Group

Page No.
Date

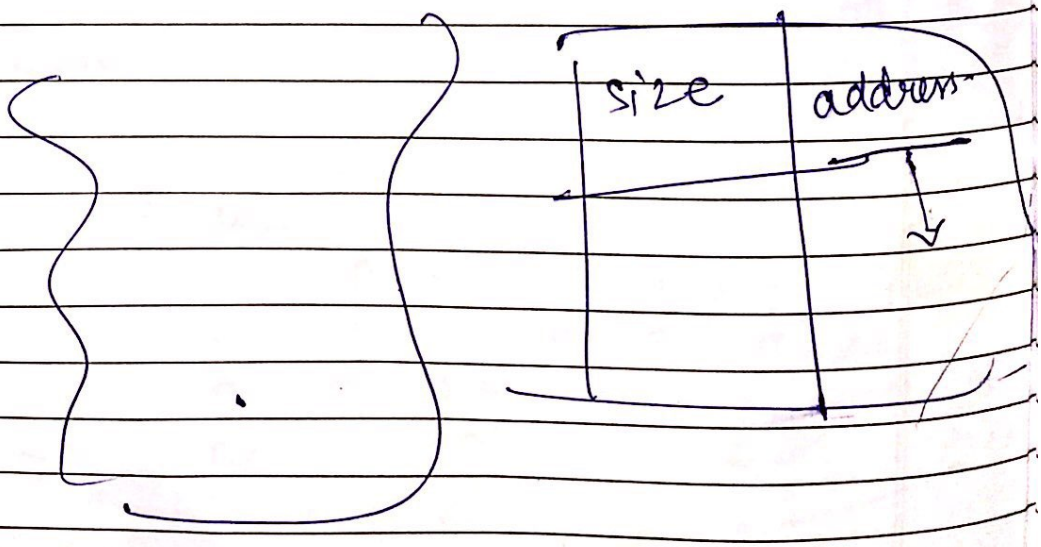
## Segmentation:

The chunks that a program is divided into which are not necessarily all of the same size called segments. In Segmentation.

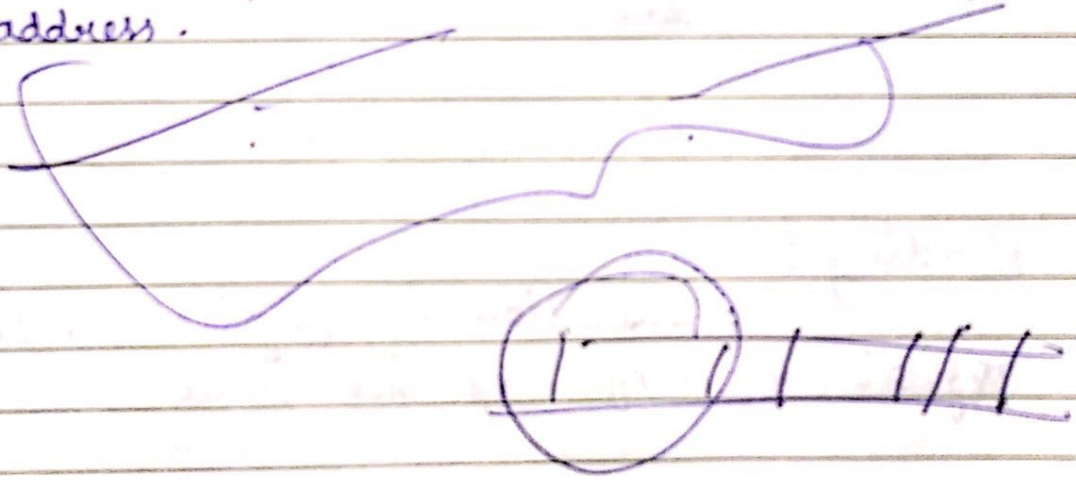
In Main Memory, we only store the segment table of process. This table which keeps the record of segment, its size and memory address.

Advantages:
- → Segmented table occupies less space
- → No internal fragmentation.

