**UNIVERSITY OF TORONTO**

**Faculty of Arts and Science**

**December 2013 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 16 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: | / 22 |
| Part B: | / 18 |
| Part C: | / 8 |
| Part D: | / 53 |
| Part E: | / 15 |
| Part F: | / 24 |
| Bonus: | / 1 |
| **Total:** | **/ 140** |

## Part A: Short Answer (22 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 6 address bits and a 64-bit architecture? **(2 marks)**

**a)** 256 **b)** 384

**c)** 4096 **d)** 512

**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)** 512 **b)** 8

**c)** 32 **d)** 9

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

**4.** True or False? A `pn` junction won't have a depletion layer unless a forward or reverse bias is applied to the junction. **(1 mark)**

**True** **False**

**5.** True or False? When assigning flip-flop values to states, it's better to use more flip-flops in your design if it avoids having multiple flip-flops change at the same time. **(1 mark)**

**True** **False**

**6.** Which of the ALU bits $S_0$, $S_1$, $S_2$ is different from the others, and in what way? **(2 marks)**

**7.** What kind of interrupt handling does the MIPS architecture use? **(1 mark)**

**8.** True or False? An accumulator circuit performs multiplication faster than a multiplication circuit made up of layers of adders. **(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 finite state machine has 70 states, what is the minimum number of flip-flops that you would need to implement it in a circuit? **(1 mark)**

**11.** Which devices require the most gates to implement? Rank each of the following devices in order of complexity, from **1** (highest gate cost) to **4** (lowest gate cost) **(6 marks)**

**RS flip-flop:** __________ **D flip-flop:** __________

**Full adder:** __________ **Multiplexer:** __________

**1-bit comparator:** __________ **T flip-flop:** __________

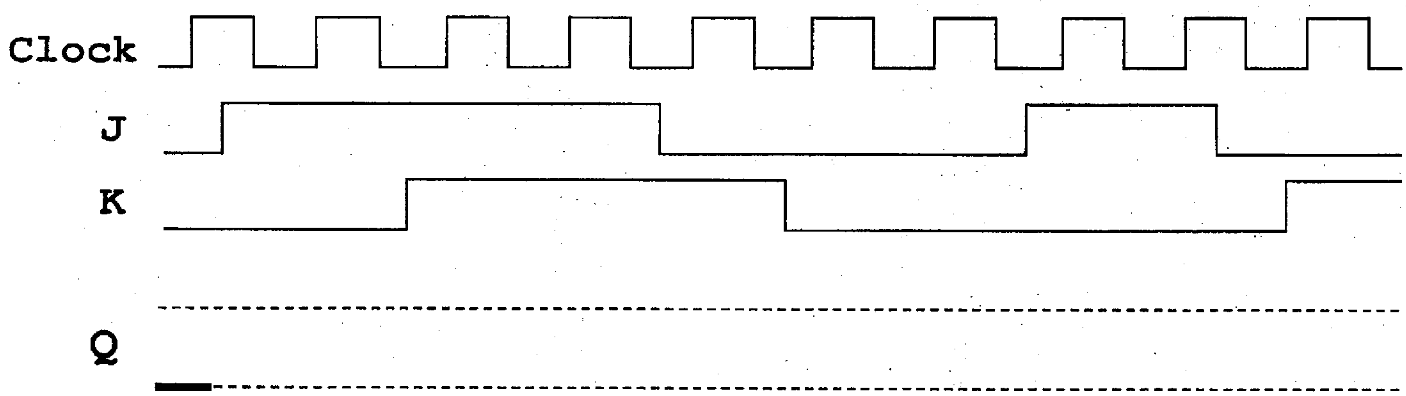
**12.** What values do the `HI` and `LO` registers store when performing integer division? **(2 marks)**

