COMPUTER SYSTEM DESIGN
(ULC401)

LABORATORY FILE

(II YEAR)

B.E. - EEC

Submitted to: Dr. Mukesh Dalal

Submitted by: Gaurav Singh **Roll No :102169002**

THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY
PATIALA

JAN-MAY 2023

LISTS OF EXPERIMENT

1. Steps to create a file and project in MPLAB (details about how to start).
2. Write a program to add/sub 2 8-bit numbers in Assembly Language.
3. Write a program to add 1H 5 times with the content of Working Register.
4. Write a program to add 5H value with the contents of Working register.
5. Write a program to store a value 20H in file memory locations 30H to 34H.
6. Write a program in assembly language to illustrate Addressing Mode.
7. Write a program in assembly language to illustrate Data transfer operations.
8. Write a program in assembly language to illustrate the following:
    a) ARITHMETIC OP.
    b) LOGICAL OP.
9. Write a program in assembly language to illustrate SHIFT Operation.
10. Write a program in assembly language to illustrate the following:
    a) LOOP
    b) LOOP INSIDE LOOP
11. Write a program to illustrate the following:
    a) bit oriented
    b) byte oriented

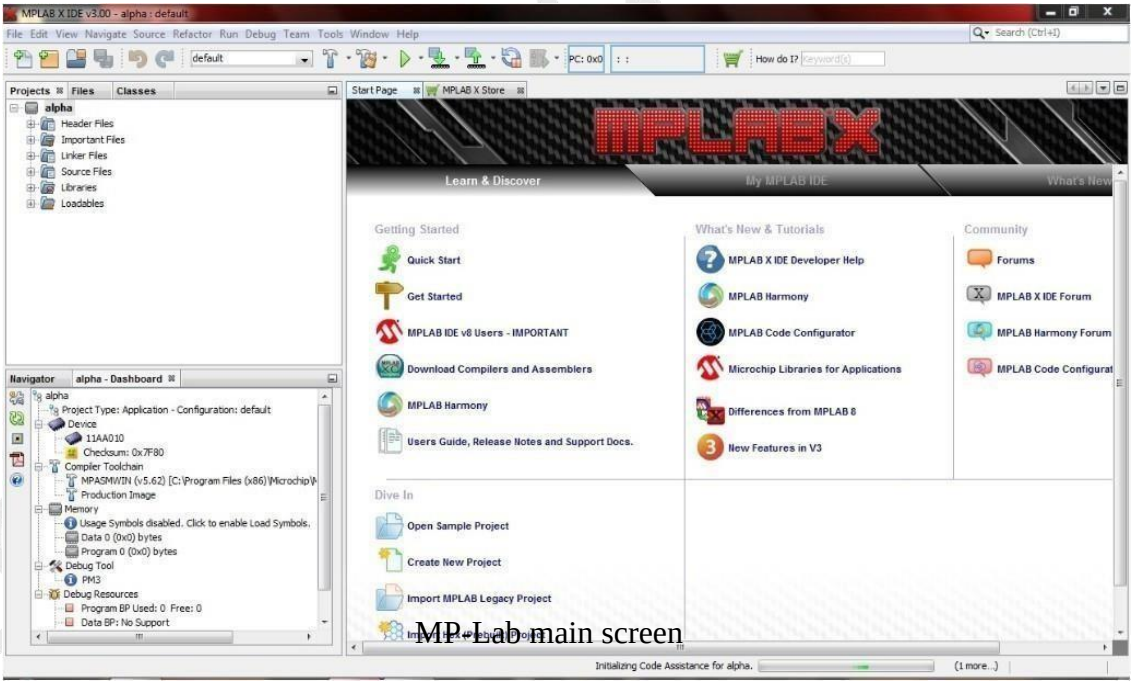
EXPERIMENT -1

Introduction:

In this lab, we will learn how to open the **MP-LAB** software and open the file for programming. Basically, it is the introduction lab for the semester wo rk in Embedded lab. This introduction lab is very useful for coming projects and work in lab.

MP-Lab:

MP-lab is a public domain software and is developed for the embedded applications on PIC and microcontroller and this is developed by multinational compan y named as Microchip technology. MP lab supports the coding, debugging and programming of microchips, microcontrollers of 8bit, 16bits and DS -PIC microcontroller and also for 32bit software asset management and Pic - Microcontrollers 32 bits.



MP-Lab main screen

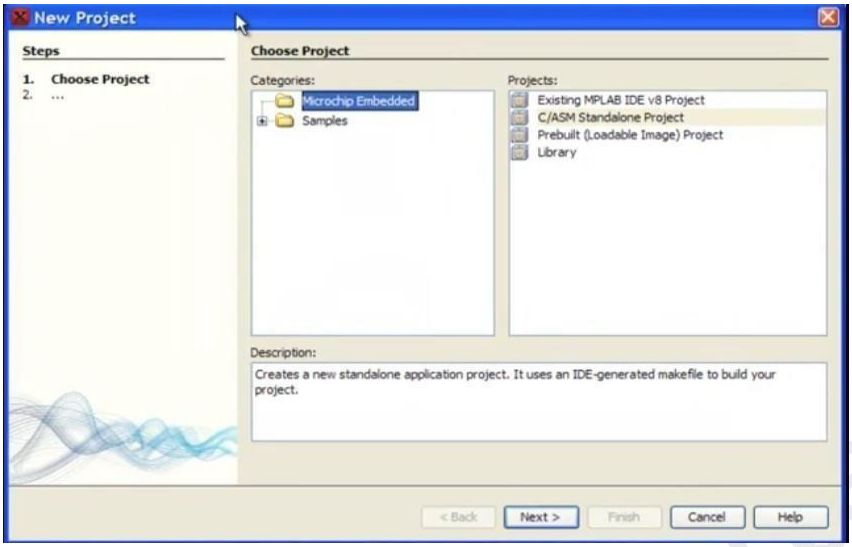
PIC-Microcontroller:

The word PIC stands for **Peripheral Interface Controller** . Firstly, PIC developed for the PDP **Program Data processor** computers for controlling the Peripheral devices. These are very easy to carry out as com pared to others. This is widely used because it is of very less cost and also easy for programming and have large base for store the program. This is 40 pin IC.

PIC 40 Pin Microcontroller

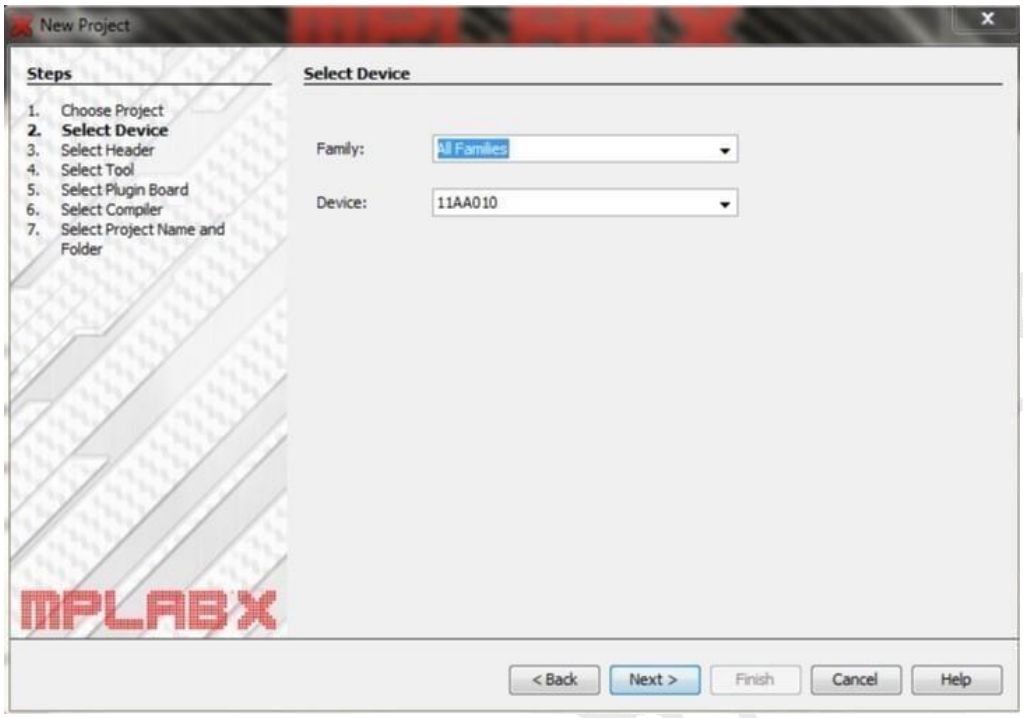
Procedure:

- First of all, we have to open the MP -Lab Software and to start the project we have to go on the left top corner where the file bar is written and select the New -Project. The pop-up screen will be shown as follow.
- Then, we have to choose the C/ASM Standalone project which is the assemblylanguage

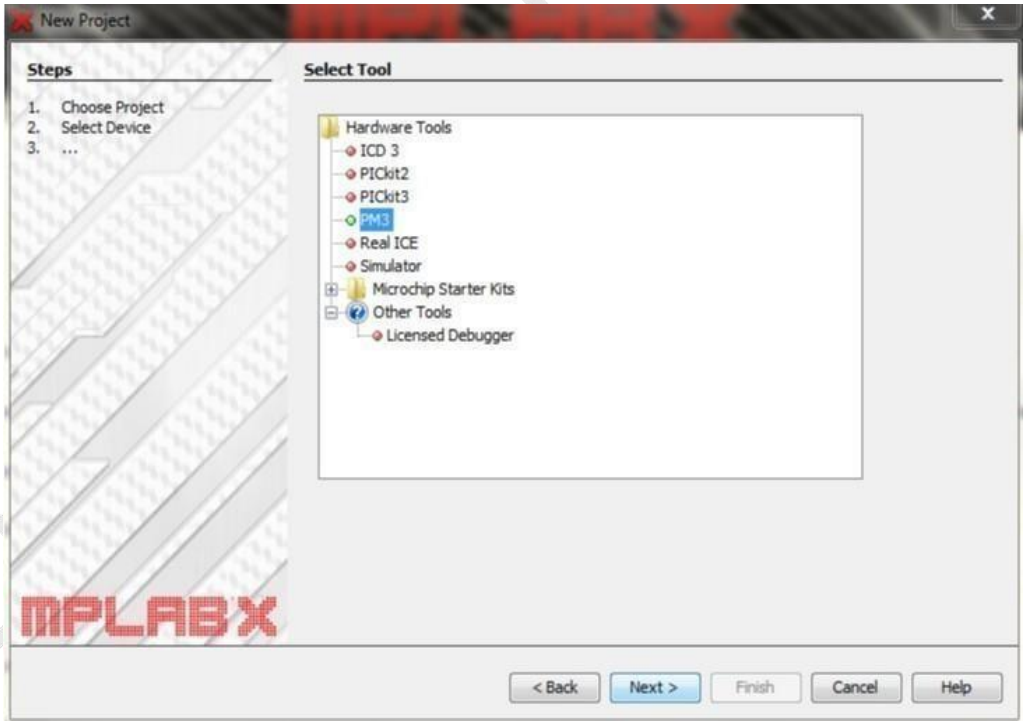


Making of new project

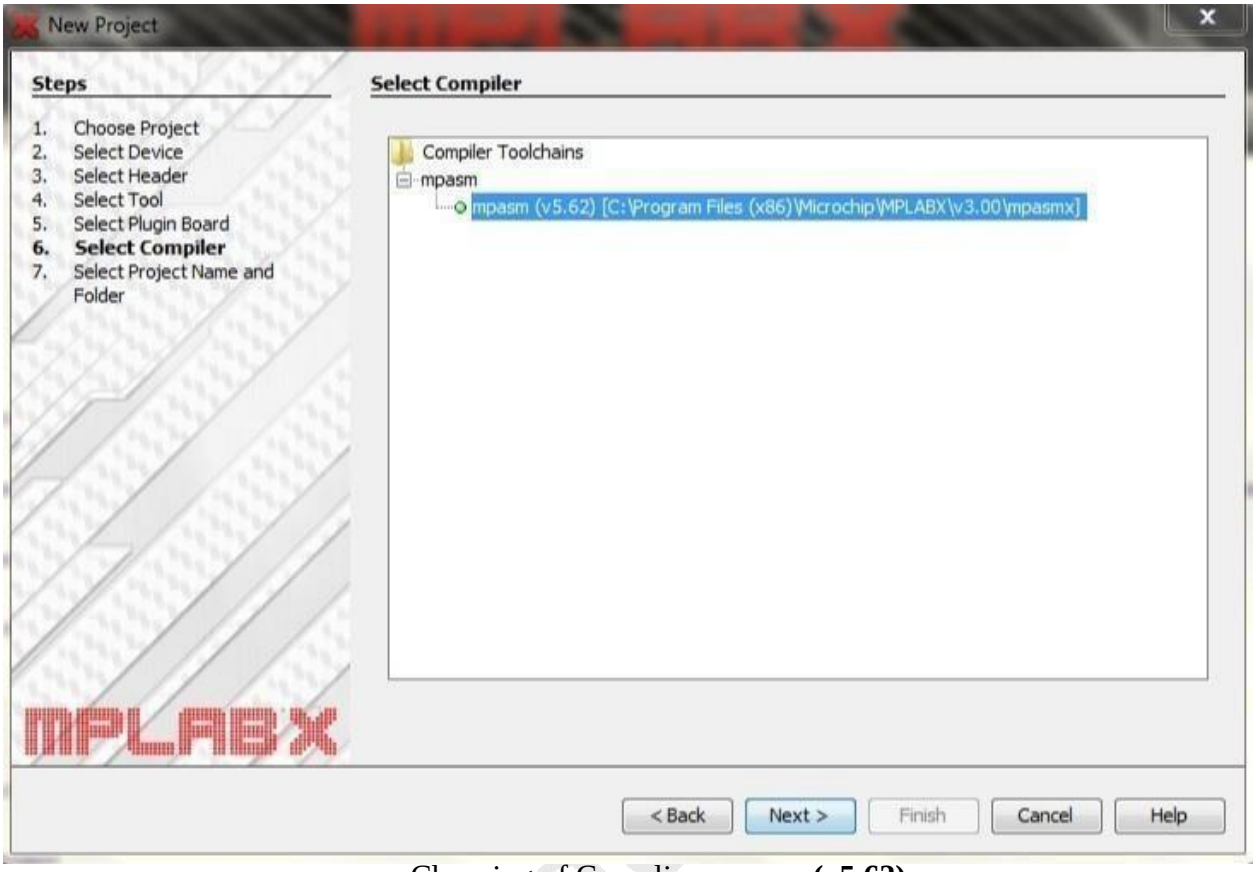
- The third step is to select the device. Before selecting the device we choose the upper bar as All-Families. We choose the device **PIC-18C-452**.
- Here, the PIC stands for **Peripheral Interface Controller** and the **18** stands for the generation and **C** is the family name.
- Now after selecting device we will next move to hardware tools where we select the **PM3** and after selecting it we will next move to select the complier.
- The compiler we choose is **mpasm (v5.62)**



