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

Winter 2019 Section 01/02 Midterm Test #2 (corrected)
Wednesday, March 20 — 7:00pm to 8:30pm

Correction note: On the original version of this test, there was a small but significant mistake in the schematic of Problem 3. That has been corrected in this version.

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* page 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 *(8 marks).* Consider the C code listed to the right. Translate the function `upcase` 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.

```
void upcase(char *s)
{
  int i;
  i = 0;
  while (s[i] != '\0') {
    if (s[i] >= 97 && s[i] < 123)
      s[i] -= 32;
    i++;
  }
}
```

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

Part a. *(4 marks.)* Suppose `$t0` contains `0x8000_0020`, and the MIPS instruction `addi $t1, $t0, -0xc0` 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 `$s0` contains `0x4000_0000`, `$s1` contains `0x9fff_fff9`, 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.

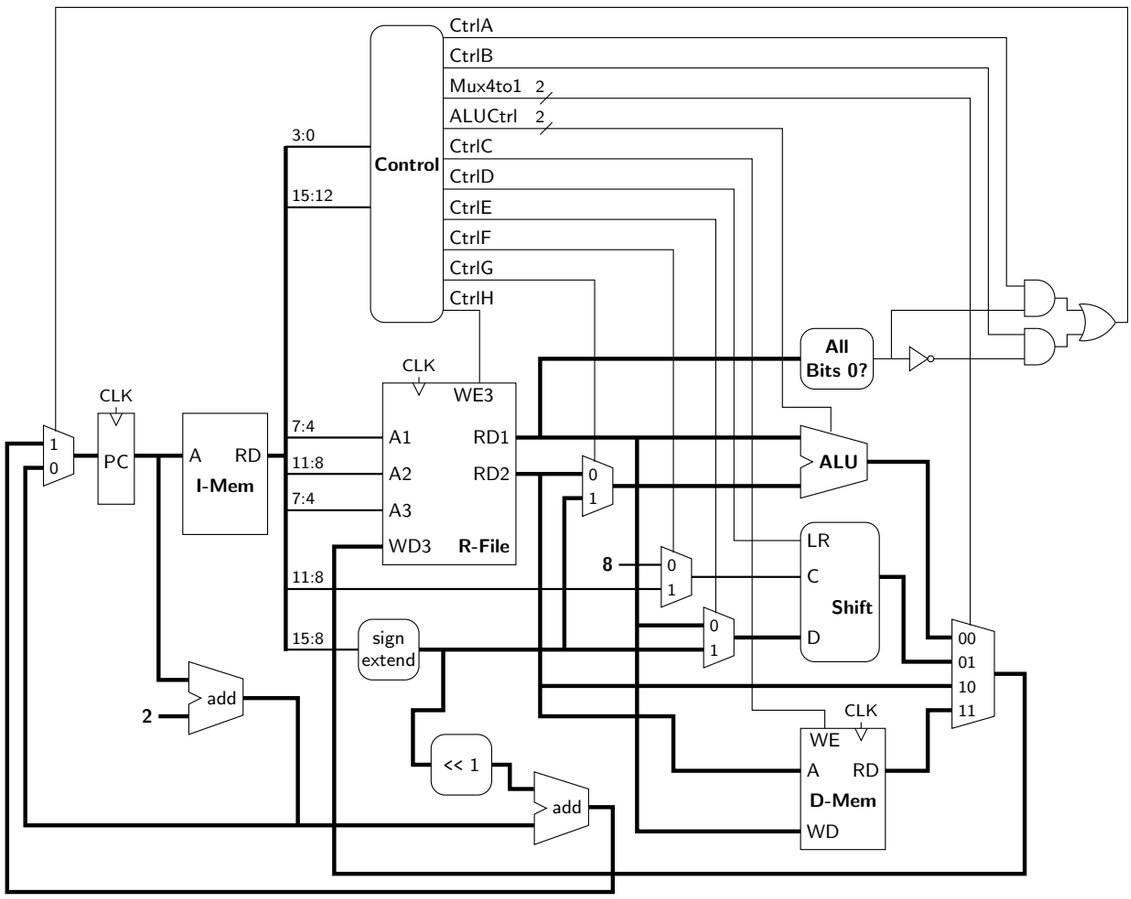
Part f. *(4 marks.)* Write a complete MARS translation of the following C function. For this part of this problem, the only instructions you may use are `sll`, `addu` and `jr`.

```
unsigned int times100(unsigned int x)
{
  return 100 * x;
}
```

