Starting at 9:05 PM

Operating Systems

Agenda

① Synchronization

→ Peterson

→ Mutex

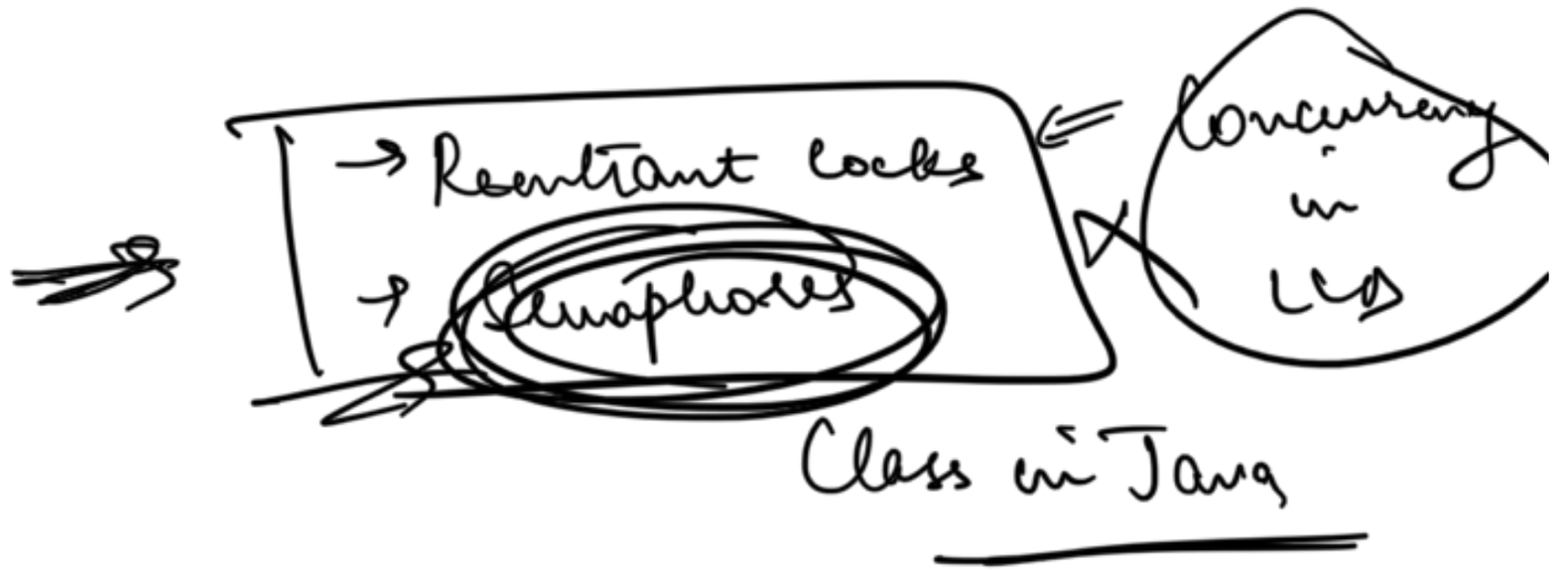
→ Semaphores

→ Implement Sync in Java

Adder | Subtractor

GIT

Synchronized



→ Deadlocks

→ Ways to handle deadlock

→ Way most OS use

→ Memory Management

→ Pages & Frames

→ Thrashing

→ IPC
- Pipes
- Shared Memory

Synchronization

→ Multiple threads that are working on a shared data in consistency

→ Peterson's Sol

[If the other person has their flag on
& it is the turn of other, I should
wait]

Atomic int
Synchronized
Adder {
Counter
Cached Value
add() {

Subtractor {
Counter
Cached
Subtract() {

Real Value

if flag[0] != 1

flag[1] = 1

turn = 0

Cached Value

flag[1]

turn

flag[0]

turn

flag[0] = 1

Cached Value

Practical Lab

1. Mutex ⇒ Mutual Exclusion

Lock

Semaphores / OS APIs

Mutex

Mutex(mutex)

Guarantee consistency & Atomicity

T1

while(!mutex.acquire())

CS

T2

while(!mutex.acquire())

CS

// shared var

mutex. release()

mutex. release()

→ Only one thread can acquire a lock at one time.

mutex. acquire() → true → if you got the lock

→ false → otherwise

while (! mutex.acquire()){

}

CS

mutex.release();

I have a shared printer

OKAY if 6 threads are accessing

in parallel but not more

Semaphore → Multi utili a counter

Semaphore s = Semaphore(n)

while(!s.acquire()) // < n threads have gotten the lock

}

1. release()

MUTEX: Only one thread will acquire

SEMAPHORE → N threads can have lock at the same

Create a new thread for every request

Server

10000 → 10000

Semaphore (500)

Only 500 parallel threads can be served at a time


1
Server
2 Take a look over kafka
3 Event
(Only 500 people can)
Server
15
10000
A
A
A
1 acquire lock over
semaphore
2 acquire lock over

