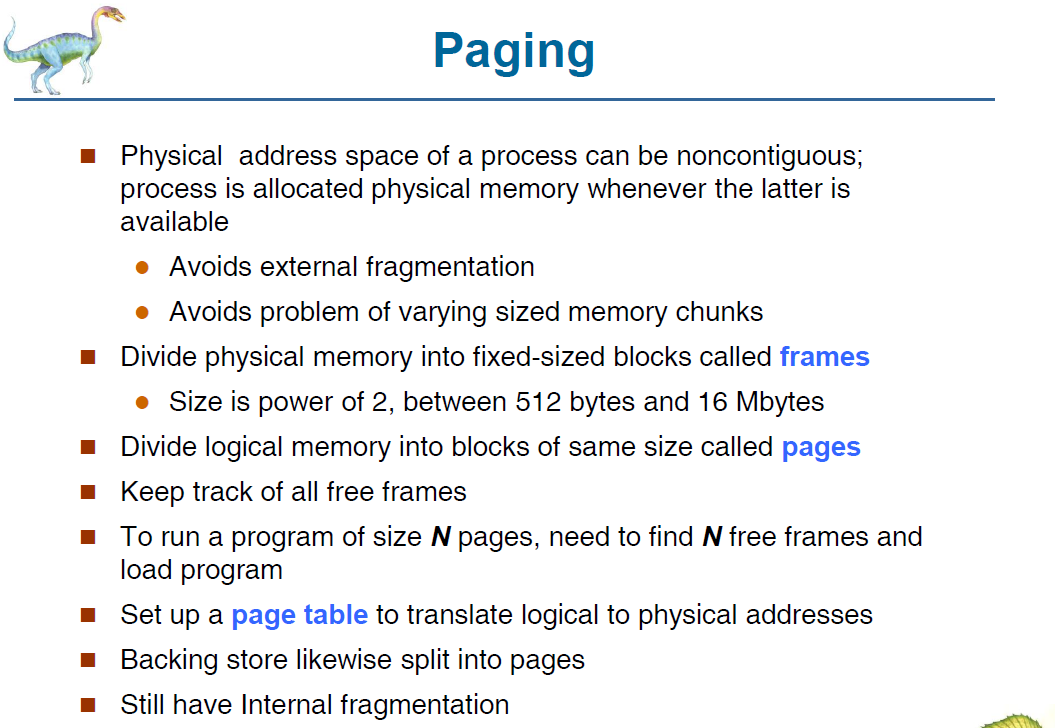
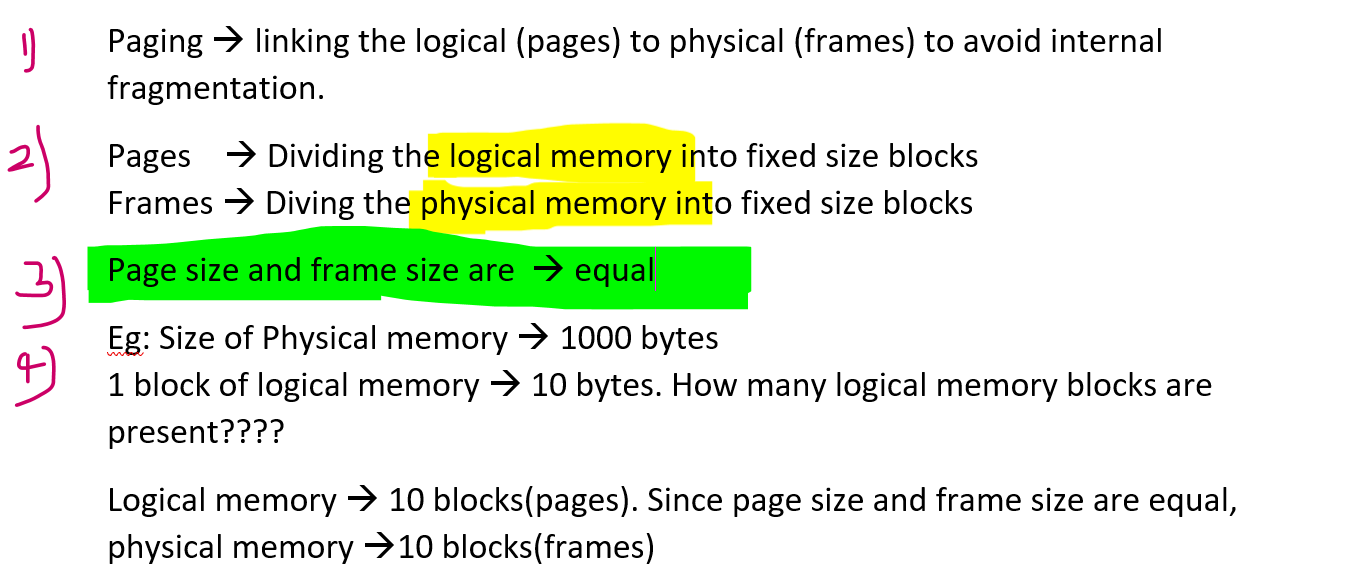
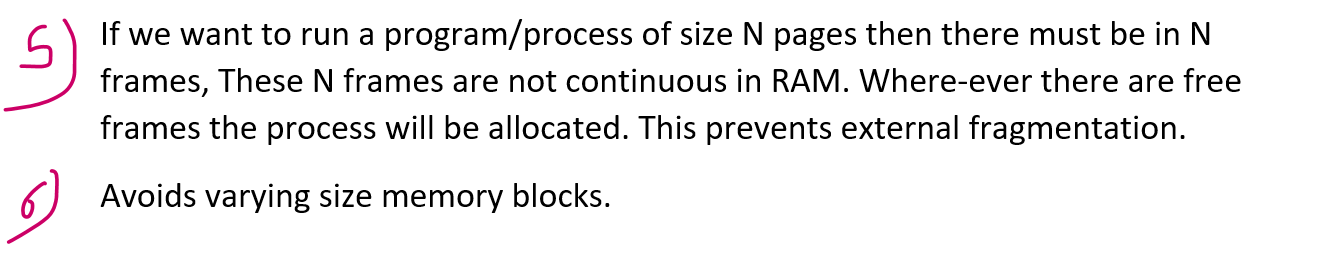
**Non-contiguous Memory Allocation**







CPU always generates logical address.



