Notation: A register and contents are written in the form "*RegisterName* = *Contents*", with contents in hexadecimal notation; like SP = 0506h, R5=0x32AF. As for memory reference, "[*Address*] = *Memory Contents*"; Contents may be in byte size (covering only one physical address) as in [0FFE0h]= 2Ah, or in word size covering in fact two physical addresses, like for example [0xF2FE] = 0A234h.

Problem 1. To define our notation, in a product $A \times B = P$, we call A the multiplier, B the multiplicand and P the product. (Notice: in Spanish the names for A and B are the other way, but since the normal multiplication is commutative, this fact is irrelevant.) Now, the maximum number of digits in P is the sum of the number of digits in A and B. In particular, to cover any possibility in binary systems, if A and B are bytes, P will be a 16-bit word.

Since there is no instruction for multiplication, one algorithm for multiplication can be as follows.

a. Load bytes multiplier and multiplicand in registers Ra and Rb, respectively.
b. Initialize product as 0
c. For bits k= 0 to 6
    (a) if bit k of multiplier is 1, then add Rb to product and rotate Rb left
    (b) if bit k of multiplier is 0, then rotate Rb left
d. If bit 7 of multiplier is 1, add Rb to product.

Your task is to

a. Test the **algorithm** starting on step (c)using the table below for 11X11, where registers in are in hex notation and already initialized (steps 1 and 2).
b. develop a code in assembly language for MSP430 for this algorithm.

	Multiplier	Multiplicand	Product
Steps c) and d)	Ra	Rb	Rc
	000B	000B	0000
Step from Bit 0			
Step from Bit 1			
Step from Bit 2			
Step from Bit 3			
Step from Bit 4			
Step from Bit 5			
Step from Bit 6			
Step from Bit 7			

Problem 2. Two word-size signed data stored in memory locations named NUM1 and NUM2 are to be used in a subroutine to generate a third signed number to be stored in location NUM3. The flow graph shown in Fig. P. 2 shows the steps to generate and store the new number. Use this flow graph or an equivalent one (you have to show it!) to generate the assembly code for your subroutine (include comments!). [NOTE: Remember that shifting or rotating left or right we can multiply or divide by 2.)

