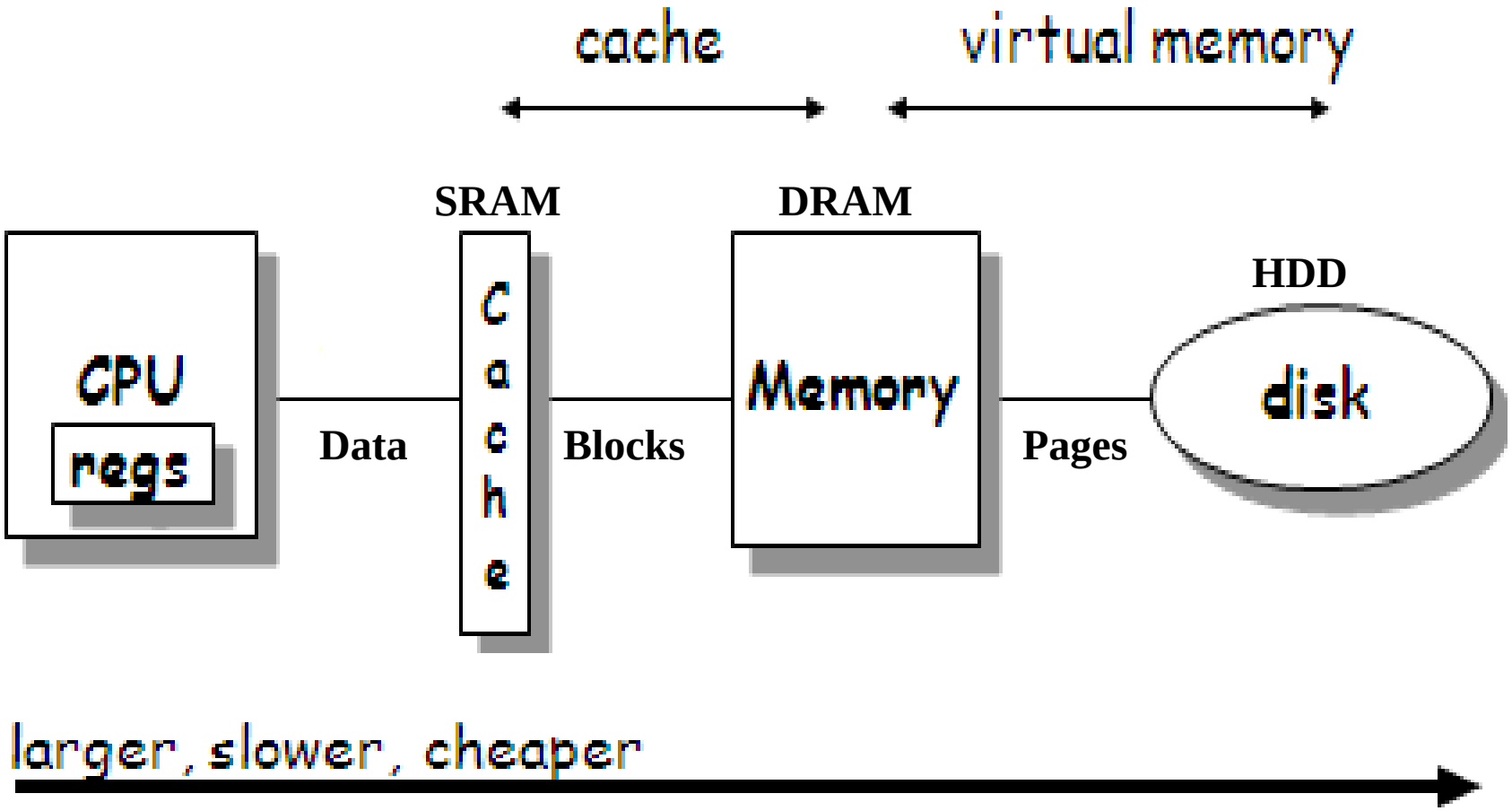
Virtual Memory

P. K. Roy

Levels of Memory



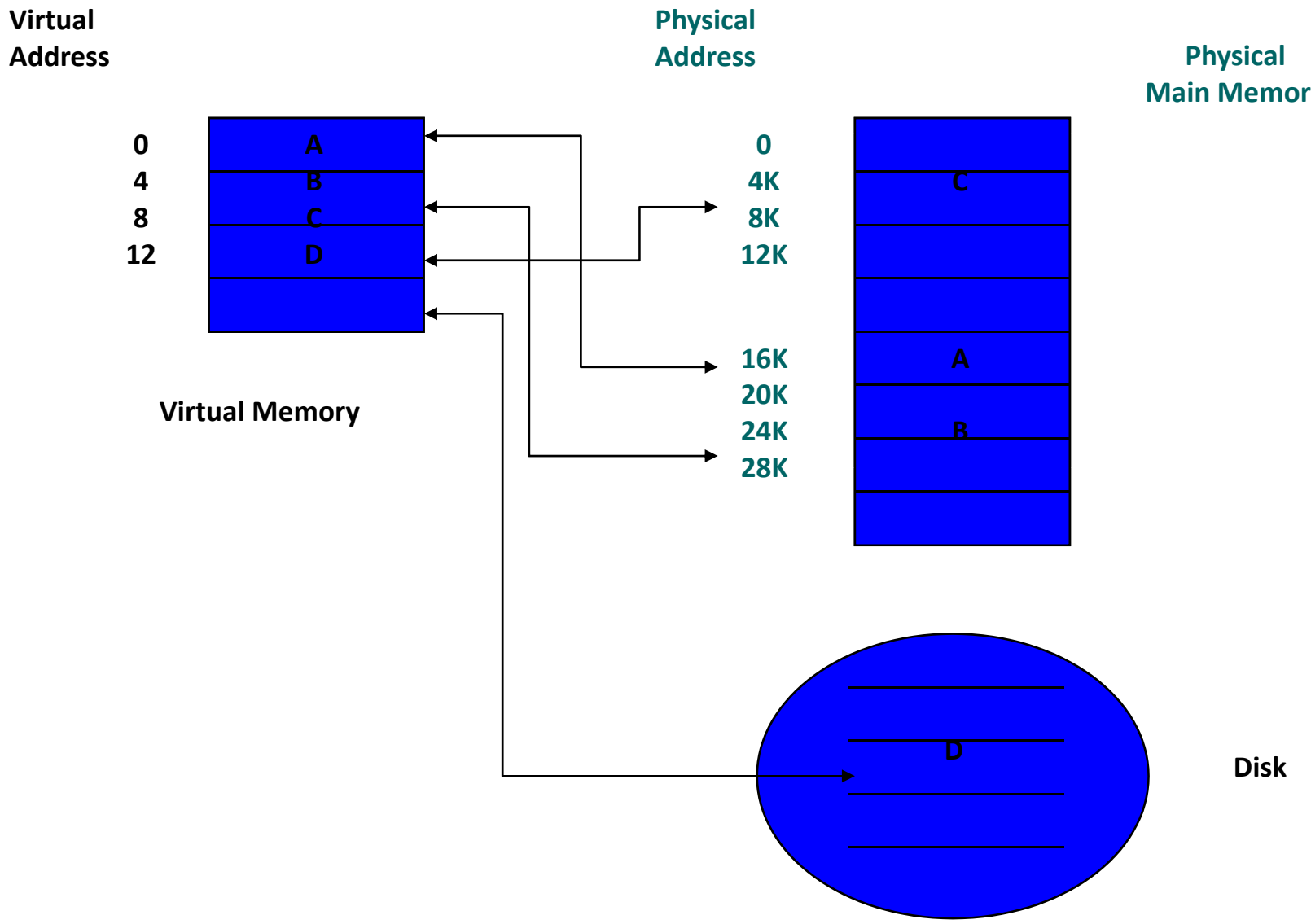
What is it?

- Provides mapping between memory addresses (auxiliary memory) used by a program, called ***virtual/logical addresses***, and ***physical addresses*** of computer memory (main memory).
- It is a ***memory management technique*** for some large computer systems, which provides the programmer an illusion of having a large main memory.
- Invisible to the programmer.
- CPU is responsible for generating ***virtual addresses***.
- The size of virtual memory (***virtual address space***) is equivalent to the size of secondary memory.
- Virtual memory allows main memory (DRAM) to act like a cache for secondary storage (magnetic disk).
