If you haven't done it yet, please fill out the course policies form!

https://forms.cloud.microsoft/r/Ac0JcTKYuK

The Hardware-Software Interface

No lab tonight!

Discussion Policy Summary

- No “requirement” to speak, especially if you’re having a bad day
- Leaving room for others to participate by being conscious of how much you’ve participated
- Electronic use during larger discussions should be non-destructive
- Be willing to be wrong
- In smaller groups, be sure chairs and bodies are positioned to include everyone in the conversation (especially for people behind you!)
- Knowing each other’s names

AI Policy Summary

- More variance in philosophy of what should be allowed… response ranges from totally disallowed to unlimited use
- Common feedback that the language around debugging is overly restrictive
  - Policy on the course website has softened the language
- One of my goals for this course is to serve as a technical practicum in a controlled environment where you are allowed to fail
  - Using these tools for development on assignments will remain against course policy
  - I am interested in your experience using these tools for some tasks, so I am inclined to allow it for non-assignment warmup labs

Image credit: https://www.ibm.com/history/650

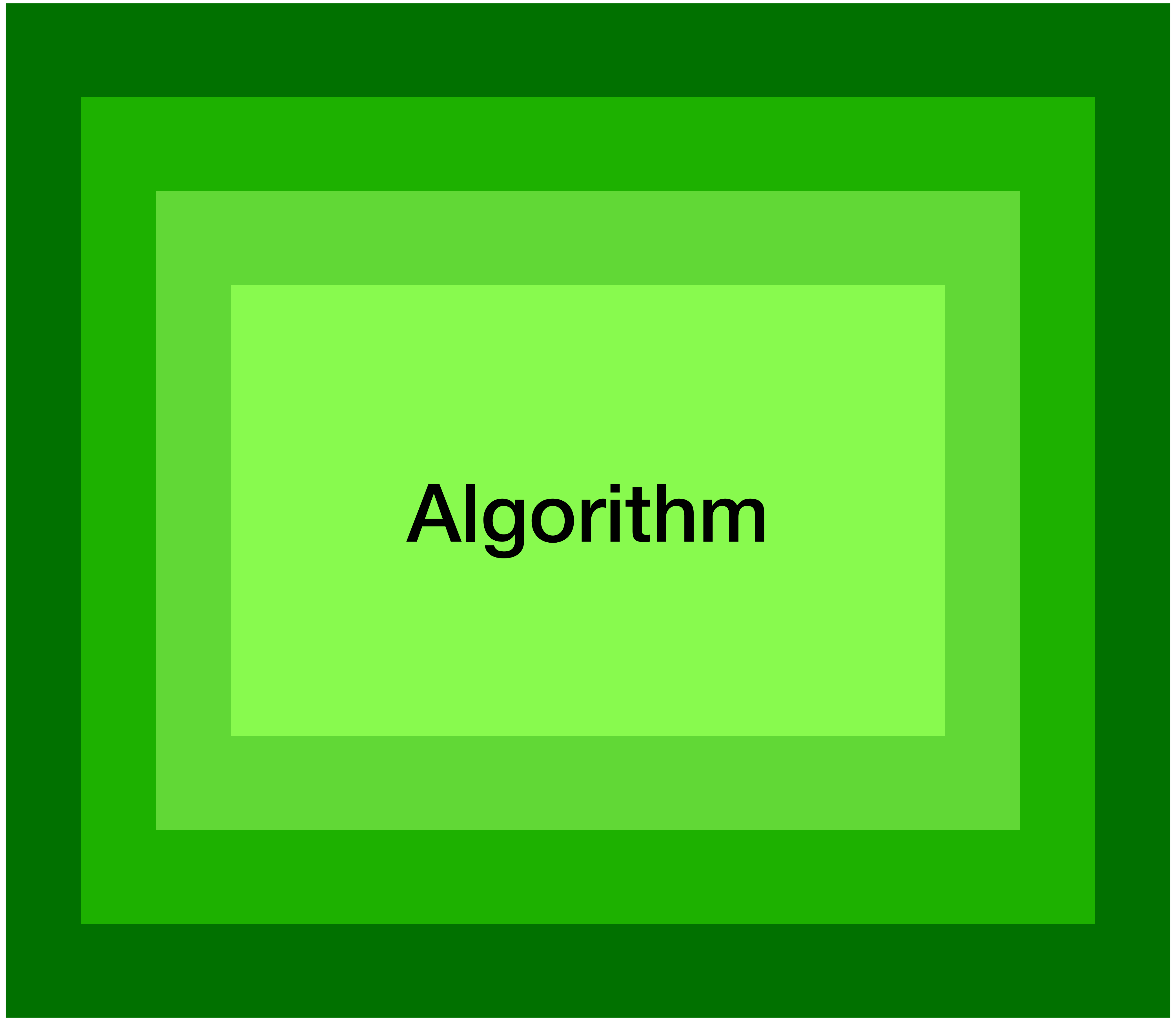
Today's Outline

- Software interfaces to hardware
- Introducing instruction sets
- Functions of a processor
- Introducing basic processor components

Goal: build an intuition for the computer architecture landscape

Unpacking a Program

Software



Unpacking a Program

Software

Algorithm

Can be expressed in pseudocode, unrestricted syntax

Programming Language

Lots of packages, syntax tricks that lead to expansion, language runtimes

System Software

Operating system, system libraries, compiler flags, etc all define the sequences of instruction to execute

Assembly

The relationship between the program instructions and what the processor knows how to run

System Software

We will explore these in more detail on Friday!

- Compilers transform *source code* to *assembly*
- Different compilers compile code down to different assembly formats…

