CSC B58 Winter 2018 Final Exam
Duration — 2 hours and 50 minutes
Aids allowed: none

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

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

Last Name: ______________________ **First Name:** ______________________

Question 0. [1 MARK]

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

Do ***not*** *turn this page until you have received the signal to start.*
(Please fill out the identification section above, **write your name on the back of the test**, and read the instructions below.)
Good Luck!

This exam is double-sided, and consists of 5 questions on 18 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.
- Do not remove any pages from the exam booklet.
- Read all instructions before completing any questions
- Write your name and student number on the back of the last page
- You may leave any question blank except for the words "I don't know" to receive 10% (rounded up to the nearest half-integer) for that question

0: _______/ 1

1: _______/ 8

2: _______/ 6

3: _______/ 5

4: _______/20

5: _______/30

TOTAL: _______/70

[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. [8 MARKS]

Answer the following questions in the space provided. Your answers should be as simple and concise as possible.

Part (a) [2 MARKS]

What are semiconductors, and why are they so important for modern computer science?

Part (b) [2 MARKS]

If the number of 1s and 0s in a truth table is roughly equal, we tend to prefer a SOM rather than a POM equation. Why would this be?

Part (c) [2 MARKS]

In assembly without delayed loads, you may sometimes see code that has `NOOP` (instructions which don't perform any operations) following `lw` instructions. Why is this?

Part (d) [2 MARKS]

Draw a diagram (with source, ground and transistors) of a circuit with the following truth table. Your circuit should be as simple as possible.

A	OUT
0	1
1	?

[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]

Answer the following questions in the space provided.

Part (a) [2 MARKS]

How would you represent `35d` in binary? Show your answer for both signed and unsigned numbers.

How would you represent `-35d` in binary? Show your answer for both signed and unsigned numbers.

Part (b) [2 MARKS]

Calculate `35d` - 51d by converting both numbers to binary and using binary subtraction. Show your work.

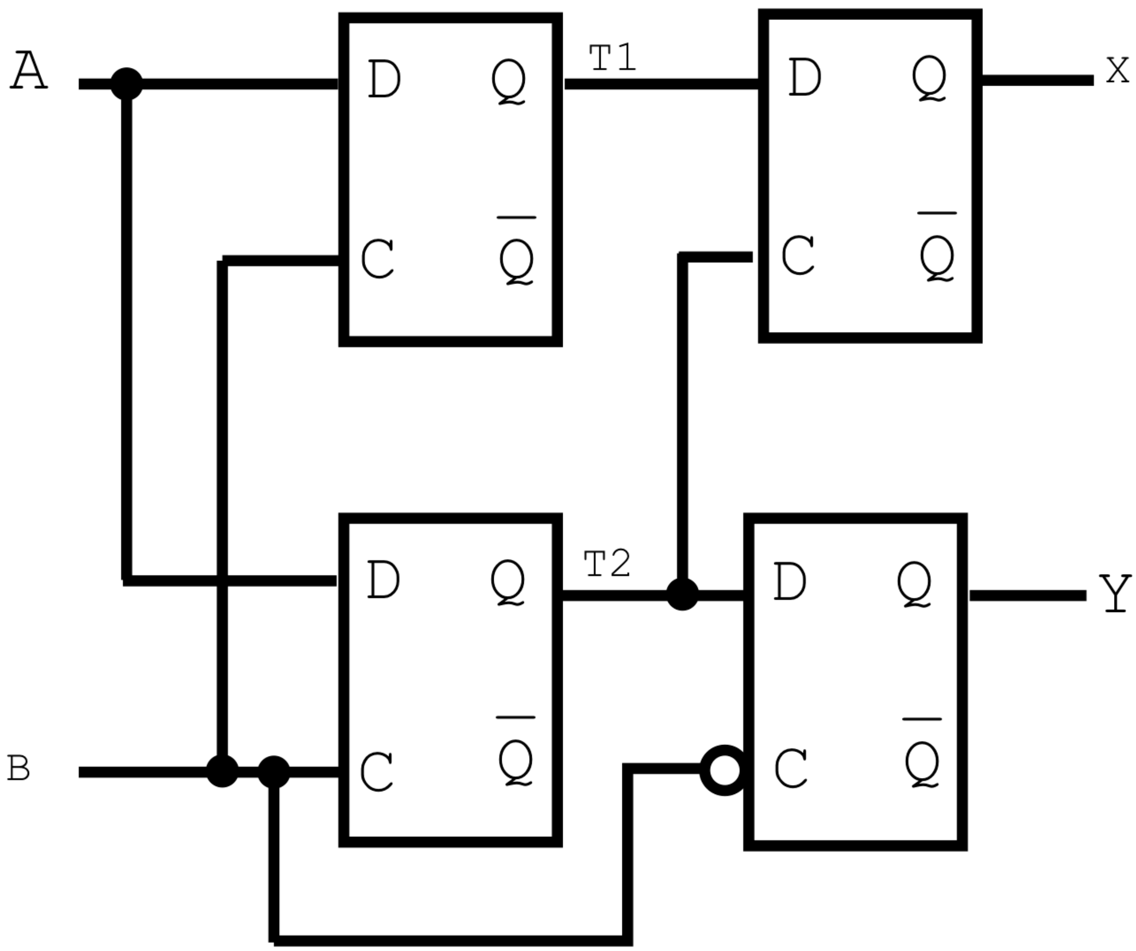
Part (c) [2 MARKS]

