CIS 452 - Operating Systems Concepts

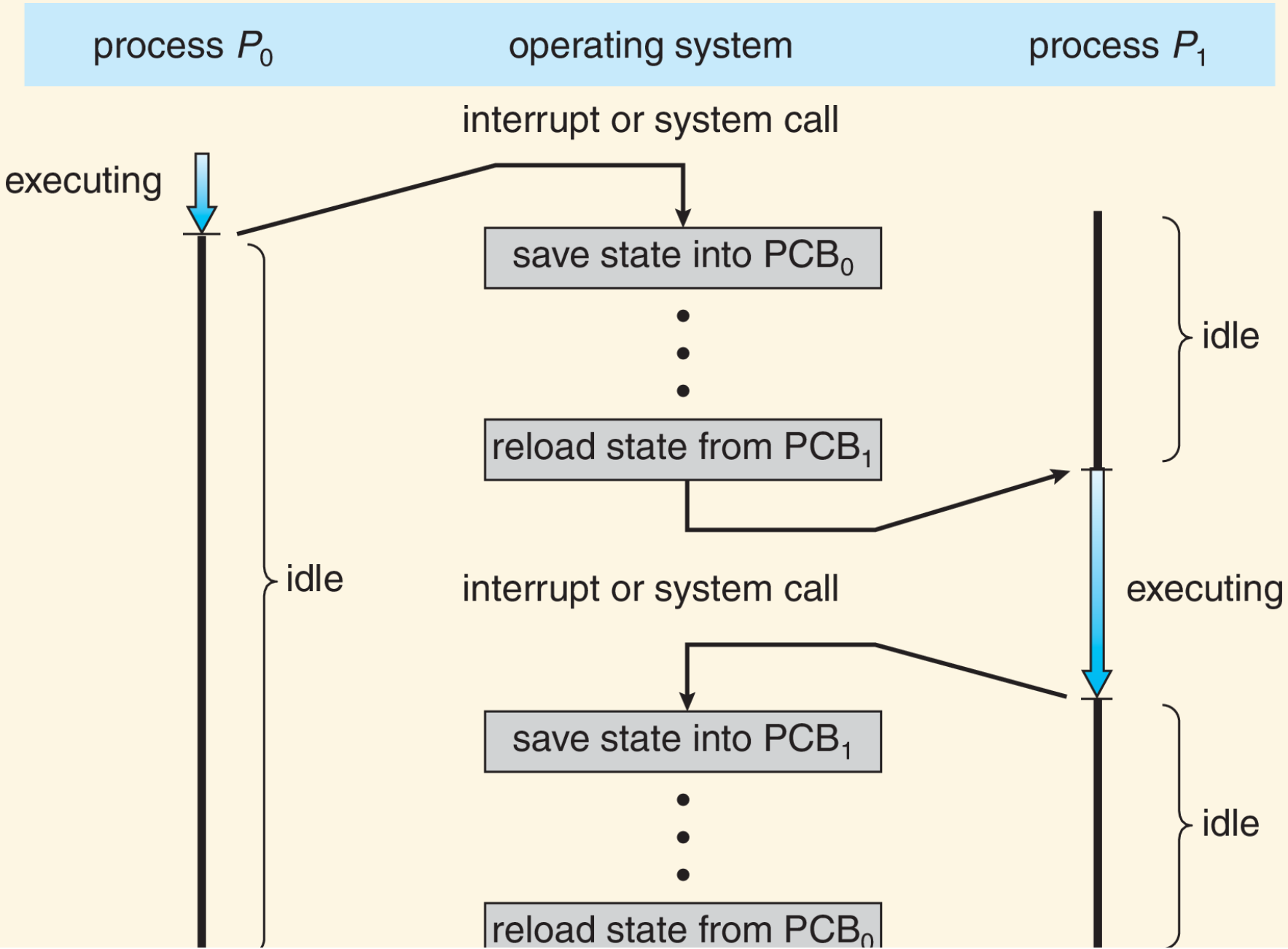
Nathan Bowman

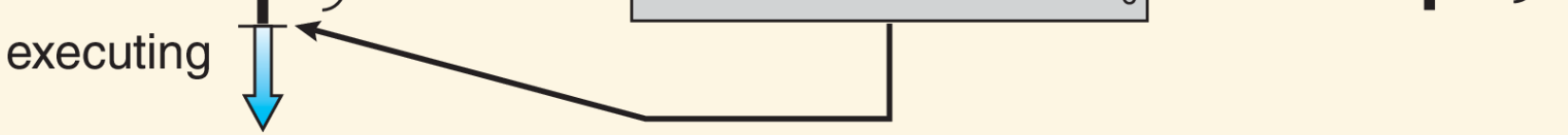
Images taken from Silberschatz book

Process Scheduling

OS regularly changes which process controls CPU

When running process changes, entire state must change with it


process P_0
operating system
process P_1
interrupt or system call
executing
save state into PCB_0
reload state from PCB_1
idle
idle
interrupt or system call
executing
save state into PCB_1
reload state from PCB_0
idle


executing

Another word for state is "context" -- this process is a **context switch**

Process's entire context (everything it needs to run correctly) is stored in PCB

Context switch is when OS saves state of running process into PCB, restores state of next process from its PCB, and puts that process on the CPU

