CITY UNIVERSITY OF HONG KONG

Course code & title	:	CS2115 Computer Organization
Session	:	Semester A 2020/21
Time allowed	:	Two hours

This paper has 26 pages (including this cover page).

1. This paper consists of 6 questions.
2. Answer ALL questions.
3. Write your answers in the space provided.

Student ID: ___________________________

Seat number: ___________________________

1	2	3	4

5	6		Total

*This is a **closed-book** examination.*

No materials or aids are allowed during the whole examination. If any unauthorized materials or aids are found on a candidate during the examination, the candidate will be subject to disciplinary action.

<u>CS2115 Final Exam 2020/21 Semester A</u>

<u>Academic Honesty</u>

I pledge that the answers in this examination are my own and that I will not seek or obtain an unfair advantage in producing these answers. Specifically,

❖ *I will not plagiarize (copy without citation) from any source;*

❖ *I will not communicate or attempt to communicate with any other person during the examination; neither will I give or attempt to give assistance to another student taking the examination; and*

❖ *I will use only approved devices (e.g., calculators) and/or approved device models.*

❖ *I understand that any act of academic dishonesty can lead to disciplinary action.*

I pledge to follow the Rules on Academic Honesty and understand that violations may lead to severe penalties.

Student ID:

Name:

Sign:

<u>CS Departmental Hotline (phone, whatsapp, wechat)</u>
+852 6375 3293

Problem 1 (20 marks)

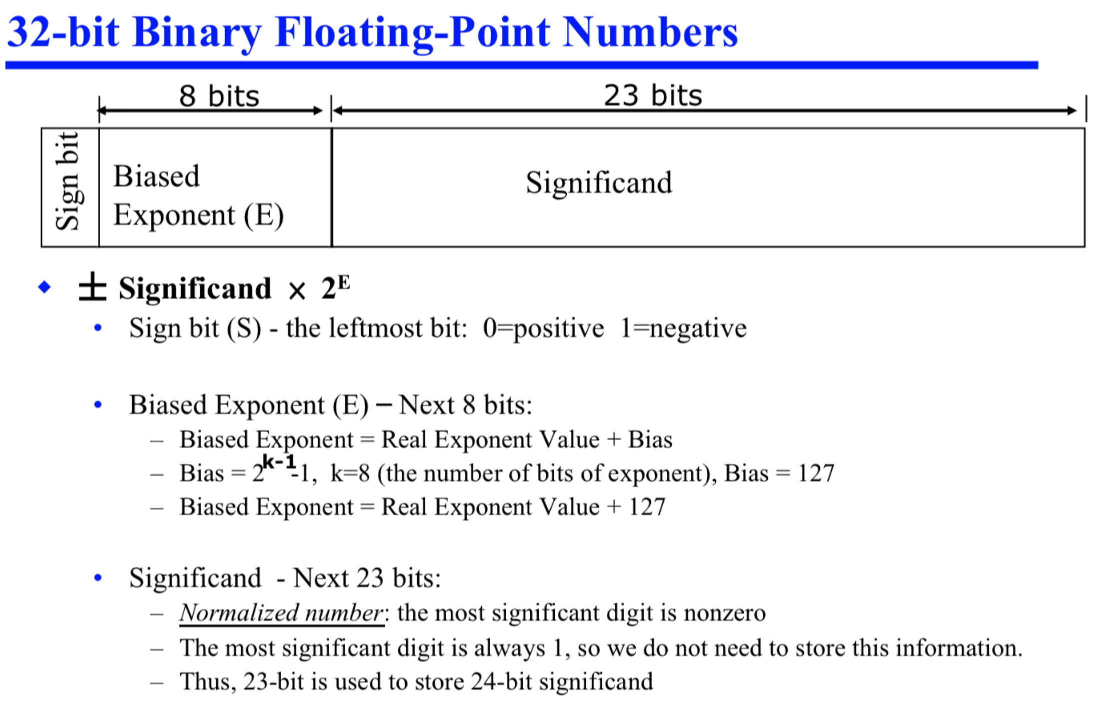
1.a) For the decimal number $(61.25)_{10}$, convert it to the **binary**, **octal** and **hexadecimal** forms, respectively. (3 marks)

