LAB NO. 3

BLINKING LEDS USING PIC ASSEMBLY PROGRAMMING

OBJECTIVES: -

- To write a code for PIC in assembly to control LED blinking.
- To observe ROM (Program Memory) and RAM (SFRs) in PIC using MPLAB X.
- To simulate the circuit on Proteus.

APPARATUS: -

- Laptop
- MPLAB X
- Proteus Circuit Simulator

CLASS TASK: -

- Write a program for PIC in assembly language.

THEORY: -

INSTRUCTIONS & ASSEMBLER DIRECTIVES: -

There are some commands in PIC assembly programming which are known as Instructions which tells the CPU what to do next. Another type of commands is known as Assembler Directives which tells the assembler what to do. Both of these types of commands are used in Assembly programs.

MOVLW, ADDLW, ADDWF are some examples of Instructions used in PIC assembly. EQU, SET, #include, ORG are some examples of Directives used in Assembly programs.

- **EQU (EQUATE): -**

This directive is used to define a constant value in assembly programming, this equates a tag to a constant and we can use that tag where the constant value is requires in program. The constant can be a number, address or a port etc.

- **ORG (ORIGIN): -**

This directive tells the CPU where to place the following instructions in Program ROM, this instruction basically set the address in ROM where the program will be burnt in PIC.

- **END: -**

This directive tells the CPU about the end of the program, it means instructions or directives after this directive cannot be processed by assembler.

- **LIST: -**

This directive specifies the assembler for a specific PIC chip, all in all this directive tells assembler which PIC models is being programmed and for which the code is being written.

- **#INLCUDE: -**

This directive is used to include an assembler's library in program these libraries helps the programmer to make the code simple and small.

- **_CONFIG (CONFIGURATION): -**

This directive configures the PIC chip for specific operations, for example which clock will be used Internal or External, Is Code Protection on or off? auto reset on or off?

Port Programming: -

PORTs are group of I/O pins in PIC used for inputting and outputting data in or from the PIC respectively. Pins are used to attach external devices with the PIC and used for input, output purposes. There are 5 ports in PIC18F452 these are PORTA, PORTB, PORTC, PORTD and PORTE. Every port has some pins associated with it and three registers with each port. PORTC, TRISX and LATX are those three registers where X indicates the port.

- **PORTX Register: -**

This register is used to send or receive some data from PIC, this register writes or fetches data on pins in PIC. This register also used to send data to external devices via pins.

- **TRISX Register: -**

This register is used to configure the port or input or as output purpose. A port or a pin in PIC can be used for input or output purpose at a time. If this register is set for respective port that port will be an INPUT port or if TRISX is zero then PORTX will work as an output port.

DECFSZ & DECFSNZ Instructions: -

In DECSFZ instruction a file register is decremented and overwritten with decremented value, then if content of the register is zero the instruction after the DECFSZ is skipped in this way a conditional loop can be created.

The DECFSNZ is vice versa of the above, the next instruction is skipped if the content is not zero.

- **PORTX Register: -**

This register is used to send or receive some data from PIC, this register writes or fetches data on pins in PIC. This register also used to send data to external devices via pins.

- **TRISX Register: -**

This register is used to configure the port or input or as output purpose. A port or a pin in PIC can be used for input or output purpose at a time. If this register is set for respective port that port will be an INPUT port or if TRISX is zero then PORTX will work as an output port.

Program: -

Code: -

```
#include "P18F452.inc"
LIST P=18F452, F=INHX32, MM=OFF

CONFIG OSC=XT
CONFIG WDT=OFF

ORG 0x00
GOTO MAIN
ORG 0x200
```

```
MAIN:

    R2 EQU 0x20
    R3 EQU 0x21

    MOVLW 0x00
    MOVWF TRISB

    ENDLESS_LOOP
     COMF PORTB, F
     CALL Delay
     BRA ENDLESS_LOOP
     ORG 0x2000

    Delay           ; 500ms Delay, F=10MHz

          MOVLW D'200'
          MOVWF R3
     AGAIN     MOVLW D'250'
               MOVWF R2
     HERE      NOP
               NOP
               DECF R2,F
               BNZ HERE
               DECF R3,F
               BNZ AGAIN
               RETURN
END
```

