# Operating Systems

# Quiz

Q. Critical Section Problem can be resolved by using

A. Binary Semaphore

B. Mutex Object

C. Classic Semaphore

D. Both A & B

E. None of the above

Q. Which of the following signal an OS send to a process for forceful termination?

A. SIGTERM

B. SIGEND

C. SIGSTOP

D. SIGKILL

# Quiz

Q Processes which shares data with another processes referred as

A. related processes

B. cooperative processes

C. independent processes

D. all of the above

E. none of the above

Q. Which of the following ipc mechanism is used for communication across the systems?

A. Pipe

B. message queue

C. chatting application

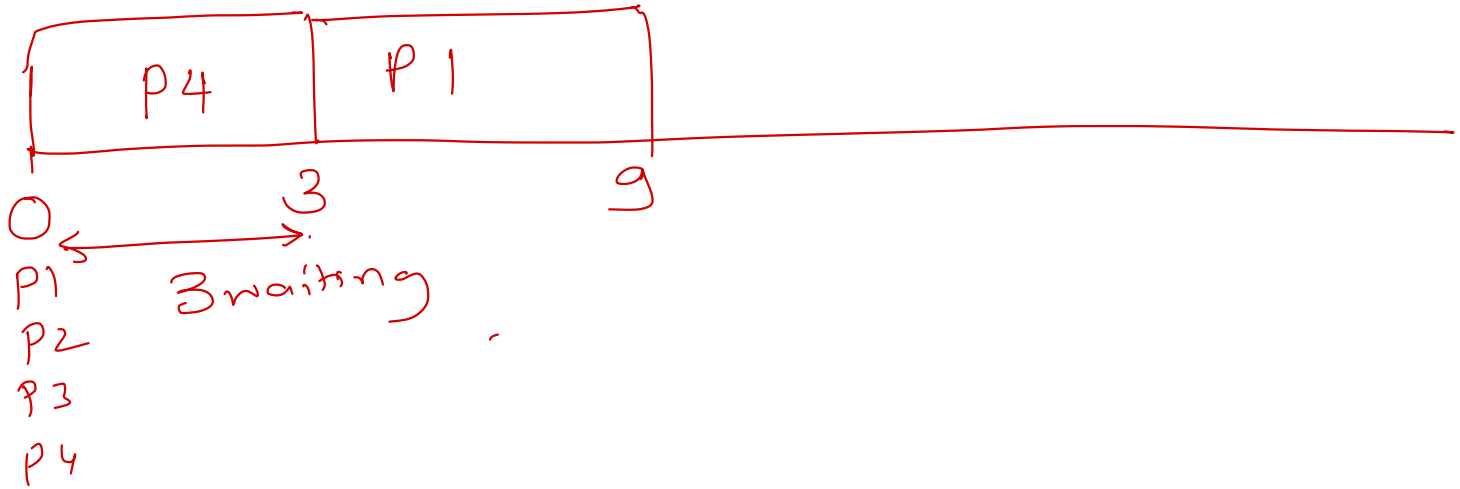
D. socket

E. shared memory model

# Quiz

Q. Consider the following set of processes, the length of the CPU burst time given in milliseconds.

| Process | Burst time |
|---|---|
| P1 | 6 |
| P2. | 8 |
| P3 | 7 |
| P4 | 3 |




Assuming the above process being scheduled with the SJF scheduling algorithm.

a) The waiting time for process P1 is 3ms
b) The waiting time for process P1 is 0ms
c) The waiting time for process P1 is 16ms
d) The waiting time for process P1 is 9ms

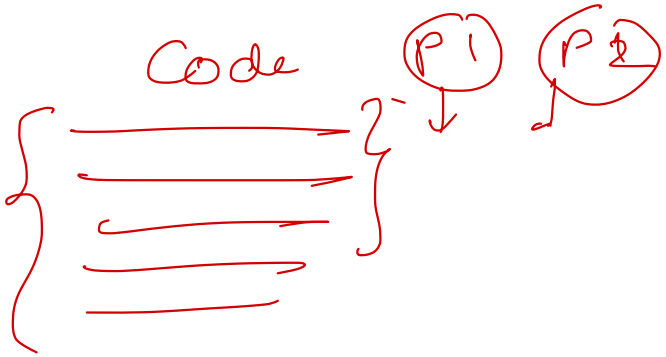
# Quiz

Q. What is ‘Aging’?

a) keeping track of cache contents
b) keeping track of what pages are currently residing in memory
c) keeping track of how many times a given page is referenced
d) increasing the priority of jobs to ensure termination in a finite time

