Toy Instruction Set Architecture

CS 2130: Computer Systems and Organization 1

Xinyao Yi Ph.D.
Assistant Professor

UNIVERSITY of VIRGINIA | ENGINEERING

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` = 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`

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!

Immediate values

icode 6 provides literals, **immediate** values

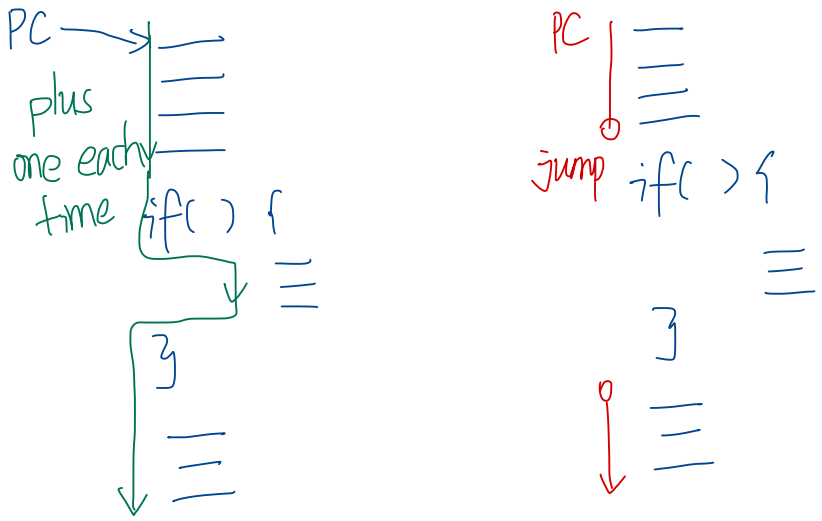
icode	b	action
6	0	`rA` = read from memory at `pc + 1`
	1	`rA` &= read from memory at `pc + 1`
	2	`rA` += read from memory at `pc + 1`
	3	`rA` = read from memory at the address stored at `pc + 1`
		For icode 6, increase `pc` by 2 at end of instruction

Jumps (control constructs)

- 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, ... in terms of machine code, these codes called jumps
- 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



when we got "if", we have a choice

{ ①. continue our code, line by line
② don't want to do the if body, magic teleport, teleports me down to the end of the if statement.

(if the first line after if has index 25, instead of PC+1, I'll say, PC=25)

Jumps

we make all our registers 8 bits, each register will hold one byte.

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

Instruction icode 7 provides a **conditional** jump

jumps if some condition is true. Specifically, we read the value in rA.

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

just set rA to 0.

Writing Code

We can now write any* program!

We are basically being what we called a "compiler"

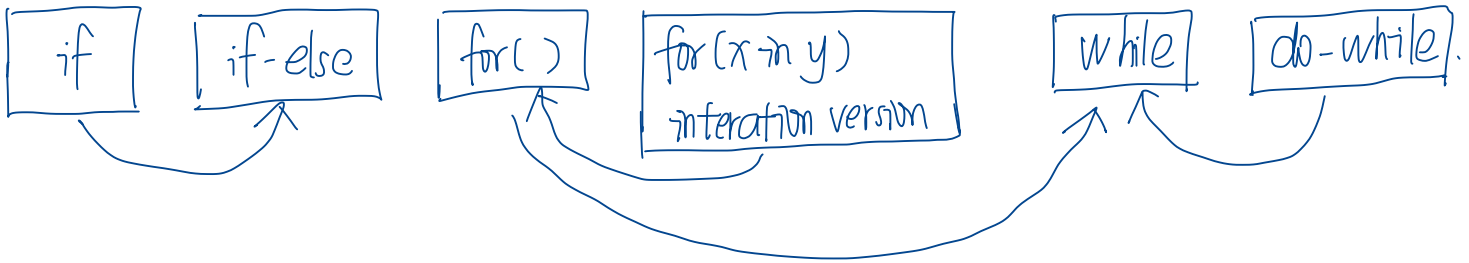
- When you run code, it is being turned into instructions like ours
- Modern computers use a larger pool of instructions than we have (we will get there)

We have 14 instructions, modern computers can have thousands of instructions.

*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?



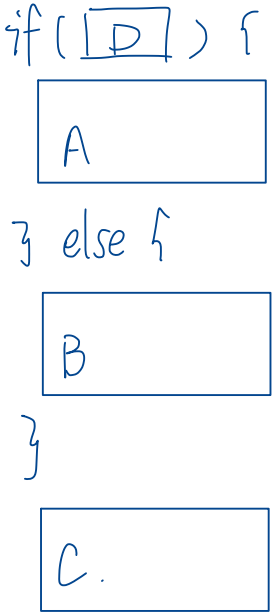
how to convert for to while:

```
for (int i=0; i<25; i++){
    ==
}
```

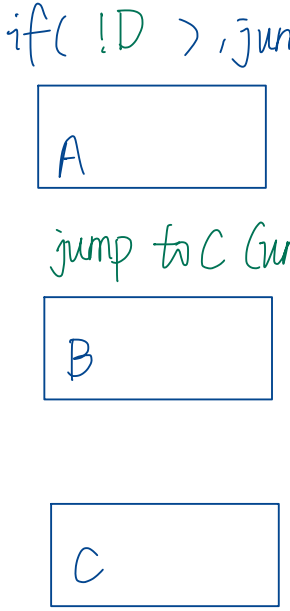
⇒

```
int i=0
while(i<25){
    ==
    i++; }
```

if/else to jump



if condition D is true, I will do A,
skip B, continue to do C



If D is true, no need to jump, continue to A.
So we can think about it from an opposite way, if(!D), jump to B.

2 situations:
① if D is true: don't jump. continue A, then C.
② if D is false: jump to B, then C.

while to jump

while (c) {

A

}

B

How do I know where to jump?
① hard code if you know the address
② icode 3-3

if (!c), jump to B

A

jump to A? No! Infinate loop!

jump to if, do the condition check!

B

each loop iteration has 2 jump checks.

if (!c), jump to B

A

if (c), jump to A.

B

each loop iteration only has 1 jump check.

Encoding 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

Example 3: if r0 < 9 jump to 0x42

I don't have an instruction say r0<9.
I need "r0<=0" for icode 7, what should I do?

r0<9 ⇔ r0<=8 ⇔ (r0-8)<=0
⇔ r0+=-8 (0xF8)
r0<=0

r1 = 0x42 — 0 110 01 00 — 42
6 4 42

r0 += F8 — 0 110 00 10 — F8
6 2 F8

if r0<=0, PC=r1 — 0 111 00 01
(R0) (R1)
7 1

b442b2F871