University of Calgary
Department of Electrical and Computer Engineering
ENCM 369: Computer Organization
Lecture Instructors: Steve Norman and Norm Bartley

Winter 2018 MIDTERM TEST ~~#1~~ #2

with corrections

Monday, March 12 — 7:00pm to 8:30pm

Please do *not* write your U of C ID number on this cover page.

Name (printed)**:**	
Signature:	
Lecture section (01 is MWF 11:00am with S. Norman, 02 is MWF 10:00am with N. Bartley)**:**	

General Instructions

- Marks will be recorded on the **last** page of this question paper. When you are told to start the test, the first thing you should do is to put your name, signature, U of C ID number, and lecture section in the appropriate spaces at the bottom of the last page.
- If you use a **calculator**, it must be one of the following models sanctioned by the Schulich School of Engineering: Casio FX-260, Casio FX-300MS, TI-30XIIS.
- The test is **closed-book**. You may not refer to books or notes during the test, with one exception: you may refer to the *Reference Material* booklet that accompanies this test paper.
- You are not required to add **comments** to assembly language code you write, but you are strongly encouraged to do so, because writing good comments will improve the probability that your code is correct and will help you to check your code after it is finished.
- Some problems are relatively **easy** and some are relatively **difficult**. Go after the easy marks first.
- To reduce distraction to other students, you are not allowed to leave during the last **ten minutes** of the test.
- Write all answers on the question paper and hand in the question paper when you are done.
- Please print or write your answers **legibly**. What cannot be read cannot be marked.
- If you write anything you do not want marked, put a large X through it and write “rough work” beside it.
- You may use the backs of pages for rough work.

PROBLEM 1 *(10 marks)*
Consider the C code listed to the right. Translate the function `compare` into MARS assembly language. Follow the usual calling conventions from lectures and labs, and use only instructions from the Midterm Instruction Subset described on the *Reference Material* page.

```
int compare(const char *a,
            const char *b)
{
  int result, ca, cb;
  result = 0;
  do {
    ca = *a;
    cb = *b;
    if (ca < cb) {
      result = -1;
      break;
    }
    else if (ca > cb) {
      result = 1;
      break;
    }
    a++;
    b++;
  } while (ca != '\0');
  return result;
}
```

PROBLEM 2 (*total of 15 marks*). Questions on integer arithmetic and logical instructions.

Part a. *(4 marks.)* Suppose `$t0` contains `0x7fff_a000`, and the MIPS instruction `addi $t1, $t0, 0x7000` is attempted. Determine the 32-bit result that will be produced by the adder that handles this instruction. Show your work, and use hexadecimal notation for your answer.

Part b. *(1 mark.)* Will the attempted `addi` instruction of **part a** update the `$t1` register? Give a precise reason for your answer.

Part c. *(4 marks.)* Suppose `$s0` contains `0x7000_0000`, `$s1` contains `0xdfff_fff6`, and the MIPS instruction `subu $s2, $s0, $s1` is attempted. Showing your work, determine the value that `$s2` will receive. Use hexadecimal notation for your answer.

Part d. *(1 mark.)* Did signed overflow occur in the subtraction of **part c**? Give a precise reason for your answer.

Part e. *(1 mark.)* Did unsigned overflow occur in the subtraction of **part c**? Give a precise reason for your answer.

Part f. *(4 marks.)* Write a sequence of MARS instructions (*not* a complete procedure) to modify the value in the `$s0` register as follows: bits 25–21 should be changed to 00000; bits 10–6 should be changed to 11111; all other bits should not be changed. (Reminder: Hexadecimal constants are allowed in MARS instructions.)

