Computers

CSE 120 Spring 2017

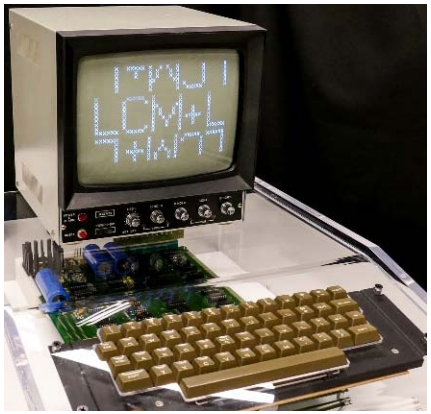
Instructor: Justin Hsia

Teaching Assistants: Anupam Gupta, Braydon Hall, Eugene Oh, Savanna Yee

Steve Jobs' custom Apple I and other historic machines are on display at Seattle museum

In the earliest days of those early days, Steves Wozniak and Jobs made their first device together: the Apple I. Few of these were sold, and fewer still survive — but the Living Computers museum in Seattle managed to get three. And one of them was Jobs' personal machine.

That's the mission of the museum, though: the Apple I, along with dozens of other ancient computers… are deliberately there to be touched and, if not truly understood, at least experienced.



- https://techcrunch.com/2017/04/17/steve-jobs-custom-apple-i-and-other-historic-machines-are-on-display-at-seattle-museum/

Administrivia

- Assignments:
  - Creativity Assignment (4/24)
  - Reading Check 5 (4/27)

- Midterm in class on Wednesday (4/26)
  - Bring 1 sheet of notes (2-sided, letter, handwritten)
  - Fill-in-the-blank(s), short answer questions, multiple choice
    - Questions 1-7 on readings and big ideas (drop 2 lowest scores) [10 pt]
    - Questions 8-10 on computational thinking (code: reading, writing, and debugging) [20 pt]
  - Time is not your friend
    - Short answers should be *short*
    - Skip questions and come back later

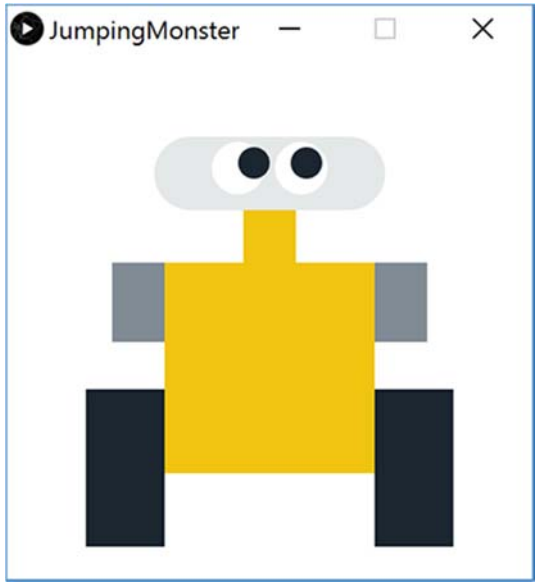
Living Computers Museum Report

- Due Sunday, May 14
  - You have weeks 5-7 to complete
- Required to visit the Living Computers Museum in SODO
  - 2245 1st Avenue South, Seattle, WA 98134
  - Your admission has been paid for! (transportation is not)
  - Lots of hands-on exhibits involving old computers and modern gadgets and other big ideas from CSE120
  - We encourage you to visit in groups!
- Worksheet to complete after visit
  - Involves some photos, a little bit of research, and some written responses

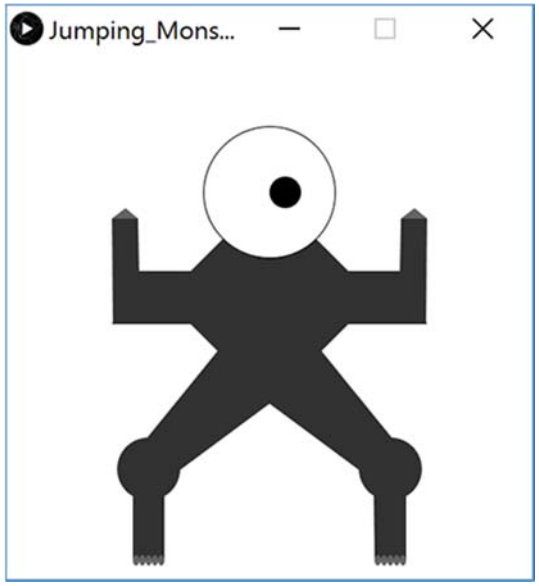
Outline

- **Student Work Showcase**
- Computer Components
- Computer Instructions
- Instruction Execution

