University of Western Ontario

Computer Science Department

CS3350B, Computer Organization

Take-Home Exam

April 13, 2020

Student ID Number:	**<Insert Your Student Number Here>**
First and Last Name:	**<Insert Your Name Here>**

Instructions: This exam consists of 9 pages (including this one) and 8 questions. Answer all questions. This exam is open-book. That is, you may use any of the class resources, lectures notes, your own notes, or the recommended textbook. However, you should not search the internet for answers nor should you consult with another person (in-person, online, or by any other means). Read every question carefully. Answer each question in the space provided below each question. This means filling in cells of a provided table or filling in the "answer box" below each question. You may fill in the answers directly in this word document, or you may use some writing application to draw your answers on top. In either case, show all workings.

Question	Possible Marks	Mark Received
1	10	
2	22	
3	8	
4	8	
5	10	
6	12	
7	20	
8	10	

Exercise 1 [10 Marks]

Consider the following table describing the number and type of instructions existing in some program, and the number of cycles needed to execute each instruction on some processor.

Operation	*Instruction Count*	*CPI*
ALU	2000	4
Load	600	5
Store	500	3
Branch	400	2

Part A

Compute the ideal CPI for this program.

Answer Box:

Part B

Compute the CPI for this program while taking into account stall cycles. That is, CPI_stall. Assume the processor has a 97% instruction hit rate, a 90% data hit rate, and a miss penalty of 50 cycles.

Answer Box:

Exercise 2 [22 Marks]

Consider a 4-way set associative cache with 4-byte cache lines, 2 sets, and an LRU (least recently used) replacement policy.

Part A

Starting with an empty cache, fill in the below table to indicate how the cache would look after all of the following sequential memory accesses have finished. The memory addresses are given as byte addresses. In the table, circle, bold, or color in red, the addresses which result in cache hits.

3, 7, 2, 1, 5, 8, 10, 17, 26, 6

Set 0				
Set 1				

Part B

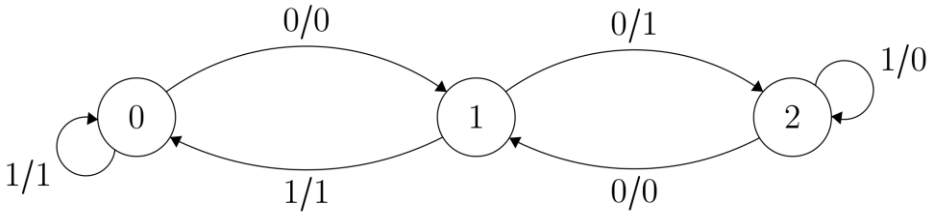
Starting with the cache in a state as in the **end of Part A,** fill in the below table to indicate how the cache would look after all the following sequential memory accesses have finished. The memory addresses are given as byte addresses. In the table, circle, bold, or color in red, the addresses which result in cache hits.

0, 1, 32, 12, 15, 35, 36, 37

Set 0				
Set 1				

Exercise 3 [8 Marks]

This question considers the following FSM.



Part A

Fill in the below truth table which describes the combinational logic circuit associated with the preceding FSM. Notice that each state has already been encoded as a number.

