20F-COMSCIM151B-1 Homework 3

NEVIN LIANG

TOTAL POINTS

57.5 / 60

QUESTION 1

R to IEEE 754 : 1 10 pts

1.1 125.5 as binary32 5 / 5

✓ **+ 5 pts** Correct

+ 0 pts Incorrect

1.2 Flipped 25th bit 5 / 5

✓ **+ 5 pts** Correct

+ 0 pts Incorrect

QUESTION 2

R to IEEE 754 : 2 7 pts

2.1 -14.25 as binary32 5 / 5

✓ **+ 5 pts** Correct

+ 0 pts Incorrect

2.2 Error? 2 / 2

✓ **+ 1 pts** No overflow error

✓ **+ 1 pts** No rounding error

QUESTION 3

Stuck-at-zero 18 pts

3.1 RegWrite 3 / 3

✓ **+ 2 pts** Correct explanation

✓ **+ 1 pts** Correct faulty instructions

+ 0 pts Incorrect

3.2 ALUop 3 / 3

✓ **+ 2 pts** Correct explanation

✓ **+ 1 pts** Correct instructions

+ 0 pts Incorrect

3.3 ALUSrc 3 / 3

✓ **+ 2 pts** Correct explanation

✓ **+ 1 pts** Correct instructions

+ 0 pts Incorrect

3.4 Branch 3 / 3

✓ **+ 2 pts** Correct explanation

✓ **+ 1 pts** Correct instructions

+ 0 pts Incorrect

3.5 MemRead 3 / 3

✓ **+ 2 pts** Correct explanation

✓ **+ 1 pts** Correct instructions

+ 0 pts Incorrect

3.6 MemToReg 3 / 3

✓ **+ 2 pts** Correct explanation

✓ **+ 1 pts** Correct instructions

+ 0 pts Incorrect

QUESTION 4

Ternary add 15 pts

4.1 Reused blocks 5 / 5

✓ **- 0 pts** Correct

- 2.5 pts One mistake

- 5 pts Incorrect

4.2 New blocks 2.5 / 5

✓ **+ 2.5 pts** Extra adder

+ 2.5 pts Extra read port

+ 0 pts Incorrect

4.3 New control signals 5 / 5

✓ **+ 5 pts** Correct

+ 0 pts Incorrect

QUESTION 5

5 Stack machine assembly 10 / 10

✓ **- 0 pts** **Correct**

- 10 pts Incorrect

QUESTION 6

6 Late submission penalty 0 / 0

✓ **- 0 pts** **On time**

- 15 pts Late submission

NEVIN LIANG
CS M151B

Homework – III

Session: Fall 2020 **Instructor: Prof. P. Gupta**

Total Points: 60 **Due Date: Oct 27 8:00 PM**

Instructions

1. **There are two sections in this homework set. The questions under Section – I are for your practice. Solutions to these problems are already provided on the course web page. Students are encouraged to solve the problems before looking at the solution provided.**
2. **Problems under Section – II are to be submitted.**
3. **Submit your solutions to gradescope. You must include all the pages in the template as part of your solution.**

Section – I

1. Different instructions utilize different hardware blocks in the basic single-cycle implementation. The next three problems in this exercise refer to the following instruction:

	Instruction	Interpretation
a.	add Rd, Rs, Rt	Reg[Rd] = Reg[Rs] + Reg[Rt]
b.	ld Rt, Offs(Rs)	Reg[Rt] = Mem[Reg[Rs] + Offs]

   a. What are the values of the control signals generated by the control in Figure 1 for this new instruction?
   b. Which resources (blocks) perform a useful function for this instruction?
   c. Which resources (blocks) produce outputs, but their outputs are not used for this instruction? Which resources produce no outputs for this instruction?

