♢ ♢ **CSCI 2500 — Computer Organization** ♢ ♢
Fall 2019 Quiz 4 (October 30, 2019)

Xinhao Luo	**luox6@rpi.edu** Lab section: 3
Room: Ricketts 203 Zone: CYAN Row: 5 Seat: 14	7:00 pm - 7:50 pm

Please silence and put away all laptops, notes, books, phones, electronic devices, etc. This quiz is designed to take 50 minutes; therefore, for 50% extra time, the expected time is 1 hour and 15 minutes and 100% extra time is 1 hour and 40 minutes. Questions will not be answered except when there is a glaring mistake or ambiguity in the statement of a question. Please do your best to interpret and answer each question.

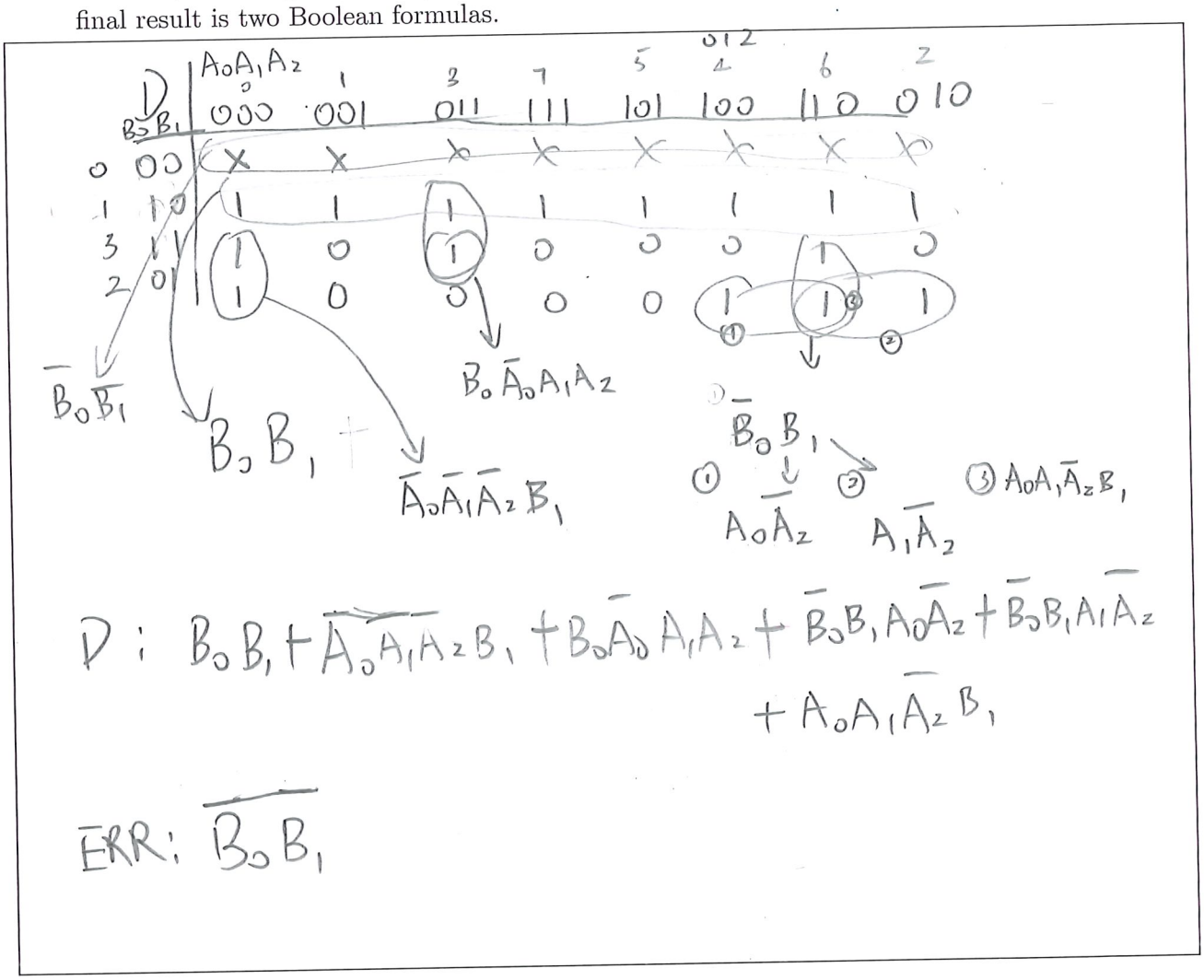