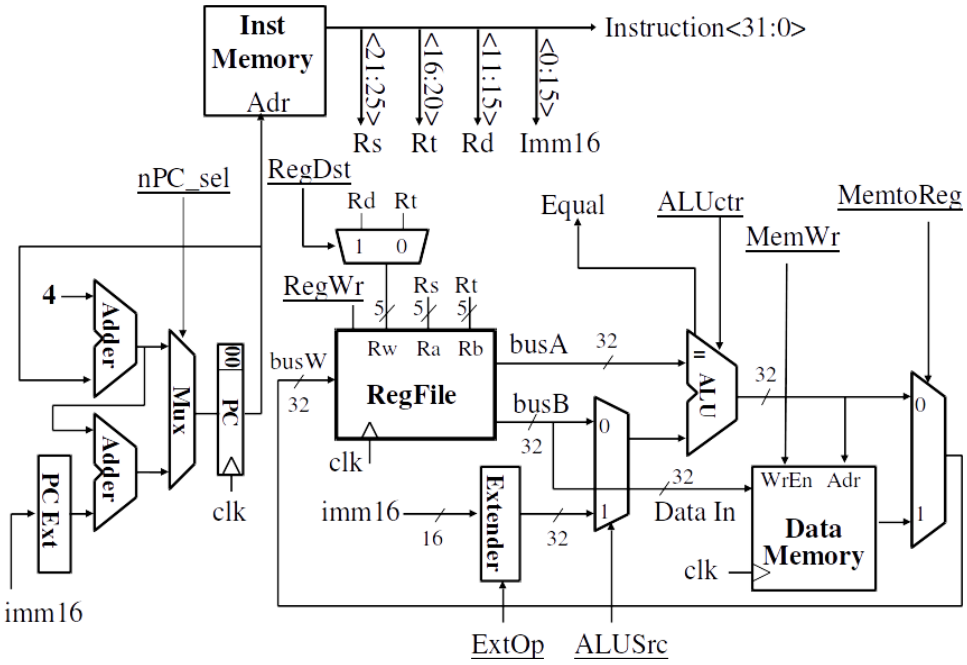
Present State	Input	Next State	Out

Part B

Consider implementing this FSM as a synchronous circuit. What is the minimum possible size, in bits, for the register to be used in that synchronous system?

Answer Box:

Exercise 4. [8 Marks]



Consider the above diagram of a single-cycle MIPS datapath and its associated control signals. This diagram shows 8 different control signals as labels on arrows where the labels are underlined. Give the value of each control signal during the execution of the instruction **slti** (set less than immediate). You may use "x" to denote a "don't care" value. You may use the semantic meaning for each control signal (e.g. ALUCtr = "add").

Answer Box:

Exercise 5 [10 Marks]

This question deals with the following MIPS code fragment.

```
lw   $t0, 0($s1)
lw   $t0, 0($s2)
addi $t2, $t0, 14
sw   $t1, 0($s1)
add  $t1 $t1 $t2
```

Part A:

Consider the basic MIPS pipeline without any structural hazards but without any forwarding. Using the table below, complete a pipeline diagram indicating the flow of the 5 instructions through the pipeline. Do not reorder instructions. Be sure to include "nop" instructions where appropriate to resolve any hazards.

	Clock Cycle														
Instr.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Part B:

Consider a MIPS pipeline without structural hazards but now with all possible types of forwarding as discussed in class. Using the table below, complete a pipeline diagram indicating the flow of the 5 instructions through the pipeline. Do not reorder instructions. Be sure to include "nop" instructions where appropriate to resolve any hazards.

	Clock Cycle														
Instr.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Exercise 6 [12 Marks]

Using register renaming, modify the following sequence of instructions to use unique logical destinations (a.k.a. value names). Complete the following table to answer the question. That is, not only should you specify the final renamed instruction, but you should also fill in each row to show the point in time when a register is assigned a new value name.

Instruction	$t0	$t2	$t4	$t6	Renamed Instr.
…	V0	V1	V2	V3	
sub $t4, $t0, $t2					
addi $t2, $t4, 12					
add $t2, $t0, $t6					
add $t6, $t2, $t0					
and $t0, $t2, $t2					
xor $t2, $t4, $t2					

Exercise 7 [20 Marks]

Consider a multi-core processor with 4 cores. All 4 cores have initially empty caches and are attempting to read and write to the same cache block. Use the MESI protocol to complete the following table, showing the state of each cache as requests are being processed. Specify the state of the cache block in each core after each request as well as the data supplier. If there are multiple possible data suppliers, list all of them. Also, specify whether each request resulted in a cache hit or a cache miss.

Request	State				Data Supplier	Cache Hit/Miss
	P1	P2	P3	P4		
Initially	-	-	-	-	-	-
P1 Read						
P2 Read						
P4 Read						
P3 Write						
P2 Read						
P1 Write						
P2 Read						
P3 Read						

Exercise 8 [10 Marks]

Consider the following code fragment. What are the possible values of p which could be printed at the end of the main method?

```
void modifyAddress(int* p, int a) {
    int val = *p;
    val *= a;
    val += 3;
    *p = val;
}

int main(int argc, char** argv) {
    int* p = new int[1];
    *p = 10;

    std::thread t1(modifyAddress, p, 2);
    std::thread t2(modifyAddress, p, 5);
    t1.join();
    t2.join();

    std::cerr << "p: " << *p << std::endl;

    return 0;
}
```

Answer Box: