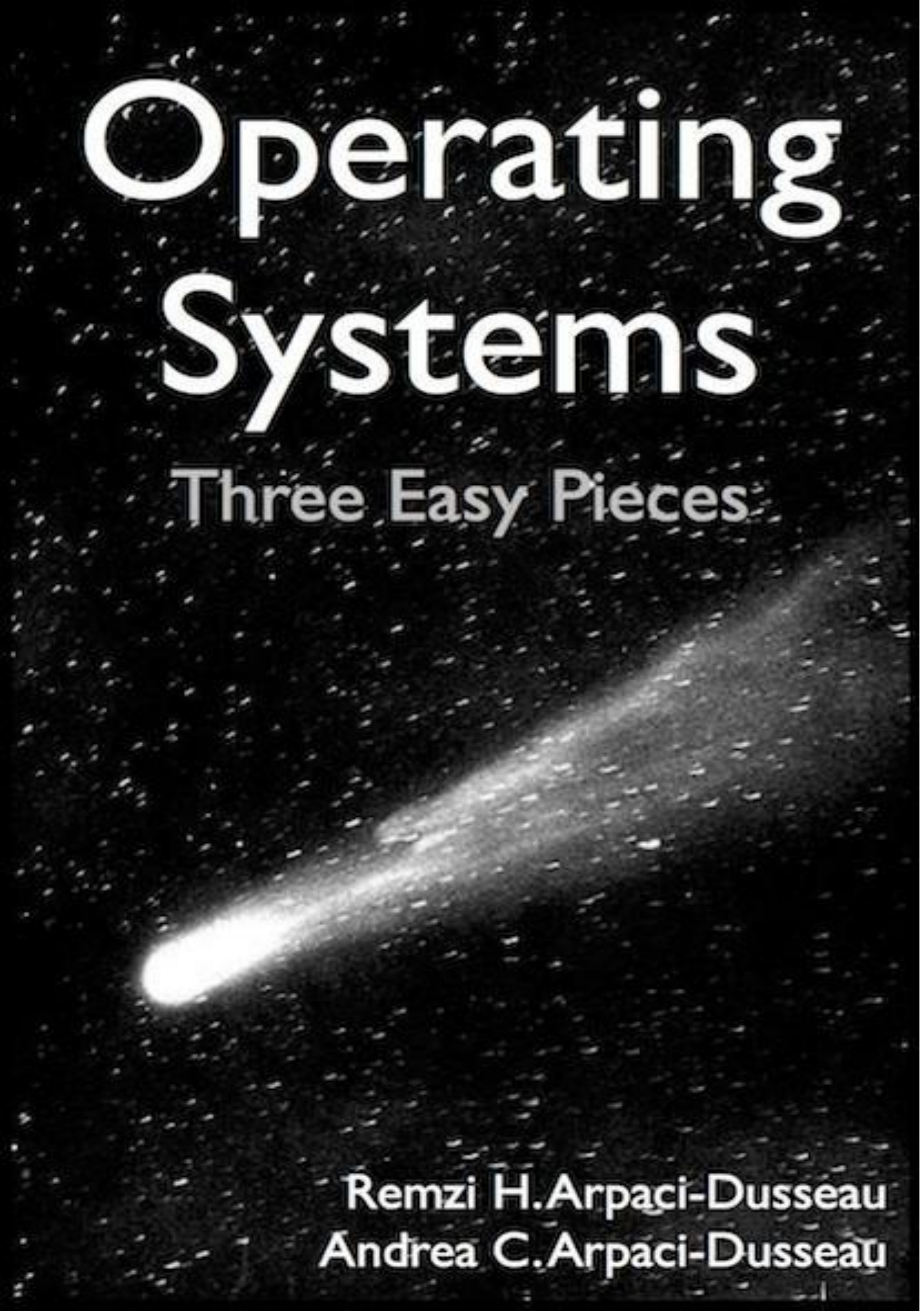
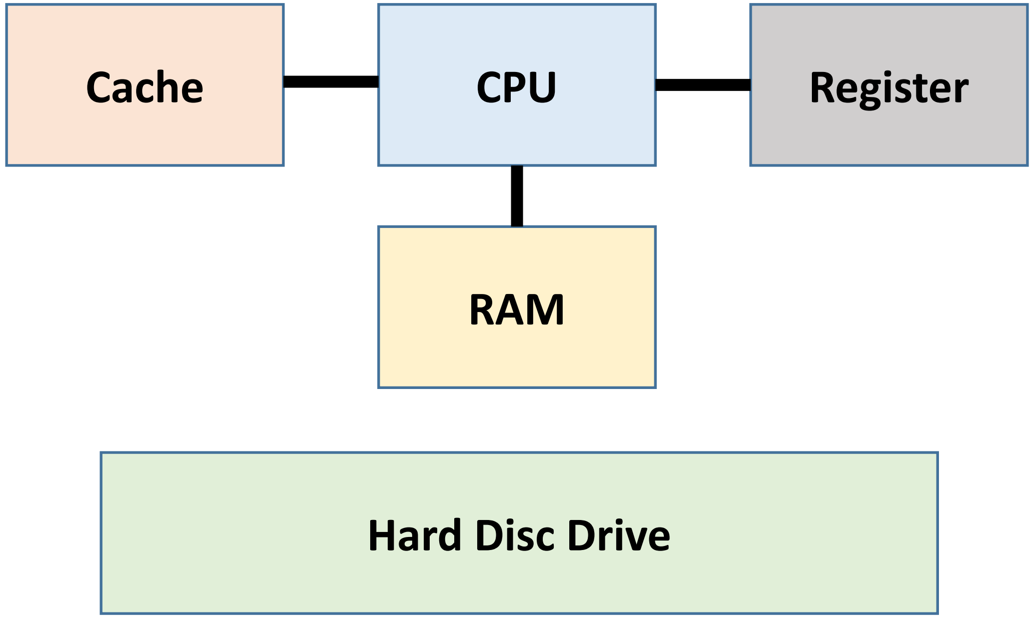
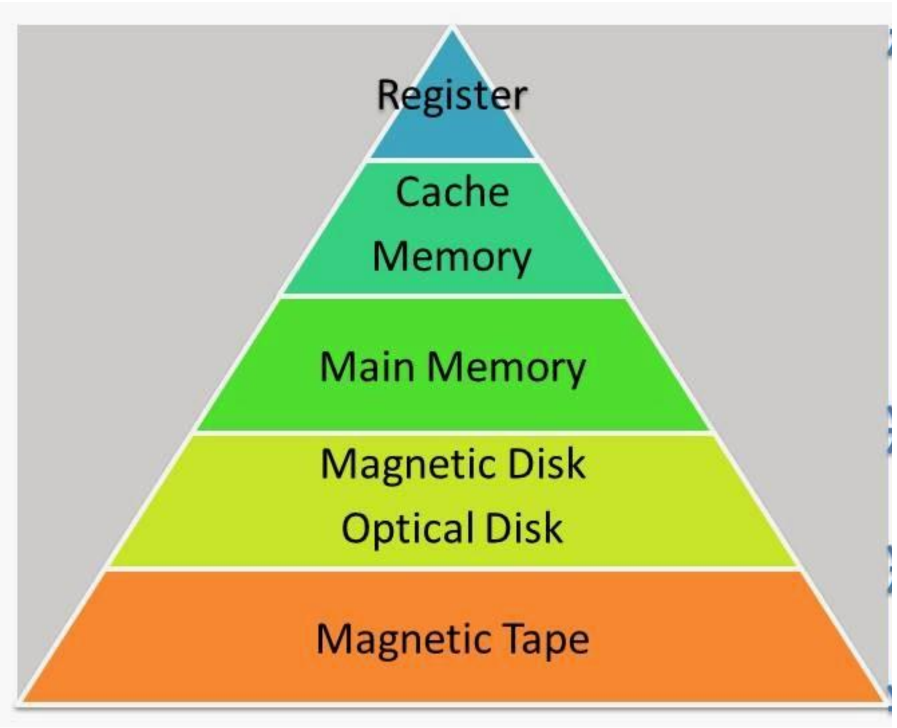



