Last Name: _____ **First Name:** _____

Student #: |_|_|_|_|_|_|_|_|_| **Signature:** _____

UNIVERSITY OF TORONTO MISSISSAUGA
APRIL 2015 FINAL EXAMINATION
CSC258H5S
Computer Organization
Andrew Petersen
Duration - 3 hours
Aids: None

The University of Toronto Mississauga and you, as a student, share a commitment to academic integrity. You are reminded that you may be charged with an academic offence for possessing any unauthorized aids during the writing of an exam. Clear, sealable, plastic bags have been provided for all electronic devices with storage, including but not limited to: cell phones, tablets, laptops, calculators, and MP3 players. Please turn off all devices, seal them in the bag provided, and place the bag under your desk for the duration of the examination. You will not be able to touch the bag or its contents until the exam is over.

If, during an exam, any of these items are found on your person or in the area of your desk other than in the clear, sealable, plastic bag; you may be charged with an academic offence. A typical penalty for an academic offence may cause you to fail the course.

*Please note, you **CANNOT** petition to **re-write** an examination once the exam has begun.*

This final examination consists of 6 questions on 20 pages (including this one). When you receive the signal to start, please make sure that your copy of the examination is complete.

If you need more space for one of your solutions, use the last pages of the exam and indicate clearly the part of your work that should be marked.

Please be sure that your handwriting is legible. Detailed answers will be given more credit than vague ones. Incorrect assertions will detract from otherwise correct answers.

MARKING GUIDE

1: _____/22
2: _____/ 6
3: _____/ 6
4: _____/ 8
5: _____/ 7
6: _____/11

TOTAL: _____/60

CONTINUED ON PAGE 2

22 marks **Question 1.** Short Answer

1 mark **Part (a)** Transistors

Sketch a transistor-level circuit for a 2-input CMOS NAND gate. You may use A, $\overline{A}$, B, and $\overline{B}$ as inputs to the transistors. (i.e., You don't need to implement a CMOS NOT gate if you need a negated input.)

1 mark **Part (b)** Binary Numbers

Provide the 6-bit two's complement representation for the decimal number -11.

1 mark **Part (c)** Two's Complement Arithmetic

Showing your work, subtract the 6-bit two's complement number 010001 from the two's complement number 100110. Indicate whether or not the computation overflows.

CONTINUED ON PAGE 3

1 mark

Part (d) Adders

Explain why a hardware designer might use a ripple-carry adder instead of a prefix adder.

2 marks

Part (e) Adders

Demonstrate how a prefix adder would compute the sum of the two 8-bit 2's complement numbers 01101111 and 00101000. (Suggestion: Draw the prefix network that would be used.)

1 mark

Part (f) Latches and Flip-flops

Succinctly describe the key difference between a latch and a flip-flop.

CONTINUED ON PAGE 4

1 mark

Part (g) Parallelism

Define the term *spatial parallelism* and provide a computer science example that demonstrates it.

2 marks

Part (h) Pipeline Hazards

Provide an example of a *control hazard* and briefly explain how a pipelined processor mitigates the impact of these hazards.

1 mark

Part (i) Instruction Set Architectures

MIPS does not have a branch when less-than assembly instruction like "*blt $s0, $s1, TARGET*". Why not?

CONTINUED ON PAGE 5

2 marks **Part (j)** MIPS Assembly

Provide a sequence of valid MIPS assembly instructions that implements "*blt $s0, $s1, TARGET*".

2 marks **Part (k)** Machine Code

Translate the instruction "*bgtz $s2, TARGET*" into MIPS machine code (in binary). Recall that register $s0 is register number 16. TARGET refers to a location in the code that is 5 instructions before the branch being translated.

2 marks **Part (l)** Memory Model

Please draw a figure representing the programmer's model of memory. Label the segments in your figure and indicate where the PC, SP, and FP registers point.

CONTINUED ON PAGE 6

2 marks

Part (m) Compiler Toolchain

List the major components of the compiler toolchain. For each component, describe the input the components requires and the output (or effect) it produces.

