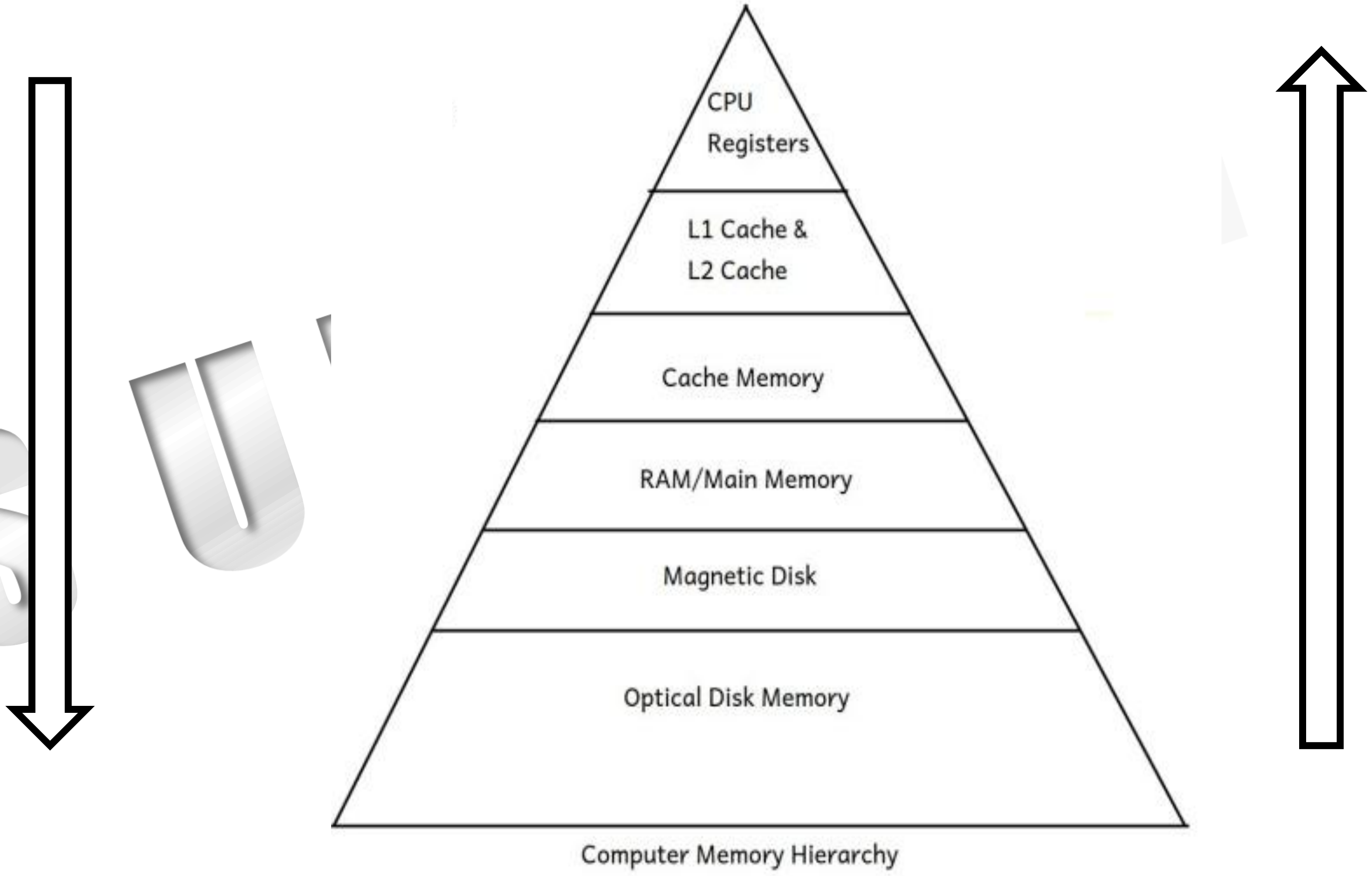
Memory Management

➢Computer Memory Technologies

As we go Top to Bottom
1.Acses speed decrease
2.Coast also decrease
3.Capacity increase



➢Computer Memory Technologies:

- **CPU Registers:** memory which is very close to the CPU are registers which is at the top in a computer memory hierarchy.
- Instructions and data currently executing by the CPU can be kept temporarily into the CPU registers.

➢Computer memory can be categorized into two categories as per its location:

- **Internal Memory & External Memory.** - Internal Memory: memory which is internal to the motherboard is referred as an internal memory.

  - e.g. CPU registers, L1 & L2 cache Cache memory, RAM.

- **External Memory:** memory which is external to the motherboard is referred as an external memory.

  - e.g. magnetic disk, optical disk, magnetic tape etc...

➢Computer Memory Technologies:

- Computer memory can also categorized into two categories**: Primary Memory & Secondary Memory**.

- **Primary Memory:** memory which can be accessible directly by the CPU is referred as primary memory, i.e. memory which can accessible by the CPU with the help of instruction set having with the CPU.

- e.g. CPU registers, L1 & L2 Cache, Cache Memory, RAM

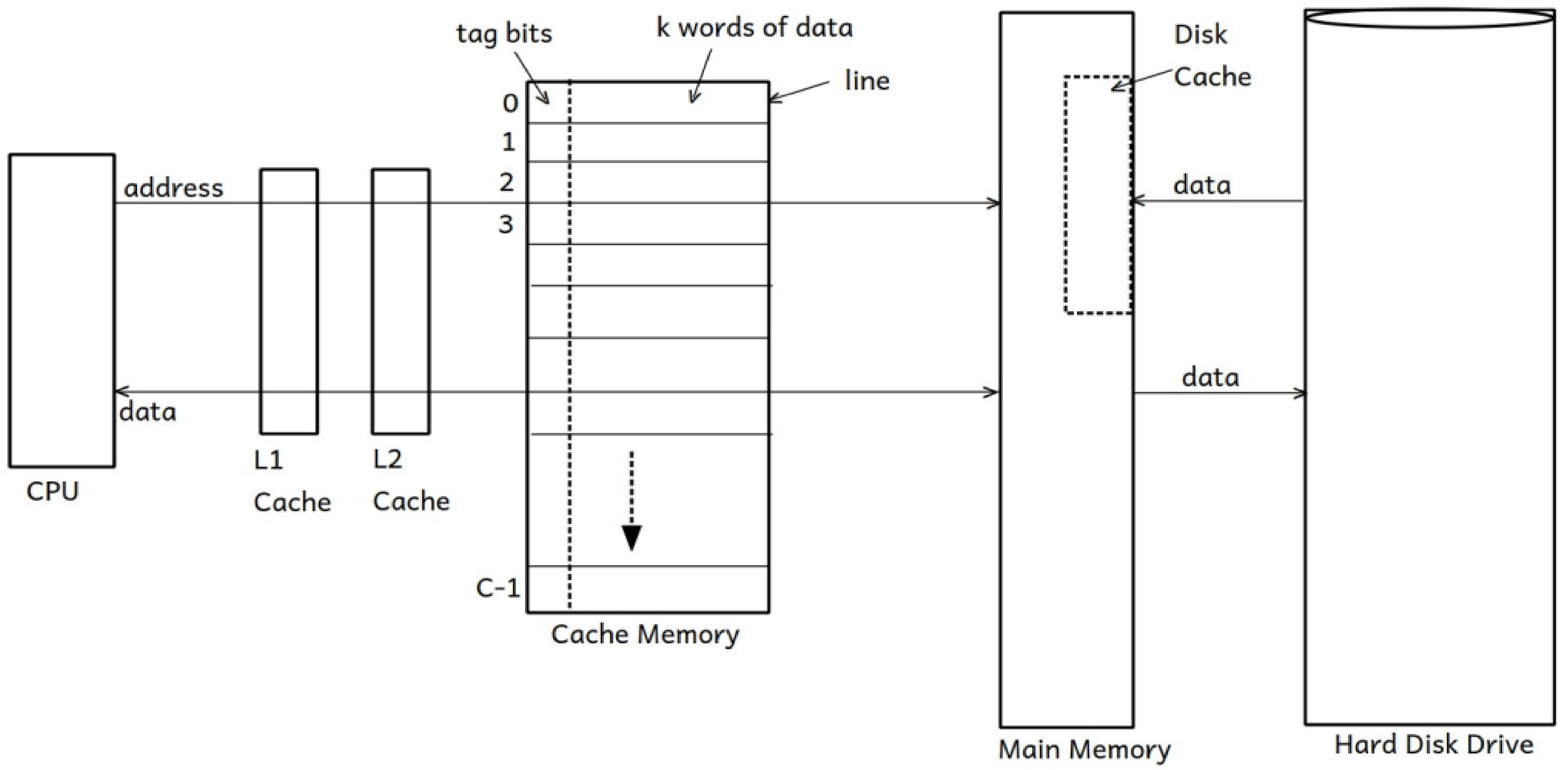
- **Secondary Memory:** memory which cannot be accessed directly by the CPU is referred as secondary memory.

