Lab 2: Arithmetic and Logic Instructions

EE 5385

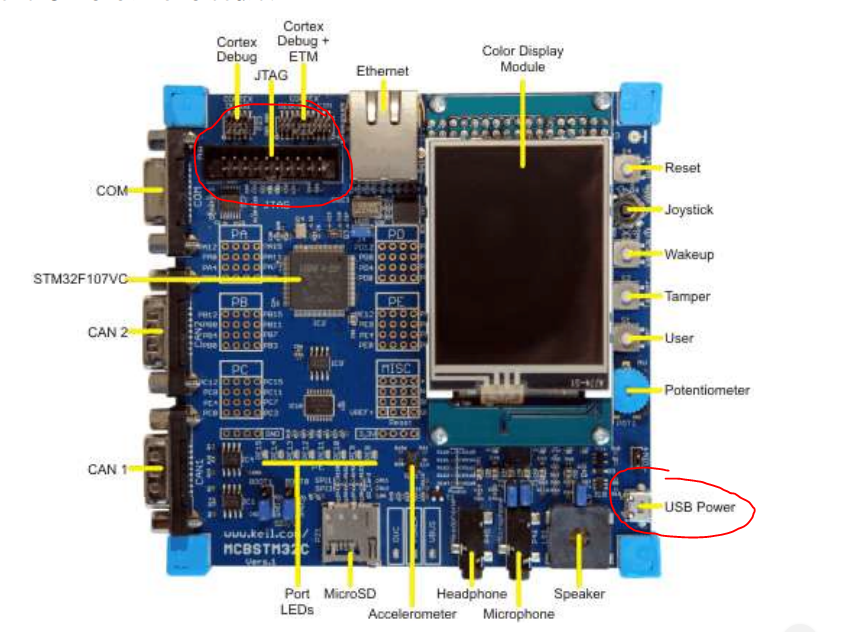
Lab Report

3/2/18

Victor Yuan

Turn on and setup (same as previous two labs)

First, I setup the board like the first lab by connecting an USB cable with an ULINK-ME adapter. Then I connect this adaptor to the JTAG pins in the figure below:



Then I connected the USB to the USB power port on the bottom right of the figure. Connect both ends of the USB cables to the computer one two power the board and one to program the board.

Then I used the program Keil uVision, first I created a new project from scratch. I selected the microcontroller vendor and device in the device database but did not use the system’s default startup code since we were provided with a custom startup file. A created a file and named it logic.s and added this file to the project along with the startup file given to us through this link “http://goo.gl/BB3mnx”.

Part 1

Type in the code line by line with comments on how each instruction works and what it does as shown below:

AREA LOGIC, CODE, READONLY

ENTRY

ldr r0, =0x00000018

ldr r1, =0x00000010

ldr r2, =0x00000000

ldr r3, =0x00000000

ldr r4, =0x00000000 ;load the registers with the values provided

add r2, r0, r1 ;adds 0x10 to 0x18 and puts the value in r2

sub r3, r1, r0 ;r1-r0 = 0xfffffff8 = -8

mul r4, r0, r1 ;multiplies 0x18 and 0x10 and stores in r4

;r4 = 0x180

add r4, r2, r3 ;adds the value in r3 with the value in r2

;r4 = 0x28 + -0x8 = 0x20

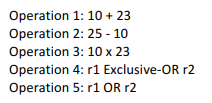
stop B stop

END

To find the values of these registers save your logic.s file and build it once there is no errors open the debug tool and look at the contents of each register. Step through each instruction line by line by pressing f11 on the keyboard.

Part2

Make another file and add it to the project or simply comment out the first part of the program and write the assembly instructions that correlates with the following arithmetic operations.



AREA LOGIC, CODE, READONLY

ENTRY

;Operation 1

mov r0, #10

mov r1, #23

add r2, r0, r1 ;10+23

;Operation 2

mov r0, #25

mov r1, #10

sub r3, r0, r1 ;25-10

;Operation 3

mov r0, #10

mov r1, #23

mul r4, r0, r1 ;10x23

;Operation 4

mov r0, #10

mov r1, #23

EOR r5, r0, r1 ;exclusive or 10111 and 1010 = 11101 = 0x1D =29

;Operation 5

mov r0, #10

mov r1, #23

ORR r6, r0, r1 ;logic or 10111 and 1010 = 11111 = 0x1F =31

stop B stop

END

The inputs of each operations where set in registers r0 and r1, there were some redundancies in the code that was done to better organize the code. The debugger was used to determine the value of each register r2, r3, r4, r5, r6 contains the result of each operation 1, 2, 3, 4, 5, respectively.

Part 3:

Once again make a new file and add it to the project or comment out parts 1 and 2. This time I am asked to perform r0 = r1 \* 16 and r2 = r3 \* 9 however I am only allowed to use 1 line for both these operations to do this i have to make use of logic shift and add.

AREA LOGIC, CODE, READONLY

ENTRY

mov r1, #1 ;test value of 1 for simple calculations

lsl r0, r1, #4 ;left shift logical 1 by 4 is the same thing as

;multiplying by 2^4 = 1 \* 16 = r1 \* 16

mov r3, #1 ;test value

add r2, r3, r3, lsl #3 ;set all the destination register to r2 as the

;operation requires.

;r3 is logically shifted by 3 which is the same

;as multiplying by 8 then it is added back onto

;r3 effectively making it 9 copies of itself

;which is the same as multiplying r3 with 9

stop B stop

END

Analysis

For this lab I only needed to look in the registers since all results where stored and operated on in registers no values were retrieved or stored to memory, so it was unnecessary.

There was a slight issue I encountered while first setting up the program I wasn’t sure if I was supposed to use RTX kernel or not. So I chose to build upon the configuration setup by lab 1 and 2, I simply removed the .s files in project 2 file and added logic.s to the project. Since the device was the same and so was the custom startup code.

Signatures:

