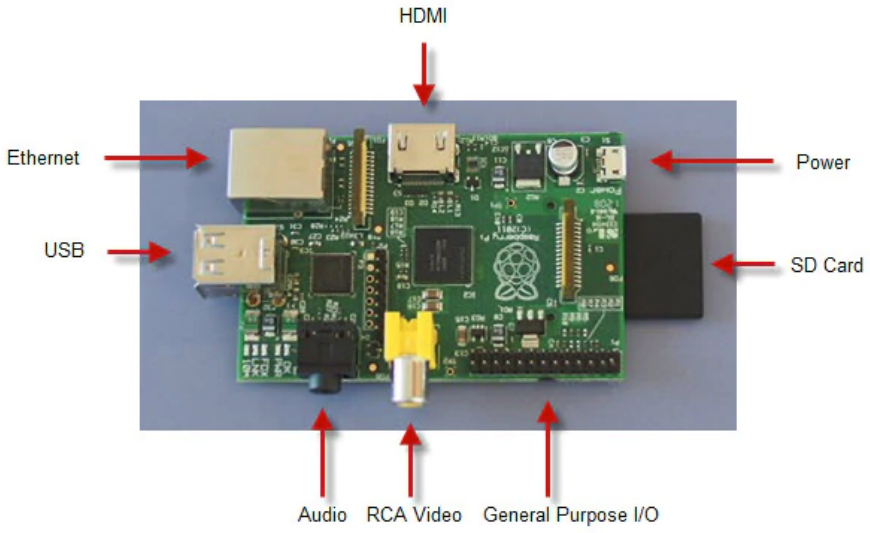
CS1520 Practical 1

Alessandro Moura

January 23, 2018

1 Setting up the Raspberry Pi

We will use the Raspberry Pi computer in our practical classes. The Raspberry Pi is a small single-board computer, build around a 32-bit ARM CPU. Each student in the course receives a Raspberry Pi in the first practical; they must be returned to the instructor at the end of the course. The students must bring their Raspberry Pis to the practical classes.



The Raspberry Pi single-board computer.

1.1 Connecting the Raspberry Pi to Peripherals

In order to use the Raspberry Pi, you must connect it to a USB mouse, a USB keyboard, a monitor, and a power supply. In addition, you must insert the SD memory card containing the operating system and the file system. Referring to the image in the previous page, this is how to connect the Raspberry Pi:

- The mouse is connected to one of the USB ports.
- The keyboard is connected to one of the USB ports.
- The video is connected to the HDMI port.
- The power source is connected to the port labelled as "Power".
- The SD card should be inserted in the slot labelled "SD card". The card's contacts should face up, touching the board.

The correct way to connect the Raspberry Pi to the peripherals will be demonstrated by your instructor in the first practical.

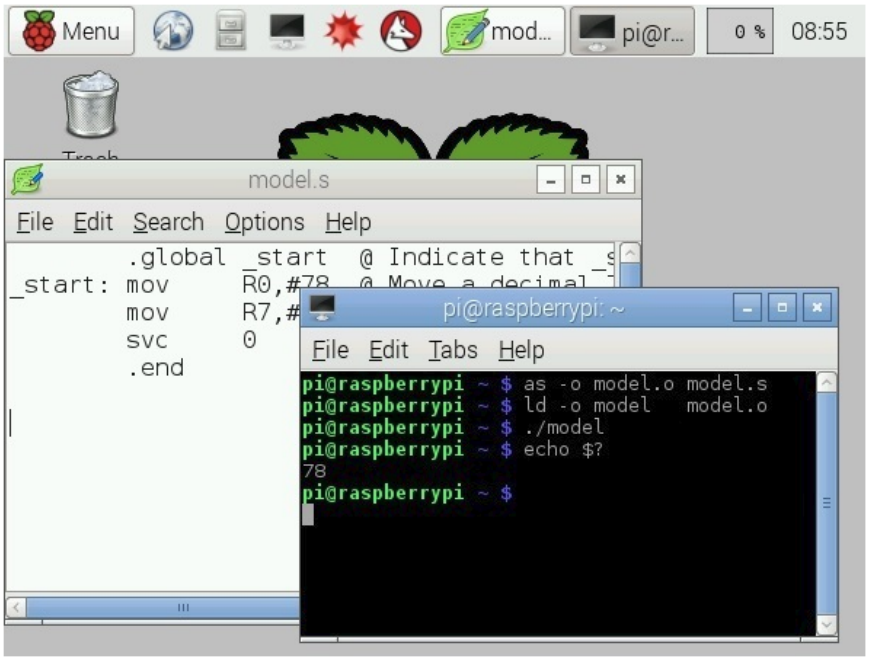
Once everything is connected and the card is inserted in the appropriate slot in the Raspberry Pi, you can turn the power on, and the Raspian Linux operating system will boot up.

