23365

Assignment 1

Name: Aditya Kangune	Batch: G11
Roll no: 23365	Date: 6/6/21.

A] Explain the status register of PIC18 in detail w.r.t the addition operation. Give example in which status register goes gets value (HEX):

i) 10 ii) 8 iii) 14 iv) 08 v) 11.

→

Status register holds the status flags for instruction 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 zero.

0 = Non-zero result.

DC :- Digital carry.

For ADDlnlf, ADDLW, instructions.

1: Carry from 4th low order low bit of result was occured.

0: No carry from 4th low order bit of result has occured.
C: Carry bit.

For ADD LW, ADDWF instructions:
1: A carry out of MSB of result has occurred.
0: No carry out of MSB of result.

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

Here, only negative bit is set. This result in the status register may occur when there is no carry or DC.
There is no overflow and there is a non-zero arithmetic answer.
The answer of addition must be negative.

Eg: 81 H
+ 11 H Status:- 0001 0000.
92 H

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

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

Eg: 62H
+ 33H Status:- 0001 1000.
95

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: FOH
+ 10H Status- 0000 0101.
100H

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: FF H
+ 03H
102H

v.) (11)16 = (0001 0001)2

Here, carry and negative flags are set. The answer is negative. There is no DC and overflow. The answer is non-zero.

F3 H
+92H Status: 0001 0001.
185 H

3.] 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 is to PORTx will make the corresponding PORTx pins low. There are 5 PORTx registers.

PORT A	F80H	Memory locations.
PORT B	F81H	
PORT C	F82H	
PORT D	F83H	
PORT E	F84H	

TRISx:

→ These registers are used to configure respective PORTx as input/output.

→ They are known as tristate or data direction.
→ Registers writing 1's to TRIS x will make corresponding PORT x pins as input.
→ Similarly, writing 0's to TRIS x will make corresponding PORT x pins as output.

TRIS A	F92H	Memory locations.
TRIS B	F93H	
TRIS C	F94H	
TRIS D	F95H	
TRIS E	F96H	

LAT X:
→ These are port latch registers when data is written to PORT x registers.
→ It is stored in LATX registers then copied to PORT x.
→ If TRIS register is not zero, data is not copied to PORT x but remains in LAT x.
→ When data is read from PORT x, it is first copied to LAT x and then WREG.
→ If TRIS x is not OFF, data is not copied from PORTx but from LAT x.

LAT A	F89H	Memory locations.
LAT B	F8AH	
LAT C	F8BH	
LAT D	F8CH	
LAT E	F8DH	

c.] Write a note on Embedded C programming:

→ 1) Embedded C is an extension of standard C programming language.
2) Additional features like addressing i/o, multiple memory addressing, fixed point arithmetic, etc.
3) Both C and Embedded C are ISO standards that have almost same syntax, datatypes, functions, etc.
4) C programming language is generally used for developing desktop applications, whereas embedded C is used to develop micro-controller based application.

→ Structure of Embedded C program:

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

X

Assignment 1 Input-

MPLAB X IDE v5.45 - Assignment_1_Adding : default

File Edit View Navigate Source Refactor Production Debug Team Tools Window Help

default

add.c | arrayAddition.c | mulanddiv.c | sorting array.c | addingmain.c | main.c

Source History

```
1   // Author: Aditya Kangune
2   /*
3   Study of Embedded C programming language
4    (Overview, syntax, One simple program like
5    addition of two numbers).
6    */
7
8   #include <xc.h>
9   #include<stdlib.h>
10  #include<stdio.h>
11  #include<PIC18F4550.h>
12  void main(void){
13      int a = 4, b = 3;
14      int c = a + b;
15      PORTB = c;
16      TRISB = 0x00;
17      return;
18  }
19
```

Assignment 1 Output-

```
12  void main(void){
13      int a = 4, b = 3;
```

I/O Pins

Pin	Mode	Value	Owner or *Mapping*
RB0	Aout	5.0V	**RB0**/AN12/INT0/FLT0/SDI/S
RB1	Aout	5.0V	**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

Configuration Bits | Program Memory | Configuration Bits | IO View | Output | Variables | Call Stack | Breakpoints | Watches | Assignment_1_Adding (Build, Load, ...)

