University of Toronto
Faculty of Applied Science and Engineering

Midterm Test
March 2020

ECE243 – Computer Organization

Examiners – Prof. Stephen Brown and Prof. Jonathan Rose

Print:

First Name _____ Last Name _____

Student Number _____

1. There are **5** questions and **20** pages. Do **all** questions. The duration of the test is 2 hours.
2. **ALL WORK IS TO BE DONE ON THESE SHEETS.** You can use the blank pages included on Pages $17 - 20$ if you need more space for any question. Be sure to indicate clearly if your work continues elsewhere.
3. Closed book. An **Aid Sheet** is included for your reference starting on Page 15.
4. No calculators are permitted.

1 [20]	
2 [10]	
3 [12]	
4 [10]	
5 [12]	
Total [64]	

[20 marks] 1. Short answers:

[2 marks] (a) For the ARM processor:

i. For the `BL` instruction shown in the code below, indicate which register(s) are changed during its execution, if any, and give the value of those register(s) after the instruction is executed.

```
Address      Instruction
-------      -----------
0x0:          BL  0x100
0x4:          ... (code not shown)
.
.
.
0x100:        ... (code not shown)
```

Answer:

ii. What instruction is used to return from a subroutine?

Answer:

ii. ~~In as few words as possible, explain what the assembly language program on the previous page would "do" if you executed it on the processor system that you created for Lab 2. In other words what would you observe on the DE1-SoC board while the program is executing? Note that device addresses for the Lab 2 processor are given on the Aid Sheet.~~

~~Answer:~~

[2 marks] (f) Consider the ARM assembly-language code shown below:

```
                .text
                .global _start
_start:
                MOV     R2, #LEDR_ADDR
                LDR     R2, [R2]
                MOV     R0, #DATA_WORD
                LDRB    R0, [R0]
                MOV     R1, #1
                CMP     R0, #0x10
                BEQ     DONE
                MOV     R1, #0x200
DONE:           STR     R1, [R2]
END:            B       END

LEDR_ADDR:    .word   0xFF200000
DATA_WORD:    .word   0x40302010
                .end
```

In as few words as possible, state what you would observe on the DE1-SoC board if you ran this program. Explain your answer.

Answer:

[10 marks] 4. An ARM program is shown below. It is supposed to be able to display on *HEX3-0* any four-letter word composed of the first eight (upper-case) letters in the alphabet. The word to be displayed is specified by using the `.ASCII` directive shown in the code. This directive produces one byte in the ASCII code for each letter given. The ASCII codes needed for this program are `A` = `0x41`, `B` = `0x42`, ..., `H` = `0x48`. The program includes appropriate patterns, starting at the label *SEG7*, that are used to render each letter on a 7-segment display. Note that if a letter is included in the `.ASCII` directive that is beyond the letter `H`, then the program displays a '–' for that *illegal* letter.

```
1          .text
2          .global _start
3  _start: MOV     R4, #STRING          // ASCII string
4          MOV     R5, #SEG7            // 7-segment codes
5          MOV     R6, #CODE            // used for results
6          ADD     R6, #3               // point to MSB
7          MOV     R0, #0               // letter counter
8  LOOP:   CMP     R0, #3
9          BEQ     DONE
10 CONT:   LDRB    R1, [R4, +R0]        // read a letter
11         SUB     R1, #41              // convert from ASCII
12         CMP     R1, #7               // valid?
13         BLE     OKAY
14         MOV     R1, #8               // show '-'
15 OKAY:   LDRB    R1, [R5, +R1]        // read 7-segment code
16         STRB    R1, [R6, -R0]        // HEX3-0 pattern
17         ADD     R0, #4               // ++ptr
18         B       CONT
19 DONE:   LDR     R1, [R6]             // get the result
20         LDR     R6, =0xFF200000      // I/O port base address
21         STR     R1, [R6, #0x20]      // write to display
22 END:    B       END
23
24 STRING: .ASCII   "FACE"
25 CODE:   .WORD   0
26 SEG7:   .byte   0b01110111  // 'A'
27         .byte   0b01111100  // 'b'
28         .byte   0b00111001  // 'C'
29         .byte   0b01011110  // 'd'
30         .byte   0b01111001  // 'E'
31         .byte   0b01110001  // 'F'
32         .byte   0b01111101  // 'G'
33         .byte   0b01110110  // 'H'
34         .byte   0b01000000  // '-'
35         .end
```

... continued on the next page

The program on the previous page contains a number of logical errors. All errors are within the code from Line 8 to Line 21. In the space below provide a corrected version of the code. You can either show all of the code between the Lines 8 and 21, or else show only the lines of code that you corrected. Either way, indicate clearly where you have made changes to the code, for example by encircling or underlining your corrections. Note: you should *not add any additional lines of code* to fix the errors; just correct the errors in the code that is there.

Answer:

[12 marks] 5. Consider the ARM program shown below. Answer parts (a), (b), and (c).

```
                .text
                .global  _start
_start:         MOV      R1, #DATA
                LDR      R1, [R1]
                MOV      R0, #0
                MOV      R2, #0
LABEL1:         CMP      R1, #0
                BEQ      LABEL2
                LSLS     R1, #1
                BCC      LABEL3
                ADD      R0, #1
                B        LABEL1
LABEL3:         CMP      R2, R0
                BGE      LABEL4
                MOV      R2, R0
LABEL4:         MOV      R0, #0
                B        LABEL1
LABEL2:         CMP      R2, R0
                BGE      END
                MOV      R2, R0
END:            B        END

DATA:          .word     0x3C303C30
               .end
```

[4 marks] (a) If this program is executed on the ARM processor, what values would be shown in a debugger. Show your answers in *hexadecimal*. for the registers below at the point when the processor reaches the label `END`. Note that for the left-shift instruction `LSLS` the bit that is shifted out goes into the carry (`C`) condition-code flag.

R0 ____ R1 ____ R2 ____

R15 ____

[3 marks] (b) What does this code "do"? That is, given the data at the label `DATA`, what does the program produce as a result?

Answer

ARM Addressing Modes

Name	Assembler syntax	Address generation
Offset:		
immediate offset	[Rn, #offset]	Address = Rn + offset
offset in Rm	[Rn, ±Rm, shift]	Address = Rn ± Rm shifted
Pre-indexed:		
immediate offset	[Rn, #offset]!	Address = Rn + offset; Rn ← address
offset in Rm	[Rn, ±Rm, shift]!	Address = Rn ± Rm shifted; Rn ← address
Post-indexed:		
immediate offset	[Rn], #offset	Address = Rn; Rn ← Rn + offset
offset in Rm	[Rn], ±Rm, shift	Address = Rn; Rn ← Rn ± Rm shifted

Examples of device addresses for the DE1-SoC Computer

```
Memory: 0x00000000
LEDR:   0xFF200000
HEX3-0: 0xFF200020
HEX5-4: 0xFF200030
SW:     0xFF200040
KEY:    0xFF200050
```