**COMP1411 (Spring 2020) Introduction to Computer Systems**

Individual Assignment 2: Programming Due Date: 10:00am, 4th May, 2020

**Notes:**

* Please submit your assignment in Blackboard.
* Please follow the naming convention as stated in Section 3.

# Section 1. Description

In this assignment, you need to implement a simulator for assembly program based on a modified version of the **Y86-64 architecture**. *For details about Y86-64, please refer to the instructions in lecture 6 and chapter 4.1 in the textbook.*

For simplicity, we use the following setting:

* We use the following registers: %r0, %r1, %r2, %r3, %r4, %r5, %r6, %r7.
* Each register can store a 8-bit signed integer (instead of 64-bit). The value range is -128..127.
* The main memory address is an integer in the range 0..15.

We only consider the following instructions in this assignment and have the following remarks.

* rA and rB denote registers.
* Dest is an address label.
* V is a constant in decimal format (e.g., $2).
* Only the instruction cmpq can update condition codes.
* The notation D(rB) refers to the memory location at the address rB+D.

|  |  |  |
| --- | --- | --- |
| *Instruction* | *Operands* | *Meaning* |
| halt |  | stop execution |
| addq | rA, rB | compute rB+rA, then store the result in rB |
| subq | rA, rB | compute rB–rA, then store the result in rB |
| cmpq | rA, rB | compare rB and rA, then update condition codes |
| je | Dest | jump to Dest if the previous comparison result is rB==rA |
| jne | Dest | jump to Dest if the previous comparison result is rB≠rA |
| jg | Dest | jump to Dest if the previous comparison result is rB>rA |
| jl | Dest | jump to Dest if the previous comparison result is rB<rA |
| rrmovq | rA, rB | move rA to rB |
| irmovq | V, rB | move the constant V to rB |
| rmmovq | rA, D(rB) | move rA to memory (at the address rB+D) |
| mrmovq | D(rB), rA | move memory (at the address rB+D) to rA |

# Section 2. Sample input and output

Your simulator takes an assembly program as input, then simulates its execution, and finally prints the content of registers and the main memory to the screen.

Your simulator should initialize the content of registers and memory to 0 before executing an assembly program.

In the following, we show the sample input (in a file) and the sample output (on screen) of your simulator. Please follow the format of sample output (on screen) exactly.

*Sample input (in a file “case000.txt”)*

|  |
| --- |
| irmovq $-5, %r0  irmovq $11, %r1 L1: addq %r0, %r1  rmmovq %r0, 0(%r1)  halt |

Assume that there is an instruction in each line. It is optional to add an address label (e.g., L0, L1, L2, L3, …) in front of an instruction. Assume that the address labels at different lines must be different.

*Sample output (on screen)*

|  |
| --- |
| %r0..7: -5 6 0 0 0 0 0 0 M0..15: 0 0 0 0 0 0 -5 0 0 0 0 0 0 0 0 0 |

The first line shows the content of registers. The second line shows the content of the main memory. The values are displayed in the decimal format and separated by space.

# Section 3. Submission and grading

You shall submit a C program **SimXXX.c**

or

a C++ program **SimXXX.cpp** in Blackboard.

(*note that XXX denotes your student ID*)

Tutor will test your simulator by using 10 test cases.   
Each test case is worth of 10 marks.

Tutor will compile your simulator in our department’s apollo server as follows.

gcc SimXXX.c –o SimXXX

or

gcc SimXXX.cpp –o SimXXX

Tutor will run your simulator as follows, where “caseYYY.txt” is a text file (assembly program).

./SimXXX caseYYY.txt

The following assumptions are true for all test cases used for grading.

1. Each test case file has at most 100 instructions, i.e., 100 lines.
2. Each line has at most 128 characters.
3. Each test case contains at most 10 address labels. These address labels are L0, L1, …, L9. Different address labels are used for different lines.
4. The time limit of each test case is 1 second. If your simulator cannot print results within 1 second, then it is regarded as incorrect.
5. You must be careful about whitespaces (e.g., tabs, line break characters), commas, colons and others. We provide an extreme test case for this (case005.txt).
6. All contents (in input and output) are case sensitive.
7. All numbers are integers in the range -128..127.
8. All operations will be in the range of memory and registers.