23865

Assignment 2

Name: Aditya Kangune.	Batch: G11
Roll no: 23865	Date: 6/6/2021.

Q.A] Write a short note on working register (WREG) of PIC 18.

→ 1.) The working register or WRG is a temporary storage for information at CPU.

2.) It has a size of CPU.

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

4.) Used to perform arithmetic and logical instructions like MOVE and ADD.

This means move literal value a is the WREG.

→ A program to ADD two numbers:

```
MOVL  W = 03H
ADDL  W = 50H
```

Now, 53H is stored in WREG.

5.) It is same as accumulator in other microprocessor.

Q.B] Memory of PIC 18F458 (all types) and memory banking.

→ 1.) Memory consists of a sequence of directly addressable locations.

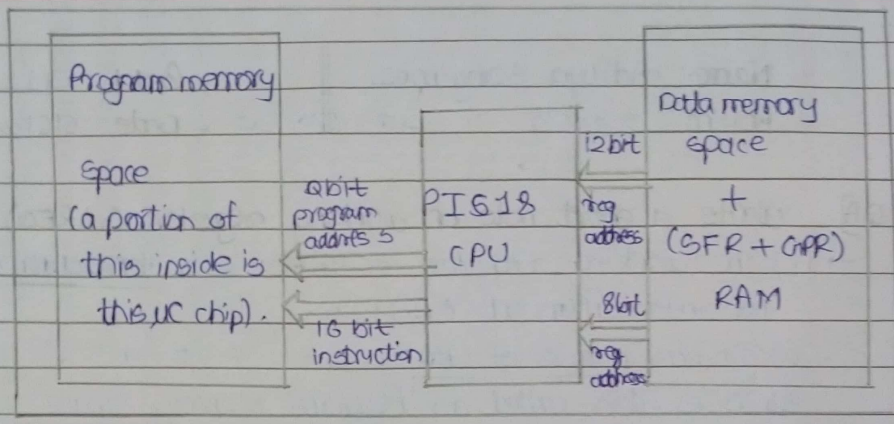
2.) A location is reffered to as an information unit.

3.) A memory location can be used to store data, instruction and status of peripheral devices.

4.) A memory called location has two components: an address and its contents.

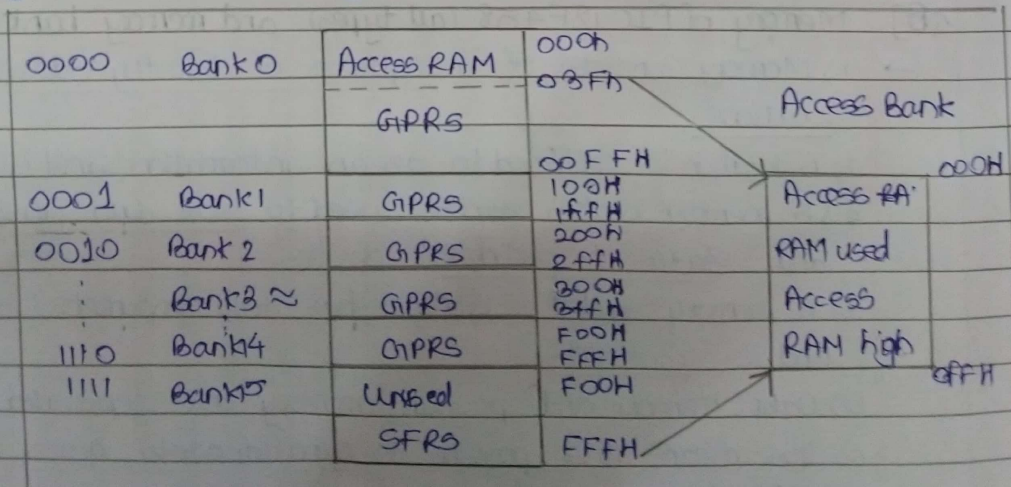
5.) Data memory and program memory are separated.

