CHAPTER 05 Memory Management

* Memory Management: It is responsible for managing the computer's primary memory. It is crucial to manage the main memory as it is available in limited space. Module of OS.

Functions of Memory Management

1. Memory Allocation: Allocate memory to a new arriving process
2. Memory deallocation: It is responsible for deallocate memory from completed process.
3. Memory Protection: While running, a process can access only that memory which is allocated to it.

Goals of Memory Management

1. Maximum Utilization of Space. (minimum wastage of space)
2. Ability to run larger programs with limited space. Using Virtual Mem.

Memory Management Scheme

Entire process must be stored in consecutive memory location

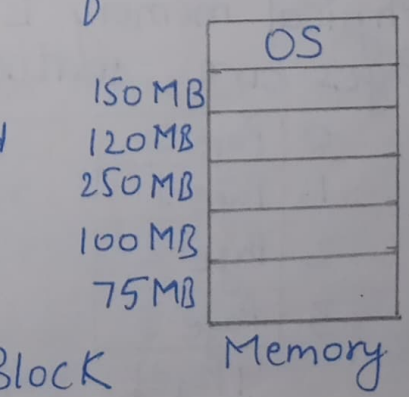
A process is divided into multiple parts can be stored anywhere in memory

- Contiguous Mem. Management Scheme
  - Single Contiguous mem. management scheme
  - Multiple Partitioning
    - Fixed Partitioning
    - Dynamic Partitioning
- Non-Contiguous Memory Management Scheme
  - Paging
  - Segmentation

* Fixed Partition Contiguous MMT Main memory is divided into several fixed sized partitions. These partitions can be of the same size or different sizes. Each partition can hold a single process. The no. of partitions determines the degree of multiprogramming. These partitions are made at the time of system generation and remain fixed after that.

* Internal Fragmentation

When space allocated to a process is more than it's required space, then the extra allocated space is wasted and that wastage of space is known as Internal Fragmentation.



Note: Solution of Internal Fragmentation is Best-Fit Block

* Partition Allocation Policy

1. First Fit : The first partition from starting which can store the process is allocated.
2. Best Fit : The smallest partition which can be used to store the process, is allocated.
3. Worst Fit : The biggest partition is allocated.
4. Next Fit : The first partition from previously allocated partition.

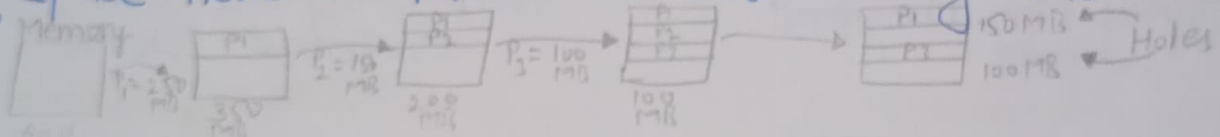
* Dynamic Partitioning

Also known as Variable Partition Contiguous MMT. It was designed to overcome the problems of fixed partitioning scheme. Whenever a new process arrives then a new partition, equal to the size of process is created and is allocated to the process. Here, no internal fragmentation.

Note: Worst fits works better in Dynamic Partitioning.

* External Fragmentation → Solution Paging

When enough space is available to store a process but not consecutively. Hence the process cannot be stored. Wastage of space here is known as external fragmentation.



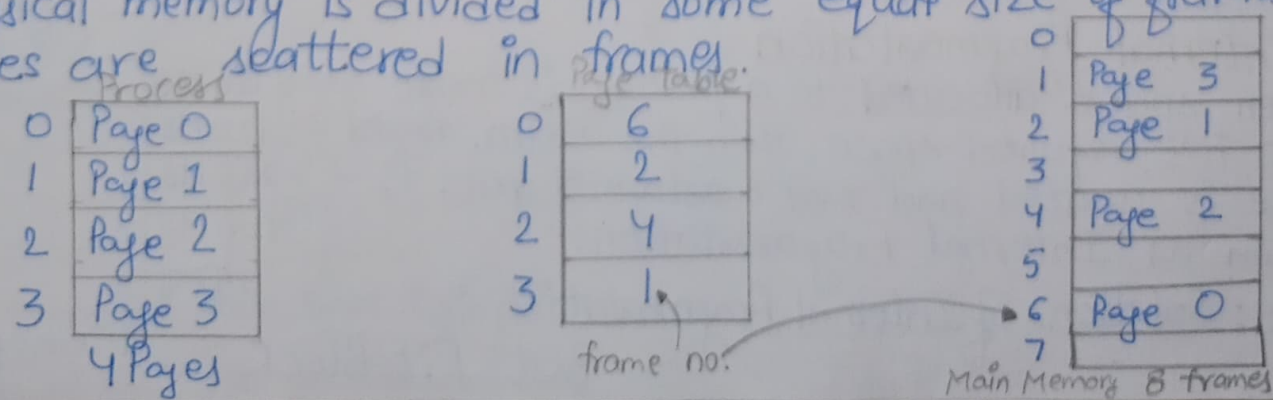
* Compaction : We know that dynamic partitioning suffers from external fragmentation. To avoid compaction, we need to collect all allocated processes into one side of memory, so that all empty spaces will be in other side of memory collectively.