# Memory Management

Prepared by:
***Azizur Rahman Anik***
Lecturer, department of CSE, UIU

# Types of Memory Storages

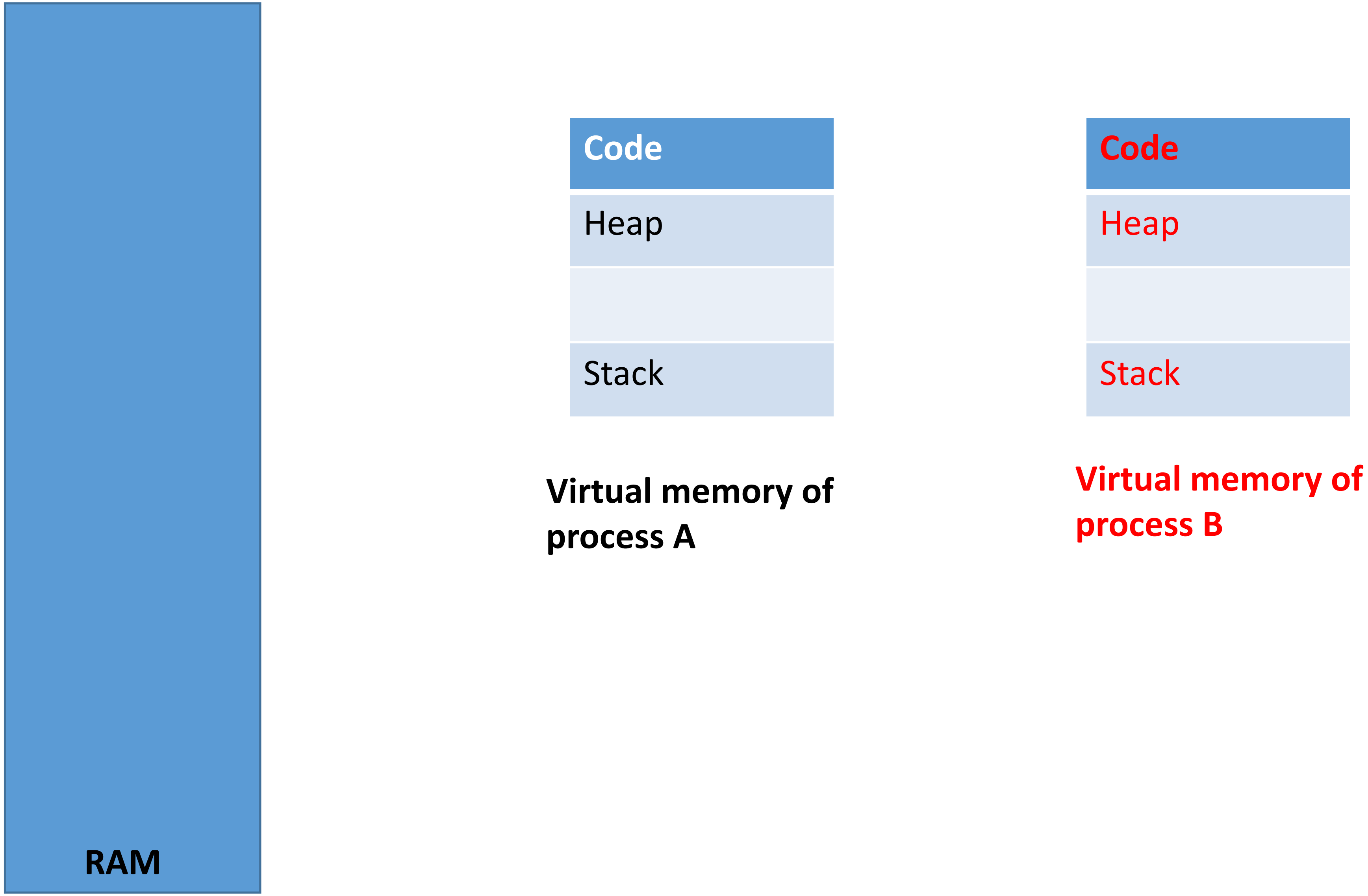


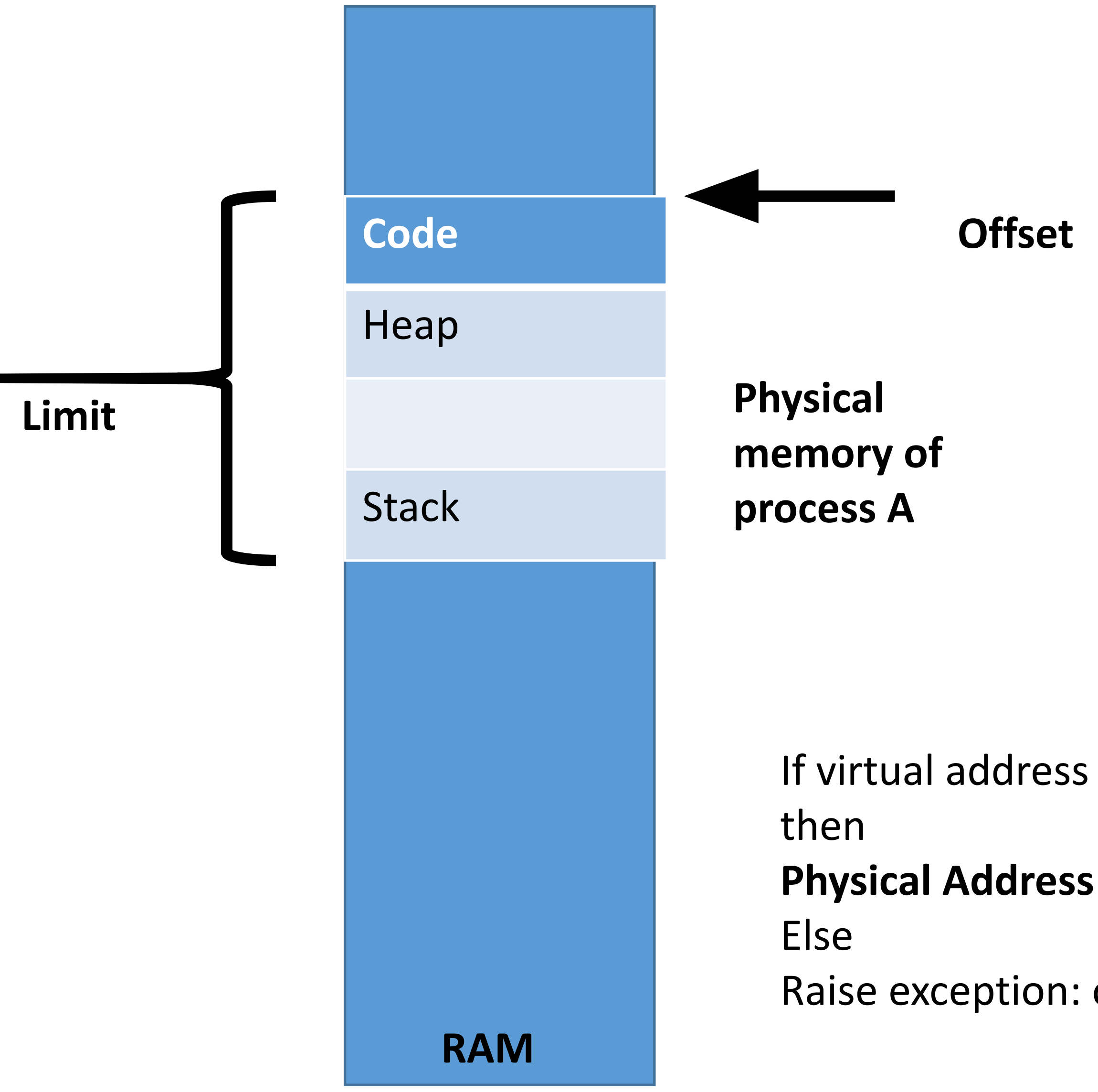
# Physical Memory vs Virtual Memory

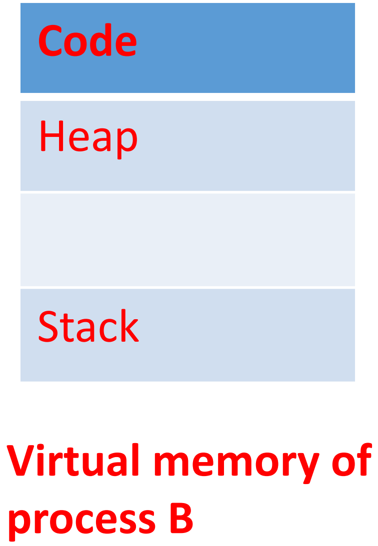
| **Physical Memory** | **Virtual Memory** |
|---|---|
| 1) Physical memory is typically the RAM, which has physical existence. | 1) Virtual memory is the software (VS code, intelliJ etc.) allocated memory for a program, which does not have any physical existence until loaded in physical memory. |
| 2) If the size of used virtual memory is greater than the physical memory, then additional support can be given from HDD | 2) Virtual memory can be greater than or less than the available physical memory in RAM. |

# Address Translation



# Address Translation




If virtual address is within limit,
then
**Physical Address = Offset + Virtual Address**
Else
Raise exception: out of limit

