

University of Puerto Rico – Mayagüez Campus
School of Engineering

INEL 4206 – Microprocessors

Problem Set 2

Due: Tuesday, March 18, 2003

Using the instruction set discussed in class for the Easy I processor, you will provide Assembly programs as solutions to the various parts of this problem set. You are given an Assembler and an Simulator with the Problem Set package file, read the instructions on how to use these programs in the following pages. You are also given a “template file” with some code, you must edit after the line that says “#Start your code here”, don’t edit this line or any line above, except in the space provided for name, student #, and Linux lab account number.

A)

Provide an Easy I Assembly program that determines the remainder for the division of two numbers. The file provided (ps2a.asm) has the locations for the dividend, divisor and remainder. Your result must be in the memory space assigned in the code for the remainder (memory location 504).

Variable	Location	Initial Value
Dividend	500	9
Divisor	502	4
Remainder	504	0

B)

Provide an Easy I Assembly program that determines the Fibonacci Number Sequence for a specified Amount of terms. 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 “allocates” space for the array of integers where you will store your result. The Amount of terms is given in the code (memory location 500). 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.

C)

You are given a set of instructions encoded into Hexadecimal digits (4-digits represent each instruction). Starting on memory location 600 the program creates an array and after execution it contains a message using the ASCII code (you can find out the code in decimal and hexadecimal notation at www.asciitable.com). The task for this problem is to determine what the message is. The message is first put into memory encoded and then decoded by the program and placed in the same location in memory. You must execute the program either manually or by changing into machine code and using the simulator, either way you must show your work and explain your steps, describe method used for encoding/decoding the message, plus provide the message. The file containing the hex-codes is called ps2-c.hex.

General information on how the simulator works.

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

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

If you use a immediate jump instruction (jumpi / brni) you may use labels and reference them in the instruction, if you are using an indirect method (jump / brn) you must use the line number of the instruction you want to jump to (starting with 1 not 0).

Comments are allowed in both the assembly and the machine code for this assembler/simulator. 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 simulator. 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 simulator 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 simulator.

`./asm ps2a.a` → this will then ask me the output file name I want and convert ps2a.a to machine code

`./sim ps2a.mc` → this will execute the instructions contained in the machine code file named ps2a.mc

Problem Set specific details

- Array length is stored in memory location 500
- Base address of the array is stored in memory location 502
- Index of last element in array is stored in memory location 504
- Array starts at memory location 620, to maintain the idea of byte addressable memory, each cell is accessed by $2 * \text{index} + \text{base address}$, so the second cell (index 1) is at 622 not 621.

You should use the memory space from 506 to 599 for your local variables, since the program will be tested with different number of terms, therefore the memory array will vary from test to test. 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.

Make sure you have no blank (empty) lines between code lines or at the end of the file, as they are not handled by the assembler/simulator.

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 \rightarrow 1000110000000000
and
jumpi 0 would be \rightarrow 0000110000000000

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
sim	→ Script to execute the simulator
ps2a.a	→ Assembly program file template with variable location for problem A
ps2b.a	→ Assembly program file with initial variables and array allocation scheme (algorithm)
ps2c.hex	→ File containing the hexadecimal code for the program in part C
submit	→ Script that submits your answer (which should be named ps2.asm and ps2.mc)
ps2b.cc	→ Fibonacci Sequence Program in C++

Files you submit:

ps2a.a	→ The assembly file with program from problem A
ps2a.mc	→ The machine code file with your program from problem A
ps2b.a	→ The assembly file with program from problem B
ps2b.mc	→ The machine code file with your program from problem B

Answer to problem C, will be handed-in (in paper) to the professor on the due date.

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:

The Fibonacci Numbers : <http://math.holycross.edu/~davids/fibonacci/fibonacci.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”.