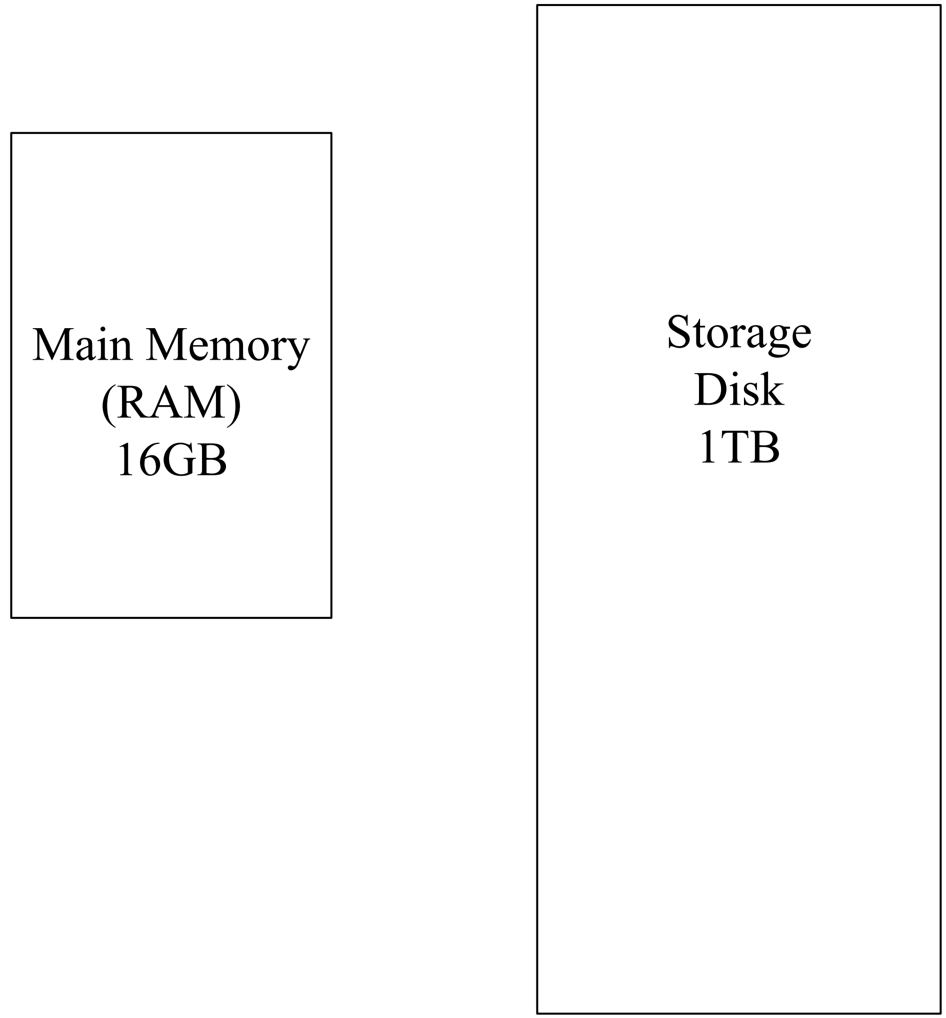
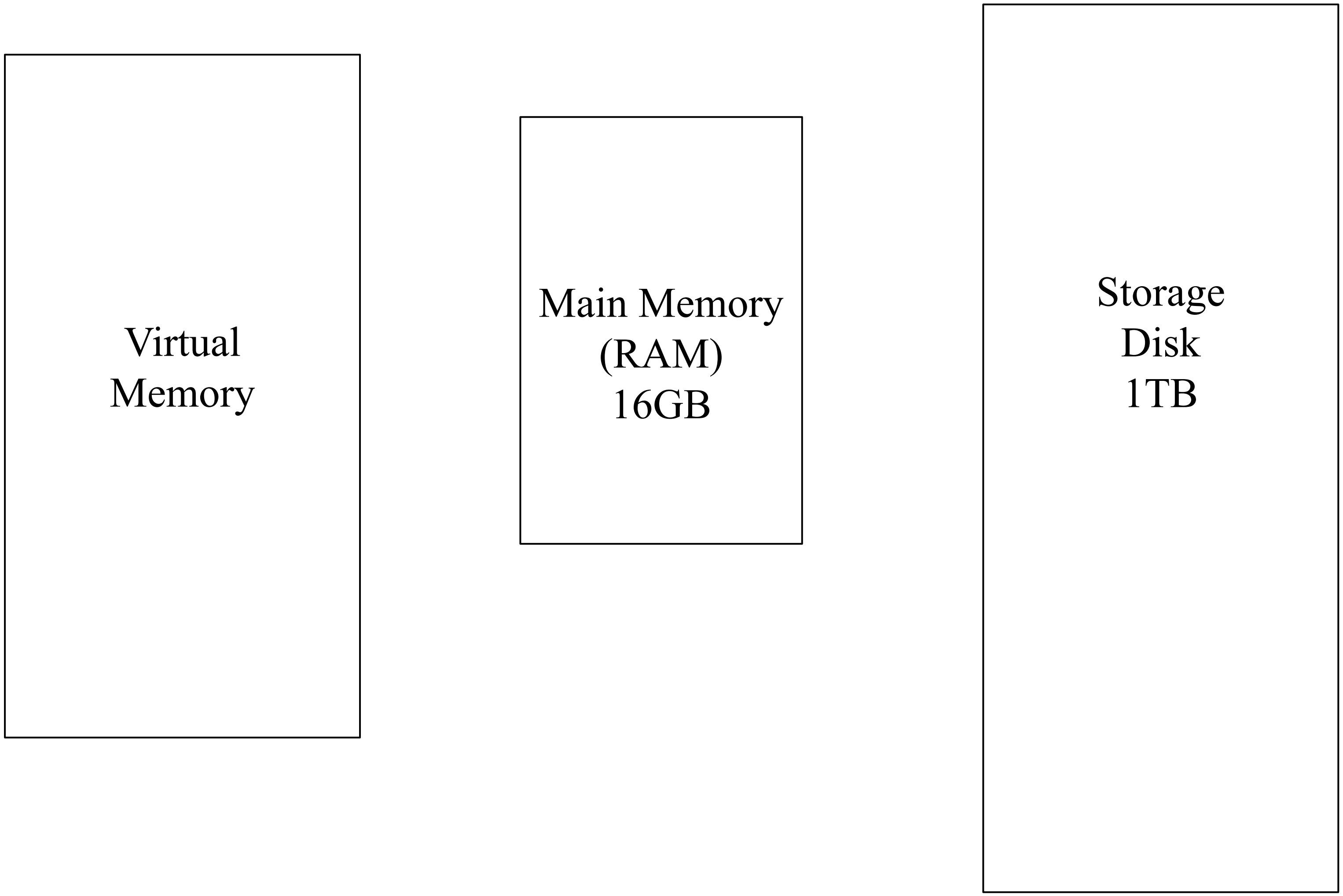
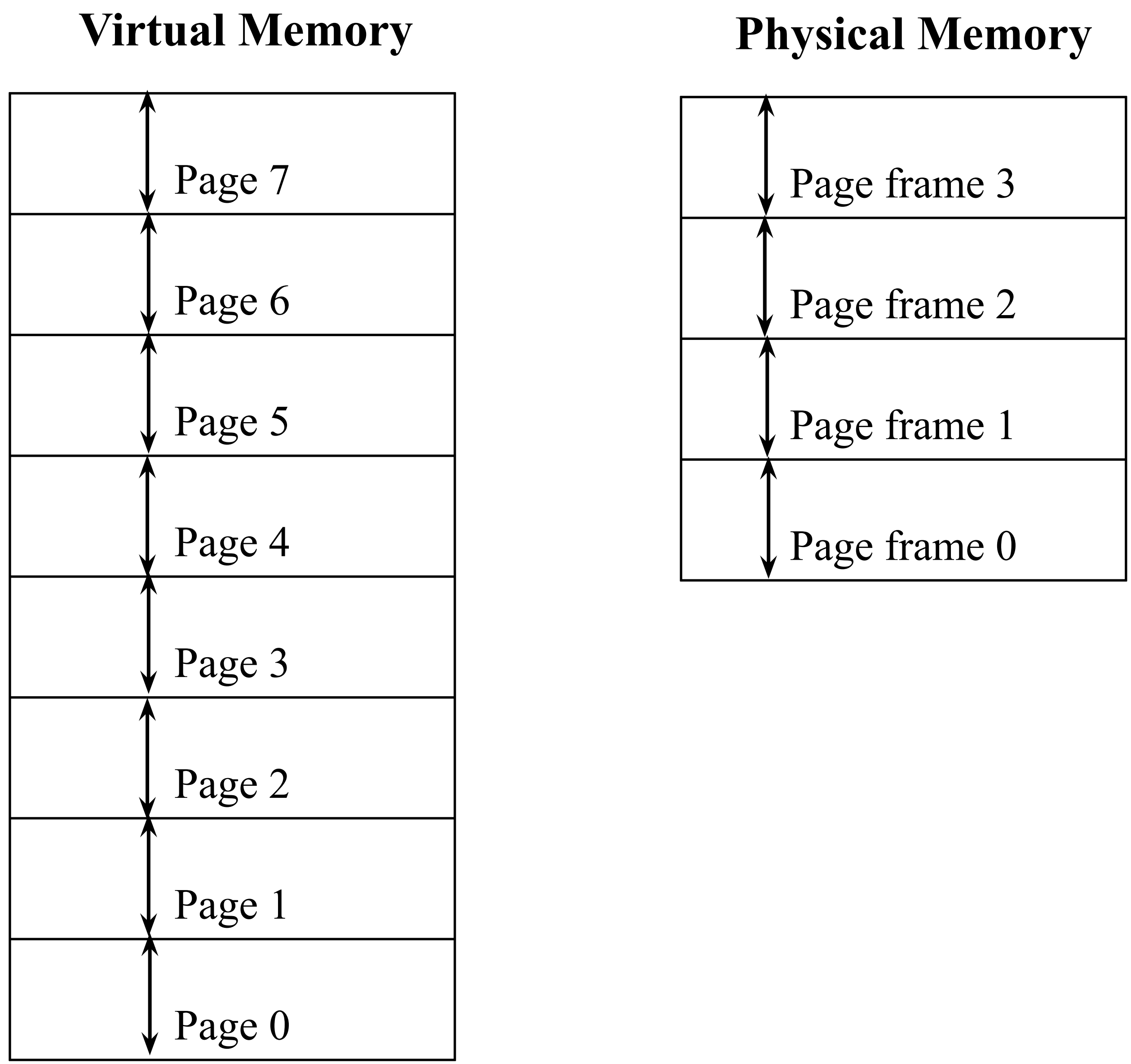
Paging