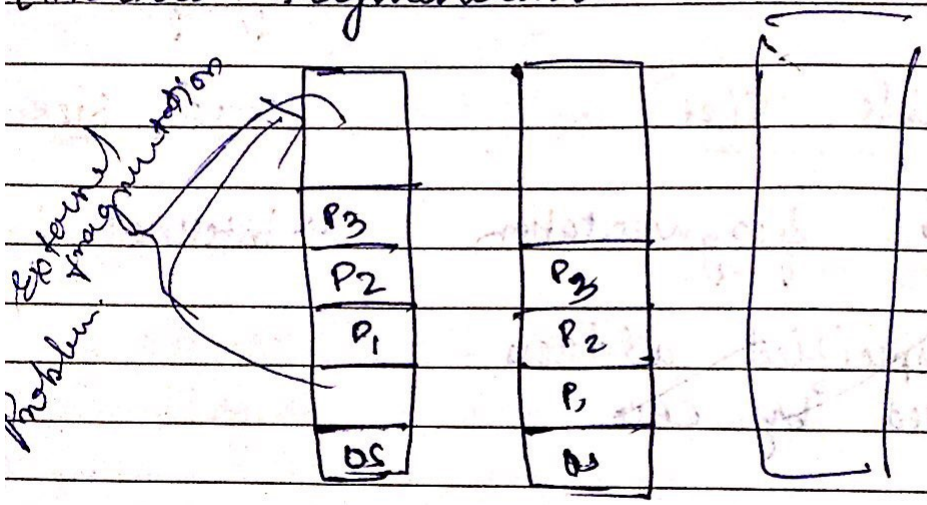
Disadvantages:
- → Due to segments external fragmentation it results in a lot of memory waste.



| Segmentation | Paging |
| --- | --- |
| Variable Size | fixed size |
| External fragmentation | → internal fragmentation |
| user specified address is divided by CPU into | |
| Hardware decides user specified size. | → Hardware decide the size |
| Size and memory address. | → Base address of each page. |



# External Fragmentation




~~Binding~~ :

~~Compile~~

~~Doo~~

## Binding :

translation of physical memory to logical memory or vice versa

- compile
- load
- execution.

ch-4

Page No.
Date

⇒ Garbage Collection:

* Memory Management:

→ Issue that prompt this →
(1) Allocation
(2) swapping, Fragmentation & Compactio

→ Garbage Collection:
The area released by a process is usually not accounted for immediately by the process - its Garbage. compaction or garbage collection is responsibility of the Os.

→ Protection:
checking for illegal access of data from another process memory area.

* Binding:

Address binding of instruction and data to memory address can happen at three different stages.

→ Compile Time:

If memory location known a priori, absolute code can be generated; must recompile code if starting location changes

→ Load Time:

Must generate relocatable code if memory location is ~~not~~ not known at compile time.

→ Execution Time:

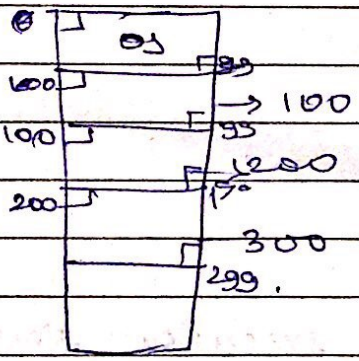
Binding delayed until run time if the process can be moved during its execution from one memory segment to another. Need hardware support for address maps.

# * Memory Management Unit

→ Hardware device that maps virtual to physical address.

→ In MMU scheme, the value in the relocation register is added to every address generated by a user process at the time it it sent to memory.

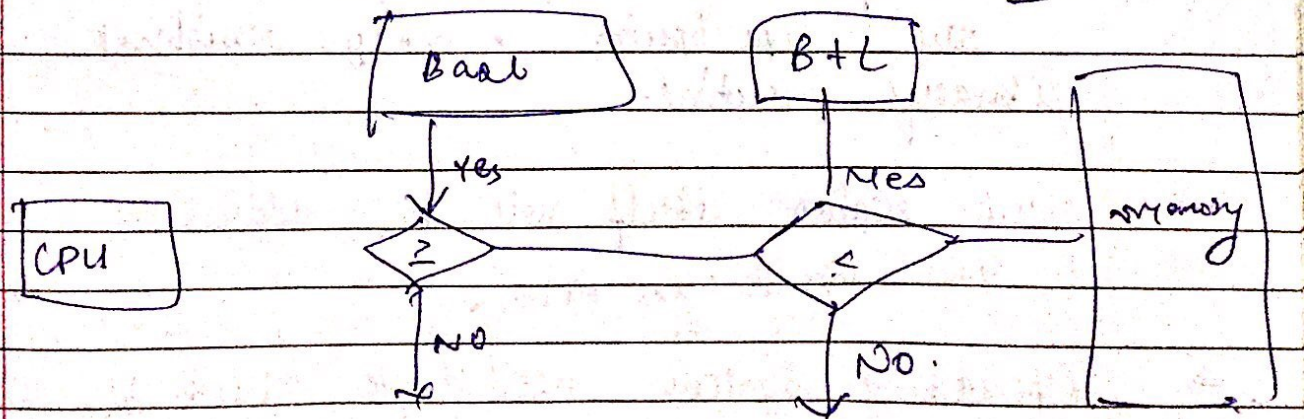
→ The user program deals with logical address, it can never see its physical address.




