NATIONAL TSING HUA UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE
CS 4100: Computer Architecture
Spring 2024, Mid-term Examination

ID: ______________________ **Name:** ______________________

☆ **You must sign and hand in this problem sheet together with the answer sheet!**

☆ **You must list all the calculation work.**

1. (16%) Suppose a benchmark of 10 million (M) instructions for a specific instruction set architecture contains four instruction classes summarized as follows:

	Class A	Class B	Class C	Class D
Number of Instructions	1M	2.5M	4.5M	2M

The latest processor product of this instruction set architecture, P1, came with the following characteristics:

	Clock Frequency	CPI of Instruction Class			
		Class A	Class B	Class C	Class D
P1	2 GHz	2	4	6	3

(a) (4%) Calculate the CPI of P1.

(b) (6%) The architects want to design the next-generation processor, P2. Their first attempt is to increase the clock frequency from 2 GHz to 2.5 GHz. However, the CPI for each instruction class will increase by 1 accordingly (e.g., the CPI of Class A will become 2+1=3; the CPI of Class B will become 4+1=5, and so on and so forth). Calculate the performance ratio of P2 and P1 (You can list the expression and leave the result as a ratio, e.g., a/b, without calculating its final value.). Does P2 improve the performance? You must give your reasons.

(c) (6%) Based on (a), the compiler team attempts to improve the CPI further with a new compiler technique that can convert one Class C instruction to two Class A instructions. Calculate the new CPI with all Class C instructions converted. (You can list the expression and leave the result as a ratio, e.g., a/b, without calculating its final value.) Does this new compiler improve the CPI or not? You must give your reasons.

2. (10%) A tech journalist is comparing two computer products, A and B. After applying three popular benchmarks, the following table summarizes the execution time on the two computers.

	Computer A	Computer B
Benchmark 1	5 s	36 s
Benchmark 2	16 s	12 s
Benchmark 3	24 s	15 s

(a) (5%) Calculate the performance ratio of A:B using the geometric mean. Which one performs better?

(b) (5%) Explain why we prefer to use the geometric mean instead of the arithmetic mean.

3. (26%) Translate the following C code snippet into RISC-V RV64 assembly code to calculate the sum of negative numbers in an array **number[]** and also count the number of negative numbers. Assume that the register **x5** stores the starting memory address of **number[]**, the loop index **i** is stored in the register **x6**, the loop bound **n** is stored in **x7**, the **sum** is stored in **x8**, and the **count** is stored in **x9**. Hint: You may use additional general-purpose registers if necessary.

```
sum = 0;
count = 0
for (i=0; i<n; i++) {
  if (number[i] < 0) {
    sum += number[i];
    count++;
  }
}
```

(a) Assume that each element in **number[]** is a double-word integer. Complete the assembly code according to the description on the right.

Assembly Code	Description
	(a-1) (6%) Initialization (you must use **add** to initialize the variables)

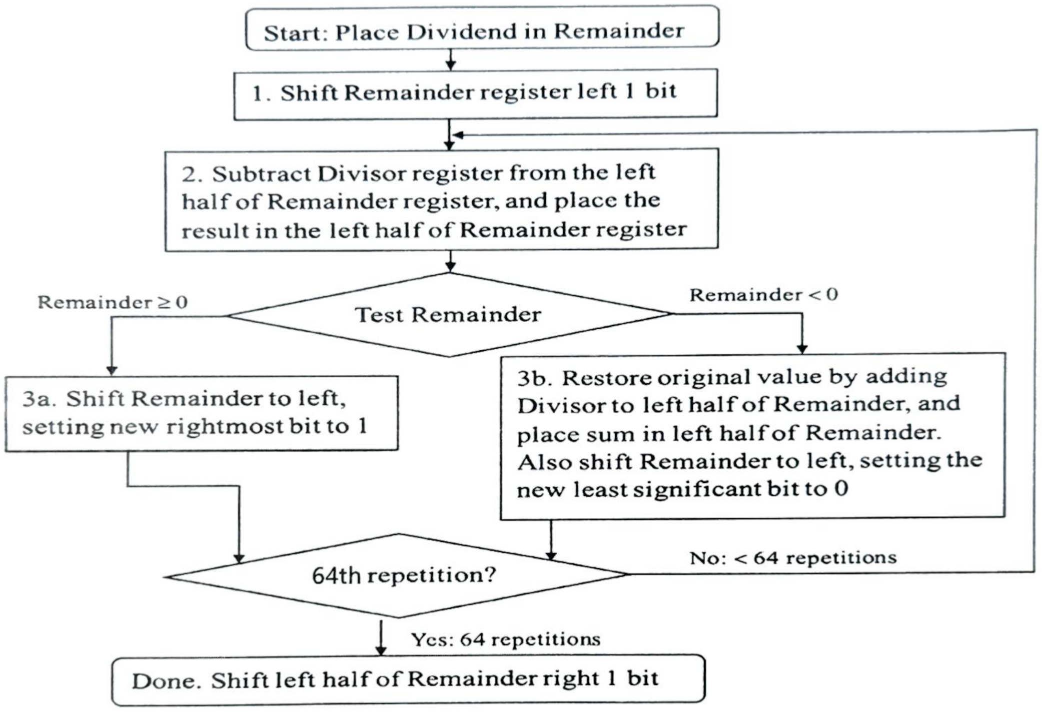
`LOOP_START:`
`    bge    x6, x7, LOOP_END`	

	(a-2) (6%) `x11` ← `number[i]`
	(a-3) (6%) Implement the `if` statement.

