ARM Assembly - getting started

Before you get into programming in assembly, it will be useful to configure your environment. If you do not have an ARM processor on your machine, then you can either connect to a lab machine or install the emulator and debugger on your own machine. I recommend using your own machine because you can install features not available on the lab machines to help you. The instructions in this sheet will work for Ubuntu and WSL.

GDB with/without GEF

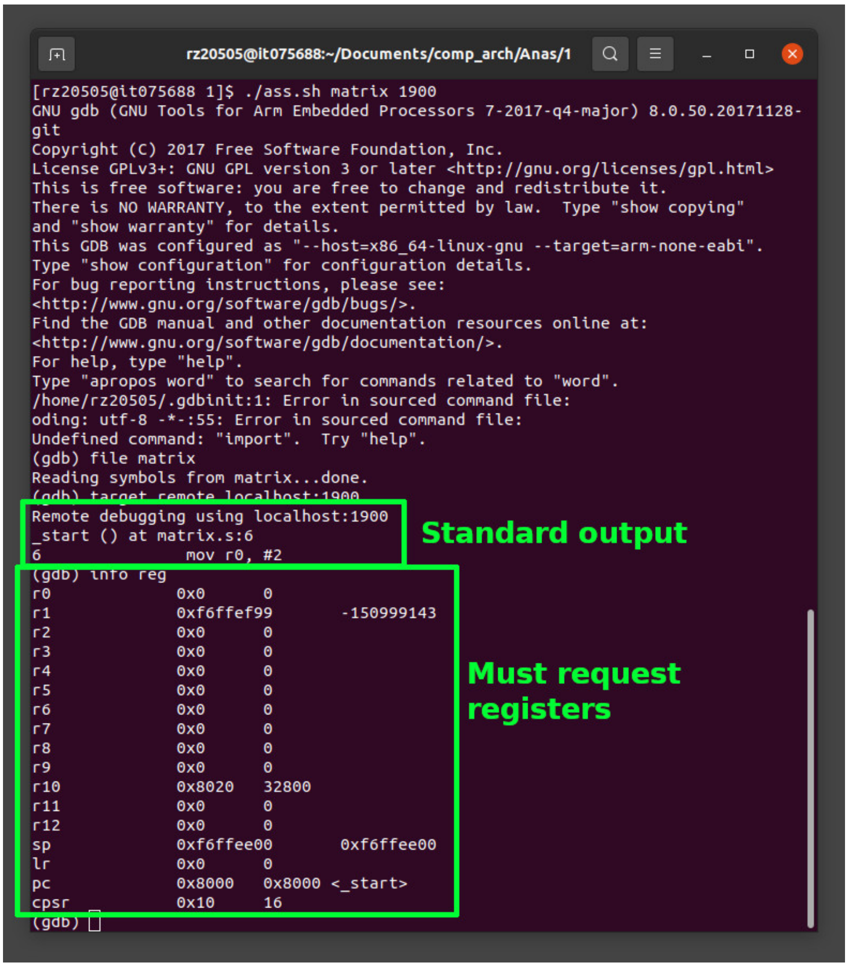

Figure 1: Standard GDB on lab machine

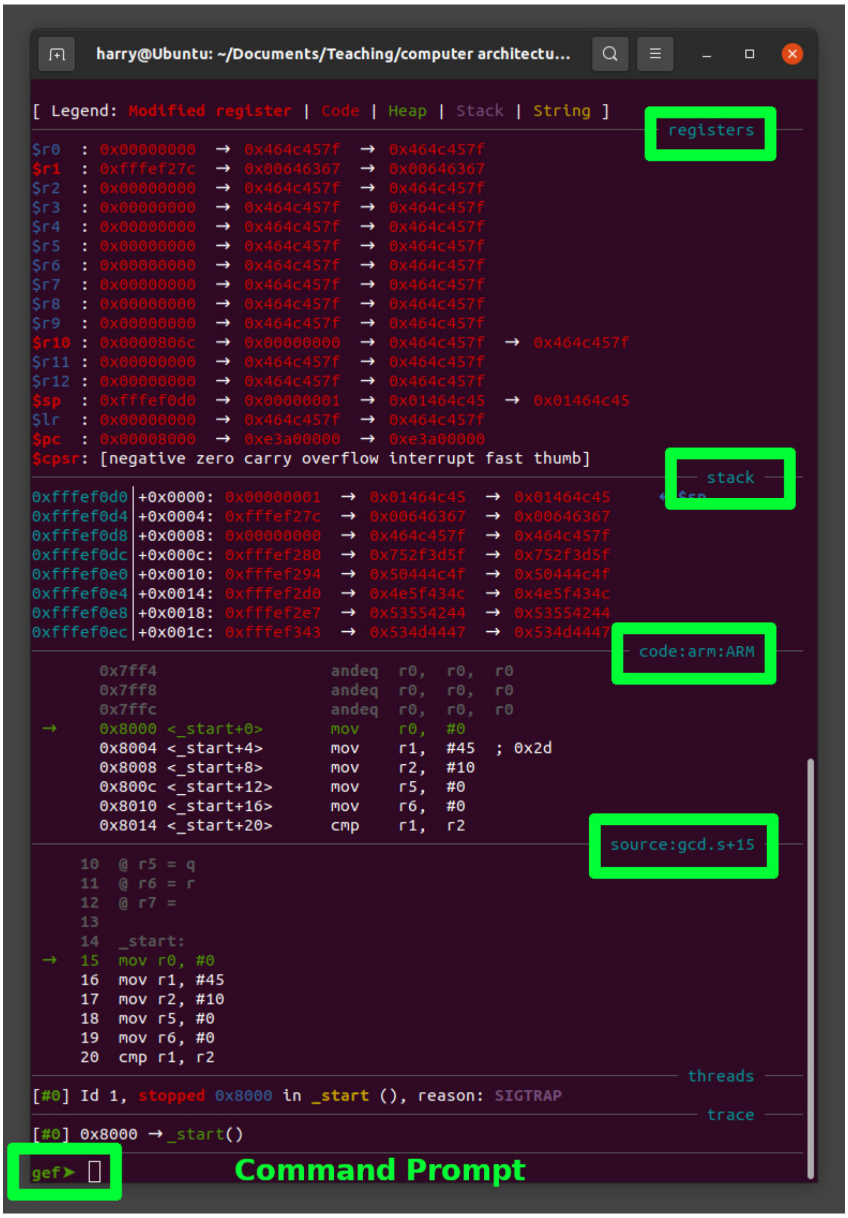

Figure 2: GDB with GEF in Ubuntu/WSL

As you can see from the interface installed locally, the registers, stack, cpsr flags and code are all visible as we step through the program. This gives us a more comprehensive view that we can use to our advantage while debugging programs.

Installation for Ubuntu and WSL

Run the following commands:

```
sudo apt update && sudo apt upgrade
sudo apt install qemu
sudo apt install binutils-arm-none-eabi gcc-arm-none-eabi
sudo apt install gdb-multiarch
sudo apt install qemu-user
sudo apt install curl
wget -q -O- https://github.com/hugsy/gef/raw/master/scripts/gef.sh | sh
bash -c "$(curl -fsSL http://gef.blah.cat/sh)"
```

Everything we need is now installed!

Compiling and executing your program

