Operating Systems

Quiz

Q. Which of the following statement/s is/are false about a process?

A. Process is a running entity of a program

B. Program in main memory is referred as a process

C. One program may has multiple running instances i.e. processes.

D. Program in execution is referred as a process.

E. All of the above

F. None of the above

Q. Process which is in the main memory waiting for the CPU time considered in a _____.

A. waiting state

B. new state

C. ready state

D. running state

Quiz

Q. _____ copies an execution context of a process which is scheduled by the scheduler from its PCB and restores it onto the CPU registers.

A. Loader

B. Interrupt Handler

C. Dispatcher

D. Job Scheduler

Q. In a timeshare operating system, when the time slot assigned to a process is completed, the process switches from the current state to?

A. Suspended state

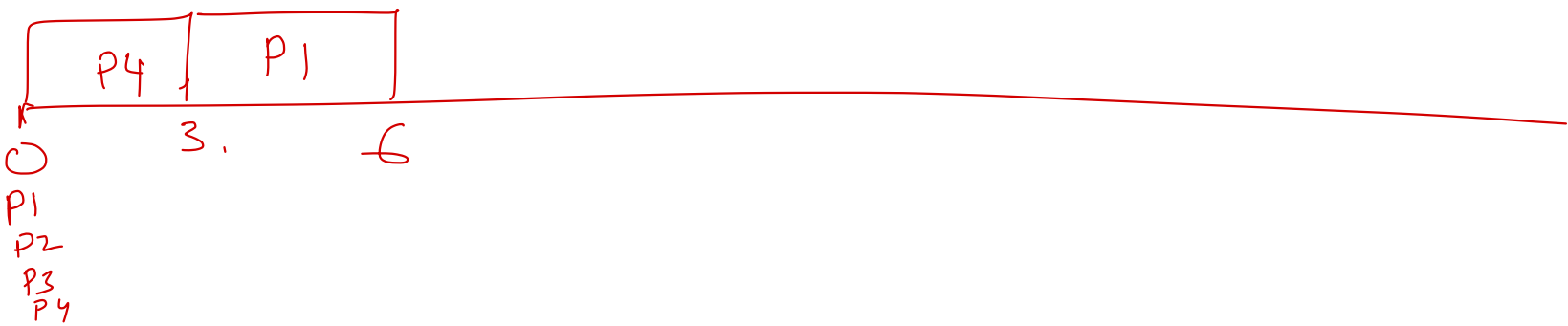
B. Terminated state

C. Ready state

D. Blocked state

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

Process	Burst time	Arrival
P1	6	0
P2.	8	6
P3	7	0
P4	3	0



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. When a process is in a “Blocked” state waiting for some I/O service. When the service is completed, it goes to the _____

a) Terminated state

b) Suspended state

c) Running state

d) Ready state

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

A. SIGTERM

B. SIGEND

C. SIGSTOP

D. SIGKILL

Inter process Communication (IPC)

- A process cannot access of memory of another process directly. OS provides IPC mechanisms so that processes can communicate with each other.

Independent process

- **do not get affected by the execution of another process**

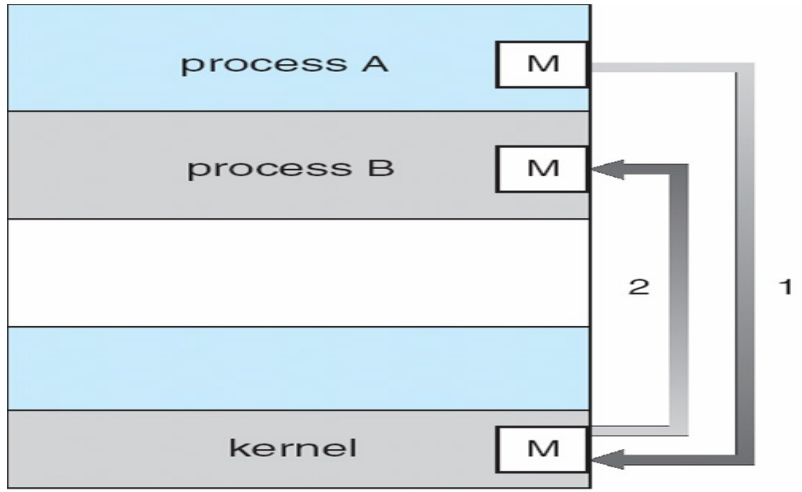
Cooperating process

- **get affected by the execution of another process**

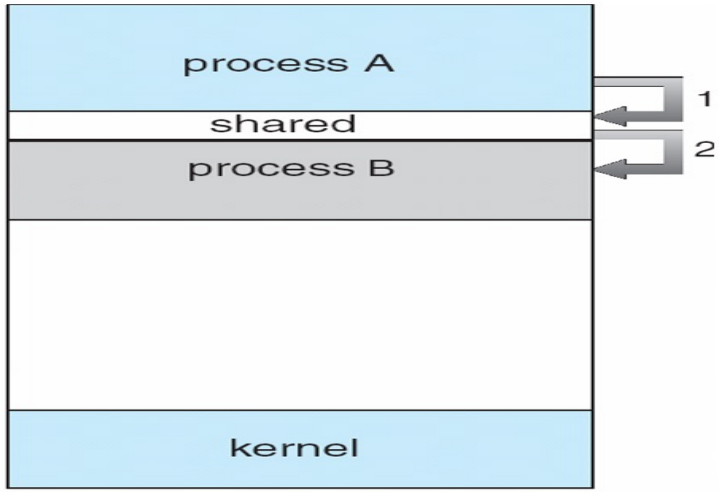
Reasons for cooperating processes:

