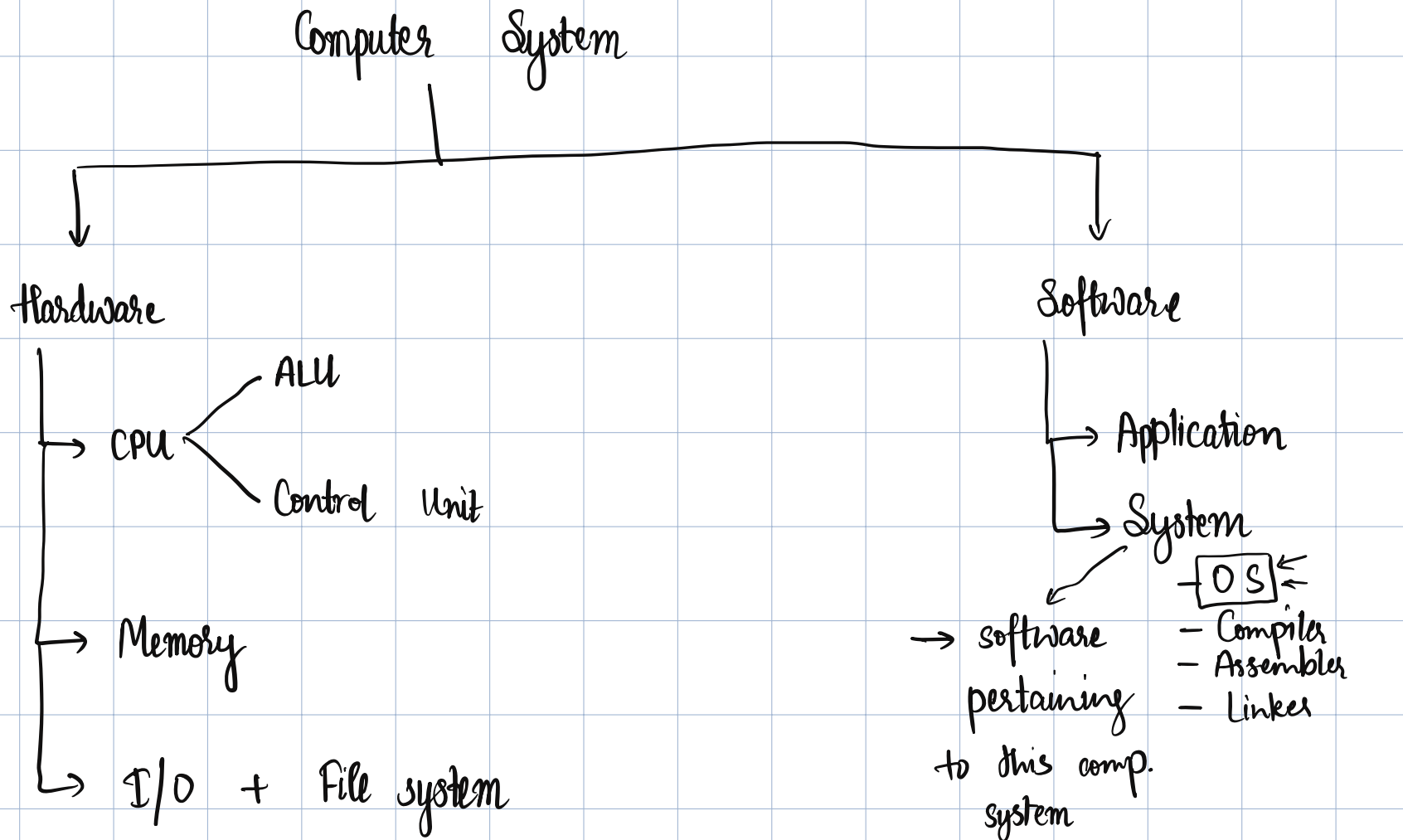
2024/10/22 - Operating Systems - Week 01

