Solutions

UNIVERSITY OF TORONTO

Faculty of Arts and Science

December 2012 Examinations

CSC258H1S: Computer Organization

Duration: 3 hours

No Aids Allowed

Solutions

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 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.

Mark Breakdown

Part A:	**/ 21**
Part B:	**/ 18**
Part C:	**/ 52**
Part D:	**/ 14**
Part E:	**/ 20**
Part F:	**/ 30**
Total:	**/ 155**

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 instructions could fit into a 256 megabyte memory unit, given a 32-bit architecture? **(2 marks)**

32-bit = 4Btye

256 M ÷ 4B = 64M

a) 256 **b)** 64 M

c) 32 **d)** 8

2. How many address bits are needed to specify each byte in a 512 byte memory unit? **(1 mark)**

512 B ÷ 1B = 512

$\log_2(512) = 9$

a) 512 **b)** 8

c) 32 **d)** 9

3. How many minterms could you have in a circuit with three inputs and two flip-flops? **(1 mark)**

32 $2^5 = 32$

4. True or False? MOSFETs act as a switch by creating a conductive channel between the two n-type sections in a p-type substrate, or vice versa. **(1 mark)**

True **False**

5. What are the HI and LO registers used for in the context of integer division? **(2 marks)**

HI : Remainder

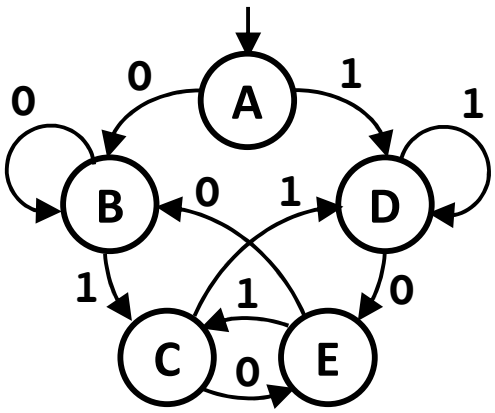
LO : Quotient

6. Assume that wires have already been declared for a, b and c. In the space below, show **TWO** different statements in Verilog that set c high when a and b are low. **(2 marks)**

```
assign c = ~a & ~b;
assign c = ~(a | b);
```

moore machine

7. Consider the finite state machine shown below. The output of this circuit goes high whenever it is in State C or State E. In the spaces below, indicate the operation that this finite state machine performs, and how many flip-flops this will need. **(5 marks)**



Operation: This circuit can recognize when the input alternates between 0 and 1.

Number of flip-flops: 3

8. In p-type semiconductors, what are "holes"? **(1 mark)**

a) absent electrons **b)** positive charge carriers

c) protons **d)** where you plant silicon flowers

9. How many bits do you shift a binary number in order to divide it by 4? **(1 mark)**

2

10. Why are the addresses of all MIPS instructions divisible by 4? **(1 mark)**

Because all instructions are four bytes long, starting at address 0

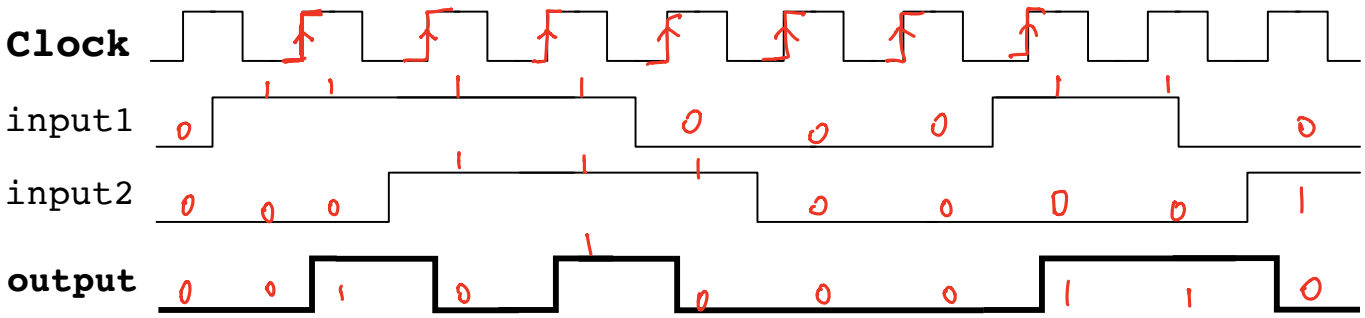
11. What do the following output signals from the ALU signify? **(4 marks)**