# Address Translation Example

Virtual Address of A's — Physical Address

1) 512 → 2048 + 512 = 2560

↳ offset, 2kB = 2048

2) 1024 → 2048 + 1024 = 3072

3) 4096 → Expecption → Out of limit

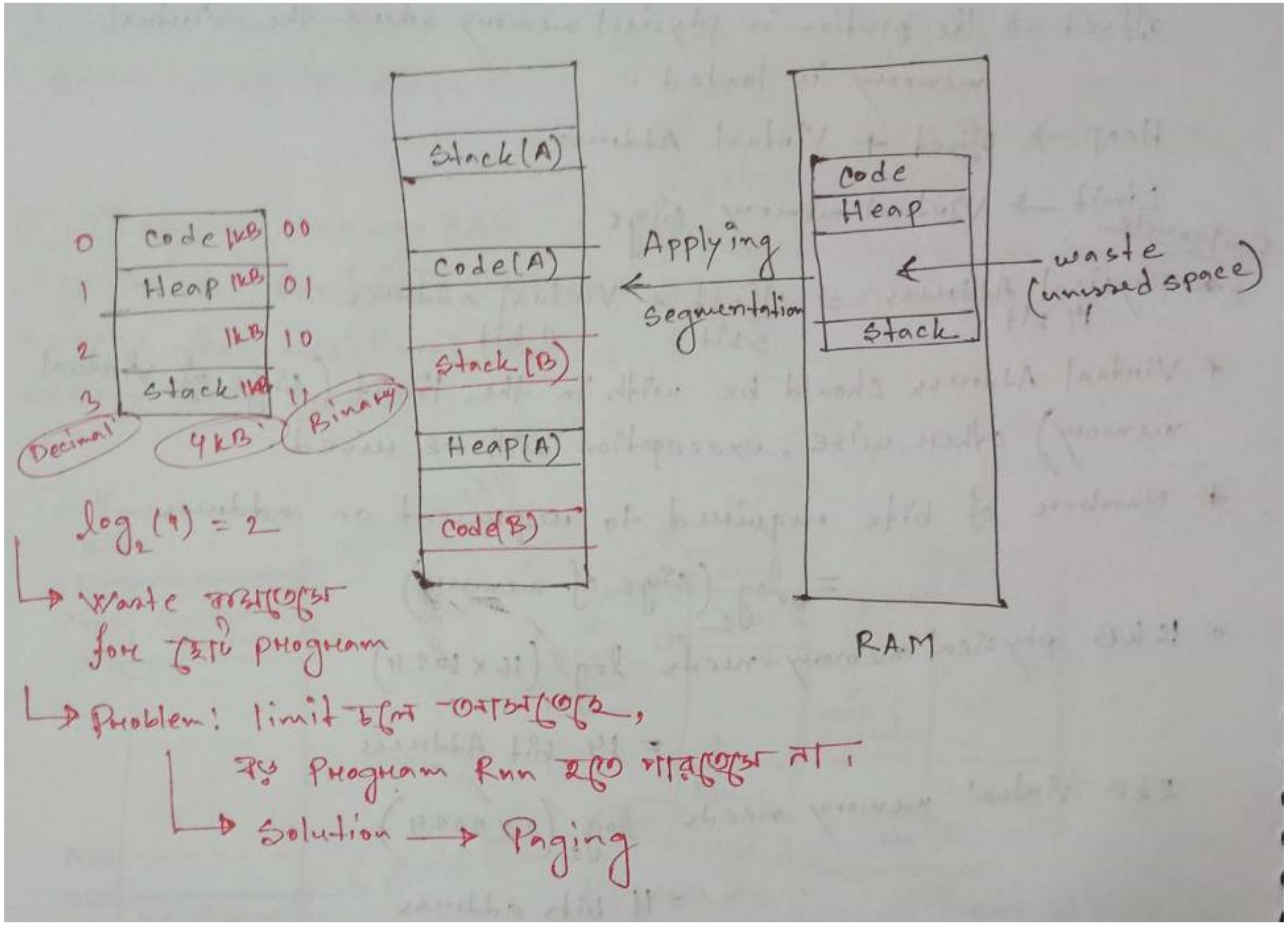
$\boxed{1\text{KB} = 2^{10} = 1024}$

limit = 2KB

= 2 × 1024

= 2048

# Problem with our current Memory Management



# Solution: Paging


- Both physical and virtual memory is divided into same size pages. That means, both type of page has equal number of bytes.

- Bytes in a page are represented by offset.
  Offset = log2(page size in byte)

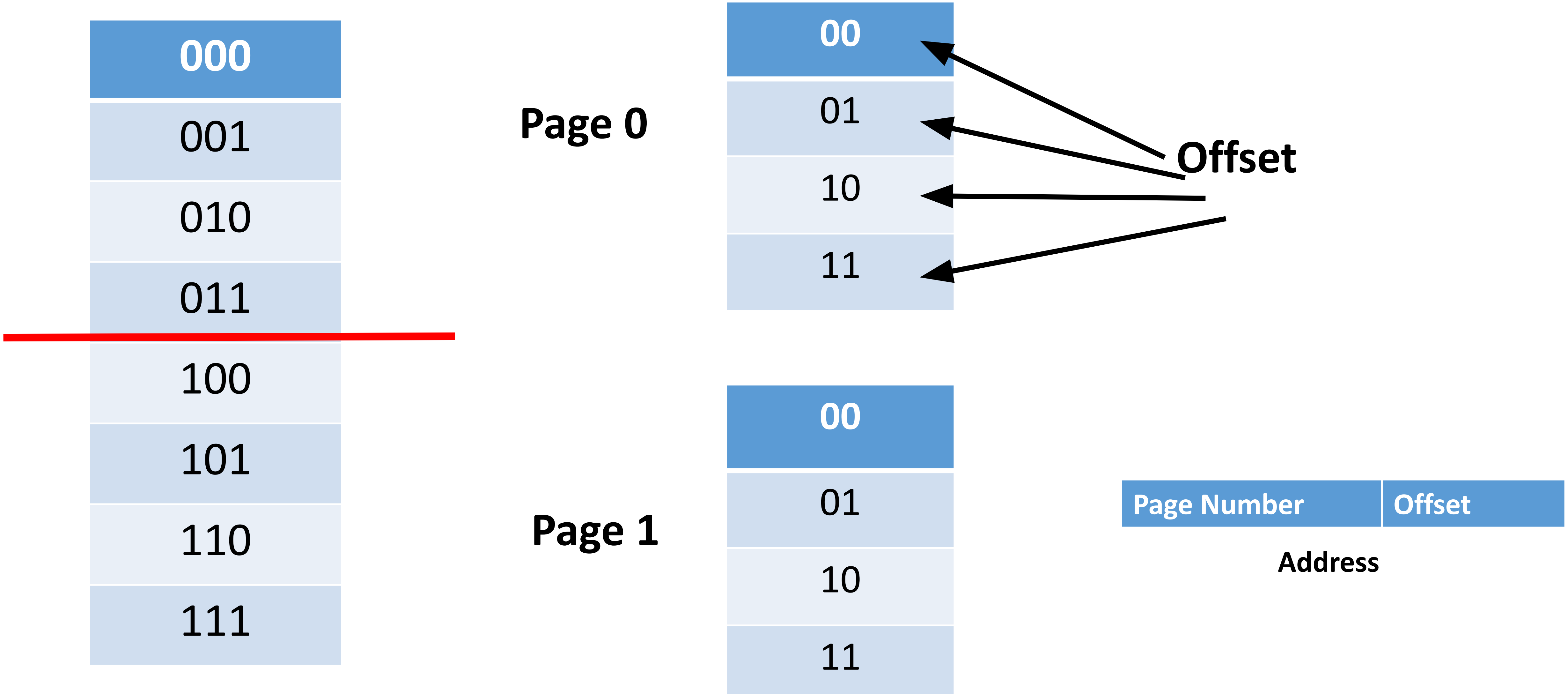
- Virtual memory's page is called virtual page
  Physical memory's page is called physical frame.

- Virtual pages are represented by VPN (Virtual Page Number)

- Physical Frames are represented by PFN (Physical Frame Number)

- Different virtual pages will get mapped to different physical frames. To keep track, we need page table.

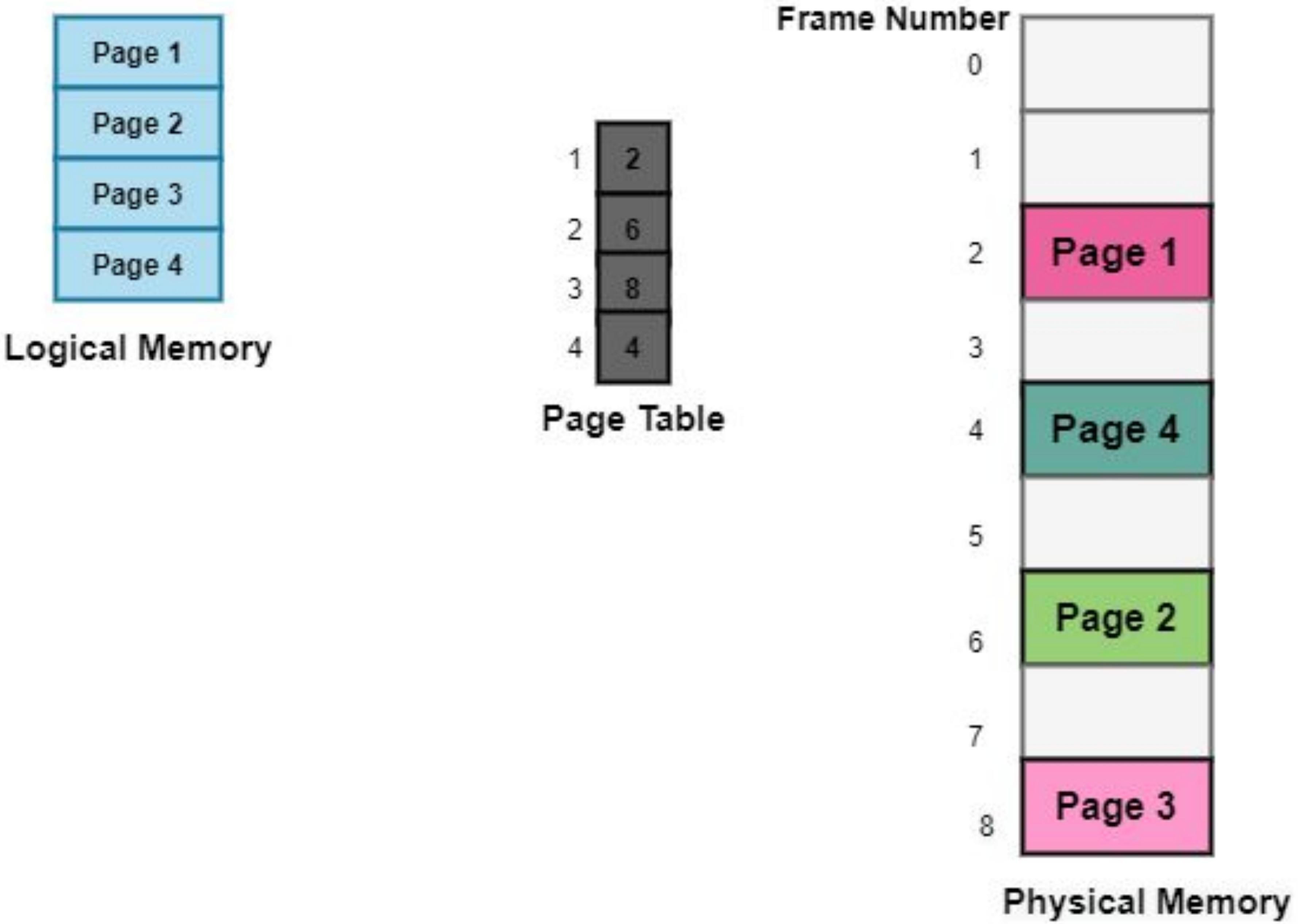
# Things to observe about addresses



# Page Table (Single Level)

- Page table is a data structure kept by the OS for each process.
- Page table resides in physical memory
- Each page table entry (PTE) has PFN (Physical frame number) to which a virtual page is loaded and some flag bits (valid bit, present bit, dirty bit etc.)
- Number of PTEs = Number of virtual pages
- Size of a page table = no. of PTEs * size of a PTE

| VPN | PFN | Valid Bit |
|---|---|---|
| 000 | 0xA12 | 1 |
| 001 | 0x121 | 1 |
| 010 | 0x0AB | 1 |
| 011 | 0x234 | 0 |
| 100 | 0xD12 | 0 |
| 101 | 0x1CA | 0 |
| 110 | 0x156 | 0 |

