PLEASE HAND IN

PLEASE HAND IN

UNIVERSITY OF TORONTO
Faculty of Arts and Science

APRIL 2016 EXAMINATIONS

CSC 258 H1S

Instructors: Chin, Papadopoulou

Duration — 3 hours

No Aids Allowed

You must earn at least 42 out of 105 marks (40%) on this final examination in order to pass the course. Otherwise, your final course grade will be no higher than 47%.

Student Number:	
Last (Family) Name(s):	
First (Given) Name(s):	

Do **not** turn this page until you have received the signal to start.
In the meantime, please read the instructions below carefully.

This final examination paper consists of 10 questions on 20 pages (including this one), printed on both sides of the paper. *When you receive the signal to start, please make sure that your copy of the paper is complete and fill in your name and student number above.*

- If you use any space for rough work, indicate clearly what you want marked.
- Do not remove pages or take the exam apart.

# 1:	/ 17
# 2:	/ 8
# 3:	/ 12
# 4:	/ 8
# 5:	/ 10
# 6:	/ 10
# 7:	/ 10
# 8:	/ 10
# 9:	/ 8
# 10:	/ 12
TOTAL:	/105

Good Luck!

Question 1. [17 MARKS]

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

Part (a) [3 MARKS]

What is the function of the PC?

Part (b) [1 MARK]

What is the function of the hi and lo registers in MIPS? ______________________________

Part (c) [4 MARKS]

A memory has 8-bit address port and a 10-bit data port.

How many rows/words of data can this memory store? ____________________

How many memory cells does the memory contain? ____________________

Part (d) [4 MARKS]

```
.data
mydata: .word 12, 6
.text
  <some more code>
```

Assuming that the text section starts at memory address 0x534532AC and the data section starts at memory address 0x88843ABC, what instruction(s) does the assembler translate the following pseudo instruction to? Do not convert hexadecimal constants to decimal values.

```
la      $a0, mydata
```

____

____

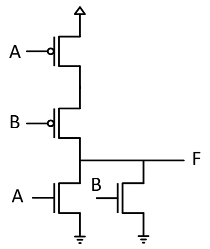
____

Part (e) [2 MARKS]

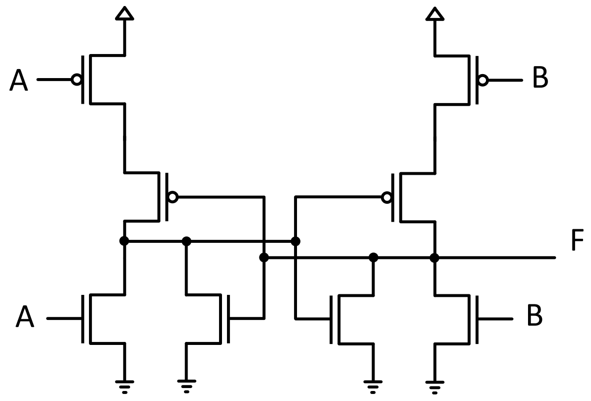
Why do we need the stack?

Part (f) [3 MARKS]

Recall the design of a NOR-gate:



What does the following circuit implement?



Question 2. [8 MARKS]

Which of the following are valid implementations of the Boolean function $f(a, b, c, d)$ shown in the Karnaugh map? For each function circle Valid or Invalid. Hint: It may help to draw more K-maps.

f $cd \backslash ab$	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	0	1	0	0
10	0	1	1	1