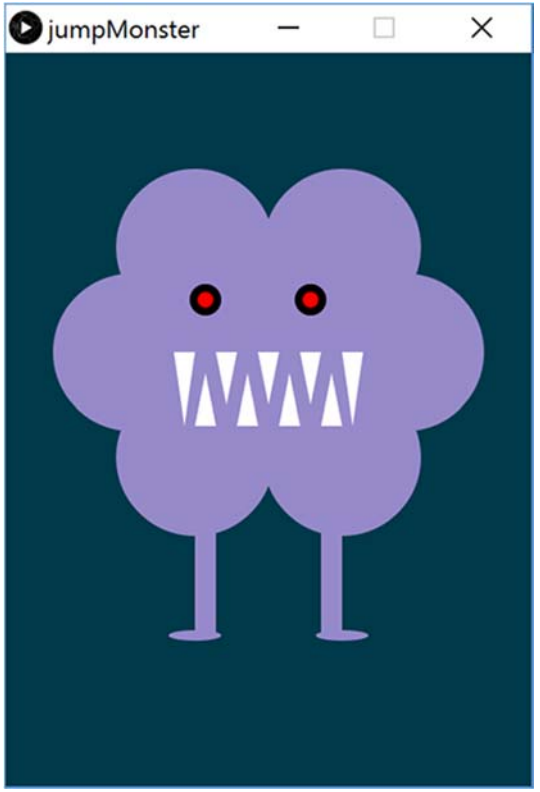
Jumping Monster



Aliakesi Zhuranko



Jack Cummings



Outline

- Student Work Showcase
- **Computer Components**
- Computer Instructions
- Instruction Execution