- Allows any program to run anywhere.
- Allows multiple programs to share the same physical memory.
- Allows program to run in any location in physical memory .

How does it work?

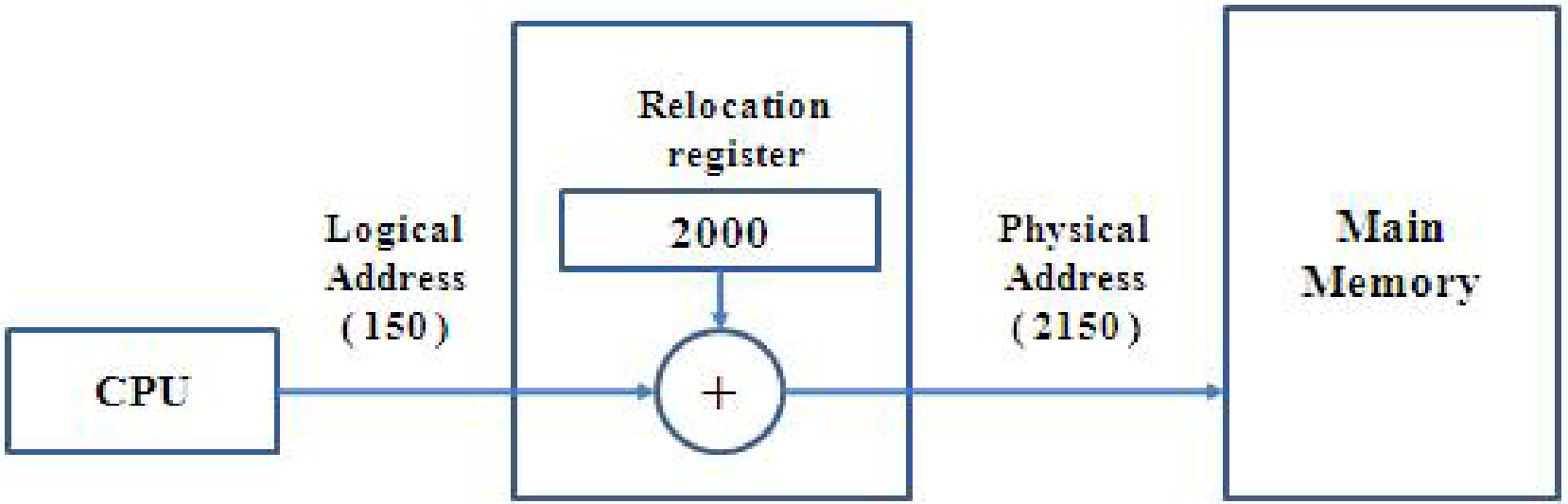
- To facilitate copying virtual memory into real memory, the operating system divides virtual memory into pages, each of which contains a fixed number of addresses.
- Each page is stored on a disk until it is needed.
- When the page is needed, the operating system copies it from disk to main memory, translating the virtual addresses into physical addresses.
- Virtual-to-physical translation by indexed table lookup
  - Add another cache for recent translations (the TLB)

