Georgia Tech Computer Science

CS2200
Systems and Networks
Spring 2022

Lecture 13: Processor Scheduling

Alexandros (Alex) Daglis
School of Computer Science
Georgia Institute of Technology
adaglis@gatech.edu

Lecture slides adapted from Bill Leahy and Charles Lively of Georgia Tech

Announcements

- Starting Chapter 6

- Assignment deadlines extended – check Canvas announcements

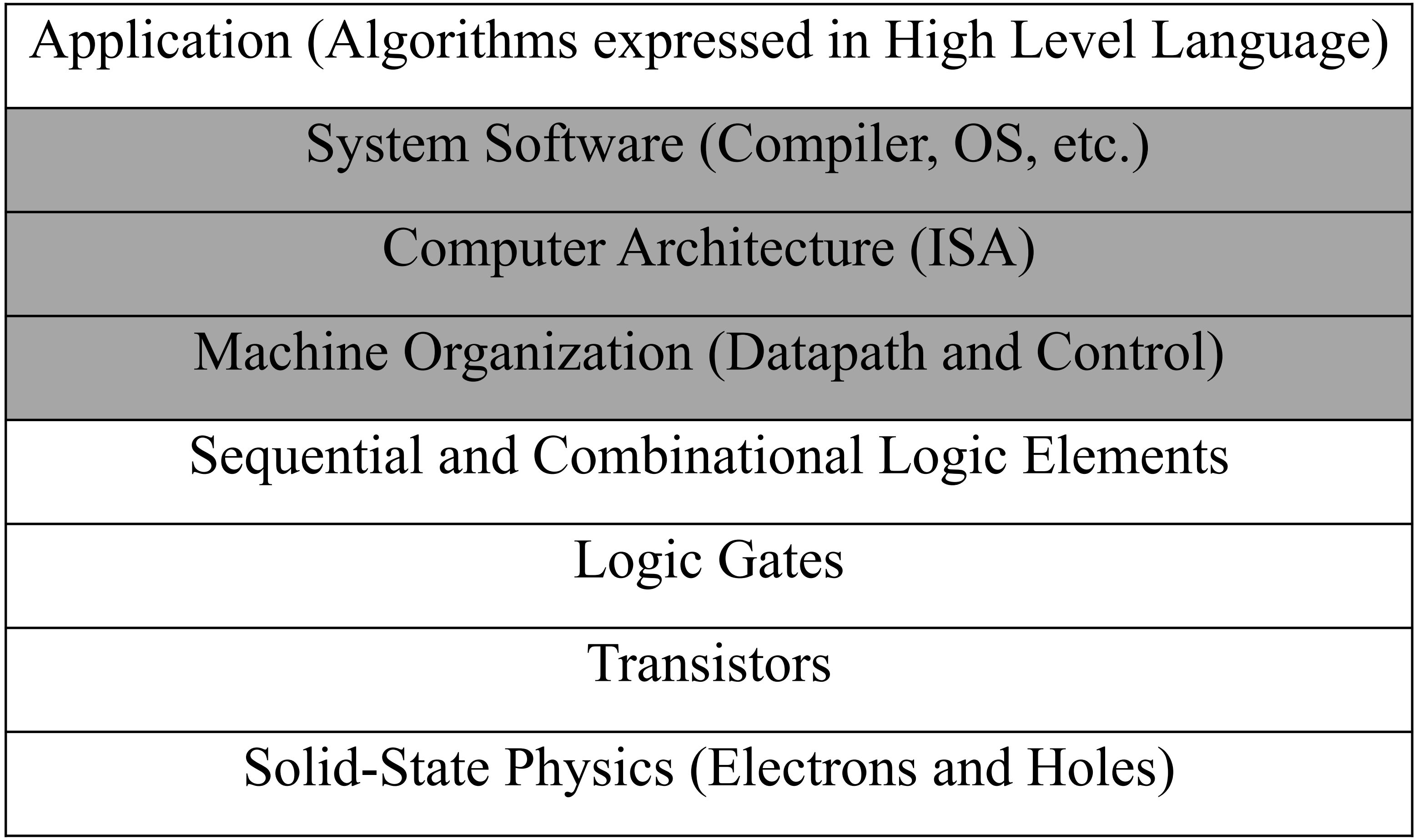
Topics

- Process as an abstraction
- Scheduler
- Process, job, thread, task
- States of process
- Process control block (PCB)
- Types of scheduler
- Algorithms
- Metrics & Evaluation

What is an operating system?

- An operating system is a special program that manages access of user programs to hardware resources
- The OS provides a number of abstractions, e.g.
  - Multiple processes (how can one physical processor run multiple programs at the same time?)
  - Memory permissions to protect other processes and the OS
  - Shared access to I/O devices, e.g. networks, file systems
  - Resource sharing (processors, memory, I/O, …)
  - Additional instructions (read, write, exit, get more memory, change permissions, …) implemented through traps

Levels of Abstraction



Application (Algorithms expressed in High Level Language)

System Software (Compiler, OS, etc.)

Computer Architecture (ISA)

Machine Organization (Datapath and Control)

