Processor Wrap-up & OS Start

Introduction to Computer Systems, Fall 2022

Instructor: Travis McGaha

TAs:

Ali Krema	Andrew Rigas	Anisha Bhatia
Audrey Yang	Craig Lee	Daniel Duan
David LuoZhang	Eddy Yang	Ernest Ng
Heyi Liu	Janavi Chadha	Jason Hom
Katherine Wang	Kyrie Dowling	Mohamed Abaker
Noam Elul	Patricia Agnes	Patrick Kehinde Jr.
Ria Sharma	Sarah Luthra	Sofia Mouchtaris

How are you?

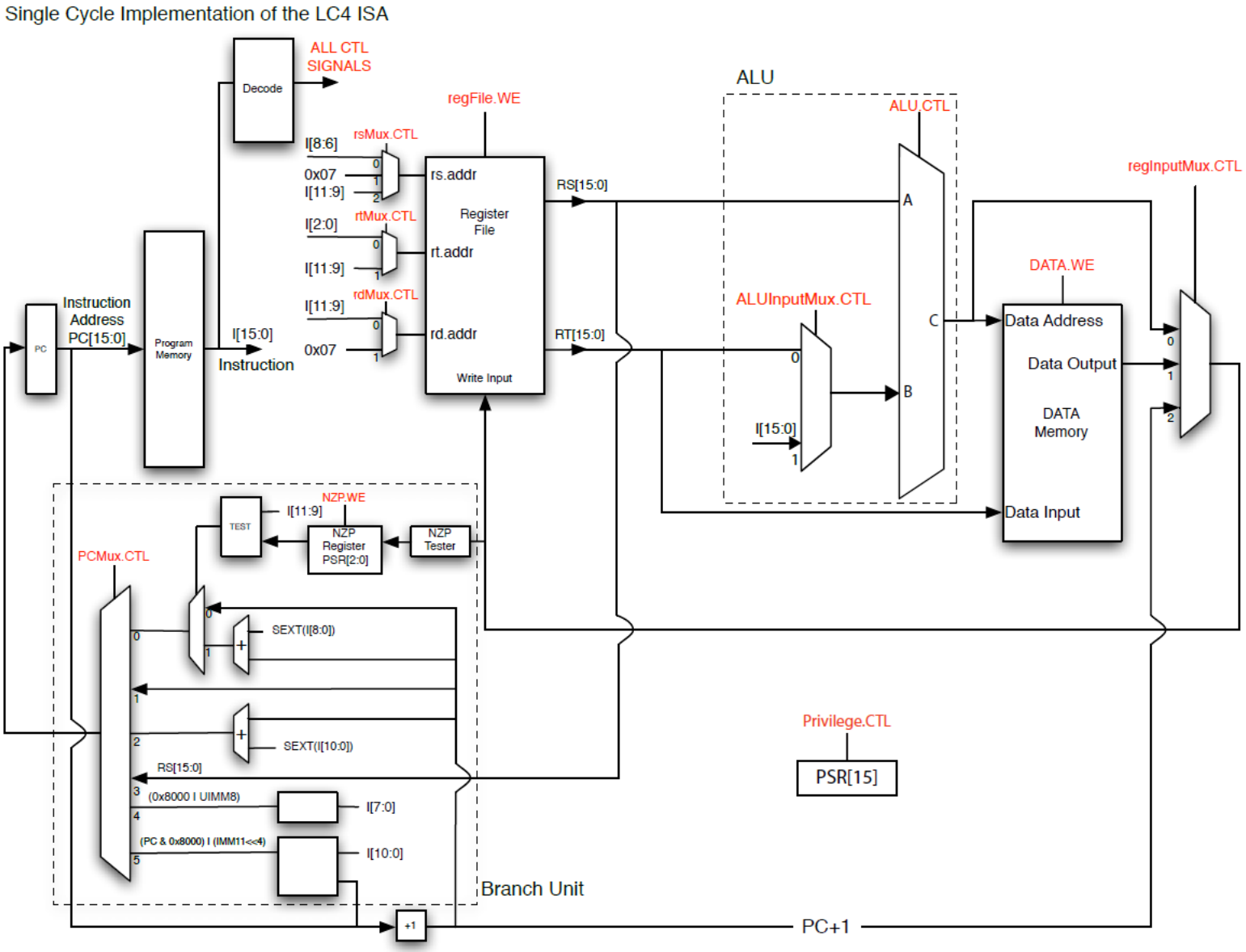
Logistics

- HW04 LC4 Programming: **This Friday** 10/14 @ 11:59 pm
  - Should have everything you need
  - Practice in Recitations this week
  - Normal programming assignment ☺
- Check-in05: releasing on Thursday (tomorrow)
- HW05 Control Signals: to be released this Friday
  - Programming assignment
- Final Exam information posted on Ed & Website

