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 (February 20, 2026) in class
    - Written, closed notes
    - If you have SDAC, please schedule ASAP
    - Review session in class next Wednesday

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?

$r0 < 9 \Leftrightarrow r0 <= 8 \Leftrightarrow (r0-8) <= 0$

$\Leftrightarrow r0 += -8 \ (0xF8)$
$r0 <= 0$

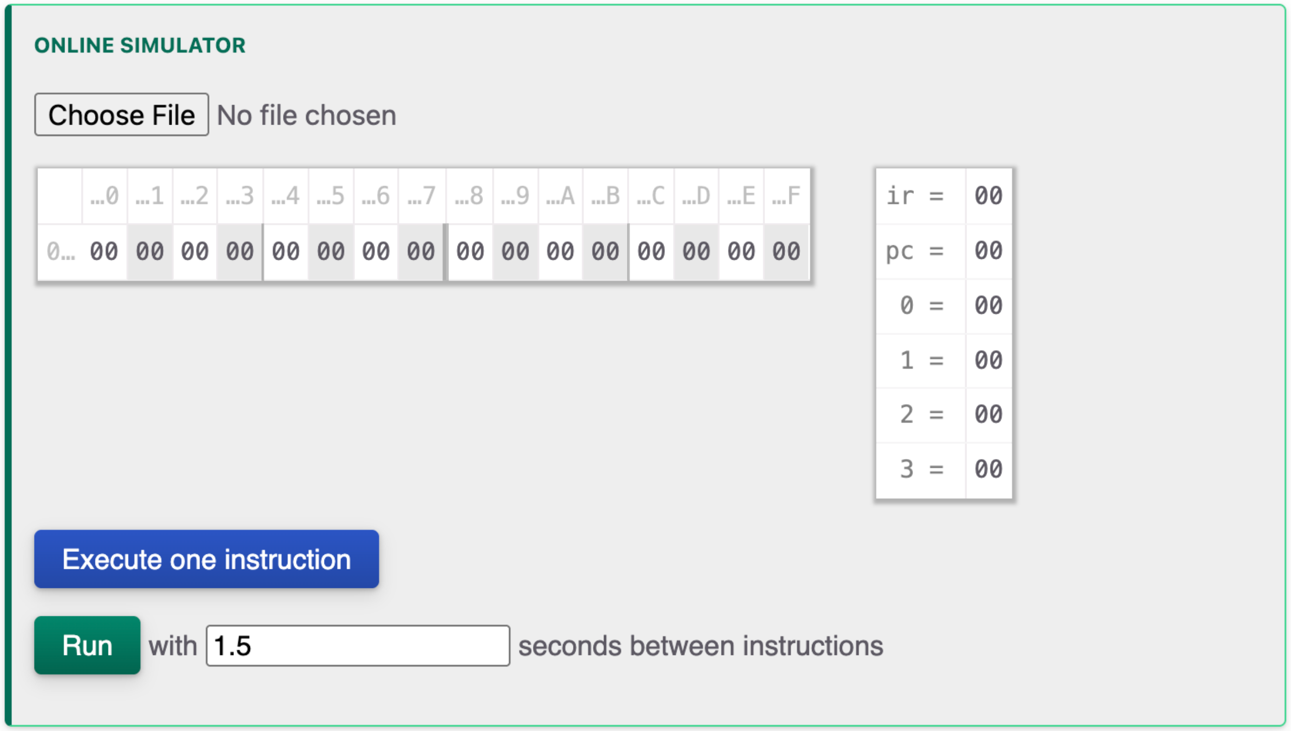
r1 = 0x42 $\quad \underbrace{0\,110}_{6}\ \underbrace{01\,00}_{4}\ \underbrace{42}_{42}$

r0 += F8 $\quad \underbrace{0\,110}_{6}\ \underbrace{00\,10}_{2}\ \underbrace{F8}_{F8}$

if r0<=0, PC=r1 $\quad \underbrace{0\,111}_{7}\ \underbrace{\overset{(R0)}{00}\,\overset{(R1)}{01}}_{1}$

