

# ENGINEERING PHYSICS

## 300 Level Laboratories

### Assembly Language Programming and simple i/o.

#### *Introduction*

These notes are an introduction to parts 5 and 6 of your laboratory course in digital electronics. There are also additional notes for parts 5 and 6 which will be given to you later. At this stage in your course you have learnt in Chapters Two and Three how memory elements and simple arithmetic circuits may be constructed from logic elements. In Chapter Four you saw how the CPU (which does the arithmetic and logical operations) communicates with the memory and i/o devices (in Chapter Four these were only leds). You also wrote simple assembly language routines and hand coded these into machine language (ones and zeros). Now you are going to build on this knowledge and write assembly routines for a real computer, get it to output signals to real devices, and then read back signals from real devices.

In order to do all of the above there are several things you may first need to learn to use. These are;

- Linux - the operating system for the computer you will be using,
- Emacs - the text editor you will be using,
- NASM - the assembler which translates your assembly code into machine code understandable by any x386 processor (this includes all flavours of 80386, 80486 and Pentium processors),
- gcc - the gnu C compiler which does the linking,
- gdb - the gnu debugger which allows you to run your program instruction by instruction, examining registers and memory locations as you go.

If this looks like a lot to learn, that is because it is. At the end of it all, however, you will have learnt a lot of useful things in addition to how to interface a computer to an instrument. The one restriction is that what you will have learnt about the instruction set for NASM will only apply to Intel (or equivalent such as AMD) type processors. This is because different processors have different architectures and hence different instruction sets.

If you are already familiar with Linux (or any Unix type of operating system) you can probably skip Exercise 5 which provides a tutorial introduction to Linux and Emacs. If not, please do Exercise 5. This exercise may be conducted on any of the Linux boxes in the 2nd year Physics Laboratory (once you have arranged for the laboratory supervisor to give you an account on one), or on any other Unix system you have an account on (you all have an account on Wumpus). The following notes are written assuming that you are doing this exercise in the lab, but they apply anywhere with minor, commonsense,

modifications.

## Laboratory Exercise 5

### Introduction to Using Linux

#### Part 1 - logging in

The first thing you need to be able to do in Linux is to log in. Thus you need to know your user name and password. Turn on the computer. If you do nothing then eventually the Linux operating system will be booted. You may, by intervening at the correct time, get the computer to boot the Windows 95 operating system but why would you want to use that? In all that follows precisely what you see is determined by how the operating environment has been setup so there will be minor differences from system to system. Once you have learnt more about Linux you will be able to customise your environment.

Once Linux has booted you should see something like this

```
login:
```

At the login prompt type your user name (case is important) and hit the enter key.

```
login:username
```

The computer should respond with

```
password:
```

Now type in your password, remembering that case is important. Note that as you type your password you will see nothing new appear on the screen. If you have typed your password correctly, then after you hit the enter key you should see something like this

```
[username@localhost username]$
```

This tells you that the user *username* is logged into the machine called *localhost*, and that you are in the sub-directory *username*. The \$ symbol is your shell prompt. To log out from Linux type **exit**. Note that you must not switch the computer off after you have exited.

#### Part 2 - navigating the directory tree

Type **pwd** and hit enter. You should now see the following

```
[username@localhost username]$ pwd
/home/username
[username@localhost username]$
```

The command **pwd** is used to find out where, in the computer's directory tree, your present working directory is located. The results of the above tell you that your present working

directory, *username*, is a subdirectory of *home*, which itself is a subdirectory of the root or highest level directory, denoted sometimes by */*.

To examine the other directories at the same level as */home*, type the following;

```
[username@localhost username]$ ls /
```

This gives you a short listing of the contents of the directory */*. You can find out more about the command **ls** by typing

```
[username@localhost username]$ man ls
```

Do this and see if you can find out how to determine the permissions of a directory or file. If you can't find this out from the above command then use the following

```
[username@localhost username]$ info ls
```

What command do you need to type to list permissions? Use this command and find a directory in */* for which you do not have any read permissions. What happens when you try to determine the contents of this directory?

Use the command **cd** to change to another directory. In particular try the following;

```
[username@localhost username]$ cd .
```

```
[username@localhost username]$ cd ..
```

```
[username@localhost username]$ cd /
```

```
[username@localhost username]$ cd ~
```

```
[username@localhost username]$ cd /usr/local
```

After each of these determine which directory you are in, either by looking at your prompt or by using **pwd**.

