Function Calls, Memory
Instruction Set Architectures

CS 2130: Computer Systems and Organization 1

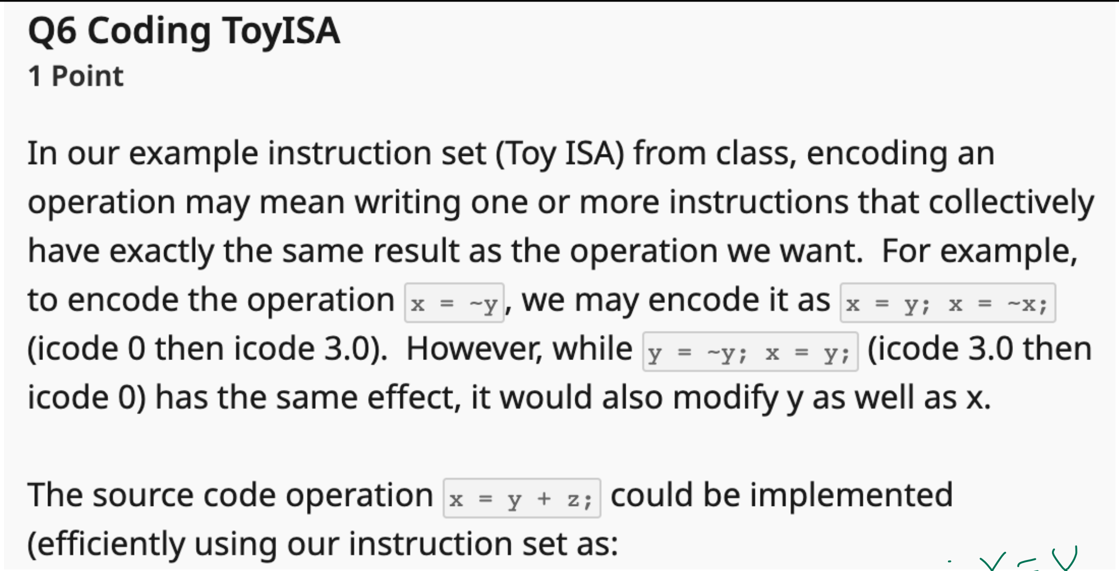
Xinyao Yi Ph.D.
Assistant Professor

Announcements

- Homework 3 due Next Monday on Gradescope
- Midterm 1 Friday in class
- Written, closed notes
- If you have SDAC, please schedule ASAP

UNIVERSITY of VIRGINIA

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

X=Y

X+=Z

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`?

idea 1: find all the instruction sets and encode them

for example: r2 =-r2 / r0 += r2 or r1 = r2 / r1 =-r1 / r0 += R1

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

you just find all possible solutions and encode them to find the answer.

idea 2: check each option to see what this binary do.

0000 0110   0011 0110   0010 0001

R1 = R2   R1 =- R1   R0 += R1

Suppose we extended the ISA simulator you wrote in Lab 4 with the following code:

```
if (reserved == 1 && icode == 7) {
    R[a] = R[b] >> M[oldPC + 1];
    return oldPC + ___;
}
```

1. [4 points] What is the value that we will need to increment the `oldPC`? (Fill in the blank below.)

```
    return oldPC + _2_;
```

① r0 = r1 >>4

② r0 &= r1

③ r0 &= 0000 0001

2. [10 points] Using the new instruction above **at least once**, write a program that determines if both of the hexadecimal digits representing the byte in register 1 are odd, storing a 1 in register 0 if both are odd and a 0 if either are even. For example, if `r1` is `0x73`, store a `0x01` in `r0`; if `r1` is `0xE9`, store a `0x00` in `r0`. Do not change the values stored in `r1`, `r2`, or `r3`. Answer in hexadecimal bytes, separated by spaces. *Hint: You may need to write additional instructions.*

Answer: F1 04 11 61 01

3. [12 points] Complete the table below listing all the register values as hex digits after the following code executes. Assume that all registers start with value `0x00` and that the first instruction is at address `0x00`.

```
64 05 09 26 4E 62 F8 14 80
```

Register	Value
0	F8
1	08
2	05
3	62

5. [15 points] Answer the following questions assuming 8-bit two's-complement numbers.

A. Compute the following sum, showing your work (such as carry bits, etc).

```
    0  0  1  1  1  0  1  1
 +  0  1  1  0  0  1  1  1
 -------------------------
    1  0  1  0  0  0  1  0
```

Overflow.

B. Is your result a positive or negative number? (circle one)

Positive [✓ **Negative**]

C. Convert your solution to decimal.

Answer
−94.

7. [8 points] The following number is encoded as an 8-bit floating point number assuming a 4-bit exponent value.

$$\underbrace{0}\underbrace{1001}\underbrace{111}$$

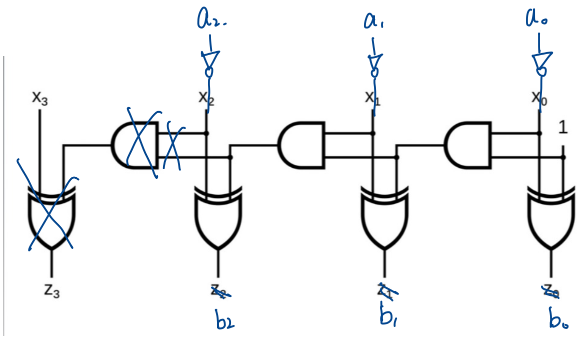
A. Write the value in binary scientific notation.

Answer
1.111×2^2

B. Write the value in decimal.

Answer
7.5.

10. [14 points] Using only 2-input `and`, `or`, `xor` gates, 1-input `not` gates, and constants `0` and/or `1`, draw a circuit that has 3-bit input a with bits labeled a_2, a_1, and a_0; 3-bit output with bits labeled b_2, b_1, b_0; which computes $b = -a$ in 3-bit two's complement. *Clearly label your inputs and output bits individually.*



$b = -a = \sim a + 1$

How to flip a ?

How to add 1 ?

(Check the increment circuit in the previous slides!)