CS 251, Fall 2025, Assignment 1.0.0

Due Wednesday, September 24th, 10:00 PM

You are required to read, complete and sign, and submit (as well as follow) the following statement of Academic Integrity.

Statement of Academic Integrity for CS 251 Fall 2025, **Assignment 1**
I declare the following statements to be true:

- I have not used any unauthorized aids.
- I recognize that while I can discuss the questions in this assignment on Piazza and other forums with the instructors and with other students in the class, the write up that I am submitting is my own.
- I am aware that misconduct related to course work can result in significant penalties, including failing the course and suspension (this is covered in Policy 71:)

  `https://uwaterloo.ca/secretariat/policies-procedures-guidelines/policy-71`

Student Name:

UW ID#:

Signature:

Date:

Note: in the ARM questions, you should only use the ARM instructions discussed in class and in the ARM document provided for the course (e.g., ADD, SUB, ADDI, SUBI, LDUR, STUR, CBZ, CBNZ, B).

1. (3 points)

   Write an ARM program starting at address 100 that reads the value stored at memory address 4104 into X1 and then adds 2025 to it. The final value is stored back to memory address 4112. You may use X2 as a temporary register. Note that the constant used by LD/ST can not be any larger than 256, and that ADDI/SUBI have a 12-bit, unsigned constant.

100	
104	
108	
112	
116	
120	

2. (8 points)

Write a sequence of ARM instructions that will store the cumulative sum of an array in the array. I.e., for initial values $v[i]$, $i = 0..n-1$, after executing the code the array will contain

$$v[i] = \sum_{j=0}^{i} v^o[j],$$

where $v^o[i]$ is the original value stored in $v[i]$. For example, starting with an array containing the initial values 2,3,1,5, the final array will contain the values 2,5,6,11.

The address of the first element of the array is stored in register X1, and number of elements in the array is stored in register X2. You may assume that X1 contains a value that is a multiple of 8, and that X2 contains a value between 1 and 100.

You may use any other registers as temporary registers, and you are free to change the values stored in X1 and X2.

Be sure to state what your temporary registers are used for.

000	
004	
008	
012	
016	
020	
024	
028	
032	
036	

3. (8 points)

Write your student number here:

(a) (3 points) For 3 inputs A, B, C, construct the truth table for the function F that is 1 when the binary number ABC is a digit in your student ID. For example, if your student ID is 20340259, the rows 000, 010, 011, 100, and 101 should be 1, while the other entries should be 0 (we are ignoring 8's and 9's in your student ID).

A	B	C	F
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

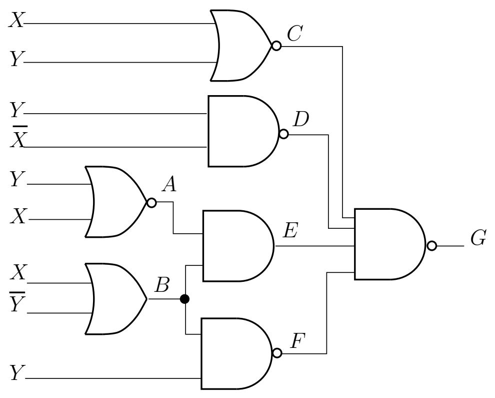
(b) (3 points) Give the unreduced sum of products expression for F:

$F =$

(c) (2 points) Draw the circuit that implements F from part (b). You should *not* reduce the formula or the circuit.

4. (8 points)

(a) (6 points) Complete the truth table table for output F in the following circuit, giving the truth table values for the internal signals A, B, and C and D as intermediate steps.