* Problem with Compaction

The efficiency of the system is decreased as huge amount of time is invested for this procedure and the CPU will remain idle for all this time.

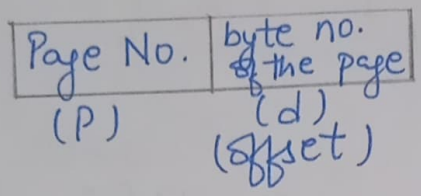
* Paging

- Process is divided into equal size of pages.
- Physical memory is divided in some equal size of frames.
- Pages are scattered in frames.



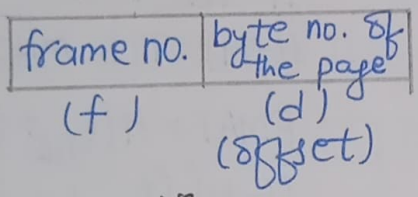
Note: 1. Page table is maintained to denote which page is stored in which frame.
2. No. of entries in a page table = no. of pages in the process.
3. Page table entry size = frame no. + extra bits.
4. OS maintain a page table for each process.
5. Paging help to solve the issue of external fragmentation.
6. Page size has impact on internal fragmentation.
7. Paging incurs memory overheads because Page Table are stored in memory.

Logical Address is divided into 2 parts:

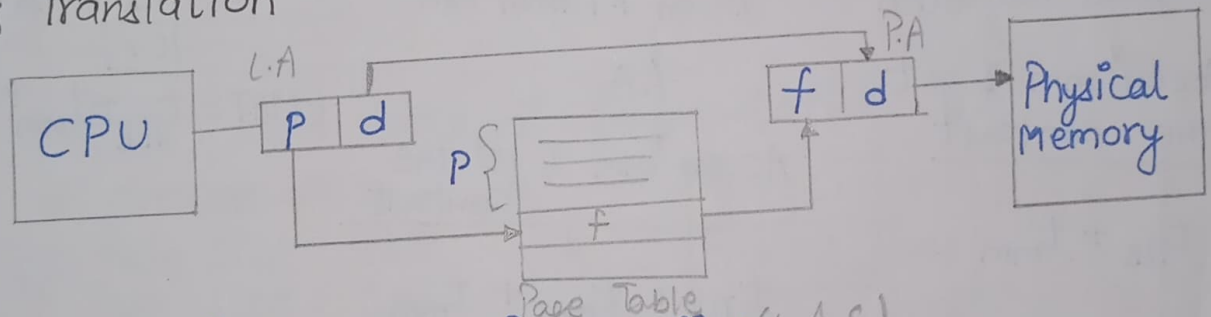


no. of bits for d = $\log_2$(Page size in bytes)

Physical Address is divided into 2 parts



Address Translation

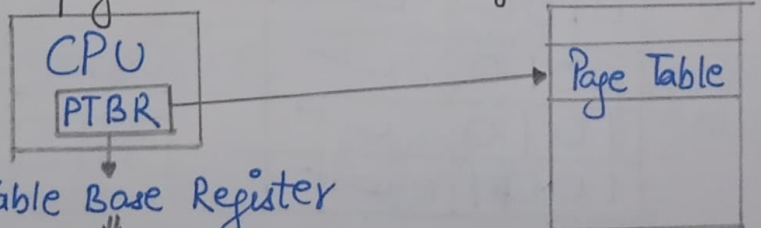


* No. of pages in process = $\frac{\text{Process Size}}{\text{Page Size}}$ (L.A.S)