6.) This means it is possible to simultaneously access data and instructions.

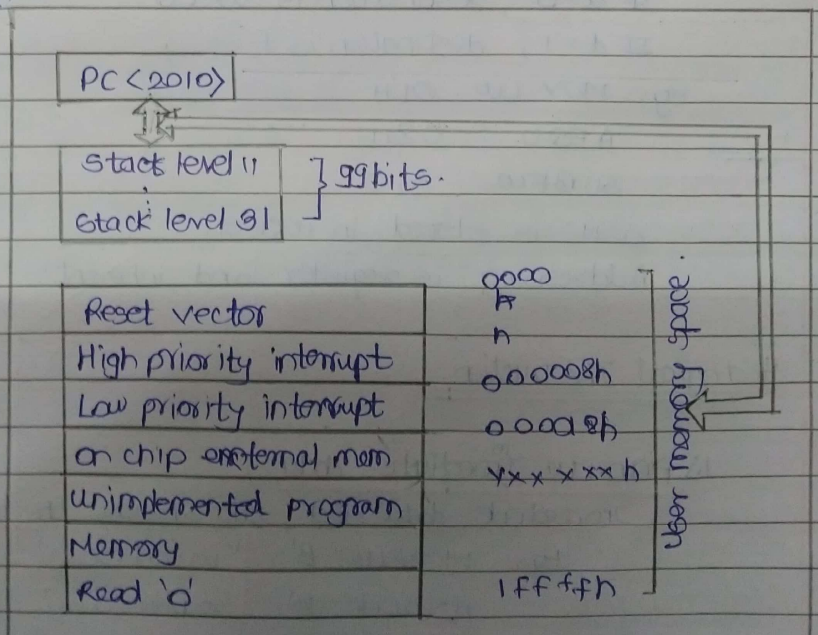


Memory features:

1.) 4096 bytes of data memory.
2.) General purpose registers are used to hold dyamic data.] Data registers.
3.) Special Function registers are used to control operation of] registers. peripheral function.
4.) only one bank is active at a time and is specified by the BSR register.
5.) PIC 18 implements access banks to reduce problems caused by bank shifting.
6.) Access bank consists of lowest 96 bytes and highest 160 bytes of data memory space.



7.) Every bank has 256 bytes.
8.) PIC18 has 2 MB program space.
1MB = 16 bytes.
Hence PIC18 has 16 bytes.
9.) 0-95 of first bank is access RAM (low) higher 160 of last bank. Access RAM (high).
10.) Unused space is for some SRF's that can be added in later versions.
11.) PC is 21 bits long which enables user program to access upto 2MB of program memory.
12.) After power on, PIC18 starts executing instructions from address.
13.) Location or address 0x08 is reserved for high priority ISR and 0x18 for low priority ISR.
14.) Upto 128 kB of program memory MCU chip at present time.



Q c] Instruction for arithmetic, logical branch and bitwise operations. Give 2 examples of each type.

→ # Arithmetic Instructions:

1.) ADDCWF f, d

Contents of file registers F and WREG are added.

If d = 0, destination is WREG.

d = 1, destination is F.

Eg: MOVLW 12H

AEU 18H

ADDAO

In this program 25H is stored in WREG, addressing mode is direct and inherent.

2.) SUBFW f, d

Contents of WREG are subtracted from the file register.

If d = 0, destination is WREG.

If d = 1, destination is F.

Eg: MOV LW 01H

AEQU 03H

SUBFW A, 0

02H is stored in WREG.

Addressing is register and inherent.

Logical Instruction.

1.) ANDLW Immediate data:

Immediate data and data inside the WREG are "ANDed."

Eg: MOVLW B' '10001000'

ANDLW B' '01011000'

(00001000)2 is stored in the WREG.

Addressing mode is immediate and inherent.

2.) IORLW intermediate data.

Intermediate data and data inside WREG is 'ORED'.

