CS 273 Lab 2: HC11 Assembler and Simulator

Dept. of Computer Science, NMSU

Fall 2006

Due Date: Next class after your lab session.

For the first part of this course, we will be writing assembly language programs for the 68HC11, but we won't be using an actual 68HC11 processor. How will we do that? Well, we have an HC11 assembler that runs on the Linux PCs, and better yet, we have a simulator that runs under Linux. The simulator is essentially a software copy of the HC11 processor. It takes the HC11 machine code and executes it just like a real HC11 processor would.

1 Before Running HC11 Assembler and Simulator

To use the assembler and simulator programs, you have to add the directory where they are located to your search `PATH`, so that your command shell will find them. To do this, add the following line to your `.cshrc` file in your home directory, and put the line near the bottom of the file:

```
source /home/CS273/cshrc.cs273
```

Be sure to hit ⟨`enter`⟩ and create an empty line below this one! Otherwise it will not work. *(If you do not know how to edit the file under Linux, you can choose an editor by the menu:*

Start → Utilities → Editor

or you can type an editor command in your terminal window. Popular editors are emacs/xemacs, kate, pico, and vi. If you are a beginner I would suggest kate or pico. Vi is extremely not-user-friendly!)

Adding this line to your `.cshrc` file will add the directory to your search path. To make this take effect, you can do one of several things: close your current terminal window and open a new one; log out and log back in; or type the command "`source .cshrc`" in your current window (you might also have to do "`rehash`").

Note that when you type "`ls`" or "`dir`" on your home directory, you do not see the `.cshrc` file. In Linux, files that begin with a dot are ignored by "`ls`" and treated as hidden files. If you want to see

them, use the command "`ls -a`".

Once you have done the above, the commands "`as11`" and "`sim11`" should be available to you. The command named "`as11`" is the assembler, and "`sim11`" is the simulator. You can read them as "*aye-ess-eleven*" and "*sim-eleven*". We are now ready to write and run some assembly programs.

2 Running HC11 Assembler and Simulator

First, copy the program

`/home/CS273/pub/test.asm`

to your own directory (you might want to create a class subdirectory from your home directory, so that you are better organized). This sample program just adds the value 5, the value in the constant `param1`, the value in the constant `param2`, and places the result into a memory location.

Now, assemble the program using the command

"`as11 test.asm`"

(Note: this is "*as-one-one test.asm*"). You should see some output saying that there were no warnings and no errors. If you do have some warnings or errors you probably did something wrong in copying the program file. After this step you should have a file called "`test.s19`" ("*test.ess-nineteen*"). This is the machine code file that the assembler created.

Now, type the command

"`as11 test.asm -l`"

Note that "`-l`" is a "*dash-ell*", not a "*dash-one*". The "`-l`" option tells the assembler to produce a listing of the program, which will be used frequently in this course. The listing is most useful, however, when it is in a file, so now do the command

"`as11 test.asm -l > test.lst`"

(Note: the list file is "*test.ell-ess-tee*", not "*test.one-ess-tee*"). This will assemble the program into machine code and save it in a file called "`test.s19`". It will also produce a listing file called "`test.lst`". (Note: in the command, the notation "`-l`" is "*dash-ell*", not "*dash-one*". Also, the "`>`" sign is a Linux shell operator that takes everything that the program would print to the screen and redirects into a file named by the given name after the "`>`" sign. You can do this with any Linux program.)

After this step you should have the files "`test.asm`", "`test.s19`" and "`test.lst`". Look at the listing file using the command

"`more test.lst`"

It should look like the one we saw in class.

Now run the program using the command

"`sim11 test.s19`"

The simulator will print out a screen that looks like the following:

```
A 00 B 00    t=0
D  0000
X  0000 Y  0000         breakpoints ->
SP 0000 PC F802         ldaa    #05
    00 (.... ....)

       0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
0000  aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa ................
0010  aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa ................
0020  aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa ................
0030  aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa ................
0040  aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa ................
0050  aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa ................
0060  aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa ................
0070  aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa ................
0080  aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa ................
0090  aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa ................
00a0  aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa ................
00b0  aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa ................
00c0  aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa ................
00d0  aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa ................
00e0  aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa ................
00f0  aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa ................
```

The above shows the status of the processor and its registers, and the RAM memory (the simulator puts the value "`aa`" in every byte of memory to start with). With the simulator, every time you hit ⟨`enter`⟩, the simulator executes one machine instruction, and displays a new processor and memory status. In this way you can watch the registers and memory change values as the program is executed. The

"`t=⟨val⟩`"

that is displayed on the first line (after registers A and B) is a count of the time that has elapsed during program execution (we will get to what this count means shortly). This counter is not a CPU register, it is just showing you how much time your program has taken to execute. This time count is displayed in decimal numbers.

The final instruction in the program just loops forever, so to quit the simulator, enter the command

"`q`"

or hit

"Ctrl-D"

Now run the program with a graphical front-end to the simulator by typing in the command "tksim11 test.s19".

Questions

1. What are the final values in the registers (in hex)? Note that the registers are the labels A, B, D, X, Y, SP, and PC at the top of the output. Their values are the two or four digit values next to the labels. They are hexadecimal values with digits 0-9, as well as digits A-F (or a-f).
2. What are the final values in memory locations $0000, $0001, and $0002 (in hex)? Note that these are the first three values in the first row of the memory matrix display. Initially, the simulator sets all the memory values to "aa" (which is a valid hexadecimal number!). This program probably changed some of them.
3. What is the final time count when this program ends? The time count is reported at the "t=" portion of the text output and in the "cycles" box in the GUI window of tksim11. The program ends before the "JMP end" instruction is executed. This instruction is just an infinite loop that halts the program. You should record the time count before this instruction executes even once (in other words, the first time you see the "jmp" instruction appear on the output). In the text mode ("sim11"), if you keep hitting ⟨enter⟩, this instruction will just keep executing, and the time count will keep going up, even though no more computation is occurring.

3 Lab Turn-in

The test assembly program that you ran, a completed lab report named "LastName-Lab2.???", with answers for the questions in the report. Submit both the source assembly (.asm) and listing (.lst) files for each program. Please write the lab report using the template available online at

http://www.cs.nmsu.edu/~joemsong/273/labtemplate.html (HTML version)

or

http://www.cs.nmsu.edu/~joemsong/273/labtemplate.pdf (PDF version)

Submit these through the web submission page at:

http://www.cs.nmsu.edu/~joemsong/273/upload/

Your initial password is the last 4 digits of the social security number.