

# Toy Instruction Set Architecture

---

## CS 2130: Computer Systems and Organization 1

**Xinyao Yi** Ph.D.  
Assistant Professor

## Announcements

---

- Homework 3 due Monday at 11:59pm on Gradescope
- Midterm 1 next Friday (October 3, 2025) in class
  - Written, closed notes
  - If you have SDAC, please schedule ASAP

## High-level Instructions

---

In general, 3 kinds of instructions

- **moves** - move values around without doing “work”
- **math** - broadly doing “work”
- **jumps** - jump to a new place in the code

## Moves

---

Few forms

- Register to register (icode 0),  $x = y$
- Register to/from memory (icodes 4-5),  $x = M[b]$ ,  $M[b] = x$

Memory

- Address: an index into memory.
  - Addresses are just (large) numbers
  - Usually we will not look at the number and trust it exists and is stored in a register

## Moves

---

icode	b	action
0		$rA = rB$
3	3	$rA = pc$
4		$rA = \text{read from memory at address } rB$
5		$\text{write } rA \text{ to memory at address } rB$
6	0	$rA = \text{read from memory at } pc + 1$
	3	$rA = \text{read from memory at the address stored at } pc + 1$

## Math

---

Broadly doing work

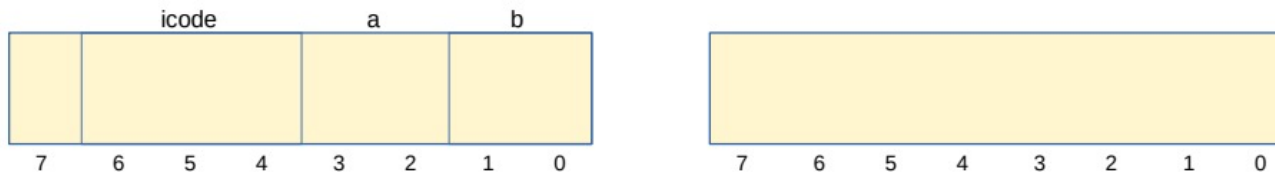
icode	b	meaning
1		$rA \&= rB$
2		$rA += rB$
3	0	$rA = \sim rA$
	1	$rA = !rA$
	2	$rA = -rA$
6	1	$rA \&= \text{read from memory at pc} + 1$
	2	$rA += \text{read from memory at pc} + 1$

*Note: We can implement other operations using these things!*

## Immediate values

icode 6 provides literals, **immediate** values

icode	b	action
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



## Jumps

---

- Moves and math are large portion of our code
- We also need **control constructs**
  - Change what we are going to do next
  - if, while, for, functions, ...
- Jumps provide mechanism to perform these control constructs
- We jump by assigning a new value to the program counter PC



# Jumps

---

- For example, consider an if

## Jumps

---

icode	meaning
7	Compare rA as 8-bit 2's-complement to 0 if $rA \leq 0$ set $pc = rB$ else increment pc as normal

Instruction icode 7 provides a **conditional** jump

- Real code will also provide an **unconditional** jump, but a conditional jump is sufficient

## Writing Code

---

We can now write any\* program!

- When you run code, it is being turned into instructions like ours
- Modern computers use a larger pool of instructions than we have (we will get there)

\*we do have some limitations, since we can only represent 8-bit values and some operations may be tedious.

## Our code to this machine code

---

How do we turn our control constructs into jump statements?

## **if/else to jump**

---

## **while to jump**

---

## Encoding Instructions

icode	b	meaning
0		$rA = rB$
1		$rA \&= rB$
2		$rA += rB$
3	0	$rA = \sim rA$
	1	$rA = !rA$
	2	$rA = -rA$
	3	$rA = pc$
4		$rA = \text{read from memory at address } rB$
5		write $rA$ to memory at address $rB$
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
7		Compare $rA$ as 8-bit 2's-complement to 0 if $rA \leq 0$ set $pc = rB$ else increment $pc$ as normal

Example 3: if  $r0 < 9$  jump to 0x42

## Dealing with Variables and Memory

---

What if we have many variables? Compute:  $x += y$



# Function Calls

---