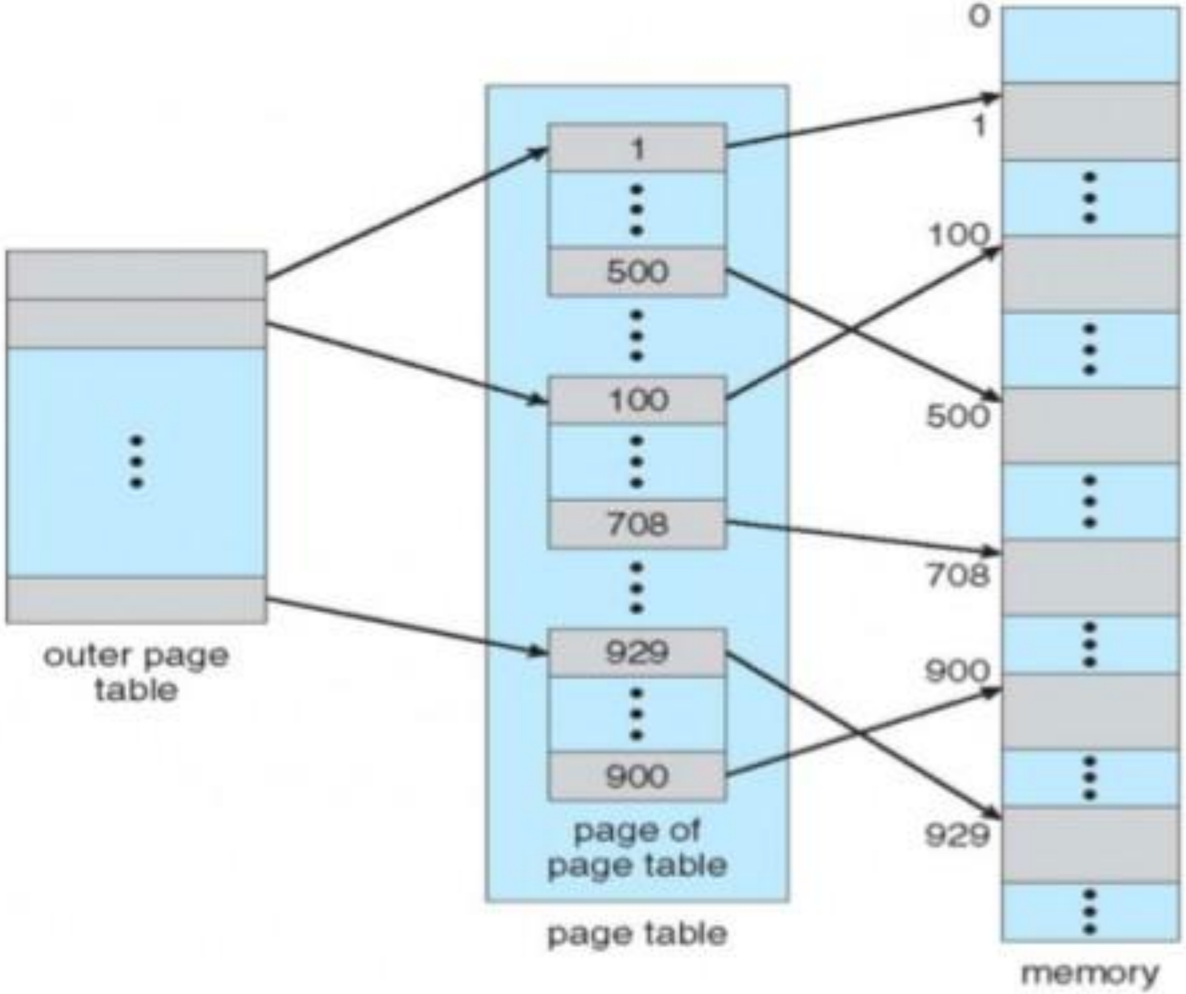

Page 1
Page 2
Page 3
Page 4
Logical Memory
1
2
3
4
2
6
8
4
Page Table
Frame Number
0
1
2
3
4
5
6
7
8
Page 1
Page 4
Page 2
Page 3
Physical Memory

# Problem with Single Page Table

- All virtual pages are not used or valid.

- Single page table can be very large and requires contiguous space in the memory!

# Solution: Multi Level Page Table



Single Page Table

| VPN | PFN | Valid Bit |
|---|---|---|
| 0000 | | 1 |
| 0001 | | 1 |
| 0010 | | 0 |
| 0011 | | 0 |
| 0100 | | 0 |
| 0101 | | 0 |
| 0110 | | 0 |
| 0111 | | 0 |
| 1000 | | 0 |
| 1001 | | 0 |
| 1010 | | 0 |
| 1011 | | 0 |
| 1100 | | 0 |
| 1101 | | 0 |
| 1110 | | 0 |
| 1111 | | 1 |

Page 00

Page 01

Page 10

Page 11

Virtual memory's page numbers

| 0000 |
|---|
| 0001 |
| 0010 |
| 0011 |
| 0100 |
| 0101 |
| 0110 |
| 0111 |
| 1000 |
| 1001 |
| 1010 |
| 1011 |
| 1100 |
| 1101 |
| 1110 |
| 1111 |

**Level 2 Page Table**

| Level 1 page number | PFN | Valid bit |
|---|---|---|
| 00 | | 1 |
| 01 | | 0 |
| 10 | | 0 |
| 11 | | 1 |

**Level 1 page table pages**

**Page 00**

| VPN | PFN | Valid bit |
|---|---|---|
| 00 | | 1 |
| 01 | | 1 |
| 10 | | 0 |
| 11 | | 0 |

**Not Allocated**

**Not Allocated**

**Page 11**

| VPN | PFN | Valid bit |
|---|---|---|
| 00 | | 0 |
| 01 | | 0 |
| 10 | | 0 |
| 11 | | 1 |