Cntd…


Virtual
Address
Physical
Address
Physical
Main Memory
0
4
8
12
A
B
C
D
Virtual Memory
0
4K
8K
12K
16K
20K
24K
28K
C
A
B
D
Disk

MMU (Memory Management Unit)

- The hardware base that makes a virtual memory system possible.
- Sometimes called ***paged memory management unit*** (PMMU).
- It is a computer hardware unit having all memory references passed through itself, primarily performing the translation of virtual memory addresses to physical addresses.
- It is usually implemented as part of the CPU, but it also can be in the form of a separate Integrated circuit.



Implementation of Virtual Memory System

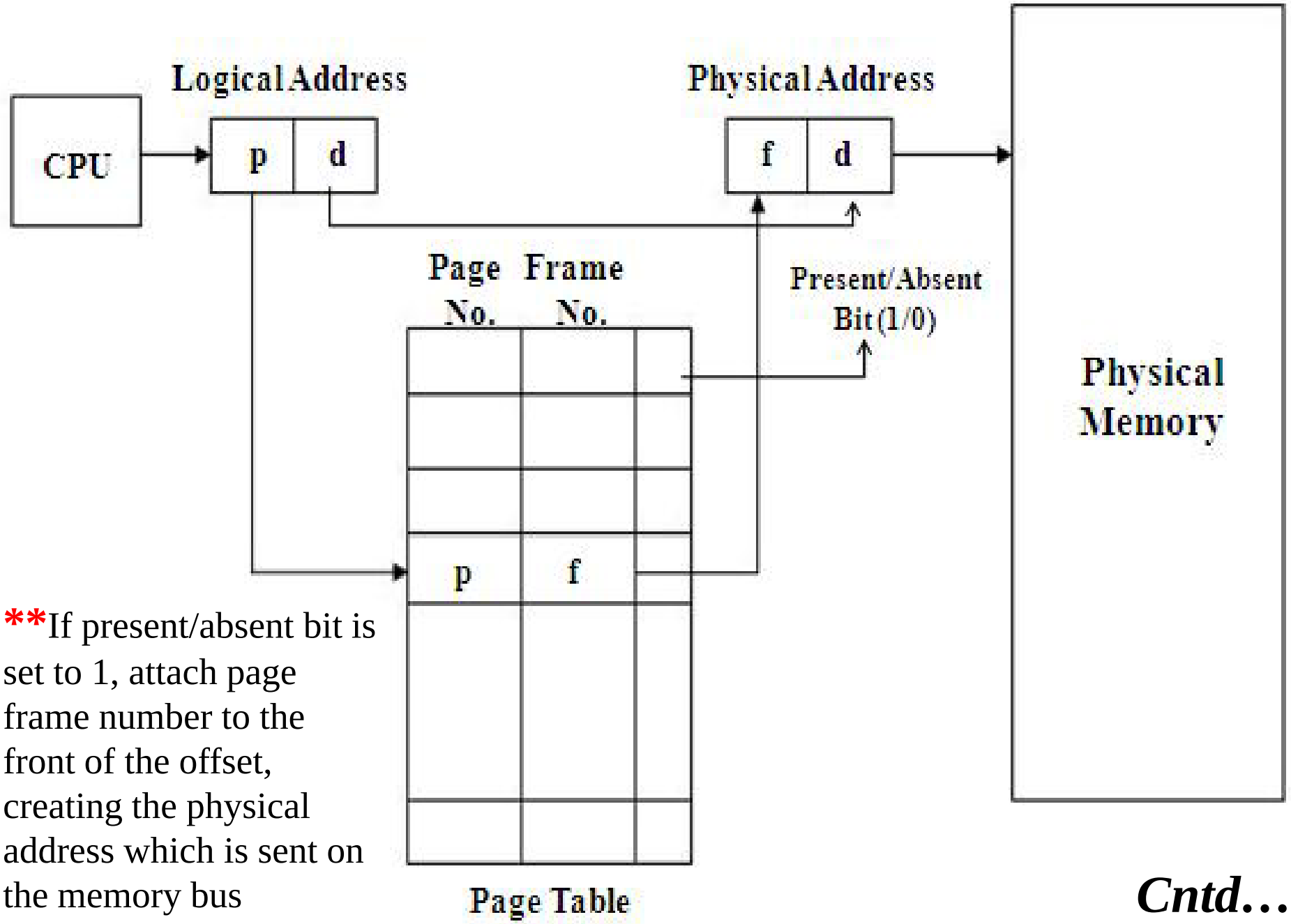
1. Paging

   => TLB(Translation Look-aside Buffer)

