UNIVERSITY OF TORONTO

Faculty of Arts and Science

April 2017 Examinations

CSC258H1S: Computer Organization

Duration: 3 hours

Permitted Aids: one ruler, one highlighter

Last Name: _____

First Name: _____

Student Number: _____

Instructors: **Steve Engels (L0101, L0201)**

Sajad Shirali-Shahreza (L5101)

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 and those mentioned above. Keep all bags and notes far from your desk before the exam begins.
- There are 6 questions on 18 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.
- **Important:** CSC258 has an minimal exam condition for passing the course. You must get at least 40% on this exam to pass the rest of the course.

Mark Breakdown	
Part A:	/ 43
Part B:	/ 8
Part C:	/ 31
Part D:	/ 39
Part E:	/ 18
Part F:	/ 36
Bonus:	/ 1
Total:	**/ 175**

Part A: Short Answer (43 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. Which ones of the following hexadecimal addresses are valid PC values when translated into 32-bit binary addresses? Circle all that apply. **(2 marks)**

a) `E45B23C1`
b) `4C32A330`
c) `9B31FF1C`
d) `1234ABCD`

2. Which is of the following registers always stores a value that is divisible by 4? Circle all that apply. **(2 marks)**

a) `$zero`
b) `$t0`
c) `$sp`
d) `$ra`

3. Which operations are not performed by the ALU? Circle all that apply **(2 marks)**

a) multiplication
b) decrementing
c) increment by 4
d) sign extending

4. The processor's control unit is basically an advanced _____. **(1 mark)**

a) adder circuit
b) finite state machine
c) register circuit
d) multiplexer

5. Which of the following 4-bit signed binary arithmetic operations will cause an overflow? Circle all that apply. **(2 marks)**

a) `1001 + 0100`
b) `1010 + 1110`
c) `1111 + 0001`
d) `0100 + 0100`

6. A one-hot decoder has a binary output of `0000100000000000`. What is the input? **(1 mark)**

第3+2位

1100 (2)

7. What is the O ( ) complexity of an **n**-bit adder-array multiplication circuit? **(2 marks)**

Time = 1 **Space =** n^2

8. What is the maximum distance that a jump instruction can jump (in bytes)? **(1 mark)**

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

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

2^{26} 4 instruction

9. What is the output of a tri-state buffer when both inputs are zero? **(1 mark)**

high - Z

10. How many bytes can a register unit store, with 8 address bits and 32-bit words? **(2 marks)**

2^8 = 4 byte

$2^8 \times 4 = 1024$ Byte = 1 kB

=4B

11. Given a 32-bit architecture, how many address bits do you need for a memory unit that can store 512 integers? **(2 marks)**

memory = 512 * 4B = 2kB

2kB/4B = 512

$\log_2(512) = 9$ address bit

12. Which of these devices are purely combinational circuits? Circle all that apply. **(6 marks)**

$\log_2(512) = 9$

a) register unit **b)** sign extender **c)** program counter

d) adder circuit **e)** "shift left 2" unit **f)** control unit

sym circuits (存储器件)

13. Where are the stack and stack pointer stored in a processor?. **(2 marks)**

Stack = memory **Stack Pointer =** register

14. The ALU has four special output signals. In the spaces below, write the letter labels for each, and the brief description of what conditions they signal. **(4 marks)**

V = _____ C = _____

N = _____ Z = zero

15. Which of the following Verilog statements cannot appear inside an `always` block? Circle all that apply. **(2 marks)**

a) `a = b;` **b)** `assign a = b;`

c) `a <= b;` **d)** `not (a, b);`

16. What is the minimum number of 2-to-1 multiplexers that you would need to implement a 12-to-1 multiplexer? **(1 mark)**

11

12-1 [8-1 / 4-1] + 2-1 = 11 (7, 3, 1)

17. Which of the following instructions modify the `LO` register? Circle all that apply. **(3 marks)**

a) `addi $t1, $t3, 25` **b)** `divu $t2, $t3` **c)** `subu $t4, $t2, $t8`

d) `mflo $t5` **e)** `mthi $t6` **f)** `mult $a0, $a1`

18. The MOSFET is made up of three types of material, listed below. Sort these materials according to the conductivity, from most conductive (1) to least conductive (3). **(2 marks)**

