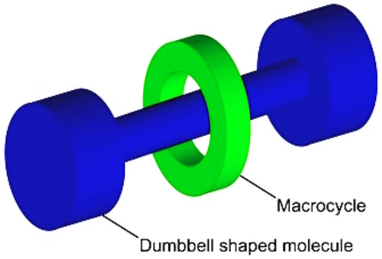
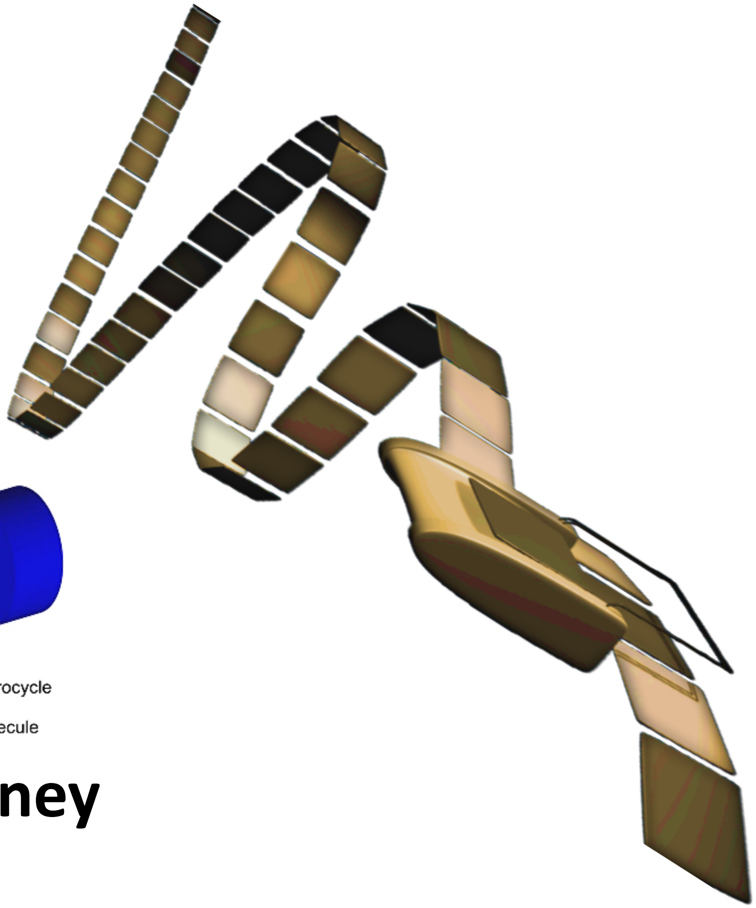
Introduction to Computer Science

CSCI 109





Andrew Goodney

Fall 2017

Lecture 8: Operating Systems *October 16, 2017*

Operating Systems

Working Together

Agenda

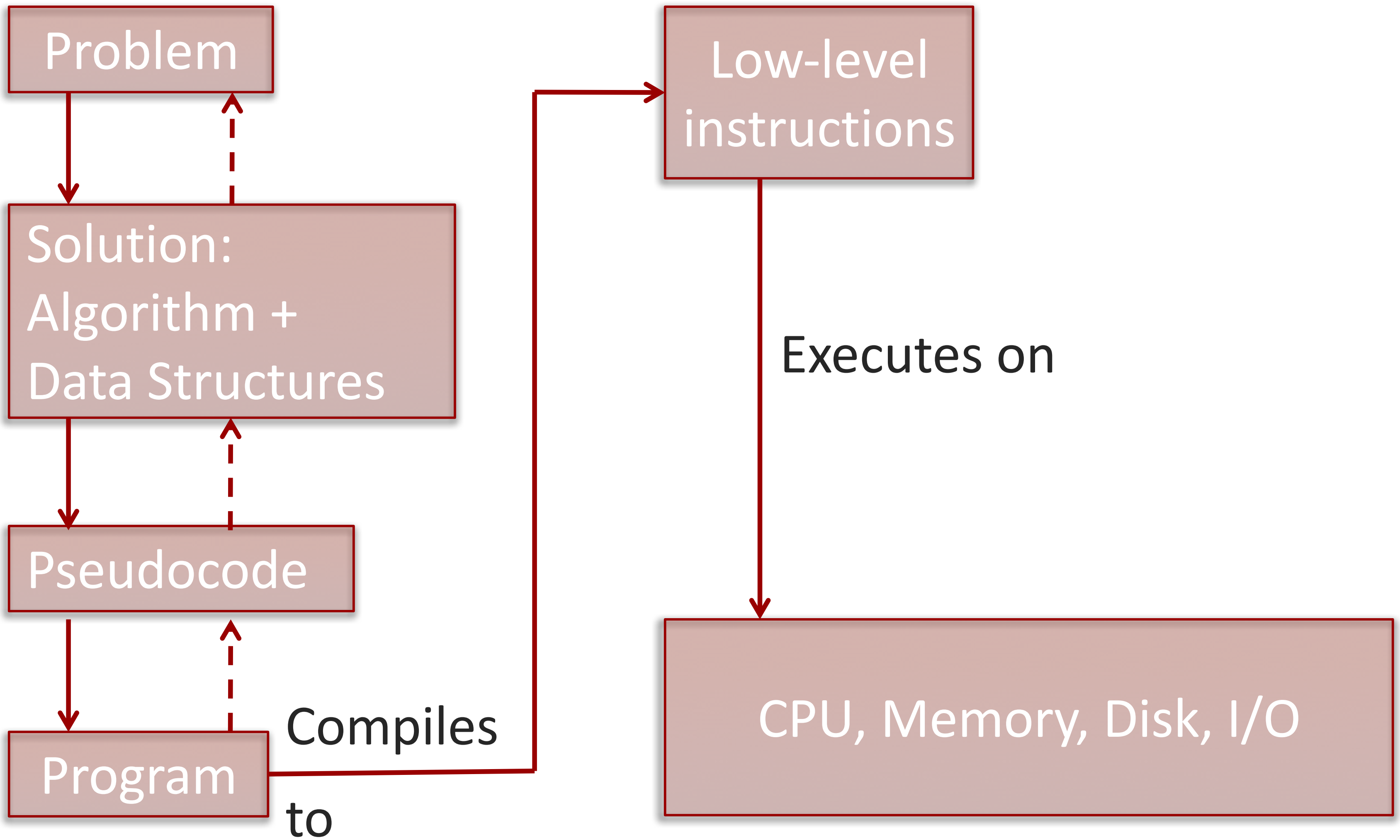
- Talk about operating systems
- Review quizzes 1-3
- Take quiz 4