Main Memory
(RAM)
16GB
Storage
Disk
1TB


Virtual Memory
Main Memory (RAM) 16GB
Storage Disk 1TB


Virtual Memory
Page 7
Page 6
Page 5
Page 4
Page 3
Page 2
Page 1
Page 0
Physical Memory
Page frame 3
Page frame 2
Page frame 1
Page frame 0

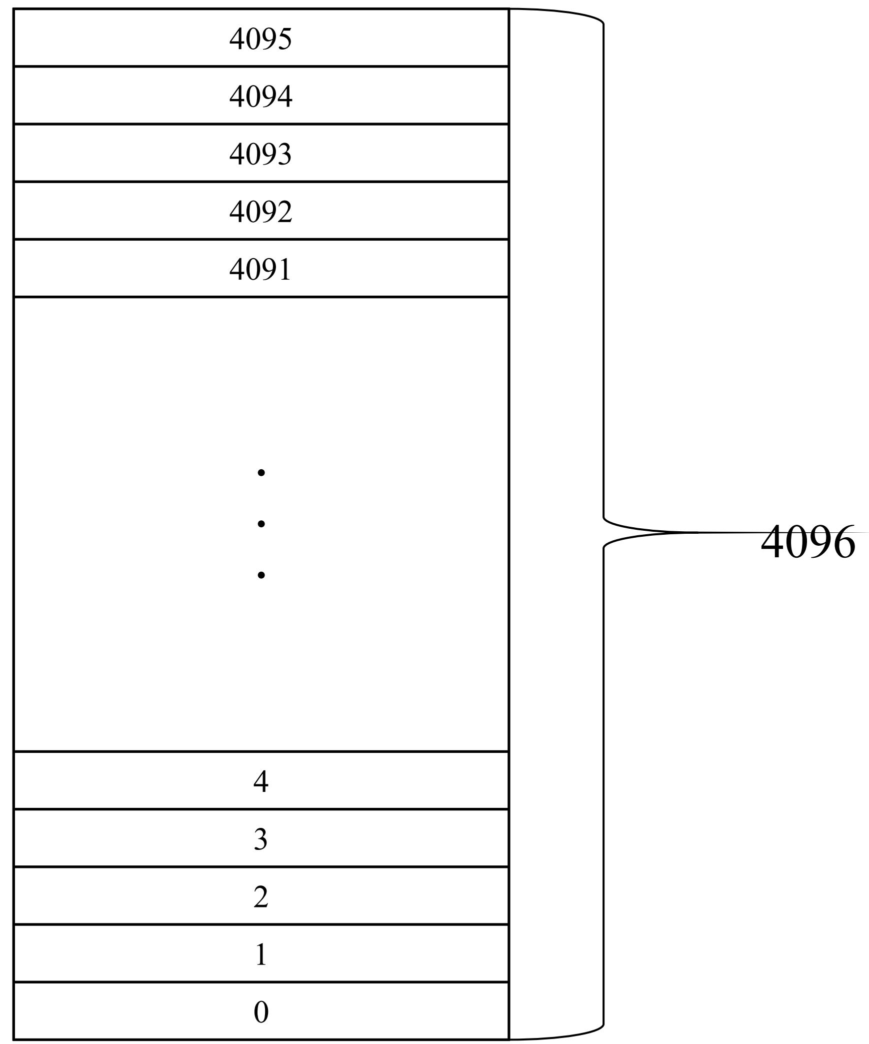
Virtual Memory (64KB)

