UNIVERSITY OF TORONTO

Faculty of Arts and Science

December 2014 Examinations

CSC258H1F: Computer Organization

Duration: 3 hours

Permitted Aids: one ruler, one highlighter

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, plus a bonus question. 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 use the overflow page, clearly indicate the section(s) that you want marked.

Mark Breakdown

Part A:	/ 19
Part B:	/ 25
Part C:	/ 22
Part D:	/ 39
Part E:	/ 14
Part F:	/ 24
Bonus:	/ 1
Total:	**/ 143**

Part A: Short Answer (19 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 32-bit architecture? **(2 marks)**

a) `256` **b)** `1024`

c) `2048` **d)** `8192`

2. How many address bits are needed to specify the location of an integer in a 1024 bit memory unit, given a 64-bit architecture? **(2 marks)**

a) `4` **b)** `8`

c) `10` **d)** `16`

3. Which of the following are valid clocks on the DE2 board? **(2 marks)**

a) `CLOCK_25` **b)** `CLOCK_27`

c) `CLOCK_50` **d)** `CLOCK_72`

4. True or False? Multiplication with Booth's Algorithm uses the same number of clock cycles as an accumulator circuit. **(1 mark)**

True **False**

5. Which of the following assembly instructions use the processor's sign extend unit? Circle all that apply. **(2 marks)**

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

c) `lb` **d)** `xori`

6. If the Verilog statement `{C, A} = A * B` performs multiplication on the binary numbers `A`, `B` and `C` without losing any bits, which of the following must be true? **(2 marks)**

a) `C` has more bits than `B` **b)** `C` has fewer bits than `B`

c) `C` is the same size as `B` **d)** More than one of the above

7. Which of the following is the FPGA that we use in the DE2 labs? **(1 mark)**

a) Cyclone I **b)** Cyclone II

c) Cyclone III LS **d)** Cyclone IV GX

8. Which is the device that is used for this FPGA in the labs? **(1 mark)**

a) EP2C35F672C6 **b)** EP2C36F672C6

c) EP2C35F672C5 **d)** EP2C36F672C5

9. True or False? Interrupts are polled on a MIPS architecture. **(1 mark)**

True **False**

10. Rank the following interrupts, from 1 (highest priority) to 4 (lowest priority). **(2 marks)**

Arithmetic overflow exception ______ **External interrupt** ______

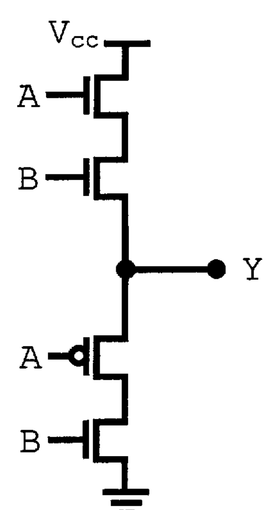
Address error exception ______ **Breakpoint exception** ______

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

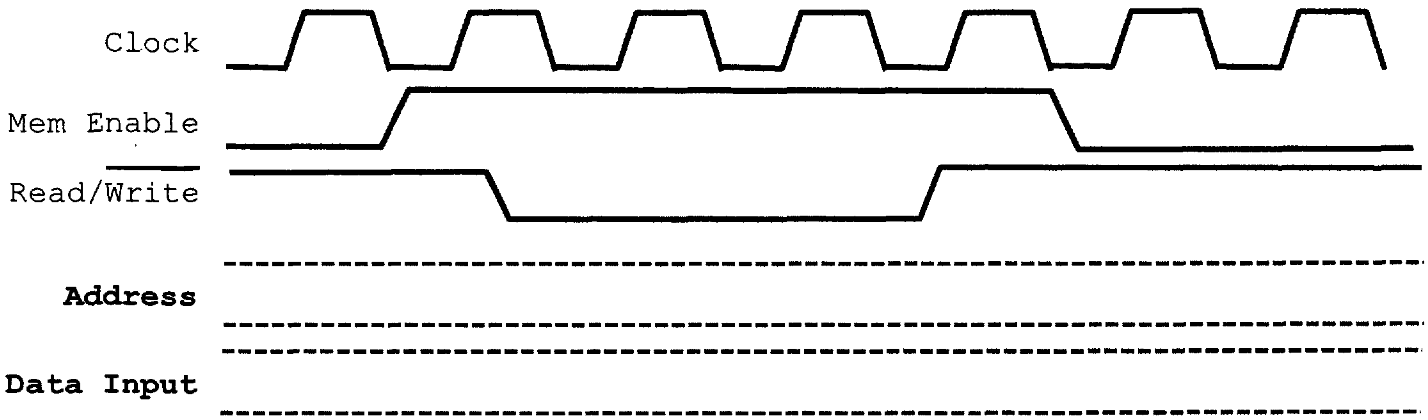
c) 2^{26} bytes **d)** 2^{28} bytes

12. What is the gate that is implemented by the transistor diagram below? **(2 marks)**



Part B: Design and Analysis (25 marks)