Operating Systems

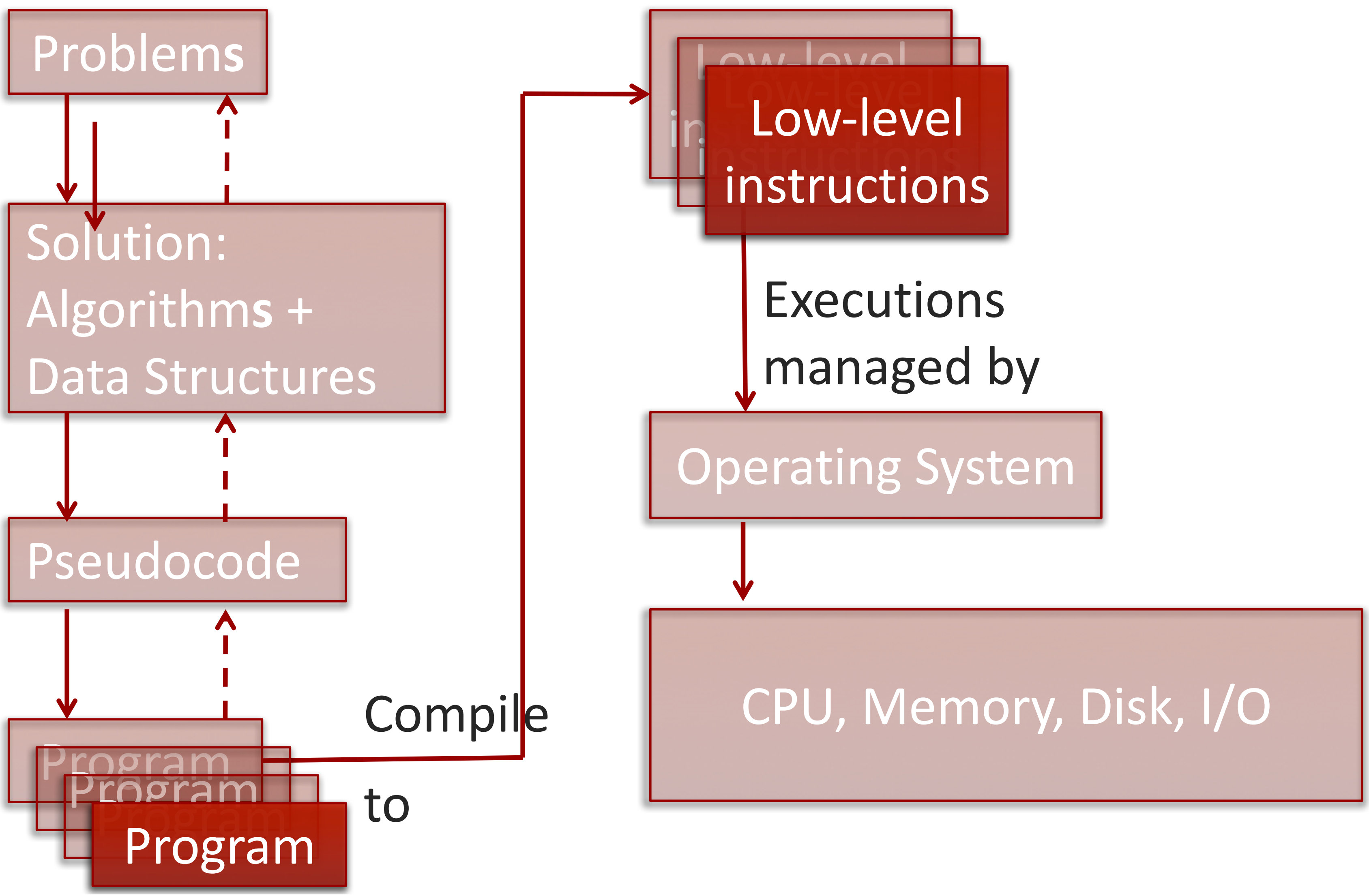
- What is an OS?
- The kernel, processes and resources
- Protection/Isolation/Security
- Competing for time
- Competing for space

Reading:
St. Amant Ch. 6

The need for an OS



The OS as a executive manager

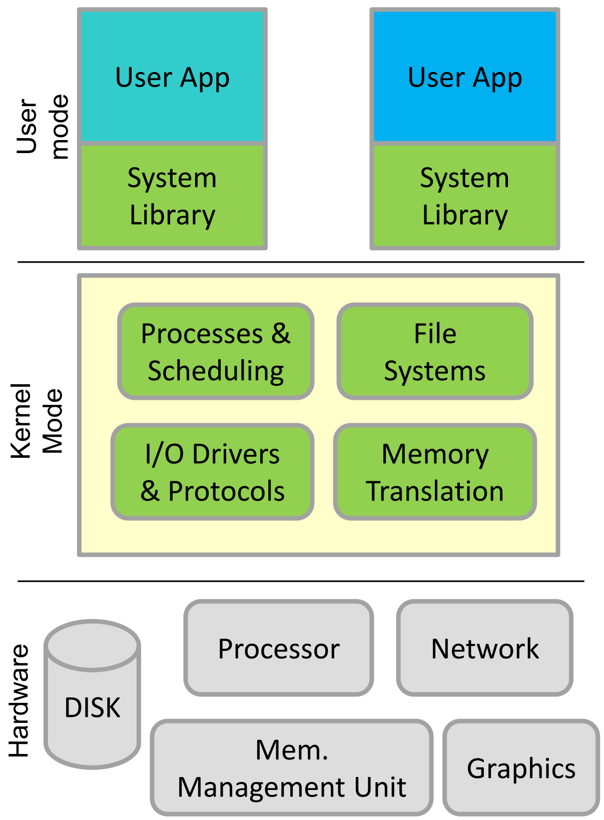


What is an Operating System?

- An executive manager for the computer
- Manages resources
  - Space (i.e. memory)
  - Time (i.e. CPU compute time)
  - Peripherals (i.e. input and output)
- OS is a program that starts, runs, pauses, restarts, and ends other programs
- (some content from the following slides is courtesy of Mark Redekopp and CS350)

Definition

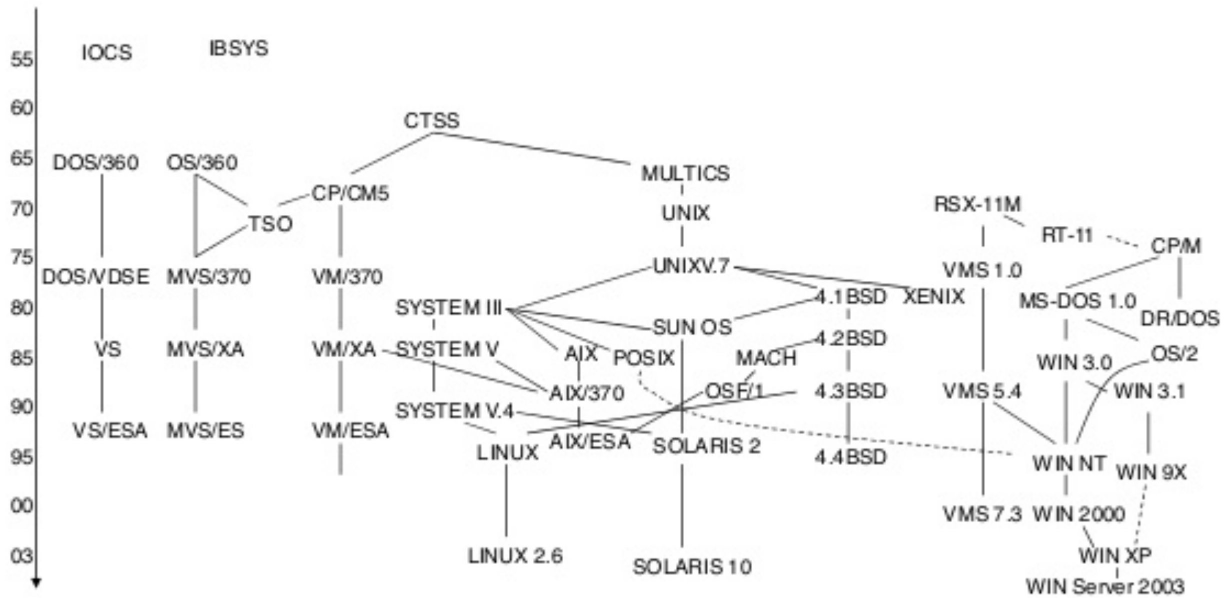
- A piece of software that manages a computer's resources
- What resources need managing?
  - CPU (threads and processes)
  - Memory (Virtual memory, protection)
  - I/O (Abstraction, interrupts, protection)



