Lab Time_____ Name_____

Lab 1b. ARM Assembly and Keil Debugging

Preparation

You will need a LaunchPad and a laptop/computer with Keil uVision5 installed. Download the Lab 1b starter project from Canvas.

Starter project Lab 1b

Purpose

The purpose of this lab is for you to become familiar with using the debugger in Keil. Additionally, you will gain exposure to ARM assembly programming in the form of arithmetic operations.

Introduction

In this lab you will learn some of the features that are available within Keil for debugging your programs. In the latter portion of the lab you will finish a program by writing three functions. The correctness of each function can be tested by holding down a specific button combination. Each test will either blink a specified LED if correct, or blink the red LED if it failed. A correct example test will run when no buttons are pressed.

Procedure

1. Let X, Y, and Z be three numbers as shown below. In the project provided for this lab write assembly instructions to perform the operations listed in the table below. After each operation record the Result (R) and the states of condition flags N, Z, V, and C in the PSR register. You will need to set breakpoints and use the debugger to observe these values. Figure 1 shows how this can be done in Keil.

X = 0x0000 0008, Y = 0xFFFF FFFD, Z = 0x7FFF FFFF

Operation	Result (R)	Condition Codes N Z V C
R = X + Y	0x0000 0005	0 0 0 1
R = X – X	0x0000 0000	0 1 0 1
R = X + Z	0x8000 0007	1 0 1 0
R = Y – Z	0x7fff fffe	0 0 1 1
R = Y + Z	0xffff fffd	1 0 1 0

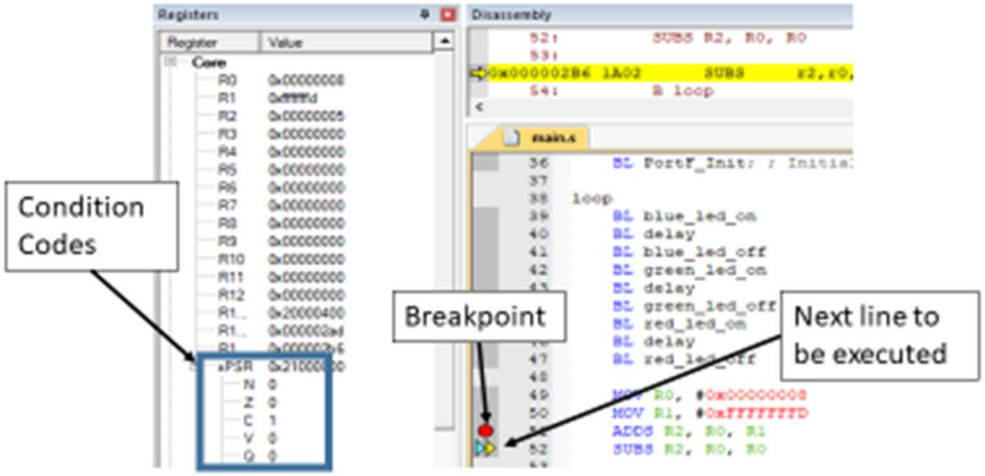

Figure 1: Debugging in Keil

All of the debugging commands can be found under the **Debug** menu in addition to on the toolbar. Breakpoints can be added and removed by selecting **Insert/Remove Breakpoints**. You can start or stop the debugger by going to **Start/Stop Debug Session** under the same menu. Once in the debugger you can use **Step** to execute one line at a time or **Run** to execute the program until it hits a breakpoint. **Stop** will halt a running program and can be useful when trying to determine why your program never reaches a desired breakpoint.

The code for this part and part 2 can be inserted after the "debug" function label in main.s. Fill out the table and demonstrate using the debugger to the TA. Explain to the TA what the N, Z, C, and V flags stand for.

Signature Carl Sorrell Date 2/14/20

2. Next you will use the Memory Window to study the various forms of the LDR instruction. The Memory Window can only be accessed by going to **View → Memory Window** and choosing a window. **Memory windows are only available while using the debugger**. In the memory window, put in the address 0x20000000 (beginning of RAM space) as shown in Figure 2. The data found there was set by some initialization function. Fill in the table below with the values loaded into R6 when using each of the various forms of the LDR instruction.

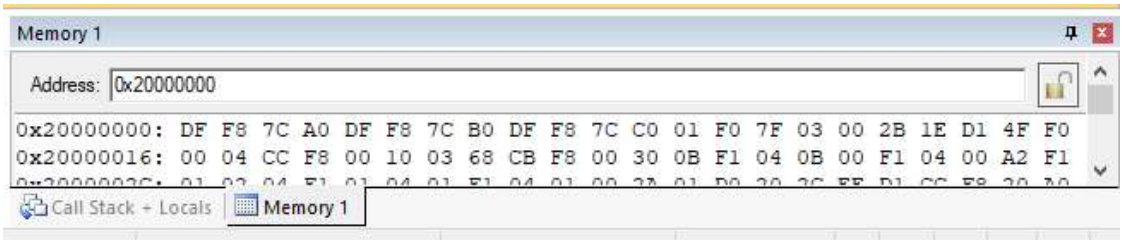

Figure 2: Memory Window

First run:
MOV R7, #0x20000000
 	
Then:
LDR R6, [R7]
R6 = a07c f8df
 	
LDRH R6, [R7]
R6 =0000 f8df
 	
LDRB R6, [R7]
R6 =0000 00df
 	
LDRSH R6, [R7]
R6 =ffff f8df
 	
LDRSB R6,[R7]
R6 = ffff ffdf
	First run:
	MOV R7, #0x20000000
	ADD R7, #2
	
	; Must use add because 0x20000002 cannot be represented as
	; an immediate value
	
	Then:
	LDR R6, [R7]
	R6 = f8df a07c
	
	LDRH R6, [R7]
	R6 =0000 a07c
	
	LDRB R6, [R7]
	R6 = 0000 007c
	
	LDRSH R6, [R7]
	R6 = ffff a07c
	
	LDRSB R6,[R7]
	R6 =0000 007c

The code for this part can be inserted in the same location as the code for part 1. Show the completed table to the TA. Be sure you are able to explain to the TA what each variation does and why you got the results you did. You should be able to relate your results to the information displayed in the memory window.

Signature Matthew Russell Date 2/14/20

3. In the file main.s fill in the code for "Part3_Function". Your code will implement the arithmetic operation Z = (A<<2) | (B&15). This code is tested by holding down SW1. A successful test of your code will result in the Blue LED blinking. Incorrect code will result in the Red LED blinking. **Be sure to read the comments above the function!** Show your correct code to the TA.

Signature Cole Farrell Date 2/14/20

4. Next fill in the code for "Part4_Function". Your code will implement the arithmetic operation Z = ((A+B)+(A-B)) | (A<<B). This code is tested by holding down SW2. A successful test of your code will result in the Green LED blinking. Incorrect code will result in the Red LED blinking. **Be sure to read the comments above the function!** Show your correct code to the TA.

Signature Carl Farrell Date 2/4/20

5. Lastly, fill in the code for "Part5_Function". Your code will implement the arithmetic operation Z = X + Y where X = ((A*8)|(B*4)|(C*2)) and Y = ((B xor 2)|(C xor 4). This code is tested by holding down both SW1 and SW2. A successful test of your code will result in the Blue and Green LEDs simultaneously blinking. Incorrect code will result in the Red LED blinking. **Be sure to read the comments above the function!** Show your correct code to the TA.

Signature Matthew Russell Date 2/4/2020