CS-382 Computer Architecture and Organization Fall 2022

Lab 4 · Hello Assembly!

Lecturer: Shudong Hao *Date: See Canvas*

In this lab, we're going to write and execute assembly programs for the first time.

1 Task 1: Install QEMU Emulator

Recall that assembly programs are closely related to the hardware platform they are performed on, so an assembly program written in ARM syntax cannot be executed on any other machines. Most of the time, however, we also want to execute ARM assembly on some other machines. In this case, we can install an **emulator** that can simulates the hardware execution of ARM machines on any type of machine. This is called **cross-compilation**. The emulator we are using is called QEMU.

To install QEMU on your virtual machine, type the following in your terminal:

```
$ sudo apt-get install qemu-user
```

Once you have finished the installation, you need to write the simplest ARM assembly program (the one listed in B.1.1 in textbook), and assemble and execute it. Upon success, there should be nothing printed out — no warnings or any type of output. To learn how to assemble, link, and execute an assembly program, read B.2.1 in textbook.

To get credits for attendance, show your CA that you have installed QEMU, and can assemble/link/execute the simplest ARM program. No submission needed.

2 Task 2: Write a Simple Program

In this task, you'll need to write a complete program to calculate the dot product of two vectors in assembly. You should write this program in a `.s` file, and you should be able to assemble/link/execute it using QEMU emulator without any warning or error. For now we haven't learned how to print things out to the terminal, so you don't need to print out anything, and the CAs will check your code manually.

The code listed in B.1.1 in textbook is your starter code. Please learn how to declare data and load address by reading B.1.3 in textbook before starting this task.

You can assume the vectors are of `long int` type, and the length can just be a small number, *i.e.*, 3. You should store the dot product into a variable. There's no need to use loops; you can just hard code the offsets for now.

The `.data` segment can be declared as follows:

```
.data
vec1:  .quad   10, 20, 30
vec2:  .quad   1,  2,  3
dot:   .quad   0
```

where `vec1` and `vec2` are two vectors, and `dot` is where we store the dot product result.

3 Requirements

- Task 1 is for attendance, so you have to finish it in the lab. No submission needed;
- For task 2:
  (1) **Note** your code is a **complete** assembly program (not just a sequence of instructions). It should be able to assemble, link, and execute without error and warnings. When executed, the program should finish without problems (also without any outputs);
  (2) If your code cannot assemble, you get no credit – this is the same as C programs that cannot be compiled;
  (3) You can use the `.data` segment provided above for coding. Since we haven't learned loops, we will not use our own tests. As long as your code works for the example above, you're good;
  (4) `MUL` instruction can only used for multiplications between the elements from two vectors, not offset calculation;
  (5) You have to put comments on each line. Without comments your code will be penalized heavily;
  (6) Put your name and honor code pledge at the top of your code in comments.

Deliverable

Submit a single `.s` file.