**Will the base register and limit register is enough to translate the logical address to physical address??? 🡪 NO**In continuous memory allocation, base and limit registers are used since we can easily find a process. Base + limit address



Non-contiguous memory allocation 🡪 Page table



Protection:

Contiguous memory allocation 🡪 Base and limit register  
Non-contiguous memory allocation 🡪 Page table

Page table (logical to physical address conversion)   
Let say there are 4 pages  
P0 🡪 base address of F15  
P1 🡪 base address of F12  
P2 🡪 base address of F59  
P3 🡪 base address of F45

Paging leads to internal fragmentation  
Non-contiguous memory allocation 🡪 External fragmentation

Eg: Suppose if there are 1000 bytes page, and the   
process requires 🡪999 bytes (1 byte will be wasted) [ best-case ]  
process requires 🡪990 bytes (100 bytes will be wasted) [worst-case]

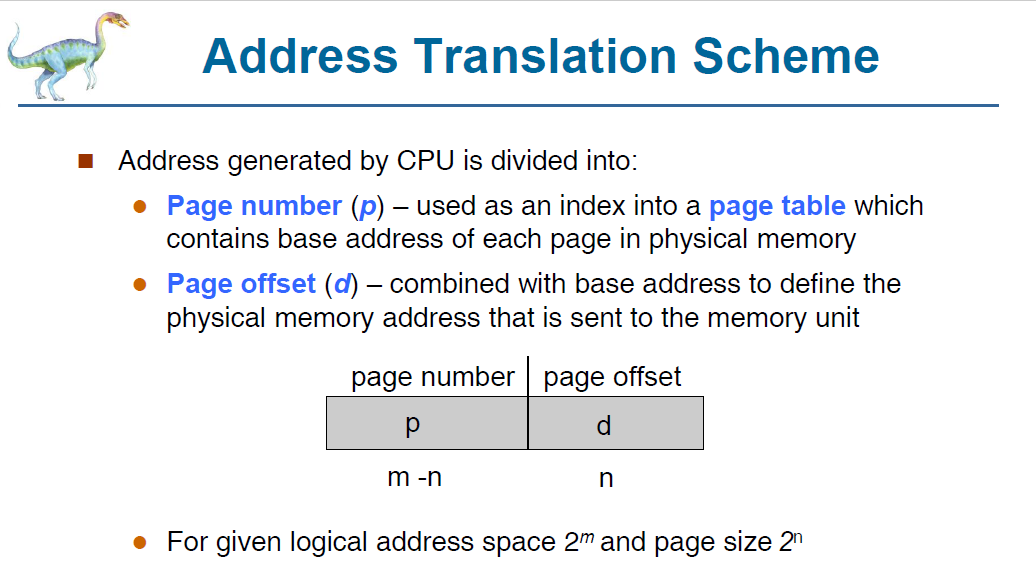


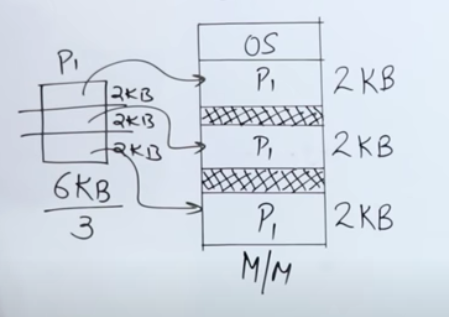
Since the pages and frame block are of fixed size.

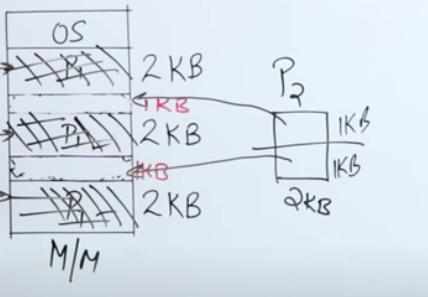
Internal fragmentation 🡪 Allocated memory might slightly larger than the requested memory. The size difference is internal to the partition, but not being be used.

In External fragmentation, empty holes can be pushed together as a big hole but in internal fragmentation It is Within the process, so waste spaces cannot be reused again.

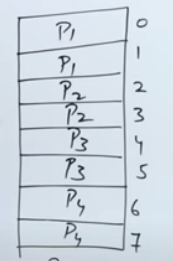
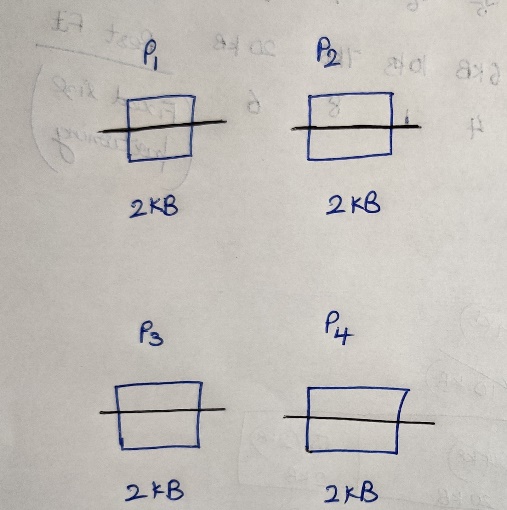
Size(i.e page size and frame size) should always be in the power of 2





