

# Building to a Computer

## Fetch, Decode, Execute

Welcome!

CS 2130: Computer Systems and Organization 1  
September 19, 2025

# Announcements

- Quiz 3 available today, due Sunday by 11:59pm
- Homework 2 due Monday

# Memory and Storage

## Registers

$\approx$  KiB

- 6 gates each,  $\approx$  24 transistors
- Efficient, fast
- Expensive!
- Ex: local variables

*These do not persist between power cycles*

# Memory and Storage

## Memory

≈ GiB

- Two main types: SRAM, DRAM
- DRAM: 1 transistor, 1 capacitor per bit
- DRAM is cheaper, simpler to build
- Ex: data structures, local variables

*These do not persist between power cycles*

# Memory and Storage

## Disk

≈ GiB-TiB

- Two main types: flash (solid state), magnetic disk
- Magnetic drive
  - Platter with physical arm above and below
  - Cheap to build
  - Very slow! Physically move arm while disk spins
- Ex: files



*Data on disk does persist between power cycles*

# Putting it all together

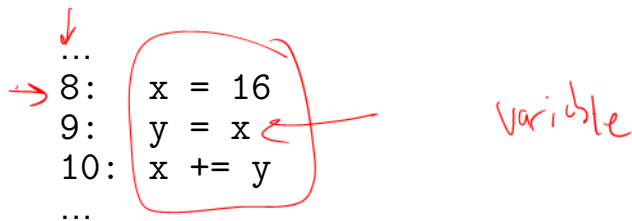
# Our story so far

- Information modeled by voltage through wires (1 vs 0)
- Transistors
- Gates:             $\&$              $|$              $\sim$              $\wedge$
- Multi-bit values: representing integers, floating point numbers
- Multi-bit operations using circuits
- Storing results using registers, clocks
- Memory

# Code

How do we run code? What do we need?

Consider the following code:



```
...  
8:  x = 16  
9:  y = x  
10: x += y  
...
```

variable

What is the value of  $x$  after line 10?



# Bookkeeping

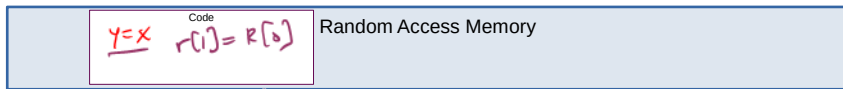
What do we need to keep track of?

- **Code** - the program we are running
  - RAM (Random Access Memory)
- **State** - things that may change value (i.e., variables)
  - Register file - can read and write values each cycle
- **Program Counter (PC)** - where we are in our code
  - Single register - byte number in memory for next instruction

