■ Comparison between fixed and variable partition methods.

Fixed Partitioning	Dynamic/variable Partitioning
1. It is the OS that decides the partition size only once at the system boot time.	1. OS has to decide about partition size, everytime a new process is chosen by long term scheduler.
2. Here the degree of multiprogramming is fixed.	2. Here the degree of multiprogramming will vary depending on program size.
3. It leads to internal fragmentation.	3. It leads to external fragmentation. No internal fragmentation is there.
4. IBM-360 DOS and OS/MFT OS used this approach.	4. IBM OS/MFT used this approach also.

Non-Contiguous Storage Allocation:

To permit sharing of code and data among processes, to resolve the problem of external fragmentation of physical memory, to enhance degree of multiprogramming and to support virtual memory concept, it was decided to have non-contiguous physical address space of a process. So now a process could be allocated memory whenever it was available. However, it involves complex implementation and an additional cost in terms of memory and processing.

There are three main methods of non-contiguous Storage allocations —

(a) Paging
(b) Segmentation
(c) Segmentation with Paging (Combined approach)

Next...

Paging :

Paging is an efficient memory management scheme because it is non-contiguous memory allocation method. The previous (single partition, multiple partition) methods support the continues memory allocation, the entire process loaded in the same partition. But in the paging the process is divided into small parts, these are loaded into else where in the main memory.

The basic idea of paging is the physical memory (Main memory) is divided into fixed sized blocks called frames, the logical address space (User job) is divided into fixed sized blocks, called pages, but page size and frame size should be equal. The size of the frame or a page is depending on operating system. Generally the page size is 4KB.

In this scheme the operating system maintains a data structure, that is page table. It is used for mapping purpose. The page table specifies some useful information, it tells which frames are allocated, which frames are avaiable, and how many total frames are there and so on. The general page table consisting of two fields, one is page number and other one is frame number. Each operating system has its own methods for storing page tables, most allocate a page table for each process.

Each address generated by the CPU divided into two parts, one is 'Page Number' and second is 'Page offset' or displacement. The page number is used index in page table. The pages are loaded into available free frames in the physical memory.

