CSC B58 Winter 2017 Final Examination
Duration — 2 hours and 50 minutes
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 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 7 questions on 20 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.
- Draw a smiley face in the bottom right corner of this page
- Do not remove any pages from the exam booklet.
- Don't draw a smiley face, instead write "Hi Brian" in the bottom right corner of this page (good thing you kept reading huh?)
- All code must include full documentation. Undocumented code will not be graded.

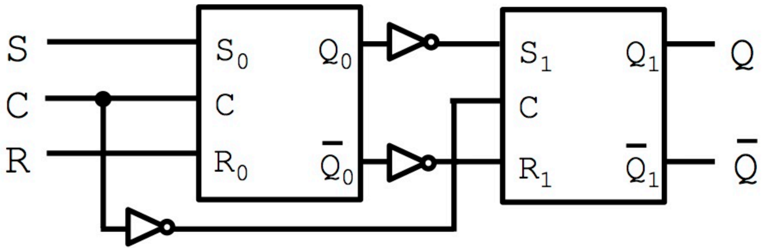
# 0:	______/ 1
# 1:	______/ 5
# 2:	______/ 5
# 3:	______/ 4
# 4:	______/ 4
# 5:	______/ 6
# 6:	______/12
# 7:	______/13
TOTAL:	______/50

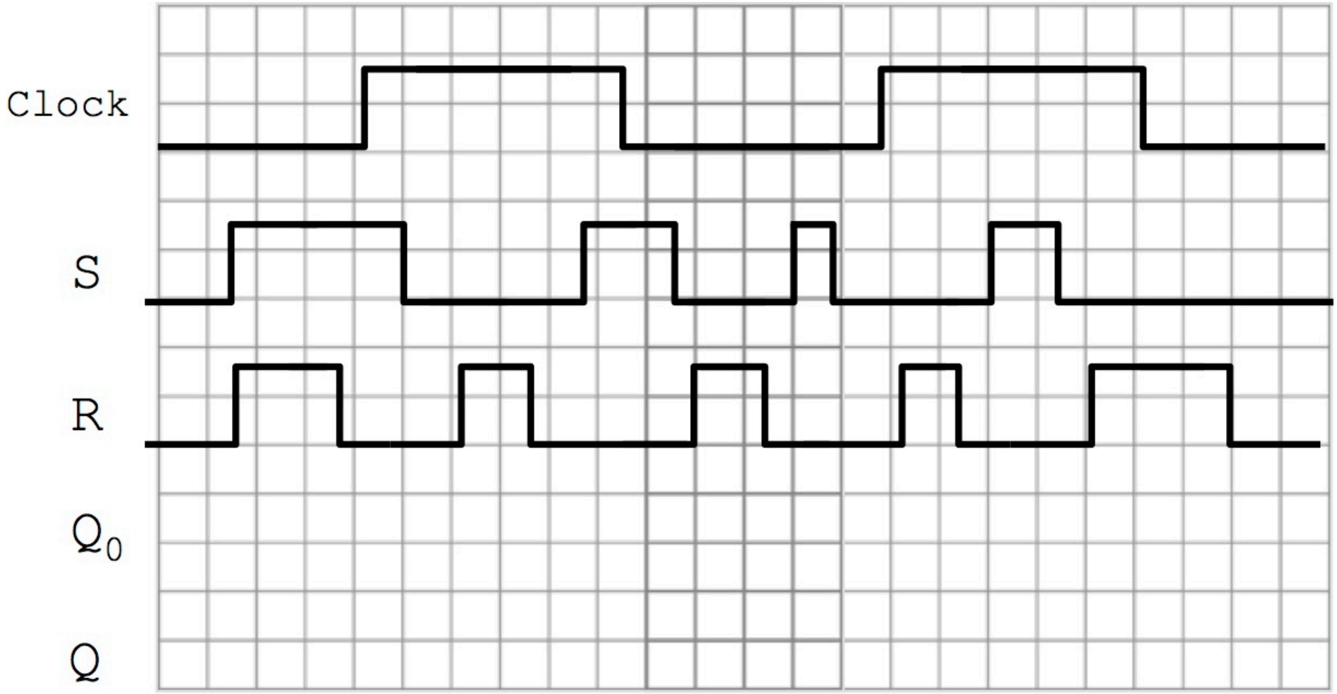
Total Pages = 20

[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. [5 MARKS]

Assuming that Q_0 starts low, complete the following timing diagram





[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. [5 marks]