X	Y	A	B	C	D	E	F	G
0	0							
0	1							
1	0							
1	1							

(b) (2 point)

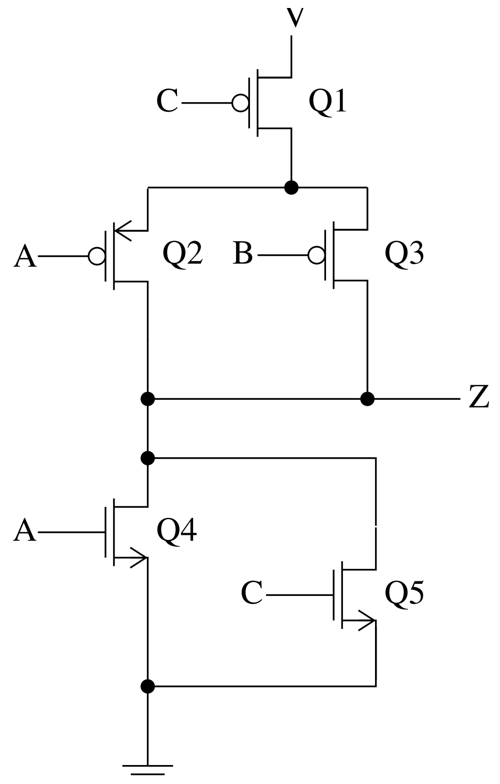
Give one operation formulas for F and G in terms of X, Y or their complements ("one operation"–you should use exactly one of AND, OR, NAND, NOR, XOR, XNOR in each of your expressions for F and G).

$F =$

$G =$

5. (8 points)

Given the transistor circuit below, complete the table below, determining the internal resistances for Q_1 to Q_6 and the final output Z. The transistors Q_1 to Q_6 resistances should be High or Low (or 'H' or 'L'). The output Z may be 0 or 1, float '–', or short-circuit '*'. You may assume all resistances are accurate when determining the final output Z.

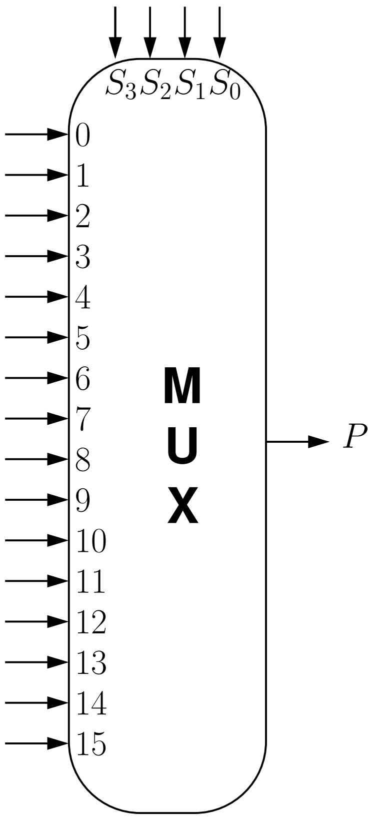


A	B	C	Q_1	Q_2	Q_3	Q_4	Q_5	Z
0	0	0						
0	0	1						
0	1	0						
0	1	1						
1	0	0						
1	0	1						
1	1	0						
1	1	1						

6. (4 points) In the diagram below a 16–1 multiplexors. Suppose we want to implement “2-Modulus-3” with this multiplexors, where our circuit has a 4-bit numbers $(A_3A_2A_1A_0)$ as input and a 1=bit result P. The truth table for 2-Modulus-3 is

A_3	A_2	A_1	A_0	P	A_3	A_2	A_1	A_0	P
0	0	0	0	0	1	0	0	0	1
0	0	0	1	0	1	0	0	1	0
0	0	1	0	1	1	0	1	0	0
0	0	1	1	0	1	0	1	1	1
0	1	0	0	0	1	1	0	0	0
0	1	0	1	1	1	1	0	1	0
0	1	1	0	0	1	1	1	0	1
0	1	1	1	0	1	1	1	1	0

The output of the multiplexor is labeled. Label the inputs and select lines to the multiplexor to build a circuit for the 2-Modulus-3 function.