**HI :** ______________________________

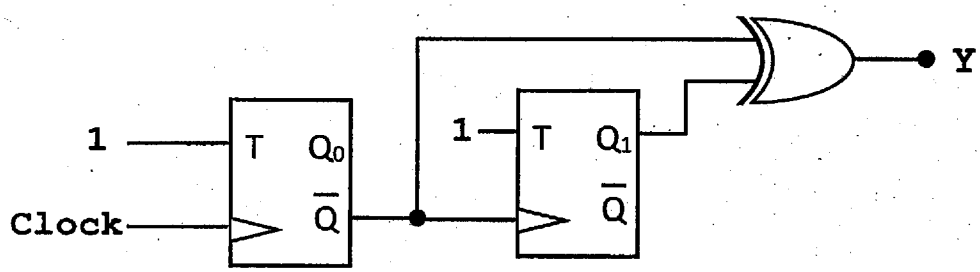
**LO:** ______________________________

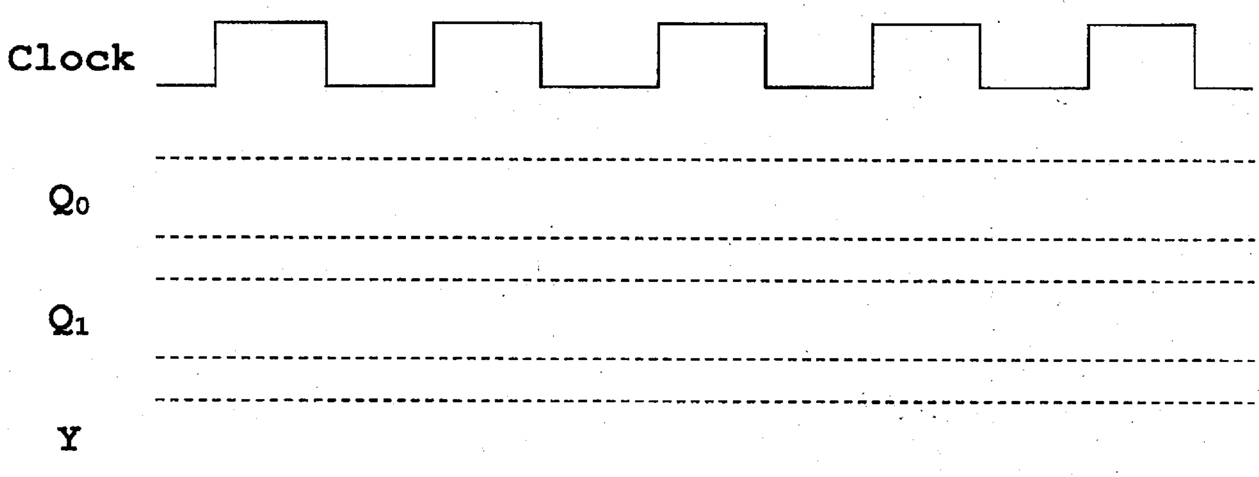
## Part B: Design and Analysis (18 marks)

**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)**

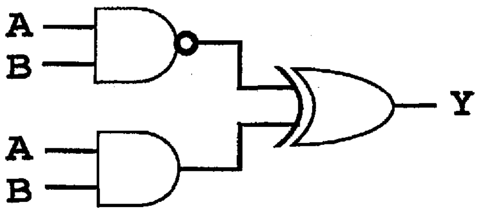


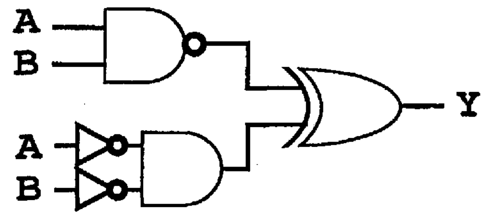

**4.** Given the following circuit, show what the output value of $Q_0$, $Q_1$ and Y will be in the waveform diagram. Assume that $Q_0$ and $Q_1$ start with initial values of zero. **(6 marks)**

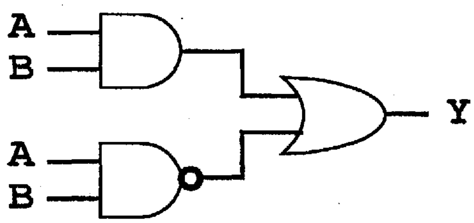


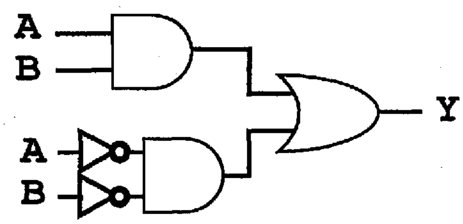



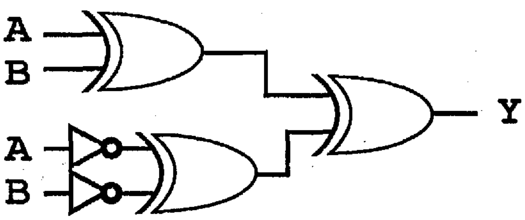
**3.** Match the circuits that perform the same logical operation by drawing lines that connect equivalent circuits in the space below. **(8 marks)**

