Microprocessor-Based Systems (E155)

D. Harris and M. Spencer **Fall 2015**

Lab 4: Life of Pi

Requirement

1) *Set up your Raspberry Pi*
2) *Write an assembly-language program to sort an array of 12 signed bytes on the Pi*

Setting up the Pi

To familiarize yourself with the Pi, Linux, and how to connect to the Pi

a) If you are an inexperience Linux user, read "Linux and the Pi"

b) Windows users should read the "Running X11 in Windows" to learn how to connect to a Pi from one of the lab computers or your own using X11 forwarding.

c) If you are rusty on ARM assembly language, review Chapter 6 of *Digital Design & Computer Architecture*.

Pick up a Raspberry Pi 2.0, SD card, USB power supply, and Pi Cobbler cable from the stockroom. This is yours to keep.

We'll be running a common OS designed for the Raspberry Pi known as Raspbian. Download and install the New Out Of the Box Software (NOOBS) onto your SD card. Instructions are at

https://www.raspberrypi.org/downloads/noobs/

After downloading the ~1GB image, follow the link to the setup guide. You will have to reformat the SD card first with the SD Formatter utility as described, then copy the image onto the SD Card. Reformatting the SD card requires Administrator permission. You probably have Admin permission on your personal laptops, so use your own laptop or a friend's. If you don't have an SD card slot on your laptop, you can borrow a USB SD Card Reader from the stockroom.

Next, boot your new baby Pi. Put the SD card into the Pi, plug one of the monitors in the lab into the Pi via an HDMI cable, plug a keyboard into the USB port, attach an Ethernet

cable, and insert in the micro USB power cord. Follow the directions at startup to install Raspbian; this may take about half an hour.

Log in with the user name "pi" and password "raspberry."

Once the Pi boots up you should get a blue options screen. If it goes to the command line you should be able to get to the configuration screen by invoking

```
sudo raspi-config
```

This opens a GUI editor that allows you to change certain system settings. Under Internationalisation Options, choose Keyboard. Accept the default keyboard (e.g. Generic 105-key (intl) PC). Change the layout to "other" and choose English(US), then follow the other prompts. Under Advanced Options, enable the serial peripheral interface (SPI) to use in future labs. Also under Advanced, change the host name to something like raspi-XX, with XX being your initials, to distinguish your Pi from the others on the network. (Note that dashes are allowed but underscores will cause mailx to fail later in this lab.) Change your password to something more secure. Finish and reboot.

To install packages we'll use a program called `apt-get` first to update itself, and then to install other packages. Now we can install some software that'll be useful in this and future labs, including the `ddd` debugger and the `sendmail` tool. This update process will take a while.

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install ddd
sudo apt-get install sendmail
sudo apt-get install heirloom-mailx
```

You will mostly use the Pi in headless mode, with an Ethernet connection but no monitor or keyboard. To connect to a headless Pi, you need to know its IP address. One of the simplest ways to deal with this is to have the Pi email its IP address to you, using various software packages.

For the labs you will need to edit code, so you can use the installed text editor `nano`, or install your own favorite one. We'll want the Pi to email us every time it boots up so we'll add a line to the `/etc/rc.local` file. This is just a text file that runs right after boot up, but before any users log in. Edit the file using the command

```
sudo nano /etc/rc.local
```

Add the following line before the exit statement:

```
curl ifconfig.me | mailx -s "my IP" YOUREMAIL@hmc.edu
```

Restart you Pi with

```
sudo reboot
```

to allow the changes to take effect and check that you received an email containing the IP address (this may take about 4 minutes). Now you should be able to follow the instructions to connect to the Pi using X11 forwarding from Windows.

From a Mac, the process is simpler. Open a terminal and type

```
ssh -Y pi@134.173.38.108
```

(or whatever IP address you were assigned.)

Programing the Pi in Assembly

Now write an assembly language program called `sort.s` to sort 12 signed bytes. See the appendix for an example to get started. Remember that assembly language code is nearly unreadable without line-by-line comments. Assemble the program into a binary executable using GCC:

```
gcc -g  -o sort sort.s
```

The `-g` flag allows a debugger to display the assembly language code which is useful for debugging. `-o` specifies the name of the executable file.

You can look at the executable using

```
objdump -S sort
```

You'll see that quite a bit of startup code has been added. Although we can run this code directly, to be able to see the values in the register you'll want to run it with the `ddd` debugger:

```
ddd sort
```

`ddd` provides a lot of useful debugging information. Click on the main label and choose Break to set a breakpoint at the start. Choose Status -> Registers to open a window to view the registers. Click on Run to start the program. Then click on Step repeatedly to step through the program and watch values in the registers. Check against your expectation at each step to debug.

To find the base address of the array, click on the array label and choose Source -> Lookup. To watch the content of memory, use the Data -> Memory function. For example, display 12 hex words starting from the base address of the array.

What to Turn In

- A listing of your program that sorts numbers.
- Explicitly state the test cases you used and the output of the tests on each program. Be sure your tests would convince a skeptic that your algorithm works.
- How many hours did you spend on the lab? Any comments, suggestions, or complaints about the assignment? This will not count toward your grade.

Appendix A: Sample Code

```
// sort.s
// 28 September 2015 david_harris@hmc.edu
// sort twelve numbers for E155 lab 4

.text
.global main
main:
        LDR R3, =array  // load base address of a into R3

        … more code here

        BX LR           // return from main

.data
array:
        .byte 0x08
        .byte 0x10
        .byte 0xFF

        … more data here

.end
```