Examples of Operating Systems

- Microsoft Family
  - MSDOS, Windows 3.1 – 98, WindowsNT -> Windows 10
  - Predominately x86 (Intel) hardware, some PowerPC, some ARM
  - FreeDOS
- POSIX (UNIX/like)
  - macOS, FreeBSD, openBSD, netBSD, Solaris, AIX, and others
    - Run on most processor architectures
  - iOS
  - Linux
    - Little side project of university student
    - ”UNIX clone” that won the war
    - 20+ popular distributions
    - Android: heavily customized Linux and Java on phone/tablet
- Others
  - PlaystationOS, VxWorks

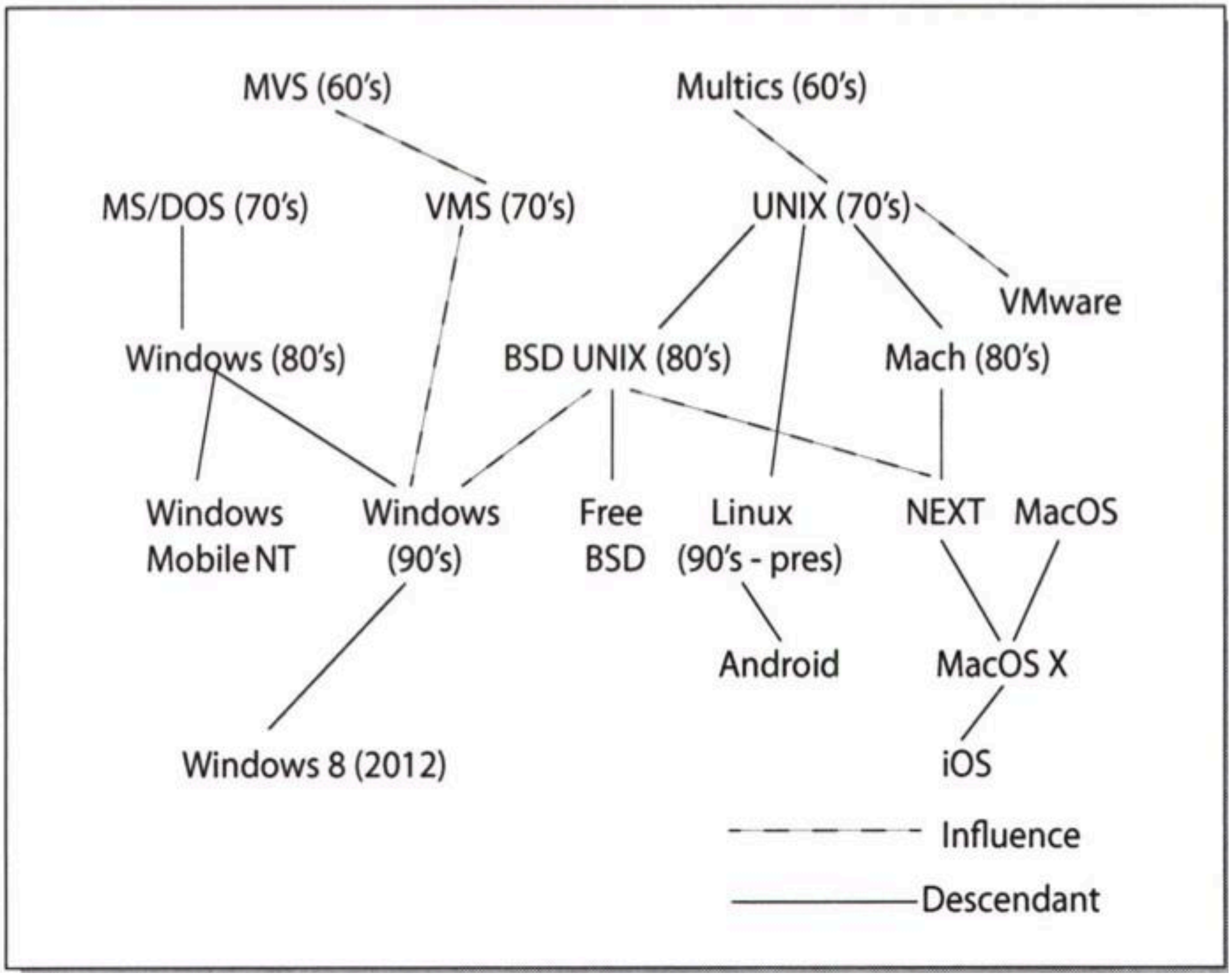
History of Modern Operating Systems



Operating Systems I | Intro, History

13

PT/FF 2014


MVS (60's)
Multics (60's)
MS/DOS (70's)
VMS (70's)
UNIX (70's)
VMware
Windows (80's)
BSD UNIX (80's)
Mach (80's)
Windows Mobile NT
Windows (90's)
Free BSD
Linux (90's - pres)
NEXT
MacOS
Android
MacOS X
iOS
Windows 8 (2012)
Influence
Descendant

Important Vocabulary

- Resource
  - Some part of the computer that programs use:
    - Memory, CPU, Input/Output devices
- Policy
  - Rules enforced by algorithms that share access to resources
- OS Developers (humans) write policies that achieve some set of goals for the operating systems

What does an Operating System do?

- A bare computer is just hardware
- Programs are written to use that hardware, but exclusive use is inefficient
- In simple terms, the OS:
  - Enables more than one program at a time to use the computer hardware
  - Present computer resources (CPU, disk, I/O) through abstract interfaces to allow sharing
  - Enforce policies to manage/regulate the sharing of resources

Roles

- Referee
  - Protection against other applications
  - Enforce fair resource sharing
    - Why doesn't an infinite loop require a reboot?
- Illusionist (Virtualization)
  - Each program thinks it is running separately
  - Each program thinks it has full access to computer's resources (or unlimited resources)
- Glue
  - Common services (such as copy/paste)
  - Files can be read by any application
  - UI routines for look & feel
  - Separate applications from hardware
    - so you don’t need to know which keyboard, disk drive, etc

OS Design Criteria

- Reliability (and availability)
- Security & Privacy
- Performance
- Portability

Reliability and Availably

- Reliable systems work properly
  - Correct (or expected) outputs are generated for a set of inputs
  - If this is not the case, the system has failed
    - Examples?
- Available systems are available to do work
- Available does not imply reliable
  - System can be available but not reliable (system has bugs, generates wrong results)
  - System can be reliable but not available
    - Crash every 5 minutes, but saves results and restarts 5 minutes later

Privacy, Security, Isolation

- For an OS security means the OS does not run unintended code or get into a compromised state
  - No virus/malware
- OS privacy means programs should not get access to data they should not have
  - Password keychains, files in other users directories
- Security and Privacy require some tradeoffs with performance, which is why OS's are not 100% secure
  - Some are better than others!