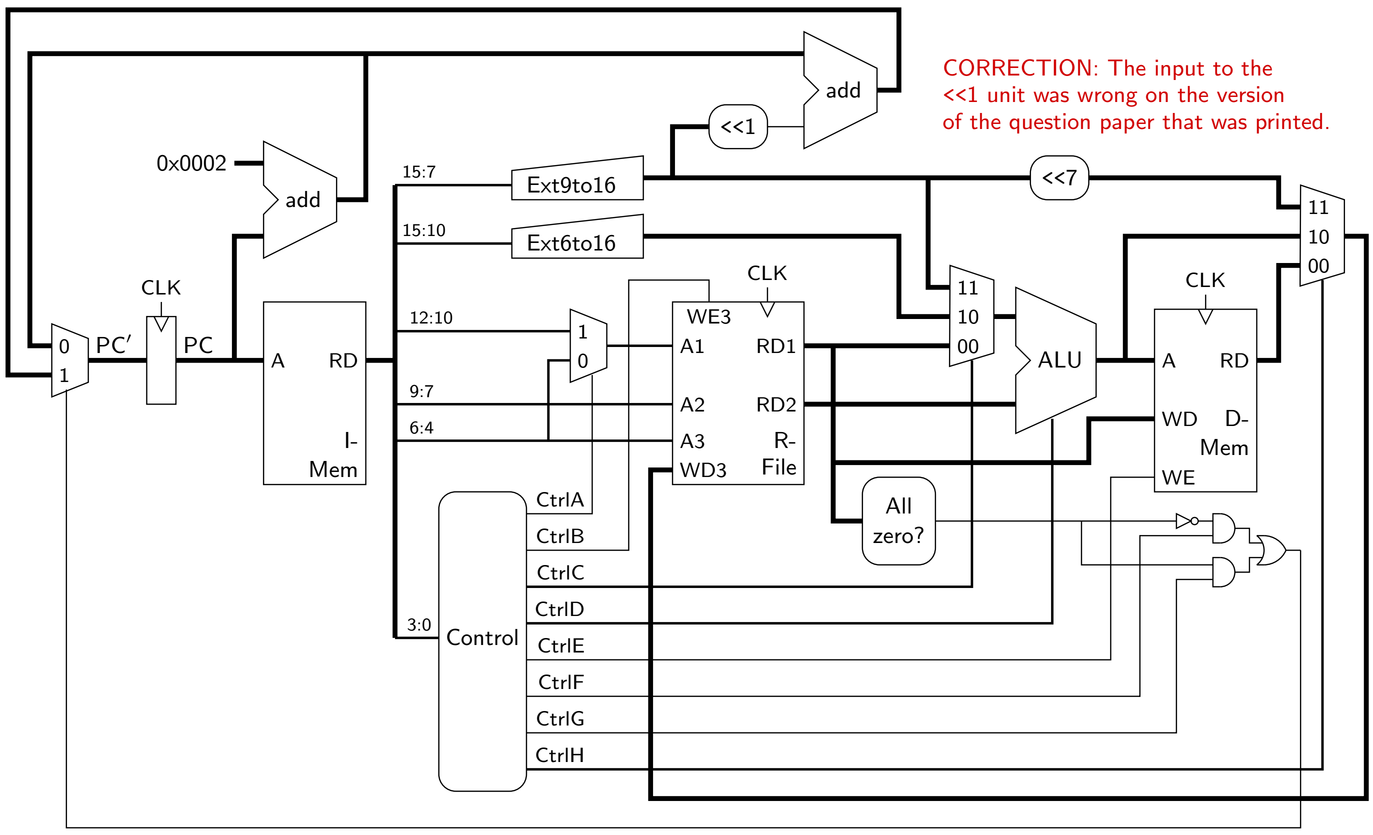
PROBLEM 3 *(10 marks)***.** The x1816 instruction set architecture (ISA) was designed specifically for this midterm test. In this ISA, there are 8 16-bit GPRs; GPR 0 is *not* special. Memory addresses, instructions, and data memory words are all 16 bits wide.

The table and notes below specify the machine code for the instruction set.

15 13	12 10	9 7	6 4	3 0	
	s s s	t t t	d d d	0 0 0 0	OR
	s s s	t t t	d d d	0 0 0 1	AND
	s s s	t t t	d d d	0 0 1 0	ADD
	s s s	t t t	d d d	0 0 1 1	SUB
	s s s	t t t	d d d	0 1 0 0	SLT
9-bit immediate			d d d	0 1 0 1	ADDI
9-bit immediate			d d d	0 1 1 0	LUI
9-bit offset			s s s	1 0 0 0	BEZ
9-bit offset			s s s	1 0 0 1	BNZ
6-bit offset		p p p	s s s	1 0 1 0	SW
6-bit offset		p p p	d d d	1 0 1 1	LW

d d d is the destination GPR number.
s s s is the first source GPR number.
t t t is the second GPR number.
In ADDI, the destination GPR is also a source GPR.
In LUI, the immediate operand is copied to bits 15–7 of the destination, and bits 6–0 of the destination will be zero.
BEZ means "branch if GPR equals zero" and BNZ means "branch if GPR is not equal to zero".
In SW and LW the memory address is the sum of the GPR given by p p p and the sign-extension of the 6-bit offset.

Below is a schematic for a single-cycle implementation of x1816. The unit labeled All zero? will have an output of 1 if all 16 of its input bits are 0; otherwise the output will be 0.



CORRECTION: The input to the <<1 unit was wrong on the version of the question paper that was printed.

Fill in the following table of control signals. Assume that the 3-bit ALU control input works the same way as the 3-bit control input for the ALU on page 4 of the Reference Material. To get full credit you must use X to indicate a don't-care bit whenever that is appropriate.

	CtrlA 1 bit	CtrlB 1 bit	CtrlC 2 bits	CtrlD 3 bits	CtrlE 1 bit	CtrlF 1 bit	CtrlG 1 bit	CtrlH 2 bits
OR								
AND								
ADD								
SUB								
SLT								
ADDI	ADDI can't be made to work with the given schematic.							
LUI								
BEZ								
BNZ								
SW								
LW								

PROBLEM 4 (*total of 7 marks*)

Below is the specification for the x1816 instruction set of Problem 3, repeated for convenience.