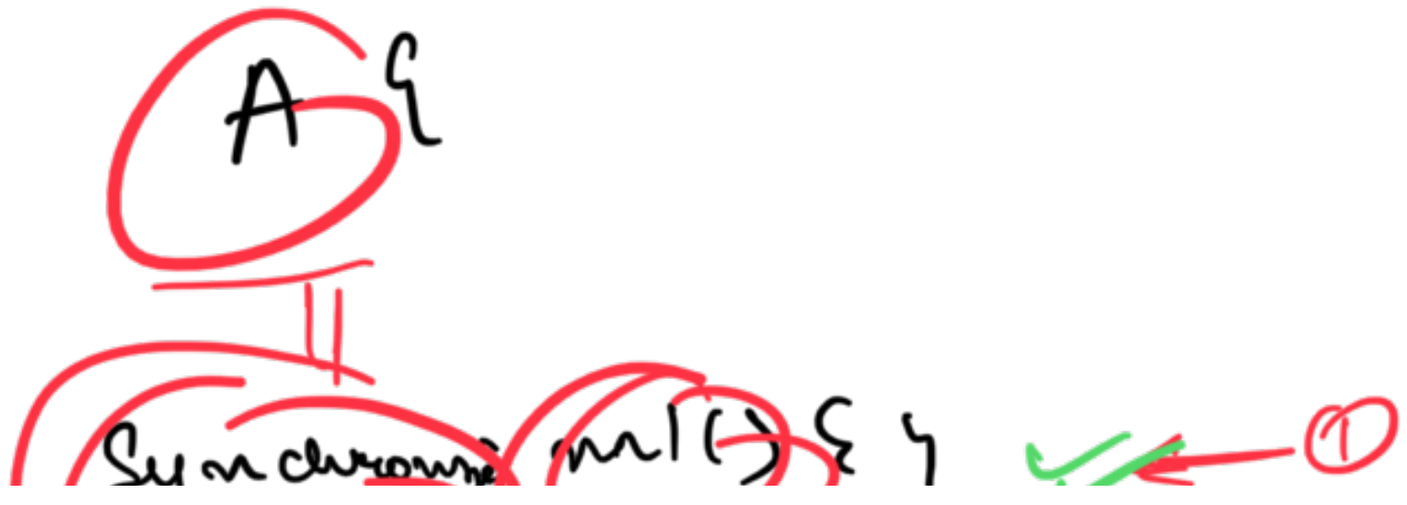
Only one thread can acces

Mutex → Semaphore with N=1

Synchronized

→ We don't want multiple people to Manipulate the same object at same time

Synchronized



} ← 2

→ Only one thread will be able to go inside the complete set of sync methods of a class

m2() { } ← 4

synchronized m3() { } ← 3

}

T1

→ A.m1()

— A.m3()

}

T2

⇒ A.m3()

}

ON ONE OBJECT

... COMPLETE SET OF SYNC methods

OVER THE [illegible]

only 1 thread can be ~~doon~~

A {

② m1()

m2()

① m3()

a1.m1()

a1.m2()

a1.m1()

a1.m1()

a1.m1()

a1.m1()

a1. m1()

a1. m1()

a1. m2()

a1. m1()

a1. m2()

a1-m1()

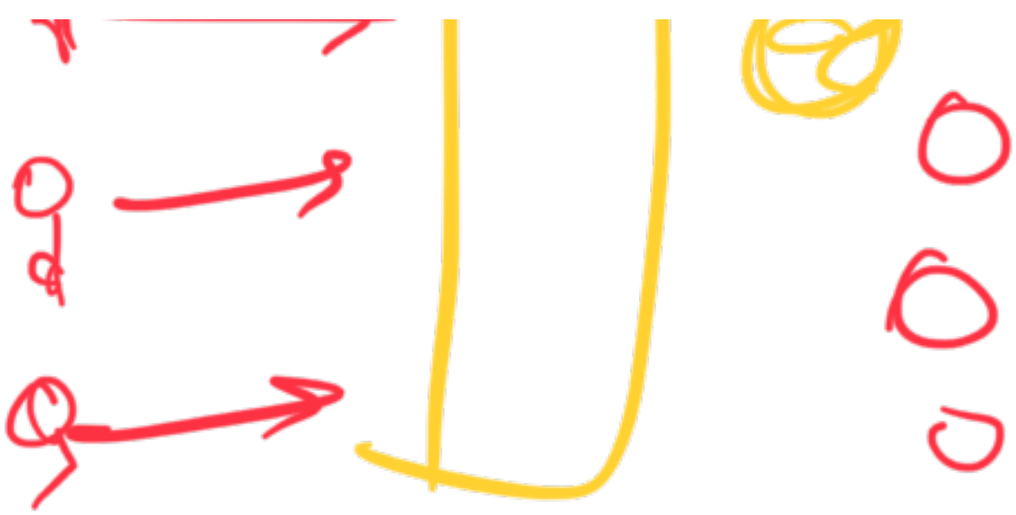
a2. m1()