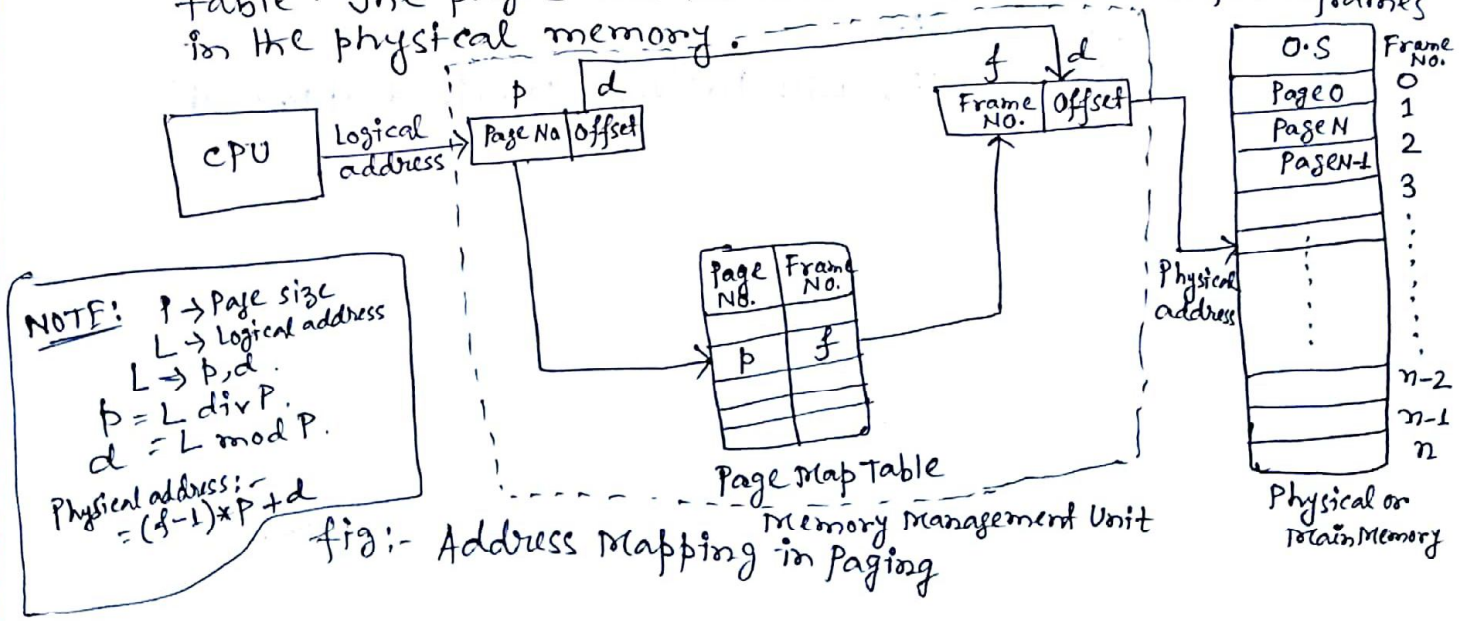
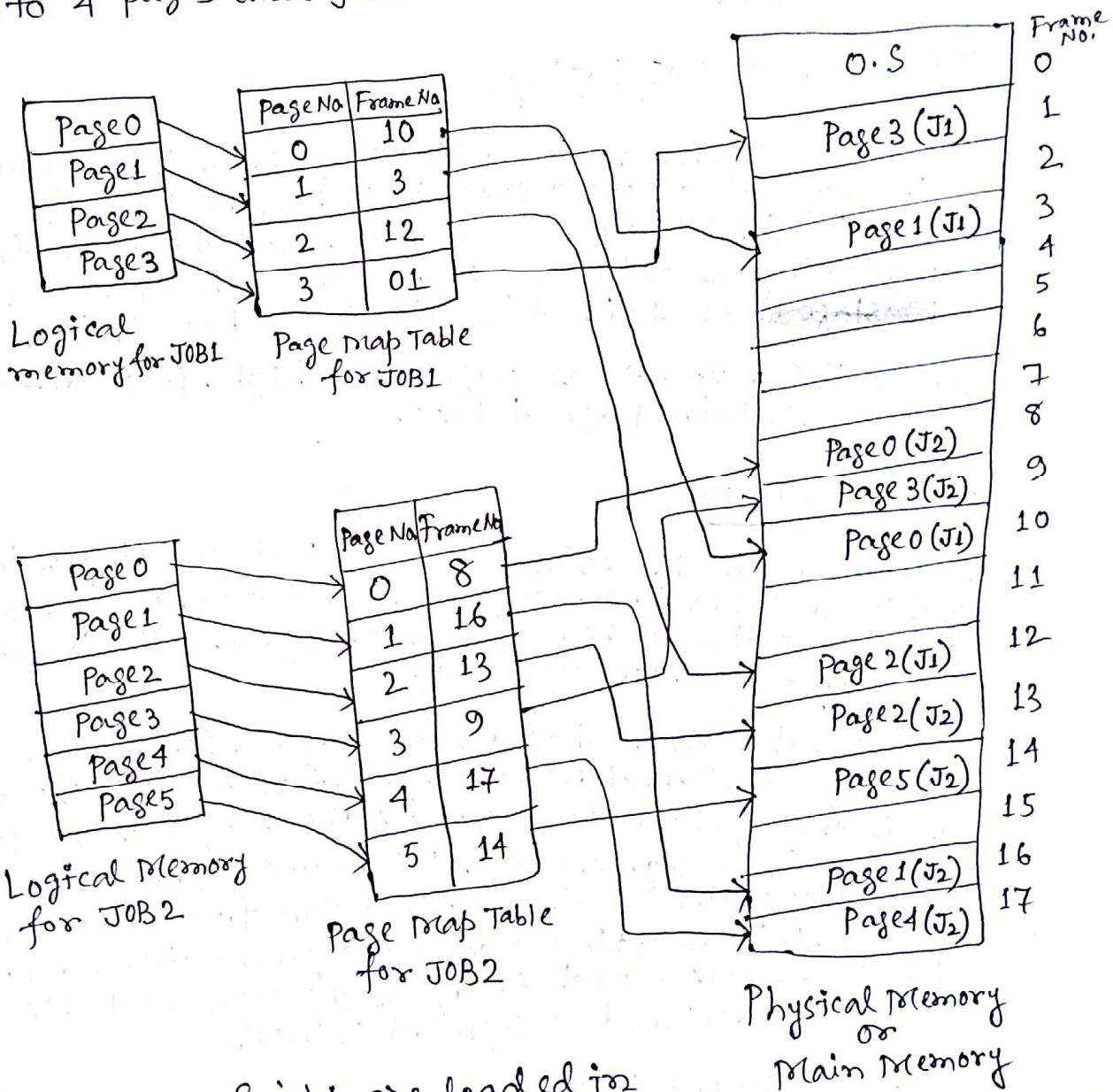