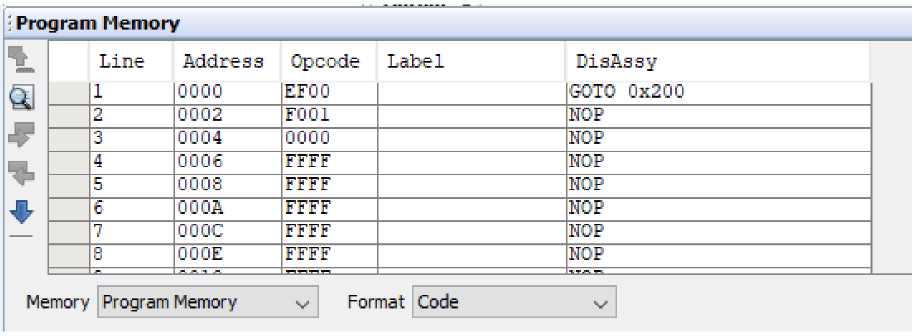
EXPLANATION: -

First of all, using include directive include files for PIC18F452 are being included and then LIST directive tells the CPU which chip is being used. _config directive tells about to use Internal Oscillator. The ORG pronounced as Origin directive specifies the location where to place the code. Line GOTO Main is place on 0x00 location while Main code is starting from 0x200 and delay from 0x2000.

The working register is being loaded by literal value 0x00 and then this value is sent to TRISB register, this operation will instruct the processor to use portb pins as output pins. It means the LEDs will be connected to portb.

In Main code we have nothing but an infinite loop is initialized and data on PORTB is complemented, if it is zero sets to FF and if is FF sets to zero. Then a delay subroutine is being called which is used for a 500ms delay and code jumps to the start of loop means the complementing of PORTB.

PROGRAM ROM: -

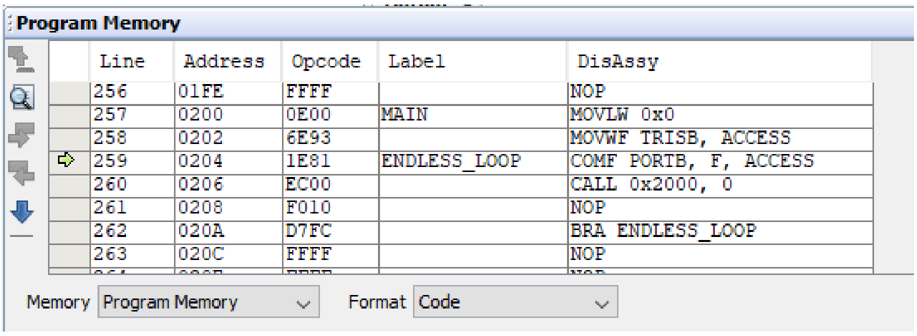


Program Memory

Line	Address	Opcode	Label	DisAssy
1	0000	EF00		GOTO 0x200
2	0002	F001		NOP
3	0004	0000		NOP
4	0006	FFFF		NOP
5	0008	FFFF		NOP
6	000A	FFFF		NOP
7	000C	FFFF		NOP
8	000E	FFFF		NOP

Memory Program Memory Format Code

Figure 1.1: Program ROM (GOTO at 0x00)

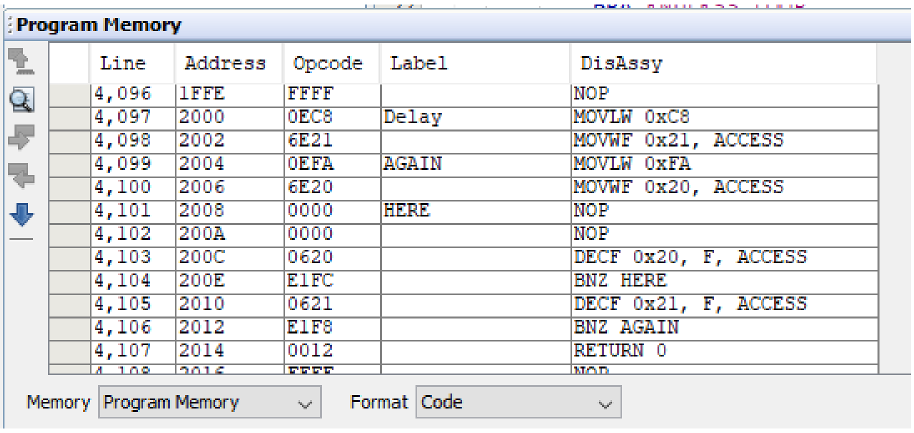


Program Memory

Line	Address	Opcode	Label	DisAssy
256	01FE	FFFF		NOP
257	0200	0E00	MAIN	MOVLW 0x0
258	0202	6E93		MOVWF TRISB, ACCESS
259	0204	1E81	ENDLESS_LOOP	COMF PORTB, F, ACCESS
260	0206	EC00		CALL 0x2000, 0
261	0208	F010		NOP
262	020A	D7FC		BRA ENDLESS_LOOP
263	020C	FFFF		NOP