selecting the device and Family

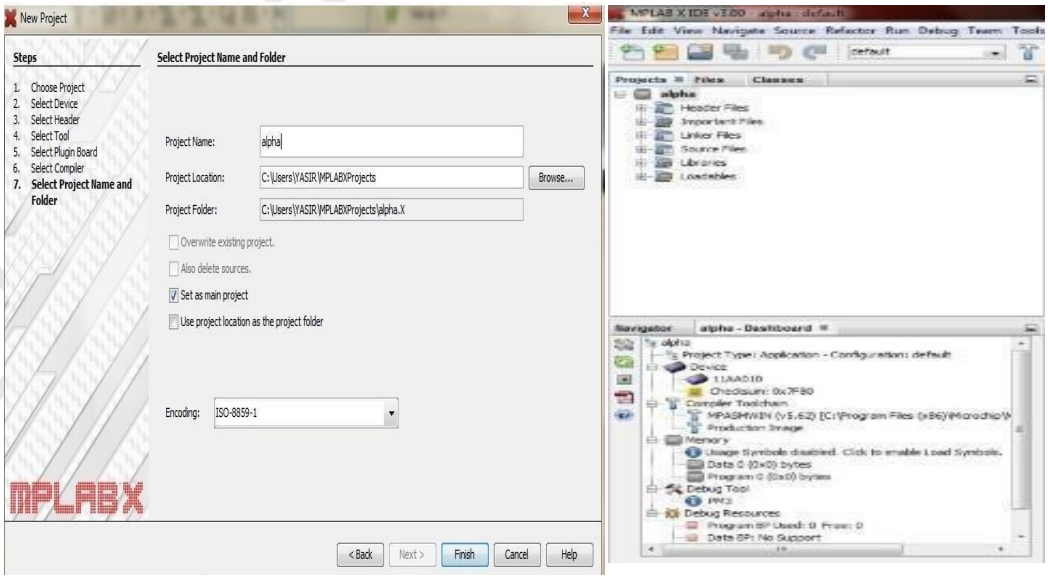


Choosing of the Hardware tool **PM3**

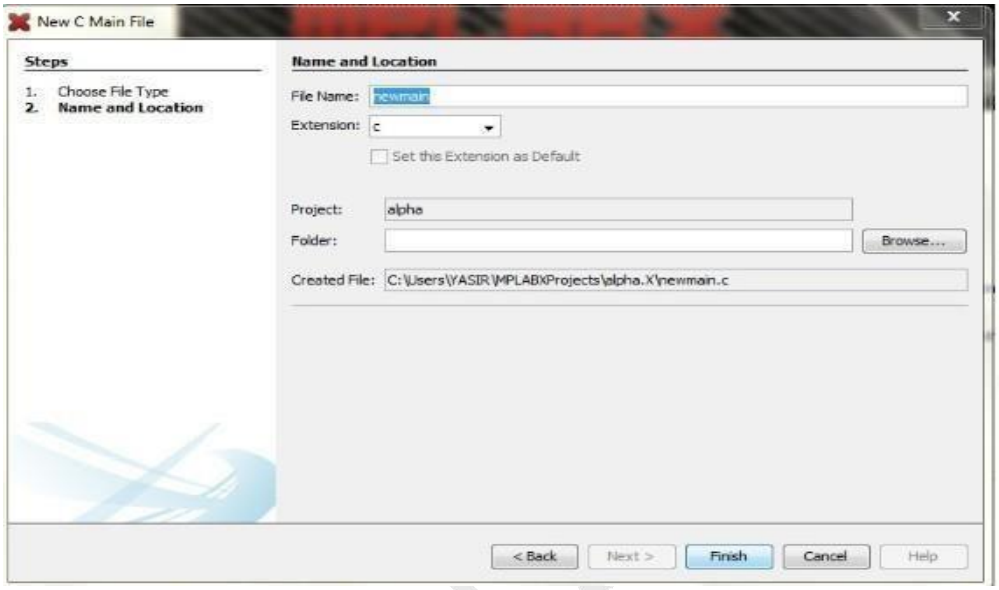


Choosing of Complier **mpasm (v5.62)**

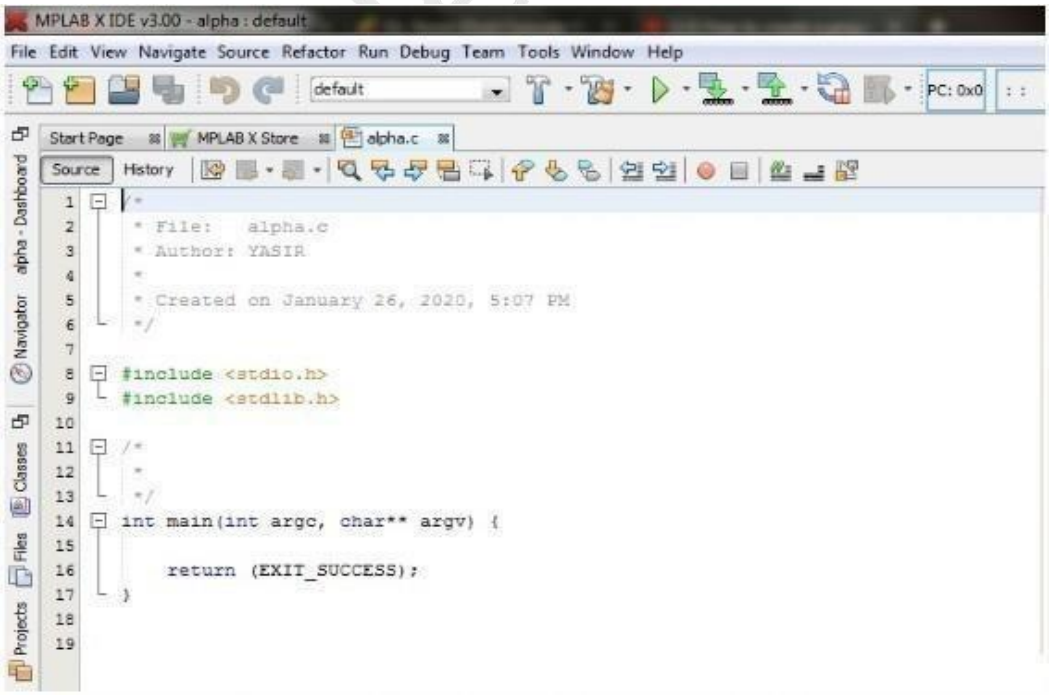
- In the next step, we have to name the project and we named it as **alpha** and then click on finish and our project is ready and the pop-up screen will appear as shown in figure.



- Now after project of alpha has been created, we will click on alpha and some new sub files will be shown to us and we then click on source file and select the new option and select the sample file and also select the location where we want to save then click on finish button and our full file has been created.



Naming and locating the new source file



New Source file

How to Debugging pic code using simulator

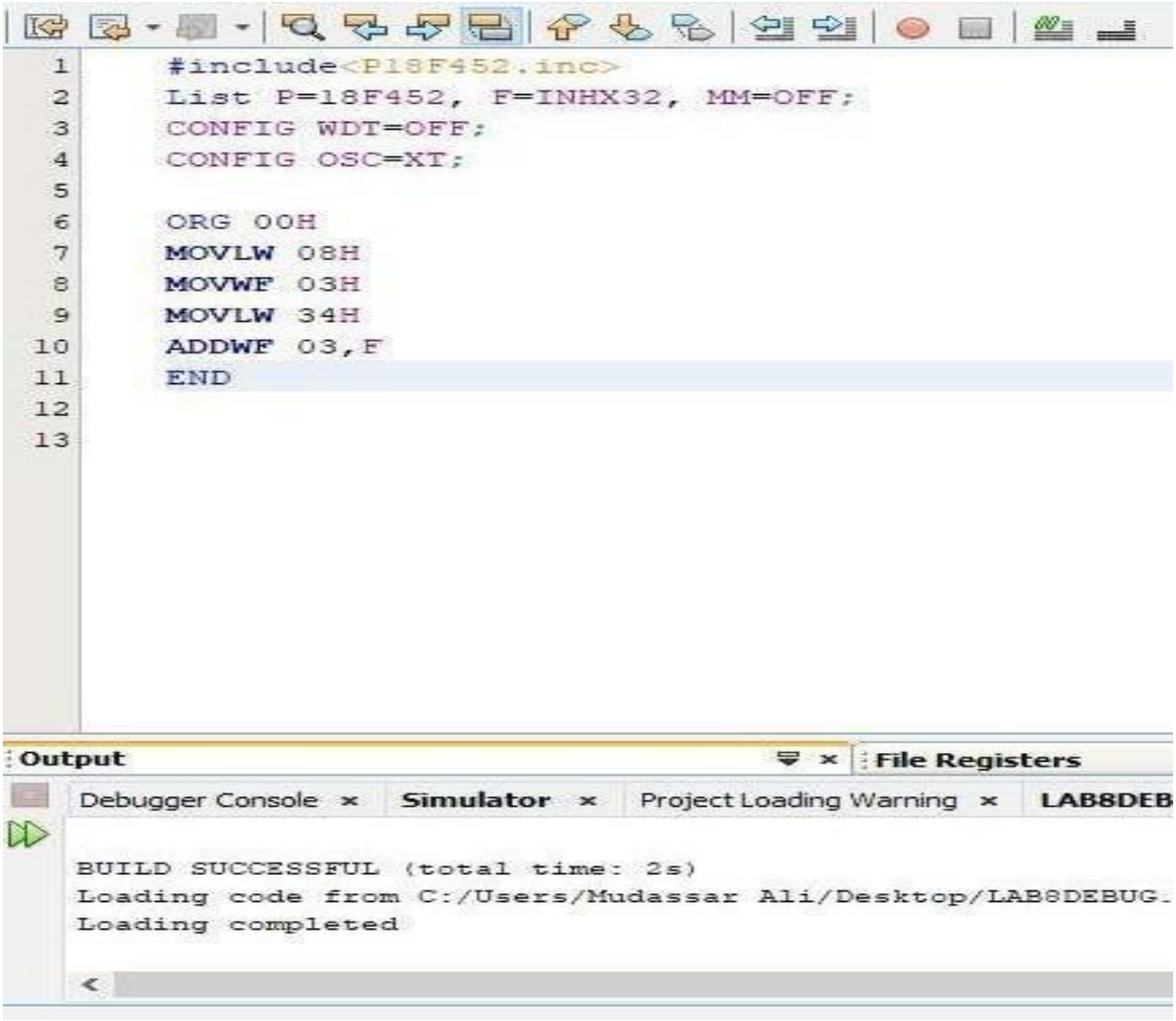
Debugging:

"Debugging is the process of finding and resolving errors within a computer program".

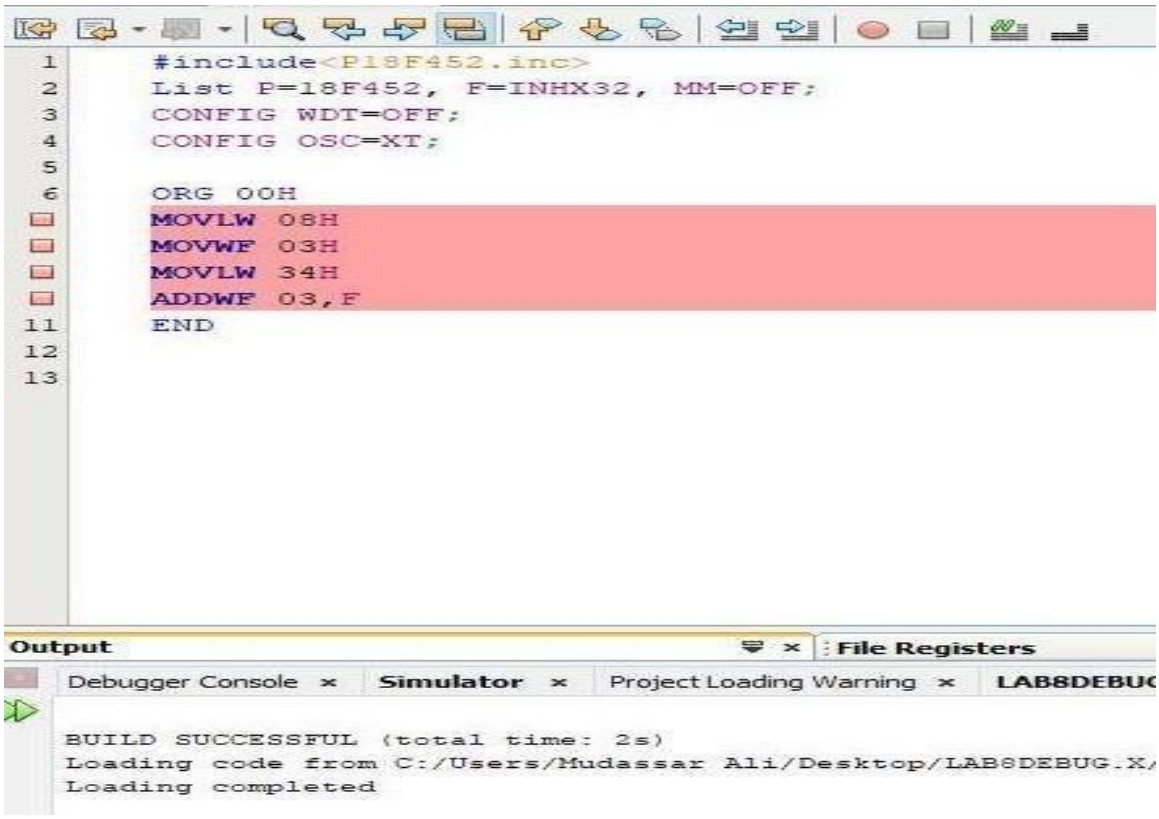
I execute program step by step and check valves of WREG and SFR's at every step. If any error occurs in program, then remove it. I check the progress of our code by using breakpoints on a single or every line of code. Our computer, when executing the code, stops on the breakpoint until permission to proceed to next line is given.

Procedure

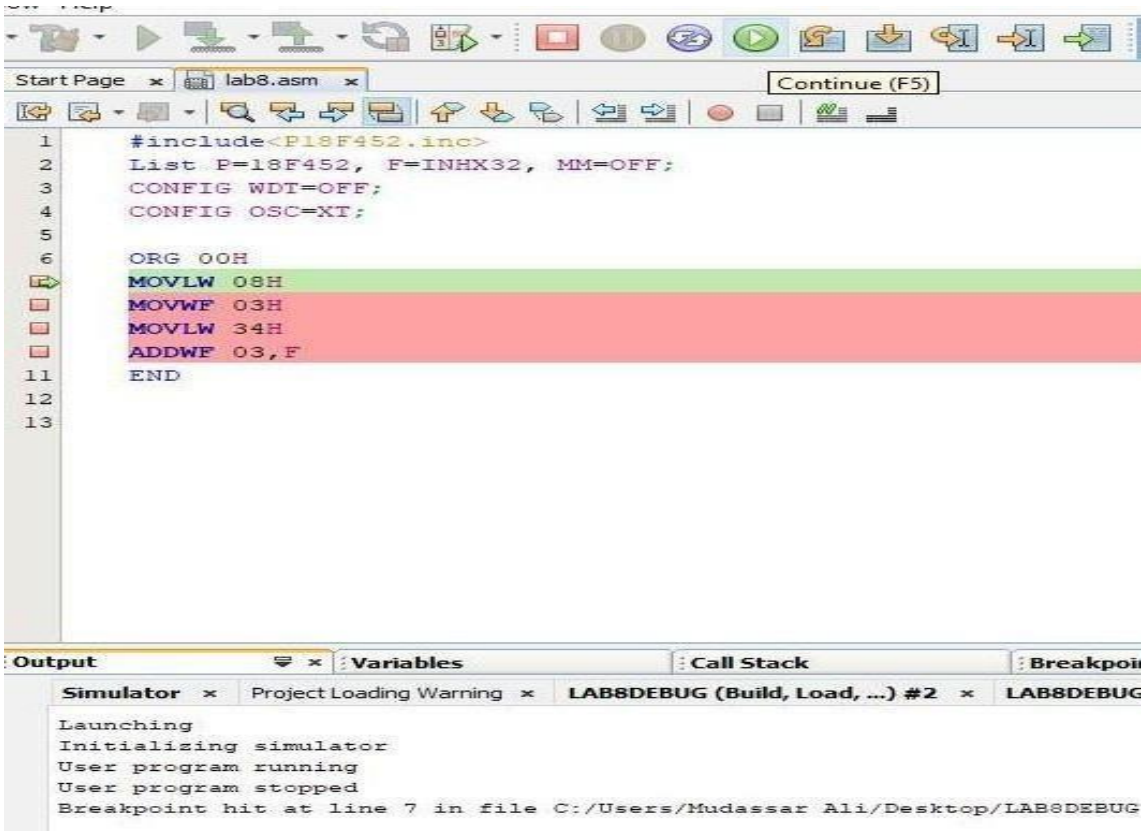
- First of all, create a new project in MPLAB.
- Now create .asm file in source folder.
- Write the code in the .asm file.



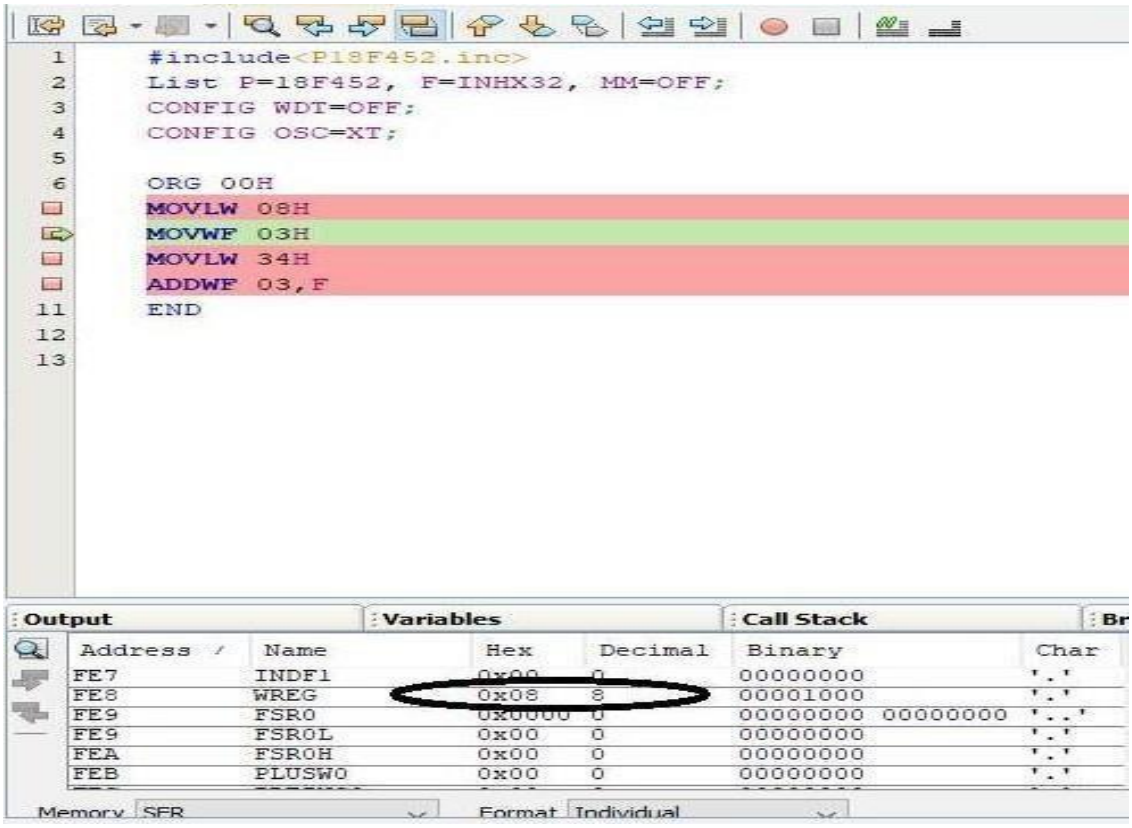
- Put breakpoint on all lines of code starting from ORG line. You can add breakpoint on a line by right clicking the corresponding numbers on the line.