x86

ARM

Power

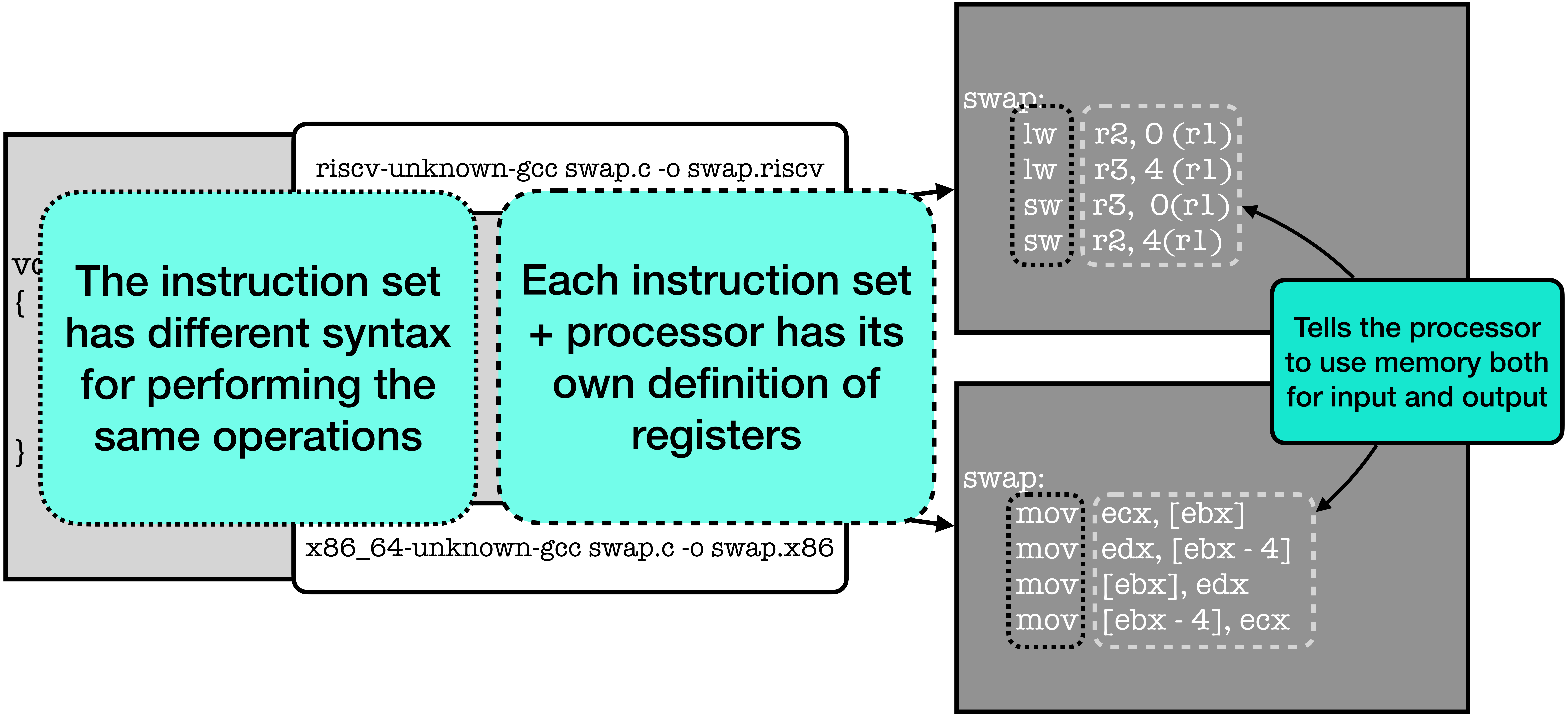
RISC-V

MIPS