3 marks

Part (n) Locality

Define *spatial locality* and *temporal locality*. Provide a single programming example that demonstrates both.

CONTINUED ON PAGE 7

6 marks

Question 2. Caches

4 marks

Part (a) Cache Attributes

Assume the base configuration of a cache is direct-mapped with 16 blocks, each of size 256 bytes. Then, consider the cache configurations below. For each, provide one positive and one negative attribute of the cache relative to the base configuration.

a) Direct-mapped with 32 blocks of 128 bytes each.

b) Fully associative with 16 blocks of 256 bytes each.

2 marks

Part (b) Eviction

Assume a cache with 2 blocks, each of size 16 bytes. The cache is provided a sequence of loads on the specified addresses. For each address, note whether it is a (H)it or a (M)iss.

a) Direct-mapped, with the 5th least significant bit (the bit in the 16 position) determining where the address should be mapped.

Addresses	0x1A	0x3A	0x30	0x1A	0x24	0x14	0x30	0x28
Hit/Miss								

b) Fully associative using the least recently used (LRU) heuristic.

Addresses	0x1A	0x3A	0x30	0x1A	0x24	0x14	0x30	0x28
Hit/Miss								

CONTINUED ON PAGE 8

6 marks

Question 3. Logic Design

A	B	C	D	Out
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	X
0	1	0	0	1
0	1	0	1	X
0	1	1	0	0
0	1	1	1	1
1	0	0	0	X
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

2 marks

Part (a) K-Maps

Use a Karnaugh map to simplify the function described in the truth table above. **For this question, treat X values as 0's.** Provide the k-map and the optimized formula you derive from it.

1 mark

Part (b) K-Maps

Use a Karnaugh map to simplify the function described in the truth table above. **For this question, use X values to simplify the circuit as much as possible.** Provide the k-map and the optimized formula you derive from it.

