Lab 5: Status flags and arithmetic instructions

EE 5385

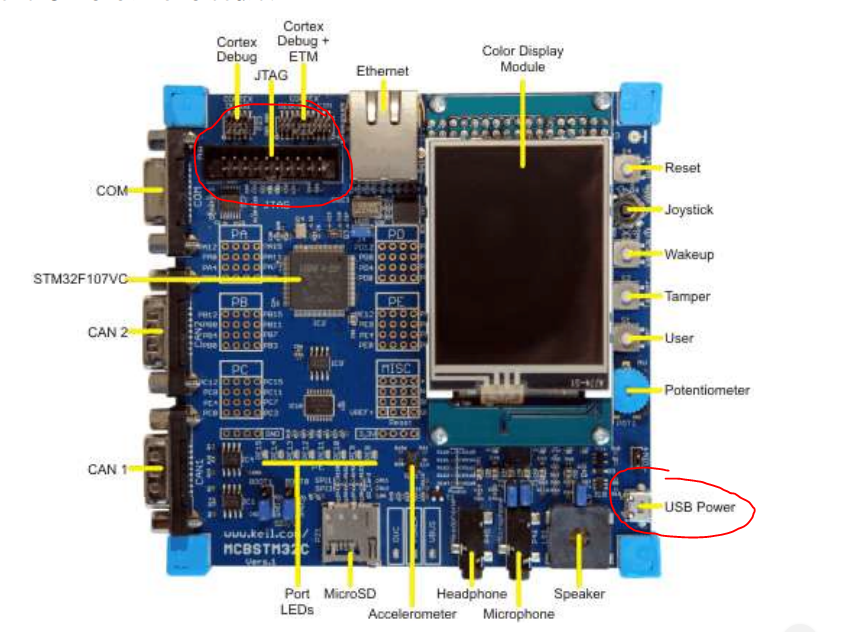
Lab Report

3/23/18

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”.

Step 1:

In step one we entered the given code and looked at the CPSR to see if the program could be ran successfully before the main portion of the project.

Step 2:

In step two we are given three arithmetic adds operations and we are asked to determine how this affects the CPSR flags.

The first operation where A=0xFFFF0000 and B=0x87654321 are added together, the decimal representation of this is -65536 + -2,023,406,815 = -2,023,472,351 this value is still less then -2^31 which is the maximum negative value therefore there is no overflow or carry, the answer is not zero either so the z flag will not be set the only flag that is set will be the N flag indicating that the signed add result was a negative answer.

The second operation where A=0xFFFFFFFF and B=0x12345678 are added together, the decimal representation of this is -1 + 305,419,896 = 305,419,895 there is no overflow no carry and the answer is a nonzero with a positive value so all the flags will not be set.

The third operation where A=0x67654321 and B=0x23110000 are added together, the decimal representation of this is 1,734,689,569 + 588,316,672 = 2,323,006,241 this answer is positive and non zero however it is well beyond the maximum value of (2^31)-1 therefore the overflow and Carry flags are both set.

Step 3:

In step three we are given a signed multiplication problem asking us to multiply the number 0xFFFFFFFF and 0x80000000 and placing the result in a different register then the two source registers. the decimal multiplication of this is -1 \* -2,147,483,648 = 2,147,483,648 this is a positive number and a non-zero number however the maximum positive value is 2,147,483,647 so there is a overflow and a carry flag set. Furthermore, having a different register would not change the flags since the CPSR is not stored on any of the regular registers.

Its important to have two long multiply instructions signed and unsigned, umull is used to preserve all bits in a multiplication storing the 64 bit result in two registers it will be useful in the future in single or double point precision calculations when we have to deal with decimals while smull can perform long multiplication for signed integers much easier

Step 4:

We are asked to write a program that would multiply two 32-bit signed integers for a result of a 64bit answer to be placed in two separate registers. we are only allowed to use UMULL and logical expressions.