1.b) Write the following pairs in the **5-bit** 2's complement form. Calculate the binary addition of each pair and write down the **5-bit** binary form summation result. State whether there is an overflow in each case. (6 marks)

1.b.1) $(14)_{10}$ and $(-9)_{10}$

1.b.2) $(6)_{10}$ and $(14)_{10}$

1.c) Write $(-61.25)_{10}$ in the 32-bit binary floating point format. The format is given as follows. Hint: you can reuse some of the results from question 1.a). (4 marks)



32-bit Binary Floating-Point Numbers

8 bits
23 bits
Sign bit
Biased Exponent (E)
Significand

- **± Significand × 2^E**
  - Sign bit (S) - the leftmost bit: 0=positive 1=negative
  - Biased Exponent (E) – Next 8 bits:
    - Biased Exponent = Real Exponent Value + Bias
    - Bias = 2^{k-1}-1, k=8 (the number of bits of exponent), Bias = 127
    - Biased Exponent = Real Exponent Value + 127
  - Significand - Next 23 bits:
    - *Normalized number*: the most significant digit is nonzero
    - The most significant digit is always 1, so we do not need to store this information.
    - Thus, 23-bit is used to store 24-bit significand

Figure 1: This is a slide from the lecture note for your reference

1.d) Let us look at the format of the 16-bit floating point number with 1-bit sign, 5-bit biased exponent, and 10-bit significand (or fraction). The calculation of the biased exponent and significand is the same as the 32-bit/64-bit floating point number, except the difference in bit-length. (7 marks)

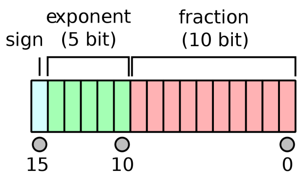


Figure 2: This is the format of the 16-bit binary floating point number.

1.d.1) What is the bias of the 5-bit biased exponent? (1 mark)

1.d.2) Represent $(-61.25)_{10}$ in the 16-bit binary floating point format. (2 marks)

1.d.3) What are the **largest positive number** and the **smallest positive number** that this 16-bit binary floating point number format can represent (You can represent them in the form of $a \times 2^b$)? (4 marks)

Problem 2 (25 marks)

We will design a digital circuit to convert a given 3-bit natural binary code to the respective Gray code. Let A, B and C be the left-most bit, the central bit and the right-most bit of the natural binary code, respectively; and let X, Y and Z be the left-most bit, the central bit and the right-most bit of the Gray code, respectively.

Binary	**Gray Code**
000	000
001	001
010	011
011	010
100	110
101	111
110	101
111	100

2.a) Get the function between the Gray code and the natural binary code in the canonical SOP form, i.e., express X, Y, Z using A, B, and C. (6 marks)

2.b) Use K-map to get the simplified function expression of X, Y, and Z. (3 marks)

2.c) Draw the circuit of the simplified functions in 2.b). (3 marks)

2.d) Draw the circuit again using only XOR gates, with no more than 3 XOR gates. (3 marks)

2.e) Design the 3-bit Gray code counter with FSM. (10 marks)

Design a 3-bit Gray code counter FSM with no inputs and three outputs. Suppose Gray code bits X, Y, Z are stored in 3 D flip flops. Following clock cycles, the output of the FSM will iterate in an order '000→001→011→010→110→111→101→100→000→001→ . . . '.

2.e.1) Draw the state transition diagram for the 3-bit Gray code counter; clearly label the state and the output. (3 marks)

2.e.2) Draw the state transition table for the 3-bit Gray code counter. (3 marks)

2.e.3) Draw the circuit for the 3-bit Gray code counter (Hint: Use K-map to simplify the logic expression and implement the circuits.) (4 marks)

Problem 3 (12 marks)

3.a) Given the following D latch and D flip-flop, complete the timing diagram below for D latch and D flip-flop. The initial values of both states of Q (D latch) and Q (D flip-flop) are 0 (low level). (4 marks)