Metal 1 (most) **Oxide** 3 (least) **Semiconductor** 2

19. Which of the following can be the destination register for an `add` operation? **(3 marks)**

a) `$pc` **b)** `$zero` **c)** `$ra`

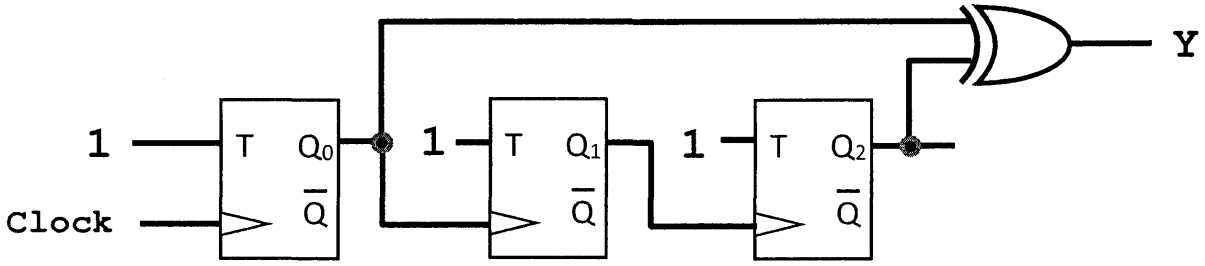
d) `$at` **e)** `$sp` **f)** `$hi`

20. In Part II of Lab 7, pixel values are loaded one at a time into the VGA adaptor, and then sent to the screen together. What is the name of the signal that sends the pixels to the screen? **(2 marks)**

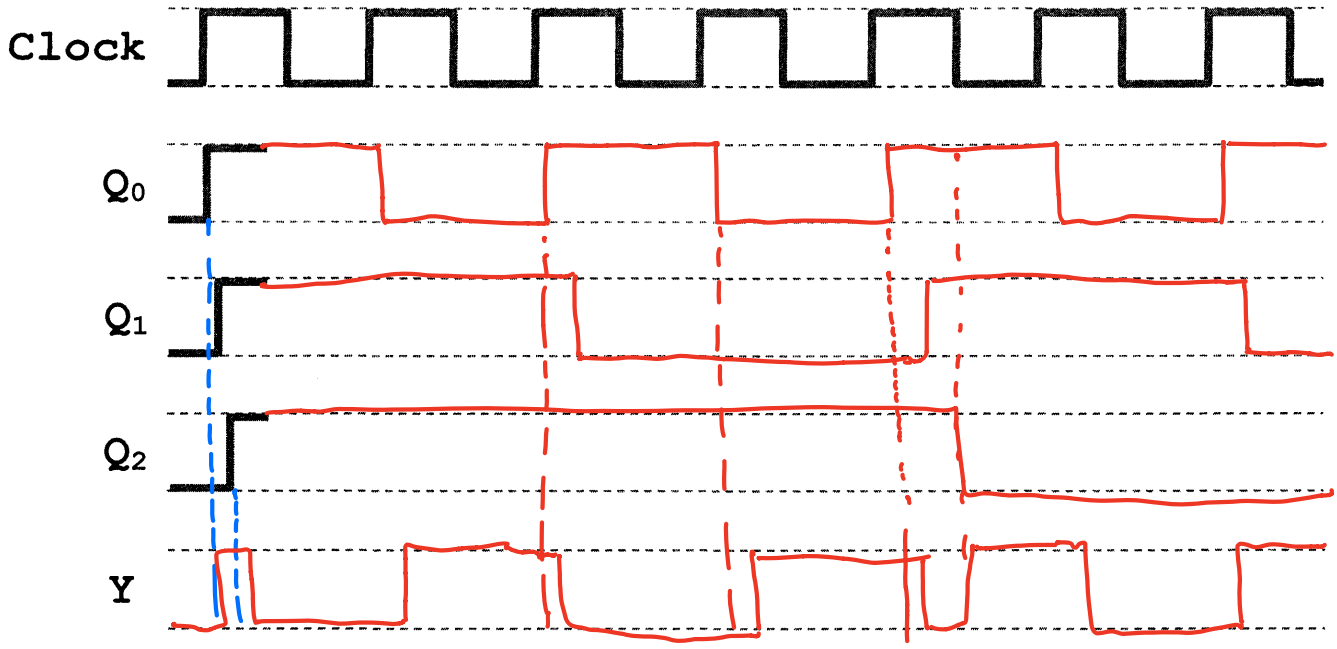
not sure, reg_en

Part B: Design and Analysis (8 marks)