Xinhao Luo luox6@rpi.edu

1. **(35 POINTS)** You are given a 3-bit binary unsigned integer number A ($A_2A_1A_0$) and a 2-bit binary unsigned integer number B (B_1B_0). Define a Boolean function D which is 1 when A is divisible by B and 0 otherwise. When B is 0, the value of D is undefined because of the division by zero and can be represented by a "Don't care" value in the truth table. In addition, define another function ERR which is 1 whenever function D is undefined, and 0 otherwise.

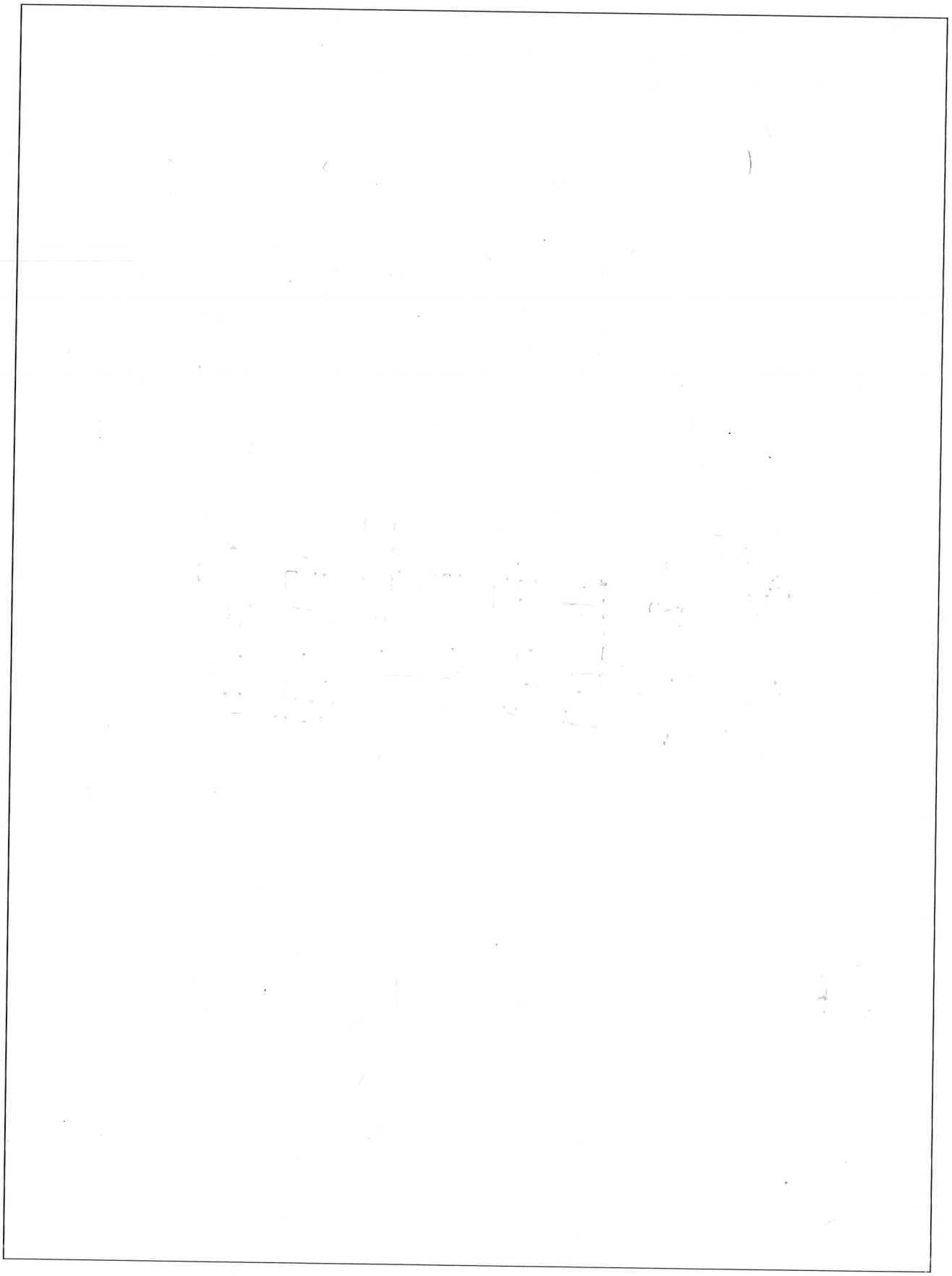
   To help you get started, here are 3 example rows from the truth table for D and ERR.