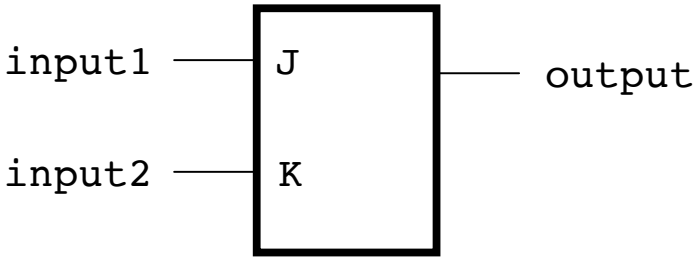
C: carry **V:** overflow

N: negative **Z:** zero

Part B: Design and Analysis (18 marks)

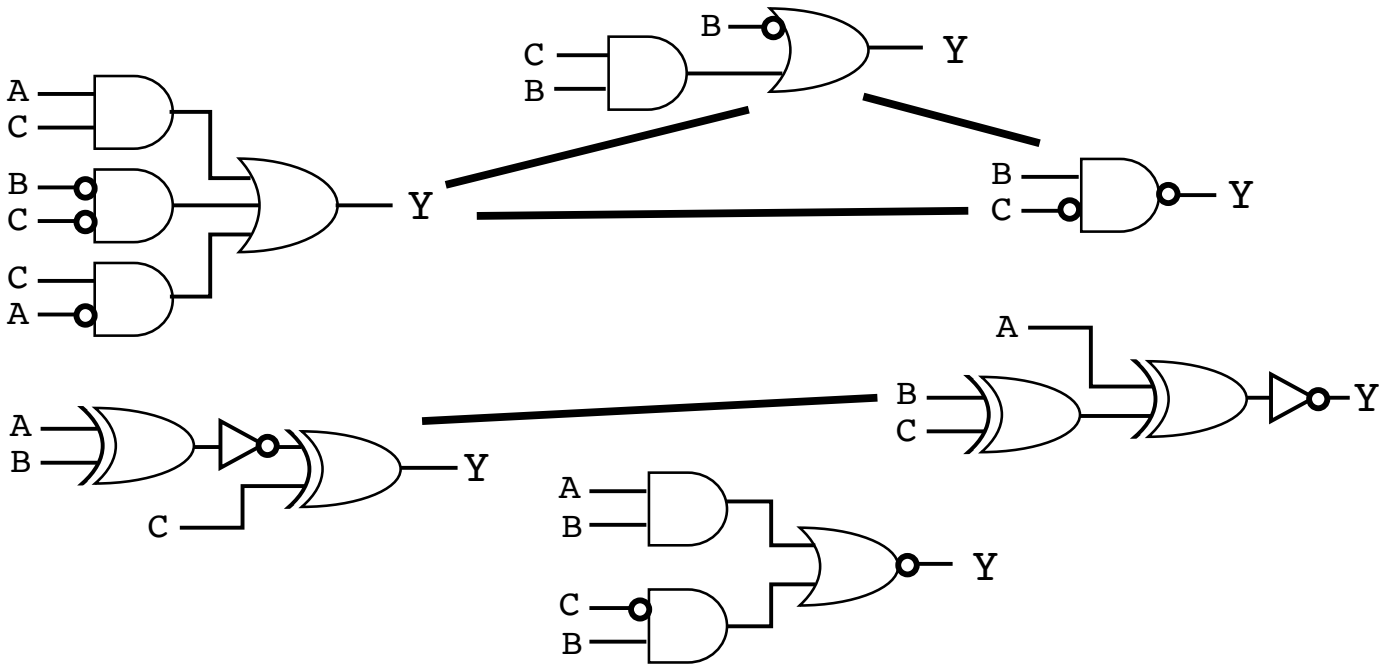
1. In the space below, draw a circuit whose behaviour matches the following waveform. **(4 marks)**