Eg: MOVLW B' '00100101'

IORLW B' '11010001'

(11110101)2 is stored in WREG A.

Addressing mode is stored in WREG A.

intermidiate and inherent.

BRANCH :-

1.) BNZ address : Branch if not zero.

2.) BZ address : Branch is zero.

BITWISE :-

1.) BCF F,b :

Clear bit B

A EQU B' '00011111'

BCF A,0

value on A becomes (0001 0111)2

2.) BSF F,b :

Set bit B

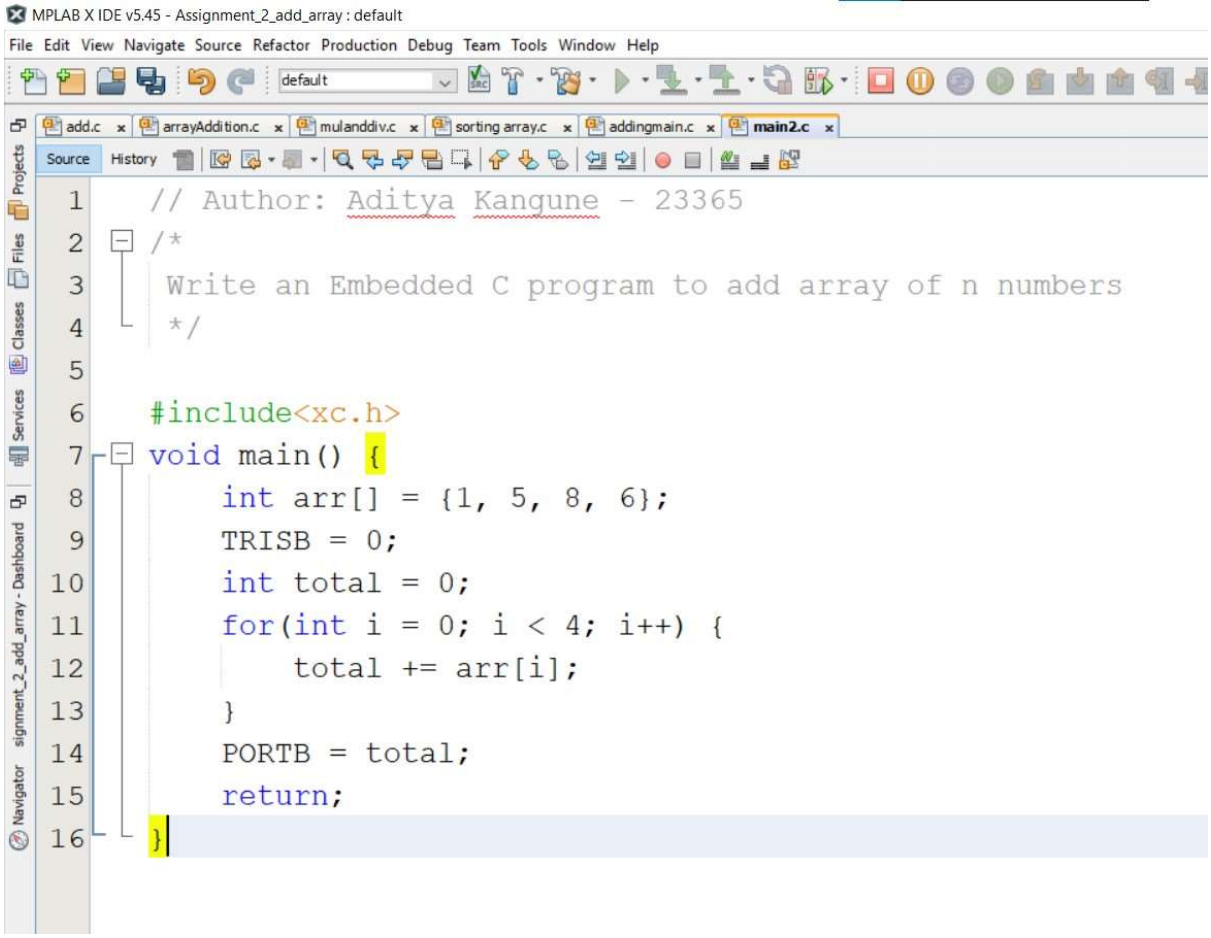
A EQU B' '0001 1111'