PROBLEM 3 (*12 marks*). Below is the specification for an instruction set called MT219, in which addresses, memory words, GPRs and instructions are all 16 bits wide. There are 16 GPRs, $0 to $15; $0 always contains zero.

	machine code bits				
instruction	15–12	11–8	7–4	3–0	description
mov *rd*, *rs*	0000	*rs*	*rd*	0001	$rd = rs$
add *rd*, *rs*	0001	*rs*	*rd*	0001	$rd = rd + rs$
sub *rd*, *rs*	0010	*rs*	*rd*	0001	$rd = rd - rs$
slt *rd*, *rs*	0011	*rs*	*rd*	0001	$rd = (rd < rs)$
sll *rd*, *shamt*	0100	*shamt*	*rd*	0001	$rd = rd$ << *shamt*
srl *rd*, *shamt*	0101	*shamt*	*rd*	0001	$rd = rd$ >> *shamt*
lw *rd*, (*ra*)	0000	*ra*	*rd*	0010	$rd = \text{DMem}[ra]$
sw *rs*, (*ra*)	0001	*ra*	*rs*	0010	$\text{DMem}[ra] = rs$
addi *rd*, *imm*	*imm*		*rd*	0011	$rd = rd + imm$
lui *rd*, *imm*	*imm*		*rd*	0100	$rd = imm$ << 8
bez *rs*, *label*	*offset*		*rs*	0101	branch if *rs* == 0
bnz *rs*, *label*	*offset*		*rs*	0110	branch if *rs* != 0

And here is a single-cycle computer for MT219 *corrected so that RD1, not RD2, is supplied to to the multiplexer that chooses the D input of Shift*:



ALUCtrl: 00 to add, 01 to subtract, 10 for SLT.

Shift: LR is 1 for left, 0 for right. C is a 4-bit shift count. D is a 16-bit data input.

Fill in the following table of control signals. You must use X for don't-care in any cell where the value of the control signal does not matter.

instruction	CtrlA	CtrlB	Mux4to1	ALUCtrl	CtrlC	CtrlD	CtrlE	CtrlF	CtrlG	CtrlH
mov										
add										
sub										
slt										
sll										
srl										
lw										
sw										
addi										
lui										
bez										
bnz										

PROBLEM 4 (*8 marks*). The specification of the MT219 instruction set from Problem 3 is repeated below. Addresses, memory words, GPRs and instructions are all 16 bits wide. There are 16 GPRs, `$0` to `$15`; `$0` always contains zero.

	machine code bits				
instruction	15–12	11–8	7–4	3–0	description
mov *rd*, *rs*	0000	*rs*	*rd*	0001	*rd* = *rs*
add *rd*, *rs*	0001	*rs*	*rd*	0001	*rd* = *rd* + *rs*
sub *rd*, *rs*	0010	*rs*	*rd*	0001	*rd* = *rd* − *rs*
slt *rd*, *rs*	0011	*rs*	*rd*	0001	*rd* = (*rd* < *rs*)
sll *rd*, *shamt*	0100	*shamt*	*rd*	0001	*rd* = *rd* << *shamt*
srl *rd*, *shamt*	0101	*shamt*	*rd*	0001	*rd* = *rd* >> *shamt*
lw *rd*, (*ra*)	0000	*ra*	*rd*	0010	*rd* = DMem[*ra*]
sw *rs*, (*ra*)	0001	*ra*	*rs*	0010	DMem[*ra*] = *rs*
addi *rd*, *imm*	*imm*		*rd*	0011	*rd* = *rd* + *imm*
lui *rd*, *imm*	*imm*		*rd*	0100	*rd* = *imm* << 8
bez *rs*, *label*	*offset*		*rs*	0101	branch if *rs* == 0
bnz *rs*, *label*	*offset*		*rs*	0110	branch if *rs* != 0

