ELEC 3300

HOMEWORK 1: INTRODUCTION TO KEIL MDK WORKSHEET

Please complete the following and submit your worksheet electronically before the deadline

Name : _Tseng, Mu-Ruei___ Student number : ___20472522___ LAB Session : ___LA3____

1. From Page 7, **after you added your code and compiled the project**, check the Build Output, record the following: (Note: Make sure your project is complied with Optimization Level 0.)

   `Program Size: Code = ______1332_________`

2. Run your program up to the instruction "`printf("Sum %d\n",sum);`"

   a. Fill the following table. Remember you are using your *OWN* student ID.
   NO MARKS WILL BE AWARDED IF YOU ARE NOT USING YOUR STUDENT ID

Expression	Value	Type
i	0x20472522	int
j	0x22527402	int
stdid	0x20000020 stdid	int[8]
[0]	0x00000002	int
[1]	0x00000000	int
[2]	0x00000004	int
[3]	0x00000007	int
[4]	0x00000002	int
[5]	0x00000005	int
[6]	0x00000002	int
[7]	0x00000002	int
counter	0x00000008	int
sum	0x00000018	int

   b. What is the Value in stdid? _______________ 0x20000020___________________________

   c. What is the meaning of the value you fill in part b? _The starting address of the array stdid.____

   d. What is the data width of the variables that declared as int? _______4________ bytes

   e. What is the starting address of the following arrays; please write it in hexadecimal format

   swapid: ___________0x20000040____

   oddid: ___________0x20000060____

3. Which SINGLE assembly instruction did the corresponding instruction take place

AND ____________________ANDS r0,r0,r1______________________

OR ____________________ORRS r0,r0,r1______________________

XOR ____________________EORS r0,r0,r1 ______________________

4. Write down the answer of your result in hexadecimal format

ANDresult ___________0x20422402____

ORresult ___________0x22577522____

XORresult ___________0x02155120____

What are the addresses of the following variables; please write it in hexadecimal format

ANDresult ___________0x200000014____

ORresult ___________0x200000018____

XORresult ___________0x20000001C____

5. What is the state of C and V ***BEFORE*** execution of ADDS instruction?

C _______1______ V _____0______

What is the state of C and V ***AFTER*** execution of ADDS instruction?

C _______0______ V _____0______

Please explain your result in detail with reference to your student ID.

C represents the carry flag and V represents the overflow flag. In addition cases, C is 0 when add operation does not result in a carry and is 1 when the operation result in a carry. For the overflow flag(V), V becomes 1 if operation result in overflow, 0 otherwise. In my case, x = 0x20472522 and y = 0x22527402, the addition result of x+y = 0x42999924. Since my highest four bits is 0010 and 0010, after addition, the result is 0100, there is no carry over. Also, the variable is declared as an integer, and in this case the summation of two positive numbers remain positive according to 2's complement. Therefore, there is no overflow in the summation.

6. What is the state of C and V ***BEFORE*** execution of SUBS instruction?

C ______0_______ V _____0______

What is the state of C and V ***AFTER*** execution of SUBS instruction?

C ______0_______ V _____0______

Please explain your result in detail with reference to your student ID.

In subtraction cases, C is 0 when subtraction operation needs a borrow bit and is 1 when don't need the borrow bit. For the overflow flag(V), V becomes 1 if operation result in overflow, 0 otherwise. In my case, x=0x20472522 and y=0x22527402 and x<y; therefore, it requires a borrow bit and C is 0. To perform the subtraction, we can write x and y in 2's complement and change y to -y by inverting the bits of y and add 1. After we change y to -y, we can perform addition and get the result:

x = 0010 0000 0100 0111 0010 0101 0010 0010
-y = 1101 1101 1010 1101 1000 1011 1111 1110
x-y = 1111 1101 1111 0100 1011 0001 0010 0000

From the 2's complement, the result of x-y where x<y is a negative number; therefore, there is no overflow (V=0).

7. From Page 16, after you added your code and compiled the project, check the Build Output, record the following: (Note: Make sure your project is complied with Optimization Level 2.)

`Program Size: Code = ________1168_________`

Any difference compared to the Question 1 you found before? _The code size is smaller.___

Please explain the optimization is being done with the help of checking the assembly language?

Optimization Level 2 results is smaller code size compares to Level 0 because it makes changes in the assembly code to use more registers to store the variables that will be reused. For example, in bitwise operations (AND, OR, XOR), it stores i, j in r1 and r2 and use other register to store the result. This can reduce the number of LDR and prevent the code from keep reloading i and j. In short, optimization level 2 uses more registers to store data in exchange for the efficiency.