- Demo!

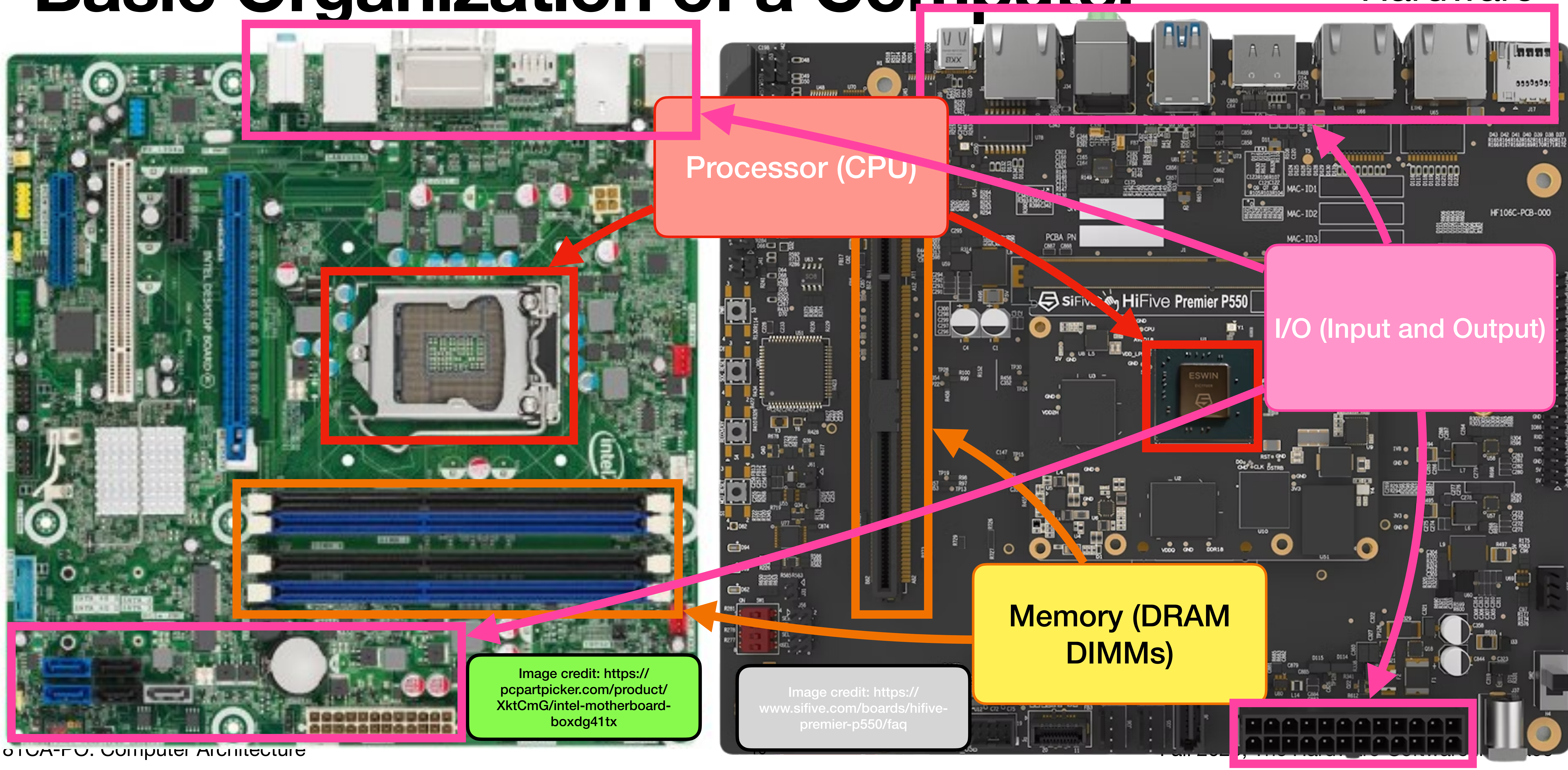
Unpacking a Program (cont.)