Page 15 (4KB)
Page 14 (4KB)
Page 13 (4KB)
Page 12 (4KB)
Page 11 (4KB)
Page 10 (4KB)
Page 9 (4KB)
Page 8 (4KB)
Page 7 (4KB)
Page 6 (4KB)
Page 5 (4KB)
Page 4 (4KB)
Page 3 (4KB)
Page 2 (4KB)
Page 1 (4KB)
Page 0 (4KB)

Physical Memory (32KB)

Page frame 7 (4KB)
Page frame 6 (4KB)
Page frame 5 (4KB)
Page frame 4 (4KB)
Page frame 3 (4KB)
Page frame 2 (4KB)
Page frame 1 (4KB)
Page frame 0 (4KB)

Page / Page frame (4KB)



- Program generates virtual address which points in Virtual Memory.

- MMU maps this Virtual address into physical address.

- Let's see how MMU works?

Page Table

15		0
14		0
13		0
12		0
11	111	1
10		0
9	101	1
8		0
7		0
6		0
5	011	1
4	100	1
3	000	1
2	110	1
1	001	1
0	010	1

Page No.

Present / Absent bit

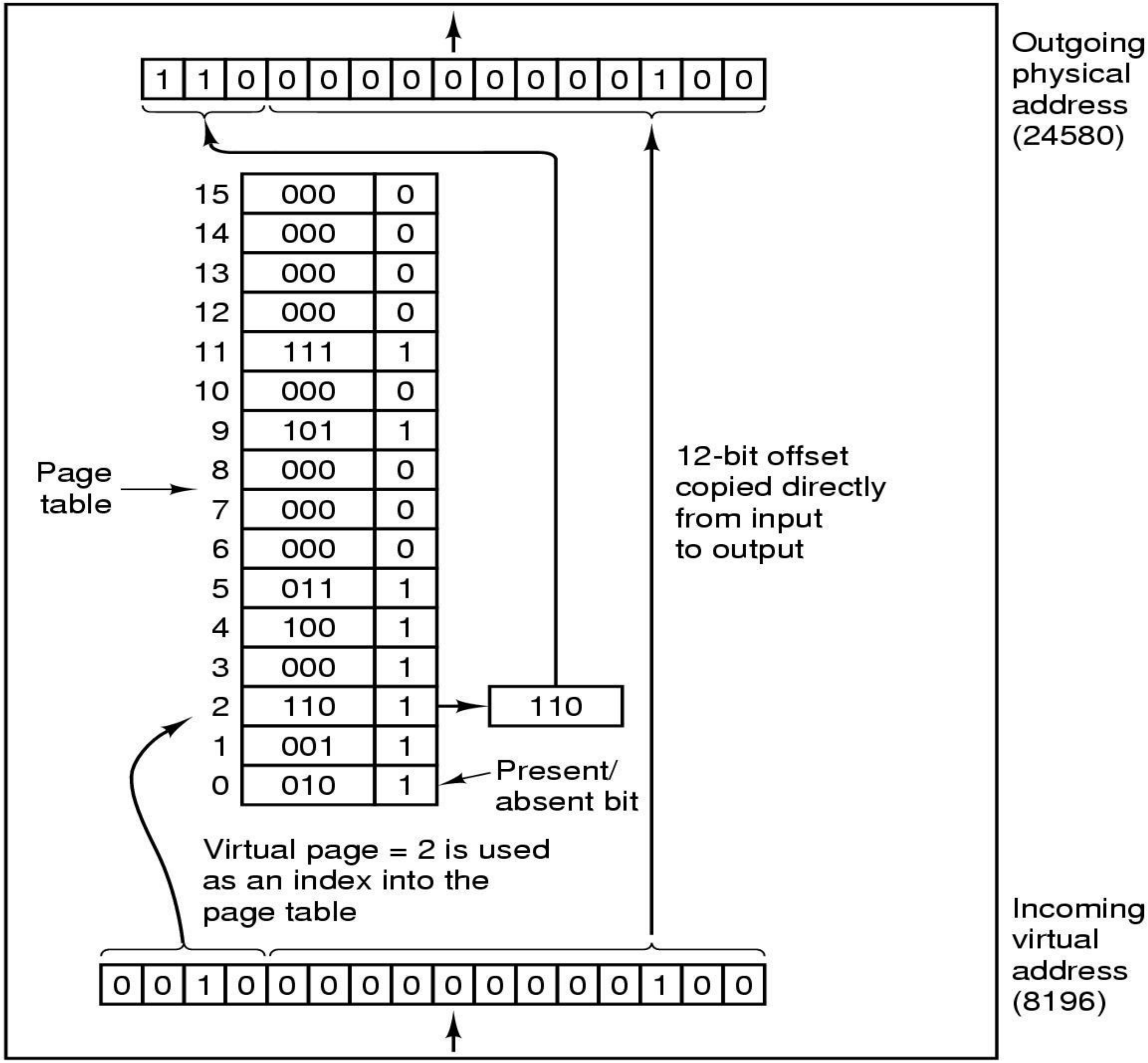
Page frame No.