fig:- Address Mapping in Paging

The mapping between page number and frame number is done by page map table. The page map table specifies which page is loaded in which frame, but offset is common.

Suppose, there are two jobs in the ready queue, the jobs sizes are 16 KB and 24 KB, the page size is 4 KB. The available main memory is 72 KB (18 frames), so job1 divided into 4 pages and job2 divided into 6 pages.



Four Pages of job1 are loaded in different locations in main memory. The O.S. provides a page table for each process, the page table specifies the location in main memory. The capacities of main memory in the above example is 18 frames. But the avaiable jobs are 2 which consists of 10 pages. So, the remaining 8 frames are free. The scheduler can use this free frames for some other jobs.

Advantages of Paging:

1. It supports the time sharing system.
2. It does not effect from external fragmentation.
3. It achieves a high degree of multiprogramming.
4. Sharing of common code among the processes is possible.

Disadvantages of Paging:

1. This scheme may suffer 'Page breaks'. For example the logical address space is 17 KB, the page size is 4KB. So, this job requires 5 frames. But the fifth frame consisting of only one KB. So the remaining 3 KB is wasted. It is said to be page breaks.
2. If the number of pages are high, it is difficult to maintain page tables.

Segmentation:

A segment can be defined as a logical grouping of instructions, such as a subroutine, array or a data area. Every program(job) is a collection of these segments. Segmentation is a technique for managing these segments.

Each segment has a name and a length. The address a of the segment specifies both segment name and the offset with in the segment. For example the length of the segment is 100KB for a segment with the name. The operating system search the entire main memory for free space to load a segment, this mapping done by segment table. The segment table is a table, each entry of the segment table has a segment 'Base' and a segment 'Limit'.

The segment base contains the starting physical address where the segment resides in memory. The segment limit specifies the length of the segment.