Computers are Complex

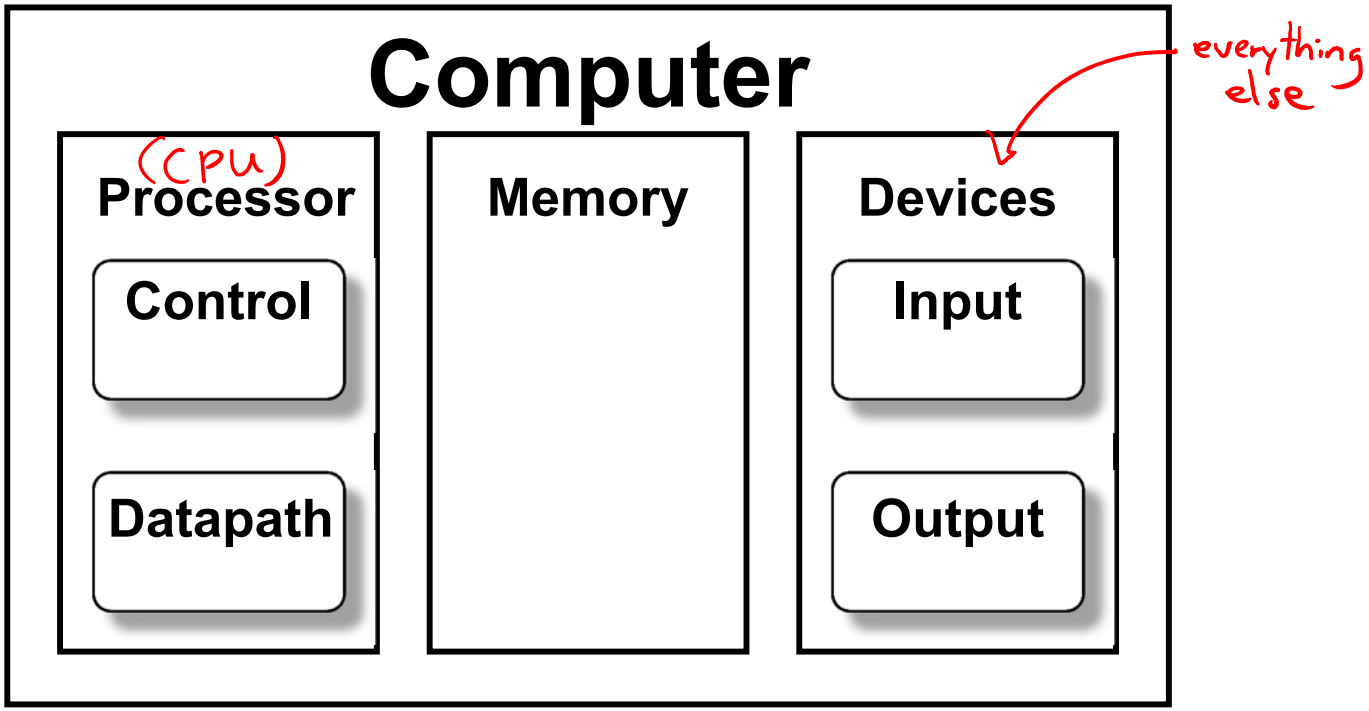
http://xkcd.com/676/

Audience Responses

- What are different computer components you've heard of?
  - Hard drive
  - RAM (memory)
  - CPU (processor)
  - Monitor
  - Graphics Card
  - Motherboard
  - Keyboard
  - Mouse
  - Speakers
  - Etc…

Five Components of a Computer

- Control
- Datapath
- Memory
- Input
- Output



Input

- Devices that send information *to* the computer
- Examples we've seen:
  - Mouse
  - Keyboard
  - Ethernet cable/wireless card
- Other examples:
  - Microphone
  - External sensors
  - Disk (read)

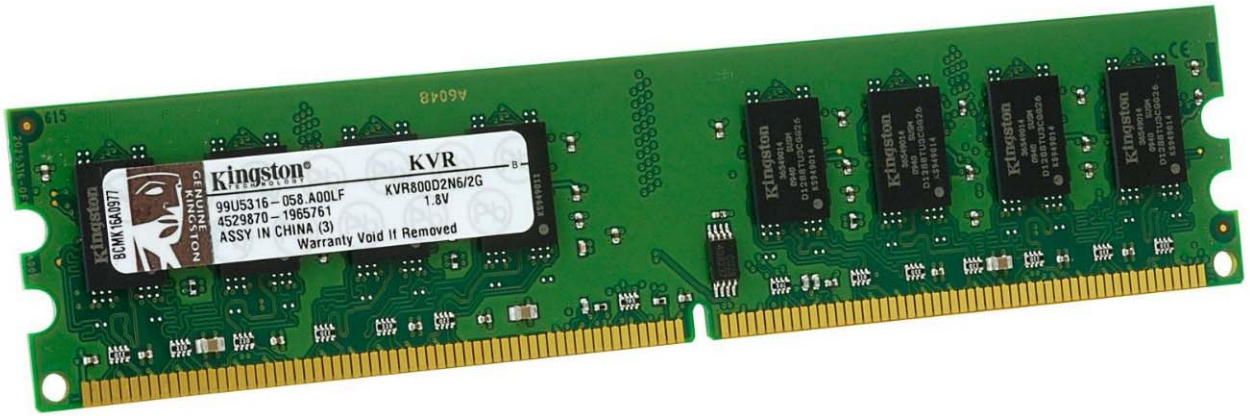
Output

- Devices that the computer sends information *to*
- Examples we've seen:
  - Monitor/graphics card
  - Ethernet cable/wireless card
- Other examples:
  - Speakers
  - Printers
  - Disk (write)

Memory

- Place for *temporary* data storage
  - Typically random access memory (RAM)
  - Data is lost when the computer loses power!
  - Permanent storage goes to disk

- Needed by programs as they run
  - Running out of available memory is typically what causes your computer to slow down