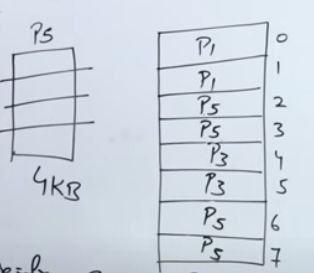
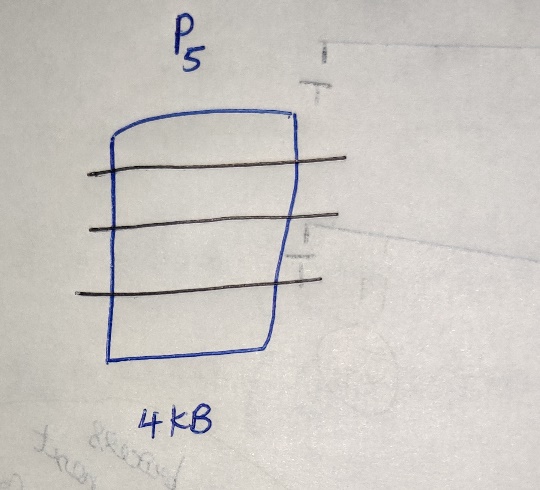


After P1 is in the RAM, P2 is coming into the RAM.

Page size = frame size = 1KB.

After P2 and P4 executes and leaves the RAM.



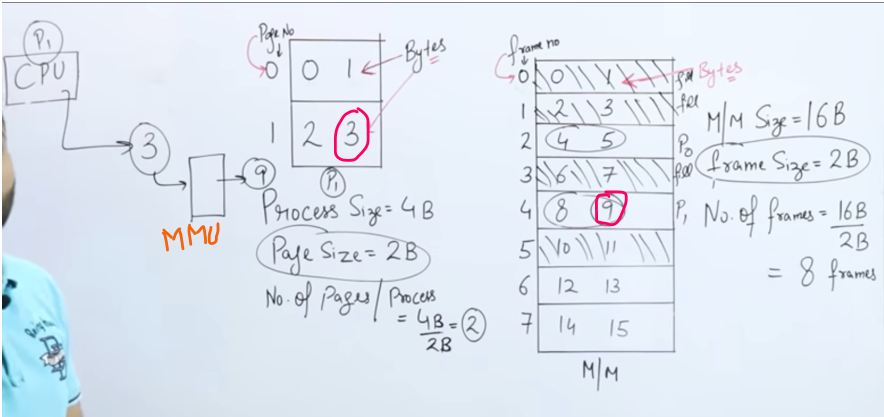
After P2,P4 leaves the RAM, P5 enters into RAM.