01. Abc "hi"

01. Abc = "hello"

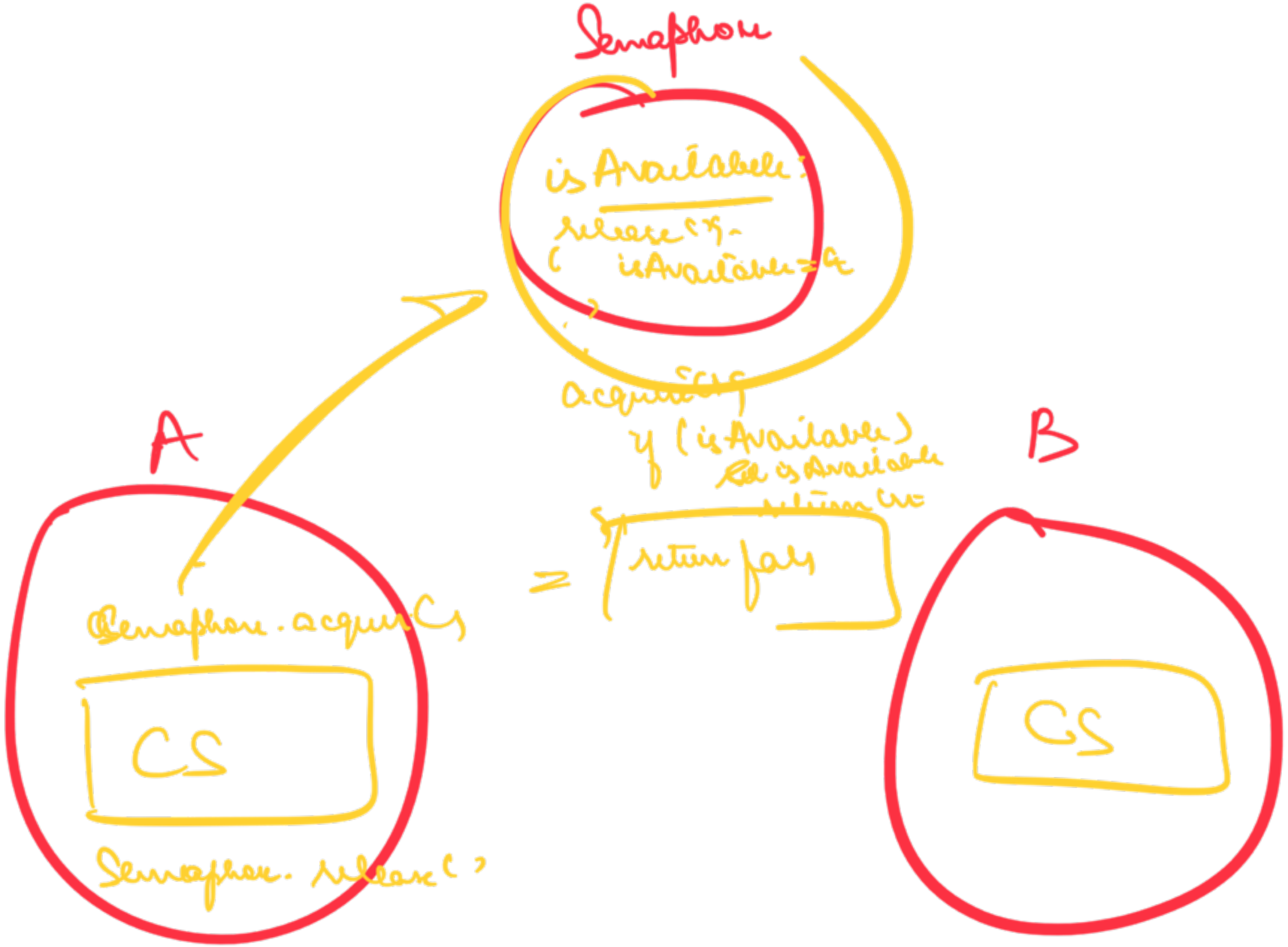
synchronized set Abc() {

}



HashSet

- add()
- get
- put()


Semaphore
isAvailable
release()
isAvailable = true
acquire()
A
B
Semaphore.acquire()
CS
Semaphore.release()
=
return fast
CS

Semaphore ⇒ Ticket

One person can have ticket at one time

SA. release()

SB. release()

SA. release()

SB. release()

Deadlock: When the whole system comes into a pause because diff threads are waiting for resources that other have lock over.

How will you take deadlock

Manage Deadlock

Prevent

Avoid

Recovery

Ignorance

(Lockdown)

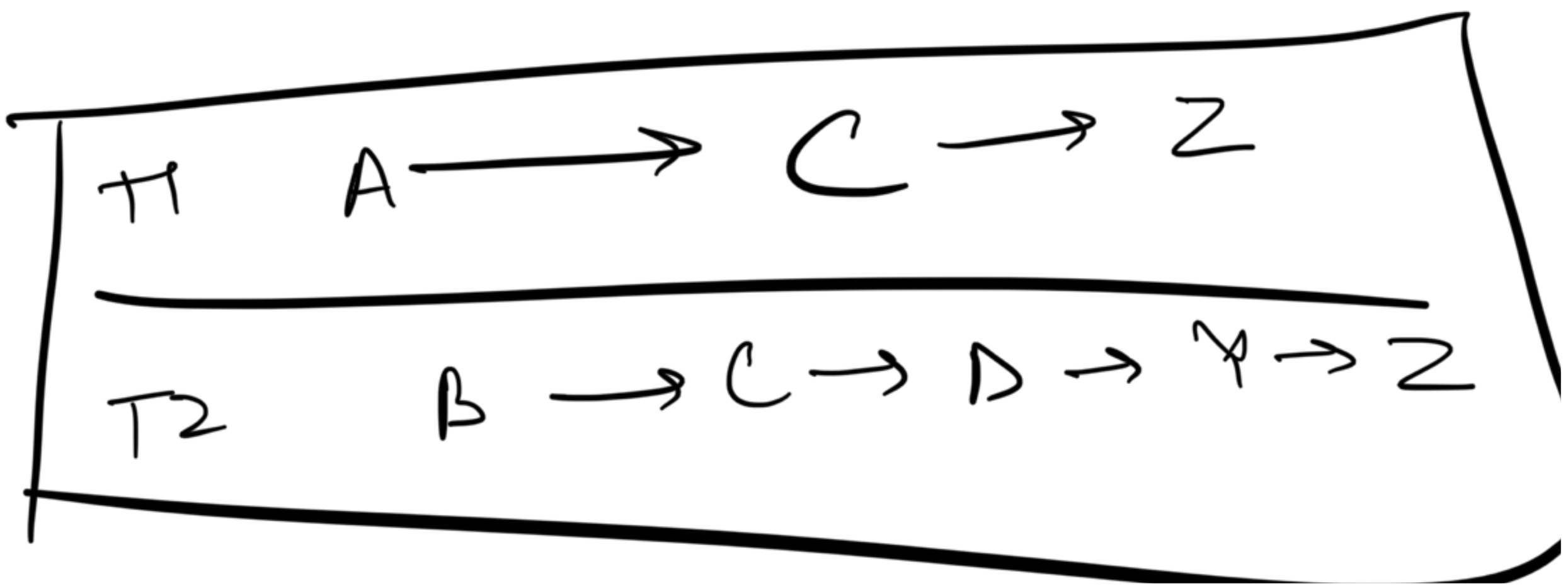
(Mark/Vacuum) (So cool bro)