1. Consider the flip-flop diagram below.



On the waveform diagram below, draw the waveform for the output signals Q_2, Q_1, Q_0 and Y, given the clock signal input shown. **(8 marks)**



f f c 6

x+3 x+2 x+1 x

Part C: Processor Operations (31 marks)

① stack grows backward

② little endian (least sig-bit's address is smallest)

1. The stack diagram on the left illustrates the top four bytes of a stack, and the empty spaces above them. What would this stack look like after the decimal integer 258 has been pushed onto it? Draw the result on the diagram on the right (using binary values for the contents).

258 = 256 + 2

For this example, assume little endian byte storage, and draw an arrow to illustrate what `$sp` points to after this operation is complete. **(6 marks)**

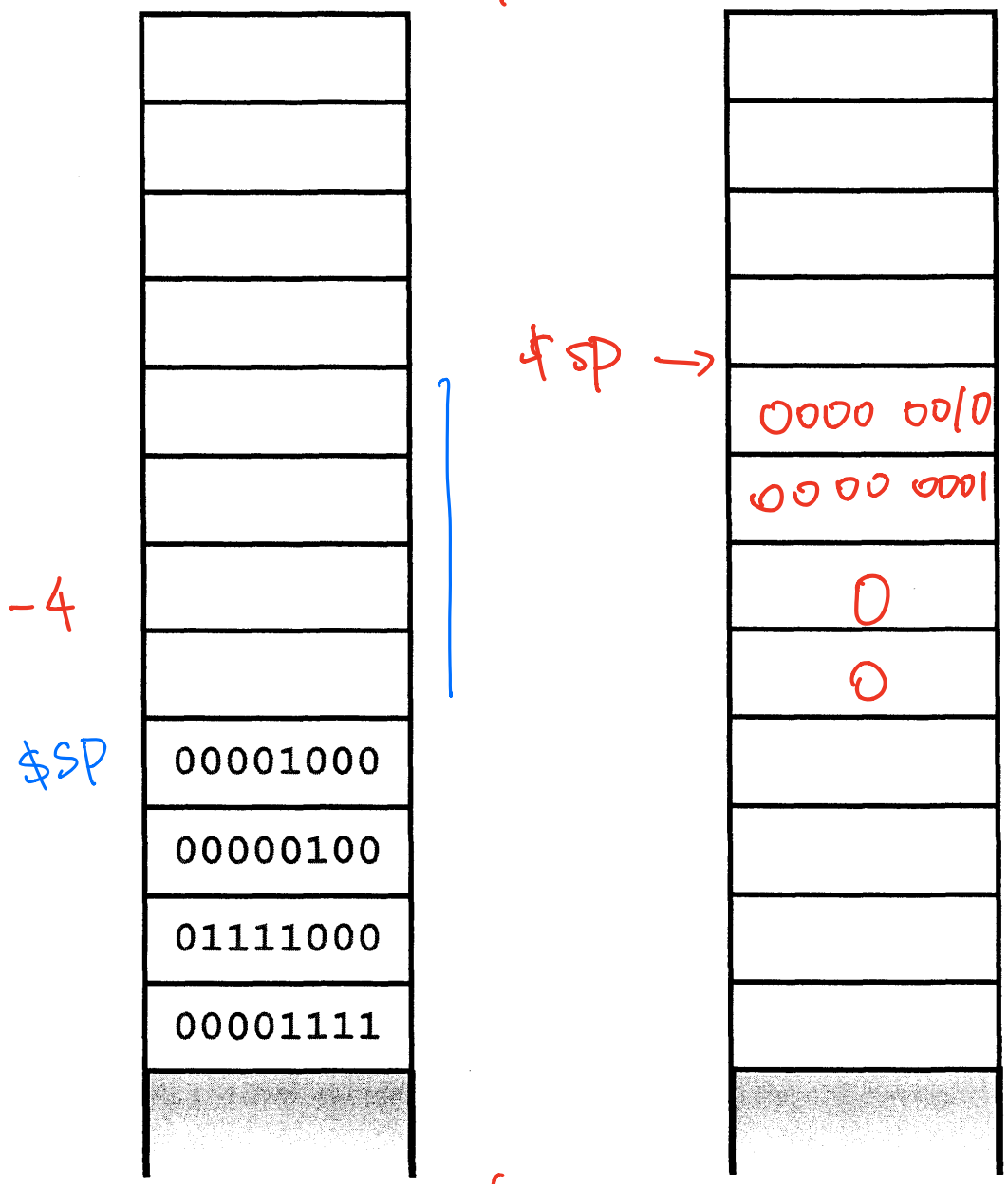
$258 = 2^8 + 2$

$= (0000,0001,0000,0010)_2$

$= (0102)_{16}$

