CS202 Computer Organization Midterm Exam - Spring 2024

1. (10 points) You are designing an embedded processor for a pacemaker. Based on an analysis of the monitoring software that it will run, you find the following mix of instructions, which have the specified execution time in your current design:

Instructions	Percentage	Time
Load	10%	7 cycles
Store	15%	10 cycles
Branch	15%	4 cycles
Add	55%	4 cycles
Multiply	5%	5 cycles

1) (2 points) What is the CPI of your processor on this mix of instructions?
2) (2 points) The analysis shows that the software has 6000 instructions and processor's execution time for the software is 63 us, what is the processor's clock rate?
3) (2 points) If a new compiler is used, the new mix of instructions is divided equally among the categories of instructions, except that for Add, which occurs twice as often as each of the others, what is the new CPI?
4) (2 points) The new compiler mentioned in question 3) generates a total number of 5000 instructions, without affecting the clock rate, what's the new execution time?
5) (2 points) Would you use the new compiler? Why?

2. (26 points) Considering the following RISC-V code.

```
1                   .include "store_value.asm"      # store_value.asm is a different file
2                   .data
3                   array: .word 1,2,3,4,5
4                   .text
5   0x00061C00      main: la a0, array
6   0x00061C08      li t1, 3
7   0x00061C0C      mv t2, zero
8   0x00061C10      loop: blt t1, zero, end
9   0x00061C14      slli t3, t1, 2
10  0x00061C18      add t3, a0, t3
11  0x00061C1C      lw t3, 0(t3)
12  0x00061C20      add t2, t2, t3
13  0x00061C24      addi t1, t1, -1
14  0x00061C28      j loop
15  0x00061C2C      end: add al, t2, zero
16  0x00061C30      jal ra, store_value             # procedure defined in store_value.asm
```

1) (3 points) For the pseudo-instruction 'j loop' at line 14, rewrite it using basic RISC-V instruction.
2) (9 points) For the following code segment, fill in the blank fields with hexadecimal number.

8	0x00061C10	loop: blt x6, x0, end	Machine code = 0x___(a)____
9	0x00061C14	slli x28, x6, 2	Machine code = 0x___(b)____
…	…		
15	0x00061C2C	end: add al, t2, zero	
16	0x00061C30	jal x1, store_value	after execution, x1=0x___(c)____

3) (3 points) What is the range of 32-bit instructions that can be approximately reached from the current PC using blt instruction?

4) (3 points) What is the range of 32-bit instructions that can be approximately reached from the current PC using jal instruction?

5) (4 points) Assume the code of store_value.asm is as follows, how the associate data segment in memory is modified after executing the procedure call (assuming address of array is 0x00010000).

```
.text
store_value: sw a1, 4(a0)
             jr ra
```

6) (4 points) Rewrite the code from line 8-14, you still can use one branch instruction, but jump instruction is not allowed to be used anymore. Do not modify code other than line 8-14, do not create new label other than existing ones.

3. (8 points) So far, the multiply hardware we learnt shown in the figure below is only used for unsigned multiplication. Thus, to deal with signed multiplication, the solution is to first convert the multiplicand and multiplier to positive numbers, and then negate the product only if the original signs disagree. Based on that, let's calculate the product of 0111×1011, where 0111 is the multiplicand and 1011 is the multiplier, both are 4-bit signed value.

1) (6 points) Please show the contents of each register on each iteration. We consider that the positive multiplicand and multiplier are 4-bit.

2) (2 points) What's the final result of the multiplication? (in 2's complement)

4. (8 points) If we want to calculate 6 divided by 4 using the hardware described in the figure below,

both dividend and divisor are unsigned 4-bit integers.

1) (4 points) Write down the initial value in Divisor and Remainder register respectively.

Divisor register:_____; Remainder register:_____

2) (4 points) In which direction should content in divisor and quotient register shift to?

Divisor register: shift_____(left/right), Quotient register: shift_____(left/right).

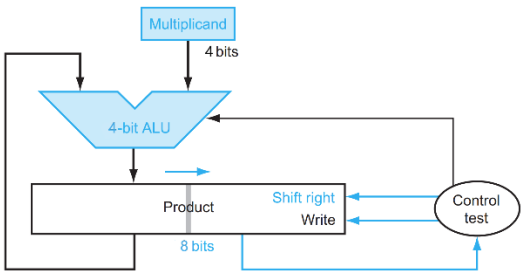


Figure for question 3

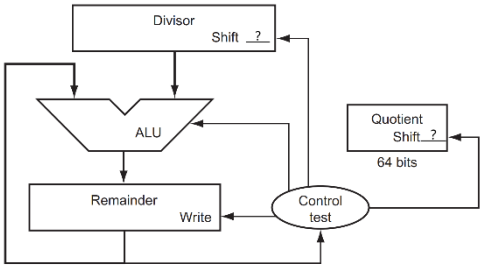


Figure for question 5

5. (12 points) Consider an 8-bit floating point representation, where the left most bit is the sign bit, the exponent is 3 bits, and the fraction is 4 bits. This 8-bit floating point follows IEEE-754 standard.

1) (2 points) For the above 8-bit floating point representation, the bias =_____?