`UPDATE_i:`
 	
`LOOP_END:`
	(a-4) (4%) Increase the loop index `i` and jump back to `LOOP_START`.

(b) (4%) Explain how to revise the code in (a) if each element in **number[]** is a 32-bit integer instead.

(b) (3%) Now consider a non-restoring division algorithm that is different from the restoring algorithm in the sense that when the Remainder is negative in iteration i, $1 \leq i \leq 63$, (1) it does not add the Divisor to the left half of the Remainder to restore the original value but it still shifts the Remainder to the left and sets the least significant bit to 0 in step 3b, and (2) in iteration $i+1$, it adds the Divisor to the left half of the Remainder rather than subtracting the Divisor in step 2. Does this non-restoring division algorithm work correctly? Justify your answer.


Start: Place Dividend in Remainder
1. Shift Remainder register left 1 bit
2. Subtract Divisor register from the left half of Remainder register, and place the result in the left half of Remainder register
Remainder ≥ 0
Test Remainder
Remainder < 0
3a. Shift Remainder to left, setting new rightmost bit to 1
3b. Restore original value by adding Divisor to left half of Remainder, and place sum in left half of Remainder. Also shift Remainder to left, setting the new least significant bit to 0
64th repetition?
No: < 64 repetitions
Yes: 64 repetitions
Done. Shift left half of Remainder right 1 bit

8. (18%) Consider the IEEE 754 floating-point standard and its arithmetic operations.
   (a) (3%) Does 0x7F800000 represent a normalized number in the single precision format? Justify your answer.
   (b) (3%) Suppose the largest negative normalized number in the single precision format exactly represents the decimal number $A \times 2^B$. What are A and B?
   (c) (6%) Represent the decimal numbers 12.8125 and -3.625 in the single precision format and write your answers in hexadecimal.
   (d) (6%) Show all the steps to perform 12.8125 + (-3.625), assuming the two numbers are given in the single precision format.

4. (8%) Consider that an RISC-V RV64 instruction of branch-if-not-equal (**bne**) is executed with its program counter value as **0x0000000000008000**:

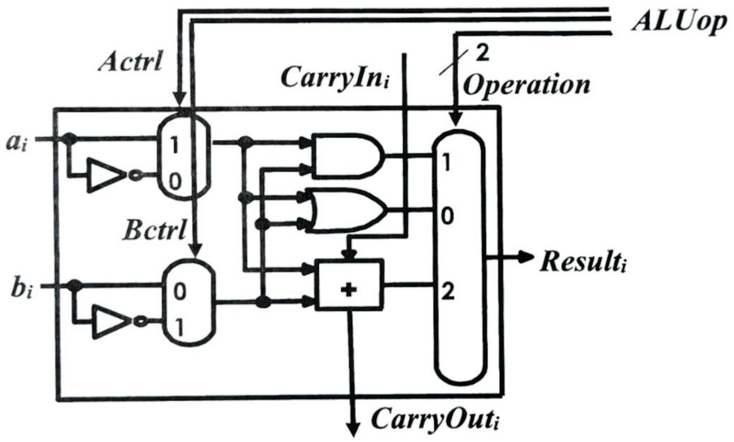
```
0x0000_0000_0000_8000: bne x5, x6, EXIT
```

(a) (4%) What is the bit width of the immediate offset stored in the binary format of the bne instruction? Explain how this offset is added to the PC (Program Counter) value to obtain the branch target address of EXIT.

(b) (2%) What is the largest absolute PC address of the label EXIT that this instruction can reach (in hexadecimal format)?

(c) (2%) What is the smallest absolute PC address of the label EXIT that this instruction can reach (in hexadecimal format)?

5. (6%) Consider an ALU which has two n-bit data inputs A = $a_{n-1}a_{n-2}\ldots a_0$, B = $b_{n-1}b_{n-2}\ldots b_0$, and one 4-bit control input ALUop. By ignoring the overflow detection, the ALU for each bit i, $0 \le i \le n-1$, is shown below. The ALU supports two's complement addition/subtraction by connecting CarryIn_0 to Bctrl.



Give the 4-bit value of ALUop (from left to right: Actrl, Bctrl, Operation) for each of the following operations: (a) A + B, (b) A – B, (c) A and B, (d) A or B, (e) A nor B, (f) A nand B.

6. (6%) Assume that the decimal value 100 is stored in the RISC-V register x10.

(a) (3%) How many different values of x11 could lead to overflow when executing the RISC-V instruction **add x12, x10, x11**?

(b) (3%) How many different values of x11 could lead to overflow when executing the RISC-V instruction **sub x12, x10, x11**?

7. (10%) The division algorithm (version 2) introduced in class for performing 64-bit division is a restoring one and shown below.

(a) (7%) Explain how to revise the algorithm such that the last step "Done. Shift left half of Remainder right 1 bit" can be removed without losing the correctness. To simplify the writing of your answer, you only need to describe what change or no change is made for each of steps 1, 2, 3a, and 3b.