CSE 112 Computer Organization 2025: Mid-Semester Exam

Total Time: 60 Minutes **Maximum: 30 Points**

Instructions:

1. **Write all the assumptions clearly, if any. *Only reasonable assumptions will be considered if any ambiguity is found in the question*.**
2. **No electronic gadgets apart from calculators are allowed.**

ISA description:

First Block Instructions	Description	Operation
addi rd,rs, imm[11:0]	Add immediate	rd = rs + sext(imm[11:0])
sub rd, rs1, rs2	Subtract registers	rd = rs1 - rs2
add rd, rs1, rs2	Add registers	rd = rs1 + rs2
xor rd, rs1, rs2	Perform XOR boolean operation	rd = rs1 xor rs2
slt rd, rs1, rs2	Set Less than	rd = 1. If signed(rs1) < signed(rs2) else rd=0
sltu rd,rs1,rs2	Set Less than unsigned	rd = 1. If unsigned(rs1) < unsigned(rs2) else rd=0
lw rd,#imm[11:0](rs1)	Load Word	rd=mem(sext(rs1+imm))
sw rd,#imm[11:0](rs1)	Store Word	mem(sext(rs1+imm))=rd
beq rs1, rs2, imm[12:1]	Branch if equal	PC = PC + sext({imm[12:1],1'b0}) if rs1 == rs2
beq x0,x0,#0	Branch if equal	**HALT**

Second Block Instructions	Description	Operation
bne rs1, rs2, imm[12:1]	Branch if not equal	PC = PC + sext({imm[12:1],1'b0}) if rs1 =! rs2
bltu rs1,rs2,imm[12:1]	Branch if less than unsigned	PC = PC + sext({imm[12:1],1'b0}) If unsigned(rs1) < unsigned(rs2)
blt rs1,rs2,imm[12:1]	Branch if Less than	PC = PC + sext[imm[12:1],1'b0}] If signed(rs1) < signed(rs2)
bge rs1,rs2, imm[12:1]	Branch if Greater than or Equal	PC= PC + sext({imm[12:1],1'b0}) If signed(rs1) > signed(rs2)
bgeu rs1,rs2, imm[12:1]	Branch if Greater than or equal unsigned	PC= PC + sext({imm[12:1],1'b0}) If unsigned(rs1) > unsigned(rs2)

Note: The ISA is a subset of RISC-V ISA. Each instruction is of 32-bit size, we have 32 registers as standard from x0 to x31 with x0 hardwired to zero. The **symbol #** indicates an immediate value in the decimal format.

Q1. [CO2] Assembly Execution:

We need to answer how many times the character ***** will be printed and what is the value in registers {x1,x2} after execution of every instruction. **[5 points]**

Assembly Program	**Instruction:** OUT * Explanation: This will print ***** at the output. And, get to the next line.
xor x2,x2,x2	
addi x2,x2,#10	
LABEL: out *	
sltu x1,x0,x2	
sub x2,x2,x1	
bne x2,x0,**LABEL**	
HERE: beq x0,x0,**HERE**	

Solution:

The character ***** will be printed 10 times. [1 Point]

Instruction	1st	2nd	3rd	9th Iteration	Last Iteration
xor x2,x2,x2	x1=xx,x2=0				
addi x2,x2,#10	x1=xx,x2=10				

LABEL: out *	**Output *** x1=xx,x2=10	**Output *** x1=1,x2=9	…	**Output *** x1=1,x2=2	**Output *** x1=1,x2=1
sltu x1,x0,x2	x1=1,x2=10	x1=1,x2=9		x1=1,x2=2	x1=1,x2=1
sub x2,x2,x1	x1=1,x2=9	x1=1,x2=8		x1=1,x2=1	x1=1,x2=0
bne x2,x0,**LABEL**	x1=1,x2=9	x1=1,x2=8		x1=1,x2=1	x1=1,x2=0
HERE: beq x0,x0,**HERE**					x1=1,x2=0

a) If the number of iterations is incorrect, and the logic is correct, deduct one point out of four.
b) If the student begins with wrong initialization, and the logic is correct, deduct two points out of four.