Memory Program Memory Format Code

Figure 2.2: Program ROM (Main Program stating at 0x200)



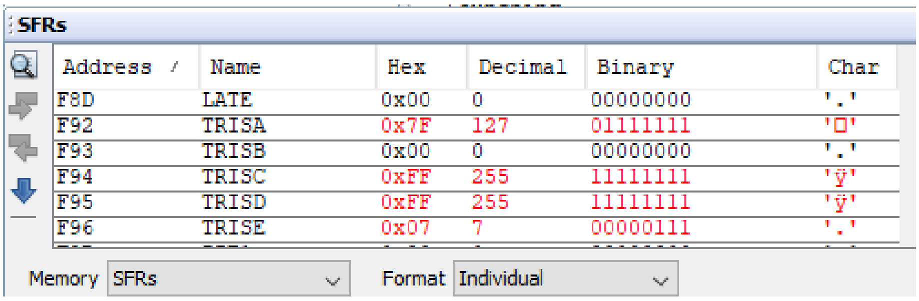
Program Memory

Line	Address	Opcode	Label	DisAssy
4,096	1FFE	FFFF		NOP
4,097	2000	0EC8	Delay	MOVLW 0xC8
4,098	2002	6E21		MOVWF 0x21, ACCESS
4,099	2004	0EFA	AGAIN	MOVLW 0xFA
4,100	2006	6E20		MOVWF 0x20, ACCESS
4,101	2008	0000	HERE	NOP
4,102	200A	0000		NOP
4,103	200C	0620		DECF 0x20, F, ACCESS
4,104	200E	E1FC		BNZ HERE
4,105	2010	0621		DECF 0x21, F, ACCESS
4,106	2012	E1F8		BNZ AGAIN
4,107	2014	0012		RETURN 0

Memory Program Memory Format Code

Figure 3.3: Program ROM (Delay sub-routine stating at 0x2000)

RAM (SFRs): -

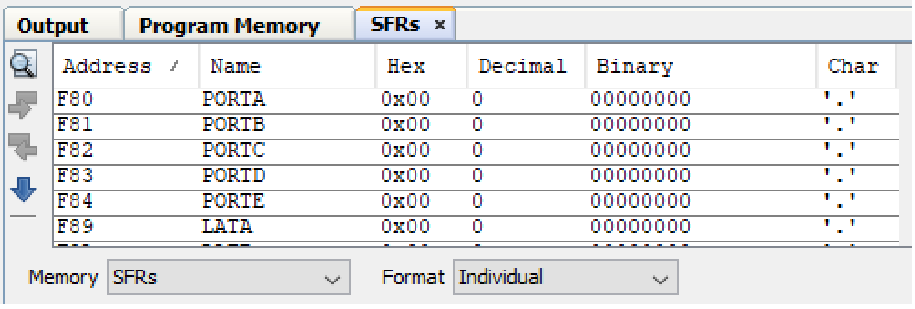


SFRs

Address	Name	Hex	Decimal	Binary	Char
F8D	LATE	0x00	0	00000000	'.'
F92	TRISA	0x7F	127	01111111	'□'
F93	TRISB	0x00	0	00000000	'.'
F94	TRISC	0xFF	255	11111111	'ÿ'
F95	TRISD	0xFF	255	11111111	'ÿ'
F96	TRISE	0x07	7	00000111	'.'

