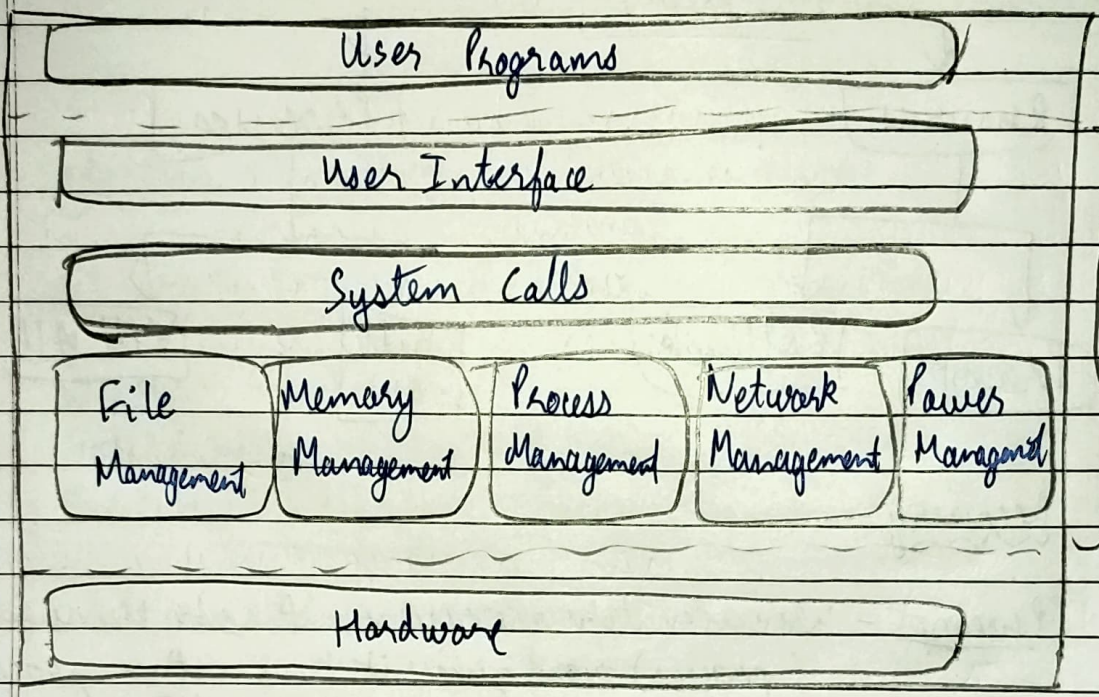
Module - 4

Page No.
Date

Memory Management

Unit 4.1

Memory Management Concepts



- Memory Management keeps tracks of each and every memory location, regardless of either it is allocated to some process or it is free.
- It checks how much pro memory is allocated to process
- It decides which which process will get memory at what time.

◇ Memory Management Techniques:

(i) Relocation: When program gets swapped out to a disk memory then it is not always possible that when it is swapped back into main memory then it occupies its previous memory location since the location may still be occupied.

by another process. We may need to relocate the process to different area of memory.

ii) Protection: Whenever we are dealing with multiple programs at same time, there always exist a danger - one program might write to address space occupied by another program.
Thus every process has to protected against all unwanted interferences when all the other processes try to write a process whether it is accidental or incidental.

iii) Sharing: Some protection mechanism allows various processes to access a similar section of main memory. Thus it allows all the processes to access the very same copy of the program instead of having their own separate copy.

iv) Logical Organisation: The main memory is organised as a linear form, or it could be an address space that's one dimensional. It comprises a sequence of words or bytes. ~~A majority of pro~~

v) Physical Organisation: The structure of computer memory has two levels referred to as main memory and secondary memory. Main memory is relatively very fast and costly as compared to secondary memory. Main memory is volatile. Thus secondary memory is provided for storage of data for long term basis while main memory holds user programs.

Page No.
Date

◇ Memory Partioning

Memory partitioning → dividing memory into chunks and to be assigned to processes in main memory.

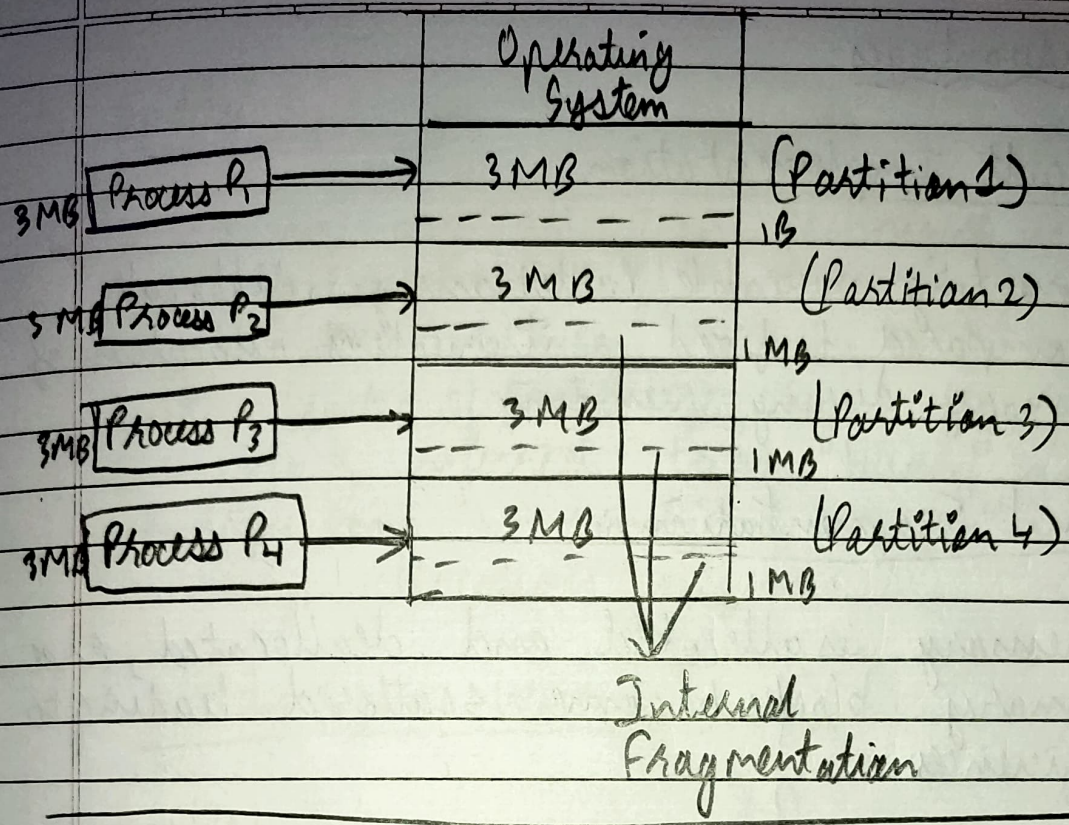
(i) Fixed - Size - In fixed-size memory partitioning is divided into blocks of the same or different sizes.
→ Fixed size memory partitioning can take place before executing any processes or during configuration of system.