Q2. [CO2] Assembly Instruction Encoding: **[10 points]**

	Pseudo Code

We need to convert the pseudo code given on the right side into an assembly program using the above-mentioned ISA. And, we do need to report the final value of the register **E** (corresponding mapped register) after execution of every instruction**.**	A=20, B=11, D=1, E=1
	while(A!=B)
	{
	A=A-D
	E=E+1
	}
	HALT

Solution:
The below-mentioned assembly code is one possible encoding; other solutions are also possible.
Let variable A,B,D,E are mapped to x1,x2,x3,x4 respectively.

Assembly Code	1st	2nd	3rd	8th	Last Iteration
addi x1,x0,#20	A=20				
addi x2,x0,#11	B=11				
addi x3,x0,#1	D=1				
addi x4,x0,#1	E=1 (x4=1)				
bne x2,x1,**LABEL2**	x4=1				
beq x0,x0,#0	x4=1				
LABEL2: sub x1,x1,x3	Code inside the loop. x4=1,x1=19	Inside loop. x4=2,x1=18	…	Inside loop x4=8,x1=12	Inside loop x4=9,x1=11
addi x4,x4,#1	x4=2	x4=3	…	x4=9	x4=10
bne x2,x1,**LABEL2**	x4=2	x4=3	…	x4=9	x4=10
beq x0,x0,#0					x4=10

The loop will execute 9 times. So, the final value of E is 10.
Assembly Encoding

a) No points just for correct initialization in assembly code.
b) If the Assembly encoding is functionally and logically correct, award five points out of five.
c) If the Assembly encoding is functionally and logically correct, and the number of loops are wrong, award four points out of five.
d) If the Assembly code encoding is functionally incorrect, award zero points.
e) If the Assembly code encoding is functionally correct, and some other instructions out of the specified instructions are used, award 2.5 points out of 5.

Assembly Code Optimization:

a) If the number of branches used apart from **HALT** are two or less than two, award 2 points out of 2.
b) If the number of branches used apart from HALT are three, award 1 point out of 2.
c) Else award 0 points out of 2.

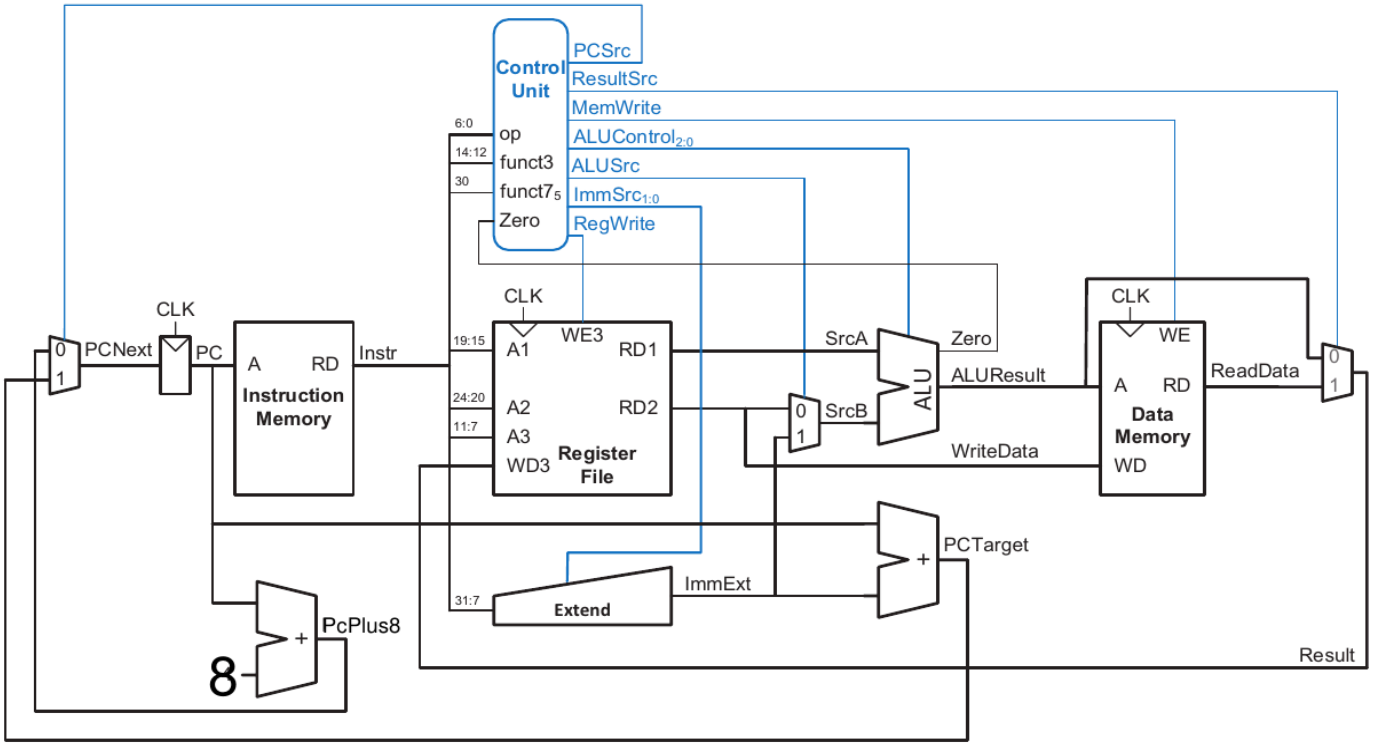
Assembly Execution:

a) If the final value of register E mapped address is correct and the number of iterations are also correct, award 3 points out of 3 points.
b) If the logic is correct (although can be evaluated from assembly code) and the final value and number of iterations are incorrect, award 1 point out of 3.
c) Else award 0 points out of 3.

Q3. [CO3, CO2] Student_A designed a processor microarchitecture for the instructions mentioned in the **First block**. **Student_A** intentionally changed the default PC update hardware (as depicted in the below block diagram). The designed processor uses a 32-bit data width bus and a byte addressable instruction memory. **Student_B** is **unaware** about the changes made by **Student_A** in the PC update hardware. **Student_B** wrote the below-mentioned assembly code to execute on the processor designed by **Student_A**.

Assembly Code:

```
addi x2,x0,#11
add  x3, x2, x0
sub x4, x5, x2
add x1, x3, x4
addi x1, x3, #4
```



Answer the following questions:

a) List out the instructions executed by the microarchitecture and the value of the registers(x0 to x5) after each instruction.
b) Does the microarchitecture designed by **Student_A** align with the intent of the code written by **Student_B**. Give reason.
c) How will **Student_A** rectify the microarchitecture to solve the issue?

Assume that the **initial** values of registers are as x1→1;....; x31→31. **[3+2+1=6 points]**

Ans:
a)

addi x2,x0,#11
sub x4, x5, x2
addi x1, x3, #4
	x2 = 11; x0 =0
	x4 = -6; x5 = 5; x2 = 11
	x1 = 7; x3 =3

b) No, since the processor follows a byte-addressable scheme and PC is added by 8 instead of 4, we are skipping some instructions, resulting into a different values instead of correct one.
c) we can rectify it by making the PC + 4 instead of 8

Q4. [CO3, CO2] If the microarchitecture described in the above question has the following control signal.

1. **RegWrite**: If the instruction writes into the register, the value is 1, else it is 0.
2. **ALUSrc:** When the value is 0, the value of register 2 is read, else the extended immediate value is read.
3. **MemWrite:** If the instruction writes into the data memory, the value is 1, else it is 0.
4. **ResultSrc**: When the value is 0, output from the execute stage is written back. When 1, output from the memory stage is written back.
5. **ALUControl**: 000 for R-type ADD instruction, 001 for R-type SUB instruction, 010 for instructions using immediate values in any form, 011 for instructions implementing boolean operations.

Assuming the instructions listed below are performed, determine the value of the control signals. Use **"x"** as a don't care value wherever necessary. Answer in the form of the given table: **[9 points]**

	RegWrite	ALUSrc	MemWrite	ALUControl	ResultSrc	Marks
add x3,x2,x1	1	0	0	000	0	0.2x5=1
lw x2, #4(x3)	1	1	0	010	1	0.4x5=2
sw x1, #8(x5)	0	1	1	010	x	0.4x5=2
addi x2, x1, #8	1	1	0	010	0	0.2x5=1
sub x1,x2,x3	1	0	0	001	0	0.2x5=1
beq x1, x2, #0	0	0	0	001	x	0.4x5=2