$\boxed{6442\,62F8\,71}$

Online Simulator


ONLINE SIMULATOR

Choose File No file chosen

	...0	...1	...2	...3	...4	...5	...6	...7	...8	...9	...A	...B	...C	...D	...E	...F
0...	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

ir =	00
pc =	00
0 =	00
1 =	00
2 =	00
3 =	00

with 1.5 seconds between instructions

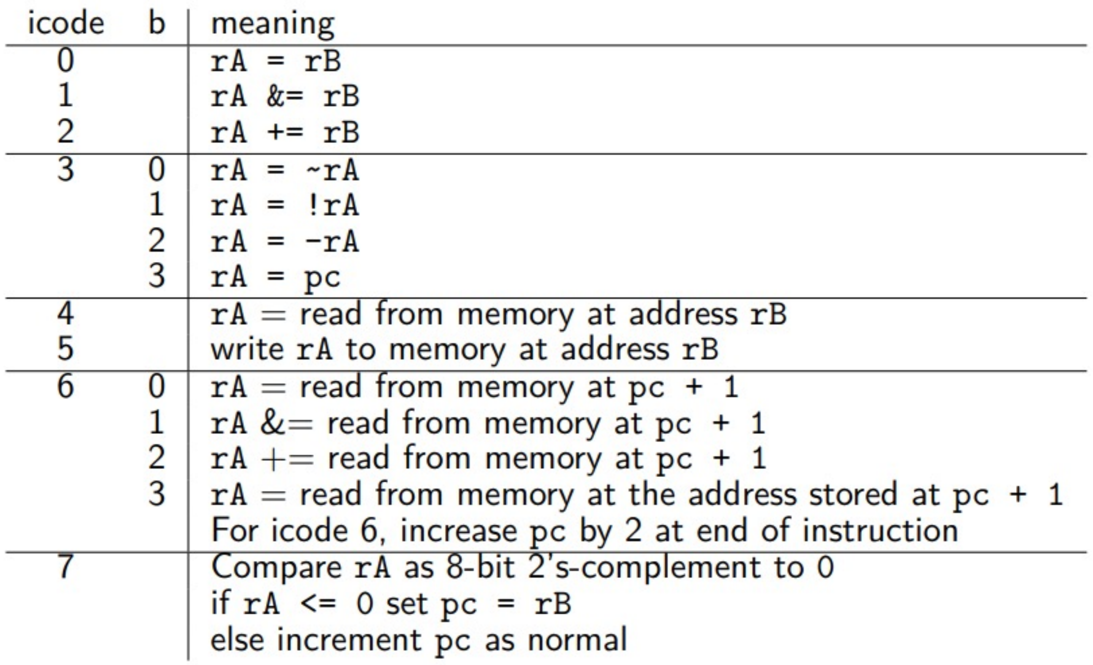
https://uva-cs.github.io/cso1-s26/homework/hw3-product.html

Homework Hints

1. Write pseudocode that does the desired task.
2. Deal with control flow.
3. Split multi-operation lines into a series of single-operation lines.
   - x = y − z; becomes x = y; x -= z;
4. Convert operations to those in our instruction set.
   - x -= z; becomes w = z; w = -w; x += w;
5. Deal with loops.
6. Assign variables to our four registers.
   - Example: r0 = x, r1 = y, r2 = z, r3 = w
   - r0 = r1; r3 = r2; r3 = -r3; r0 += r3
7. Write those instructions into triples, then hex.