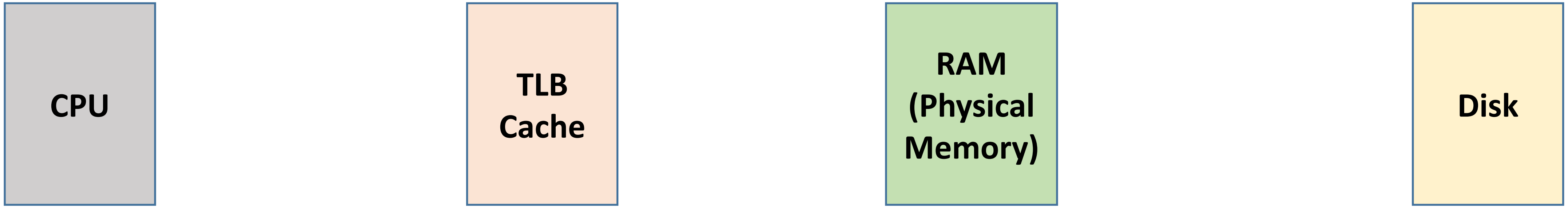
**Virtual memory's page numbers**

| 0000 |
|---|
| 0001 |
| 0010 |
| 0011 |
| 0100 |
| 0101 |
| 0110 |
| 0111 |
| 1000 |
| 1001 |
| 1010 |
| 1011 |
| 1100 |
| 1101 |
| 1110 |
| 1111 |

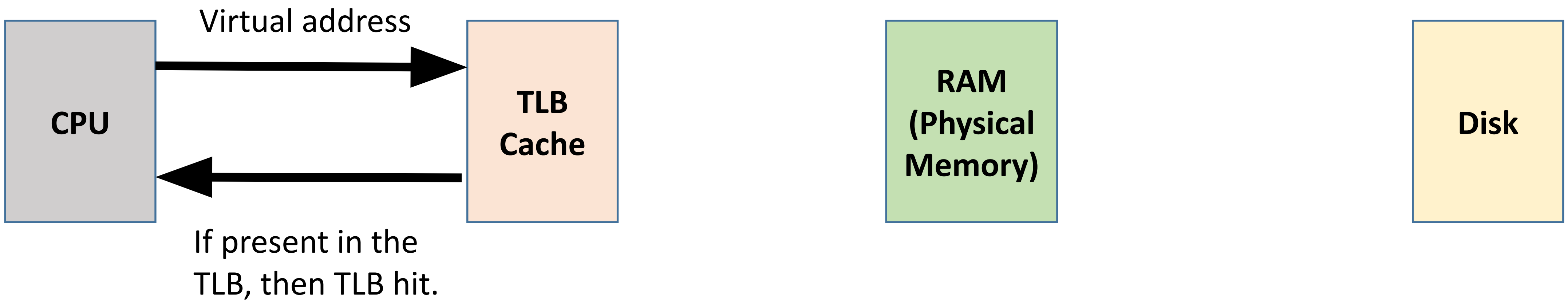
# Multi Level Page Table

- Each level's page table will be divided into page sized segments so that each segment can be kept in a single page.
- Level 1 page table keeps the VPN to PFN mapping
- Level 2 page table keeps the track of Level 1 pages
- Level 3 page table keeps the track of Level 2 pages and so on.