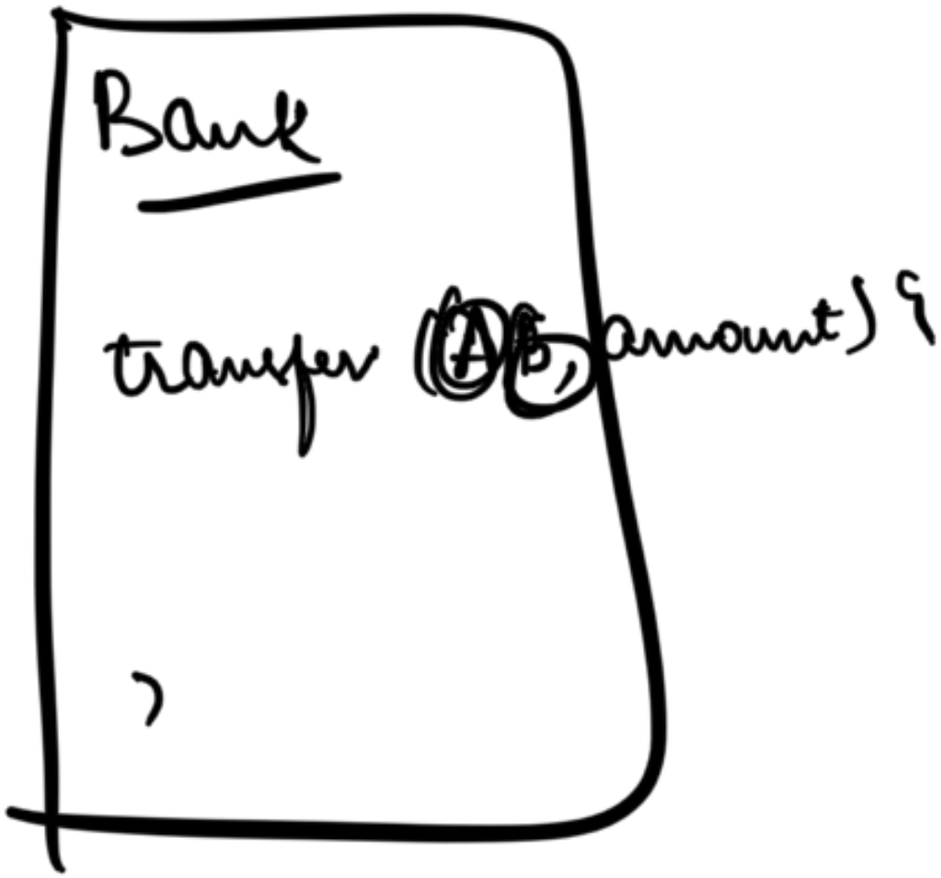
Multiple algo for each of

OS uses these days

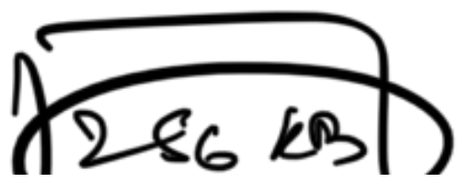
Prevent

⇒ Take lock in alphabetical orde





Memory Management ⇐



512MB

2004

8GB

