Arlan Prado
CS 118-02

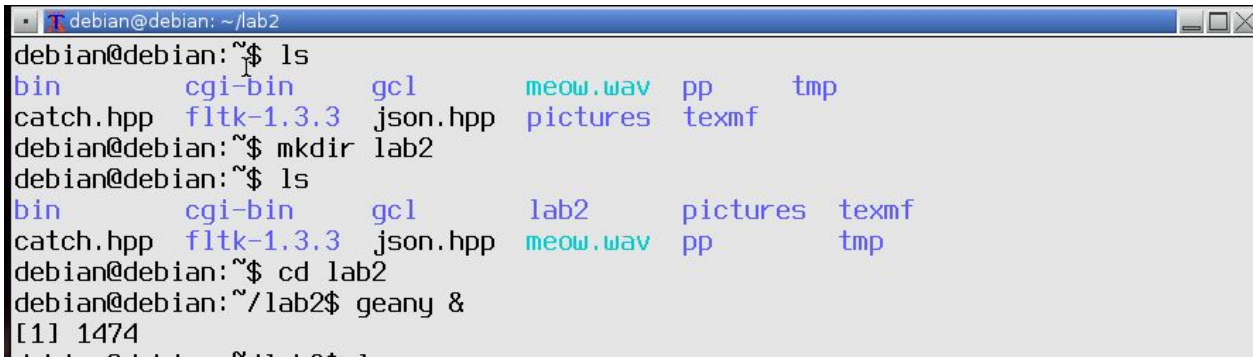
Lab 2

Purpose:

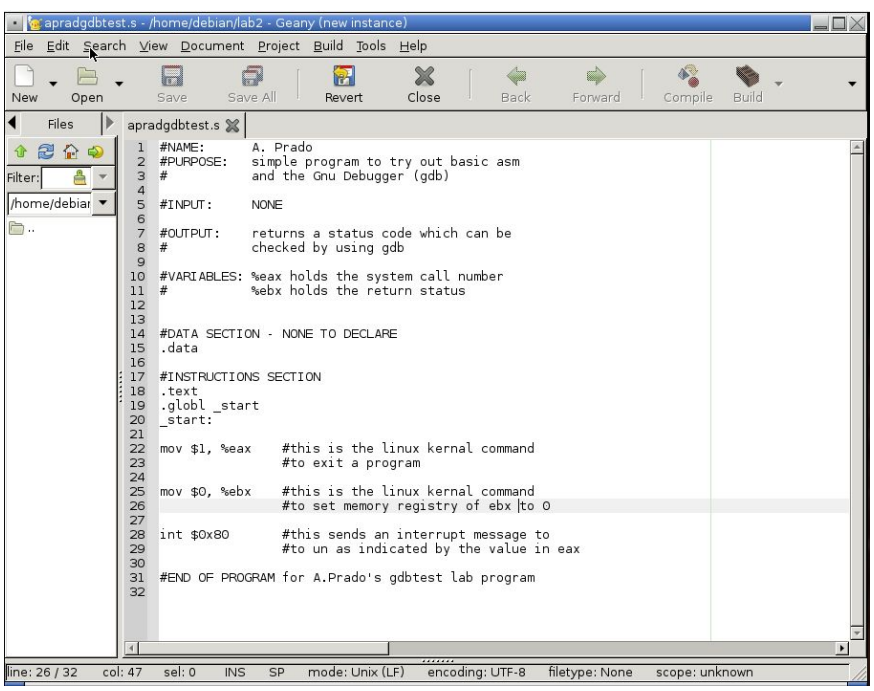
This assignment is to start me with assembly language. It helps me get used to the commands and gdb, the syntax of assembly language, and to ensure the virtualbox can run the assembly language.

Process:

The first thing to do is open the virtualbox and create a directory for lab2 using the debian terminal and running geany through the new directory.



Then I typed in the code that was given to me by the professor and make the edits that are required. The # lines are comments to clarify who made it, what for, and how it was made. The mov line is to copy a value of one register to another register.



After writing the code, save the file as apradgdbtest.s. Then access the terminal and enter the command “as apradogdbtest.s -g -o apradgdbtest.o” and fix any errors that occur. After getting the as command to come up with no errors, run the ld command to allow the .s file to be executable. Once making sure the file is an executable, then run gdb with the executable.

```
debian@debian:~/lab2$ as apradgdbtest.s -g -o apradgdbtest.o
debian@debian:~/lab2$ ld apradgdbtest.o -o apradgdbtest
debian@debian:~/lab2$ gdb apradgdbtest
```

The next thing to do is to set up breakpoints at the first variable (line 22) and first integer (line 28) with “b [line #]”. Run the debugger with “r” and when it runs into a breakpoint, take note and enter “c” to continue the debugger. I did this until the debugger finished.

```
---Type <return> to continue, or q <return> to quit---
Type "apropos word" to search for commands related to "word"...
Reading symbols from apradgdbtest...done.
(gdb) b 22
Breakpoint 1 at 0x8048054: file apradgdbtest.s, line 22.
(gdb) b 28
Breakpoint 2 at 0x804805e: file apradgdbtest.s, line 28.
(gdb) r
Starting program: /home/debian/lab2/apradgdbtest

Breakpoint 1, _start () at apradgdbtest.s:22
22      mov $1, %eax    #this is the linux kernal command
(gdb) info r eax ebx
eax            0x0      0
ebx            0x0      0
(gdb) c
Continuing.

Breakpoint 2, _start () at apradgdbtest.s:28
28      int $0x80       #this sends an interrupt message to
(gdb) info r eax ebx
eax            0x1      1
ebx            0x0      0
(gdb) c
Continuing.
[Inferior 1 (process 1538) exited normally]
(gdb) q
debian@debian:~/lab2$
```

This ended the project successfully.

Pitfalls:

The instructions were clear and it had very little problems through the process. The only problems I had were compile errors. But they were easily solved as they were spelling errors.

```
debian@debian:~$ cd '/home/debian/lab2'
debian@debian:~/lab2$ as apradgdbtest.s -g -i apradgdbteset.o
as: unrecognized option '-i'
debian@debian:~/lab2$ as apradgdbtest.s -g -o apradgdbteset.o
apradgdbtest.s: Assembler messages:
apradgdbtest.s:15: Error: unknown pseudo-op: `.dlobl_start'
debian@debian:~/lab2$ as apradgdbtest.s -g -o apradgdbteset.o
apradgdbtest.s: Assembler messages:
apradgdbtest.s:15: Error: unknown pseudo-op: `.globl_start'
debian@debian:~/lab2$ as apradgdbtest.s -g -o apradgdbteset.o
apradgdbtest.s: Assembler messages:
apradgdbtest.s:15: Error: unknown pseudo-op: `.global_start'
debian@debian:~/lab2$ as apradgdbtest.s -g -o apradgdbteset.o
debian@debian:~/lab2$ ld apradgdbtest.o -o apradgdbtest
ld: cannot find apradgdbtest.o: No such file or directory
debian@debian:~/lab2$ as apradgdbtest.s -g -o apradgdbtest.o
debian@debian:~/lab2$ ld apradgdbtest.o -o apradgdbtest
debian@debian:~/lab2$ gdb apradgdbtest
```

Possible Improvements:

An improvement could be to have more of an understanding of the gdb because I know in the future that we are going to be using the debugger a lot. It would be a lot smoother for me if I knew basic commands for the gdb.