2. Convert the decimal 1/10 i.e., 0.1 to a floating-point representation. Do you get rounding error?

Section – II

1. In modern chips, it is very expensive to ensure that no errors ever happen. Imagine a RISC-V processor where single precision floating point registers are especially error prone. Errors are modeled as flipping of exactly *one bit* in the 64-bit register. Represent 125.5 in IEEE 754 FP representation. **[5+5 = 10 points]**

$125.5_{10} = 64 + 32 + 16 + 8 + 4 + 1 + 1/2 = 1111101.1_2$

$1111101.1_2 = 1.1111011 * 2^6$

$E = 6 + 127 = 133 = 10000101_2$
$F = 0.1111011, S = 0$

$125.5_{10} = 01000010111110110000000000000000_2$

01000010111110110000000000000000

Now assume due to an error, bit number 25 flips (sign bit is bit #31). What is the floating point number represented now?

$S = 0$
$E = 10000101_2 \rightarrow 10000001_2 = 129$
$F = 1111011_2 \rightarrow 1.1111011_2$
$exp = 129 - 127 = 2$

$x = 111.11011_2 = 7.84375$

7.84375

2. The value of a float type variable is represented using the single-precision 32-bit floating point format IEEE-754 standard that uses 1 bit for sign, 8 bits for biased exponent and 23 bits for mantissa. A float type variable X is assigned the decimal value of −14.25. The representation of X in hexadecimal notation is: **[5 points]**

