CSC 3210 Computer Organization and programming

LAB 2

Learn about mov, and, or, not commands, along with local C variables

INSTRUCTIONS FOR LAB 2

Tasks to be done in this lab:

- log into a remote server
- examine assembly language code generated from a .c file
- learn about the "movl", "andl", "orl", and "notl" commands
- learn about how local variables are stored
- compile an assembly language file to an object file, then call "gcc" again to link the object file to an executable program
- compile an assembly language file to an executable directly
- run the executable program

ASSIGNMENT FOR LAB 2

- You need to create a log file, and demonstrate your ability to complete the tasks given in the previous slide. The log file is created with the "script" command, and any input or output after that will be stored in the log file, until you give it the "exit" command. Upload the log file to iCollege, making sure to use a .txt extension. Do not upload screen-shots.
- There are questions to answer throughout this lab. You can write the answers to a text file, and append it to your lab. Or you can write the answers to a text file, and "cat" it to the log. For example, you can use the vi editor to create your response, followed by cat lab2notes.txt (or whatever you call the file if it is not "lab2notes.txt"). For each answer, make sure to include the question text.

STEP-1

- First, get the "lab2_example.s" file. You can download it from here. You will need to put this under your account on SNOWBALL. You can use sftp as shown in the lab (remember that there is an example here). There are many other ways to do this, too, and sftp is not required for this assignment.

```
C:\Users\harsh>sftp bsrikakulapu1@snowball.cs.gsu.edu
bsrikakulapu1@snowball.cs.gsu.edu's password:
Connected to snowball.cs.gsu.edu.
sftp> ls
hello               hello.c             hello.o             hello.s             hello2              hello_original.s
log1.log            log2.log
sftp> lls lab2*
 Volume in drive C is Windows
 Volume Serial Number is 4A3A-14EA

 Directory of C:\Users\harsh

01/24/2025  10:25 PM             1,054 lab2_example.s
               1 File(s)          1,054 bytes
               0 Dir(s)  249,315,950,592 bytes free
sftp> put lab2_example.s
Uploading lab2_example.s to /home/bsrikakulapu1/lab2_example.s
lab2_example.s                                                                100% 1054   114.4KB/s   00:00
sftp> ls
hello               hello.c             hello.o             hello.s             hello2              hello_original.s
lab2_example.s      log1.log            log2.log
sftp> get lab2_example.s
Fetching /home/bsrikakulapu1/lab2_example.s to lab2_example.s
lab2_example.s                                                                100% 1054   114.4KB/s   00:00
sftp> lls lab2*
 Volume in drive C is Windows
 Volume Serial Number is 4A3A-14EA

 Directory of C:\Users\harsh

01/25/2025  01:17 AM             1,054 lab2_example.s
               1 File(s)          1,054 bytes
               0 Dir(s)  249,311,391,744 bytes free
sftp> quit
```

STEP-2

- Log in to SNOWBALL server
- Use the command “**ssh *your_account*@snowball.cs.gsu.edu**”
- Create a log file with the "**script**" command.
- Use the command “**script lab2.log**”
- You can use “**ls**” command to view the files that are already available in your server

```
C:\Users\harsh>ssh bsrikakulapu1@snowball.cs.gsu.edu
bsrikakulapu1@snowball.cs.gsu.edu's password:
Last login: Sat Jan 25 00:53:35 2025 from 162-194-60-9.lightspeed.tukrga.sbcglobal.net
   +
   |   GSU Computer Science
   |   Instructional Server
   |   SNOWBALL.cs.gsu.edu
   +
[bsrikakulapu1@gsuad.gsu.edu@snowball ~]$ script log2.log
Script started, file is log2.log
[bsrikakulapu1@gsuad.gsu.edu@snowball ~]$ ls
hello  hello2  hello.c  hello.o  hello_original.s  hello.s  lab2_example.s  log1.log  log2.log
```