lsb

low



2. When Booth's Algorithm is performed on the 4-bit binary inputs A=6 and B=-7, 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. **(5 marks)**

Initial Values: A= 01100 B= _____ -B= _____

Step #1:

A = 01100 **Initial P value =** 0000 0000

P value before shift = 0000 0000

Step #2:

A = _____ **Initial P value =** _____

P value before shift = _____

Step #3:

A = _____ **Initial P value =** _____

P value before shift = _____

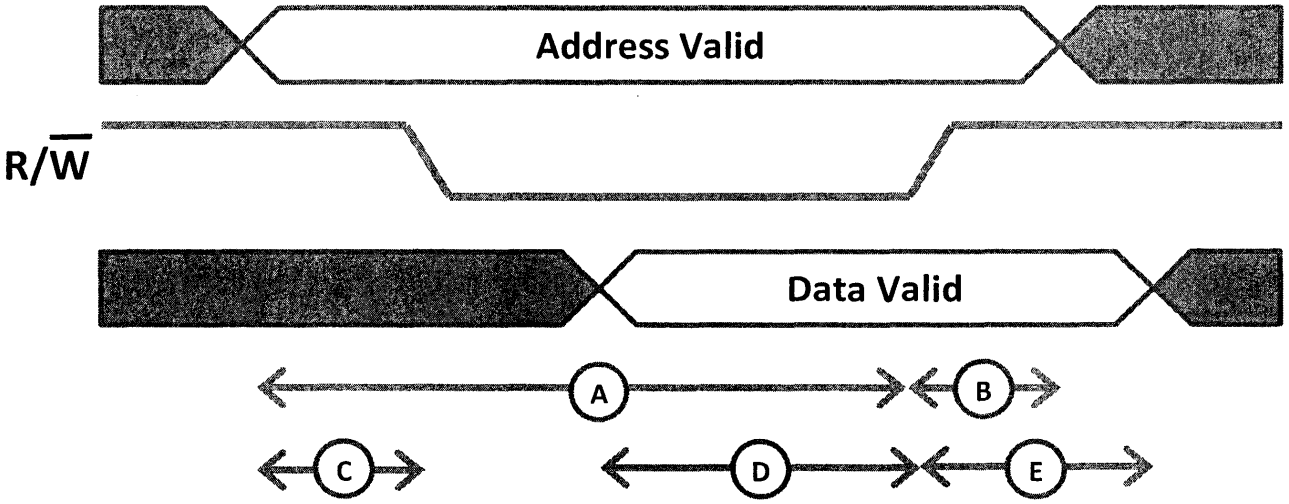
Step #4:

A = _____ **Initial P value =** _____

P value before shift = _____

Final P value (binary) = _____ **Final P value (decimal) =** _____

3. The diagram below represents the signals for a memory write operation.



In the spaces below, give the name of each delay shown above, and the reason why it's needed in this write operation. The first one is provided as an example. **(8 marks)**

