10/30/23

Scheduling

(10–100 ms)

time slice

* when a job blocks another job is scheduled immediately

→ FIFO, SJF, RR

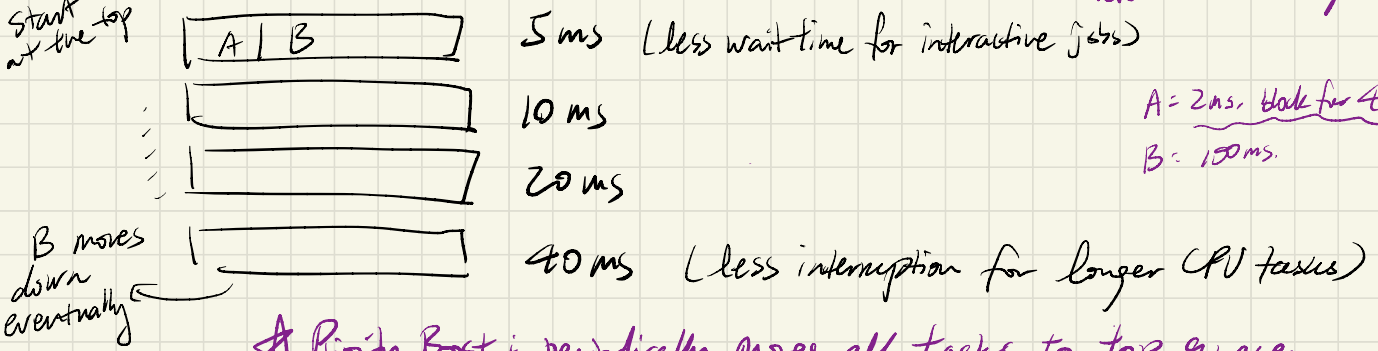
→ Problem w/ RR : one time slice can't fit all jobs

→ interactive task might block before time slice expires (ran for < quantum), but wait for up to N * time quantum before its next turn (# of jobs in ready queue)

→ MLFQ

→ one time slice doesn't work for all, so let's do multiple time slices!

↳ RR within each queue



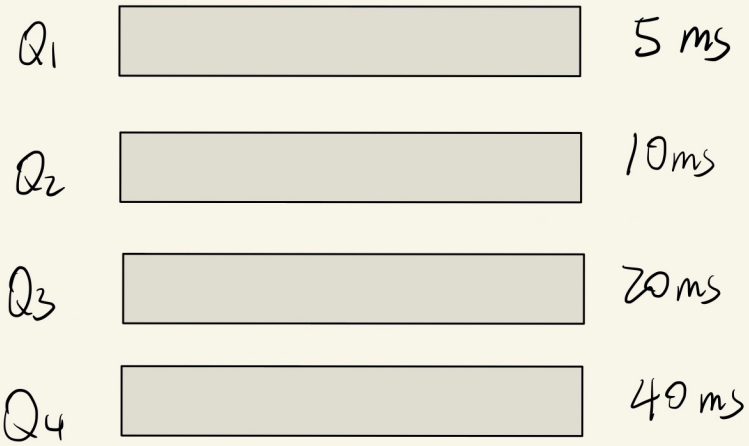
A = 2 ms. block for 4 ms.

B = 150 ms.

* Priority Boost : periodically moves all tasks to top queue

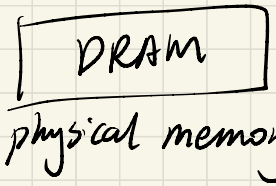
tasks arriving around the same time

A: [2ms CPU, blocks for 6ms] ×2 B: 30ms



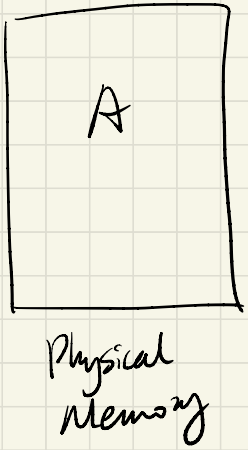
A runs for 2ms, runs
blocks (5ms), 2ms
B runs for 5 ms, blocks.
scheduled again,
1 ms, (24 ms to go)
wait for 2ms,
10 ms, finishes

Virtual Memory = Physical Memory Management



byte addressable, ~200 cycles access latency

★ Resource Allocation Problem: How should processes share the DRAM?



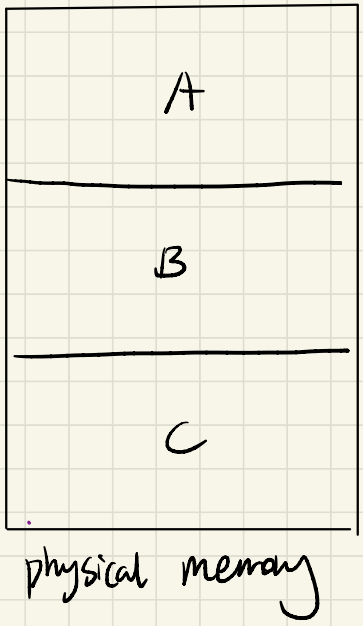
→ Simple case: don't share, just run one process at a time

→ give the entire physical memory to the process

→ no translation needed!

Any Problem? That's not how we use the computer!