- Information sharing
- Computation speedup
- Modularity
- Convenience
- Cooperating processes need **inter process communication** (**IPC**)

Mechanisms of IPC



(a)Message Passing



(b) Shared Memory Model

Message Passing

- communication takes place by means of messages exchanged between the cooperating processes
- Uses two primitives : Send and Receive

Shared Memory

- In the shared-memory model, a region of memory that is shared by cooperating processes is established.
- Processes can then exchange information by reading and writing data to the shared region.

Mechanisms of IPC

Unix/Linux IPC mechanisms

- Signals
- Shared memory
- Message queue
- Pipe
- Socket

Mechanisms of IPC

Signals

- OS have a set of predefined signals, which can be displayed using command
  - terminal> kill -l
- A process can send signal to another process or OS can send signal to any process.

Important Signals

- SIGINT (2): When CTRL+C is pressed, INT signal is sent to the foreground process.
- SIGKILL (9): During system shutdown, OS send this signal to all processes to forcefully kill them. Process cannot handle this signal.
- SIGSTOP (19): Pressing CTRL+S, generate this signal which suspend the foreground process. Process cannot handle this signal.
- SIGCONT (18): Pressing CTRL+Q, generate this signal which resume suspended the process.
- SIGSEGV (11): If process access invalid memory address (dangling pointer), OS send this signal to process causing process to get terminated. It prints error message "Segmentation Fault".

Mechanisms of IPC

Message Queue

- Used to transfer packets of data from one process to another.
- It is bi-directional IPC mechanism.
- Internally OS maintains list of messages called as "message queue" or "mailbox".

Pipe

- Pipe is used to communicate between two processes.
- It is stream based uni-directional communication.
- Pipe is internally implemented as a kernel buffer, in which data can be written/read.
- There are two types of pipe:
  - Unnamed Pipe
  - Named Pipe (FIFO)

Mechanisms of IPC

Socket

- Socket is defined as communication endpoint.
- Sockets can be used for bi-directional communication.
- Using socket one process can communicate with another process on same machine (UNIX socket) or different machine (INET sockets) in the same network.
- Sockets can also be used for communication over Bluetooth, CAN, etc.

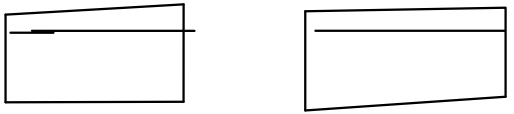
Shared memory

- OS creates a memory region that is accessible to multiple processes.
- Multiple processes accessing a shared memory need to be synchronized to handle race condition problem.
- Fastest IPC mechanism.
- Both processes have direct access to shared memory(no os invoked)

Multitasking

MS-Word

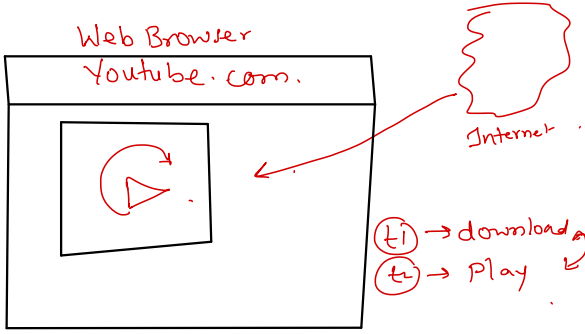
MS-Paint



Independent process.

Web Browser

Youtube.com.



Internet

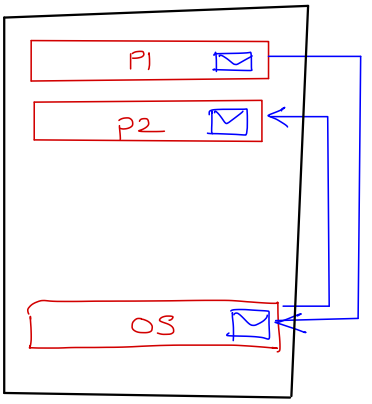
t_1 → download data (IPC)

t_2 → Play

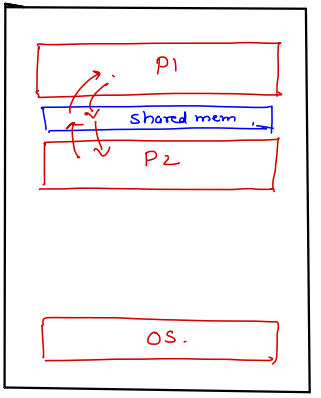
Co-operative Process.

IPC techniques.

① Message Passing



② Shared Memory



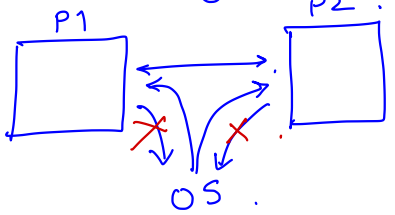
LINUX / UNIX IPC mechanism.

