Computer Science

COMPSCI 210 S1

Assignment ONE

Due:	**11:59 pm Friday 10th April 2020**
Worth:	**6% of the final mark**
Late Submission	**30% penalty**

Introduction

This assignment is to be done using LC-3 simulator. You can download the JAVA version from Canvas.

You can use the simulator to compile and test the program.

Section 1: Running the Simulator [1] (0.5 marks)

You can execute the simulator ('LC3sim.jar'). We need to first load some software. The first piece of software we should load is, naturally, an operating system. The LC-3 operating system is very basic: it handles simple I/O operations and is responsible for starting other programs. Download the LC-3 OS ('LC3os.asm') and you can understand what the operating system does.

The LC-3 machine doesn't understand assembly directly; we first have to 'assemble' the assembly code into machine language (it is an '.obj' file containing binary data). The LC-3 simulator has a built-in assembler, accessible (as is the case for most of its functionality) via the Command Line text box. To assemble the operating system, type ***as lc3os.asm*** at the command line and hit enter. Make sure that the OS file is in the same directory as the '.jar' file; the as command also understands relative and absolute paths if the OS is in a different directory. Output from the assembly process is displayed in the CommandLine Output Pane. After assembling the OS, you should notice that 2 new files, 'lc3os.obj' and 'lc3os.sym', have been created. The '.obj' file is the machine language encoding of the assembly language file, and the '.sym' file is a text file that holds symbol information so the simulator can display your symbols. Recall that symbols are really just a convenience for silly humans; the machine language encoding knows only about offsets.

Now we can load the 'lc3os.obj' file into the simulator, either via the command ***load lc3os.obj*** or by going to the File menu and selecting Open '.obj' file. Notice that the contents of the memory change when the OS is loaded. Now assemble and load the solution file for Problem 0 (Q0.asm) into the simulator. The memory has changed again, but you may not notice since the relevant memory addresses (starting at x3000) aren't visible unless you've scrolled the screen. User-level programs (i.e., non-OS code) start, by convention, at x3000. If you type the command ***list x3000*** the memory view will jump to x3000 and you can see the 1-instruction solution to this problem.

To actually run code, you can use the 4 control buttons at the top of the simulator, or type commands into the command line interface (the command names are the same as the buttons). Note that the PC register is set to x0200, which is the entry point to the operating system by convention. You can set the value in the registers. Example: You can set the value of R2, either by double-clicking it in the Registers section, or via the command ***set R2 (value)***. Now, actually run the code by hitting the continue button. You can find more details of operations from [1].

In section 1, you are going to revise the program below. This program will take two input operands and output the AND results of those inputs. You first assemble all the files: 'lc3os.asm', 'data.asm' and 'Q0.asm'. Hence, you execute the following commands: ***load lc3os.obj, load data0.obj*** and ***load Q0.obj***. Click 'continue' to run the program. You can see the results from the display at the bottom-left.

```
…….
;
; A subroutine to AND the values from R2 and R3 (R2 AND R3). The result is saved at R3.
;
MyAND           AND             R3, R2, R3
                RET

;
; A subroutine to OR the values from R2 and R3 (R2 OR R3). The result is saved at R3.
;
MyOR
                RET

;
; A subroutine to add the values from R2 and R3 (R2 + R3). The result is saved at R3.
;
MyADD
                RET

;
; A subroutine to subtract the value of R3 from R2 (R2 - R3). The result is saved at R3.
;
MySUB
                RET

;
; A subroutine to multiply the value from R3 and R2 (R2 * R3). The result is saved at R3.
;
MyMULT
                RET

;
; A subroutine to calculate the value from R2 to the power of the value from R3 (R2 ^ R3). The result
is saved at R3.
;
MyPOW
                RET
…….
```

You now start revising the program of the sample file (Q0.asm) so the output will display the result of AND operations of every two input values (after the '&' character) from the "data0.asm". The first character of the data file is used to identify which the operation is going to be executed. You can save the program as the file Q1.asm. You are going to revise the highlighted area of the program to finish the whole assignment. The output of executing Q0 with data file data0 is shown below.

