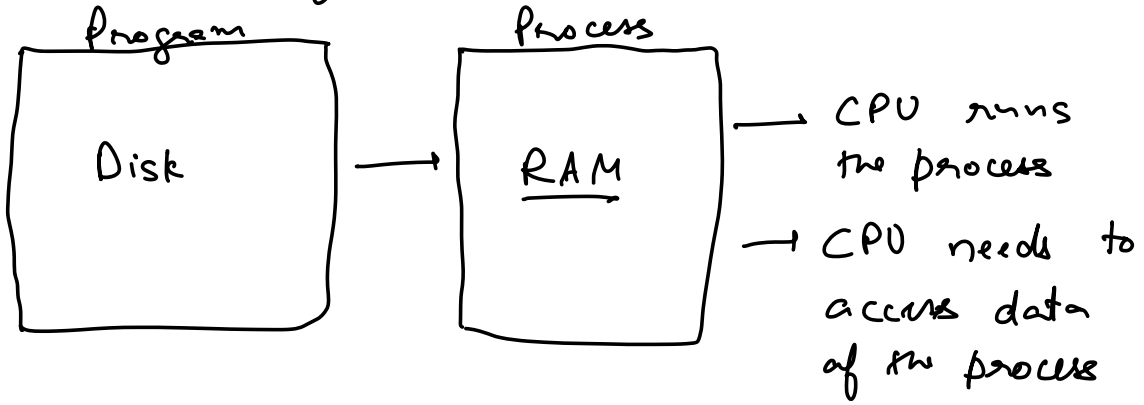
1. Good Evening
2. We will begin at ~~9:00~~ 9:10 pm
3. Topic - Memory Management

Agenda

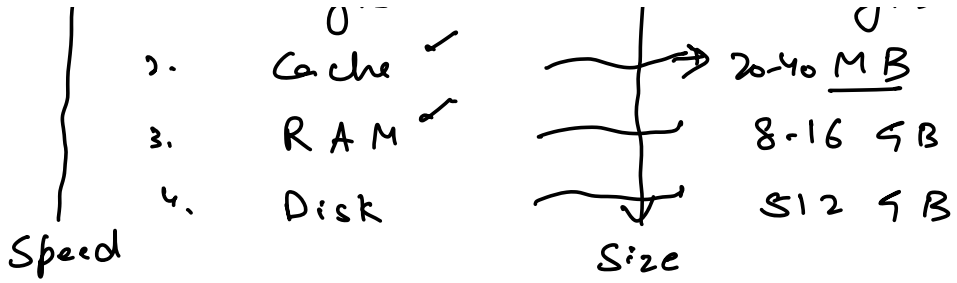
1. Recap on program vs process
2. Storage Devices
3. Fragmentation
4. Paging [Logical vs Physical Addresses]
5. Page Fault
6. Thrashing
7. Belady's Anomaly.
8. ☆ Concurrent DS

Recap → Program vs Process



Storage Devices in a Computer

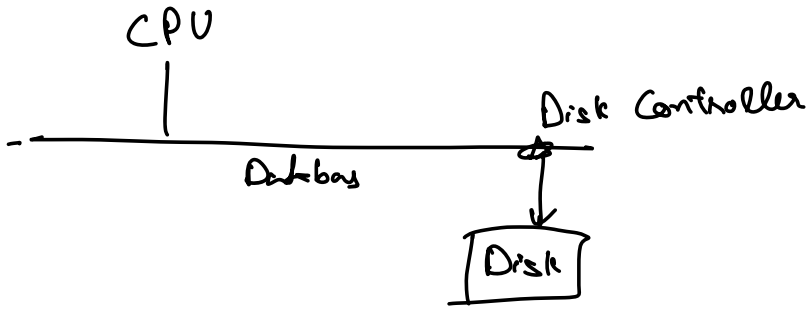
↑ 1. Register → 16 bytes



* Volatility vs Persistence
  ↓
  If we switch off & turn on the machine again, data is not lost

Persistent → Disk
Volatile → Register, Cache, RAM

* Which ones can the CPU access directly?
  - Register, Cache, RAM
  - Disk can't be accessed directly by CPU



* If CPU needs to access something on the disk, we will first need

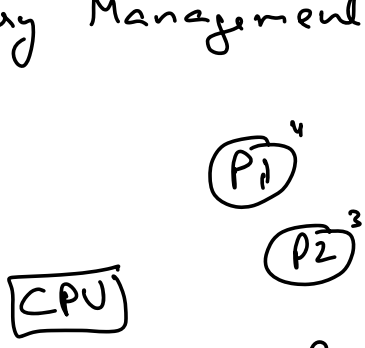
to bring it to RAM

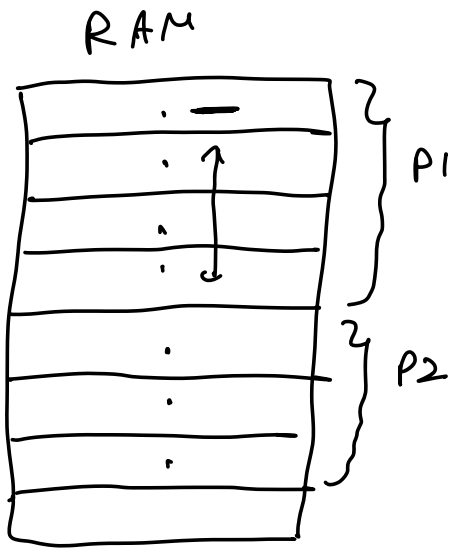
Questions

* How is it possible for an OS to run a huge process e.g. Games
  - RAM → 4 GB
  - 64 GB

* How is it possible for an OS to run a large no of processes
  - Memory need in excess of the RAM on the system.

Memory Management





1. Contiguous memory allocation

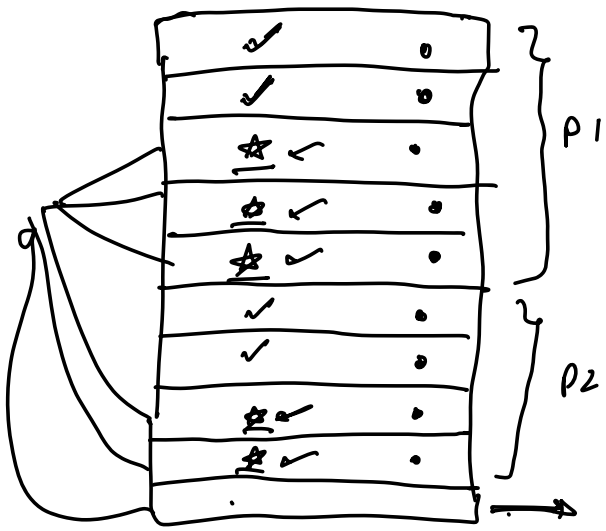
Advantages

1. Faster : Access via index
   OS will read something from RAM, it will bring nearby blocks to the cache.
2. Simple
3. Process knows where data lives physically

Disadvantage

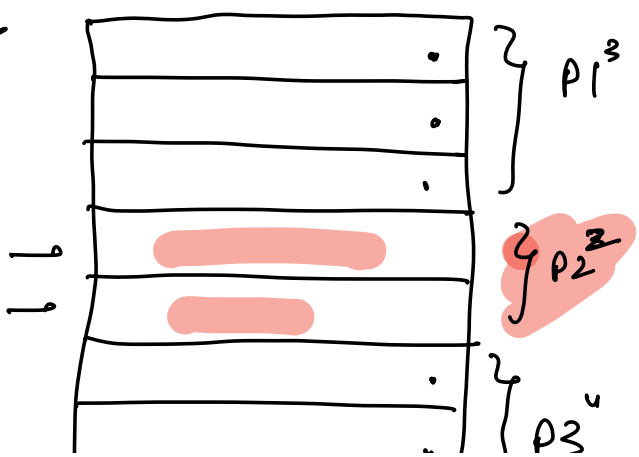
1. Memory Wastage

1.1.



P1⁵, P2⁴, P3 (4)

1.2.



P4 (3)

★ Fragmentation

↳ Free RAM is available but is fragmented / divided in many parts.

→ Problem : Memory Wastage

★ How to run a process which requires 6 GB, but we have 4 GB of RAM

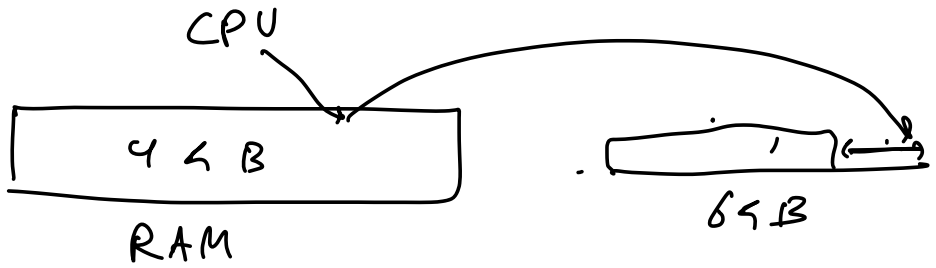
★ How to run 10 process of 600 MB ~ 6 GB but we have 4 GB of RAM

PAGING

★ Store what is possible in the RAM & let the rest remain on DISK

★ Whenever CPU wants to access the part on the disk, bring it to RAM first

MMU [Memory Management Unit]



Details of Paging

1. Two types of addresses

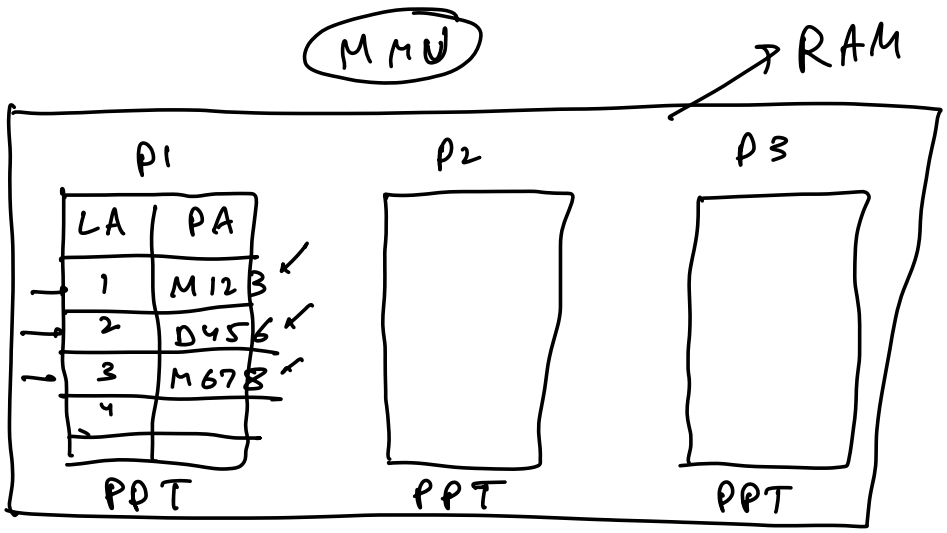
- Logical Address or Conceptual Address
- Physical Address or Real Address

→ An application / process is aware only of the logical addresses.

→ On the process start, MMU allocates logical address to the process.

→ MMU maintains a (table) to map these conceptual addresses to physical addresses.

☆ A table for every process → Process Paging Table



☆ Memory Constraints are solved

RAM of 4GB → Physical address limited

2GB of RAM
10GB of Disk

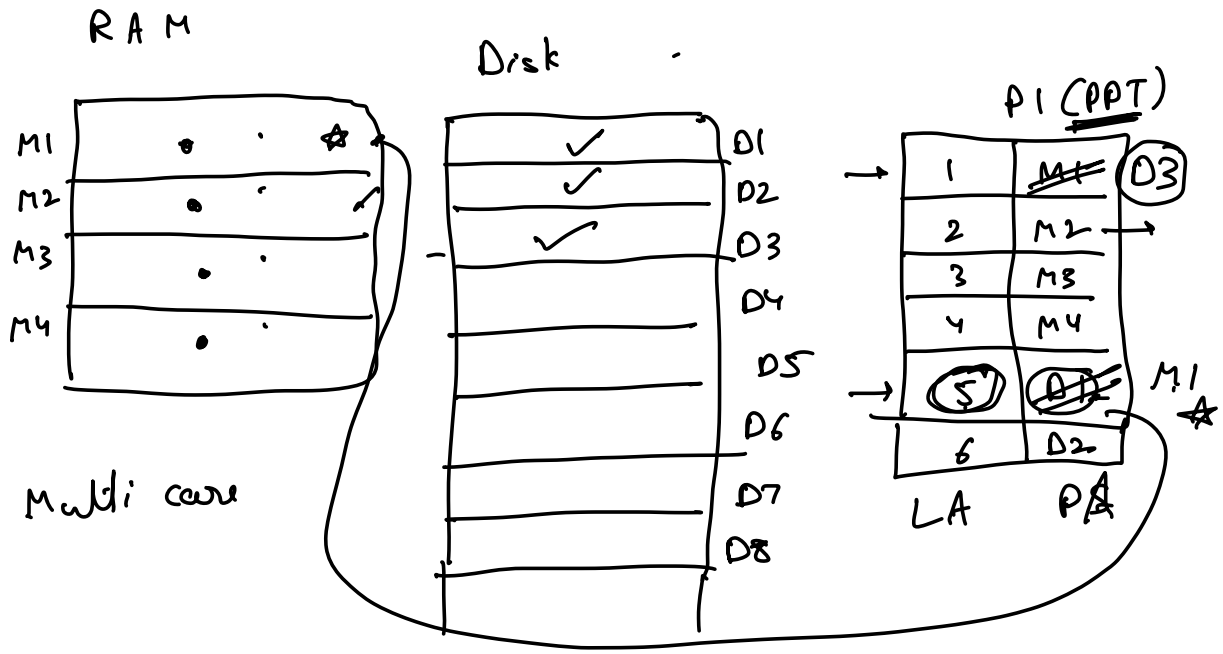
5GB + Same

PPT

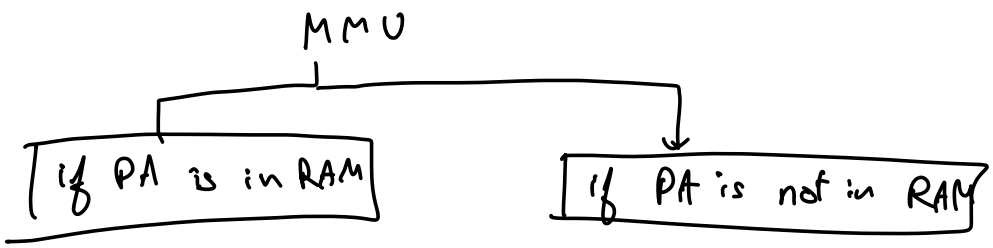
LA	PA	
1.	M:—	2GB
2	M —	
3	M —	
4.	D:—	3GB
5	D —	
6	D —	

★ How will CPU actually access physical addresses

↳ ★ CPU can only access RAM & not disk



1. CPU wants to access LA 1

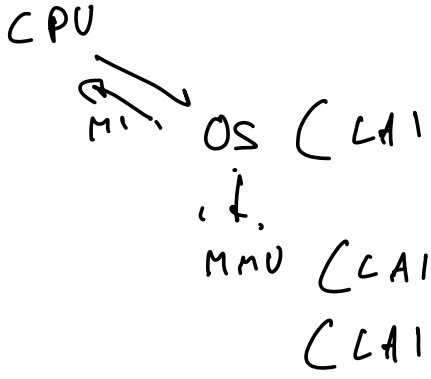


2. CPU wants to access LA 5

M1 ⟶ D3

D1 ⟶ M1

CPU can now read from M1

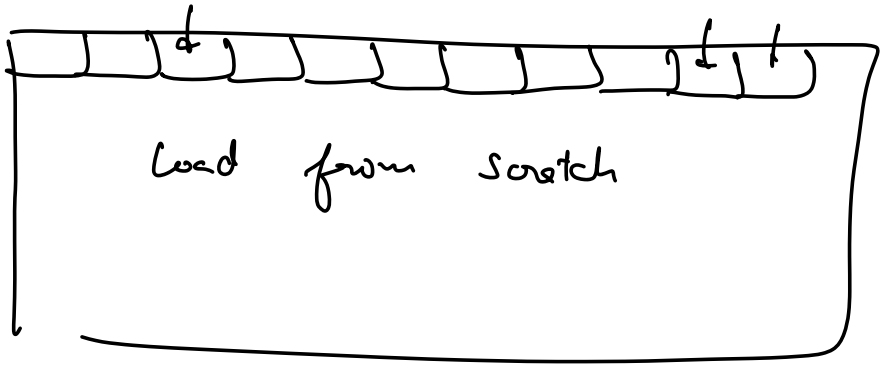


* 1. CPU is accessing a LA mapped to RAM's PA ✓

2. CPU is accessing a LA mapped to Disk [Page Fault]
   1. Page Replacement Algorithm [
   2. Move a block from RAM to disk
   3. Move the necessary block from disk to RAM.

1. What is paging?
2. What is Process Paging table
3. LA vs PA

4. How does a CPU read via a CA
   - L 1.
   - L 2.

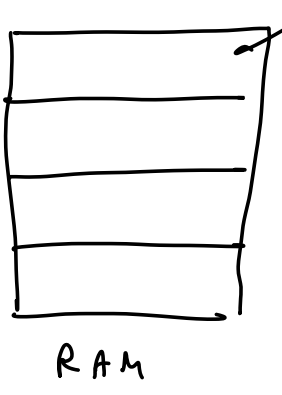


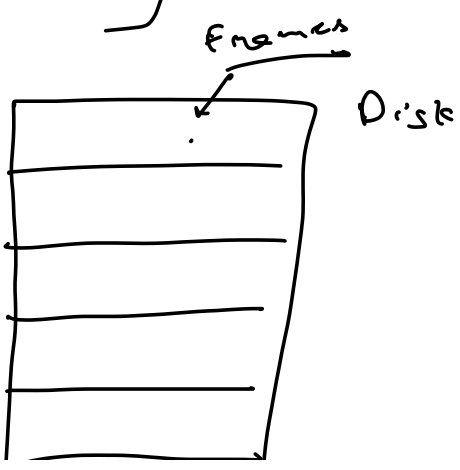
PPT → MMU

Break Here

10:23 → 10:32

1. Page Replacement Algo
2. Thrashing
3. Belady's Anomaly





1 Page size = 1 Frame Size

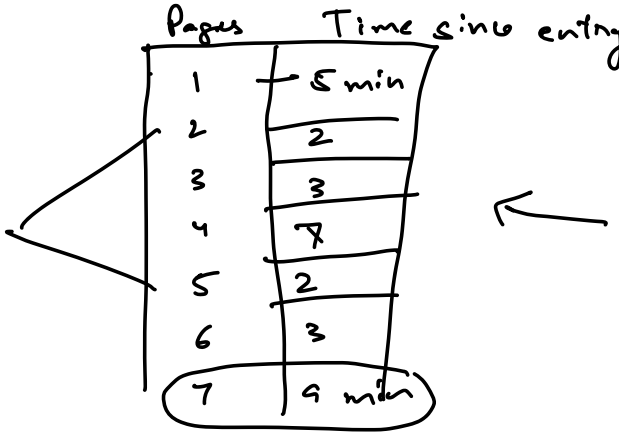
frames > # pages

* Page Fault → Page Replacement will need to be done

Page Replacement Algorithm

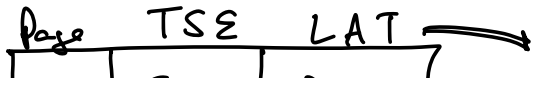
1. FIFO

Replace the page that exist for the longest time



2. LIFO → Either move Page 2 or Page 5 out

3. LRU → Least Recently used



1	5 min	2
2	2 min	1
3	3	2
4	7	3
5	2	1
6	3	2
7	9	1

when CPU accessed this Page

Wisdom → Page which has not been used for a long time is less likely to be used in future

Assignment → LRU algorithm.

↙ Interview?

↘ Pg to Pg Use?

Thrashing

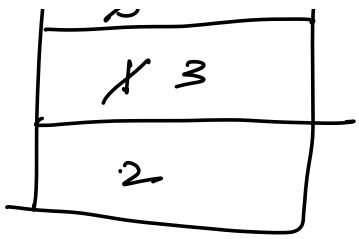
Run LRU once

Logical → 3 1 2 4 3 1 4 2 1 4 2 3 4

Address ✡ ✡ ✡

RAM

4



Page Fault?
Page Replced?

A lot of page faults → A lot of page replacements → Slower performance

Thrashing → A lot of page faults are happening :o CPU is doing less work △ is more busy waiting for memory updates.

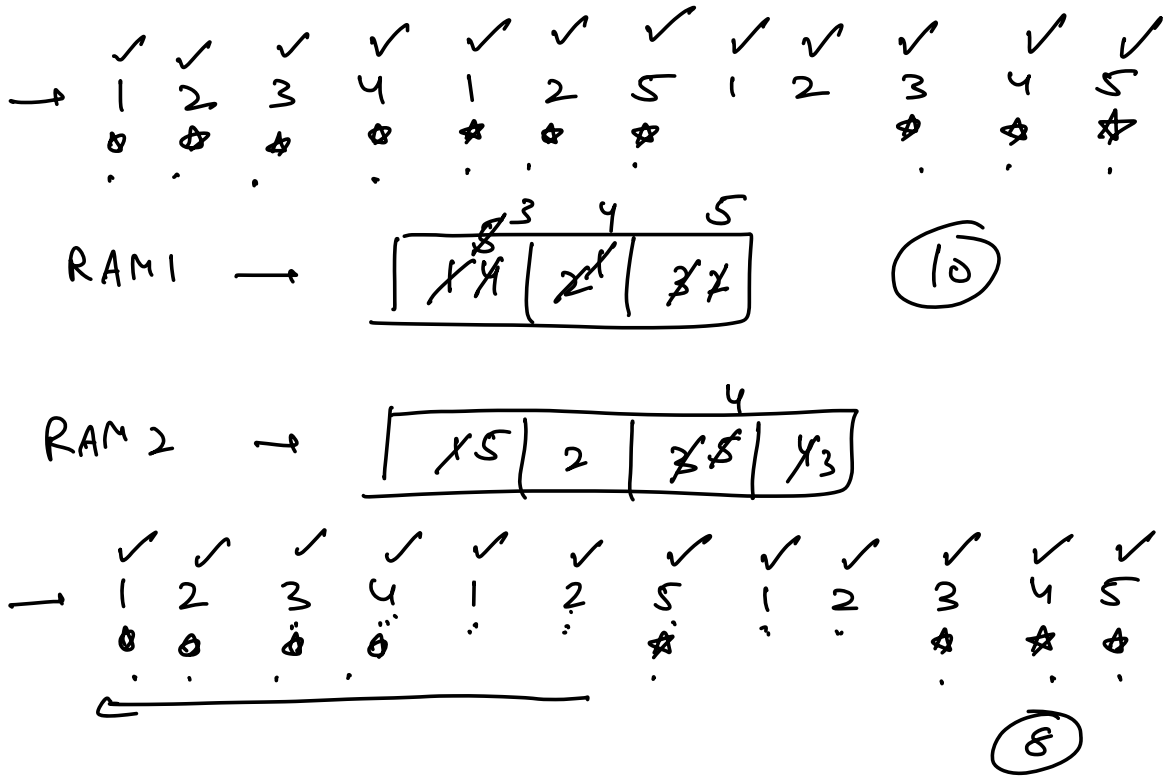
Reasons

1. Low RAM
2. [Memory need is excessive]
   - 2.1 Running a huge program
   - 2.2 Running a lot of processes.

☆ Belady's Anomaly [Can happen]

with all Page Replacement algorithms

* Larger RAM may lead to more page faults.



Such a sequence is possible for which larger RAM may give more page faults

Larger RAM generally (not always) means

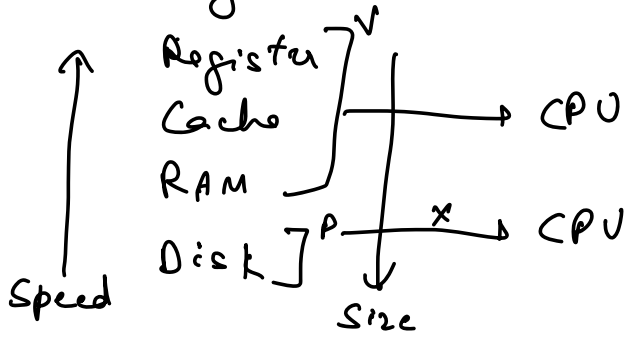
less page faults.

* In a certain sequence it is possible that larger RAM leads to more page faults
  - Belady's Anomaly.