STEP-3

- Use the "cat" command to display the contents of the "lab2_example.s" file. You should see content like the following.

```
[bsrikakulapu1@gsuad.gsu.edu@snowball ~]$ cat lab2_example.s
        .file   "lab2_example.c"
        .section        .rodata
.LC0:
        .string "%d %d\n"
        .text
        .globl  main
        .type   main, @function
main:
.LFB0:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        subq    $16, %rsp
        movl    $6, -4(%rbp)
        movl    $12, -8(%rbp)
        movl    $3, -12(%rbp)
        movl    -12(%rbp), %eax
        movl    -8(%rbp), %edx
        andl    %eax, %edx
        movl    -8(%rbp), %eax
        movl    -4(%rbp), %ecx
        andl    %ecx, %eax
        movl    %eax, %esi
        movl    $.LC0, %edi
        movl    $0, %eax
        call    printf
        movl    -12(%rbp), %eax
        movl    -8(%rbp), %edx
        orl     %eax, %edx
        movl    -8(%rbp), %eax
        movl    -4(%rbp), %ecx
        orl     %ecx, %eax
        movl    %eax, %esi
        movl    $.LC0, %edi
        movl    $0, %eax
        call    printf
        movl    -8(%rbp), %eax
        notl    %eax
        movl    %eax, %edx
        movl    -4(%rbp), %eax
        notl    %eax
        movl    %eax, %esi
        movl    $.LC0, %edi
        movl    $0, %eax
        call    printf
        movl    $0, %eax
        leave
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE0:
        .size   main, .-main
        .ident  "GCC: (GNU) 4.8.5 20150623 (Red Hat 4.8.5-44)"
[bsrikakulapu1@gsuad.gsu.edu@snowball ~]$
```