1. Signal
2. Message Queue
3. Pipe
4. Socket
5. Shared Memory → Shared memory.

(2, 3, 4) → Message passing.

① Signal

- Predefine. Signal.



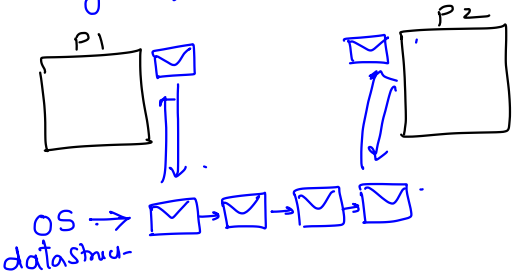
① SIGINT → ctrl+C.

② SIGKILL →.

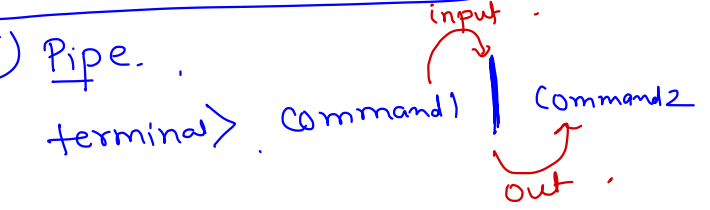
③ SIGSEGV →.

② Message queue.

✓ Bidirectional
✓ Packet Based.



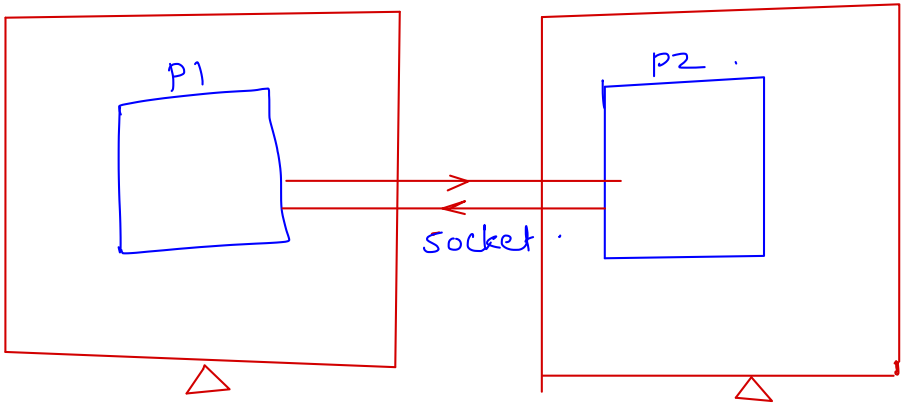
③ Pipe.



→ Unidirectional

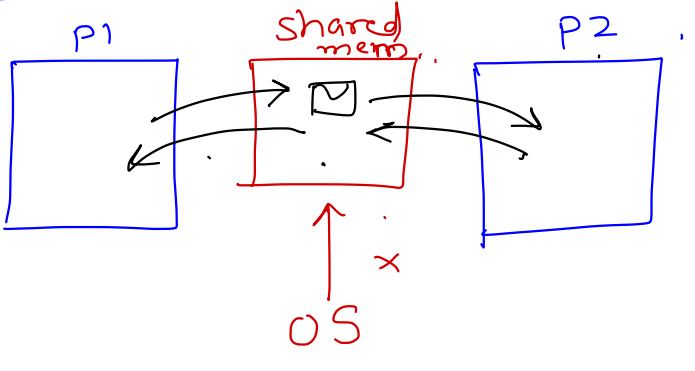
→ Stream based.

④ Socket



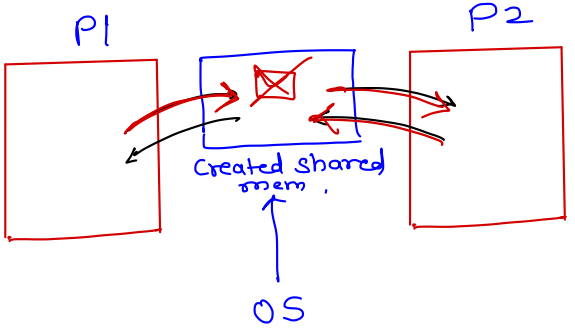
---

⑤ Shared Memory



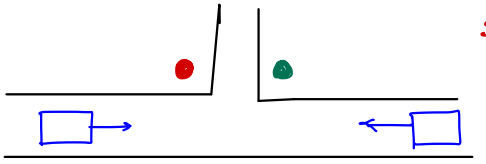
✓ fastest

⑤ Shared Memory

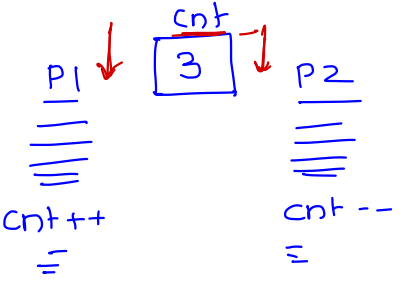


Shared memory is fastest.