$x = -14.25 = -(8 + 4 + 2 + 1/4) = -1110.01_2 = -1.11001 * 2^3$

$F = 1101_2$
$E = 127 + 3 = 130 = 10000010_2$
$S = 1$

$x = 11000001011001000000000000000000_2$
$= C1640000_{16}$

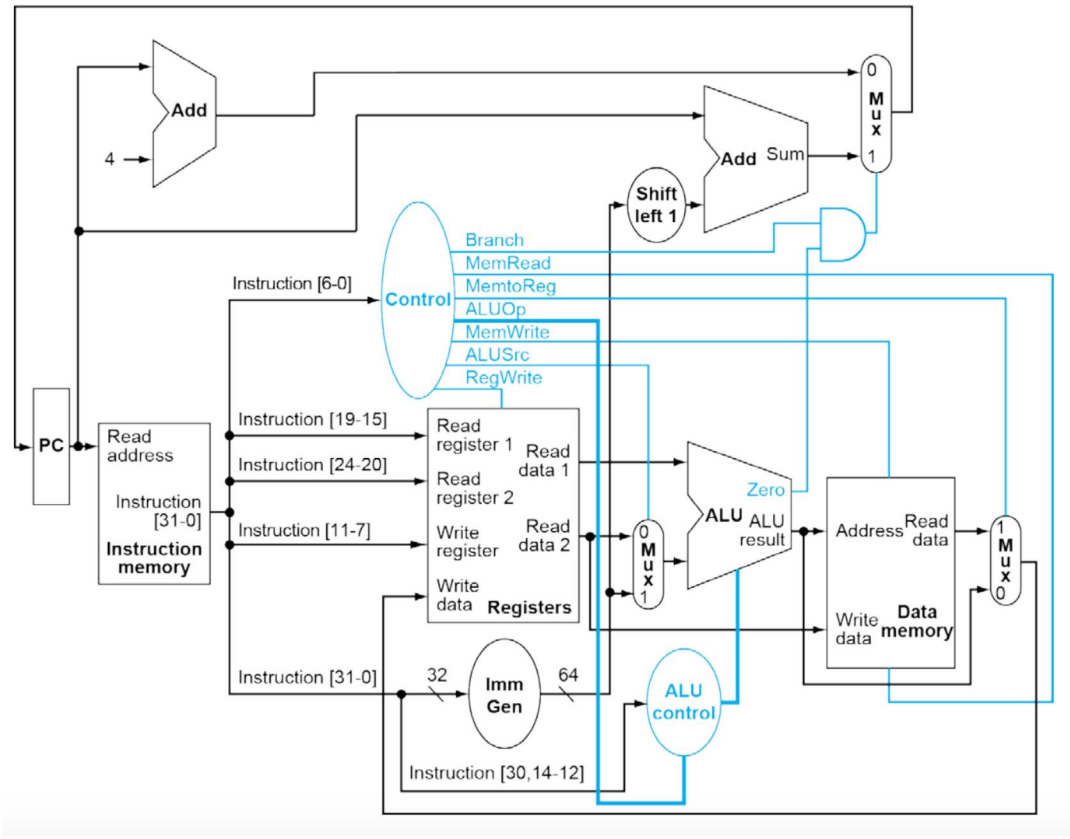
0xC1640000

Indicate whether an overflow or rounding error occurs. **[2 points]**

no

3. Describe the effects of a single stuck-at-0 fault (i.e. regardless of what it should be, the signal is always 0) would have for the signals shown below, in the single-cycle datapath in Figure 1. Which instructions, if any, will not work correctly? Explain why. **[18 points]**
   a. RegWrite = 0
   b. ALUop0 = 0
   c. ALUSrc = 0
   d. Branch = 0
   e. MemRead = 0
   f. MemtoReg = 0

Figure 1:



a.

R-format instructions won't work because they have a destination register, and regwrite = 0 means it physically cannot write a value into a destination register. Similarly, I-format and U-format will not work either.

Load instructions (I-format) don't work because you can't write something into a register.

[It also appears that UJ-Format instructions aka Jump and Link and Jump and Link Register don't work, because they have to write to a destination register as well.

b.

Input or output	Signal name	R-format	ld	sd	beq
Inputs	I[6]	0	0	0	1
	I[5]	1	0	1	1
	I[4]	1	0	0	0
	I[3]	0	0	0	0
	I[2]	0	0	0	0
	I[1]	1	1	1	1
	I[0]	1	1	1	1
Outputs	ALUSrc	0	1	1	0
	MemtoReg	0	1	X	X
	RegWrite	1	1	0	0
	MemRead	0	1	0	0
	MemWrite	0	0	1	0
	Branch	0	0	0	1
	ALUOp1	1	0	0	0
	ALUOp0	0	0	0	1

When ALUOp0 is 1, only the instruction beq doesn't work.

c.

From the previous table, only load and store fail. For an explanation, The MUX before the ALU, the ALUOp if stuck at 0, always reads data 2 register and never the immediate. Thus, branch would work (because the immediate goes up into the top adder), and R-type instructions would also work (because no immediate). However, I-type, Load, and Store would all fail because ALUOp is 0 and those need ALUOp to be 1.

d.

Branch is only 1 when the instruction BEQ is executed. Any non-branch statement would work. ALL Branch instructions will fail.

e.

If MemRead is stuck at 0, then all instructions that read memory would fail. This includes the ld instruction. No other instruction reads memory so only ld will fail.

f.

MemtoReg, if stuck at 0, is actually a special one because for sd and beq, it doesn't matter what the control output it. Because regwrite is 0, the register isn't even being written to. Forcing MemtoReg = 0 would only make ld fail, because ld loads from memory to a register and therefore does indeed write to a register.

4. The basic single-cycle RISC-V implementation can only implement some instructions. New instructions can be added to an existing ISA, but the decision whether or not to do that depends, among other things, on the cost and complexity such an addition introduces into the processor datapath and control. The next three problems refer to the new instruction given in the table below: **[15 Points]**

Instruction	Interpretation
add3 Rd, Rs, Rt, Rx	Reg[Rd] = Reg[Rs] + Reg[Rt] + Reg[Rx]

a. Which existing blocks (if any) can be used for this instruction? **[5 points]**

The Register block, Instruction Memory block, and ALU have to all be used for this new instruction.

The Register block's read register ports and write ports have to all be used.

b. Which new functional blocks (if any) do we need for this instruction? **[5 points]**

If we put another ALU that adds Rs to Rt + Rx, then you're able to add 3 registers together.

If we had an ALU with 3 register read ports, that would work as well :)

c. What new signals do we need (if any) from the control unit to support this instruction? **[5 points]**

If we made a new control signal like TRIPLEADD, then we could tell the ALU to feed the sum of two registers into the new ALU and therefore add the third number. That way, the instruction would be supported.

5. Assuming all relevant arithmetic operations are available, write assembly code $c = A(x+b*y)+d$ for a stack machine. (Be careful about that the exact instruction format would be different from RISC-V). Write each instruction in a separate box. **[10 points]** Partial points will be provided.

```
PUSH Y
PUSH B
MULT
PUSH X
ADD
PUSH A
MULT
PUSH D
ADD
POP C
```