Base address : 100
Limit = 200.

100 + 199 = 299.

Ans.



* Dynamic Loading:

→ Routine is not loaded until it is called.

→ Better memory space utilization; unused routine is never loaded

→ useful when large amount of codes are needed to handle infrequent occuring cases.

→ NO special support from the operating system is required implemented through program design.

* Dynamic Linking:

→ Linking postponed until execution time

→ Small piece of code, stub, used to locate the appropriate memory resident library routine.

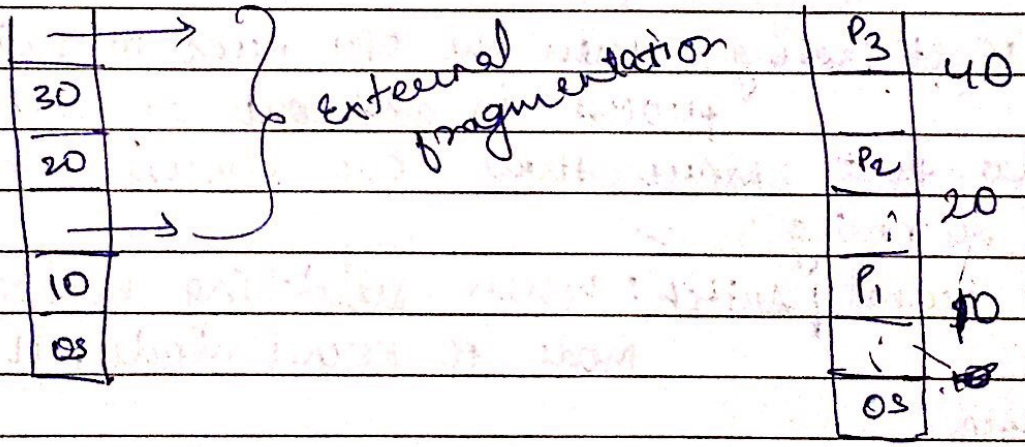
→ Stub replace itself with the address of the routine, and execute the routine.

→ Operating System needed to check if routine is in process memory address.

→ Dynamic linking is particularly useful for libraries.

→ Memory allocation technique
- First fit → 40
- Best fit → 10
- Worst fit → 40
- Next fit → allocated at the same as first fit

Technique use to solve External fragmentation.




Ans 5 KB

* Context Switching :

It occurs when a computer's CPU switches from one process to a different process.

It allows for one CPU to handle numerous process without the need for

additional processor.

A content switch is the mechanism to store and restore the data state or context of a CPU in process Control Block so that a process execution can be resumed from the same point at a later time.

Any operating system that allows for multi tasking relies heavily on the use of content switching to allow different processes to run at the same time.

⇒ Typically, there are three situation that a context switch is necessary

→ Multi tasking: when the CPU needs to switch process in and out of memory, so that more than one process can be running.

→ Kernel switch: When switching between user mode to kernel mode, it may be used.

→ Interrupts: when the CPU is interrupted to retrieve data from a disk read.

⇒ The steps in the full process switch are:

(1) Save the content of the processor, including program counter and other registers.

(2) Update the process control block of the process that is currently in the running state. This includes changing the state of process to one of other states (ready, blocked, suspend, Exit). other relevent fields must also be updated, including the reason for leaving the running state and accounting information.

(3) Move the PCB of the process to appropriate queue (Ready; Blocked on event i; Ready (suspend)

(4) select another process for execution

(5) update the PCB of the process selected. This includes changing the state of the process to Running.

(6) Update Memory Management data structure. This may be required, depending on how address translation is managed.

Unit-4.

* Universal Serial Bus

→ Classification:

1. Human Interaction device class
2. Communication device class
3. Printer device class
4. Mass storage device class

→ Modes of Communication:

1. Interrupt
2. Bulk transfer
3. High Synchronise
4.

→ Three tier Architecture:

* Address Binding: It is the process of mapping the program's logical or virtual addresses to corresponding physical or main memory address.

(OR)

A given logical address is mapped by MMU to a physical address.

* Internal Fragmentation: It occurs when the memory is divided into fixed size blocks, whenever a process request for the memory, the fixed size block is allocated to the process. In case, if the memory assigned to the process is somewhat larger that the memory requested, then the difference between assigned and request memory is known as IF.