Software



Basic Organization of a Computer

Hardware



Basic Organization of a Computer

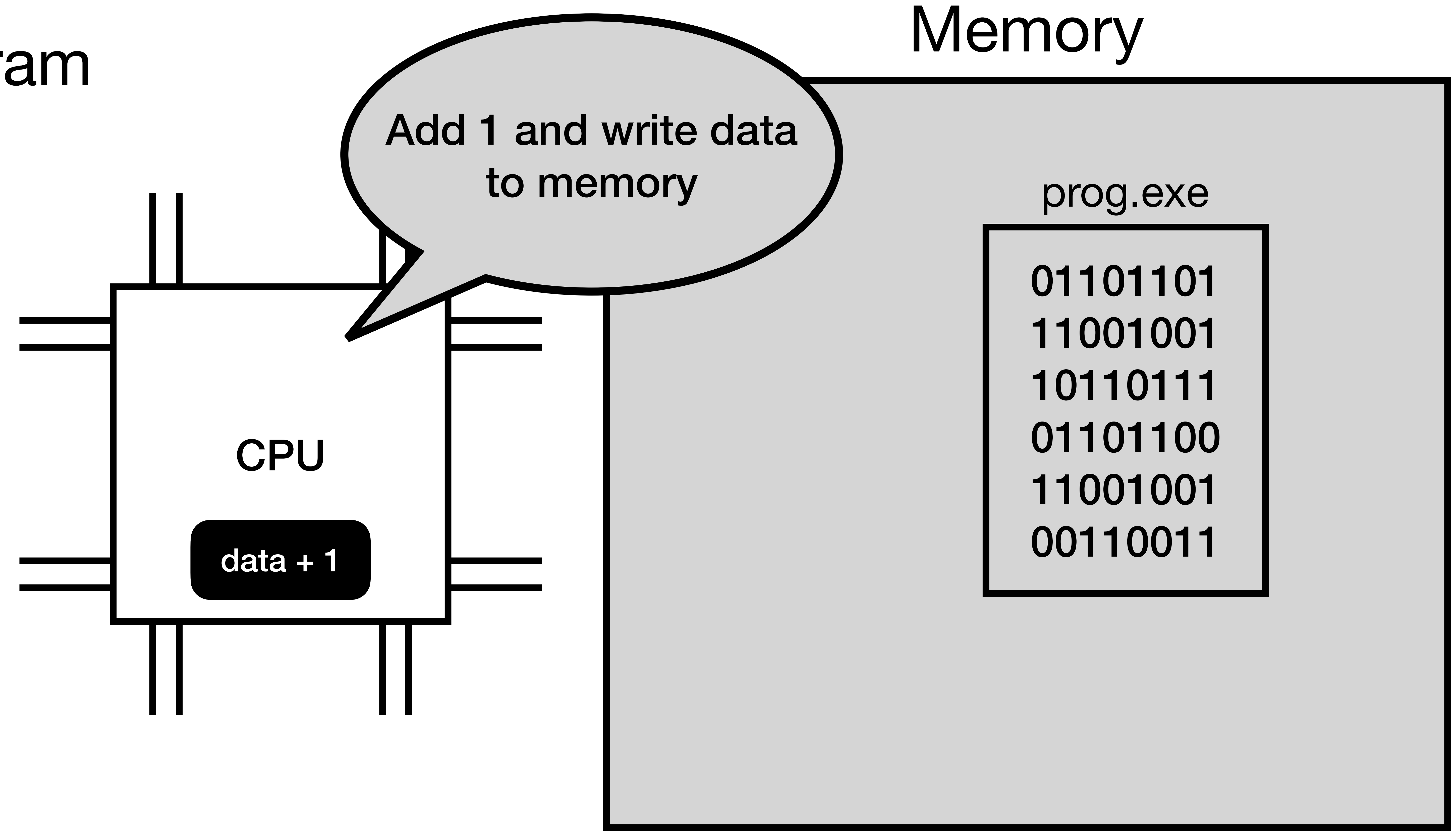
Hardware

- Processor: the processor executes programs by implementing the *data path* and *control* that defines how instructions should be executed
  - Data Path: how data flows through a processor
  - Control: the configuration of custom logic in processor components
- Memory: *load* and *store* interface between the processor and memory for data that doesn't fit in registers
- I/O: mouse, keyboard, stdin/stdout files, screen, network card, etc.

Basic Functions of a Processor

Hardware

- Track location in the program
- Fetch instructions from memory
- Interpret instructions
- Execute instructions
- Update state



Chat with your neighbor(s)!

Hardware

Think about our two assembly programs. What from the assembly syntax will lead to similarities in the basic processor design? Differences?