Memory SFRs Format Individual

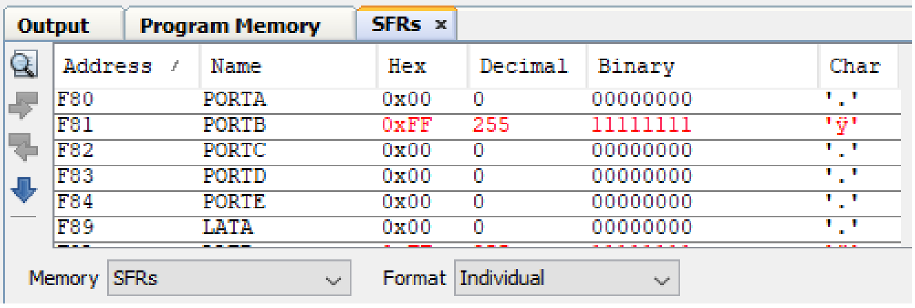
Figure 4.4: SFR's view for TRIS registers



Address	Name	Hex	Decimal	Binary	Char
F80	PORTA	0x00	0	00000000	'.'
F81	PORTB	0x00	0	00000000	'.'
F82	PORTC	0x00	0	00000000	'.'
F83	PORTD	0x00	0	00000000	'.'
F84	PORTE	0x00	0	00000000	'.'
F89	LATA	0x00	0	00000000	'.'

Memory SFRs Format Individual

Figure 5.5: SFR's View for PORT registers (Before Compliment)



Address	Name	Hex	Decimal	Binary	Char
F80	PORTA	0x00	0	00000000	'.'
F81	PORTB	0xFF	255	11111111	'ÿ'
F82	PORTC	0x00	0	00000000	'.'
F83	PORTD	0x00	0	00000000	'.'
F84	PORTE	0x00	0	00000000	'.'
F89	LATA	0x00	0	00000000	'.'

Memory SFRs Format Individual

Figure 6.6: SFR's View for PORT registers (After Complement)

CIRCUIT: -

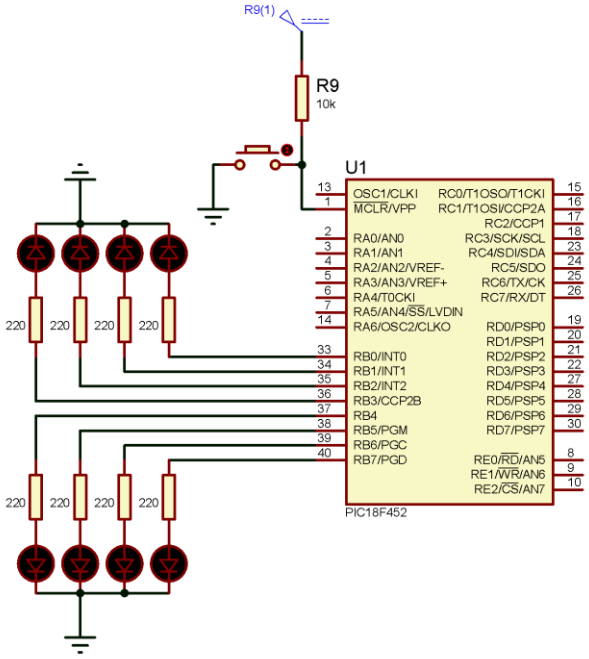


Figure 7.7: Circuit for above PIC program (Class Task)

HOME TASK: -

- A push button is connected to a pin of Microcontroller, every time it is pressed, a data is sent to a PORT. Data starts from decimal value 100 and ends at 1000, as 1000 value reaches reset the count to 100. During counting another switch can be pressed to restart the data at any instant of time.

THEORY: -

PORT BIT MANIPULATION: -

This concept is related manipulation of ports bit by bit, one can set, clear or test the bits of the ports. There are several commands available to do that some are

- BSF (Bit Set File Register)
- BCF (Bit Clear File Register)
- BTG (Bit Toggle File Register)
- BTFSS (Bit Test File Register Skip if Set)
- BTFSZ (Bit Test File Register Skip if Zero)

These commands or instructions directs the pins of ports to be set or clear or to do some task if it is set or is clear.

