QUIZ 1 RUBRICS (SET A &B)

[CO1] Q1. INSTRUCTION DECODING: [10 points]

Consider the subset of R-type instruction of RISC V ISA given below:

R-type instruction:

Format:

funct7	rs2	rs1	funct3	rd	opcode
[31:25]	[24:20]	[19:15]	[14:12]	[11:7]	[6:0]

funct7	rs2	rs1	funct3	rd	opcode	Semantics	Explanation
[31:25]	[24:20]	[19:15]	[14:12]	[11:7]	[6:0]		
0000000			000		0110011	add rd, rs1, rs2	rd = rs1 + rs2
0100000			000		0110011	sub rd, rs1 rs2	rd = rs1 - rs2.
0000000			010		0110011	slt rd, rs1, rs2	Set less than rd=1 only if signed(rs1) < signed(rs2)
0000000			011		0110011	sltu rd, rs1, rs2	Set less than unsigned rd=1 only if unsigned(rs1) < unsigned(rs2)
0000000			100		0110011	xor rd, rs1, rs2	***Bitwise Xor*** rd=rs1 ⊕ rs2
0000000			110		0110011	or rd, rs1, rs2	***Bitwise Or*** rd=rs1 | rs2
0000000			111		0110011	and rd, rs1, rs2	***Bitwise And*** rd=rs1 & rs2

We have 32 registers each of size 32-bit, these are uniquely addressed using 5-bit, we shall name as x0, x1, x2,..., x31 and the addressing is done in such a way that the address 0x0 → x0; 0x1 →x1; 0x2 →x2 and so on

Answer the following question:

A. Convert the following instruction to binary format

(SET A Questions)

add x2, x3, x4

sub x7, x6, x5

xor x1, x0, x1

and x24, x12, x10

sltu x6, x1, x2

(SET B Questions)

add x2, x3, x4

and x24, x12, x10

xor x1, x0, x1

sub x7, x6, x5

sltu x6, x1, x2

B. How many maximum number of instructions can this given ISA support? [consider funct7; funct3; opcode]
C. What are the minimum and maximum signed and unsigned values that can be stored in any register?
D. If this ISA supports an address space of 256 Kilobytes with byte-addressable memory. What are the minimum bits required to represent a memory address uniquely?

Ans:

A. Set A answer:- [1x5 = 5 points]

add x2, x3, x4	0000000 00100 00011 000 00010 0110011
sub x7, x6, x5	0100000 00101 00110 000 00111 0110011
xor x1, x0, x1	0000000 00001 00000 100 00001 0110011
and x24, x12, x10	0000000 01010 01100 111 11000 0110011
sltu x6, x1, x2	0000000 00010 00001 011 00110 0110011

Set B answer [1x5 = 5 points]

add x2, x3, x4	0000000 00100 00011 000 00010 0110011
and x24, x12, x10	0000000 01010 01100 111 11000 0110011
xor x1, x0, x1	0000000 00001 00000 100 00001 0110011
sub x7, x6, x5	0100000 00101 00110 000 00111 0110011
sltu x6, x1, x2	0000000 00010 00001 011 00110 0110011

B. It can support 2^(7+3+7) i.e 2^17 instructions [1 point]
C. As each register is of 32 bit. [1x2 = 2 points]
   a. The maximum positive value is 2^(32-1) - 1. Or 2^{32-1} - 1.
   b. The minimum value or maximum negative value is - 2^(32-1).
D. For 256 KiloBytes with a byte-addressable location, the number of bits can be calculated as:

$$2^n = 256 \times 1024 \Rightarrow 2^n = 2^8 \times 2^{10} \Rightarrow 2^n = 2^{18} \Rightarrow n = 18$$ [2 points]

[CO2]

Q2. Instruction Execution and Program Counter Flow [10 points]

A Processor is designed based on the ISA mentioned in question1, with each instruction encoded in 32 bits. The processor follows a byte addressable memory scheme and has 32 bits read data width. The instructions of the assembly program are stored in the contiguous memory locations.

Assume all registers are initialized with their index numbers (x0 with 0, x1 with 1, x2 with 2, etc.). We need to execute the assembly program mentioned below under the three program counter-update mechanisms. Provide the values of the registers (x6, x7, x8, x9) during the execution of the program under these three mechanisms.