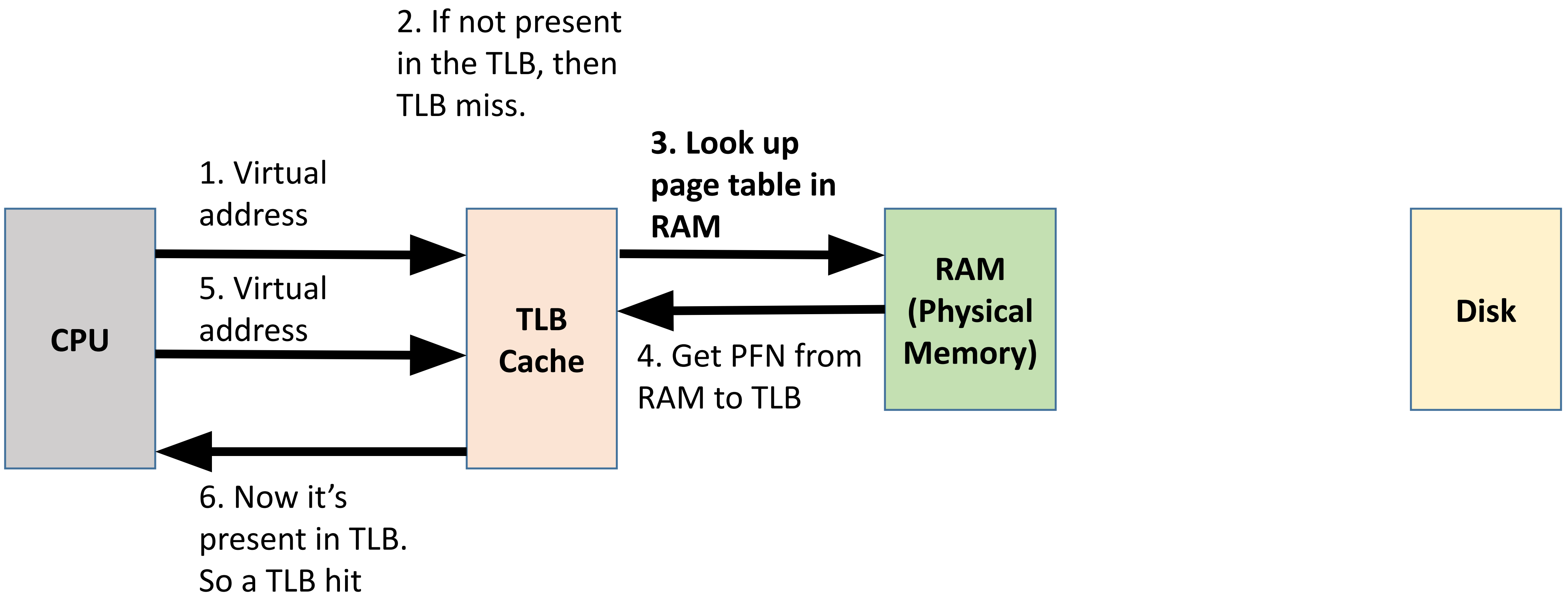
# The Big Picture of Memory Access



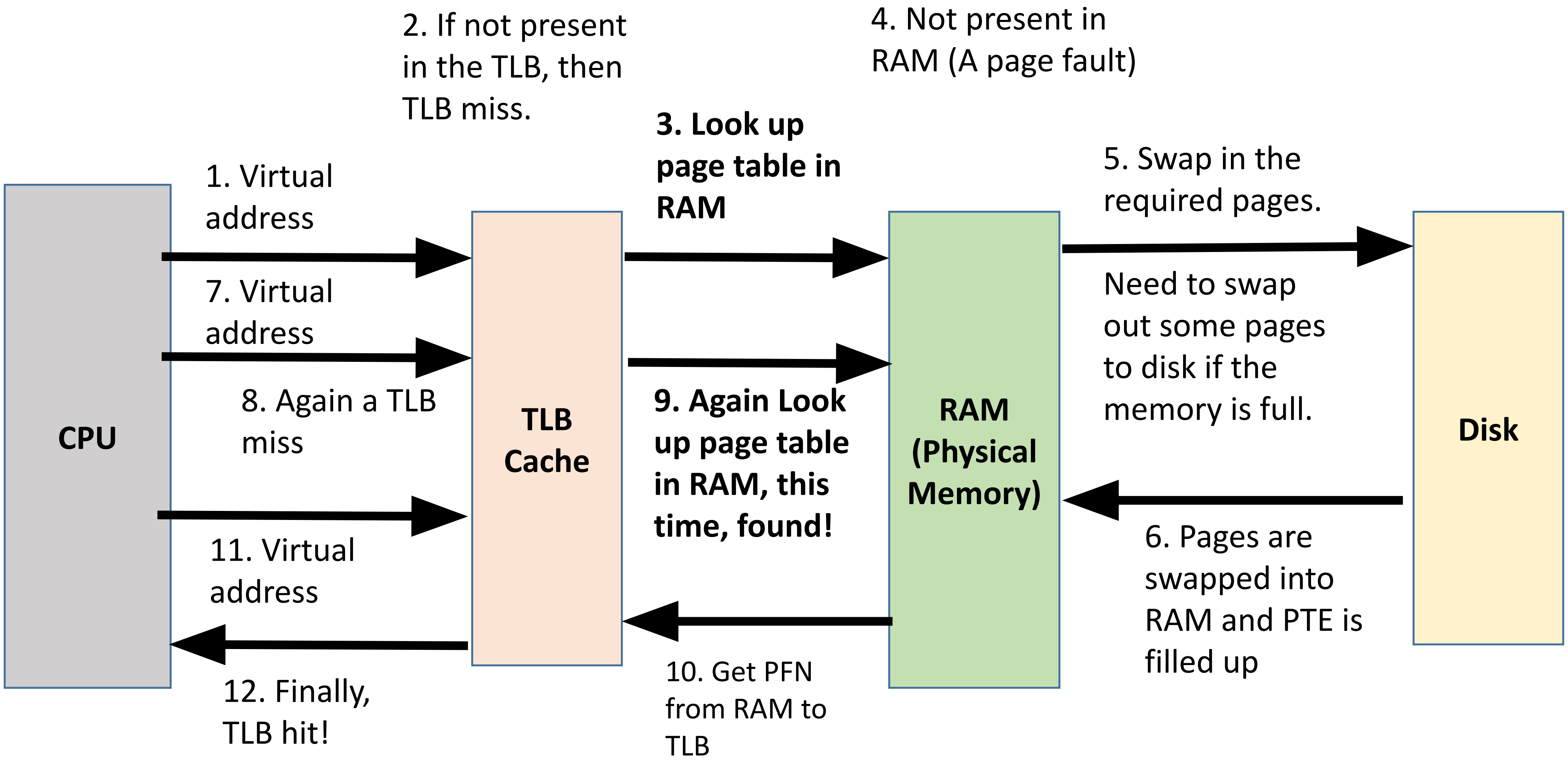
# The Big Picture of Memory Access



# The Big Picture of Memory Access



# The Big Picture of Memory Access



# Memory Access Time Calculation

Suppose, a system uses Translation Lookaside Buffer (TLB) while calculating physical address from logical address.

The size of virtual memory is 64KB and physical memory is 32 KB. Page size is 4KB. CPU requests for five pages: 0xAF50, 0x20BA, 0xB8C2, 0x8CDA, 0xEBCD.

Calculate the Effective Memory Access Time if TLB access time 20ns and memory access time 100ns. TLB and page table is shown in figure-1.

| Page Number | Frame Number |
|---|---|
| 1011 | 010 |
| 1001 | 001 |
| 0010 | 000 |
| 1000 | 100 |

(a) TLB

| | Frame | v |
|---|---|---|
| 15 | 111 | 1 |
| | 110 | 1 |
| | 100 | 0 |
| | 010 | 0 |
| | 010 | 1 |
| | 011 | 1 |
| | 001 | 1 |
| | 100 | 1 |
| | 110 | 0 |
| | 000 | 0 |
| | 001 | 0 |
| | 111 | 0 |
| | 010 | 0 |
| | 000 | 1 |
| | 010 | 0 |
| 0 | 101 | 1 |

(b) Page Table

# Page Replacement Policy

- FIFO
- Random Replacement
- Optimal Replacement
- Least Recently Used (LRU)