Advantages:

- It is very simple and easy to implement.
- No overhead - There is no overhead associated with managing varying partitions sizes during execution of processes.
- Better control over memory allocation.

Disadvantages -

- Wasteful memory usage (Internal Fragmentation) - It can lead to inefficient memory allocation of small processes which are allocated to large partitions.
- Limited flexibility.
- ~~Predictable~~
- ~~Better control or~~
- External fragmentation



(ii) Dynamic Partitioning -

Dynamic Partitioning is a memory management technique used in computing systems to allocate and deallocate memory resources when needed by them.
It allows for creation of new memory partitions or the resizing of existing ones at runtime.
~~It helps in memory~~

Advantages:

- No internal Fragmentation - Space in main memory is strictly allocated according to needs of the process hence no wastage of memory space.
- No restriction on degree of concurrency:

More processes can be accomodated due to absence of internal fragmentation.

Page No.
Date

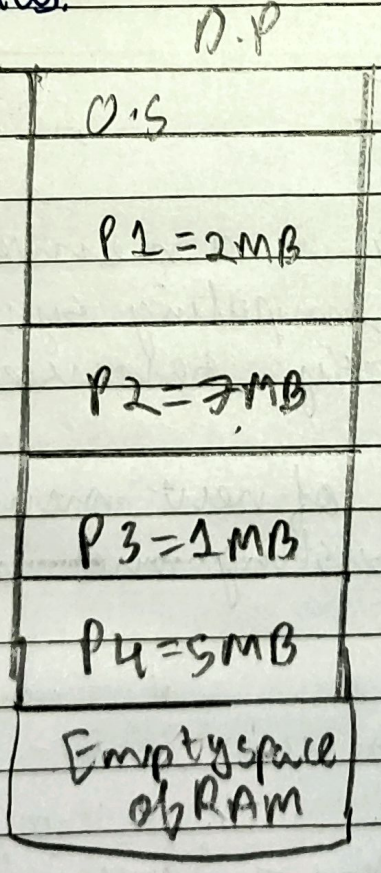
Disadvantages:

- **Difficult Implementation:**

Implementing variable partitioning is difficult as compared to fixed as it involves allocation of memory during run time.

- **External Fragmentation:**

As memory is allocated and deallocated, free memory blocks become scattered leading to inefficiencies.



Block size = 2MB

Block size = 7MB

Block size = 1MB

Block size = 5MB

Partition size = process size.

(iii) Buddy Systems

→ It is a memory allocation technique used in OS to manage memory efficiently.
- It involves dividing the available memory into blocks of fixed size; typically powers of 2.
→ Each block referred to as "buddy" and these buddies can be combined or split to fulfill memory allocation requests.

Buddy System Working

- Initial Allocation
- Buddy Splitting
- Buddy Combination
- Memory Allocation

- If process needs certain size it is allocated the nearest available block size.
- Reduces external fragmentation.

