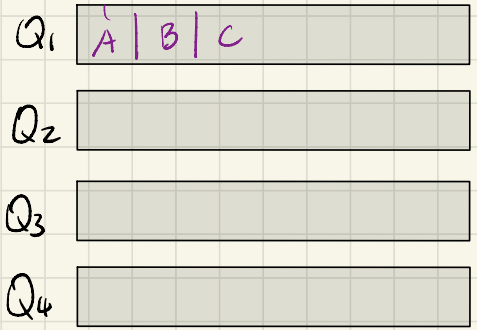
10/31 Scheduling Wrap-up & Address Translation

Scheduling

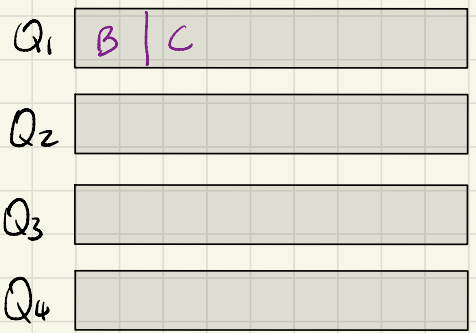
- Average response time (interactivity)
- FIFO
- SJF
- RR (a single time quantum, impacts long & short jobs differently)
- MLFQ
  - -> Multiple queues, each with its own time quantum
  - -> start all jobs at the top queue
  - -> if task blocks before the time is up, stay or move up a queue
  - -> if task uses all of its time, need more time on CPU, moves down a queue

(use the past to predict the future)

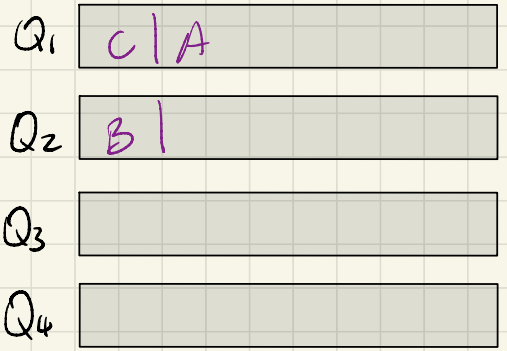
MLFQ Example



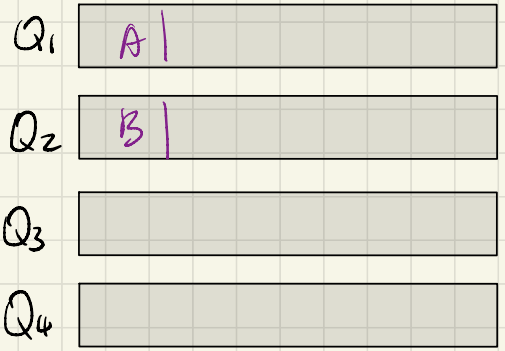
① A, B, C arrives, scheduler picks A to run



② A ran for 1 ms, then blocks scheduler picks B to run



③ While B runs, A unblocks, B runs for 5 ms



④ C gets to run, blocks after 2ms

Completely Fair Scheduler (CFS)

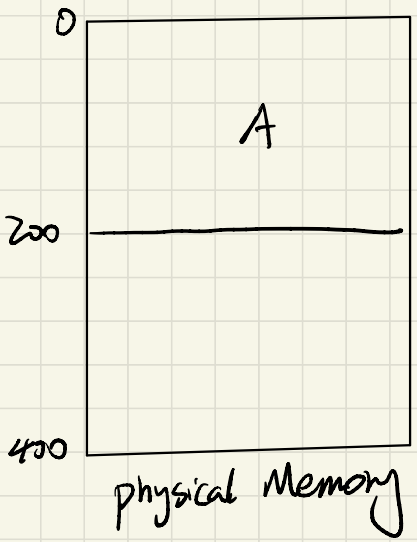
- ideally, if there are only 2 tasks, each should take 50% of CPU time
- track how long a task had been on the CPU
- red-black tree (self balanced, CPU time-ordered, binary search tree) efficiently tells you which process has the least time on the CPU.

☆ schedule one w/ least amount of CPU share (to make it fair!)

XK scheduler?