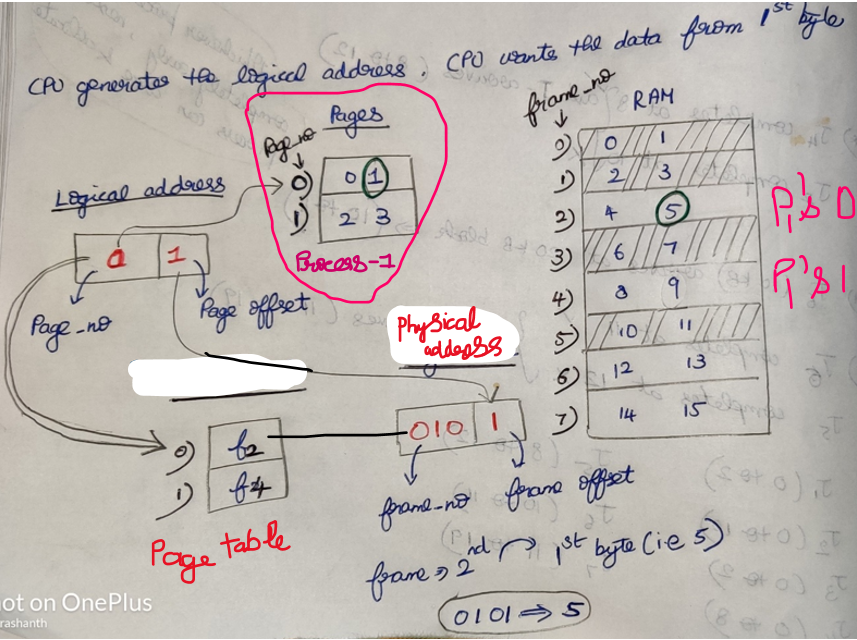
**Approach-1**



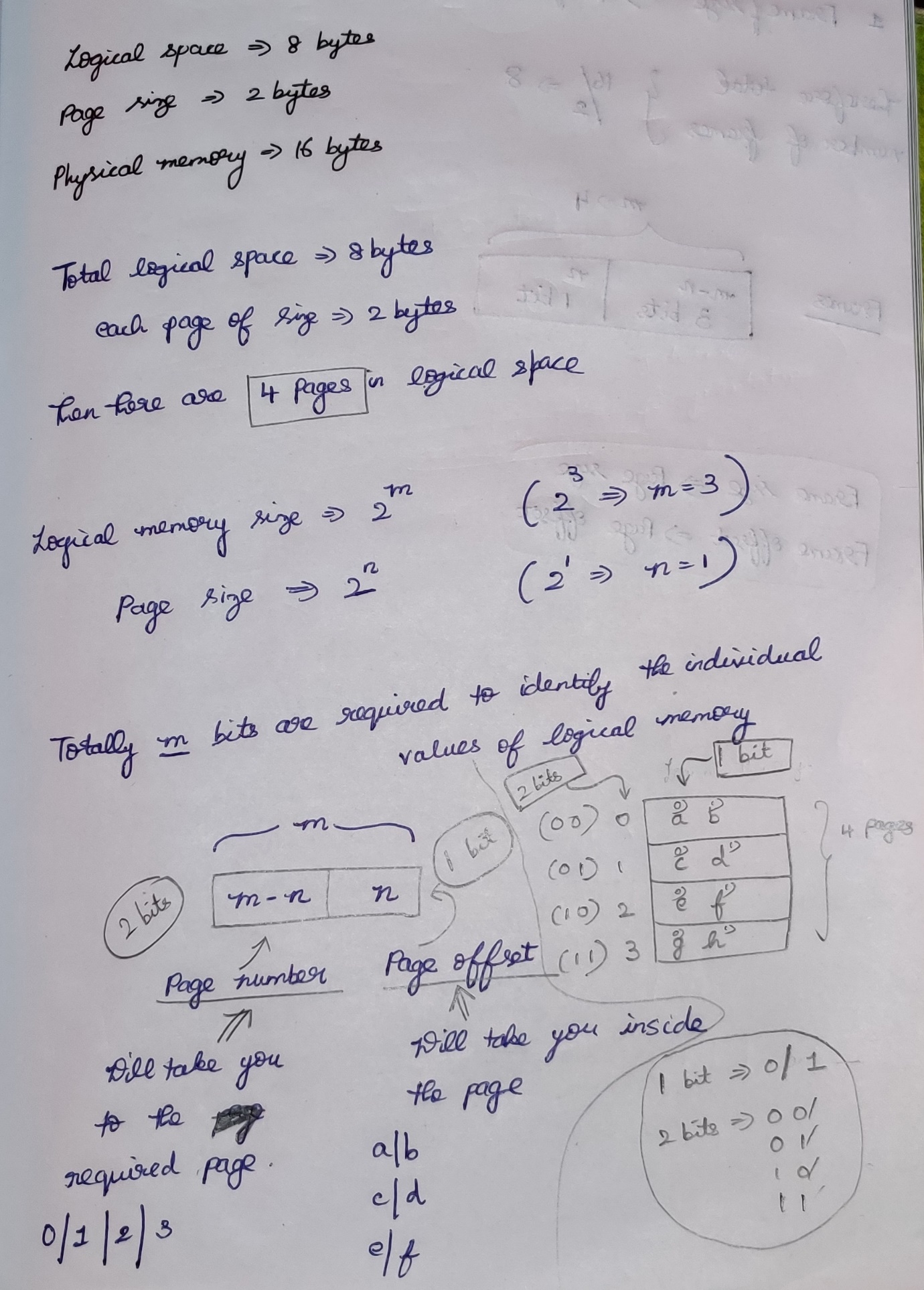
CPU generates a logical address of a process which is in secondary memory.  
How this logical address can be converted into physical address, so as to find in the main-memory????