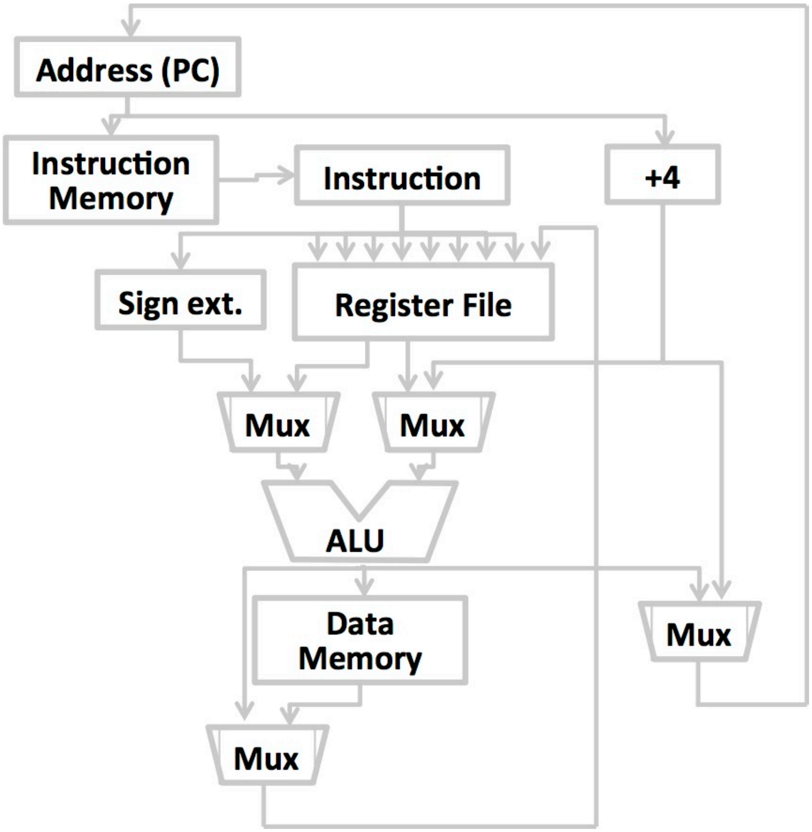
Use Booth's Algorithm to compute $-43 * 37$ (numbers given in decimal). Show all your work.

[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 3. [4 MARKS]

In the image below, highlight the datapath for the following instruction:

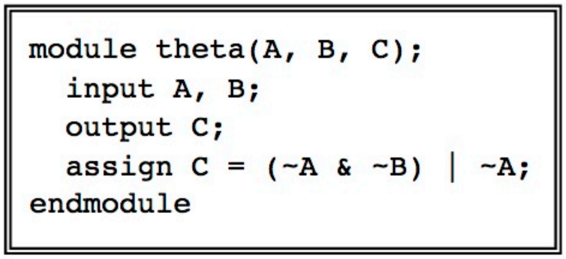
`bgtz $t0, LABEL1`



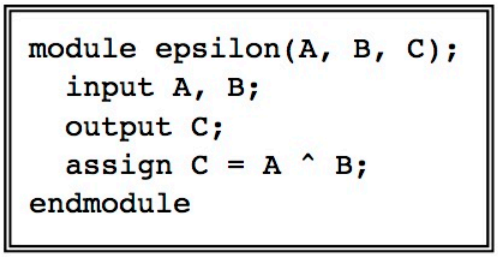
[Use the space below for rough work. This page will not be marked unless you clearly indicate the part of your work that you want us to mark.]

Question 4. [4 MARKS]

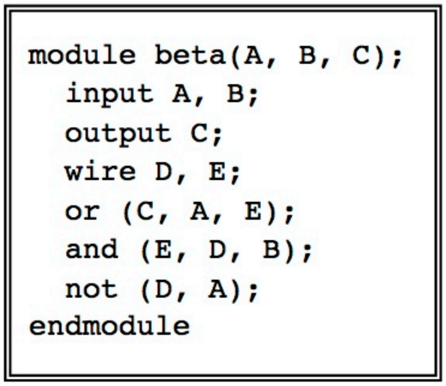
Draw lines connecting the Verilog modules which have equivalent behaviour



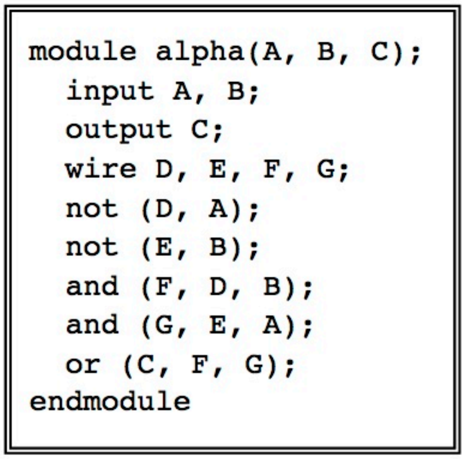
```
module theta(A, B, C);
  input A, B;
  output C;
  assign C = (~A & ~B) | ~A;
endmodule
```