Lecture Outline

- **Processor Wrap-up**
  - **When we don't care about signals**
  - **"Single Cycle"**
- What is an OS?

Last Lecture:



Last Lecture:

- Last Lecture we went over how to determine control signals for an instruction
- The process given works pretty well for determining which signal to use, if that signal matters
- Other problem: How do we determine if that signal value matters for that instruction?
  - We saw this with the JMP activity

Determining if a Control Signal Matters

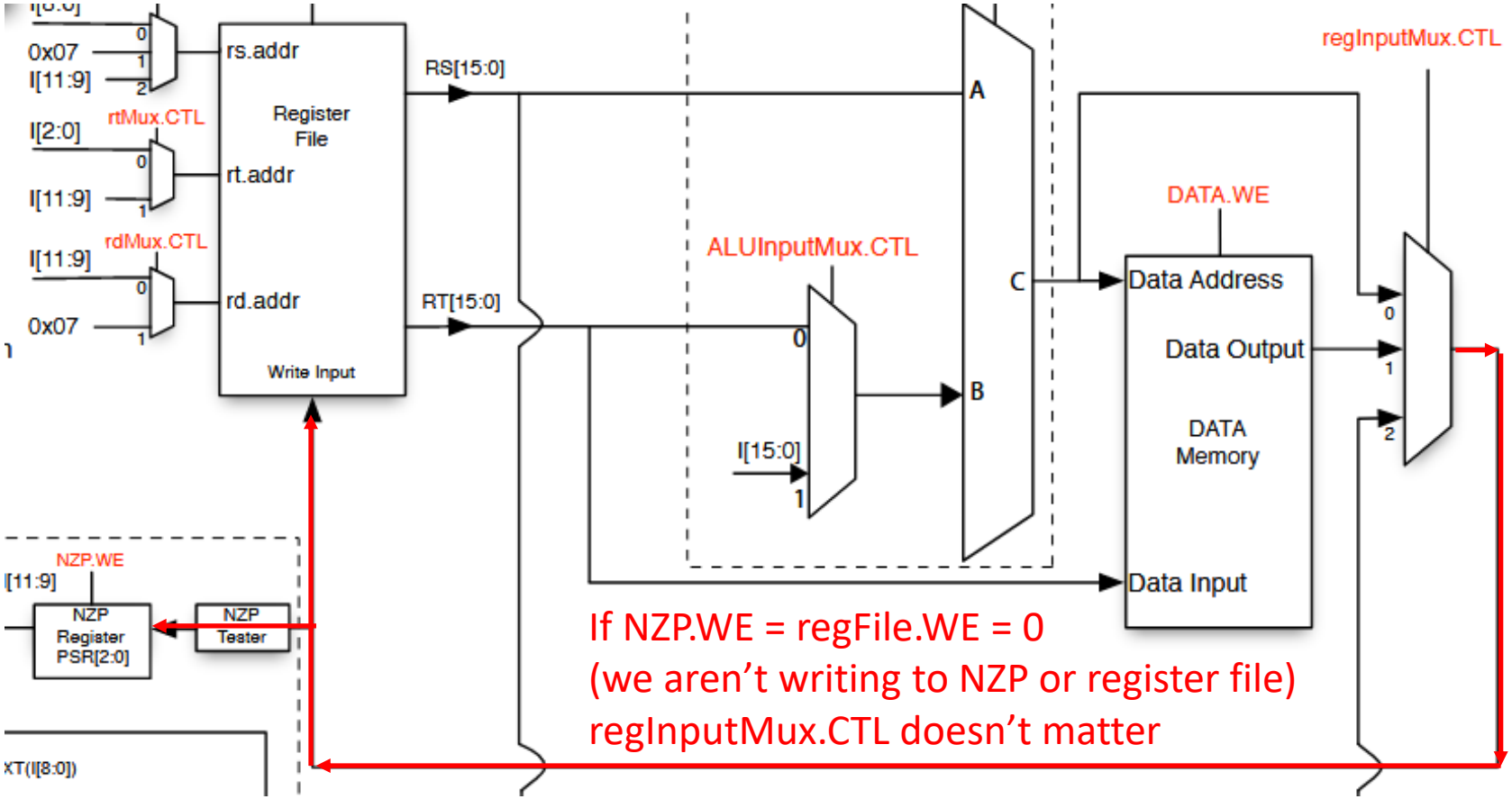
- Privilege.CTL, PCMux.CTL, and WE signals **always** matter
  - Why PCMux.CTL?
    - All instructions update the PC
  - Why WE?
    - All wires have a signal. If WE is ever 1, then something (even if it is garbage) will be written. Instructions either write or they don't, never "maybe"
  - Why Privilege.CTL?
    - Similar to WE, the signal decides if Privilege is updated