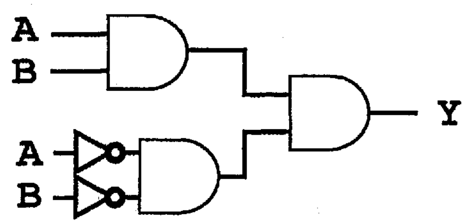


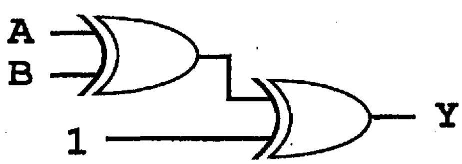



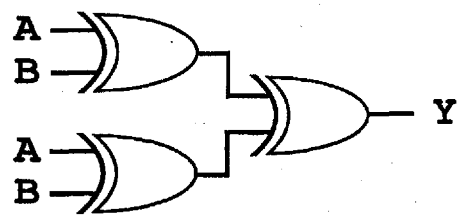



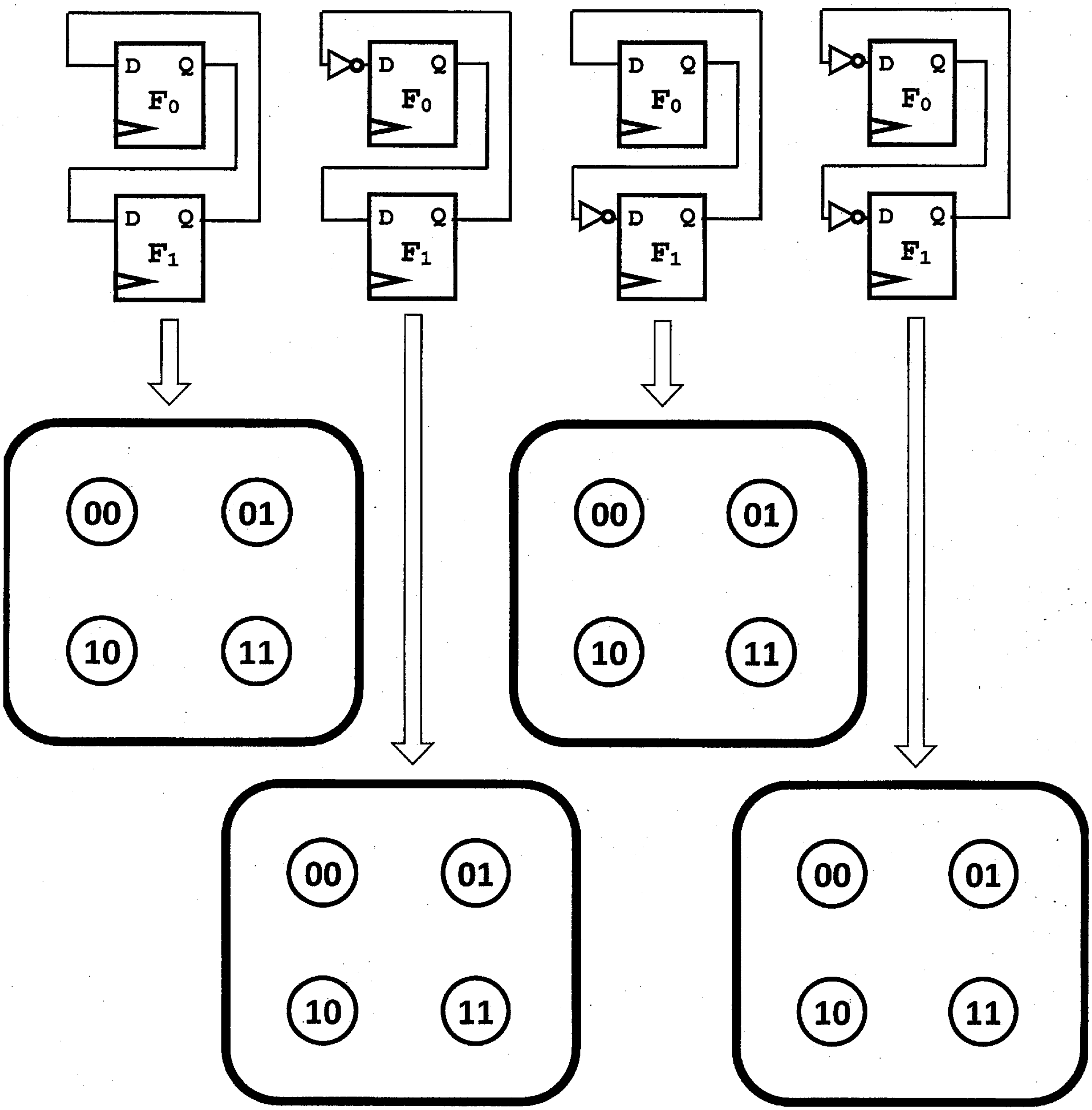




Student Number: ____________________

## Part C: Sequential Circuit Design (8 marks)

**1.** Consider the four flip-flop circuits below. For each circuit, fill in the corresponding state transitions in the state diagrams beneath each circuit. **(8 marks)**



## Part D: Processors (53 marks)

**1.