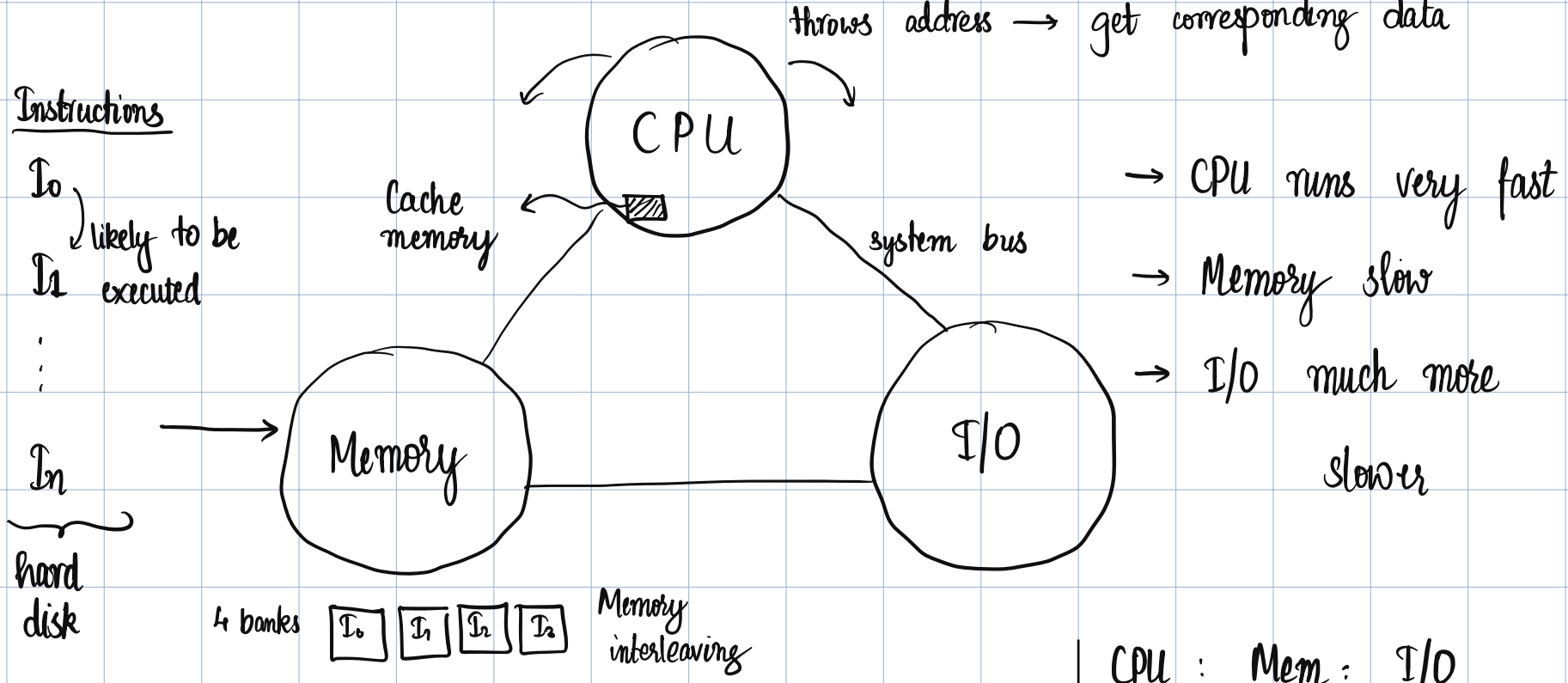
Textbooks : - "Operating Systems" → Andrew S Tenenbaum
→ Silber → OS-II
→ Stallings

1 final examination → Nov 24 sun (9 - 10:30)



* O.S manages all the hardware resources efficiently (CPU, memory, I/O, file sys)

- keyboard, mouse, display → cannot be directly connected to a computer
  → need drivers



→ CPU runs very fast
→ Memory slow
→ I/O much more slower

→ Cache memory → cannot be large (cost of designing)
→ locality of reference.

