Introduction to Computer Architecture - Midterm Exam (2022 Fall Semester)

Total 8 questions, 80 pts, Total 3 pages

1. CPI (10 pts, 5 pts each): Processor A's clock frequency is 2.5 GHz.

   a. The number of cycles for processing an instruction on Processor A is 2 cycles except for the following cases:
   - Load: 5 cycles
   - Multiply / Divide: 4 cycles
   - Taken branch / Jump: 6 cycles
   - Not taken Branch: 2 cycles

   On average, 20% of the executed branch instructions are taken branches. Program B executes the instructions in the given ratio in the table. **What is the average CPI** when processor A executes program B?

Arithmetic	Logic	Branch	Load	Store	Multiply	Divide	Jump	Shift
20%	15%	20%	10%	10%	5%	10%	5%	5%

		cycles	
arithmetic	20%	2	0.4
logic	15%	2	0.3
not taken branch	16%	2	0.32
taken branch	4%	6	0.24
load	10%	5	0.5
store	10%	2	0.2
multiply	5%	4	0.2
divide	10%	4	0.4
jump	5%	6	0.3
shift	5%	2	0.1
		Avg. CPI	**2.96**

   b. If program B executes 10 billion (10^{10}) instructions, **how long does it take to complete program B** on processor A? Ignore any additional delay except for the CPU time.

$$2.96 \times 10^{10} / 2.5 \times 10^{9} = 2.96 \times 4 = \mathbf{11.84}\ (\boldsymbol{seconds})$$

2. There are two types of CPUs, CPU ***A*** and CPU ***B***. CPU ***A***'s clock frequency is 4GHz and CPU ***B***'s clock frequency is 2GHz. Other than the difference in the clock frequency, they have the same specifications (i.e., same CPI). You want to build a processor chip that has multiple CPU cores. Due to the limited budget, you need to choose one of the following options: 1) Two CPU ***A***s, 2) One CPU ***A*** + Two CPU ***B***s

   The following are the programs that need to be executed on the processor.

Program	P1	P2	P3	P4	P5
Number of cycles	55 billion	40 billion	25 billion	20 billion	10 billion

   Which option would be faster, and by how much? (10 pts, 5 pts each) The following are the detailed conditions:

- Each program is executed once.
- All programs can be executed any of CPU A or CPU B
- Once a program is started on a CPU, the program completes execution on that CPU without interruption.
- All programs are independent. They can be executed in parallel (i.e., simultaneously) on different CPUs.
- Write the performance difference in fraction: e.g., A (or B) is faster by $^{x}/_{y}$ times.

Option 1): CPU A1: [P1, P4], CPU A2: [P2,P3,P5] : Each CPU executes 75B instructions @ 4GHz
Option 2): CPU A1: [P1, P4], CPU B1: [P2], CPU B2: [P3,P5] : The slowest CPU is CPU B1 that executes 40B instructions @ 2GHz

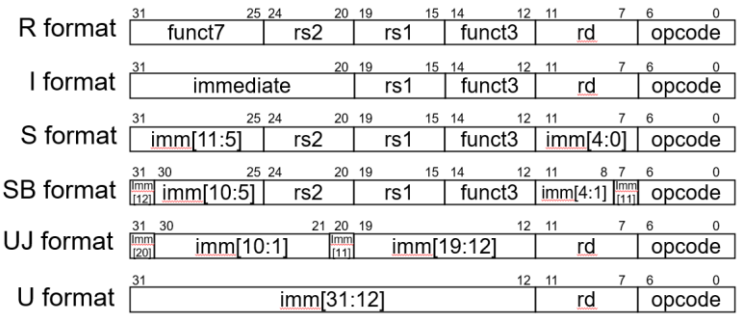
Option 1 is faster by $\left.{}^{40B}/_{2GHz}\right/{}_{75}/_{4GHz} = {}^{16}/_{15}$ **times**

3. Find the results of the following RISC-V Program Execution. **Write the values of the destination register after executing each instruction** one at a time. Assume you can freely read / write from any memory address. Write the values in hexadecimal**.** (10 pts, 1pts each)

`addi x1, x0, 0x100`	**x1: 0x100**
`slli x2, x1, 2`	**x2: 0x400**
`addi x3, x2, -75`	**x3: 0x3B5**
`sll x4, x2, x3`	**x4: 0x80000000**
`slt x5, x4, x0`	**x5: 0x1**
`lui x6, 0x10000`	**x6: 0x10000000**
`lui x7, 0xABCDE`	**x7: 0xABCDE000**
`sw x7, 0(x6)`	
`lw x8, 0(x6)`	**x8: 0xABCDE000**
`lh x9, 2(x6)`	**x9: 0xFFFFABCD**
`lbu x10, 3(x6)`	**x10: 0x000000AB**

4. Binary representation (10 pts, 5 pts each). Please use the following opcode/funct table and the RISC-V binary encoding formats. The codes are written in binary (base 2)

Instruction	Opcode	Funct3	Funct7
add	0110011	000	0000000
sub	0110011	000	0100000
sll	0110011	001	0000000
slt	0110011	010	0000000
lw	0000011	010	-
sw	0100011	010	-
lui	0110111	-	-
jalr	1100111	000	-
addi	0010011	000	-
slti	0010011	010	-
slli	0010011	001	-



a. Convert the following RISC-V assembly instruction **into the machine code**. Write the 32-bit machine code in hexadecimal number.

