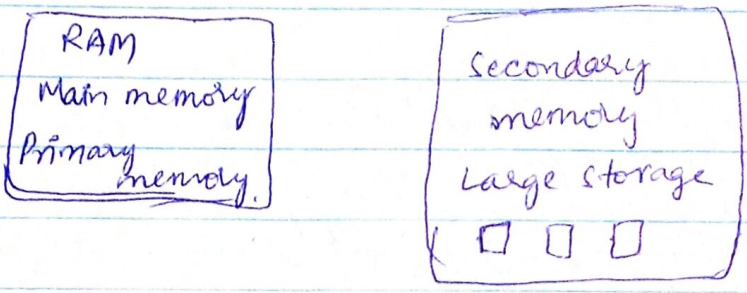
⑥ → O.S

Memory Allocation, Fragmentation, Paging, Segmentation, Virtual Memory, Thrashing

Basic Concepts :-



1) Program (or) instruction is executed when it is brought into main memory (or) primary memory.

2) It will not going to be executed itself when it is present in secondary memory

3) A program that is loaded into main memory and is executed is commonly referred to as "process"

Memory Allocation:-

- "Contiguous memory allocation" is one of the oldest memory allocation schemes. When a process needs to execute, memory is requested by the process. The size of the process is compared with the amount of contiguous main memory available to execute the process.

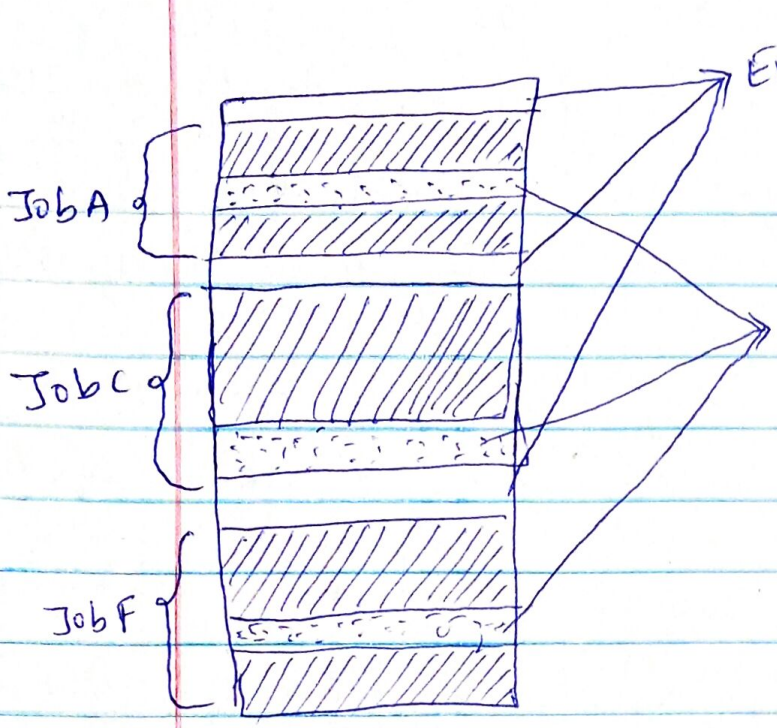
PROBLEMS:- External Fragmentation.

INTERNAL FRAGMENTATION	EXTERNAL FRAGMENTATION
1. When a process is allocated more memory than required, few space is left unused and this is called "INTERNAL FRAGMENTATION"	1. After execution of processes when they are swapped out of memory and other smaller processes replace them, many small non contiguous (adjacent) blocks of unused spaces

(2) → OS

are formed which can serve a new request if all of them are put together but as they are not adjacent to each other, a new request can't be served and this is known as "EXTERNAL FRAGMENTATION"

2) It occurs when memory is divided into fixed-size partitions	2. It occurs when memory is divided into variable-sized partitions based on size of process.
3) It can be cured by allocating memory dynam-ically (or) having partitions of different sizes	3) It can be cured by "Compaction", "Paging" and "Segmentation".



Job B

"There is enough memory to run 'Job B', but the memory is not contiguous"

— External Fragmentation

⇓

Because of "External Fragmentation" concept of paging arises

→ Non-Contiguous memory allocation; a program is divided into several blocks that may be placed in different parts of main memory.

→ ~~Using~~ Paging uses the concept of non-contiguous memory allocation.

③ → 0.5

De-fragmentation:-

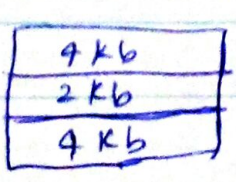
1) De-fragmentation is a process by which fragmented files that are stored on a disk are re-arranged to occupy contiguous storage locations allowing for both space and performance optimization.