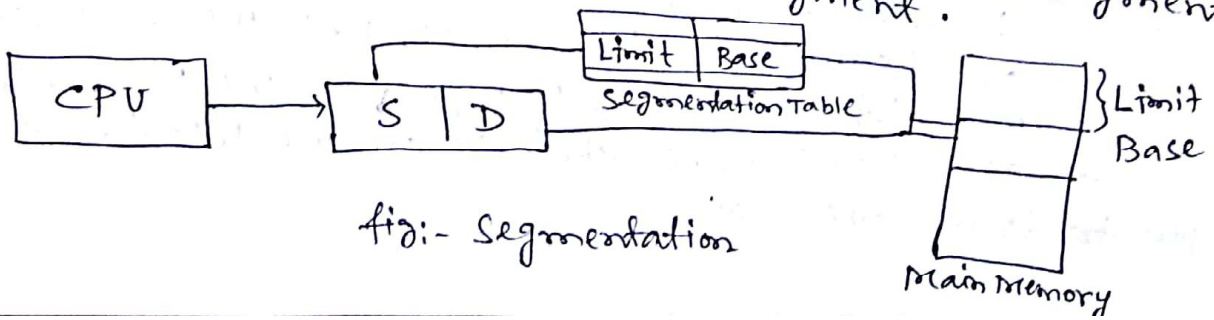


fig:- Segmentation

Segmentation

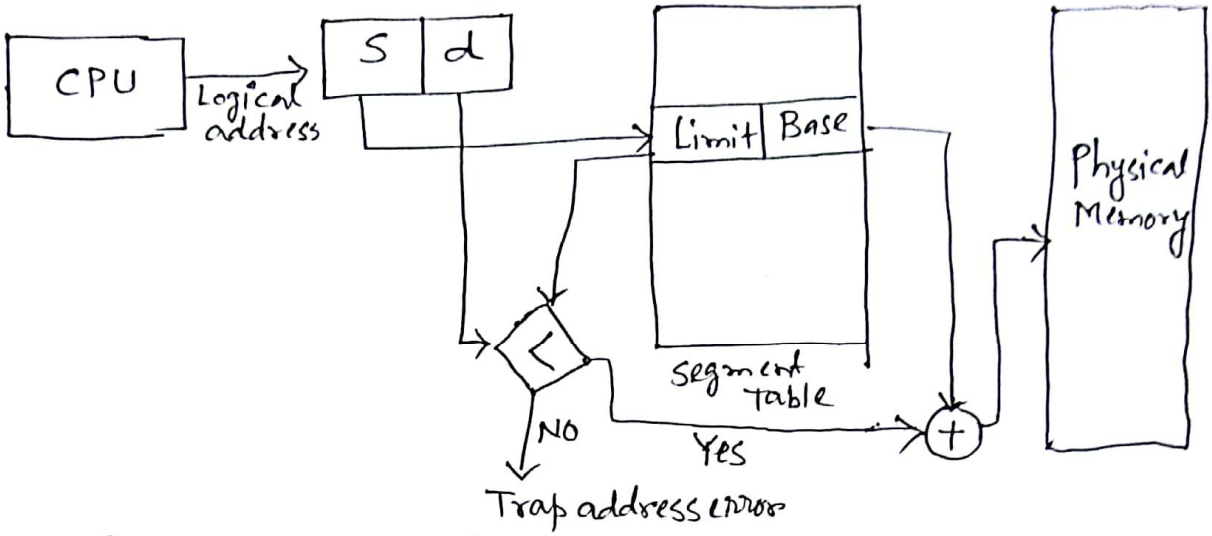
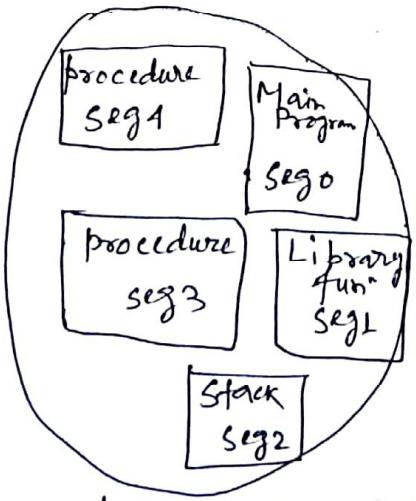


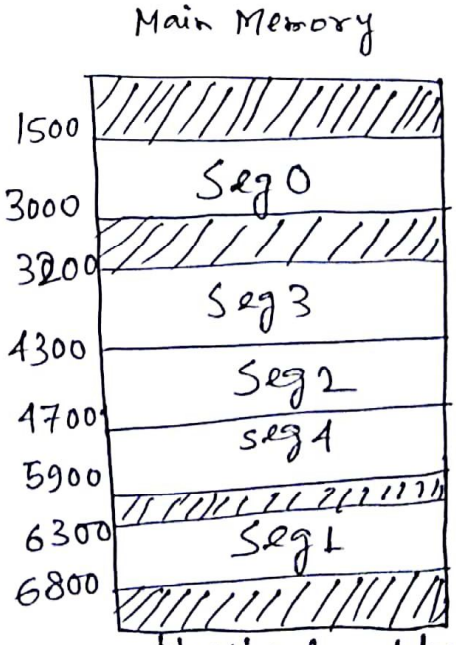
fig:- Hardware for segmentation

if Limit = 400, then d → (0 to 399).

