UNIVERSITY OF TORONTO

Faculty of Arts and Science

April 2014 Examinations

CSC258H1S: Computer Organization

Duration: 3 hours

Permitted Aids: one ruler

Last Name: _____

First Name: _____

Student Number: _____

Instructor: Steve Engels

Instructions:

- **Write your name on every page of this exam.**
- **Do not open this exam until you hear the signal to start.**
- Have your student ID on your desk.
- No aids permitted other than writing tools. Keep all bags and notes far from your desk before the exam begins.
- There are 6 questions on 14 pages. When you hear the signal to start, make sure that your exam is complete before you begin.
- Read over the entire exam before starting.
- If you use any space for rough work or have to user the overflow page, clearly indicate the section(s) that you want marked.

Mark Breakdown

Part A:	/ 21
Part B:	/ 18
Part C:	/ 27
Part D:	/ 38
Part E:	/ 14
Part F:	/ 24
Bonus:	/ 1
Total:	**/ 142**

Part A: Short Answer (21 marks)

Answer the following questions in the space provided. When providing a written answer, write **as clearly and legibly as possible**. Marks will not be awarded to unreadable answers.

1. How many bytes can be stored in a memory unit that uses 8 address bits and a 16-bit architecture? **(2 marks)**

a) `4096` **b)** `1024`

c) `128` **d)** `256`

2. How many address bits are needed to specify the location of an integer in a 1024 byte memory unit, given a 32-bit architecture? **(2 marks)**

a) `32` **b)** `1024`

c) `10` **d)** `5`

3. What does the assembly language command `rfe` stand for? **(1 mark)**

4. True or False? Doping a semiconductor involves adding impurities in the form of atoms from neighbouring groups in the periodic table. **(1 mark)**

True **False**

5. Which of the following assembly instructions use the processor's shift left units? Circle all that apply. **(2 marks)**

a) `jal` **b)** `bne`

c) `lb` **d)** `sub`

6. Which of the following assembly instructions use the processor's sign extend unit? Circle all that apply. **(3 marks)**

a) `addu` **b)** `addi` **c)** `beq`

d) `lw` **e)** `sw` **f)** `srl`

7. Which ALU output signal triggers an exception condition? **(1 mark)**

8. True or False? Counters are asynchronous circuits. **(1 mark)**

True **False**

9. Assuming that `A` and `B` are one-bit wire values, which of the following Verilog statements will cause `B` to have a high value at all times? Circle all that apply. **(2 marks)**

a) `xor(B, A, ~A);` **b)** `nand(B, A, ~A);`

c) `or(B, A, ~A);` **d)** `nor(B, A, ~A);`

10. If a control unit implements a finite state machine with 250 states, what is the minimum number of flip-flops that you would need to implement it in a circuit? **(1 mark)**

11. What is the maximum distance that a program counter can traverse in memory, as a result of a jump instruction? **(1 mark)**