- We can determine if a signal matters by looking at:
  - The whole Single Processor Schematic
  - The signal decides some output, is this output used anywhere in the circuit?

regInputMux.CTL

- Example: when does regInputMux.CTL not matter?

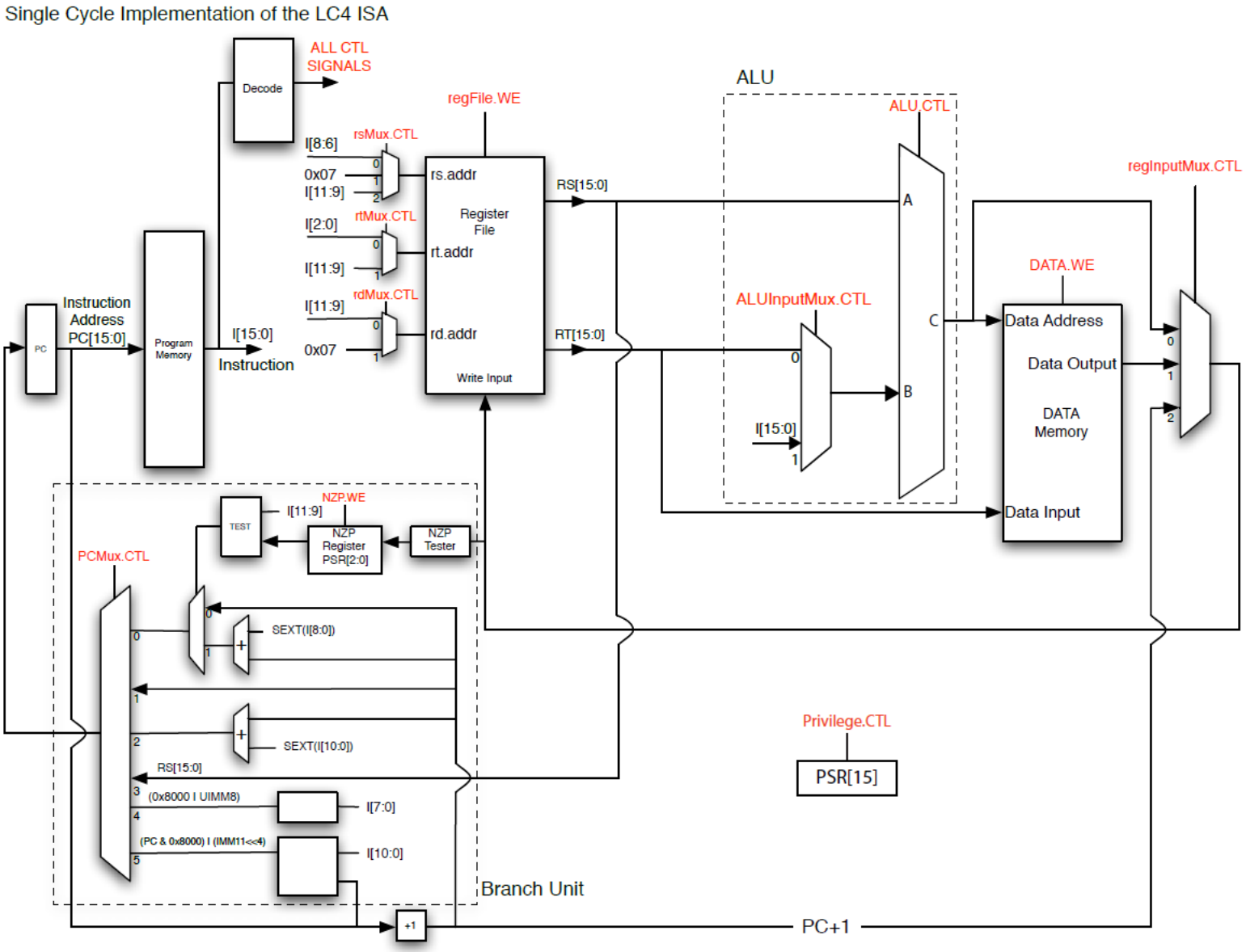
1st: Where does regInputMux.CTL output go? Answer: Register file and NZP