CPU

RAM

Storage

when a process needs CPU

Swapping

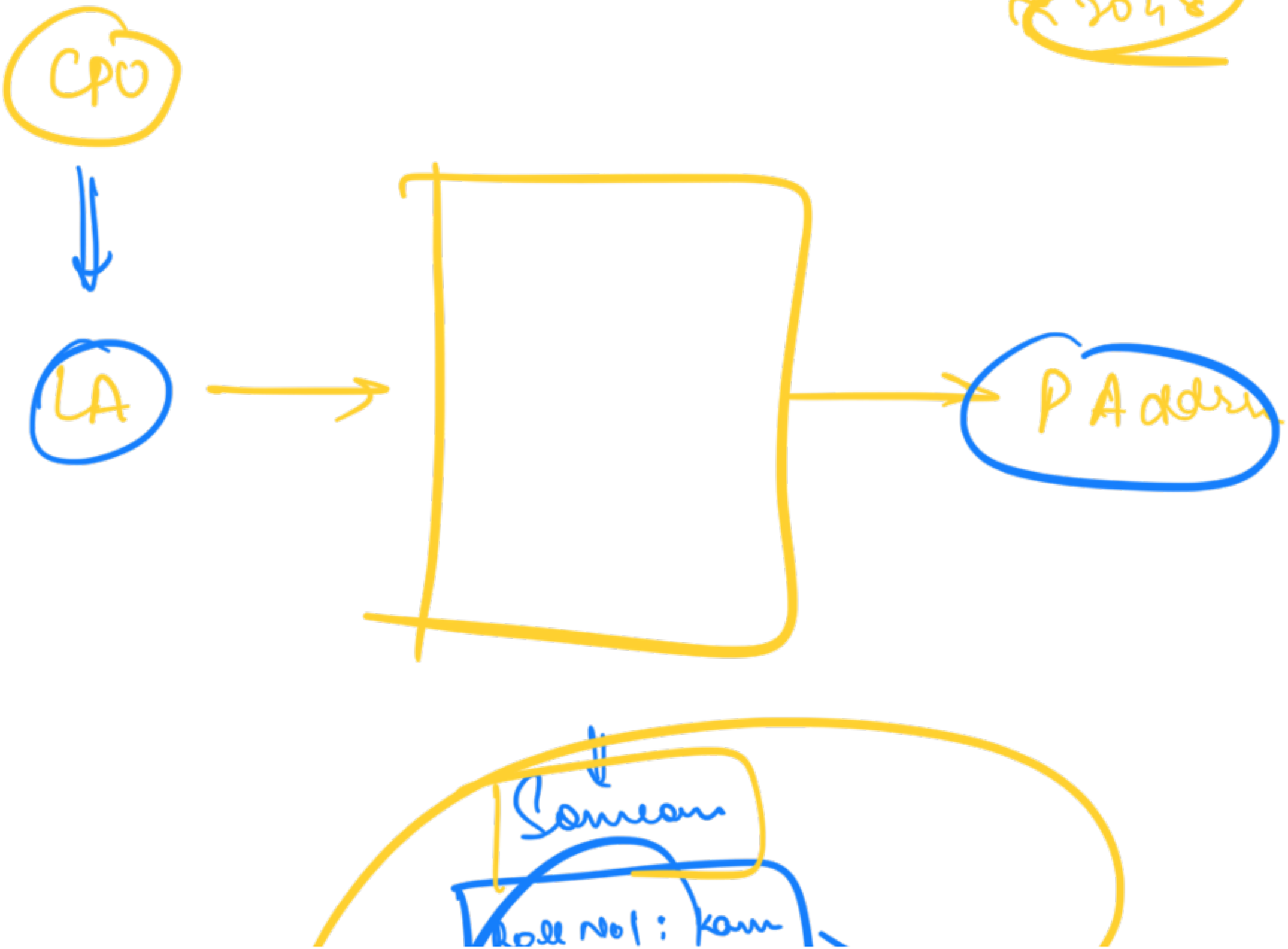
2 types of address

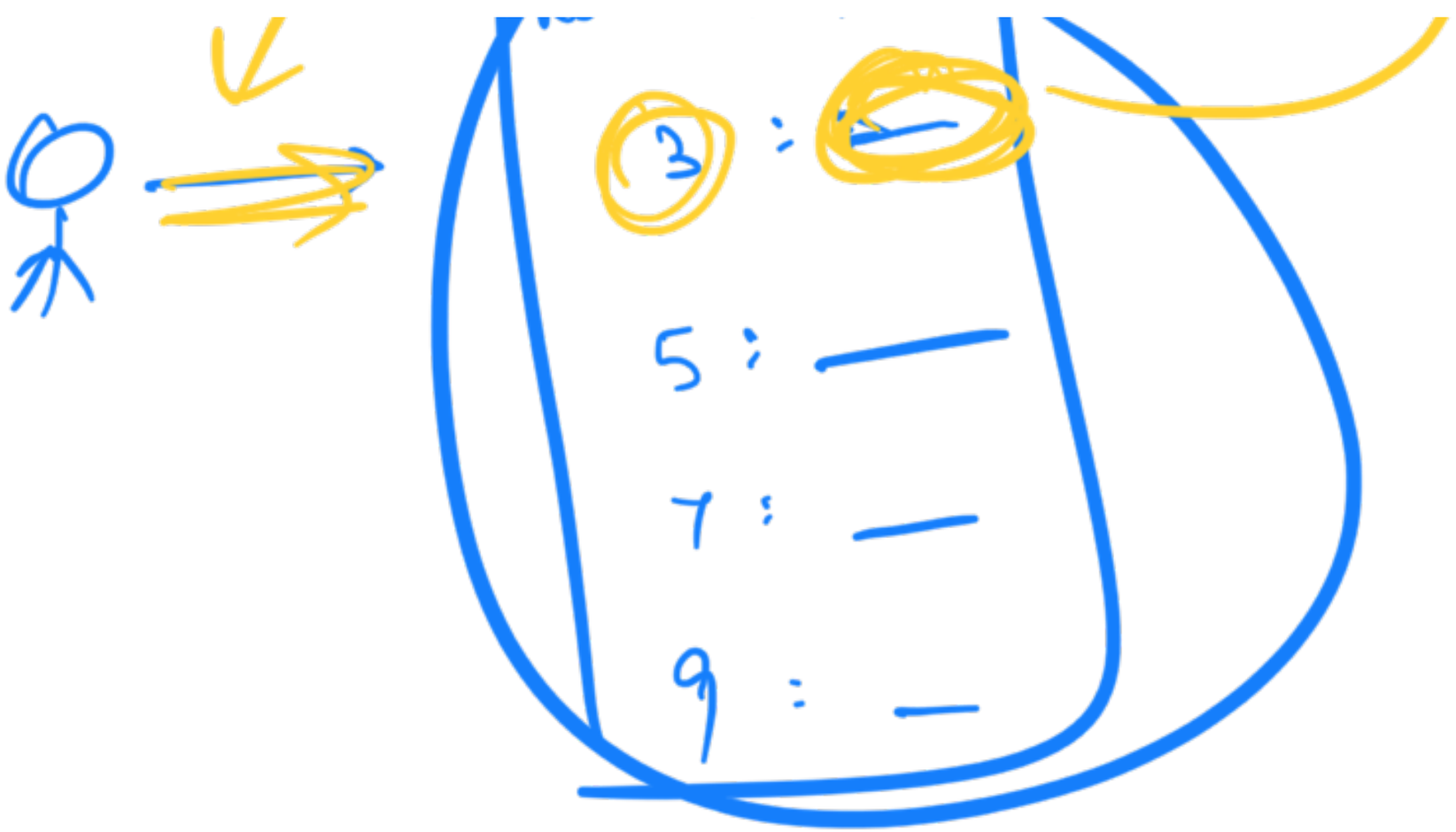
① Logical Address : Address that procs in CPU knows about

