Name: Yu Ching Hei
SID: 1155193237
email: chyu2@cse.cuhk.edu.hk

Question 1. (15%)

Consider the following RISC-V instructions. Please note that we treat NUM_1%2 and NUM_1%2+1 as decimal values.

```
1     li a1, 1    #li a1, NUM_1%2
2     li a2, 2    #li a2, NUM_1%2+1
3     li a3, 6
4     LOOP:
5     slti t0, a3, 1
6     bne t0, zero, DONE
7     add a4, a1, a2
8     addi a1, a2, 0
9     addi a2, a4, 0
10    addi a3, a3, -1
11    jal x0, LOOP
12    DONE:
13    # end of the program
```

1. How many times is the branch instruction executed? (7%)
2. What are the final values of a1 and a2. (8%)

Answer:

1. 7 times, the first 6 times didn't fulfill the check, and the 7^{th} time got the check fulfilled
2. a1 = 21, a2 = 34

Question 2. (20%)

Read through the multiplication / devision algorithm:

Left: multiplication algorithm, Right: division algorithm

Write down the step by step procedure to calculate 5×2 or 0101×0010. Use Multiplier0 to indicate the least significant bit of the multiplier. List the initial values and the values in 1st to 4th iterations of Multiplier, Multiplier0, Multiplicand and Product. In each iteration, list the values after 1, 2 and 3 steps in Figure 1 left separately. (Represent Multiplier as 4bits, Multiplier0 as 1bit, Multiplicand as 8bits, Product as 8bits.)

Answer:

Iteration	Step	Multiplier	Multiplier0	Mcand	Product
0	Initial value	010<u>1</u>	1	0000 0010	0000 0000
1	1⇒Prod=Prod+Mcand	0101	1	0000 0010	0000 0010
	Shift left Multiplicand	0101	1	0000 0100	0000 0010
	Shift right Multiplier	001<u>0</u>	0	0000 0100	0000 0010
2	Shift left Multiplicand	0010	0	0000 1000	0000 0010
	Shift right Multiplier	000<u>1</u>	1	0000 1000	0000 0010
3	1⇒Prod=Prod+Mcand	0001	1	0000 1000	0000 1010
	Shift left Multiplicand	0001	1	0001 0000	0000 1010
	Shift right Multiplier	000<u>0</u>	0	0001 0000	0000 1010
4	Shift left Multiplicand	000<u>0</u>	0	0010 0000	0000 1010
	Shift right Multiplier	000<u>0</u>	0	0010 0000	0000 1010

Question 3. (20%)

IEEE 754 Floating-Point Standard

1. What decimal number does this single precision float $\texttt{C13C0000}_{16}$ represent? (Show your work.) (10 %)
2. What is -1.5$_{10}$ in IEEE single precision binary floating point format? (Show your work.) (10%)

Answer:

1. (a) by breaking the hexadecimal number into binary, we get

$$\texttt{C13C0000}_{16} = \texttt{11000001001111000000000000000000}_2$$

   (b) the first bit indicate the sign, in this case it is a negative number

   (c) the next 8 bits are used to express the exponent of 2 offset by -127, so the exponent in this case is $\texttt{10000010}_2 - \texttt{127}_{10} = \texttt{130}_{10} - \texttt{127}_{10} = \texttt{3}_{10}$

   (d) the following 23 bits($\texttt{01111000000000000000000}_2$) are the mantissa which is a decimal number with a leading 1. i.e. $\texttt{1.01111}_2$, it is then shift right by times of the exponent and therefore $-\texttt{1011.11}_2$

   (e) $\texttt{1011}_2 = \texttt{11}_{10}$ & $\texttt{0.11}_2 = \texttt{0.5} + \texttt{0.25} = \texttt{0.75}_{10}$

$$\therefore -11.75_{10}$$

2. (a) Since it is negative, the first bit is 1

   (b) Since the part before the decimal point is -1 only, there is no need to shift the bits exponent = 127

   (c) then convert the number after the decimal point by multiplying 2 and extract the decimal number,

$$0.5 \times 2 = \underline{1}.0 \Rightarrow 1$$

   so we have mantissa 100 0000 0000 0000 0000 0000

   (d) so the single precision floating point number is

$$1\ 01111111\ 10000000000000000000000_2 = \text{BFC00000}_{16}$$

Question 4. (10%)

Consider the following instruction:
Instruction: `xor rd, rs1, rs2`
Interpretation: `Reg[rd] = Reg[rs1] XOR Reg[rs2]`

1. What are the values of control signals generated by the control in figure 2 for this instruction?
2. Which resource (block) produces no output for this instruction?

Answer:

1. RegWrite = true, ALUSrc = 0, ALU Control = "xor", MemWrite = false, MemRead = false, MemToReg = 0
2. Data Memory

Question 5. (15%)

The following figures show the format and datapath of an R format instruction.

1. Assume we have an instruction whose machine code is 0x00c5d533, please write down the instruction in assembly language. (10%)
2. Which ports in the datapath do we use for addressing rs1, rs2, and rd? (5%)

Answer:

1. By breaking down the machine code, it is 0000 0000 1100 0101 1101 0101 0011 0011_2.
   (a) By bit0-7(0110011), we can know that it is a R-type instruction.
   (b) By bit7-11(01010), we can know that rd is x10 which is a0.
   (c) By "funct3" (bit12-14 = 101_2 = 0x5), we know that it is a shift right function.
   (d) By bit15-19(01011) we can know that r1 is x11 which is a1.
   (e) By bit20-24(01100) we can know that r2 is a2.
   (f) By "funct7"(bit25-31 = 0000000_2 = 0), so we can know that it is a shift right logical function.

   Therefore the code instruction in assembly is:

   ```
   srl a0, a1, a2
   ```

2. - Read Addr 1: rs1
   - Read Addr 2: rs2
   - Write Addr: rd

Question 6. (20%)

In this exercise, we examine how pipelining affects the clock cycle time of the processor. Problems in this exercise assume that individual stages of the datapath have the following latencies:

IF	ID	EX	MEM	WB
300ps	500ps	200ps	350ps	250ps

1. What is the clock cycle time in a pipelined and non-pipelined (single-cycle) processor?
2. What is the total latency of an lw instruction in a pipelined and non-pipelined (single-cycle) processor?
3. If we can split one stage of the pipelined datapath into two new stages, each with half the latency of the original stage, which stage would you split and what is the new clock cycle time of the processor?

Answer:

1. Clock cycle time:
   - pipelined: max(300, 500, 200, 350, 250) = 500ps
   - non-pipelined: $300 + 500 + 200 + 350 + 250 = 1600$ps
2. Total latency:
   - pipelined: $500 \times 5 = 2500$ps
   - non-pipelined: Single clock cycle time = 1600ps
3. I would split the stage "ID" as it has the longest latency which "EX" and "WB" only occupies less than or equal to half of its latency.
   New clock cycle time = 350ps