CS 341L Fall 2009 Lab 1

Assigned: Wednesday, 26 August 2009

Due: 10:00am, Monday 31 August 2009 by attachment as an email to labturnin.cs341l@gmail.com

You should attach your program with your last name and the lab number in the name of the file, e.g., “crandall-lab1.S”.

No late assignments will be accepted unless circumstances warrant an extension for the whole class (e.g., server is down all weekend, etc.). You must type your own Hello World program in a text editor, no copying and pasting or using a compiler is acceptable.

The purpose of this assignment is to make you familiar with the basic format of a MIPS program for the gcc framework in Linux, and make sure you are able to log into tomalesbay, assemble your labs, and perform all the basic tasks necessary for future labs.

Step 1: Log into tomalesbay.cs.unm.edu with an SSH client and change your password.

Your username is your last name in all lower-case with no apostrophes (e.g. “Crandall” => “crandall”, “Knockel” => knockel, “C'Debaca” => “cdebaca”, ...).

Your password is “MIPSFun1234” where 1234 is replaced by the last four digits of your LoboID.

Change your password right away as soon as you log in, using the UNIX command “passwd”.

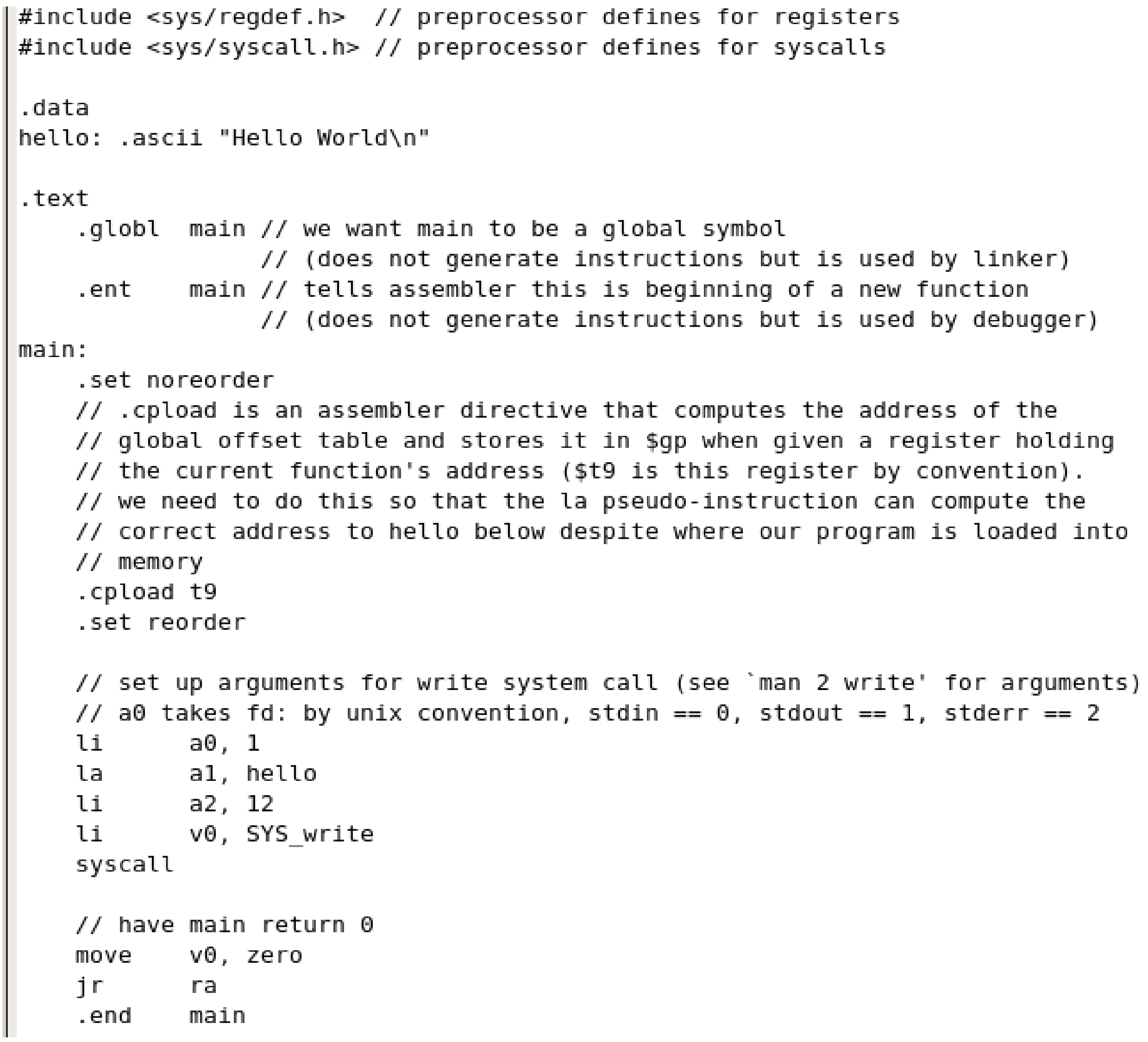
Step 2: Create a file with your last name and the lab number and a capital S as the extension (e.g., crandall-lab1.S). Type Jeff's hello world program (attached below) into the file manually, feel free to change the comments or use your own spacing preferences, etc., but make sure it's well-commented and readable. The purpose of typing it manually is to make sure you understand what every line does. If you don't understand what a line does as you type it, ask Jeff or I (I know it makes Jeff squirm when I say, “Jeff or I” instead of “Jeff or me”, so I'm going to do it regularly :-).

Step 3: Assemble your program with the following command (replacing the filename, of course)

```
gcc -Wall -Wextra -g crandall-lab1.S -o crandall-lab1
```

Now you should be able to execute your hello world program like this:

```
./crandall-lab1
```



```
#include <sys/regdef.h>  // preprocessor defines for registers
#include <sys/syscall.h> // preprocessor defines for syscalls

.data
hello: .ascii "Hello World\n"

.text
    .globl  main // we want main to be a global symbol
                 // (does not generate instructions but is used by linker)
    .ent    main // tells assembler this is beginning of a new function
                 // (does not generate instructions but is used by debugger)
main:
    .set noreorder
    // .cpload is an assembler directive that computes the address of the
    // global offset table and stores it in $gp when given a register holding
    // the current function's address ($t9 is this register by convention).
    // we need to do this so that the la pseudo-instruction can compute the
    // correct address to hello below despite where our program is loaded into
    // memory
    .cpload t9
    .set reorder

    // set up arguments for write system call (see `man 2 write' for arguments)
    // a0 takes fd: by unix convention, stdin == 0, stdout == 1, stderr == 2
    li      a0, 1
    la      a1, hello
    li      a2, 12
    li      v0, SYS_write
    syscall

    // have main return 0
    move    v0, zero
    jr      ra
    .end    main
```