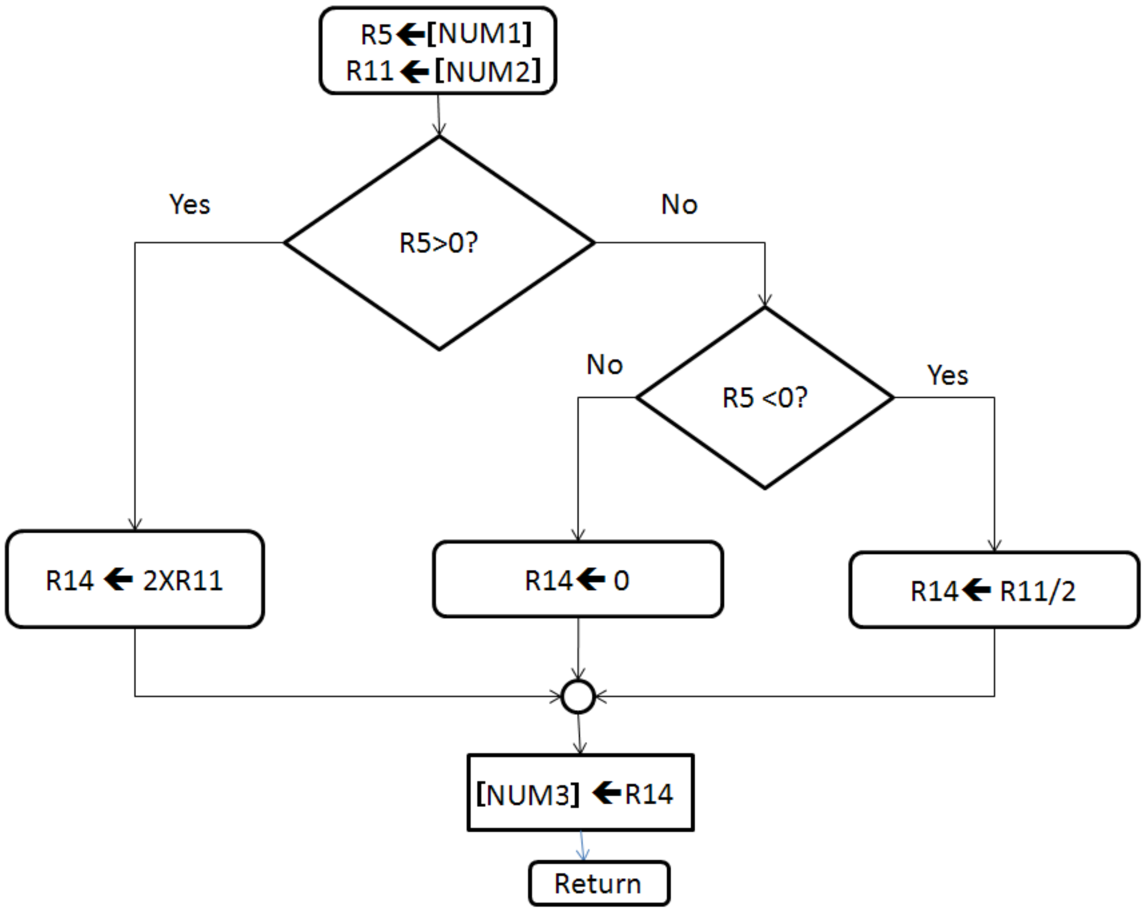

Figure P. 2: Flowgraph for subroutine.

Problem 3. Test your code for the previous problem using **NUM1 = the hex equivalent of twice the number formed with the last four digits of your student ID number** and NUM2= 0x802A. To test your code, fill up the following table, showing the registers you used with the respective information using hex notation AND in the left column indicating the instruction being applied. Add columns to the right if necessary according to your code.

[NUM1]=

[NUM2]=

(NOTE: in FLAGS column, apply the flag of interest to decide)

Instruction	R5	R11	R14	FLAGS	[NUM3]
	XXXX	XXXX	XXXX	– – – –	XXXX

Problem 4. A special device connected to the microcontroller as shown in Fig. P. 4 works as follows:

a. The microcontroller sends a pulse to the device, which receives it at the terminal TaskACK, and starts doing its work

b. When the device ends the task, sends a pulse to the microcontroller through its terminal TaskEND, which causes an interrupt for the CPU

c. The CPU then reads the result at port 1, and stores it at register 10.

You have to

(1) Configurate ports 1 and 2 (Do not forget that interrupt is being used)

(2) Write the instructions needed to ask the device for a task, and to receive the result in R10.

(3) Turn off the interrupt flag after reading the result

{NOTE: Remember that the registers associated to port X are: PxDIR (1 out, 0 in), PxIN, PxOUT, PxSEL (0 for port, 1 for peripheral), PxIE (1 enabled, 0 disabled), PxIES (1 high to low; 0 low to high), PxIF) }

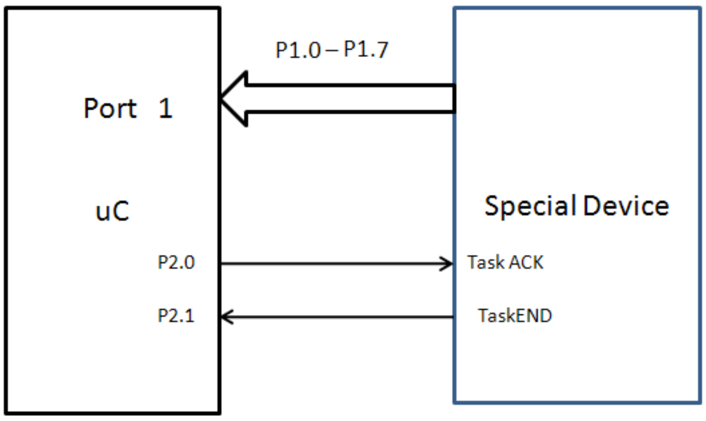

Figure P. 4: Connection to ports.