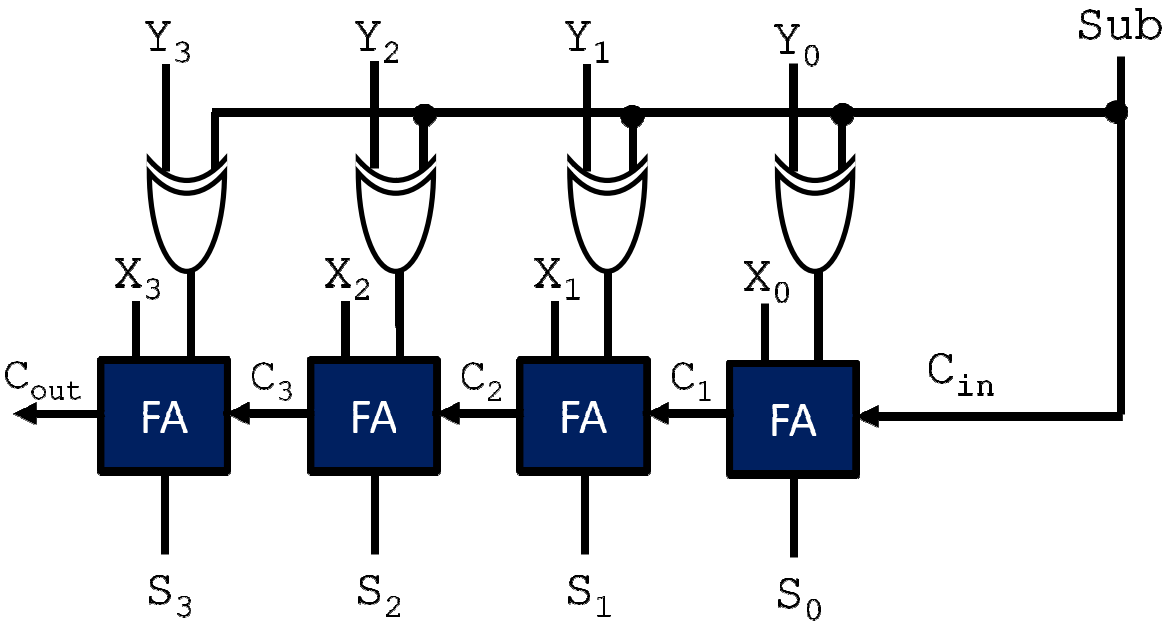


in1	in2	prev.	cur.
0	0	0	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

2. Which of the circuits below have equivalent behaviour? Draw lines that connect any circuits that match. **(6 marks)**

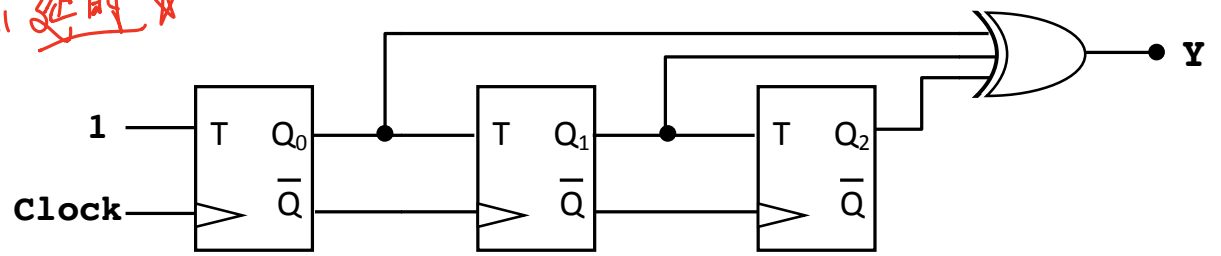