`sw x3, 0x693(x10)` → **0x683529A3**

b. Convert the following machine code written in hexadecimal number **into the RISC-V assembly instruct**ion.

0x407209B3 → **sub x19, x4, x7**

5. For each of the following five instructions, indicate whether it is a **valid** or **invalid** RISC-V RV32I instruction. An instruction that is not supported by the RISC-V RV32I CPU (e.g., pseudoinstructions) should be indicated as invalid instruction. (10 pts, 2pts each)

   a. xori x0, x1, 10 → **valid**

   b. lm x1, -3(x4) → **invalid**

   c. sla x10, x20, x30 → **invalid**

   d. and x0, x0, x0 → **valid**

   e. srli x31, x31, 0 → **valid**

6. New ISA (10 pts, 2pts each). Suppose you want a more powerful ISA than the baseline RISC-V RV32I. You want to design an ISA that supports following features

   - 128 general-purpose registers (x0-x127)
   - Opcode and funct fields are the same as the current RV32I

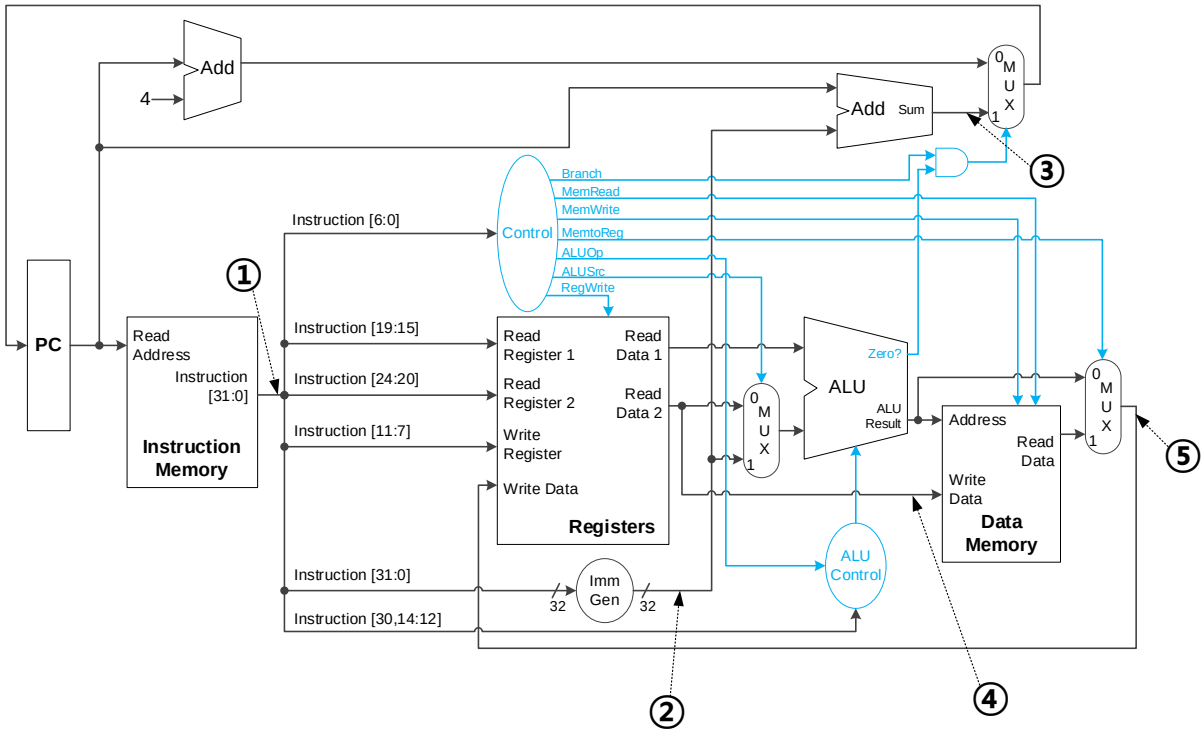
   a. **How many bits are required** to address the index of a register? → **7bits**
   b. **How many bits are required** to express the new R-format arithmetic instruction? → **38bits**
   c. What is the **minimum number of bytes** to represent the new R-Format arithmetic instruction? → **5 bytes**
   d. Suppose all instructions are represented using the number of bytes you answered in the previous question. For now, let's not worry about the non-power-of-two bytes per instruction. For an I-format instruction, **how many immediate bits are available**? → **16 bits**
   e. The instructions will be stored in memory consecutively (i.e., no empty bytes between instructions). For this new ISA, the supported branch offset range is from -2^x to $+2^y - 1$. **Write the values of x and y.** → **x: 15 y:15**
   The new ISA uses 5-byte instructions, and this means that the address of an instruction does not always end with "0". The branch offset needs to indicate the lowest bit as well. Since we can use 16 bits of immediate value, the branch offset range will be the same as normal 16-bit 2's complement singed integer range, -2^{15} ~ $+2^{15}-1$.

