Getting Started: ARM Assembly for Android

I wanted to play with some ARM over the weekend, but unlike x86, I don't have an arm development environment. I'll need a way to compile arm binaries within x86 and then test them out on an ARM device. One easy (and free) solution for this is to write the assembly in linux, cross compiled for ARM, and then test the code inside the Android Emulator. The Android Emulator is a full emulator, and so it not only gives developers the ability to test their Java code, but also any native ARM code they developed for Android.

Steps:

1. Setting up the Development Environment
2. Writing helloarm assembly file
3. Assemble and Link helloarm into executable
4. Test helloarm in the Android Emulator

Setting up the Development Environment

Because I want this tutorial to be about getting started quickly with writing ARM Assembly for Android, the most convenient way for everyone to be on the same page is use the **Android Reverse Engineering (ARE) Virtual Machine** from the Honeynet Project. Following the link will give you instructions for downloading the Virtual Machine and running it inside **VirtualBox**. If you don't want to use a Virtual Machine, you can follow the guides on the developer.android.com to install the **Android SDK** and the **Android NDK**.

Writing helloarm assembly file

The assembler we'll be using is the **GNU Assembler**. The syntax of the assembler allows for commenting and use of directives. Here is the helloworld.as program we'll be using:

```
        .syntax unified
.data
message:
        .asciz "Hello, world.\n"
len = . - message
.text
        .global _start
_start:
        mov     r0, $1
        ldr     r1, =message
        ldr     r2, =len
        mov     r7, $4
        swi     $0

        mov     r0, $0
        mov     r7, $1
        swi     $0
```

Assemble and Link helloarm into executable

Note: I left the full paths of the assembler(as) and linker(ld) to avoid confusion, but in the future, you'll probably want to create soft links or add them to your PATH.

```
ebuilt/linux-x86/bin/arm-linux-androideabi-as -o helloworld.S helloworld.s
ebuilt/linux-x86/bin/arm-linux-androideabi-ld -o helloworld.exe helloworld.S
```

For both tools, the -o parameter is the output file, and the last parameter is the input file.

Test your new executable on the emulator

First, start the emulator. Be sure to include the "&" so that the program runs in the background.

```
$ emulator -avd Android21 &
[1] 16467
```

Hit enter in the terminal after the emulator pops up, this will restore your prompt. Wait for the emulator to finish starting up (this could take a few).
Use adb to create a new test folder, and push the executable into it.

```
# adb -e shell will let you execute any command on the device
abd -e shell "mkdir /data/data/test"
adb -e push "helloworld.exe /data/data/test"
# need to set permissions for execution
adb -e shell "chmod 777 /data/data/test/helloworld.exe"
# lets try to execute
adb -e shell "/data/data/helloworld.exe"
Hello, world.
```

References

- **Hello World in ARM Assembly**
- **Hello World In (ARM) Assembly**

Posted by alex on November 3, 2012

http://www.amccormack.net/getting-started-arm-assembly-for-android/

☺