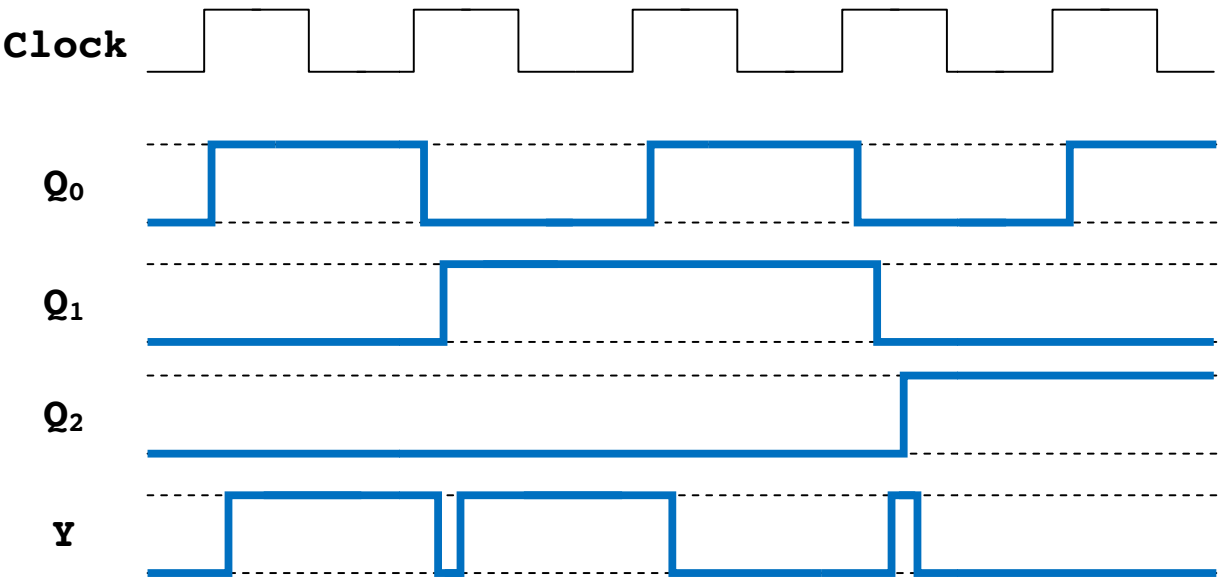
3. Draw gates into the following diagram to implement a subtractor circuit. **(2 marks)**



4. Given the following circuit, show what the output value of Q_0, Q_1, Q_2 and Y will be in the waveform diagram. Assume that Q_0, Q_1 and Q_2 start with initial values of zero. **(6 marks)**