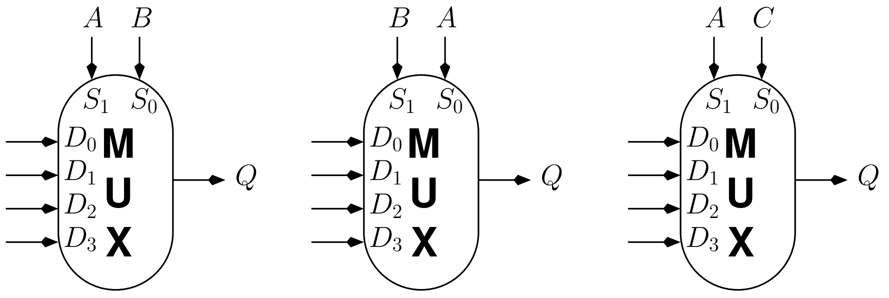


The remaining questions will NOT be used to compute your assignment mark; they are included here as additional questions you may want to try to aid your understanding of the course material. Solutions to these questions will not be provided.

7. In the diagram below is a 4–1 multiplexor with its select lines tied to A and B. Suppose we want to implement the following function using this multiplexor:

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

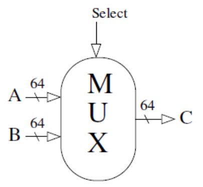
Label the inputs D_0, D_1, D_2, D_3 so that the output Q is given by the truth table for F. For the left and center MUXes, you may use the labels C, $\bar{C}$, 0, or 1; for the right MUX, you may use the labels B, $\bar{B}$, 0, or 1.

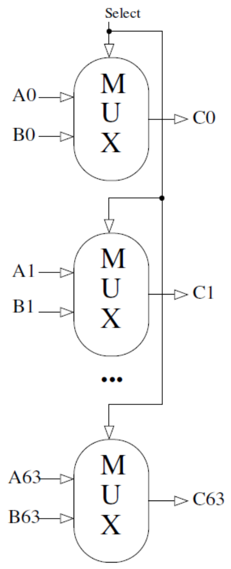


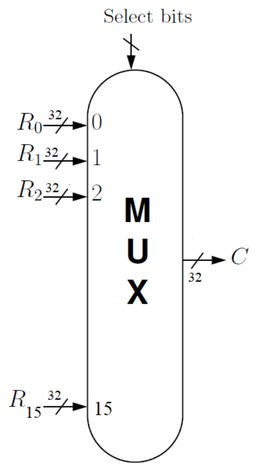
8. Draw a CMOS circuit that takes two inputs, A and B, and performs the **NOR** operation at an output Z. You must only use NMOS and PMOS transistors. Be sure to clearly indicate a Power connection and grounding of your circuit. You may look up this configuration on the web, but it must be accurately drawn below. You should look up 'CMOS NOR'. The circuit must be implemented with CMOS configurations as was the NAND operation from class.

9. Below is a diagram of a 2×1 multiplexor taking two 64-bit inputs and pro ducing one 64-bit output. On the right, is the internal configuration of 2×1 MUXes that will implement this operation.

   Further down, we have drawn a 16×1 multiplexor taking 16 inputs that are 32-bits each and producing one 32-bit result. Draw the internal configuration of MUXes that will implement the 16×1 MUX. You do not need to draw all of the inputs or all the necessary multiplexors, but you should use (...) where necessary to simplify your diagram. The number of inputs and the number of multiplexors you use should be clearly marked. Be sure to label the width of the input lines, the width of the select bits, etc. Also, be sure to clearly indicate where the select bits feed into each multiplexor. You should label the inputs to the multiplexors and the bit position clearly using the convention R_{X_Y} (for example R_{2_0} indicates the zero bit of R_2).


Select
A
64
B
64
MUX
64
C


Select
A0
B0
MUX
C0
A1
B1
MUX
C1
...
A63
B63
MUX
C63


Select bits
R_0 32 0
R_1 32 1
R_2 32 2
MUX
32 C
R_{15} 32 15