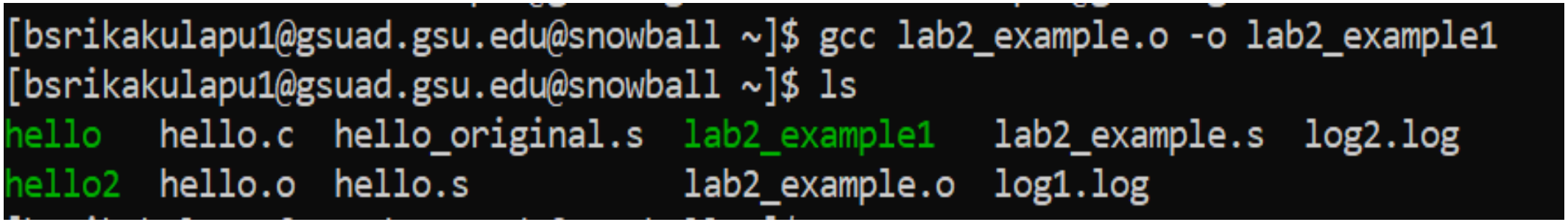
STEP-4

- Compile the "lab2_example.s" program. Remember that "gcc" is the GNU C compiler, which also works as an assembler. Use "gcc" to create an object file called “lab2_example.o". An object file is created before linking, and linking produces the executable file. Notice the "-c" argument that is passed to "gcc". Use command “**gcc -c lab2_example.s**”
- After compiling, use "**ls -l**" command to get a list of files. The "-l" option says that we want a long list, i.e. showing details. Notice how the "ls" output shows the new file "**lab2_example.o**".

```
[bsrikakulapu1@gsuad.gsu.edu@snowball ~]$ gcc -c lab2_example.s
lab2_example.s: Assembler messages:
lab2_example.s: Warning: end of file not at end of a line; newline inserted
[bsrikakulapu1@gsuad.gsu.edu@snowball ~]$ ls -l
total 68
-rwxrwxr-x. 1 bsrikakulapu1@gsuad.gsu.edu bsrikakulapu1@gsuad.gsu.edu  8360 Jan 20 02:44 hello
-rwxrwxr-x. 1 bsrikakulapu1@gsuad.gsu.edu bsrikakulapu1@gsuad.gsu.edu  8360 Jan 20 02:53 hello2
-rw-rw-r--. 1 bsrikakulapu1@gsuad.gsu.edu bsrikakulapu1@gsuad.gsu.edu   125 Jan 20 02:44 hello.c
-rw-rw-r--. 1 bsrikakulapu1@gsuad.gsu.edu bsrikakulapu1@gsuad.gsu.edu  1512 Jan 20 02:54 hello.o
-rw-rw-r--. 1 bsrikakulapu1@gsuad.gsu.edu bsrikakulapu1@gsuad.gsu.edu   504 Jan 20 02:47 hello_original.s
-rw-rw-r--. 1 bsrikakulapu1@gsuad.gsu.edu bsrikakulapu1@gsuad.gsu.edu   504 Jan 20 02:49 hello.s
-rw-rw-r--. 1 bsrikakulapu1@gsuad.gsu.edu bsrikakulapu1@gsuad.gsu.edu  1704 Jan 25 02:01 lab2_example.o
-rw-rw-r--. 1 bsrikakulapu1@gsuad.gsu.edu bsrikakulapu1@gsuad.gsu.edu  1054 Jan 25 01:16 lab2_example.s
-rw-rw-r--. 1 bsrikakulapu1@gsuad.gsu.edu bsrikakulapu1@gsuad.gsu.edu 20480 Jan 20 02:29 log1.log
-rw-rw-r--. 1 bsrikakulapu1@gsuad.gsu.edu bsrikakulapu1@gsuad.gsu.edu     0 Jan 25 01:21 log2.log
```

STEP-5

- In this step, we will use the object file to create an executable file. Again, we use "gcc" for this. Notice the "-o" argument that is passed to "gcc". The arguments "-o lab2_example" specify that we want the output to be a new file called "lab2_example". **Be careful with this step; if you were to type "-o lab2_example.s", it would over-write the S file, and you would have to start over.**
- The output file that this command creates is called "lab2_example1".
- Use the command "**gcc lab2_example.o -o lab2_example1**"



STEP-6

- You might be wondering if we must compile and link the program separately. In fact, we can put them together, as in the following.
- Use command "**gcc lab2_example.s -o lab2_example2**"

```
[bsrikakulapu1@gsuad.gsu.edu@snowball ~]$ gcc lab2_example.s -o lab2_example2
lab2_example.s: Assembler messages:
lab2_example.s: Warning: end of file not at end of a line; newline inserted
[bsrikakulapu1@gsuad.gsu.edu@snowball ~]$ ls -l
total 92
-rwxrwxr-x. 1 bsrikakulapu1@gsuad.gsu.edu bsrikakulapu1@gsuad.gsu.edu  8360 Jan 20 02:44 hello
-rwxrwxr-x. 1 bsrikakulapu1@gsuad.gsu.edu bsrikakulapu1@gsuad.gsu.edu  8360 Jan 20 02:53 hello2
-rw-rw-r--. 1 bsrikakulapu1@gsuad.gsu.edu bsrikakulapu1@gsuad.gsu.edu   125 Jan 20 02:44 hello.c
-rw-rw-r--. 1 bsrikakulapu1@gsuad.gsu.edu bsrikakulapu1@gsuad.gsu.edu  1512 Jan 20 02:54 hello.o
-rw-rw-r--. 1 bsrikakulapu1@gsuad.gsu.edu bsrikakulapu1@gsuad.gsu.edu   504 Jan 20 02:47 hello_original.s
-rw-rw-r--. 1 bsrikakulapu1@gsuad.gsu.edu bsrikakulapu1@gsuad.gsu.edu   504 Jan 20 02:49 hello.s
-rwxrwxr-x. 1 bsrikakulapu1@gsuad.gsu.edu bsrikakulapu1@gsuad.gsu.edu  8368 Jan 25 02:14 lab2_example1
-rwxrwxr-x. 1 bsrikakulapu1@gsuad.gsu.edu bsrikakulapu1@gsuad.gsu.edu  8368 Jan 25 02:19 lab2_example2
-rw-rw-r--. 1 bsrikakulapu1@gsuad.gsu.edu bsrikakulapu1@gsuad.gsu.edu  1704 Jan 25 02:01 lab2_example.o
-rw-rw-r--. 1 bsrikakulapu1@gsuad.gsu.edu bsrikakulapu1@gsuad.gsu.edu  1054 Jan 25 01:16 lab2_example.s
-rw-rw-r--. 1 bsrikakulapu1@gsuad.gsu.edu bsrikakulapu1@gsuad.gsu.edu 20480 Jan 20 02:29 log1.log
-rw-rw-r--. 1 bsrikakulapu1@gsuad.gsu.edu bsrikakulapu1@gsuad.gsu.edu     0 Jan 25 01:21 log2.log
```