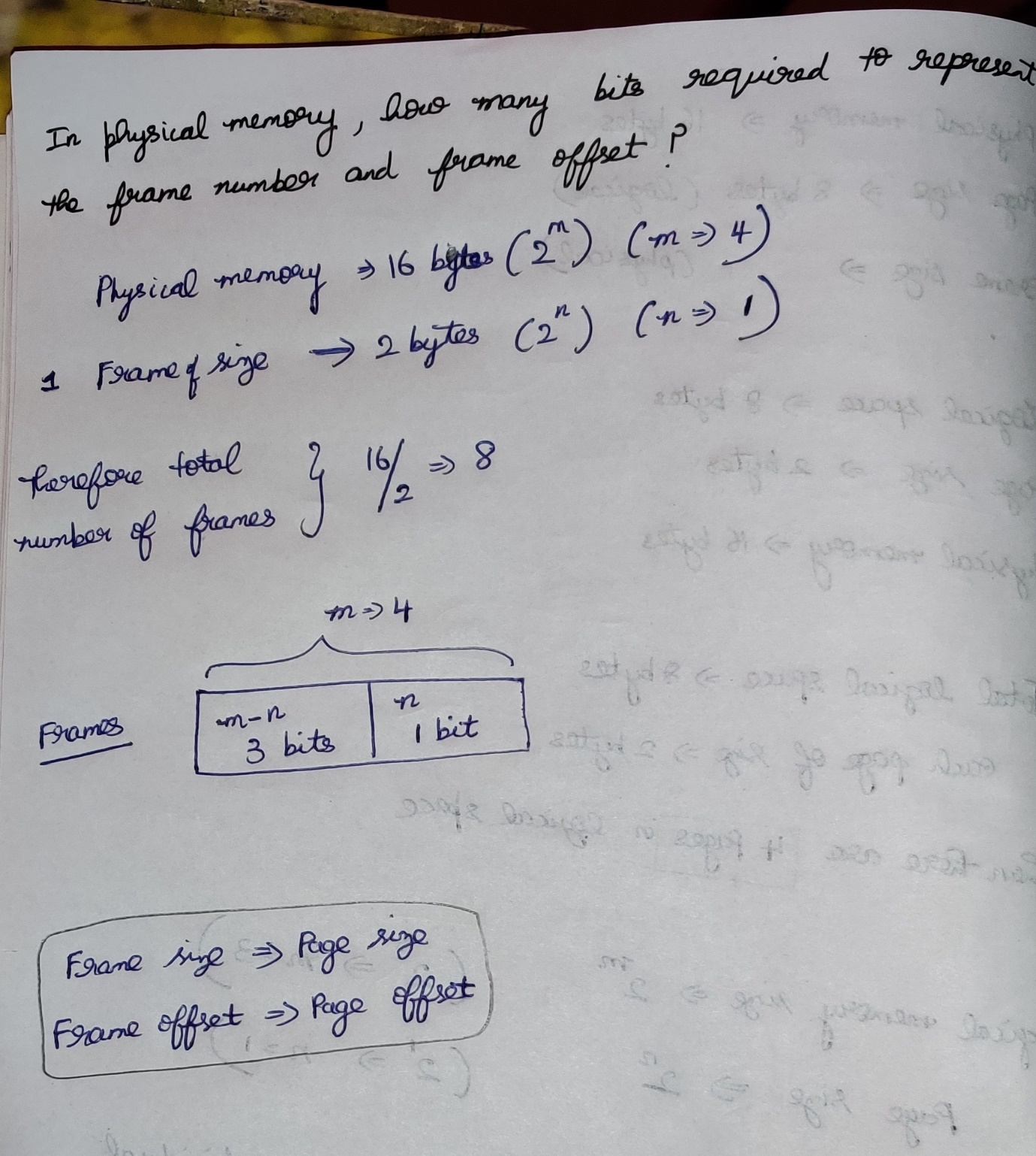
**Approach-2**

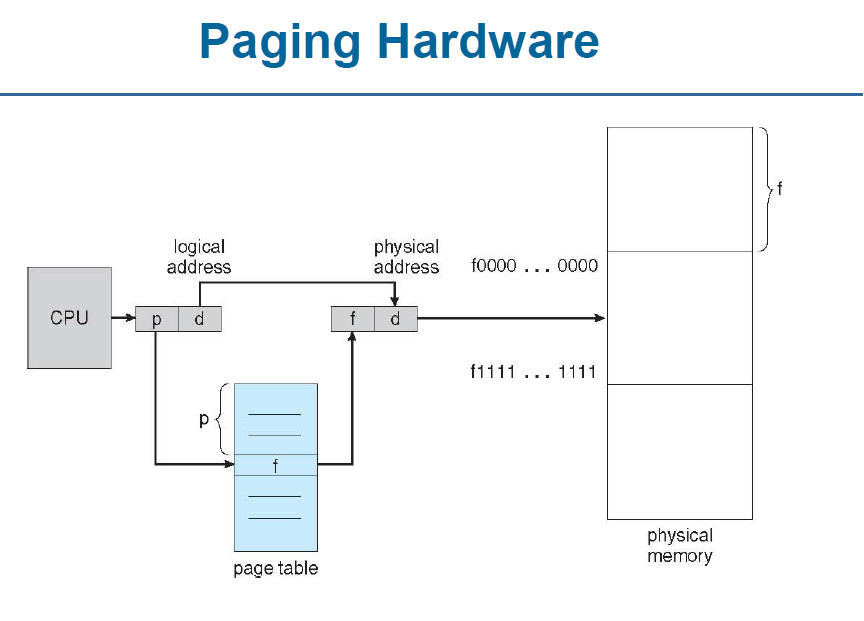
CPU generates the byte 3 which is present at 9.   
So 3 should be converted into 9 🡪 Done by Memory Management Unit  
Page table 🡪 Converting logical address(CPU generated) into Physical address  
(Memory management unit)

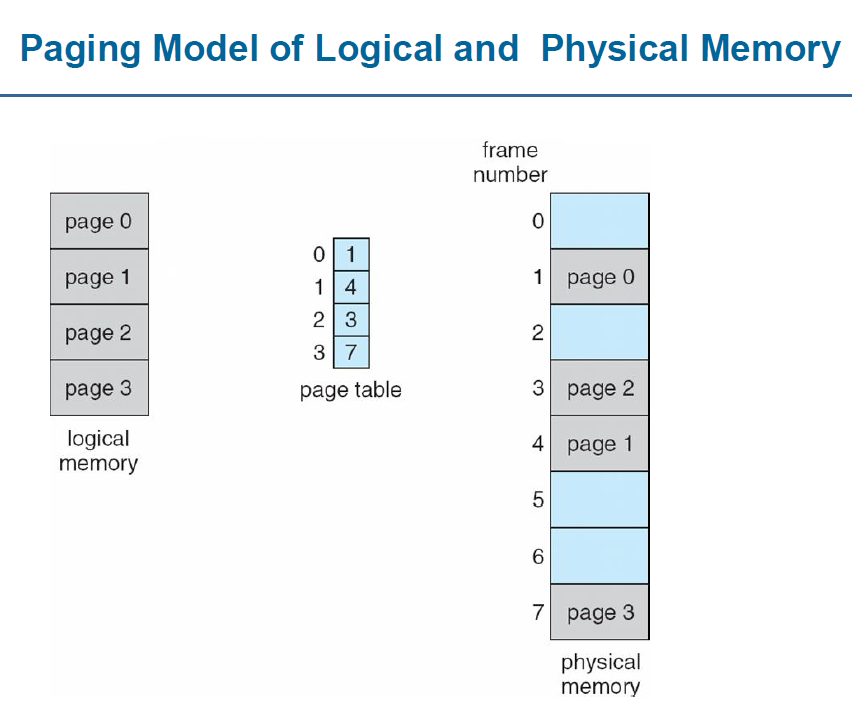
Total number of frames[Frame number] 🡪 8 (3 bits)  
Frame size [Frame offset] 🡪 2 (1 bit)  