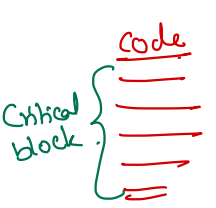
If two process try to access same resource at the same time, it is referred as "race condition"



Sync mechanism ensure that only one process will access resource at a time and thus data inconsistency is avoided.



expected: cnt will be unchanged (3)
But, if race condition occurs, then cnt may be 2 or 4. This is data inconsistency.



A block of code executed by multiple processes at same time cause data inconsistancy

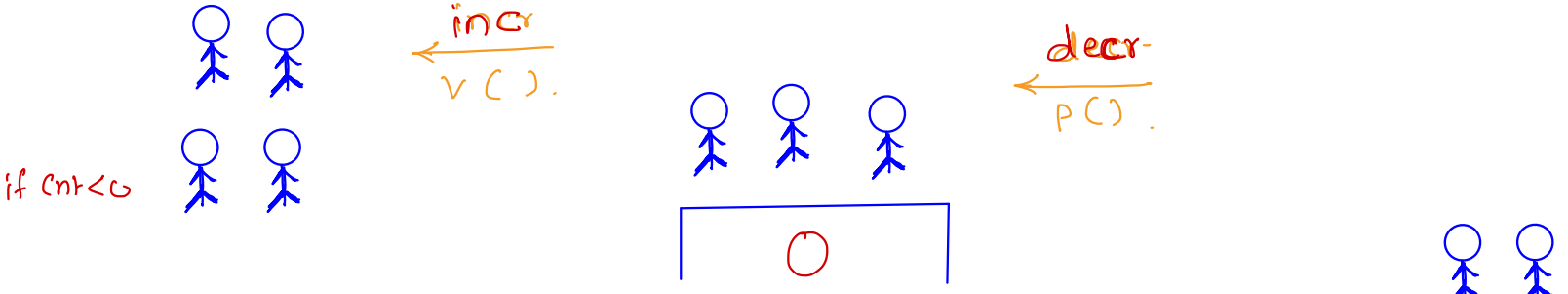
Synchronization

- If two/multiple processes try to access same resource at same time , it is referred as “race condition”
- When race condition occurs, resource may get corrupted (unexpected results).
- Peterson's problem: If two processes are trying to modify same variable at the same time, it can produce unexpected results.
- Synchronization Mechanism ensure that only one process will access resource at a time and thus data inconsistency is avoided.
- A block of code ,executed by multiple processes at same time cause data inconsistency. Such kind of code block is called Critical section.
- To resolve race condition problem, one process can access resource at a time. This can be done using sync objects/primitives given by OS.
- OS provide some Synchronization objects

1) Semaphore
2) Mutex

Semaphore

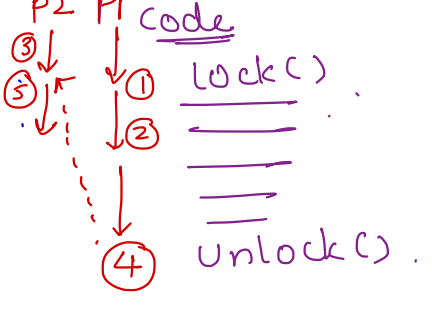
- Semaphore was suggested by Dijkstra scientist (dutch math)
- Semaphore is a counter
- On semaphore two operations are supported:
- wait operation: decrement op: P operation:
  - semaphore count is decremented by 1.
  - if cnt < 0, then calling process is blocked(block the current process).
  - typically 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



If semaphore cnt is -n, number of waiting processes are n

Semaphore
- Counting / Classic. (when multiple processes can access a resource at a time.)
- Binary (0 or 1). (when a single process can access a resource at a time.)

Mutex (Mutual exclusion): when only one process access resource like Binary semaphore. Mutex has two states i.e 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

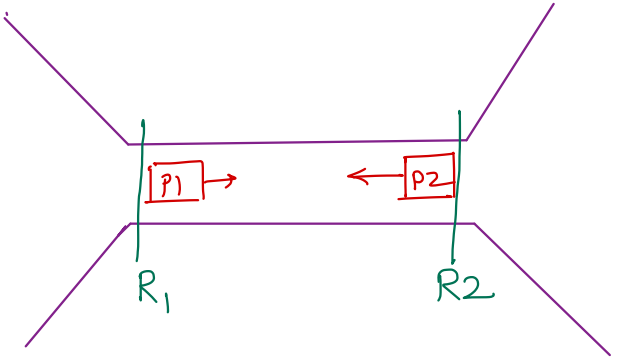
Mutex(Mutual Exclusion)

- 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.



Characteristic of Deedlock.

1. No pre-emption.
2. No Mutex.
3. Hold and wait.
4. Circular wait.

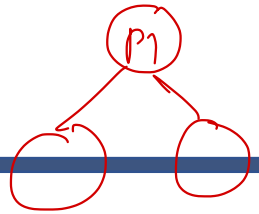
Deadlock

- 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:**
  1. 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.
  2. Banker's algorithm: A bank always manage its cash so that they can satisfy all customers.

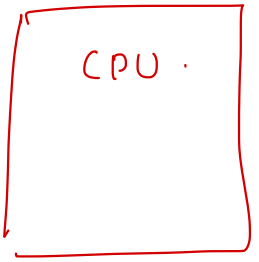
- **Deadlock Recovery**
  - Deadlock can be solved by resource preemption or terminating one of the process (involved in deadlock).

