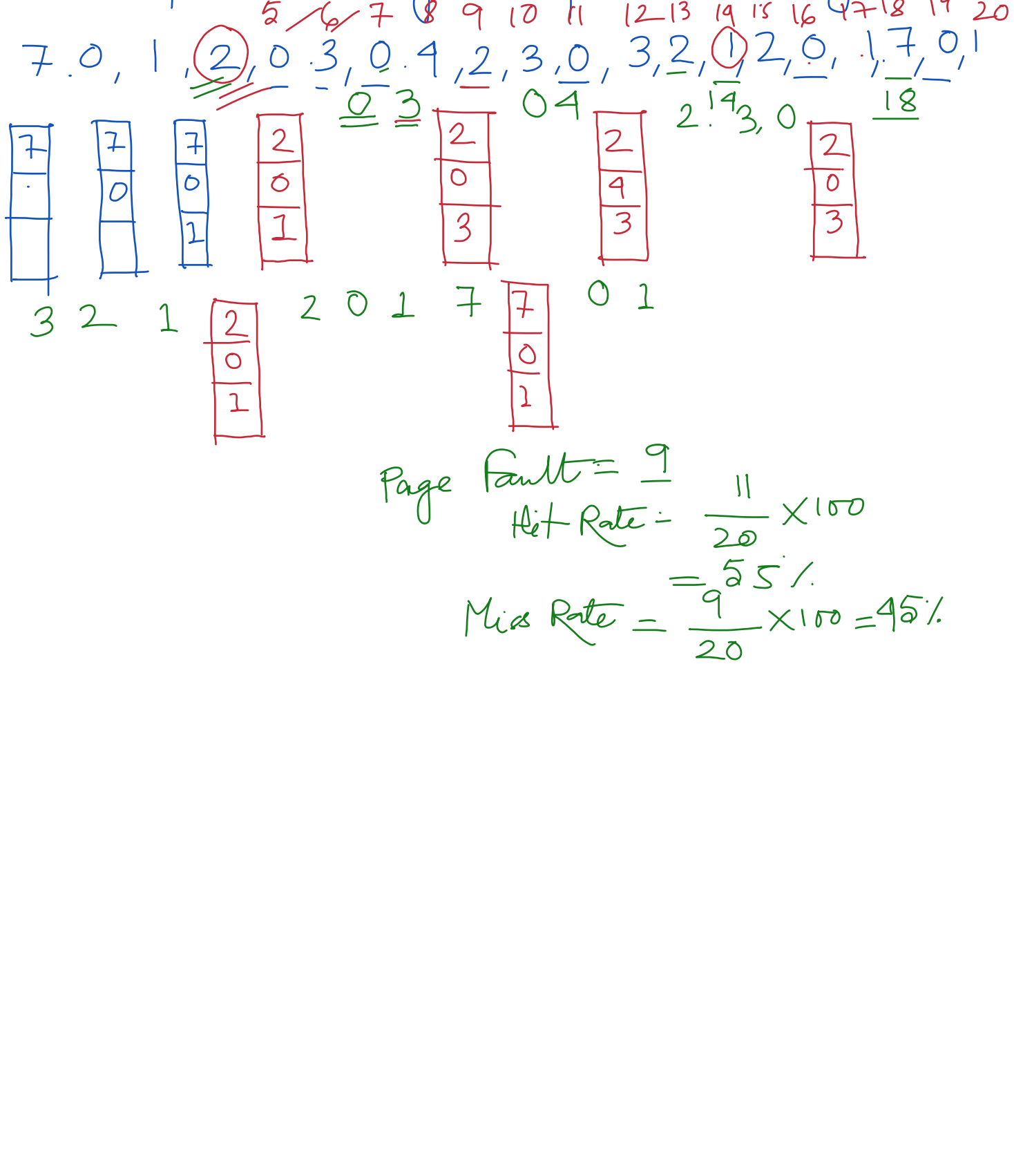
0, 1, 2 — 2 bit — (3)

NRU

8 frames

Pseudo LRU

Optimal Page Replacement Algorithm



Page Fault = 9

Hit Rate = $\frac{11}{20} \times 100 = 55\%$

Miss Rate = $\frac{9}{20} \times 100 = 45\%$