** In the space below, perform Booth's Algorithm on the binary values A=1101 and B=1011. Show your steps in the space provided. **(6 marks)**

**P =**

**Step 1:**

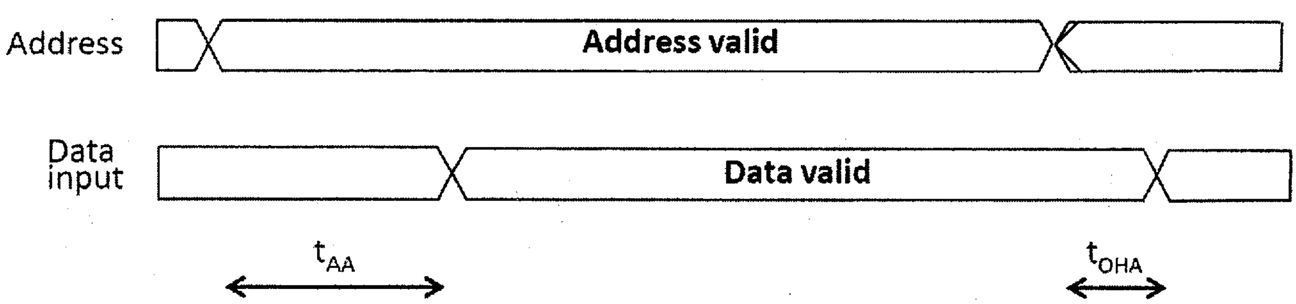
**Step 2:**

**Step 3:**

**Step 4:**

**P =**

**2.** For the following read signal diagram, describe what each labeled time segment is called, and the purpose of each segment during a memory read operation. **(4 marks)**