load-store architecture

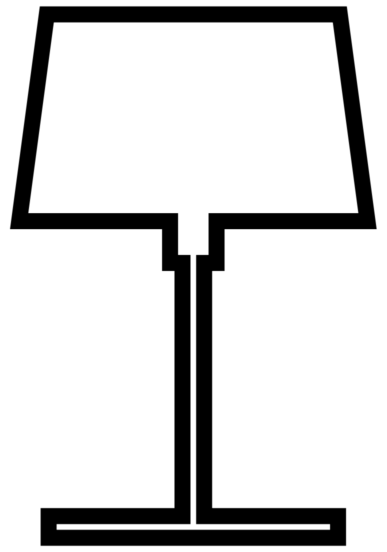
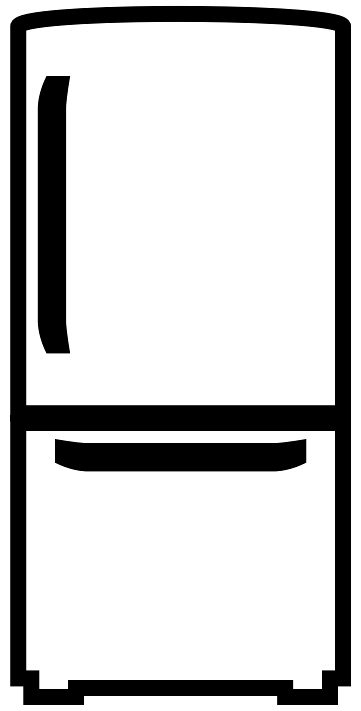
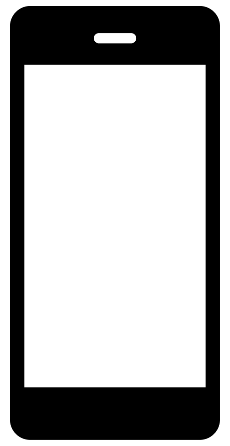
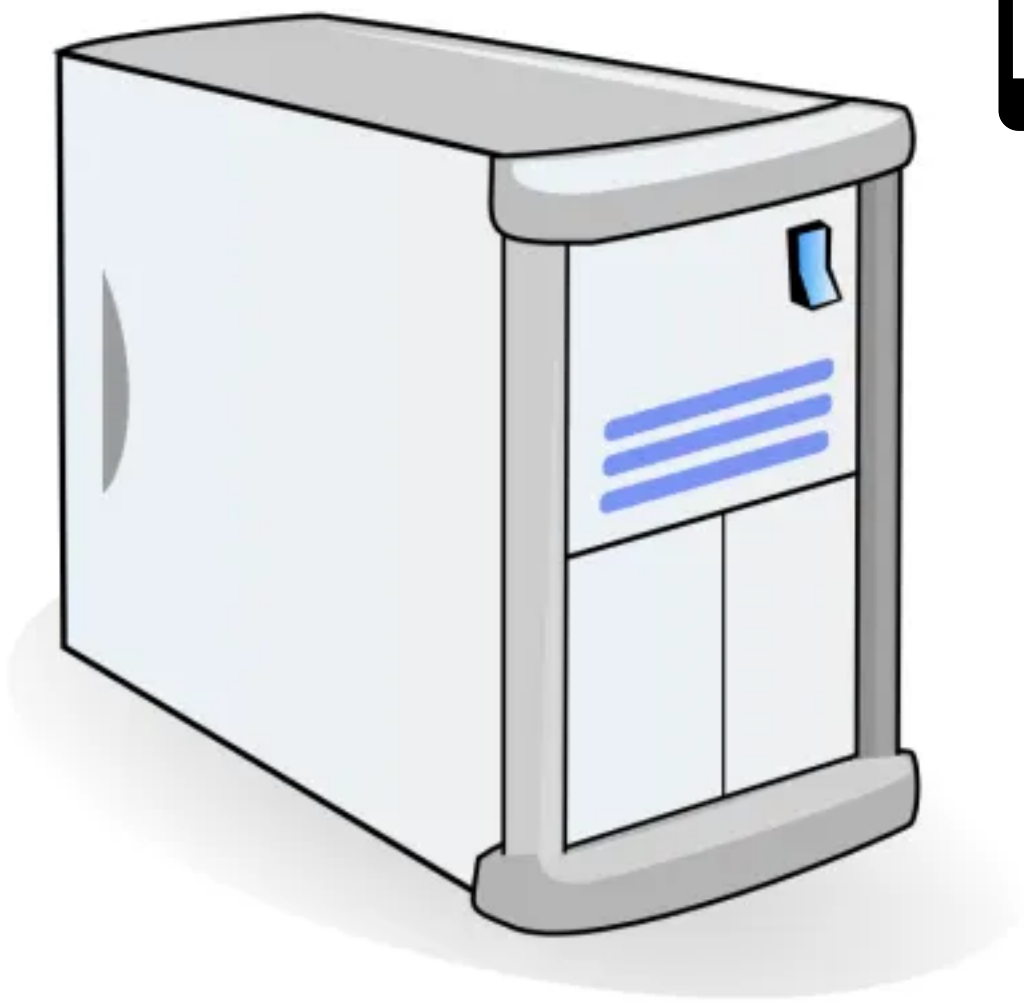
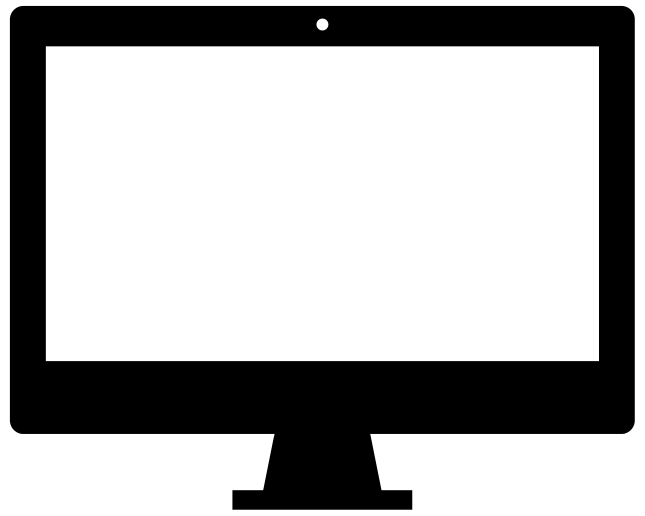