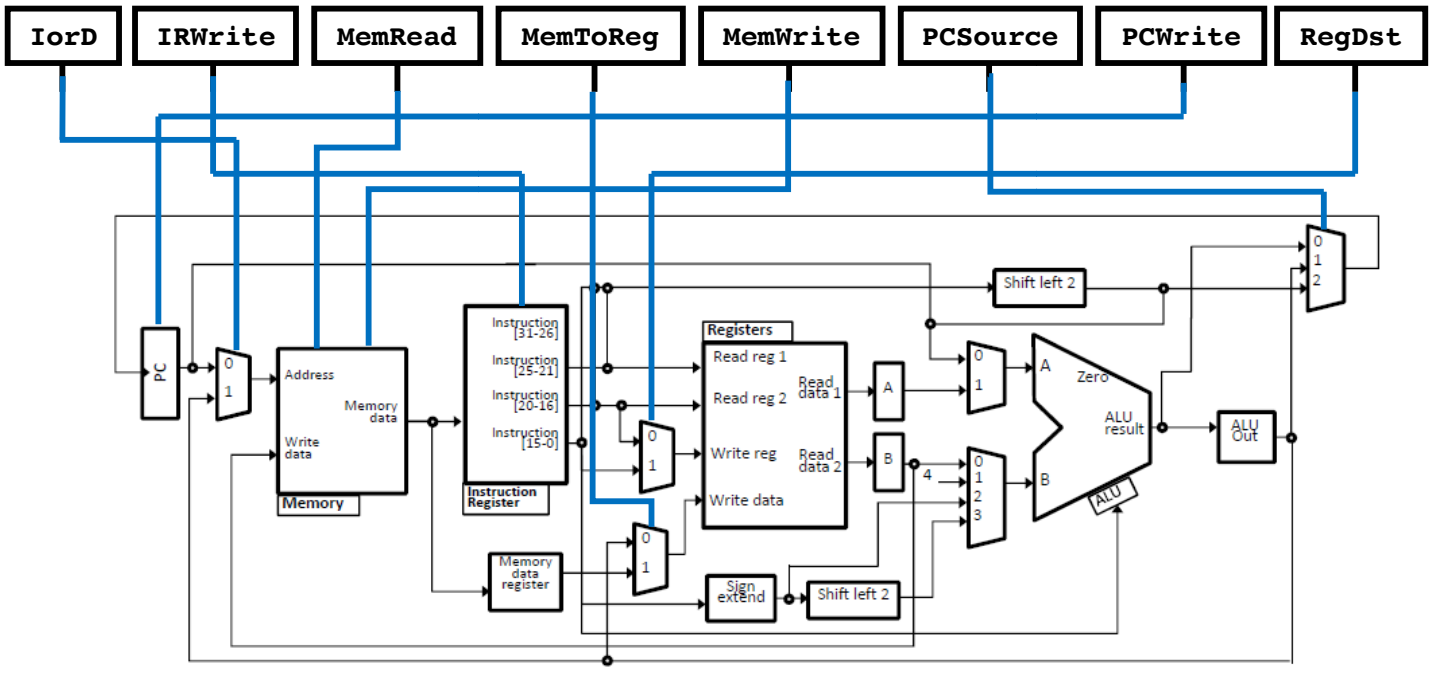
注意画延时★



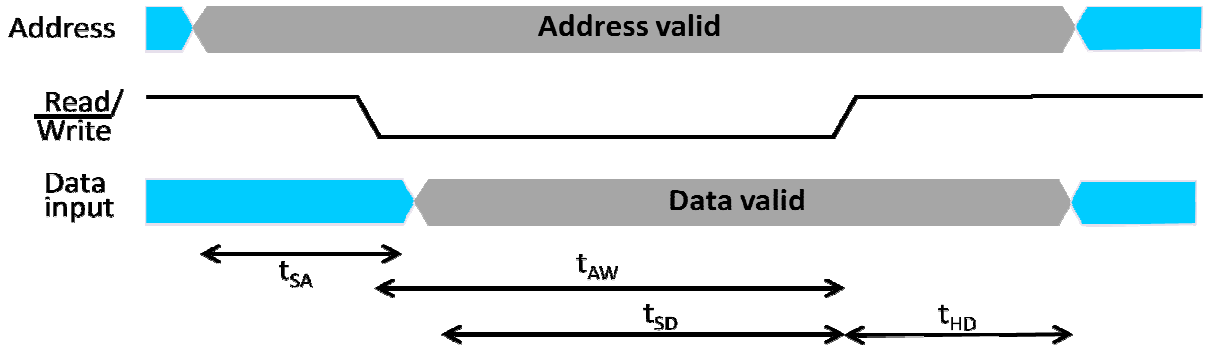


Part C: Processors (52 marks)

1. In the datapath diagram below, connect the following signals to the units that they affect. **(8 marks)**



2. For the following write signal diagram, describe what each labeled time segment is called, and the purpose of each segment during a memory write operation. **(8 marks)**



t_{SA}: **Setup address time: Time needed for address to be stable before writing.**

t_{AW}: **Address width time: Time that write signal is high.**

t_{SD}: **Setup data time: Time needed for data values to be set up for write.**

t_{HD}: **Hold data time: Time needed to maintain data values after write signal falls.**

3. In the space below, perform Booth's Algorithm on the binary values A=10110 and B=01101. Show your steps in the space provided. **(6 marks)**

$=-10_{(10)}$ $=13_{(10)}$

P = 00000 00000 $\quad -B = 10011$, $A = 10110\ 0$

Step 1:

$A' = 1011\ \boxed{00}$

P = 00000 00000
(no action)
P = 00000 00000

Step 2:

A' >>> 1

$A' = 101\ \boxed{10}\ 0$

P = 00000 00000
−B = 10011
P = 10011 00000

Step 3:

$A = 10\ \boxed{11}\ 00$

P = 11001 10000
(no action)
P = 11001 10000

Step 4:

$A = 1\ \boxed{01}\ 100$

P = 11100 11000
+B = 01101
P = 01001 11000

Step 5:

$A = \boxed{10}\ 1100$

P = 00100 11100
−B = 10011
P = 10111 11100

P = 11011 11110

4. Verify your answer above by writing the decimal values for A, B and the final product in the spaces below. **(2 marks)**