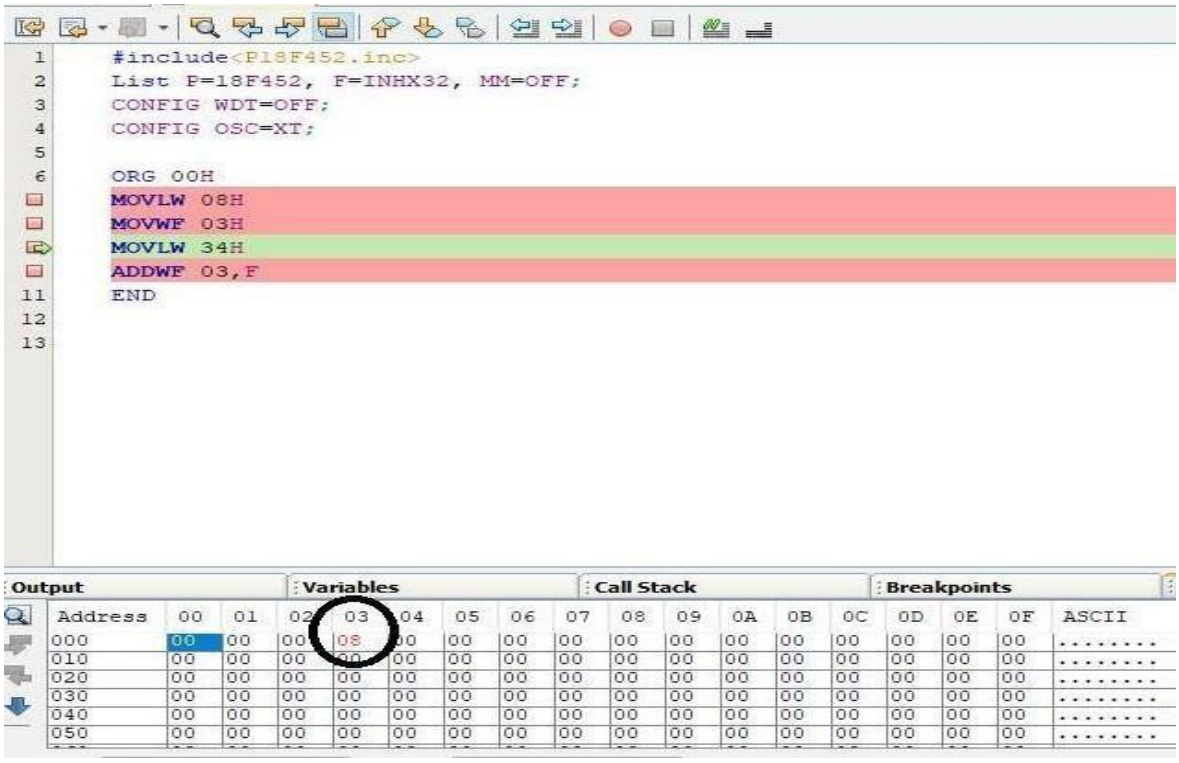
- Now, moving the next step, for this press F5 or press the ‘continuous icon’ by mouse. The line moves further.



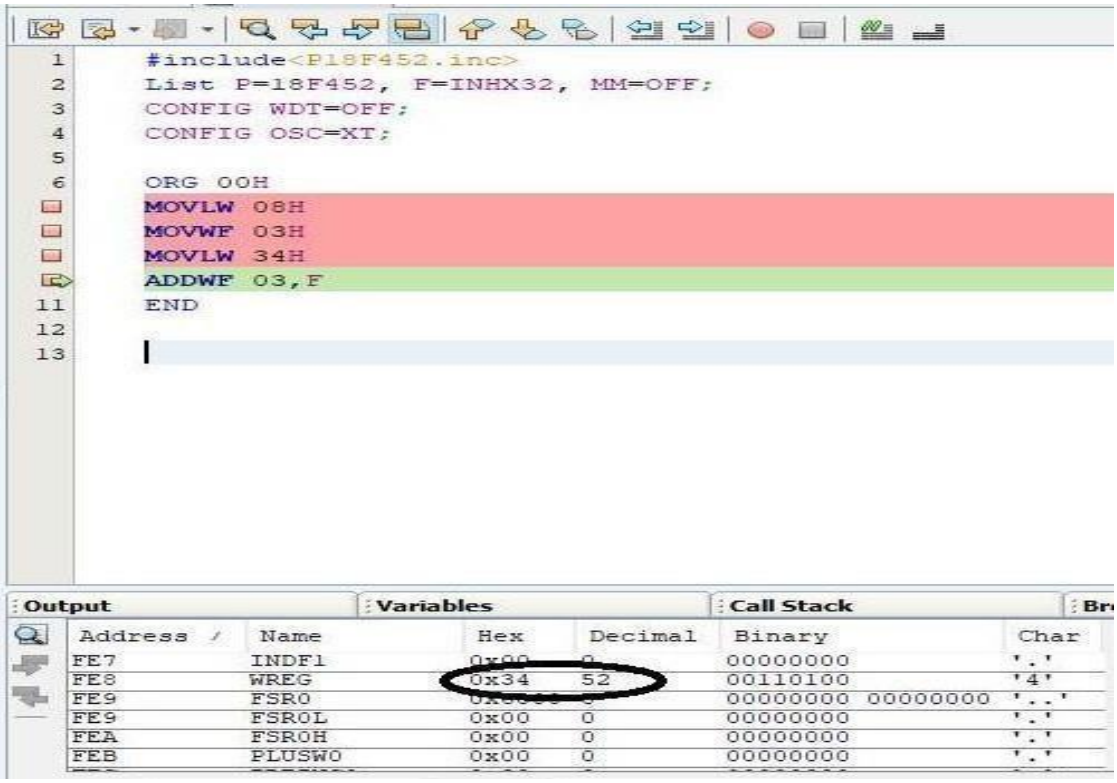
- Execution of line 7 will move the value 08 in the WREG.



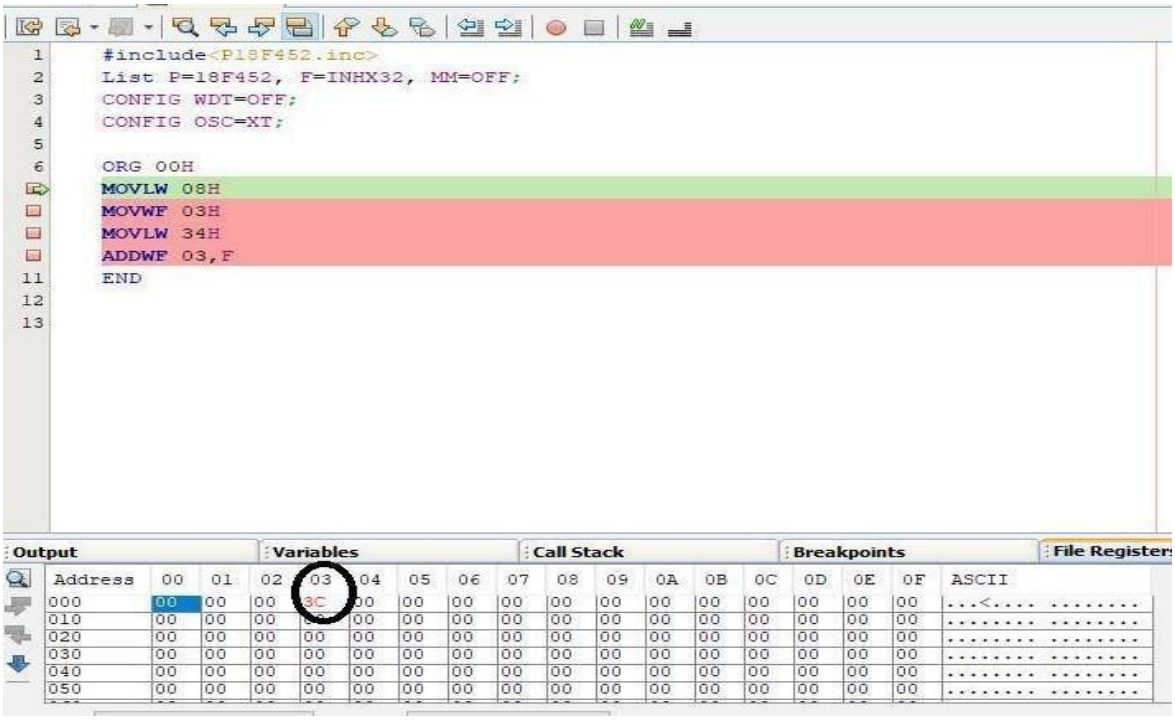
- Then line 8 execute and value 08 move in the SFR's at location 03H.



- Then line 9 execute and WREG update by new value 34.



- After this line 10 execute and add WREG and SFR's values and result value (3C) store in file register at 03H location. And cursor go to back first break point line 7.



And similarly, if you want to debug other lines set break point and proceed onward as described above.

How to storing a value in working and file register and learn the basic commands in MPLAB IDXE

Theory:

- **Assembly Language Programming:**

  An assembly language, often abbreviated .asm, in a low-level programming language for a computer, or other programmable device, in which there is a very strong correspondence between the language and the architecture machine code instructions. Each assembly language is specific to a particular computer architecture.

- **Assembler:**

  An assembler program creates object code by translating combinations of mnemonics and syntax for operations and addressing modes into their numerical equivalents. This representation typically includes an operation codes as well as other control bits and data. The assembler also calculates constant expressions and resolves symbolic names for memory locations and other entities.

- **Program Structure:**

Basic elements of a program line in MPLAB are, Label, Mnemonics, Operands and comments and they are structured as:
[Label] [Mnemonics] [Operands] [; Comments]
Mnemonics and Operands are necessary elements of program line while other two can be used when needed.

- **Working register:**

working register is a special register in the PIC construction, that is used for 2 operands, ALU operations, and can also be the destination for any ALU operation.

- **File register:**

File register is used for indirect file register addressing. The address of the register required is placed in the file select register. When data is written to or read from file location, it is actually written to or read from the file register pointed to by file select register.

Use different commands in MPLAB for storing values in working and file registers.

MOVLW Command:

MOVLW command use for copy the contents contained in the literal value to the working register. Syntax,

MOVLW k

K is literal value 0<k>255

ADDLW Command:

ADDLW command used for performing addition operation, adding a constant with W register. Syntax,

ADDLW k Given

constant(k) added with W register.

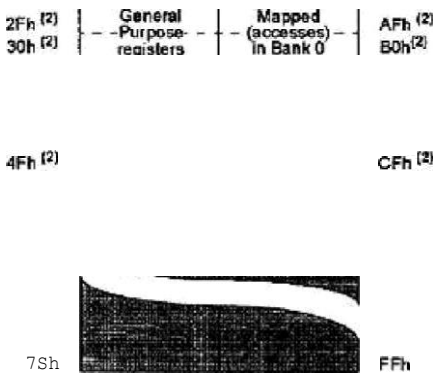
MOVWF command:

MOVWF command used for move the data from W register to flag register F.
Syntax,

MOVWF f

‘fi is file register any location.

FT* Address £Xlh	lndired eddi."’	In4troct addr.”’	F-lie Adéreec th
OZN	PCL	FCL	BZh
OOh	4T*TU0	STATUS	O3t
04h	**FSR**	**FSR**	**84h**
05h	PDRTA	TRISA	85t
Obh	PDRTB	TRISB	**86h**
0Bb	EEDATA	EECON1	B8h
09h	EEADR	EECON/’!	Nh
OAH	PGLATH	IPCLATH	BAh
OBE	**INTCDN**	**INTCON**	**BBh**



DDWF command:

|ADDWF is also used for performing the addition operation. This ADDWF instruction adds the 'file register value with working register value.

Procedure

i. First of all, create a new project in MPLAB.
ii. Now create .asm file in source folder.
iii. Write the code in the .asm file.

iV. Execute the program step by step and check the value of working and SFR registers.

V. ORG give information about starting of program.

YL When second command **“MOVLW 33H”** execute value 33 store in WREG.

vii. Then third command **"ADDLW l6H"** execute value of WREG and 16 add and store in WREG know the value stored in WREG is 4S.

3ri1L Then forth command **"MOVWF 02H"** execute, value that stored in WREG copy in SFR at 02H location.

ü. When fifth command "ADDWF 02H" execute, value that stored in WREG and file register (SFR) add and result store in SFR. And value of WREG same as it is.

x. When sixth command **"ADDWF 02H.** W" execute, value that stored in WREG and file !register (SFR) add and result store in WREG. And value of SFR same as it is..

x'l. Then seventh command **"MOVFF 02H, 04H"** execute data copy 02H location to 04H location in SFR.

m". After this eighth command "MOVFF 02H, 04H" execute data copy 02H location to 03H location in SFR.

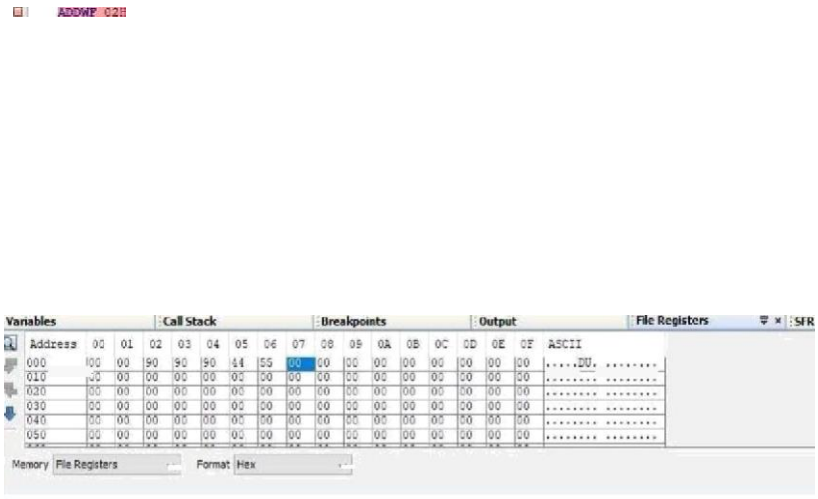
xiii. Then ninth command **"MOVLW 44H"** execute upgrade the value of WREG new value in WREG is 44H.

xiv. After this tenth command "MOVWF 05H" execute and value of WREG copy in SFR at |05H location.

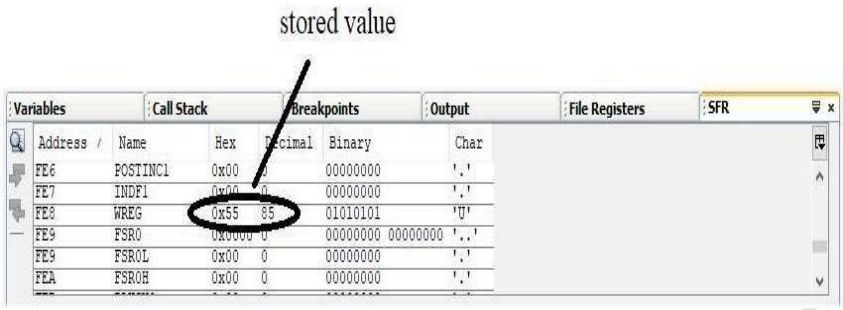
xv. Then eleventh command **"MOVLW 44H"** execute upgrade the value of WREG new value in WREG is 55H.

xvi. After this twelfth command **"MOVWF 06H"** execute and value of WREG copy in SFR eat 06H location.

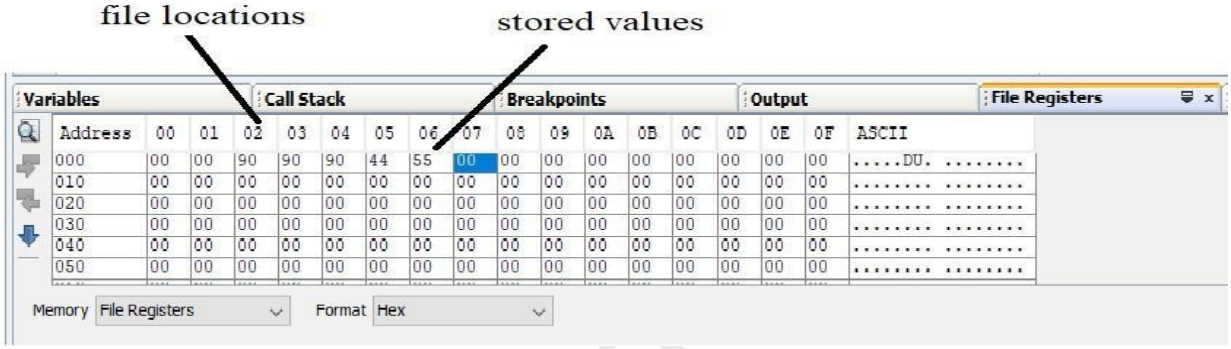
xvii. lThen END instruction that show, program end.



Main screen



Values stored in File Register



Value stored in WREG at the end:

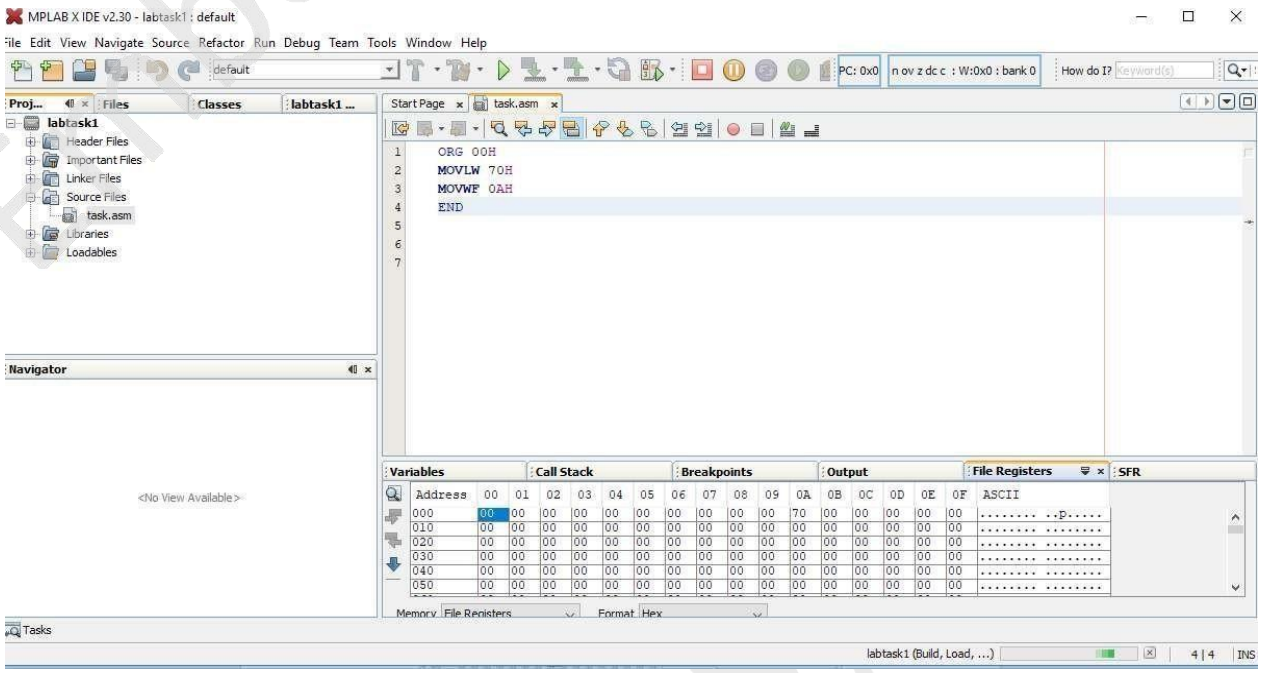
Task:

Save last two digits of your roll number into the register which is equal to first two digits of your birthday.

Procedure

i. First of all, create a new project in MPLAB.

ii. Now create .asm file in source folder.



EXPERIMENT-2

Aim: Write programs in assembly language to illustrate the following arithmetic operations

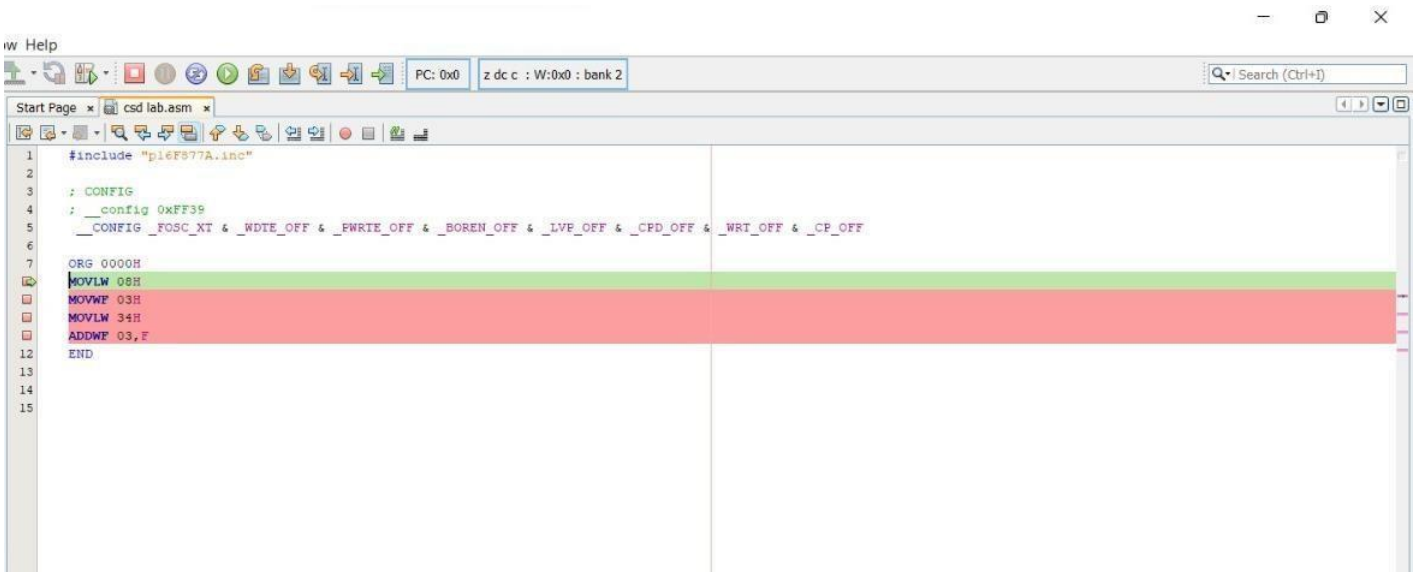
a) ADD values in assembly language.
b) SUB values in assembly language

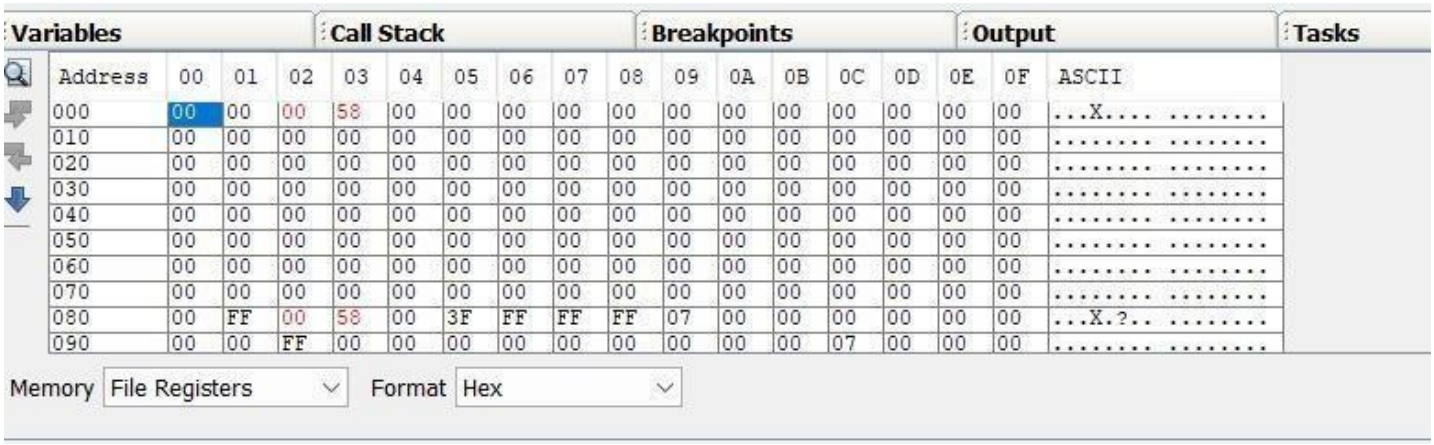
CODE:

ADDITION:

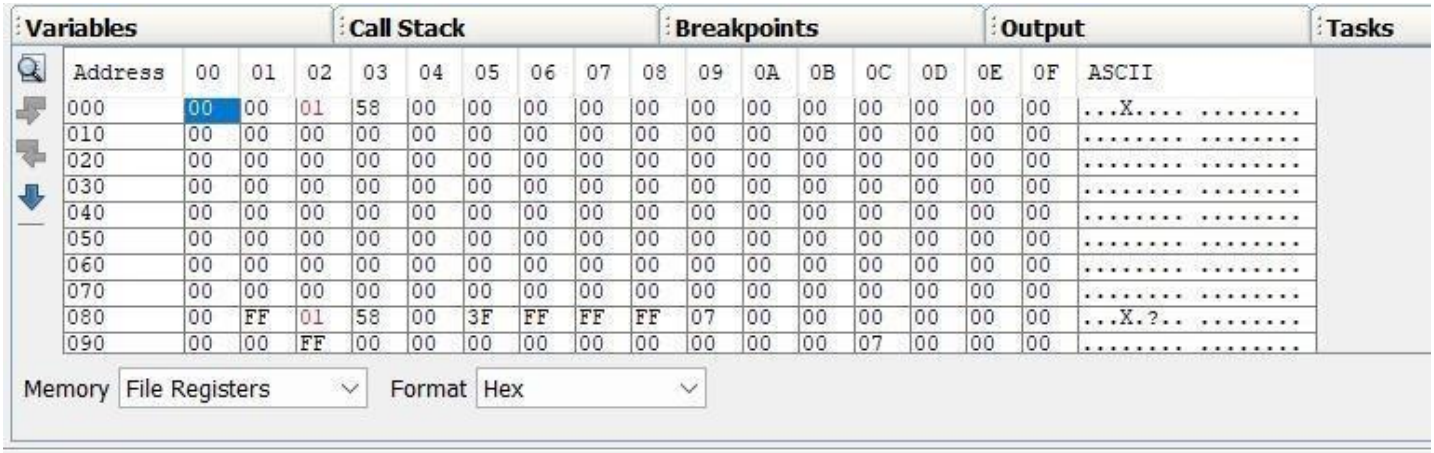
```
ORG 0000H
MOVLW 08H
MOVWF 03H
MOVLW 34H
ADDWF 03,F
```

OUTPUT: (Applying Breakpoints)

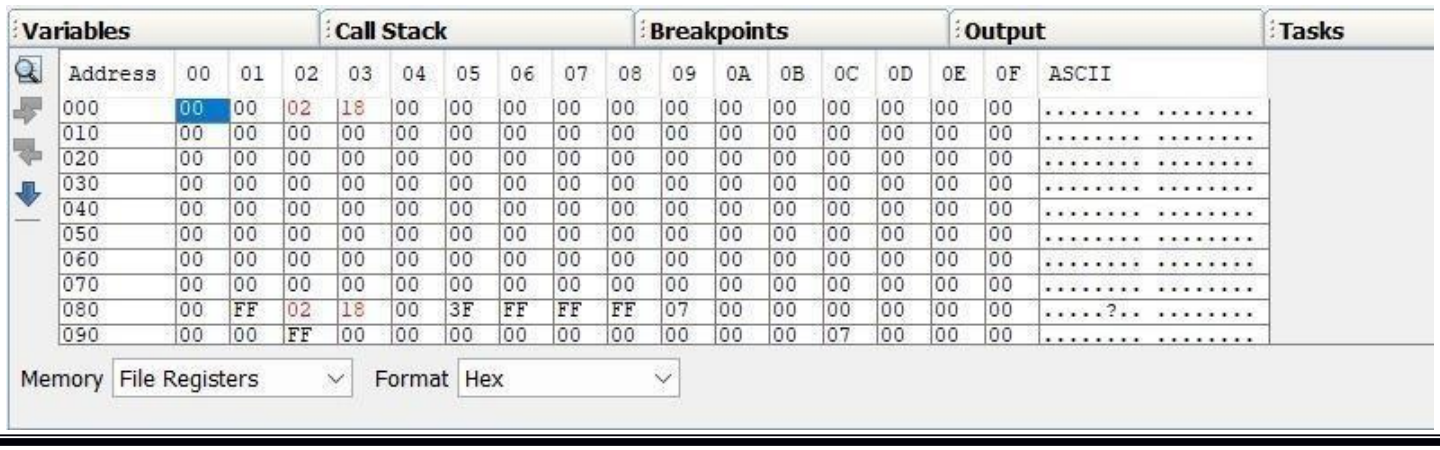




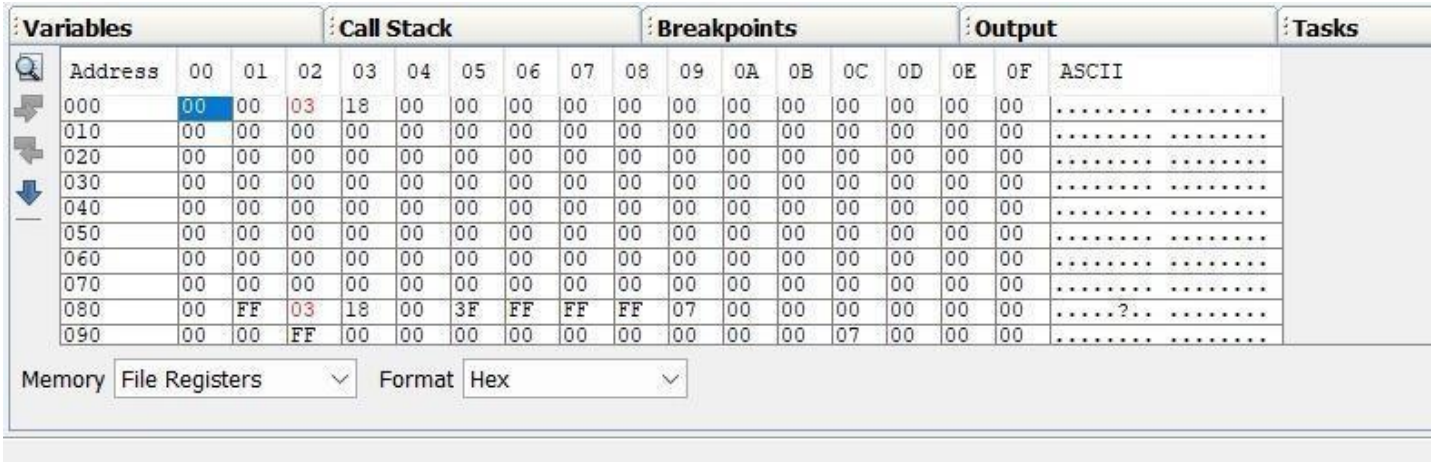

indow Help
PC: 0x1
z dc c : W:0x8 : bank 2
Search (Ctrl+I)
Start Page
csd lab.asm
#include "p16F877A.inc"
; CONFIG
; __config 0xFF39
__CONFIG _FOSC_XT & _WDTE_OFF & _PWRTE_OFF & _BOREN_OFF & _LVP_OFF & _CPD_OFF & _WRT_OFF & _CP_OFF
ORG 0000H
MOVLW 08H
MOVWF 03H
MOVLW 34H
ADDWF 03,F
END

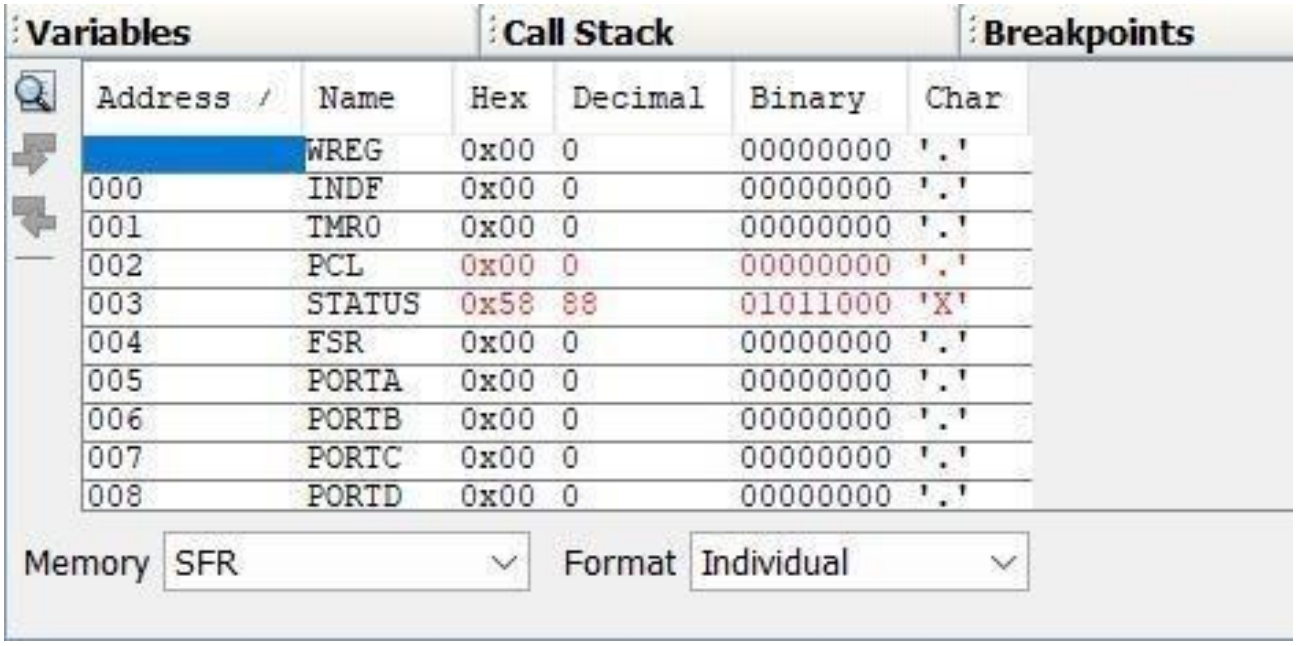

Variables
Call Stack
Breakpoints
Output
Tasks
Address 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F ASCII
000 00 00 01 58 00 00 00 00 00 00 00 00 00 00 00 00 ...X.... ....
010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
080 00 FF 01 58 00 3F FF FF FF 07 00 00 00 00 00 00 ...X.?.. ....
090 00 00 FF 00 00 00 00 00 00 00 00 00 00 07 00 00 00 ....
Memory File Registers
Format Hex


w Help
PC: 0x2
z dc c : W:0x8 : bank 0
Search (Ctrl+I)
Start Page
csd lab.asm
#include "p16F877A.inc"
; CONFIG
; __config 0xFF39
__CONFIG _FOSC_XT & _WDTE_OFF & _PWRTE_OFF & _BOREN_OFF & _LVP_OFF & _CPD_OFF & _WRT_OFF & _CP_OFF
ORG 0000H
MOVLW 08H
MOVWF 03H
MOVLW 34H
ADDWF 03,F
END


Variables
Call Stack
Breakpoints
Output
Tasks
Address 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F ASCII
000 00 00 02 18 00 00 00 00 00 00 00 00 00 00 00 00 ....
010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
080 00 FF 02 18 00 3F FF FF FF 07 00 00 00 00 00 00 .....?.. ....
090 00 00 FF 00 00 00 00 00 00 00 00 00 00 07 00 00 00 ....
Memory File Registers
Format Hex

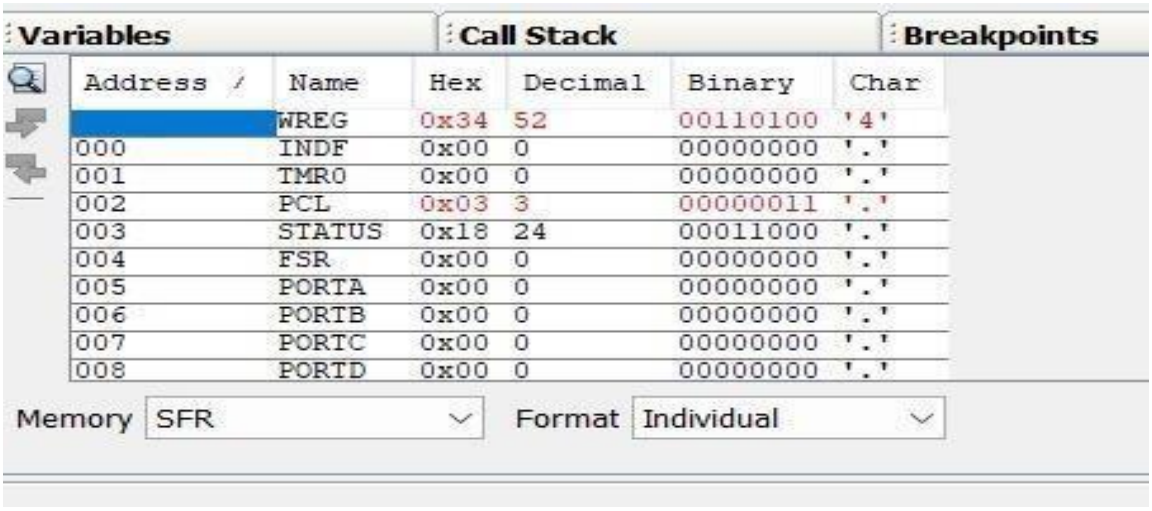

w Help
PC: 0x3
z dc c : W:0x34 : bank 0
Search (Ctrl+I)
Start Page
csd lab.asm
#include "p16F877A.inc"
; CONFIG
; __config 0xFF39
__CONFIG _FOSC_XT & _WDTE_OFF & _PWRTE_OFF & _BOREN_OFF & _LVP_OFF & _CPD_OFF & _WRT_OFF & _CP_OFF
ORG 0000H
MOVLW 08H
MOVWF 03H
MOVLW 34H
ADDWF 03,F
END


Variables
Call Stack
Breakpoints
Output
Tasks
Address 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F ASCII
000 00 00 03 18 00 00 00 00 00 00 00 00 00 00 00 00 ....
010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
080 00 FF 03 18 00 3F FF FF FF 07 00 00 00 00 00 00 .....?.. ....
090 00 00 FF 00 00 00 00 00 00 00 00 00 00 07 00 00 00 ....
Memory File Registers
Format Hex


Variables
Call Stack
Breakpoints
Address / Name Hex Decimal Binary Char
WREG 0x00 0 00000000 '.'
000 INDF 0x00 0 00000000 '.'
001 TMR0 0x00 0 00000000 '.'
002 PCL 0x00 0 00000000 '.'
003 STATUS 0x58 88 01011000 'X'
004 FSR 0x00 0 00000000 '.'
005 PORTA 0x00 0 00000000 '.'
006 PORTB 0x00 0 00000000 '.'
007 PORTC 0x00 0 00000000 '.'
008 PORTD 0x00 0 00000000 '.'
Memory SFR
Format Individual


Variables
Call Stack
Breakp
Address
Name
Hex
Decimal
Binary
Char
WREG 0x08 8 00001000 '.'
000 INDF 0x00 0 00000000 '.'
001 TMR0 0x00 0 00000000 '.'
002 PCL 0x02 2 00000010 '.'
003 STATUS 0x18 24 00011000 '.'
004 FSR 0x00 0 00000000 '.'
005 PORTA 0x00 0 00000000 '.'
006 PORTB 0x00 0 00000000 '.'
007 PORTC 0x00 0 00000000 '.'
008 PORTD 0x00 0 00000000 '.'
Memory SFR
Format Individual


Variables
Call Stack
Breakpoin
Address
Name
Hex
Decimal
Binary
Char
WREG 0x08 8 00001000 '.'
000 INDF 0x00 0 00000000 '.'
001 TMR0 0x00 0 00000000 '.'
002 PCL 0x01 1 00000001 '.'
003 STATUS 0x58 88 01011000 'X'
004 FSR 0x00 0 00000000 '.'
005 PORTA 0x00 0 00000000 '.'
006 PORTB 0x00 0 00000000 '.'
007 PORTC 0x00 0 00000000 '.'
008 PORTD 0x00 0 00000000 '.'
Memory SFR
Format Individual


Variables
Call Stack
Breakpoints
Address / Name Hex Decimal Binary Char
WREG 0x34 52 00110100 '4'
000 INDF 0x00 0 00000000 '.'
001 TMR0 0x00 0 00000000 '.'
002 PCL 0x03 3 00000011 '.'
003 STATUS 0x18 24 00011000 '.'
004 FSR 0x00 0 00000000 '.'
005 PORTA 0x00 0 00000000 '.'
006 PORTB 0x00 0 00000000 '.'
007 PORTC 0x00 0 00000000 '.'
008 PORTD 0x00 0 00000000 '.'
Memory SFR
Format Individual

EXPERIMENT-3,4

Aim: **a)** In MPLAB X IDE v2.00, Working register of PIC-18F has a value of 0FFH.Write a program to add 1H 5 times with the content of working register.

b) Write a program to add 5H value with the contents of WREG & check the status of SRF status.

CODE:

1.
```
#include "p16F877A.inc"

 ; CONFIG

 ;_config 0xFFFF

  __CONFIG _FOSC_EXTRC & _WDTE_ON & _PWRTE_OFF & _BOREN_ON & _LVP_ON & _CPD_OFF & _WRT_OFF & _CP_OFF

 ORG 0000H

 MOVLW OFFH

 ADDLW 01H

 ADDLW 01H

 ADDLW 01H

 ADDLW 01H

 ADDLW 01H

 END
```

2.
```
ORG 0000H

 MOVLW OFFH

 ADDLW 05H

 END
```

Start Page | csd lab.asm | KANISHK.asm | csd lab 2.asm | EXP3B.asm

```
1   #include "p16F877A.inc"
2
3   ; CONFIG
4   ; __config 0xFF3B
5    __CONFIG _FOSC_EXTRC & _WDTE_OFF & _PWRTE_OFF & _BOREN_OFF & _LVP_OFF & _CPD_OFF & _WRT_OFF & _CP_OFF
6   ORG 0000H
7   MOVLW 0FFH
8   ADDLW 01H
9   ADDLW 01H
10  ADDLW 01H
11  ADDLW 01H
12  ADDLW 01H
13  END
14
15
16
17
```

Variables | Call Stack | Breakpoin

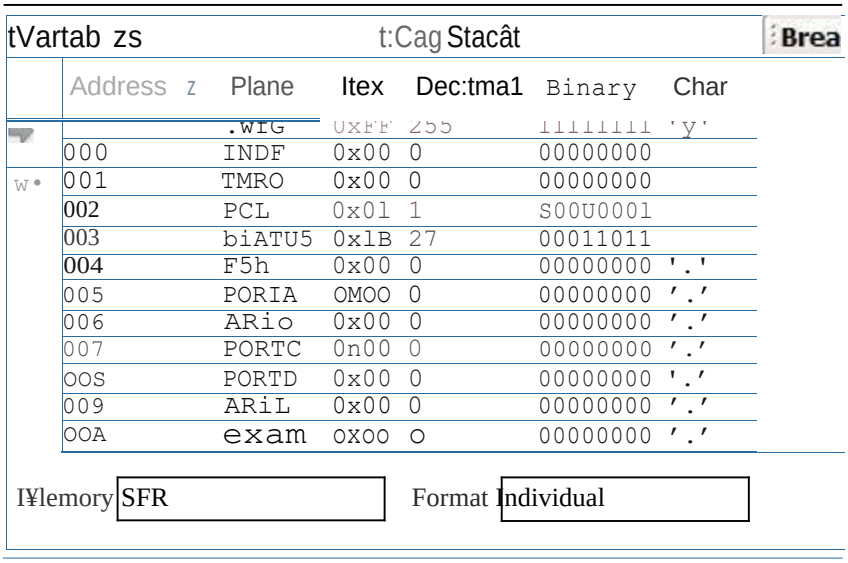
Address	Name	Hex	Decimal	Binary	Char
	WREG	0xFF	255	11111111	'ÿ'
000	INDF	0x00	0	00000000	'.'
001	TMR0	0x00	0	00000000	'.'
002	PCL	0x01	1	00000001	'.'
003	STATUS	0x1B	27	00011011	'.'
004	FSR	0x00	0	00000000	'.'
005	PORTA	0x00	0	00000000	'.'
006	PORTB	0x00	0	00000000	'.'
007	PORTC	0x00	0	00000000	'.'
008	PORTD	0x00	0	00000000	'.'
009	PORTE	0x00	0	00000000	'.'
00A	PCLATH	0x00	0	00000000	'.'

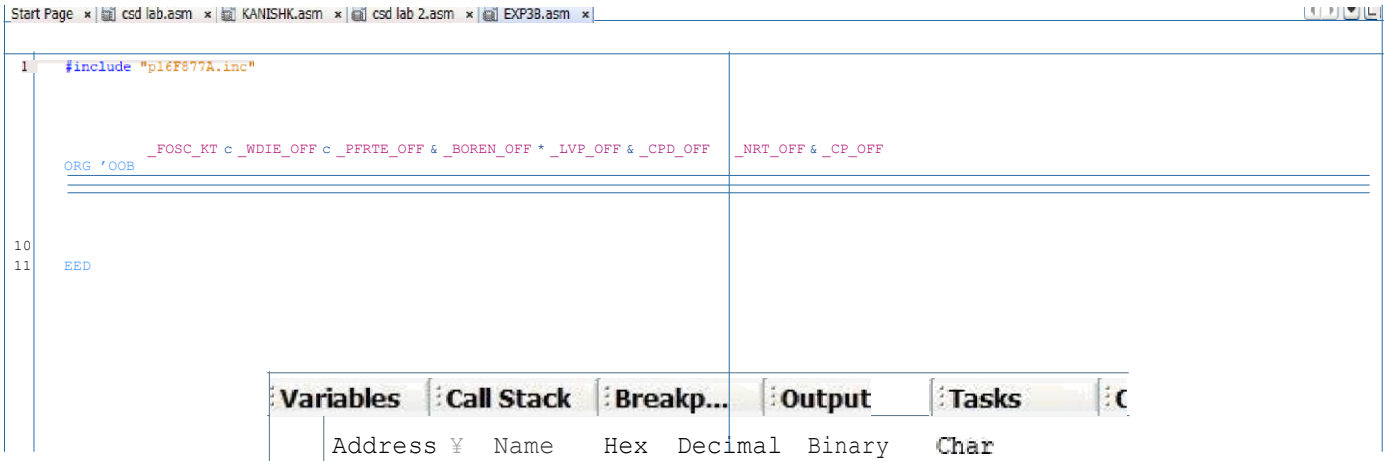
Memory SFR Format Individual

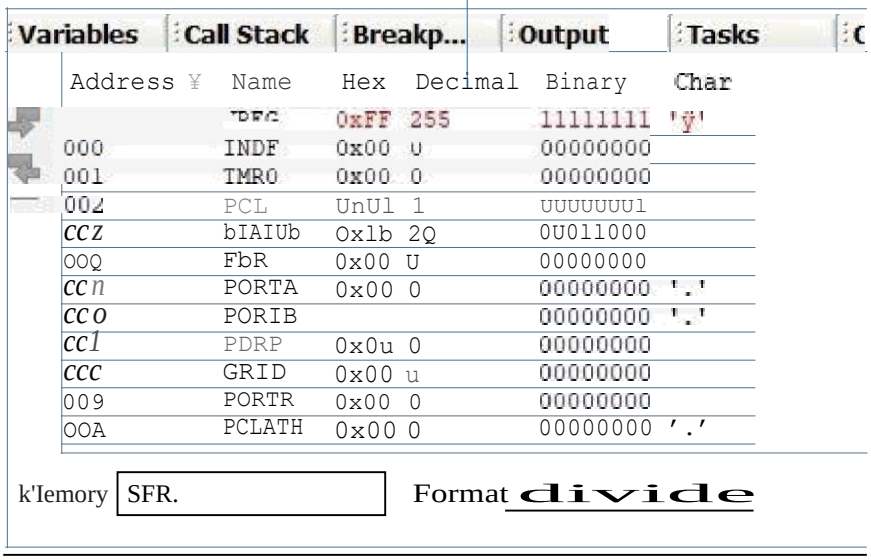
Variables | Call Stack | Brea

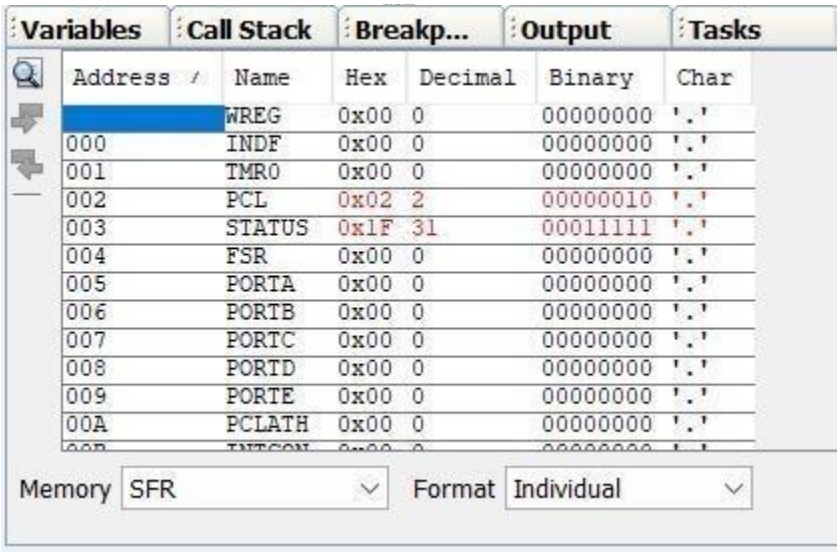
Address	Name	Hex	Decimal	Binary	Char
	WREG	0x00	0	00000000	'.'
000	INDF	0x00	0	00000000	'.'
001	TMR0	0x00	0	00000000	'.'
002	PCL	0x00	0	00000000	'.'
003	STATUS	0x1B	27	00011011	'.'
004	FSR	0x00	0	00000000	'.'
005	PORTA	0x00	0	00000000	'.'
006	PORTB	0x00	0	00000000	'.'
007	PORTC	0x00	0	00000000	'.'
008	PORTD	0x00	0	00000000	'.'
009	PORTE	0x00	0	00000000	'.'
00A	PCLATH	0x00	0	00000000	'.'

Memory SFR Format Individual


tVartab zs
t:Cag Stacât
Brea
Address z
Plane
Itex
Dec:tma1
Binary
Char
.WIG 0xFF 255 11111111 'y'
000 INDF 0x00 0 00000000
001 TMRO 0x00 0 00000000
002 PCL 0x01 1 S00U0001
003 biATU5 0x1B 27 00011011
004 F5h 0x00 0 00000000 '.'
005 PORIA OMOO 0 00000000 '.'
006 ARio 0x00 0 00000000 '.'
007 PORTC 0n00 0 00000000 '.'
OOS PORTD 0x00 0 00000000 '.'
009 ARiL 0x00 0 00000000 '.'
OOA exam 0X00 o 00000000 '.'
I¥lemory SFR
Format Individual


Start Page
csd lab.asm
KANISHK.asm
csd lab 2.asm
EXP38.asm
#include "p16F877A.inc"
_FOSC_KT c _WDIE_OFF c _PFRTE_OFF & _BOREN_OFF * _LVP_OFF & _CPD_OFF _NRT_OFF & _CP_OFF
ORG 'OOB
EED

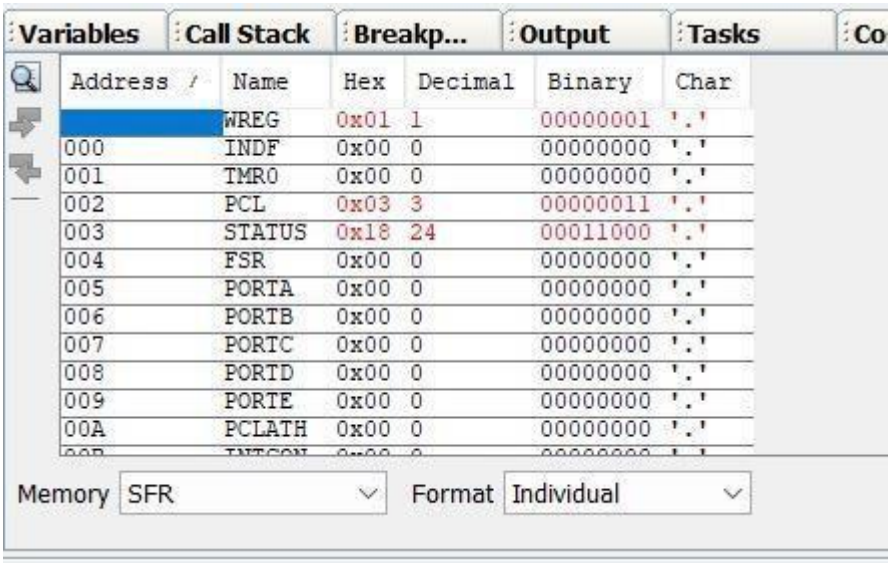

Variables
Call Stack
Breakp...
Output
Tasks
Address ¥
Name
Hex
Decimal
Binary
Char
0xFF 255 11111111 'ÿ'
000 INDF 0x00 0 00000000
001 TMRO 0x00 0 00000000
002 PCL UnUl 1 UUUUUUU1
ccz bIAIUb Oxlb 2Q 0U0ll000
OOQ FbR 0x00 U 00000000
ccn PORTA 0x00 0 00000000 '.'
cco PORIB 00000000 '.'
cc1 PDRP 0x0u 0 00000000
ccc GRID 0x00 u 00000000
009 PORTR 0x00 0 00000000
OOA PCLATH 0x00 0 00000000 '.'
k'Iemory SFR.
Format divide



Variables | Call Stack | Breakp... | Output | Tasks

Address	Name	Hex	Decimal	Binary	Char
	WREG	0x00	0	00000000	'.'
000	INDF	0x00	0	00000000	'.'
001	TMR0	0x00	0	00000000	'.'
002	PCL	0x02	2	00000010	'.'
003	STATUS	0x1F	31	00011111	'.'
004	FSR	0x00	0	00000000	'.'
005	PORTA	0x00	0	00000000	'.'
006	PORTB	0x00	0	00000000	'.'
007	PORTC	0x00	0	00000000	'.'
008	PORTD	0x00	0	00000000	'.'
009	PORTE	0x00	0	00000000	'.'
00A	PCLATH	0x00	0	00000000	'.'

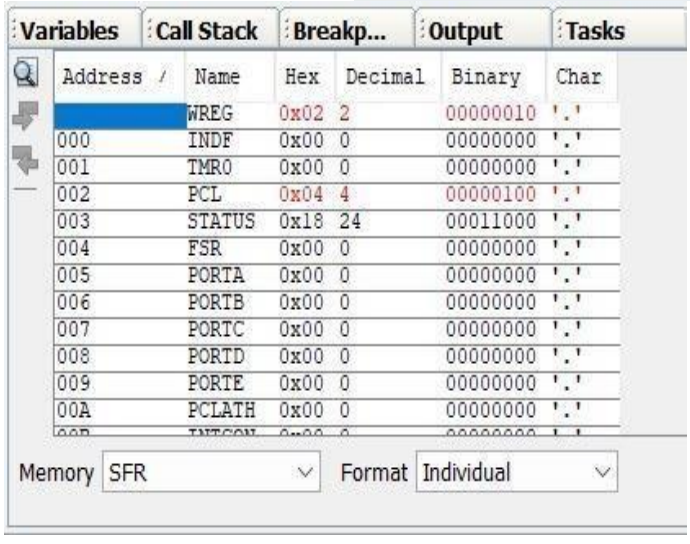
Memory SFR Format Individual



Variables | Call Stack | Breakp... | Output | Tasks | Co

Address	Name	Hex	Decimal	Binary	Char
	WREG	0x01	1	00000001	'.'
000	INDF	0x00	0	00000000	'.'
001	TMR0	0x00	0	00000000	'.'
002	PCL	0x03	3	00000011	'.'
003	STATUS	0x18	24	00011000	'.'
004	FSR	0x00	0	00000000	'.'
005	PORTA	0x00	0	00000000	'.'
006	PORTB	0x00	0	00000000	'.'
007	PORTC	0x00	0	00000000	'.'
008	PORTD	0x00	0	00000000	'.'
009	PORTE	0x00	0	00000000	'.'
00A	PCLATH	0x00	0	00000000	'.'

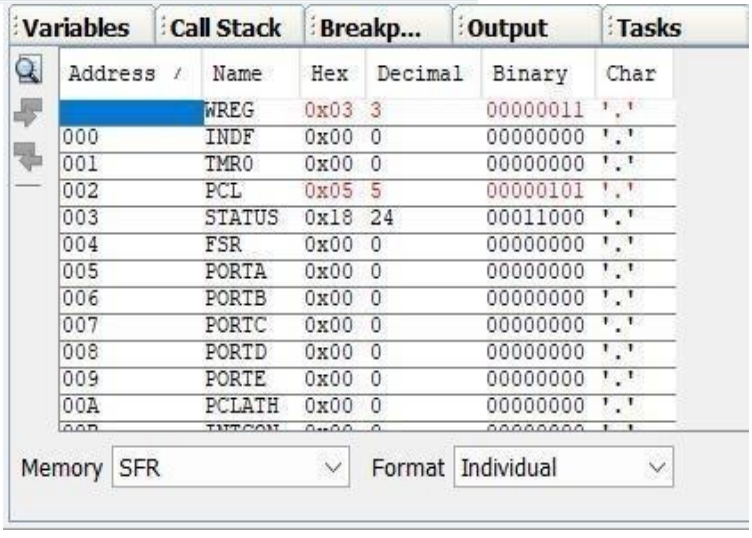
Memory SFR Format Individual



Variables | Call Stack | Breakp... | Output | Tasks

Address	Name	Hex	Decimal	Binary	Char
	WREG	0x02	2	00000010	'.'
000	INDF	0x00	0	00000000	'.'
001	TMR0	0x00	0	00000000	'.'
002	PCL	0x04	4	00000100	'.'
003	STATUS	0x18	24	00011000	'.'
004	FSR	0x00	0	00000000	'.'
005	PORTA	0x00	0	00000000	'.'
006	PORTB	0x00	0	00000000	'.'
007	PORTC	0x00	0	00000000	'.'
008	PORTD	0x00	0	00000000	'.'
009	PORTE	0x00	0	00000000	'.'
00A	PCLATH	0x00	0	00000000	'.'

Memory SFR Format Individual



Variables | Call Stack | Breakp... | Output | Tasks

Address	Name	Hex	Decimal	Binary	Char
	WREG	0x03	3	00000011	'.'
000	INDF	0x00	0	00000000	'.'
001	TMR0	0x00	0	00000000	'.'
002	PCL	0x05	5	00000101	'.'
003	STATUS	0x18	24	00011000	'.'
004	FSR	0x00	0	00000000	'.'
005	PORTA	0x00	0	00000000	'.'
006	PORTB	0x00	0	00000000	'.'
007	PORTC	0x00	0	00000000	'.'
008	PORTD	0x00	0	00000000	'.'
009	PORTE	0x00	0	00000000	'.'
00A	PCLATH	0x00	0	00000000	'.'

Memory SFR Format Individual

EXPERIMENT-5

Aim: Write a program to store a value 20H in file memory locations 30H to 34H.

CODE:

```
#include "p16F877A.inc"

; CONFIG

config OxFF3D

ORG 0000H

MOVLW 020H

MOVWE 30H

MOVWE 31H

MOVWE 32H

MOWWE 33H

MOVWE 34H

CONFIG _FOSC_XT & _WDIE_ON &
_PWRIE_OFF & _BOREN_OFF & _IVP_OFF &
_CPD_OFF & _WRT_OFF & _CP_OFF
```

Start Page | csd lab.asm | KANISHK.asm | csd lab 2.asm | EXP3B.asm | tostore 20h from 30h to 34h file register.asm

```
1   #include "p16F877A.inc"
2
3   ; CONFIG
4   ; __config 0xFF3D
5     __CONFIG _FOSC_XT & _WDTE_ON & _PWRTE_OFF & _BOREN_OFF & _LVP_OFF & _CPD_OFF & _WRT_OFF & _CP_OFF
6   ORG 0000H
7   MOVLW 020H
8   MOVWF 30H
9   MOVWF 31H
10  MOVWF 32H
11  MOVWF 33H
12  MOVWF 34H
13
14  END
15
16
17
```

Output — File Registers

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
000	00	00	05	18	00	00	00	00	00	00	00	00	00	00	00	00	....
010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
030	20	20	20	20	20	00	00	00	00	00	00	00	00	00	00	00	... .
040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
080	00	FF	05	18	00	3F	FF	FF	FF	07	00	00	00	00	00	00	.....?.. ....
090	00	00	FF	00	00	00	00	00	00	00	00	00	07	00	00	00	....
0A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....

Memory: File Registers Format: Hex

Output — File Registers

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
000	00	00	05	18	00	00	00	00	00	00	00	00	00	00	00	00	....
010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
030	20	20	20	20	20	00	00	00	00	00	00	00	00	00	00	00	... .
040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
080	00	FF	05	18	00	3F	FF	FF	FF	07	00	00	00	00	00	00	.....?.. ....
090	00	00	FF	00	00	00	00	00	00	00	00	00	07	00	00	00	....
0A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....

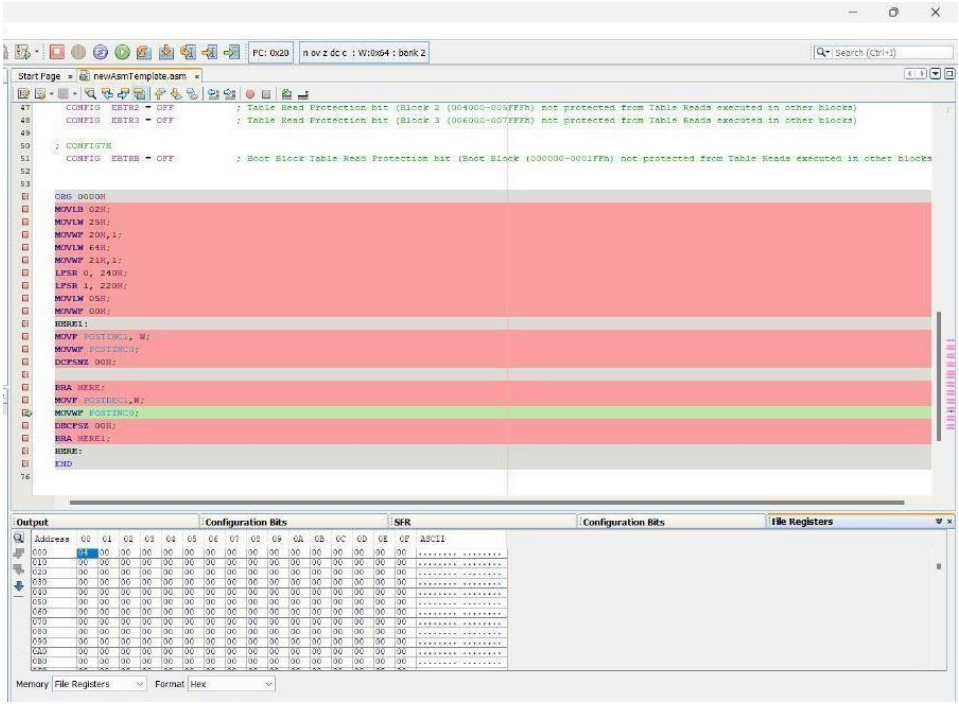
Memory: File Registers Format: Hex

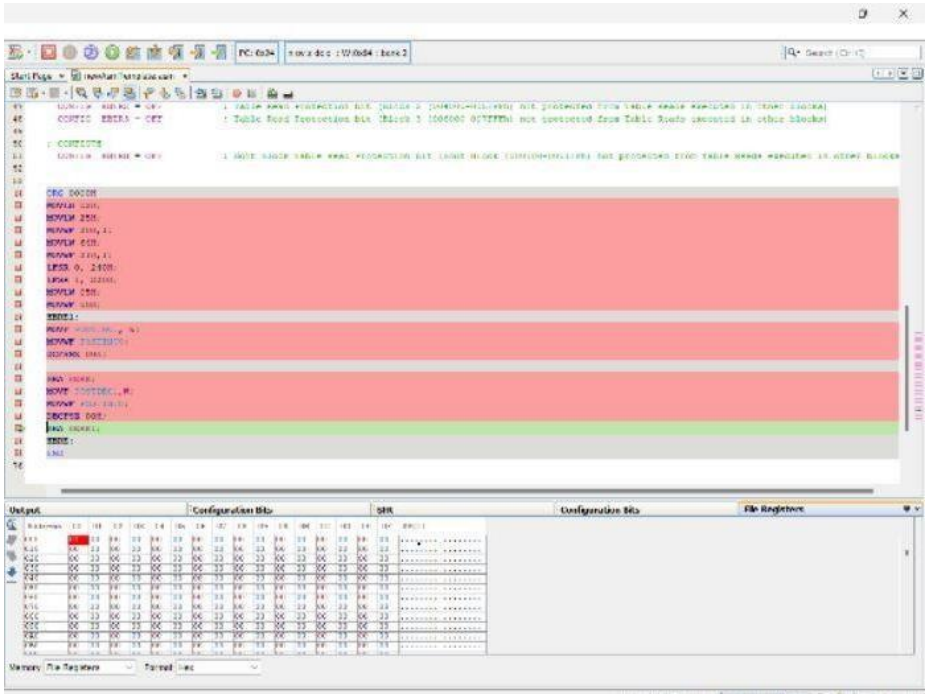
EXPERIMENT-6

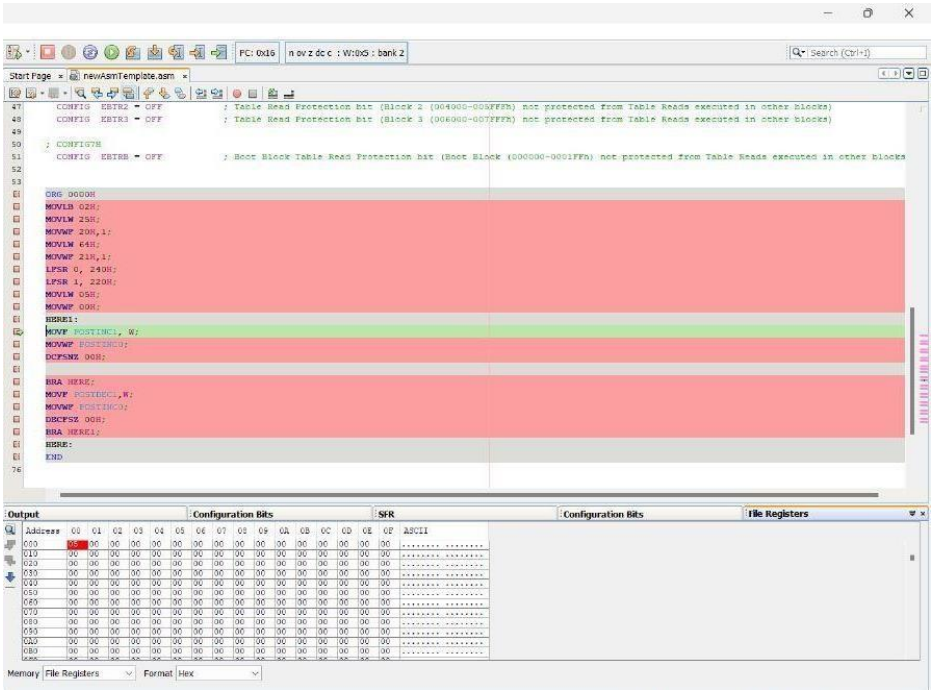
Aim: Write a program in assembly language to illustrate the addressing modes.

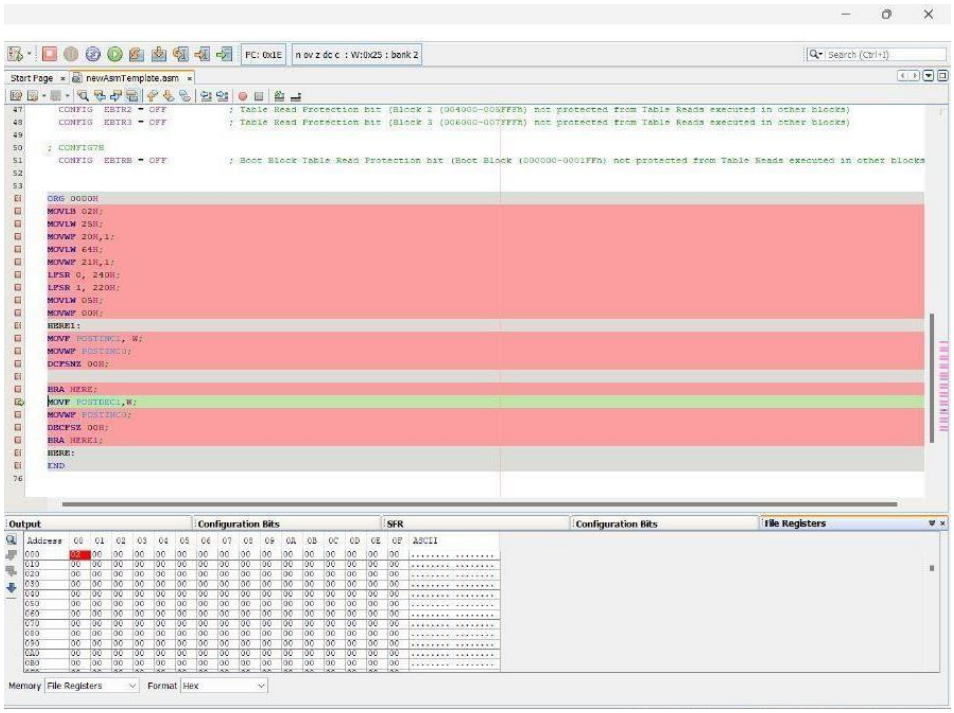
CODE:

```
ORG 0000H
MOVLW D'2'
MOVWF 12H;
HERE2: MOVLW D' 100';
MOVWF 11H;
HERE1: MOVLW D'100':
MOVWF 10H;
HERE: COMP 30H;
DECFSZ 10H;
BRA HERE;
DECEFZ 11H;
BRA HERE1;
DECEFZ 12H;
BRA HERE2:
END
```






PC: 0x16
n ov z dc c : W:0x5 : bank 2
Search (Ctrl+I)
Start Page
newAsmTemplate.asm
CONFIG EBTR2 = OFF ; Table Read Protection bit (Block 2 (004000-005FFFh) not protected from Table Reads executed in other blocks)
CONFIG EBTR3 = OFF ; Table Read Protection bit (Block 3 (006000-007FFFh) not protected from Table Reads executed in other blocks)
; CONFIG7H
CONFIG EBTRB = OFF ; Boot Block Table Read Protection bit (Boot Block (000000-0001FFh) not protected from Table Reads executed in other blocks
ORG 0000H
MOVLB 02H;
MOVLW 25H;
MOVWF 20H,1;
MOVLW 64H;
MOVWF 21H,1;
LFSR 0, 240H;
LFSR 1, 220H;
MOVLW 05H;
MOVWF 00H;
HERE1:
MOVF POSTINC1, W;
MOVWF POSTINC0;
DCFSNZ 00H;
BRA HERE;
MOVF POSTDEC1,W;
MOVWF POSTINC0;
DBCFSZ 00H;
BRA HERE1;
HERE:
END
Output
Configuration Bits
SFR
Configuration Bits
File Registers
Address 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F ASCII
Memory File Registers
Format Hex

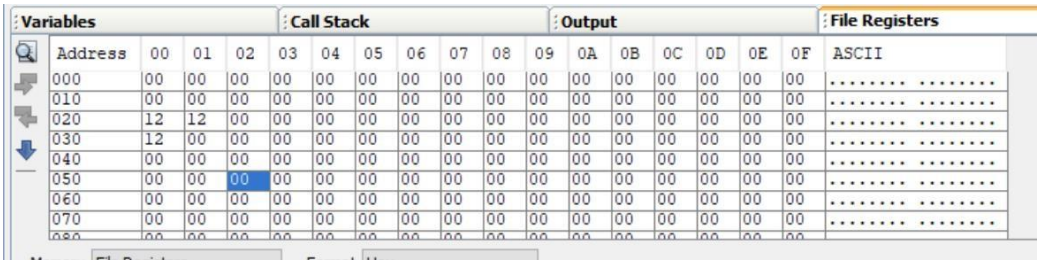

PC: 0x1E
n ov z dc c : W:0x25 : bank 2
Search (Ctrl+I)
Start Page
newAsmTemplate.asm
CONFIG EBTR2 = OFF ; Table Read Protection bit (Block 2 (004000-005FFFh) not protected from Table Reads executed in other blocks)
CONFIG EBTR3 = OFF ; Table Read Protection bit (Block 3 (006000-007FFFh) not protected from Table Reads executed in other blocks)
; CONFIG7H
CONFIG EBTRB = OFF ; Boot Block Table Read Protection bit (Boot Block (000000-0001FFh) not protected from Table Reads executed in other blocks
ORG 0000H
MOVLB 02H;
MOVLW 25H;
MOVWF 20H,1;
MOVLW 64H;
MOVWF 21H,1;
LFSR 0, 240H;
LFSR 1, 220H;
MOVLW 05H;
MOVWF 00H;
HERE1:
MOVF POSTINC1, W;
MOVWF POSTINC0;
DCFSNZ 00H;
BRA HERE;
MOVF POSTDEC1,W;
MOVWF POSTINC0;
DSCFSZ 00H;
BRA HERE1;
HERE:
END
Output
Configuration Bits
SFR
Configuration Bits
File Registers
Address 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F ASCII
000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
0A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
0B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
Memory File Registers
Format Hex

EXPERIMENT-7

AIM: Write a program in assembly language to illustrate Data Transfer Operation.

CODE:

```
; PIC18F452 Data Transfer Operations Example
; Copy a byte from memory to a register, and then copy it back to a different memory location

    ; define processor
  include "p18f452.inc"    ; include PIC18F452 header file

  ; define variables
  cblock  0x20            ; define block of memory for variables
    temp1 ; temporary variable 1
    temp2 ; temporary variable 2
  endc

  ; program code
  org     0x0000          ; start at address 0

  ; initialize variables
  movlw   0x12            ; move value 0x12 into W
  movwf   temp1           ; move W into temp1
  clrf    temp2           ; clear temp2

  ; copy byte from memory to register
  movf    temp1, W        ; move value of temp1 into W
  movwf   temp2           ; move W into temp2

  ; copy byte from register to memory
  movf    temp2, W        ; move value of temp2 into W
  movwf   0x30            ; move W into memory location 0x30

  ; end of program
  End
```



EXPERIMENT-8

AIM: Write a program in assembly language to illustrate:

a) arithmetic operations
b) logical operations.

Code:

```
#include<p18F452.inc>

ORG 0000H

MOVLW OD2H;

MOVWF 20H;

MOVLW OASH;

ADDWF 20H, W:

BNC HERE;

INCE 40H;

HERE: MOVWF 41H;

END
```

Code:

```
: PICI6F8774 Configuration Bit Settings ASM

source line config statements #include

"p16F877A.inc”

;CONFIG

;_contig OxFF39
```

```
_CONFIG FOSC XT &

_WDIE_OFF & _PWRIE_OFF & _BOREN _OFF & _LVP_OFF &
_CPD_OFF & _WRT_OFF & _CP_OFF ORG

0000H

MOVLW      1100

ADDWF     05H,0

END


:PICI6F8774 Configuration Bit Settings

;ASM source line config statements

;include p16F877A.inc" CONFIG

config OxFF39

CONFIG _FOSC_XI & _WDIE_OFF & _PIRIE_OFF &
_BOREN_OFF & _IVP_OFF & _CPD_OFF & _WRI_OFF &
_CP_OFF ORG

0000H MOVLW

1100

XORWF 06H, 0

END
```


```
; PIC16F877A Configuration Bit Settings

; ASM source line config statements

#include "p16F877A.inc"
```

memory rae Registers Format | Hex

Start Page | csd lab.asm | KANISHK.asm | csd lab 2.asm | EXP38.asm | tostore 20h from 30h to 34h file register.asm | as.asm

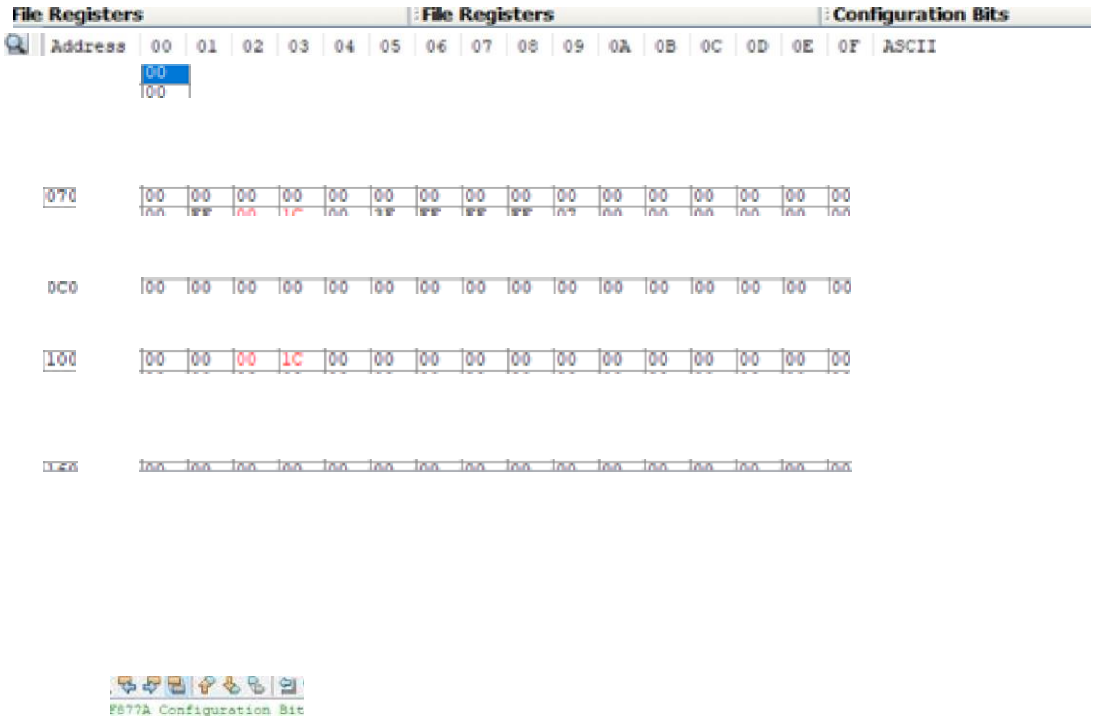
```
1   ; PIC16F877A Configuration Bit Settings
2
3   ; ASM source line config statements
4
5   #include "p16F877A.inc"
6
7   ; CONFIG
8   ; __config 0xFF39
9    __CONFIG _FOSC_XT & _WDTE_OFF & _PWRTE_OFF & _BOREN_OFF & _LVP_OFF & _CPD_OFF & _WRT_OFF & _CP_OFF
10
11  ORG 0000H
12  MOVLW 1100
13  ADDWF 05H,0
14  END
15
16
17
```

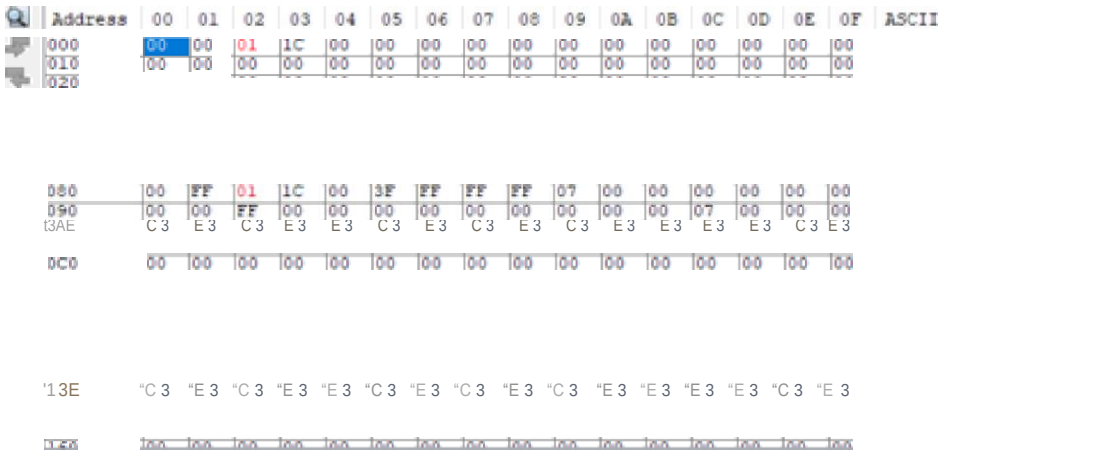
Output | File Registe

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
000	00	00	00	1C	00	00	00	00	00	00	00	00	00	00	00	00	....
010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
080	00	FF	00	1C	00	3F	FF	FF	FF	07	00	00	00	00	00	00	.....?.. ....
090	00	00	FF	00	00	00	00	00	00	00	00	00	07	00	00	00	....
0A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
100	00	00	00	1C	00	00	00	00	00	00	00	00	00	00	00	00	....
110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
150	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....

Memory File Registers Format Hex

```
1   ;PIC16F877A Configuration Bit Settings
2
3   ; ASM source line config statements
4
5   #include "p16F877A.inc"
6
7   ; CONFIG
8   ; __config 0xFF39
9    __CONFIG _FOSC_XT & _WDTE_OFF & _PWRTE_OFF & _BOREN_OFF & _LVP_OFF & _CPD_OFF & _WRT_OFF & _CP_OFF
10
11
12  ORG 0000H
13  MOVLW 1100
14  ;ANDWF 05H,0
15  XORWF 06H,0
16  END
17
18
19
```


File Registers
File Registers
Configuration Bits
Address 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F ASCII
070
0C0
100
F877A Configuration Bit

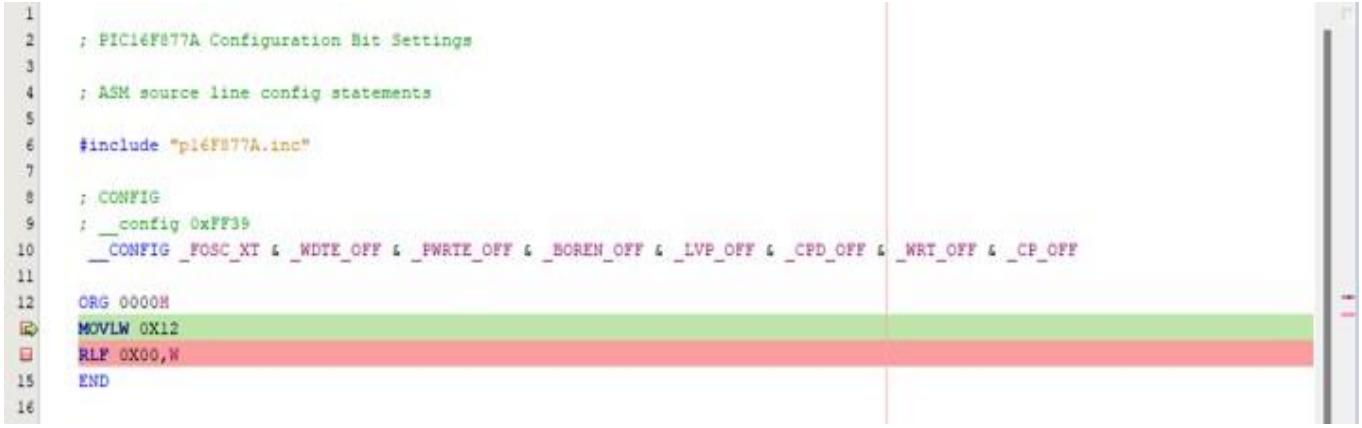

Address 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F ASCII
000
010
020
080
090
0C0
13E

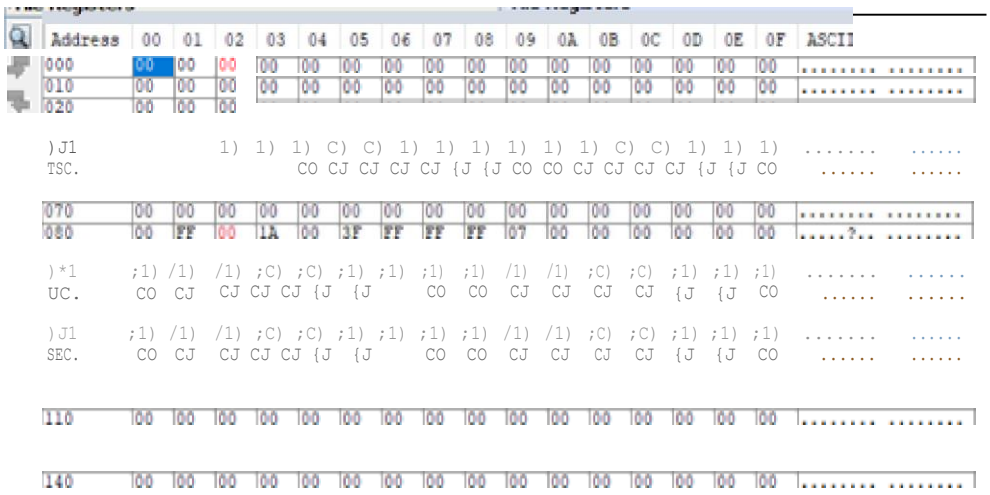
EXPERIMENT-9

AIM: Write a program in assembly language to illustrate shift operation.

CODE:

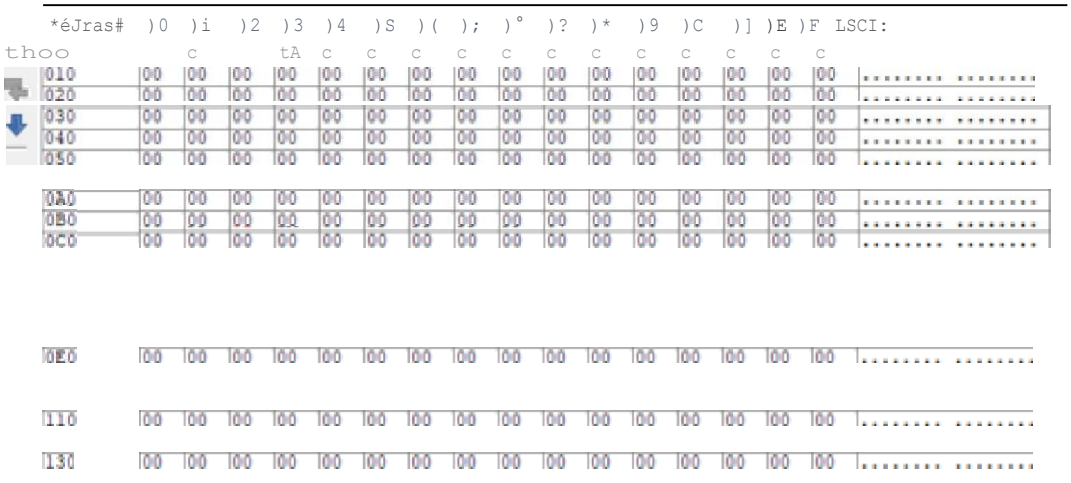
```
; PICI6F877A Configuration Bit Settings
;ASM source line config statements
#include "p16F877A.inc"
: CONFIG
;_config OxFF39
_CONFIG _FOSC _XT & _WDTE_OFF &
PWRTE_OFF 6 _BOREN_OFF & _IVP_OFF & _CPD_OFF & _WRT_OFF &
_CP_OFF
ORG 0000H
MOVLW 0X12
RLF 0X00,W
END
```





fde¥nory File Registers •• Format HeX -

```
3

6   #include "pl6F877A.inc"

10      CONFIG  FOSC XT & WDTE OFF & PWRTE OFF & BOREN OFF & LVP OFF & CPD OFF & WRT OFF & CP OFF
```



EXPERIMENT-10

AIM: Write a program in assembly language to illustrate:

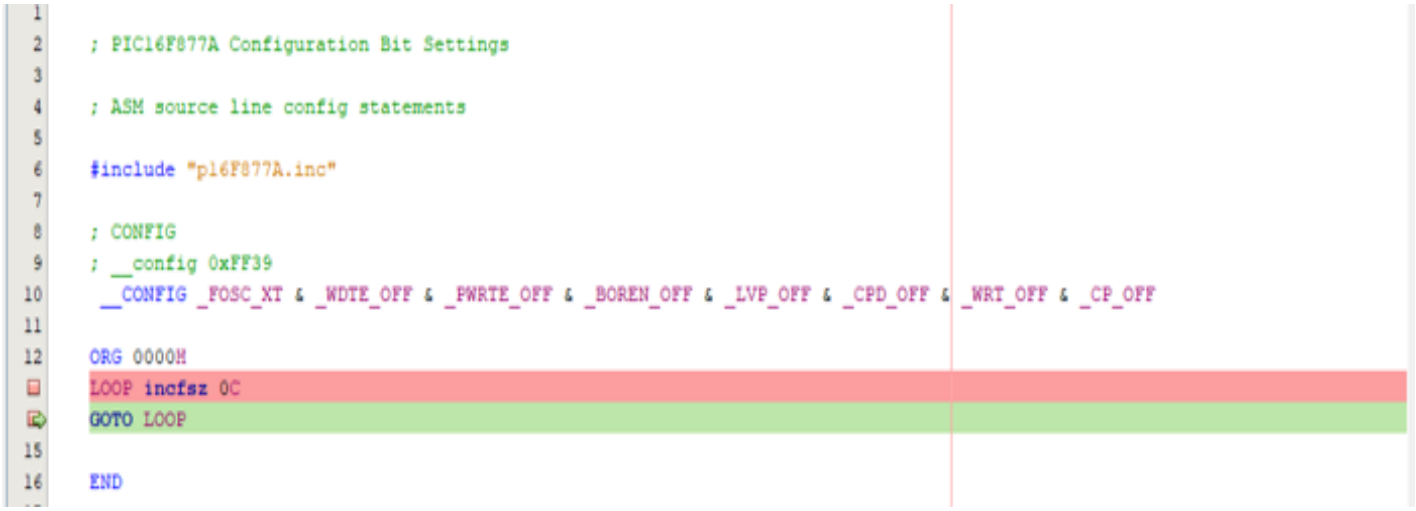
a) Loop

b) Loop inside Loop

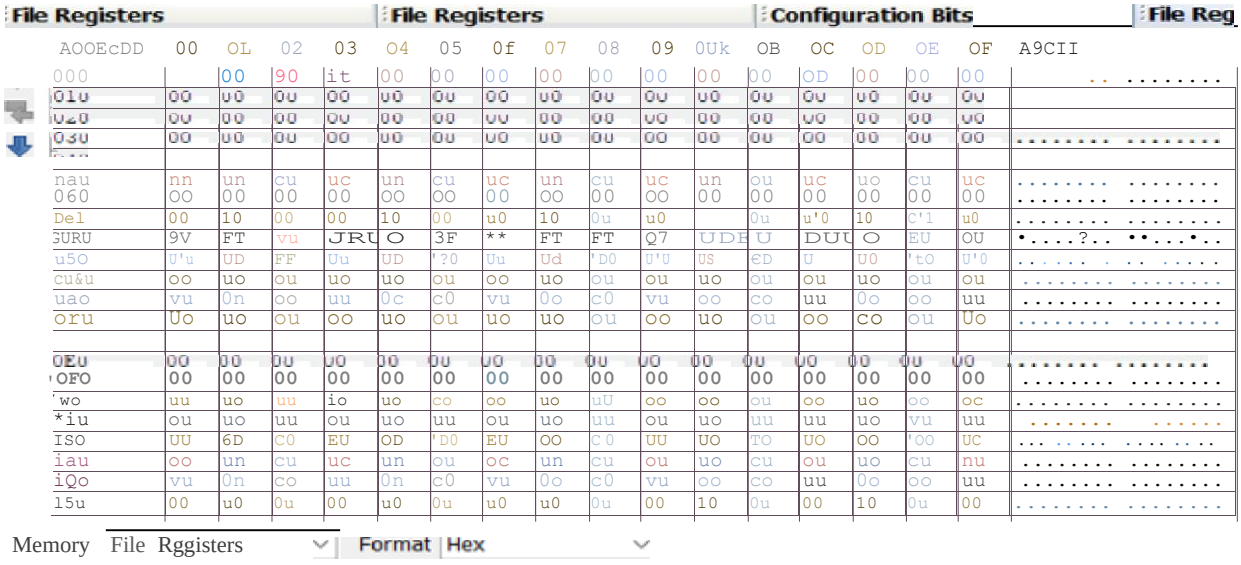
CODE:

```
ORG 0000H;

LOOP incsfz 0c;

GOTO LOOP;

END
```

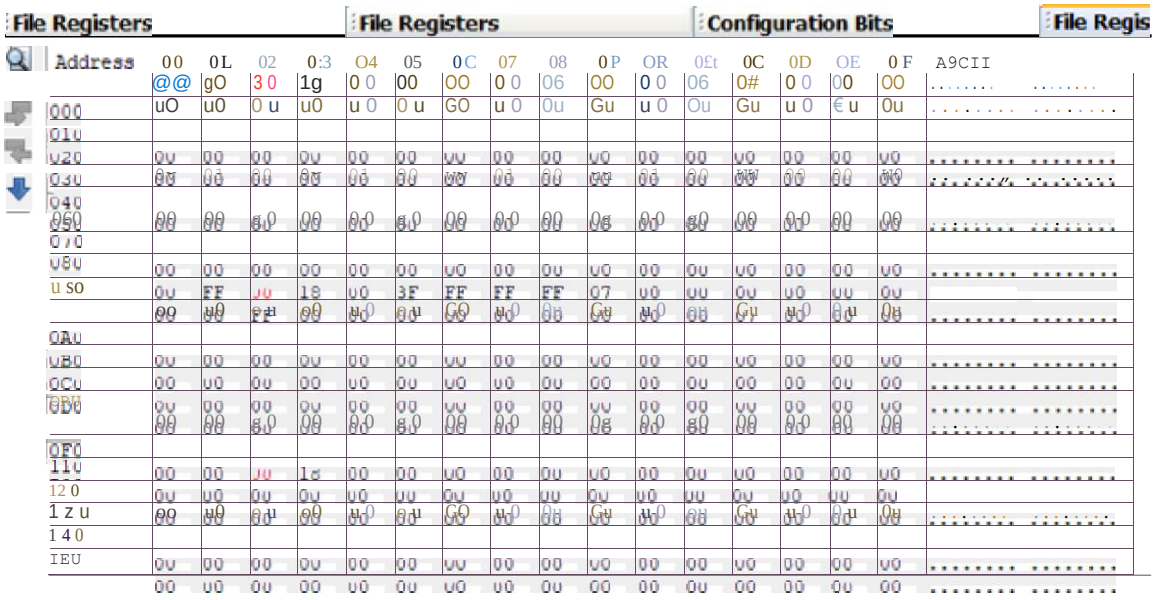
OUTPUT:

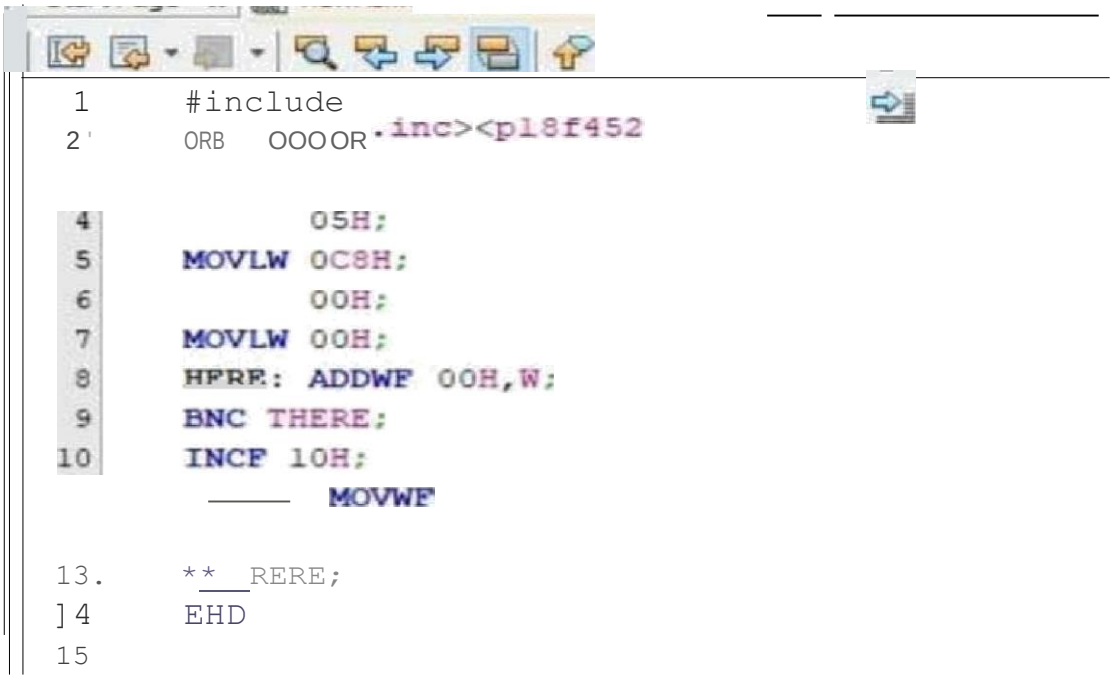


```
; PIC16F877A Configuration Bit Settings

; ASM source line config statements

#include "p16F877A.inc"

; CONFIG
; __config 0xFF39
 __CONFIG _FOSC_XT & _WDTE_OFF & _PWRTE_OFF & _BOREN_OFF & _LVP_OFF & _CPD_OFF & _WRT_OFF & _CP_OFF

ORG 0000H
LOOP incfsz 0C
GOTO LOOP

END
```

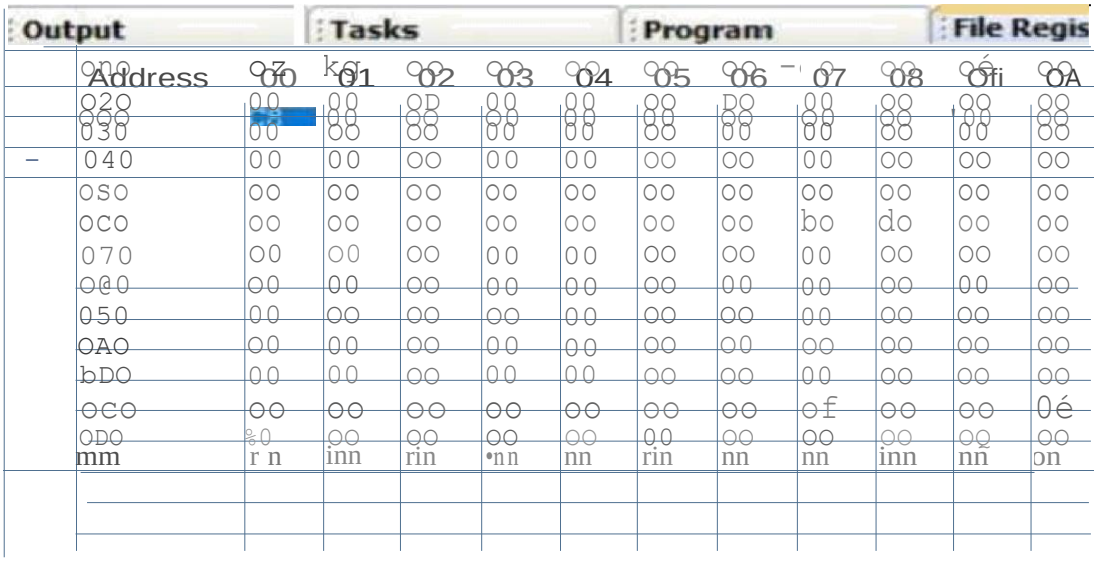



```
2   ; PIC16F877A Configuration Bit Settings


5
6   #include "p16F877A.inc"

8   ; CONFIG
9   ; __config 0xFF39
10   __CONFIG _FOSC_XT & _WDTE_OFF & _PWRTE_OFF & _BOREN_OFF & _LVP_OFF & _CPD_OFF & _WRT_OFF & _CP_OFF

12  ORG 000 Od

15
16  END
```





```
1       #include
2       ORB   OOOOR·inc><p18f452

4             05H;
5       MOVLW 0C8H;
6             00H;
7       MOVLW 00H;
8       HERE: ADDWF 00H,W;
9       BNC THERE;
10      INCF 10H;
          ———   MOVWF

13.     ** RERE;
]4      EHD
15
```



Output | Tasks | Program | File Regis

Address	00	01	02	03	04	05	06	07	08	Ofi	OA
020	00	00	OD	00	00	00	DO	00	00	00	00
030	00	00	00	00	00	00	00	00	00	00	00
040	00	00	00	00	00	00	00	00	00	00	00
050	00	00	00	00	00	00	00	00	00	00	00
060	00	00	00	00	00	00	00	bo	do	00	00
070	00	00	00	00	00	00	00	00	00	00	00
080	00	00	00	00	00	00	00	00	00	00	00
090	00	00	00	00	00	00	00	00	00	00	00
0A0	00	00	00	00	00	00	00	00	00	00	00
0B0	00	00	00	00	00	00	00	00	00	00	00
0C0	00	00	00	00	00	00	00	of	00	00	0é
0D0	%0	00	00	00	00	00	00	00	00	00	00
mm	r n	inn	rin	•nn	nn	rin	nn	nn	inn	nn	on

EXPERIMENT-11

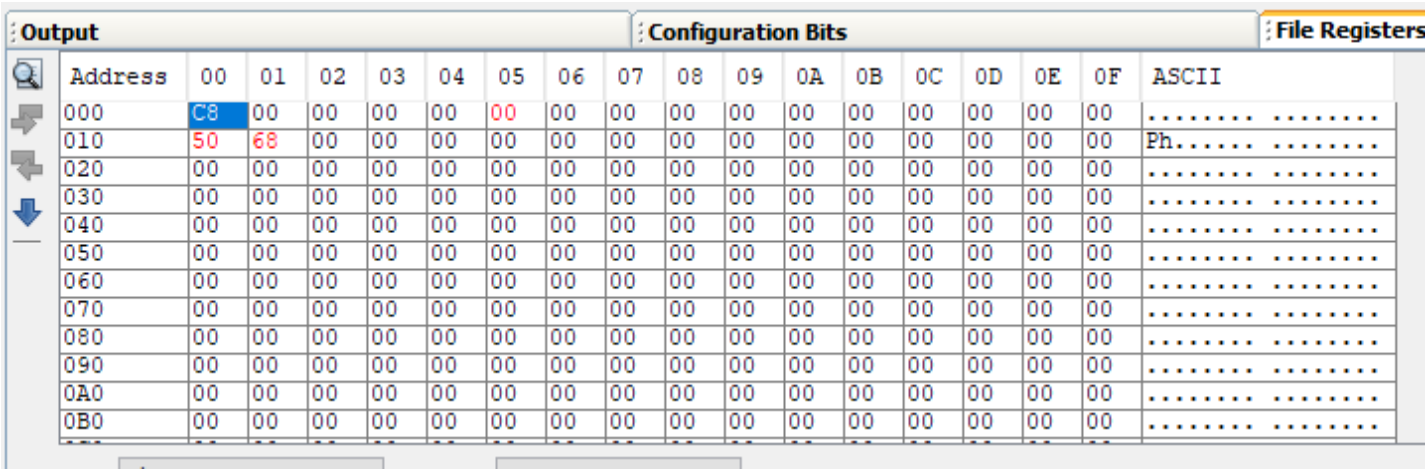
AIM: Write a program in assembly language to illustrate the following:

a) bit oriented
b) byte oriented

CODE:

```
; PIC16F877A Configuration Bit Settings


; ASM source line config statements


#include "p16F877A.inc"


; CONFIG

;__config 0xFFFF

 __CONFIG _FOSC_EXTRC & _WDTE_ON & _PWRTE_OFF & _BOREN_ON & _LVP_ON & _CPD_OFF & _WRT_OFF & _CP_OFF


; Bit-oriented operations

  ; Load value 0x55 into register W

  MOVLW 0x55

  ; AND the value in W with the mask 0xF0, result stored in W

  ANDLW 0xF0

  ; Store the result in register 0x0A

  MOVWF 0x0A
```

```
; OR the value in W with the mask 0x0F, result stored in W
IORLW 0x0F
; Store the result in register 0x0B
MOVWF 0x0B


; Byte-oriented operations
; Load value 0x0A into register W
MOVLW 0x0A
; Store the value in register W into register 0x0C
MOVWF 0x0C
; Load value 0x05 into register W
MOVLW 0x05
; Add the value in W to the value in register 0x0C, result stored in W
ADDWF 0x0C, W
; Store the result in register 0x0D
MOVWF 0x0D
```



Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
000	C8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
010	50	68	00	00	00	00	00	00	00	00	00	00	00	00	00	00	Ph..... .....
020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....

END