7. The single-cycle CPU we studied in class can be easily extended to support I-format arithmetic instructions. When the CPU is processing the following RISC-V instruction loaded from memory address 0x1000, **write the values at the indicated datapath locations** in the figure (you need to write a total of 5 values and write in hexadecimal). (10 pts, 2pts each)

addi x1, x2, -48

Currently, the register file has the following values

x1:	0xBAD	x3:	0xFEED	x15:	0xFACE	x17:	0xDEAD
x2:	0xDECAF	x4:	0xBAAD	x16:	0xCAFE	X18:	0xBEEF



1) 0xFD010093

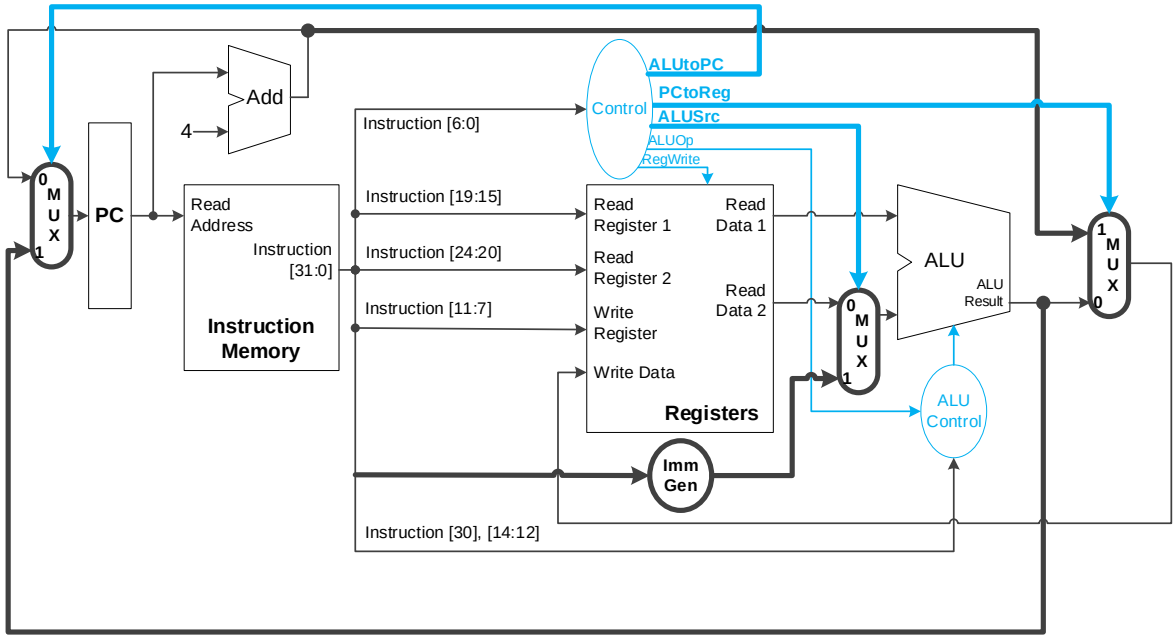
2) 0xFFFFFFD0

3) 0xF30

4) 0xCAFE

5) 0xDEC7F

8. Design a CPU that supports the R-format instructions (add, sub, and, or) and the JALR instruction. Do not need to support the lw, sw, and beq instructions. The JALR instruction is using the I-format machine code. **Draw or explain the required datapath and control signals** to support both instruction types. If you add control signals, indicate their value when the CPU is executing the JALR instruction. Do not need to exactly specify the bit ranges (i.e., don't need to write "Instruction [11:7]"). (10 pts)



You don't need to draw a perfect datapath. However, the following features must be clearly explained either in drawing or in written explanation.

1) **A datapath that connects PC+4 to the "write data" port of the register file and control signals to distinguish this new datapath from the original datapath from the ALU**
2) **A datapath that connects the output of the ALU to the input of the PC register and the control signals to distinguish this new datapath from the original PC+4 input.**