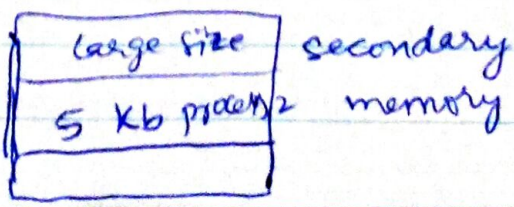
2) Disk - fragmentation may cause crashes, slowdowns, freeze-ups, and even total system failures. You can use the de-fragmentation operations to analyze the disk, and defragment it to consolidate free space.

→ ←

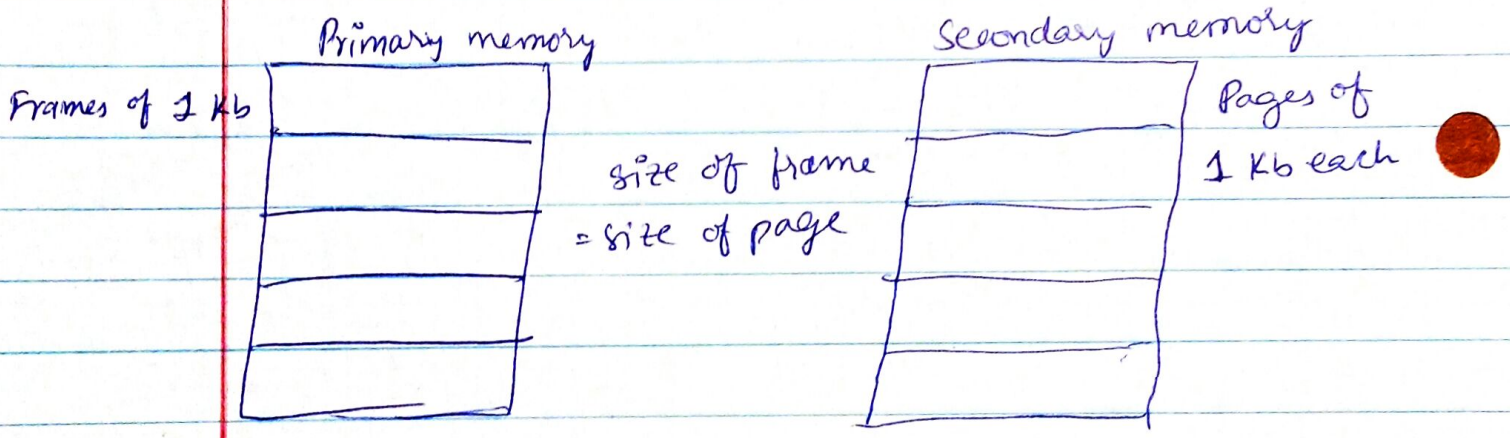
- when a program in secondary memory is loaded into primary memory for execution, then it is process.

→ Because of external - fragmentation, below 5 Kb process in secondary memory cannot be loaded into primary memory.





- To tackle above problem, paging concept is introduced, where, "Secondary memory" is divided into pages (physical divisions) of equal size. Secondary memory divisions are called "Pages"
- Primary memory is divided into physical divisions called "Frames".
- In a machine, size of a page = size of a frame



SEGMENTATION vs PAGING

- Paging never cares about what is actually present in instruction or program. It is just physical division
- Suppose if program contains loops, their pages should remain together

→ "Segmentation", ④→O.S is Logical division of programs together rather than having fixed page sizes

- If a process (or) program is trying to access, other than its allocated (or) logical segment than it throws "segmentation fault".
- Therefore, segments can be of different sizes.

