Experiment 1: Basic Assembly Coding

16.10.2023
Res. Asst. Abdullah Cihan Ak
akab@itu.edu.tr
Res. Asst. Yusuf Huseyin Sahin
sahinyu@itu.edu.tr

*N*o! My heart shall be a tower,
and I myself set at its highest rim:
where nothing else is, neither one last hurt,
nor the ineffable, where the world shall stop
Still one thing alone in immensity,
to darken first and then again grow light

Rainer Maria Rilke, The Solitary

"сила приносит свободу,
Побеждай и станешь звездой
А может, Обретешь покой"

Ария, Колизей

1 -Introduction and Preliminary

Welcome to the Microcomputer Lab! Your first experiment will be about the LEDs on your experiment kit. Please check the following documents before the lab.

- MSP430 Education Board Manual
- MSP430 Architecture Chapter 4,6
- MSP430 Instruction Set

Since this is an introductory lab, we shared the skeleton codes with you for some parts. You will make changes on them to adapt the codes to the given tasks.

2 (30 pts) - Part 1

Create a new CCS Project. Be sure that you selected **"MSP430G2553"** as the target and **Empty Assembly-only Project** as the project template. Write the given code inside main.asm. Since this code is a more complex version of the code from the Introduction video[1], please watch the video before delving deeper.

```
1  SetupP1      mov.b   #11111111b,&P1DIR
2               mov.b   #11111111b,&P2DIR
3               mov.b   #00000000b,&P1OUT
4               mov.b   #00000000b,&P2OUT
5               mov.b   #00001000b, R6
6               mov.b   #00010000b, R7
7               mov.b   #0d, R8
8
9  Mainloop1    bis.b   R6, &P1OUT
10              bis.b   R7, &P2OUT
11              inc     R8
12              rra     R6
13              rla     R7
14              mov.w   #00500000, R15
15 L1           dec.w   R15
16              jnz     L1
17              cmp     #4d, R8
18              jeq     SetupP1
19              jmp     Mainloop1
```

Before changing the code, let's check it line-by-line carefully.

In Lines 1 and 2, both Port1 and Port2 are set. Remember that, both of these ports have 8-bit read/write registers. Thus, using #**11111111b**, we can set every bit as output. When defining constant numbers, we use **b** for bit representation, **h** for hexadecimal representation and **d** for decimal representation. Also note that this is a byte operation because of **".b"**. Since the ports have the capacity of one byte, it is a suitable operation.

Another point to emphasise at this point is the difference between **bis** and **mov** operations. **Please check the difference between them, you will report your findings.**

In Lines 3 and 4, we initially move zeroes to the ports. Thus, initially, all leds are off.

In Lines 5 and 6, we moved #**00001000b** and #**00010000b** to registers R6 and R7 respectively. We will use these registers in the lines below: In the main loop (Mainloop1), we will move the values in these registers to P1OUT and P2OUT for different led states.

In Line 7, we initially start R8 from zero. We will use R8 to keep the loop variable. Our led pattern will consist of four states. Thus we will further use R8 to create the **for(int**

[1] https://youtu.be/MJIjhCs6Hek

R8=0, R8<4; R8++) loop.

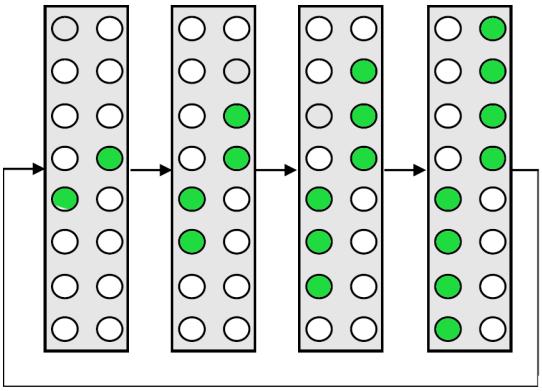
In Lines 9 and 10, we set some bits of P1OUT and P2OUT according to R6 and R7.

In Line 11, we incremented R8 by one.

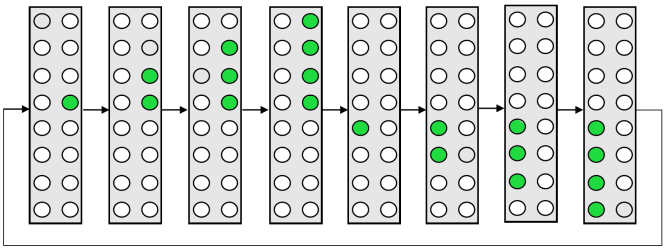
In Lines 12 and 13, the values inside R6 and R7 are rotated arithmetically to right and left. Thus, in the next iteration different leds will be activated.

Lines between 14-15 are very similar with the code in the Introduction. Here a huge value is assigned to R15 and this value is decreased step by step. Thus, we have a **busy-waiting** here. If the value of R15 decreases down to zero, we can pass the "jnz L1"(jump to label L1 if not zero) line. Then we check whether the value in R8 reached to 4, which means all four states are shown on the leds. If the value is reached, we restart the process by jumping to label SetupP1.

After running the code in the device, we will obtain the pattern given below.

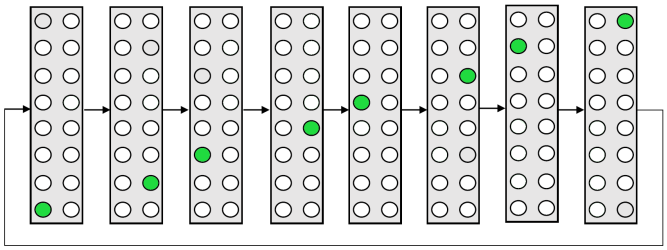


Run the code and observe its results. Then, change the code so that, P1 LEDs and P2 LEDs to work successively as in the figure below.



3 (30 pts) - Part 2

Change the code in Part 1 so that, it works as in the figure below.



4 (40 pts) - Part 3

Change the code so that, it works as in the figure below. Then, assign a button from PORT2, which stops the system when pressed.