CONTINUED ON PAGE 9

3 marks

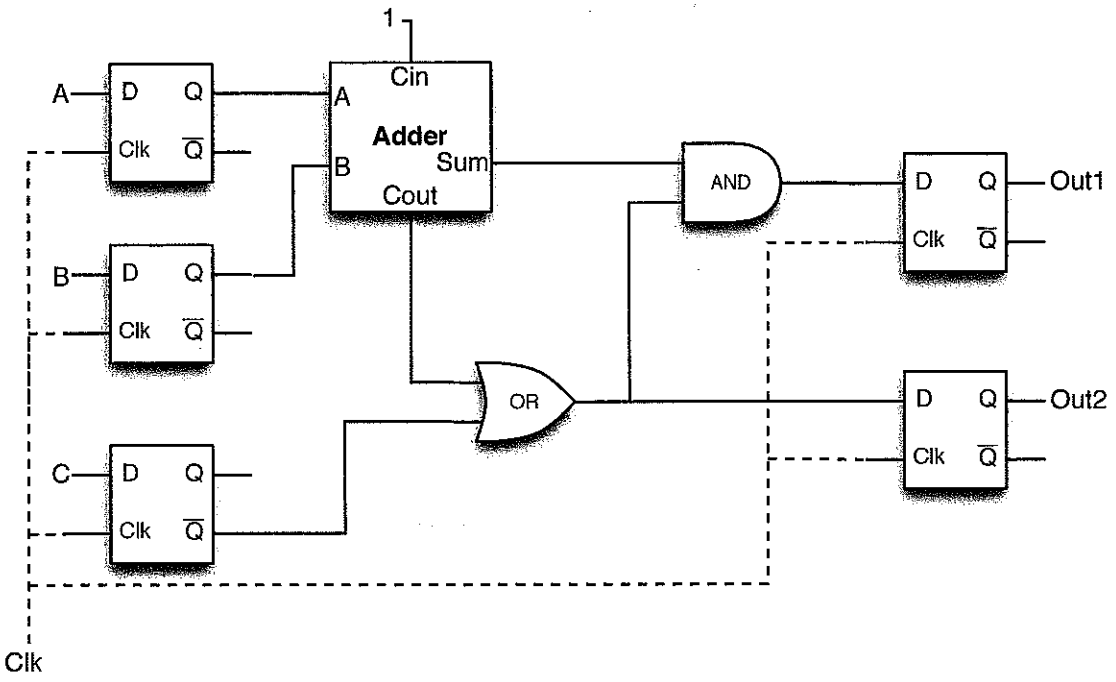
Part (c) Flip-flops

A toggle (T) flip-flop takes an input T and toggles to the complement of its previous value on the rising edge of the clock if T has value 1. Provide the **truth table and schematic** for the toggle flip-flop. In the schematic, you may only use SR latches (not flip-flops!) and combinational logic.

CONTINUED ON PAGE 10

8 marks

Question 4. Latency



For the following subquestions, use the circuit above and assume that each 2-input boolean gate has a propagation delay of 20 ns and contamination delay of 10 ns. (NOT gates are free.) Assume that a D flip-flop has a clock-to-q propagation delay of 60 ns, setup time of 25 ns, hold time of 35 ns, and clock-to-q contamination delay of 30 ns.

1 mark

Part (a) Adders

Draw the schematic for a **1-bit adder** like the one used in the circuit.

2 marks

Part (b) Combinational Logic Delays

What are the propagation and contamination delays of a **1-bit adder** like the one used in the circuit? Please specify the path taken for the delays you have calculated (e.g., "from A to Cout" or "Cin to Sum".)

CONTINUED ON PAGE 11

2 marks **Part (c)** Sequential Logic Propagation Delay

What is the propagation delay of the full sequential circuit on the previous page? Please identify the longest path (by highlighting it on the circuit schematic) and show your work.

1 mark **Part (d)** Hold Time

Does the circuit above contain a hold time violation? Explain why or why not.

1 mark **Part (e)** Clock Skew

What is the propagation delay of the circuit if the circuit can experience up to 20 ns of skew?

1 mark **Part (f)** Clock Skew

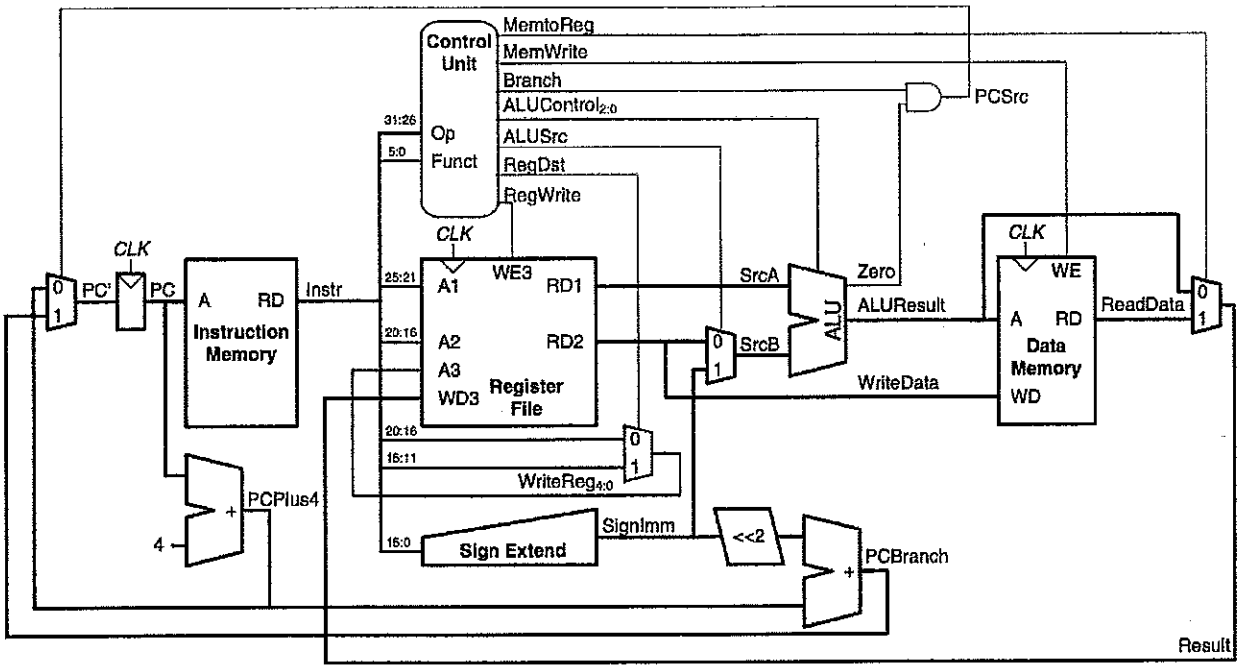
Does your answer to the question about hold time change if the circuit can experience up to 20 ns of skew?