- MMU splits Virtual address into 2 part;
    - High order bits as page no.
    - Low order bits as an offset

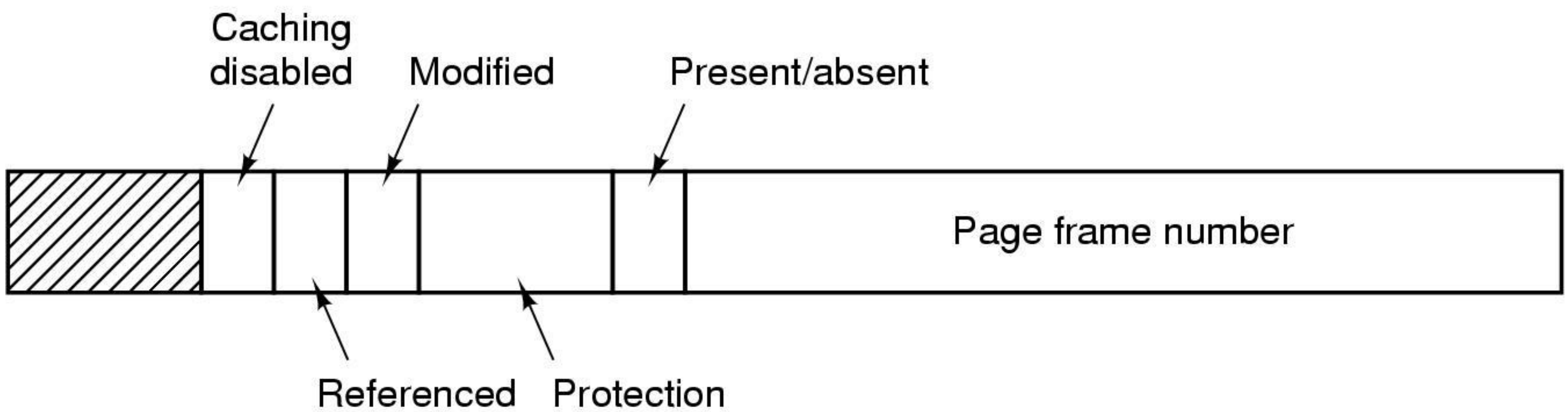
- Here we consider Virtual Memory of 64KB
  ie 16 bits virtual address (2^{16} = 65536 Bytes).
    - We have 16 pages of 4KB each.
    - Upper 4-bits of VA used for page no (as $2^4 = 16$) and
    - Lower 12-bits as an offset ie location of data byte in page frame.
      2^{12} = 4096 (4KB)

- This virtual address split depends on number of pages and page size.