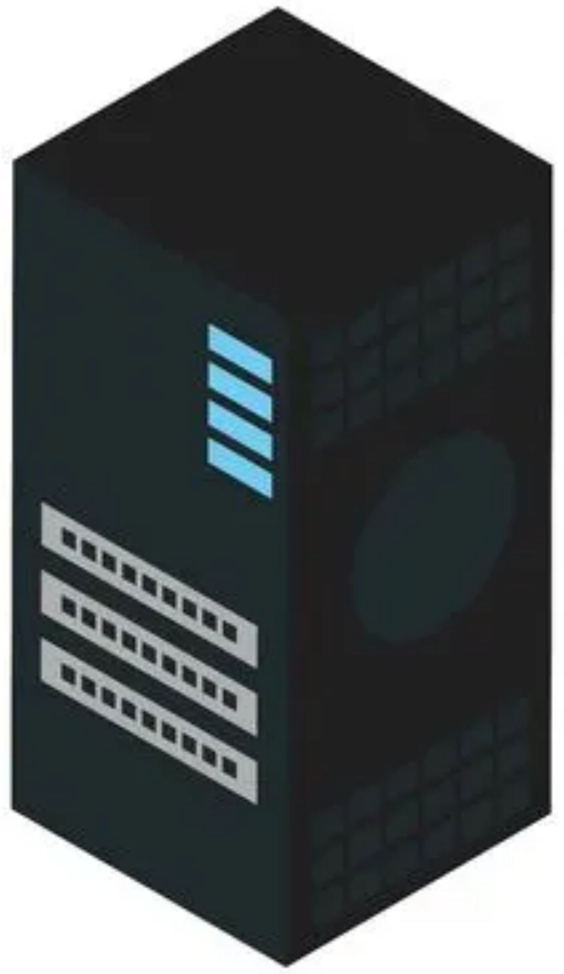
```
swap:
    lw   r2, 0 (r1)
    lw   r3, 4 (r1)
    sw   r3,  0(r1)
    sw   r2, 4(r1)
```

```
swap:
    mov  ecx, [ebx]
    mov  edx, [ebx - 4]
    mov  [ebx], edx
    mov  [ebx - 4], ecx
```