CPU :	Mem :	I/O
1	5	100

time

↓

CPU idle most of the time

* Temporal and spacial locality

CPU	Mem	I/O
1	5	1000

→ Cache

→ instruction cache

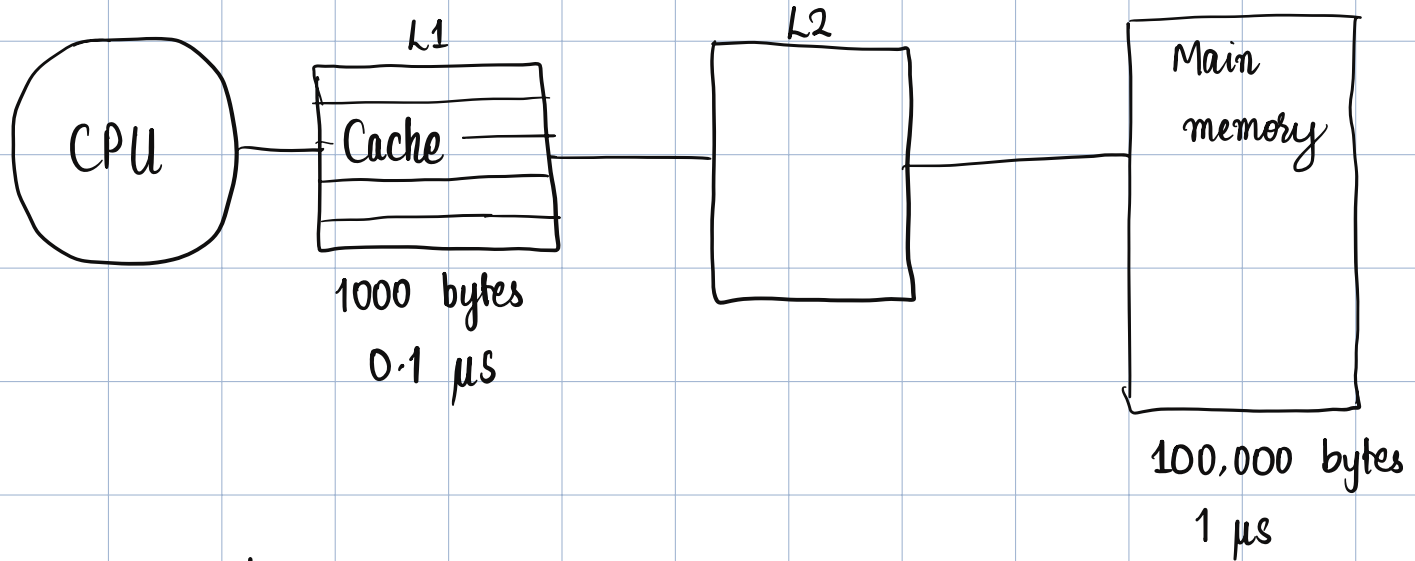
→ data cache

should not be mixed

2024|10|24

→ Mismatch in speeds : CPU vs Memory → Cache , Memory interleaving

Cache circumvents this problem to some extent.