A =	**-10**
B =	**13**
Product =	**-130**

$A = 10110 = -10_{(10)}$ $\quad -A = 01010 = 10_{(10)}$

$B = 01101 = 13_{(10)}$ $\quad -B = 10010 = -13_{(10)}$

$A' = 10100$. $A * B = -130_{(10)}$

① $P = 00000, 00000 >>> 1 = 00000, 00000$

$A' >>> 1 = 11010$

② $P = 00000, 00000$

10011

$P = 10011, 00000 >>> 1 = 11001, 10000$

$A' >>> 1 = 11011$

③ $P >>> 1 = 11100, 11000$

$A' >>> 1 = 11101$

④ $P = 11100, 11000$

01101

$P = 01001, 11000 >>> 1 = 00100, 11100$

$A' >>> 1 = 1111110$

⑤ $00100, 11100$

$-B = 10011$

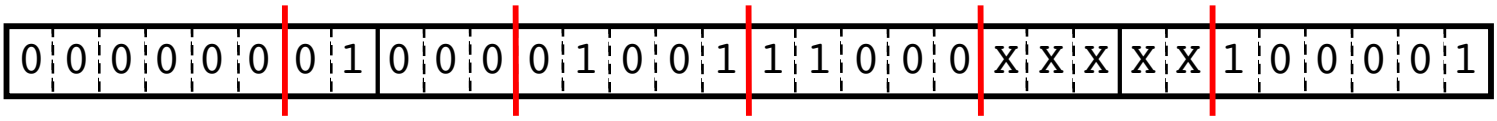
$P = 10111, 11100 >>> 1 = 11011, 11110$

$-P = 00100, 00010 = 128+2 = 130_{(10)}$

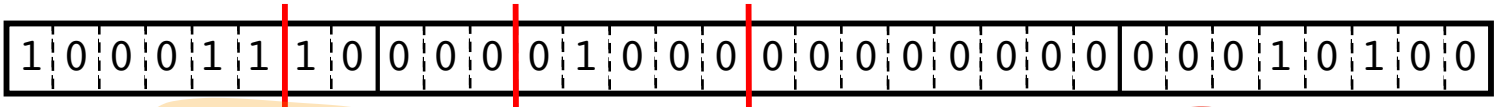
∴ P = -130

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. **(10 marks)**

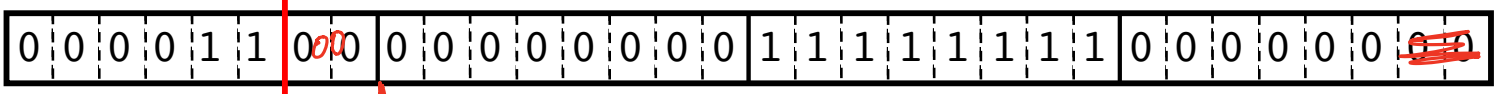
a) `addu $t2, $t0, $t1`



b) `lw $t0, 20($s0)`



c) `jal top` (where top is at hexadecimal address 0xFF00)



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

a) 00111001000000100000000011111111

xori $v0, $t0, 255

b) 00000010000001000100000000100011

subu $t0, $s0, $a0

c) 00100111111100000000000000000000

addiu $zero, $ra, 0

1. Enable Signal: PCwrite, PCwriteCond, MeRead, MemWrite, IRWrite, RegWrite, → 不是 0, 就是 1

2. MUX Signal: IorD, MemToReg, PCSource, ALUOp, ALUSrcA, ALUSrcB, RegDst → 要么是一个数, 要么是 'x'

if PCwrite=1 ⇒ PcWriteCond=X

else ⇒ PcWriteCond = 0 不写PC

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. **(12 marks)**

- If a control signal doesn't affect the operation, fill in its value with an X.
- For ALUOp, if you don't know the values, just write what kind of operation is taking place.

Reduce the program counter by the value stored in $t0. PC=PC-$t0

PCWrite	1	PCWriteCond	X	IorD	X	MemRead	0	MemWrite	0
MemToReg	X	IRWrite	0	PCSource	00	ALUOp	010 ← 减法		
ALUSrcA	0	ALUSrcB	00	RegWrite	0				