- e.g. Magnetic Disk, CD/DVD, PD etc..
- If the CPU want to access disk contents, first it gets fetched into the RAM and then it can be accessed by the CPU from RAM.

- As for an execution of every program RAM memory is must and hence RAM is also called **as Main memory.**

➢ Why there is a need of cache memory?

- As the rate at which the CPU can execute instructions is faster than the rate at which data can be accessed from the main memory, so even the CPU is very fast, with the same speed data do not gets fetched from the main memory for execution, hence due to this speed mismatch overall system performance gets down.
- To reduce speed match between the CPU and the main memory Cache memory (hardware) can be added between them and system performance can be increases by means **reducing speed mismatch.**


tag bits
k words of data
line
0
1
2
3
C-1
Cache Memory
CPU
address
data
L1
Cache
L2
Cache
Disk
Cache
data
data
Main Memory
Hard Disk Drive

➢Memory Management

❖. Why there is a need of memory (main memory)management ?

- As main memory is must for an execution of any program and it is a limited memory, hence an OS manages main memory.
- To achieve maximum CPU utilization, an OS must support **multitasking, and to support multi-tasking multiple processes** must be submitted into the system at a time i.e. it must support multiprogramming, but **as main memory is limited** to support multiprogramming an **OS has to do memory management to complete an execution of all submitted processes.**
- **Memory space of one process should gets protected from another process**.

- Addresses generated by compiler (i.e. compiler + linker) are referred as **logical addresses.**
- Addresses which can be see by the process when it is in the main memory referred as **physical addresses**.
- **MMU (Memory Management Unit):** which is a hardware unit converts logical address into physical address.
- **MMU is a hardware** contains adder circuit, comparator circuit, base register and limit register. Values of base register and limit registers gets change during context-switch, and memory space **of one process gets protected from another process**.
- CPU always executes program in its **logical memory space**.

➢Memory Allocation:

- When a process is requesting for the main memory, there are two methods by which memory gets allocated for any process

1. Contiguous Memory Allocation

2. Non - contiguous Memory Allocation

1. Contiguous Memory Allocation:

- Under this method, process can complete its execution only if memory gets allocated for it in a contiguous manner.
- There are two methods by which memory gets allocated for process under contiguous memory allocation method.

1. Fixed Size Partitioning

2. Variable Size Partitioning

1. Fixed Size Partitioning:

- In this method, physical memory i.e. main memory (user space) is divided into fixed number of partitions and size of each partition is remains fixed.
- If any process is requesting for the memory it can be loaded into main memory in any free partition in which it can be fit.

❖ Advantages:

- This method is simple to implement.

❖Disadvantages:

- **Internal fragmentation**: if memory remains unused which is internal to the partition.
- **Degree of multi-programming** is limited to the number of partitions in the main memory.
- **Maximum size of a process is limited** to max size partition in the main memory.
- To overcome limitations/disadvantages of fixed size partitioning method, variable/dynamic size partitioning method has been designed.

2. Variable/Dynamic Size Partitioning:

- In this method, initially whole user space i.e. physical memory is considered as a single free partition, and processes gets loaded into the main memory as they request for it.
- Size of partition and number of partitions are not fixed in advanced, it gets decided dynamically.
- Memory is allocated to each process as per its availability in the RAM.
- After **allocation** and **deallocation** of few processes, RAM will have few used slots and few free slots.
- OS keep track of free slots in form of a table. For any new process, OS use one of the following mechanism to allocate the free slot.
  - **First Fit**: Allocate first free slot which can accommodate the process.
  - **Best Fit**: Allocate that free slot to the process in which minimum free space will remain.
  - **Worst Fit**: Allocate that free slot to the process in which maximum free space will remain.
- Statistically it is proven that First fit is faster algo; while best fit provides better memory utilization.
- Memory info (physical base address and size) of each process is stored in its PCB and will be loaded into MMU registers (base & limit) during context switch.
- CPU request virtual address (address of the process) and is converted into physical address by MMU as shown in diag.
- If invalid virtual address is requested by the CPU, process will be terminated.