register-memory architecture

Types of Hardware Platforms

- Embedded Devices/Internet of Things (IoT)
- Personal Mobile Devices (PMD)
- Desktop
- Server
- Cluster/Warehouse-Scale

Types of Hardware Platforms

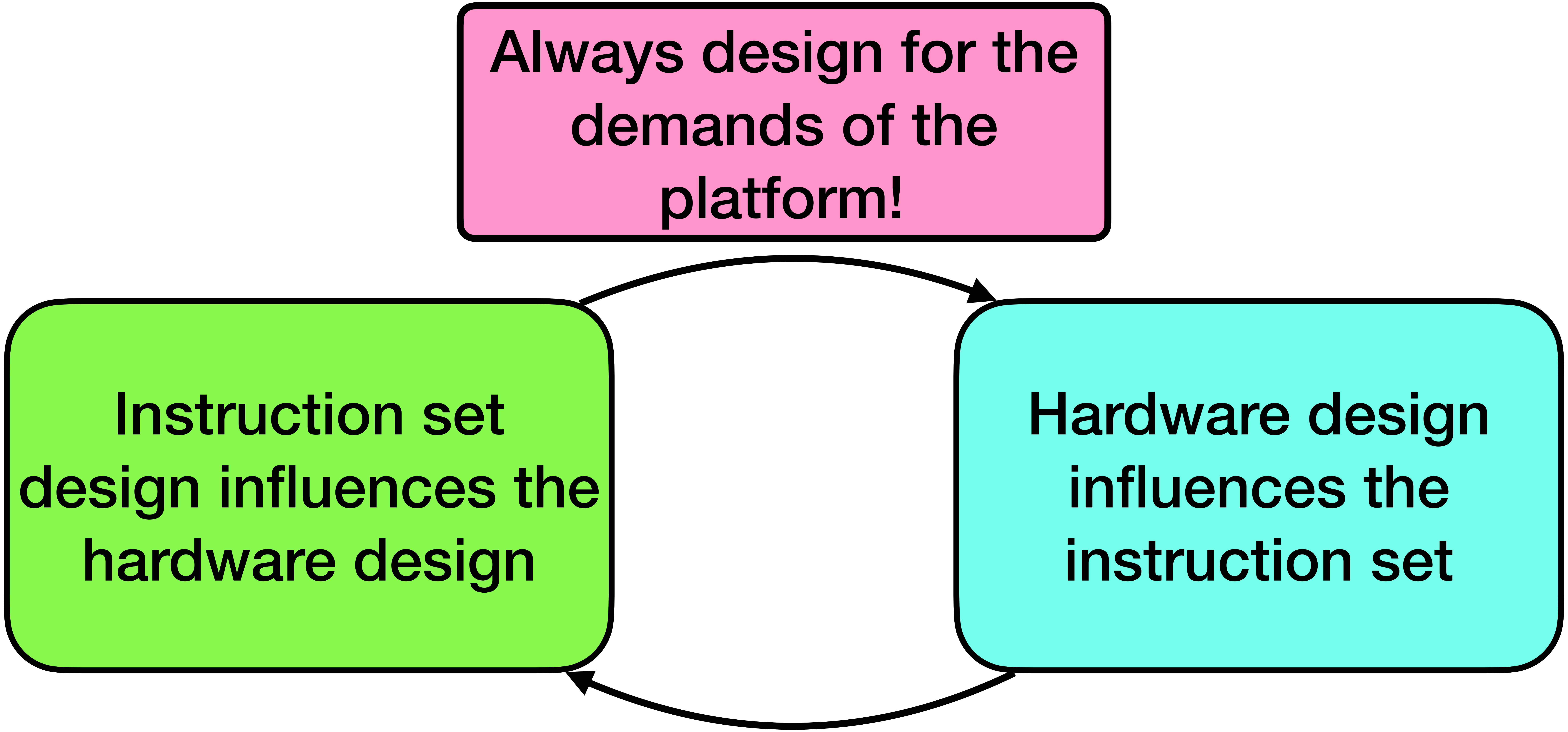
- Embedded Devices/Internet of Things (IoT): cost, energy, specialized application performance
- Personal Mobile Devices (PMD): cost, energy, media performance, responsiveness
- Desktop: combination of price and performance, energy, graphics performance
- Server: throughput, availability, energy, scalability
- Cluster/Warehouse-Scale: throughput, combination of price and performance, energy proportionality

RISC-V

ARMv8-32, x86_64

ARM, x86

Takeaways


Always design for the demands of the platform!
Instruction set design influences the hardware design
Hardware design influences the instruction set