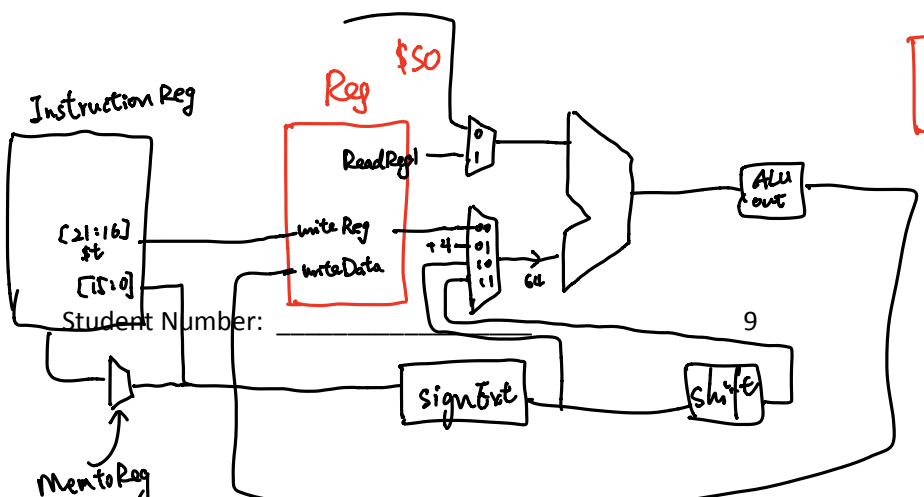
01 → 加4 进行下一个指令

Fetch the next instruction from the address in the program counter.

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

Add 64 to $s0 and store the result back in $s0. $S0 = $S0 + 64

PCWrite	0	PCWriteCond	0	IorD	X	MemRead	0	MemWrite	0
MemToReg	0	IRWrite	0	PCSource	XX	ALUOp	001 ← 加法		
ALUSrcA	1	ALUSrcB	~~01~~ 10	RegWrite	1				



addi, $s0, $s0, 64

Part D: Verilog (14 marks)

Consider the piece of Verilog code on the right.

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

It's an ALU.

2. Given your answer to part 1, what input signal is missing from this device? **(2 marks)**

It's missing a carry bit input.

```
module foo(a, b, s, r);

   input [31:0] a, b;
   input [2:0] s;
   output reg [31:0] r;

   always @(*) begin
     case(s[2:0])
       3'b000: r = a;
       3'b001: r = a + b;
       3'b010: r = a - b;
       3'b011: r = a - 1;
       3'b100: r = a & b;
       3'b101: r = a | b;
       3'b110: r = a ^ b;
       3'b111: r = ~a;
     endcase

   end

endmodule
```

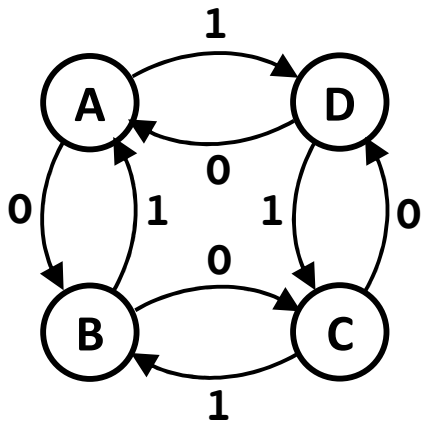
3. In the space below, write a Verilog module called `counter` that takes in input signals called `clock`, `reset` and `enable`, and has a 4-bit output signal called `value`. **(3 marks)**

- Make the `value` output increment if `enable` is on when the clock goes high **(3 marks)**
- Implement an asynchronous reset that is also positive-edge triggered. **(2 marks)**

```
module counter (clock, reset, enable, value);

     input clock, reset, enable;
     output value;

     reg value;

     always @ ( posedge clock or posedge reset)
     if (reset) begin
       q <= 1'b0;
     end else if (enable) begin
       q <= q+1;
     end

endmodule
```

Part E: Finite State Machines (20 marks)

1. Consider the following state machine and flip-flop assignments below. Assume that the input to this state machine is a single input called X.



Flip-Flop Assignments

State A → $F_1 = 0$ $F_0 = 0$

State B → $F_1 = 0$ $F_0 = 1$

State C → $F_1 = 1$ $F_0 = 1$

State D → $F_1 = 1$ $F_0 = 0$

Fill in the truth table below with the values that correspond to the FSM above. **(8 marks)**

F_1	F_0	X	F_1	F_0
0	0	0	**0**	**1**
0	0	1	**1**	**0**
0	1	0	**1**	**1**
0	1	1	**0**	**0**
1	0	0	**0**	**0**
1	0	1	**1**	**1**
1	1	0	**1**	**0**
1	1	1	**0**	**1**

2. Given the truth table values on the previous page, fill in the Karnaugh map shown below, and circle the largest minterm groupings possible. **(6 marks)**

F_1:

	$\overline{F}_1 \cdot \overline{F}_0$	$\overline{F}_1 \cdot F_0$	$F_1 \cdot F_0$	$F_1 \cdot \overline{F}_0$
$\overline{X}$	0	1	1	0
X	1	0	0	1