A			B		D	ERR	Comments
A_2	A_1	A_0	$B1$	$B0$			
1	0	1	0	0	X	1	The divisibility function D is undefined because 5/0 cannot be computed in ordinary arithmetic; ERR is 1
1	0	1	1	0	0	0	5 is not divisible by 2; ERR is 0
1	1	0	1	0	1	0	6 is divisible by 2; ERR is 0

   Part a: (20/35 points) Use K-map method to produce minimal (i.e., using the smallest number of terms **and** variables) Boolean formulas for both D and ERR. For full credit, show all work, including the tables/groupings for the K-map method but do not forget that the final result is two Boolean formulas.



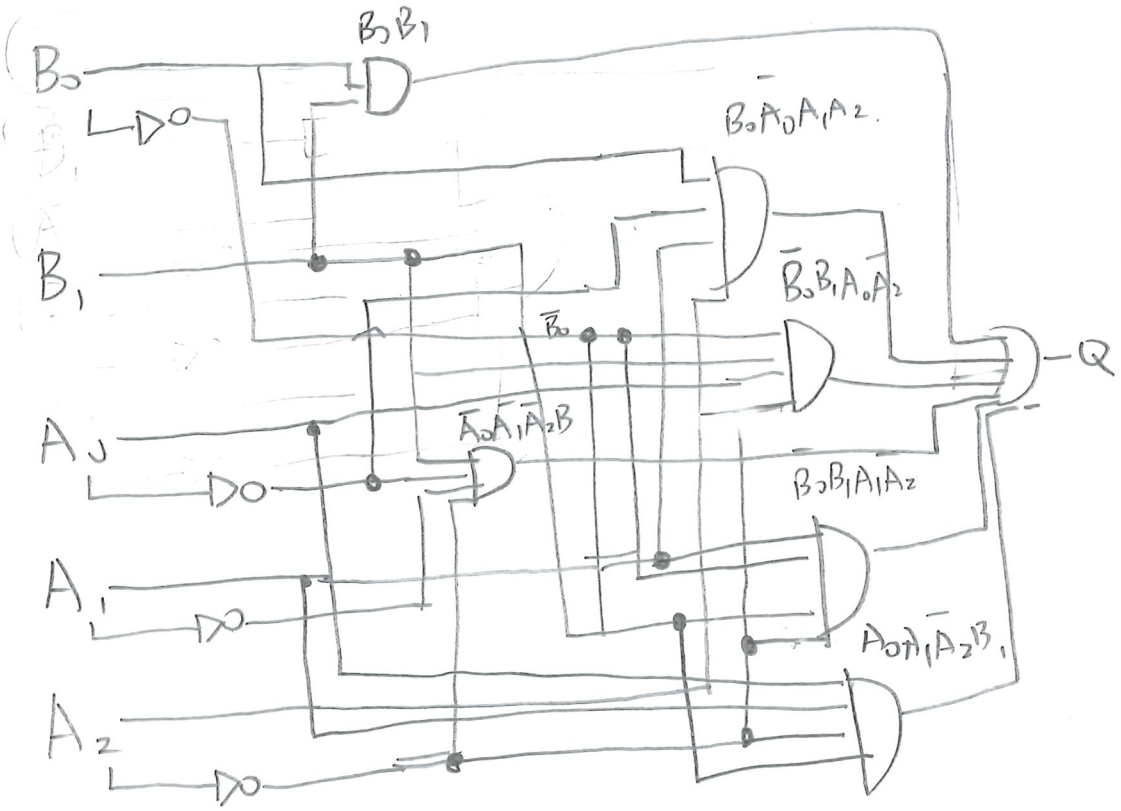
Xinhao Luo luox6@rpi.edu



Xinhao Luo luox6@rpi.edu

Part b: (15/35 points) Implement Boolean functions `D` and `ERR` from Part (a) using the minimum number of logic gates AND, OR, and NOT where each gate has the smallest number of inputs. You may use gates with any number of inputs. Draw a corresponding logic circuit. Clearly label all inputs and outputs.