Context switch metaphor is quite useful for real life

Think about moving from reading your textbook to responding to an email, for example

Time spent in a context switch is completely wasted

Duration of context switch depends on several factors:

- complexity of stored state
- hardware speed
- special instructions from architecture

Few milliseconds is a good approximation

OS be careful how it decides which processes to schedule

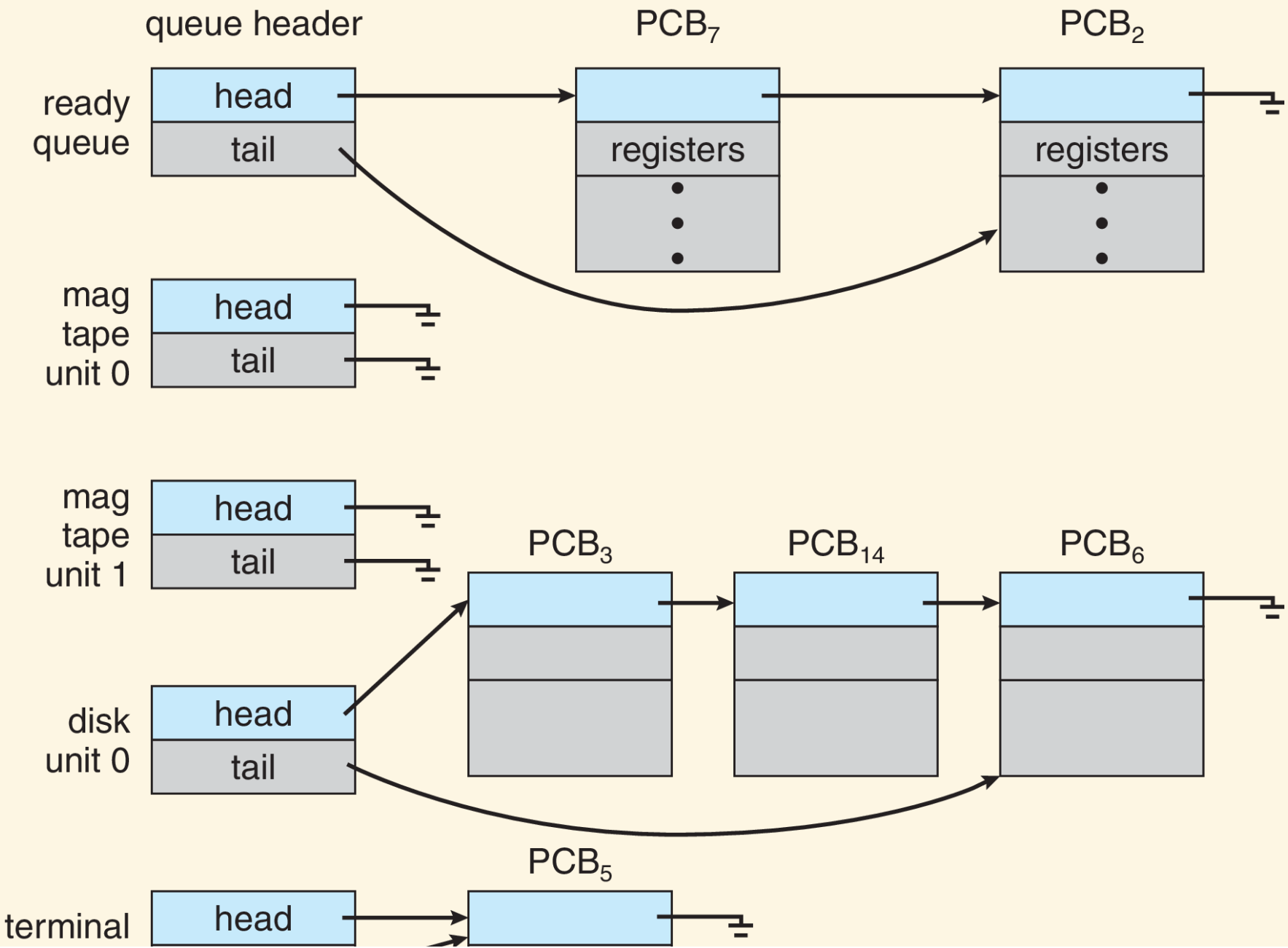
Must consider resource utilization (via multiprogramming) and time sharing (how reactive is computer?)