1. Resource pre-emption -> forcibly withdraw resources given to the processes.
2. 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).



↳ storage.

digital.

Binary → 0 or 1.

1Byte = 8bit.

KB, MB, TB, GB, b B

Memory

primary

Secondary.

eg:- RAM, CPU Reg, Cache.

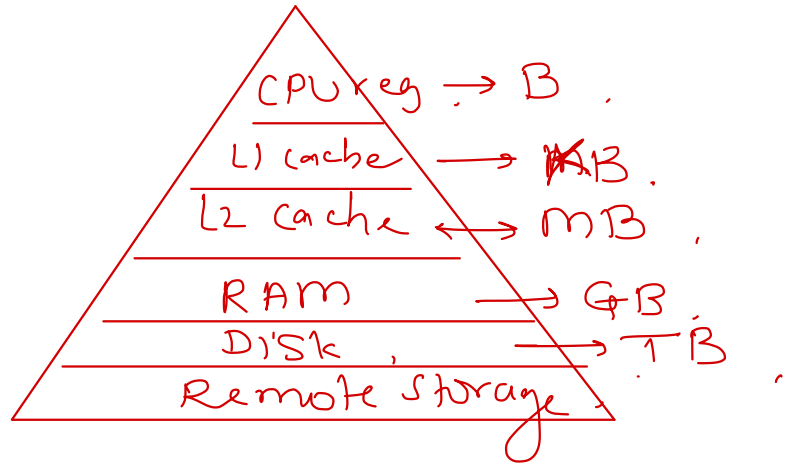
HDD, ROM.

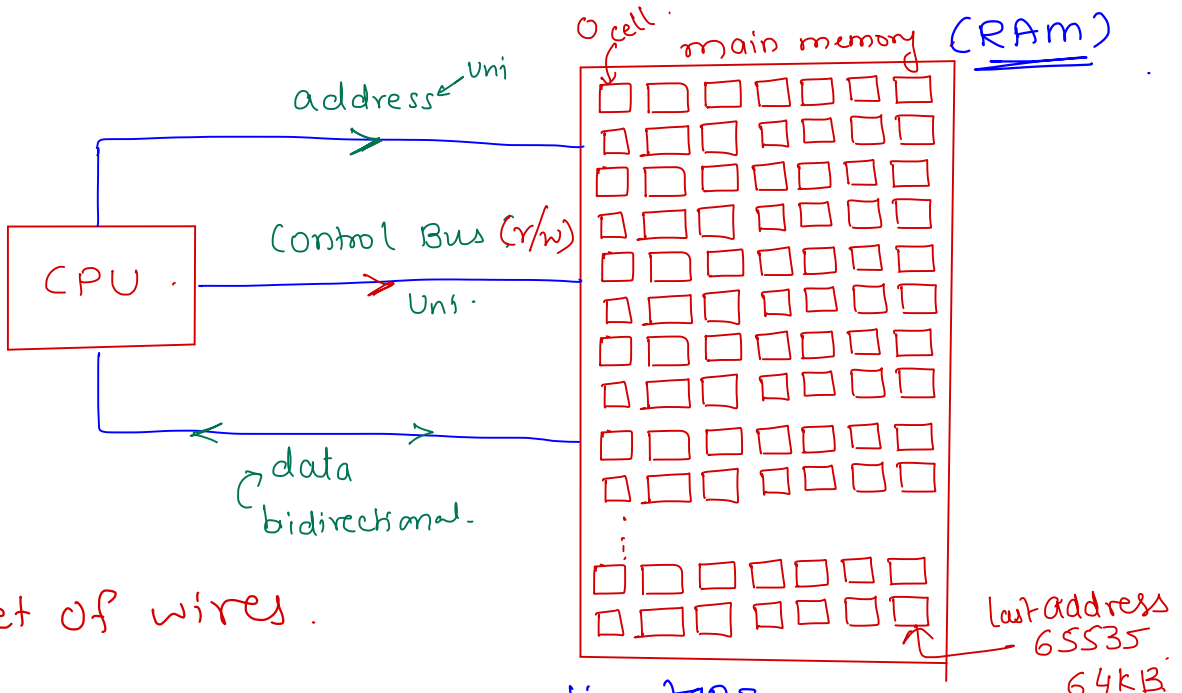
Memory

- 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





Bus :- Set of wires.

1. Sequential Access :- eg :- magnetic taps.
2. Direct Access : Block by Block → eg :- HDD.
3. Random Access : Byte By Byte → eg :- RAM
4. Associative Access :- Key and Value → eg :- cache.

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)

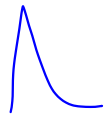
- RAM is packaged as a chip, Basic storage unit is a cell (one bit per cell)
- Internal memory of the CPU for storing data, program, and program result
- Used for Read/ Write
- Volatile (Temporary Storage)

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.