2. Segmentation

1. Paging

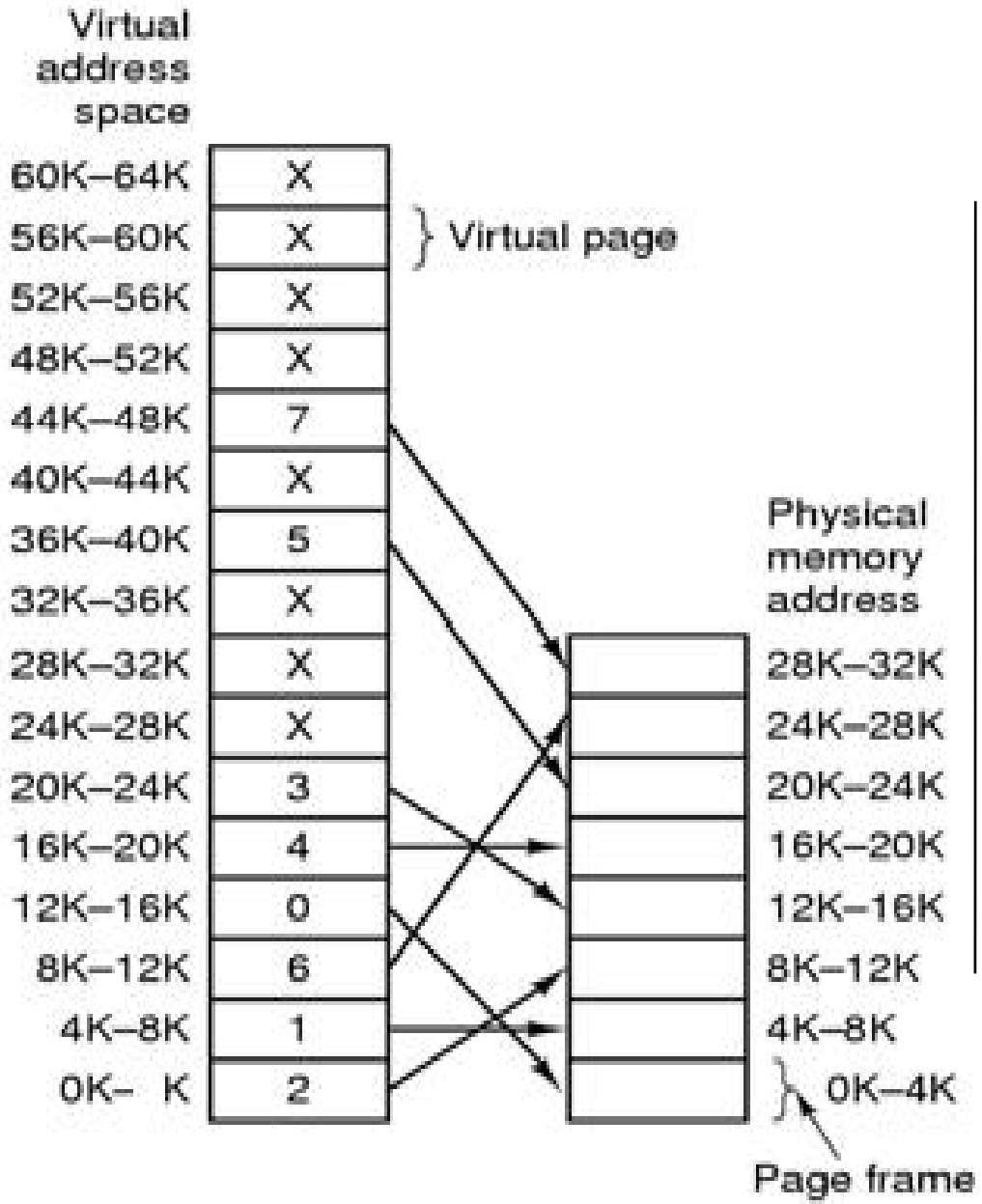
1. Non-contiguous memory allocation.
2. ***Virtual address space*** is divided into equal size blocks called ***pages***.
3. ***Physical address space*** is divided into equal size blocks called ***frames***.
4. Size of a ***page*** = Size of a ***frame***
5. Size of a page or frame is dependent on the OS (generally 4KB).
6. OS maintains separate ***page tables*** for different programs to map logical address into physical address.
7. A page table consists of 2 fields, ***page number & frame number***, to specify that which page would be mapped to which frame.
8. CPU-generated address (virtual/logical address) is divided into 2 parts: ***page number (p) & offset/displacement (d)***.
9. The page no. p is used as index in the page table and the offset d is the word no. within the page p.



**If present/absent bit is set to 1, attach page frame number to the front of the offset, creating the physical address which is sent on the memory bus

Cntd…

Example



•Non-contiguous memory allocation.