Portability

- Many machine types exist: x86, x86_64, PPC, ARM, MIPS
- Many different motherboards or hardware platforms exist: server with 8 CPUs 12 PCIe slots to RaspberryPi, to AppleTV, etc.
- OS with good portability abstracts these differences into a stable API so programmers don't notice
- Also, can the OS itself be ported to new hardware easily?
- Good portability leads to wide adoption
  - Linux, Windows

Performance

- What does performance mean?
  - Lots of computation?
  - Fluid GUI for game?
  - Low latency disk for database?
- OS balances these with policies
  - Major axis is throughput vs. response time
  - Different OS's are tuned based on use case
  - DB server has different policies than Windows gaming rig

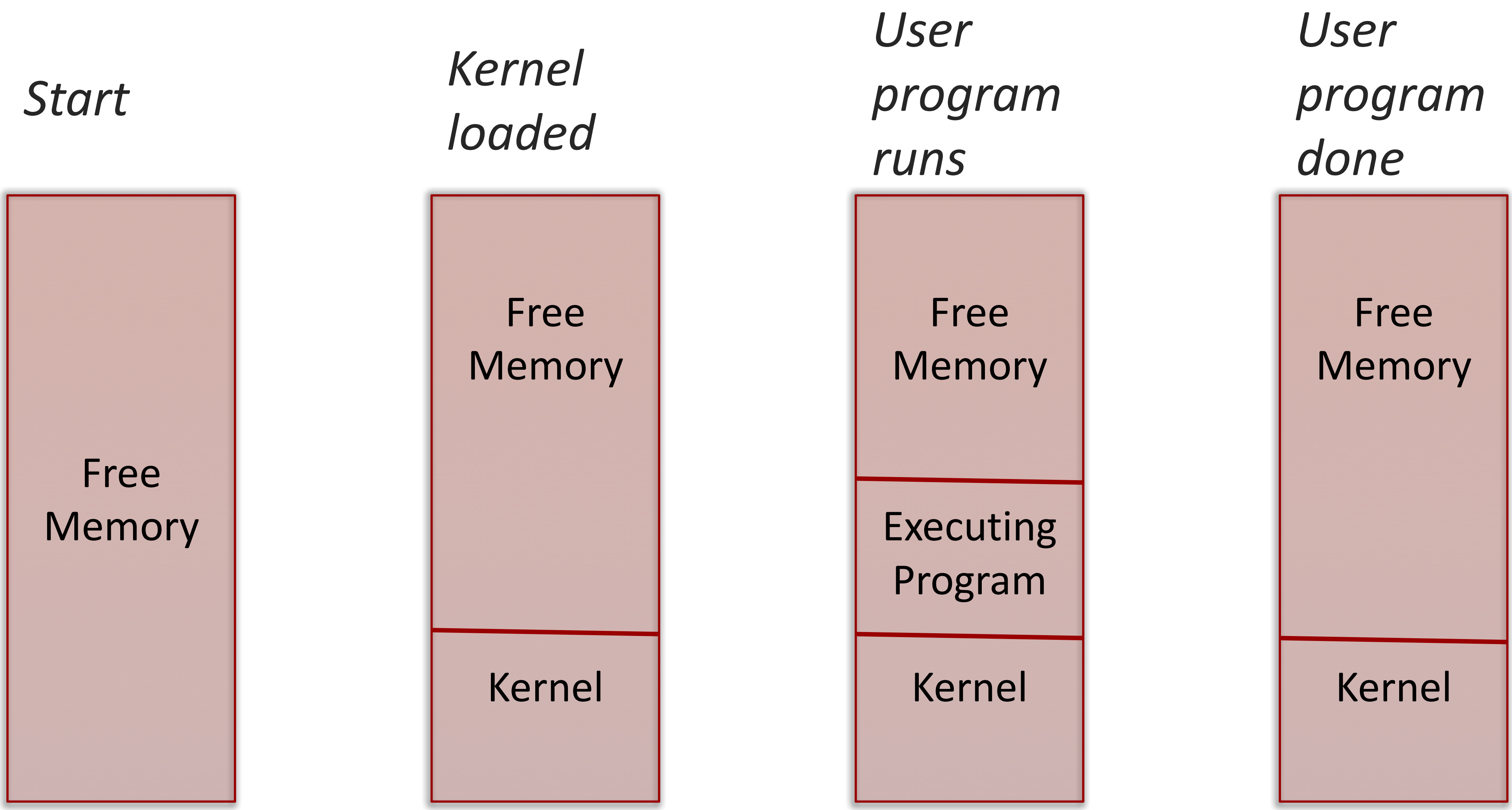
Examples of Policies

- Tasks are given priorities; higher priority tasks are handled first
- Some kind of tasks are never interrupted
- All tasks are equal priority; round-robin
- Some tasks can only use part of a disk
- Some tasks can use network

The kernel

- The kernel is the core of an OS
- Kernel coordinates other programs
- When the computer starts up the kernel is copied from the disk to the memory
- Kernel runs until some other program needs to use the CPU
- Kernel pauses itself to run other program

How memory is used



Multitasking

- One program uses the CPU at a time
- OS switches CPU usage (rapidly)
- Creates an illusion that all the programs are running at the same time
- Changeover from one program to another is called a *context switch*
- Examples of context switching?
- Can context switching be good for a program?
- Can context switching be good for a CPU?

Abstractions: Processes and Resources

- Resources
  - Space (memory)
  - Time (CPU)
  - Peripherals (printers etc.)
- Process: an executing program
  - Program counter
  - Contents of registers
  - Allocated memory & contents

- OS doesn't worry about what each program does
- Instead OS cares about
  - What resources does a process need?
  - How long will it run?
  - How important is it?

Protection/Isolation

- Other processes have to be prevented from writing to the memory used by the kernel
- Crash in one program shouldn't crash OS or other programs
- OS has access to all resources: privileged mode
- User programs have restricted access: user mode
- When a user program needs access to protected resources it makes a *system call* (e.g., managing files, accessing a printer)
- Principle of *least privilege* (kernel has highest privilege)