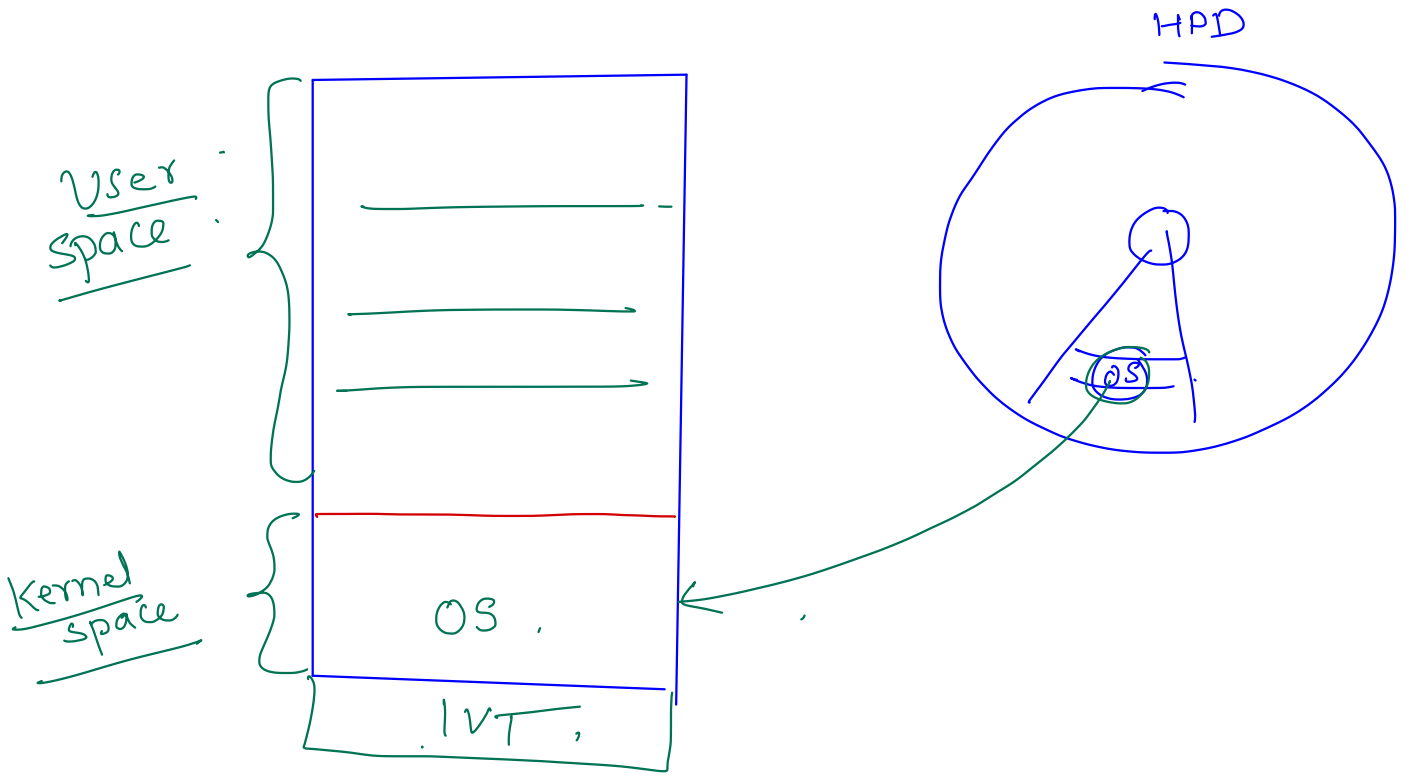
Dynamic RAM (DRAM)

- 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, ...