Q. The segment of code in which the process may change common variables, update tables, write into files is known as _____

a) program
b) critical section
c) non – critical section
d) synchronizing



# Quiz

Q. If the semaphore value is negative _____

a) its magnitude is the number of processes waiting on that semaphore

b) it is invalid

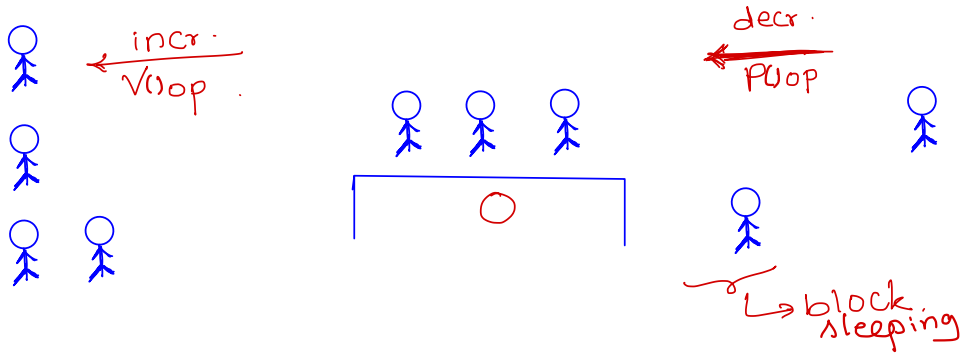
c) no operation can be further performed on it until the signal operation is performed on it

d) none of the mentioned

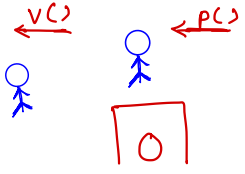
# Semaphore

- Semaphore was suggested by  Dijkstra scientist (dutch math)
- Semaphore is a counter
- On semaphore two operations are supported:
- wait operation: decrement op: P operation:
  - semaphore count is decremented by 1.
  - if cnt < 0, then calling process is blocked(block the current process).
  - typically wait operation is performed before accessing the resource.
- signal operation: increment op: V operation:
  - semaphore count is incremented by 1.
  - if one or more processes are blocked on the semaphore, then wake up one of the process.
  - typically signal operation is performed after releasing the resource.
- Q. If sema count = -n, how many processes are waiting on that semaphore?
  - Answer: "n" processes waiting

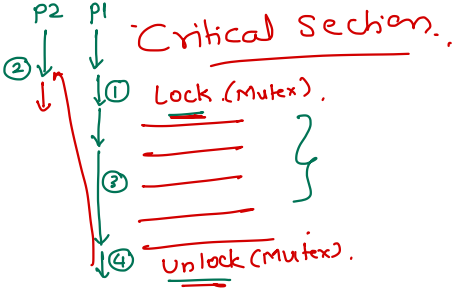
# Semaphore




Semaphore
- Multiple processes can access a resource
  - Counting/Classic.
- Binary (0 or 1).
  - When single process can access a resource.




- Mutex (Mutual Exclusion): when only one process access resource. Like binary Semaphore. Mutex has two states ie. Unlocked or locked



# Semaphore

- **Counting Semaphore/classic**
  - When multiple processes can access a resource.
  - Allow "n" number of processes to access resource at a time.
  - Or allow "n" resources to be allocated to the process.
- **Binary Semaphore**
  - When a single process can access a resource at a time.
  - Allows only 1 process to access resource at a time or used as a flag/condition

# Mutex

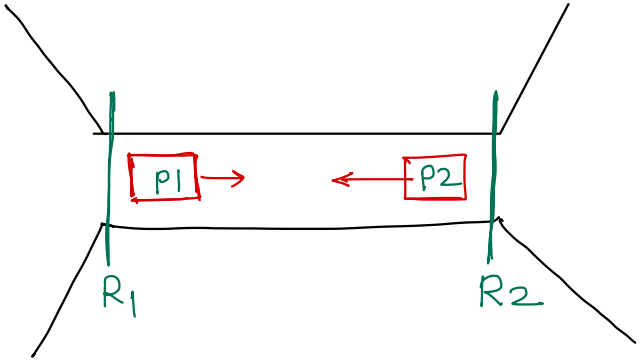
**Mutex( Mutual Exclusion)**

- When only one process access resource like binary semaphore
- Mutex has two states ie. Unlocked or locked state.
- Rule of mutex is that which process has lock the mutex can only unlock the mutex.

**Semaphore vs Mutex**

- S: Semaphore can be decremented by one process and incremented by same or another process. M: The process locking the mutex is owner of it. Only owner can unlock that mutex.
- S: Semaphore can be counting or binary.
  M: Mutex is like binary semaphore. Only two states: locked and unlocked.
- S: Semaphore can be used for counting, mutual exclusion or as a flag.

  M: Mutex can be used only for mutual exclusion.




## Deadlock:

## Characteristic Deadlock.

1. No preemptive
2. Mutal condition.
3. Hold and wait.
4. Circular wait.

# Deadlock

- The processes are blocked indefinitely in deadlock
- Deadlock occurs when four conditions/characteristics hold true at the same time.
  - No preemption: A resource should not be released until task is completed.
  - Mutual exclusion: Resources is not sharable.
  - Hold & Wait: Process holds a resource and wait for another resource.
  - Circular wait: Process P1 holds a resource needed for P2, P2 holds a resource needed for P3 and P3 holds a resource needed for P1.

- **Deadlock Prevention**
  - OS system are designed so that at least one deadlock condition is never true.
  - This will prevent deadlock

.

# Deadlock

## Deadlock Avoidance

- **The processes should inform system about the resources it needs later. OS will accept or reject resource request based on deadlock possibility.**
- **Algorithms used for this are:**
  - Resource allocation graph: OS maintains graph of resources and processes. A cycle in graph indicate circular wait will occur. In this case OS can deny a resource to a process.
  - Banker's algorithm: A bank always manage its cash so that they can satisfy all customers.


- **Deadlock Recovery**
  - Deadlock can be solved by resource preemption or terminating one of the process (involved in deadlock).

  Resource pre-emption -> forcibly withdraw resources given to the processes.