→ bigger cache ⟹ cost increases

$$\langle \text{access time} \rangle = h \cdot t_c + (1-h) \cdot (t_c + t_m)$$

$$= 80\% \ (0.1\,\mu s) + 20\% \ (0.1 + 1)\,\mu s$$

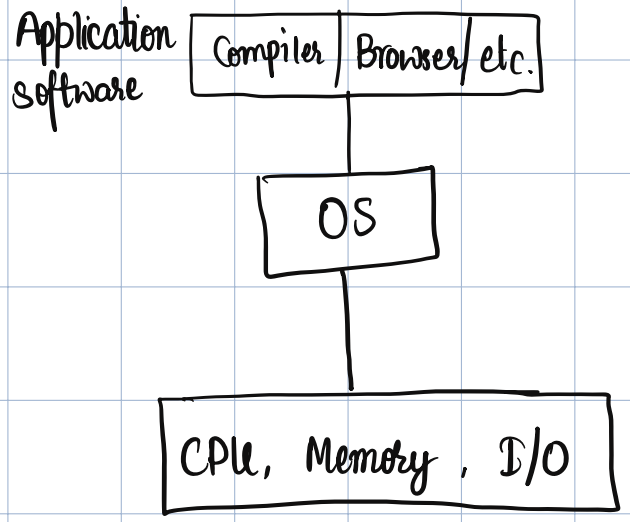
assume
h = hit rate
= 80%

→ Mismatch in speeds: CPU vs. I/O → Multiprogramming

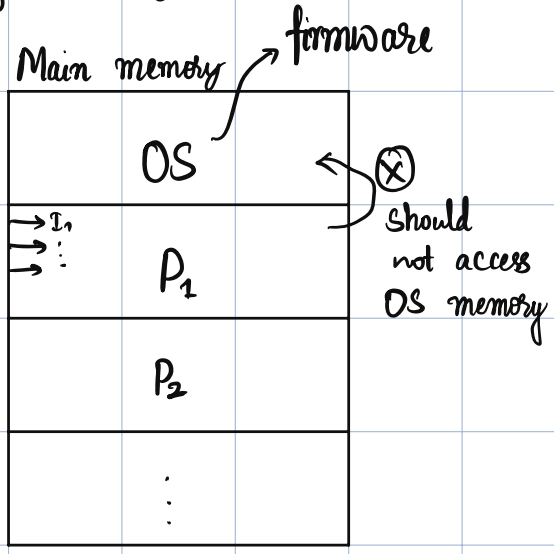
→ Problems

→ DMA (direct memory access)
→ keeps CPU & I/O busy

→ Mismatch in speeds: memory & I/O
↓
Multiport memories



??

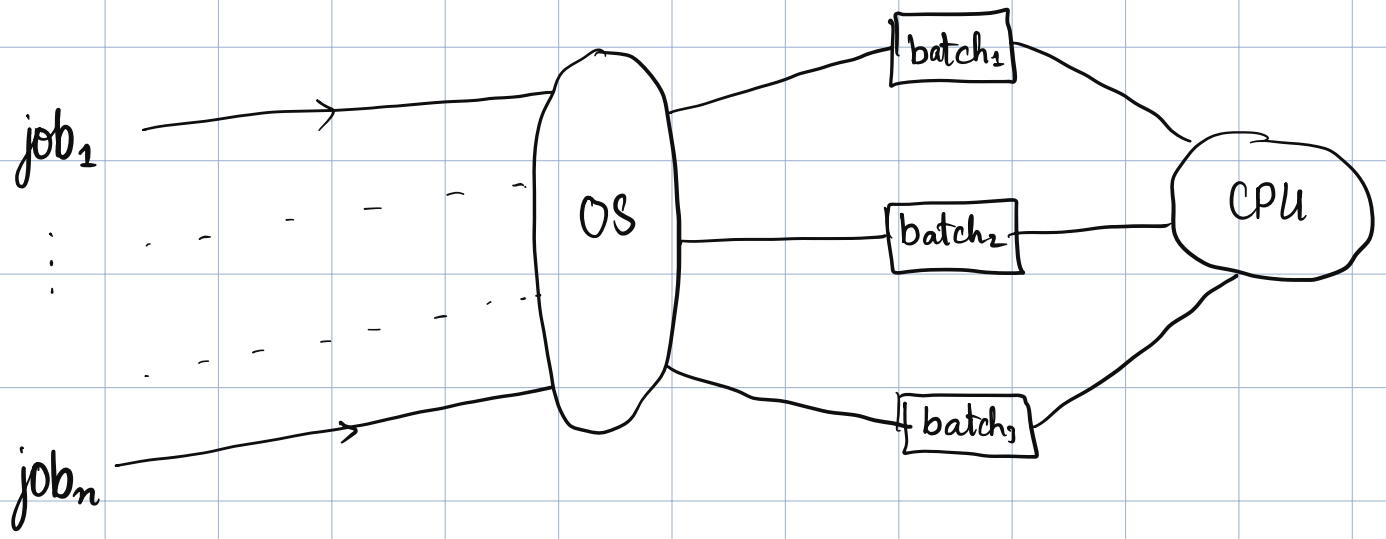


P_1: only a part of the program (remaining part in disk) → because we want to support multiprogramming

Batch O.S

→ Cache is transparent to the CPU

in CS:
what is not seen



job = program + data
(Matrix multiply) (input matrices)

job takes a lot of time ⇏ program is large
⇒ computational complexity of program ↑

→ throughput : number of jobs per hour completed by the CPU.

→ turnaround : time difference : $t_{completion} - t_{submission}$

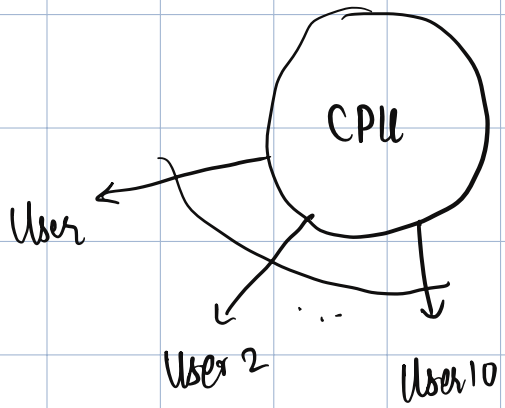
↑ throughput , ↓ turnaround

Time Sharing O.S [Interactive O.S]

→ Virtualization

Gives an illusion that there are 10 CPUs

→ minimize response time,
maximize throughput.