009&009=009
009&008=008
008&007=000
007&006=006
006&005=004
005&004=004
004&003=000
003&002=002
000&001=000
009&000=000
001&001=001

WARNING: We will use the JAVA simulator for marking. In particular, you should make sure that your answer will produce ONLY the exact output expected. The markers simply makes an exact comparison with the expected output. If you have any debug printouts or other code which produces some **unexpected output**, the markers will give you **zero marks**. If your files **cannot be compiled** successfully or they **cannot be executed** after compilation, the markers will also give you **zero marks**.

Section 2: OR Operation (0.5 marks)

You now complete the subroutine “MyOR“. It is a subroutine to “OR” the values from R2 and R3 (R2 OR R3). The result is saved at R3. You can check Lecture 5 and 6 for more information.

For example (data from ‘data1.asm’):
009|001=009
004|004=004
008|002=010
007|003=007
006|005=007
005|004=005
004|003=007
003|002=003
001|000=001
009|000=009
001|001=001

Section 3: Addition (0.5 marks)

You now complete the subroutine “MyADD“. It is a subroutine to add the values from R2 and R3 (R2 + R3). The result is saved at R3. You can check Lecture 5 and 6 for more information.

For example (data from ‘data2.asm’):
009+001=010
004+004=008
008+002=010
007+003=010
006+005=011
005+004=009
004+003=007
003+002=005
001+000=001
009+000=009
001+001=002

Section 4: Subtraction (0.5 marks)

You now complete the subroutine “MySUB“. It is a subroutine to subtract the value of R3 from R2 (R2 - R3). The result is saved at R3. You can check Lecture 5 and 6 for more information.

For example (data from ‘data3.asm’):
009-001=008
004-004=000
008-002=006
007-003=004
006-005=001
005-004=001
004-003=001
003-002=001
001-000=001
009-000=009
001-001=000

Section 5: Multiplication (2 marks)

You now complete the subroutine "MyMULT". It is a subroutine to multiply the value from R3 and R2 (R2 * R3). The result is saved at R3. You can check Lecture 12 for more information.

For example (data from 'data4.asm'):
009*001=009
004*004=016
008*002=016
007*003=021
006*005=030
005*004=020
004*003=012
003*002=006
001*000=000
009*000=000
001*001=001

Section 5: Power Function (2 marks)

You now complete the subroutine "MyPOW". It is a subroutine to calculate the value from R2 to the power of the value from R3 (R2 ^ R3). The result is saved at R3. You can check Lecture 13 for more information.

For example (data from 'data5.asm'):
009^001=009
004^004=256
008^002=064
007^003=343
006^003=216
005^004=625
004^003=064
003^002=009
001^000=001
009^000=001
001^001=001

Remarks:
1. All input value should be between 000_{10} – 009_{10}.
2. The results of all output should be between 000_{10} – 999_{10}.
3. All inputs and outputs should be positive.
4. There should not be any invalid inputs from the input data file.

Submission

You may electronically submit your assignment through the Web Dropbox (https://adb.auckland.ac.nz/) at any time from the first submission date up until the final date. You can make more than one submission. However, every submission that you make replaces your previous submission. Submit ALL your files in every submission. Only your very latest submission will be marked. Please double check that you have included all the files required to run your program.

No marks will be awarded if your program does not compile and run. You are to electronically submit all the following files:

1. **Q1.asm**

There will be 30% penalty on late submission. The period of late submission will be 2 weeks after the deadline. No more submission will be allowed after that period.

Integrity

Any work you submit must be your work and your work alone. To share assignment solutions and source code is not permitted under our academic integrity policy. Violation of this will result in your assignment submission attracting no marks, and you will face disciplinary actions in addition.

Reference

[1] http://www.cis.upenn.edu/~milom/cse240-Fall05/handouts/lc3guide.html