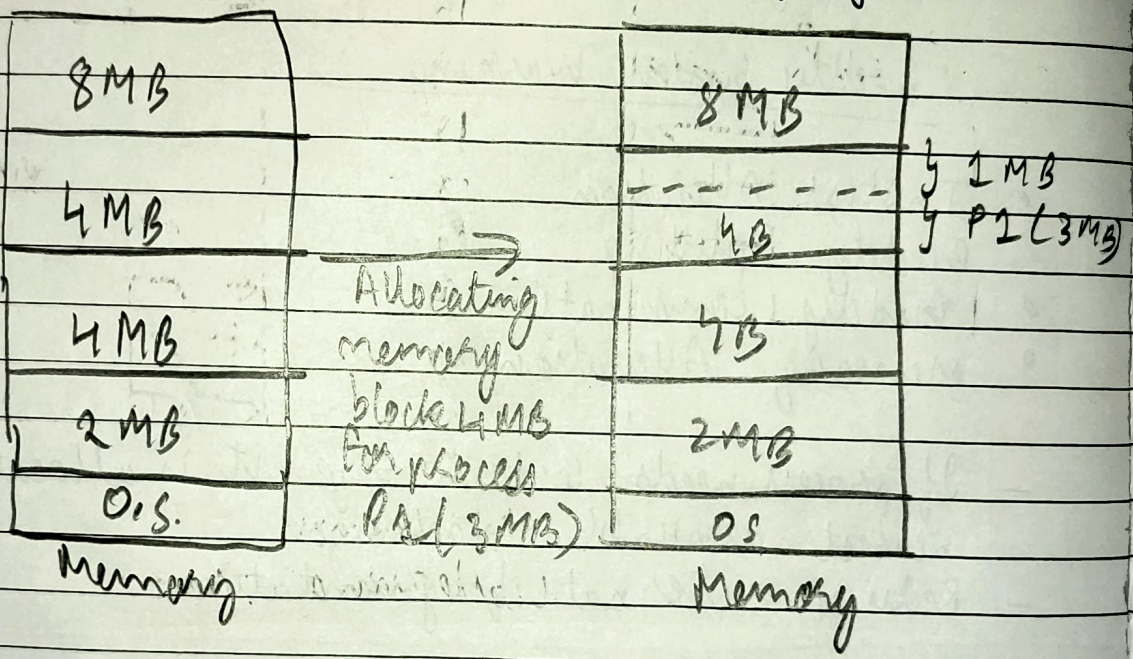
(iv) Fragmentation :

Fragmentation is unwanted problem in O.S in which the processes are loaded and unloaded from memory space is fragmented. Processes can't be assigned to memory blocks due to their small size.

Types

◊ Internal Fragmentation

When a process is allocated to a memory block and if the process is smaller than the amount of memory requested, a free space is created in the given memory block.
Due to this, the free space of the memory block is unused, which causes internal fragmentation.



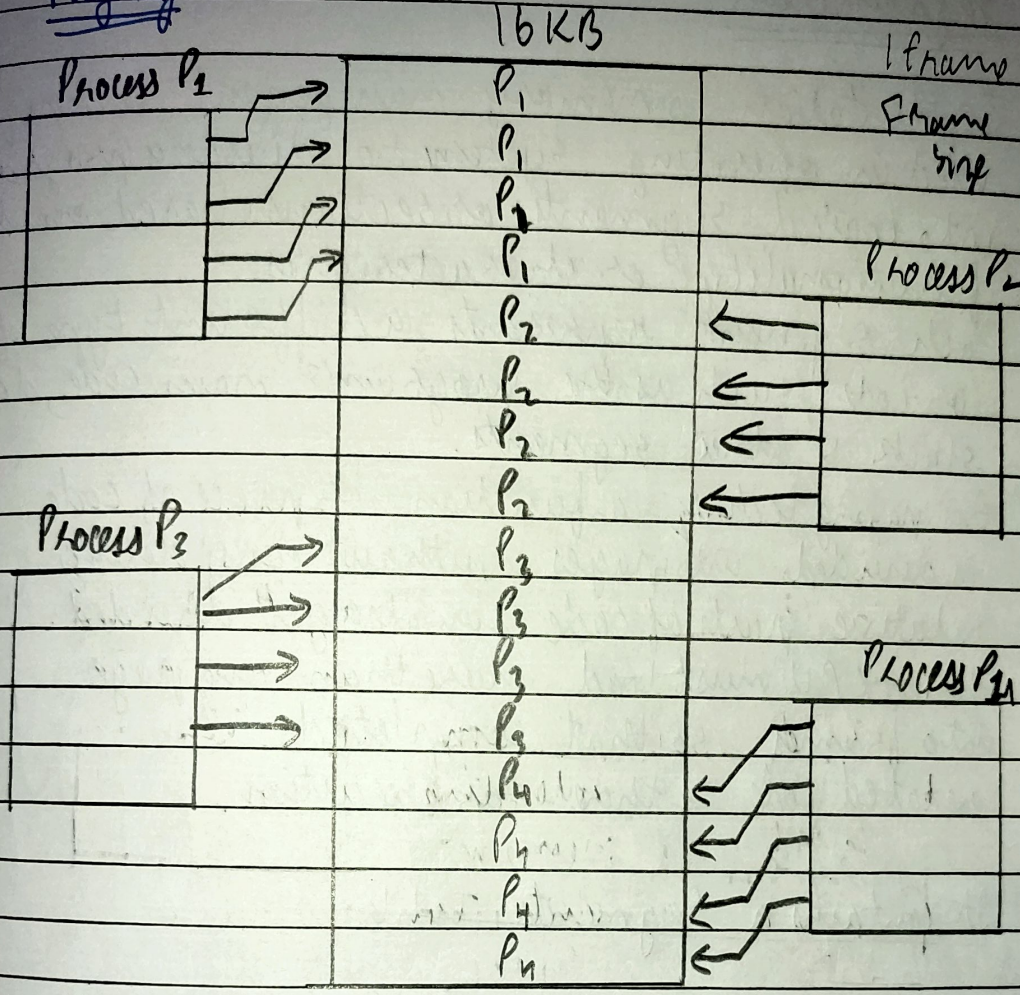
1MB of free space in this block is unused

To avoid internal fragmentation - Dynamic Partitioning.

◊ External Fragmentation

External fragmentation is a memory management issue where free memory blocks are scattered throughout the address space making it difficult to allocate contiguous memory for larger resources. This fragmentation occurs over time as they are loaded & unloaded.

(V) Paging



→ Paging is a storage mechanism used in OS to retrieve processes from secondary storage to main memory as pages.

→ The primary concept behind paging is to break each process into individual pages. Thus primary memory would also be separated into frames.