(a) PC = PC+4
(b) PC = PC+8
(c) PC = PC + 12 (SET B)
(d) PC = PC+16 (SET A)

add x5,x6,zero
sub x7,x6,x6
xor x9,x9,x1

sub x9,x9,x1
or x7, x9, x1
HALT
HALT
HALT
HALT

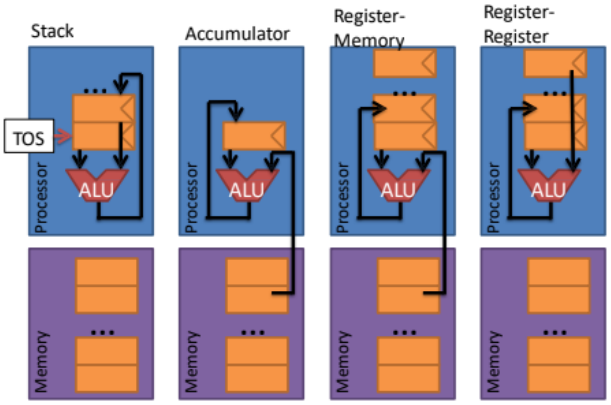
Solution:
N.B. This Address mapping is just for reference only and will not be considered in the evaluation

Address	Instruction	a) PC=PC+4	b) PC=PC+8	c) PC=PC+12	d) PC=PC+16
0x0000_0000	add x5,x6,zero	x5=6	x5=6	x5=6	x5=6
0x0000_0004	sub x7,x6,x6	x7=0	SKIP	SKIP	SKIP
0x0000_0008	xor x9,x9,x1	x9=8	x9=8	SKIP	SKIP
0x0000_000C	sub x9,x9,x1	x9=7	SKIP	x9=8	SKIP
0x0000_0010	or x7, x9, x1	x7=7	x7=9	SKIP	x7=9
0x0000_0014	HALT	STOP	SKIP	SKIP	SKIP
0x0000_0018	HALT	No Further execution	STOP	STOP	SKIP
0x0000_001C	HALT		No Further Execution	No Further Execution	SKIP
0x0000_0020	HALT				STOP

Marking Scheme:

a) 2.5 = 0.5 * 5
b) 2.5=0.75+0.75+1
c) PC=PC+12
    i) 5=2.5+2.5
d) PC=PC+16
    i) 5=2.5+2.5

[CO1, CO2] Q3. Machine Model [7 points]
For the machine model shown below, write the assembly to perform A=C*(B+D).
Note that the ALU can do mul, add operations, and you have registers R1, R2, and R3 in the register model.



Ans: [1.75 points for each model] [1.75 x 4 = 7 points]

1. Stack model

```
PUSH C    ; Push C onto stack
PUSH B    ; Push B onto stack
PUSH D    ; Push D onto stack
ADD       ; B + D (Result at top of stack)
MUL       ; C * (B + D) (Result at top of stack)
POP A     ; Store result in A
```

2. Accumulator model

```
load B          ; Load B into accumulator
add D           ; Acc = B + D
mul C           ; Acc = C * (B + D)
store A         ; Store result in A
```

3. Register-Memory model

```
load R1, B      ; Load B into R1
add R1, D       ; R1 = B + D
mul R1, C       ; R1 = C * (B + D)
Store A, R1 ; Store result in A
```

4. Reg-Reg model

```
load R1, B              ; load B into R1
load R2, D              ; load D into R2
Add R3, R1, R2          ;  R3 = R1 + R2
Load R1, C              ; load C into R1
Mul R3, R1, R3          ; R3 = R1 * R3
Store R3, A             ; store R3 into A
```

FOR SET B A= C+(B*D)

1. Stack model

```
PUSH C     ; Push C onto stack
PUSH B     ; Push B onto stack
PUSH D     ; Push D onto stack
MUL        ; B * D (Result at top of stack)
ADD        ; C + (B * D) (Result at top of stack)
POP A      ; Store result in A
```

2. Accumulator model

```
load B          ; Load B into accumulator
mul D           ; Acc = B * D
add C           ; Acc = C + (B * D)
store A         ; Store result in A
```

3. Register-Memory model

```
load R1, B      ; Load B into R1
mul R1, D       ; R1 = B * D
add R1, C       ; R1 = C +(B * D)
Store A, R1 ; Store result in A
```

4. Reg-Reg model

```
load R1, B              ; load B into R1
load R2, D              ; load D into R2
mul R3, R1, R2          ;  R3 = R1 * R2
Load R1, C              ; load C into R1
add R3, R1, R3          ; R3 = R1 + R3
Store R3, A             ; store R3 into A
```

Note: There are multiple ways to answer this question.

[CO1] Numerical [3 points]

Propose logical conditions to detect Overflow in 2's Complement Addition and make a suitable table.

Ans: [1+1+1 points]

If suppose we have to add two numbers in binary of "n" bits each, A + B i.e., and let the MSBs for the Summands be An & Bn and the MSB of SUM be Sn
Now, overflow occurs during the following conditions:
1. Two negative numbers are added, and an answer comes positive or
2. Two positive numbers are added, and an answer comes as negative
Summarizing in a table:

An	Bn	Sn	Overflow
0	0	1	1
1	1	0	1
0	1	x	0
1	0	x	0