Calculate `99900000099000d` * `3`. Using the technique shown in class, you do not need to convert to binary, but you do need to 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 3. [5 MARKS]

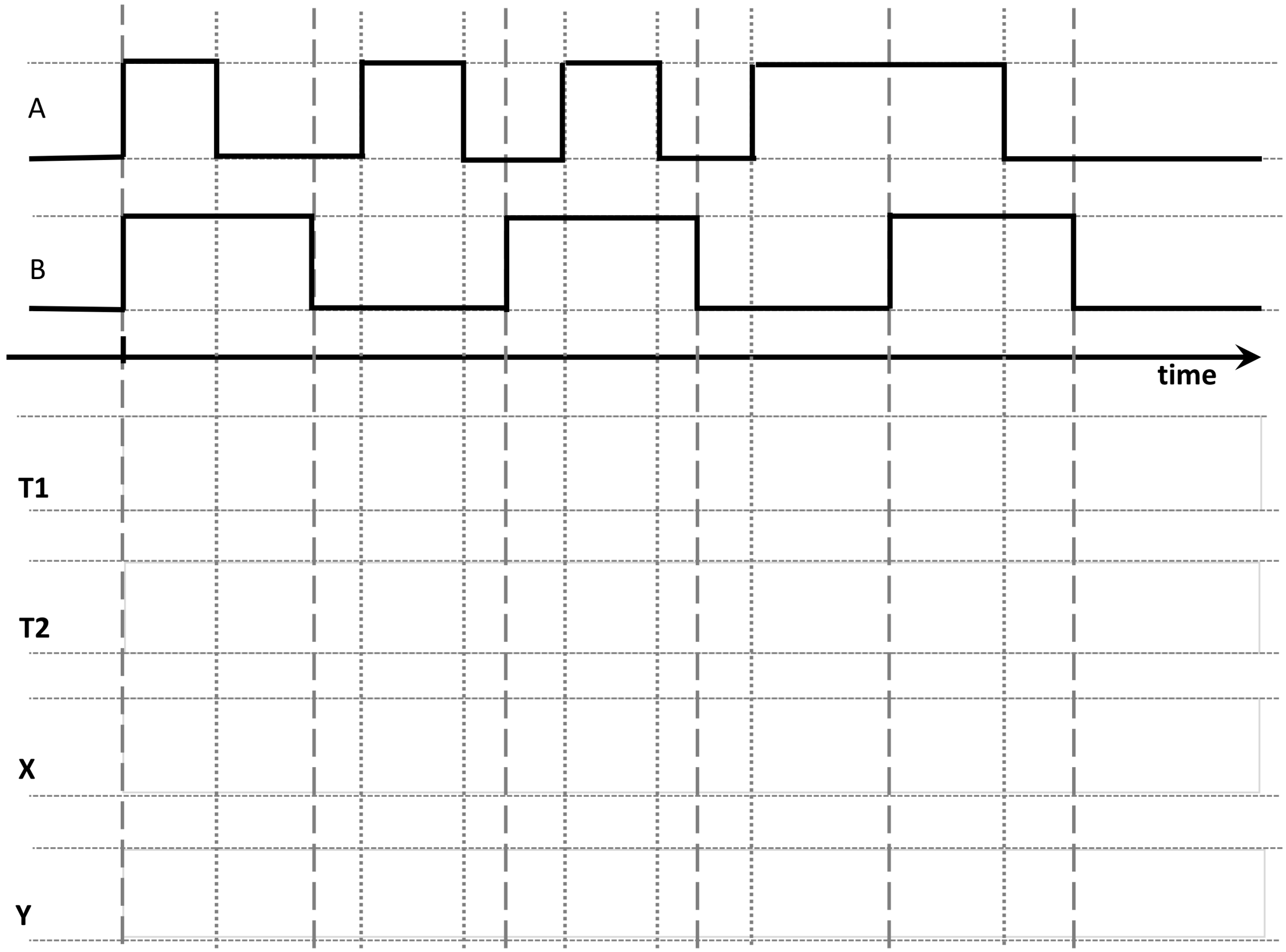
Given only HA and FA gates, along with the standard primitive gates (AND, OR, NAND, NOR, XOR, and NOT). Construct a 4 bit adder-subtractor.



Note: All blocks shown are D-latches

Question 4. [20 MARKS]

Complete the timing diagram for the gates shown on the opposite page.