* External Fragmentation: It occurs when there is a sufficient amount of space in the memory to satisfy the memory request of a process. But the process's memory request cannot be satisfied as the memory available is in a non-contiguous manner. Either you apply first-fit or best fit memory allocation strategy it will cause EF.

When a process is loaded and removed from the memory the free space creates the hole in the memory space, and there are many such holes in the memory space.

Compaction may be the solution.

Compaction algorithm shuffles all memory contents to one side and free one large block of memory. But Compaction algorithm

Page No.
Date

* Key Difference between IF & EF

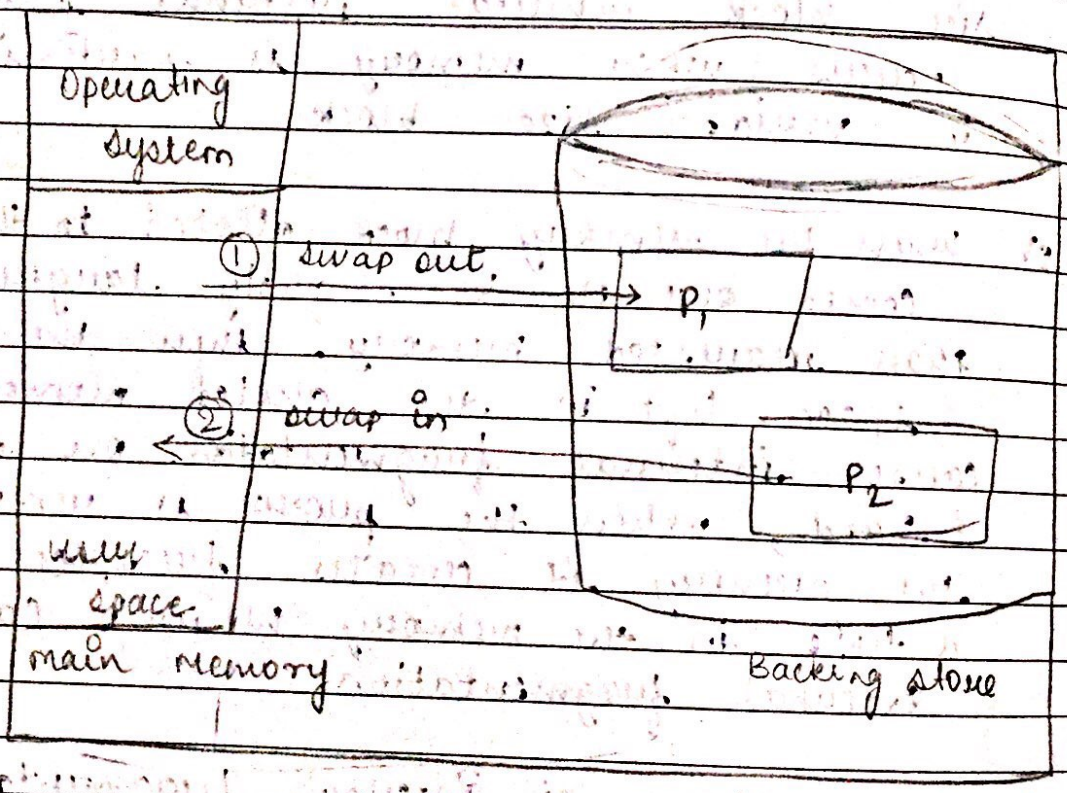
1) The basic reason behind the occurrence of internal and external fragmentation is that internal fragmentation occurs when memory is partitioned into fixed size block, whereas external fragmentation occurs when memory is partitioned in variable size block.

2) When the memory block alloted to the process comes out to be slightly larger than requested memory, then the free space left in the alloted memory block cause internal fragmentation. On other hand, when the process is removed from the memory it creates freespace causing a hole in the memory which is called external fragmentation.

3) The problem of Internal fragmentation can be solved by partioning the memory into variable size block and assign the best fit block to the requesting process. However, the solution for external fragmentation is compaction, but it is expensive to implement, so the process must be allowed to acquire