Valid Invalid $f = \bar{a}d + ac\bar{d} + a\bar{b}\bar{c}$

Valid Invalid $f = a\bar{d} + \bar{a}\bar{c}d + \bar{a}bc$

Valid Invalid $f = (\bar{a} + \bar{d})(a + c + d)(a + b + \bar{c})$

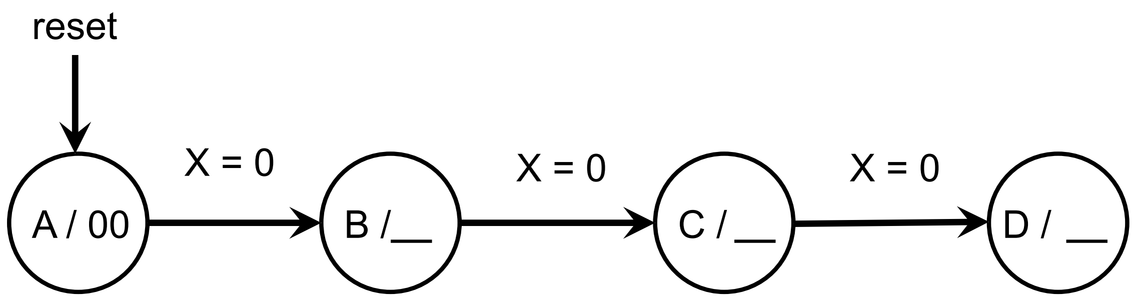
Valid Invalid $f = (\bar{a} + \bar{d})(a + c + d)(a + b + \bar{c})(a + b + d)$

Question 3. [12 MARKS]

You need to design a Moore type FSM that given a single bit input X is able to recognize over time 3-bit input sequences 010 and 000. This FSM should have a 2-bit output Y. The most significant bit of Y should be 1 if we have just recognized a 3-bit sequence starting and ending with 0. The least significant bit of Y should match the middle bit of the recognized sequence (i.e., should be 0 for 000 and 1 for 010) or 0 if none of these two sequences was recognized. Overlapping sequences should also be recognized.

Part (a) [8 MARKS]

Draw the state diagram for the aforementioned FSM in the space below. We have already drawn the initial state (A) and its 2-bit output for you, along with a couple more states (B to D). You will need to add a few more states and any necessary transitions and outputs to that state diagram. Please continue to use capital letters for your state names. Make sure you clearly annotate the input value X for each state as well as its 2-bit output Y.



Part (b) [2 MARKS]

Answer the following questions for the FSM you just designed:

- How many flip-flops would you need if you were using fully-encoded states? ____________
- How many flip-flops would you need if you were using one-hot encoding? ____________

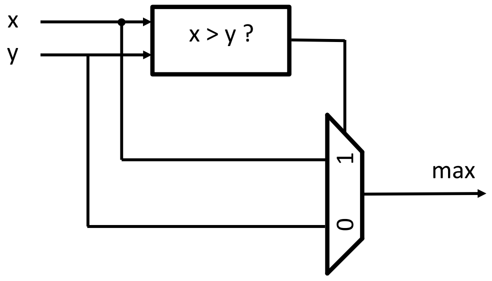
Part (c) [2 MARKS]

Assume you have the following Verilog snippet where *current_state* corresponds to outputs Q of your flip-flops and *next_state* corresponds to the inputs D of your flip-flops. Fill in the blanks so that the 2-bit FSM output Y has the correct value. You can assume that N, used in the Verilog snippet below, has the appropriate value for your FSM. Refer to the beginning of this question for the meaning of each bit of the output Y.

```
reg [N:0] current_state, next_state;
wire [1:0] Y;
parameter A = N'd0, B = N'd1, C = N'd2, D = N'd3, ....; // Assume other states as needed.


assign Y[1] =  ____________________________________________________________ ;

assign Y[0] =  ____________________________________________________________ ;
```

Question 4. [8 MARKS]

The combinational circuit shown in the diagram below implements a max function on two $12-bit$ unsigned inputs x and y.

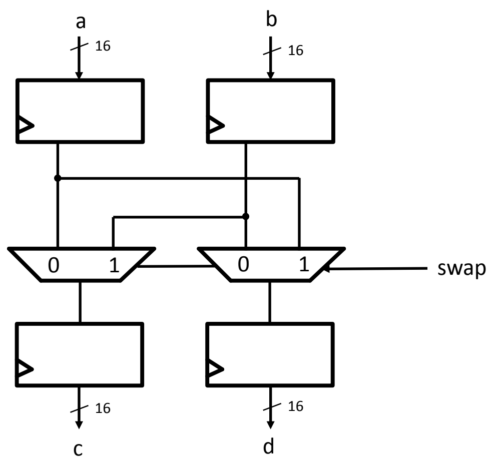


The following code has bugs. Find and correct all mistakes in the Verilog code by crossing out incorrect lines and adding necessary code.

```
module buggy_max(x, y, max);
    input x, y;



    reg  [12:0] out;



    always@(*) begin



        if(x > y)



            out = x;



    end



    assign max = out;



endmodule
```

Question 5. [10 MARKS]