  Process termination -> forcibly kill one of the process.

# Deadlock

## Starvation vs Deadlock

- Starvation: The process not getting enough CPU time for its execution.
  - Process is in ready state/queue.
    Reason: Lower priority (CPU is busy in executing high priority process).
- Deadlock: The process not getting the resource for its execution.
  - Process is in waiting state/queue indefinitely.
    Reason: Resource is blocked by another process (and there is circular wait).

Memory
↳ Storage data → 1 & 0 → Binary → 2.

Size.

→ $2^3 = 8$

1 Byte = 8 bits.

Primary

CPU directly.

CPU register, Cache, RAM.

Secondary

Primary mem → Secondary.

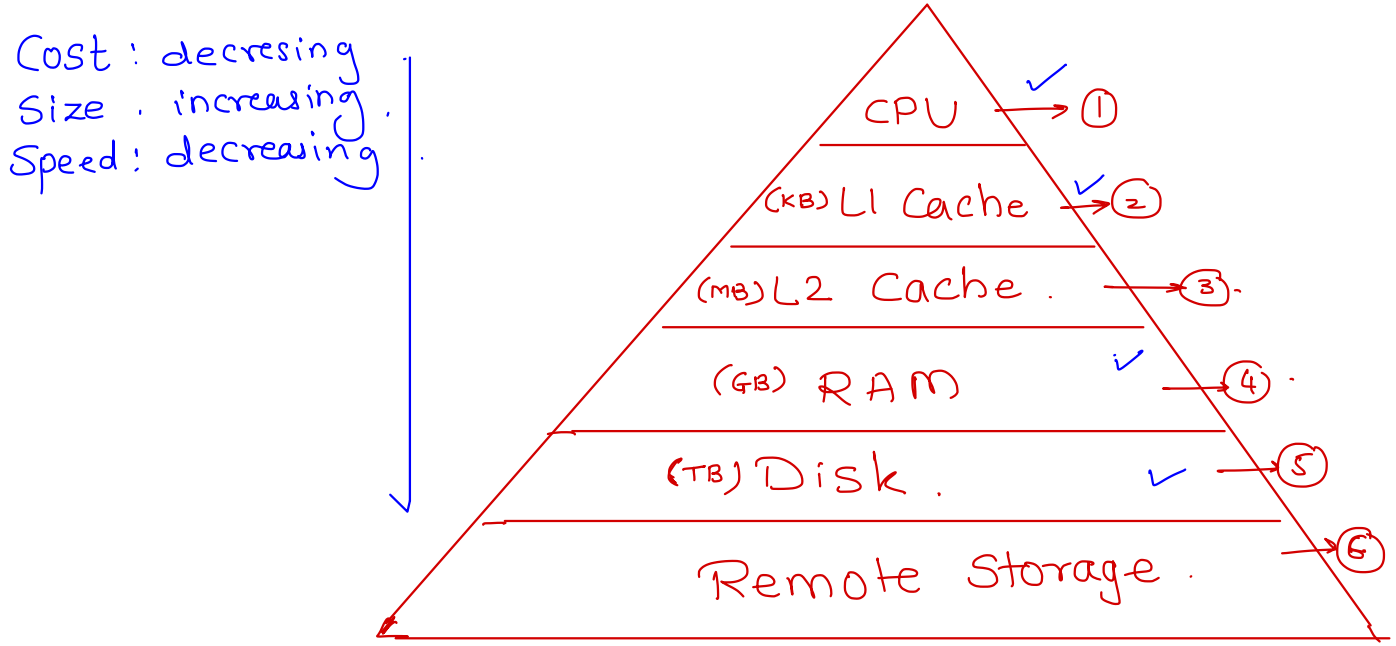
HDD, ROM.

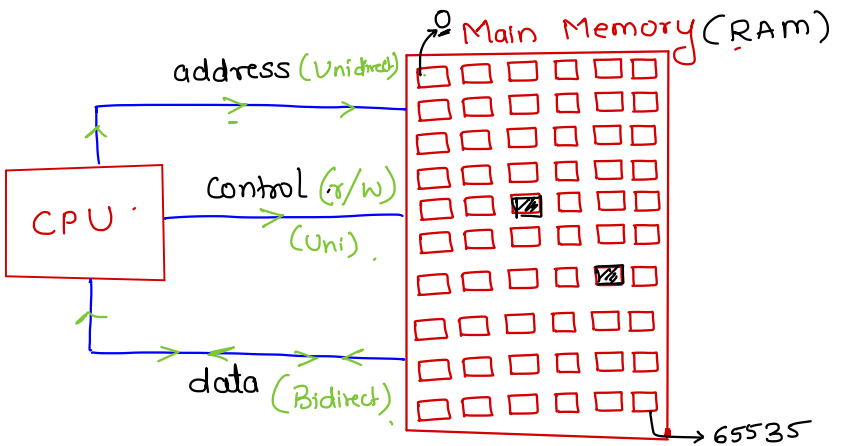
## Memory

- Memory holds (digital) data or information.
  - Bit = Binary Digit (0 or 1) --> Internally it is an electronic circuit i.e. FlipFlop.
    1 Byte = 8 Bits
    B, KB (2^10), MB (2^20), GB (2^30), TB (2^40), PB (2^50), XB (2^60), ZB (2^70)
- Primary memory – Memory
  - Directly accessible by the CPU.
    E.g. CPU registers, Cache, RAM.
- Secondary memory -- Storage
  Memory accessible via Primary memory. E.g. Disk, CD/DVD, Tape, ROM.
- Volatile vs Non-volatile memory
  - Volatile memory: The contents of memory are lost when power is OFF.
    Non-volatile memory: The contents of memory are retained even after power is OFF.

## Memory Hierarchy

- Level 0: CPU Registers
- Level 1: L1 Cache
- Level 2: L2 Cache
- Level 3: RAM
- Level 4: Disk (Local)
- Level 5: Remote storage (NFS)
- Comparison
  - Capacity: Level 0 is smallest (B) to Level 5 is largest (TB)
  - Speed: Level 0 is fastest and Level 5 is slowest
  - Cost: Level 0 is costlier and Level 5 is cheaper






RAM = 16GB. → $2^{16} = 65536$ (0 to 65535)

Bus ⇒ set of wires.

① Sequential Access ⇒ eg :- Magnetic tap.

② Direct Access ⇒ eg :- HDD → block by block.

③ Random Access ⇒ eg :- RAM → direct.

④ Associative Access ⇒ eg :- Cache → Key and Value

**Memory Access**

- CPU <----> Memory
- Address bus
  - Unidirectional from CPU to the memory
  - Address represent location of the memory to read/write
  - Number of lines = number of locations
- Data bus
  - Bi-directional from/to CPU to/from the memory
  - Carries the data
  - Number of lines = width of data unit
- Control bus
  - Read/Write operation
- CPU <---> Cache <---> RAM <---> Disk
- Sequential access: Read/write sequentially from start to end. e.g. Magnetic tapes
- Direct access: Read/write to the block address e.g. Hard disk
- Random access: Read/write to the memory (byte) address to e.g. RAM
- Associative access: Read/write to the scan/tag lines(Key –value storage )e.g. Cache

**RAM (Random Access Memory)**

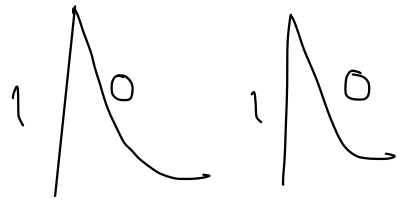
- RAM is packaged as a chip, Basic storage unit is a cell (one bit per cell)
- Internal memory of the CPU for storing data, program, and program result
- Used for Read/ Write
- Volatile (Temporary Storage)

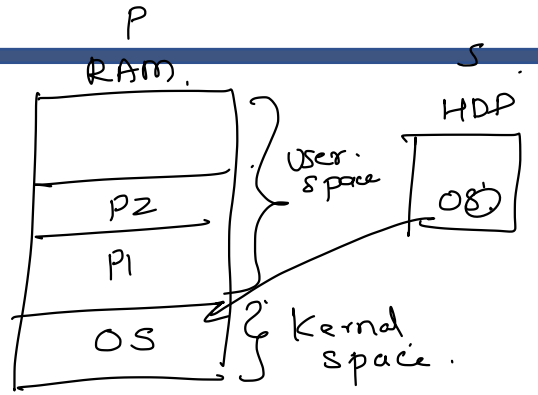
**Static RAM (SRAM)**

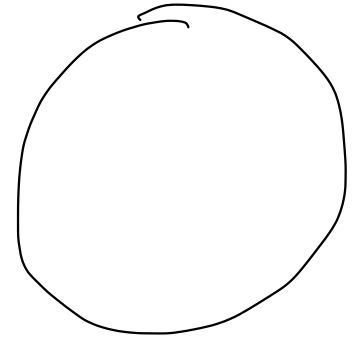
- Retains its contents as long as power is being supplied.
- Made up of transistors.
- SRAM is more often used for system cache.
- SRAM is faster than DRAM.

## Dynamic RAM (DRAM)

- Must be constantly refreshed or it will lose its contents.
- This is done by placing the memory on a refresh circuit that rewrites the data several hundred times per second.
- Made up of memory cells composed of capacitors and one transistor.
- DRAM is typically used as the main memory in computers.