BCF A,1

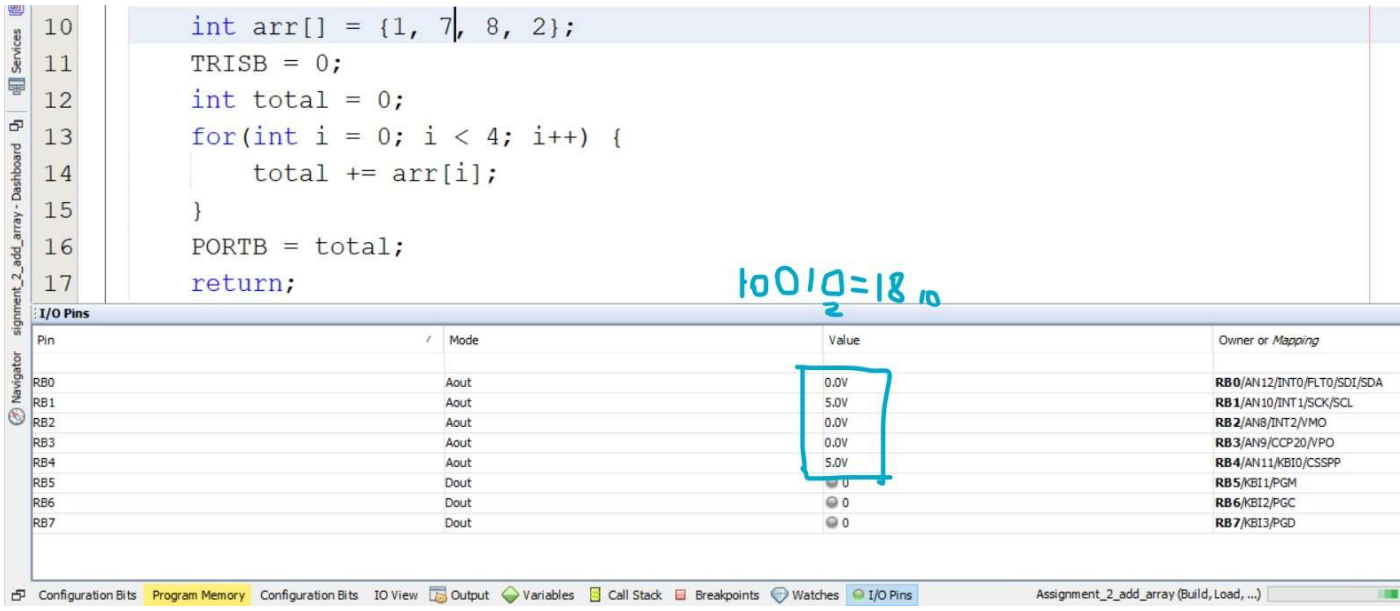
Value of A becomes (1001 0111)2.

X

Assignment 2 Input-



Assignment 2 Output-



23365

Assignment 3

Name: Aditya Kangune.
Roll no: 23365

Batch: G11
Date: 5/6/2021.

Q1.] Write a note on MPLAB IDE :

→ 1.) MPLAB is property freeware integrated development environment for the development of embedded applications on PIC and dsPIC µcontrollers, and is developed by Microchip Technology.

2.) MPLAB X is the latest edition, and is developed on the NETBeans platform.

3.) MPLAB and MPLABx support project management, code editing, debugging and programming of Microchip 8-bit PIC and AVR µcontrollers, 16bit PIC24 and dsPIC µcontrollers, as well as 32 bit SAM (ARM) and PIC32 (MIPS) µcontrollers.

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

Q2.] Write an assembly code for PIC18 to multiply two 8 bit numbers using consecutive addition.