❖Advantages:

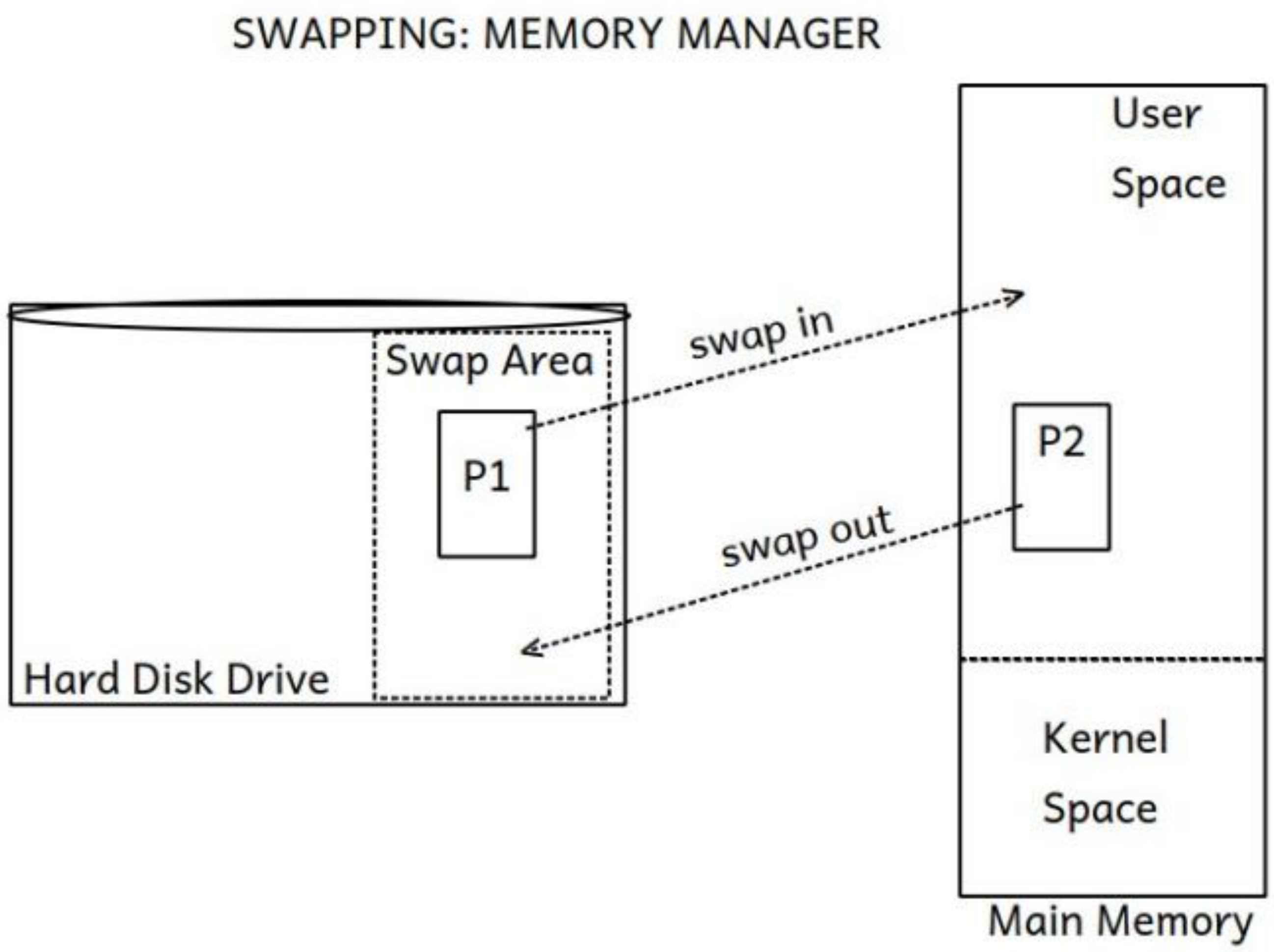
- There are very less chances of **internal fragmentation** - Degree of multi-programming is not limited/fixed
- Size of the process is not also limited, any size process may get load into the main memory.