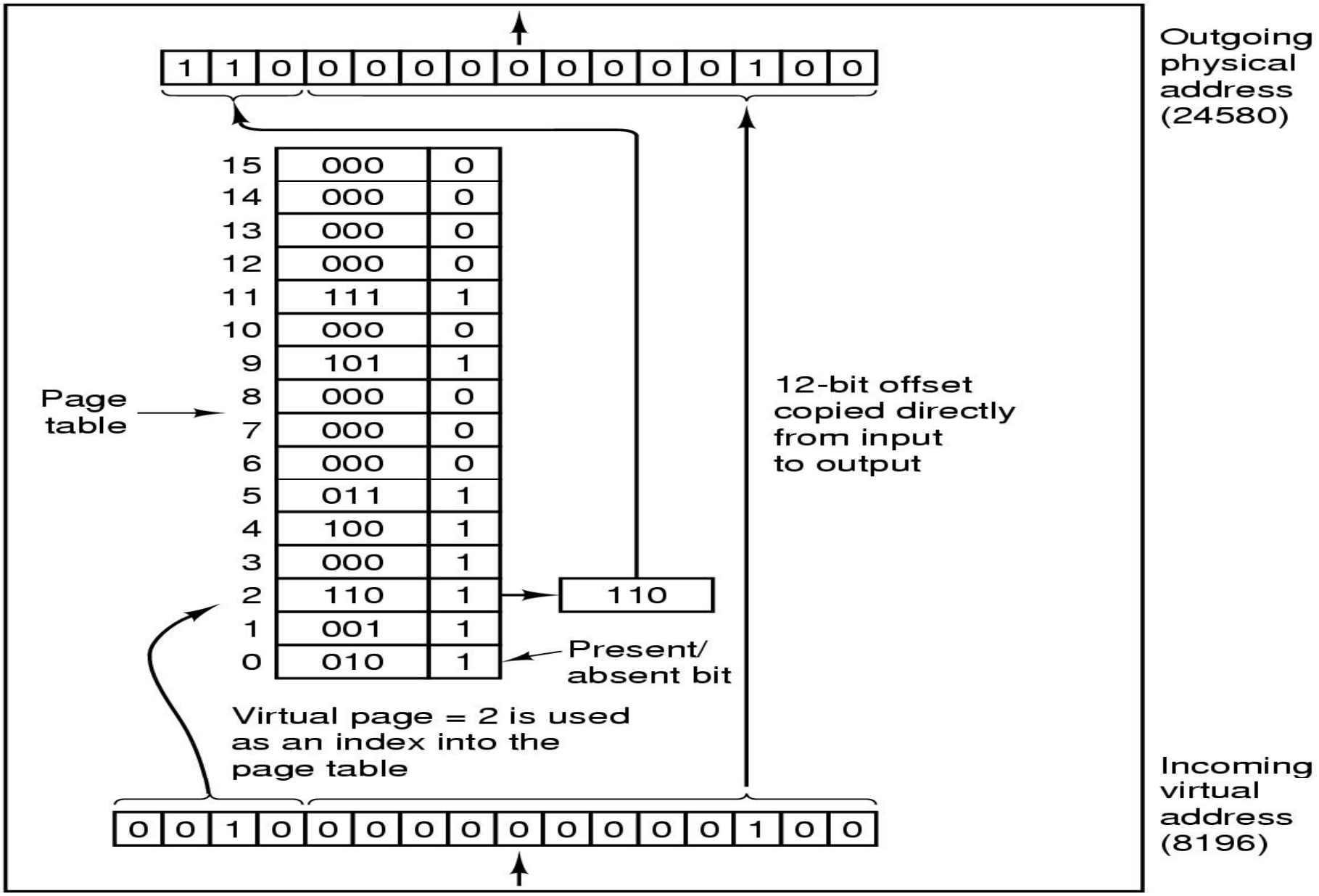
64KB of virtual address space

32KB of physical address space

Size of each page/frame = 4KB

=> 16 virtual pages
=> 8 page frames

Example (MMU Operation)



Page Fault & Page break

1. **Page Fault :**

   i) When the current page (during execution of a program) is not in main memory (still in secondary memory).

   ii) Present/absent bit tells whether page is in memory or not.

   0 => page fault

   1 => no page fault

2. **Page Break:**

   18 KB of virtual address space & page size is 4KB

   => no. of frames = 5

   2KB is wasted for the 5^{th} frame.

Demand Paging

- When ***page fault*** occurs (required page is not in main memory, Present/Absent bit is ‘0’), then the required page needs to be copied from secondary memory (HDD) and placed into the primary memory through ***page table***. This situation is known as ***demand paging***.

  => swapping-out & swapping-in of pages is referred as ***thrashing.***

  => The processor spends most of its time swapping pieces rather than executing user instructions.

- Demand paging is also used when a process first starts up.

- When a process is created, it has

  => A brand new page table with all Present/Absent bits off (0)

  => No pages in memory

Page Fault occurs
Implies Demand paging

Cntd…

- **Advantage:**
  - i) Supports time-sharing systems.
  - ii) Efficient memory utilization.
  - iii) Supports non-contiguous memory allocation.
  - iv) Easy to implement.

- **Disadvantage:**
  - i) Suffers from page-break.
  - ii) Difficult to maintain page table if virtual address space is large.
  - iii) Virtual to physical mapping is done on every memory reference => mapping must be fast

Exercise

1. 64-bit virtual address & 32-bit physical address. Each page is of 8KB.
   i) How many bits are required for the virtual page number & physical frame number?
   ii) How many page table entries are required for this system.

2. Virtual address space = 64KB
   Physical address space = 4KB
   Page size = 512B
   i) What are the physical address corresponding to the virtual addresses : 001111000001 , 1100111001110, 11110110000001, 111111111111
   ii) For which addresses (given above) there is a page fault.

Page No.	Frame No.	P/A bit
0	6	1
3	1	0
7	4	0
4	2	1
10	0	1
12	5	1
24	3	0
30	7	1

Translation Look-aside Buffer(TLB)

- In ***paging***, main memory is accessed twice, one for retrieving the frame no. and another for accessing the desired word in main memory. This increases latency.

The solution is to use ***TLB***.

- The TLB is a fast associative memory which is used to hold most recently used page table entries.
- Whenever CPU needs to access a particular page, the TLB is accessed first.

