CSC B58 Summer 2017 Final Examination
Duration — 2 hours and 50 minutes
Aids allowed: none

Student Number: |_|_|_|_|_|_|_|_|_|_| **A**

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)
Good Luck!

This exam is double-sided, and consists of 4 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.
- All code must include full documentation. Undocumented code will not be graded.

0: _____/ 1
1: _____/ 9
2: _____/10
3: _____/10
4: _____/20

TOTAL: _____/50

Total Pages = 18

[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. [9 MARKS]

Part (a) [2 MARKS]

Briefly explain what doping means in the context of this course.

Part (b) [2 MARKS]

Briefly explain why D flip-flops are used instead of S-R flip-flops in this course.

Part (c) [2 MARKS]

Briefly explain why I can't simply set my system clock to a faster rate in order to speed up my computer.

Part (d) [2 MARKS]

Briefly explain why we need to use the stack to pass values into/out of functions in assembly.

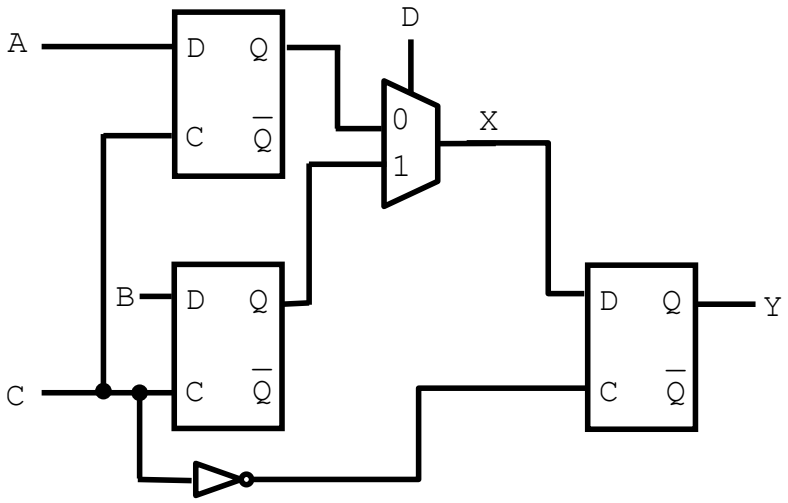
Part (e) [1 MARK]

