University of Puerto Rico – Mayagüez Campus
School of Engineering

INEL 4206 – Microprocessors

Problem Set 2
Due: Tuesday October 29, 2002 Midnight

Using the instruction set discussed in class for the Easy I processor, you will provide a program that sorts an array of integers in ascending order (from the lowest to the highest number). To achieve this, you'll use the Insertion Sort algorithm. An example program in C++ is provided to you in the problem set files, so basically what you have to do is translate the HLL program (C++ program) into instructions from the EasyI Instruction Set (assembly language). You are given the first part of assembly program which creates the array of integers out of order, and stores the length of the array. For convenience the index of the last element in the array is also stored in memory, if you wish to use it in your program.

General information on how the emulator (simulator) works.

Memory addresses start at 500 (in this case up to 699)

500-599	Defined variables (local vars *declared vars in HLL*)
600-699	Heap Memory (Dynamic Memory i.e. arrays)

Instructions are referenced by their line number starting in line 1. Thus the PC will start pointing to memory space 1 where the first instruction will be. Lines that only contain a comment are counted but are evaluated as a no operation instruction, so when doing jumps and branches the line numbers of lines with comments only are not ignored. For convenience reasons, the instruction memory holds a complete instruction per cell, so unlike the example in the class here, the PC is incremented only by one (instead of two) to go to the next instruction.

Comments are allowed in both the assembly and the machine code for this assembler/emulator. Comments are preceded by the pound sign (#), anything after this will be ignored. Comments can be made after an instruction, or on a line by themselves.

You are given two programs, one that is an assembler and one that is the EasyI emulator. The assembler will take your assembly program and convert it to machine code (EasyI machine code) and save the output on a file with a filename specified by you during execution. The emulator will read the machine code file, and process the instructions, at the end of the processing it will show you the status of the accumulator and any memory location that has been used during the execution of the program. Both programs are made in Java and are packaged in a jar file. To make it easier for you, two

scripts have been included in the files provided. You will use the scripts to execute the program. Here are examples of how to run the assembler and the emulator.

./asm prob2.asm ✍ this will then ask me the output file name I want and convert prob2.asm to machine code

./emu prob2.mc ✍ this will execute the instructions contained in the machine code file named prob2.mc

Problem Set specific details

- Array length is stored in memory location 500
- Index of last element in array is stored in memory location 502
- Array starts at memory location 600, to maintain the idea of byte addressable memory, each cell is accessed by 2*index + base address, so the second cell (index 1) is at 602 not 601.

You should use the memory space from 504 to 599 for your local variables, since the program will be tested with other arrays, different than the one given to you in the ps2.asm file. (Your program should work with arrays of different lengths and contents as long as they are integer arrays). For example if you use memory location 620 for a local variable, and one of the tests cases has an array that uses that memory location as a cell, your program will not work correctly.

Valid instructions for the assembly language of the EasyI (make reference to class slides for their functionality)

add & addi
and & andi
comp
shr
jump & jumpi
brn & brni
load & loadi
store & storei

A new instruction used for this problem set: **end**

- This new instruction marks the end of your program, please use it, even though your program should work the same without it.

Note: Remember that the 'i' at the end of the instructions is immediate and is not the same as the I in the machine code which represents indirect instruction, in fact they are opposites. So

jump 0 would be ✍ 100011000000000
and
jumpi 0 would be ✍ 000011000000000

For the immediate versions of jump and branch, (jumpi and brni) labels can be used so you don't have to mess with line numbers. A label will mark the place where to jump to.

ex.

```
line 1:    looplabel:              #jump here
line 2:                andi  0
line 3:                jumpi  looplabel
```

would be equivalent to:

```
line 1:    #jump here
line 2:                andi  0
line 3:                jumpi  1
```

FILES

File you receive: ***ps2pkg.tar.gz***

This is a compressed file, which you have to decompress using the command, "`tar –xzf ps2pkg.tar.gz`" on the console of the Linux workstation. This will create a folder named ***ps2files*** and inside you will find:

ps2.pdf	✍ This file (or future versions)
ps2.jar	✍ Actual java programs packaged inside here
asm	✍ Script to execute the assembler
emu	✍ Script to execute the emulator
ps2.asm	✍ Assembly program file with the instructions to create the array, and store length and last index, where you will make your program
submit	✍ Script that submits your answer (which should be named ps2.asm and ps2.mc)
prog.cc	✍ Insertion sort program in C++

Files you submit:

ps2.asm ✍ The assembly file with your program in it
ps2.mc ✍ The machine code file with your program

This files will be submitted using the submit script, by just running the command "./submit" inside the p2files folder. You must use the account provided for this class to submit the files since they will be identified by the username (account name).

References:

Insertion Sort explanation: http://www.sparknotes.com/cs/sorting/insert/

You can find a visual example of how Insertion Sort works at:
http://www.cs.ust.hk/faculty/tcpong/cs102/summer96/aids/insert.html

Correction criterias:

Criteria	Weight(%)
Correctness	60%
Design	20%
Efficiency	10%
Style	10%

Remember the policy on late submission stated on the "Course Information Sheet".