EE 337: Introduction to the Pt-51 Kit
Lab 3

14th August, 2017

Familiarization with Pt-51 kit and simple peripherals

You should browse through the manual for Keil software, the data sheet for 89C5131A, and the Pt-51 user manual.

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 Take home assignments

1. Repeat this part from Lab 2 homework, but execute your program on the Pt-51 kit. 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: $1 \leq D \leq 10$.

1. Write a subroutine `readNibble` (as per the algorithm template given next) which will read the binary value which is set on the port using slide switches (P1.3-P1.0). The subroutine should display this value on the LEDs for 5 seconds and store the nibble as the last four bits of location $4EH$.
   Once it is displayed, the program should clear the pins P1.7-P1.4 for one second and call readNibble again to read the new value from the switches. If the read value equals to $0FH$, the program should display the value stored at $4EH$(the previously read nibble), otherwise, display the new value. Here readNibble is the subroutine, which is to be called only once for reading the nibble. In order to come out of the loop you should enter 0FH.

```
readNibble :
;Routine to read a nibble and confirm from user
;First configure switches as input and LEDs as Output.
;To configure port pin as input, set it.
;Logic to read a 4 bit number (nibble) and get confirmation from  user
```

```
loop:
  ;turn on all 4 LEDs (routine is ready to accept input from the user)
  ;wait for 5 sec during which user can give input through switches
  ;turn off all LEDS
  ;read the input from switches (nibble)

value_changed:
  ;show the read value on LEDs and store in 4EH
  ;wait for 5 sec
    ;clear LEDs (pin P1.7 - p1.4)
    ;read the input from switches
    ;if  read value != 0Fh go to loop
display_old:
;otherwise display previously stored nibble from location 4EH
;wait for 5 seconds
;return from the subroutine.
```

Note: you should push / pop all registers being used in the algorithm

This algorithm provides a visual handshake and is to be used for taking nibble inputs from slider switches in the lab work problems. The user is to setup the slider switches to specific value of interest during the 5 sec period when all LEDs are ON to confirm the previous nibble entered.

2. (Optional now but will be required for the next lab)

   Write a subroutine `packNibbles`. Two successive 4 bit values read using `readNibble` should be combined to form a byte (with most significant nibble being read first followed by least significant nibble), which should be stored at location $4FH$.