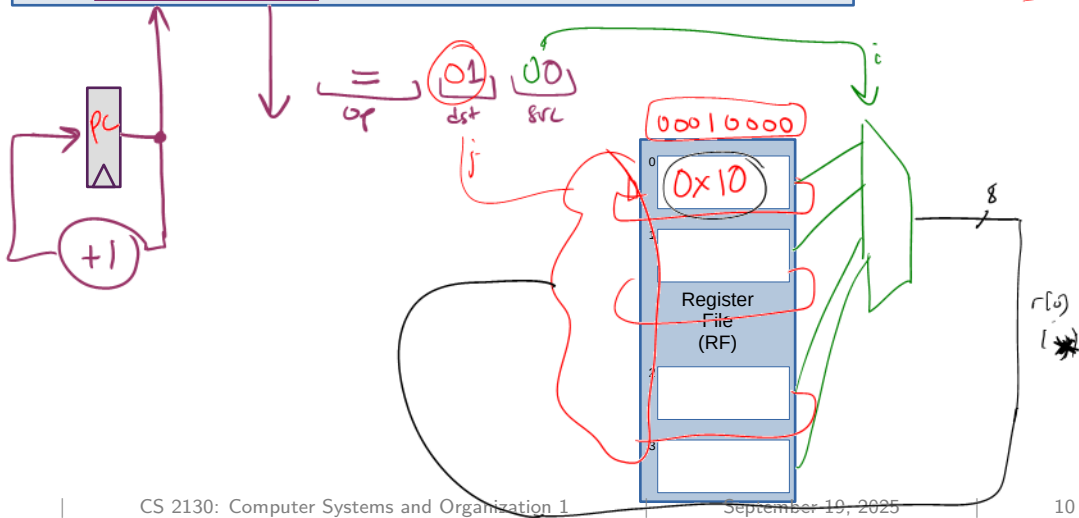


# Building a Computer

$y = x$



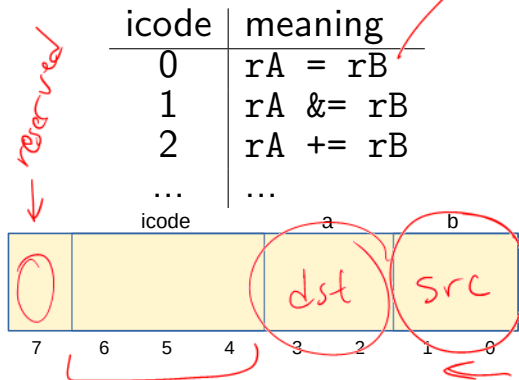
$y \rightarrow R[1]$   
 $x \rightarrow R[0]$



# Encoding Instructions

## Encoding of Instructions (**icode** or **opcode**)

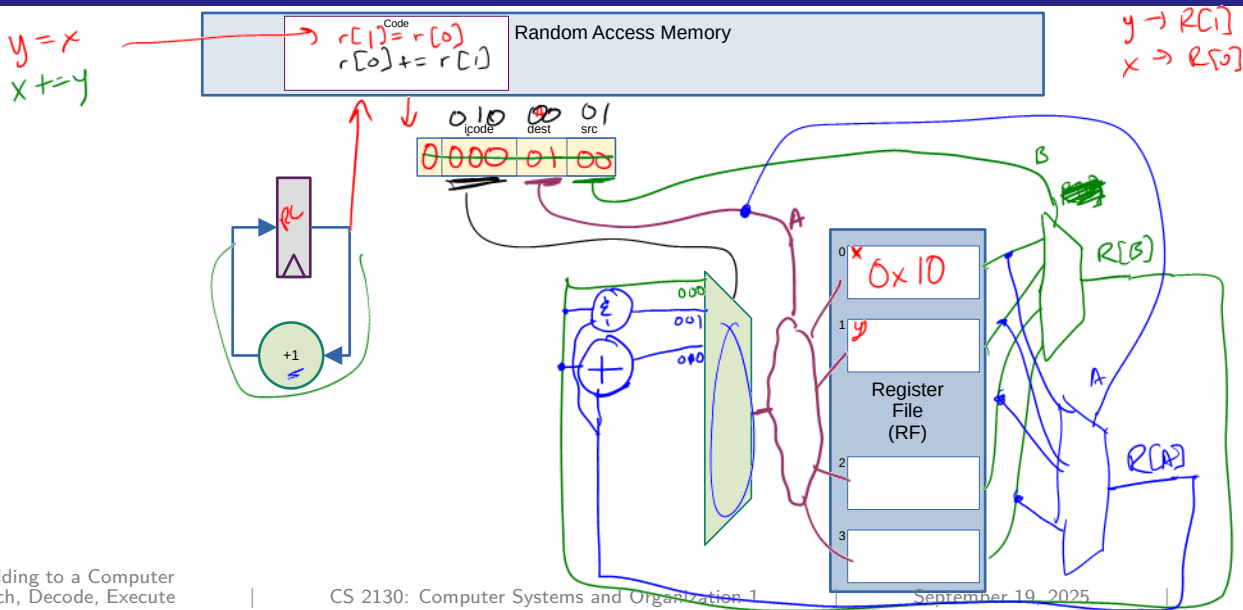
- Numeric mapping from icode to operation



~~$z = x + y$~~   $x + = y$   
 $R[0] = R[1]$

8 possible icode

# Building a Computer





# Question

What happens if we get the 0-byte instruction? 00

# Our Computer's Instructions

## Toy ISA 3-bit icode

icode	meaning
0	$rA = rB$
1	$rA \&= rB$
2	$rA += rB$
...	...
4	$rA = \text{read from memory at address } rB$
5	write $rA$ to memory at address $rB$
...	...
7	Compare $rA$ as 8-bit 2's-complement to 0 if $rA \leq 0$ set $pc = rB$ else increment $pc$ as normal

# Our Computer's Instructions

## Toy ISA 3-bit icode

icode	b	action
3	0	$rA = \sim rA$
	1	$rA = !rA$
	2	$rA = -rA$
	3	$rA = pc$
6	0	$rA = \text{read from memory at } pc + 1$
	1	$rA \&= \text{read from memory at } pc + 1$
	2	$rA += \text{read from memory at } pc + 1$
	3	$rA = \text{read from memory at the address stored at } pc + 1$
		For icode 6, increase pc by 2 at end of instruction