* No. of frame in main memory = $\frac{\text{Physical Mem. Size}}{\text{Page Size}}$ (P.A.S)

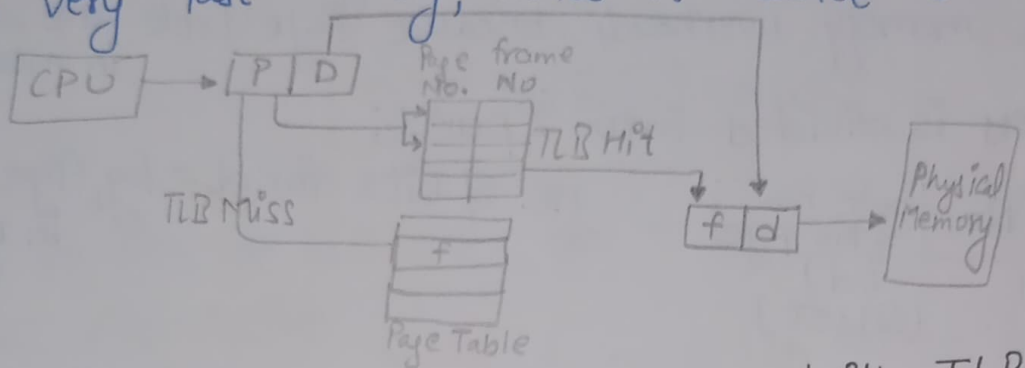
* Page Table Size = No. of pages in process × 1 P.T. entry size

* Where the page table stored? In main memory.



Page Table Base Register
it stores starting address of P.T of current running process.

* Performance of Paging: Eff. Mem. Acc. Time = $2 \times t_{mm}$
  - one for P.T
  - one for content
* Special Case: If page table is very small and kept in registers. EMAT = t_{mm} [P.T access time is negligible]
* TLB (Translation LookAside Buffer)
It is a memory hardware used to store some most frequently and recently referred page table entries. It is very fast memory, hence it reduce E.M.A.T.



CPU Generates Logical Address

Search in TLB

Hit

Physical Address (P.A)

Access mm & get the content

$t_{TLB} + t_{mm}$

Miss

Access P.T from mm

P.A

Access mm & get the content

$t_{TLB} + t_{mm} + t_{mm}$

With TLB:

H = TLB hit ratio

(1-H) = TLB miss ratio

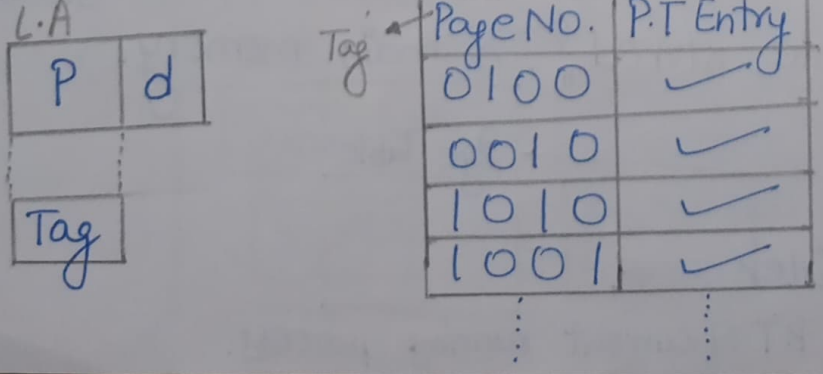
$EMAT = H(t_{TLB} + t_{mm}) + (1-H)(t_{TLB} + t_{mm} + t_{mm})$

OR

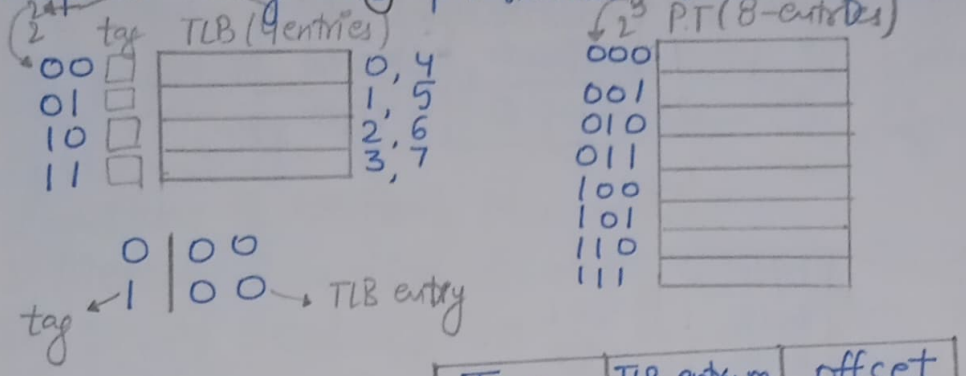
$EMAT = t_{TLB} + t_{mm} + (1-H)t_{mm}$

* TLB Mapping: Which P.T entries is stored where in TLB.