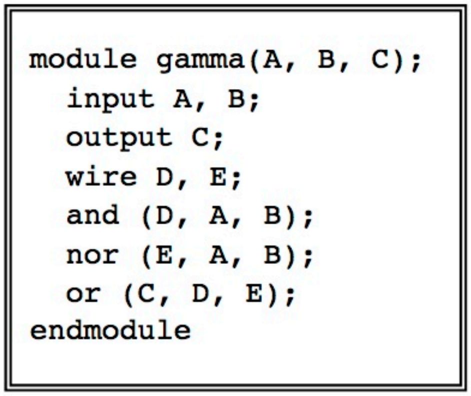
```
module epsilon(A, B, C);
  input A, B;
  output C;
  assign C = A ^ B;
endmodule
```



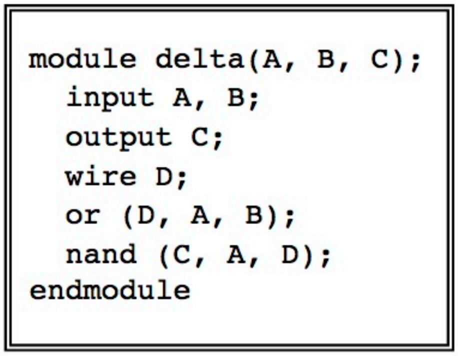
```
module beta(A, B, C);
  input A, B;
  output C;
  wire D, E;
  or (C, A, E);
  and (E, D, B);
  not (D, A);
endmodule
```



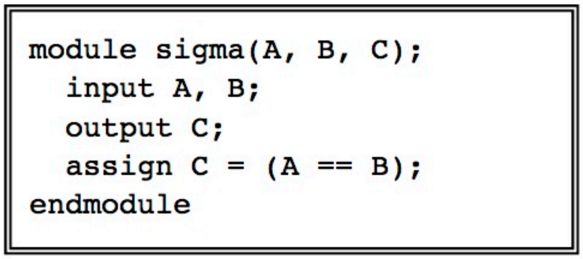
```
module alpha(A, B, C);
  input A, B;
  output C;
  wire D, E, F, G;
  not (D, A);
  not (E, B);
  and (F, D, B);
  and (G, E, A);
  or (C, F, G);
endmodule
```



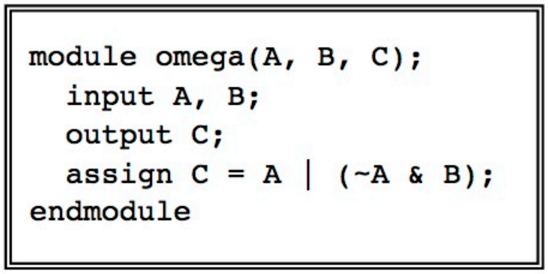
```
module gamma(A, B, C);
  input A, B;
  output C;
  wire D, E;
  and (D, A, B);
  nor (E, A, B);
  or (C, D, E);
endmodule
```



```
module delta(A, B, C);
  input A, B;
  output C;
  wire D;
  or (D, A, B);
  nand (C, A, D);
endmodule
```



```
module sigma(A, B, C);
  input A, B;
  output C;
  assign C = (A == B);
endmodule
```



```
module omega(A, B, C);
  input A, B;
  output C;
  assign C = A | (~A & B);
endmodule
```

[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. [6 MARKS]

Consider the following Verilog module:

```
module mystery (Q, D, L, E, C, R);
    input [7:0] D;
    input L, E, C, R;
    output reg [7:0] Q;

    always @posedge C, negedge R)
    if (~R)
        Q <= 0;
    else if (L):
        Q <= D;
    else if (E)
        Q <= Q + 1;
endmodule
```

Part (a) [3 MARKS]

In one sentence, what does the module do?

Part (b) [3 MARKS]

What is the purpose/function of each of the following signals?

- Q
- D
- L
- E
- C
- R

.

Question 6. [12 MARKS]

Part (a) [4 MARKS]

In the opposite page, draw the flow-chart for a function `between` that takes 3 parameters, max, min and x (in that order), and returns 1 if MIN $\leq$ x $\leq$ MIN, and 0 otherwise.

`<-- Your flow chat goes there`

Part (b) [8 MARKS]

In the space below, write the assembly code for `between` including any data declarations and all comments and labels.

[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 7. [13 marks]

Assuming you have a properly designed and coded function called `is_vowel`, which takes the ascii value of a letter as input, and returns 1 if that letter is a vowel, and 0 otherwise. Write a program that declares a string, replaces all of the vowels in that string with the letter 'X', and then prints the result to the console. You must include all data declarations and complete comments.

[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	$v0 is int read
read_string	8	$a0 is address of buffer, $a1 is buffer size in bytes
exit	10	