CPU wants the 3rd byte. It is present at Page-1 | Frame-4(100), in that 1st bit(1) where 9th byte is present.  
**Question: CPU wants the 1st byte**

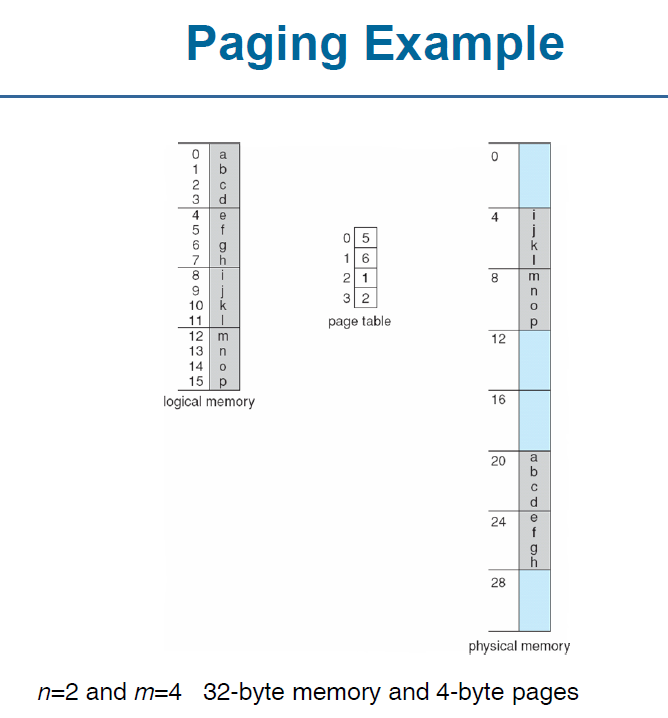
* ***Process P1 is divided into 2 pages, then page table is divided into 2 pages. Therfore   
  Number of pages a = Number of pages a   
  process is divided page table is also divided***
* Logical address space always specifies the size of the process
* ***Number of entries in*** the ***page table = Number of pages in a process***