CONTINUED ON PAGE 12

7 marks

Question 5. Architecture

The image below is from your textbook and represents the single-cycle MIPS processor design from chapter 7.



1 mark

Part (a) Program Counter

What is the role of the program counter (PC)?

2 marks

Part (b) Control Path

What is the function of the *ALCSrc* control line, and in particular, what instructions cause it to have the value *1*?

CONTINUED ON PAGE 13

4 marks

Part (c) Processor Extension

Consider how to modify the processor to implement the instruction *jal*. Describe the changes you would make below. If it would be helpful, you may mark up the figure on the previous page and refer to the diagram in your description. Be sure to provide a summary of the values of the control lines emitted by the decoder.

CONTINUED ON PAGE 14

11 marks **Question 6.** Programming in Assembly

2 marks **Part (a)** Function Calls

Explain how functions implemented in MIPS assembly (1) return to the correct calling location and (2) return a value to the caller.

3 marks **Part (b)** Exceptions

In the week 12 lab, you were asked to update the exception handler to handle the *address error in load* exception. Assuming you implemented the handler correctly, list the major events that occurred from the instant the bad load was executed to the instant control was returned to the user program.

CONTINUED ON PAGE 15

6 marks

Part (c) Assembly Programming

Provide the MIPS assembly code that allocates an integer array of size 1000 on the stack, assigns 42 to the elements in the array with an even index, and assigns 0 to the elements in the array with odd index. If you do not remember the opcode for an instruction you want to use, write a comment that explains what the assembly instruction should do.

Total Marks = 60

CONTINUED ON PAGE 16

MIPS Reference

Machine Encoding Aids

Key

o/f	instruction/function opcodes
s/t/d	first/second/third register
a/i	shift amount/immediate

Instruction Encoding Formats

Register	000000ss sssttttt dddddaaa aafffffff
Immediate	ooooooss sssttttt iiiiiiii iiiiiiii
Jump	oooooooi iiiiiiii iiiiiiii iiiiiiii

Instruction Syntax

Encoding	Syntax	Template
Register	ArithLog	f $d, $s, $t
	DivMult	f $s, $t
	Shift	f $d, $t, a
	ShiftV	f $d, $t, $s
	JumpR	f $s
	MoveFrom	f $d
	MoveTo	f $s
Immediate	ArithLogI	o $t, $s, i
	LoadI	o $t, immed32
	Branch	o $s, $t, label
	BranchZ	o $s, label
	LoadStore	o $t, i($s)
Jump	Jump	o label
	Trap	o i

