Goals

• Modifying a C Program

• Creating a MIPS Program from a C Program

General Instructions

Paste all your programs and program output (including the `gcc` and execute commands that you used in the case of C programs) into a document (PDF or open office or plain text). Upload this document to Sakai and hand in a hard copy by the due date. *BE SURE YOU PUT YOUR NAME AND THE LAB NUMBER* at the top of the document!

Also upload the programs themselves as separate files. All programs must be FULLY COMMENTED — header comments should provide information about author, date, program purpose (input/output and summary of processing performed); in-code comments should describe variables and summarize major steps in the algorithm. For C programs, indent properly (imitate K&R). For MIPS programs, be as generous with comments as possible—they are mostly to help YOU, but they’ll help me, too.

In both C and MIPS programs, use white space, e.g., blank lines, to make the program more readable. (For instance, note how I grouped the various syscalls in the sample programs by inserting blank lines.)
Part 1: Modifying C for a New Purpose [15 points]

In Week 2’s Lab, you wrote a C program that plays the “Guess My Number” game. This week, you will modify that program to switch roles:

- YOU will choose the number.
- The COMPUTER will attempt to guess the number.
- On each iteration, YOU must inform the computer whether its guess is too high, too low, or correct.
- The COMPUTER will repeat your feedback (useful for debugging).
- When guessing correctly, the COMPUTER will report how many guesses it took, and then ask if you want it to guess again.

When giving feedback to the computer about its guess, you should use the following:

- 1 = The computer’s guess is too high.
- 0 = The computer’s guess is correct.
- -1 = The computer’s guess is too low.

Your input prompts and output labels should look similar to the ones in this sample run:

```
aldenv27:lab3 jwenskovitch$ gcc lab3part1.c -o lab3part1
aldenv27:lab3 jwenskovitch$ ./lab3part1
Your Name
Lab 3
Tue Sep 16 07:39:02 2014

Hello user! Please pick a number between 1 and 100.
My first guess is 50. How did I do?
1

Hmmm, my guess was too high. My next guess is 25. How did I do?
-1

OK, my guess was too low. My next guess is 37. How did I do?
1

Hmmm, my guess was too high. My next guess is 31. How did I do?
1

Hmmm, my guess was too high. My next guess is 28. How did I do?
```
Yay, I got it in 5 tries! Your number was 28.
Would you like me to guess again (y/n)? y

Hello user! Please pick a number between 1 and 100.
My first guess is 50. How did I do?
1

Hmmm, my guess was too high. My next guess is 25. How did I do?
0

Yay, I got it in 2 tries! Your number was 25.
Would you like me to guess again (y/n)? n

There are two ways that you can have the computer guess your number. In the first case, the computer can simply try every number between 1 and 100 in order. This will give you partial credit. In the second case, you can keep track of the largest and smallest number that it could possibly be, and always guess the number in the middle. This will give you full credit.

Note that I do not expect your program to detect when you are lying to the computer about your number. If you want to feel superior to the machine, be my guest, but make sure that your program will respond to “real” input as well.

Paste both the C program and several sample outputs into your report document. Include the Linux commands used to compile and execute the program, as well as all prompts and all input. (Just select text from the terminal window.) Also, upload the .C file to Sakai when you are ready to submit the lab.
Part 2: Convert Your Program to MIPS [15 points]

Write a MIPS assembly language program that does exactly what your C program (from part 1) does: play the “Guess My Number” game in reverse, with YOU picking the number and the COMPUTER guessing the number. Use the MARS simulator.

You will need to use various syscall command codes and make appropriate use of the $v0 and $a0 registers as illustrated in the sample programs we have completed together this semester.

For information about syscall codes, see the website link posted on the Schedule page of the course website. You can also read about them in the MARS documentation—select “Help”, select the “MIPS” tab, look for the tab labeled “Syscalls”.

You will need to create separate strings (".asciiz" directives) for each “piece” of the input prompts and output labels, a separate string for the terminating newline character at the end of each line, etc.

Run it a few times with different values. Copy the contents of the “Run I/O” window (showing your program output) and paste it, along with the text of your program, into the report document. Also, upload the .ASM file to Sakai when you are ready to submit the lab.