→ Multiply 0x05 and 0x0A.

```
COUNT EQU 0x30
      MOVLW 0x05
      MOVWF COUNT
      MOVWF 0

LOOP  ADDLW 0x0A
      DECFSZ COUNT, F
      GOTO LOOP
```

Q3] Write an Assembly language code program for PIC18 to divide one 8 bit number by another 8 bit number using consecutive subtractions.

→ Divide 0x0A by 0x02

```
QUOTIENT EQU 0x30

        MOVLW 0x0A

LOOP    SUBLW 0x02
        INCF QUOTIENT, F
        BN LOOP

        DECF QUOTIENT, F.
```

—— X ——

23365

Assignment 4

Name: Aditya Kangune | Batch: G1
Roll no: 23365 | Data: 6/6/2021.

Q1.] Write a note on Timer 0, 1, 2, and 3.

→ 1] Timer 0:

i.) Timer contol registers

- TOCON is Timer 0 control registers.
- TOCON is an 8 bit register with following bits:

2.) TMR00N
T08BIT
TOCS
TOSE
PSA
TOPS1
TOPS2
TOPS0
} Prescalar select bits.

ii.) Timer registers:

-Timer zero uses a 16 bit register represented by TMR0H (Timer 0 higher byte) as higher byte register and TMR0L (Timer 0 Lower byte) as Lower byte register.

iii.) Timer interrupt flag:

- The interrupt flag bit for Timer 0 is TMR01F which is a port of INTCON (Interrupt contol) register.

iv) Delay calculations:

a.) Crystal frequency = X

b.) Timer frequency = t = X/4 , t/Prescalar in case prescale ON.

c.) Timer period = $T = \frac{1}{t}$

d) count = FFFH - (TMR0H.TMR0L) + 1 = C
e) Timer delay = D = CXT
f) Time delay of half cycle = 2 * D.

2) Timer 1.

i) Time control registers:
- T1CON is Timer 1 control register.
- T1CON is an 8 bit register with following bits:
  - RD16
  - Not used
  - T1CKPS1] Prescaler select bits
  - T1CKPS0
  - T1OSCEN
  - T1SYNC
  - TMR1CS
  - TMR1ON

ii) Timer registers:
- Timer 1 uses a 16 bit register which is split into two bytes, referred as TMR1L (Timer 1 low byte) and TMR1H (Timer 1 high byte).
- Timer 1 can be programmed in 16 bit mode only.

iii) Timer interrupt flag:
- The interrupt flag bit for Timer 1 is TMR1IF which is a part of PIR1 register.

iv) Delay calculations:
a) Crystal frequency = X
b) Timer frequency = $t = \frac{X}{4}$

$= t / \text{prescalar value}$, when prescalar > 1.

c.) Timer period = $T = \frac{1}{t}$

d.) Count = C = FFFH - (TMR1H . TMR1L) + 1

e.) Time delay = D = T1 x C.

f.) Time delay of half cycle = 2 x D.

3.] Timer 2

i.) Timer control register:

- T2CON is Timer 2 control register.
- T2CON is an 8 bit register.
- T2CON contains following 8 bits.

NOT used
TOUTPS3]
TOUTPS2]
TOUTPS1] → Prescale select bits.
TOUTPS0]
TMR2ON
T2CKPS1] → Prescale select bits.
T2CKPS0]

ii.) Timer registers:

- The 8 bit register of Timer 2 is called TMR2.
- Timer 2 also has an 8 bit register called period register (PR2).
- The value of PR2 can be select set to a fixed value and Timer 2 can then increments till the same value in PR2.

iii.) Timer interrupt flag:

- The interrupt flag for Timer 2 is TMR2IF, which is a part of PIR1.

iv.) Delay calculations:

a.) Crystal frequency = X

b.) Timer frequency = $\frac{X}{4}$ = t.

c.) Timer period = Postscale value × Prescale value × $\frac{1}{t}$ = T.

d.) Count = C = TMR2H = PR2.

e.) Time delay = C × T = D

f.) Time delay of half cycle = 2 × D.