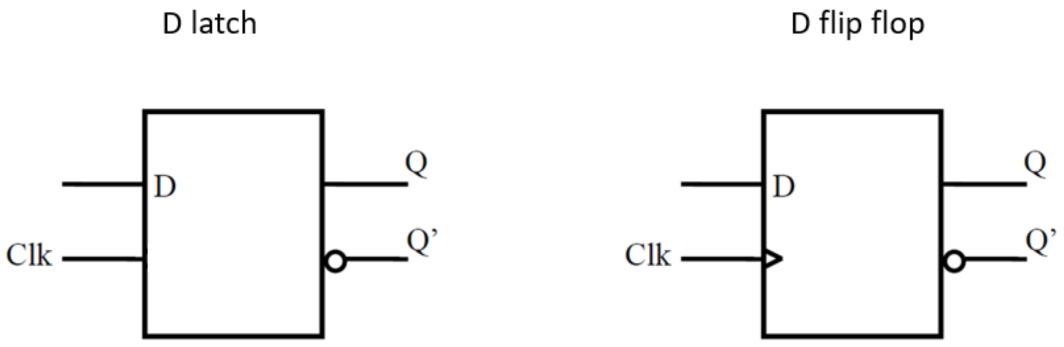
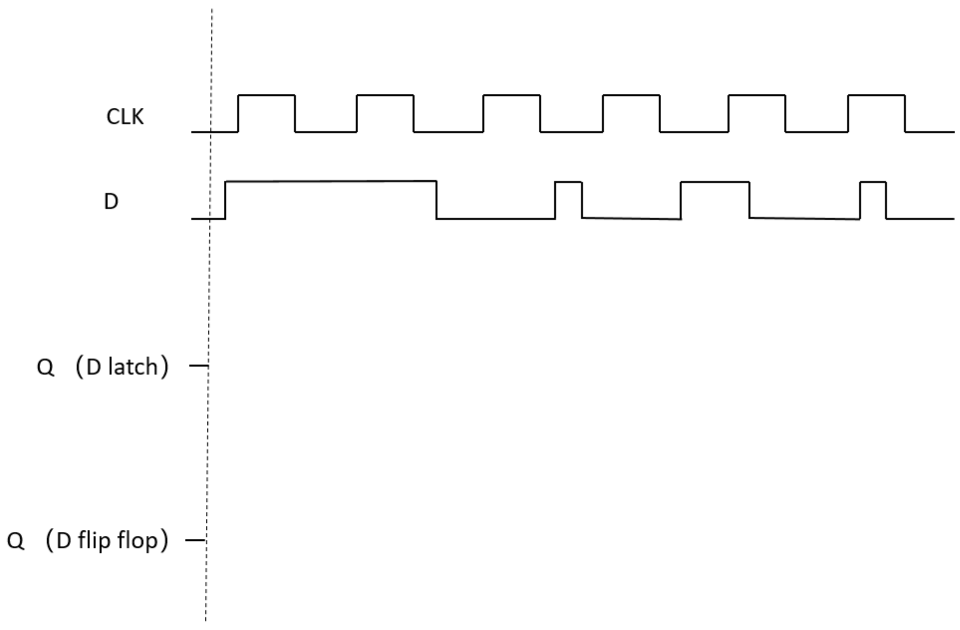
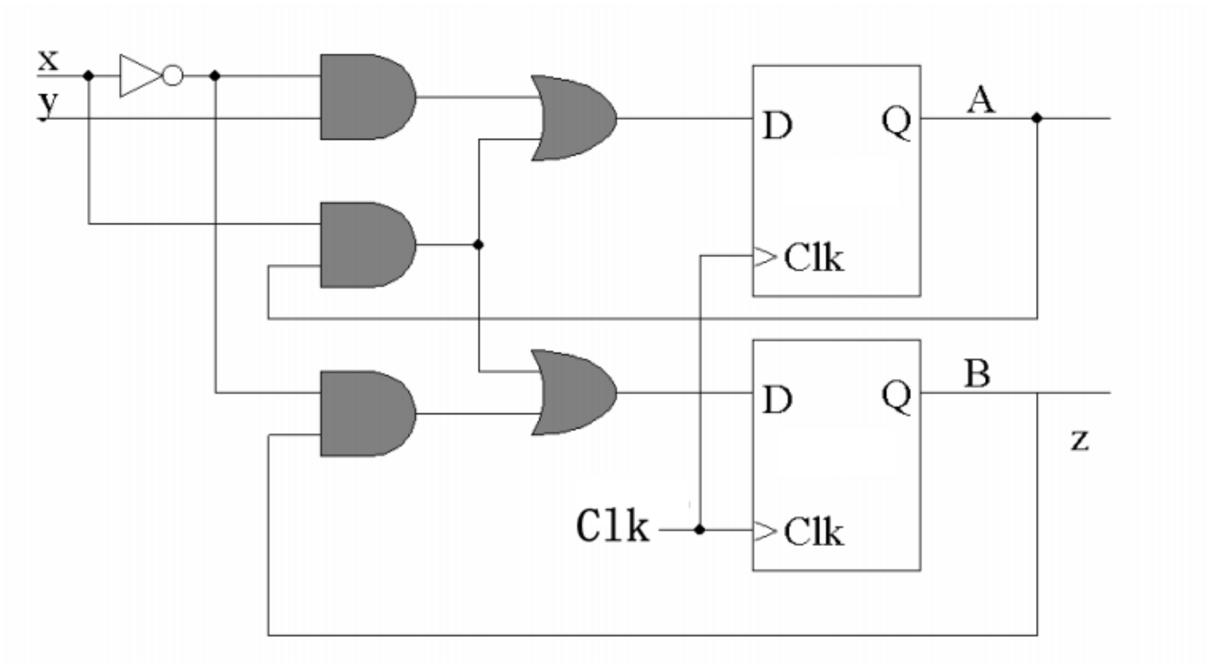