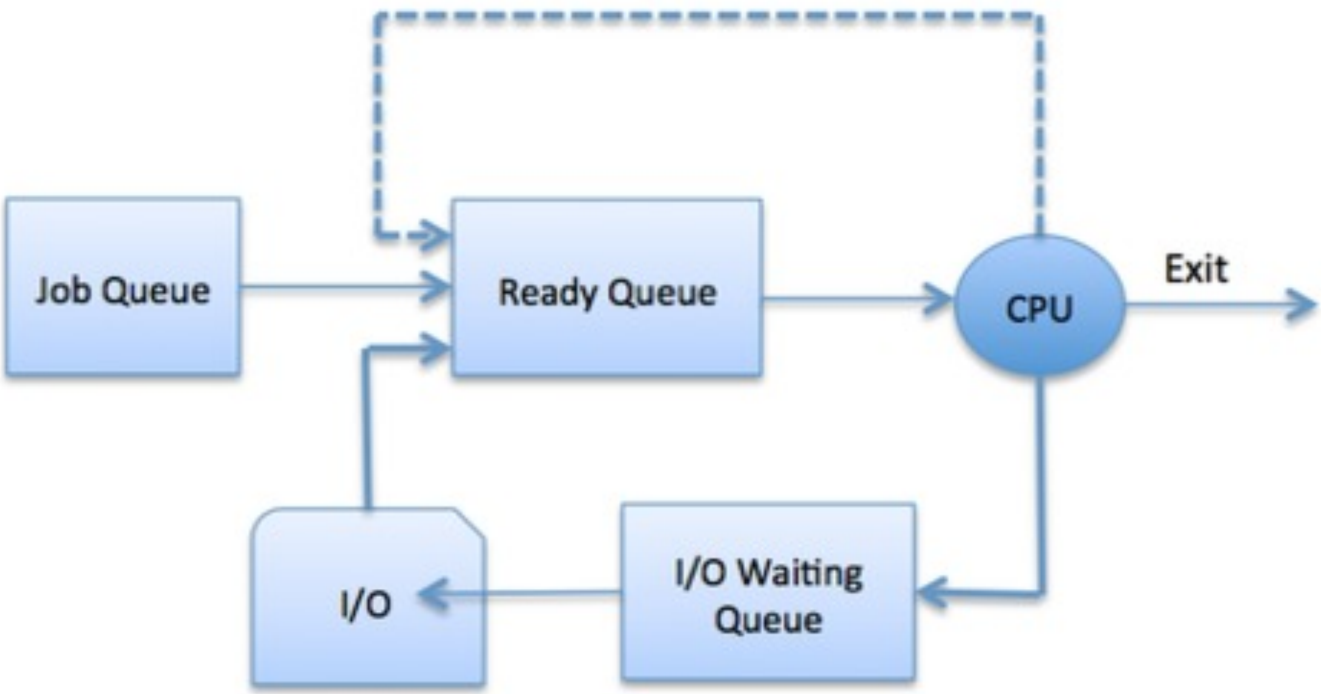
Competing for time

- Think of the time the CPU spends in chunks or blocks
- How can blocks of time be allocated to different processes so that work can be done efficiently?
- Policy: rules to enforce process prioritization

Process Scheduling Policies

- The process *queue*
- Round-robin
- First-come, first-served
- Priority-based
  - Preset priority for each process
  - Shortest-remaining-time
- All these policies keep the CPU busy
- Are there other ways to judge a policy?

Keeping CPU busy



How to evaluate a policy?

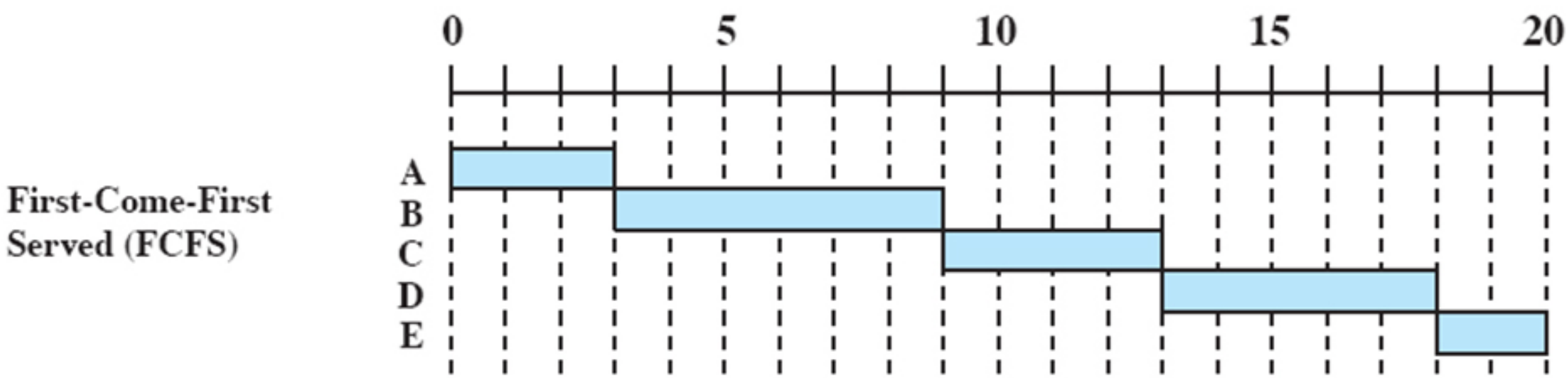
- Utilization: how much work the CPU does
- Throughput: # of processes that use the CPU in a certain time
- Latency: average amount of time that processes have to wait before running
- Fairness: every process gets a chance to use the CPU

	CPU utilization	Throughput	Latency	Fairness
Round-robin	Good	Variable	Potentially high	Yes No starvation
First-come first-served	Good	Variable		Yes
Shortest remaining time	Good	High	Potentially high	No Could have starvation
Fixed priority	Good			