a) 2^{16} bytes **b)** 2^{18} bytes

c) 2^{26} bytes **d)** 2^{28} bytes

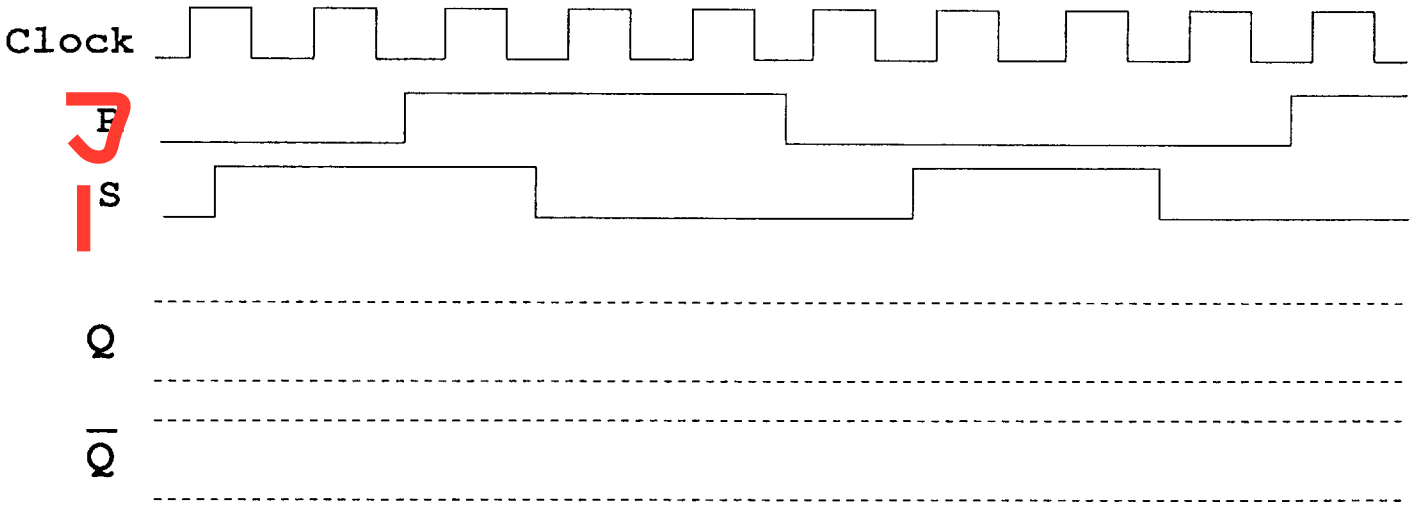
12. The three types of instructions are R-type, I-type and J-type. What do the R, I and J stand for? **(3 marks)**

R: _____ **I:** _____ **J:** _____

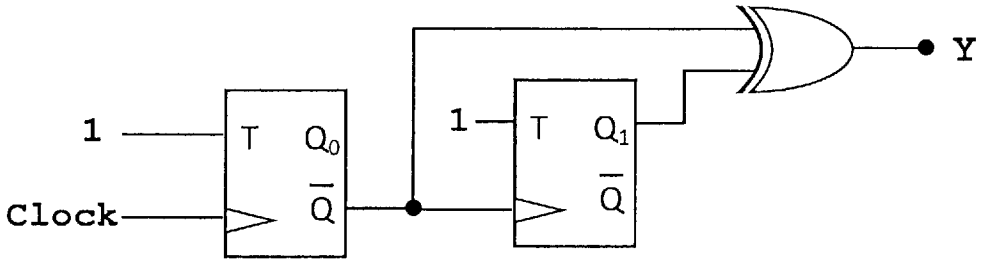
13. What register can be used as a source operand in assembly instructions, but not a destination? **(1 mark)**

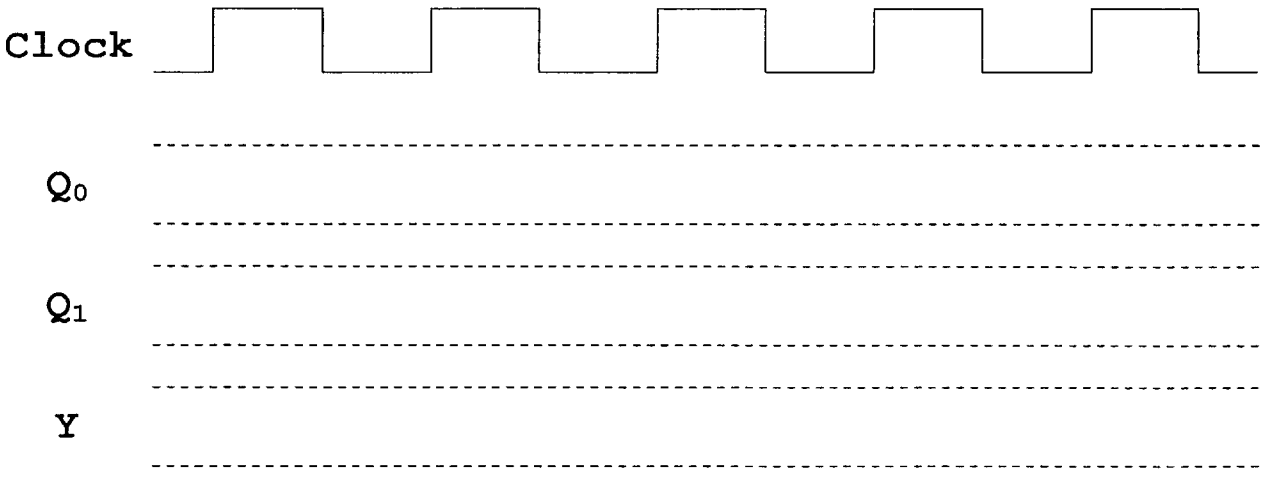
Part B: Design and Analysis (18 marks)