	Limit	Base
0	1500	1500
1	500	6300
2	400	4300
3	1100	3200
4	1200	4700

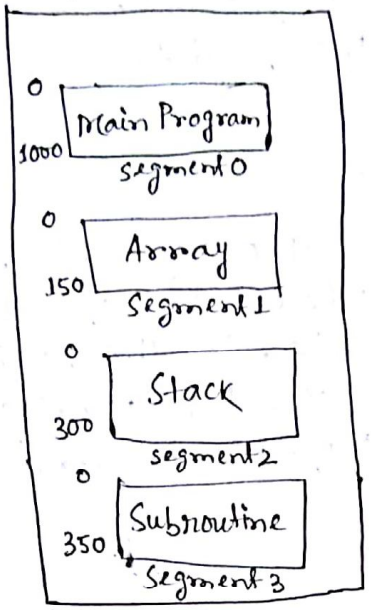


Logical address



physical address

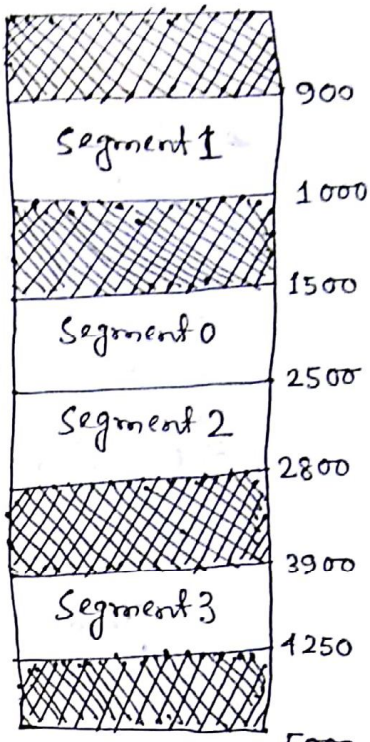
The logical address consists of 2 parts – a segment number (s) and an offset into that segment (d). The segment number (s) is used as an index into the segment table.



Logical address space

Seg. No.	Size	Base
0	1000	1500
1	150	900
2	300	2500
3	350	3900

Segment Table



Main Memory

fig:- Example of Segmentation.

The logical address space (a job) is divided into 4 segments, numbered from 0 to 3. Each segment has an entry in the segment table. The limit specifies the size of the segment and the base specifies the starting address of the segment. For example segment 0 loaded in the memory from 1500 KB to 2500 KB, so 1500 KB is the base and (2500 – 1500) = 1000 KB is the limit or size.

Advantages of Segmentation:

1. Eliminate fragmentation, by moving segments around, fragmented memory space can be combined into a single free area.
2. Segmentation support virtual memory.
3. Allow dynamically growing segments.
4. Dynamic linking and loading of the segments.
5. Sharing of code or data is possible.

Disadvantages of Segmentation:

When the number of segments is large then the size of the segment table will also grow. So, it cannot be accomodated in any of the registers and has to be kept in memory.

Segmentation With Paging:

The combined scheme is 'Page the segments'. Each segment in this scheme is divided into pages and each segment maintains a page table. So, the logical address is divided into 3 parts: (S, P, D) where S is the segment number, P is the page number and D is the offset or displacement.