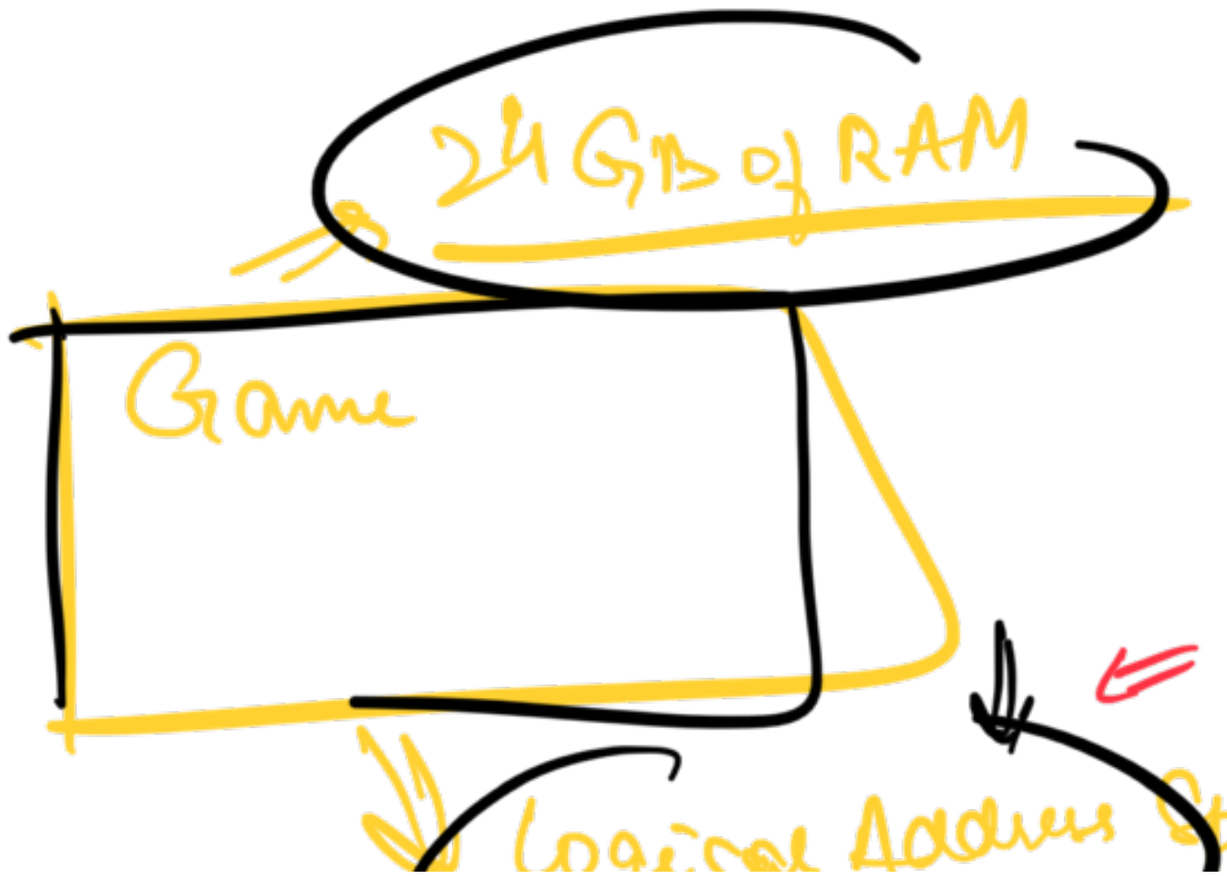
② Physical Address : Real address of data in memory