15 13	12 10	9 7	6 4	3 0	
	s s s	t t t	d d d	0 0 0 0	OR
	s s s	t t t	d d d	0 0 0 1	AND
	s s s	t t t	d d d	0 0 1 0	ADD
	s s s	t t t	d d d	0 0 1 1	SUB
	s s s	t t t	d d d	0 1 0 0	SLT
9-bit immediate			d d d	0 1 0 1	ADDI
9-bit immediate			d d d	0 1 1 0	LUI
9-bit offset			s s s	1 0 0 0	BEZ
9-bit offset			s s s	1 0 0 1	BNZ
6-bit offset		p p p	s s s	1 0 1 0	SW
6-bit offset		p p p	d d d	1 0 1 1	LW

d d d is the destination GPR number.
s s s is the first source GPR number.
t t t is the second GPR number.
In ADDI, the destination GPR is also a source GPR.
In LUI, the immediate operand is copied to bits 15–7 of the destination, and bits 6–0 of the destination will be zero.
BEZ means "branch if GPR equals zero" and BNZ means "branch if GPR is not equal to zero".
In SW and LW the memory address is the sum of the GPR given by p p p and the sign-extension of the 6-bit offset.

Assembly language for x1816 instructions has the same syntax as MARS assembly language, except that GPRs have numbers only, not names, and the ADDI, BEZ and BNZ instructions have only one GPR operand where MARS ADDI, BEQ and BNE instructions have two register operands. Here is an example two-instruction sequence to put the value `0x2a` into register `$0`:

```
lui     $0, 0
addi    $0, 0x2a
```

On an x1816 machine, the width of the `int` type is 16 bits.

For both parts of this problem use *only* the given instructions—you are not allowed to make up new instructions. (You are allowed to use ADDI even though that instruction does not work in the hardware of Problem 3.)

Part a. *(2 marks)* Suppose that `i` is of type `int`, in register `$4`. Translate the following C statement into x1816 assembly language.

```
i = 0x7bcd;
```

Part b. *(5 marks)* Assume the following types and register allocations:

name	type	register
a	int*	$3
n	int	$4
p	int*	$5
guard	int*	$6
count	int	$7

Translate the following sequence of C statements into x1816 assembly language. You may use registers `$0`, `$1` and `$2` for intermediate results.

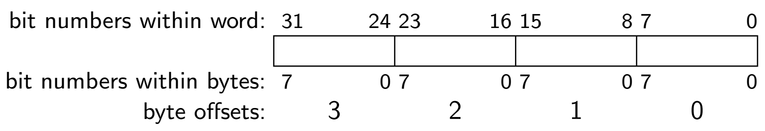
```
count = 0;
p = a;
guard = a + n;
do {
  if (*p > 0)
    count++;
  p++;
} while (p != guard);
```

PROBLEM 5 (*total of 8 marks*). Questions about miscellaneous topics.

Part a. *(2 marks.)* The single-cycle microarchitecture of textbook Figure 7.11 (which can be found on page 4 of the *Reference Material* booklet) processes exactly one instruction per clock cycle. Pipelined microarchitectures aim to start one instruction per clock cycle. Explain briefly why a pipelined microarchitecture is likely to be able to process many more instructions per second than the single-cycle microarchitecture.

Part b. *(2 marks.)* *Forwarding* is a technique used to solve data hazards in pipelined execution of instructions. Briefly explain what forwarding is, using a simple example with a pair of MIPS R-type instructions.

Part c. *(4 marks.)* The following diagram shows the organization of bytes within a MARS memory word:


bit numbers within word: 31 24 23 16 15 8 7 0
bit numbers within bytes: 7 0 7 0 7 0 7 0
byte offsets: 3 2 1 0

Find the values of the `$t0, $t1 and $t2` after the following MARS code runs. Express your answers in hexadecimal notation.

```
lui     $t9, 0x1001
lui     $t8, 0xbc67
addi    $t8, $t8, 0x0123
sw      $t8, 0($t9)
srl     $t7, $t8, 8
sw      $t7, 4($t9)
lb      $t0, 3($t9)
lb      $t1, 2($t9)
lbu     $t2, 6($t9)
```

MARKS: The space below will be used to record your marks for each question and your overall test mark. Please put your name, signature, and U of C ID number in the appropriate boxes.

Name (printed):

Signature:

U of Calgary ID number:

Lecture Section (01 is MWF 11:00am with S. Norman, 02 is MWF 10:00am with N. Bartley)**:**

Problem	Mark
1	/ 10
2	/ 15
3	/ 10
4	/ 7
5	/ 8
TOTAL	/ 50