### Part 3 - directory and file manipulation

Go back to your home directory. To make a new directory which is a sub-directory of your home directory enter

```
[username@localhost username]$ mkdir assembly
```

What you have done is tell the computer to make a directory called *assembly* which is a sub-directory of your present working directory, which in this case will be your home directory. Note that if you were in any directory other than your home directory, you could have achieved the same result with

```
[username@localhost anywhere]$ mkdir /home/username/assembly
```

If you now enter

```
[username@localhost username]$ ls assembly
```

you will get a listing of the contents of the directory *assembly*. As it is a newly created directory there will be no files in it. Copy a file from your home directory into this new directory. If you don't have any files in your home directory type the following

```
[username@localhost username]$ ls -a
```

This should list for you some hidden files that you don't see with the **ls** command. Copy one of these (*.bashrc* for example) into the new directory by typing

```
[username@localhost username]$ cp .bashrc assembly
```

If you now enter

```
[username@localhost username]$ ls -a assembly
```

you should see *.bashrc* listed. Note that the file */home/username/.bashrc* still exists. You have made a copy of it called */home/username/assembly/.bashrc*. If you had wanted to call the copy something else, e.g. *woof*, you would enter

```
[username@localhost username]$ cp .bashrc assembly/woof
```

What happens if you want to copy a file from the hard drive to a floppy? Obviously you need to insert a floppy disc into the drive but how do you access it? The contents of the drive need to be mounted as a filesystem. This is done using the **mount** command

```
[username@localhost username]$ mount /mnt/floppy
```

on the 3rd yr lab machines, or sometimes on other machines

```
[username@localhost username]$ mount /floppy
```

Once the floppy is mounted you can read and write to it. Try listing the contents of a floppy (ask the supervisor for one if you don't have one handy), and then copying a file from the hard drive to the floppy. When you have finished with the floppy remember to unmount it using

```
[username@localhost username]$ umount /mnt/floppy
```

Sometimes, you will not want to copy a file, but simply to move it. To do this you use the command **mv**. Note that moving a file is equivalent to renaming it. For information about using this command use **info** or **man**. However, do not use it to move any files starting with *.* from your home directory.

The command to remove files is **rm**. Try using it to remove the directory *assembly*. As *assembly* is a directory this should not work. The command to remove directories is **rmdir**: this only works if the directory is already empty. So to remove a directory you can manually delete every file in every sub-directory, then delete every sub-directory, and then when the directory is completely empty it too may be removed. Alternatively, you can use the **-r** option for the command **rm**. For example

```
[username@localhost username]$ rm -r assembly
```

will remove every file in *assembly*, then enter any sub-directories in *assembly* and remove all their files and so on, and finally remove *assembly*.

For all the commands used above except **cd**, you can get further information on their usage using either **info** or **man**. Now that you know how to move around the filesystem, create and delete directories and move and copy files, it is time to learn how to create and edit text files. This is the topic of the next section on using the Emacs text editor. There are several other editors available for Linux and Unix machines, but Emacs is probably the most popular.

## Introduction to Emacs

Most of what you need to know about using Emacs can be found out by using the Emacs tutorial or typing **info emacs**. To access the Emacs tutorial you need to be running the command **emacs**. Make a directory */home/username/assembly/intro* and copy into it the file */home/username/.bashrc*. Get into the directory *~/assembly/intro*. Now run the following command

```
[username@localhost intro]$ emacs .bashrc
```

You are now running **emacs** on the file *.bashrc*. How do you quit out of Emacs? This is where you learn the most important thing about Emacs - the use of the CTRL key. While holding down the CTRL key type **x** then **c** (also written as **CTRL-x CTRL-c**). If you have done this correctly you should now be back at your prompt. Open *.bashrc* again in Emacs. To save any changes you make type **CTRL-x CTRL-s**. Now you know how to open files in Emacs, save changes to them, and quit out of Emacs. Everything else you need to know can be found out by typing **CTRL-h t** (release the CTRL key before typing **t**) while Emacs is running. This gets you into the Emacs tutorial where you learn how to navigate around the text file, change text, kill, yank, paste, etc.,.

When you have finished with the tutorial try making the following change to the *.bashrc* file you have copied into *~/assembly/intro*. On any new line type **echo "Plant more trees"**. Save this change and exit emacs. Change the name of the *.bashrc* file in your home directory to *.bashrc\_old*. Now copy the *.bashrc* file in *~/assembly/intro* to *~* (your home directory). Exit from Linux and then log back in. What do you see above your prompt?

You should now be ready to move on to Exercise 6 on using NASM, gcc and gdb.