With assembly, we have to run the following 3 commands in order to assemble our code and start the debugger: (Step 3 will produce a warning when using WSL. Give permission to access local/private networks but not public ones.)

```
arm-none-eabi-as -o <FILENAME>.o -g <FILENAME>.s # Create object
arm-none-eabi-ld -o <FILENAME> <FILENAME>.o      # Create executable
qemu-arm -singlestep -g <PROTNUM> <FILENAME> &   # Start qemu
arm-none-eabi-gdb                                # Start gdb
```

2 scripts are provided with this worksheet. au.sh is intended for people running an Ubuntu system or VM, and al.sh is for people using the lab machines. These files do all of the above steps for you. Think of this like a makefile in C, in that it needs to be in the same directory as the file you want to assemble. With this script in the same directory as your assembly file, the command to run the script is:

```
./au.sh <FILENAME> <PORTNUM>    # For Ubuntu
./al.sh <FILENAME> <PORTNUM>    # For lab machines
```

The port number is included because you often need to change it when you reassemble and restart the debugger. The filename should not have any '.s' or any other appending characters. If your file is called tst.s, then the command might be

```
./au.sh tst 1234
```

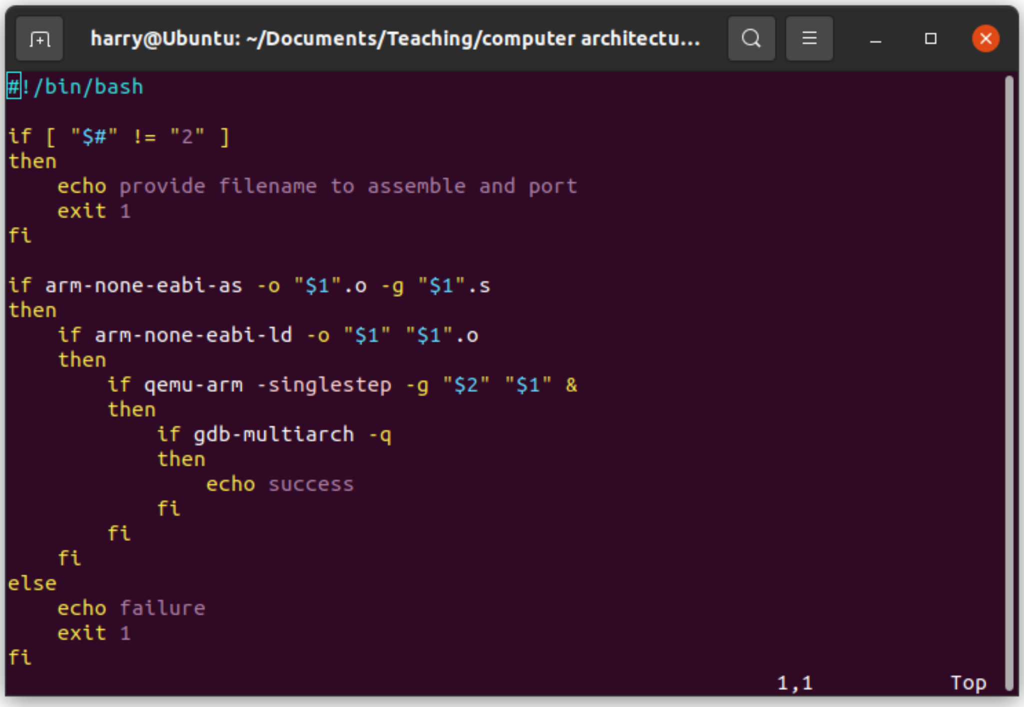

Figure 3: Script to automate assembling code and starting debugger