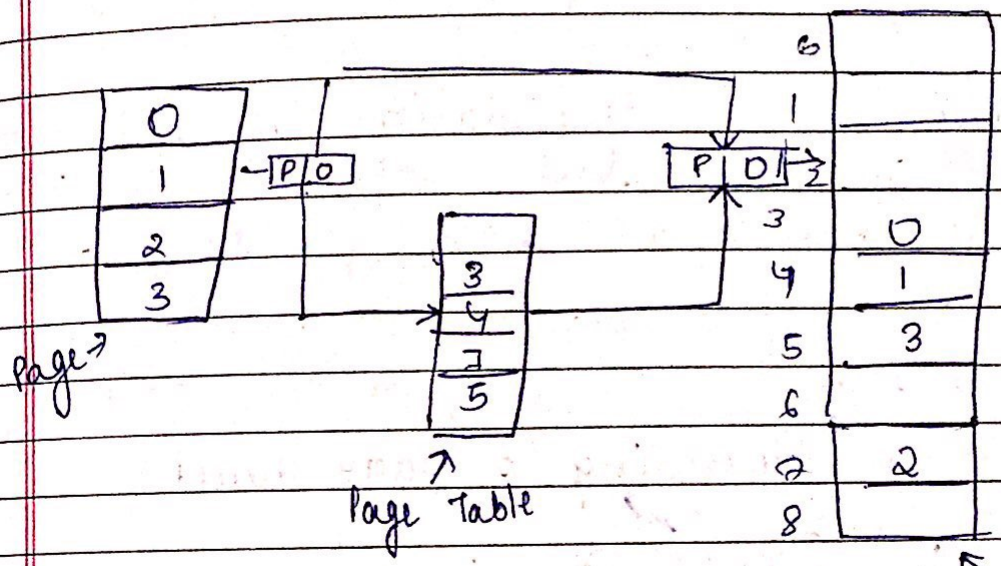
physical memory is non-contiguous manner; to achieve the technique of paging and segmentation is introduced.

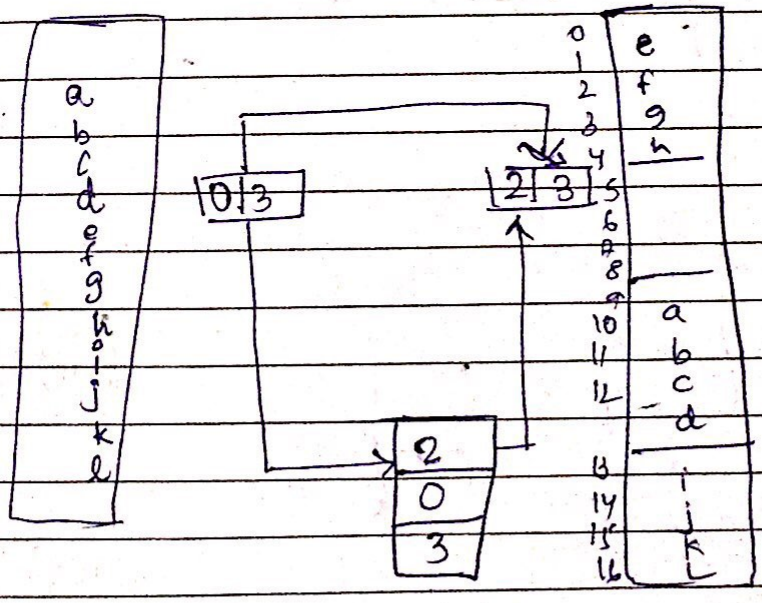
* Swapping :




A process can be swapped temporarily out of memory to a backing store, and then brought back into memory for continued execution.