STEP-7

- Now run the first executable program.
- Use command "**./lab2_example1**"

```
[bsrikakulapu1@gsuad.gsu.edu@snowball ~]$ ./lab2_example1
4 0
14 15
-7 -13
```

STEP-8

- Now run the second executable program.
- Use command "**./lab2_example2**"

```
[bsrikakulapu1@gsuad.gsu.edu@snowball ~]$ ./lab2_example2
4 0
14 15
-7 -13
```

movl command

- The movl command is used to move a piece of "long" data (32 bits) from one place to another, zeroing out the upper 32 bits of the destination register.
- The base pointer holds the address for the frame pointer, and can be used for local variables. That is, a local variable may be at (%rbp - 4), and another local variable at (%rbp - 8), etc.
- For example, **movl -8(%rbp), %eax** says to move the value from the base pointer, minus 8, to the EAX register
- **movl -4(%rbp), %ecx** says to move the value from the base pointer, minus 4, to the ECX register
- **movl $6, -4(%rbp)** moves the "immediate" number 6 to the base pointer - 4. The base pointer indicates a local memory area, and subtracting 4 or 8 from it allows us to specify a location to hold a 4 byte value.
- Now, look up the commands andl, orl, notl.

STEP-9

- Check what happens if we copy the executable code from one machine to another. Use sftp to download the example onto another x86 based machine. When attempting to run it, see if the following error occurs.

```
C:\Users\harsh>sftp bsrikakulapu1@snowball.cs.gsu.edu
bsrikakulapu1@snowball.cs.gsu.edu's password:
Connected to snowball.cs.gsu.edu.
sftp> ls
hello               hello.c             hello.o             hello.s             hello2              hello_original.s

lab2_example.o      lab2_example.s      lab2_example1       lab2_example2       log1.log            log2.log

sftp> get lab2_example1
Fetching /home/bsrikakulapu1/lab2_example1 to lab2_example1
lab2_example1                                                                   100% 8368   454.0KB/s   00:00
sftp> lls lab2*
 Volume in drive C is Windows
 Volume Serial Number is 4A3A-14EA

 Directory of C:\Users\harsh

01/25/2025  01:17 AM             1,054 lab2_example.s
01/25/2025  03:01 AM             8,368 lab2_example1
               2 File(s)          9,422 bytes
               0 Dir(s)  249,313,259,520 bytes free
sftp> quit
```

```
mweeks@air:csc3210_temp$ ./lab2_example
-bash: ./lab2_example: cannot execute binary file
```

STEP-10

- Finally, exit from the script command. This informs the "script" utility that we are done.
- You might want to next use "cat" on the "lab1.log" file, copy the output, and paste it into a text file called "lab1.txt". This should get rid of any extraneous (control) characters, though you should look it over to make sure.

```
[bsrikakulapu1@gsuad.gsu.edu@snowball ~]$ exit
exit
Script done, file is log2.log
```

- Next, exit again to log out of SNOWBALL.
- You can use "sftp" to get your log file from the server.

```
[bsrikakulapu1@gsuad.gsu.edu@snowball ~]$ exit
logout
Connection to snowball.cs.gsu.edu closed.
```

Submit your log file on icollege, with your name, email and session number!

NOTE:

You can use the filename "lab2.txt" for the version of the log file that you turn in. Do not upload files to iCollege without an extension, or with an extension of .log.