1. The waveforms below shows the input values to a JK flip-flop. Assuming that the initial value for Q is low, draw the output waveform that results, given the following input behaviour. **(4 marks)**

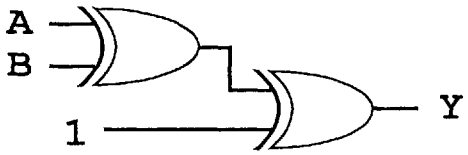


4. Given the following circuit, show what the output value of Q_0, Q_1 and Y will be in the waveform diagram. Assume that Q_0 and Q_1 start with initial values of zero. **(6 marks)**



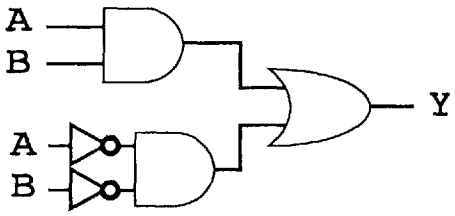


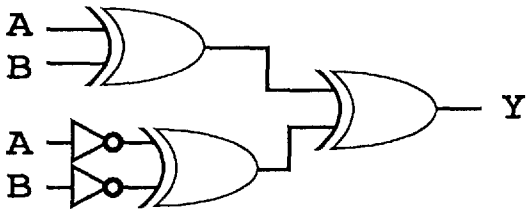
3. Match the circuits that perform the same logical operation by drawing lines that connect equivalent circuits in the space below. **(8 marks)**



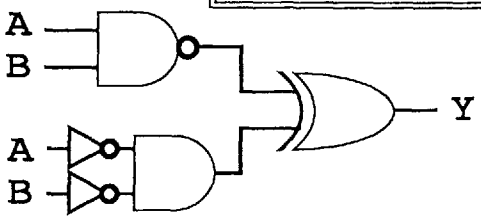

```
module aye(A, B, Y);
input A, B;
output Y;
wire V, W;
xor(W, A, B);
nand(V, A, B);
xor(Y, V, W);
endmodule
```

```
module bee(A, B, Y);
input A, B;
output Y;
wire V, W;
and(W, A, B);
nand(V, A, B);
or(Y, V, W);
endmodule
```






```
module cee(A, B, Y);
input A, B;
output Y;
wire V, W;
assign Y = (A & B) &
     (~A & ~B);
endmodule
```




```
module cee(A, B, Y);
input A, B;
output Y;
wire V, W;
and(W, A, B);
nand(V, A, B);
xor(Y, V, W);
endmodule
```

Part C: Processors (27 marks)

1. When Booth's Algorithm is performed on the binary inputs A=1101 and B=1011, the values for A and P change at each step of the algorithm. On the left hand side are possible values for A, and on the right hand side are possible value for P. Answer this question by doing the following:

- On the left, circle all the values for A that occur during Booth's algorithm. **(4 marks)**
- On the right, indicate which value for A correspond to this value for P (if any). Leave any invalid values of P blank. **(4 marks)**

You can use the space to show your work, but it's not required.

a) 10110

b) 11111

c) 11101

d) 01101

e) 11010

f) 11110

☐	00101000
☐	11101100
☐	00011110
☐	00011110
☐	00000000
☐	00101000
☐	11101100
☐	00001111

2. Below are errors that can arise when writing data values to memory. For each one, write the name and brief description of the timing delay that would solve the problem. One has been provided for you as an example. **(6 marks)**

Error #1: The write signal wasn't on long enough for the data to get set up in memory.