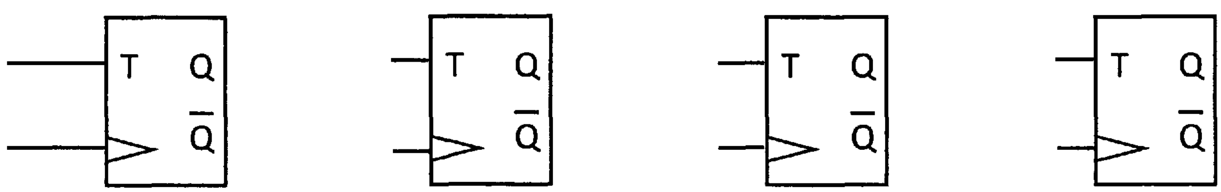
1. In the memory write diagram below, label the regions with valid address and data values. Also label the memory delays listed below, and describe the reasons behind them. **(8 marks)**



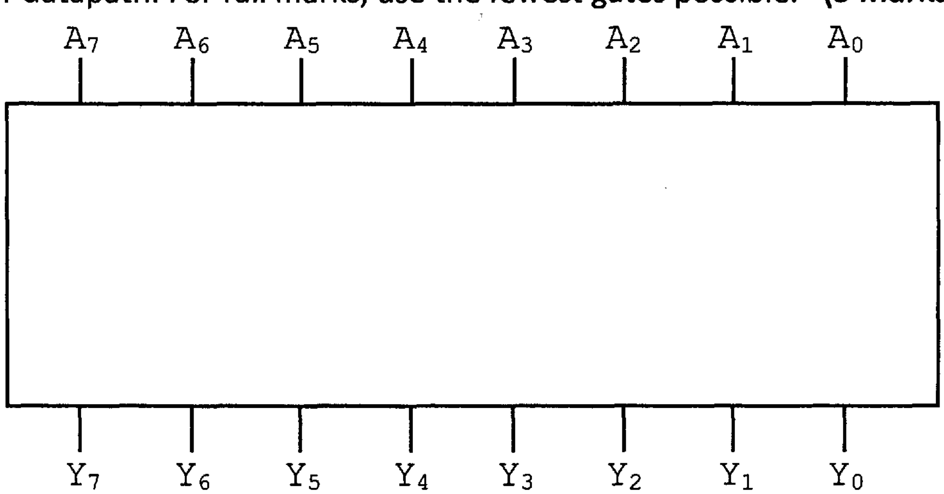
Label these on diagram & describe:

- Setup Address Time ______________________________
- Hold Data Time ______________________________

2. Using the flip-flops shown below, design a ripple counter diagram that counts down instead of up. Make sure to label the inputs and outputs properly. **(6 marks)**



3. In the diagram below, fill in the circuitry for an 8-bit shift-by-two circuit, like those found in the processor datapath. For full marks, use the fewest gates possible. **(3 marks)**



4. For each of the circuits below, circle the ones that will compile without errors. **(4 marks)**
For circuits that do compile, draw lines between the ones with the same behaviour. **(4 marks)**

```
module final(A, B, C);
input [1:0] B, C;
output [1:0] A;
assign A = ~B & ~C | ~C & B;
endmodule
```

```
module final(A, B, C);
input [1:0] B, C;
output [1:0] A;
assign A = B ^ C;
endmodule
```

```
module final(A, B, C);
input [1:0] B, [1:0] C;
output [1:0] A;
wire W;
xor(W, B, C);
not(A, W);
endmodule
```

```
module final(A, B, C);
input B[1:0], A[1:0];
output C[1:0];
wire R, S;
nand(R, B, A);
or(S, B, A);
xor(C, R, S);
endmodule
```

```
module final(A, B, C);
input [1:0] B, A;
output [1:0] C;
wire R, S;
nand(R, B, A);
or(S, B, A);
xor(C, R, S);
endmodule
```

```
module final(A, B, C);
input [1:0] B, C;
output [1:0] A;
reg W;
xor(W, B, C);
not(A, W);
endmodule
```

```
module final(A, B, C);
input [1:0] B, C;
output [1:0] A;
assign A = B & ~C | ~B & C;
endmodule
```

```
module final(A, B, C);
input [1:0] B, C;
output [1:0] A;
A <= B ^ C;
endmodule
```

Part C: Processors (22 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. The framework is provided below, with a few values filled in for you. Fill in the rest, according to the steps shown in class. **(10 marks)**

