- PSDL -

ROLL NO : 23330

PICT, PUNE

- ASSIGNMENT-1 -

NAME : NANDINI PATIL
~~D.O.P~~ CLASS : SE II
~~D.O.S~~ ~~DIV~~ BATCH : FII.

a) Explain the status register of PIC 18 in detail w.r.t the addition operation. Give example in which status register gets value (HEX)

i) 10 ii) 8 iii) 14 iv) 03 v) 11

⇒ Status register holds the status flags for instructions execution. It is stored at memory location (0XFD8).

-	-	-	N	OV	Z	DC	C
7	6	5	4	3	2	1	0

N → Negative Bit.
1 = arithmetic result is negative
0 = arithmetic result is positive.

OV → overflow bit.
1 = Overflow occured for signed arithmetic.
0 = No overflow.

Z = zero flag.
1 = result of arithmetic or logical operation is 0.

PICT, PUNE

ROLL NO : 23330

- ASSIGNMENT - 1 -

0 = Non - zero result.

DC = Digit carry

For ADDWF, ADDLW. instructions.

1 = carry from 4th low order bit of result has occured

0 = NO carry from 4th low order bit of result has occured.

C = carry bit

For ADDLW, ADDWF instructions.

1 = A carry out of MSB of result has occured.

0 = No carry out at MSB of result.

(i) $(10)_{16} = (00010000)_2$

Here, only negative bit is set. This result in the status register may occur only When there is NO carry or DC. There is non-zero arithmetic answer.

The answer of addition must be negative.

eg. $\begin{array}{r} 81\,H \\ 11\,H \\ \hline 92\,H. \end{array}$ status : 0001 0000

PICT, PUNE

ROLL NO : 23330

ASSIGNMENT - 1

ii) $(18)_{16} = (0001\ 1000)_2$

Here, the negative and overflow flags are set. The answer must have no carry or DC. It must be non-zero and negative.

Eg: $\begin{array}{r} 62\ H \\ +\ 33\ H \\ \hline 95\ H \end{array}$ STATUS : 0001 1000

iii) $(05)_{16} = (0000\ 0101)_2$

Here, the carry and zero flags are set. The answer must have no DC or overflow. The sum must be zero.

Eg: $\begin{array}{r} F0\ H \\ 10\ H \\ \hline 100\ H \end{array}$ STATUS : 0000 0101.

iv) $(03)_{16} = (0000\ 0011)_2$

Here, the DC and carry flag is set. The answer must be positive, non-zero and the sum must have no overflow.

Eg: $\begin{array}{r} FF\ H. \\ 03\ H \\ \hline 102\ H. \end{array}$

v) $(11)_{16} = (0001\ 0001)_2$

Here, carry and negative flags are set. The answer is negative. There is no DC

PICT, PUNE

ASSIGNMENT-1 ROLL NO: 28330

and overflow. The answer is non-zero

$$\begin{array}{r} F3\ H \\ +\ 92\ H \\ \hline 185\ H \end{array}$$

STATUS : 0001 0001

b] Explain the registers PORTx, TRISx and LATx:

⇒ Each port in PIC18 is associated with 3 special function registers.

PORTx :

This register is used to read/write data from/to port pins. Writing it to PORTx will make the corresponding PORTx pins high and writing '0's to PORTx will make the corresponding PORTx pins low. There are 5 PORTx Registers.

PORTA	F80H	memory locations
PORTB	F81H	
PORTC	F82H	
PORTD	F83H	
PORTE	F84H	

TRISx :

These registers are used to configure respective PORTx as output/input. They are known as Tristate or data direction registers. Writing 1's to TRISx will

PICT, PUNE

ROLL NO : 23330

ASSIGNMENT - 1

make corresponding PORTx pins as ~~output~~ input. Similarly, writing 0's to TRISx will make corresponding PORTx pins as output.

TRIS A	F92 H.
TRIS B	F93 H
TRIS C	F94 H
TRIS D	F95 H
TRIS E	F96 H

} memory locations.

LATx :-

These are port latch registers when data is written to PORTx registers, it is first stored in LATx registers then copied to PORTx. If TRISA register is not 0, data is not copied to PORTx but remains in LATx, when data is read from PORT, it is first copied to LATx and then WREG. If TRISx is not 0xFF, data is not copied from PORTx but from LATx.

LAT A	F89 H
LAT B	F8A H
LAT C	F8B H
LAT D	F8C H
LAT E	F8D H.

} memory locations.

c] Write a note on embedded C programming.

PICT, PUNE

ROLL NO: 28930

ASSIGNMENT-I

Q] Write a note on Embedded C Programming.

⇒ Embedded C is an extension of standard C programming language with additional features like addressing i/o, multiple memory addressing, fixed point arithmetic, etc.

⇒ Both C and embedded C are ISO standards that have almost same syntax, datatypes, functions etc.

⇒ C programming language is generally used for developing desktop applications, whereas embedded C is used to develop microcontroller based applications.

Structure of Embedded C program:

```
# include
# define                    /* Processor
Global variables.                directives */
   function declaration
main() {
      local variables      # or infinite loops.
      function calls
} function defination
```

PICT, PUNE

ROLL NO: 23880

- ASSIGNMENT - 1 -

INPUT : Input code is attached.

OUTPUT : output of code is attached.

CONCLUSION: Thus we have studied embedded C program for addition of two numbers.

INPUT:

main.c

Source | History

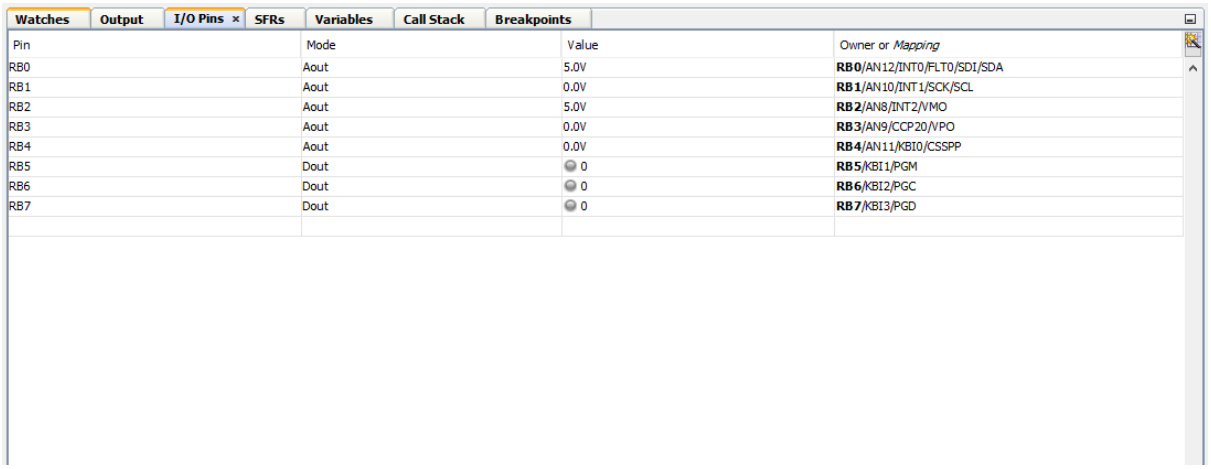
```
1  /*
2   * File:   main.c
3   * Author: NANDINI
4   *
5   * Created on 3 March, 2021, 10:33 AM
6   */
7
8
9  #include <xc.h>
10 #include <stdlib.h>
11 #include <stdio.h>
12 #include <PIC18F4550.h>
13
14 void main(void) {
15     int a = 2;
16     int b = 3;
17     int sum = a+b;
18     PORTB = sum;
19     TRISB = 0x00;
20     return;
21 }
22
```

OUTPUT:

Watches | Output | I/O Pins | SFRs | Variables | Call Stack | Breakpoints

Address	Name	Hex	Decimal	Binary	Char
FDF	INDF2	0x00	0	00000000	'.'
FF2	INTCON	0x03	3	00000011	'.'
FF1	INTCON2	0xF5	245	11110101	'õ'
FF0	INTCON3	0xC2	194	11000010	'Â'
F9F	IPR1	0xFF	255	11111111	'ÿ'
FA2	IPR2	0xFF	255	11111111	'ÿ'
F89	LATA	0x00	0	00000000	'.'
F8A	LATB	0x05	5	00000101	'.'
F8B	LATC	0x00	0	00000000	'.'
F8C	LATD	0x00	0	00000000	'.'
F8D	LATE	0x00	0	00000000	'.'
FD3	OSCCON	0x48	72	01001000	'H'
F9B	OSCTUNE	0x00	0	00000000	'.'
FF9	PCL	0xE6	230	11100110	'æ'
FF9	PCLAT	0x0000E6	230	00000000 00000000 11100110	'..æ'
FFA	PCLATH	0x00	0	00000000	'.'
FFB	PCLATU	0x00	0	00000000	'.'
F9D	PIE1	0x00	0	00000000	'.'
FA0	PIE2	0x00	0	00000000	'.'
F9E	PIR1	0x00	0	00000000	'.'
FA1	PIR2	0x00	0	00000000	'.'
FEB	PLUSW0	0x00	0	00000000	'.'
FE3	PLUSW1	0x00	0	00000000	'.'
FDB	PLUSW2	0x00	0	00000000	'.'
F80	PORTA	0x00	0	00000000	'.'
F81	PORTB	0x00	0	00000000	'.'

Memory SFRs Format Individual



Watches | Output | I/O Pins × | SFRs | Variables | Call Stack | Breakpoints

Pin	Mode	Value	Owner or *Mapping*
RB0	Aout	5.0V	**RB0**/AN12/INT0/FLT0/SDI/SDA
RB1	Aout	0.0V	**RB1**/AN10/INT1/SCK/SCL
RB2	Aout	5.0V	**RB2**/AN8/INT2/VMO
RB3	Aout	0.0V	**RB3**/AN9/CCP20/VPO
RB4	Aout	0.0V	**RB4**/AN11/KBI0/CSSPP
RB5	Dout	0	**RB5**/KBI1/PGM
RB6	Dout	0	**RB6**/KBI2/PGC
RB7	Dout	0	**RB7**/KBI3/PGD

PICT, PUNE

ROLL NO: 23380

- ASSIGNMENT -2 -

NAME: NANDINI PATIL

BATCH: F11.

AIM: Write an embedded c program to add array of 'n' numbers.

SOFTWARE: MPLAB XIDE V5.45

THEORY: An array is defined as the collection of similar type of data items stored at contiguous memory locations. Arrays are the derived data types in C programming language which can store the primitive type of data such as int, char, double, float etc.

→ It is necessary to understand basic IO operations of PIC18f4550 before dealing with its complexities. This is a way to take simple output from a PIC microcontroller.

→ PIC18f4550 has a total of 33 I/O (input/output) pins which are distributed among 5 ports, the following table shows the names and number of IO pins of these 5 ports.

PICT, PUNE

ROLL NO : 28330

- ASSIGNMENT - 2 -

Port Name	No. of pins	Pins.
PORT A	6	RA0 - RA5
PORT B	8	RB0 - RB7
PORT C	8	RC0 - RC5, RC6 - RC7
PORT D	8	RD0 - RD7
PORT E	3	RE0 - RE2.

The 33 I/O pins are also multiplexed with one or more alternative functions of controller's various peripherals. Each port of PIC microcontroller corresponds to three 8-bit register which should be configured to use the port for general I/O purpose. The registers are :→

1) TRISx :→ This is a data direction which sets the direction of each port pin as input or output.

2) PORTx :→ This register istores the input levels of pins (High / low). When a pin configured as input, signal from external source is read from RORTx register.

PICT, PUNE

ROLL NO: 23330

- ASSIGNMENT - 2 -

3) LAT x :→ This is output latch register. The data which has to be sent to external hardware as output is stored in LAT x register.

INPUT: Input code is attached.

OUTPUT: Output of the code is attached.

CONCLUSION: Thus we have studied embedded C program to add array of 'n' numbers.

—x—

INPUT:

main.c

Source | History

```
/*
 * File:   main.c
 * Author: NANDINI
 *
 * Created on 24 February, 2021, 11:56 AM
 */


#include <xc.h>
#include <stdlib.h>
#include <pic18f4550.h>

void main(void) {
    int sum=0;
    int n=10;
    int arr[10] = {1,2,3,4,5,6,7,8,9,0};

    for(int i=0 ; i<n ; i++)
    {
        sum+=arr[i];
    }

    PORTB = sum;
    TRISB = 0x00;

    return;
}
```

OUTPUT:

Watches | Output | I/O Pins | SFRs | Start Page | Variables | Call Stack | Breakpoints

Address	Name	Hex	Decimal	Binary	Char
FD2	HLVDCON	0x05	5	00000101	'.'
FEF	INDF0	0x00	0	00000000	'.'
FE7	INDF1	0x00	0	00000000	'.'
FDF	INDF2	0x00	0	00000000	'.'
FF2	INTCON	0x03	3	00000011	'.'
FF1	INTCON2	0xF5	245	11110101	'õ'
FF0	INTCON3	0xC2	194	11000010	'Â'
F9F	IPR1	0xFF	255	11111111	'ÿ'
FA2	IPR2	0xFF	255	11111111	'ÿ'
F89	LATA	0x00	0	00000000	'.'
F8A	LATB	0x2D	45	00101101	'-'
F8B	LATC	0x00	0	00000000	'.'
F8C	LATD	0x00	0	00000000	'.'
F8D	LATE	0x00	0	00000000	'.'
FD3	OSCCON	0x48	72	01001000	'H'
F9B	OSCTUNE	0x00	0	00000000	'.'
FF9	PCL	0x64	100	01100100	'd'
FF9	PCLAT	0x0...	100	00000000 00000...	'..d'
FFA	PCLATH	0x00	0	00000000	'.'
FFB	PCLATU	0x00	0	00000000	'.'
F9D	PIE1	0x00	0	00000000	'.'
FA0	PIE2	0x00	0	00000000	'.'
F9E	PIR1	0x00	0	00000000	'.'
FA1	PIR2	0x00	0	00000000	'.'
FEB	PLUSW0	0x00	0	00000000	'.'
FE3	PLUSW1	0x00	0	00000000	'.'

Memory SFRs Format Individual

PICT, PUNE

ROLL NO: 23330

-ASSIGNMENT-2-

NAME: NANDINI PATIL

D.O.S:

D.O.P:

Q.1.) Write short note on working register (WREG) of PIC18.

=> 1] The working register or WREG is a temporary storage for information at the CPU.

2] It has a size of 8 bits.

3] It is also called as "Program Memory Space."

4] It can be used for arithmetic and logical instructions like move or Add.

Eg. MOVLW 'N' means,
move literal value N in the WREG.

Eg. A program to add two numbers in WREG.

```
MOVLW  11H
ADDLW  22H.
```

83 H is stored in the WREG.

5] It is the same as an accumulator in other microprocessors.

Q.2] memory of PIC18F458 (all types) and memory banking.

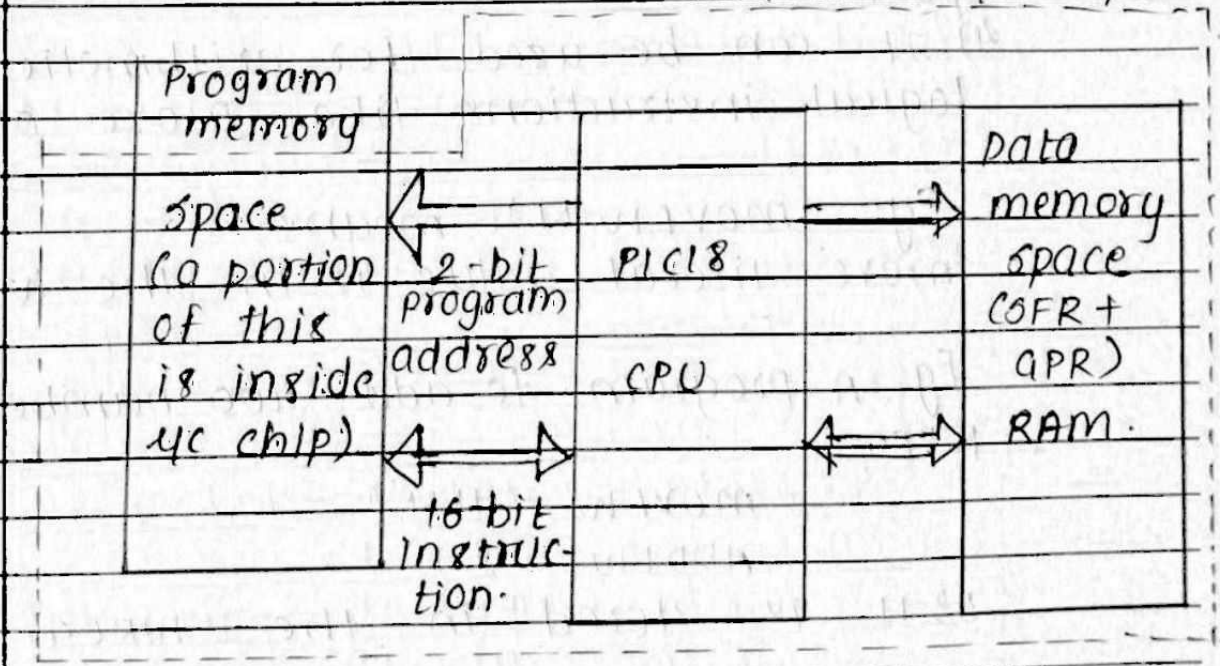
PICT, PUNE

ROLL NO : 23330

ASSIGNMENT - 2

⇒ 1] memory consists of a sequence of directly addressable locations. A location is reffered to as an information unit. A memory location can be used to store data, instruction and status of peripheral devices. A memory location has two components : an address and its contents.

2] Data memory and Program memory are separated. This means it is possible to simultaenously access data and instruction.



3 memory features:

⇒ 4096 bytes of data memory.
⇒ General purpose registers are used to hold dynamic data.

PICT, PUNE

ROLL NO: 23330

ASSIGNMENT-2

⇒ Special function registers are used to control operations of peripheral function.

⇒ only one bank is active at a time and is specified by the BSR registers.

⇒ Bank switching is overhead and can be error prone.

⇒ PIC 18 implements access bank to reduce problems caused by bank switching.

⇒ Access bank consists of lowest 96 bytes and highest 160 bytes of data memory space.

BSR <3:0>		
0000 Bank 0	Access RAM	000h } 0-95 05Fh
	GPRs	00FFh
0001 Bank 1	GPRs	100h 1FFh
0010 Bank 2	GPRs	200h 2FFh
		300h
Bank 3 to 13	GPRs	DFFh
1110 Bank 14	GPRs	E00h EFFh
1111 Bank 15	unused	F00h F5Fh
	SFRs	F60h FFFh

Access bank.

000h	
Access RAM low	5Fh
Access RAM high	60h
SFRs	FFh
	FFFh

⇒ Every bank is 256 bytes.

⇒ PIC 18 has 2 MB program space.

1 MB = 16 Banks.

PICT, PUNE

ASSIGNMENT-3 ROLL NO: 23330

Hence, PIC18 has $\frac{16}{32}$ banks.

⇒ 0 to 156 of first bank : Access RAM (low).

higher 196 of last bank : Access RAM (high)

⇒ Unused space is for some SFRs that can be added in later versions.

⇒ PIC is 21 bit long and enables user program to access upto 2 MB of program memory.

⇒ PIC 18 has 31 entry return address stack to return address for subroutine call.

⇒ After POWER ON, PIC18 starts executing instruction for from-0 address 0.

⇒ Location at address 0x08 is reserved for high priority ISR and 0x18 for low priority ISR.

⇒ upto 128 KB of program memory inside mcu chip (at present time).

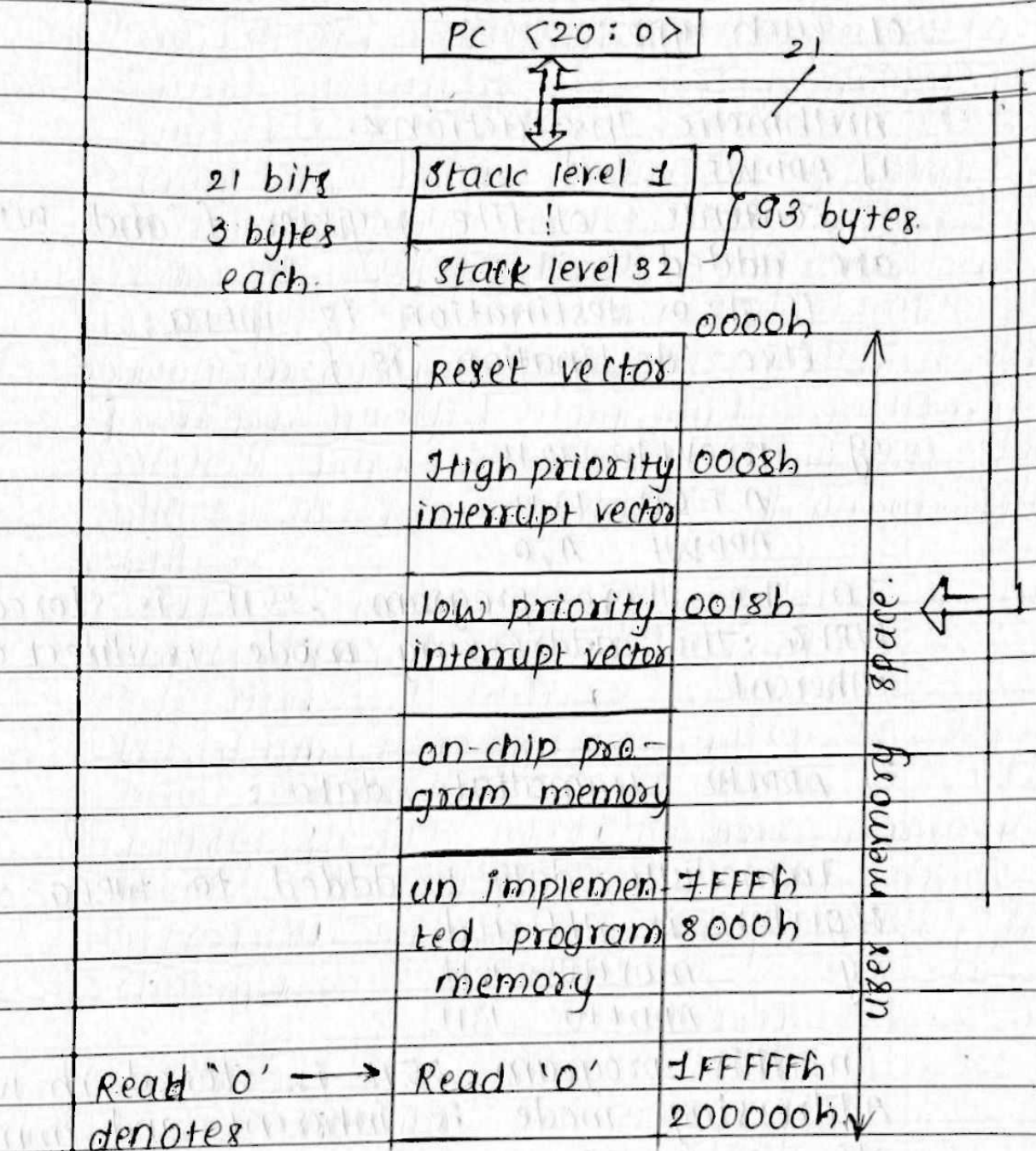
⇒ Part of program memory outside mcu chip.

DIAGRAM ON NEXT PAGE.

PICT, PUNE

ROLL NO: 23330

ASSIGNMENT-3



program memory Orientation.

37 Instructions for arithmetic, logical, branch and bitwise operations. Give 5 examples

PICT, PUNE

ROLL NO : 23330

ASSIGNMENT -3

of each type.

=> Arithmetic Instructions.

1) ADDWF f,d

contents of file register 'f' and WREG are added.

If d=0 destination is WREG,
else destination is f.

eg. MOVLW 12H
AEQU 13H.
ADDWF A,0

In the above program, 25H is stored in WREG. The addressing mode is direct and inherent.

2) ADDLW Immediate data :

Immediate data is ddded to WREG. Status register is affected.

eg. MOVLW 12H.
ADDLW 13H.

In this program, 25H is stored in WREG. Addressing mode is inherent and immediate.

3) SUBFW f,d

contents of WREG are subtracted from the file register 'f' (f - WREG)

PICT, PUNE

ROLL NO: 23330

ASSIGNMENT-3

if d=0, destination is WREG.
else it is f.
eg.

```
MOVLW   01H
AEQU    03H
SUBFW   A,0
```

In the above program, 02H is stored in WREG. Addressing mode is register and inherent.

4] INC f,d : Increment file register F

eg:
```
A EQU 8H
INC A.
```

9H is stored in A.
Addressing mode is register.

5. DECF f,d : Decrement file register f

eg:
```
A EQU 8H
DECF A.
```

7H is stored in A.
Addressing mode is register.

logical instructions:

1. ANDLW Immediate data.
Immediate data and data inside the WREG are 'Anded.,

PICT, PUNE

ASSIGNMENT-3 ROLL NO = 28330

E.g. MOVLW B' 10001000'.
ANDLW B' 01011000'

$(0000\ 1000)_2$ is stored in the WREG. Addressing mode is immediate and inherent.

2] ANDWF f,d.
The data in file register 'f' is 'anded' with data in the WREG.

Eg: A EQU B' 00100101'
MOVLW B' 00000011'
ANDWF A,0.

$(0000\ 0001)_2$ is stored in WREG. Addressing mode is register and inherent.

3] IORLW Immediate data.
Immediate data and data inside the WREG is 'OR ed'. Addressing mode is immediate and inherent.

eg. movlw B' 00100101'
IORLW B' 11010001'

$(1111\ 0101)_2$ is stored in the WREG A.

4] IORWF f,d.
Data in file register 'f' is 'OR-ed' with data in WREG.

PICT, PUNE

ASSIGNMENT-3

ROLL NO : 23330

E.g. A EQU B' 00000001'
MOVLW B' 10000000'
IORWF A,0

(1000 0001)$_2$ is stored in the WREG.
Addressing mod is inherent and register.

5] XORLW Immediate data :
Immediate data is 'Xclusively OR-ed' with data in the WREG.

eg. MOVLW B' 1111 1111'
XORLW B' 1111 0000'

(0000 1111)$_2$ is stored in WREG.
Addressing mode is immediate and inherent.

Branch
1. GOTO xyz :
control goes to xyz
Eg : C EQU 5H
MOVLW 00H.
loop ADDLW 06H.
DECFSZ C
GOTO loop
} Multiply 6 and 5 (Hexadecimal)

2. BNZ address :
Branch if not zero.

3. BZ address :
Branch if zero.

PICT, PUNE

ROLL NO: 28330

ASSIGNMENT-3

4. BNC address : Branch if not Zero carry.

5. BC address : Branch if carry

Bitwise

1. BCF f,b :
clear bit 'b'
Eg: A EQU B' 0001 1111'
BCF A,4
value in A becomes $(0001\ 0111)_2$

2. BSF f,b : set bit 'b'
eg. A EQU B' 0000 0000'
BSF A,3
value in A becomes $(0000\ 0100)_2$

3. BTG f,b ; Toggle bit 'b'.
e.g. A EQU B' 0010 0010'
BTG A,2
BTG A,1
value in A becomes $(0010\ 0001)_2$

4. RLCF f,d : Rotate left file register through CY.

MSB → CY 0000 1100 1
CY → LSB 0001 1001 0

5. RRCF f,d : Rotate right file register through CY.

LSB → CY 0000 1100 1
CY → MSB 1000 0110 0

— x —

PICT, PUNE

ROLL NO: 28330

- ASSIGNMENT - 3 -

~~INPUT : Input~~

CONCLUSION: Thus we have studied embedded C program to transfer elements from one location to another for internal to internal memory transfer and internal to external memory transfer.

PICT, PUNE

ROLL NO : 28330

- ASSIGNMENT - 4 -

NAME : NANDINI PATIL
DIVISION : SE - 11
BATCH : F11

Q1) Write a note on MPLAP IDE.

=> 1) MPLAB is a proprietary freeware integrated development environment for the development of embedded applications on PIC and dsPIC microcontrollers and is developed by microchip Technology.

2) MPLAB IS the latest edition of MPLAB and is developed on the NetBeans platform.

3) MPLAB and MPLABX support project management, code editing, debugging and programming of microchip 8-bit PIC24 and dsPIC and AVR microcontrollers, as well as 32 bit SAM (ARM) and PIC32 (MIPS) microcontrollers.

4) MPLABX supports automatic code generation with the MPLAB code configurator and the MPLAB Harmony configurator plugins.

Q2// Write an assembly language program

PICT, PUNE

ROLL NO- 23330

ASSIGNMENT-4

for PIC18 to multiply two 8 bit numbers using consecutive addition.

=> Multiply 0x05 and 0x0A.

```
COUNT   EQU   0x30
        MOVLW 0x05
        MOVWF COUNT
        MOVLW 0
LOOP    ADDLW 0x0A
        DECFSZ COUNT,F
        GOTO  LOOP.
```

x

Q3// Write an assembly language program for PIC 18 to divide one 8 bit number by another 8 bit number using consecutive subtractors.

=> Divide 0x0A by 0x02

```
QUOTIENT EQU 0x30
         MOVLW 0x0A
LOOP     SUBLW 0x02
         INCF QUOTIENT,F
         BN LOOP

         DECF QUOTIENT,F.
```

x

ROLL NO: 28330

PICT, PUNE

- ASSIGNMENT-4 -

4) Registers of PIC18F

→ The memory of the PIC microcontroller is devided into a series of registers each of the registers has its own address and memory locations.

2) According to the working and usage the registers in PIC are classified as:

⇒ Special function Registers (SFRs):→ used for control and status of the controller and peripheral functions.

⇒ General purpose registers (GPRs) :→ used for data storage and scratch pad operations in the user's application.

⇒ WREG :→ Working Register acts as an accumulator) → used to perform arithmetic and logic operations.

⇒ Status register :→ stores flags → indicates the status of the operation done by ALU.

Registers:→ hold memory address →

⇒ Bank select registers (BSR) : 4 bit register used in direct addressing the data memory

⇒ File selected register : 16 bit register that holds the ~~register~~ program memory address while executing programs. This

PICT, PUNE

ROLL NO: 28330

-ASSIGNMENT-4-

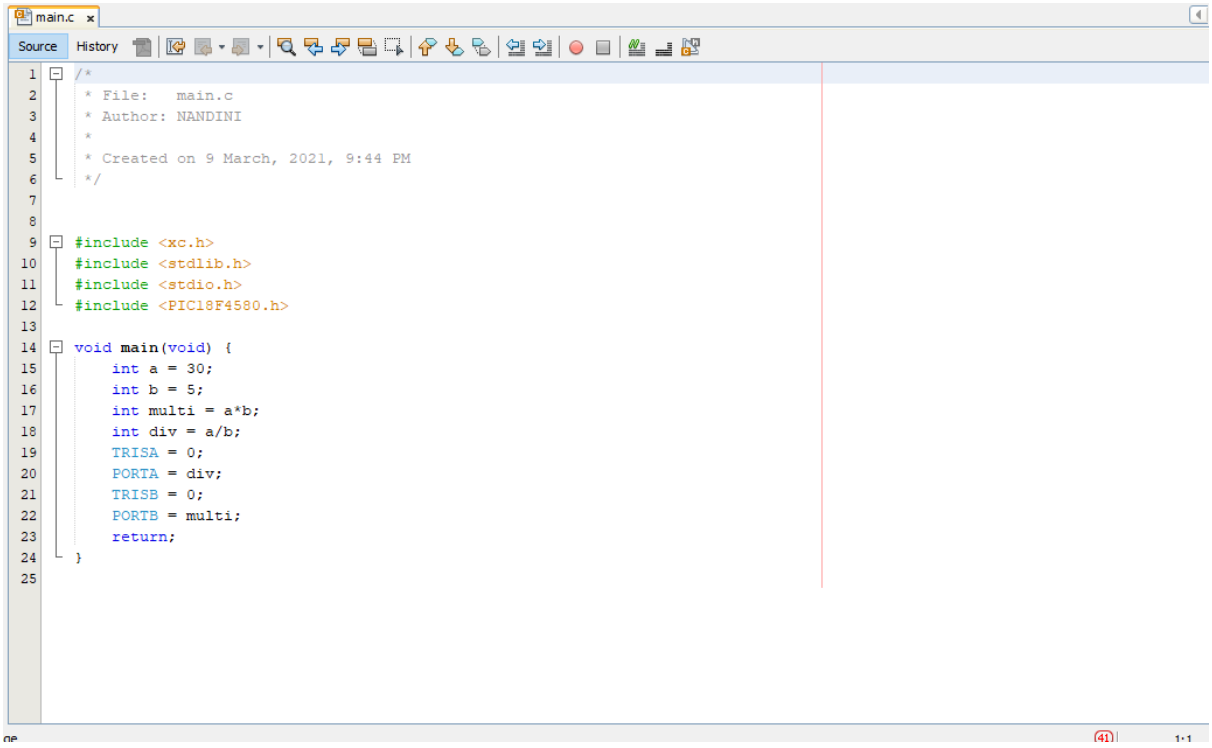
means that the PIC18 Family can access program address 000000 to 1fffffH a total of 2M bytes of code.

⇒ Stack pointer (SP): PIC18 has a 8 bit stack pointer. It is used to access the stack.

CONCLUSION: Thus we have studied embedded C program for menu driven program to multiply and divide 8-bit numbers.

—*—

INPUT:

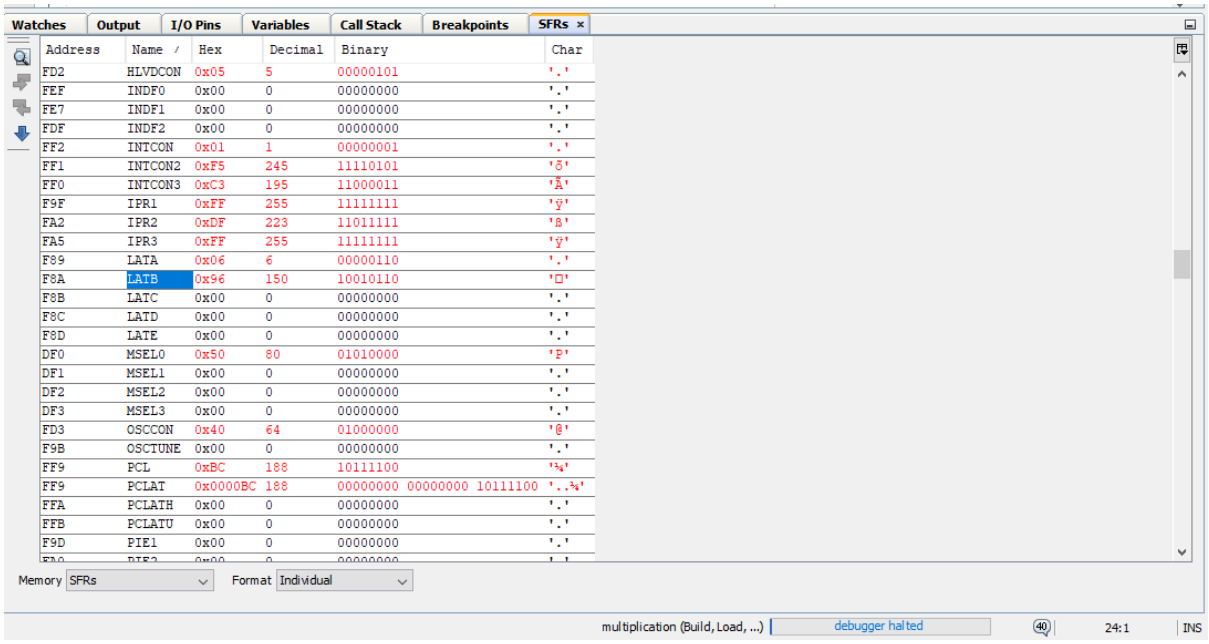


```
/*
 * File:   main.c
 * Author: NANDINI
 *
 * Created on 9 March, 2021, 9:44 PM
 */


#include <xc.h>
#include <stdlib.h>
#include <stdio.h>
#include <PIC18F4580.h>

void main(void) {
    int a = 30;
    int b = 5;
    int multi = a*b;
    int div = a/b;
    TRISA = 0;
    PORTA = div;
    TRISB = 0;
    PORTB = multi;
    return;
}
```

OUTPUT:



Watches | Output | I/O Pins | Variables | Call Stack | Breakpoints | SFRs

Address	Name	Hex	Decimal	Binary	Char
FD2	HLVDCON	0x05	5	00000101	'.'
FEF	INDF0	0x00	0	00000000	'.'
FE7	INDF1	0x00	0	00000000	'.'
FDF	INDF2	0x00	0	00000000	'.'
FF2	INTCON	0x01	1	00000001	'.'
FF1	INTCON2	0xF5	245	11110101	'õ'
FF0	INTCON3	0xC3	195	11000011	'Ã'
F9F	IPR1	0xFF	255	11111111	'ÿ'
FA2	IPR2	0xDF	223	11011111	'ß'
FA5	IPR3	0xFF	255	11111111	'ÿ'
F89	LATA	0x06	6	00000110	'.'
F8A	LATB	0x96	150	10010110	'□'
F8B	LATC	0x00	0	00000000	'.'
F8C	LATD	0x00	0	00000000	'.'
F8D	LATE	0x00	0	00000000	'.'
DF0	MSEL0	0x50	80	01010000	'P'
DF1	MSEL1	0x00	0	00000000	'.'
DF2	MSEL2	0x00	0	00000000	'.'
DF3	MSEL3	0x00	0	00000000	'.'
FD3	OSCCON	0x40	64	01000000	'@'
F9B	OSCTUNE	0x00	0	00000000	'.'
FF9	PCL	0xBC	188	10111100	'¼'
FF9	PCLAT	0x0000BC	188	00000000 00000000 10111100	'..¼'
FFA	PCLATH	0x00	0	00000000	'.'
FFB	PCLATU	0x00	0	00000000	'.'
F9D	PIE1	0x00	0	00000000	'.'

Memory: SFRs Format: Individual

multiplication (Build, Load, ...) | debugger halted | 24:1 | INS

[30]

ROLL NO: 23330.

PICT, PUNE

— ASSIGNMENT-5

AIM: Write an Embedded C program for sorting the numbers in ascending order.

SOFTWARE USED: MPLAB X IDE V5.45

THEORY: The array can be stored in ascending order by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning the algorithm maintains two subarrays in a given array.

1. The subarray which is already sorted.
2. Remaining array which is unsorted.

PROCEDURE:

Step 1: Creating a new project
⇒ Go to the File tab.
⇒ Click on new project.
⇒ Step 1: Choose project
⇒ Select: microchip embedded ⇒ Standalone project. Click 'Next'.

Step 2: Select Device:
⇒ Select: Family ⇒ Advanced 8 Bit MCU (PIC 18).
⇒ select: Device: PIC18F4550. Click Next.

Step 3: Select Tool: Simulator. Click Next.

PICT, PUNE

ROLL NO : 23330

- ASSIGNMENT - 5 -

Step 4: Select Compiler → XC8 . Click Next.

Step 5: Select project Name and Folder.

⇒ Give project name

⇒ Select project location using browse button.

⇒ Uncheck set as main project option.

⇒ Click Finish.

Step 6: Creating a new Source File.

⇒ Go to the project location in the Project Window.

⇒ Click the '+' sign to open the project space.

⇒ Right click on the Source files folder (for a c file).

⇒ New → C Source file.

Step 7: Select Build or clean and Build.

⇒ Debug the project - Check for the errors. If No errors - Build successful will appear on the OUTPUT window.

⇒ Go to Debugg window - Discrete debug operation - build for debugging - launch for debugger option - For single stepping press F8.

⇒ Go to window option - select target memory views - File register.

PROGRAM: Attached the code.

PICT, PUNE

ROLL NO : 23330

- ASSIGNMENT - 5 -

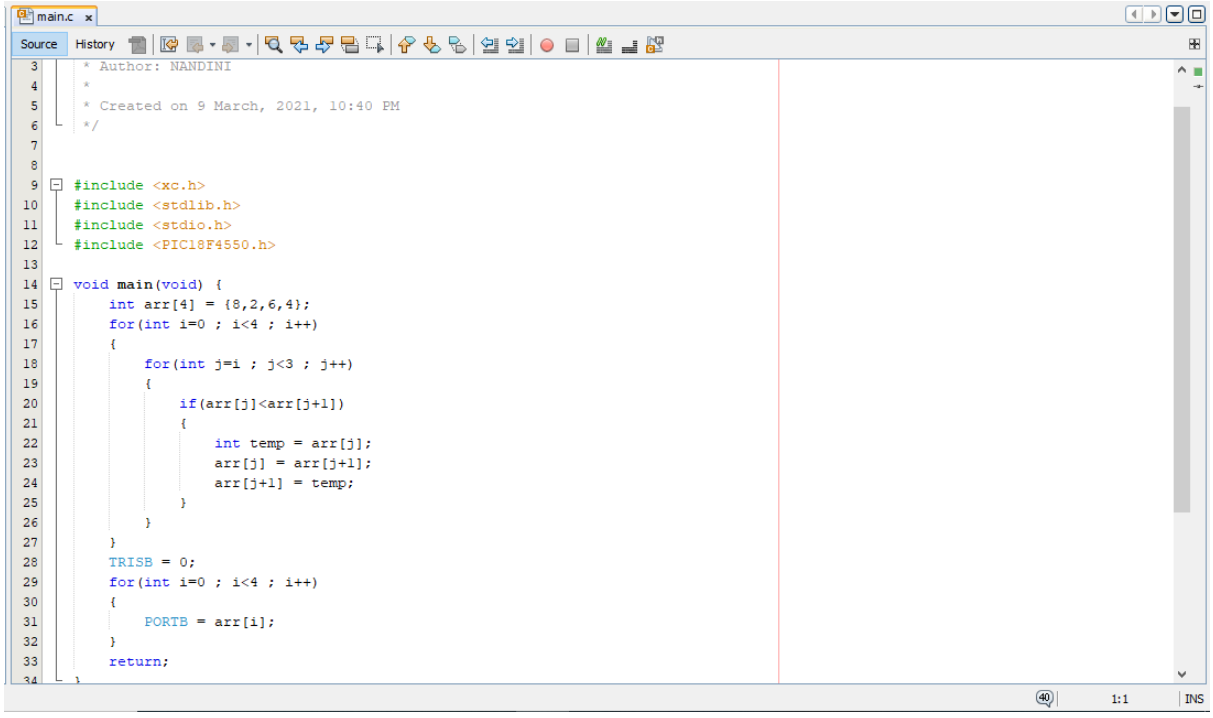
OUTPUT : It is attached as an image.

CONCLUSION : Thus we have studied embedded C program to sorting the numbers in ascending order.

- X -

INPUT:

Descending

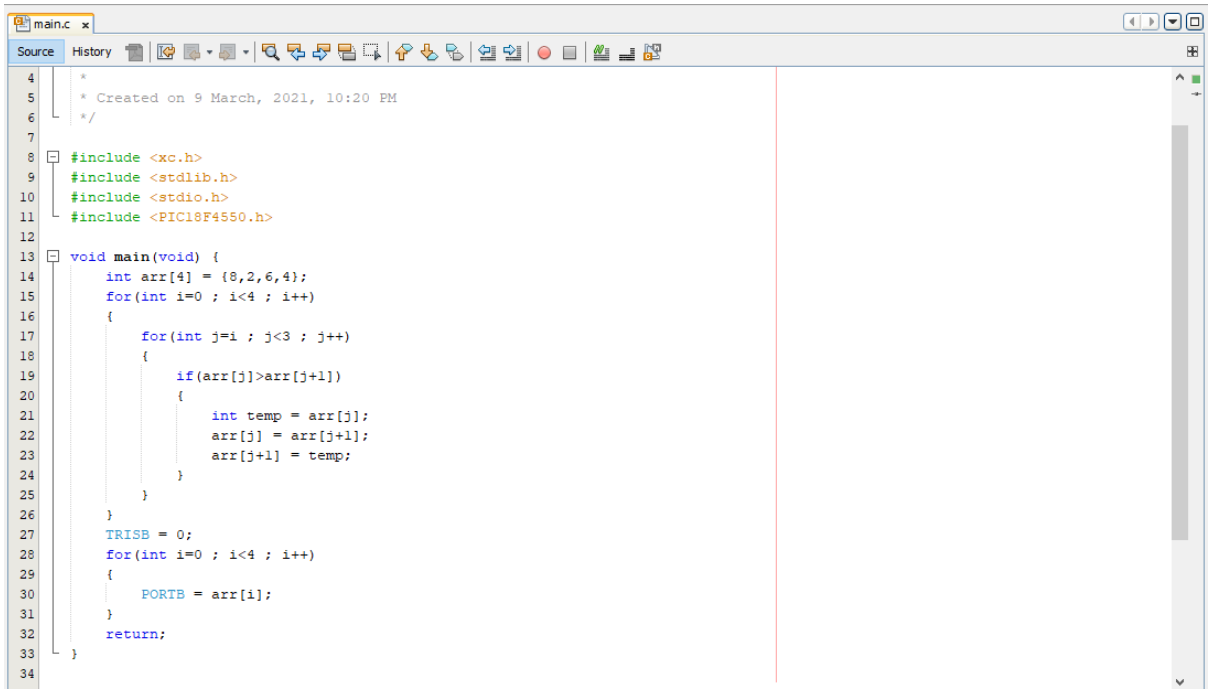


```
/*
 * Author: NANDINI
 *
 * Created on 9 March, 2021, 10:40 PM
 */


#include <xc.h>
#include <stdlib.h>
#include <stdio.h>
#include <PIC18F4550.h>

void main(void) {
    int arr[4] = {8,2,6,4};
    for(int i=0 ; i<4 ; i++)
    {
        for(int j=i ; j<3 ; j++)
        {
            if(arr[j]<arr[j+1])
            {
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
    TRISB = 0;
    for(int i=0 ; i<4 ; i++)
    {
        PORTB = arr[i];
    }
    return;
}
```

Increasing



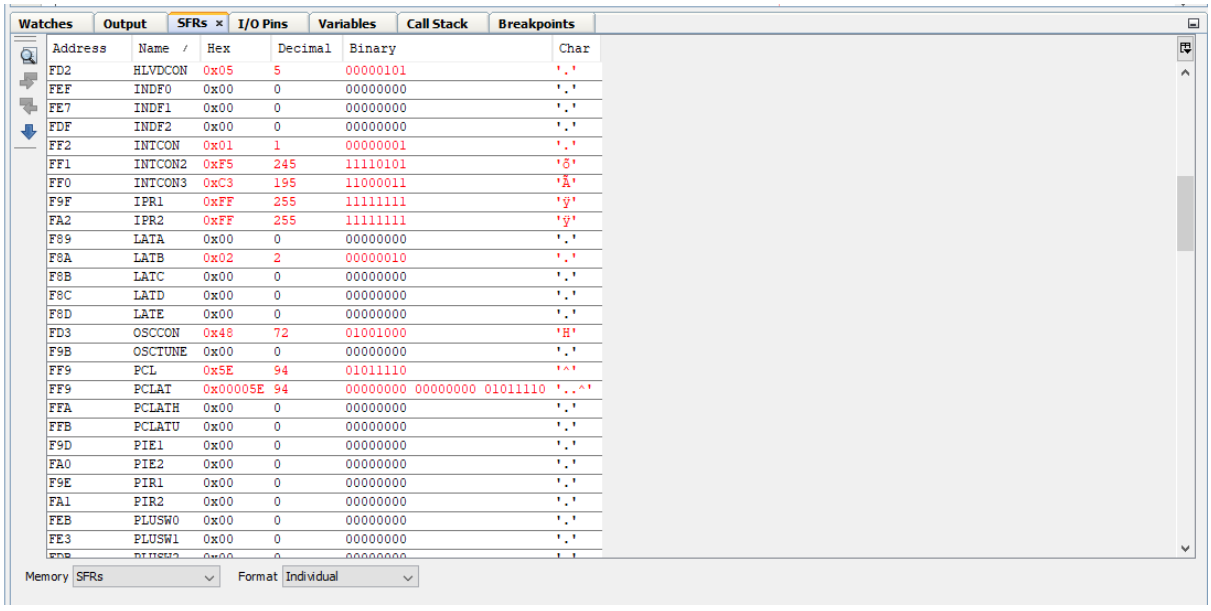
```
/*
 * Created on 9 March, 2021, 10:20 PM
 */

#include <xc.h>
#include <stdlib.h>
#include <stdio.h>
#include <PIC18F4550.h>

void main(void) {
    int arr[4] = {8,2,6,4};
    for(int i=0 ; i<4 ; i++)
    {
        for(int j=i ; j<3 ; j++)
        {
            if(arr[j]>arr[j+1])
            {
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
    TRISB = 0;
    for(int i=0 ; i<4 ; i++)
    {
        PORTB = arr[i];
    }
    return;
}
```

OUTPUT:

Decreasing

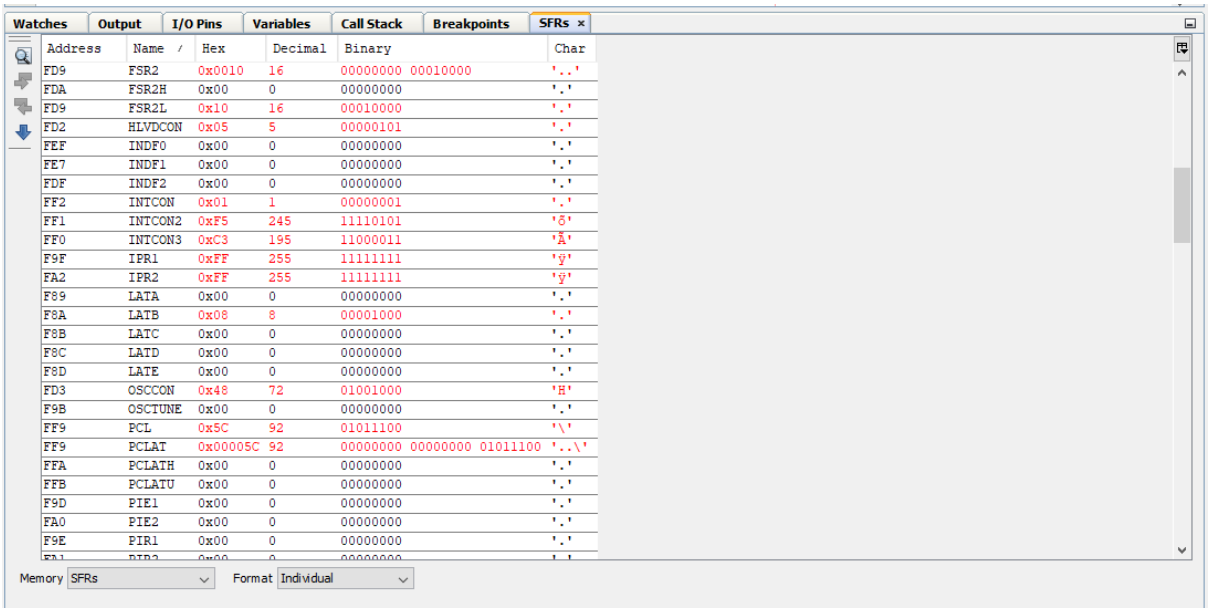


Watches | Output | SFRs × | I/O Pins | Variables | Call Stack | Breakpoints

Address	Name	Hex	Decimal	Binary	Char
FD2	HLVDCON	0x05	5	00000101	'.'
FEF	INDF0	0x00	0	00000000	'.'
FE7	INDF1	0x00	0	00000000	'.'
FDF	INDF2	0x00	0	00000000	'.'
FF2	INTCON	0x01	1	00000001	'.'
FF1	INTCON2	0xF5	245	11110101	'õ'
FF0	INTCON3	0xC3	195	11000011	'Ã'
F9F	IPR1	0xFF	255	11111111	'ÿ'
FA2	IPR2	0xFF	255	11111111	'ÿ'
F89	LATA	0x00	0	00000000	'.'
F8A	LATB	0x02	2	00000010	'.'
F8B	LATC	0x00	0	00000000	'.'
F8C	LATD	0x00	0	00000000	'.'
F8D	LATE	0x00	0	00000000	'.'
FD3	OSCCON	0x48	72	01001000	'H'
F9B	OSCTUNE	0x00	0	00000000	'.'
FF9	PCL	0x5E	94	01011110	'^'
FF9	PCLAT	0x00005E	94	00000000 00000000 01011110	'..^'
FFA	PCLATH	0x00	0	00000000	'.'
FFB	PCLATU	0x00	0	00000000	'.'
F9D	PIE1	0x00	0	00000000	'.'
FA0	PIE2	0x00	0	00000000	'.'
F9E	PIR1	0x00	0	00000000	'.'
FA1	PIR2	0x00	0	00000000	'.'
FEB	PLUSW0	0x00	0	00000000	'.'
FE3	PLUSW1	0x00	0	00000000	'.'

Memory SFRs Format Individual

Increasing:



Watches | Output | I/O Pins | Variables | Call Stack | Breakpoints | SFRs ×

Address	Name	Hex	Decimal	Binary	Char
FD9	FSR2	0x0010	16	00000000 00010000	'..'
FDA	FSR2H	0x00	0	00000000	'.'
FD9	FSR2L	0x10	16	00010000	'.'
FD2	HLVDCON	0x05	5	00000101	'.'
FEF	INDF0	0x00	0	00000000	'.'
FE7	INDF1	0x00	0	00000000	'.'
FDF	INDF2	0x00	0	00000000	'.'
FF2	INTCON	0x01	1	00000001	'.'
FF1	INTCON2	0xF5	245	11110101	'õ'
FF0	INTCON3	0xC3	195	11000011	'Ã'
F9F	IPR1	0xFF	255	11111111	'ÿ'
FA2	IPR2	0xFF	255	11111111	'ÿ'
F89	LATA	0x00	0	00000000	'.'
F8A	LATB	0x08	8	00001000	'.'
F8B	LATC	0x00	0	00000000	'.'
F8C	LATD	0x00	0	00000000	'.'
F8D	LATE	0x00	0	00000000	'.'
FD3	OSCCON	0x48	72	01001000	'H'
F9B	OSCTUNE	0x00	0	00000000	'.'
FF9	PCL	0x5C	92	01011100	'\'
FF9	PCLAT	0x00005C	92	00000000 00000000 01011100	'..\'
FFA	PCLATH	0x00	0	00000000	'.'
FFB	PCLATU	0x00	0	00000000	'.'
F9D	PIE1	0x00	0	00000000	'.'
FA0	PIE2	0x00	0	00000000	'.'
F9E	PIR1	0x00	0	00000000	'.'

Memory SFRs Format Individual

PICT, PUNE

ROLL NO : 23330

- ASSIGNMENT-7 -

NAME: NANDINI PATIL
BATCH: F11
ASSIGNMENT : 7

AIM: Write an embedded C program for Timer programing ISR based buzzer on/off.

SOFTWARE: MPLAB IDE.

THEORY:

PIC Interupt:

PIC microcontroller consists of both hardware and software interrupts. If the interrupts are generated by external hardware at certain pins of microcontroller or by inbuit devices like timer, they are called Hardware interrupts. while software interrupts are generated by a piece of code in the program.

IVT Table:

IVT table stands for Interrupt Vector Table. It holds the addresses of the interrupt service routines for various interrupts supported by the microcontroller.

⇒ Interrupt vector table for PIC18:

PICT, PUNE

ROLL NO : 23330

- ASSIGNMENT - 7 -

Interrupt	Vector Address (RAM location)
Power-on Reset	0000H.
High priority Interrupt	0008H.
low priority Interrupt	0018H.

Programming of timer using interrupt.

⇒ The timer interrupt is generated when the time register overflows from ffh to 00h in 8-bit mode or ffffh to 0000h in 16-bit mode. This overflow sets the TMRxIf bit. The timer interrupt can be masked by cleaning the TMRxIE bit

⇒ The TMRxIf bit must be cleared in software by the timer module Interrupt service routines before re-enabling this interrupt.

⇒ Note : X may take value 0, 1, 2, 3.

⇒ Table for the timers and their corresponding interrupt enable flag bits and interrupt flag bit.

PICT, PUNE

ROLL NO: 28330

- ASSIGNMENT - 7 -

Timer	Interrupt Enable		Interrupt flag	
	Bit	Register	Bit	Register
Timer 0	TMR0IE	INTCON	TMR0If	INTCON
Timer 1	TMR1IE	PIE 1	TMR1If	PIR1
Timer 2	TMR2IE	PIE 1	TMR2If	PIR1
Timer 3	TMR3IE	PIE 2	TMR3If	PIR2

External hardware interrupts.

⇒ PIC18 microcontroller has three external hardware interrupts INT0, INT1, INT2. They are available on PORTB pins RB0, RB1 and RB2.

⇒ These interrupts are edge trigged interrupts i.e triggered by either a rising edge or by falling edge.

PROCEDURE:

STEP 1: ⇒ MPLAB IDE → Go to Project → Project Wizard → select device PIC18F4550 → next → Active toolsuit → microchip C18 Toolsuite → next.

⇒ Create new Project file → Create new Folder → click on next → Finish.

PICT, PUNE

ROLL NO : 23330.

- ASSIGNMENT - 7 -

⇒ Go to file - new - write code - save file with .c extension → add this file in source File → Right click on source file → add C-file.

STEP-2 : ⇒ Go to project → Build all Project.
→ Check for errors if No errors → Build successful will appear on the output window.

Step-3 : After Build project HEX file is created. Burn the HEX file in Proteus 8.

CONCLUSION: Thus we have studied embedded C program for timer programming ISR based buzzer on/off and external interrupt input switch press.