UNIVERSITY OF TORONTO
Faculty of Arts and Science
Summer 2016 Final Examination
CSC 258H1 Y
Duration — 3 hours
Aids allowed: none

Student Number: |_|_|_|_|_|_|_|_|_|_|

UTORid: |_|_|_|_|_|_|_|_|_|_|

Last Name: ____________________ **First Name:** ____________________

Question 0. [1 MARK]

Read and follow all instructions on this page, and fill in all fields.

*Do **not** turn this page until you have received the signal to start.*
(Please fill out the identification section above and read the instructions below.)
Good Luck!

This midterm is double-sided, and consists of 9 questions on 16 pages (including this one). When you receive the signal to start, please make sure that you have all pages.

- If you use any space for rough work, indicate clearly what you want marked.
- Write "Hi Brian" in the bottom left corner of this page
- In lieu of answering, you may write "I don't know" on any question to receive partial credit (20% rounded up to the nearest half mark) for the question. Answers which do not demonstrate a sensible understanding of the question, will not receive partial marks. In other words, don't guess if you don't know.
- Do not remove any pages from the exam booklet.

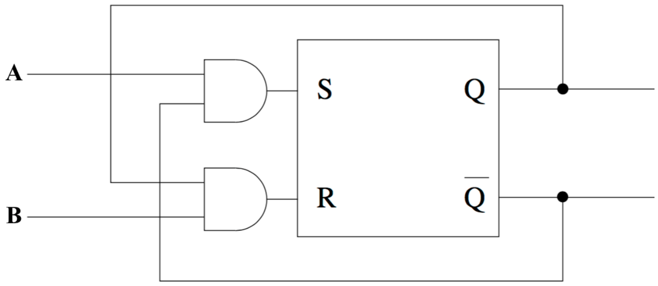
# 0:	______/ 1
# 1:	______/10
# 2:	______/ 6
# 3:	______/ 5
# 4:	______/10
# 5:	______/15
# 6:	______/ 3
TOTAL:	______/50

[Use the space below for rough work. This page will not be marked unless you clearly indicate the part of your work that you want us to mark.]

Question 1. [10 MARKS]

Part (a) [2 MARKS]

In the space below, provide the truth table for the following circuit:



Part (b) [3 MARKS]

Using only the circuit above (you can use a block diagram to represent the circuit) and standard logic gates (and/or/not/xor/nand/nor), draw a counter that counts the sequence 0, 1, 2, 3, 0, 1, 2, 3, ...

[Use the space below for rough work. This page will not be marked unless you clearly indicate the part of your work that you want us to mark.]

Part (c) [3 MARKS]

Using only the circuit from part a and standard logic gates (and/or/not/xor/nand/nor), draw a "counter" that counts the sequence 3, 2, 1, 3, 2, 1, 3, 2, 1, ...

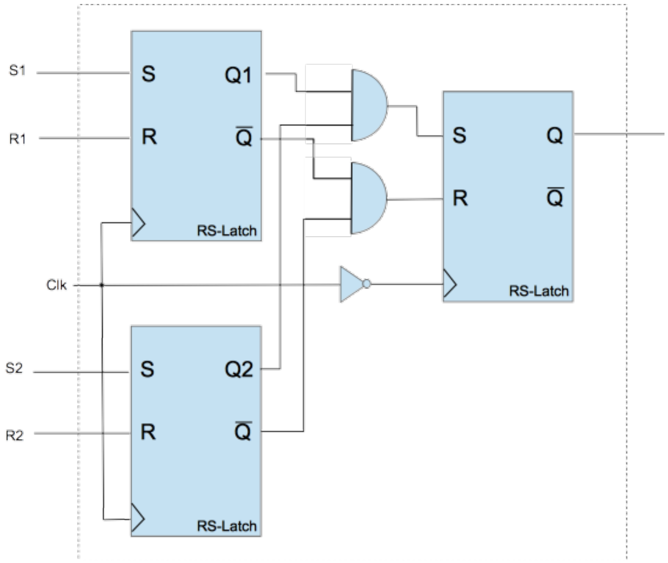
Part (d) [2 MARKS]

