**ECE 212: Microprocessors, Hardware, Software, and Interfacing**

# Lab 1: Getting Started with DE1-SoC board

Karysse Hay
*College of Engineering*
*University of Miami*
Miami, United States
kdh85@miami.edu

***Abstract*— ARM assembly language is a low-level language that many professionals in the engineering field use. And so, to better understand specific components of ARM programming, this lab aims to teach students how to manipulate strings and memory addressing in assembly.**

In the previous lab, the development board and the ARM processor was explored. Additionally, a sample program was successfully installed onto the ARM processor and the outcome was observed.

***Keywords— ARM, processor, memory, manipulation***

## I. CHALLENGE

The objective of this lab is to learn about the ARM processor on the DE1-SoC development board and how to compile, install, and debug assembly programs. The operation of the emulator environment, reading of the memory content, and debugging of assembly programs will also be covered.

## II. APPROACH

Individual programs will be written, and a unique message will be placed in memory. This program will be modified to satisfy the lab requirements. The information learned in Chapter 2 (GNU Assembly Syntax) and the tools provided by the Altera/Intel Monitor Program (AMP) will assist in the debugging and/or navigating and the explaining of the program.

## III. EXPERIMENT SETUP

A new project was created in AMP. The directory was selected, and the project name was typed in the "ARM Cortex A9" architecture and the "ARM Cortex-A9 System" was selected. The program was assembled and a text file (MyName.s) was created and added to the program. The host connection was verified to be set to "DE-SoC [USB-1]" and the Linker Section Presets was set to Basic.

### *A. Final Code*

```
. global _start
_start:

.data

.align 4
x: .byte 'E', 'C', 'E', '2', '1', '2'

.align 4
y: .hword 'E', 'C', 'E', '2', '1', '2'

.align 3
z: .word 'E', 'C', 'E', '2', '1', '2'

.text

ldr r1, =x
ldr r2, [r1]

.end
```

## IV. RESULTS AND ANALYSIS

The initial program saved "Hello, my name is Karysse Hay." in the memory address 0x00000018. At first, the Memory tab was not set to 'Show ASCII equivalent characters' nor 'view as byte' but after that was done, Figure I was visible. In the "Info & Errors" section, the program stopped at address 0x0000009C.

After, the string "ECE212" was typed as a .byte, .halfword and .word characters. Three labels were used: x, y, and z. This can be seen in Figure II. Additionally, all three strings of characters were aligned so that they would sit on different lines. Three align directives were used to accomplish this and can be seen in Figure III. In total, six directives were needed for this program: .data, .byte, .hword, .word, .text and .global.

The project was disconnected and another directive, ".data" was added under the memory options. The starting address of this directive was set to the value of my C number: C23722255. However, this number was not divisible by 8 and had to adjusted so that an error was not received. This number eventually became 0x0169F920. The ending address range of .text also had to be adjusted so that it would not overlap the range of .data. The information eventually saved at the memory address mentioned and this can be seen in Figure IV. The difference between this section and the previous one was that the starting address was preset manually and the memory address ranges of the directives: .data and .text were altered.

In the final section, the first word was loaded into the "r1" register. This was done by using the 'ldr' instruction. The memory address for r1 then changed from 0xFFFF139C to the starting address of the previous section which was 0x0169F920. This can be seen in Figure V. Four values in the register

changed: PC, r1, r2, r5, r10, SP, LR and CPSR. The PC value changed from 0x00000000 to 0x00000048.

## A. *Figures and Tables*

*a)*

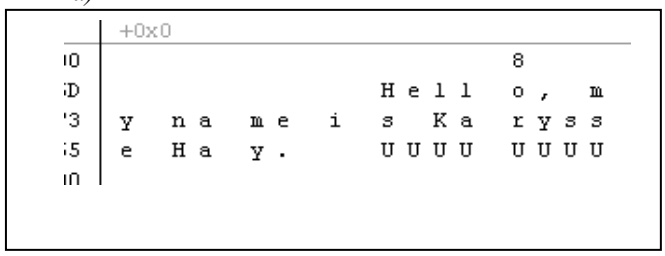


FIGURE I. PART I A

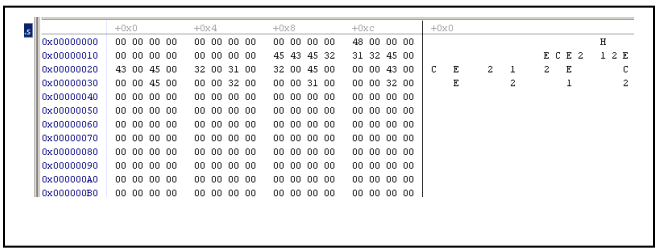


FIGURE II. PART I B (I)

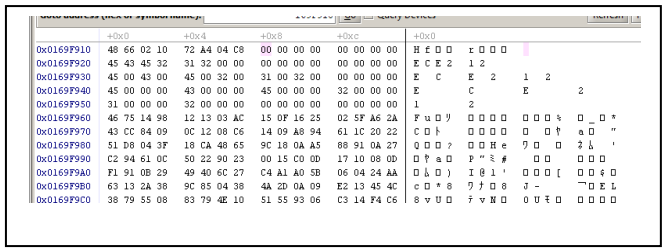


FIGURE III. PART I B(II)

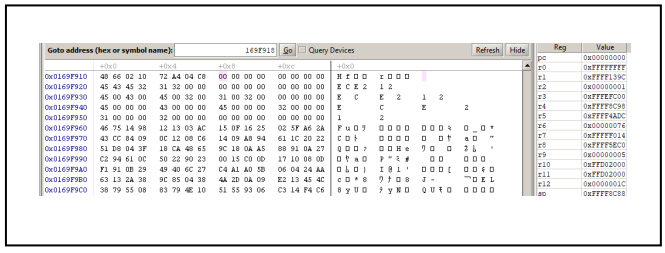

FIGURE IV. PART II

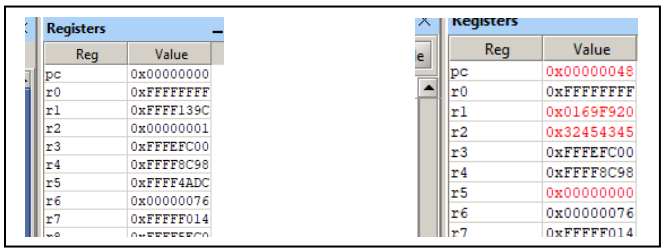


FIGURE V. PART III

## V. CONCLUSION

In this lab, we were able to become more familiar with directives, labels and memory addresses and apply them to theory learnt in the classroom. We changed the starting points of certain directives and altered the memory address ranges for others. Changes in memory address for the registers were also monitored an recorded after the starting address was changed.

## REFERENCES


[1] A. Holdings, "Cortex-a9 technical reference manual, revision r2p0," ARM Holdings. Report number: DDI0388E, 2009.

[2] ARM Ltd and ARM Germany GmbH, " Writing ARM Assembly Language," Assembler User Guide: Writing ARM Assembly Language. [Online]. Available: https://www.keil.com/support/man/docs/armasm/armasm_dom1359731144635.htm. [Accessed: 20-Mar-2021].

[3] "ASCII," Wikipedia, 19-Aug-2020. [Online]. Available: https://simple.wikipedia.org/wiki/ASCII#/media/File:ASCII-Tablewide.svg. [Accessed: 20-Mar-2021].