Microprocessor-Based Systems (E155)

Lab 4: Sort in ARM Assembly

Learning Objectives

By the end of this lab you will have…

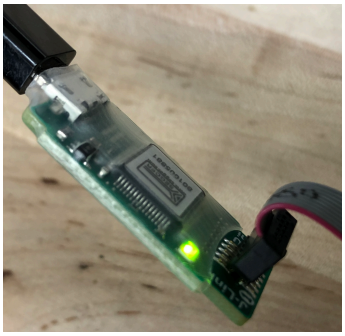
- Gained familiarity with writing and debugging ARM assembly code
- Refreshed your understanding of the Keil Microcontroller Development Kit (MDK)

Requirements

Write an assembly-language program to sort an array of 12 signed bytes on the ATSAM4S4B

Setting up the Programmer

Pick up a J-Link EDU Mini Programmer from the MicroPs Cabinet. Plug the JTAG ribbon cable directly into the port on your µMudd 5.1 board labeled JTAG – SAM. The socket on the µMudd board is polarized so you can only insert the cable one way. The socket on the J-Link is not polarized. Insert it with the cable running toward the middle of the J-Link board. This allows you to program the microcontroller directly. Power your board with the power supply as you have done in previous labs.



The J-Link EDU Mini is inexpensive so you might wish to buy your own from AdaFruit to avoid the crowds in the lab.

Setting up Keil

If you want to run on your own Windows laptop, you can download and install the free Keil MDK-Lite-ARM tools.

1. Download the tools from https://www.keil.com/demo/eval/arm.htm. You will be asked for your contact information, and will also need to indicate that you are using the SAM4S family.

2. Install the device drivers for the SAM4S series microcontrollers on your computer. Download the Device family pack for the SAM4S series from http://www.keil.com/dd2/microchip/atsam4s4b/
3. Launch **Keil µVision** from the start menu.
4. Choose **Project → New µVision Project** and name the project '**SortSAM'**. Save it in a new folder in your Charlie Directory.
5. In the '**Select Device for Target'** window, navigate to, or simply search for, the ATSAM4S4B microcontroller (**Microchip → SAM4 Series → ATSAM4S → ATSAM4S4B (dep…)**). Select this device, and click **OK**.
6. In the '**Manage Run-Time Environment'** window, expand '**CMSIS'**. Select the checkbox in the '**CORE'** row, then click **Resolve**, then **OK**.
7. Expand **Target 1** in the left-hand **Project** pane. Select it, then navigate to **Options for Target 'Target1'** under the **Project** menu (or press ALT+F7).
8. In the '**Options for Target 'Target 1''** window, under the **Debug** tab, open the dropdown menu by '**Use:'** and choose '**J-LINK/J-TRACE Cortex**' as your programmer. Then click **'Settings'**.
9. In the **'Cortex JLink/JTrace Target Driver Setup'** window, under the **Flash Download** tab, select **'Erase Full chip'**. Click **OK**, then **OK** again. This fully rewrites the program and data memory every time you build your code onto the microcontroller.

Writing the Code

1. In the Project pane, right-click on **'Source Group 1'**, and select **'Add New Item to Group 'Source Group 1''**. Create a an assembly file **('Asm File'**) named **sort.s**.
2. Download sort.s from the E155 web page or paste the code found in Appendix A in the sort.s file. This assembly program starts by loading an array of decimals into the data memory of the microcontroller and handles the code execution such that you can build a standalone assembly file without a .c file to run it, as you did in E85.
3. Finish writing the assembly language program to sort 12 signed bytes (the ones on the "ARR DCD…" line). Remember that assembly language code is nearly unreadable without line-by-line comments. If needed, use online reference sheets for ARM Thumb2 Assembly language or refer to Chapter 6 of *Digital Design and Computer Architecture.*

Running and Testing your Code

1. With the programmer and µMudd board set up, choose **Build Target** under the **Project** menu (or click the build icon on the top panel; or press F7). Scroll through the **Build Output** panel at the bottom to look for warnings and errors.
2. Next, click the **Load** icon next to the **Build** icons in order to download the program onto the device's memory (or press F8).
3. Navigate to **Debug → Start/Stop Debug Session** to invoke the Debugger (or press CTRL+F5). Under the **View** menu, select **Memory Window**. Within the memory window, you can choose to view the decimal values of the bytes in memory by right -clicking, which makes debugging much easier. Step through the program with the step icons on the top panel. You will be able to see the values in the registers in the left-hand window change as you step through the program, and you can search for the address you want to look at in

the bottom-right-hand memory window, which displays all the values in the program's memory. This information is crucial for debugging.

4. To test your code, try various cases with the array in the "ARR DCD…" line. Rebuild and start a new Debug Session every time you make changes. To view the actual values in the array (and watch them change when your assembly code works) search for the 0x20000000 address in the Memory Window, which is where the given assembly code stores the array.

Clean Up After Yourself

Return the J-Link programmer to the cabinet and clear your workbench.

What to Turn In

- A listing of your program that sorts numbers.
- Explicitly state the test cases you used and the output of the tests on each program. Be sure your tests would convince a skeptic that your algorithm works.
- How many hours did you spend on the lab? Any comments, suggestions, or complaints about the assignment? This will not count toward your grade.

Credits

This lab was originally developed by Professors David Harris and Matthew Spencer in 2017 and redesigned for the μMudd Mark 5.1 by Tejus Rao '22 and Kaveh Pezeshki '21 in Spring 2019.

Appendix A: Starter Assembly Code

```
; sort.s
; E155 Lab 4 Starter code 2019
;
; Reset handler copies an array from program memory into data memory where it can be
sorted
;

;Directives
       THUMB ;This is the variant of ARM Assembly language that the ATSAM4S4B uses

;Exporting _Vectors for linker
       AREA RESET, DATA, READONLY
       EXPORT __Vectors
__Vectors
       DCD 0x20001000 ;stack pointer when stack is empty
       DCD Reset_Handler
       ALIGN

; Reset_Handler is called when code starts
       AREA MYCODE, CODE, READONLY
       ENTRY
       EXPORT Reset_Handler

Reset_Handler
       B loadarray
loadarray ;moves array from program memory to data memory
       LDR R3, =ARR ;load base address of array in program memory into R3
       LDR R4, =0x20000000 ;load new base address of array into R3, in data memory
       MOV R6, #0 ;current element index to be copied
loaditeration ;moves a single element of array from program -> data memory
       CMP R6, #12 ;if R5 >= 12, array has been copied, so jump to sort
       BGE sort ;once the array has been copied, sort the array
       LDR R7, [R3, R6, LSL #2] ;load first element of arr into R7, left shifting as
word-addressing
       STRB R7, [R4, R6] ;storing this element in data memory
       ADD R6, R6, #1 ;incrementing index
       B loaditeration

sort
       LDR R3, =0x20000000 ;load base address of array into R3
       ...YOUR CODE HERE
done
       B done
;creates an array of decimals in the order specified (separate values with commas)
ARR DCD 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1
; replace this with your test array YOUR TEST ARRAYS HERE
       ALIGN
       END
```