CS160 ARM Programming Assignment 1

ARM TOOLS

Problem description

This lab is an introduction to the ARM tools including `armasm`, `armcc`, `armlink`, and `armsd`. You will also be walked through some basic ARM assembly language topics.

You will need the ARM Command Line and Assembler Workbooks. You are expected to have read Sections 3.1 through 3.1.2 and Appendix B from the Hamacher textbook, which you will need to bring. The ARM Instruction Set Quick Reference Card handed out in class will also come in handy.

The assignment is simply to do the exercises in the workbooks.

A few things that are good to know

- When multiplying (e.g., `MUL Rd, Rm, Rs`), `Rd` and `Rm` cannot be the same register. A simple way around the problem is to switch `Rm` and `Rs` because `Rm` $\times$ `Rs` = `Rs` $\times$ `Rm`.
- When `Oprnd2` is an immediate value, you can only specify numbers between 0 and 255 explicitly. Numbers greater than 255 must be shifted to be divisible by 4. You can thus not specify constants such as 299 and 402.
- The command `LDR Rd,=constant` allows you to load a 32-bit number via a literal pool – see the manual.
- Function arguments are passed via the registers (`r0` through `rN-1` for N arguments).

Machines to use

The current version of the ARM assembly tools are compiled for Solaris, not Linux (used on the Hydra machines). To get around this problem, you will need to login to a Solaris machine by typing `ssh compname` where `compname` is a machine running SunOS. The following machines are available for use: `virgo`, `leo`, `libra`, `katana`, `claymore`, and `rapier`. You may use other Solaris machines as well (if you know of any). To check if a given machine is running Solaris, SSH into the machine and type `uname` – you should get `SunOS`.

Check with your neighboring classmates to see what machines they are using so that the connections are evenly distributed between the six machines mentioned above. In other words, we don't want everyone logged into the same machine.

A few machine-related tips:

- If you try to run ARM tools from a non-Solaris machine, you will get a "command not found" error message from your shell.
- When you get to Exercise 1.8 in the *ARM Command Line Workbook*, you will use `armlib.32l` – not `armlib.32b` as the machines you are using are little-endian.

Alternate ARM setup instructions

The lab manual from Graphic Creations, *ARM Command Line Workbook*, contains instructions on setting up your account so that the ARM tools work properly. Specifically, these are under "Introduction: Pre-requisites". The instructions are geared for those of you who use C-Shell; however, most of you are using a newer shell – Z-Shell. If this is the case, use the instructions in this document to get everything working correctly. If you need help with this process or experience problems while following the steps, don't hesitate to ask your GTA for assistance.

Adding to your path

Go to your Z-Shell files directory (`cd ~/.zsh_files`). Edit the file `.zsh.path` by adding the following line: `export PATH="$PATH:/pkgs/arm202u/bin"`.

Adding an environment variable

Go back to your home directory and edit the file `.zshrc`. Scroll down until you see several `export` statements. Within that group add the following line: `export ARMLIB="/pkgs/arm202u/lib"`. Next, you'll need to have the shell reread the file so that the `export` line and the new path will take effect. Do this by typing `source .zshrc`.

Running the setup script

To copy the files needed for this lab, simply run the setup script by typing `~cs160/bin/armsetup.pl`. This script will create a directory called `arm` within your `cs160` directory. The manual references various files that will be in this subdirectory (`~/cs160/arm`).

Due date and submission policy

You will submit the result of the last exercise (7.3) from the *ARM Assembler Workbook* – `block2.s`. Submit this file to your lab GTA no later than 5:00pm on Sunday, November 5, 2006. Programs submitted late will not be graded, instead a score of zero will be assigned.