1	/10
2	/15
3	/13
4	/16
5	/14
6	/18
7	/14

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

6.004 Computation Structures
Fall 2019

Practice Quiz #1A
(adapted from Quiz #1 and Quiz #2 Spring 2018)

Name	*Athena login name*	*Score*
Recitation section ☐ WF 11, 35-308 (Silvina) ☐ WF 12, 35-308 (Silvina) ☐ WF 1, 38-166 (Andy) ☐ None (pick up quiz in 32-G846)		

Please enter your name, Athena login name, and recitation section above. Enter your answers in the spaces provided below. Show your work for partial credit. You can use the extra white space and the backs of the pages for scratch work.

Problem 1. Binary Arithmetic (10 points)

(A) (4 points) What is ~(0x5A) ^ 0x3D, where ~ is bitwise NOT and ^ is bitwise XOR? Provide your result in both binary and hexadecimal.

Result in binary (0b):_________________________

Result in hexadecimal (0x):_________________________

(B) (4 points) What is 15 in 8-bit 2’s complement notation? What is –22 in 8-bit 2’s complement notation? Show how to compute 15–22 using 2’s complement addition. What is the result in 8-bit 2’s complement notation?

15 in 8-bit 2’s complement notation (0b):_____________________

–22 in 8-bit 2’s complement notation (0b):_____________________

15–22 in 8-bit 2’s complement notation (show your work) (0b):_____________________

(C) (2 points) What range of numbers encoded using two’s complement representation can be expressed using 5 bits? Provide your answer in decimal.

Smallest 5-bit two’s complement number (in decimal):_______________

Largest 5-bit two’s complement number (in decimal):_______________

Problem 2. Assembly Language (15 points)

For the RISC-V instruction sequences below, provide the hex values of the specified registers after each sequence has been executed. Assume that each sequence execution ends when it reaches the `unimp` instruction. Also assume that all registers are initialized to 0 before execution of each sequence begins.

(A) (7 points)

```
. = 0x0
     li x11, 0x400
     lw x11, 0x0(x11)
     bge x11, x0, L1
     xori x12, x11, 0xfff
     j end

L1:  srli x12, x11, 8
end: unimp

. = 0x400
X:   .word 0xC0C0A0A0
```

Value left in x11: 0x_______________

Value left in x12: 0x_______________

(B) (8 points)

```
. = 0x0
     lw x11, 0x200(x0)
     mv x12, x0
loop:
     andi x13, x11, 1
     add x12, x12, x13
     srli x11, x11, 1
     bnez x11, loop
     unimp

. = 0x200
X:   .word 0x00140083
```

Value left in x11: 0x_______________

Value left in x12: 0x_______________

Value left in x13: 0x_______________

Problem 3. RISC-V Assembly and Calling Conventions (13 points)

For each of the code segments below, specify whether or not the RISC-V assembly code properly implements the desired functionality described in C and satisfies the RISC-V calling convention. If either the functionality *or* the calling convention are not satisfied, then provide code that is functionally correct and satisfies the calling convention.

Note that '**g**' refers to another function whose specification is not provided to you. **Also, note that there may be multiple errors in the code. You should correct them all.**

(A) (7 points)

<table>
<tr>
<td>

```
int f(int a, int b) {
   return g(a + b, b) + a;
}


f:   mv s1, a0
     add a0, a1, a0
     jal ra, g
     add a0, a0, s1
     ret

g:
```

</td>
<td>

Assembly code produces expected results and follows calling convention:

YES NO

If not, provide correct code here:

</td>
</tr>
</table>

(B) (6 points)

<table>
<tr>
<td>

```
int f(int a, int b, bool c) {
   if (c) return a * 2;
   else return a – b;
}


f:   bnez a2, L1
     slli a0, a0, 2
     ret

L1:  sub a0, a1, a0
     ret
```

</td>
<td>

Assembly code produces expected results and follows calling convention:

YES   NO

If not, provide correct code here:

</td>
</tr>
</table>

Problem 4. Procedures and Stacks (16 points)

You are given an incomplete listing of a C procedure and its translation to RISC-V assembly code (shown on the right):

```
int f(int a, int b) {
  if (a < b)
    return ???;
  else
    return a | b;
}
```

```
f:  bge a0, a1, L3
    addi sp, sp, -12
    sw a0, 0(sp)
    sw a1, 4(sp)
    sw ra, 8(sp)
L1: addi a0, a0, 1
    addi a1, a1, -1
    jal ra, f
    lw t0, 0(sp)
    lw t1, 4(sp)
    lw ra, 8(sp)
L2: addi sp, sp, 12
    xor t0, t0, t1
    or a0, a0, t0
    jr ra

L3: or a0, a0, a1
    jr ra
```

(A) (3 points) Give the HEX encoding of the `'sw ra, 8(sp)'` instruction.

Hex encoding of sw ra, 8(sp): 0x ______________

(B) (3 points) What is the missing C expression corresponding to the `'???'` in the above program?

(give C code expression)

(C) (2 points) How many words are stored on the stack every time we want to execute function `f` recursively?

Number of words pushed onto the stack for each recursive call to f? __________

The program's initial call to function `f` occurs outside of the function definition via the instruction `'jal ra, f'`. The program is interrupted at an execution (not necessarily the first) of function `f`, *just prior* to the execution of the `addi sp, sp, 12` instruction at label `L2`. The diagram on the right shows the contents of a region of memory. All addresses and data values are shown in hex. **The current value in the SP register is 0xF40 and points to the location shown in the diagram.**

Memory Contents

	Addr	Data
	0xF30	0x01
	0xF34	0x05
	0xF38	0x05
	0xF3C	0x124
SP→	0xF40	0x04
	0xF44	0x06
	0xF48	0x124
	0xF4C	0x03
	0xF50	0x07
	0xF54	0xA08
	0xF58	0x02
	0xF5C	0x08

(D) (3 points) What are the arguments to the ***original*** call to `f`? Write CAN'T TELL if you can't tell.

Original arguments to f, a = ___________ ; b = ___________

(E) (3 points) What are the arguments to the ***current*** call to f? Write CAN'T TELL if you can't tell.

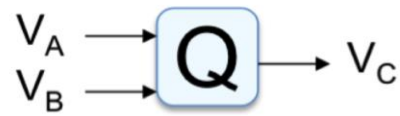
Current arguments to f, a = ___________; b = ___________

(F) (2 points) What value was in SP just prior to the initial call to `f`?

Initial contents of SP: 0x __________

Problem 5. Static Discipline (14 points)

The Q-module shown to the right has two inputs carrying voltages V_A and V_B and a single output carrying V_C. The output V_C is 1 volt if the sum of the input voltages, V_A+V_B, has been more that 4 volts for at least 10 ns; it will be 5 volts if V_A+V_B, has been less than 3 volts for at least 10 ns; and otherwise is at some undetermined voltage between 0 and 5 volts.



To summarize: after inputs have been stable for 10ns,

$$\mathbf{V_C} = \begin{cases} \textbf{1 volt} & \textbf{if } \mathbf{V_A + V_B > 4} \textbf{ volts} \\ \textbf{5 volts} & \textbf{if } \mathbf{V_A + V_B < 3} \textbf{ volts} \\ \mathbf{0 \le ??? \le 5} \textbf{ volts} & \textbf{otherwise} \end{cases}$$

You may assume that no negative voltages are used in the circuits of this problem.

In this problem, you will explore the possibility of using the Q-module as the basis for a new family of logic devices. To this end, we need a convention for representing logic values (0 and 1) as voltages. This convention should yield acceptably large noise margins. In particular, it should maximize **noise immunity, defined as the width of the smaller of the two noise margins**.

(A) (2 points) Suppose constant voltages are applied to the V_A and V_B input terminals of a Q-module, and an output voltage of 5 volts is measured at V_C after 20 ns. Assuming the Q-module obeys the above specification, what can you conclude about the sum V_A+V_B of the input voltages? Choose the best answer.

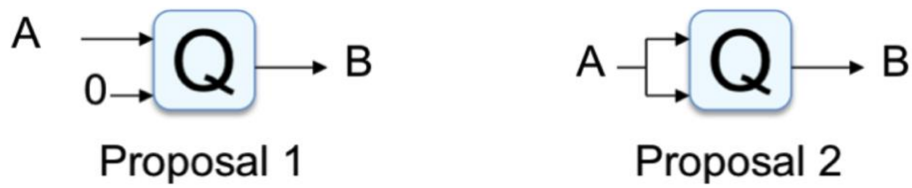
C1: $V_A+V_B < 3$ volts

C2: $V_A+V_B \le 4$ volts

C3: None of the above.

Best conclusion about V_A+V_B (circle one): C1 … C2 … C3

You begin by exploring configurations of the Q-module that will perform as an inverter. You consider two different inverter proposals:



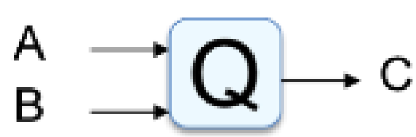
(B) (8 points) Select the proposal that gives the best noise immunity, and specify parameters for an appropriate logic mapping and the resulting noise immunity.

Best proposal (1 or 2): ______

Values for V_{OL}: ______; V_{IL}: ______; V_{IH}: ______; V_{OH}: ______

Noise immunity: ______

Next, you consider logic mappings that allow using a single Q-module directly as a 2-input combinational device, as depicted to the right. Your goal is to find a set of logic mapping parameters for which the 2-input circuit at the right computes a useful logic function, and does so with acceptable noise margins.



(C) (2 points) Consider a logic convention for which a Q-module can serve as a logic device computing an interesting (non-constant) function of its inputs and that maximizes the noise immunity. Specify the resulting noise immunity.

Noise immunity: ______volts

(D) (2 points) Identify the function computed by the single Q-module given the above convention, by specifying a Boolean expression for C in terms of inputs A and B.

Boolean expression for computed function:________________________________

Problem 6. Boolean Algebra and Combinational Logic (18 points)

(A) (6 points) Simplify the following Boolean expressions by finding a minimal sum-of-products expression for each one. (*Note:* These expressions can be reduced into a minimal SOP by repeatedly applying the Boolean algebra properties we saw in lecture.)

1. $\overline{(a + b \cdot \bar{c})} \cdot d + c$

2. $a \cdot \overline{(b + c)}(c + a)$

(B) (6 points) The following Minispec function `f` performs a basic operation using `a` and `b`. We want `f2` to implement the same function as `f`. Fill in the blank in `f2` to make the two functions equivalent. Use a single expression. Assume `n` is a power of 2.

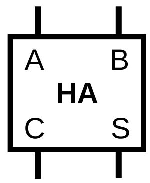
```
// Assume that n is a power of 2; log2(n) is the log base 2 of n
function Bit#(1) f#(Integer n)(Bit#(n) a, Bit#(log2(n)) b);
   Bit#(n) x = a;
   for (Integer i = 0; i < log2(n); i = i+1) begin
         // (2**i) is 2 to the ith power
         x = (b[i] == 1) ? x >> (2**i) : x;
   end
   return x[0];
endfunction

function Bit#(1) f2#(Integer n)(Bit#(n) a, Bit#(log2(n)) b);

   return __________________;

endfunction
```

(C) (6 points) Show that the half-adder device (HA) shown below can be used to implement any combinational circuit by implementing an inverter, an AND gate, and an OR gate using only half-adder circuits. Make sure to clearly label the output. You may tie inputs to 1 or 0 if necessary, and may use multiple half-adder circuits.



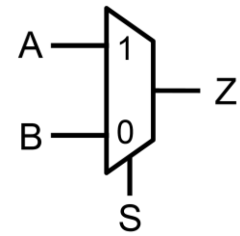
C = A & B;
S = A ^ B;

Logic diagram of inverter implementation using half-adders:

Logic diagram of AND gate implementation using half-adders:

Logic diagram of OR gate implementation using half-adders:

Problem 7. CMOS Logic (14 points)

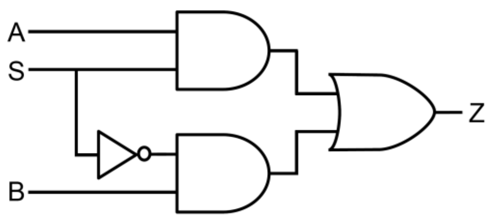


Muxes are used often so it is important to optimize them. In this problem you will design several variants of a 1-bit, 2-to-1 mux (shown to the right) using CMOS gates, and will compare their costs in number of transistors.

Note: Remember that a CMOS gate consists of an output node connected to a *single* pFET-based pullup circuit and a *single* nFET-based pulldown circuit. Gates obtained by combining multiple CMOS gates are not a CMOS gate.

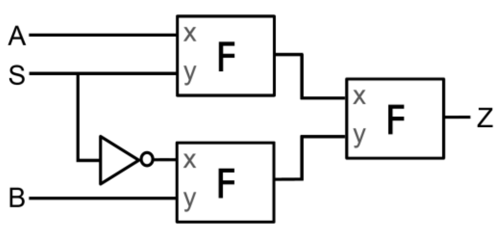
$$\mathbf{Z = A \cdot S + B \cdot \bar{S}}$$

(A) (2 points) Consider the implementation shown below, which uses two AND gates and an OR gate. Because a single CMOS gate cannot implement AND or OR, each AND gate is implemented with a CMOS NAND gate followed by a CMOS inverter, and the OR gate is implemented with a CMOS NOR gate followed by a CMOS inverter. How many transistors does this implementation have?



Number of transistors in mux: ___________________

(B) (4 points) Consider the implementation shown below, which uses three instances of gate F. Find the Boolean expression for F. If F can be built using a single CMOS gate, draw its CMOS implementation. Otherwise, give a convincing explanation for why F cannot be implemented as a CMOS gate. How many transistors does this implementation have?

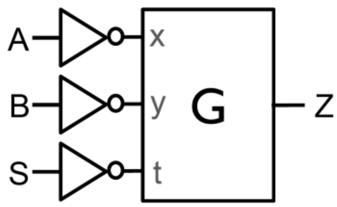


F(x,y) = ______________________

Draw CMOS gate that implements F or explain why it cannot be built.

Number of transistors in mux (if F can be built as a CMOS gate): ___________________

(C) (4 points) Consider the implementation shown below, which uses gate G. Find the Boolean expression for G. If G can be built using a single CMOS gate, draw its CMOS implementation. Otherwise, give a convincing explanation for why G cannot be implemented as a CMOS gate. How many transistors does this implementation have?

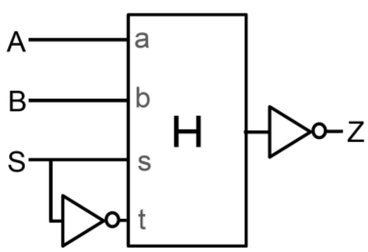


G(x,y,t) = ____________________

Draw CMOS gate that implements G or explain why it cannot be built.

Number of transistors in mux (if G can be built as a CMOS gate): _________________

(D) (4 points) Consider the implementation shown below, which uses gate H. Find the Boolean expression for H. If H can be built using a single CMOS gate, draw its CMOS implementation. Otherwise, give a convincing explanation for why H cannot be implemented as a CMOS gate. How many transistors does this implementation have?



H(a,b,s,t) = ____________________

Draw CMOS gate that implements H or explain why it cannot be built.

Number of transistors in mux (if H can be built as a CMOS gate): _________________

END OF QUIZ 1!