Assuming that main memory is 16 KB & frame size 1KB the main memory will be partitioned into a collection of 16 (1) KB frames. P_1, P_2, P_3 & P_4 are four processes each of which is 4 KB in size.

(ii) Segmentation

→ Segmentation is memory management technique used in operating system to divide a program into logical segments or sections based on its functionality or characteristics.

→ Each segment represents a different type of data or code, such as the program's main code, data, stack & other segments.

→ In paging technique a function or piece of code is divided into pages without considering relative parts of code can also get divided. Hence CPU must load more than one page into frames so that completed related code is there for execution.

Why segmentation is used?

It contains a Segment Table

- Base address - contains starting physical address where segments reside
- Segment limit - Also known as segment offset.

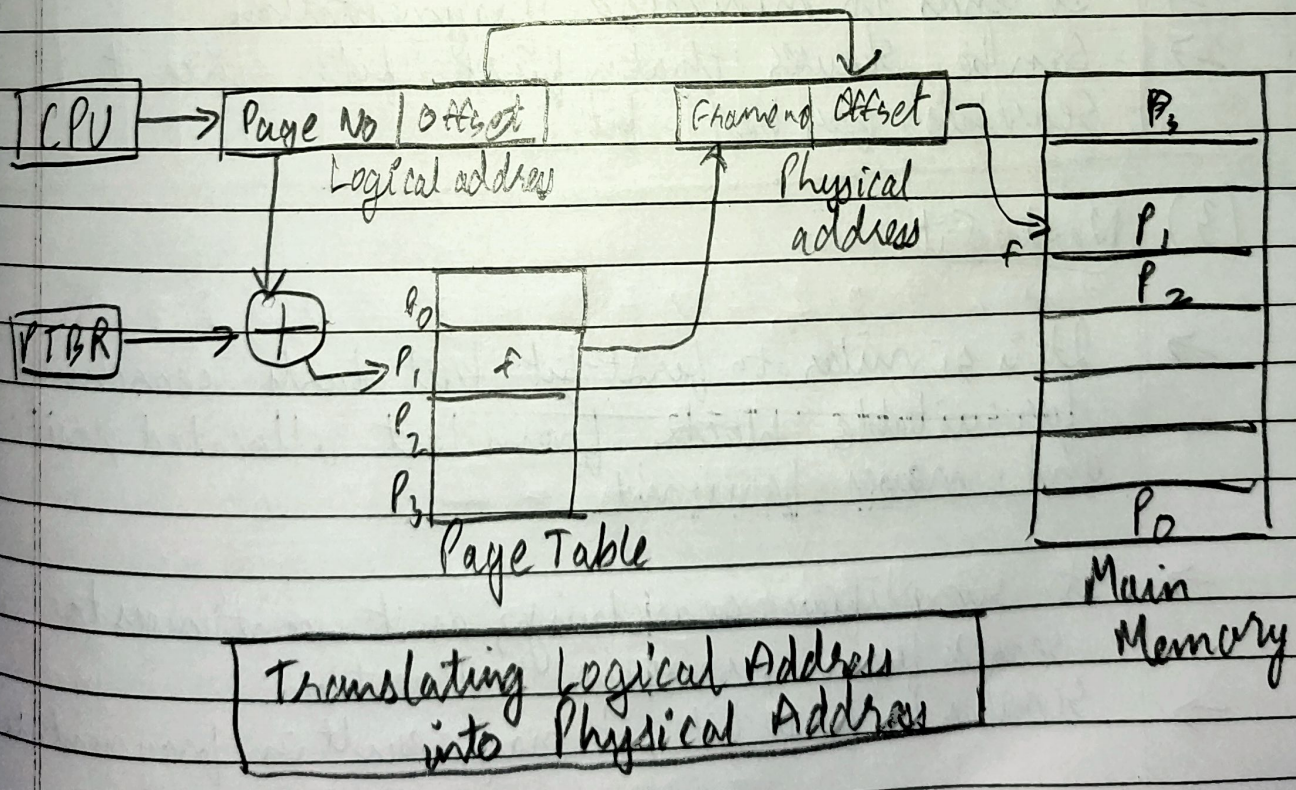
Paging	Segmentation
1) In paging, a program is divided into fixed size pages.	1) The program is divided into variable size sections.
2) Paging is easier to implement & faster.	2) It is slower than Paging
3) Page size is determined by hardware.	3) The section size is given by the user.
4) Paging results in less efficient system.	4) segmentation results in more efficient system.

vii) Address Translation :

→ Address translation in OS refers to the process of converting virtual addresses used by program into physical addresses in computer memory.

→ The operating system, with the help of hardware components like (Memory Management Unit) MMU performs address translation.

→ It uses techniques like paging or segmentation to manage the mapping b/w virtual and physical addresses.



Translating Logical Address into Physical Address

Advantages

- It allows to store parts of a single process in non-contiguous fashion.
- It solves the problem of external fragmentation

Unit 4.2

Placement Strategies

(1) First Fit

→ The first available block that is large enough to accomodate a process is allocated.
→ Simple & easy to implement
→ May lead to fragmentation.

(2) Best Fit (searches whole list)

→ It allocates the smallest available block that is large enough to accomodate a process.
→ It tends to minimise fragmentation.
→ Can be slower than first fit since it searches for best-fit.

(3) Next-Fit

→ It is similar to first-fit but starts searching for suitable block from last allocated position and moves forward.

→ It simplifies searching, as it continues to search from previous allocation.
→ Similar to first fit, it may result in fragmentation.

Efficiency - First & Next fit are faster than best fit due to less searching.
Fragmentation - best fit is designed to minimise fragmentation.