VIRTUAL MEMORY:-

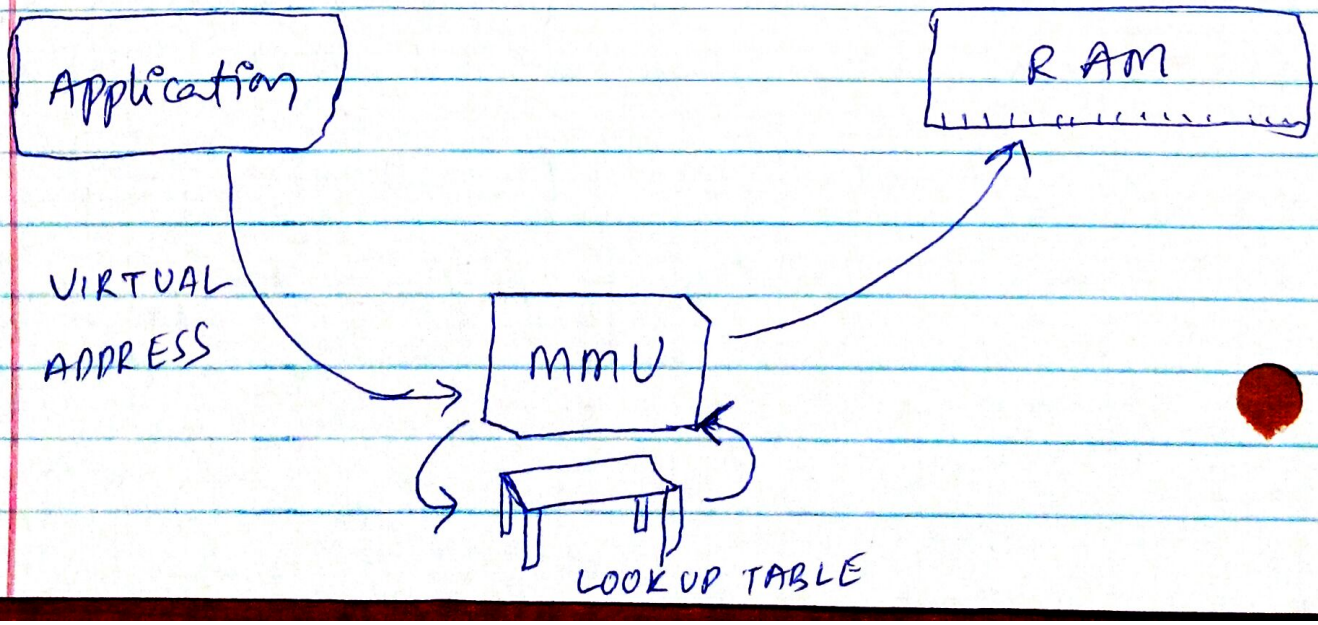
DISK Thrashing:-

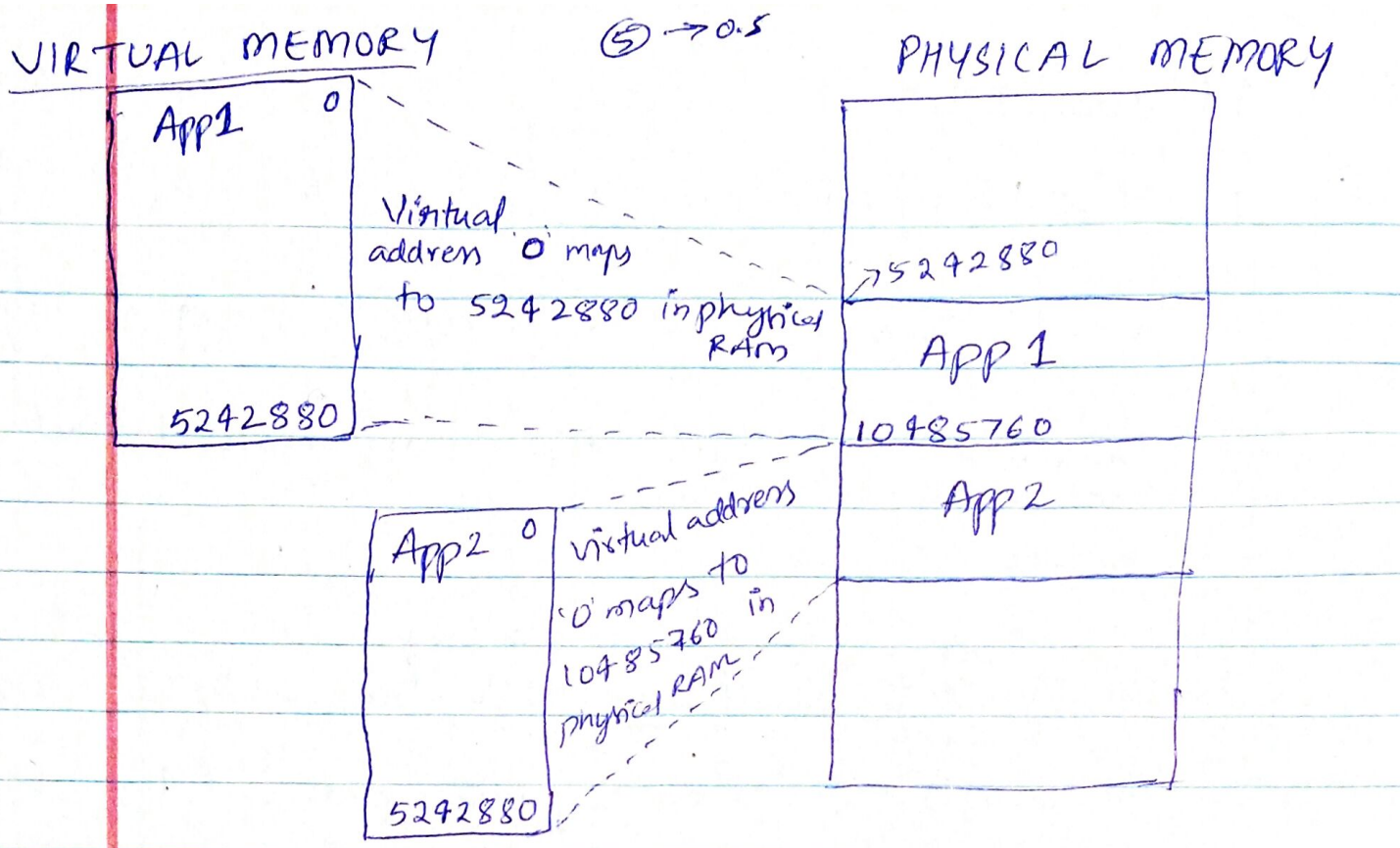
- Disk Thrashing is a problem that may occur when virtual memory is being used.
- As the main memory fills up, then more and more pages need to be swapped in and out of virtual memory. This swapping leads to a very high rate of hard disk access.

- Compared to RAM, moving a hard disk head takes thousands of times longer. And so eventually more time is spent swapping the pages than on processing the data.
- If disk-thrashing continues, then most likely programs will lock-up (or) not run.

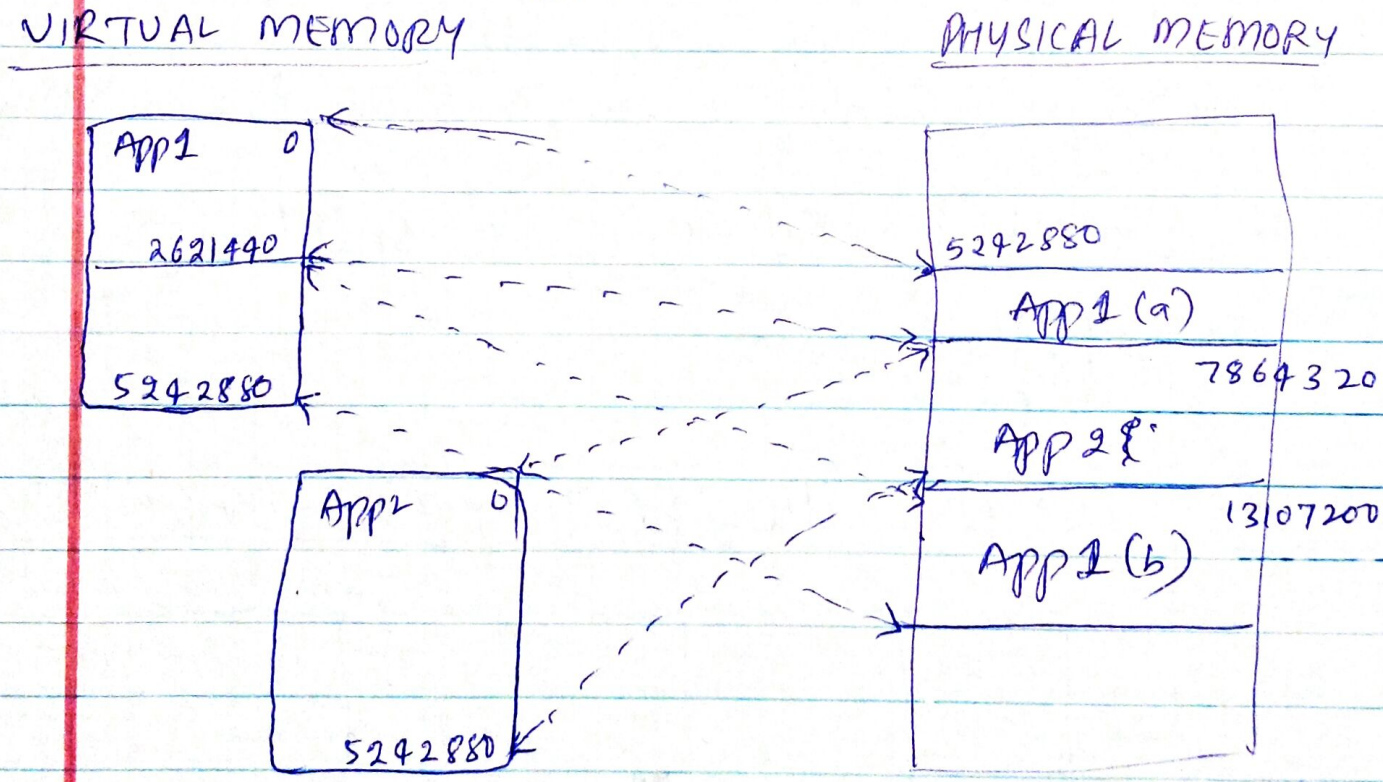
VIRTUAL MEMORY:-

- It is a memory management technique, that provides an "idealized abstraction of the storage resources that are actually available on a given machine", which "creates the illusion to users of a very large (main) memory".





(OR)



ONE TO ONE MAPPING

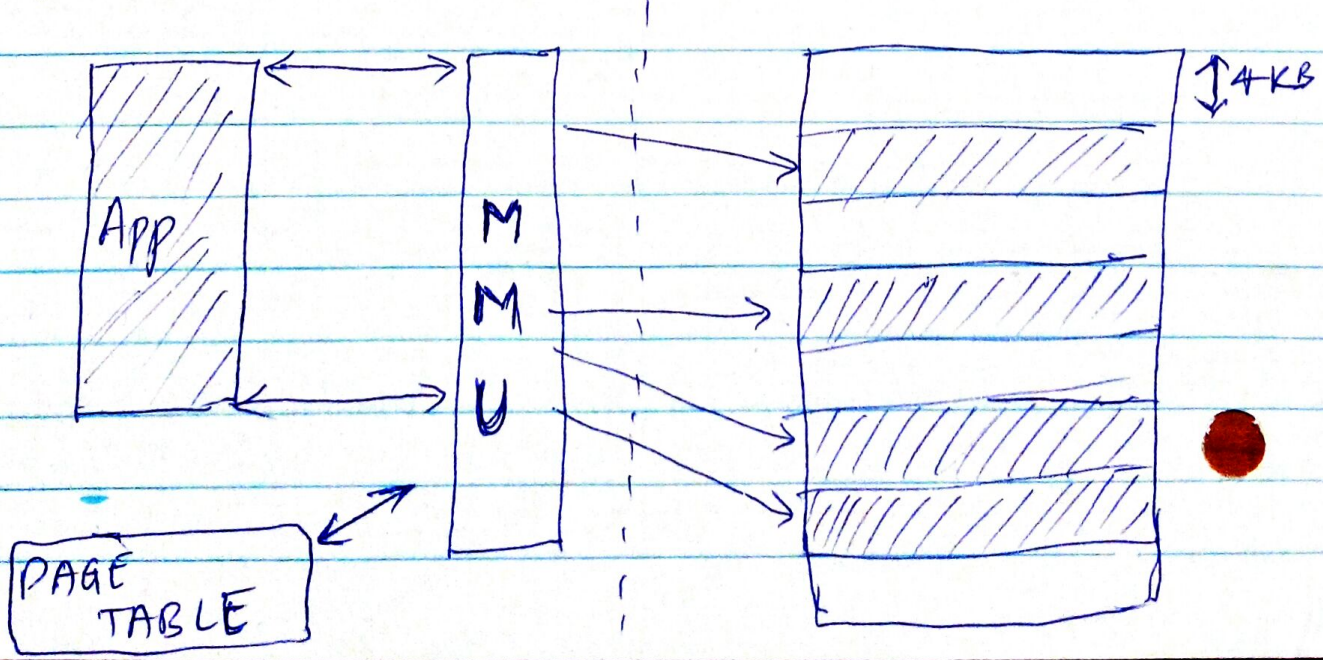
Advantages:-

1. Each application is self-contained. It doesn't write-over the other app's memory space because it has it's own virtual address space

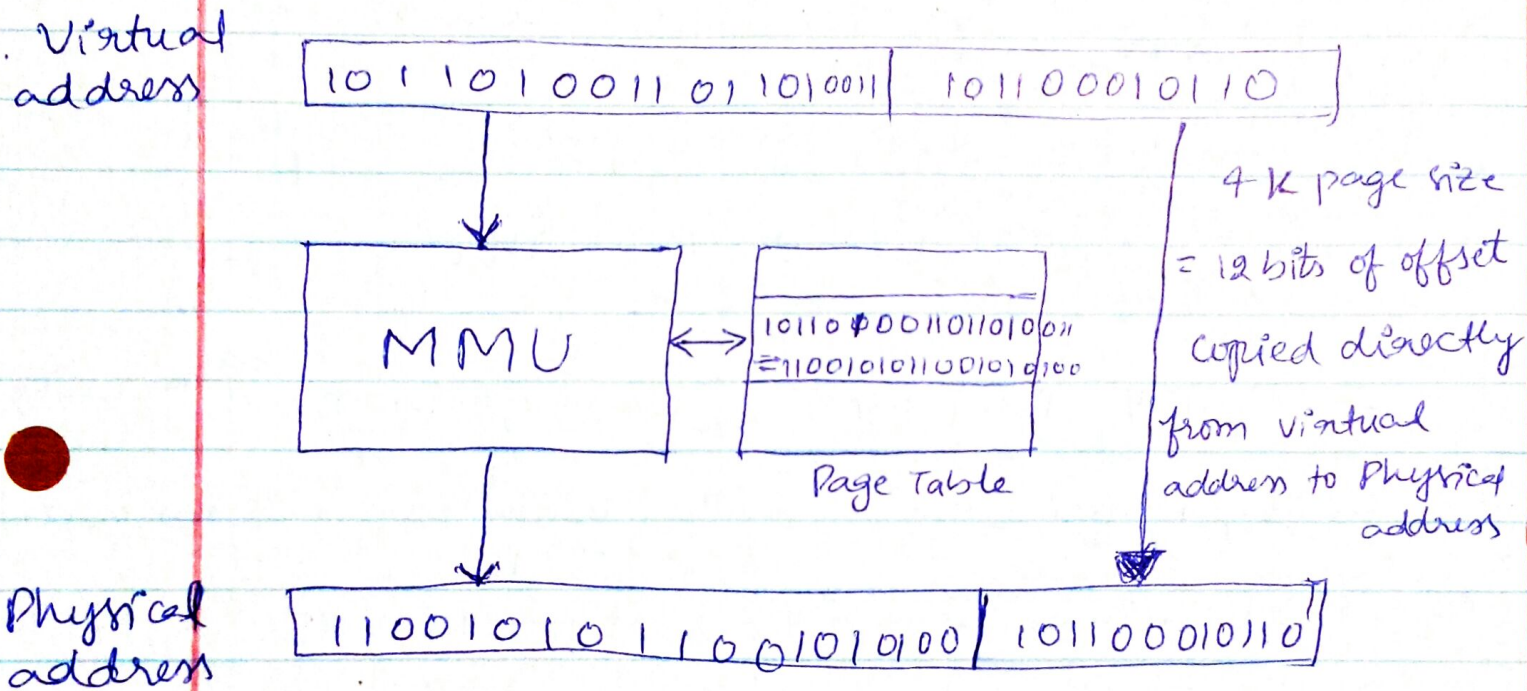
2. It doesn't matter where the app is, because MMU does the mapping between the virtual addresses and physical addresses.

3. App doesn't need to be in continuous block, ~~not~~ it can be split up in many different parts.

→ ## USING THE PAGING AND PAGE TABLE:-



→ But, if the program chunk is placed in the middle of the page (or) different location apart from starting address of page, then we have to use "Address Translation" scheme



ADDRESS - TRANSLATION

→ This "page Table" is held in 'RAM', but to get one memory location, ① App → MMU → RAM (PAGE TABLE) → RAM (Address in RAM)

→ It's tedious process, so a "CACHE" is maintained of recently looked up addresses, it is called "TRANSLATION LOOKASIDE BUFFER" (TLB).

PAGE FAULT

When MMU is trying to access a virtual address which is not present in its table, then it rises page fault. It happens because of three reasons

1. App is trying to access address which is not allowed to access and it has not got that memory allocated to it. Results in "SEGMENTATION FAULT"

2. "LAZY ALLOCATION", when App is given memory by Kernel, but it will allocate that address until something is written, So, when MMU goes to Kernel, Kernel says, yes, I have allocated memory to App, these is the address & updates the MMU table and process continues

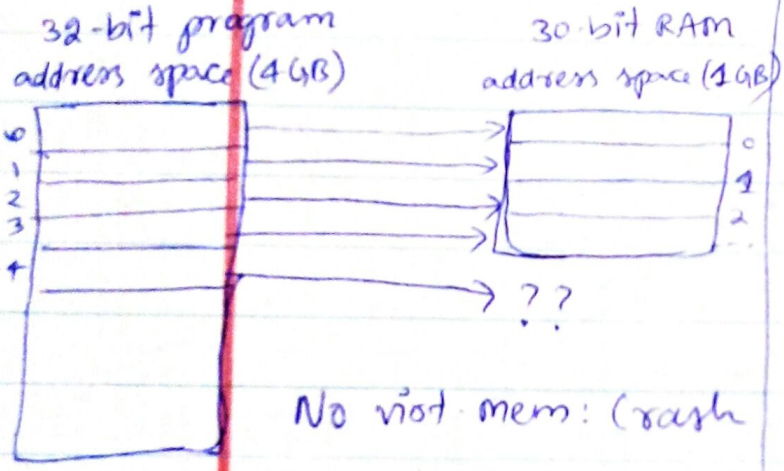
3). When the page is swapped, then Kernel gets the page, & puts it in the RAM, and re programs (updates) MMU table & process execution resumes.