[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. [30 marks]

For this question, no marks will be given for code without proper comments. If your code does not work, partial marks may be awarded for a well designed flow chart.

Part (a) [10 marks]

Write an assembly function called `max_min` that takes the address of a list as a parameter and uses a loop (not recursion) to find the maximum and minimum values in that list.

[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 (b) [10 MARKS]

Write an assembly function called `rec_max_min` that takes the address of a list as a parameter and usesrecursion to find the maximum and minimum values in that list.

[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) [10 MARKS]

Write some global assembly code that declares a list of integers (assume they are filled with data by some other process), calls `max_min` and `rec_max_min` on the list, and prints `SUCCESS` if the two functions return the same values and `FAILURE` otherwise.

[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 Sheet

You may remove this sheet, nothing on this page will be marked

Arithmetic Instructions

Instruction	Opcode/Function	Syntax	Operation
add	100000	$d, $s, $t	$d = $s + $t
addu	100001	$d, $s, $t	$d = $s + $t
addi	001000	$t, $s, i	$t = $s + SE(i)
addiu	001001	$t, $s, i	$t = $s + SE(i)
div	011010	$s, $t	lo = $s / $t; hi = $s % $t
divu	011011	$s, $t	lo = $s / $t; hi = $s % $t
mult	011000	$s, $t	hi:lo = $s * $t
multu	011001	$s, $t	hi:lo = $s * $t
sub	100010	$d, $s, $t	$d = $s - $t
subu	100011	$d, $s, $t	$d = $s - $t

Logical Instructions

Instruction	Opcode/Function	Syntax	Operation
and	100100	$d, $s, $t	$d = $s & $t
andi	001100	$t, $s, i	$t = $s & ZE(i)
nor	100111	$d, $s, $t	$d = ~($s | $t)
or	100101	$d, $s, $t	$d = $s | $t
ori	001101	$t, $s, i	$t = $s | ZE(i)
xor	100110	$d, $s, $t	$d = $s ^$t
xori	001110	$t, $s, i	$t = $s ^ZE(i)

Shift Instructions

Instruction	Opcode/Function	Syntax	Operation
sll	000000	$d, $t, a	$d = $t << a
sllv	000100	$d, $t, $s	$d = $t << $s
sra	000011	$d, $t, a	$d = $t >> a
srav	000111	$d, $t, $s	$d = $t >> $s
srl	000010	$d, $t, a	$d = $t >>> a
srlv	000110	$d, $t, $s	$d = $t >>> $s

Data Movement Instructions

Instruction	Opcode/Function	Syntax	Operation
mfhi	010000	$d	$d = hi
mflo	010010	$d	$d = lo
mthi	010001	$s	hi = $s
mtlo	010011	$s	lo = $s

Branch Instructions

Instruction	Opcode/Function	Syntax	Operation
beq	000100	$s, $t, label	if ($s == $t) pc <- label
bgtz	000111	$s, label	if ($s > 0) pc <- label
blez	000110	$s, label	if ($s <= 0) pc <- label
bne	000101	$s, $t, label	if ($s != $t) pc <- label

Jump Instructions			
Instruction	Opcode/Function	Syntax	Operation
j	000010	label	pc <- label
jal	000011	label	$ra = pc; pc <- label
jalr	001001	$s	$ra = pc; pc = $s
jr	001000	$s	pc = $s

Comparison Instructions			
Instruction	Opcode/Function	Syntax	Operation
slt	101010	$d, $s, $t	$d = ($s < $t)
sltu	101001	$d, $s, $t	$d = ($s < $t)
slti	001010	$t, $s, i	$t = ($s < SE(i))
sltiu	001001	$t, $s, i	$t = ($s < SE(i))

Memory Instructions			
Instruction	Opcode/Function	Syntax	Operation
lb	100000	$t, i ($s)	$t = SE (MEM [$s + i]:1)
lbu	100100	$t, i ($s)	$t = ZE (MEM [$s + i]:1)
lh	100001	$t, i ($s)	$t = SE (MEM [$s + i]:2)
lhu	100101	$t, i ($s)	$t = ZE (MEM [$s + i]:2)
lw	100011	$t, i ($s)	$t = MEM [$s + i]:4
sb	101000	$t, i ($s)	MEM [$s + i]:1 = LB ($t)
sh	101001	$t, i ($s)	MEM [$s + i]:2 = LH ($t)
sw	101011	$t, i ($s)	MEM [$s + i]:4 = $t

Pseudo Instructions			
Instruction	Opcode/Function	Syntax	Operation
la	N/A	$t, label	$t = SE (MEM [label]:1)
li	N/A	$t, i	$t = i
syscall	N/A		Call system trap, trapcode is in $v0

Trap Codes		
Service	Trap Code	Input/Output
print_int	1	$a0 is int to print
print_string	4	$a0 is address of ASCIIZ string to print
read_int	5	$a0 is int read
read_string	8	$a0 is address of buffer, $a1 is buffer size in bytes
exit	10	