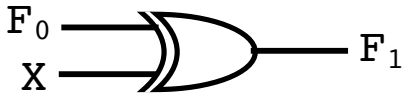
F_0:

	$\overline{F}_1 \cdot \overline{F}_0$	$\overline{F}_1 \cdot F_0$	$F_1 \cdot F_0$	$F_1 \cdot \overline{F}_0$
$\overline{X}$	1	1	0	0
X	0	0	1	1

3. Given the Karnaugh map groupings from the previous part, write the boolean equations that express these groupings, and draw the resulting circuit diagram that implements this behaviour in the space below. **(6 marks)**

$$F_1 = F_0 \oplus X$$

$$F_0 = F_1 \oplus \overline{X}$$



Part F: Assembly Language (30 marks)

1. In the spaces provided below, write the assembly language instruction(s) that corresponds to each of the tasks provided. **(12 marks total)**

a) Perform a right arithmetic shift on the value in `$t0`. The number of bits to shift `$t0` by is stored in `$s0`. The result will be stored back into `$t0`. **(3 marks)**

```
srav $t0, $t0, $s0
```

b) Load half a word from a memory address stored at `$a0`, and store that value in register `$t0`. The upper bits of `$t0` should be filled with zeroes. **(3 marks)**

```
lhu $t0, 0($a0)
```

c) Jump back to the instruction at location top if the value in `$t4` is anything other than zero. **(3 marks)**

```
bne $t4, $zero, top
```

d) Take the value that is already stored in `$t0`, and set all of its bits to zero except for the last 3 digits, which are kept as their original values. **(3 marks)**

```
andi $t0, $t0, 3
```

Consider the assembly language program in the box below.

```
          .data
list:     .word     3, 0, 1, 2, 6, -2, 4, 7, 3, 7
          .text
main:     addi $s0, $zero, list     # store the first list location in $s0
          addi $t9, $zero, 10       # set the stopping value for the loop
top:      lw $t1, 0($s0)            # get the current list value
          lw $t2, 4($s0)            # get the next list value
          sub $t3, $t2, $t1         # store list[i+1] – list[i] in $t3
          bgtz $t3, inc             # if list[i+1] < list[i],
          sw $t2, 0($s0)            #    swap the values of
          sw $t1, 4($s0)            #    list[i] and list[i+1]
inc:      addi $s0, $s0, 4          # move $s0 to the next index in list
          subi $t9, $t9, 1          # reduce our counter value
          bgtz $t9, top             # repeat if the counter > 0
end:      jr $ra                    # return to calling program
```

2. For each line in the code, provide a short descriptive comment on the right **(6 marks)**

3. In the space below, provide a one-sentence description of the overall task this code is trying to perform. **(2 marks)**

It's a single iteration of the inner loop of bubble sort.

4. One of the lines has a bug in it. What would need to be changed in order for this operation to be correct? **(2 marks)**

The initial value of `$t9` should be 9, not 10.

5. In the space below, write a short assembly language program that is a translation of the program on the right. You can assume that `i` has been placed on the top of the stack, and should be replaced by the return value before returning to the calling program. Make sure that you comment your code so that we understand what you're doing. **(8 marks)**

```
int make_even (int i) {
   if (i % 2 == 1)
      return i-1;
   else
      return i;
```

```
.data
i:              .word

.text
make_even:      addi $sp, $sp, 4          # move stack pointer
                lw $t0, 0($sp)            # get parameter from the stack
                addi $a0, $zero, i        # store a pointer to i
                sw $t0, 0($a0)            # store the parameter in i
                addi $t1, $zero, 2        # store 2 in $t1
                div $t0, $t1              # divide $t0 by 2
                mfhi $t1                  # store remainder in $t1
                beq $t1, $zero, end       # if $t0 is even, then branch
odd:            addi $t0, $t0, -1         # reduce i by 1
                sw $t0, 0($a0)            # store updated value of i
end:            sw $t0, 0($sp)            # push parameter onto stack
                addi $sp, $sp, -4         # move stack pointer
                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 table:

Register values : Processor role

- Register 0 ($zero): value 0.
- Register 1 ($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)

Instruction table:

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

This page is left blank intentionally for answer overflows.

This page is left blank intentionally for answer overflows.

Total Marks = 155

Total Pages = 18

End of exam