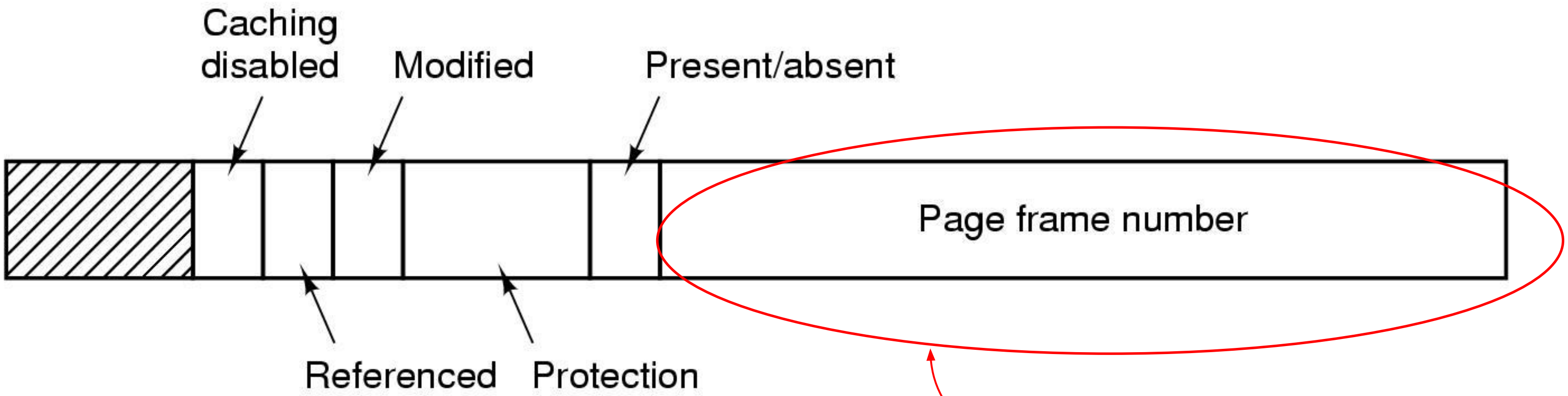
MMU operation



Structure of Page Table Entry



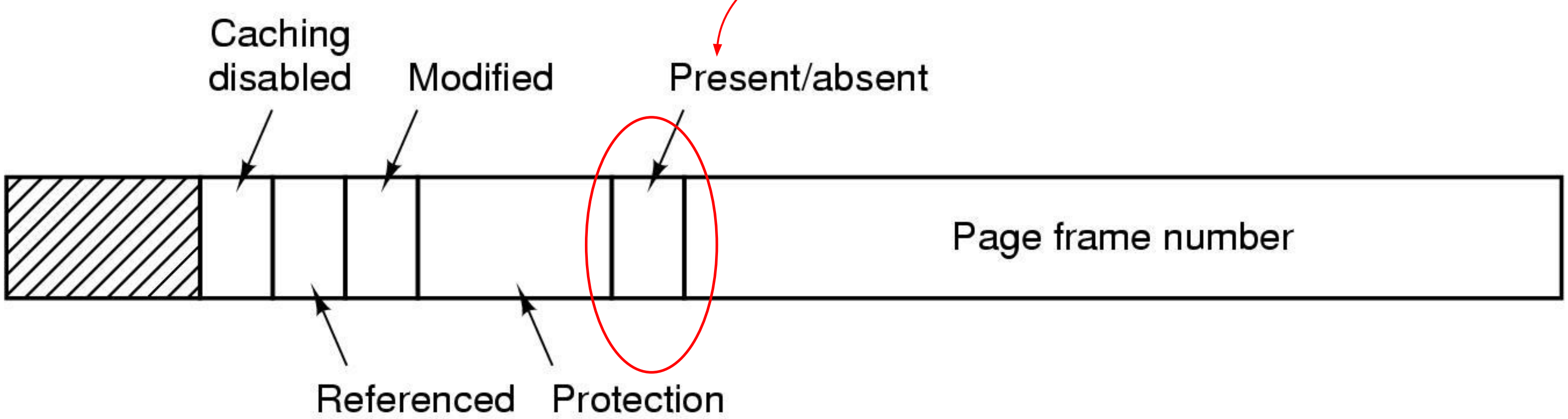
Structure of Page Table Entry



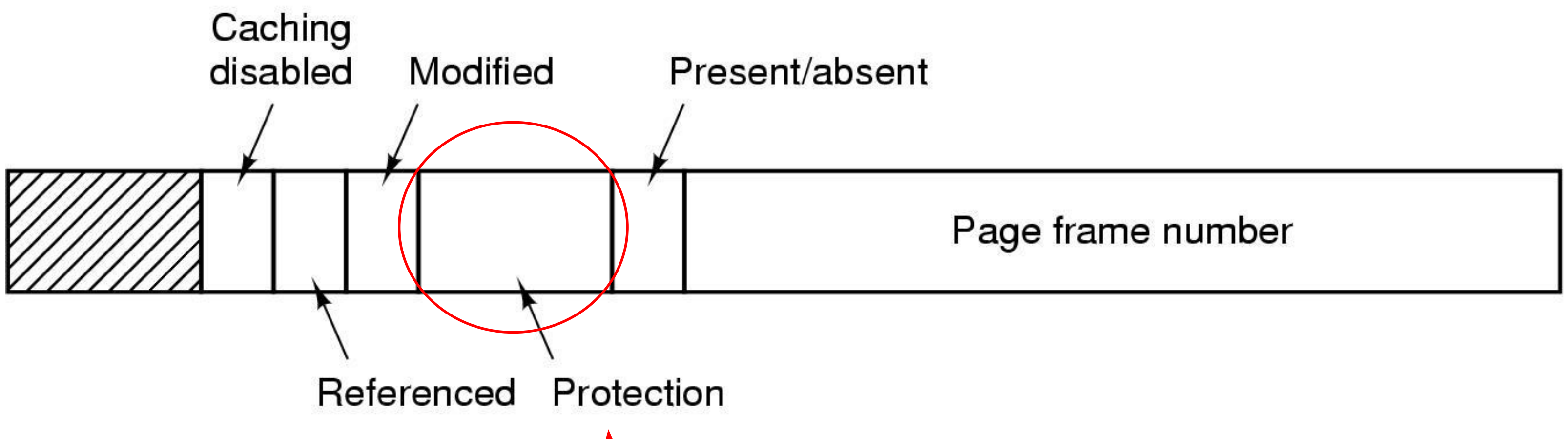
Most important field that gives which frame is mapped into the page.

Structure of Page Table Entry

If this bit is 1 that means page frame is in memory. Otherwise page fault occurs.