The following schematic implements a circuit that optionally swaps two input values a and b to the outputs c and d depending on input *swap*. Both the inputs and the outputs are registered and carry $16-bit$ unsigned data. The connections to *clk* are not shown. There is no reset. The *swap* input controls both multiplexers. **Write Verilog that implements this circuit in a module named *OptionalSwap16*. Behavioural Verilog may be a good option...**

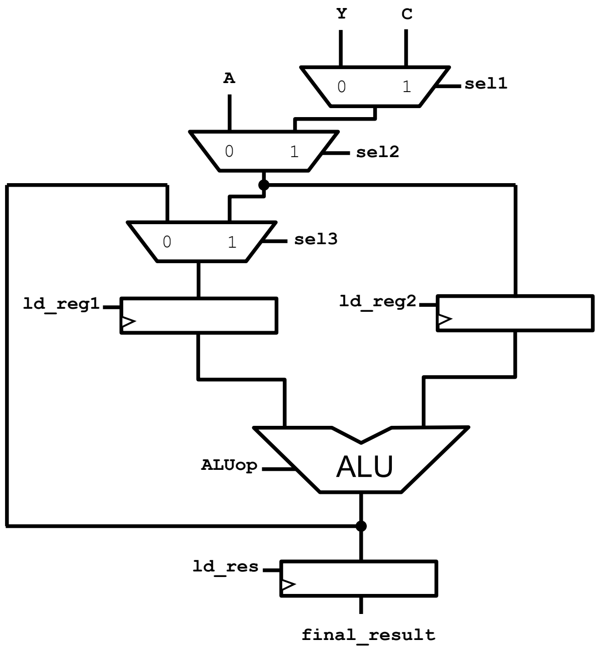


Question 6. [10 MARKS]

You need to use the following datapath to perform the operation $A * Y + C$. Fill in the table below to show the values the various control signals should have cycle by cycle. You will not need all rows of this table. If the value of a given control signal does not matter, **mark it as a *don't care* (i.e., X).**

You may assume the following:

- All registers share the same clock and reset signals, and all registers are initially reset to 0.
- The provided ALU supports addition ($ALUop = 0$) and multiplication ($ALUop = 1$) operations.
- A, Y, and C are all 8-bit unsigned values and the signals will remain stable.
- All registers are 8-bit wide, and the ALU result can always fit in 8-bits.
- Only the final result should be loaded into the result register.



Cycle #	Brief Description	sel1	sel2	sel3	ld_reg1	ld_reg2	ld_res	ALUop
Cycle 1	load value of A into register reg1							
Cycle 2								
Cycle 3								
Cycle 4								
Cycle 5								
Cycle 6								
Cycle 7								

Question 7. [10 marks]

```
C0: bgtz    $a0, C3
C1: sub     $v0, $zero, $a0
C2: j       C4
C3: addi    $v0, $a0, 0
C4: jr      $ra
```

Part (a) [4 marks]

In one sentence, what does this assembly do?

Part (b) [6 marks]

Write the machine code in binary for these three instructions:

C1:

31 0

C3:

31 0

C4:

31 0

Question 8. [10 MARKS]

Convert the following code to MIPS assembly in the space below. We have already written the initialization code for you. The values of a, b, and the constant 100 are in registers `$t0`, `$t1` and `$t3` respectively.

You may **not** need all the lines below.

```
-----------------
a = 0
b = 1
while a != 100:
    a = a + 1
    b = b - a
a = b * 4
-----------------
```

MIPS Assembly Code:

```
      addi $t0, $zero, 0;  # $t0 holds value of a
      addi $t1, $zero, 1;  # $t1 holds value of b
      addi $t3, $zero, 100;

LOOP: ____________________________________________
```

Question 9. [8 marks]

The current status of the relevant MIPS processor registers and part of data memory are as follows. Show the status of the processor's registers and the contents of memory **after** the following code executes. Assume little-endian for loads and stores. Only show the changes. Leaving empty spaces will mean no change took place.