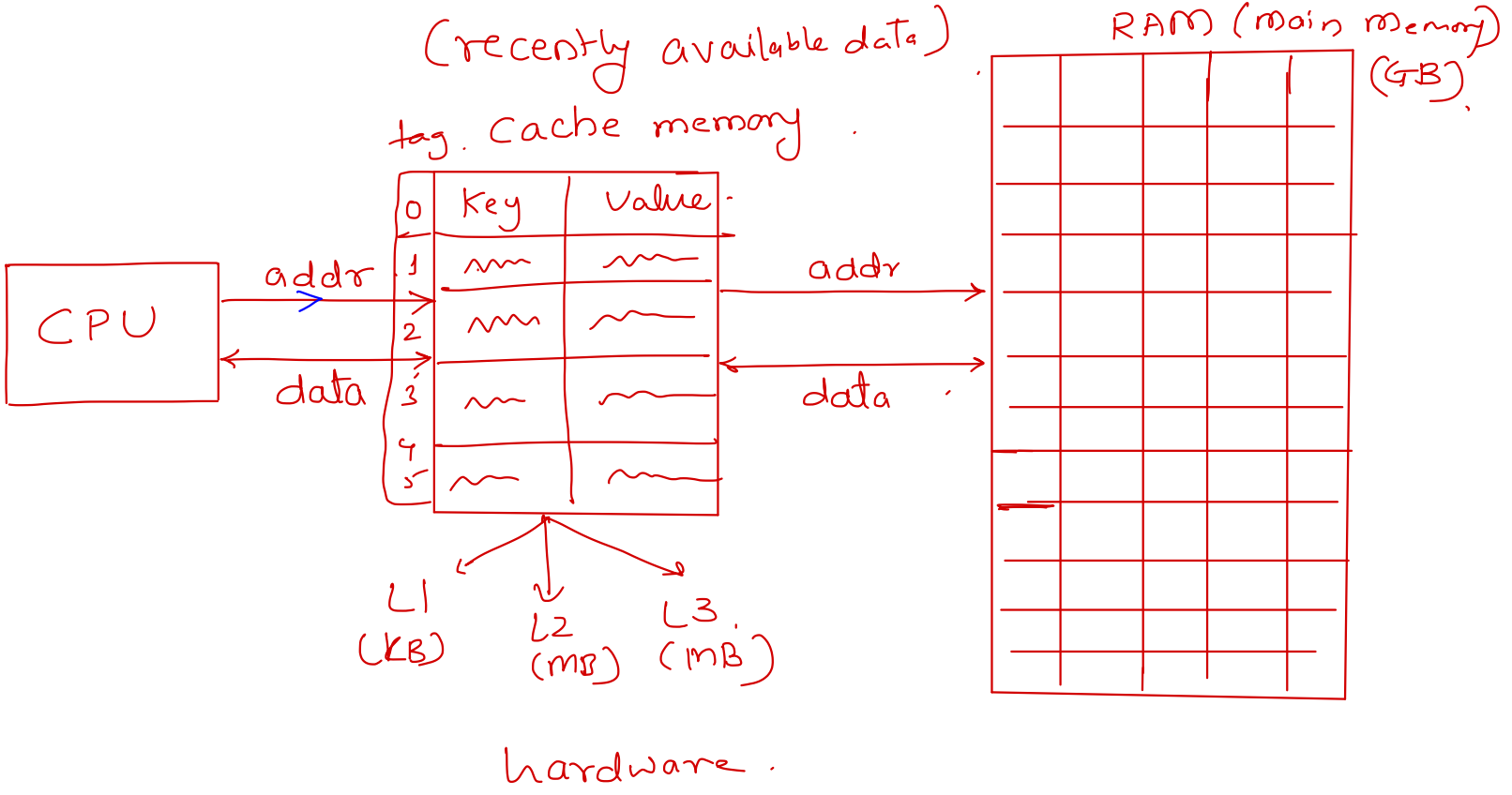



**ROM (Read Only Memory)**

- Read-only memory (Not writable).
- This type of memory is non-volatile.
- The information is stored permanently.
- A ROM stores such instructions that are required to start (bootstrap) a computer.
- BIOS – Basic Input Output System
  - Set of programs stored in PC Base ROM (on Motherboard).
  1. POST(Power On Self Test) – To test the peripherals.
  2. Bootstrap Loader – To find OS in disk/usb/cd.com.
  3. Basic/minimal device drivers for basic device functionality.
  4. BIOS setup utility (F1 or ESC).
- Programs (executable instructions) stored in ROM are called -- Firmware. e.g. BIOS, Bootstrap loader, POST, ...

Types of ROM

- Masked ROM (MROM) -- contents are fixed while manufacturing
- Programmable ROM (PROM) -- one time writable
- Erasable Programmable ROM (EPROM) -- written multiple times with special circuit
  - Ultra-Violet EPROM (UV-EPROM) -- all contents erased using UV rays
  - Electrical EPROM (E-EPROM) -- erase selected bytes using high electric current
- Flash (like E-EPROM) -- erase selected blocks - high speed


(recently available data)
RAM (main memory)
(GB)
tag. Cache memory
0
Key
Value
1
2
3
4
5
CPU
addr
data
addr
data
L1
(KB)
L2
(MB)
L3
(MB)

Hardware.

## Cache memory

- Associative memory. Key = memory address, Value = memory contents.
- Made up of cache lines tagged by tag index.
- In CPU chip or outside CPU chip.
- If requested data is found in cache (cache hit), contents are sent from cache itself to CPU (faster access).
- If requested data is not found in cache (cache miss), contents are accessed from main memory, copied in cache and then sent to CPU (slower access).
- Cache memory size is limited (KB or MB).
- When cache is full, oldest data is overwritten (Least Recently Used).
- Cache always stores recently accessed data.

# Memory Management

IDE

(high level to low level)

human readable

arch depend.

.C

Source file.