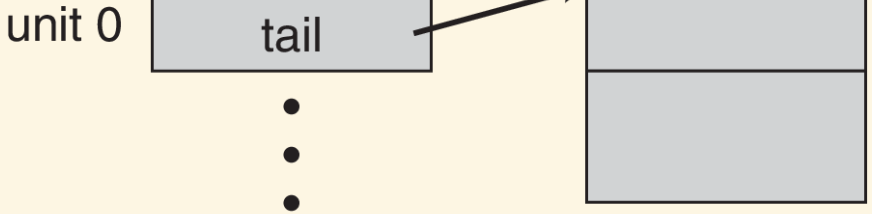
Correct balance depends on use case

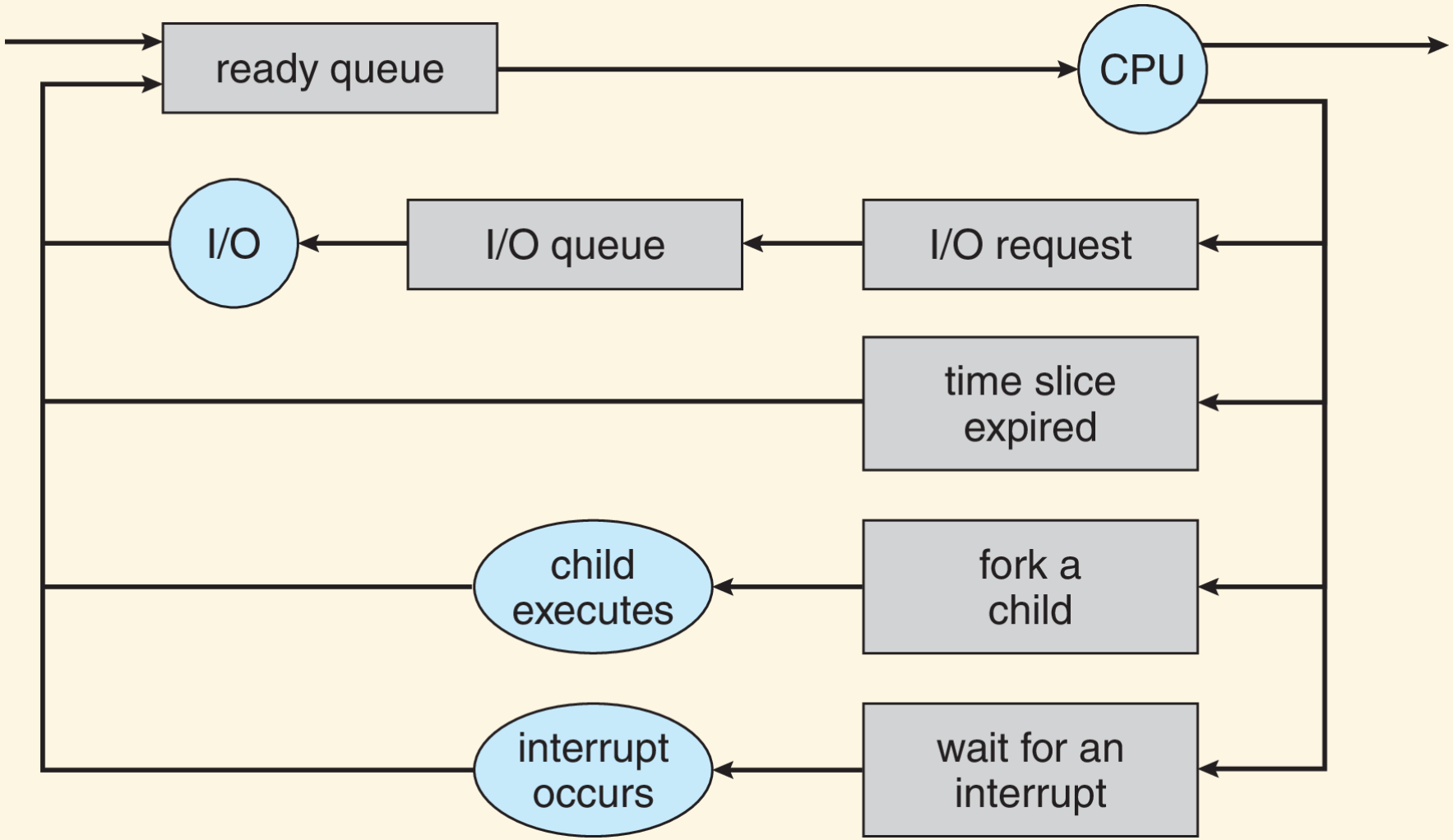
Process scheduler -- deciding who gets the CPU

