Problem 1 – ILOC programming

Appendix A in our textbook (EaC) discusses ILOC, a linear assembly code for a simple abstract RISC machine. Here, you will also be able to use an additional instruction that allows you to print out a value: OUTPUT memory_address. This instruction allows you to print integer values from a memory location.

The target machine memory layout is as follows. Byte addresses smaller than 1024 are reserved and should not be used for program variables. Address 1020 is reserved as a location used for the output instruction. Program variables are addressed through offsets from address 1024. The reserved register $r_0$ needs to contain this value. Every program variable has the size of four bytes. Given this code shape, the simple example program

```c
a := 6;
b := 10;
PRINT a + b;
```

can be written in ILOC as

```c
//
// Simple example program
//
// Assign STATIC_AREA_ADDRESS to register "r0"
loadI 1024 => r0
// Store value 6 into variable a with @a = 0;
loadI 6 => r1
storeAI r1 => r0, 0
// Store value 10 into variable b with @b = 4;
loadI 10 => r2
storeAI r2 => r0, 4
// Compute a + b
loadAI r0, 0 => r3
loadAI r0, 4 => r4
```
add r3, r4 => r5

// Print the result using the fixed memory address 1020
loadI 1020 => r6
store r5 => r6
output 1020

Write (by hand) ILOC code that corresponds to the following high-level programs written in pseudo code.

1. factorial:

   a := 5; // input value; must be > 0 -- not tested
   result := 1;
   WHILE a > 1 DO
       result := result * a;
       a := a - 1;
   END WHILE
   PRINT result;

   Your code has to work correctly for changing values of variable “a”. Think of “a” as an input parameter.

2. Inner Summation:

   length := 100; // input value; must be > 0 -- not tested
   result := 0;
   FOR i := 1 TO length DO
       a[i] := i;
   END FOR
   FOR i := 1 TO length DO
       result := result + a[i];
   END FOR
   PRINT result;

   Your code has to work correctly for changing values of variable “length”. Think of “length” as an input parameter.
For each example program, give two versions: one that uses a register-register model where every value that can safely reside in a register is assigned a “new” virtual register, and one that uses a memory-memory model where all values reside in memory and are only loaded into registers just before they are used. Compare the number of instructions executed in both cases.

You can execute your ILOC code using the ILOC simulator sim. You can execute your ILOC code in file “test.i” by saying “./sim < test.i”. The simulator is available on the ilab cluster at (~zz124/cs415_2014/ILOCsimulator).

We will use the ilab cluster for our programming projects. The “http://ilab.rutgers.edu” ilab cluster page contains the listing of valid hostnames available for this homework. Be aware that ilab.rutgers.edu, pasta.rutgers.edu and soup.rutgers.edu are NOT to be used. For instance, you can use vi.rutgers.edu. If a hostname appears to be unresponsive (e.g., ssh claims host is not available), that host may have failed. Pick another hostname and try to login. If multiple hosts are unavailable, try again in 30 minutes. If more than 5 hosts are still down, post a message to Sakai. You have the same home directory across all machines of the ilab cluster.

Problem 2 – Feasible Registers for ILOC

What is the minimal number of physical registers to execute any ILOC instruction? What is the number of feasible registers in ILOC (minimal number of physical registers to execute any ILOC code, not instruction), i.e., if all the operands in the instruction need to be loaded from memory and the result needs to be written back to memory. Remember to take into account the memory address calculation instructions as illustrated in Problem 1. Give an argument and use examples to justify your answer.