Register	Contents
$t0	00000010
$t1	BAADF00D
$t2	11BA05E0
$t3	CA5E77EE
$t4	FFF87A4C
$t5	FFF87A5C
$t6	00000000
$t7	FFF87A54

Address	+0	+1	+2	+3
0xFFF87A4C	FF	AB	CD	42
0xFFF87A50	42	42	42	42
0xFFF87A54	C0	00	FF	42
0xFFF87A58	EF	BE	AD	DE
0xFFF87A5C	F0	EE	FE	CA
0xFFF87A60	FF	AB	CD	42

```
add     $t2, $t2, $t0
sw      $t2,  0($t4)
lw      $t0,  4($t7)
lhu     $t6,  10($t7)
sb      $t3,  3($t7)
lb      $t7, 8($t7)
```

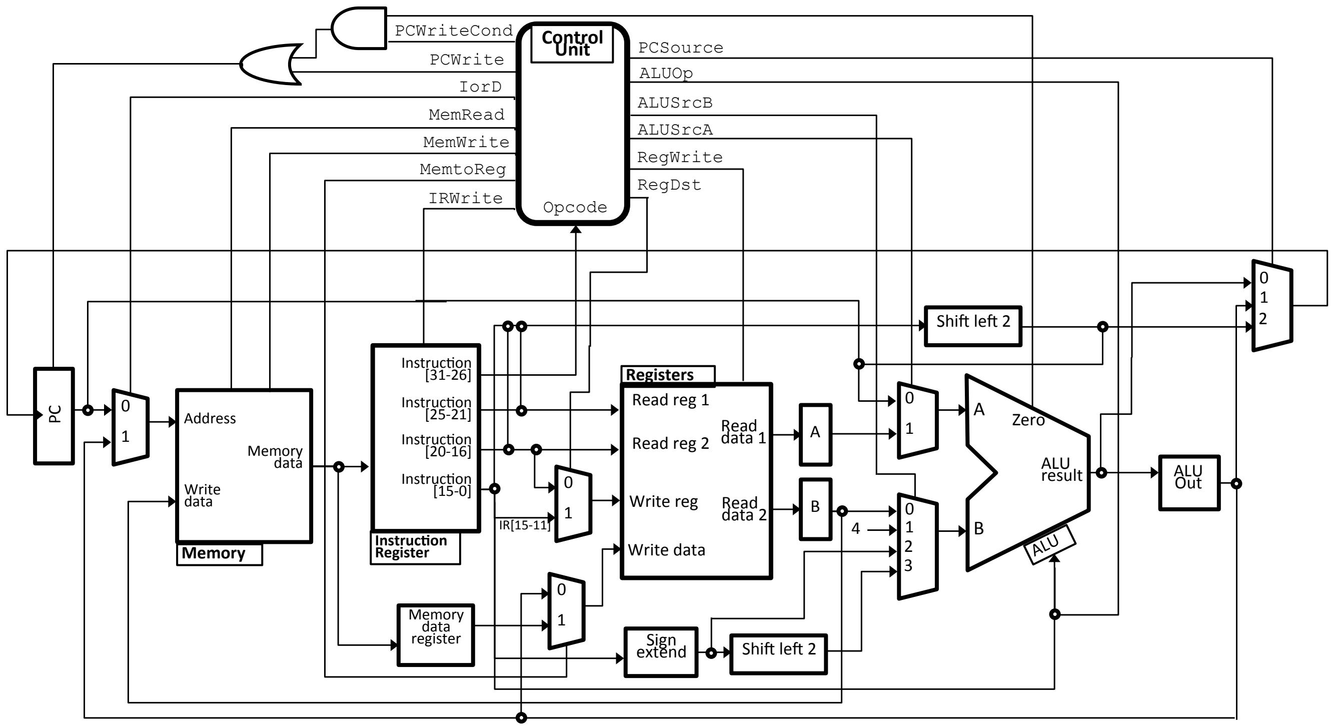
Register	Contents
$t0	
$t1	
$t2	
$t3	
$t4	
$t5	
$t6	
$t7	

Address	+0	+1	+2	+3
0xFFF87A4C				
0xFFF87A50				
0xFFF87A54				
0xFFF87A58				
0xFFF87A5C				
0xFFF87A60				

Question 10. [12 MARKS]

Part (a) [3 MARKS]

Answer the following questions based on the MIPS datapath provided below.