Sequential and Combinational Logic Elements

Logic Gates

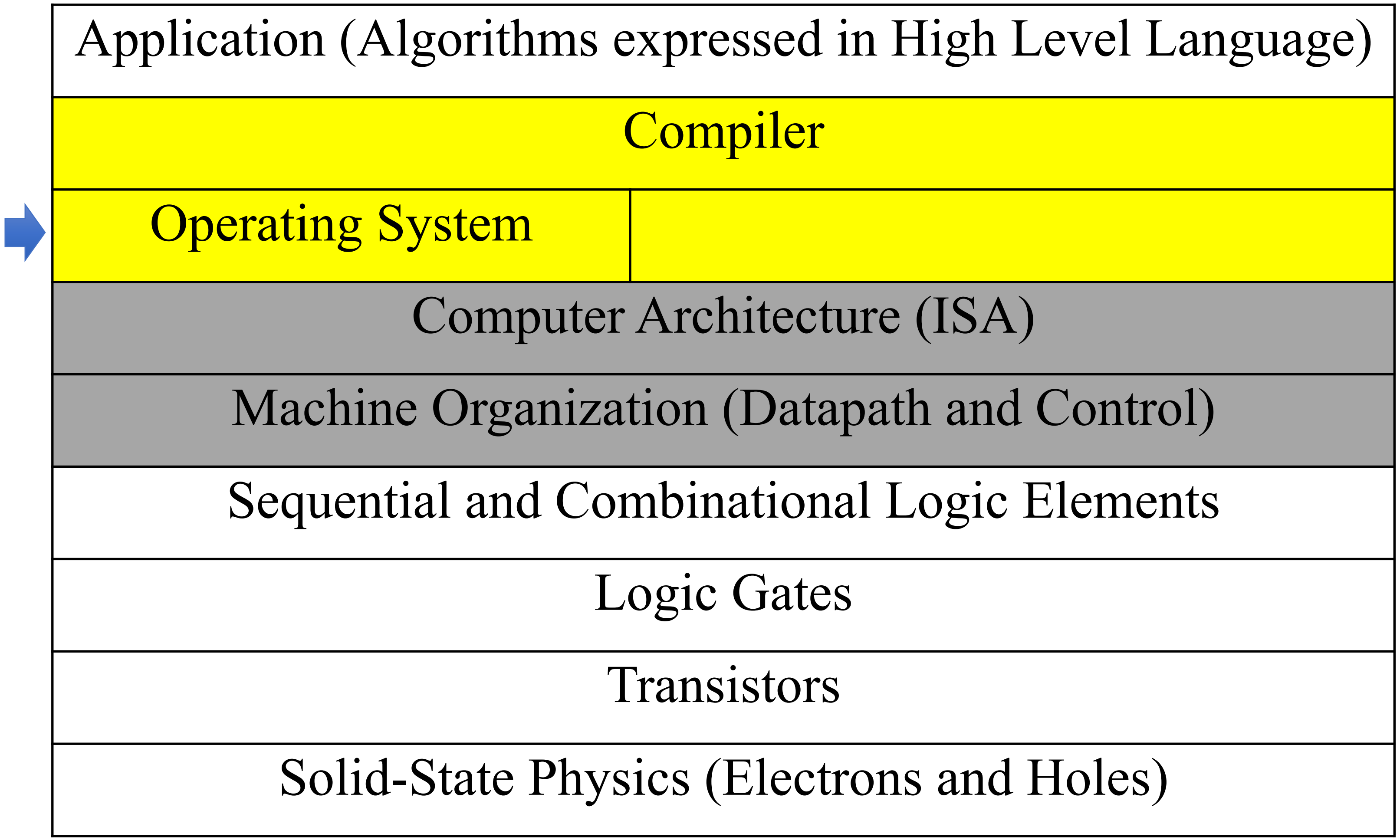
Transistors

Solid-State Physics (Electrons and Holes)

Levels of Abstraction

Application (Algorithms expressed in High Level Language)
System software (Compiler, OS, etc.)
Computer Architecture (ISA)
Machine Organization (Datapath and Control)
Sequential and Combinational Logic Elements
Logic Gates
Transistors
Solid-State Physics (Electrons and Holes)

Levels of Abstraction



Application (Algorithms expressed in High Level Language)

Compiler

Operating System

Computer Architecture (ISA)

Machine Organization (Datapath and Control)

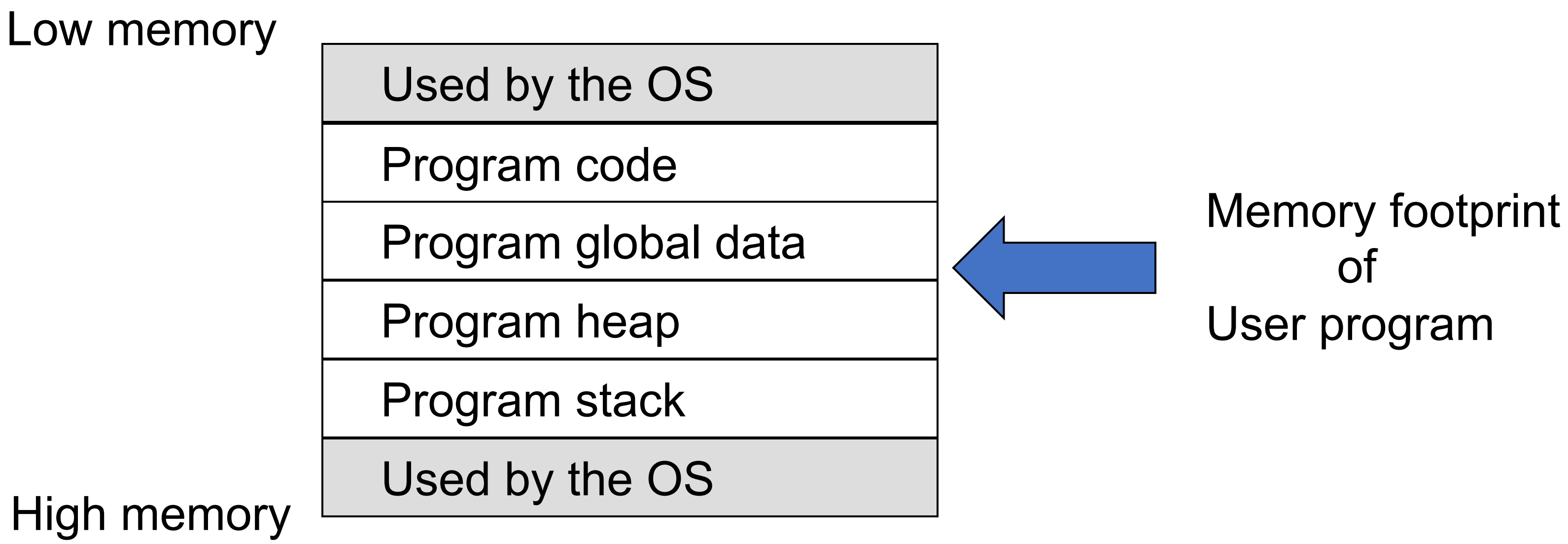
Sequential and Combinational Logic Elements

Logic Gates

Transistors

Solid-State Physics (Electrons and Holes)

A program in memory



Low memory

Used by the OS

Program code

Program global data

Program heap

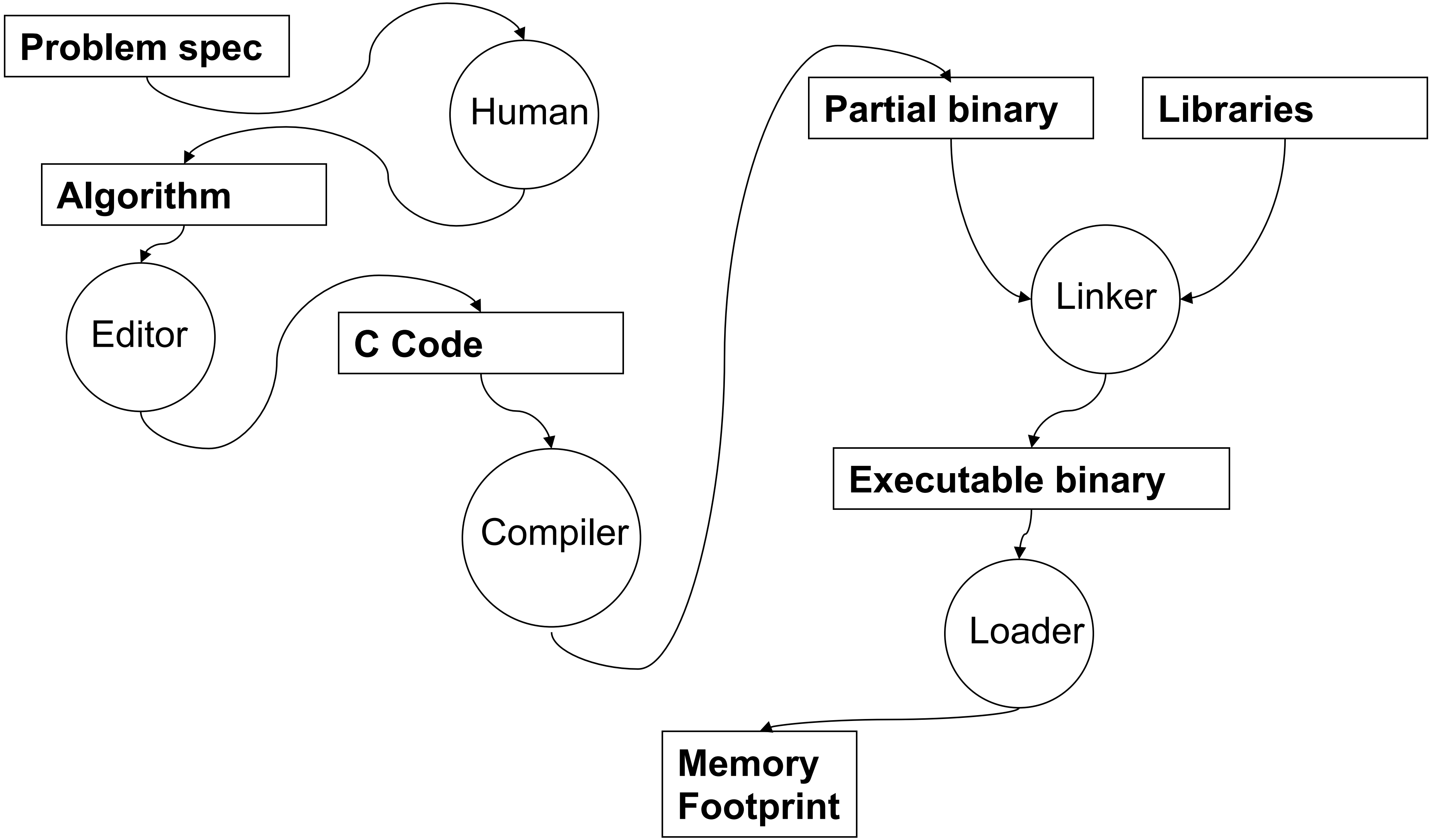
Program stack

Used by the OS

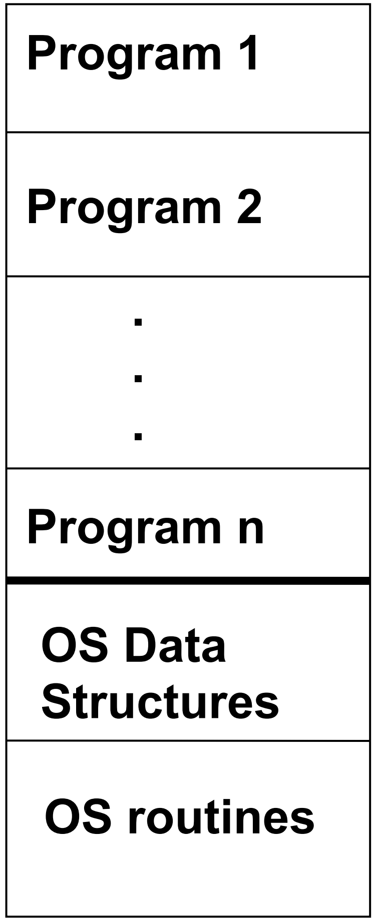
High memory

Memory footprint
of
User program

How do we create a program?



More Than 1 Program in Memory?



Program 1

Program 2

.
.
.

Program n

OS Data Structures

OS routines

- What's the difference between a process and a program?
- A process is a program AND all of the state that represents its execution, e.g.
  - Registers
  - Memory
  - PC
  - Stack
- So far we've been talking about computers that have one single process with the state maintained in hardware
- Interrupts were our first clue that it doesn't have to be this way

Session ID: dag2200

Process differs from program in that

0% A. There is no difference

11% B. A process is a machine language representation of a program

68% C. A process is a program in execution

0% D. A process is a program that executes correctly

21% E. A process is a special kind of program that is part of the operating system

0% F. No clue

Concepts

- Multitasking
    - Shortest Job First
    - Priority
    - Round Robin
    - Preemption
    - First Come First Served

Concepts

- Multitasking
- Shortest Job First
- Priority
- Preemption
- Round Robin
- First Come First Served

In the context of your life

Your To Do List:

- Laundry
- Prepare food and eat
- Call Mom
- Prepare for tests

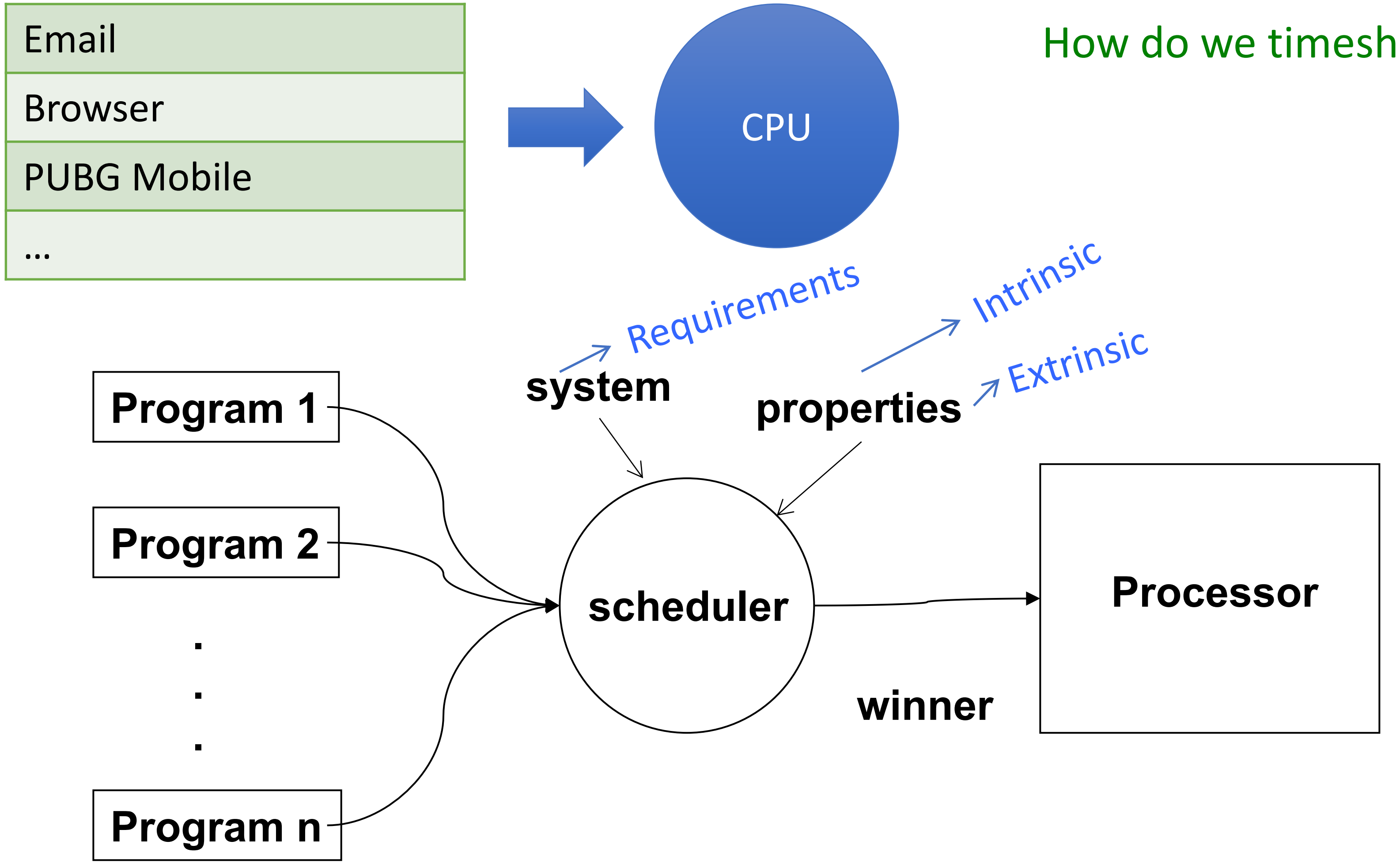
You:

How do you timeshare yourself?!

Laundry ----------------------
Prep & Eat ----------
Call Mom --------
Test Prep ----------

Multitasking in the computer world

How do we timeshare a processor?



Email
Browser
PUBG Mobile
…

Session ID: dag2200

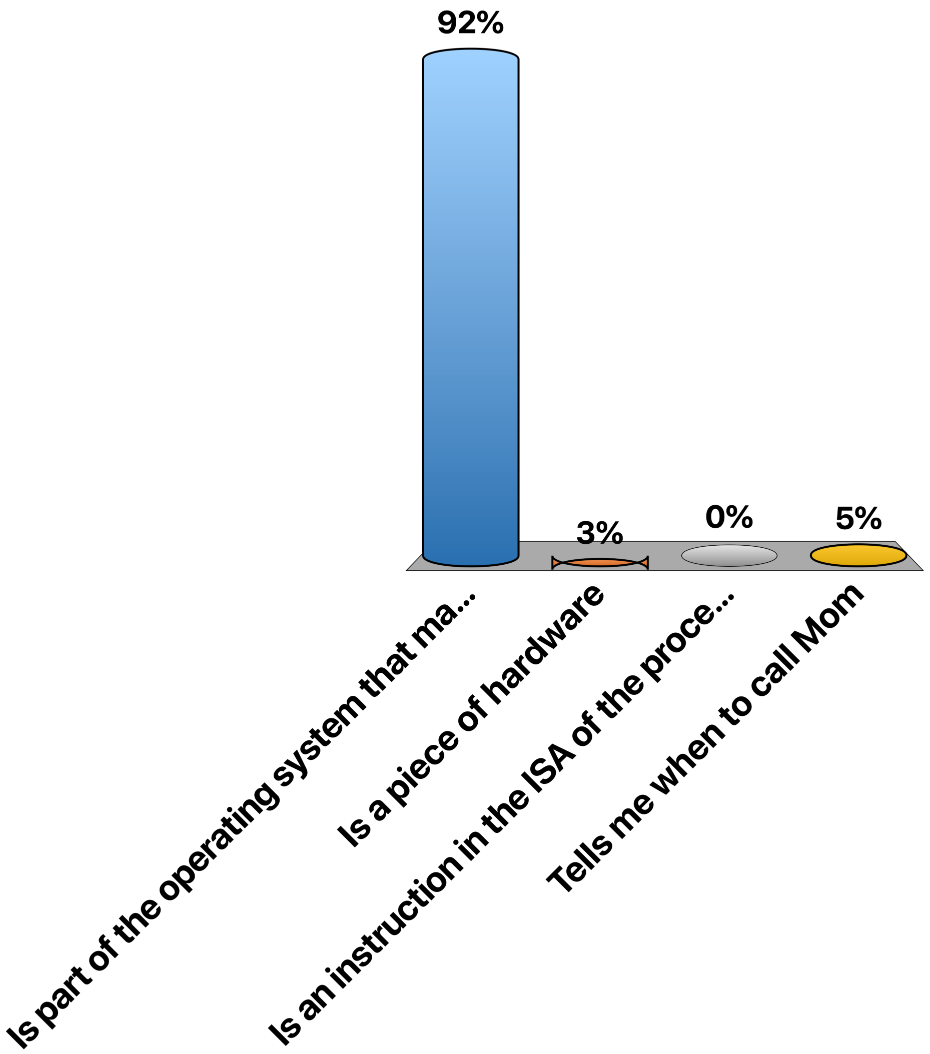
A scheduler

A. Is part of the operating system that makes decisions to allocate the processor resources to processes

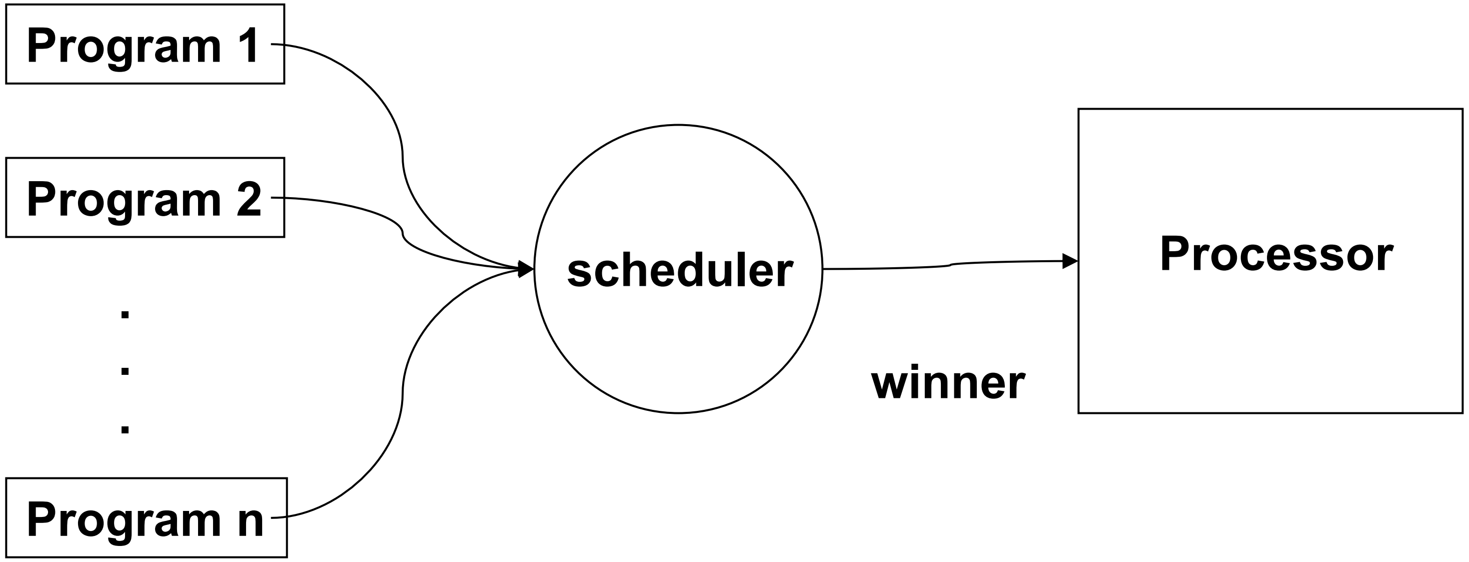
B. Is a piece of hardware

C. Is an instruction in the ISA of the processor

D. Tells me when to call Mom



Quick aside: Threads



Program 1

Program 2

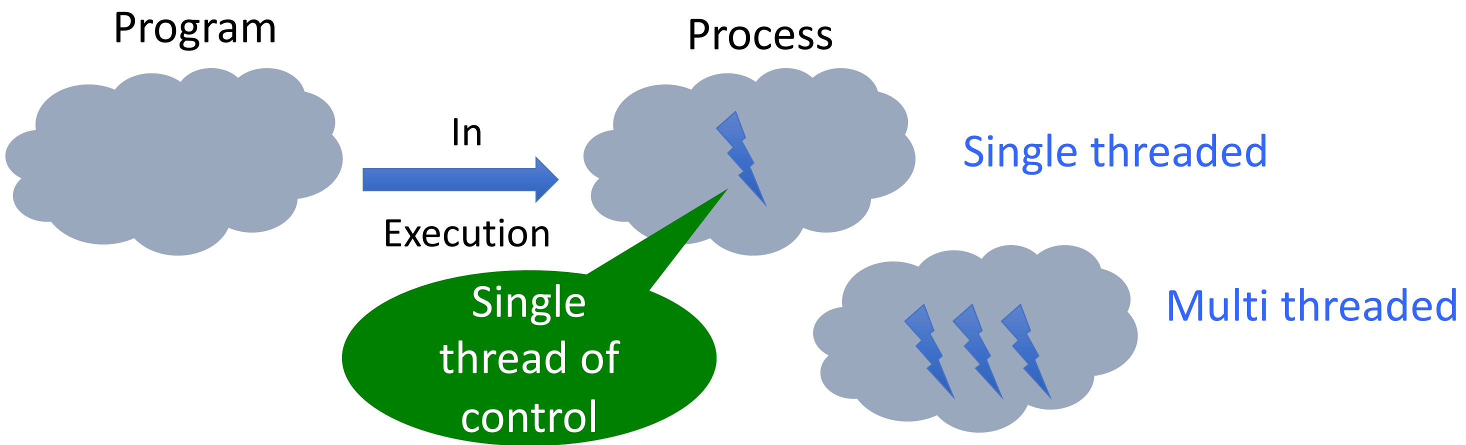
.
.
.

Program n

scheduler

winner

Processor



Program

In

Execution

Process

Single threaded

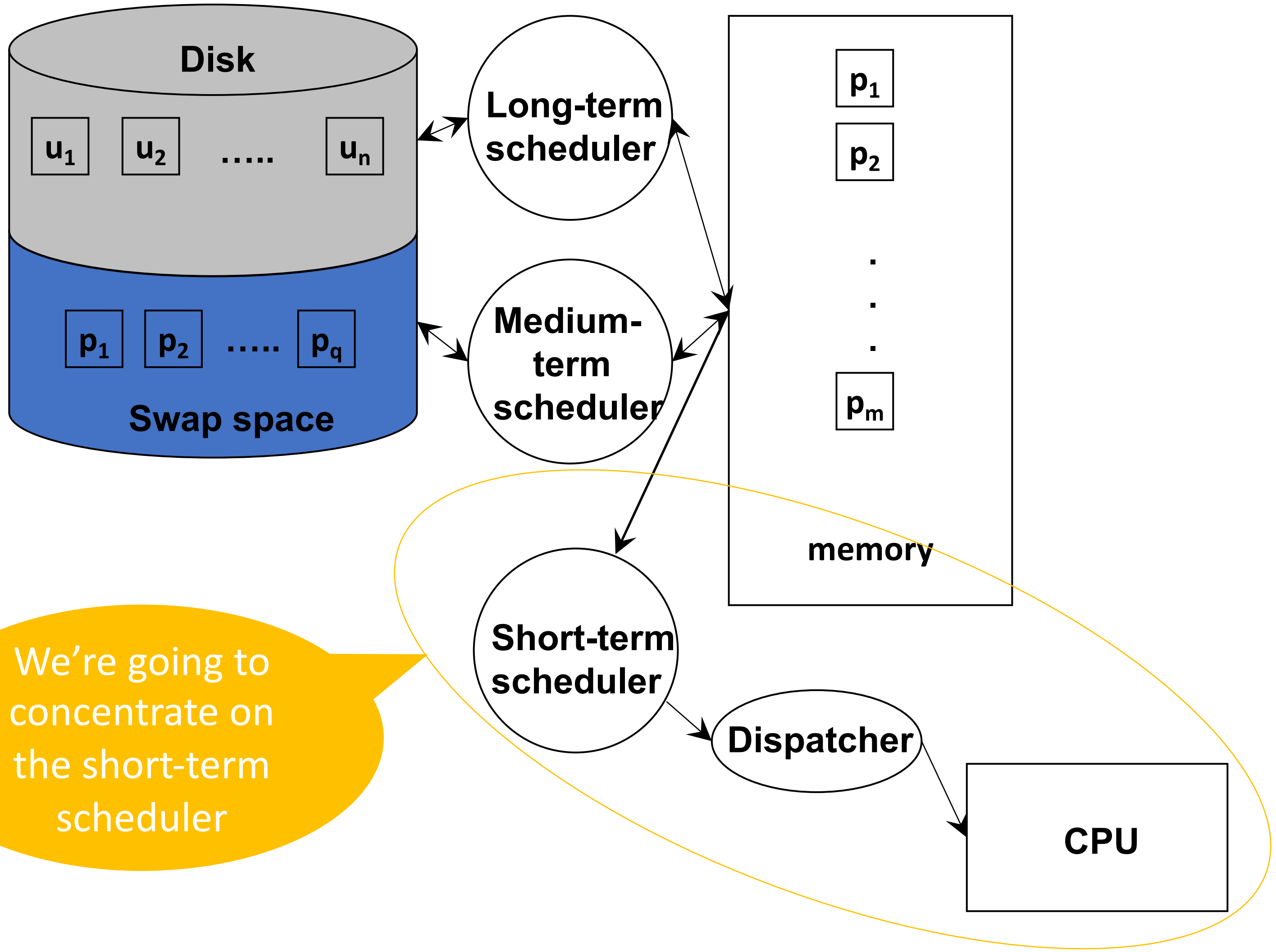
Single thread of control

Multi threaded

Terminology

Name	Usual Connotation	Use in this chapter
Job	Unit of scheduling	Synonymous with process
Process	Program in execution; unit of scheduling	Synonymous with job
Thread	Unit of scheduling and/or execution; contained within a process	Not used in the scheduling algorithms described in this chapter
Task	Unit of work; unit of scheduling	Not used in the scheduling algorithms described in this chapter, except in describing the scheduling algorithm of Linux

Schedulers



We're going to concentrate on the short-term scheduler

Terminology

Name	Environment	Role
Loader	In every OS	Load user program from disk into memory
Long term scheduler	Batch oriented OS	Control the job mix in memory to balance use of system resources (CPU, memory, I/O)
Medium term scheduler	Every modern OS (time-shared, interactive)	Balance the mix of processes in memory to avoid thrashing
Short term scheduler	Every modern OS (time-shared, interactive)	Schedule the memory resident processes on the CPU
Dispatcher	In every OS	Populate the CPU registers with the state of the process selected for running by the short-term scheduler

Session ID: dag2200

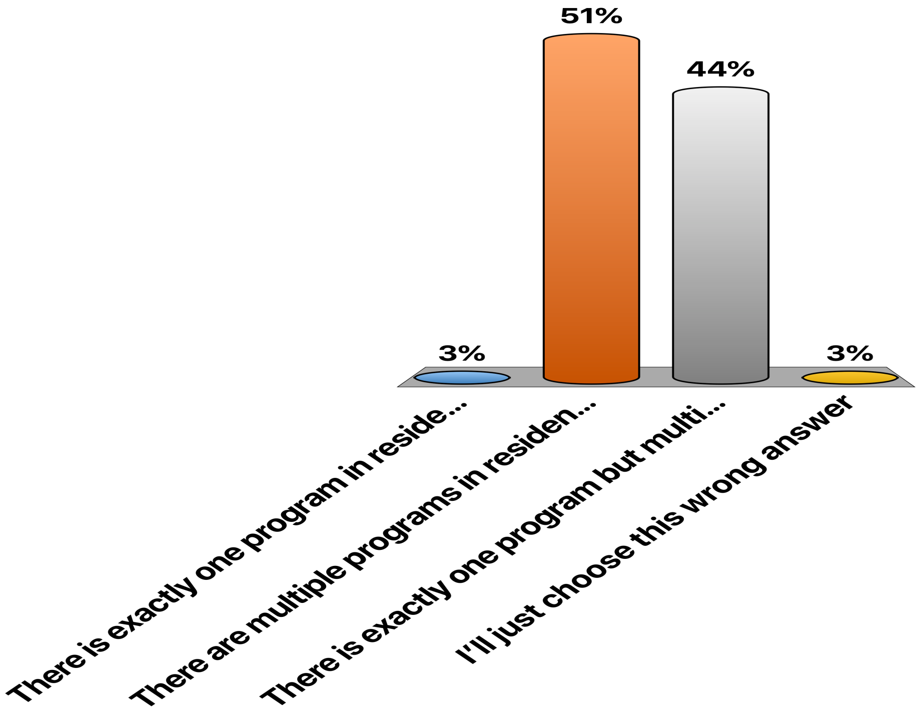
In most modern-day operating systems

A. There is exactly one program in residence in memory at any point in time

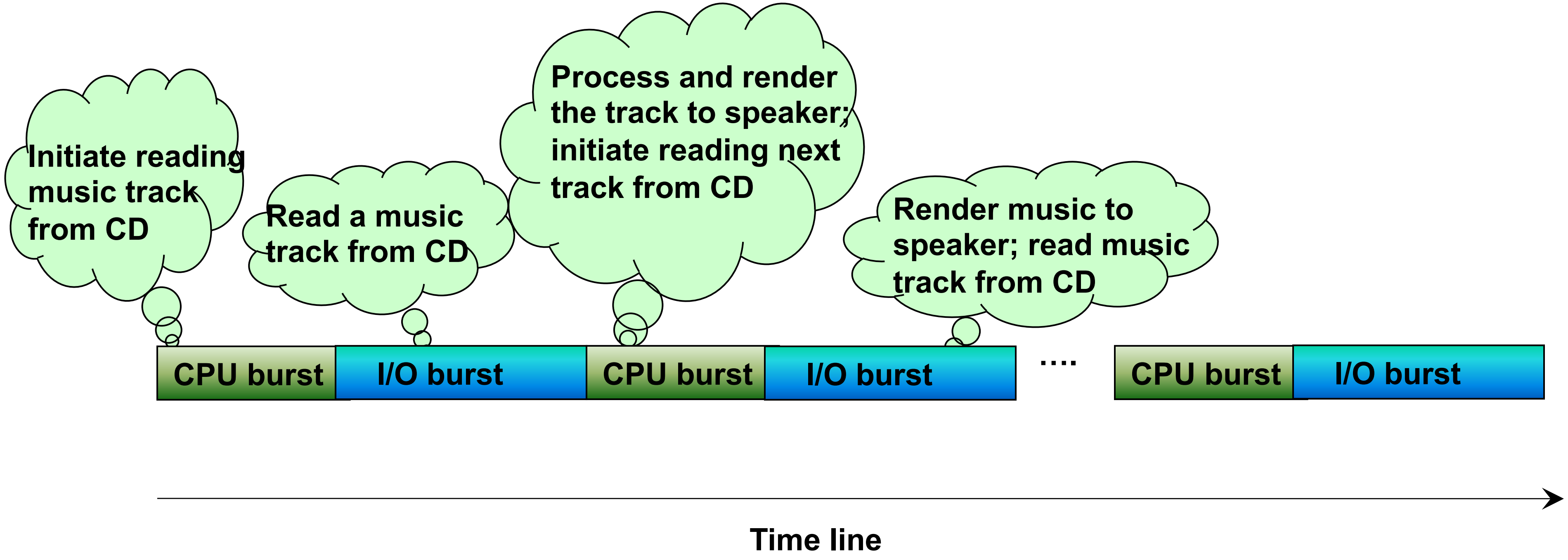
B. There are multiple programs in residence in memory at any point in time

C. There is exactly one program but multiple processes in residence in memory at any point in time

D. I'll just choose this wrong answer



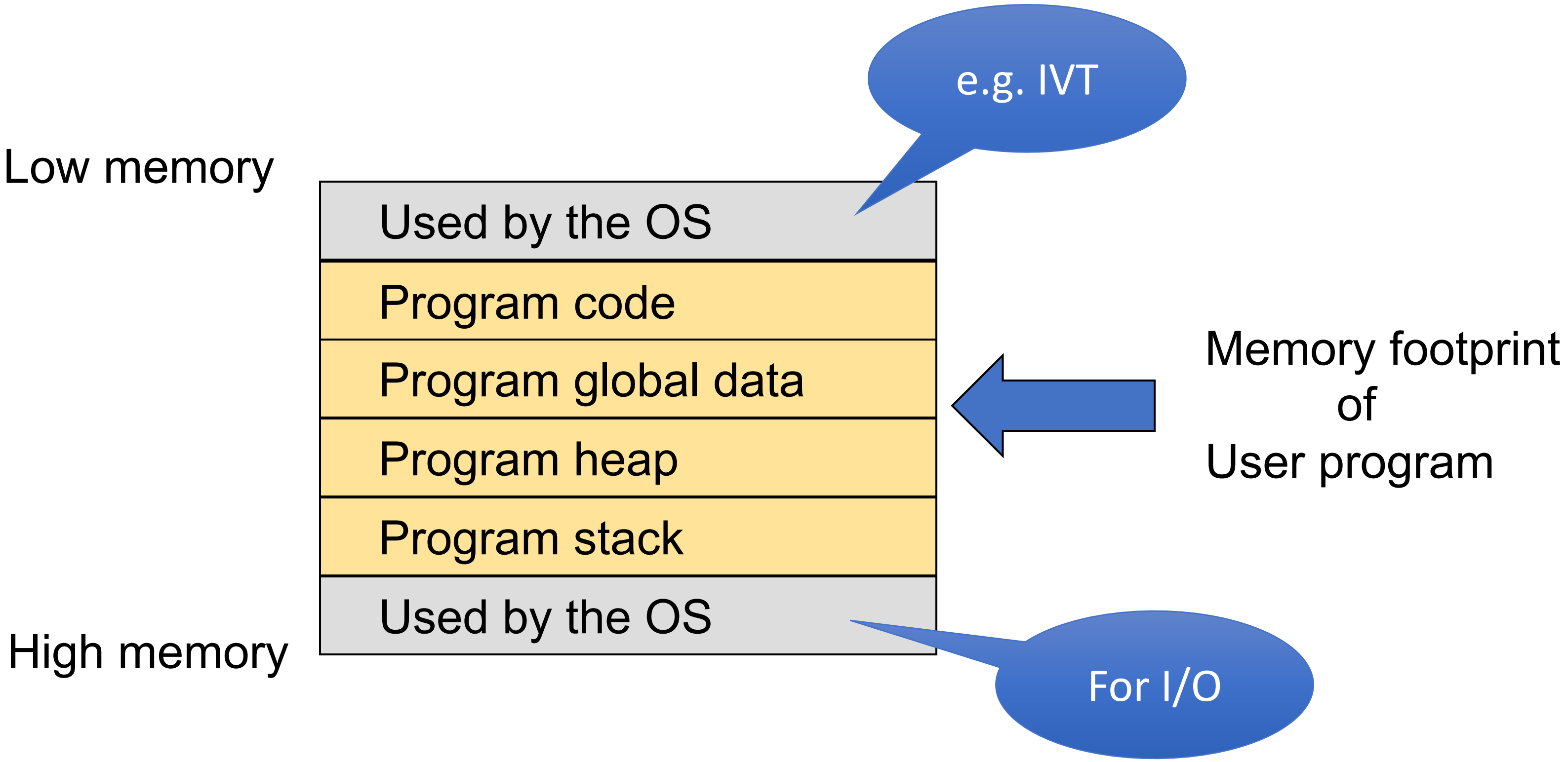
What's the process doing?


Initiate reading music track from CD
Read a music track from CD
Process and render the track to speaker; initiate reading next track from CD
Render music to speaker; read music track from CD
CPU burst
I/O burst
CPU burst
I/O burst
….
CPU burst
I/O burst
Time line