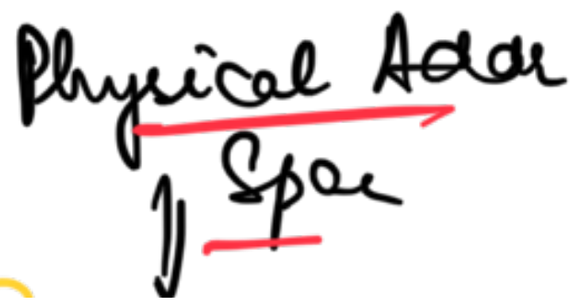
2 MB

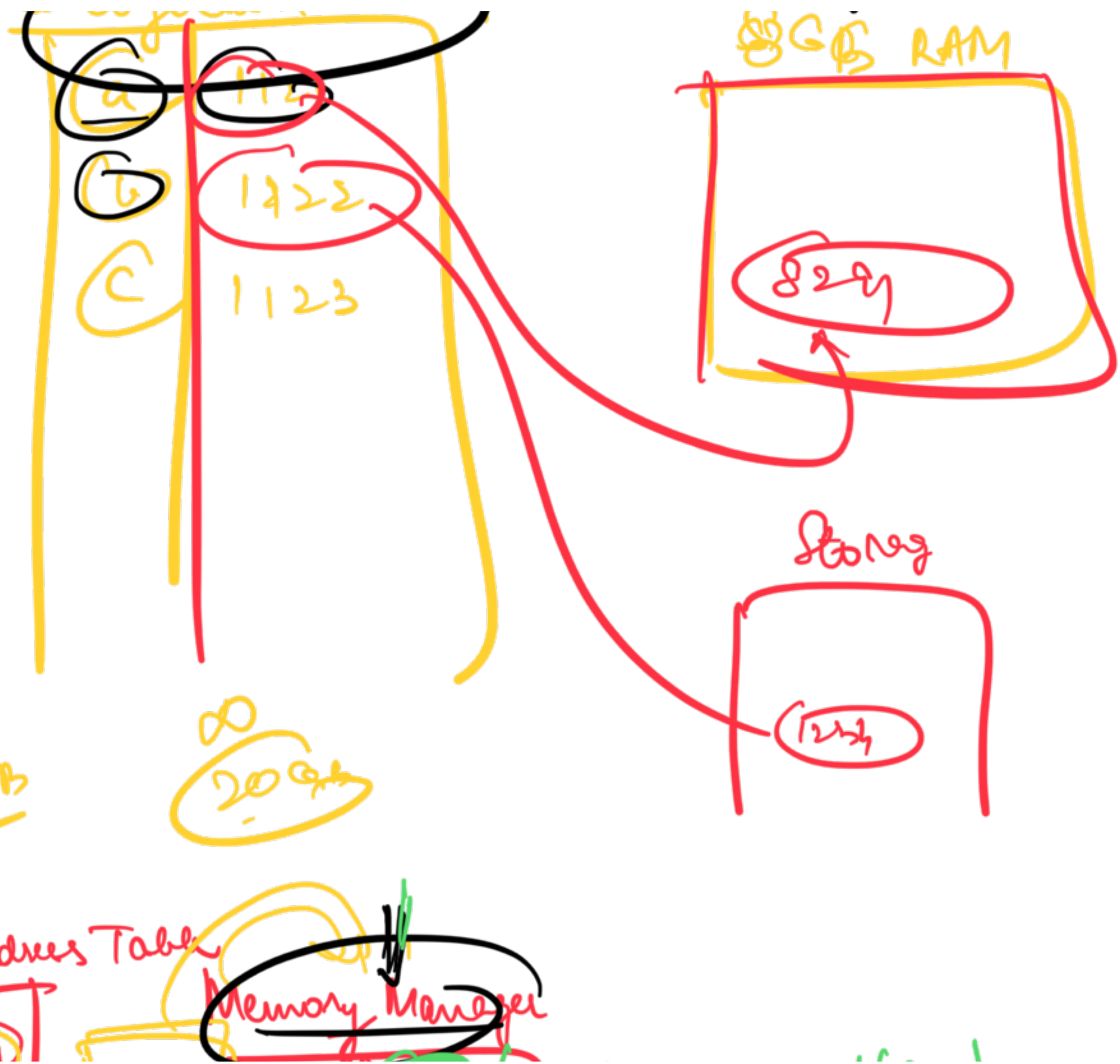
1 ⇒ 1KB 2^1024 KB


CPU
LA
PA addr
Name
Roll No: Kam


3 :
5 :
7 :
9 :


24 GB of RAM
Game
Logical Address Space


Physical Addr
Spa


8GB RAM
829
Store
1234
100 GB
20 GB
Logical Address Table
Memory Manager

CPU

LA

PA

(addr of page/part

ABC

8941 HD

HD

ABC

8941

int cnt

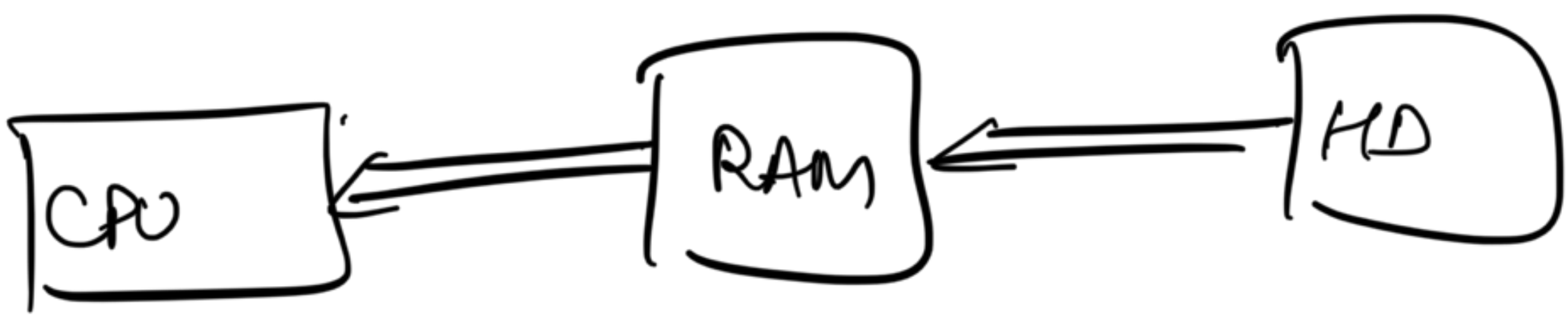
&B

Store 123 at locan ABC

If I Store in Hard Disk ⇒ Slow

⇒ RAM is full

⇒ Some other process needs to run



To fetch from HD

① load onto RAM

② Swap something out

Which data to swap?

Page Replacement Algorithm


Memory
Page
2
3
4
5
6
7
Storage
Frame
3
4
5
6
1KB
Size (page) = =
Size (frame)
1KB