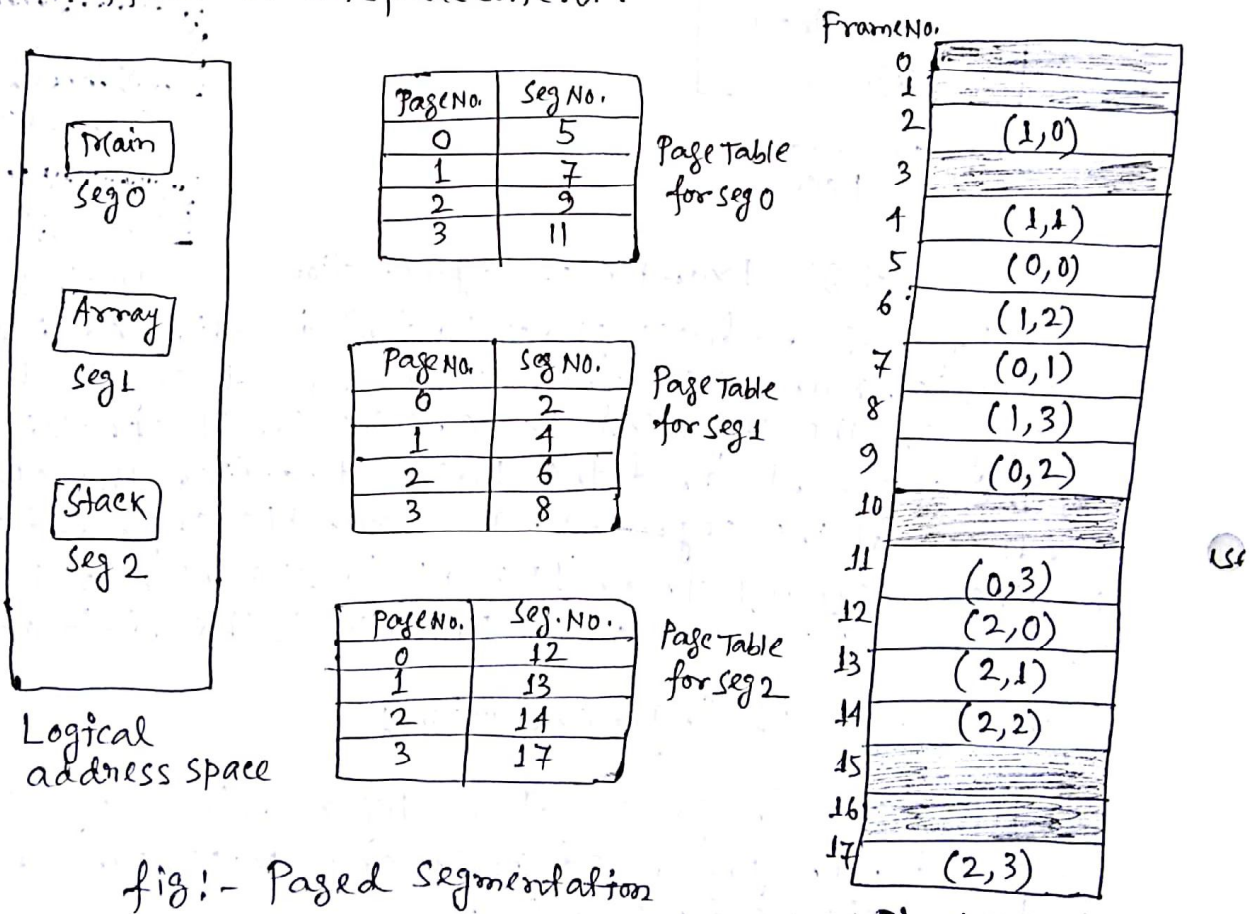


fig:- Paged Segmentation

In fig, the logical address space is divided into three (3) segments from 0 to 2. Each segment maintains a page table, which does mapping between pages and frames. For example, the frame number 6 shows the address (1,2), where '1' stands for segment number and '2' stands for the page number.

The main advantage of this scheme is to avoid fragmentation.

Paging vs. Segmentation

Paging	Segmentation
1. The main memory partitioned into frames or blocks.	1. The main memory partitioned into segments.
2. The logical address space divided into pages by compiler or memory management unit (MMU)	2. The logical address space divided into segments, specified by the programmer.
3. This scheme suffering from internal fragmentation or page break.	3. This scheme suffering from external fragmentation.
4. The OS maintains a free frames list need not to seach for free frame.	4. The OS maintains the particulars of available memory.
5. The OS maintains a page map table for mapping between frames and pages.	5. The OS maintains a segment map table for mapping purpose.
6. This scheme does not support users view of memory.	6. This scheme supports users view of memory.
7. Processor uses the page number and displacement to calculate the absolute address (P,D).	7. Processor uses the segment number and displacement to calculate the absolute address (S,D).
8. Multilevel paging is possible.	8. Multi level segmentation is also possible, but no use.

Translation Look Aside Buffer (TLB):

Whenever the processor need to access a particular page, first of all it search the translation look aside buffer (TLB). The TLB acts like a cache and contains page table entries, that entries must have been recently used or frequently used. If the desired page table entry present in the TLB (it is said to be 'TLB Hit') then the frame number is retrieved and the real address is accessed. If the desired page table entry is not present in TLB ("TLB Miss"), then the processor search the page map table for corresponding page table entry. ~~If the valid entry~~