10^5 MIPS
Millions of instructions per second

10^{11} IPS

→ O.S requires hardware support

→ Time sharing (day) + batch (night) O.S

2024|10|25

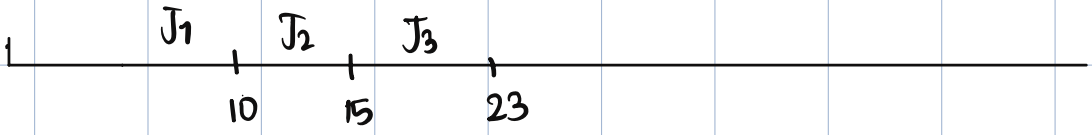
→ Batch O.S

→ Time Sharing O.S

→ Real - time O.S.

Batch OS : example

Job	Arrival time	processing time	Completion time	Turnaround time
1	0	10	10	10
2	2	5	15	13
3	4	8	23	19



→ OS schedules jobs to the CPU

→ Turnaround time ↓ ⇒ better

Time sharing OS:

→ Response time (time taken to start responding)

virtual CPU

Batch OS

turnaround time

Time sharing

Response time

Real time OS

→ programs have "deadlines"; must be completed by deadlines.

→ e.g.: satellite systems

If p_1 is done first

$t_{completion} = 35$

$t_{completion} = 55$

Process	deadline	Processing time
p_1	100	35
p_2	50	20

Batch OS — Time Sharing — Real time

→ turnaround, response time, meeting deadline

If program doesn't complete deadline

Hard RTOS

Missile launch at approaching target

Soft RTOS

Multimedia streaming

?? ??

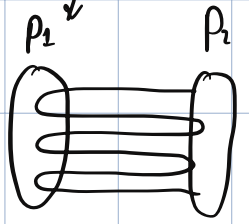
Process Program in execution

requires → ID, state, priority, program (instructions), data, stack (function calls), heap (dynamic memory), context

...

→ stored in process control block (PCB)

or process table ⇝ array of records (structs)



resume at what instruction?

→ not required in Batch OS → only context, right?

(each program executes completely. doesn't need to switch)

context to be stored and loaded:

- → store program counter (PC),
- → registers
- → list of files in use
- → accounting info
  - → no. of CPU cycles spent

→ switching is not free of cost.

context switching overhead

→ Process table → in memory

→ cannot be accessed by the user.