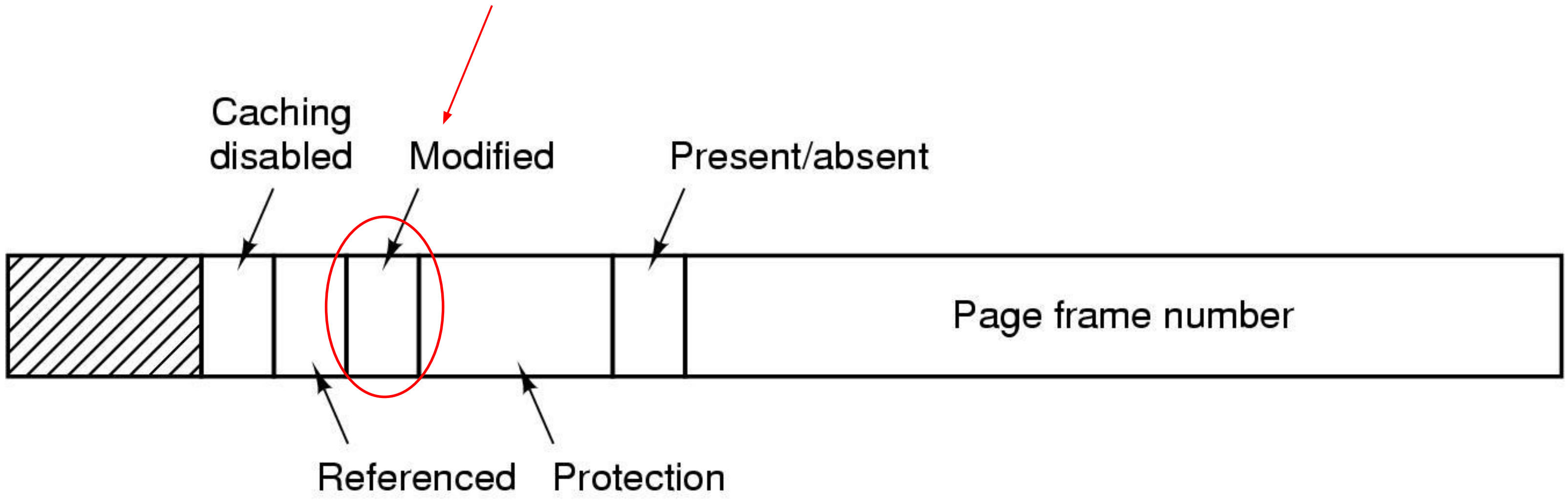
Structure of Page Table Entry



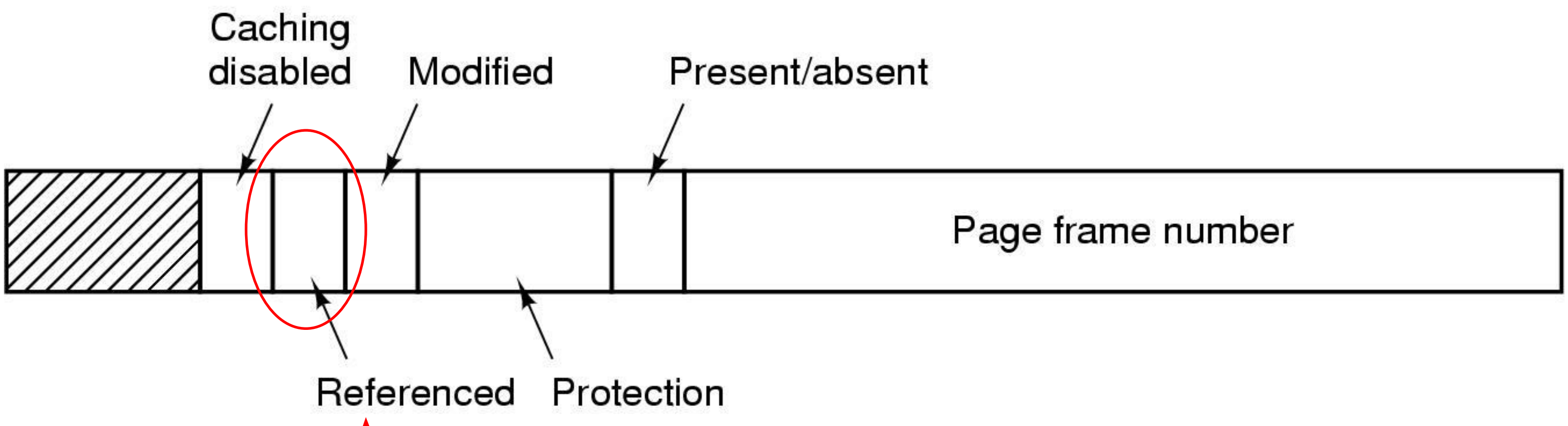
Tells what kind of access are permitted ie read/ write.

Structure of Page Table Entry

When write operation takes place on page then H/W set this bit.
1- mean page has been modified ie dirty, it must be written back to disk. Otherwise clean, not necessary to write. Also called as *dirty bit*.



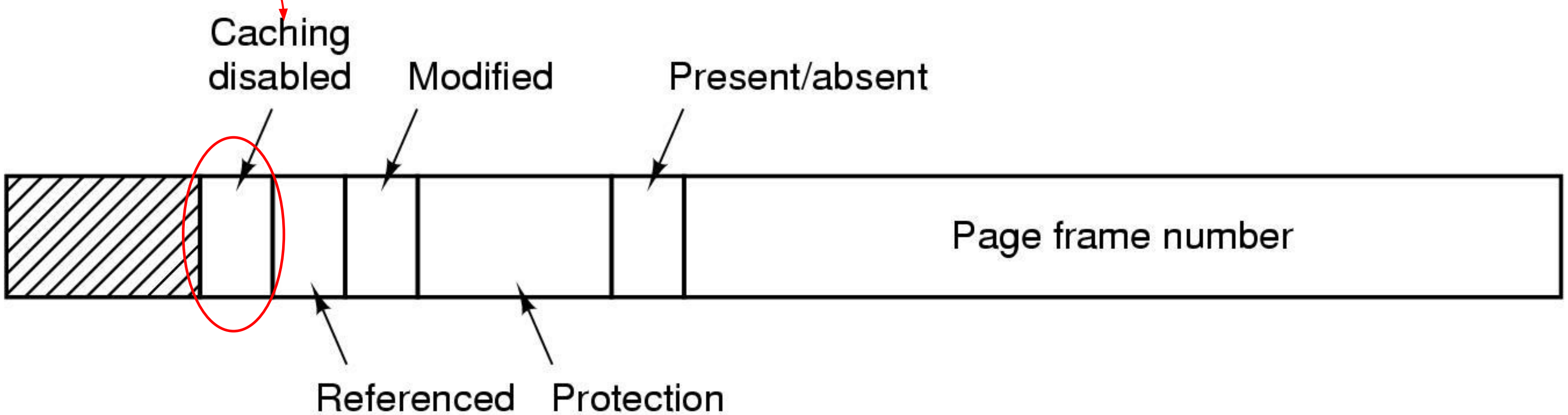
Structure of Page Table Entry



This bit set whenever a page is referenced for reading /writing. This bit used by OS for page replacement.

Structure of Page Table Entry

This bit allows caching to be disabled.



Structure of the Page Table

Some most common techniques for structuring the page table.