Figure 3: D latch and D flip-flop.



3.b) Consider the following sequential logic circuit with two D flip-flops, A and B; two inputs, x and y; and one output z. (8 marks)



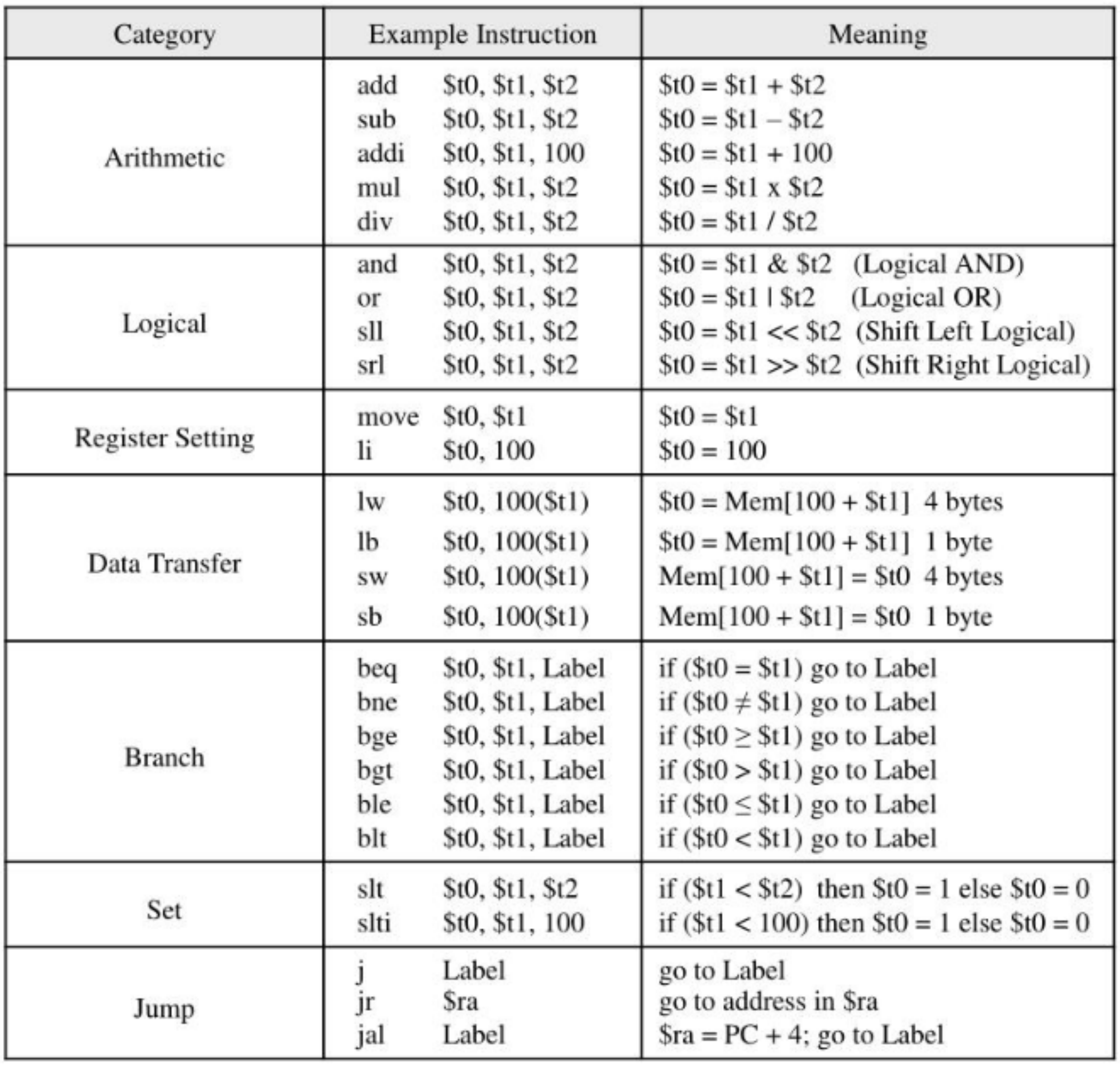
3.b.1) Based on the circuit, please write down the state equations, i.e., express $z(t)$, $A(t + 1)$, and $B(t + 1)$ using $A(t)$, $B(t)$, $x(t)$, and $y(t)$. (4 marks)