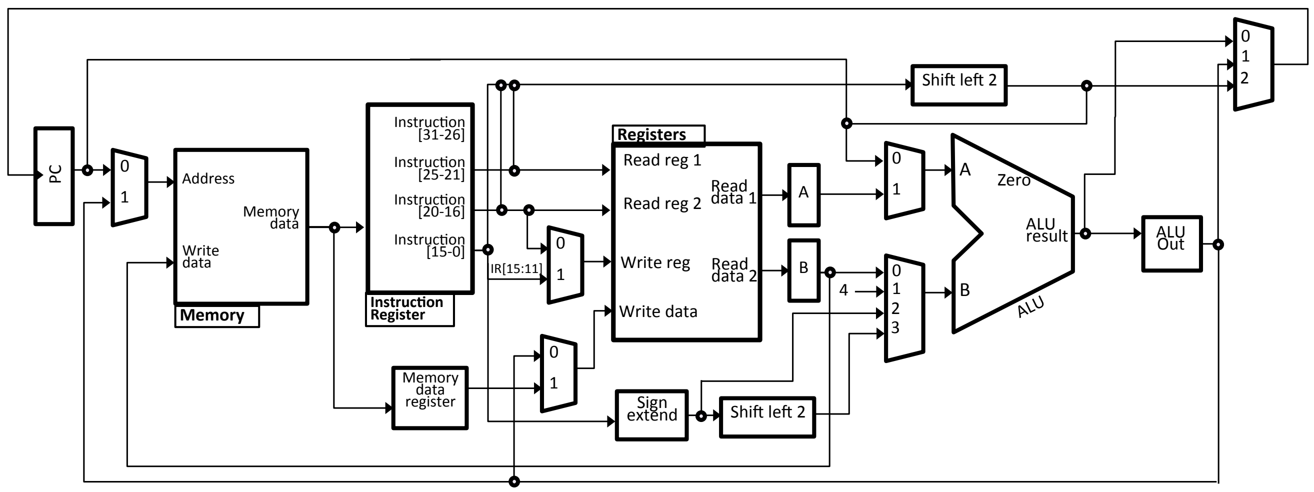


(i) If AluSrcA is 0 and AluSrcB is 1 and the ALU performs an addition operation, what will the ALUResult represent? Briefly explain the significance of this value.

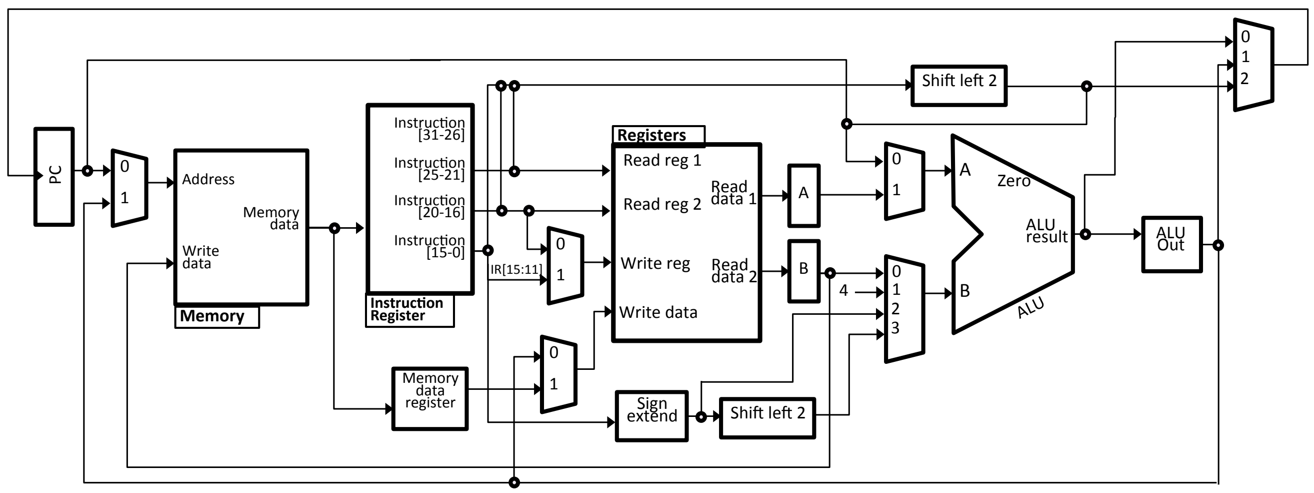
(ii) During the execution of which MIPS instruction(s) should the `MemToReg` signal be 1?