Summary

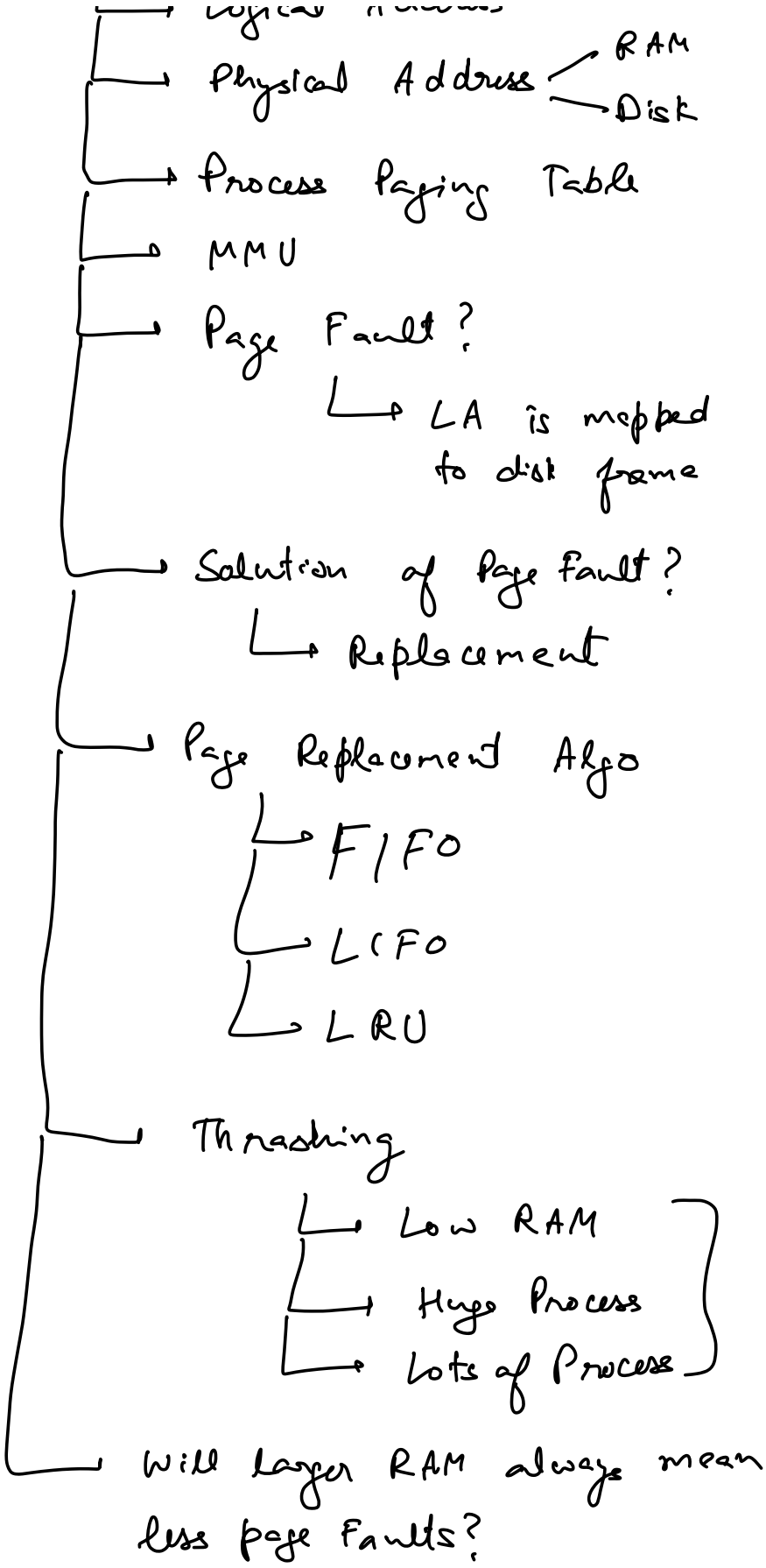
1. Program —Disk→ Process —RAM→ CPU

2. Storage Devices



3. Contiguous Memory Allocation
   - Memory wastage (Fragmentation)
   - Huge program won't run

4. Paging
   - Logical Address


Physical Address
RAM
Disk
Process Paging Table
MMU
Page Fault?
LA is mapped to disk frame
Solution of Page Fault?
Replacement
Page Replacement Algo
FIFO
LIFO
LRU
Thrashing
Low RAM
Huge Process
Lots of Process
Will Larger RAM always mean less page Faults?

↓

Belady's Anomaly.

Monday's Lecture → DBMS

1. SP
2. UDF's
3. Deadlocks
4. Full-text Indexes
5. Doubts — Schema Design
6. Semaphore questions.

- ☆ Revise 1 day's notes
- ☆ Solve 1-2 old question