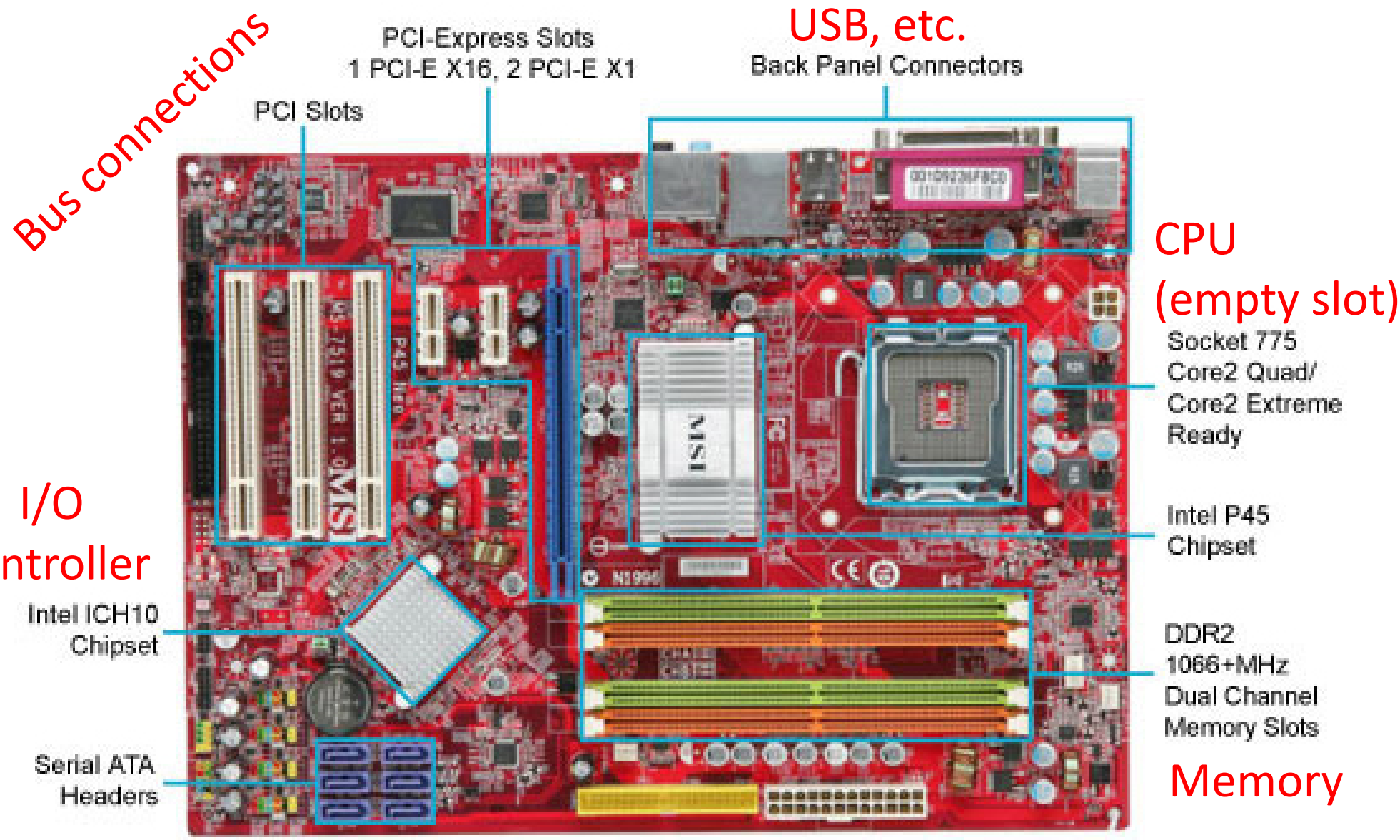


Central Processing Unit (CPU)

- The "brain" of the computer – the circuitry that carries out the instructions
  - *Datapath* – all of the hardware components needed to handle all possible instructions
  - *Control* – the routing (decision-making) that determines correct instruction execution

- Confusingly, modern CPUs contain multiple processors that can each run multiple processes simultaneously
  - More on this in a later lecture

Computer Internals: Physical View

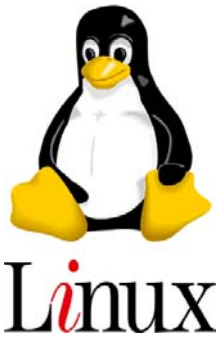


The Operating System

- Everything mentioned up to this point was *hardware*

- The piece of *software* that ties everything together and coordinates between hardware and software is the operating system (OS)
  - It's a program, just like everything else, but more important
  - The OS has special privileges
  - The OS is heavily responsible for the security of your computer (programs and data)

Ls you should update regularly!



Outline

- Student Work Showcase
- Computer Components
- **Computer Instructions**
- Instruction Execution

What Does an Instruction Look Like?

- Just like all other data on a computer, instructions are also just binary data!