Program properties

CPU to I/O ratio

Recall the memory footprint of a program



Low memory

Used by the OS

Program code

Program global data

Program heap

Program stack

Used by the OS

High memory

e.g. IVT

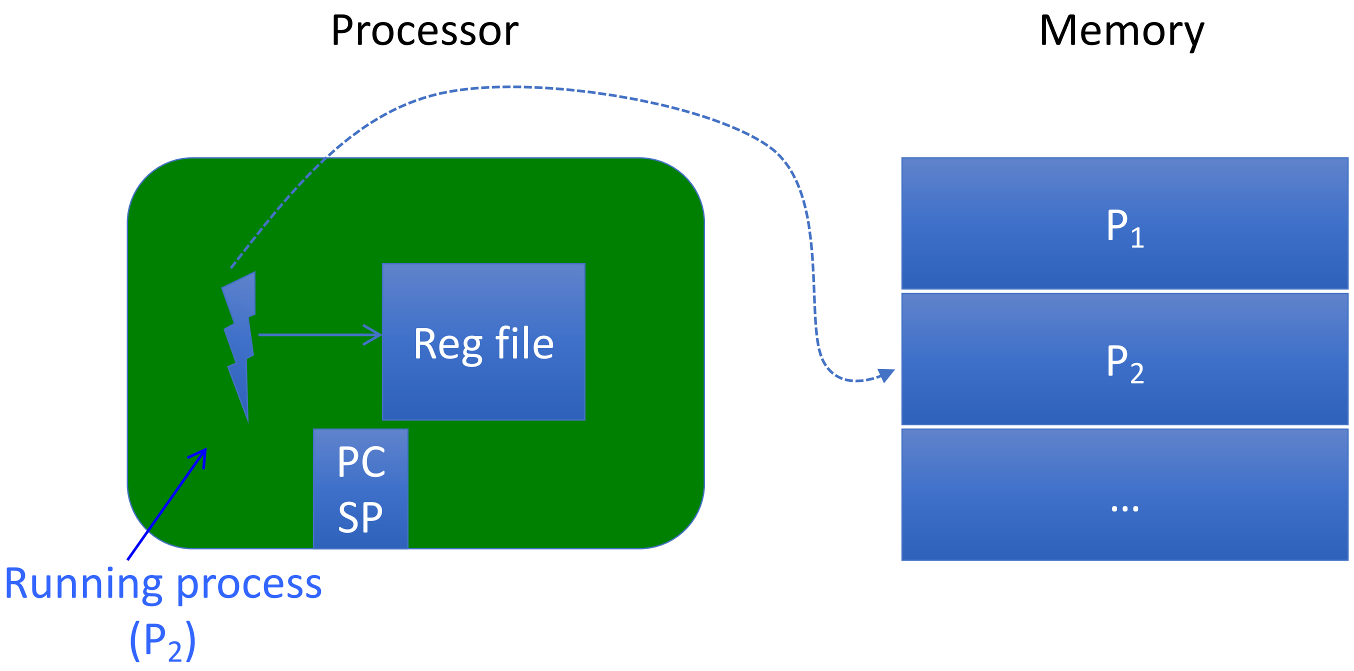
For I/O

Memory footprint
of
User program

Process state

Process$_2$ is running. The program is in memory. Where is its state?

Where will its state be when some other process is running?



PCB – Process Control Block

We need a data structure to represent a process' state

```
enum  state_type {new, ready, running,
                     waiting, halted};
typedef struct control_block_type {
   enum state_type state;
   address PC;
   int reg_file[NUMREGS];
   struct control_block *next_pcb;
   int priority;
   address memory_footprint;
   ….
   ….
} control_block;
```

Session ID: dag2200

One of these is NOT part of the PCB

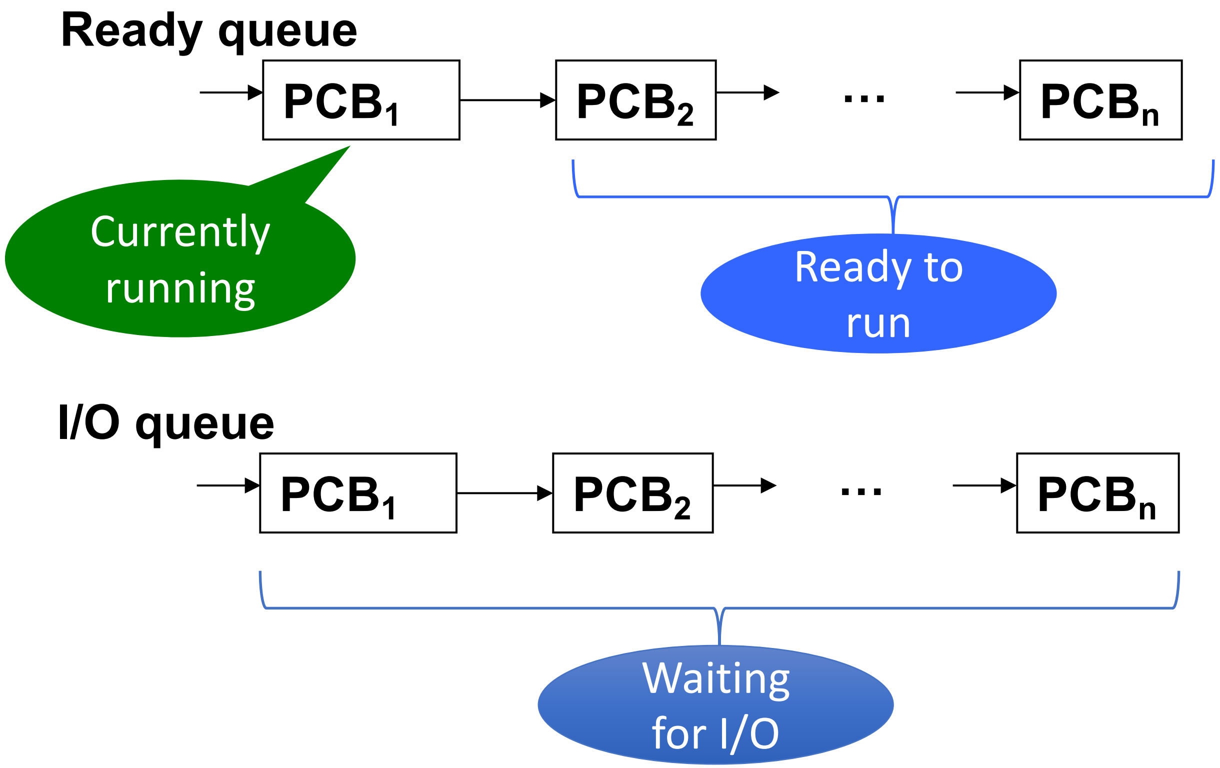
- 20% A. General purpose registers that are visible to the instruction set
- 9% B. Program counter and the register that represents the stack pointer
- 31% C. Layout of the program in memory
- 34% D. Internal registers in the datapath of the processor
- 6% E. Priority information

PCB – Process Control Block

```
enum  state_type {new, ready, running,
                   waiting, halted};
typedef struct control_block_type {
    enum state_type state;
    address PC; // where to resume
    int reg_file[NUMREGS]; // reg contents
    struct control_block *next_pcb;
    int priority; // extrinsic attribute
    address memory_footprint; // memory occupancy
    ….
    ….
} control_block;
```

volatile state

Data structure used by scheduler



Steps in scheduling

- Grab the attention of the processor
- Save the state of the current process
- Select a new process to run
- Dispatch the selected process

- Preemptive vs. non-preemptive

External interrupt (e.g. timer)

System call (trap), I/O request, process exit

Steps in scheduling

- Grab the attention of the processor
- Save the state of the current process
- Select a new process to run
- Dispatch the selected process

- Dump the "state" (PC, registers) into PCB of currently running process

Steps in scheduling

- Grab the attention of the processor
- Save the state of the current process
- Select a new process to run
- Dispatch the selected process

- This is the short-term scheduling algorithm result → select a PCB to "dispatch"

Steps in scheduling

- Grab the attention of the processor
- Save the state of the current process
- Select a new process to run
- Dispatch the selected process

- What is "dispatch"?
  ➔ load "state" of the selected PCB into processor registers (PC, reg file)

Steps in scheduling

This whole process is called a "context switch"

- Grab the attention of the processor
- Save the state of the current process
- Select a new process to run
- Dispatch the selected process