# Least Recently Used (LRU)

- the page that has been used least recently will be selected for removal from physical memory when a new page needs to be loaded into memory, and there is no free space available.

- LRU assumes that pages that have not been used recently will not be used in the near future.

# Least Recently Used

- Imagine a computer system with a physical memory that can hold up to 4 pages. This system uses virtual memory management with a page replacement policy based on the Least Recently Used (LRU) algorithm. You are provided with a sequence of page references and must simulate the behavior of the system as it processes this sequence.

- The sequence of page references is as follows:

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

# Least Recently Used

- The sequence of page references is as follows: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0

| Reference | Hit/Fault | Evicted | Memory State |
|---|---|---|---|
| 7 | Fault | None | 7 |
| 0 | Fault | None | 7, 0 |
| 1 | Fault | None | 7, 0, 1 |
| 2 | Fault | None | 7, 0, 1, 2 |
| 0 | Hit | None | 7, 0, 1, 2 |
| 3 | Fault | 7 | 3, 0, 1, 2 |
| 0 | Hit | None | 3, 0, 1, 2 |
| 4 | Fault | 1 | 3, 0, 4, 2 |
| 2 | Hit | None | 3, 0, 4, 2 |
| 3 | Hit | None | 3, 0, 4, 2 |
| 0 | Hit | None | 3, 0, 4, 2 |