Instruction Reference

Arithmetic and Logical Instructions

Instruction	Operation	Opcode or Function	Syntax	Comments
add $d, $s, $t	$d = $s + $t	100000	ArithLog	
addu $d, $s, $t	$d = $s + $t	100001	ArithLog	
addi $t, $s, i	$t = $s + i	001000	ArithLogI	i is sign-extended
addiu $t, $s, i	$t = $s + i	001001	ArithLogI	i is sign-extended
and $d, $s, $t	$d = $s & $t	100100	ArithLog	
andi $t, $s, i	$t = $s & i	001100	ArithLogI	i is zero-extended
div $s, $t	lo = $s / $t; hi = $s % $t	011010	DivMult	
divu $s, $t	lo = $s / $t; hi = $s % $t	011011	DivMult	
mult $s, $t	hi:lo = $s * $t	011000	DivMult	
multu $s, $t	hi:lo = $s * $t	011001	DivMult	
nor $d, $s, $t	$d = ~($s | $t)	100111	ArithLog	
or $d, $s, $t	$d = $s | $t	100101	ArithLog	
ori $t, $s, i	$t = $s | i	001101	ArithLogI	i is zero-extended
sll $d, $t, a	$d = $t << a	000000	Shift	Zero is shifted in
sllv $d, $t, $s	$d = $t << $s	000100	ShiftV	Zero is shifted in
sra $d, $t, a	$d = $t >> a	000011	Shift	Sign bit is shifted in
srav $d, $t, $s	$d = $t >> $s	000111	ShiftV	Sign bit is shifted in
srl $d, $t, a	$d = $t >> a	000010	Shift	Zero is shifted in
srlv $d, $t, $s	$d = $t >> $s	000110	ShiftV	Zero is shifted in
sub $d, $s, $t	$d = $s - $t	100010	ArithLog	
subu $d, $s, $t	$d = $s - $t	100011	ArithLog	
xor $d, $s, $t	$d = $s ^ $t	100110	ArithLog	
xori $d, $s, i	$d = $s ^ i	001110	ArithLogI	i is zero-extended

Movement Instructions

Instruction	Operation	Opcode or Function	Syntax	Comments
lhi $t, i	$t = i << 16	011001	LoadI	i is zero-extended
llo $t, i	$t = i	011000	LoadI	i is zero-extended
mfhi $d	$d = hi	010000	MoveFrom	
mflo $d	$d = lo	010010	MoveFrom	
mthi $s	hi = $s	010001	MoveTo	
mtlo $s	lo = $s	010011	MoveTo	

CONTINUED ON PAGE 20

Comparison Instructions

Instruction	Operation	Opcode or Function	Syntax	Comments
slt $d, $s, $t	$d = $s < $t	101010	ArithLog	
sltu $d, $s, $t	$d = $s < $t	101001	ArithLog	
slti $t, $s, i	$d = $s < i	001010	ArithLogI	i is sign-extended
sltiu $t, $s, i	$d = $s < i	001001	ArithLogI	i is sign-extended

Branch and Jump Instructions

Instruction	Operation	Opcode or Function	Syntax	Comments
beq $s, $t, label	if ($s == $t) pc += i << 2	000100	Branch	label is a line reference in the code
bgtz $s, label	if ($s > 0) pc += i << 2	000111	BranchZ	label is a line reference in the code
blez $s, label	if ($s <= 0) pc += i << 2	000110	BranchZ	label is a line reference in the code
bne $s, $t, label	if ($s != $t) pc += i << 2	000101	Branch	label is a line reference in the code
j label	pc += i << 2	000010	Jump	label is a line reference in the code
jal label	$ra = pc; pc += i << 2	000011	Jump	label is a line reference in the code
jalr $s	$ra = pc; pc = $s	001001	JumpR	
jr $s	pc = $s	001000	JumpR	

Memory Instructions

Instruction	Operation	Opcode or Function	Syntax	Comments
lb $t, i($s)	$t = MEM[$s + i]	100000	LoadStore	Sign-extends the loaded byte
lbu $t, i($s)	$t = MEM[$s + i]	100100	LoadStore	Zero-extends the loaded byte
lh $t, i($s)	$t = MEM[$s + i]	100001	LoadStore	Sign-extends the loaded bytes
lhu $t, i($s)	$t = MEM[$s + i]	100101	LoadStore	Zero-extends the loaded bytes
lw $t, i($s)	$t = MEM[$s + i]	100011	LoadStore	
sb $t, i($s)	MEM[$s + i] = $t	101000	LoadStore	Lowest order byte is stored
sh $t, i($s)	MEM[$s + i] = $t	101001	LoadStore	2 lowest order bytes are stored
sw $t, i($s)	MEM[$s + i] = $t	101011	LoadStore	

Exception and Interrupt Instructions

Instruction	Operation	Opcode or Function	Syntax	Comments
trap i	Exception	0011010	Trap	i is a trap code; implements syscall

END OF EXAMINATION