binary → 0100 1000 0000 0001 1111 1110

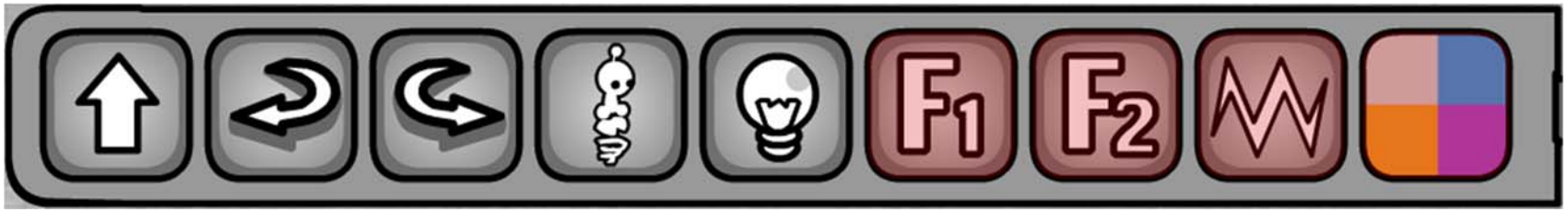
- Example: 0x4801FE tells my computer to add one number to another
  - Similar to `x = x + y;`

hex →

- An executable (program) contains the binary encoding of all of its instructions and data
  - Plus some other information
  - Hex view demo [see Panopto]

Limited Instructions

- The number and type of instructions is always limited



  - The agent can only do certain pre-defined actions

- In a computer, this is determined by the *Instruction Set Architecture* (ISA)
  - The CPU and other hardware is designed to execute *only* these instructions

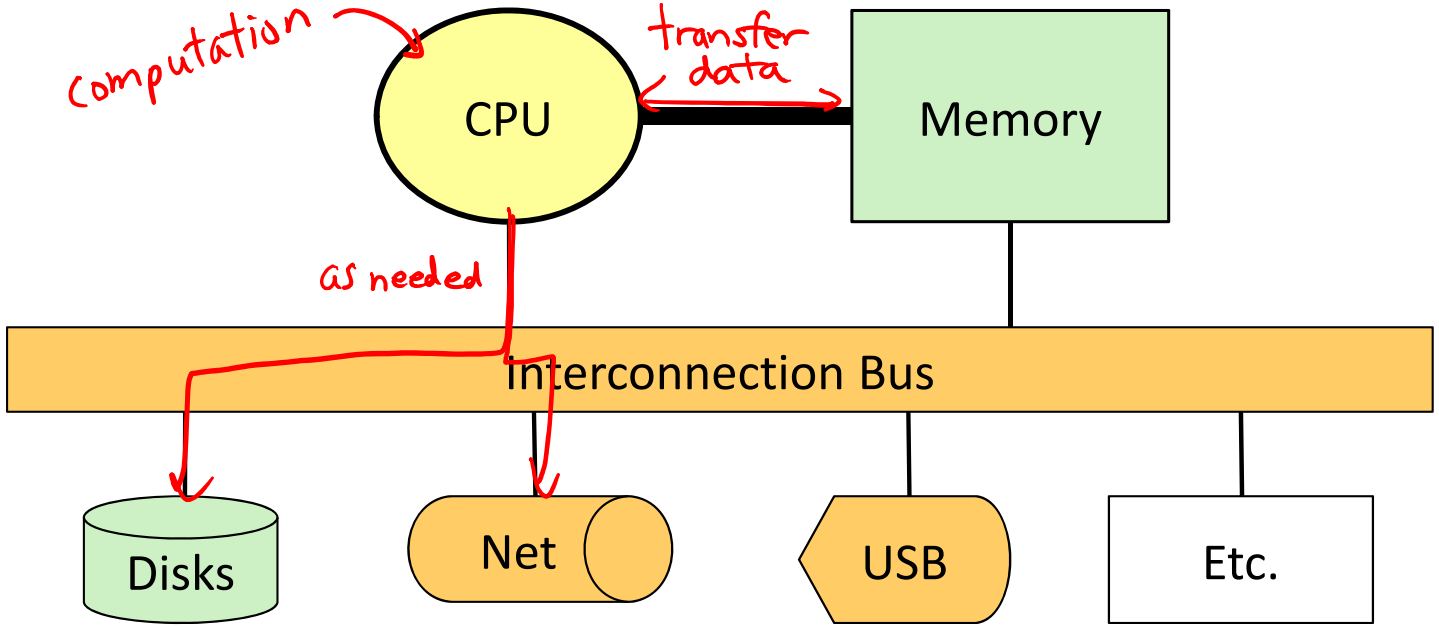
Types of Instructions

1) Perform arithmetic operation in the CPU
   - `c = a + b;` `z = x * y;` `i = h && g;`
2) Control flow: what instruction to execute next
   - Normally whatever instruction comes next in sequence
   - *Jump* to function calls
   - *Possibly jump* on conditional branches
   - *Possibly jump* for loops
3) Transfer data between the CPU and memory
   - *Load* data from memory into CPU
   - *Store* CPU data into memory

How Can We Get Away With This?