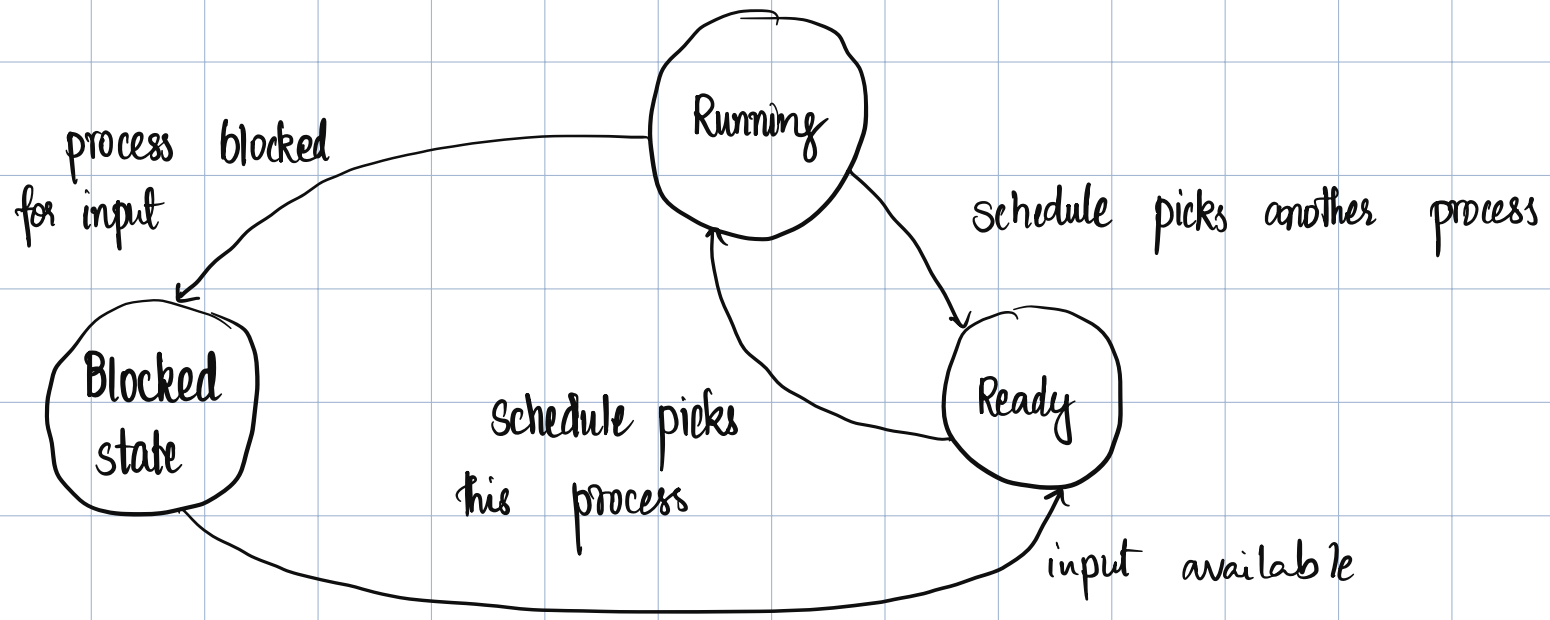
→ only by the O.S.

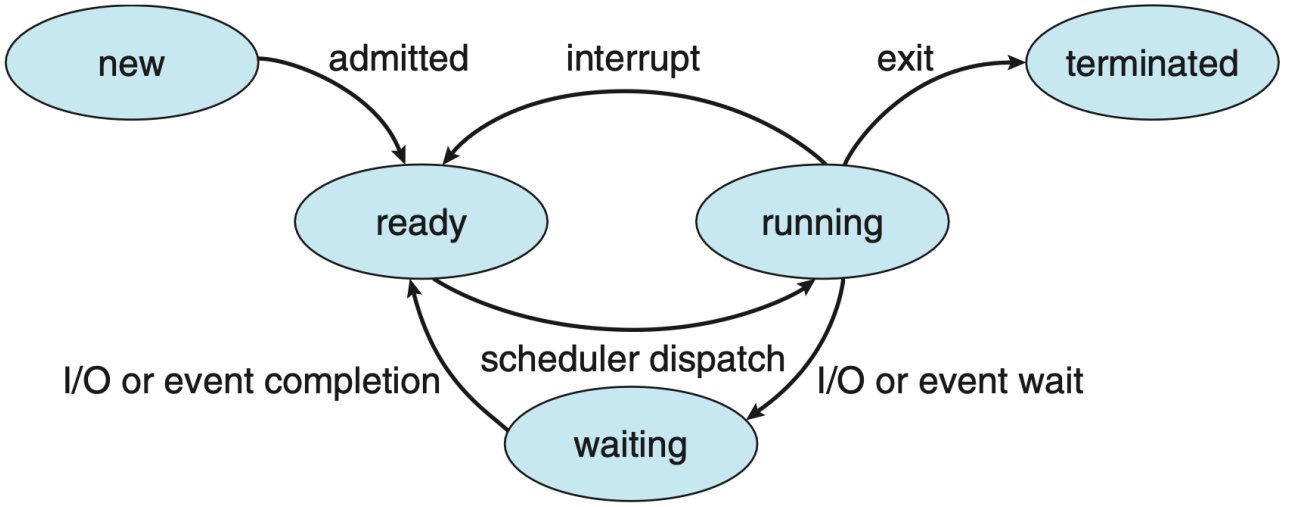
→ Who creates the processes ? OS.

UNIX	Windows
fork( )	create process( )
exit( )	exit process( )
kill( )	terminate process( )

→ only 1 state at a time.

Process state diagram




new
admitted
interrupt
exit
terminated
ready
running
scheduler dispatch
I/O or event completion
I/O or event wait
waiting