Relies on several queues to keep track of process state

- job queue -- process waiting to get into main memory
- ready queue -- processes in main memory ready and waiting for CPU
- device queues -- processes waiting for I/O on a particular device (one queue per device)


queue header
PCB7
PCB2
ready queue
head
tail
registers
registers
mag tape unit 0
head
tail
mag tape unit 1
head
tail
PCB3
PCB14
PCB6
disk unit 0
head
tail
PCB5
terminal
head


unit 0
tail


ready queue
CPU
I/O
I/O queue
I/O request
time slice expired
child executes
fork a child
interrupt occurs
wait for an interrupt

Levels of schedulers

- CPU (short-term) scheduler
- medium-term scheduler
- job (long-term) scheduler

CPU scheduler determines which process to run next

Scheduler is running every few milliseconds to keep system responsive

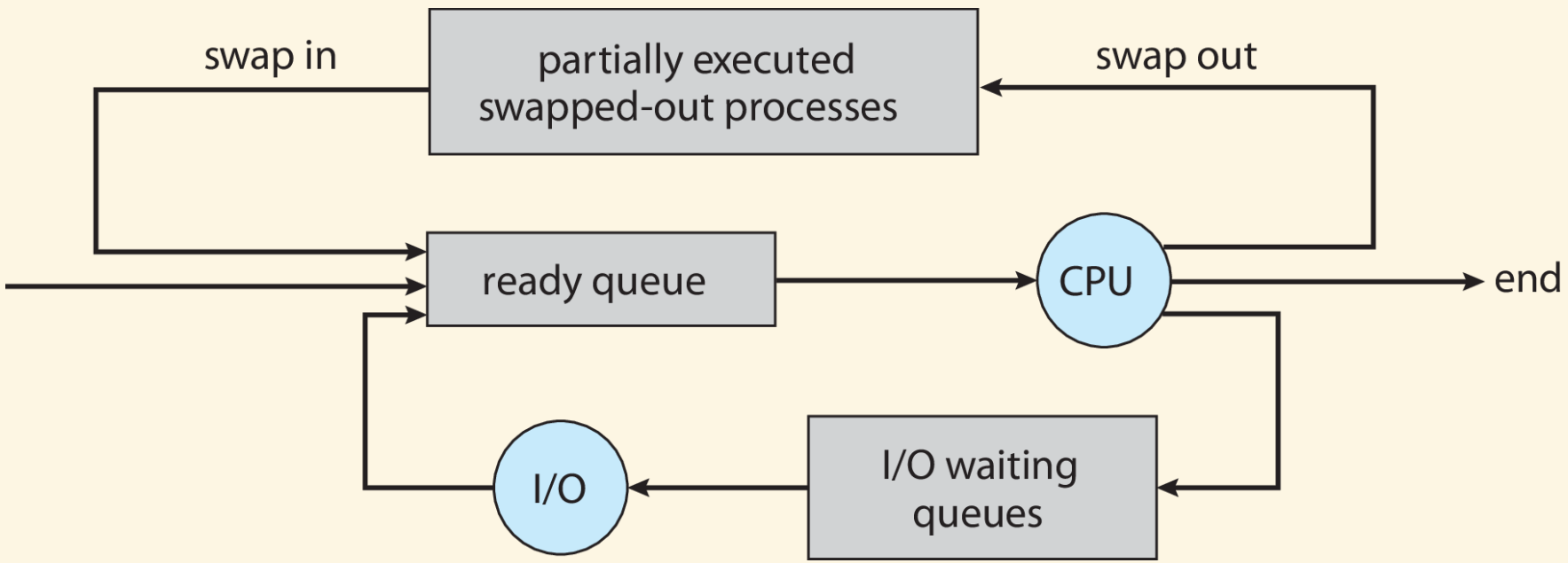
Must be **fast** (remember: time spent context switching is time wasted)

Medium-term scheduler is responsible for temporarily removing processes from memory as needed

Moving programs into and out of memory is **swapping**

Sometimes done to achieve better balance of CPU-bound and I/O-bound processes

Also done if memory is overtaxed (this is when you will notice your computer getting *very* slow)


swap in
partially executed
swapped-out processes
swap out
ready queue
CPU
end
I/O
I/O waiting
queues

Swapping is another nice metaphor for life

Memory is computer's place to do useful work

Running out of memory is like trying to cook in a tiny kitchen or work on an overcrowded desk

Job scheduler determines which processes get into memory

This applies more to batch systems, whereas medium-term scheduling applies more to time-sharing systems

Usually jobs in == jobs out, so this scheduler runs only every few minutes and can afford to be much slower

Tries to strike good balance between I/O-bound and CPU-bound processes