- Most computation handled by CPU, but it can only hold a small amount of information at a time
  - For reasons of cost and speed

- As needed, transfer data between CPU and Memory
  - As an added trick, treat all Input and Output "like memory"

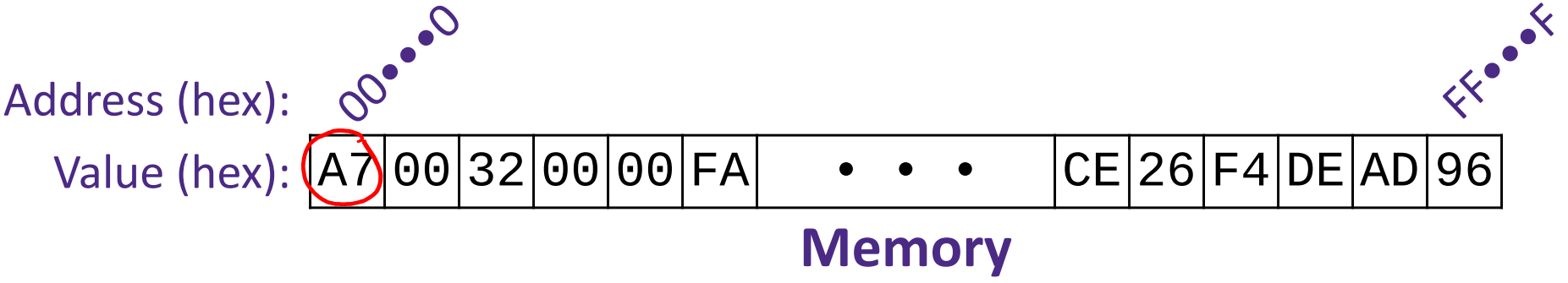


Memory

- We can think of memory as a single, massive array
  - Every memory entry is the same size (1 byte)
  - Every memory entry has an index (memory address) and a value (*data*)

Mem[0] returns 0xA7
index value

- If our computer instructions need to use data that is stored in memory, it can reference it by using the correct memory address

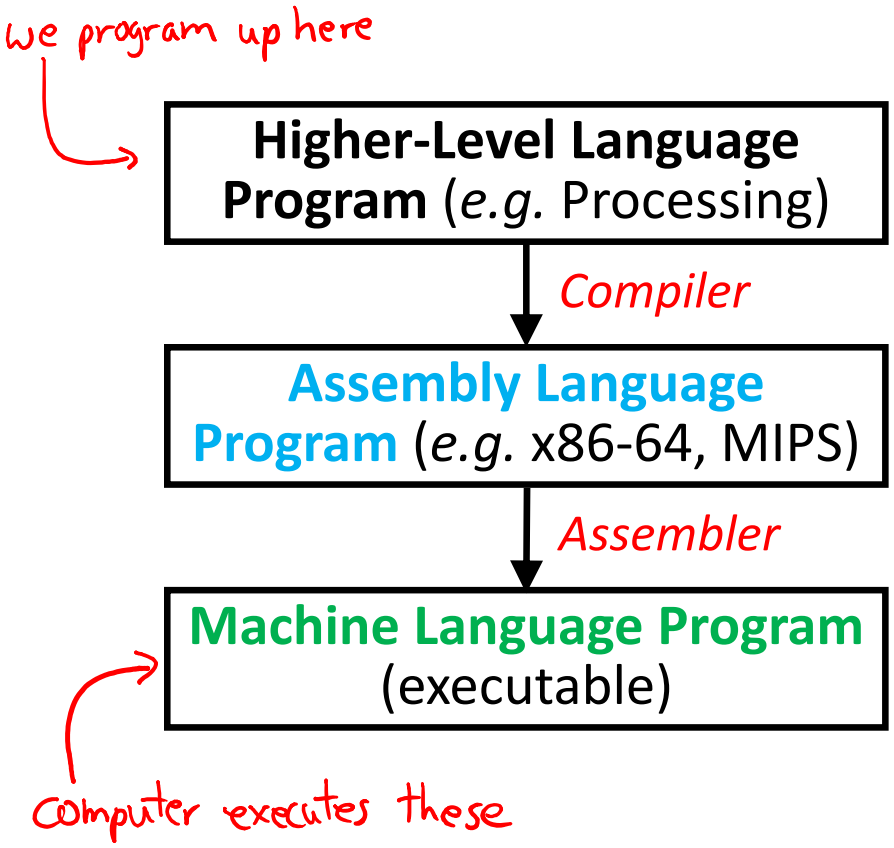


Where are the Instructions?