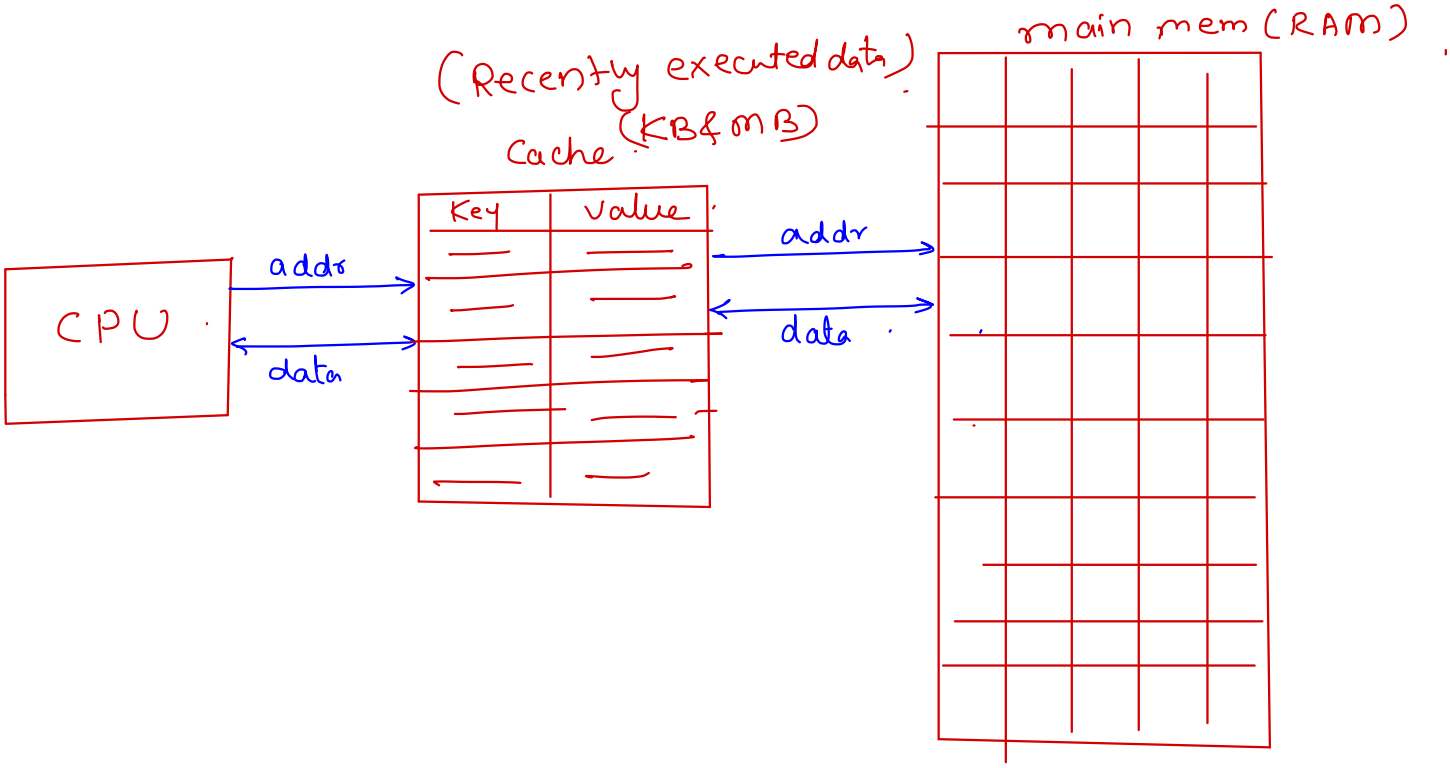

HPD
User space
OS
Kernel space
OS
IVT

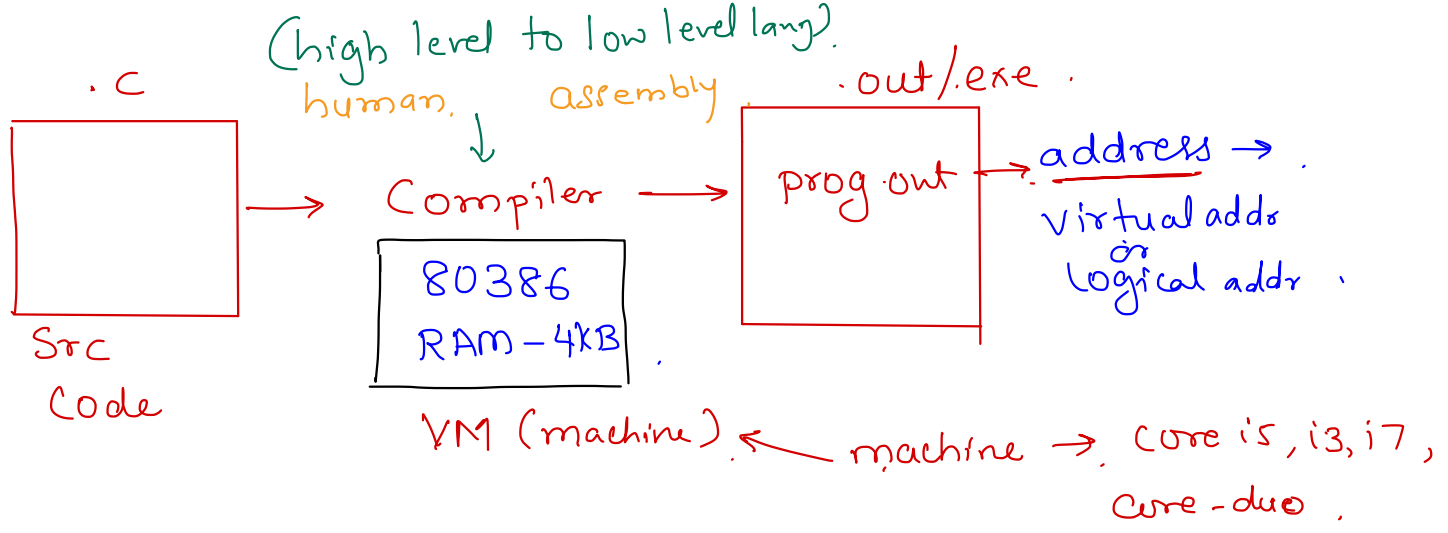
Types of ROM

- 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

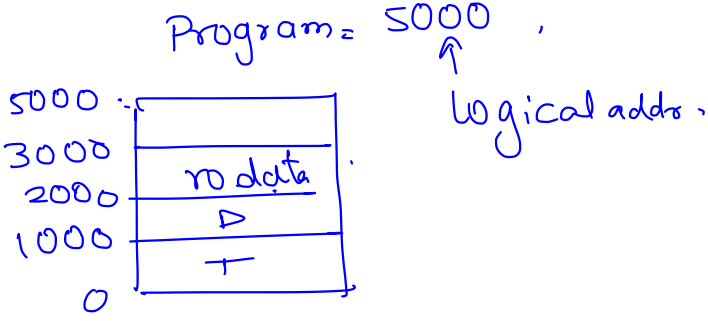
Cache memory

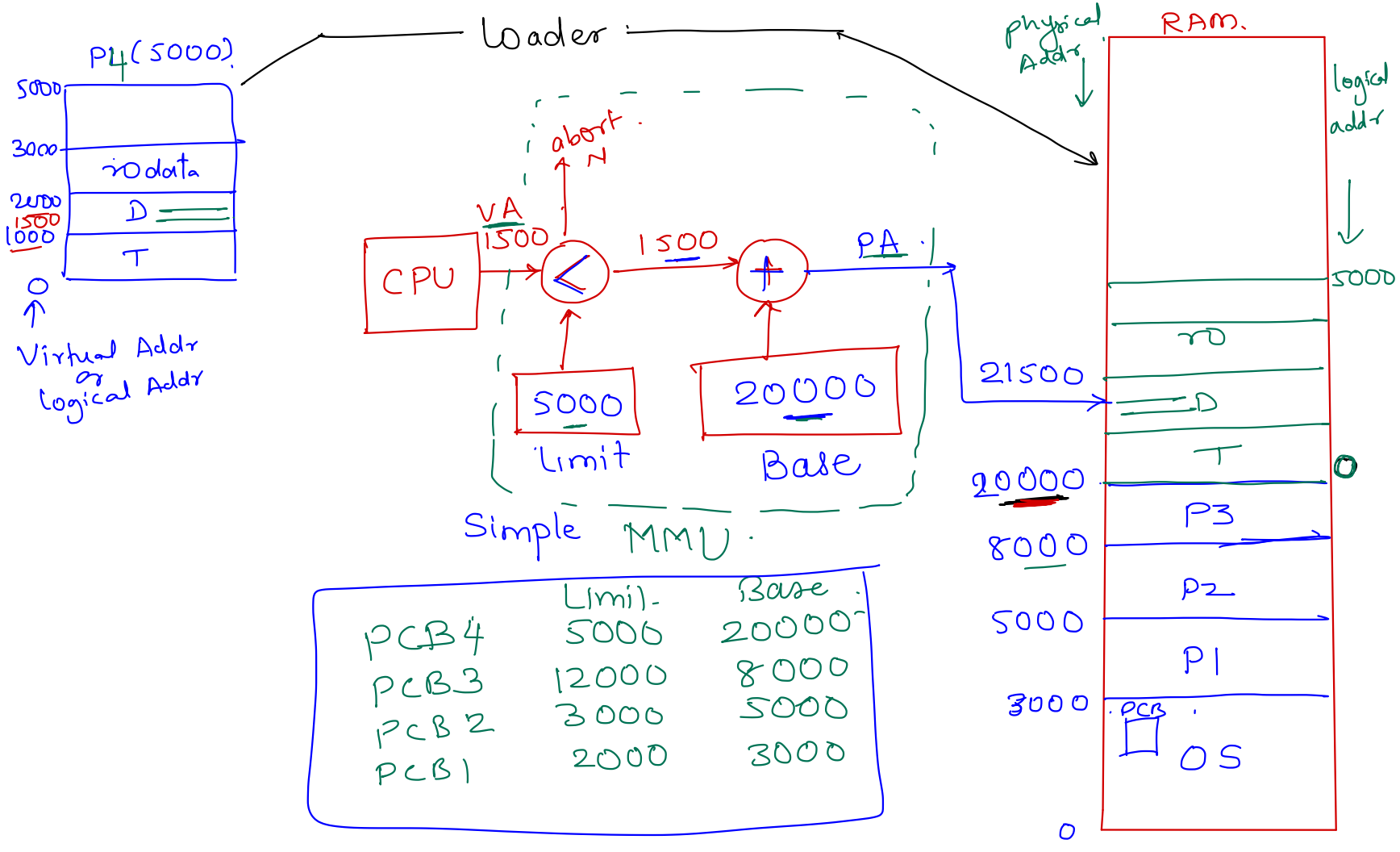
- Associative memory. Key = memory address, Value = memory contents.
- Made up of cache lines tagged by tag index.
- In CPU chip or outside CPU chip.
- If requested data is found in cache (cache hit), contents are sent from cache itself to CPU (faster access).
- If requested data is not found in cache (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.


main mem (RAM)
(Recently executed data)
Cache (KB & MB)
Key
Value
CPU
addr
data
addr
data


(high level to low level lang).
human.
assembly
.c
.out/.exe
Compiler
80386
RAM - 4KB
prog.out
address →
virtual addr
or
logical addr
Src
Code
VM (machine)
machine → core i5, i3, i7,
core-duo

gcc -32


Program = 5000
logical addr.
5000
3000
2000
1000
0
no data
D
T



	Limit	Base
PCB4	5000	20000
PCB3	12000	8000
PCB2	3000	5000
PCB1	2000	3000

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 addr" or "logical addr". 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 add" or "real add". The set of physical addresses used by the process is referred "Physical address space".
- CPU always executes a process in its virtual addr space i.e. CPU always request virtual addressed (on addr 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.