Compiler

.out

80386 assembly.

also assign addr to all instn and data. } w.r.t Vm memory } virtual addr or logical addr

80386

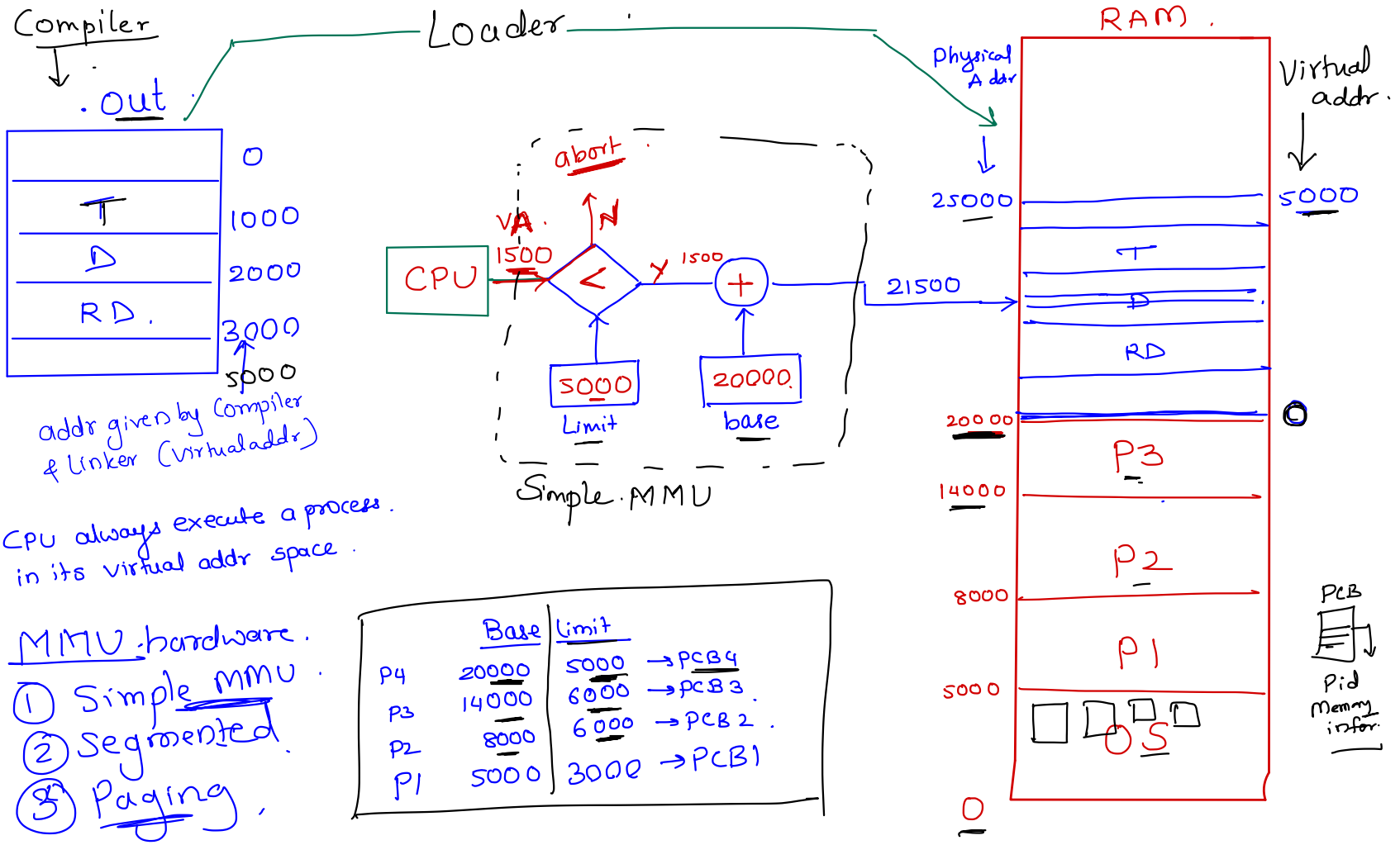
RAM 4 GB

VM (Virtual Machine)