3.b.2) Based on the circuit and equations, please complete the following state table for this circuit. Fill your answers in the table. (4 marks)

Present state		**Inputs**		**Next State**		**Output**
A(t)	B(t)	x(t)	y(t)	A(t+1)	B(t+1)	z(t)
0	0	0	0			
0	0	0	1			
0	0	1	0			
0	0	1	1			
0	1	0	0			
0	1	0	1			
0	1	1	0			
0	1	1	1			
1	0	0	0			
1	0	0	1			
1	0	1	0			
1	0	1	1			
1	1	0	0			
1	1	0	1			
1	1	1	0			
1	1	1	1			

Problem 4 (15 marks)

This question is about the ISA, MIPS assembly language, and how CPU executes an instruction. We have attached the MIPS instruction set for your reference.



Category	Example Instruction	Meaning
Arithmetic	add $t0, $t1, $t2	$t0 = $t1 + $t2
	sub $t0, $t1, $t2	$t0 = $t1 – $t2
	addi $t0, $t1, 100	$t0 = $t1 + 100
	mul $t0, $t1, $t2	$t0 = $t1 x $t2
	div $t0, $t1, $t2	$t0 = $t1 / $t2
Logical	and $t0, $t1, $t2	$t0 = $t1 & $t2 (Logical AND)
	or $t0, $t1, $t2	$t0 = $t1 | $t2 (Logical OR)
	sll $t0, $t1, $t2	$t0 = $t1 << $t2 (Shift Left Logical)
	srl $t0, $t1, $t2	$t0 = $t1 >> $t2 (Shift Right Logical)
Register Setting	move $t0, $t1	$t0 = $t1
	li $t0, 100	$t0 = 100
Data Transfer	lw $t0, 100($t1)	$t0 = Mem[100 + $t1] 4 bytes
	lb $t0, 100($t1)	$t0 = Mem[100 + $t1] 1 byte
	sw $t0, 100($t1)	Mem[100 + $t1] = $t0 4 bytes
	sb $t0, 100($t1)	Mem[100 + $t1] = $t0 1 byte
Branch	beq $t0, $t1, Label	if ($t0 = $t1) go to Label
	bne $t0, $t1, Label	if ($t0 ≠ $t1) go to Label
	bge $t0, $t1, Label	if ($t0 ≥ $t1) go to Label
	bgt $t0, $t1, Label	if ($t0 > $t1) go to Label
	ble $t0, $t1, Label	if ($t0 ≤ $t1) go to Label
	blt $t0, $t1, Label	if ($t0 < $t1) go to Label
Set	slt $t0, $t1, $t2	if ($t1 < $t2) then $t0 = 1 else $t0 = 0
	slti $t0, $t1, 100	if ($t1 < 100) then $t0 = 1 else $t0 = 0
Jump	j Label	go to Label
	jr $ra	go to address in $ra
	jal Label	$ra = PC + 4; go to Label

4.a) Take a look at the following MIPS assembly code that executes the following 6 instructions in sequence. Suppose the initial values in $s0$, $s1$, and $s2$ are all 0. Analyze the code and give the values of each register after executing each instruction. Fill your answers in the table. (6 marks)

```
(1) li      $s0,   5
(2) li      $s1,   8
(3) li      $s2,   10
(4) addi    $s0,   $s0,    4
(5) add     $s1,   $s1,   $s0
(6) mul     $s2,    $s2,  $s1
```

	$s0	$s1	$s2
Initial value	0	0	0
(1)			
(2)			
(3)			
(4)			
(5)			
(6)			

4.b) Please write down the pseudocode for implementing $X = B+C\times D-E+F+A$ using the following three types of instruction sets: stack architecture, accumulator architecture, register-memory architecture, respectively. (9 marks)
Possible instructions that you may use.

- Stack.

PUSH A (S(i+1) ← M[A], push the memory data at address A into the stack)
POP A (M[A] ← S(i), pop one stack data to the memory at address A)
ADD (S(i-1) ← S(i) + S(i-1), fetch two stack data, add them together and push the result back to the stack)
SUB (S(i-1) ← S(i-1)-S(i), fetch two stack data, subtract the first one from the second one and push the result back to the stack)
MUL (S(i-1) ← S(i) × S(i-1), fetch two stack data, multiply them together and push the result back to the stack)

- Accumulator.

LOAD A (AC ← M[A], load the memory data at address A into the register AC)
STORE A (M[A] ← AC, store the register AC data to the memory at address A)

ADD A (AC $\leftarrow$ AC + M[A], add the register AC data and the data at memory address A together and put the result back to the register AC)
SUB A (AC $\leftarrow$ AC - M[A], subtract the data at memory address A from the register AC and put the result back to the register AC)
MUL A (AC $\leftarrow$ AC $\times$ M[A], multiply the register AC data and the data at memory address A together and put the result back to the register AC)
• Register-memory. You have four registers to use R1, R2, R3, R4.
MOV R A (R $\leftarrow$ M[A], move memory data at address A into the register R)
MOV A R (M[A] $\leftarrow$ R, move the register R data to the memory at address A)
ADD R A (R $\leftarrow$ R + M[A], add the register R data and the data at memory address A together and put the result back to the register R)
SUB R A (R $\leftarrow$ R - M[A], subtract the data at memory address A from the register R and put the result back to the register R)
MUL R A (R $\leftarrow$ R $\times$ M[A], multiply the register R data and the data at memory address A together and put the result back to the register R)

Problem 5 (16 marks)

In class, we have learned the full adders in CPU. In a 2's complement number system, we can use the adder and 2's complement calculation to do the subtraction. Now in this problem, we use similar ideas to design the system with only subtractors.

5.a) Design a binary full subtractor. Given two binary inputs X, Y and the borrow in B_{in}, the binary full subtractor will perform $X - Y - B_{in}$ and output a difference D with a borrow out B_{out}. The truth table is given in Fig. 4 and the representation diagram is shown in Fig. 5. **Write down the logic expressions of D and B_{out} for the full subtractor. Draw the circuit of the full subtractor.** You can use AND/OR/NOT/XOR gates, each with no more than 3 inputs. (6 marks)

Inputs			Outputs	
X	Y	B_{in}	D	B_{out}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Figure 4: Truth table of the full substractor.

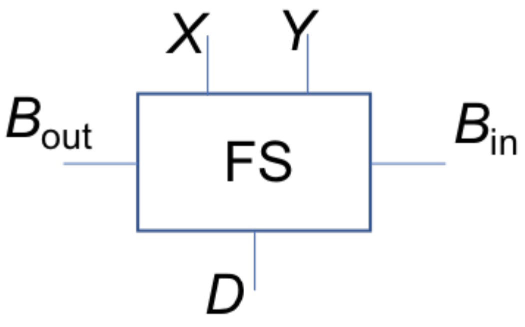
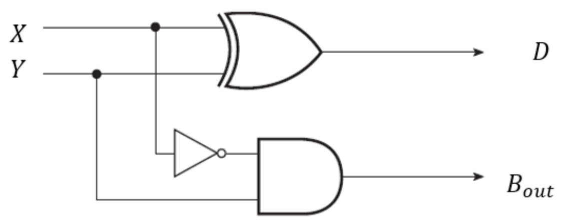
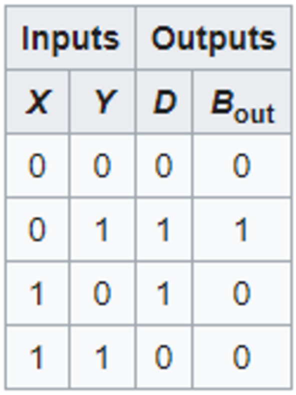


Figure 5: Full subtractor diagram.

5.b) Given the truth table, circuit and the diagram of the half subtractor. Please design a full subtractor using **two** half subtractors and no more than 3 AND/OR/NOT/XOR gates. (4 marks)

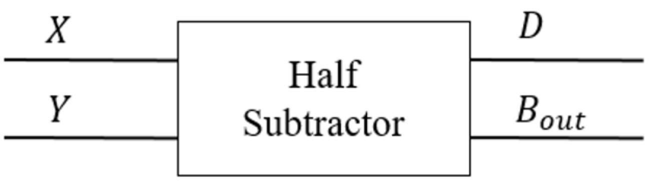


a) Logic circuit of the half subtractor



Inputs		Outputs	
X	**Y**	**D**	**B_{out}**
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

b) Truth table of the half subtractor



c) Half subtractor diagram

5.c) Delay Analysis. We first analyze the gate-delays for the full subtractor and then for a 4-bit subtractor.

Based on your circuit design in Fig. 5 of 5.a), **how many gate delays (including the gates AND/OR/NOT/XOR) are there from the input X to the output B_{out}? How about Y to B_{out}, B_{in} to B_{out}, X to D, Y to D, and B_{in} to D.**

Given a 4-bit subtractor in Fig. 6 as below, **how many gate delays are there from X_0 to D_3, from Y_0 to D_3, from B_{in} to D_3, from X_0 to B_{out}, from Y_0 to B_{out}, from B_{in} to B_{out}?** (Note that for the 4-bit subtractor, B_{out} is the borrow out $B_{out,3}$ of the highest-bit's full subtractor, as shown in Fig. 6.) (6 marks)

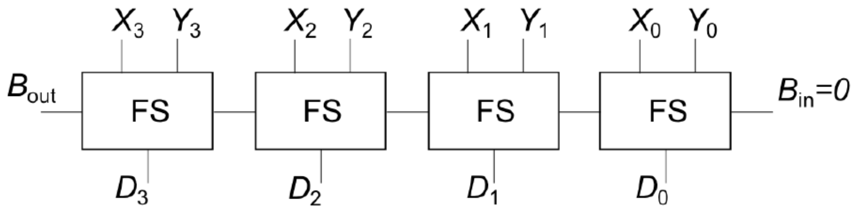


Figure 6: 4-bit subtractor.

Problem 6 (12 marks)

Please answer your questions in a short and concise way!
6.a) What are the 5 stages of an instruction cycle in RISC (e.g., MIPS)? Explain them. Explain what does pipeline operations mean when executing instructions. Why do we often use such pipeline operations? (6 marks)

6.b) Compare the differences between DRAM and SRAM (list at least 3 differences). What do we usually use DRAM and SRAM for in the computer systems? Why? (3 marks)

6.c) What are the three techniques for I/O operation? Explain them. (3 marks)