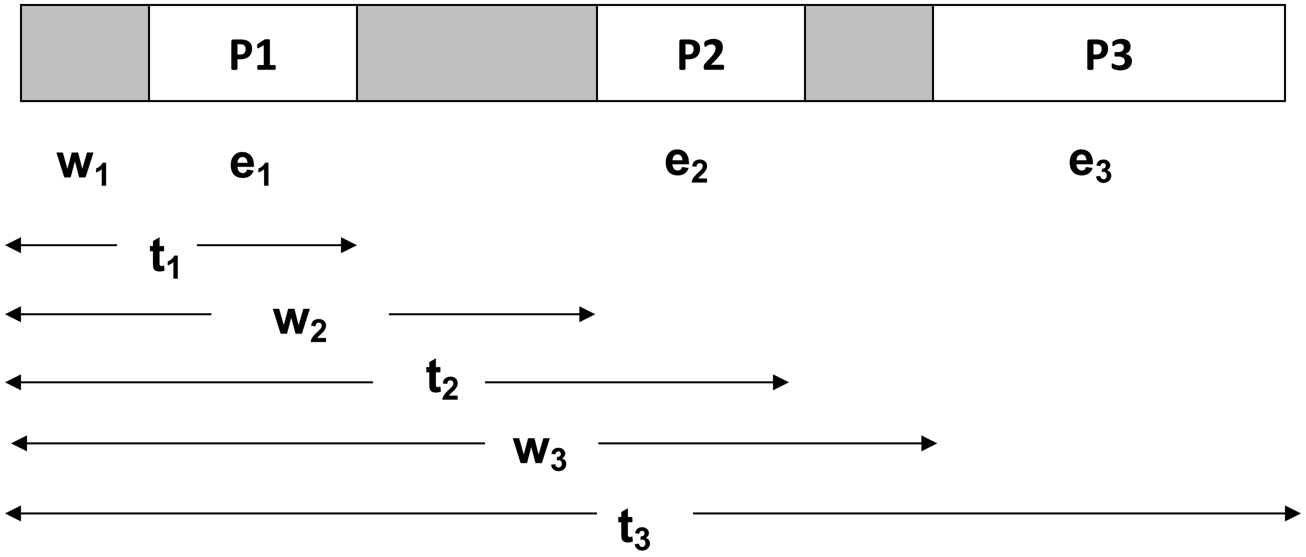
Name	Description
CPU burst	Continuous CPU activity by a process before requiring an I/O operation
I/O burst	Activity initiated by the CPU on an I/O device
PCB	Process context block that holds the state of a process (i.e., program in execution)
Ready queue	Queue of PCBs that represent the set of memory resident processes that are ready to run on the CPU
I/O queue	Queue of PCBs that represent the set of memory resident processes that are waiting for some I/O operation either to be initiated or completed
Non-Preemptive algorithm	Algorithm that allows the currently scheduled process on the CPU to voluntarily relinquish the processor (either by terminating or making an I/O system call)
Preemptive algorithm	Algorithm that forcibly takes the processor away from the currently scheduled process in response to an external event (e.g. I/O completion interrupt, timer interrupt)
Thrashing	A phenomenon wherein the dynamic memory usage of the processes currently in the ready queue exceed the total memory capacity of the system

Session ID: dag2200

A preemptive scheduling algorithm requires

- 40% A. A trap instruction
- 54% ✓ B. An external interrupt
- 3% C. The currently running process to terminate
- 3% D. The currently running process to make an I/O request

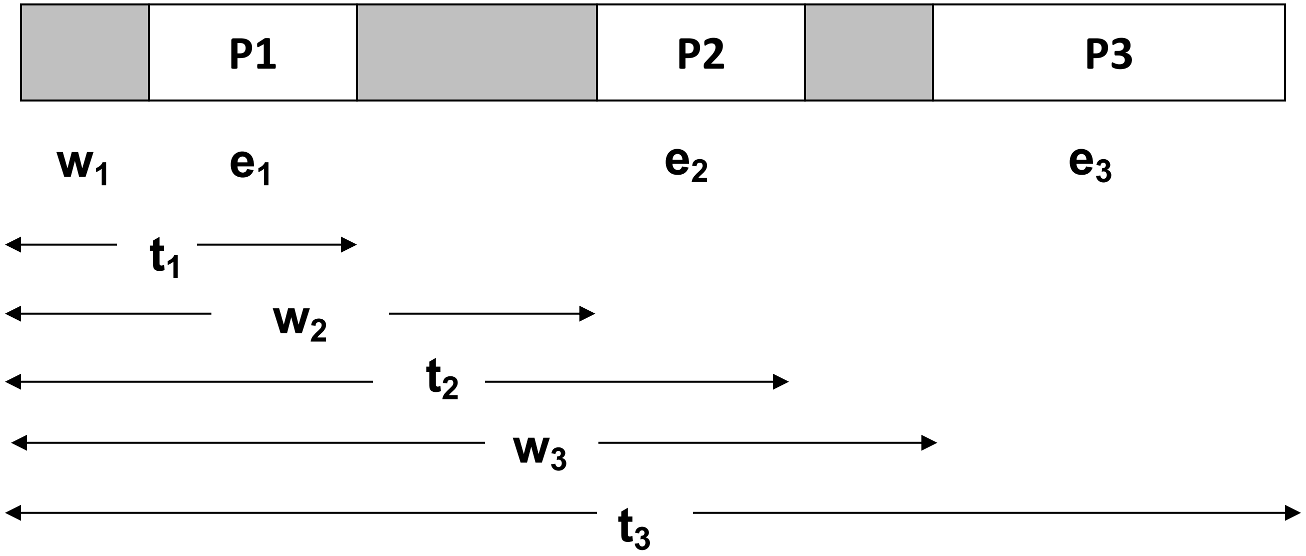
Metrics



w_i, e_i, and t_i, are respectively the wait time, execution time, and the elapsed time (turnaround time) for a job j_i

Throughput?
Avg. Turnaround Time?
Avg. Wait Time?
Response time?

Metrics



w_i, e_i, and t_i, are respectively the wait time, execution time, and the elapsed time (turnaround time) for a job j_i

System Centric

Throughput?	$3 / t_3$ jobs/sec
Avg. Turnaround Time?	$(t_1+t_2+t_3)/3$ sec
Avg. Wait Time?	$(w_1+w_2+w_3)/3$ sec

User Centric → Response time? $R_{P1}=t_1, R_{P2}=t_2, R_{P3}=t_3$

Name	Notation	Units	Description
CPU Utilization	-	%	Percentage of time the CPU is busy
Throughput	n/T	Jobs/s	System-centric metric quantifying the number of jobs *n* executed in time interval *T*
Avg. Turnaround time (t_{avg})	$(t_1+t_2+\ldots+t_n)/n$	Secs	System-centric metric quantifying the average time it takes for a job to complete
Avg. Waiting time (w_{avg})	$(w_1+w_2+ \ldots+w_n)/n$	Secs	System-centric metric quantifying the average waiting time that a job experiences
Response time	t_i	Secs	User-centric metric quantifying the turnaround time for a specific job *I*
Variance in Response time	$E[(t_i - t_{avg})^2]$	Secs^2	User-centric metric that quantifies the statistical variance of the actual response time (t_i) experienced by a process (P_i) from the expected value (t_{avg})
Starvation	-	-	User-centric qualitative metric that signifies denial of service to a particular process or a set of processes due to some intrinsic property of the scheduler
Convoy effect	-	-	User-centric qualitative metric that results in a detrimental effect to some set of processes due to some intrinsic property of the scheduler [This often appears as a “convoy” of short jobs waiting for the completion of a long job; non-preemptive FCFS is the convoy effect’s native habitat.]

Session ID: dag2200

The most user-centric metric of a scheduler is…

6% A. Throughput

28% B. Average waiting time

11% C. Average turnaround time

3% D. CPU utilization

50% E. Response time

3% F. None of the above