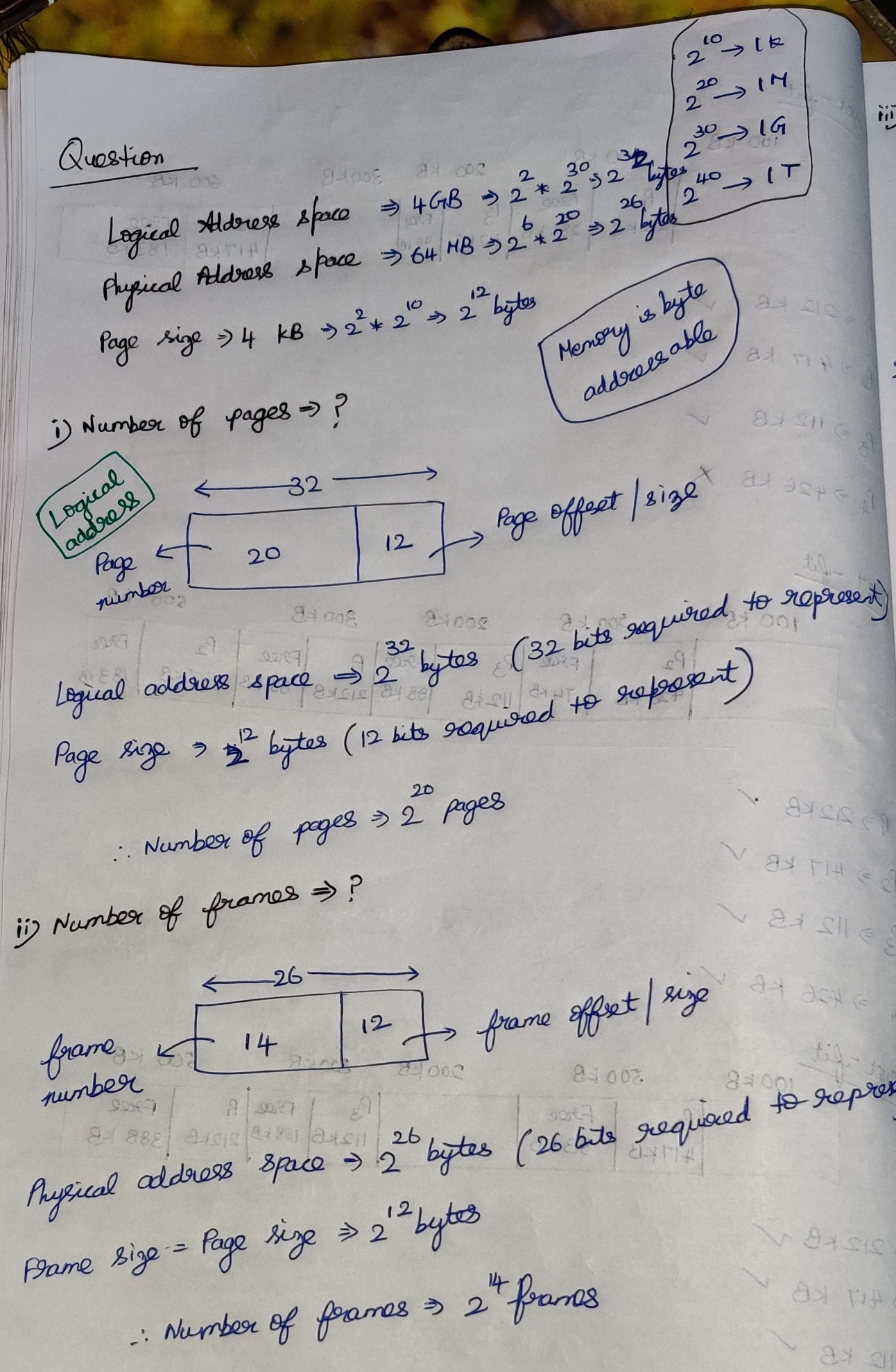


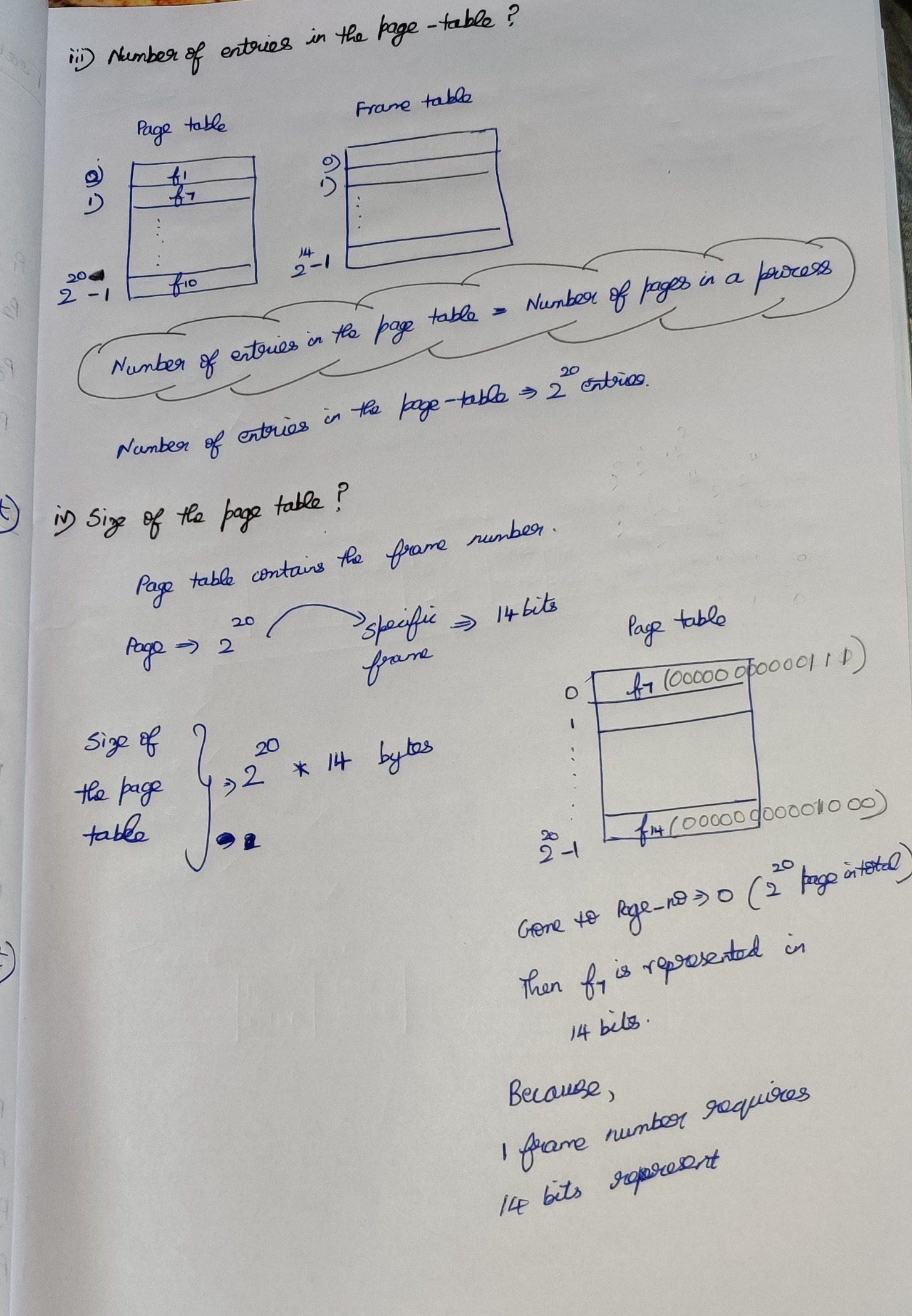
 4 pages 8 frames

2^4 🡪 16 bytes (m=4) 2^5 🡪 32 bytes (m=5)  
 ( 0 to 15) ( 0 to 31 )