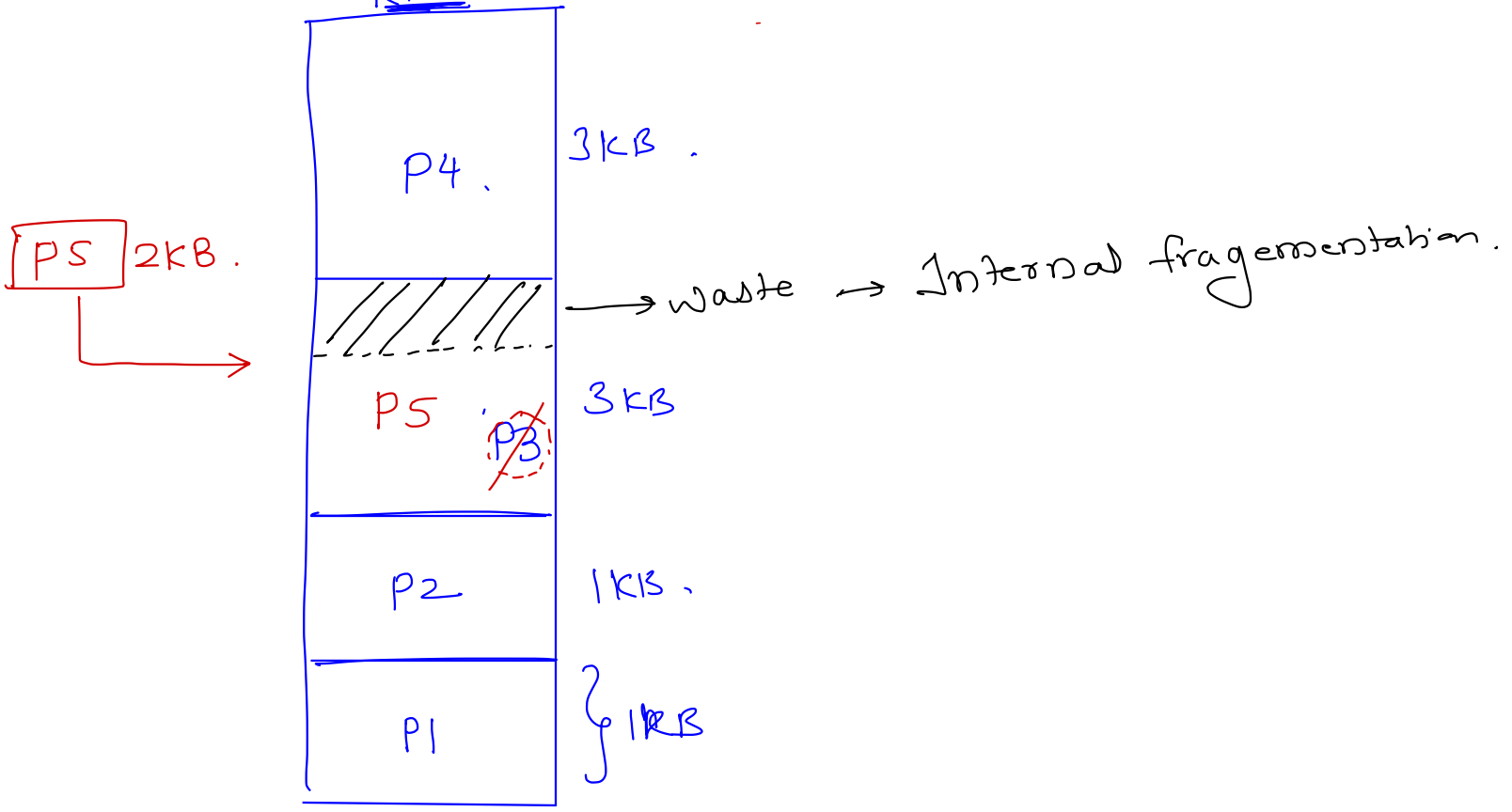
Can execute on any CPU like. pentium, core-2 duo, core-i x.

gcc -m32

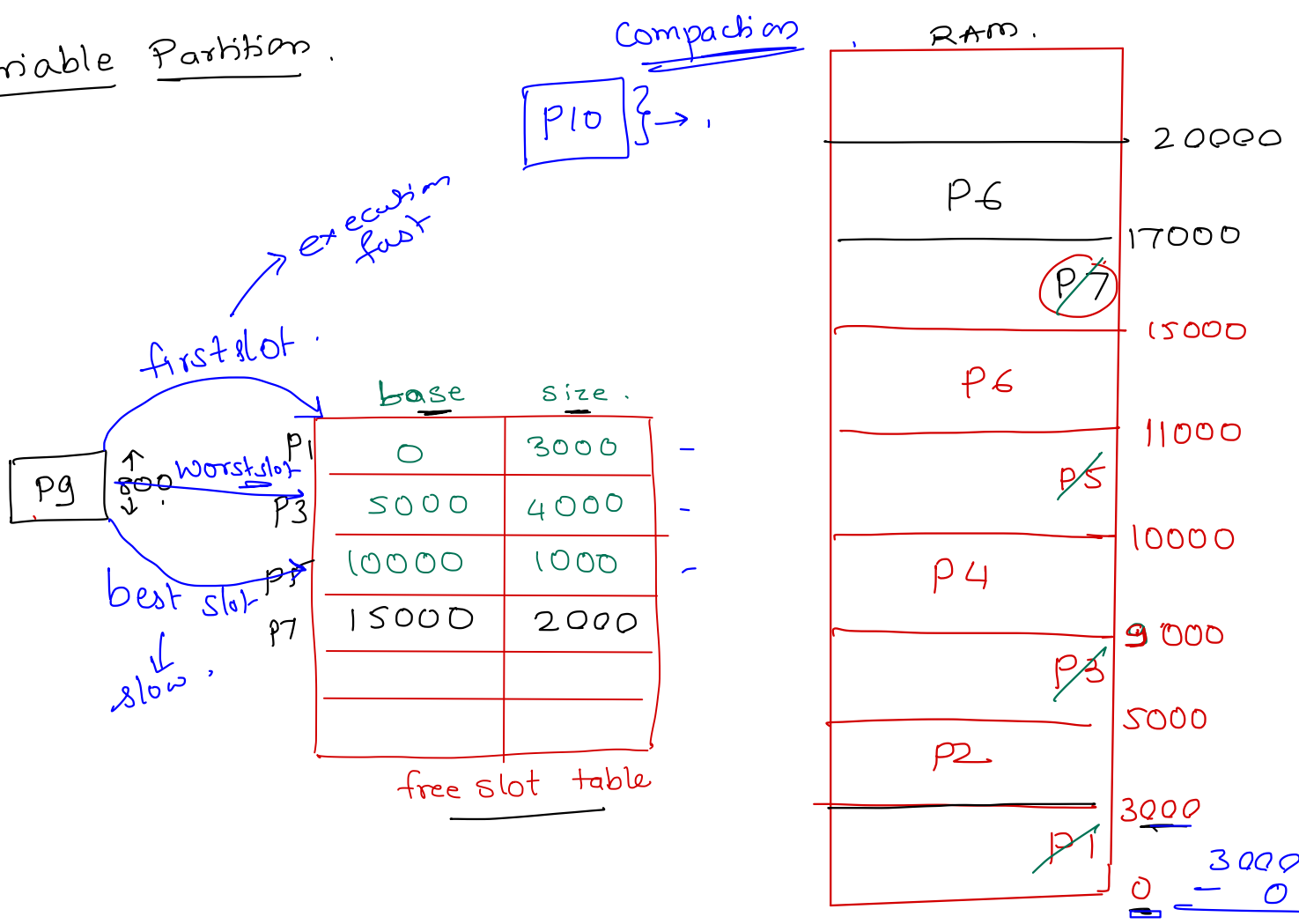
32-bit compilation.