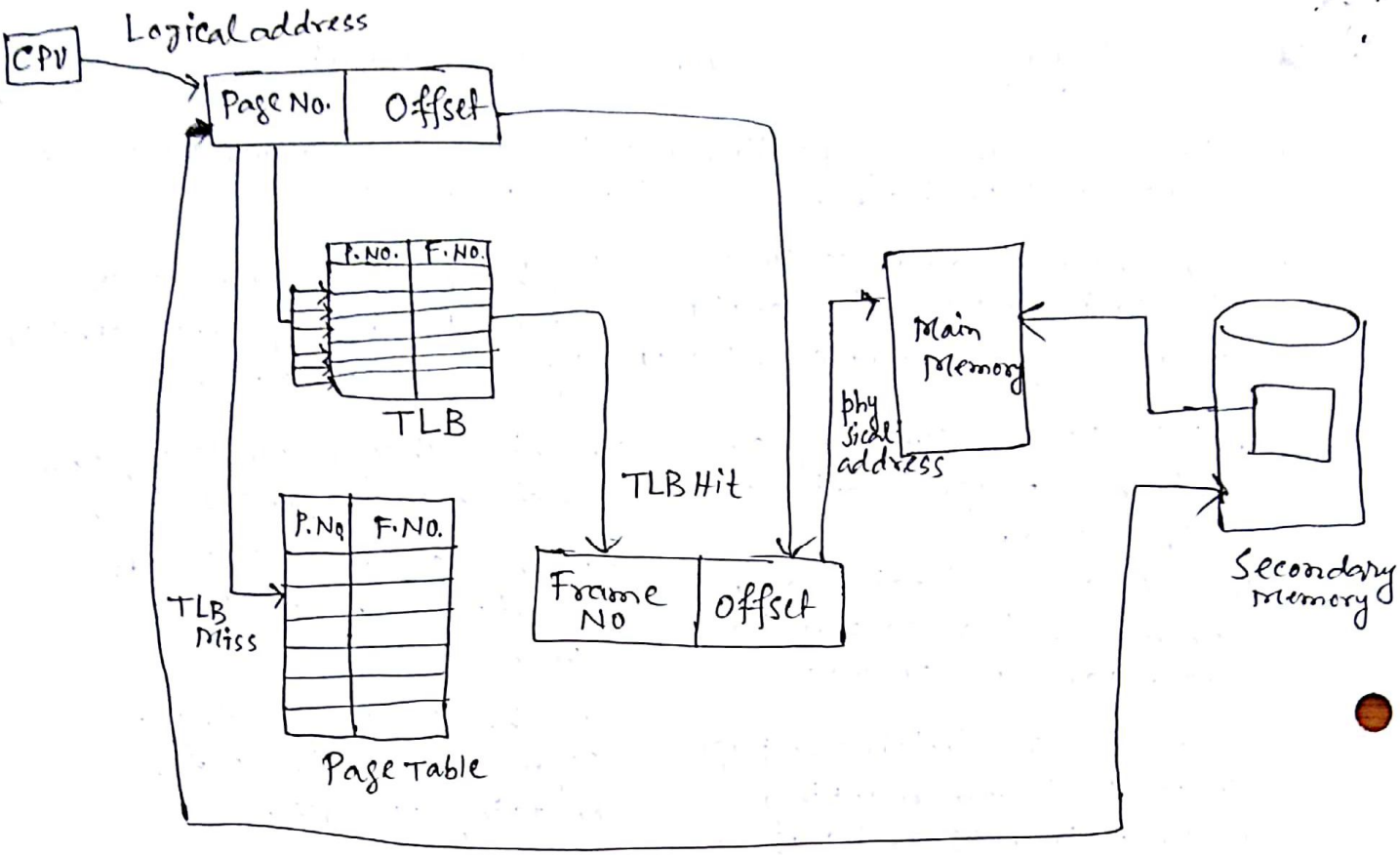


fig:- Functioning of TLB

Question: Why are page sizes always a power of 2?

Ans:- Paging implemented by breaking up an address into a page (p) and offset(d). It is most efficient to break the address into X page bits and Y offset bits, rather than to perform arithmatic on the address to calculate the page number and offset. Because each bit position represents a power of 2, splitting an address between bits results in a page size that is power of 2.

Ques: Main Memory and TLB are having accessing time of 220ns and 120ns respectively. The TLB hit ratio is about 98%. Calculate the effective access time.

$$EAT = h \times t_{TLB} + (1-h) \times t_{M.M}$$

$$= 0.98 \times 120 + 0.02 \times 220 = 122 \text{ ns}$$