ISA + Arrays

CS 2130: Computer Systems and Organization 1

September 23, 2022

Announcements

- Homework 3 due Monday at 11pm on Gradescope
  - Please remember that homeworks are **individual** assignments if not stated otherwise on the assignment
  - Your code should be space-separated bytes as hex values
- Exam 1 next Friday (in class)
  - For SDAC accommodations, please schedule a time with their testing center

High-level Instructions

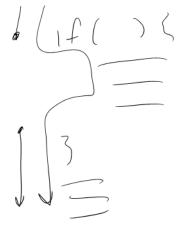In general, 3 kinds of instructions

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

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

Example 3-bit icode

$R[a]$

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

- 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)

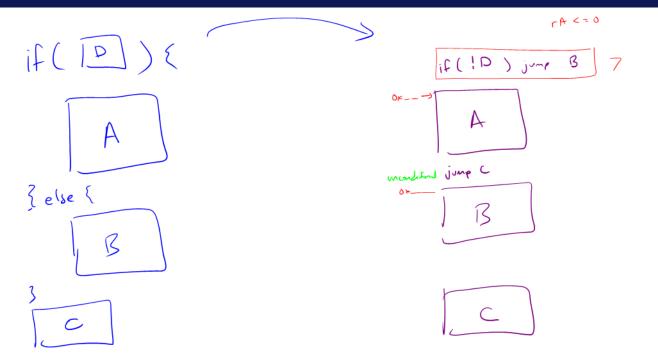*we do have some limitations, since we can only represent 8-bit values and some operations may be tedious.

Our code to this machine code

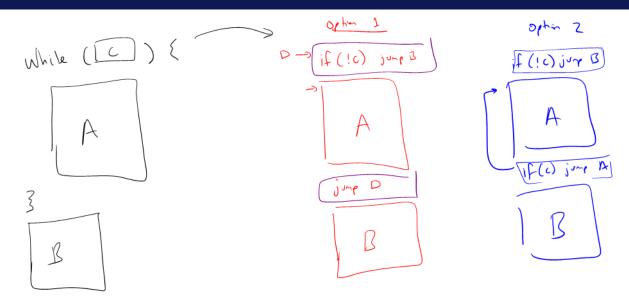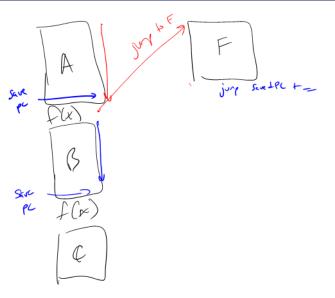How do we turn our control constructs into jump statements?

if else   while

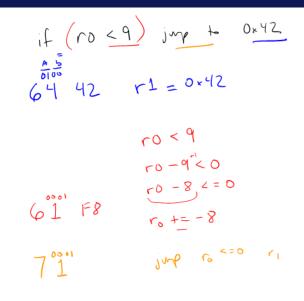While ($\boxed{c}$) {

A

}

B

Option 1

D → $\boxed{\text{if (!c) jump B}}$

→ A

jump D

B

Option 2

$\boxed{\text{if (!c) jump B}}$

A

$\boxed{\text{if(c) jump A}}$

B

9

Function Calls

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

icode	b	meaning
0		rA = rB
1		rA += rB
2		rA &= rB
3		rA = read from memory at address rB
4		write rA to memory at address rB
5	0	rA = ~rA
	1	rA = -rA
	2	rA = !rA
	3	rA = pc
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

00
01
02
03
⋮
0F
10

00   01   02   03   04   05   06

read the
inredick

Memory

What kinds of things do we put in memory?

- Code: binary code like instructions in our example ISA
  - Intel/AMD compatible: x86_64
  - Apple Mx and Ax, ARM: ARM
  - And others!
- Variables: we may have more variables that will fit in registers
- Data Structures: organized data, collection of data
  - Arrays, lists, heaps, stacks, queues, …

Dealing with Variables and Memory

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

x	0x80
y	81
z	82
a	83
b	84
c	85

$$x = M[80]$$
$$y = M[81]$$
$$x += y$$
$$M[80] = x$$
$$M[81] = y$$