# * Paging



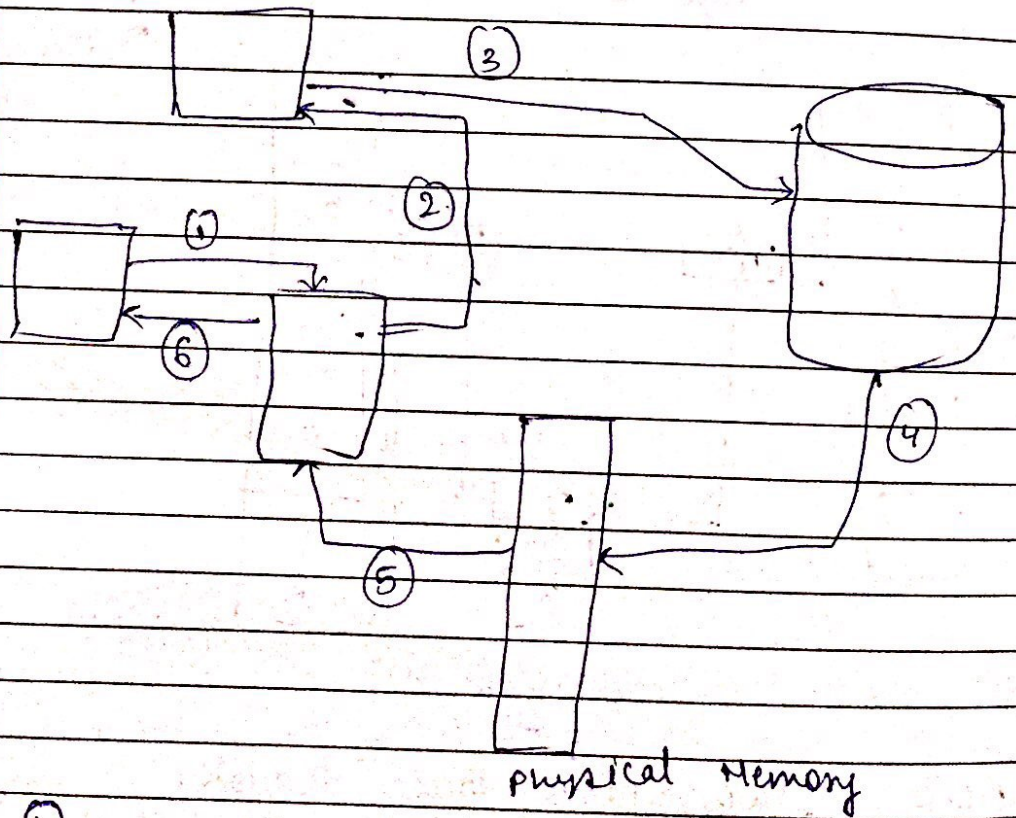




PTBR : Page Table : Base Register
PTLR : Page Table length register
ASID : Address Space Identifier
TLB : Transaction Loop Buffer.

* Short note on Paging & Buddy System.

⇒ During address translation, if valid-invalid bit in page table entry is 0 then it is known as page fault.

⇒ Steps in Handling a page fault




1.
2.
3. Page is on Baking store
4. bring is missing page
5.
6. Restart instruction

* Advantage of Demand Paging

→ Bring a page into memory only when needed

→ less I/O needed
→ less memory needed
→ faster Response
→ More users.

⇒ Page is needed ⇒ reference to it

→ invalid Reference → abort
→ not in memory → bring to memory

* Basic Page Replacement

① Find the location of the desired page on disk
② Find a free frame:
③ Read the desired page, update
④ Restart the instruction.

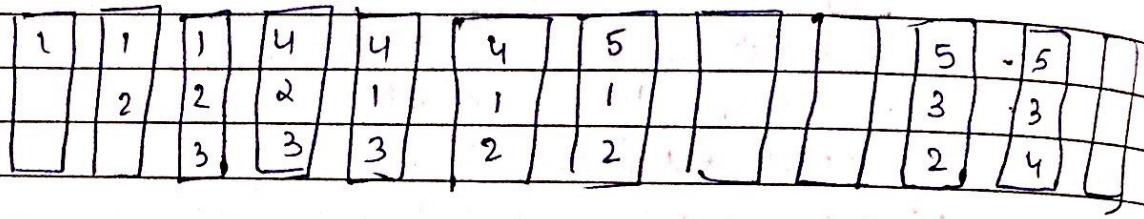
# Algorithm

Page No.
Date

* First In First Out

(1)

1 2 3 4 1 2 5 1 2 3 4 5

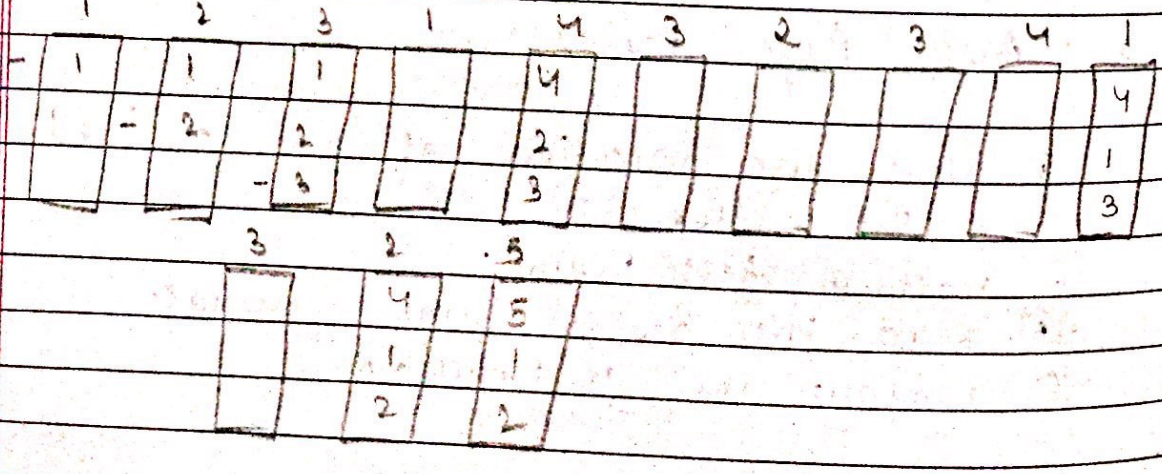


3

Page default = 9.

(2)

1 2 3 1 4 3 2 3 4 1 3 2 5



Page default : 7

(3)

(4)

Page No.
Date

(3)

| 1 | 2 | 3 | 4 | 1 | 2 | 5 | 1 | 2 |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | | | 5 | 5 | 5 |
| | 2 | 2 | 2 | | | 2 | 1 | 1 |
| | | 3 | 3 | | | 3 | 3 | 2 |
| | | | 4 | | | 4 | 4 | 4 |

| 3 | 4 | 5 |
|---|---|---|
| 5 | 4 | 4 |
| 1 | 1 | 5 |
| 2 | 2 | 2 |
| 3 | 3 | 3 |

[Problem of fifo Belady Anomaly.]

Page fault : 10

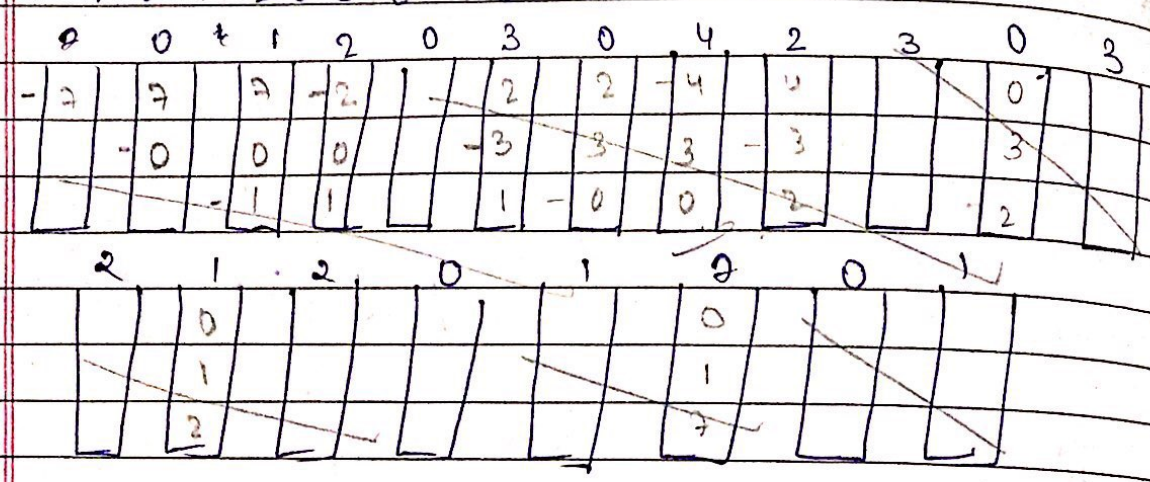
(4)

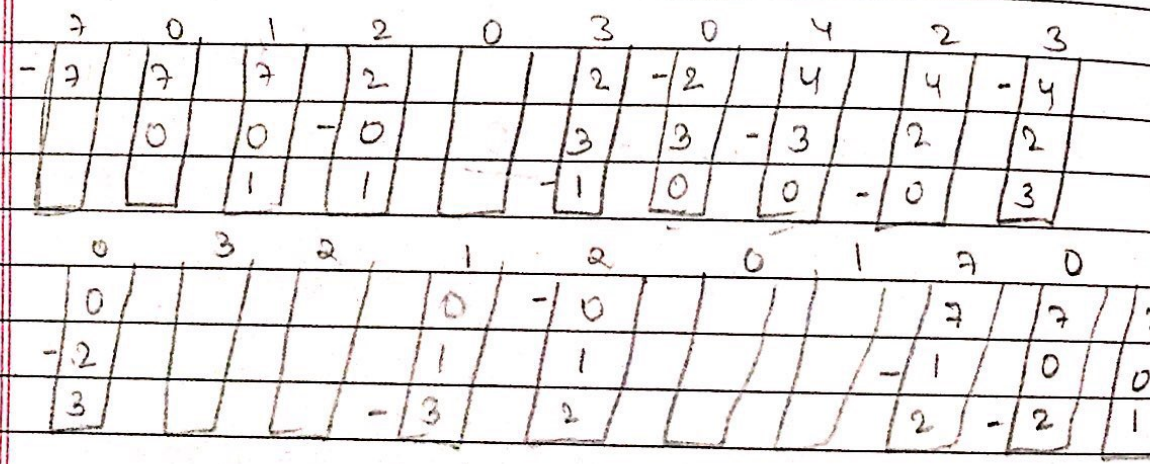
| 1 | 2 | 3 | 1 | 4 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | | 1 | | | |
| | 2 | 2 | | 2 | | | |
| | | 3 | | 3 | | | |
| | | | | 4 | | | |

| 4 | 1 | 3 | 2 | 5 |
|---|---|---|---|---|
| | | | | 5 |
| | | | | 2 |
| | | | | 3 |
| | | | | 4 |

Page fault : 5.