(4) Worst Fit (requires searching for largest available block)

→ It Allocates the largest available block to the process.
→ The idea is to leave largest possible free block for future allocations.
→ It requires searching for largest available block, which can be computationally expensive.
→ It aims to maximise available free space.

Eg

Memory block	Size
1	100 Kb
2	500 Kb
3	200 Kb
4	300 Kb
5	600 Kb

First fit

Process Requests : ~~250Kb~~ 212 Kb
Allocates to - 500 Kb block & stops searching

Next Fit

Process Requests = 250 Kb
Last Allocated block = 500 Kb.
- It will start searching after 500 Kb
- Allocates to - 300 Kb (It is large enough)

Best Fit (search the whole memory block)

Process Requests = 150 Kb.
- ~~Allocator~~ searches whole memory block.
- Allocates to = 200 Kb block.

Worst Fit (searches whole memory for largest partitions)
Requests = 250 Kb
- Allocates to = 600 Kb block.

Unit 4.4

Page Replacement Policies

Strategy used by OS to determine which page to add and remove from main memory (limited no of frames).

(1) FIFO (First In First Out)

→ It replaces the oldest page in memory
→ It uses queue to keep track of the order in which pages were brought into memory.
→ May not consider usage patterns.

f_3			1	1	1	2	0	0	0	3	3	3	2	2	2		
f_2		0	0	0	0	3	3	3	2	2	2	0	1	1	1		
f_1	7	7	7	2	2	2	2	4	4	4	0	0	0	0	0		
	*	*	*	*	Hit	*	*	*	*	*	*	Hit	*	*	Hit		

Reference String: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0

Page hit = 3
Page fault/miss = 12

Hit ratio = $\frac{3}{15} \times 100^{20}$ = 20%

Miss ratio = $\frac{12}{15} \times 100^{20}$ = 80%

(2) LRU (Least Recently used)

→ It replaces the page that has not been used in Longest time.

→ It considers usage patterns, tends to retain more relevant pages.

f_4				2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
f_3			1	1	1	1	3	4	4	4	4	4	4	1	1	1	1	1	1	1
f_2		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
f_1	7	7	7	7	7	3	3	3	3	3	3	3	3	3	3	3	3	7	7	7
	*	*	*	*	Hit	*	Hit	*	Hit	Hit	Hit	Hit	Hit	*	Hit	Hit	Hit	*	*	Hit

Reference String : 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

Page Hit = 12

Page Miss = 8

Hit Ratio = $\frac{12}{20} \times 100$ = 60%

Miss Ratio = $\frac{8}{20} \times 100$ = 40%

(3) Optimal

→ Replaces the page that will not be used for the Longest time in the future.

→ Theoretical optimum for minimising page faults

f_4				2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
f_3			1	1	1	1	1	4	4	4	4	4	4	1	1	1	1	1	1	1
f_2		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
f_1	7	7	7	7	7	3	3	3	3	3	3	3	3	3	3	3	7	7	7	7
	*	*	*	*	Hit	*	Hit	*	Hit	Hit	Hit	Hit	Hit	*	Hit	Hit	Hit	*	Hit	Hit

Ref String:

7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

Page Hits = 12

Page Miss = 8

Hit Ratio = $\frac{12}{20} \times 100$

$= 60\%$

Miss Ratio = $\frac{8}{20} \times 100$

$= 40\%$

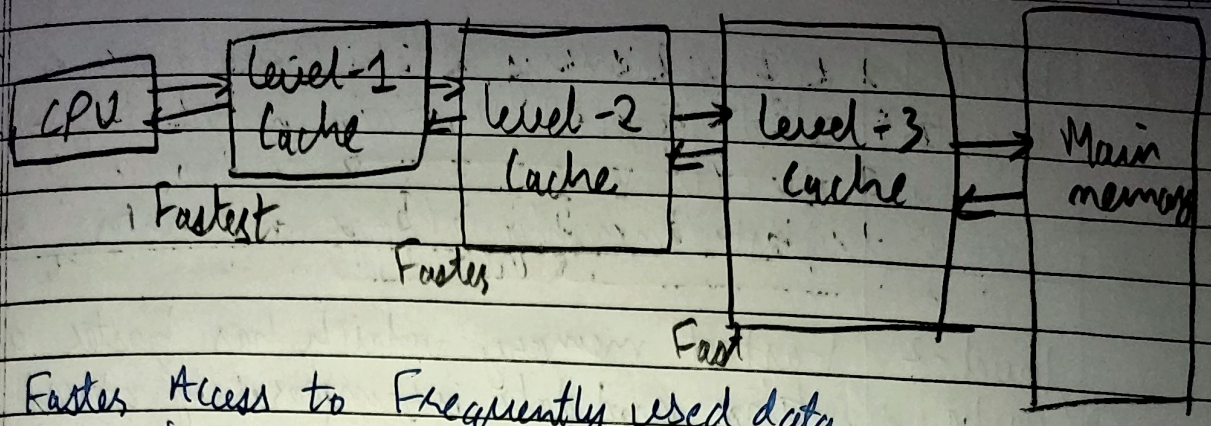
Unit 4.3

Virtual Memory

Virtual Memory is a storage scheme that provides user an illusion of having a very big main memory. This is done by treating a part of secondary memory as the main memory.