4.] Timer 3

i.) Timer control register:

- T3CON is timer 3 control register.
- T3CON is 8 bit register with following bits.

RD16

T3CCP2] Clock source for compare/capture of CCP module.

T3CCP1]

T3CKPS1] Prescale value.

T3CKPS0]

T3SYNC

TMR3CS

TMR3ON

ii.) Timer registers:

- Timer 3 is 16 bit register split as TMR3L (Timer 3 low byte) and TMR3H (Timer3 High byte).

iii.) Timer interrupt flag:

- The interrupt flag bit of Timer 3 is called TMR3IF which is a part of PIR2 register.

iv.) Delay calculations:

a.) Crystal frequency = X

b.) Timer frequency = $t = \frac{X}{4}$

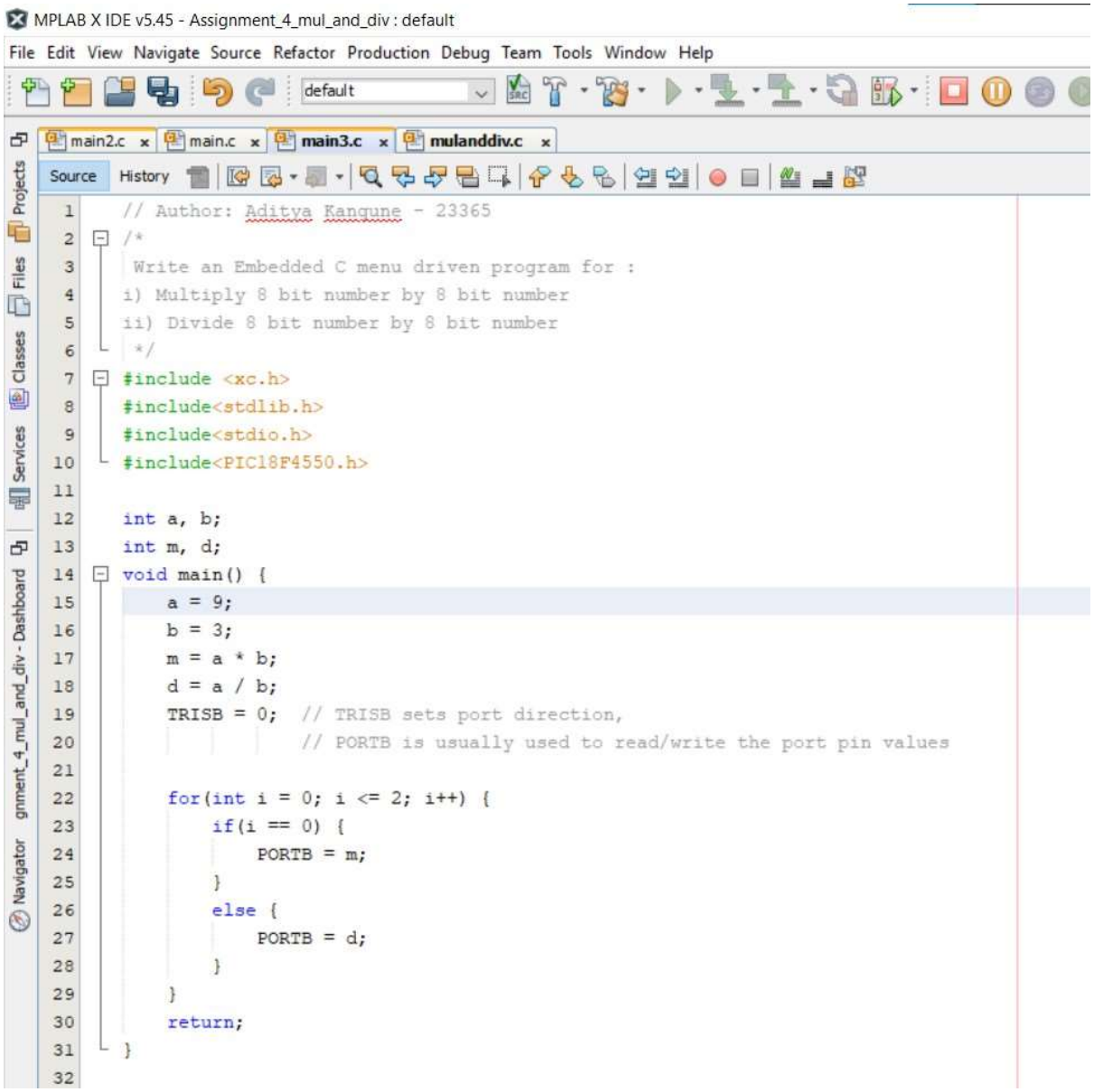
c.) Timer period = $\frac{1}{t}$ x prescale value = T.