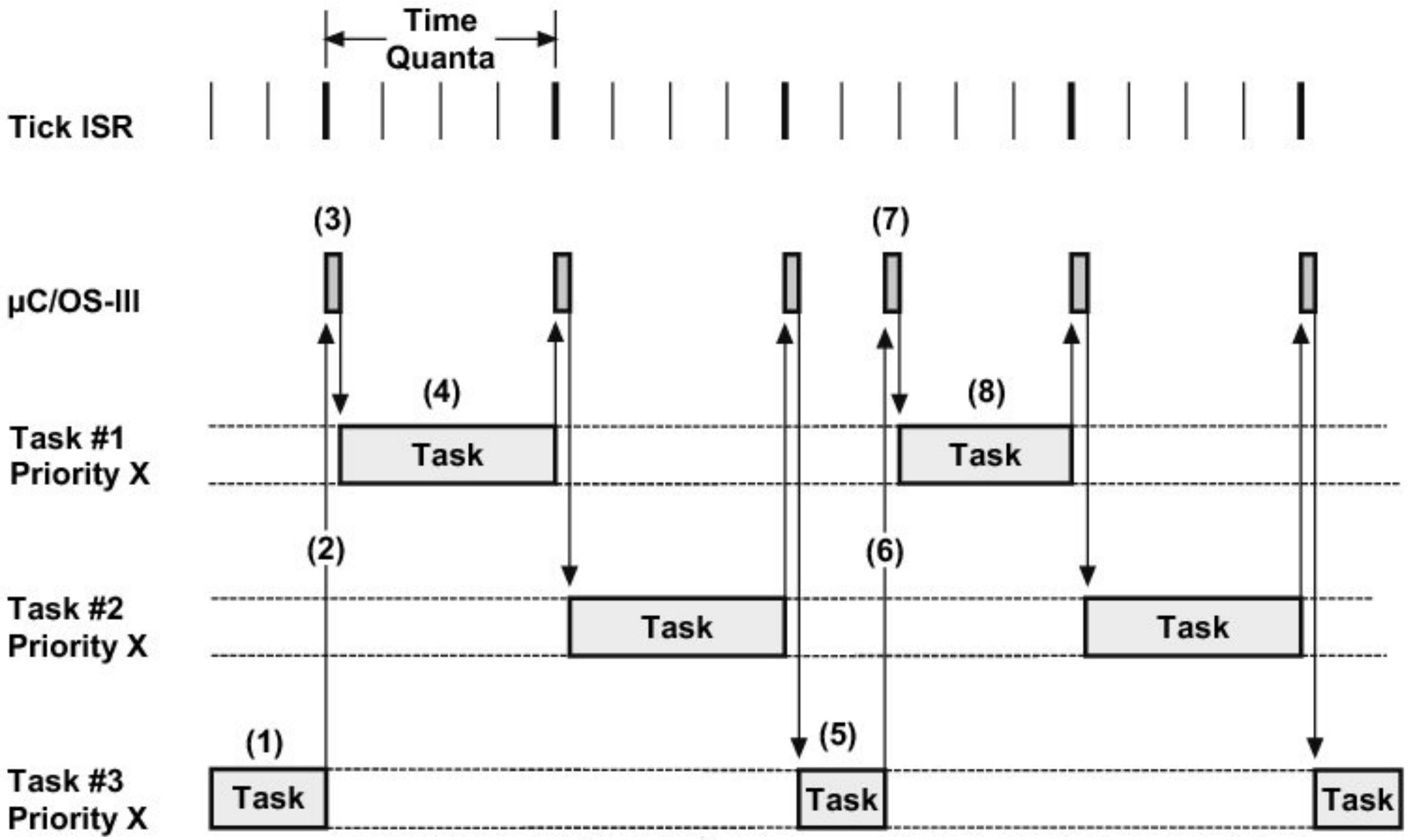
Everyday policies

- Planes taking off: first come first served
  - High efficiency for the runway
  - If several smaller planes in line before a large one, not efficient for the average passenger
- Traffic being directed at accident: round robin
  - First traffic in one direction, then another
  - If a police car arrives, then switch to priority-based
  - Unlikely to ever be shortest remaining time

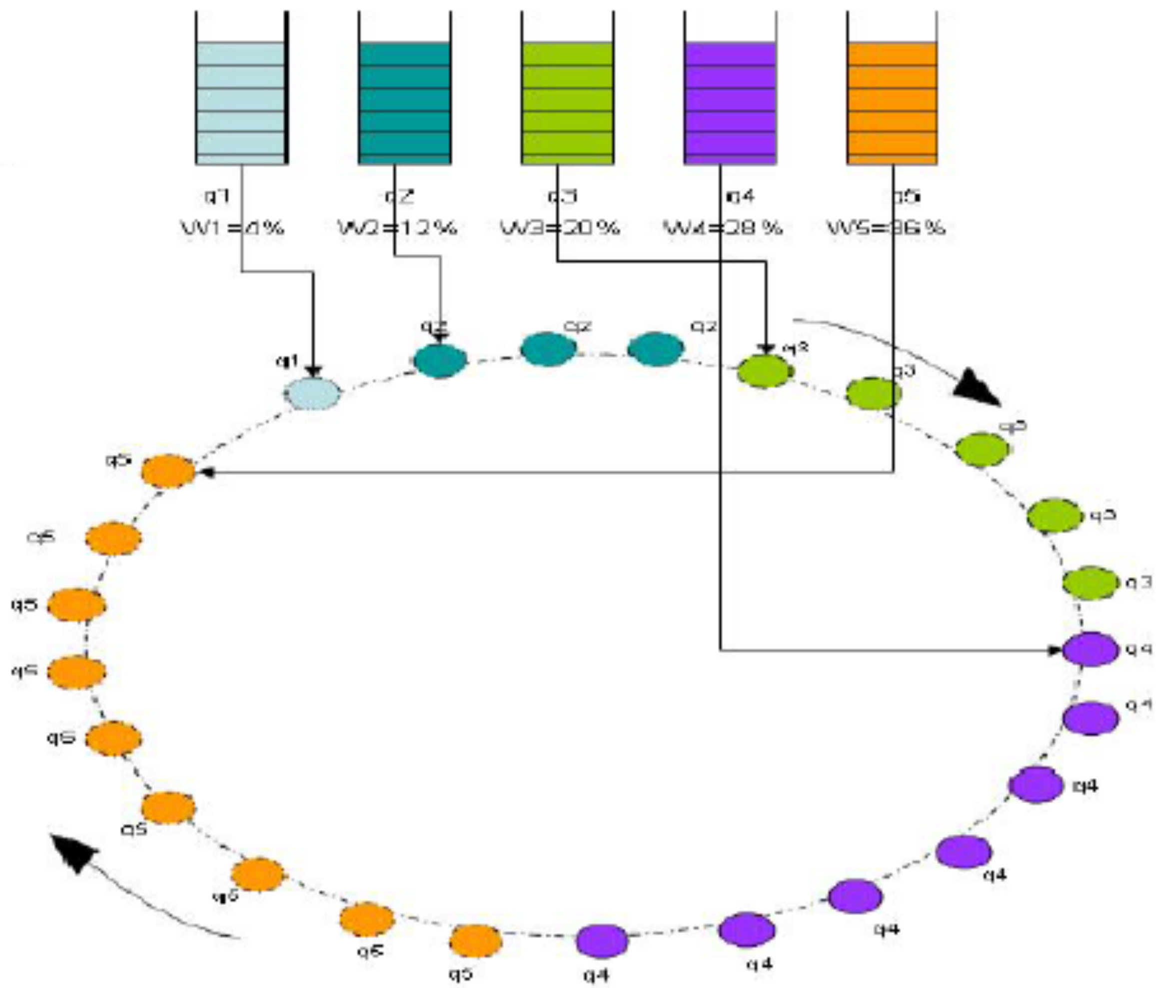
First-come, first serve (non-pre-emptive)



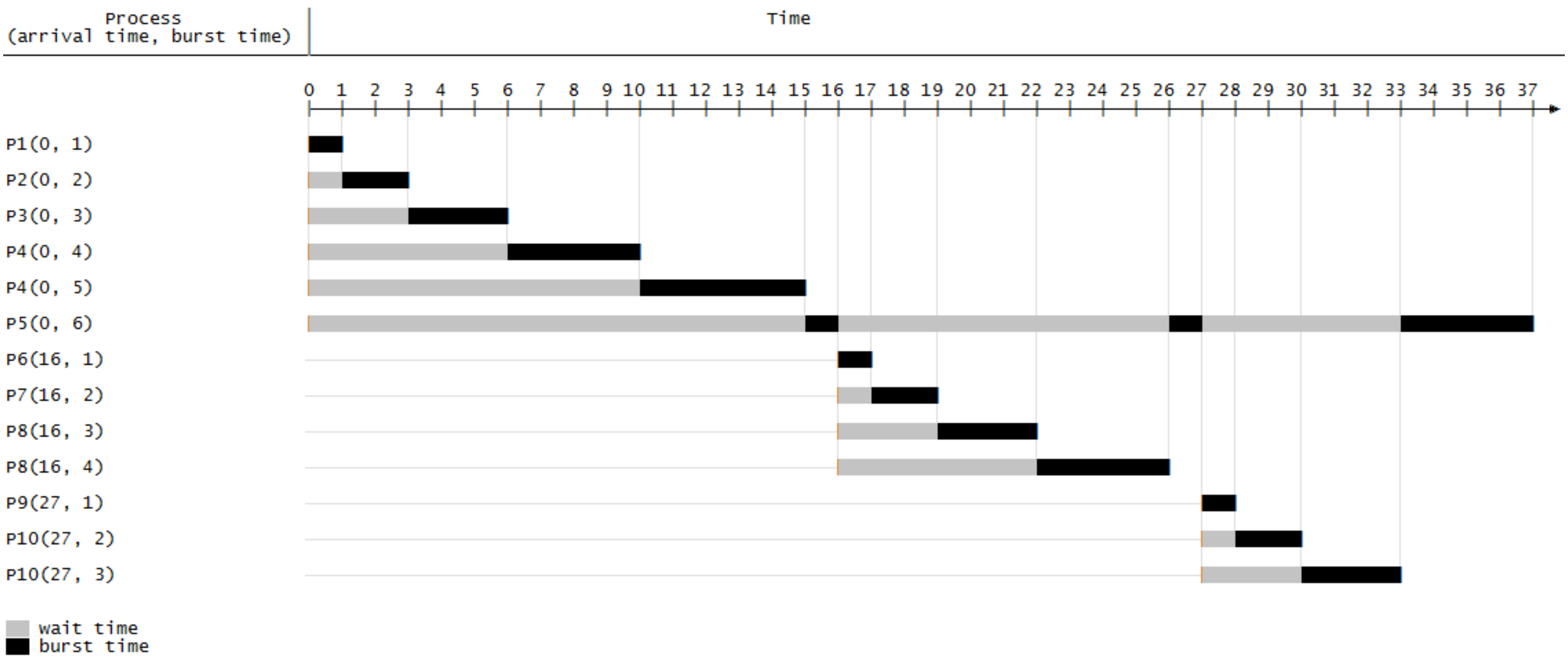
Round Robin (pre-emptive)