◇ Cache-Memory (temperary storage)

It is used to store frequently accessed data for quicker retrieval

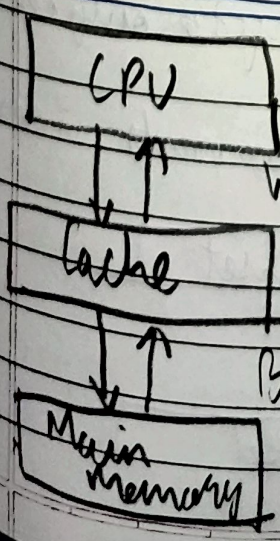


- Fastes Access to Frequently used data.
- Reducing Memory Access time.
- Improving CPU Performance:

By minimising time the CPU has to wait for data, cache memory contributes to improved processor performance.

- Optimizing Memory Hierarchy: Cache memory includes registers, cache, main memory & secondary memory storage.
- Reducing Power Consumption: Accessing data from cache memory requires less energy than fetching it from main memory.

◊ Cache Architecture:



L1 Cache: Closest to CPU, small in size & fastest.

L2 cache: Larger and slightly slower than L1

L3 cache: Shared among multiple cores, larger but slower than L2.

Level-1 (Registers)	type of memory in which data is stored and accepted that are immediately stored in CPU. Most commonly Program Counter, registers
Level-2 (Cache memory)	Fastest memory which has faster access time and data is temporarily stored.
Level-3 (Main Memory)	It is memory on which computer works currently. It is small in size and once power is off data is no longer stays in memory.
Level-4 (Secondary Storage)	It is external memory which is not as fast as main memory but data stays permanently in this memory.

◊ Address Mapping Techniques:

(i) Direct Mapping: [K mod n]

K → Block no in main memory

No of Blocks in (Cache) = Total memory / 4

→ Simplest Technique
→ maps each block of main memory to only one possible cache line.
→ An address space is split into two parts index field & tag field
→ Directly proportional to hit ratio.

Main memory [Tag | Word-offset]

Cache memory. [Tag | line offset | Word-offset]

1 Byte = 8 bits
1 KB = 1024 (Bytes) = 2^{10} Bytes
1 MB = 1024 KB = 2^{20} Bytes
1 GB = 1024 MB = 2^{30} Bytes.

Eg Given Main memory size: 4 GB
Cache size: 1 MB
Block size: 4 KB
Word size = 1 Byte.

Calculate in Cache (direct mapping technique)

Ans Main memory
size = 4 GB = $2^2 \times 2^{30} = 2^{32}$ Bytes.

Block size = 4 KB = $2^2 \times 2^{10} = 2^{12}$ Bytes.
Cache size = 1 MB = 2^{20} Bytes

(a) Physical Address

Size = 2^{32} Bytes

Physical Address Bits = $\log_2(2^{32}) = 32$ bits

(b) Block no Number bits & block offset.

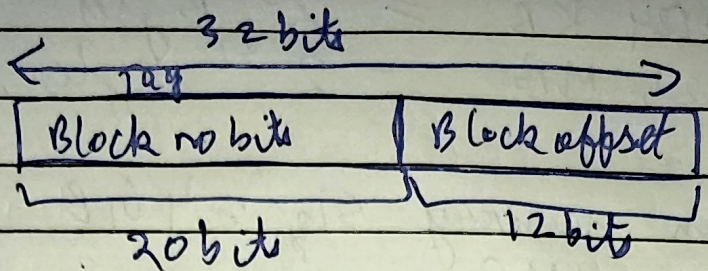
Block size = 2^{12} Bytes.

Block Number bits = $\log_2(2^{12}) = 12$ bits.

Number of blocks = $\frac{\text{Main memory size}}{\text{Block size}} = \frac{2^{32}}{2^{12}} = 2^{20}$

Block offset = $\log_2(2^{12}) = 12$ bits.

(b) Physical Address Split

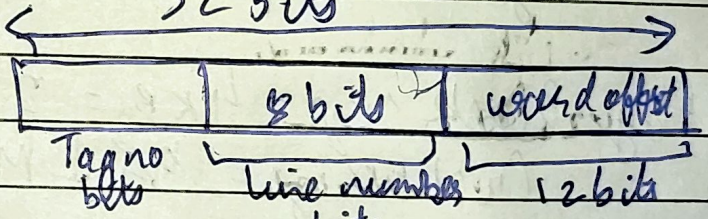


No of blocks $= 2^{20}$

(c) No of lines in Cache

Cache size $= 2^{20}$ Bytes

No of lines in Cache $= \frac{\text{Cache size}}{\text{Block size}} = \frac{2^{20}}{2^{12}} = 2^8$



Line number bits $= \log_2(2^8) = 8$

(d) No of Tag bits in Cache

No of Tag bits = P.A − (line no of bits + offset)

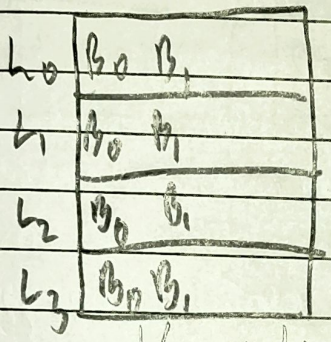
$= 32 - (8 + 12)$

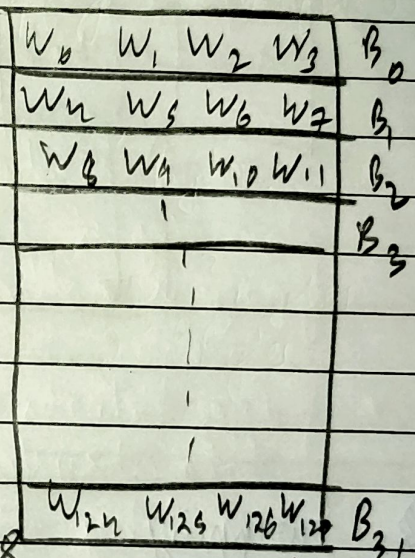
$= 12$ bits

Fully.