- Hierarchical Paging
- Hashed Page Tables
- Inverted Page Tables

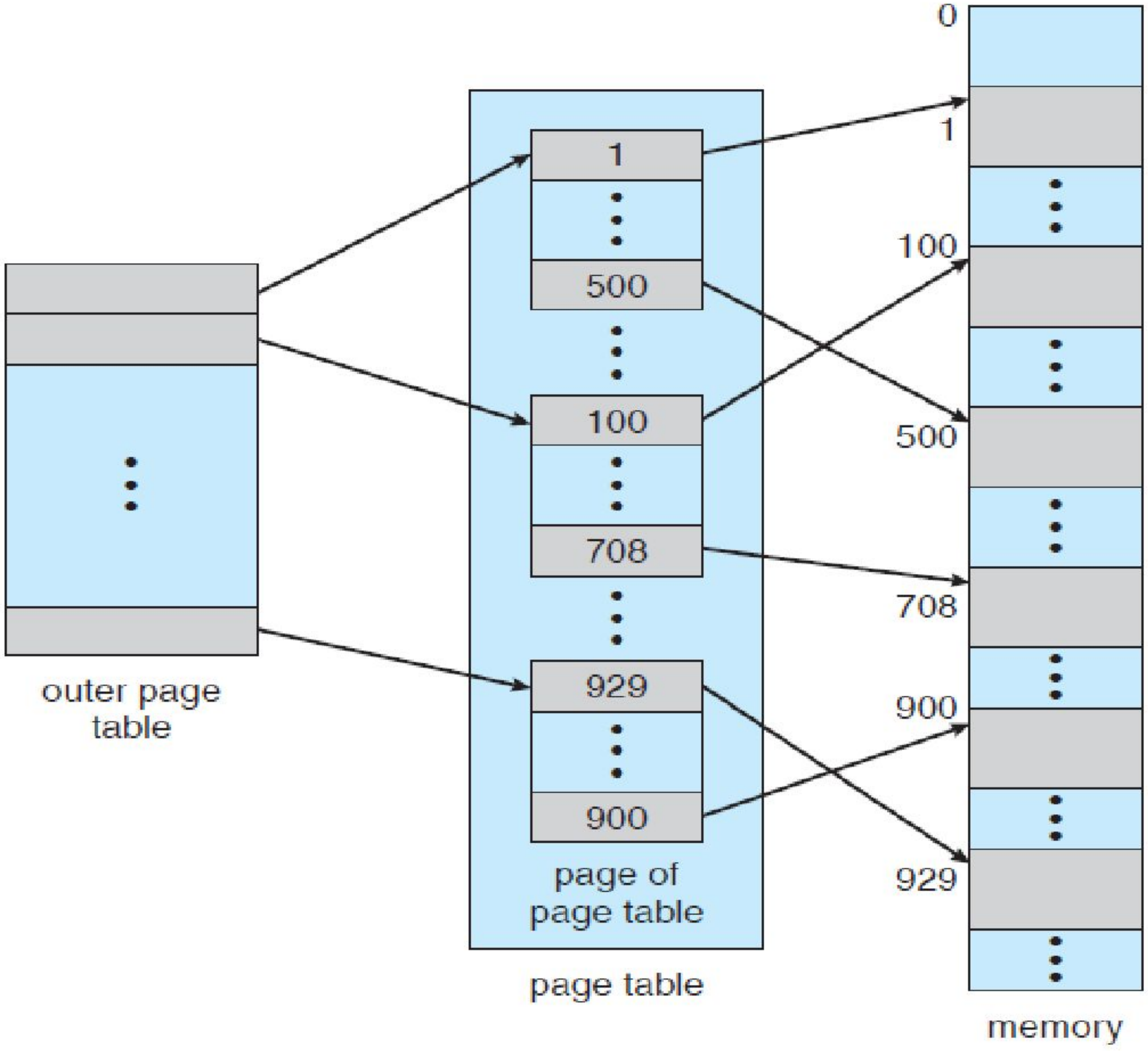
Structure of the Page Table

Hierarchical Paging:

- If logical address space is large, then the page table itself becomes excessively large.
- One simple solution to this problem is to divide the page table into smaller pieces.
- One way is to use a two-level paging algorithm, in which the page table itself is also paged.

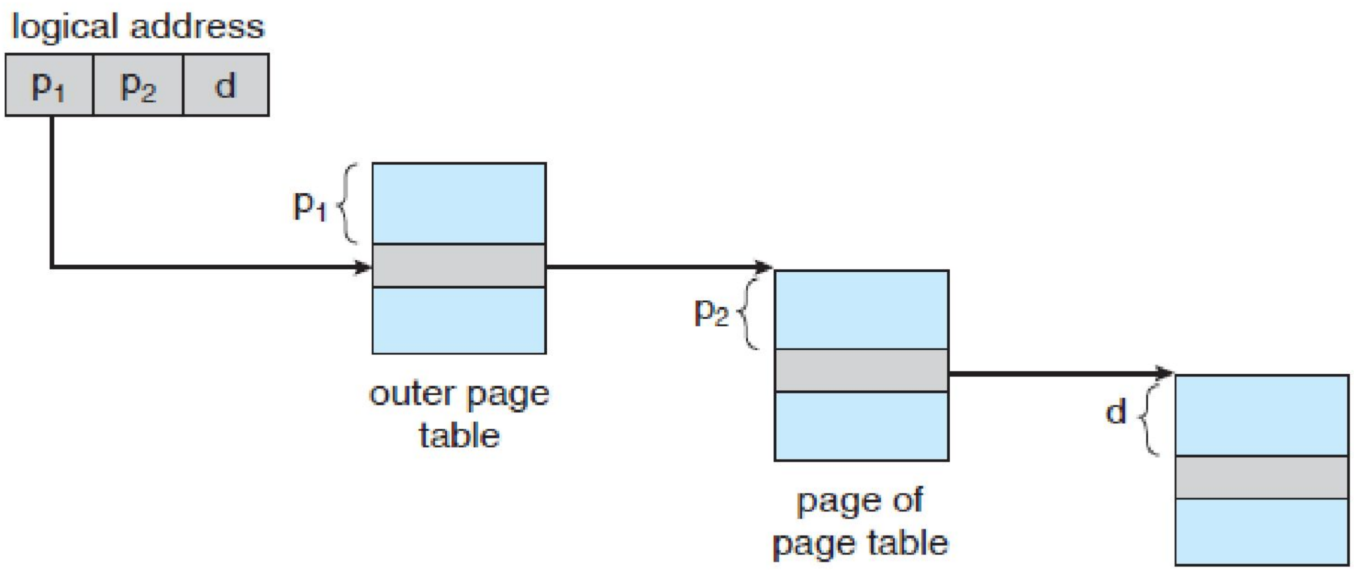
Structure of the Page Table

Hierarchical Paging:



Structure of the Page Table

Hierarchical Paging:



Structure of the Page Table

Hashed Page Tables:

- A common approach for handling address spaces larger than 32 bits is to use a **hashed page table.** with the hash value being the virtual page number.
- Each entry in the hash table contains a linked list of elements that hash to the same location.
- Each element consists of three fields:
  1. the virtual page number,
  2. the value of the mapped page frame, and
  3. a pointer to the next element in the linked list.