A. Name: t_{AW} **(Address to Write end time)**

Reason: **Time needed to hold address high until data has transmitted.**

B. Name: _____

Reason: _____

C. Name: address set up time.

Reason: _____

D. Name: _____

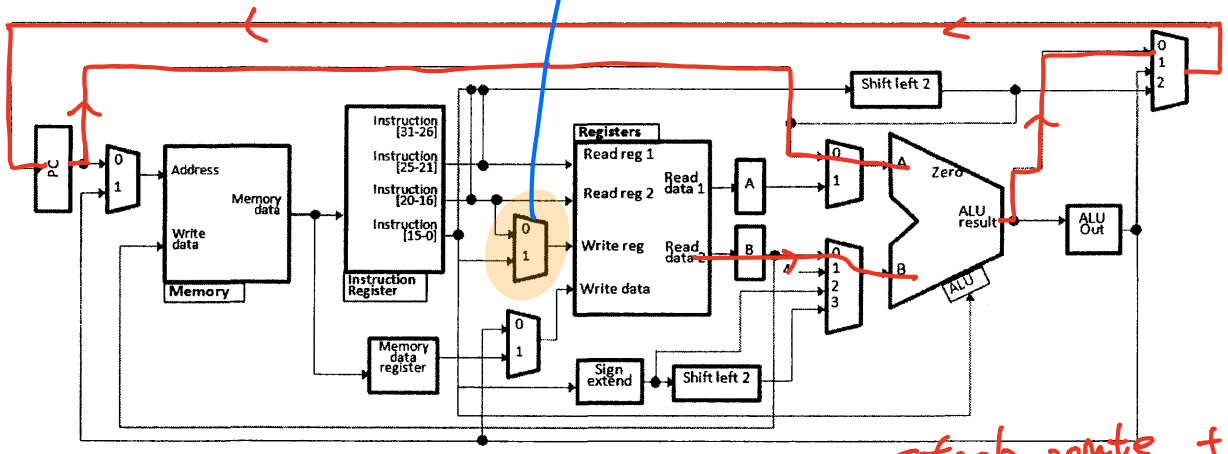
Reason: _____

E. Name: _____

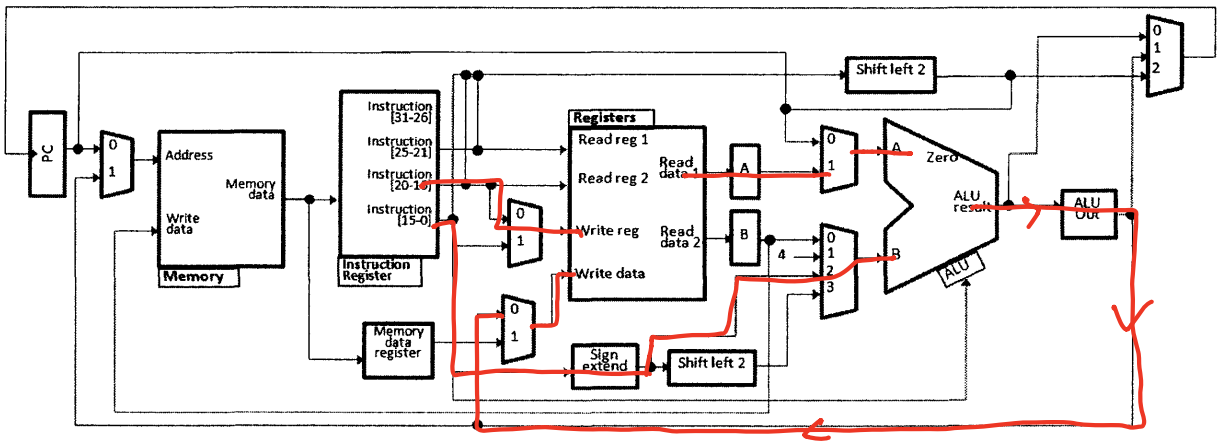
Reason: _____

4. 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 total, 4 marks each)**

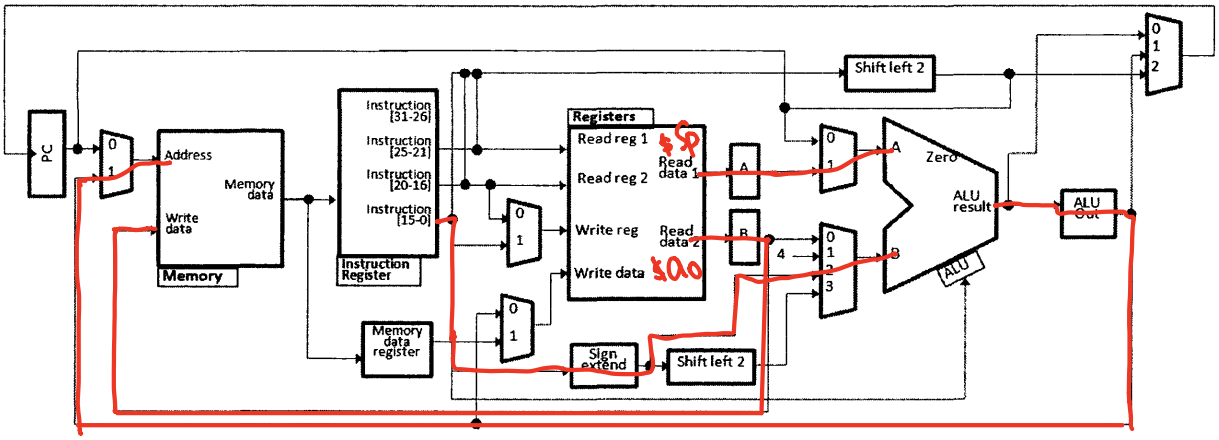
a) Increment the program counter by the value in `$s0`.



b) Move the stack pointer up by 4.



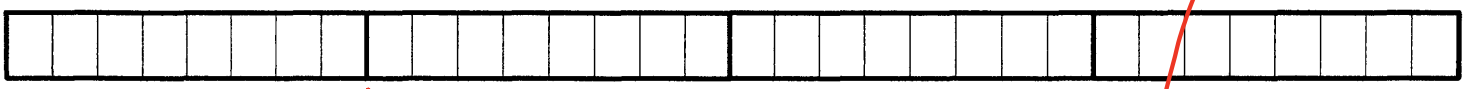
c) Store the value in `$a0` at the new stack pointer location.



Part D: Processor Instructions (39 marks)

1. For each assembly language instruction below, fill in the blanks with the corresponding 32-bit machine code instruction. For bits that could be either a **0** or a **1**, fill in the blank with an **X** value instead. **(12 marks total, 4 marks each)**

a) `addu $t8, $t0, $t1`



b) `blez $s0, top` *(where the "top" label occurs 128 bytes earlier)*

c) `sw $ra, 4($sp)`