◇ Associative Mapping (Line size = block size)

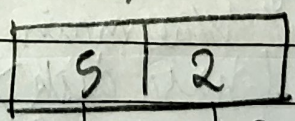
→ The associative mapping is used to store content and addresses of the memory word.
→ Any block can go into any line of cache.
→ It is considered to be the fastest and the most flexible mapping form.
→ This enables the placement of any word at any place in cache memory.





Line number bits = 0.

Size of memory = 2^7 = 128 bytes 128 words
→ 0 - 127



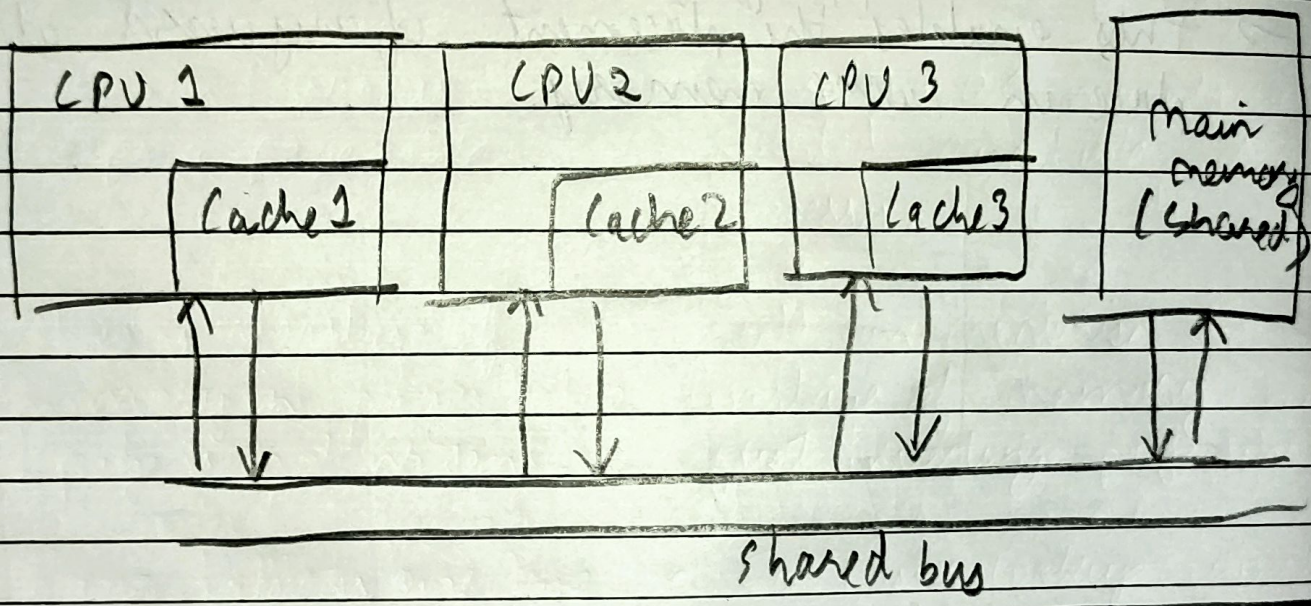
2^2 = 4 words in a block
2^5 = 32 Blocks

◇ Set - Associative Mapping

→ Most popular and widely used
→ enhanced form of direct mapping where drawbacks of direct mapping is removed.
→ Set associative (combine direct and associative mapping techniques to give best results.

Cache Coherence

→ Situation where multiple processors ~~in~~ cores shares the same memory hierarchy but have their own independent data & instruction caches.



Swapping

→ Swapping is a technique used in OS to move processes b/w main memory & secondary memory (hard disk) when there is a need to free up space in RAM.

→ It helps to manage memory efficiently.

Issues:

(a) Performance head- Due to time required to transfer data b/w main memory & secondary memory.

(b) Increased disk I/O.

(c) System Thrashing:

(d) Risk on Data loss - If system experience power failure or unexpected shutdown while a process is swapped out, there is a risk of data loss.

Thrashing:

⇒ In virtual storage system (an O.S that manages its logical storage or memory in units called pages), thrashing is a condition in which excessive paging operations are taking place.

⇒ Here the majority of processor time goes to servicing page faults.

Reasons - Insufficient RAM
Improper page replacement Algo's.

Page Table Structure & Inverted Page Table

Inverted Page Table

Frame	Pid	Page no.
0	1	1
1	1	2
2	2	0
3	2	1
4	1	0
5	2	2

Page Table

Pages	Frame
0	2
1	3
2	5
3	
4	
5	

Process 2

5 2 2

2 0

Process 2 Page 0.

Frame 2

P_2

Page Table	Inverted Page Table
• Used by virtual memory systems (to store the mapping b/w logical & Physical addresses).	• There is an virtual page for each occupied physical memory frame. It can be used to overcome the limitations of Page table.
• For each process the OS keeps a page table.	• The OS maintains an inverted page table for all processes.
• There is wastage of memory in page table if page is not present.	• We can minimise the wastage of memory by just inverting page table.
• Page table is one type of data structure that is used by virtual memory systems.	• For faster lookup, inverted page tables can be implemented using a hash table data structure.
• Logical address: <virtual page no (p), offset (d)>	• Logical address: <pid (if it), virtual page no, pid, frame>

◇ Page Size : The smaller the page size, the lesser amount of internal fragmentation
- however more pages required per process
- more pages means larger page tables.

