Toy Instruction Set Architecture

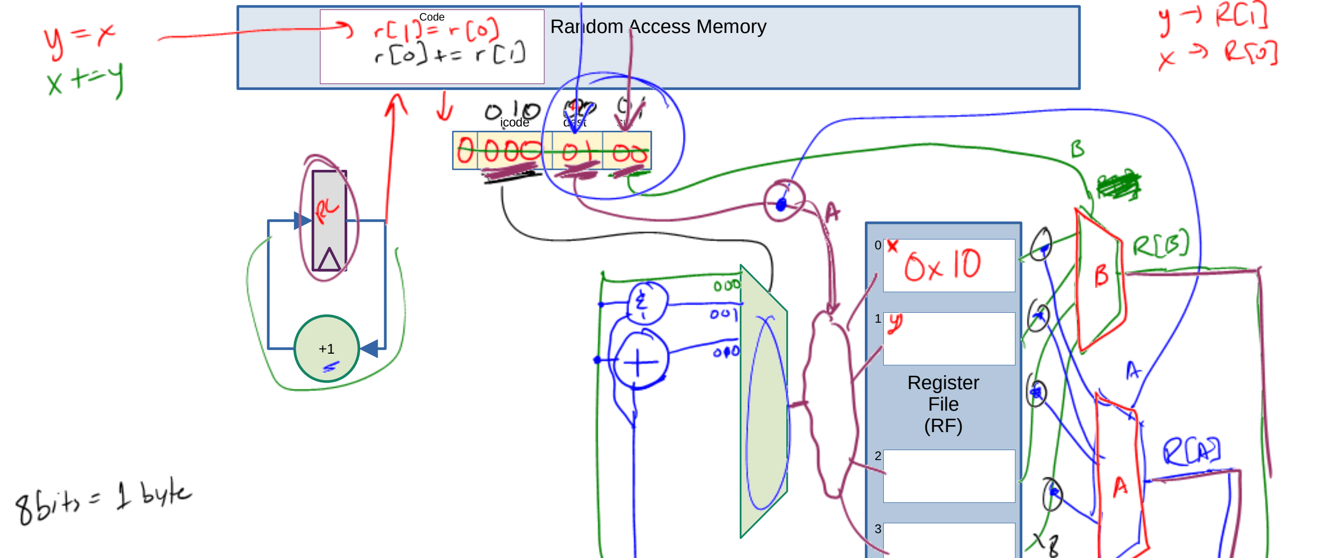
CS 2130: Computer Systems and Organization 1

September 22, 2025

Announcements

- Homework 2 due tonight at 11:59pm on Gradescope
- Homework 3 out today, due next Monday at 11:59pm on Gradescope

Building a Computer



Quiz Review

$x = y + z; \longrightarrow$

$x = y$

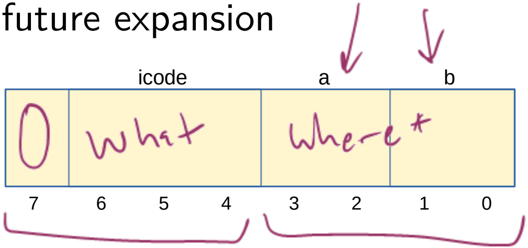
$x \mathrel{+}= z$

$rA \mathrel{+}= rB$

Encoding Instructions

Encoding of Instructions

- 3-bit icode (which operation to perform)
  - Numeric mapping from icode to operation
- Which registers to use (2 bits each)
- Reserved bit for future expansion



Question

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

0x00

0000 0000

0 r[0] r[0]

r[0] = r[0]

noop

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

R[a] = R[b]

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

x = M[R[b]]

M[R[b]] = R[a]

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` = read from memory at address `rB`
5		write `rA` to memory at address `rB`
6	0	`rA` = read from memory at `pc + 1`
	3	`rA` = read from memory at the address stored at `pc + 1`

rA = M[pc+1]