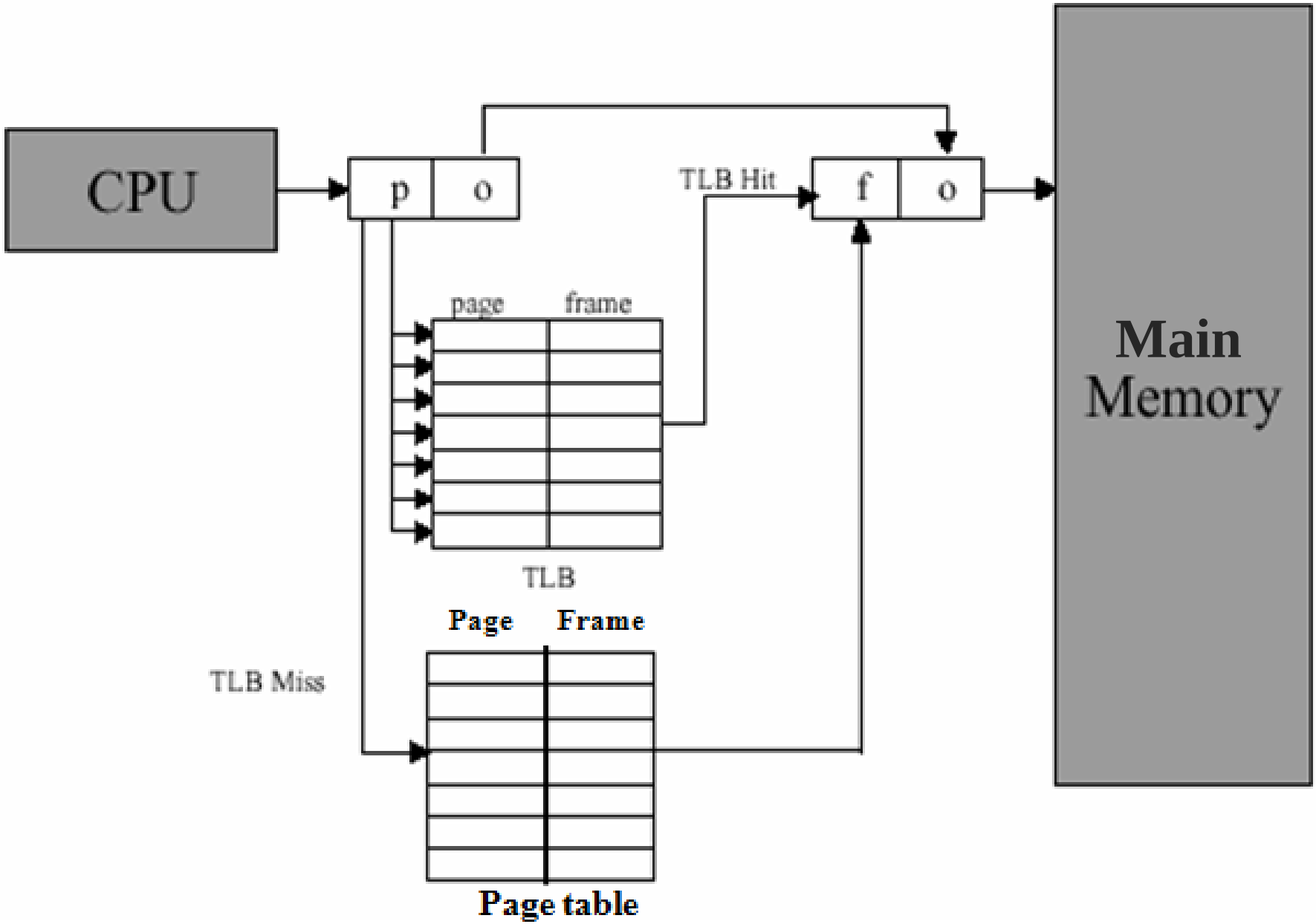
  => If the desired page table entry is found then there is a ***TLB hit***.

  => If there is a ***TLB miss*** then CPU searches the original page table in main memory.

- The effective access time

$$= (\mathbf{TLB_{hit}} \times \mathbf{Time_{hit}}) + (\mathbf{TLB_{miss}} \times \mathbf{Time_{miss}})$$

Cntd…


CPU
p
o
TLB Hit
f
o
Main Memory
page
frame
TLB
Page
Frame
TLB Miss
Page table

Exercise

TLB hit ratio = 80%

TLB access time = 20ns

Main memory access time =100ns

The effective access time = (TLB Hit Rate x Hit Time)
+ (TLB Miss Rate x Miss Time)

= (0.8 x (20 + 100)) + (0.2 x(20+100+100)