UNIVERSITY of VIRGINIA

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

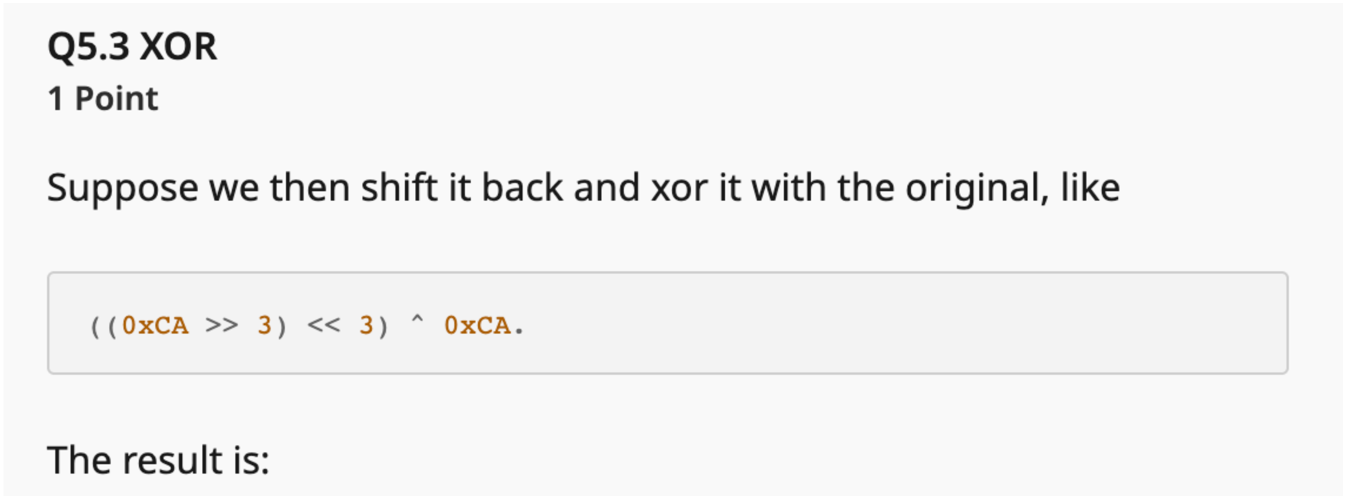
Example 4: 0x17 * 3

Dealing with Variables and Memory

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

Function Calls

Exercises



Q5.3 XOR
1 Point

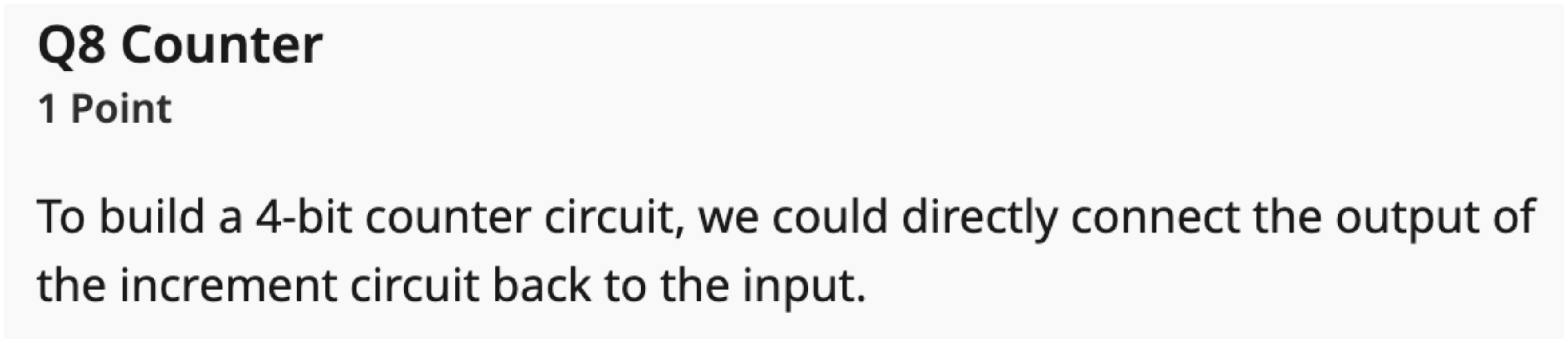
Suppose we then shift it back and xor it with the original, like

```
((0xCA >> 3) << 3) ^ 0xCA.
```

The result is:

- The same for both signed and unsigned integers
- Larger for signed than unsigned integers
- Larger for unsigned than signed integers
- There is no way to know

Exercises



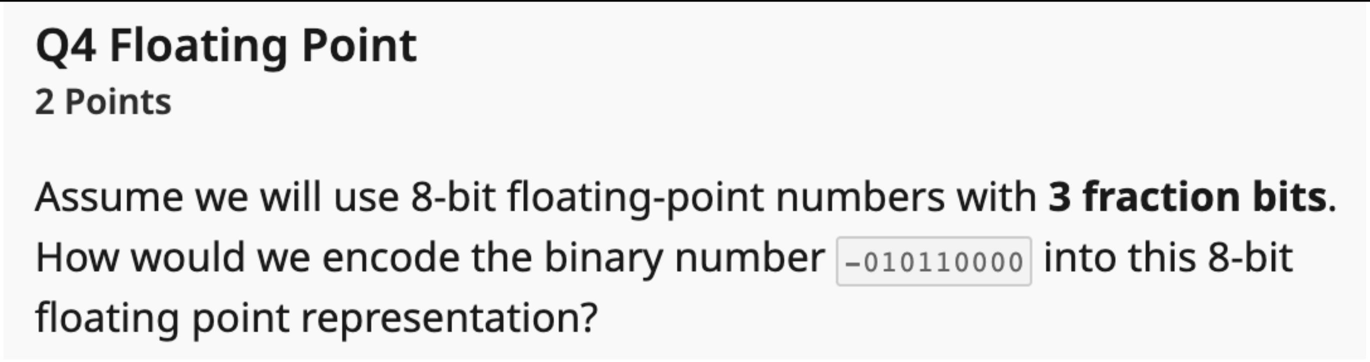
Q8 Counter

1 Point

To build a 4-bit counter circuit, we could directly connect the output of the increment circuit back to the input.

True/False

Exercises



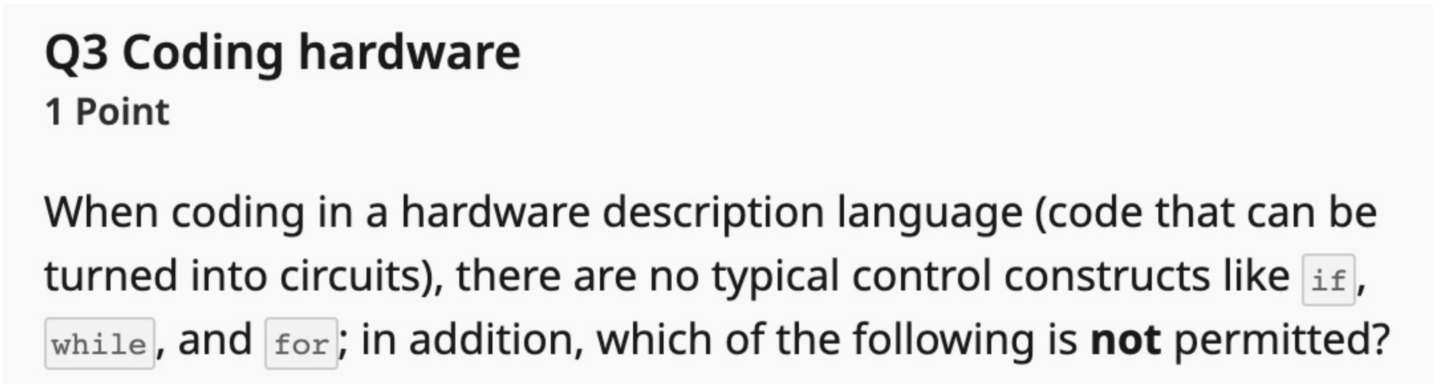
Q4 Floating Point

2 Points

Assume we will use 8-bit floating-point numbers with **3 fraction bits**. How would we encode the binary number `-010110000` into this 8-bit floating point representation?

- 0 011 1110
- 1 011 1110
- 0 0111 011
- 0 1110 011
- 1 0111 011
- 1 1110 011

Exercises



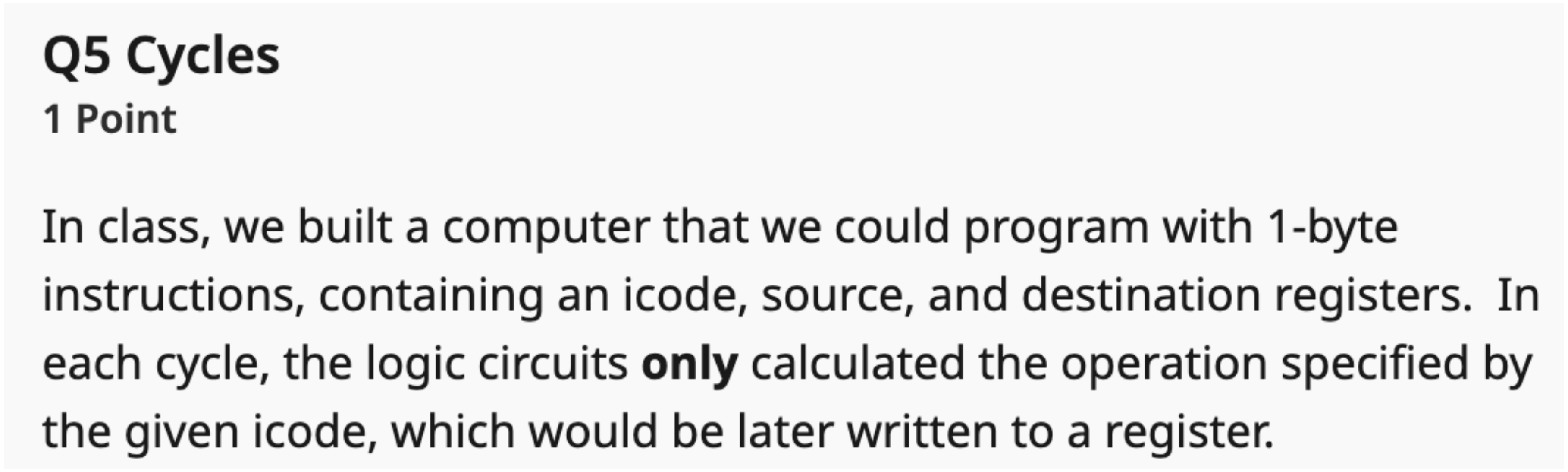
Q3 Coding hardware

1 Point

When coding in a hardware description language (code that can be turned into circuits), there are no typical control constructs like `if`, `while`, and `for`; in addition, which of the following is **not** permitted?