Weighted Round Robin

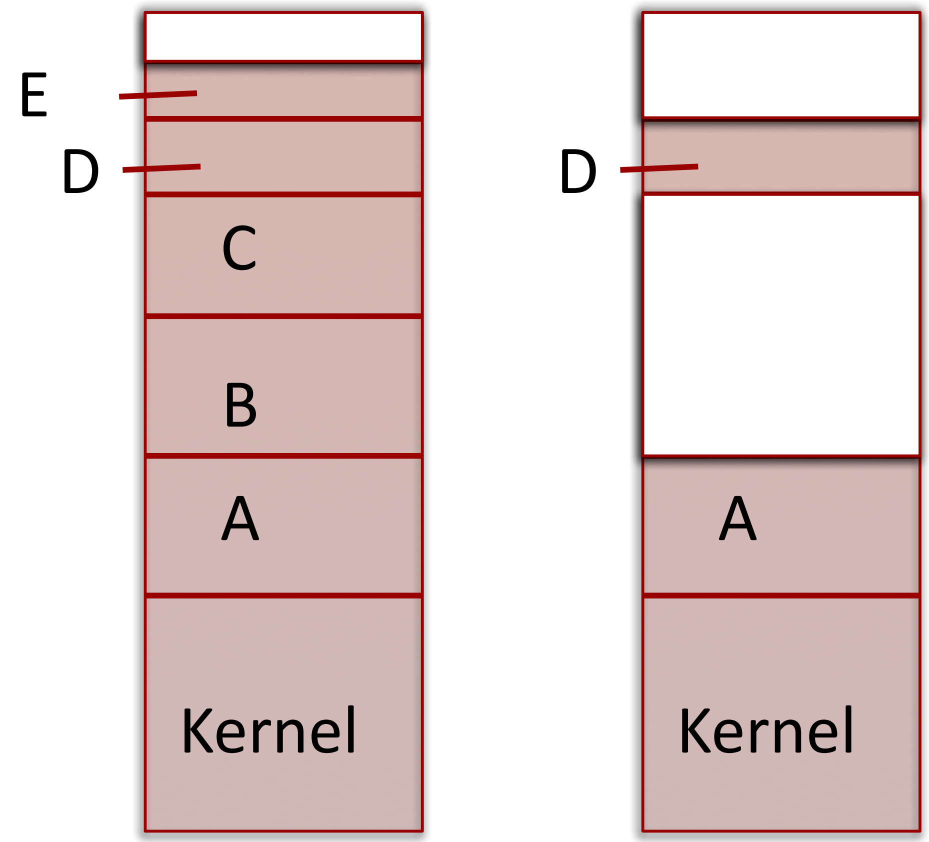


Shortest-time Remaining



Competing for space

- Unlike time, space can be reused
- When a process is running, it is allocated a single region of memory
- When a process finishes running, the memory allocated to it is given back to the OS
- *Fragmentation*



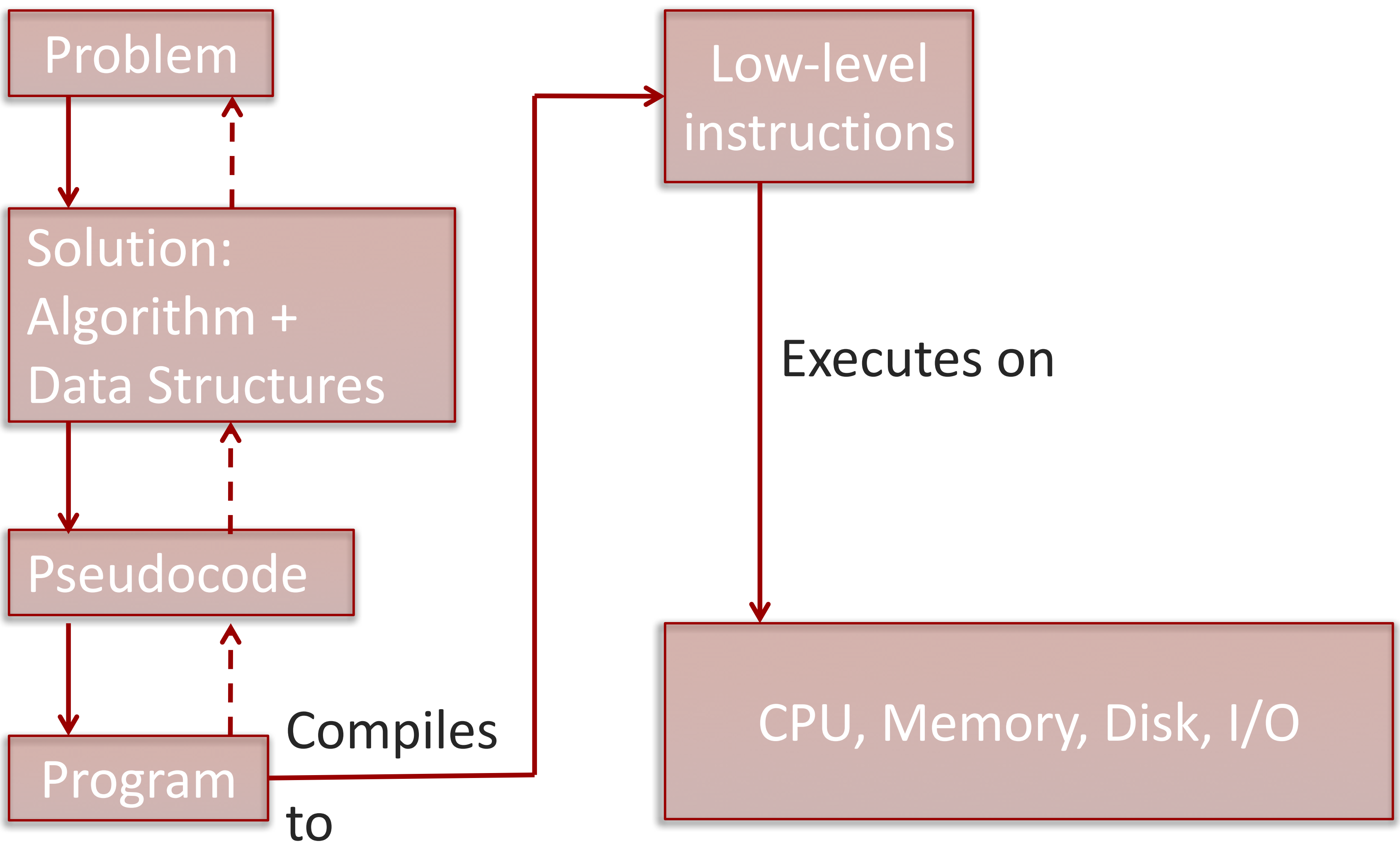
Dealing with fragmentation

- Move memory allocation around all the time in the background so when a new process needs memory, there is a large enough contiguous block to allocate
- Indirection
  - Physical memory ordering: memory divided into fixed size blocks called frames
  - Logical memory ordering: in logical memory each frame corresponds to a page (a renumbered frame)
  - When a process uses memory assigned to it, it uses logical addresses; the OS translates logical memory locations to physical memory locations