1. Fully Associative: A block of main memory can be mapped to any freely available cache line. More flexible than Direct mapping. A replacement algorithm is needed to replace a block if the cache is full.

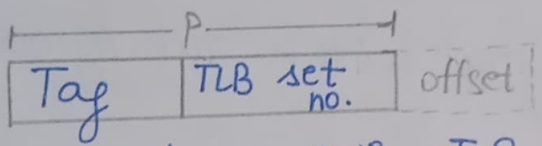


2. Direct Mapping : A particular block of main memory can map to only one particular line of the cache.

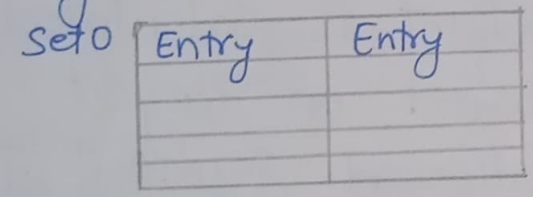


No. of bits in TLB entry no. = $\log_2$(no. of entries in TLB)

3. Set Associative



2-way set associative TLB



no. of bits in TLB set no. = $\log_2$(no. of sets in TLB)

no. of sets in TLB = $\frac{\text{no. of entries in TLB}}{\text{associativity}}$

* Paging suffers from internal fragmentation.