Initial Values: A= ____ B= ____ −B= ____

Step #1:

A = ____ **Initial P value =** 0000 0000

New P value = 0111 0000

Step #2:

A = ____ **Initial P value =** ____

New P value = ____

Step #3:

A = ____ **Initial P value =** ____

New P value = ____

Step #4:

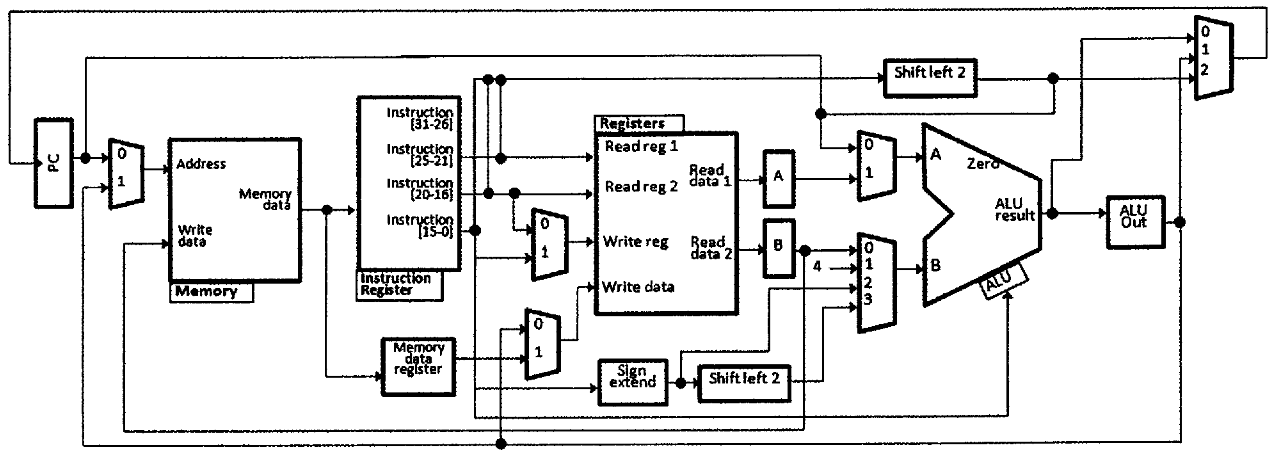
A = 11110 **Initial P value =** ____

New P value = ____

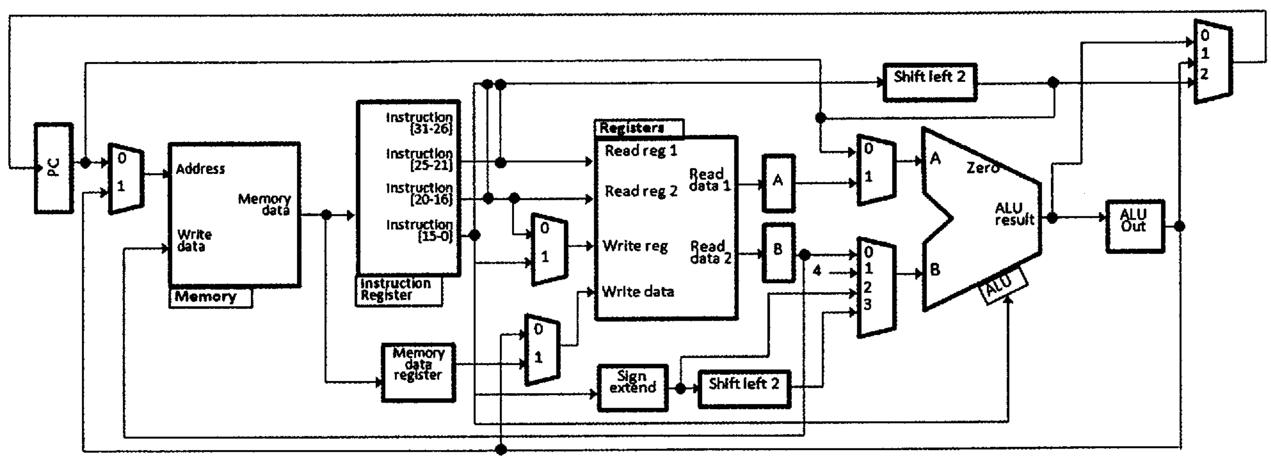
Final P value (binary) = ____ Final P value (decimal) = ____

2. Consider the datapaths below. For each of the following operations, highlight the path that the data needs to take, from start to finish. **(12 marks)**