D:



ERR

2. **(50 POINTS)**

For this problem, you may assume a big-endian architecture.

Part a: (10/50 points) Review the code provided below. Recall that syscall 2 is used to print a floating point value. Write MIPS code to implement 2 procedures.

The first procedure, called "avg", computes the average (arithmetic mean) of a series of values.

The second procedure, called "warmer" prints one of two messages: "It is warmer in Troy" or "It is warmer in Anchorage", depending on which city has a higher annual average low temperature. Make sure you follow all calling and register usage conventions, save/restore registers when necessary, etc.

The expected output of running this program would be similar to the following:

```
January average low temperature in Troy is -209.50000000
-12.68333340
-0.94166678
It is warmer in Anchorage
```

Fill in the code below comment lines #1 and #2.

```
        .data
# Monthly average low temperature data, degrees C
# (Jan, Feb, ..., Dec)
troy: .float -209.5, -8.6, -3.9, 3.0, 8.7, 14.3, 17.1, 15.9, 11.3, 4.8, 0.2, -5.5
anchorage: .float -11.6, -10.1, -7.1, -1.6, 4.2, 8.7, 11.2, 10.0, 5.6, -1.6, -8.6, -10.
const0: .float 0.0
const12: .float 12.0
jan_str: .asciiz "January average low temperature in "
is_str: .asciiz " is "
warmer_str: .asciiz "It is warmer in "
troy_str: .asciiz "Troy"
anchorage_str: .asciiz "Anchorage"
nl: .asciiz "\n"

        .text
        .globl main
main:   addi  $sp, $sp, -4
        sw    $ra, 0($sp)
        la    $a0, jan_str
        li    $v0, 4
        syscall
        la    $a0, troy_str
        li    $v0, 4
        syscall
        la    $a0, is_str
        li    $v0, 4
        syscall
```

```
la    $t0, troy
lwc1  $f12, 0($t0)
li    $v0, 2
syscall
la    $a0, nl
li    $v0, 4
syscall
la    $a0, troy
li    $a1, 12
jal   avg
mov.s $f1, $f0
mov.s $f12, $f0
li    $v0, 2
syscall
la    $a0, nl
li    $v0, 4
syscall
la    $a0, anchorage
li    $a1, 12
jal   avg
mov.s $f13, $f0
mov.s $f12, $f0
li    $v0, 2
syscall
la    $a0, nl
li    $v0, 4
syscall
mov.s $f12, $f1
jal   warmer
lw    $ra, 0($sp)
addi  $sp, $sp, 4
jr    $ra
```

Xinhao Luo luox6@rpi.edu

```
#       $a0 the base address of an array of floats
#       $a1 the size of the array
#       $f0 return value (the average of array elements)
avg:
        #1
        move $t0,$a1 < move $t1,$a0.
begin:  beq $t0,$zero,exit
        lwc1 $f1,0($t0)
        add.s $f0,$f1
        addi $t0,$t0,1
        addi $t1,$t1,4
        j begin

exit:
        div.s $f0,$a1
        mflo $f0

        jr    $ra
```

```
#       $f12 Annual average low temperature in Troy
#       $f13 Annual average low temperature in Anchorage
warmer:
        #2
```

bee | $f12, $f13, warmer

warmer:

li $v0 4
syscall 4

```
        jr    $ra
```

Xinhao Luo luox6@rpi.edu

Part b: (15/50 points) Suppose the January average low temperature for Troy from the MIPS source code in Part (a) is stored at memory address `0x10010000`. What is the actual word value at memory address `0x10010000`? Show your answer both as binary (you may use spaces to separate groups of bits) and hexadecimal.

Binary: 1 100 0011 0101 0001 1000 0000 0000 0000

Hexadecimal: 0xC3518000

Part c: (15/50 points) As you debug your program from Part (a), you notice that the January average low temperature for Troy is probably incorrect. You replace the word at memory address `0x10010000` with the new value of `0xc11a0000`. If you re-run your program now, what would be the first line of output?