T_{AW} **Address width time (time to hold write signal high)**

Error #2: The data values came down before the write signal turned off.

_____ _____

Error #3: The write signal turned on before the address bits arrived.

_____ _____

Error #4: The data bits weren't high long enough before the write signal turned off.

_____ _____

Student Number: _____

3. Consider the datapath diagram below. The control unit signals have been removed. In the diagram below, draw lines to connect the control unit signal labels at the topwith the processor units that they affect on the bottom. **(13 marks)**

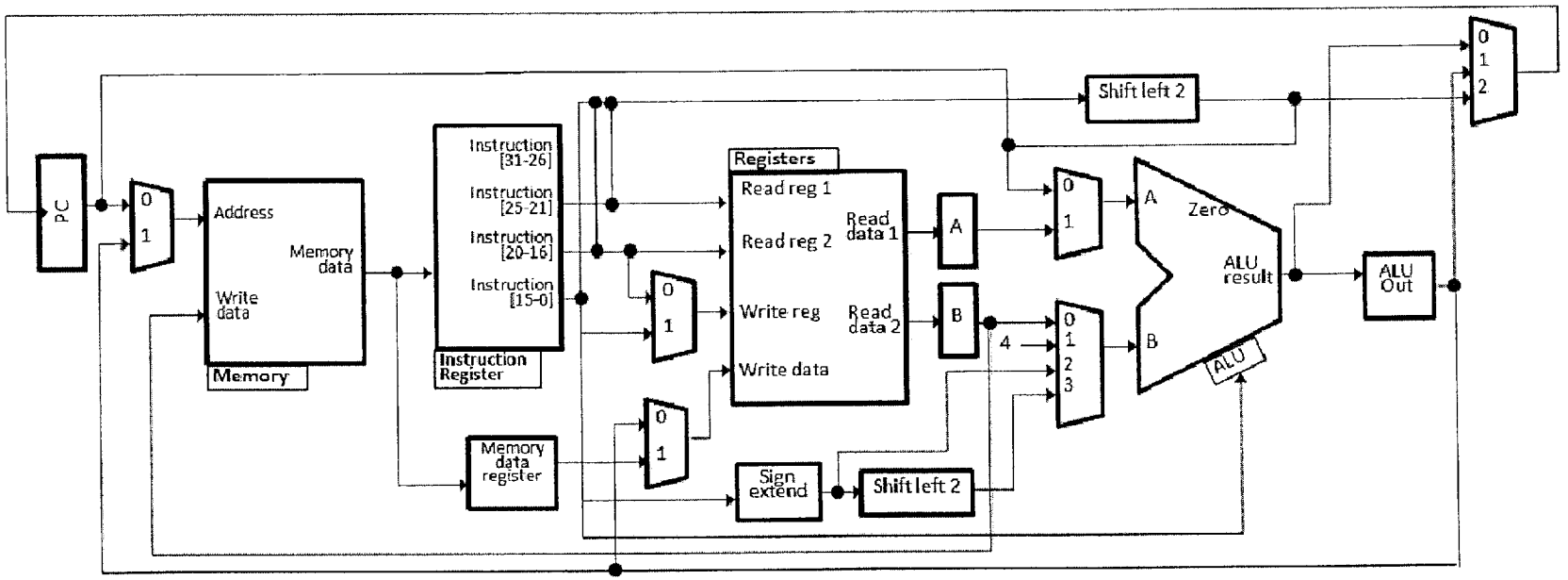
ALUSrcB | IorD | MemRead | MemToReg | MemWrite | PCWrite | PCSource

ALUSrcA

PCWriteCond

ALUOp

RegDst



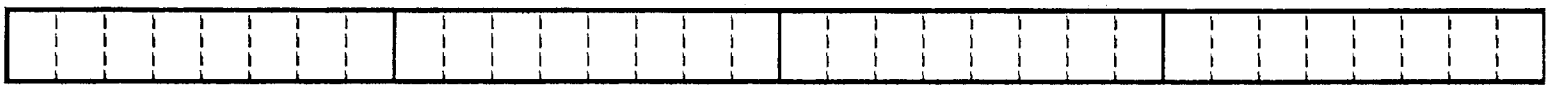
Part D: Processor Instructions (38 marks)