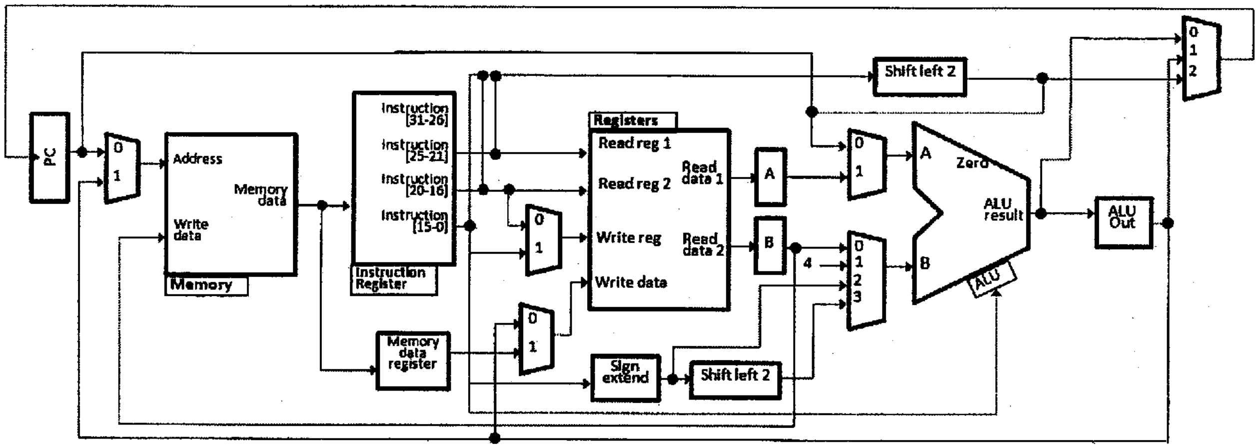

$t_{AA}$: ______________________________

$t_{OHA}$: ______________________________

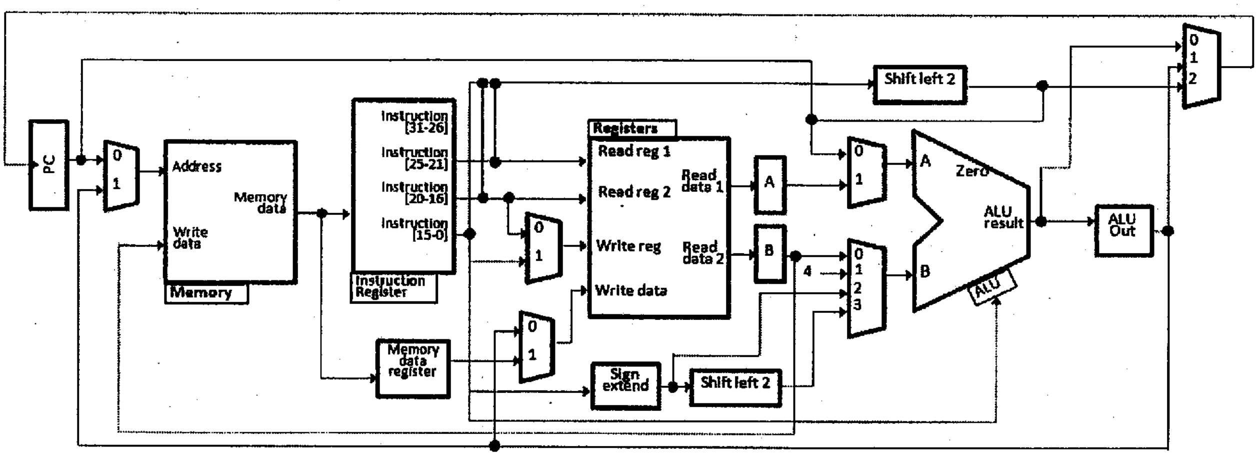
Studen

**3.** 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)**

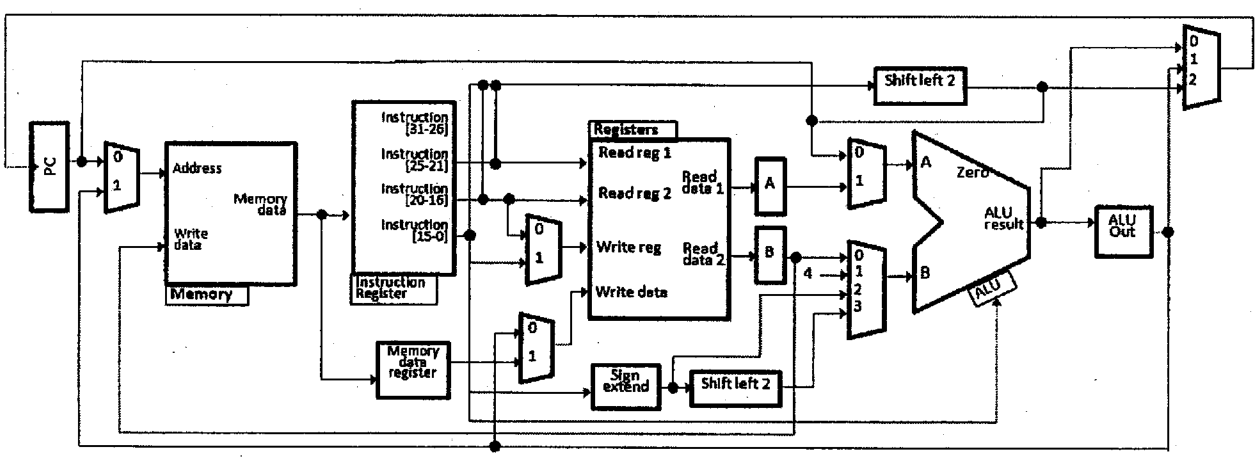
**a) Increment the program counter**




**b) Increment register $t0 by 42**




**c) Push the value in $s0 onto the stack**



**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. **(10 marks)**

**a)** `jr $ra`

**b)** `sra $t1, $t0, 4`

**c)** `lb $t0, -1($v0)`

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

**a)** `00110000000010011111111111100011`

**b)** `00000010000111110000000000001001`

**c)** `10010001000010011111111111100110`

**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.

## Fetch the next instruction from memory.

`PCWrite` ☐ `PCWriteCond` ☐ `IorD` ☐ `MemRead` ☐ `MemWrite` ☐

`MemToReg` ☐ `IRWrite` ☐ `PCSource` ☐☐ `ALUOp` ☐☐☐

`ALUSrcA` ☐ `ALUSrcB` ☐☐ `RegWrite` ☐ `RegDst` ☐

## Add 24 to the program counter and write the result to \$s0.

`PCWrite` ☐ `PCWriteCond` ☐ `IorD` ☐ `MemRead` ☐ `MemWrite` ☐

`MemToReg` ☐ `IRWrite` ☐ `PCSource` ☐☐ `ALUOp` ☐☐☐

`ALUSrcA` ☐ `ALUSrcB` ☐☐ `RegWrite` ☐ `RegDst` ☐

## Branch to the new PC value from the previous part if \$t0 is equal to \$t1.

`PCWrite` ☐ `PCWriteCond` ☐ `IorD` ☐ `MemRead` ☐ `MemWrite` ☐

`MemToReg` ☐ `IRWrite` ☐ `PCSource` ☐☐ `ALUOp` ☐☐☐

`ALUSrcA` ☐ `ALUSrcB` ☐☐ `RegWrite` ☐ `RegDst` ☐

## Part E: Verilog (15 marks)

Consider the piece of Verilog code on the right.

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

**2.** Given your answer to part 1, when would the default case be activated? **(2 marks)**

```
module X(o, i);
  output reg [9:0] o;
  input [3:0] i;

  always @(i)
    case (i)
      4'h0: o = 10'd1;
      4'h1: o = 10'd2;
      4'h2: o = 10'd4;
      4'h3: o = 10'd8;
      4'h4: o = 10'd16;
      4'h5: o = 10'd32;
      4'h6: o = 10'd64;
      4'h7: o = 10'd128;
      4'h8: o = 10'd256;
      4'h9: o = 10'd512;
      default: o = 10'd0;
    endcase
 endmodule
```