2. Given the machine code instructions below, write the corresponding assembly language instruction in the space below each machine code instruction. **(12 marks total, 4 marks each)**

a) `00000011111010001111111111111111`

b) `10000001000001000000000011111111`

c) `00000000000000000000000000000000`

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 total, 5 marks each)**

- 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.

$sp = $sp + 4 addi 指令

Move the stack pointer address forward 4 bytes in memory. 001

PCWrite	0	PCWriteCond	0	IorD	X	MemRead	0	MemWrite	0
MemToReg	0	IRWrite	0	PCSource	XX	ALUOp	001		
ALUSrcA	1	ALUSrcB	10	RegWrite	1	RegDst	0		

$ra := pc + 44

Add 44 to the program counter, and store the result in `$ra`.

PCWrite	0	PCWriteCond	0	IorD	X	MemRead	0	MemWrite	0
MemToReg	X	IRWrite	0	PCSource	XX	ALUOp	001		
ALUSrcA	0	ALUSrcB	10	RegWrite	1	RegDst	0		

lw $t4, 0($sp)

Move the top 4 bytes of the stack into register `$t4`.

PCWrite	0	PCWriteCond	0	IorD	1	MemRead	1	MemWrite	0
MemToReg	1	IRWrite	0	PCSource	XX	ALUOp	001		
ALUSrcA	1	ALUSrcB	10	RegWrite	1	RegDst	0		

Part E: Verilog (18 marks)

Consider the piece of Verilog code on the right.