Part (b) [9 MARKS]

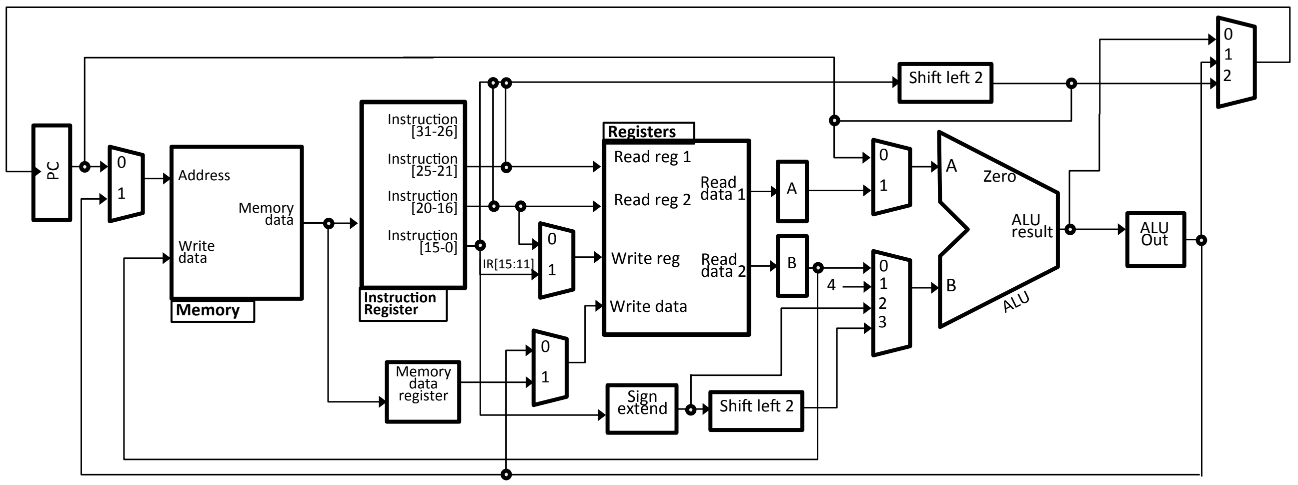
(i) Assume the instruction in your Instruction Register is `add $t0, $t1, $t2`. Highlight the parts of the datapath that will be exercised for the remaining execution of this instruction.



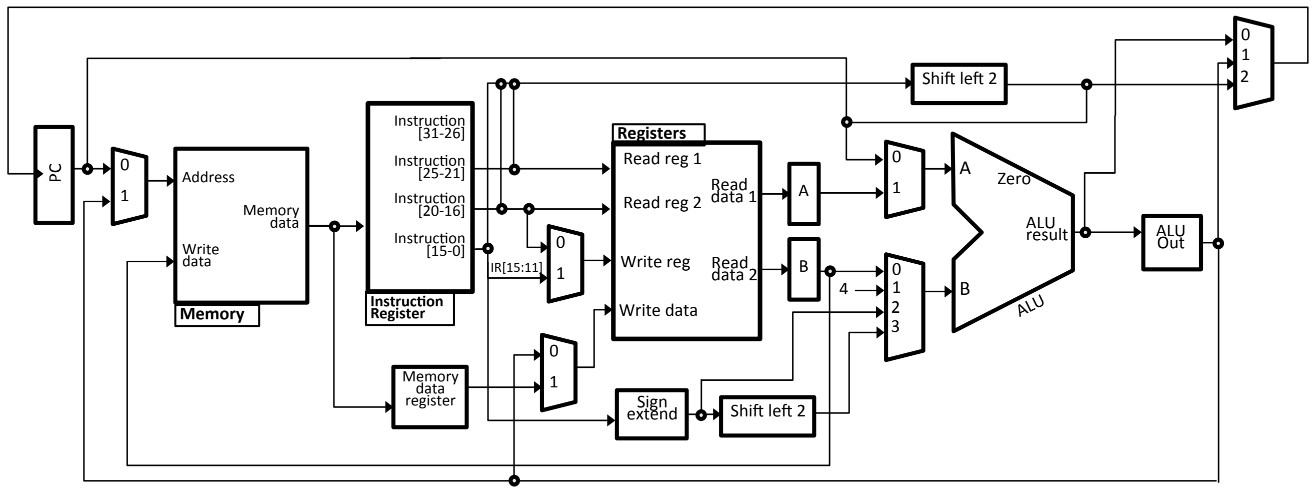
(ii) Assume the instruction in your Instruction Register is `addi $t0, $t3, -4`. Highlight the parts of the datapath that will be exercised for the remaining execution of this instruction.