pollev.com/tqm

- What are the control signals for the CMPIU instruction?
  - 11 different control signals questions on PollEv
- Probably want to pull up the Control Signals Description, LC4 Instruction, and Single Cycle Sheet
- If you are reading the slides after lecture and want to go over this, should probably watch the lecture recording

Questions?


Single Cycle Implementation of the LC4 ISA
Decode
ALL CTL SIGNALS
regFile.WE
rsMux.CTL
rtMux.CTL
rdMux.CTL
I[8:6]
0x07
I[11:9]
I[2:0]
rs.addr
rt.addr
rd.addr
Register File
Write Input
RS[15:0]
RT[15:0]
ALU
ALU.CTL
A
B
C
ALUInputMux.CTL
I[15:0]
regInputMux.CTL
DATA.WE
Data Address
Data Output
DATA Memory
Data Input
PC
Instruction Address PC[15:0]
Program Memory
Instruction
NZP.WE
TEST
NZP Register PSR[2:0]
NZP Tester
PCMux.CTL
SEXT(I[8:0])
SEXT(I[10:0])
(0x8000 | UIMM8)
I[7:0]
(PC & 0x8000) | (IMM11<<4)
I[10:0]
Privilege.CTL
PSR[15]
Branch Unit
+1
PC+1

"Single Cycle"

- These past two lectures I've been talking about processor with the term "Single Cycle"
  - This means that one instruction is executed in one clock cycle.
  - That means the length of the program is directly proportional to the number of instructions executed

- "Single Cycle" is a convenient way for programmers to think about the processor, but modern processors are **not** like this
  - LC4 stands for "Little Computer 4", provides a nicer environment to get used to systems concepts

Single Cycle Clock Speed

- When we guarantee each instruction completes in one cycle, the clock period must be timed to make sure the longest instruction still completes in one clock cycle
  - Instruction execution time is affected by the number of gates and the propagated gate delay

- Hypothetical Example:
  - Most LC4 instructions take 1 ns to fully execute
  - The longest instructions take 5 ns to fully execute
  - Clock period must be at least longer than 5 ns for our single cycle processor
  - In this case, most instructions will cause us to "waste" ~4ns of each clock cycle

Component Usage

- Our computer at this point is made of many components

- Not all components are used at all times
  - While an instruction is being fetched from memory only the program memory is being used
  - While an instruction is still being decoded, the control signals haven't been set properly and rest of the hardware is not being used for computation
  - Some instructions don't use all components
  - Etc.

Instruction Level Parallelism

Note Testable Material (but still cool)

- When multiple instructions are executed on hardware at the same point in time

- Many related concepts:
  - Pipelining
  - Superscalar execution
  - Out of Order Execution
  - Speculative Execution
  - …
  - More in CIS 4710 ☺

Pipelining

Note Testable Material
(but still cool)

- What if we broke an instruction into stages and did one stage per clock cycle?
  - Example: Fetch -> Decode -> Compute -> Update Registers
- While one instruction is being fetched, another instruction is being decode, another is being computer, and another is updating registers
- We still complete one instruction per clock cycle, but are making progress on multiple instructions per clock cycle
  - Clock can be faster since we only need to consider maximum gate delay for a single stage

Superscalar Processors

Note Testable Material (but still cool)

- What if we had multiple instruction decoders, ALU's, and could fetch multiple instructions at a time?
  - We could run multiple instructions at the same time!