Math

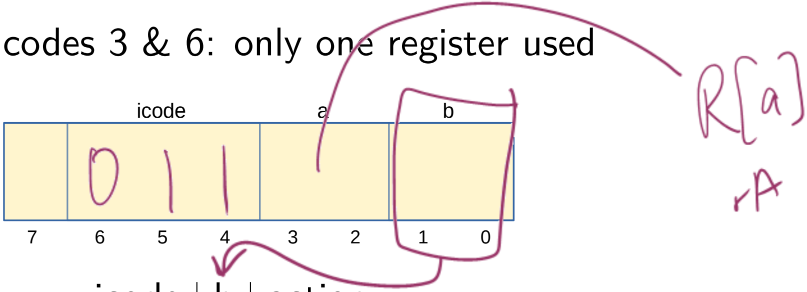
Broadly doing work

icode	b	meaning
1		`rA &= rB`
2		`rA += rB`
3	0	`rA = ~rA`
	1	`rA = !rA`
	2	`rA = -rA`
6	1	`rA` &= read from memory at `pc + 1`
	2	`rA` += read from memory at `pc + 1`

Note: We can implement other operations using these things!

icodes 3 and 6

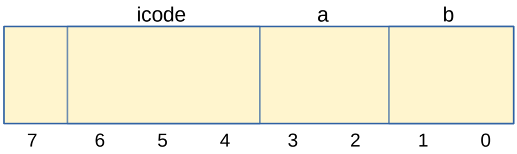
Special property of icodes 3 & 6: only one register used



icode	b	action
3	0	`rA = ~rA`
	1	`rA = !rA`
	2	`rA = -rA`
	3	`rA = pc`

icodes 3 and 6

Special property of 3 & 6: only one register used

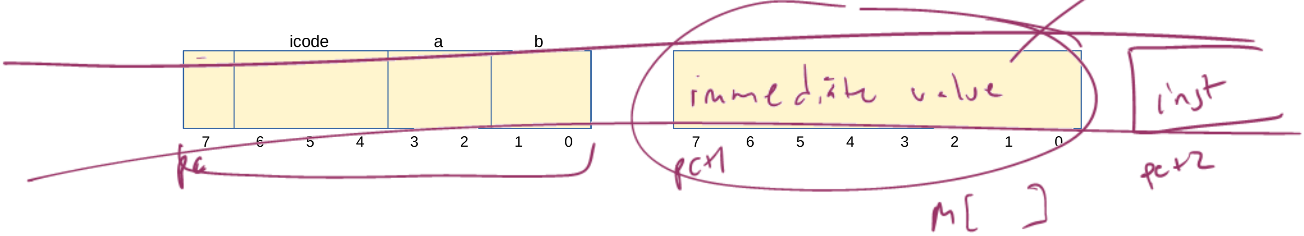


- Side effect: all bytes between 0 and 127 are valid instructions!
- As long as high-order bit is 0
- No syntax errors, any instruction given is valid

Immediate values

icode 6 provides literals, **immediate** values

icode	b	action
6	0	`rA` = read from memory at `pc + 1`
	1	`rA` &= read from memory at `pc + 1`
	2	`rA` += read from memory at `pc + 1`
	3	`rA` = read from memory at the address stored at `pc + 1`
		For icode 6, increase `pc` by 2 at end of instruction



Encoding Instructions

Example 1: `r1 += 19`

Instructions

icode	b	meaning
0		`rA = rB`
1		`rA &= rB`
2		`rA += rB`
3	0	`rA = ~rA`
	1	`rA = !rA`
	2	`rA = -rA`
	3	`rA = pc`
4		`rA` = read from memory at address `rB`
5		write `rA` to memory at address `rB`
6	0	`rA` = read from memory at `pc + 1`
	1	`rA` &= read from memory at `pc + 1`
	2	`rA` += read from memory at `pc + 1`
	3	`rA` = 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 <= 0` set `pc = rB`
		else increment `pc` as normal