Scheduling

→ 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 \( N \times \) 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!
  \( \downarrow \) RR within each queue

Start at the top

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\( 5 \text{ ms} \) (less wait time for interactive jobs)

\( 10 \text{ ms} \)

\( 20 \text{ ms} \)

\( 40 \text{ ms} \) (less interruption for longer CPU tasks)

\( \downarrow \) Priority Boost: periodically moves all tasks to top queue
tasks arriving around the same time

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

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A runs for 2ms, runs block 5 (5ms), 2ms block
B runs for 5ms, scheduled again, 1ms, (24 ms to go)
wait for 2ms, 10 ms, finishes
Virtual Memory: Physical Memory Management

- DRAM
  - 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 processes $A$, $B$, $C$ 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

Processor's View

 mapped to $A$'s location in physical memory.
Address Translation?

Physical memory

PA = VA + base (VA < bound)

base register \(\overline{0}\) for process A

bound register \(\overline{100}\) for process A

base \(\overline{100}\)  bound \(\overline{1200}\) for process B

Implementation

Physical Memory

Processor

Virtual Address

Base

Physical Address

Bound

Raise Exception
Still lots of problems

\[ \rightarrow \text{variable sized memory allocation leads to fragmentation} \]
\[ \rightarrow \text{fragmented section might be too small to fit new process (poor utilization)} \]
\[ \rightarrow \text{hard to grow} \]
\[ \rightarrow \# \text{ of processes dependent on how large their memory requirements} \]
Want to solve:

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

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

A divide physical memory into fixed sized chunks, allocate & translate in unit on a page level