Computer Science

COMPSCI 210 S1

Assignment ONE

Due:	**11:59 pm Friday 26th April 2019**
Worth:	**8% 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] (1 mark)

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 data.obj*** and ***load Q0.obj***. Click 'continue' to run the program. You can see the results from the display at the bottom-left.

!!Have change lc3os.asm to data0.asm!!

```
;
; Initialization
;
                .ORIG   x3000
                LD      R6, EMPTY       ; R6 is the stack pointer
                LD      R5, PTR         ; R5 is pointer to characters
                AND     R0, R0, #0
                ADD     R0, R0, #10     ; Print a new line
                OUT
;
REDO            LDR     R3, R5, #0      ; R3 gets character
;
; Test character for end of file
;
                ADD     R4, R3, #-10    ; Test for end of line (ASCII xA)
                BRz     EXIT            ; If done, quit
                LD      R4, ZERO
                ADD     R3, R3, R4      ; Get the decimal value from ASCII
                JSR     CONV
                ADD     R5, R5, #1
                AND     R4, R5, #1      ; check odd/even
                BRz     EVEN
                ADD     R2, R3, #0      ; Save the first operand to R2
                LD      R0, PLUS        ; '+'
                OUT
                BRnzp   REDO
EVEN            LD      R0, EQUAL       ; '='
                OUT
; Start calculation
                AND     R3, R2, R3      ; The second operand is at R3
;
                JSR     CONV
                AND     R0, R0,  #0
                ADD     R0, R0,  #10    ; Print a new line
                OUT
                BRnzp   REDO
;
; A subroutine to output a 3-digit decimal result.
;
CONV            …….
……..……..
EXIT            HALT                            ; Halt machine
```

R5,R6 are been used in order to collect the character from the data file

convert the result into decimal value.

check whether is the first/second operand

for wrong result: 009+008=008 is do own code

009:1001,008:1000
using AND is 1000 = 008

You now revise the program of the sample file (Q0.asm) so the output will display the addition result (sum) of every two input values from the "data.asm". Save the program as the file Q1.asm.

009+008=017
008+007=015
007+006=013
006+005=011
005+004=009
004+003=007
003+002=005
002+001=003

first operand, symbol(+/-...), second operand, symbol(=), result

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**.

Just change AND operation to ADD!!

!!!The end result should be in R3.
can only use R0,R1,R2,R3,R4

Section 2: Subtraction (2 marks)

You now revise the program of the sample file (Q0.asm) so the output will display the correction equation and the subtraction result of every two input values from the "data.asm". Save the program as the file Q2.asm.

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

change symbol to "-", since there is not direct "-" command, so need to use two compliment, do NOT operation and addition operation and save it to the result.

modify a few lines

Section 3: Multiplication (2 marks)

You now revise the program of the sample file (Q0.asm) so the output will display the correction equation and the multiplication result of every two input values from the "data.asm". Save the program as the file Q3.asm.

For example (data from 'data.asm'):
009*008=072
008*007=056
007*006=042
006*005=030
005*004=020
004*003=012
003*002=006
002*001=002

Section 4: Factorial (3 marks)

You now revise the program of the sample file (Q0.asm) so the output will display the correction equation and the factorial result of every input value from the "data1.asm". Save the program as the file Q4.asm.

For example (data from 'data1.asm'):
006!=720
005!=120
004!=024
003!=006
002!=002
001!=001

006!=001*002*003*004*005*006

only display 3 digits only

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.

need to modify the code to delete some codes to only use one operand, and do not need to check the operand and directly jump to the result

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** for section 1
2. **Q2.asm** for section 2
3. **Q3.asm** for section 3
4. **Q4.asm** for section 4

need to submit all four files!!

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