5. For the following assembly language instructions, write the equivalent machine code instruction in the space provided. You might find the reference information in the appendix helpful for this question. Fill in the space with an X if the value doesn't matter. **(14 marks)**

a) `andi $a0, $zero, $-16`

b) `jr $ra`

c) `sll $t7, $t0, 15`



6. For the following machine code instructions, provide the equivalent assembly language instruction in the space provided. **(9 marks)**

a) 00000011000010000100100000100101

b) 00000000010111110000000000001001

c) 10010001000010011111111111100001

7. For each of the processor tasks below, indicate what the values of the following control unit signals will be by filling in the boxes next to each signal with the signal values. **(15 marks)**

- If a control signal doesn't affect the operation, fill in its value with an X.
- For ALUOp, full marks will only be given for binary values. If you don't know what the values are, just write what kind of operation is taking place instead.

Add 4 to $t0 and store the result in $t1.

PCWrite ☐ PCWriteCond ☐ IorD ☐ MemRead ☐ MemWrite ☐
MemToReg ☐ IRWrite ☐ PCSource ☐ ALUOp ☐☐
ALUSrcA ☐ ALUSrcB ☐☐ RegWrite ☐ RegDst ☐

Write the result of the previous sum to the location specified by $s0.

PCWrite ☐ PCWriteCond ☐ IorD ☐ MemRead ☐ MemWrite ☐
MemToReg ☐ IRWrite ☐ PCSource ☐ ALUOp ☐☐
ALUSrcA ☐ ALUSrcB ☐☐ RegWrite ☐ RegDst ☐

Make the program counter jump 100 instructions ahead in memory.

PCWrite ☐ PCWriteCond ☐ IorD ☐ MemRead ☐ MemWrite ☐
MemToReg ☐ IRWrite ☐ PCSource ☐ ALUOp ☐☐
ALUSrcA ☐ ALUSrcB ☐☐ RegWrite ☐ RegDst ☐

Part E: Verilog (14 marks)

Consider the piece of Verilog code on the right.

1. In one sentence, describe what operation this code performs. **(4 marks)**

2. This code will cause an error when compiled. What is it, and how would you fix it? **(2 marks)**

```
module final (A, B, M, Clock);

input [7:0] A, B;
input Clock;
output reg [15:0] M;
reg [7:0] N;
integer i;

always @ (posedge Clock)
  begin
    for (i=0; i<8; i=i+1)
      begin
        if (A[i]==0 && A[i+1]==1)
          N = B;
        if (A[i]==1 && A[i+1]==0)
          N = -B;
      end
    M = M + N << i;
  end
endmodule
```

3. In the space below, complete the short Verilog module called `8bit_summer`, that implements an **8-bit accumulator circuit** with a **synchronous enable** and an **asynchronous negative clear**. **(8 marks)**

```
module 8bit_summer (A, Q, en, clock, clear);
```

Part F: Assembly Language (24 marks)

1. In the spaces provided below, write the assembly language instruction(s) that perform the following tasks. Full marks will only be given for one-instruction answers. **(12 marks total)**

a) Set the value in `$t0` to one eighth of its original value. **(3 marks)**

b) Set all of the bits of register `$t2` high. **(3 marks)**

c) Assuming that `$s0` stores the address of a character in memory, fetch that character and store it in `$t0`. **(3 marks)**