2) (3 points) What's the range and relative precision of the 8-bit floating point representation, assuming the all-one-bit stream and all-zero-bit stream in the exponent domain are reserved.

3) As we all know, the complex numbers are in the form a + bi, where a is the real component, b is the imaginary component. We decide to create a new complex number format, using 16-bit to store both the real and imaginary components. The real component is placed on the left half and imaginary component is placed on the right half, both are represented in 8-bit floating point as we mentioned above.

a) (4 points) Given a new format complex number 0xB248, translate it in the form of a + bi.

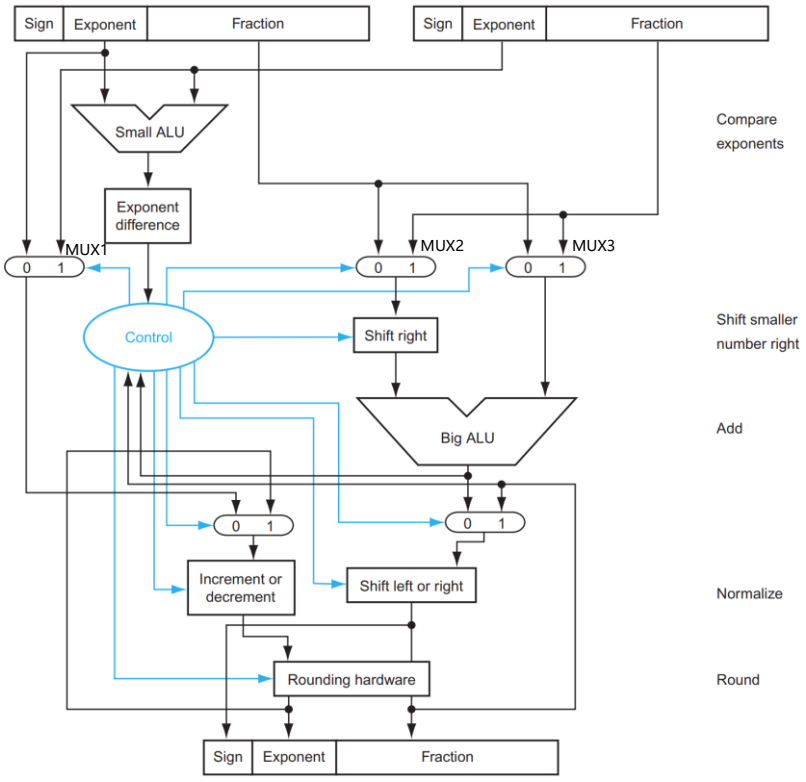
b) (3 points) Can the complex number 16.5+8i be converted into new complex format? If yes, write down the binary sequence, otherwise, explain the reason.

6 (12 points) In adder for floating points shown below, assume the left floating-point number is 1.125, the right floating-point number is -0.875. Both numbers follow the 64-bit double-precision floating point number format.

1) (5 points) Write down the control input for mux1, mux2 and mux3. Please show the calculation procedure explaining how you get the results.

2) (5 points) In the normalize stage, should the operation in the module "Increment or decrement" be increment or decrement? Should the operation in the module "shift left or right" be shifting left or right? If shift, by how many bits should it shift? Why?

3) (2 points) At which step of the floating-point addition should we check for overflow or underflow?



7 (24 points) These questions refer to the simplified single-cycle RISC-V datapath shown in the figure below, and this datapath supports the following 5 instructions: add, addi, lw, sw and beq.

1) (4 points) Consider the multiplexer labelled 'A' on the datapath diagram, which is controlled by the ALUSrc signal. Which of the supported instruction(s) mentioned above could NOT be executed correctly if ALUSrc was 0?

2) (4 points) Suppose the RegWrite signal was 0. Which of the supported instruction(s) mentioned above could NOT be executed correctly?

3) (4 points) For some instruction(s), it does not matter whether the MemtoReg signal (in the multiplexer labelled 'B') is set to 0 or 1 (meaning this signal is set to X). For which supported instruction(s) mentioned above is that true?

4) (3 points) For the modules PC, Instruction memory, Registers, ALU, MUX, Imm Gen, Data memory, which modules are combinatorial elements, which modules are state/sequential elements?

We want to add support for a new instruction 'LWPC' (Load Word Using PC address), while not affecting any other instructions. This new instruction has the format shown below and will load data from memory using the address given by PC + the shifted sign-extended immediate (i.e. LWPC rd, imm # rd=M[PC+{imm,1'b0}]. Implement changes to the datapath and control signals to support this new LWPC instruction on the single-cycle CPU. Assume when this instruction executes, a LWPC control signal will be generated and set to '1'.

a) (2 points) Among the 6 RISC-V formats, which format do you think the LWPC instruction should belong to?

b) (4 points) Explain clearly and briefly your modification with minimum change to the datapath, there's no need to draw unless you cannot clearly explain. If you want to draw, please draw neatly to make sure it's understandable.

c) (3 points) Fill in the values of the following control signals to implement this new instruction

	LWPC	ALUSrc	MemToReg	RegWrite	MemRead	MemWrite	Branch
Value(0/1/X)	1