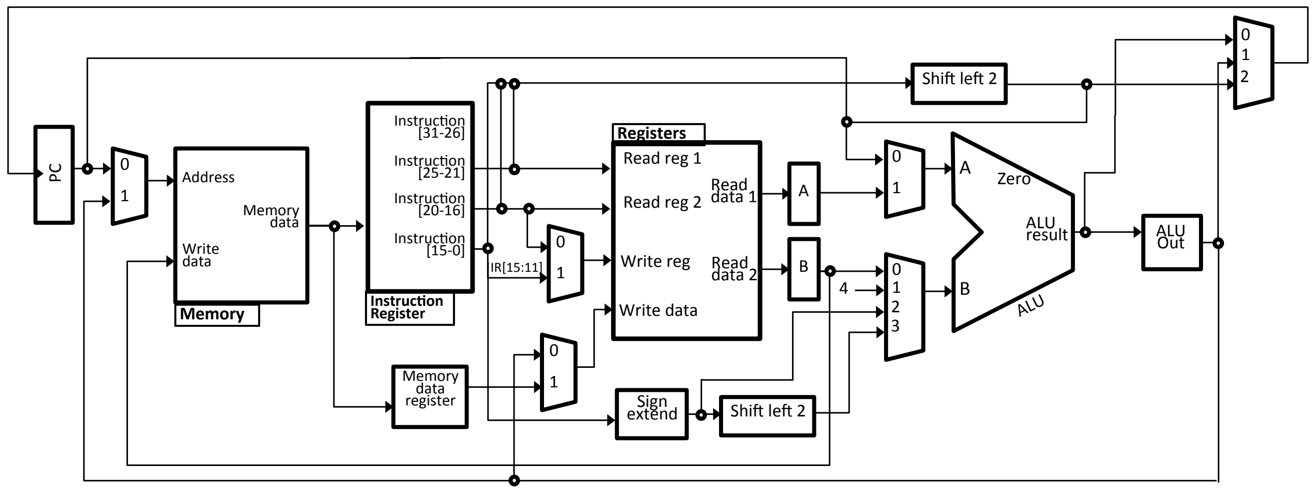


(iii) Assume the instruction you just fetched into the Instruction Register is a store. Specifically it is: `sw $t0, -4($t3)`. Highlight the parts of the datapath that will be exercised by the rest of the execution of this instruction.



You may want to use the space below for rough work for Question 10. This page will **not** be marked, unless you clearly indicate the part of your work that you want us to mark.





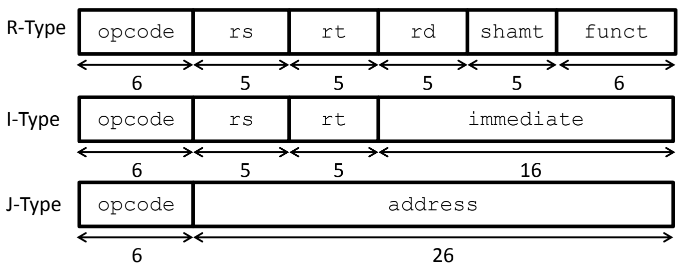
*[Use the space below for rough work. This page will **not** be marked, unless you clearly indicate the part of your work that you want us to mark.]*

[Use the space below for rough work. This page will ***not*** *be marked, unless you clearly indicate the part of your work that you want us to mark.]*

CONT'D...

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
lui	I	001111	\$t, i
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)
mflo	R	010010	\$d
mfhi	R	010000	\$d

Register Number	Name	Function
0	\$zero	reserved value
1	\$at	reserved for the assembler
2-3	\$v0, \$v1	return values
4-7	\$a0-\$a3	function arguments
8-15, 24-25	\$t0-\$t9	temporaries
16-23	\$s0-\$s7	saved temporaries
28-31	\$gp, \$sp, \$fp, \$ra	special purpose



Total Marks = 105

End of Final Examination