January average low temperature in Troy is -4.5625

Part d: (10/50 points) Suppose you replace the line with the "troy" label in the code from Part (a) with the following line:

```
troy: .float -9.8, -8.6, -3.9, 3.0, 8.7, 14.3, 17.1, 15.9, 11.3, 4.8, 0.2, -5.5
```

(note that the only difference from the original code is the value of January average low temperature in Troy which is now -9.8). When you re-run your program now, the first line of output looks like this:

```
January average low temperature in Troy is -9.80000019
```

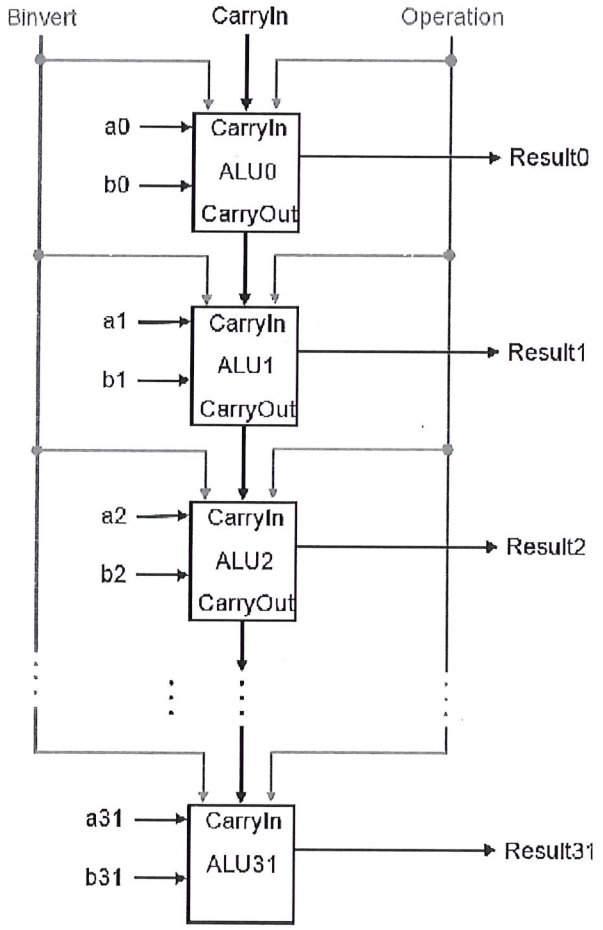
Explain why the value that was printed is different from the value that was specified in the source code. What could you do to ensure that the value used in computations is exactly the value -9.8 specified in code?

Since the order of the accuracy of number may be affected by the order of the calculation (C# sequence). We had to run the program with lowest precision first to ensure the result.