- Superscalar (and pipelining) have dependency issues:
  - Some instructions depend on the output of previous instructions, how do we know if we can execute two instructions at the same time and get the same output?

Additions are independent,
Could compute both at the same time

```
ADD R3, R2, R0
ADD R6, R7, R5
```

Second addition is dependent on first,
cannot run both in parallel

```
ADD R3, R2, R0
ADD R6, R7, R3
```

Lecture Outline

- Processor Wrap-up
  - When we don't care about signals
  - "Single Cycle"
- **What is an OS?**

LC4 is Little

- "LC4" -> Little Computer 4

- What is LC4 missing when you think of a "typical" modern computer?
  - Graphics
  - Keyboard & Mouse input
  - Files
  - Printing
  - Multiple Programs running at once
  - …

I/O

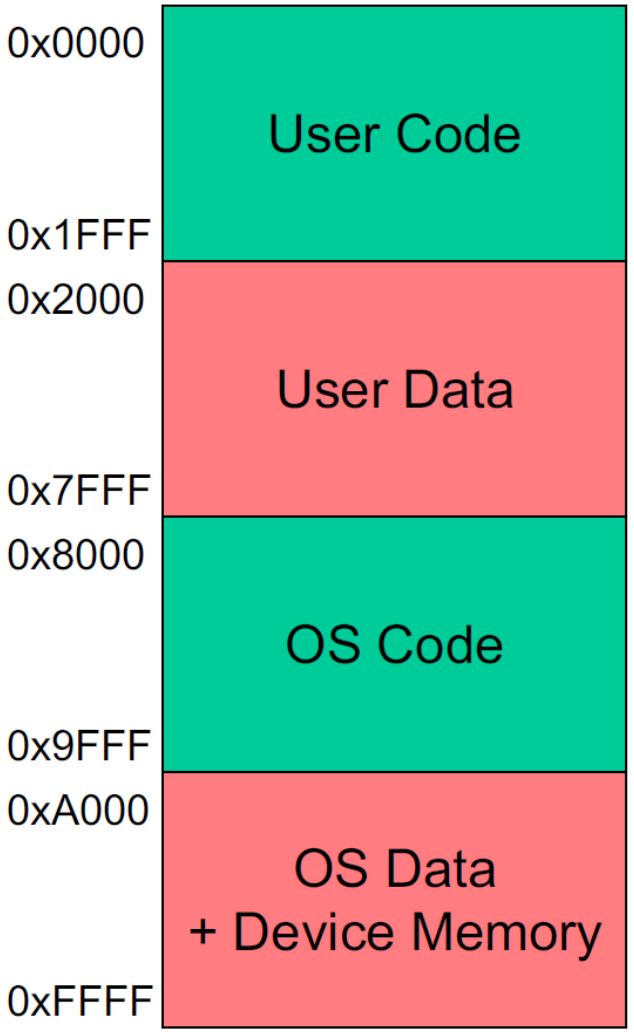
- Reading/writing anything "beyond" memory is called I/O
  - We call the locations we read/write to I/O devices
- I/O devices include:
  - Keyboard
  - Mouse
  - Files
  - Graphics Displays
  - Networks
  - Etc.
- I/O is handled by the Operating System (OS)

Operating Systems

- An Operating systems is software the directly interacts with hardware. The OS is trusted to do this for a few reasons:
  - To prevent users from breaking things
  - To abstract away messy details about hardware devices into standardized and more portable/convenient interface
    - Think of how there are many types of keyboards, computer mice, network cards, hard drive types etc. OS abstracts away these details
  - Manages (allocates, schedules, protects) hardware resources
    - Modern computers will have more than one program running, how are resources (files, screen display, etc) shared across these programs?

Memory Protection

- Memory is protected in a few ways:
  - You cannot execute anything stored in data memory
  - You cannot access any OS memory if the program is not in privileged or "super user" mode
  - Privilege is only modified when entering/leaving the OS
    - TRAP/RTI



Next Lecture:

- I/O in LC4
- OS & System Calls in LC4