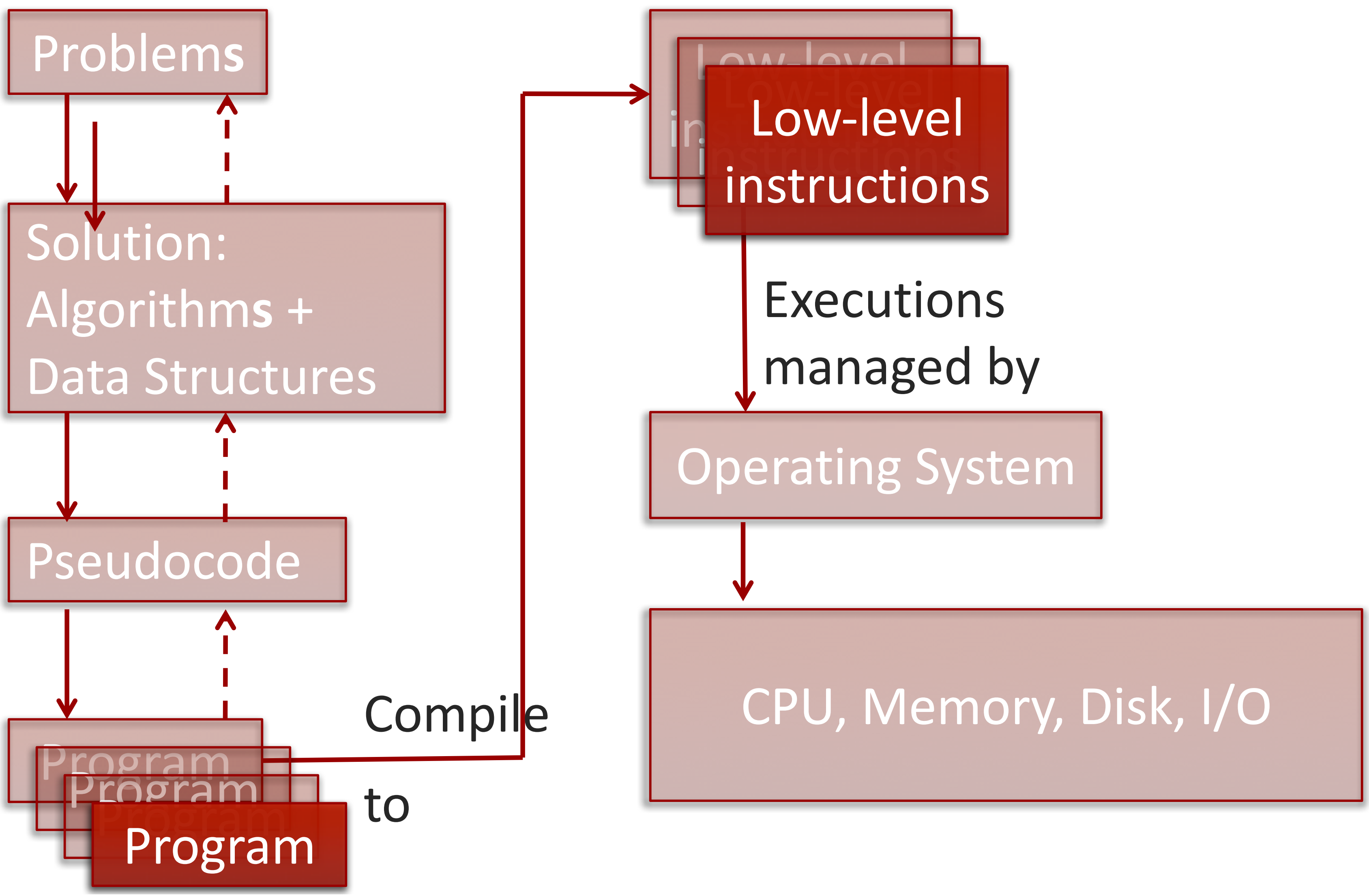
Virtual memory

- Use indirection to pretend RAM is bigger than it is
- Demand paging
  - When a process starts, the OS doesn't allocate it all the memory it needs
  - Instead only allocate memory needed to do immediate work
  - Keep rest of the requested memory in secondary storage (disk)
  - If a page in memory isn't used for a while, it's moved to disk
  - When a page is needed by a process, it's copied from disk to RAM
- Other examples of virtualization: virtual environments (render on demand), half-court basketball, etc.

The need for an OS



The OS as a executive manager



Schedule

Date	Topic		Assigned	Due	Quizzes/Midterm/Final
21-Aug	Introduction	What is computing, how did computers come to be?			
28-Aug	Computer architecture	How is a modern computer built? Basic architecture and assembly	HW1		
4-Sep	Labor day				
11-Sep	Data structures	Why organize data? Basic structures for organizing data		HW1	
12-Sep	Last day to drop a Monday-only class without a mark of “W” and receive a refund or change to Pass/No Pass or Audit for Session 001				
18-Sep	Data structures	Trees, Graphs and Traversals	HW2		**Quiz 1** on material taught in class 8/21-8/28
25-Sep	More Algorithms/Data Structures	Recursion and run-time			
2-Oct	Complexity and combinatorics	How "long" does it take to run an algorithm.		HW2	**Quiz 2** on material taught in class 9/11-9/25
6-Oct	Last day to drop a course without a mark of “W” on the transcript				
9-Oct	Algorithms and programming	(Somewhat) More complicated algorithms and simple programming constructs			**Quiz 3** on material taught in class 10/2
16-Oct	Operating systems	What is an OS? Why do you need one?	HW3		**Quiz 4** on material taught in class 10/9
23-Oct	Midterm	Midterm			**Midterm** on all material taught so far.
30-Oct	Computer networks	How are networks organized? How is the Internet organized?		HW3	
6-Nov	Artificial intelligence	What is AI? Search, planning and a quick introduction to machine learning			**Quiz 5** on material taught in class 10/30
10-Nov	Last day to drop a class with a mark of “W” for Session 001				
13-Nov	The limits of computation	What can (and can't) be computed?	HW4		**Quiz 6** on material taught in class 11/6
20-Nov	Robotics	Robotics: background and modern systems (e.g., self-driving cars)			**Quiz 7** on material taught in class 11/13
27-Nov	Summary, recap, review	Summary, recap, review for final		HW4	**Quiz 8** on material taught in class 11/20
8-Dec	Final exam 11 am - 1 pm in SAL 101				**Final** on all material covered in the semester