Structure of the Page Table

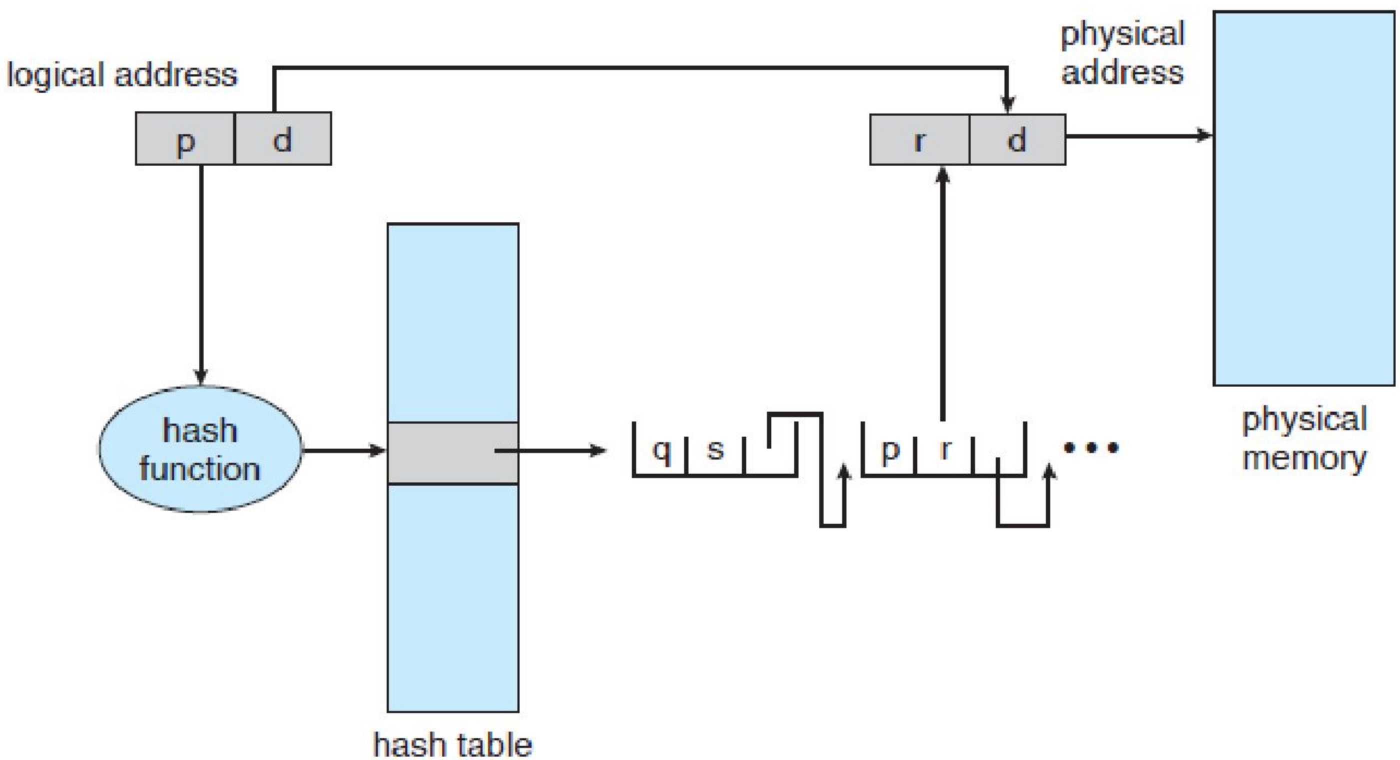
Hashed Page Tables:

The algorithm works as follows:

- The virtual page number in the virtual address is hashed into the hash table.
- The virtual page number is compared with field 1 in the first element in the linked list.
- If there is a match, the corresponding page frame is used to form the desired physical address.
- If there is no match, subsequent entries in the linked list are searched for a matching virtual page number.

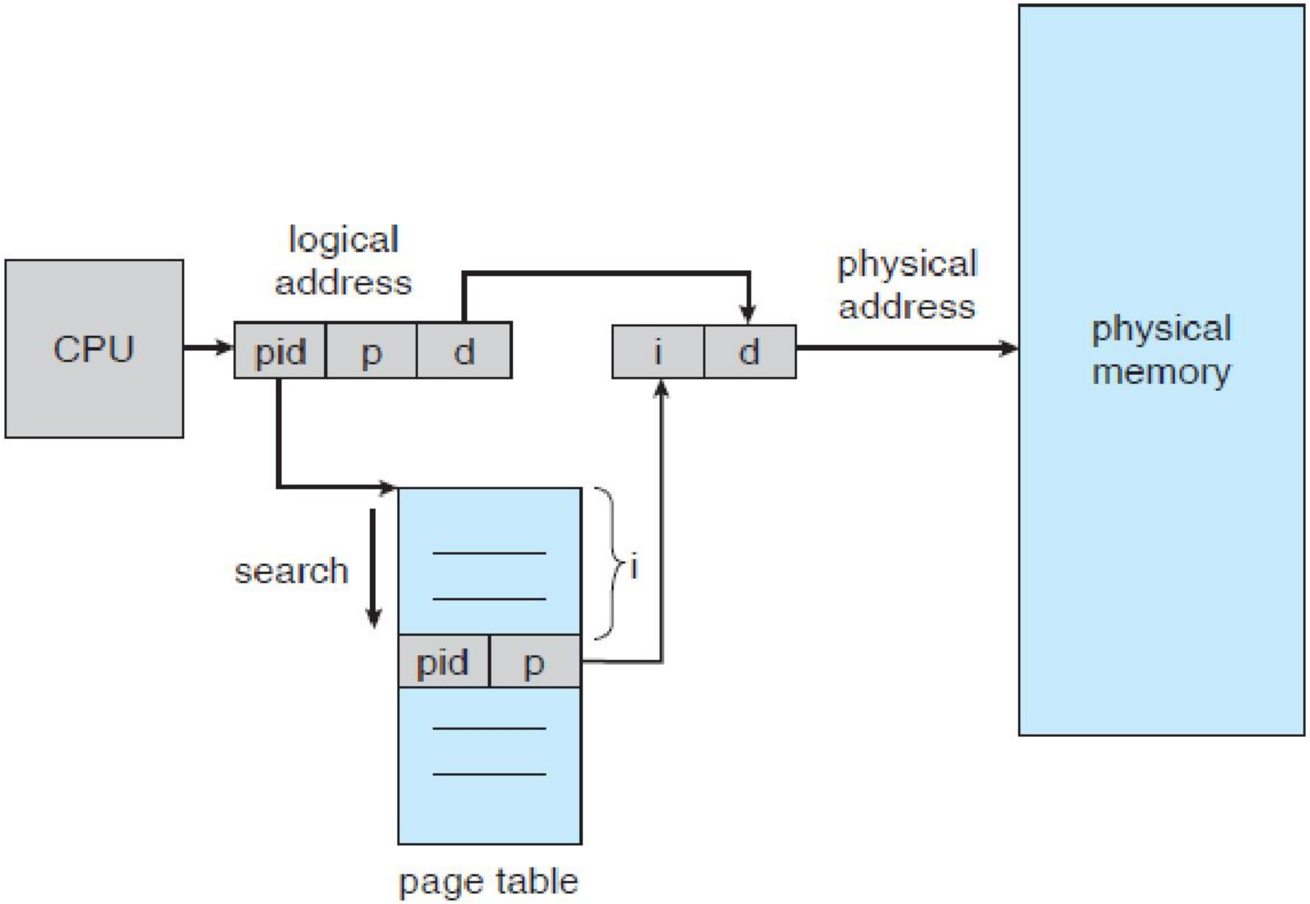
Structure of the Page Table

Hashed Page Tables:



Structure of the Page Table

Inverted Page Tables:



Thank You