EE 337: Advanced Assembly Programming
Lab 2

14^{th} August, 2019

This set of experiments has the following objectives:

- Familiarization with advanced instructions of 8051 family.
- Familiarization of breaking down a problem into subproblems and using subroutines to solve a problem.

You should browse through the manual for Keil software and the data sheet for 89C5131A uploaded on moodle site. Specifically, refer to these to understand the memory-map.

89C5131A has 256 bytes of RAM, of which the top 128 bytes can only be accessed using indirect addressing. Since the stack is always accessed through indirect addressing using the stack pointer, we shall locate the stack in this part of the memory. In our assembly programs, we shall initialize the stack pointer to 0CFH, which provides 32 bytes for the stack. It is common to place arrays also in this part of this memory, because arrays are also easily accessed using a pointer. The address range from 80H to CFH is available for this (since memory beyond this is reserved for the stack).

Refer to the textbook by Mazidi and Mazidi for syntax of instructions.

1 Homework

1. Understand the sample (`sample_led.asm`) program provided (in `lab1.zip` file) before getting started on these exercises. Specifically, understand how the code is written down as subroutines to make the code readable. Also, the subroutines use `PUSH` and `POP` instructions for storing and restoring the callers' registers. (Viva 5 points)

   You will be using a board in the later weeks which uses a 24MHz crystal. In order for your simulation to match design running on the board, you have to setup the crystal frequency in the simulator to be 24MHz.

2. Write an assembly program to blink an LED at port P1.7 at specific intervals. At location $4FH$ a user specified integer D is stored. You should write a subroutine called `delay`. When it is called it should read the value of D and insert a delay of $D/2$ seconds. Then write a main program which will call `delay` in a loop and blink an LED by turning it ON for $D/2$ seconds and OFF for $D/2$ seconds. D will satisfy the following constraint: $1 \leq D \leq 10$.

   You can use the code sequence below to introduce a delay of $\sim 50ms$.

```
MOV R2,#200
BACK1:
    MOV R1,#0FFH
    BACK :
         DJNZ R1, BACK
    DJNZ R2, BACK1
```

The counters R1 and R2 are setup in such a way that the code above would take roughly $50ms$. See the Appendix to know how the delay is calculated.

You have to nest this inside another loop with appropriate counters to get the required delay.(5 points)

You should assemble and debug these programs using Keil software on a PC or laptop. You should check that the program is operating correctly. If necessary, use single stepping and breakpoints provided by Keil Software.

2 Appendix

The code segment below can be used to get a delay of $\sim 50ms$:

```
MOV R2,#200
BACK1:
    MOV R1,#0FFH
    BACK :
         DJNZ R1, BACK
    DJNZ R2, BACK1
```

The delay calculations are as follows. `MOV` and `DJNZ` instructions take 1 and 2 machine cycles respectively. Therefore, the number of machine cycles required for the for loop is 1+255*2 = 511 machine cycles. Since this is inside a loop where R2 = 200, the outer loop at `BACK1` takes 200*511 + 200*2 = 102,600 machine cycles. Each machine cycle is 12 clock cycles making the total number of clock cycles 1, 231, 200 clock cycles. Dividing this by the crystal frequency (24 MHz), we get a delay of $\sim 50ms$.