❖Disadvantages:

- **External fragmentation**: due to loading and removing of processes into and from the main memory, main memory gets fragmented i.e. it gets divided into used partitions and free partitions. In such case, if any new process is requesting for the memory and even if the requested size of memory is available, but due to unavailability of the memory in a contiguous manner request of that process cannot be accepted, this problem is referred as an external fragmentation.
- External fragmentation is the biggest problem under contiguous memory allocation, and hence these solve using compaction.
- **Compaction**
  - o shuffling of main memory can be done in such a way that all used partitions can be shifted to one side and all free partitions can be shifted to other side and contiguous large free partition will be made available for the new processes.
  - o Compaction is practically not feasible as there is need to do recalculations of addresses every time.

Virtual Memory

- The portion of the hard disk which is used by OS as an extension of RAM, is called as "**virtual memory".**
- If sufficient RAM is not available to execute a new program or grow existing process, then some of the inactive process is shifted from main memory (RAM), so that new program can execute in RAM (or existing process can grow).
- It is also called as "**swap area" or "swap space".**
- Shifting a process from RAM to swap area is called as "swap out" and shifting a process from swap to RAM is called as "swap in".
- In few OS, swap area is created in form of a partition. E.g. UNIX, Linux, ...
- In few OS, swap area is created in form of a file E.g. Windows (pagefile.sys), ...
- **Virtual memory advantages:**
  - Can execute more number of programs.
  - Can execute bigger sized programs.


SWAPPING: MEMORY MANAGER
User Space
Swap Area
P1
swap in
P2
swap out
Hard Disk Drive
Kernel Space
Main Memory

❖ Non- contiguous Memory Allocation:

- Under this method, process can complete its execution even if memory gets allocated for it in a non - contiguous manner, and it can be achieved by two memory management techniques:

  1. Segmentation

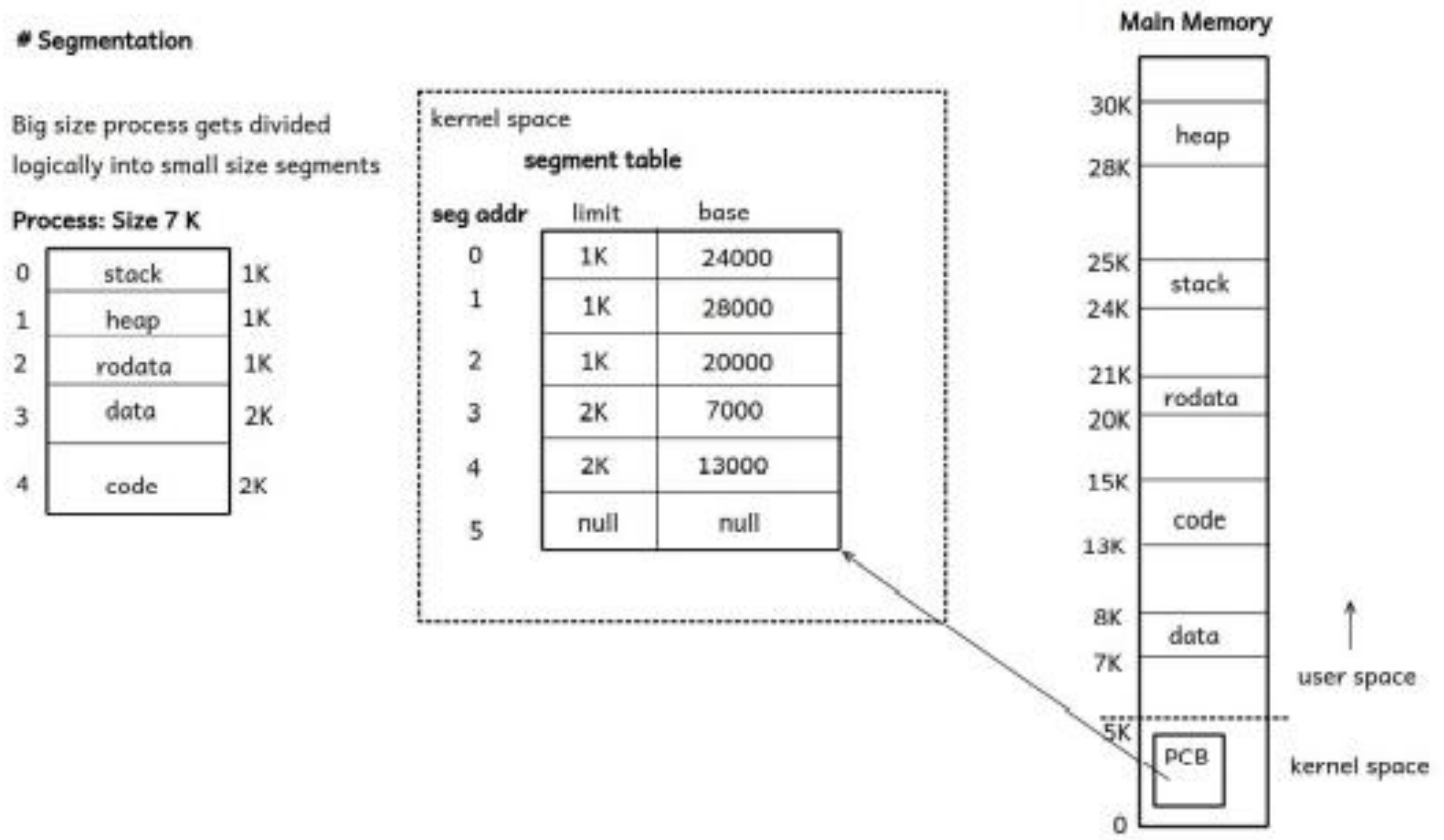
  2. Paging

- So by using segmentation & paging techniques, process can complete its execution even after memory gets allocated for it in a **non- contiguous manner.**

1. Segmentation

- Instead of allocating contiguous memory for the whole process, contiguous memory for each segment can be allocated. This scheme is known as "s**egmentation**".
- Since process does not need contiguous memory for entire process, external fragmentation will be reduced.
- In this scheme, PCB is associated with a segment table which contains base and limit (size) of each segment of the process. During context switch these values will be loaded into MMU segment table.
- CPU request virtual address in form of segment address and offset address.
- Based on segment address appropriate base-limit pair from MMU is used to calculate physical address as shown in diag.
- MMU also contains STBR (Segment Table Base Register) register which contains address of current process's segment table in the RAM.

Operating Systems Concepts



# Segmentation

Big size process gets divided logically into small size segments

Process: Size 7 K

0	stack	1K
1	heap	1K
2	rodata	1K
3	data	2K
4	code	2K

kernel space

segment table

seg addr	limit	base
0	1K	24000
1	1K	28000
2	1K	20000
3	2K	7000
4	2K	13000
5	null	null

❖Demand Segmentation

- If virtual memory concept is used along with segmentation scheme, in case low memory, OS may swap out a segment of inactive process.
- When that process again start executing and ask for same segment (swapped out), the segment will be loaded back in the RAM. This is called as "**demand segmentation**".
- Each entry of the segment table contains base & limit of a segment. It also contains additional bits like segment permissions, valid bit, dirty bit, etc
- If segment is present in main memory, its entry in segment table is said to be valid (v=1).
- If segment is swapped out, its entry in segment table is said to be invalid (v=0).

2. Paging :

- In this technique, physical memory (i.e. user space of a main memory) is divided into fixed size of blocks referred as **frames**, and process's logical memory space is divided into same size of blocks referred as **pages,** whereas maximum size of page must be equal to size of frame, i.e.
- **if e.g. size of frame = 4K, then maximum size of each page must be 4K, size of page may be less than 4K.**
- As process is divided into pages, so when it is requesting for memory, pages of one process may gets loaded into the main memory at any free frames, and for a process memory gets allocated in a non - contiguous manner.
- One page is allocated to one empty frame.
- OS keep track of free frames in form of a linked list
- There is no external fragmentation in paging.
- Internal fragmentation may exists in paging when the size of page is less than size of frame.

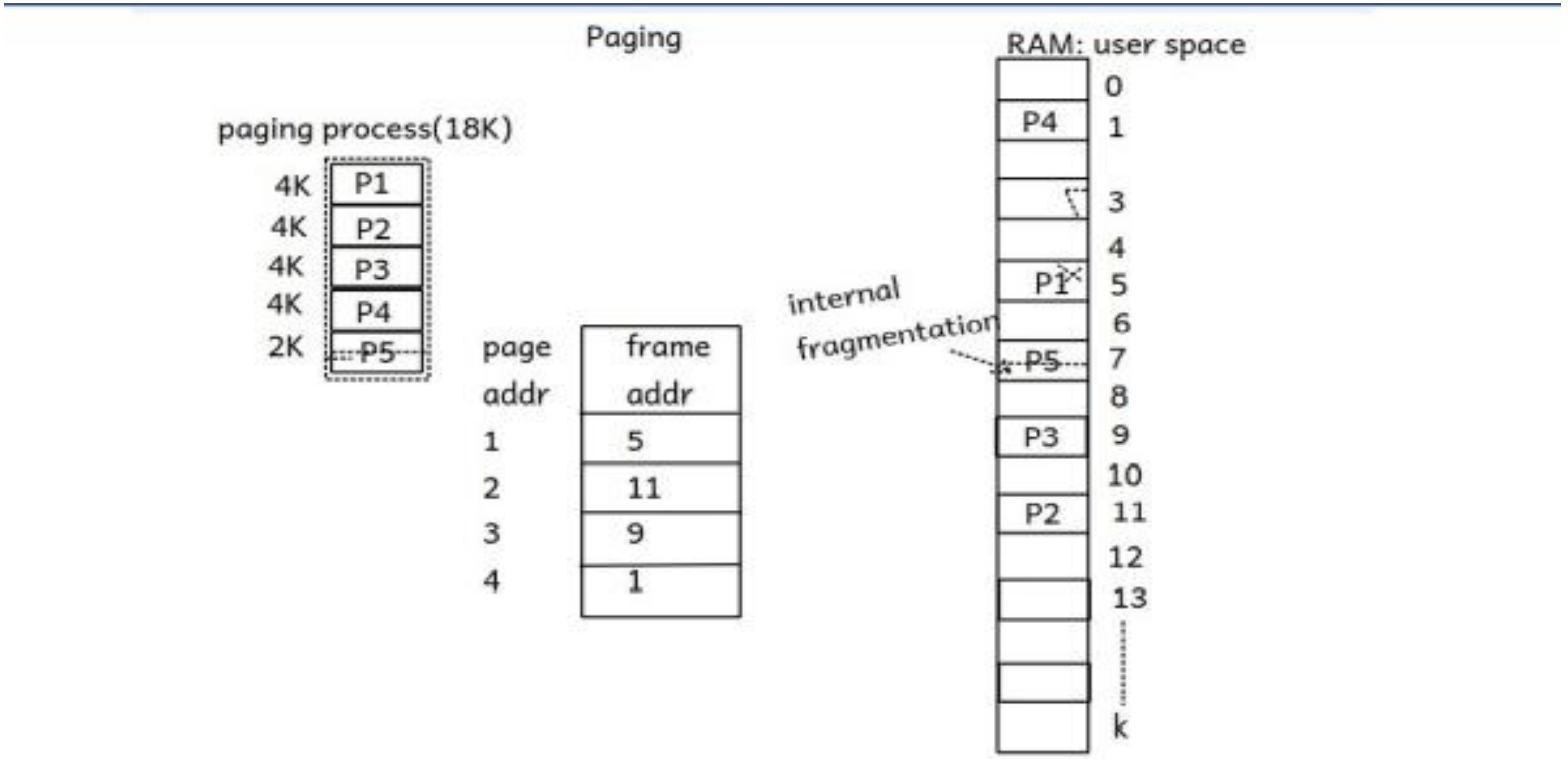
- As pages of a one process gets loaded randomly into the main memory, and in a system thousands processes are running at a time, so to keep track on all the pages of each process, an OS maintains one table per process referred as **a page table** in which information about all the pages of that process can be kept.
- During context switch this table is loaded into MMU.
- CPU requests a virtual address in form of page address and offset address.
- It will be converted into physical address as shown in diag. MMU also contains a PTBR, which keeps address of page table in RAM.
- Frame size can be configured in the hardware. It can be 1KB, 2KB or 4KB, ...
- **Page table entry**
  - Each PTE is of 32-bit (on x86 arch) and it contains
    - Frame address
    - Permissions (read or write)
    - Validity bit
    - Dirty bit ...