= 140 ns

2. Segmentation

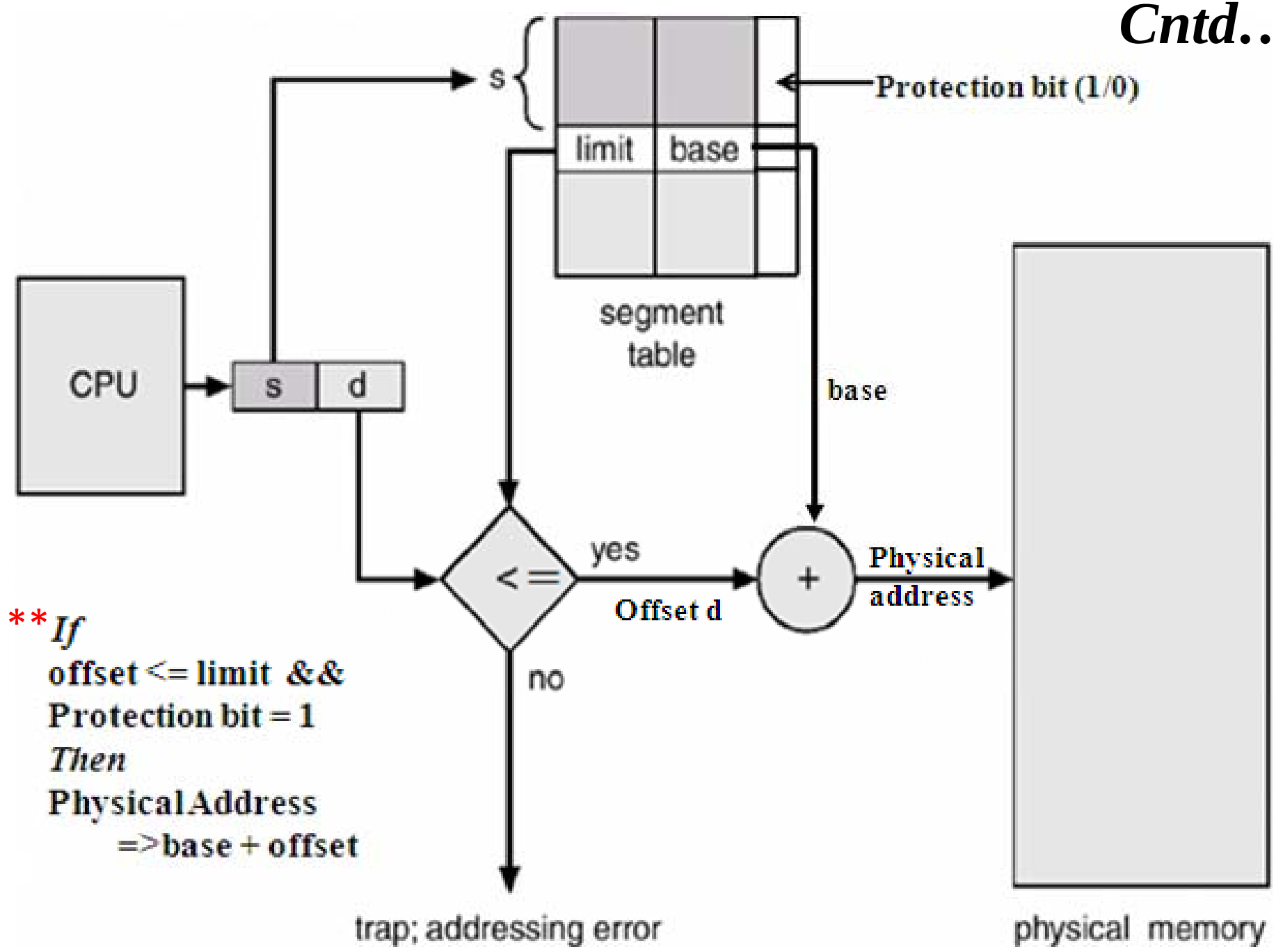
- Divide virtual address space into separate logical segments; each is part of physical memory.
- Segments are not necessarily be of equal size. The size is dependent on the virtual address space.
- For a particular process there is an unique segment.
- Each segment allocates contiguous memory.
- Segmentation defines logical grouping of instructions, such as subroutines, array or data area.
- Virtual address: ***<segment-number(s), offset(d)>***
- Segment table maps segment number to segment information