- When a program is running, the instructions for that program are stored in *memory*
  - Faster to read instructions from memory than from executable file on disk
- The CPU reads instructions for execution in the exact same way that it reads data from memory
  - Take advantage of existing hardware

Generating Instructions

- Must learn how to specify complex tasks using just these simple actions
  - In reality, this is often done automatically for us:



these are all equivalent!

```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

swap data

```
lw   $t0, 0($2)
lw   $t1, 4($2)
sw   $t1, 0($2)
sw   $t0, 4($2)
```

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```

Outline

- Student Work Showcase
- Computer Components
- Computer Instructions
- **Instruction Execution**

Instruction Execution

- The agent follows the instructions *flawlessly* and *mindlessly*
  - A computer will follow the instructions given by a program and should produce identical results if given identical inputs

- The CPU knows where to find the next instruction using the program counter (PC)
  - PC contains the address of the next instruction in memory
  - PC gets updated after each instruction is executed, sometimes jumping around based on the program's *control flow*

Fetch-Execute Cycle

- The most basic operation of a computer is to continually perform the following cycle:
  1) Fetch the next instruction (read from memory)
  2) Execute the instruction based on its purpose and specified data

- Furthermore, the Execution portion can be broken down into the following rough steps:
  1) Instruction decode – what instruction is this?
  2) Data fetch – get data operands %rdi and %rsi in this case
  3) Instruction computation – addition in this case
  4) Store result

add two things together

```
Machine:  0x4801FE   fetched
Assembly: add %rdi, %rsi
```

Arithmetic and Logic Unit (ALU)

- A special logic block that is part of the CPU and performs the arithmetic operations
  - Performs the operations needed to cover *all* the instructions that the CPU can understand
  - Typically limited to two operands, which restricts the overall instruction set

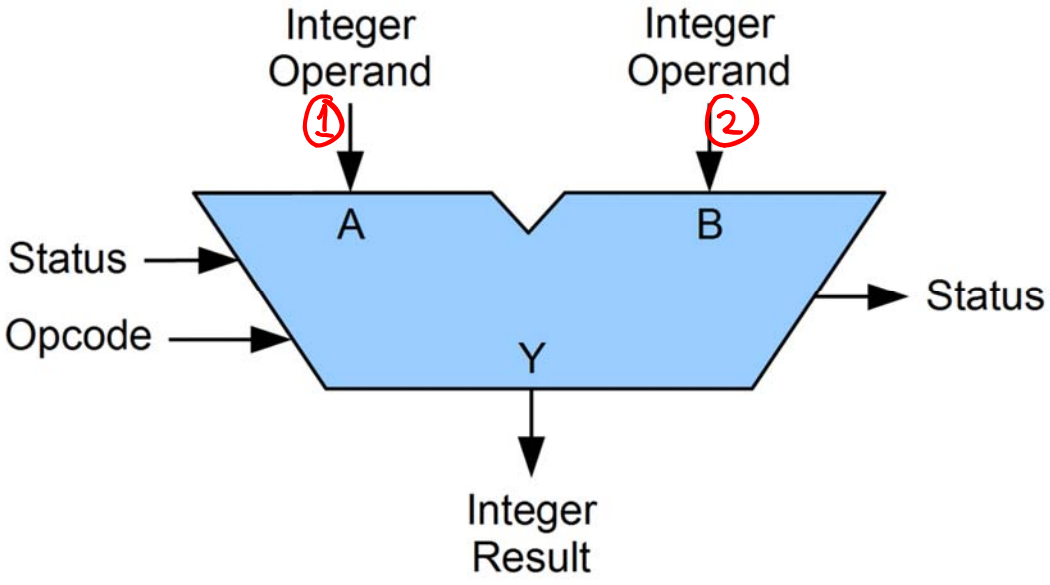
can't do three-operand addition:

```
X = X+Y+Z;
```

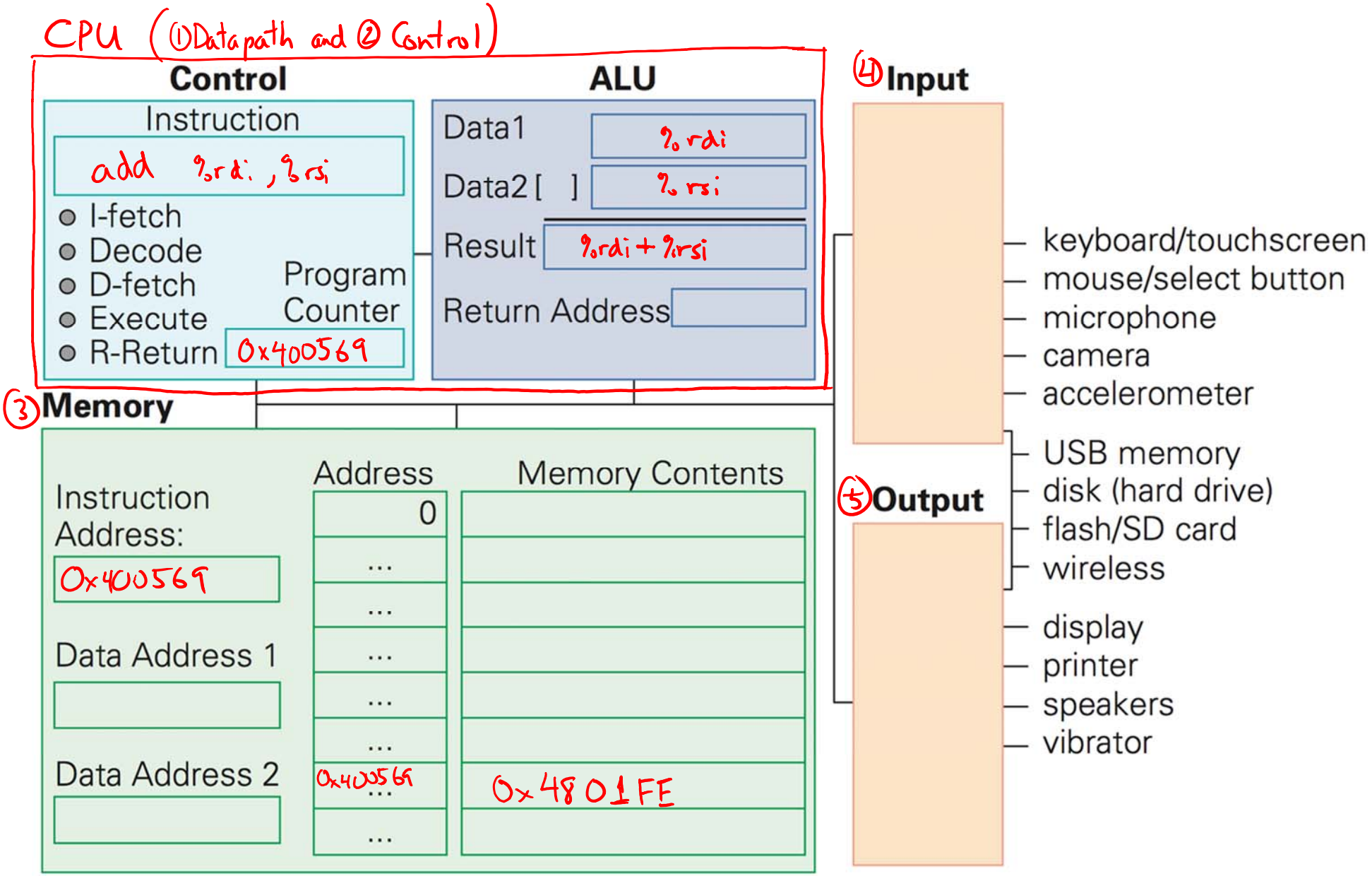
↓ actually becomes

```
X = X+Y;
X = X+Z;
```

} same result, but two instructions



Computer Components Revisited



Clock Rate

- The rate at which your CPU can perform a Fetch-Execute cycle
  - Must make sure that clock speed is slow enough to accommodate the *slowest* instruction

- Clock rate is usually given in Hertz (Hz = second^{-1})
  - Example: 2 GHz = $2 \times 10^9\ s^{-1}$ $\leftrightarrow$ one instruction every 5×10^{-10} s = 500 ps
  - Clock rate is *not* a good indicator of speed
    - CPU must often wait for data from memory or I/O devices

Example: Running a Processing Program

1) The Processing environment compiles your code into machine language
   - Your Processing code gets converted into *machine code* (just 0's and 1's)

2) When you run the program, memory is allocated for the program's instructions, variables, and data

3) Starting from the beginning (`setup()`, in this case), the computer will perform the Fetch-Execute cycle
   - Keeps going unless end of program is reached or an error is encountered before then