- **TLB (Translation Look-Aside Buffer)**
- Cache TLB is high-speed associative cache memory used for address translation in paging MMU.
- TLB has limited entries (e.g. in P6 arch TLB has 32 entries) storing recently translated page address and frame address mappings.
- The page address given by CPU, will be compared at once with all the entries in TLB and corresponding frame address is found.
- If frame address is found (TLB hit), then it is used to calculate actual physical address in RAM .
- If frame address is not found (TLB miss), then PTBR (Page Table Base Register)is used to access actual page table of the process in the RAM (associated with PCB).
- Then page-frame address mapping is copied into TLB and thus physical address is calculated.
- If CPU requests for the same page again, its address will be found in the TLB and translation will be faster.

❖Demand Paging

- When virtual memory is used with paging memory management, pages can be swapped out in case of low memory.
- The pages will be loaded into main memory, when they are requested by the CPU. This is called as "demand paging".
    - Swapped out pages
    - Pages from program image (executable file on disk)
    - Dynamically allocated pages
- **Virtual pages vs Logical pages**
    - By default all pages of user space process can be swapped out/in.
    - This may change physical address of the page. All such pages whose physical address may change are referred as "Virtual pages".
    - Few kernel pages are never swapped out.
    - So their physical address remains same forever. All such pages whose physical address will not change are referred as "Logical pages"

Operating Systems Concepts



Page Fault

- Each page table entry contains frame address, permissions, dirty bit, valid bit, etc.
- If page is present in main memory its page table entry is **valid** (**valid bit = 1**).
- If page is not present in main memory, its page table entry is **not valid (valid bit = 0).**
- This is possible due to one of the following reasons:
  - Page address is not valid (dangling pointer).
  - Page is on disk/swapped out.
- **If CPU requests a page that is not present in main memory (i.e. page table entry valid bit=0), then "page fault" occurs.**

- **Then OS's page fault exception handler is invoked, which handles page faults as follows:**
  1. Check virtual address due to which page fault occured. If it is not valid (i.e. dangling pointer), terminate the process (sending SEGV signal). (Validity fault).
  2. Check if read-write operation is permitted on the address. If not, terminate the process (sending SEGV signal). (Protection fault).
  3. If virtual address is valid (i.e. page is swapped out), then locate one empty frame in the RAM.
  4. If page is on swap device or hard disk, **swap in** the page in that **frame**.
  5. Update **page table entry** i.e. add new frame address and valid bit = 1 into PTE.
  6. Restart the instruction for which page fault occurred.

Page Replacement Algorithms

- While handling page fault if no empty frame found (step 3), then some page of any process need to be swapped out. This page is called as "**victim" page**.
- The algorithm used to decide the victim page is called as "**page replacement algorithm".**
- **There are three important page replacement algorithms.**
    1. **FIFO**
    2. **Optimal**
    3. **LRU**

1. FIFO

- The page brought in memory first, will be swapped out first.
- Sometimes in this algorithm, if number of frames are increased, number of page faults also increase.
- This abnormal behaviour is called as **"Belady's Anomaly**“

2. **OPTIMAL**
   - The page not required in near future is swapped out.
   - This algorithm gives minimum number of page faults.
   - This algorithm is not practically implementable.
3. **LRU**
   - The page which not used for longer duration will be swapped out.
   - This algorithm is used in most OS like Linux, Windows, ...
   - LRU mechanism is implemented using "stack based approach" or "counter based approach".
   - This makes algorithm implementation slower.
   - **Approximate LRU algorithms** provide **nearly the same performance** as true LRU but are **simpler and faster** to implement in operating systems.

- **Dirty Bit**
- Each entry in page table has a dirty bit.
- When page is swapped in, dirty bit is set to 0.
- When write operation is performed on any page, its dirty bit is set to 1.
- It indicate that copy of the page in RAM differ from the copy in swap area.
- When such page need to be swapped out again, OS check its dirty bit.
- If bit=0 (page is not modified) actual disk IO is skipped and improves performance of paging operation.
- If bit=1 (page is modified), page is physically overwritten on its older copy in the swap area.

➢Thrashing:

- If any process spends more time on paging rather than execution, then this high paging activity is referred as thrashing.
- If number of programs are running in comparatively smaller RAM, a lot of system time will be spent into page swapping (paging) activity.
- Due to this overall system performance is reduced.
- The problem can be solved by increasing RAM size in the machine

SUNBEAM

Thank you!

Kiran Jaybhave

email – kiran.jaybhave@sunbeaminfo.com