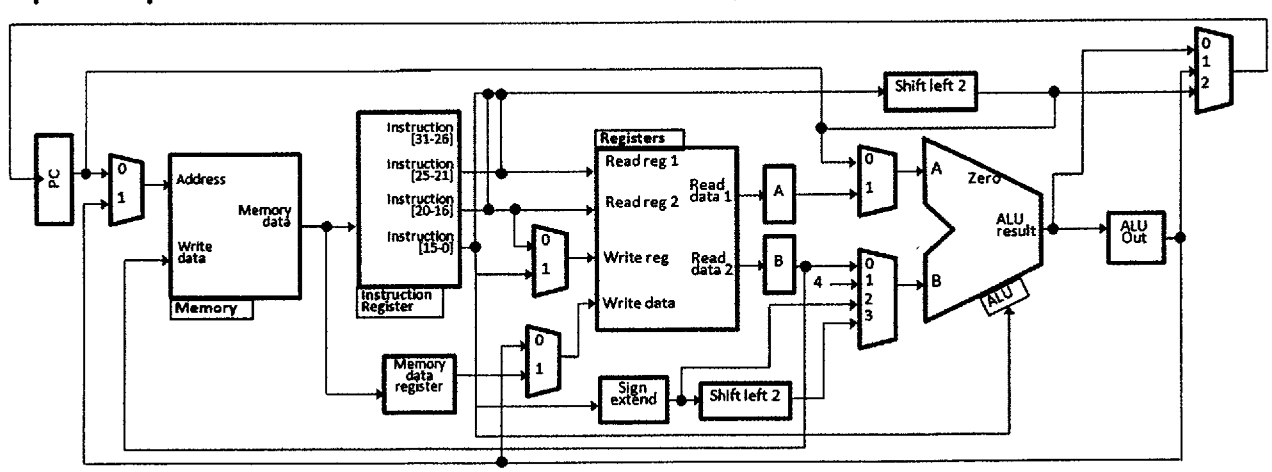
a) Set register $t0 to 100



b) Store the program counter in $ra



c) Pop the top element from the stack and store it in $a0



Part D: Processor Instructions (39 marks)

1. 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. **(12 marks)**

a) `sllv $t0, $t1, $t2`

b) `jr $ra`

c) `lhu $t0, 15($s0)`

2. For the following machine code instructions, provide the equivalent assembly language instruction in the space provided. **(12 marks)**

a) `10001111101000101111111111111111`

b) `00000000101001100010011101000110`

c) `00100100000111110001011111111111`

3. 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 the stack pointer.

PCWrite ☐ PCWriteCond ☐ IorD ☐ MemRead ☐ MemWrite ☐

MemToReg ☐ IRWrite ☐ PCSource ☐ ALUOp ☐

ALUSrcA ☐ ALUSrcB ☐ RegWrite ☐ RegDst ☐

Add 100 to the program counter, and store the result in $ra.

PCWrite ☐ PCWriteCond ☐ IorD ☐ MemRead ☐ MemWrite ☐

MemToReg ☐ IRWrite ☐ PCSource ☐ ALUOp ☐

ALUSrcA ☐ ALUSrcB ☐ RegWrite ☐ RegDst ☐

Jump to the newly-calculated $ra (from previous part) if $t0 is equal to zero.

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 a sentence or less, describe what operation this code performs. **(4 marks)**

2. Based on your answer above, what do the `i` and `o` reg values represent? **(2 marks)**

i: ____________________________

o: ____________________________

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

input [31:0] m;
input w;
output reg [5:0] o;
output reg [4:0] s, t, d;
output reg [15:0] i;
reg [31:0] c;

always @ (*)
  begin
    if (w)   c <= m;
    o <= c[31:26];
    s <= c[25:21];
    t <= c[20:16];
    d <= c[15:11];
    i <= c[15:0];
  end
endmodule
```

3. In the space below, complete the short Verilog module called `register_unit`, that implements the **register unit** from the processor datapath. **(8 marks)**

```
module register_unit (reg_write, read_reg1, read_reg2,
          write_reg, write_data, read_data1, read_data2);
```

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 stored in `$s0` to one eighth of its original value. **(3 marks)**

b) Invert all the bits of the value stored in `$a0`. **(3 marks)**

c) Pop a character off the stack 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 $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, $t0, $t0
        sw $t0, 0($s0)
        addi $t1, $t1, -1
        addi $s0, $s0, 4
        bne $t0, $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)
        addi $t1, $t1, -1
        addi $s0, $s0, 4
        add $t0, $t0, $t0
        sw $t0, 0($s0)
        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)
        lw $t0, 0($s0)
top:    addi $t1, $t1, -1
        add $t0, $t0, $t1
        bne $t1, $zero, top
        addi $s0, $s0, 4
        sw $t0, 0($s0)
end:    jr $zero
```

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:

What are the names of two of the current TAs for CSC258? **(1 mark)**

____

____

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 = 143

Total Pages = 14

End of exam