- Accessing the same variable twice, like y = x + 1; z = x - 1;
- Assigning to the same variable twice, like y = x + 1; y = w - z;
- Conditional operations, like y = (x < 0) ? x : -x;
- Including several operators in a single expression, like y = (x + y) ^ z;

Exercises



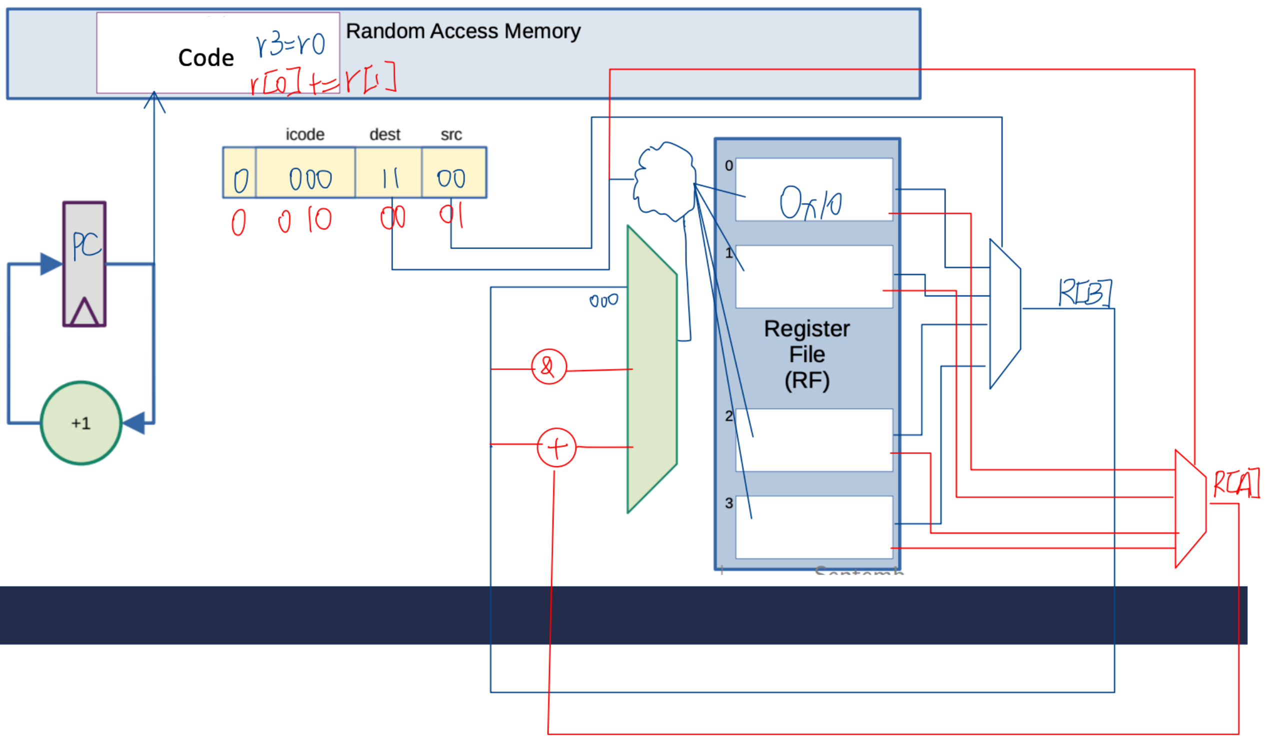
Q5 Cycles

1 Point

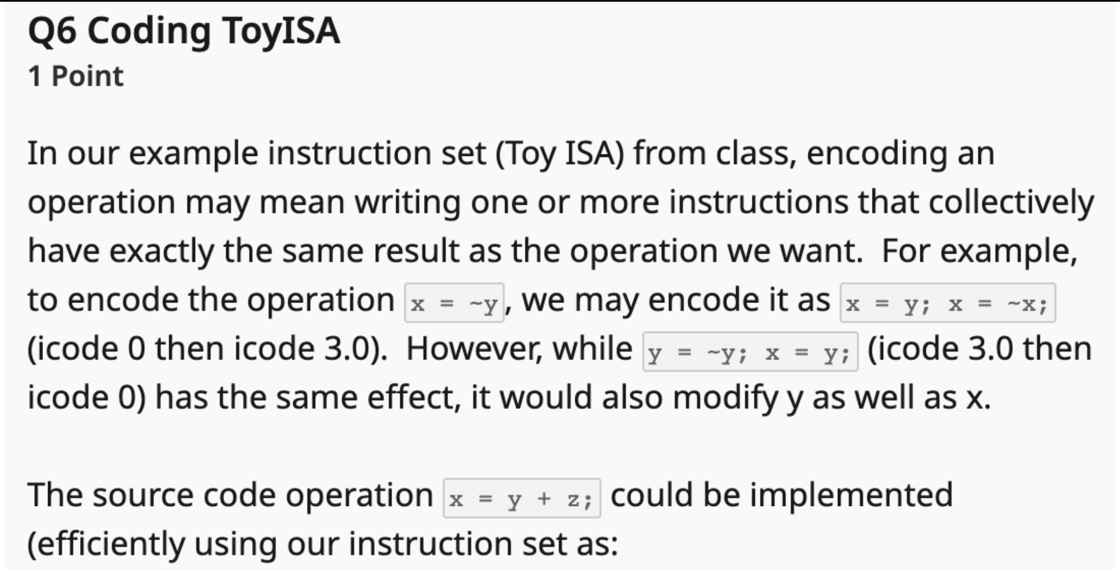
In class, we built a computer that we could program with 1-byte instructions, containing an icode, source, and destination registers. In each cycle, the logic circuits **only** calculated the operation specified by the given icode, which would be later written to a register.

True/False

Building a Computer



Exercises



Q6 Coding ToyISA

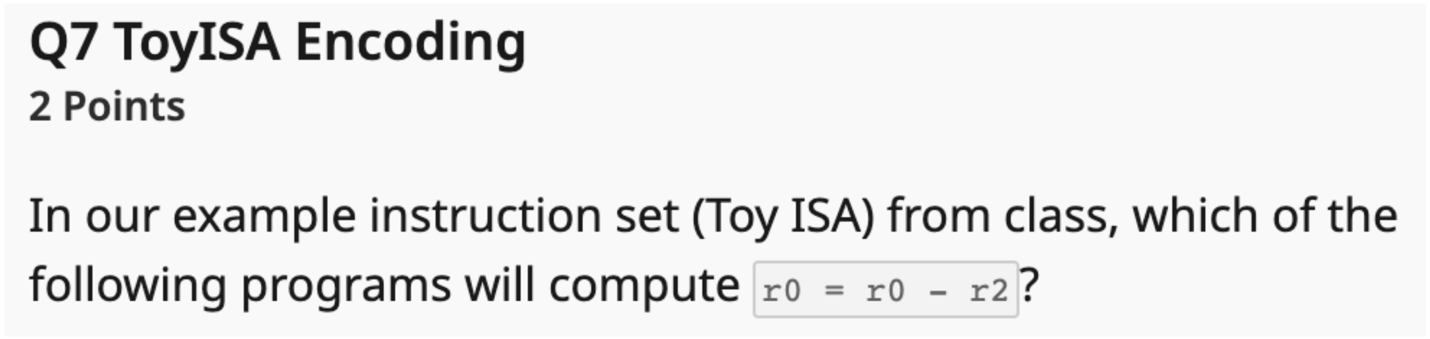
1 Point

In our example instruction set (Toy ISA) from class, encoding an operation may mean writing one or more instructions that collectively have exactly the same result as the operation we want. For example, to encode the operation `x = ~y`, we may encode it as `x = y; x = ~x;` (icode 0 then icode 3.0). However, while `y = ~y; x = y;` (icode 3.0 then icode 0) has the same effect, it would also modify y as well as x.

The source code operation `x = y + z;` could be implemented (efficiently using our instruction set as:

- One instruction
- Two instructions
- Three or more instructions
- It cannot be implemented with the Toy ISA instructions

Exercises



Q7 ToyISA Encoding

2 Points

In our example instruction set (Toy ISA) from class, which of the following programs will compute `r0 = r0 - r2`?

- 3a 02
- 06 36 21
- 3a 12 00
- 06 34 21
- None of the above