All the peripherals will be available in the computer lab; students will receive the SD card and the power source along with the Raspberry Pi; they must be brought to every practical class. If you have a USB keyboard and mouse, and a monitor that accepts HDMI input, you can use the Raspberry Pi at home as well.

1.2 Health and safety warning

Do not touch the raspberry pi boards while they are plugged into power. You should also be careful after you unplug them in case they have become hot.

2 Raspian environment, and basic Unix commands



A text editor and a terminal window in Raspian.

The image above shows the graphical interface for the Raspian operating system, which will greet you after you successfully boot your Raspberry Pi. In this example, two applications are running: a text editor and a command terminal. We will use this text editor to type our assembly programs, and we will type commands in the terminal to assemble and link the programs into executable files.

Open a terminal window: click on the upper-left raspberry icon to bring up the applications menu; select "Accessories", and then select "Terminal".

2.1 Basic Unix commands

We will interact with the Raspberry Pi mainly through commands we enter in the terminal. To start with, the command `ls` lists the directories (folders) and files in the current directory. Type `ls`, followed by enter, and see what you have on your system.

It is a good idea to create a directory to put all the files you create in the course. To create a folder named **cs1520**, type

`mkdir cs1520`

You can see the new directory by typing `ls` again.

Make the new directory your current directory with the command

`cd cs1520`

Then type `ls` again, just to check that the new directory is empty. To return to the directory that contains the current directory, type

`cd ..`

To return to your home directory (the root folder of your files), type

`cd`

2.2 Writing your first assembly program

Listing 1: hello.asm

```
.global main

main:
mov r7, #4
mov r0, #1
ldr r1, =message
mov r2, #16
svc 0

mov r7, #1
svc 0

.data
message: .ascii "What's up, doc?\n"

.end
```

Let us create a new file, where we will type our first assembly program. Enter the command

`leafpad hello.asm &`

Note the ampersand (&) at the end of the command! This will bring up a new window with a text editor. Your screen should now look similar to the

image above.

The first thing you should do is to make the editor show line numbers — that will be useful for programming. To do that, click "Options" in the editor's menu bar, and then select "Line Numbers".

Now type in the assembly program listed above.

When assembled and executed, it simply outputs the message "What's up, doc?" to the terminal. We will not explain in detail what this program works: for now, it is just an example that we use to learn how to write assembly programs.

After you have entered the program in the editor, save it (**File→Save**).

2.3 Assembling and linking your program

Now that we have a file with the assembly program, we need to generate an executable from it. The file **hello.asm** itself cannot be executed as a program — it is just a text file. We need to run the assembler, to translate **hello.asm** into a file containing machine instructions that the computer can run. This is done with the command

```
as -g -o hello.o hello.asm
```

This creates a new file **hello.o**, containing the CPU instructions corresponding to our assembly program. If you made a mistake when typing the program, an error message will appear, usually pointing out the line number where the error occurred. If that is the case, go ahead and fix it.

hello.o is still not the executable file we want: it must be linked with the command

```
gcc -g -o hello hello.o
```

This generates a new file, **hello**, which can be executed. Go ahead and enter the command

```
./hello
```

and see the message appear in the terminal. Success! Note the period and the backslash before the word "hello".

2.4 Simplifying the assembler and linking process

In the folder where your programs will be, create a new file named **build** by entering the command

```
leafpad build
```

The file consists of only two lines, and is listed below

Listing 2: build

```
#!/bin/bash
as -g -o $1.o $1.asm && gcc -g -o $1 $1.o
```

Save the file, and then enter the command

`chmod +x build`

The file **build** is now a command that executes the two steps above (assembling and linking), making the process less tedious. In order to generate the executable for **hello.asm**, for example, we enter the command

`./build hello`