**3.** In the space below, write a short Verilog module called `2bit_mult`, that implements a multiplier circuit for 2-bit input values. For full marks, do not use an `always` block. **(9 marks)**

## 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)** Divide the value in `$t0` by 16, and store the result back into `$t0`. **(3 marks)**

**b)** Set register `$t2` to be the negative of its current value. **(3 marks)**

**c)** Assuming that `$s0` stores the address of a location in memory, fetch a two-byte number from that location and store it in `$t0`. **(3 marks)**

**d)** Execute the subroutine whose location is specified by `$a0`. **(3 marks)**

**2.** State the overall operation each assembly program performs in the space provided. **(12 marks)**

```
        .data
len:    .word   5
list:   .word   8, 1, -5, 0, -4
        .text
main:   addi $s0, $zero, list
        addi $t2, $s0, 4
        lw $t0, 0($s0)
top:    lw $t1, 4($s0)
        sw $t1, 0($s0)
        addi $s0, $s0, 4
        bne $t2, $s0, top
end:    sw $t0, 0($s0)
        jr $ra
```

```
        .data
len:    .word   5
list:   .word   8, 1, -5, 0, -4
        .text
main:   addi $s0, $zero, list
top:    lw $t0, 0($s0)
        bgtz $t0, bottom
mid:    addi $s0, $s0, 4
        j top
bottom: sub $t0, $zero, $t0
        sw $t0, 0($s0)
        bne $t0, $zero, mid
end:    jr $ra
```

```
        .data
len:    .word   5
list:   .word   8, 1, -5, 0, -4
        .text
main:   add     $t0, $zero, list
        add     $t1, $zero, len
        lw      $t1, 0($t1)
top:    addi    $t1, $t1, -1
        sw      $t1, 0($t0)
        addi    $t0, $t0, 4
        bne     $t1, $zero, top
end:    jr      $ra
```

```
        .data
len:    .word   5
list:   .word   8, 1, -5, 0, -4
        .text
main:   addi $s0, $zero, list
top:    lw $t0, 0($s0)
        bne $t0, $zero, end
mid:    addi $s0, $s0, 4
        j top
bottom: sw $t0, 4($s0)
        bne $t0, $zero, mid
end:    jr $t0
```

Student Number: ____________



(continued)

# 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)

**<u>Bonus Question:</u>** **(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.***

*This page is left blank intentionally for answer overflows.*

Total Marks = 140

Total Pages = 16

End of exam