word size → 32 bit

64 Bit

Process wants to access address ABC

Page Fault →

But the address is not in memory

MM will swap from Disk

CPU

Virtual Memory

Thrashing

MM

When PF ⇒ load from storage

→ take time

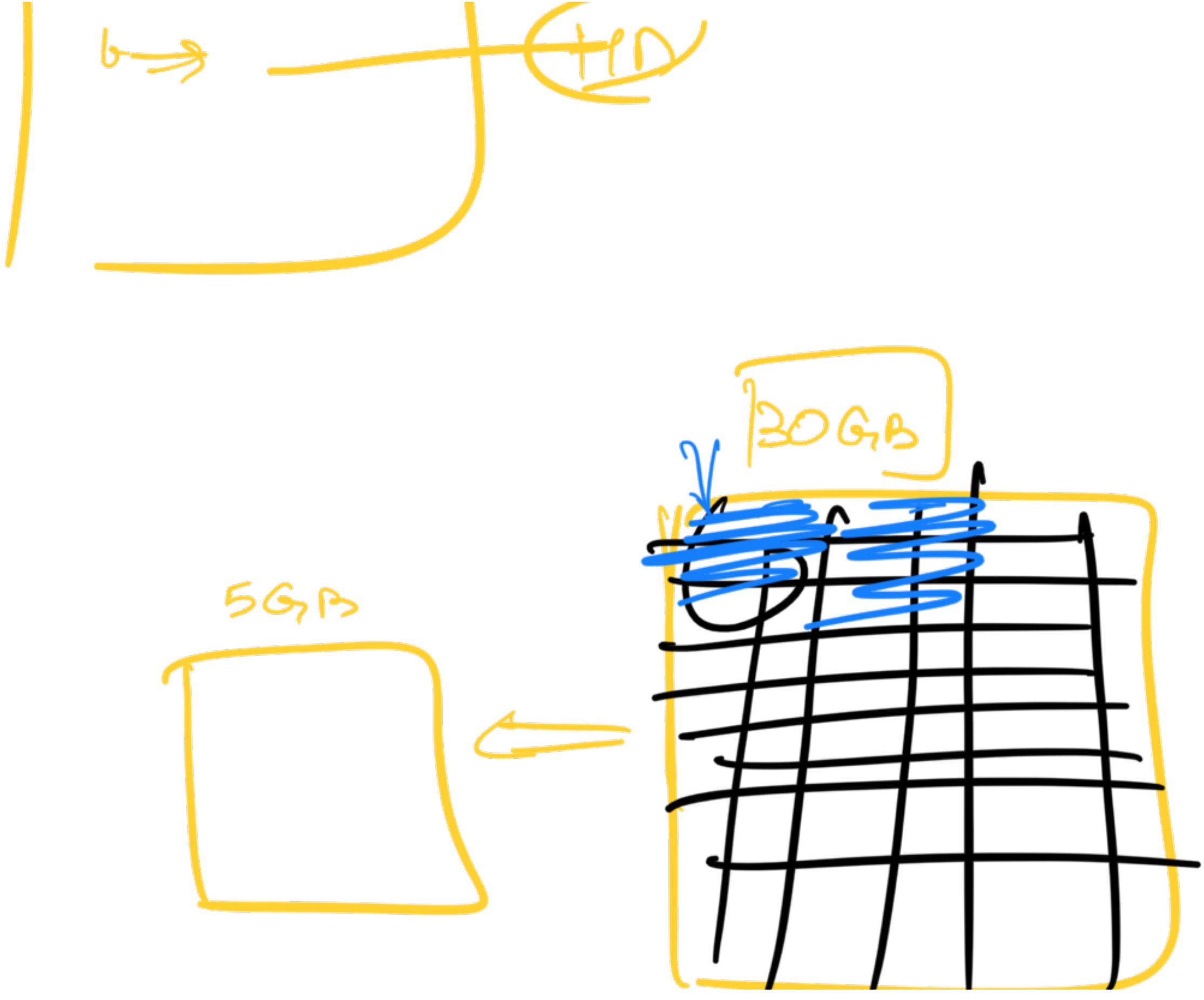
→ replace another page

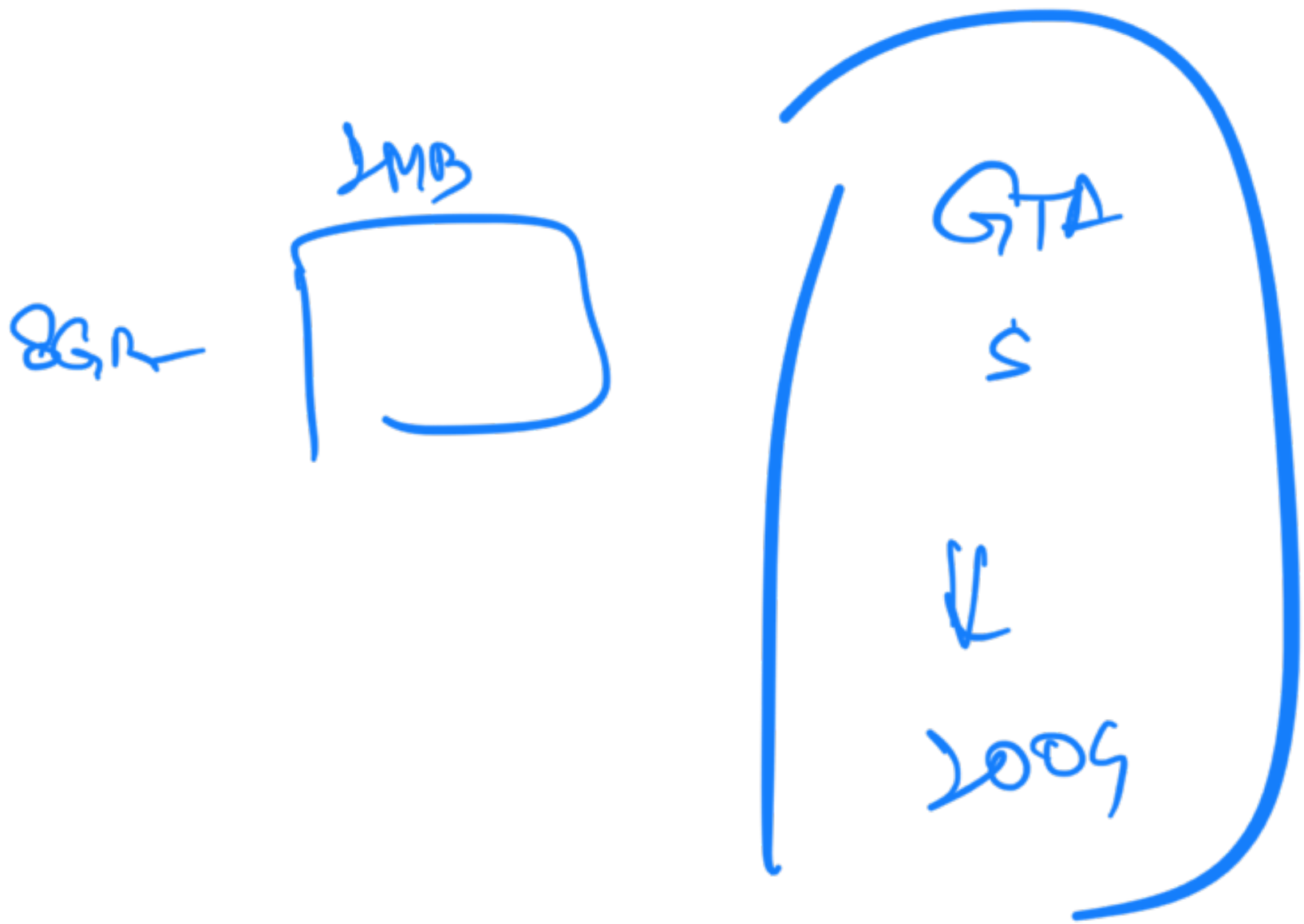
Secondary Mem

Class

a →

RAM


30GB
5GB


LMB
GTA
5
2005