Suppose that `i` is of type `int` in GPR `$2` and `a` is of type `int*` in GPR $3. Translate the following C code fragment (*not* a complete function definition) into MT219 assembly language, using *only* the instructions in the above table. You may use any or all of GPRs `$10`–`$15` for intermediate results. Note that the C `int` type for MT219 is a 16-bit two's complement type.

Warning: You will find that the sequence of instructions needed will be longer and more awkward than MARS assembly language would be.

```
for (i = 0; i < 0x45c; i++)
  a[i] *= 8;
```

PROBLEM 5 (*total of 4 marks*). Questions about miscellaneous topics.

Part a *(2 marks).* Give a brief definition of the term *control hazard* as it applies to pipelined processor microarchitecture. You may provide an example if you like.

Part b *(2 marks).* Explain briefly why neither a simple array of SRAM nor a simple array of DRAM is suitable for a laptop computer in 2019.

PROBLEM 6 (*total of 6 marks*). This problem refers to the **pipelined** computer of **Figure 7.47** of your course textbook. A copy of the figure is printed on page 4 of the *Reference Material* booklet.

Suppose the clock period is 0.5 ns and $t = 40.0$ ns marks the start of the Fetch phase for the `lw` instruction in the following sequence:

```
address      machine code   disassembly
0x0040_0140  0x0000_0000         nop
0x0040_0144  0x0000_0000         nop
0x0040_0148  0x8d09_0010         lw    $9, 16($8)
0x0040_014c  0x010a_4022         sub   $8, $8, $10
0x0040_0150  0x0169_5820         add   $11, $11, $9
```

At time $t = 40.0$ ns,

```
 $8 = 0x1001_0080    $9 = 0x0000_0007
$10 = 0x0000_0020   $11 = 0x0000_000c
```

Part a. *(1 mark.)* What value does PCPlus4F take on in the clock cycle between $t = 40.0$ ns and $t = 40.5$ ns? Answer in hexadecimal.

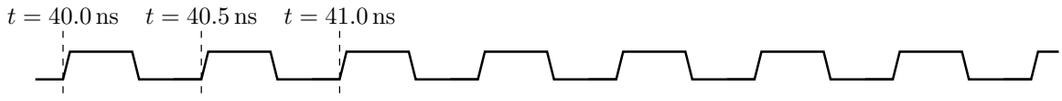
Part b. *(1 mark.)* What value does InstrD take on in the clock cycle between $t = 41.0$ ns and $t = 41.5$ ns? Answer in hexadecimal.

Part c. *(1 mark.)* What value does ALUOutM take on in the clock cycle between $t = 42.0$ ns and $t = 42.5$ ns? Answer in hexadecimal.

Part d. *(2 marks.)* What value does ResultW—see the label near the lower right corner of the schematic— take on in the clock cycle between $t = 42.0$ ns and $t = 42.5$ ns? If you cannot give a number, be as precise as you can be in describing where the value would come from.

Part e. *(1 mark.)* Will the `add` instruction use the correct value of `$9`? Why or why not?

The clock signal and blank space below may be helpful for rough work.



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

Name (printed): _____

Signature: _____

UCID number: _____

Lecture section: _____

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

Problem	Mark
1	/ 8
2	/ 15
3	/ 12
4	/ 8
5	/ 4
6	/ 6
TOTAL	/ 53