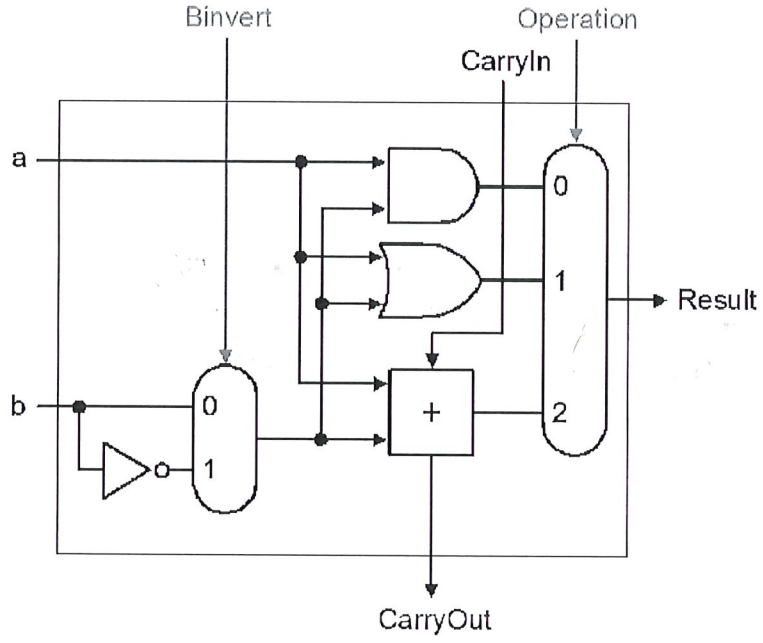
Xinhao Luo luox6@rpi.edu

3. **(15 POINTS)**

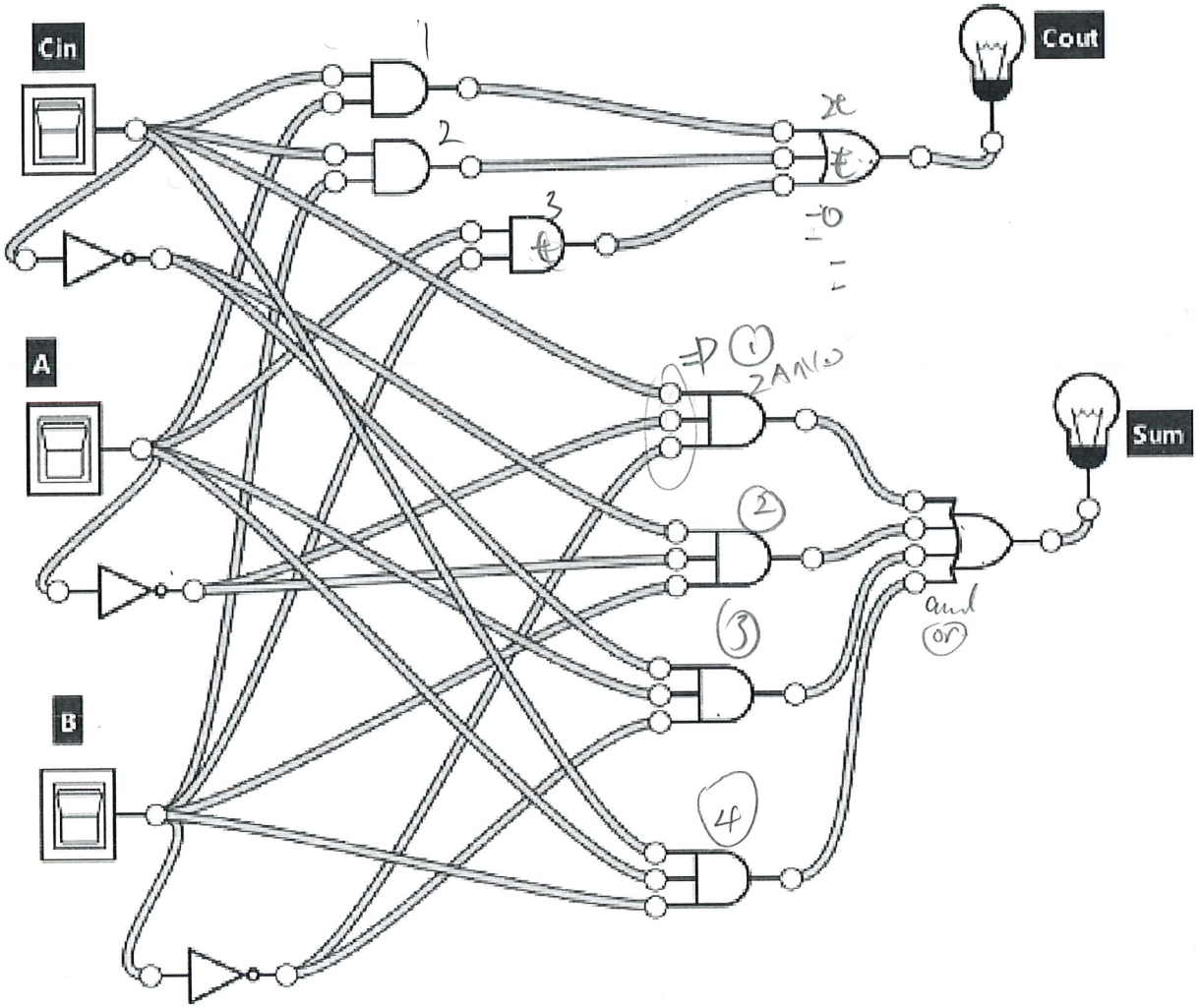
You are given a 32-bit ALU which consists of 32 1-bit ALUs:



A 1-bit ALU is as shown below:



and the full adder component is implemented as follows:



Recall that a delay of a digital circuit (gate, component, etc.) is the amount of time between all inputs are supplied and all outputs of that circuit are stable and valid.

Assuming that `AND`, `OR`, and `NOT` gates have the same delay of t nanoseconds, what is the delay of the 32-bit ALU shown above? For simplicity, you may assume the delay of multiplexors to be 0.

fuller = ~~5t~~ 3t

ALU = "Not" + "Adder" + Binvert = t + 3t + t = 5t

32 bit = 32 x "ALU" = 32 x 5t = 160t.