Code:

AREA asm\_code, CODE, ALIGN=2

ENTRY

mov r0, #4

mov r1, #-2 ;test values

lsr r2, r0, #31 ;extract signs of each 32-bit value

lsr r3, r1, #31

cmp r2, #1 ;determines if number is negative or not

bne skip2sComp1

eor r0, #0xFFFFFFFF ;XRL assuming XRL is xor

add r0, r0, #1

skip2sComp1

;skips process if the value is positive

cmp r3, #1 ;same as before for 2nd number

bne skip2sComp2

eor r1, #0xFFFFFFFF

add r1, r1, #1

skip2sComp2

umull r7, r8, r0, r1 ;r8,r7 highest value numbers are in r8

;lowest is r7

;determine sign

eor r4, r2, r3 ;r4 contains the sign that the answer needs to

;be in finale result

cmp r4, #1 ;if it is equal, then the answer contained in

;r8 r7 needs to be flipped with a 2's comp so

;that it an negative number

;representation

bne stop

eor r7, #0xFFFFFFFF ;if negative flip everything for 1's comp and

;prepare for 2's

eor r8, #0xFFFFFFFF

ADDS r7, #1 ;add 1 to the 1’s comp for 2’s comp

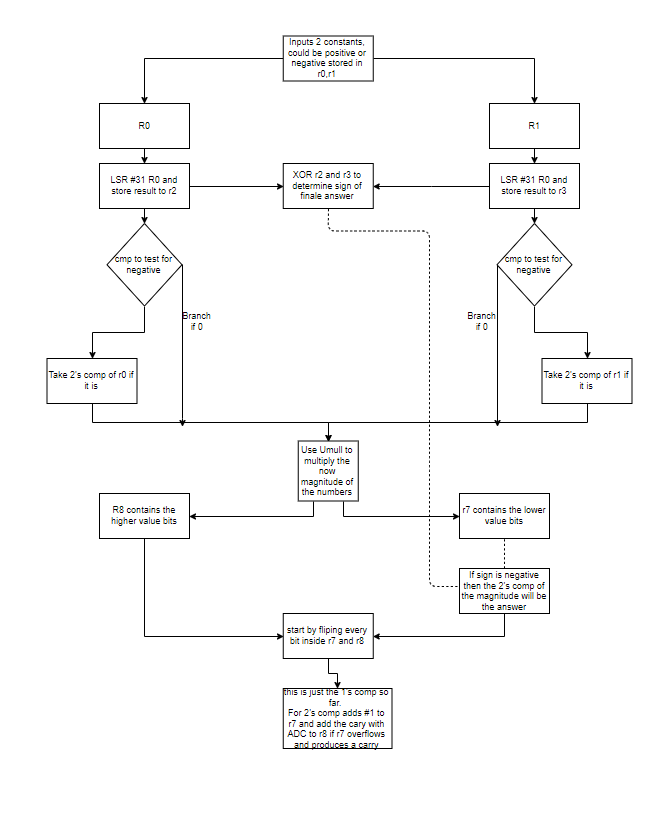
ADC r8, #0 ;if r7 has a overflow and carry then it is

;added to r8 otherwise r8 remains unchanged

stop b stop

END

Flow Chart:



The most notiable issue I had with this lab was with the conditional instructions for some reason they did not work with Eor, so I had to use branches instead

After I finished the lab, I learned I could have simply used MVN to achieve the same thing eor with 0xffffffff.

Additional questions:

Overflow occurs when a arithmetic operation results in a answer greater then +(2^31)-1 or less then -2^31, it means that the register can not hold a signed value of that magnitude.

The carry can occur when with 0xFFFF FFFF + 0x7FFF FFFF or with 0x0000 0001 – 0x0000 0002

- Update the destination register only if the Carry flag C is set

The C suffix can be used ie ADDC SUBC

- Update the destination register only if the Negative flag N is set

The flags NE, Lt can be used IE ADDNE, ADDLT