Compiler
Loader
.out
T
D
RD
0
1000
2000
3000
5000
addr given by Compiler & Linker (Virtual addr)
CPU always execute a process in its virtual addr space.
abort
VA
1500
N
CPU
<
Y
1500
+
5000
Limit
20000
base
Simple MMU
Physical Addr
Virtual addr
RAM
25000
5000
21500
T
D
RD
20000
P3
14000
P2
8000
P1
5000
OS
0
PCB
Pid
Memory info
MMU hardware.
1. Simple MMU
2. Segmented
3. Paging
Base
Limit
P4 20000 5000 → PCB4
P3 14000 6000 → PCB3
P2 8000 6000 → PCB2
P1 5000 3000 → PCB1

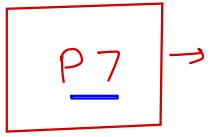
① Fixed Partition Scheme. RAM (8MB). Continuous store prog.

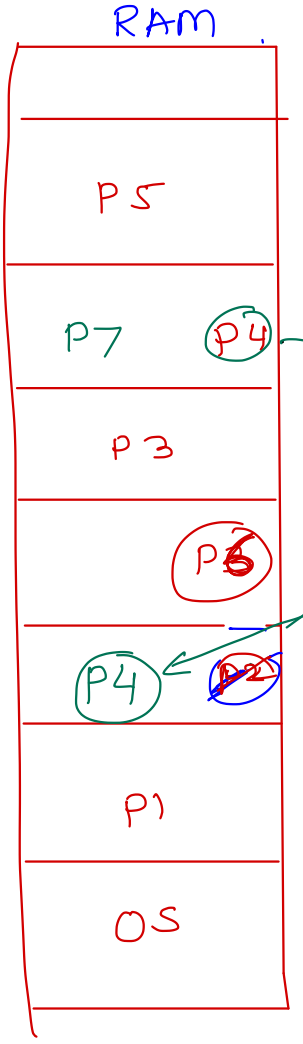


## (2) Variable Partition.



# Virtual mem (Swap in & out)

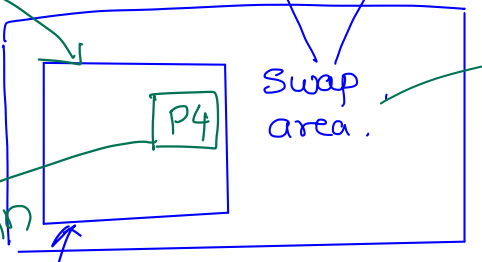



page file.sys
↑
window
↑
Swap file

Swap part
↑
Linux
↑
Swap partition

Swap-out




inactive process or sleeping process

Swap-in

disk

extended main memory or Virtual memory

1. Browser
2. Media player
3. MS-paint
4. MS-word

# Memory Management

- Compiler convert code from high level language to low level language.
- Compiler assumes a low config machine, while converting high level code to low level code called as "Virtual Machine".
- Compiler and Linker assign addresses to each variable/instruction assuming that program will execute in VM RAM. These addressed are called as "virtual address" or "logical address". The set of virtual addresses used by the process is referred "Virtual address space".
- However while execution these addresses might be occupied by other processes. Loader relocates all instructions/variables to the address available in RAM. The actual addresses given to the process at runtime are called as "physical address" or "real address". The set of physical addresses used by the process is referred "Physical address space".
- CPU always executes a process in its virtual address space i.e. CPU always request virtual addressed (on address bus).
- These virtual addresses are verified and then converted into corresponding physical addresses by a special hardware unit called as "Memory Management Unit (MMU)".
- Simple MMU holds physical base address and limit (length) of the process. The base & limit of each process is stored in its PCB and then loaded into MMU during context switch.
- In multi-programming OS, multiple programs are loaded in memory.

# Memory Management

- RAM memory should be divided for multiple processes running concurrently.
- Memory Mgmt. scheme used by any OS depends on the MMU hardware used in the machine.
- There are three memory management schemes are available (as per MMU hardware).
  1. Contiguous Allocation
  2. Segmentation
  3. Paging

**Contiguous Allocation**

**Fixed Partition**

- RAM is divided into fixed sized partitions.
- This method is easy to implement.
- Number of processes are limited to number of partitions.
- Size of process is limited to size of partition.
- If process is not utilizing entire partition allocated to it, the remaining memory is wasted. This is called as "internal fragmentation".

# Memory Management

## Dynamic Partition

- 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.

# Memory Management

- 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.
- If amount of memory required for a process is available but not contiguous, then it is called as "external fragmentation".
- To resolve this problem, processes in memory can be shifted/moved so that max contiguous free space will be available. This is called as "compaction".

# Memory Management

## 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.