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







Page Default

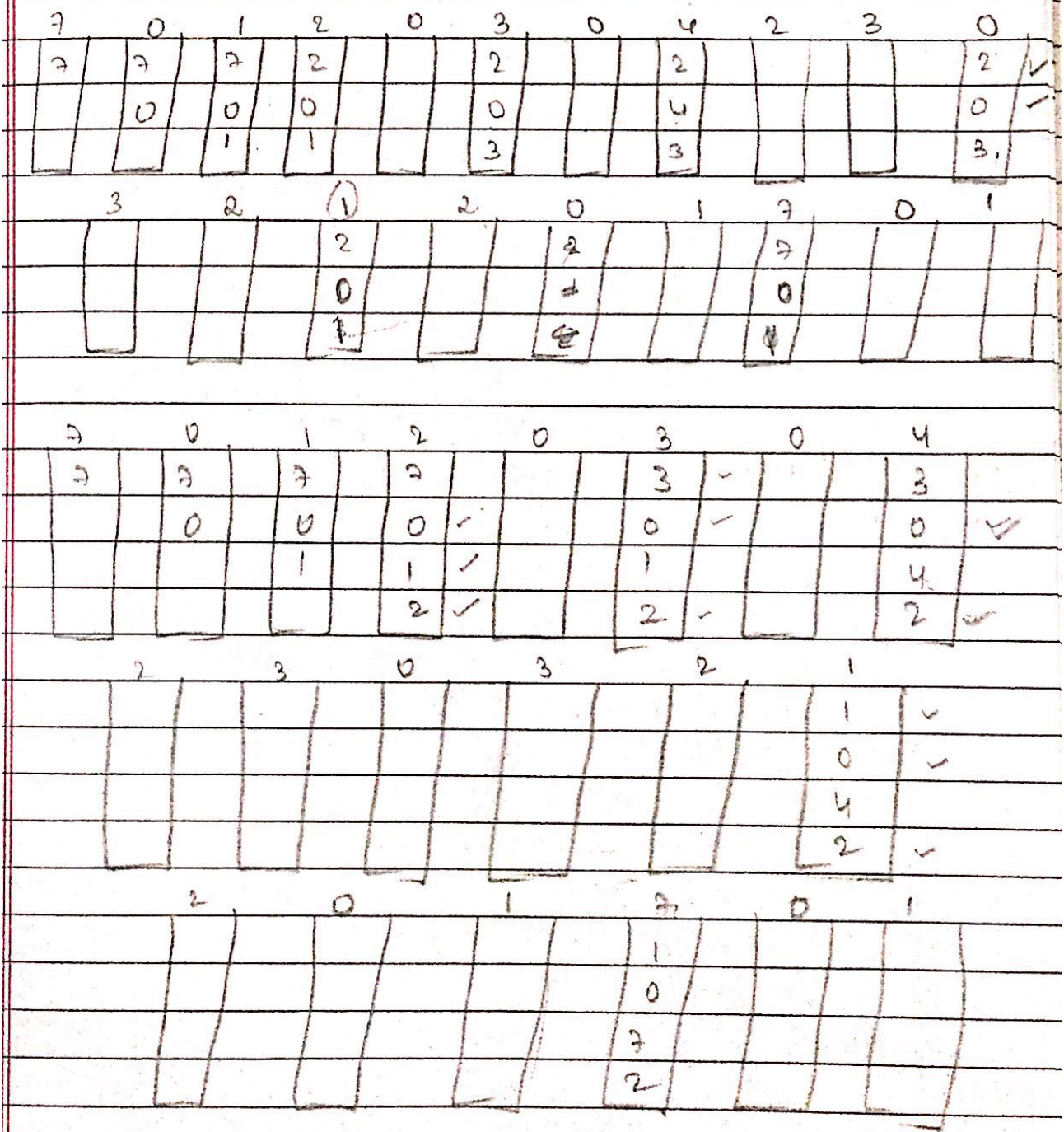
15 - 3.

10 - 4

09 - 5

Page No.
Date

# Optimal Page Replacement




Page fault : 8