Support Multiple Processes

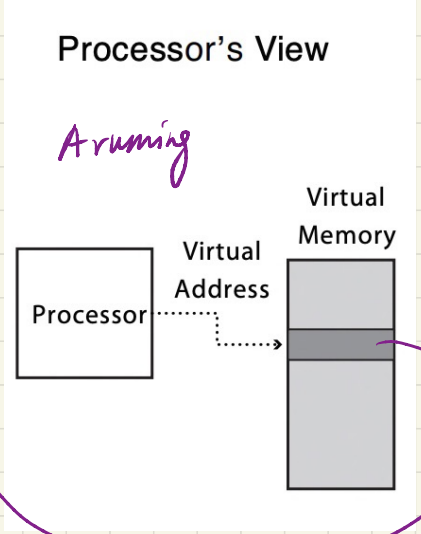


Let process A, B, C run in disjoint sections of physical memory

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

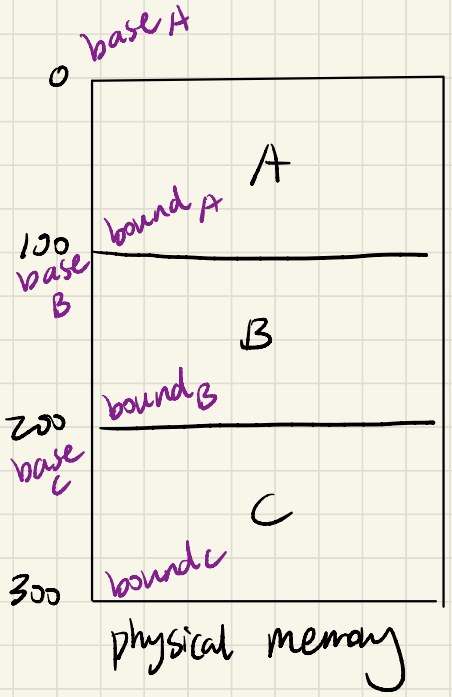
→ virtual memory ("infinite" & private memory) vs. physical memory

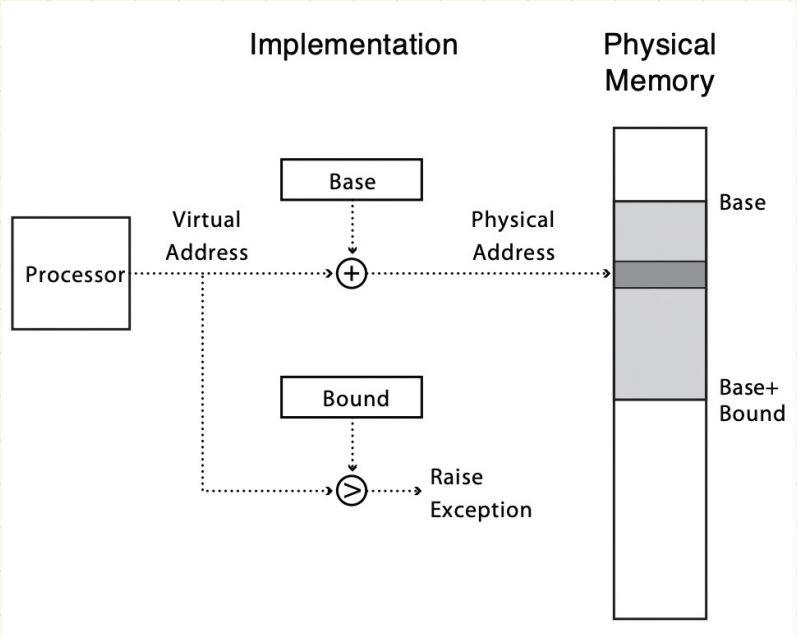
- virtual address vs. physical address



mapped to A's location in physical memory.

Address Translation?

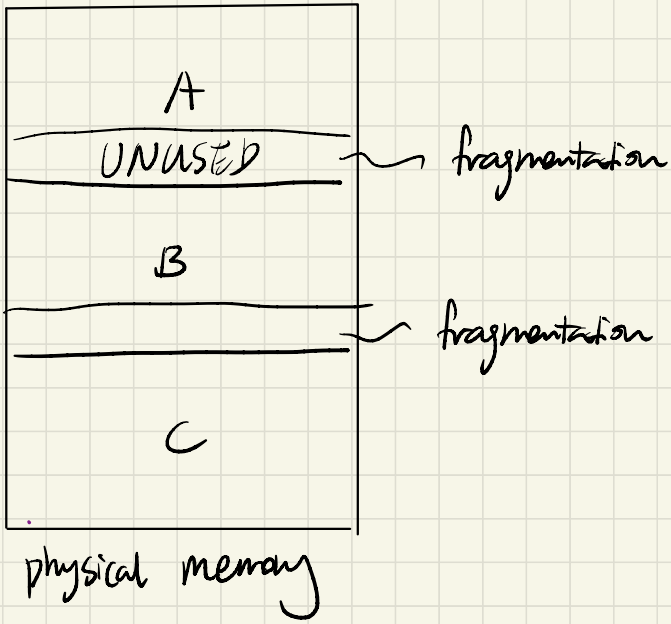




PA = VA + base (VA < bound)

base register 0 for process A
bound register 100

base 100 bound 200 for process B



Still lots of problems

→ variable sized memory allocation leads to fragmentation

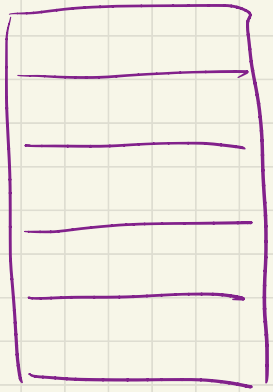
→ fragmented section might be too small to fit new process (poor utilization)

→ hard to grow

→ # of processes dependent on how large their memory requirements

Want to solve:

- → poor memory utilization
- → external fragmentation
- → # of processes in DRAM
- → flexible growth



* process don't need all of its memory at once, load as it uses each page

* divide physical memory into fixed sized chunks (pages), allocate & translate in unit on a page level