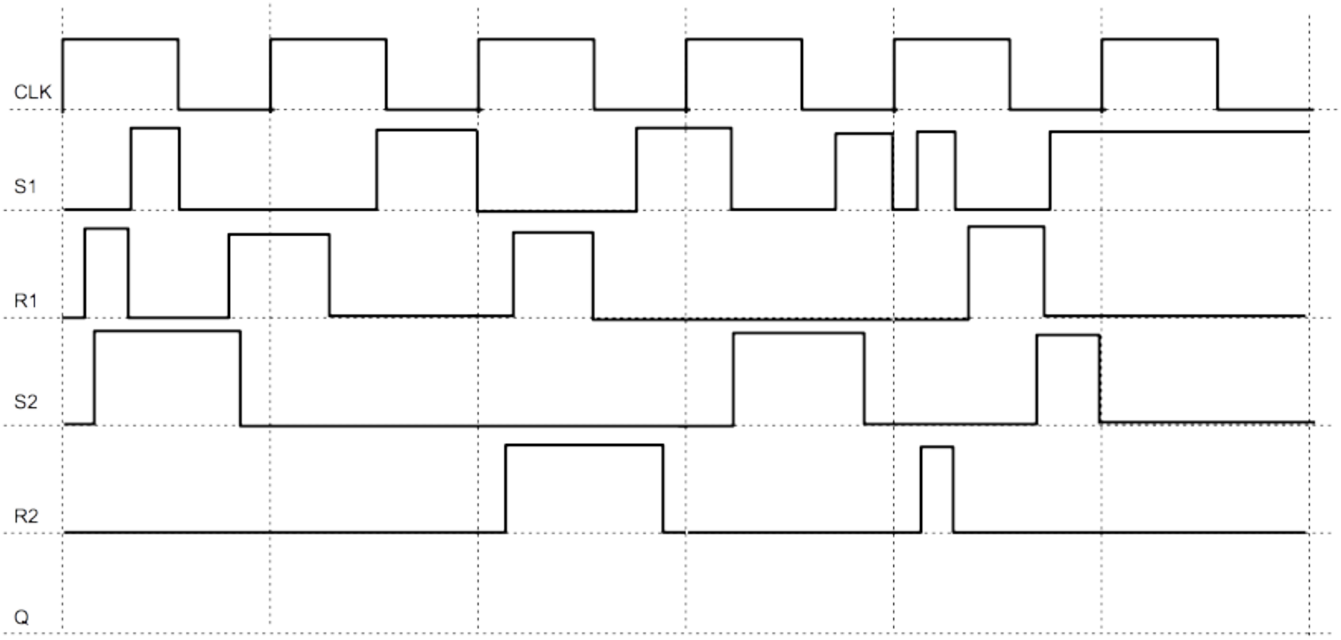
Using only the circuit from part a and standard logic gates (and/or/not/xor/nand/nor), draw a "counter" that counts the sequence 3, 4, 3, 4, 3, 4, ...

[Use the space below for rough work. This page will not be marked unless you clearly indicate the part of your work that you want us to mark.]

Question 2. [6 MARKS]

Complete the timing diagram below





[Use the space below for rough work. This page will not be marked unless you clearly indicate the part of your work that you want us to mark.]

Question 3. [5 MARKS]

Use booth's algorithm to calculate -15 * 10. Show your work.

[Use the space below for rough work. This page will not be marked unless you clearly indicate the part of your work that you want us to mark.]

Question 4. [10 MARKS]

```
.data
A:      .asciiz "I love CSC258!!"
B:      .asciiz "I like assembly"
C:      .asciiz "XXXXXXXXXXXXXXX"

.text
main:   add $t0, $zero, $zero
        addi $t1, $zero, 40
        la $t7, A
        la $t8, B
        la $t9, C
label1: add $t4, $t7, $t0
        add $t5, $t8, $t0
        add $t6, $t9, $t0
        lb $s4, 0($t4)
        lb $s5, 0($t5)
        beq $s4, $s5, label2
        sb $s4, 0($t6)
label2: addi $t0, $t0, 1
        bne $t0, $t1, label1
        li $v0, 4
        la $a0, C
        syscall
end:
```

Part (a) [6 MARKS]

Provide comments for the code above

Part (b) [4 MARKS]

What is printed to the console when this code is run?

[Use the space below for rough work. This page will not be marked unless you clearly indicate the part of your work that you want us to mark.]

Question 5. [15 marks]

Part (a) [8 marks]

In the space below, write an assembly function IS_MULT which takes two parameters a and b, and returns 1 iff a is a multiple of b, otherwise it returns a 0. To make things interesting, you **may not** use multiplication or division. Remember that no marks will be given for uncommented code.

Part (b) [7 marks]

In the space below, write an assembly program which allocates two arrays A and B of 10 integers each, and then uses your function above (assuming it is in the same file) to fill a boolean array C, using the logic C[i] = IS_MULT(A[i], B[i]).

[Use the space below for rough work. This page will not be marked unless you clearly indicate the part of your work that you want us to mark.]

MIPS Reference

Machine Encoding Aids

Key

o/f	instruction/function opcodes
s/t/d	first/second/third register
a/i	shift amount/immediate

Instruction Encoding Formats

Register	000000ss sssttttt dddddaaa aaffffff
Immediate	ooooooss sssttttt iiiiiiii iiiiiiii
Jump	ooooooii iiiiiiii iiiiiiii iiiiiiii

Instruction Syntax

Encoding	Syntax	Template
Register	ArithLog	f $d, $s, $t
	DivMult	f $s, $t
	Shift	f $d, $t, a
	ShiftV	f $d, $t, $s
	JumpR	f $s
	MoveFrom	f $d
	MoveTo	f $s
Immediate	ArithLogI	o $t, $s, i
	LoadI	o $t, immed32
	Branch	o $s, $t, label
	BranchZ	o $s, label
	LoadStore	o $t, i($s)
Jump	Jump	o label
	Trap	o i

Instruction Reference

Arithmetic and Logical Instructions

Instruction	Operation	Opcode or Function	Syntax	Comments
add $d, $s, $t	$d = $s + $t	100000	ArithLog	
addu $d, $s, $t	$d = $s + $t	100001	ArithLog	
addi $t, $s, i	$t = $s + i	001000	ArithLogI	i is sign-extended
addiu $t, $s, i	$t = $s + i	001001	ArithLogI	i is sign-extended
and $d, $s, $t	$d = $s & $t	100100	ArithLog	
andi $t, $s, i	$t = $s & i	001100	ArithLogI	i is zero-extended
div $s, $t	lo = $s / $t; hi = $s % $t	011010	DivMult	
divu $s, $t	lo = $s / $t; hi = $s % $t	011011	DivMult	
mult $s, $t	hi:lo = $s * $t	011000	DivMult	
multu $s, $t	hi:lo = $s * $t	011001	DivMult	
nor $d, $s, $t	$d = ~($s | $t)	100111	ArithLog	
or $d, $s, $t	$d = $s | $t	100101	ArithLog	
ori $t, $s, i	$t = $s | i	001101	ArithLogI	i is zero-extended
sll $d, $t, a	$d = $t << a	000000	Shift	Zero is shifted in
sllv $d, $t, $s	$d = $t << $s	000100	ShiftV	Zero is shifted in
sra $d, $t, a	$d = $t >> a	000011	Shift	Sign bit is shifted in
srav $d, $t, $s	$d = $t >> $s	000111	ShiftV	Sign bit is shifted in
srl $d, $t, a	$d = $t >> a	000010	Shift	Zero is shifted in
srlv $d, $t, $s	$d = $t >> $s	000110	ShiftV	Zero is shifted in
sub $d, $s, $t	$d = $s - $t	100010	ArithLog	
subu $d, $s, $t	$d = $s - $t	100011	ArithLog	
xor $d, $s, $t	$d = $s ^ $t	100110	ArithLog	
xori $d, $s, i	$d = $s ^ i	001110	ArithLogI	i is zero-extended

Movement Instructions

Instruction	Operation	Opcode or Function	Syntax	Comments
lhi $t, i	$t = i << 16	011001	LoadI	i is zero-extended
llo $t, i	$t = i	011000	LoadI	i is zero-extended
mfhi $d	$d = hi	010000	MoveFrom	
mflo $d	$d = lo	010010	MoveFrom	
mthi $s	hi = $s	010001	MoveTo	
mtlo $s	lo = $s	010011	MoveTo	

Comparison Instructions

Instruction	Operation	Opcode or Function	Syntax	Comments
slt $d, $s, $t	$d = $s < $t	101010	ArithLog	
sltu $d, $s, $t	$d = $s < $t	101001	ArithLog	
slti $t, $s, i	$d = $s < i	001010	ArithLogI	i is sign-extended
sltiu $t, $s, i	$d = $s < i	001001	ArithLogI	i is sign-extended

Branch and Jump Instructions

Instruction	Operation	Opcode or Function	Syntax	Comments
beq $s, $t, label	if ($s == $t) pc += i << 2	000100	Branch	label is a line reference in the code
bgtz $s, label	if ($s > 0) pc += i << 2	000111	BranchZ	label is a line reference in the code
blez $s, label	if ($s <= 0) pc += i << 2	000110	BranchZ	label is a line reference in the code
bne $s, $t, label	if ($s != $t) pc += i << 2	000101	Branch	label is a line reference in the code
j label	pc += i << 2	000010	Jump	label is a line reference in the code
jal label	$ra = pc; pc += i << 2	000011	Jump	label is a line reference in the code
jalr $s	$ra = pc; pc = $s	001001	JumpR	
jr $s	pc = $s	001000	JumpR	

Memory Instructions

Instruction	Operation	Opcode or Function	Syntax	Comments
lb $t, i($s)	$t = MEM[$s + i]	100000	LoadStore	Sign-extends the loaded byte
lbu $t, i($s)	$t = MEM[$s + i]	100100	LoadStore	Zero-extends the loaded byte
lh $t, i($s)	$t = MEM[$s + i]	100001	LoadStore	Sign-extends the loaded bytes
lhu $t, i($s)	$t = MEM[$s + i]	100101	LoadStore	Zero-extends the loaded bytes
lw $t, i($s)	$t = MEM[$s + i]	100011	LoadStore	
sb $t, i($s)	MEM[$s + i] = $t	101000	LoadStore	Lowest order byte is stored
sh $t, i($s)	MEM[$s + i] = $t	101001	LoadStore	2 lowest order bytes are stored
sw $t, i($s)	MEM[$s + i] = $t	101011	LoadStore	

Exception and Interrupt Instructions

Instruction	Operation	Opcode or Function	Syntax	Comments
trap i	Exception	0011010	Trap	i is a trap code; implements syscall