d) Set the program counter to zero. **(3 marks)**

2. In the space provided, describe the overall result of each assembly program. **(12 marks)**

```
            .data
len:        .word     5
list:       .word     -4, -1, 0, 1, 4
            .text
main:       addi $s0, $zero, list
            add $s1, $zero, len
            lw $t1, 0($s1)
top:        lw $t0, 0($s0)
            add $t0, $t0, $t0
            sw $t0, 0($s0)
            addi $t1, $t1, -1
            addi $s0, $s0, 4
            bne $t1, $zero, top
end:        jr $t1
```

```
            .data
len:        .word     5
list:       .word     -4, -1, 0, 1, 4
            .text
main:       addi $s0, $zero, list
            add $s1, $zero, len
            lw $t1, 0($s1)
top:        lw $t0, 0($s0)
            add $t1, $t1, $t0
            sub $t0, $t0, $t0
            sw $t0, 0($s0)
            addi $s0, $s0, 4
            bne $t1, $zero, top
end:        jr $ra
```

```
            .data
len:        .word     5
list:       .word     -4, -1, 0, 1, 4
            .text
main:       addi $s0, $zero, list
            add $s1, $zero, len
            lw $t1, 0($s1)
top:        lw $t0, 0($s0)
            sub $t0, $zero, $t0
            sw $t0, 0($s0)
            add $t1, $t1, $t0
            addi $s0, $s0, 4
            bne $t1, $zero, top
end:        jr $t1
```

```
            .data
len:        .word     5
list:       .word     -4, -1, 0, 1, 4
            .text
main:       addi $s0, $zero, list
            add $s1, $zero, len
            lw $t1, 0($s1)
top:        lw $t0, 0($s0)
            addi $t1, $t1, -1
            add $t0, $t0, $t0
            sw $t0, 0($s0)
            bne $t1, $zero, top
            addi $s0, $s0, 4
end:        jr $ra
```

Reference Information

ALU arithmetic input table:

Select		Input	Operation	
S_1	S_0	Y	C_{in}=0	C_{in}=1
0	0	All 0s	G=A	G=A+1
0	1	B	G=A+B	G=A+B+1
1	0	B	G=A-B-1	G=A-B
1	1	All 1s	G=A-1	G=A

Register assignments:

Register values : Processor role

- Register 0 ($zero): reserved value.
- Register 1 ($at): reserved for the assembler.
- Registers 2-3 ($v0, $v1): return values
- Registers 4-7 ($a0-$a3): function arguments
- Registers 8-15, 24-25 ($t0-$t9): temporaries
- Registers 16-23 ($s0-$s7): saved temporaries
- Registers 28-31 ($gp, $sp, $fp, $ra)

Bonus Question: (1 mark)

Name one of your CSC258 TAs.

Instruction table:

Instruction	Type	Op/Func	Syntax
add	R	100000	$d, $s, $t
addu	R	100001	$d, $s, $t
addi	I	001000	$t, $s, i
addiu	I	001001	$t, $s, i
div	R	011010	$s, $t
divu	R	011011	$s, $t
mult	R	011000	$s, $t
multu	R	011001	$s, $t
sub	R	100010	$d, $s, $t
subu	R	100011	$d, $s, $t
and	R	100100	$d, $s, $t
andi	I	001100	$t, $s, i
nor	R	100111	$d, $s, $t
or	R	100101	$d, $s, $t
ori	I	001101	$t, $s, i
xor	R	100110	$d, $s, $t
xori	I	001110	$t, $s, i
sll	R	000000	$d, $t, a
sllv	R	000100	$d, $t, $s
sra	R	000011	$d, $t, a
srav	R	000111	$d, $t, $s
srl	R	000010	$d, $t, a
srlv	R	000110	$d, $t, $s
beq	I	000100	$s, $t, label
bgtz	I	000111	$s, label
blez	I	000110	$s, label
bne	I	000101	$s, $t, label
j	J	000010	label
jal	J	000011	label
jalr	R	001001	$s
jr	R	001000	$s
lb	I	100000	$t, i ($s)
lbu	I	100100	$t, i ($s)
lh	I	100001	$t, i ($s)
lhu	I	100101	$t, i ($s)
lw	I	100011	$t, i ($s)
sb	I	101000	$t, i ($s)
sh	I	101001	$t, i ($s)
sw	I	101011	$t, i ($s)
trap	I	001100	i
mflo	R	010010	$d

This page is left blank intentionally for answer overflows.

Total Marks = 142

Total Pages = 14

End of exam