PROGRAM: -

CODE: -

```
#include "P18F452.inc"
LIST P=18F452, F=INHX32, MM=OFF

    CONFIG OSC=XT
    CONFIG WDT=OFF

    ORG 0x00
    GOTO MAIN
    ORG 0x200

MAIN:
    NUM EQU D'100'

    CHK_1 EQU 0x20
    CHK_2 EQU 0x21

    SETF TRISE
    CLRF TRISB
    CLRF TRISC

Here        MOVLW NUM
            MOVWF PORTB

            CLRF PORTC
```

```
          MOVLW d'225'
          MOVWF CHK_1
          MOVLW d'4'
          MOVWF CHK_2

Again     BTFSC PORTE, 1
          BRA Here
          BTFSS PORTE, 0
          BRA Again
          DECF CHK_1, F
          BZ DEC_L
          INCF PORTB, F
          BC New
          GOTO Again

New       INCF PORTC, F
          GOTO Again

DEC_L     DECFSZ CHK_2, F
          MOVLW d'225'
          MOVWF CHK_1
          RETURN
          BRA Here
END
```

EXPLANATION: -

In this program all the pre requisites are loaded and then data is loaded into the ports, here PORTB and PORTC are used to send 100-1000 values. Here two registers are used because the PIC is a 8-bit architecture controller so we can send a maximum data of decimal value 0-255 to a port for further data we can use a combination of two ports as 16-bits so a data of 0-65534 can be sent to the combined ports PORTC and PORTB. The data is represented as follows: -

RC7 RC6 RC5 RC4 RC3 RC2 RC1 RC0 RC7 RC6 RC5 RC4 RC3 RC2 RC1 RC0

Then two registers are used to test whether count reached to 1000. For this purpose these two are named as CHK_1 and CHK_2 the CHK_1 is loaded with value 225 and CHK_2 with 4 then after every button press 225 is decremented and if it reaches zero then CHK_2 is incremented and CHK_1 is again loaded with 225 so if CHK_2 becomes zero then it means 900 has been completed and count reaches 1000 at that point count becomes 100.

The count is incremented at every HIGH of PORTE, RE0. Reset button is connected to the PORTE, RE1 if it is HIGH every register will go to the default value as set in program.

CIRCUIT: -

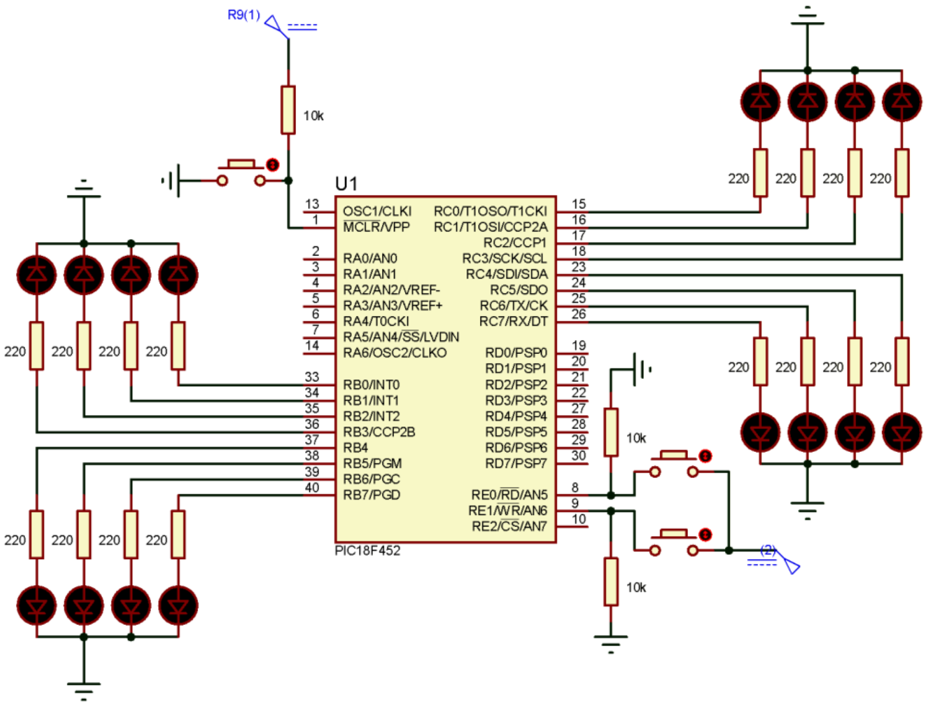


Figure 8.8: Circuit for above PIC program (Home Task)

CONCLUSION: -

- In this lab I have learnt how to program PIC using assembly language.
- Some commonly used are introduced and were manipulated with PIC assembly programming.
- How an assembly program is started and what are instructions and directives commonly used in assembly for PIC programming.
- Then PORT programming was introduced and a simple blinking program is being written in the assembly.
- In home work task I learnt how to use buttons with PIC and how to do a specific task if certain conditions meet in the PIC assembly programming.