Virtual Memory

- It is a layer of indirection

Without Virtual Memory

Program Address = RAM address

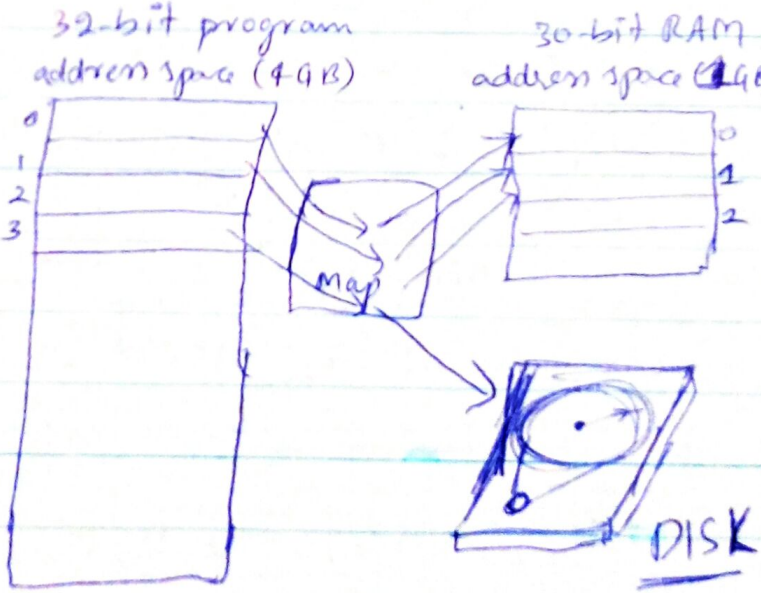


No virt mem: Crash if we try to access more RAM than we have

With Virtual Memory

Program Address Maps to RAM address

- Virtual memory takes "Program addresses" and maps them to "RAM addresses".



* Mapping gives us flexibility in how we use the RAM.

Three Problems Virtual Memory Solves:

1) Not enough RAM
2) Holes in our address space (kind of fragmentation)
3) Programs writing over each other.

What is virtual memory:

→ Indirection

Problems:-

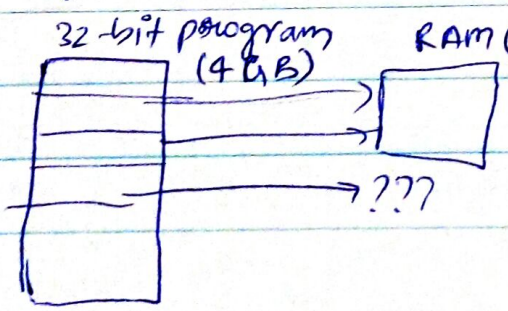
#1. what if we don't have enough memory

- MIPS gives each program its own "32-bit address space"
- Programs can access any byte in their "32-bit address space"

How much memory can you access with a 32 bit address?

⇒ 2^{32} bytes = 4 GB. ⇒ O.S reserves some if it so, it is closer to 2 GB of usable space.

- what if you don't have 4 GB (2^{32} bytes of memory)?



'crash if we try to access more than 1 GB'

(5) → 0.5

#2. Holes in our address space:

- Fragmentation

#3. How do we keep Programs Secure:-

- Each program can access any 32-bit memory address
- what if multiple programs access the same address?

How do we solve this?

- Key to the problem: "Same memory space"
- Can we give each program, it's own "virtual memory space"?
  - ↳ Separately map each "program's memory space" to the "RAM Memory" space.
  - (We can even move some of the program's memory space to disk, if we run out of memory in "RAM") (using hard disk as RAM)

→ This is the **KEY** to the **TERM** "VIRTUAL MEMORY"

(It is not actually the primary memory)

① → Networks

→ OSI Model → Open Systems Interconnection

• It is a conceptual model that characterizes and standardizes the communication functions of a telecommunication or computing system without regard to its underlying internal structure & technology

