Microprocessor-Based Systems (E155)

D. Harris and M. Spencer **Fall 2017**

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) Connecting to a Pi from a Mac terminal is straightforward. 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*.

Hardware Setup

Pick up a Raspberry Pi 3 Model B, SD card, USB power supply, and Pi Cobbler cable from the stockroom. This is yours to keep.

We'll be running a common Linux OS designed for the Raspberry Pi known as Raspbian. Your 8 GB SD card should already come installed with the NOOBS (New Out Of the Box) Raspbian installation. If it weren't installed, you could download yourself using the setup guide and a SD card writer on your laptop.

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

Next, connect your new baby Pi. Pull the microSD card out of the SD card holder and plug it in to the socket on the back of the Pi.

This first time you use the Pi, you'll connect it to a monitor, keyboard, and mouse to set it up. In the future, you can access it over a wireless network from a terminal on your

computer. Find an HDMI-DVI monitor cable in the supply cabinet to connect the Pi to a lab monitor. Plug in the mouse and keyboard to USB ports. Plug in the microUSB power cord, and your Pi should boot. Note that the Pi autodetects the monitor resolution at boot time; if you plug in the HDMI cable after booting, you may get lousy resolution until you reboot. The first time you boot will take a little while.

Configuration

Once the Pi boots up, open a terminal window by clicking on on the terminal icon near the upper left. Type

```
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 SSH (to accept network connections) and the serial peripheral interface (SPI) to use in future labs. Under Advanced Options, 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 from the default of 'raspberry' to something more secure. Finish and reboot.

Wireless Network Access (what a pain!)

Next we need to configure the Pi to be able to access Claremont-WPA. First, create a hash of your HMC network password so you won't have to store it in plain text (replacing YOUR_PASSWORD with your actual password).

```
echo -n 'YOUR_PASSWORD' | iconv -t utf16le | openssl md4
```

The output should look like

```
(stdin)= a1234exxxxxxxx72f6db0263
```

Open a new terminal window and edit the secure network configuration file by typing:

```
sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

Change the file to read as follows, substituting your username and the 32-digit hashed password (preferably cut and pasted). Proofread carefully to avoid frustration.

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=US

network={
     ssid="Claremont-WPA"
     proto=RSN
     key_mgmt=WPA-EAP
     pairwise=CCMP TKIP
     group=CCMP
     identity="YOUR_USER_NAME@hmc"
     password=hash:YOUR_HMC_PASSWORD_HASH
```

```
        phase1="peaplabel=0"
        phase2="auth=MSCHAPV2"
}
```

Now edit the network interface file:

```
sudo nano /etc/network/interfaces
```

Find the section regarding the wlan0 interface and replace it with:

```
auto wlan0
allow-hotplug wlan0
iface wlan0 inet dhcp
          pre-up wpa_supplicant -B -Dwext -i wlan0 -c/etc/wpa_supplicant/wpa_supplicant.conf
          post-down killall -q wpa_supplicant
```

Shut down the network interface and bring it back up:

```
sudo ifdown wlan0
```

```
sudo ifup wlan0
```

Ignore some known warnings about ioctl invalid arguments.

If all went well, you should be able to reach a server:

```
ping hmc.edu
```

Otherwise, check for typos in your interfaces and wpa_supplicant.conf files.

Update and Install Packages

To install packages we'll use a program called `apt-get` first to update itself, and then to install other packages. (Expect some warnings along the way, and possibly no packages to upgrade.) Now we can install some software that'll be useful in this and future labs, including the `ddd` debugger and the `festival` text-to-speech tool. This upgrade process will take a while, so go onto the Programming the Pi part of the lab in another terminal while you wait. Pay attention to the upgrade because it may prompt you for things.

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

For the labs you will need to edit code, so you can use the installed text editor `nano`, or install your own favorite one.

Ask the Pi to send you its IP address

You will mostly use the Pi in headless mode, with a wireless connection but no monitor or keyboard. To connect to a headless Pi, you need to know its IP address.

Type

```
ifconfig
```

and write down the inet addr that it reports. It should be in the form 134.173.x.y where x and y are some more digits.

The IP address assigned by DHCP (Dynamic Host Configuration Protocol) and might change from time to time, especially if the network goes through any reconfiguration. Thus it is helpful to have a way to get the current IP address. One of the simplest ways to deal with this is to have the Pi email its IP address to you at startup.

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 lines before the exit statement:

```
sleep 30
/sbin/ifconfig wlan0 | grep 'inet addr:' | sendmail YOUREMAIL@g.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, and it might appear in your Junk folder).

Access from Windows

Follow the "Running X11 in Windows" instructions to connect to the Pi using X11 forwarding from Windows.

Access from a Mac

From a Mac, you need to install Xquartz instead of Cygwin to display X windows.

www.xquartz.org

Reboot after installing. Then click on xquartz to open a terminal. Type

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

(or whatever IP address you were assigned.)

There's a risk that your spam/junk filter will catch the message; try looking there if it doesn't show up.

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 -D 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 just left of the main label and choose Break to set a breakpoint at the start and another at done. 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 watch the content of memory, use the Data -> Memory function. Display 12 hex bytes starting from the base address &array to show your sorted values.

If you are a command line person, you can use gdb instead.

When you are all done, gracefully power down the pi using

```
sudo shutdown now
```

Clean Up After Yourself

Plug the keyboard, mouse, and monitor back into the Windows computer at your station and make sure it still boots! Put your HDMI to DVI cable back into the supply cabinet.

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

done:NOP            // dummy instruction for breakpoint
      BX LR         // return from main

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

            … more data here

.end
```