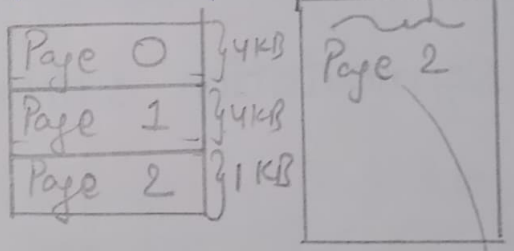
Assume a process = 9KB

Page Size = 4KB

if Page Size = 2KB

⇓

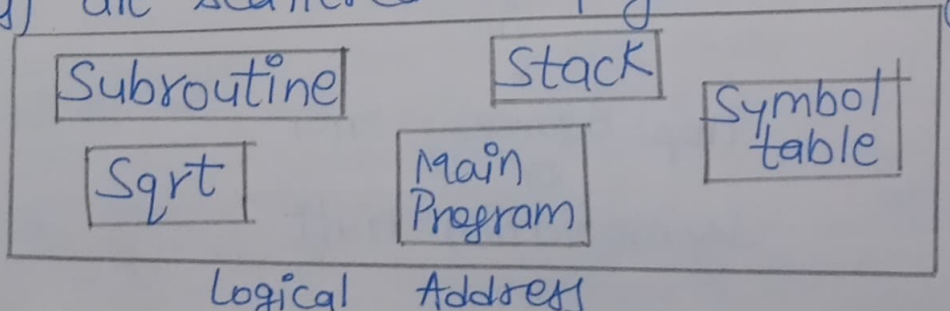
Internal fragmentation = 1KB

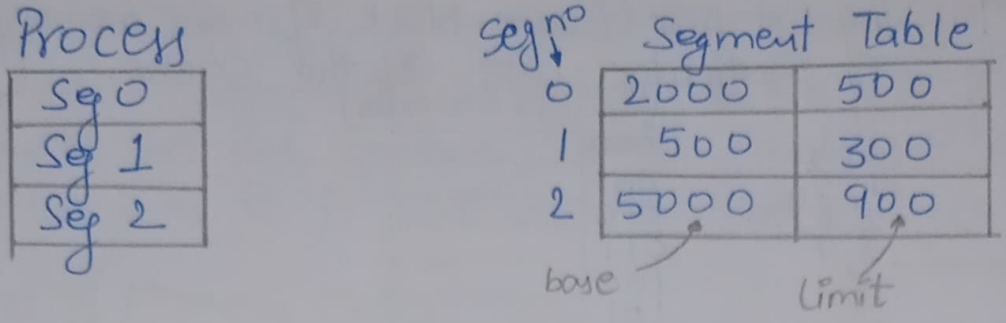


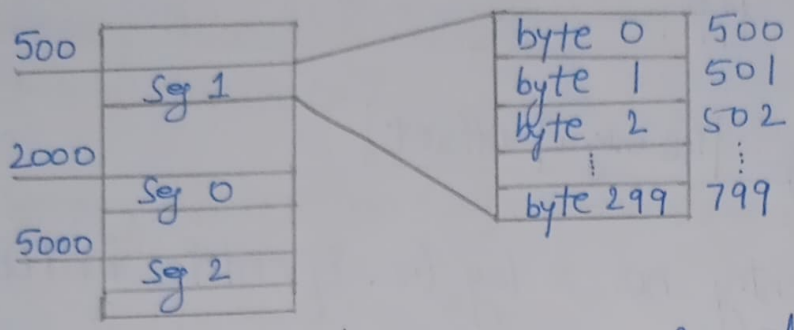
Reduce in page size can reduce internal fragmentation.

* Segmentation

A process is divided into segments (logically related partitions) are scattered in physical memory.

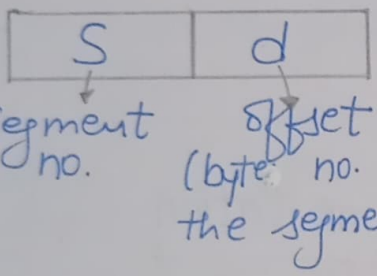






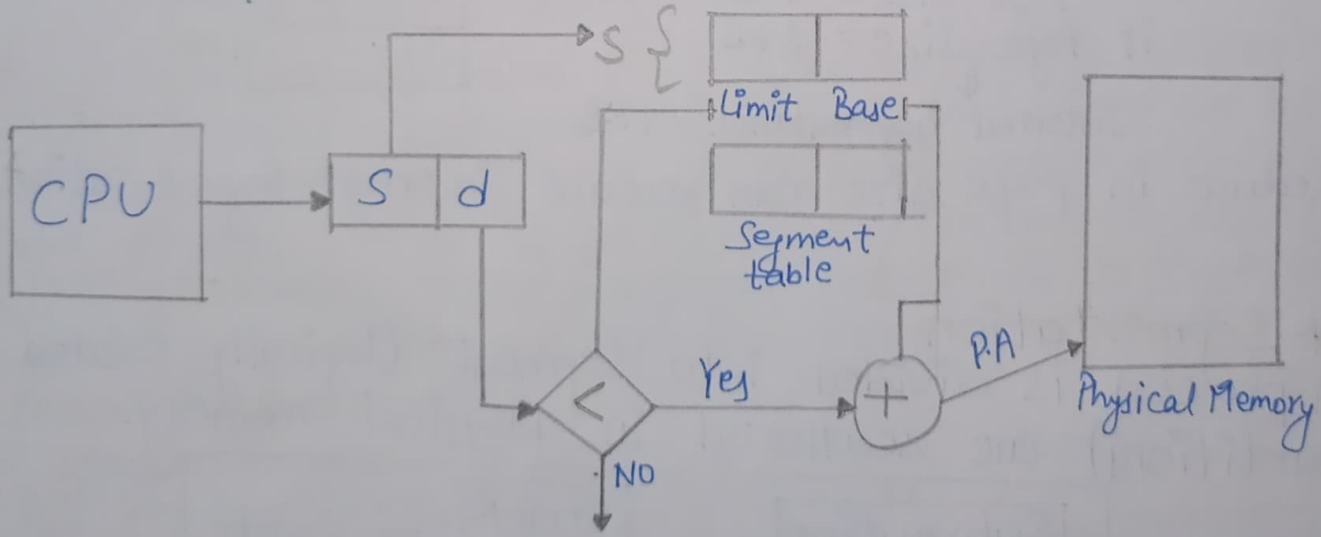
* Size of segment can vary, so along with base, keep limit information also.
* Limit defines max number of bytes within the segment.

CPU generates logical Address



Offset is calculated based on largest allowed segment size.

* Segmentation suffers from external fragmentation.



trap: addressing error
or
Segmentation fault