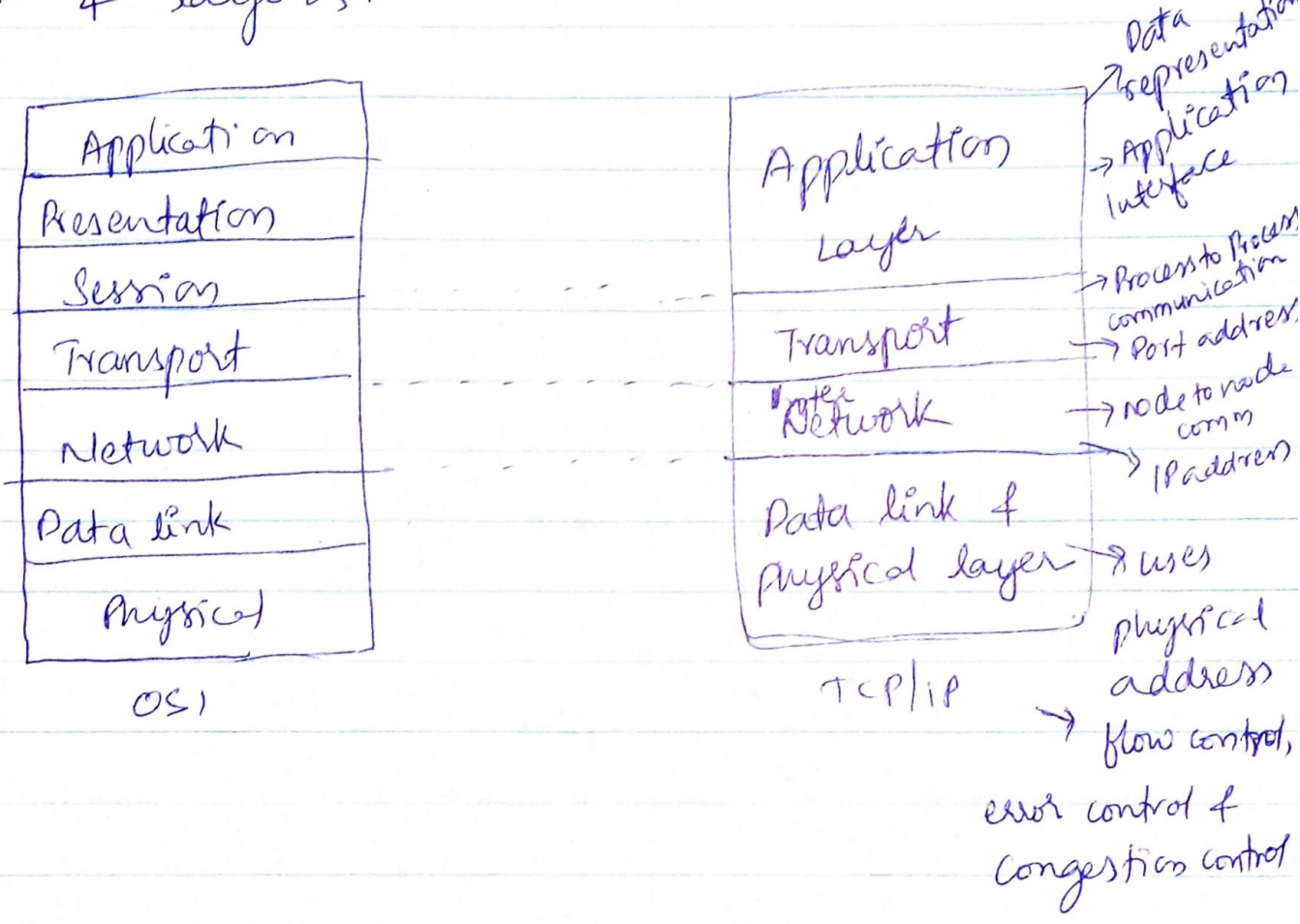
↓↓

Standardize communication with standard protocols

Application.	SMTP, FTP, Telnet
Presentation	Format Data, Encryption
Session	Start & Stop sessions
Transport	TCP, UDP, Port Numbers
Network	IP Address, Routers
Data link	MAC address, switches
Physical	Cable, Network Interface cards, Hubs

TCP/IP Reference model

- Transmission Control Protocol / Internet Protocol
- 4 layers.



→ **Application layer:**

Protocol: set of rules

- HTMP, SMTP, RTP, DNS

* Provides interface to the user
* Present data in user readable format
* Dat Encapsulation, Encryption / Decryption

Transport Layer:

- Process to Process communication
- It acts as an entity between the network layer & Application layer
- Provides error control
- Multiplexing & De-multiplexing
- TCP, UDP → User Datagram protocol
  - ↳ connection less protocol

→ Transmission control protocol
→ connection-oriented protocol.

③ Inter-network layer:

- Allows host to inject packets into any network
- Protocol: IP , ICMP

IP → proper delivery of the packet to the destination

ICMP →
- Error reporting protocol
- Internet control messaging protocol:
  - Used by n/w devices, routers, to send error messages & operational info. for example: a requested service is not available or host or router could not be reached

4) Network Access layer: (data link + physical layer)

- error control
- flow control
- Access control
- bit conversion.