1 Page 🡪 4 bytes (2^n where n=2) . Total 🡪 4 pages

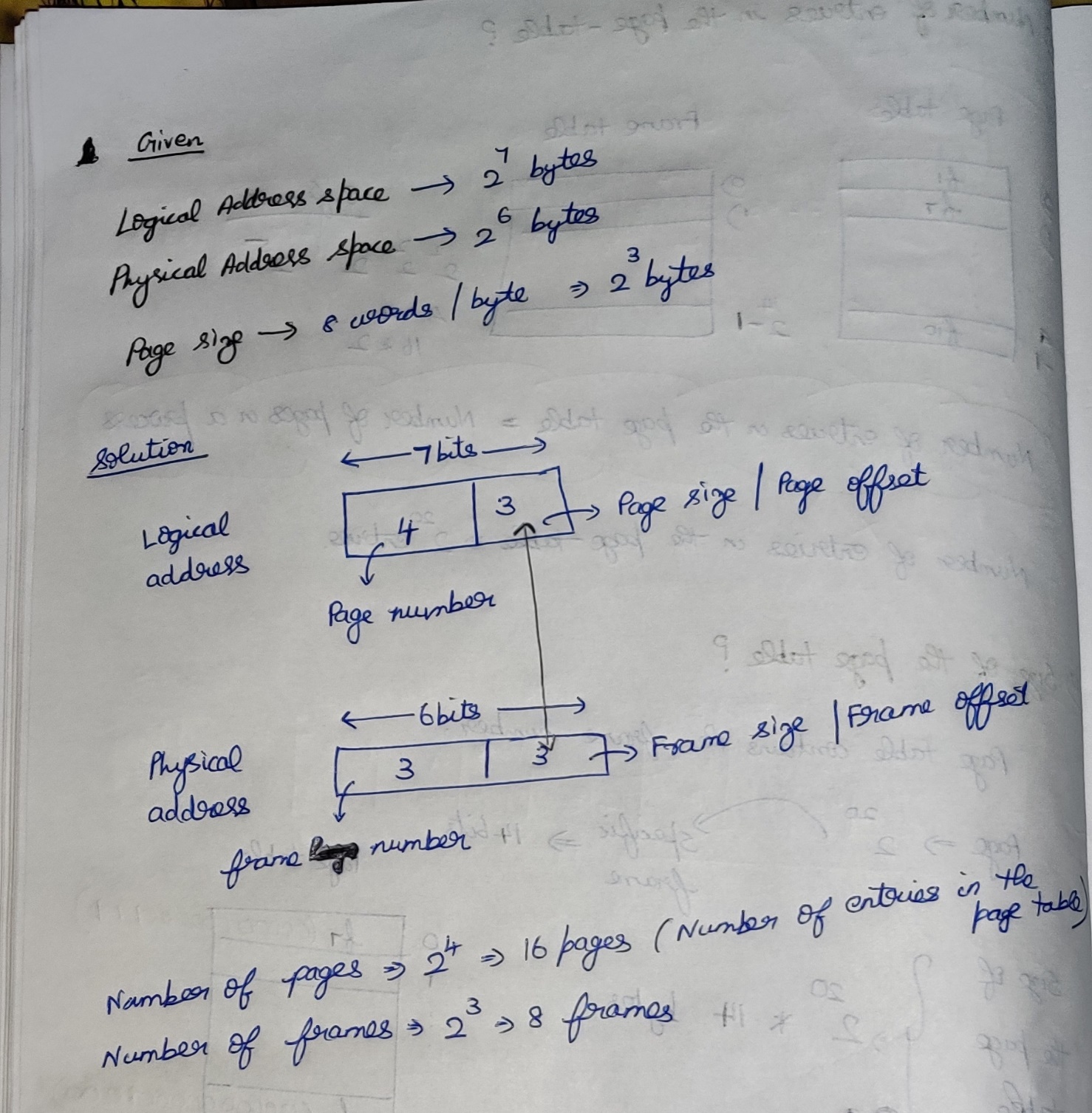
1 frame 🡪 4 bytes, total 🡪 8 frames



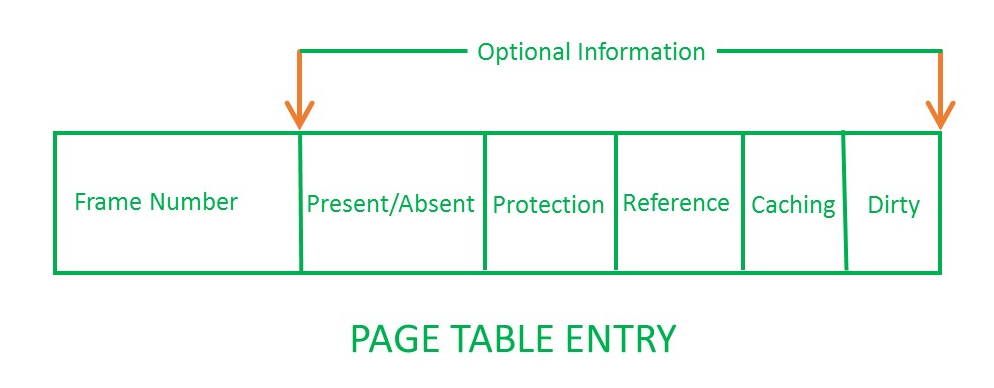


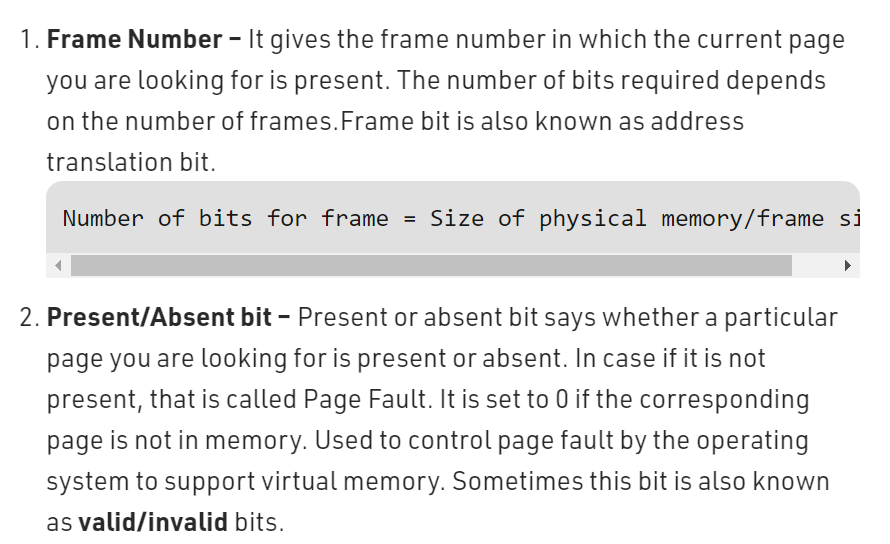
**Question**  
Consider a system which has Logical Address 🡪 7 bits and Physical address 🡪6 bit, page size=8 words. Calculate the number of pages and number of frames??

***Number of words = Number of bytes (Will change accordingly)  
32 bit PC, then 4 byte PC, then total number of words 🡪 4  
64 bit PC, then 8 byte PC, then total number of words 🡪 8***

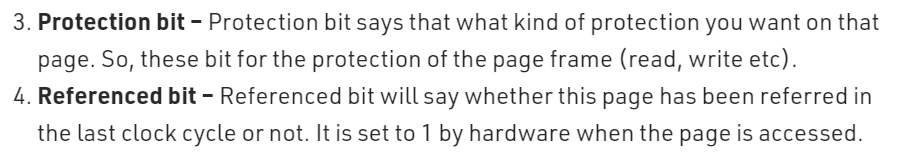


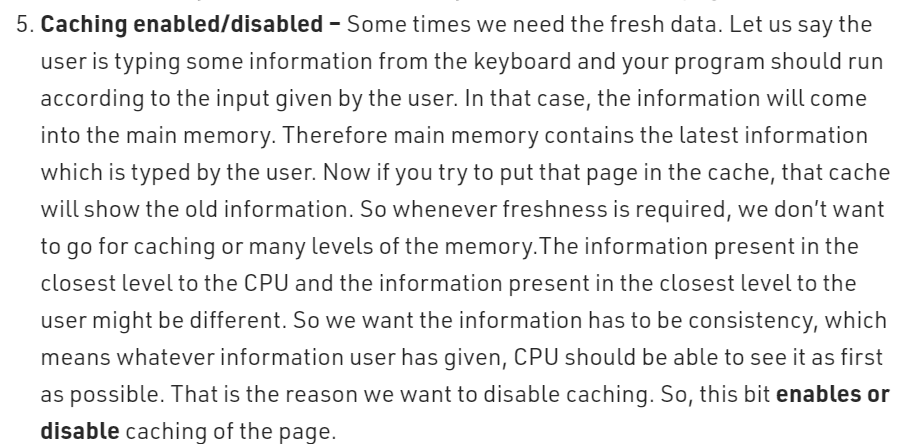
**Page table entries**





**Valid/Invalid 🡪 Page hit/Page fault**







**Level Paging**

Need of Level Paging 🡪 Page table size > frame size

Each process has its own page table.  
Page table will be in main memory.

Physical address space 🡪 256 MB  
Logical address space 🡪 4GB  
Frame size 🡪 4KB  
Page table entry 🡪 2B