Page No.
Date

Example Page-Size

Comp	Page size
1) Honeywell-Multics	1024 36-bit words
2) IBM Power	4 Kbytes
3) Pentium	4 K bytes
4) Atlas	512 48-bit words

VM with Paging && VM with Segmentation

Paging	Segmentation
• In paging program is divided into fixed or mounted size pages	• In segmentation the program is divided into variable size sections
• For paging O.S is accountable.	• For segmentation compiler is responsible.
• It is faster in comparison to segmentation.	• It is slower to paging.
• Paging could result in internal fragmentation.	• Segmentation could result in external fragmentation.
• In paging, the logical address is split into a page number & page offset.	• In segmentation logical address is split into section number & section offset.
• Paging is invisible to user as it done by hardware.	• It is visible to user because the section size is given by user.

VM with both combined Paging & Segmentation

It combines the benefit of both paging and segmentation.

(i) Paged Segmentation
(ii) Segmented Paging.

Segmented Paging

In segmented paging, main memory is divided into variable size segments which are further divided into fixed size pages.

→ The logical pages are smaller than segments
→ Each The logical address is represented as Segment Number (base address), Page Number & Page Offset.

- Each page table contains various information about every page of the segment.
- The segment table contains the information about every segment.
- Each segment table entry points to page table & each page table points to pages within segment.

Advantages:

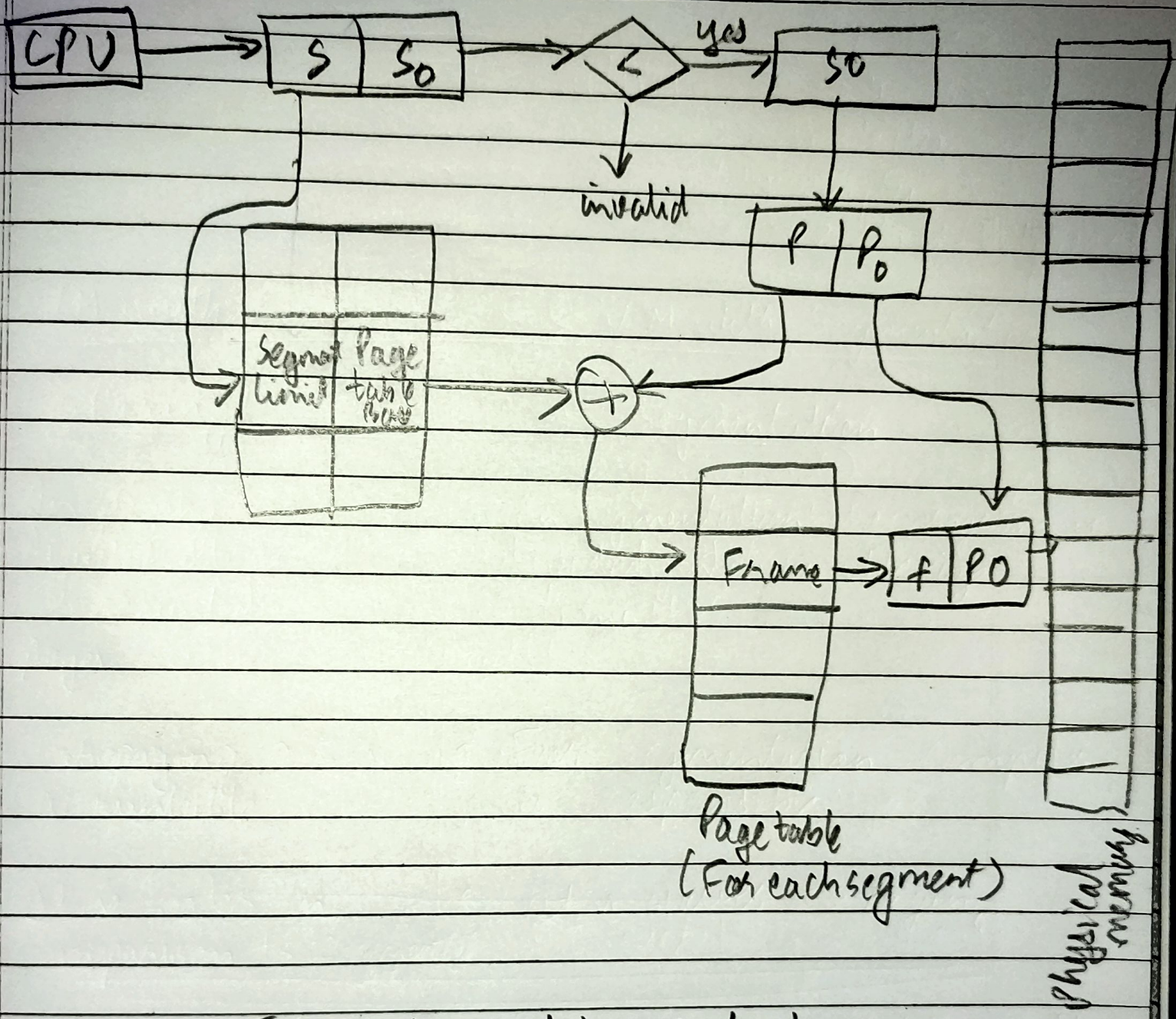
1) Reduces Memory usage
2) External Fragmentation not
3) It simplifies memory allocation

Disadvantages

1) Internal Fragmentation
2) The complexity level will be much higher than to Paging.

S → Segment table
SO → Segment Offset

P → Page Number
PO → Page Offset
f → Frame number



Translation of Logical to Physical Address