A variation of RR (based on ptable order)

Address Translation (Memory System)



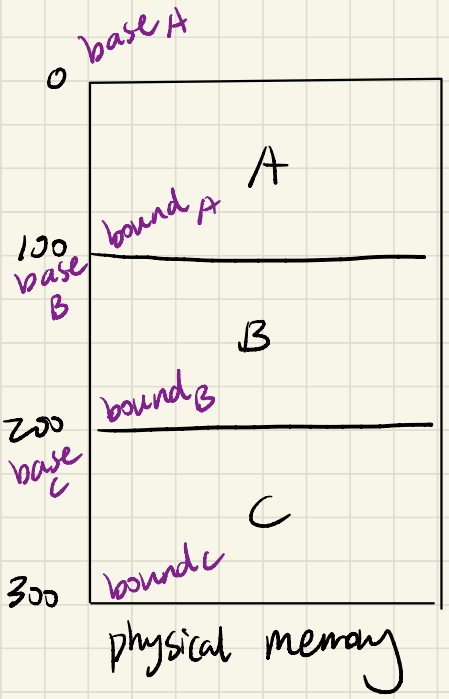
- → byte addressable
- → ~200 cycles access latency
- → process needs memory to run

Resource Allocation Problem

- → How do we allocate?
- → Simple approach: one process at a time
  - → give the entire physical memory to the process
  - → no translation needed!

Problem?

Support Multiple Processes



Let process A, B, C run in disjoint sections of physical memory

→ should processes be aware of where it is in physical memory?

→ virtual memory ("infinite" & private memory) vs. physical memory

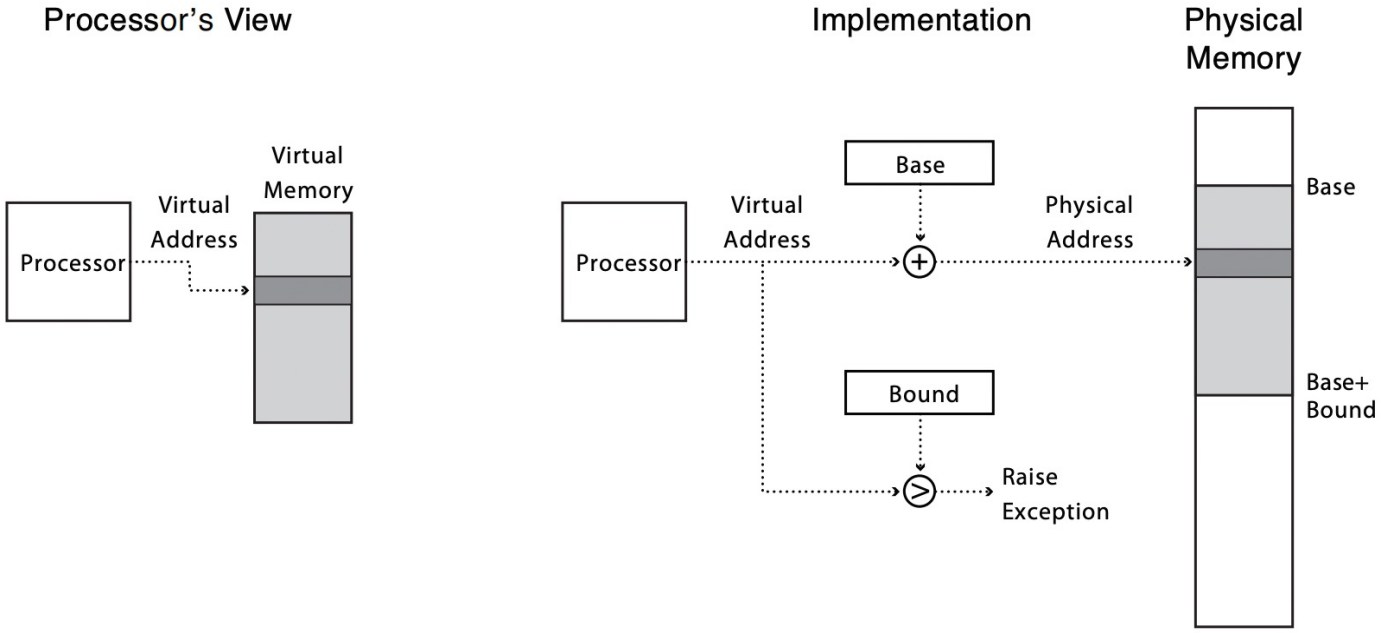
- virtual address vs. physical address

☆ How do we do address translation?

PA = VA + base (VA < bound)

base register [0] for process A
bound register [100]

base [100] bound [200] for process B



Limitations = ① processes need to fit into available space in physical memory

② hard to grow the process

③ is it possible to share memory btwn processes?

Another Approach = Paging

→ Processes don't need all of its memory at once

→ Divide process memory into fixed size chunk & only have what it needs in physical memory