Lab4: Software Organization

The Assembly Language Level

In this lab you will develop and run some SPARC assembly language programs.

Getting to Work

Our laptop computers have Intel processors, so we will be using menehune to assemble and run our programs. The first step in this lab is to ensure that you can log in, and work from (one or more) terminal window(s).

To connect to Menehune (Windows)

- Click on the Windows Start logo in the lower left of your screen
- A search box will pop up. Search for "putty".
- Click on the top option found "PuTTY". The PuTTY software should start and open a window.
- Enter the hostname: `menehune.opt.wfu.edu`
- Select connection type: `ssh`
- If you would like better colors, or larger font, use the menu on the left. You can also name your session and save your choices.
- Click on `open` to start your session. Log in using your user name and password on menehune.

For each of the lab problems, I have created a sub-directory (folder) for each. Your terminal session has the concept of the **current working directory**. Directories are organized in a tree structure, (similar in concept to a family tree). There are several UNIX commands that help you navigate the directories (folders).

pwd – tells you the current working directory

ls – list the files and sub-directories in the current directory

cd – Use the **cd** command to change the current working directory to a sub-directory. For example, if you have a sub-directory named "Prob1", use the command **cd Prob1** to change to that directory. Use the command: **cd ..** to change to the parent directory.

mv – Use the **mv** command to change the name of a file. For example **mv a.out demo0** changes the name of a the file "a.out" to "demo0".

To create and edit a file on menehune

- Type **pico f.s** to create and edit a file named **f.s**.

Some Naming Conventions The part of a file name that comes after the dot, is known as the file name extension. The file name extension gives software a clue (and reminds us) of the type of information stored in that file. Here are some common file name extensions.

- **.html** – indicates a web page
- **.doc** – indicates a Microsoft word document
- **.xls** – indicates an Excel spreadsheet document
- **.c** – indicates C-language source code
- **.s** – indicates Assembly-language source code
- **.o** – indicates an object file. This file contains code format that can be combined with other **.o** files to create an executable file.

Another useful name to remember is **a.out**. This is the default name of a file of executable machine code when it is created by the compiler (or assembler). To start that program running, type "a.out" and press the Enter key.

Problem 0

This problem is to walk you through the process of creating a file containing assembly language instructions and how to get it running. To get yourself started, change directory to the directory named "Prob0". Type "ls" and press the Enter key. You should see a file named "prog0.o"

Your task(s):

1. Use the pico editor to create a file named **f.s**. Copy the function from example 2 in the handout, and save the file.
2. Assemble your source code into an object code file using the command **gcc -c f.s**
3. If you get error messages, use **pico** to correct the file. If you get no error messages, type **ls** and you should see an object file listed named **f.o**.
4. Link the main program (provided for you) in the file **prog0.o** to your newly created object file **f.o**. To link the two files, Use the command **gcc prog0.o f.o** to link the two files into an executable. If you type **ls**, you should see a file named **a.out**
5. Run your newly linked program by typing **a.out** and pressing the Enter key. You should get the number 12 in the output.

Problem 1

In this problem we are ready to tackle our first assembly language function. Look for a directory named **Prob1** and change to that directory. You should see a file named **prog1.o**.

Your task(s):

1. Write an assembly language function with entry point named **g** which will accept three inputs in registers **%i0**, **%i1**, and **%i2**. Save your code in a file named **g.s** Compute the product (multiplication) of these three values and return the total via the usual return mechanism on SPARC systems. Add comments to your code to explain the role of each register you use.

2. Assemble your source code into an object code file using the command **gcc -c g.s**

3. Link the main program (provided for you) in the file **prog1.o** to your newly created object file **g.o**. To link the two files, Use the command **gcc prog1.o g.o** to link the two files into an executable. If you type **ls**, you should see a file named **a.out**

4. Run your newly linked program by typing **a.out** and pressing the Enter key. You should get the number 105 in the output.

Problem 2

In this problem we are ready to tackle our second assembly language function. Look for a directory named **Prob2** and change to that directory. You should see a file named **prog2.o**.

Your task(s):

1. Write an assembly language function with entry point named **max3** which will accept three numbers in registers **%i0**, **%i1**, and **%i2** and find the largest of these three values. Save your code in a file named **max3.s**. Return the largest of the three input values using the usual return method on SPARC systems. Add comments to your code to explain the role of each register you use.

2. Assemble your source code into an object code file using the command **gcc -c max3.s**

3. Link the main program (provided for you) in the file **prog2.o** to your newly created object file **max3.o**. To link the two files, Use the command **gcc prog2.o max3.o** to link the two files into an executable. If you type **ls**, you should see a file named **a.out**

4. Run your newly linked program by typing **a.out** and pressing the Enter key. You should get the number 7 in the output.

Problem 3

Look for a directory named **Prob3** and change to that directory. You should see a file named **prog3.o**.

Your task(s):

1. Write an assembly language function with entry point named **largest**, which accepts two input parameters in registers **%i0**, and **%i1**. Register **%i0** contains a number `n`. Register **%i1** contains the memory address of an array containing `n` numbers in it. Find the largest number in the array, and return it using the usual method for returning a number from a function. Save your program in a file named **largest.s**

2. Assemble your source code into an object code file using the command **gcc -c largest.s**

3. Link your object file **largest.o** to the provided main program using the command **gcc prog3.o largest.o**.

4. Run your newly linked program by typing **a.out** and pressing the Enter key. You should get the number 11 in the output.