CS 2130 Exam 1

Name ______________________________

You MUST write your e-mail ID on **EACH** page and put your name on the top of this page, too.

If you are still writing when "pens down" is called, your exam will not be graded – even if you are still writing the honor pledge. So please do that first. Sorry to have to be strict on this!

There are 6 pages to this exam. Once the exam starts, please make sure you have all the pages. Questions are worth different amounts of points, so be sure to look over all the questions and plan your time accordingly.

This exam is CLOSED text book, closed-notes, closed-cell phone, closed-smart watch, closed-computer, closed-neighbor, etc. You may **not** discuss this exam with anyone until after all the exam times have ended. Please write and sign the honor pledge below.

I am just a child who has never grown up. I still keep asking these 'how' and 'why' questions. Occasionally, I find an answer.

–Stephen Hawking

Page 2: Binary

1. [6 points] What is `0x25` in decimal?

Answer

37

2. [6 points] What is `0b011010111101` in hexadecimal?

Answer

0x6bd

3. [14 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).

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

`0 1 1 1 1 0 1 0`, but there was a carry 1

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

Positive **Negative**

Positive

Page 3: Logic and FP

4. [11 points] Write the following binary number as an 8-bit floating point number assuming a 4-bit exponent value.

`-10111.01100`

Answer

`1 1011 100`, partial credit for 0100 exponent or 011 fraction

5. [14 points] Assume x and y are 8-bit two's complement integers. For each of the following pairs of expressions, if the two expressions are equivalent for all x and y, write "same"; otherwise, write an example x (and y if used in the expressions) in binary for which the two are different.

   A. `(x >> 5) << 4` and `(x << 4) >> 5`

   different: x = 11110000

   B. `~(x ^ y)` and `(x & y) | (~x & ~y)`

   same, x and y are same 1s, not-x and not-y are same 0s.

6. [6 points] Assume 32-bit signed integers. Which of the following are 1 if and only if x is negative? (Fill in the bubble for each that applies.)

   ◯ `x >> 31`

   ◯ `(x >> 31) & 0x1`

   ◯ `!(x >> 31)`

   ◯ `x & 0x80000000`

   ◯ `(x & 0x80000000) | 0x1`

   ◯ `!((~x) & 0x80000000)`

   B, F

Page 4: Programming our Computer

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

```
if (reserved == 1 && icode == 4) {
    R[a] = R[b] & M[oldPC + 1];
    return oldPC + ___;
}
```

7. [4 points] What is the value that we will need to increment the oldPC? (Fill in the blank below.)

```
return oldPC + _____;
```

2

8. [15 points] Using the new instruction above at least once, write a program that determines if the contents of register 2 is a negative number in two's complement and stores the result in register 0. That is, if the high order bit of register 2's value is a `1`, your program should store a `0x01` in register 0. Answer in hexadecimal bytes, separated by spaces. *Hint: you may need to write additional instructions.*

Answer: ____________________

C2 80 52 52

9. [8 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`.

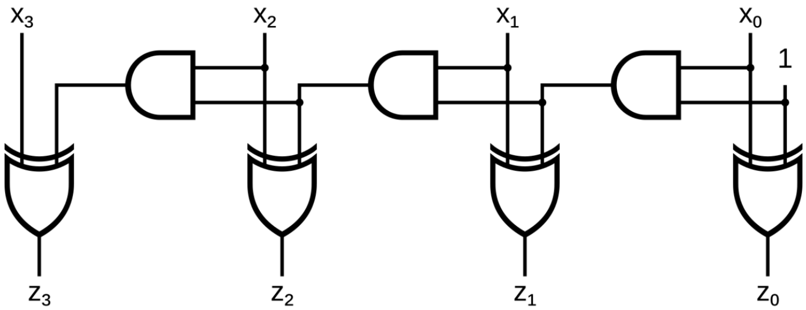
```
6c 20 60 FF 2c 04 54 19
```

Register	Value
0	
1	
2	
3	

0=FF, 1=00, 2=00, 3=20

Page 5: Circuits

10. [16 points] In class, we discussed a 4-bit increment circuit below that added 1 to the input.



How can we change this circuit to instead increment by 2, i.e., `x += 2`? Draw the new circuit below. *Note: you should not use more gates than the original circuit.*

Directly connect x0 to z0; replace the and gate from x0 to x1's xor with a 1 input (incrementer starting at the 2^1 position).

Page 6: Scratch Paper

Our Example ISA. This is the ISA described in class and used in Lab 4 and Homework 3. Each instruction is one (or two) bytes, defined as:

7	6 5 4	3 2	1 0
R	icode	a	b

byte at `pc`

7 6 5 4 3 2 1 0
immediate

byte at `pc + 1`

If the reserved bit (bit 7) in our instruction is 0, the following table defines our instruction encoding.

icode	b	behavior
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 two's-complement to `0`
		if `rA <= 0` set `pc = rB`
		else increment `pc` as normal

Nothing on this page will be graded.