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

8-bit rotator (from right to left based on input S)

2. Based on your answer above, what do the S input bits represent? **(2 marks)**

shift bits. (left)

how many bits going to be rotated

```
module final (y, d, s);

input [7:0] d; data
input [2:0] s;
output [7:0] y;

always @(*)
  case(s)
    3'b000: y = d;
    3'b001: y = {d[6:0],d[7]};
    3'b010: y = {d[5:0],d[7:6]};
    3'b011: y = {d[4:0],d[7:5]};
    3'b100: y = {d[3:0],d[7:4]};
    3'b101: y = {d[2:0],d[7:3]};
    3'b110: y = {d[1:0],d[7:2]};
    3'b111: y = {d[0],d[7:1]};
  endcase
endmodule
```

4. In the spaces, implement a Verilog module for a Fibonacci counter. A Fibonacci counter outputs a sequence of Fibonacci numbers, starting at 1. For example, when this counter first starts, the output sequence is: 1, 1, 2, 3, 5, 8, 13...., where each number in the sequence is the sum of the previous two.

Your design should fulfill the following specifications:

- The module has a 4-bit output called **`result`** that displays the current Fibonacci number.
- The output changes at every rising clock edge, but only if **`enable`** is high.
- The module also has a single asynchronous **`reset`** bit that sets the output back to the beginning of the sequence whenever reset is low.

Hint: Your design should include an `always` statement. **(12 marks total)**

1 1 2 3 5 8 13 ....

```
module fibonacci_counter (reset, clock, enable, result);
    input   reset, enable, clock;
    output  reg [3:0] result;
    reg [3:0] prev_result

always @ (posedge clock, negedge reset)
begin
    if reset == 0
        result <= 1;
    else
        prev_result = result
        result = result + prev_result
end

endmodule
```

Part F: Assembly Language (36 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) Multiply the value stored in `$t4` by 4 and stored it back in `$t4`. **(3 marks)**

sll $t4, $t4, 2

b) Store the remainder of dividing `$t3` by 32 in `$t9`. **(3 marks)**

~~sra $t9, $t3, 5~~

andi $t9, $t3, 31

c) Store the integer 258 into register `$t1`. **(3 marks)**

addi $t1, $zero, 258

d) Store 1's complement of `$t6` (inverting all the bits) in `$t8`. **(3 marks)**

xori $t8, $t6, -1

2. We need to extend the MIPS assembler to support the following new pseudo instructions. For each new pseudo-instructions, write the real MIPS instructions that will perform that operation. Only implementations that uses at most 2 operations will get full marks. **(12 marks total)**

a) `divi $t, i` Divide `$t` by the constant `i` and store result in `lo` and `hi`. **(4 marks)**

```
addi $s $zero, i
div  $s, $t
```

b) `push $t` Push the value in `$t` onto the stack **(4 marks)**

```
addi $sp, $sp, -4
sw   $t , 0($sp)
```

c) `bev $t, label` Branch to label if the value in `$t` is even. **(4 marks)**

```
andi $d, $t, 1
beq  $d, $zero, label
```

Given array contains numbers non-orderly

[2 2 3 4 4 5 5 6 7 7 6 ...]

la : load adress

3. In the space provided, write the operation performed by each assembly program. **(12 marks)**

```
            .data
len:        .word     5
list:       .word     -4, 6, 7, -2, 1
            .text
main:       la $s1, len
            lw $t1, 0($s1)
            addi $t1, $t1, -1
            la $s0, list
            lw $t2, 0($s0)
alpha:      addi $t1, $t1, -1
            addi $s0, $s0, 4
            lw $t0, 0($s0)
            sub $t3, $t2, $t0
            blez $t3, beta
            addi $t2, $t0, $zero
beta:       bgtz $t1, alpha
            add $v0, $zero, $t2
            jr $ra
```

$s1 → len 指向
$t1 = 5
$t1 = 4
$s0 → list
$t2 = -4
$t1 = 3
$s0 → list[1]
$t0 = 6
$t3 = list[0] - list[1]

find the minimum element in array.

```
            .data
len:        .word     5
list:       .word     -4, 6, 7, -2, 1
            .text
main:       la $s0, list
            la $s1, len
            lw $t1, 0($s1)
            addi $t4, $zero, 0
alpha:      lw $t0, 0($s0)
            add $t4, $t4, $t0
            addi $t1, $t1, -1
            addi $s0, $s0, 4
beta:       blez $t1, alpha
            sub $v0, $t4, $zero
            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)

Draw something in the space below.

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 = 175

Total Pages = 18

End of exam