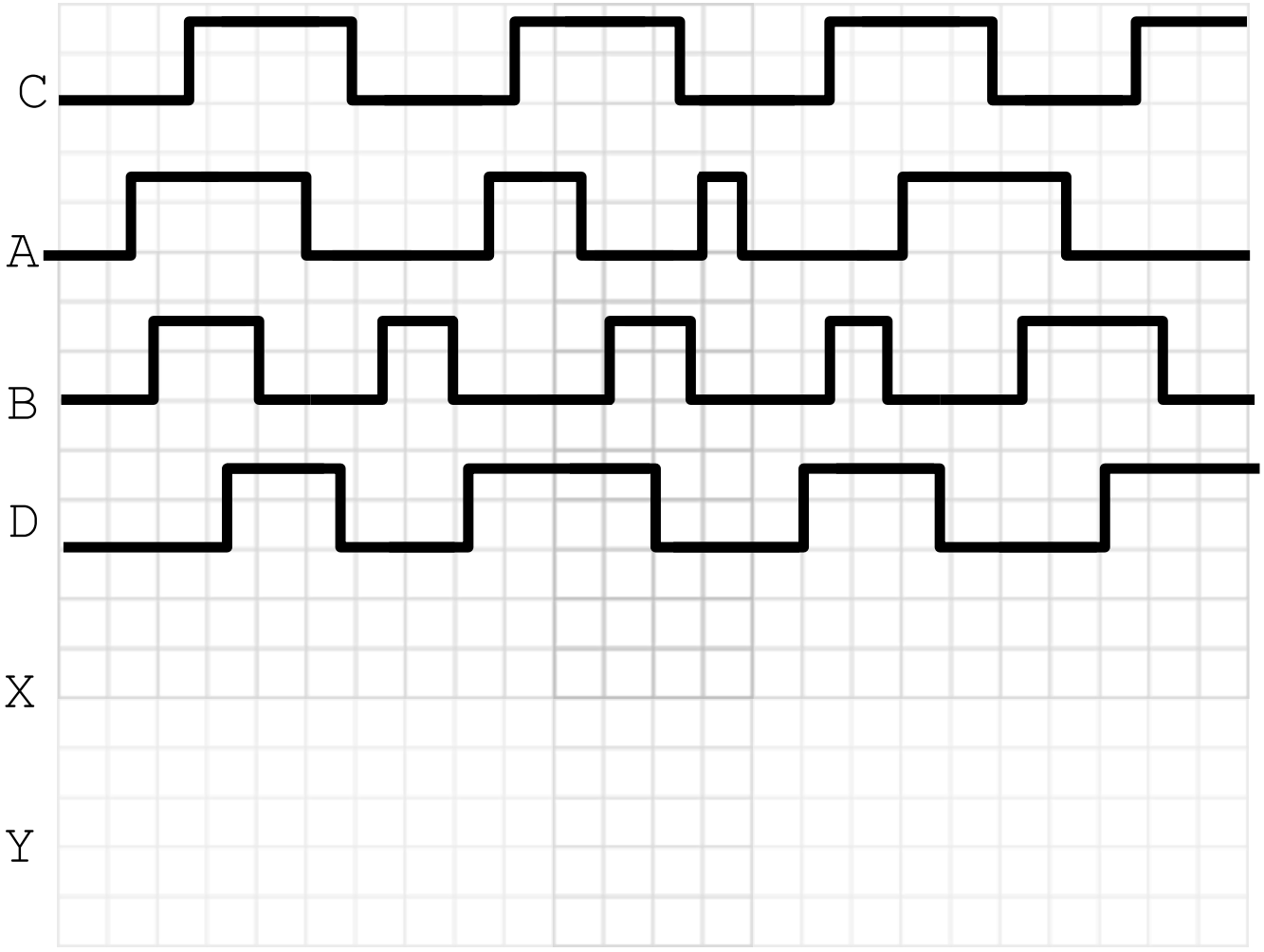
Briefly explain why Tickle Me Elmo was relevant to this course.

[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. [10 MARKS]

Making no assumptions about the starting state of the wires, complete the following timing diagram. The block diagrams represent standard D latches.





[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. [10 MARKS]

You've been tasked with building a simple elevator system for a 3 story building. Each floor has a button, if that floor's button is pressed, the elevator will move to that floor. If the button is pressed for the current floor, nothing happens. If two buttons are pressed simultaneously, nothing happens. If all three buttons are pressed simultaneously, the elevator goes straight to the bottom floor.

Part (a) [2 MARKS]

Draw a FSM for this system. Give the states meaningful names, do not worry about flipflop assignments yet. Marks will be deducted for poorly laid out or difficult to read models.

Part (b) [3 MARKS]

Draw a FSM for this system showing the flip-flop assignments for all states.

[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) [5 MARKS]

Derive a series of boolean expressions for your FSM. Show all 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. [20 MARKS]

Part (a) [8 MARKS]

In the space below, write an assembly function called `sum` that takes the address of a list of positive integers, and returns the sum of all the elements in that list. Remember that undocumented/poorly commented code will not be marked.

[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) [8 MARKS]

In the space below, write an assembly function called `rec_sum` that performs the same task as `sum` from the previous part of the question, but does so recursively. (If you implemented `sum` recursively, write this one iteratively and swap the names around).

Part (c) [4 MARKS]

In the space below, write some code that creates a list, finds the sum with the two functions you just wrote, and if they both return the same sum, prints out `Success`, otherwise prints out `Failure`.

[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.]

Bonus. (CONTINUED)

Predict your score on each question of this exam: Get within 10% to get this bonus mark.

Q0: /1
Q1: /9
Q2: /10
Q3: /10
Q4: /20

Bonus. (CONTINUED)

What would you recommend we do to change/update/improve this course in the future?

Bonus. (CONTINUED)

If you could send a message back in time to yourself on the first day of term. What would that message be (no winning lottery numbers allowed)? ‘

Bonus. (CONTINUED)

Draw something/write something/tell us a joke. Make at least one TA laugh or smile to get this bonus 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	$v0 is int read
read_string	8	$a0 is address of buffer, $a1 is buffer size in bytes
exit	10	