  Base: starting address of the segment in physical memory

  Limit: length of the segment

  Additional metadata includes ***protection bits***

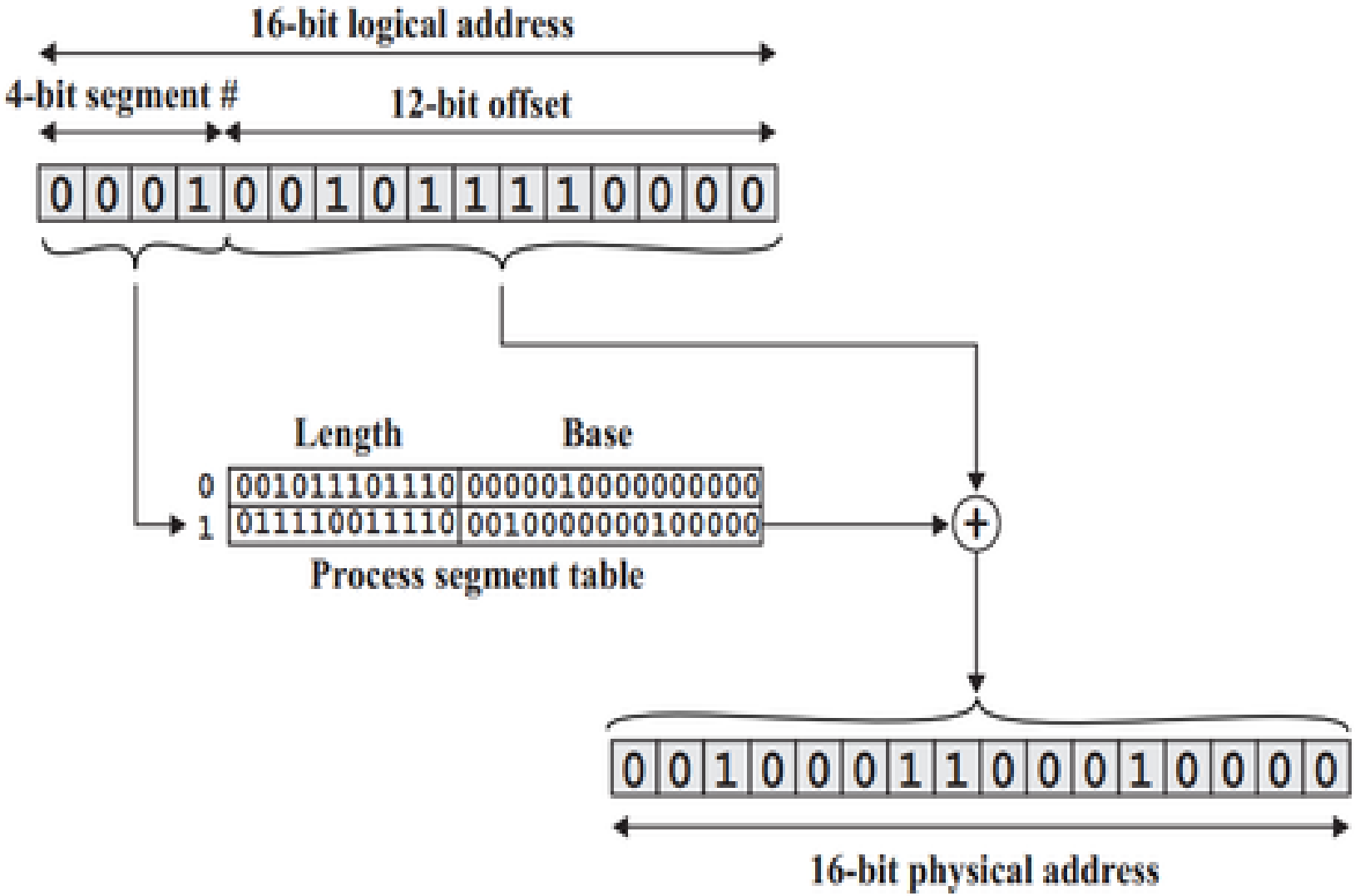
Cntd…



** *If*
offset <= limit &&
Protection bit = 1
Then
PhysicalAddress
=>base + offset

Example

4 bits for representing segment no.
=> Segment table can hold 16 entries



Cntd…

Advantage:

1. Segment sharing

2. Easier to relocate segment than entire program

3. Avoids allocating unused memory

4. Flexible protection

   - Permission checking using ***protection bit***

5. Efficient translation

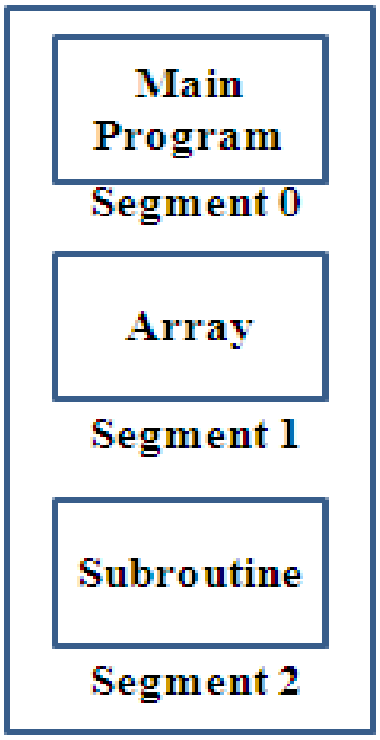
   •Segment table small => fits in MMU

Disadvantage:

1. Segments have variable lengths => how to fit?

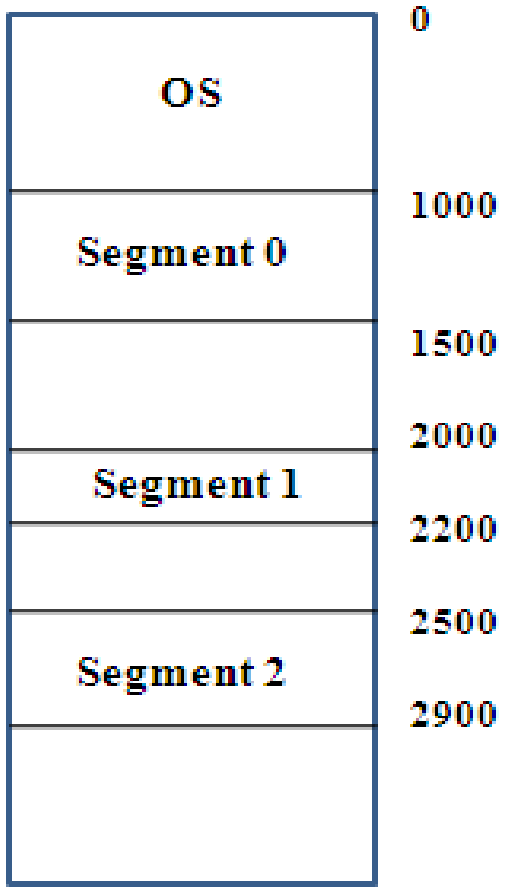
2. Segments can be large => fragmentation

Exercise



Logical Address Space

Seg no.	Limit	Base	P. bit
0	500	1000	1
1	200	2000	1
2	400	2500	1



Physical Address Space

What will be the corresponding physical addresses of the given logical addresses ***(0,20) , (1,150), (2,450)***


Q+A

1. Size of the virtual memory is equivalent to the size of ……..

   a) Primary memory b) Secondary memory

   c) Cache memory d) none of these

2. Virtual address is generated by ……..

   a) CPU b) OS c) Main memory d) Auxiliary memory

3. Who is responsible for mapping of logical address into physical address ?

   a) CPU b) OS c) Main memory d) Auxiliary memory

4. Paging follows ……. memory allocation.

   a) Contiguous b) Non-contiguous c) both d) none

5. Assume that there are 5 different program running in a virtual memory system. How many ***page table*** should be maintained by OS?

   a) 1 b) 2 c) 5 d) multiple of 5

6. Page fault occurs when ……..

   a) the page is not in main memory

   b) Present/ Absent bit is ‘0’

   c) both

   d) none

7. Page break occurs when ……..

   a) process size exceeds page size b) process size less than page size

   c) process size equal to page size d) none of these

8. There is always a page fault when the process ……..

   a) starts 1st time b) at the middle of execution

   c) at the end of execution d) all of these

9. Segmentation follows ……memory allocation.

   a) contiguous b) non-contiguous c) both d) none

Reference:

1. Computer Organization & architecture – T. K. Ghosh
2. Advanced Operating System - Tanenbaum

Thank You