d.) Count = C = FFFFH - (TMR3H. TMR3L) + 1.

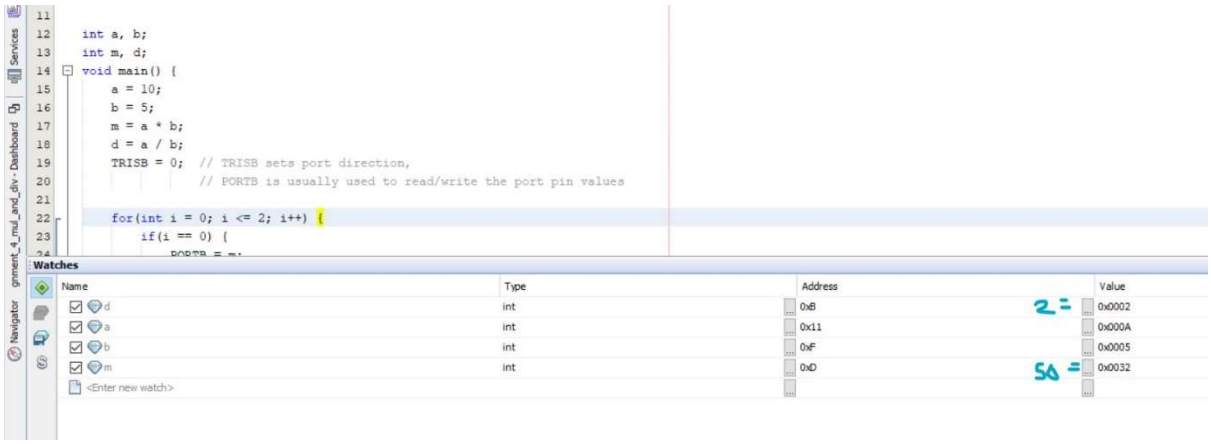
e.) Timer delay = C X T = D.

f.) Timer delay for half cycle = 2xD.

Assignment 4 Input-



Assignment 4 Output-



23365

Assignment 5

Aim: write an Embedded C program for sorting the numbers in Ascending order.

Software Used: MPLABX IDE v5.45

Theory: The array can be stored in ascending order by repeatedly finding the minimum element, in ascending order, from the unsorted part and putting it at the beginning.
The algorithm maintains two subarrays in an array.
1.) The subarray which is already sorted.
2.) Remaining unsorted subarray.

Procedure:

Step1: Creating a new project:
a) Go to file tab.
b) Click on new project.
c) Microchip ⇒ Standalone.
d) Click 'Next'.

Step2: Select device:
- Select : Family ⇒ Advanced 8 bit MCU (PIC 18)
- Select : Device : PIC18F4550 → click next.

Step3: Select tool : Simulator : Next.

Step4: Select compiler XC8, next.

Step5: Select project name and folder.
- Give project name.
- Select project location
- Uncheck set as starting main position.

- Click finish.

Step6: Creating new source file.
- Go to project location in project location.
- Click the '+' sign to open the project space.
- Right click on the source file folder (for a C file).
- New → C source file.

Step7: Select build or clean build
- Debug the project - check for errors.
- If NO errors, build successful will appear.
- Go to Debug window - discrete debug operation - build for debugging - launch for debugger option.
- For single stepping press F8.
- Go to window option, select target memory views
- then File register.

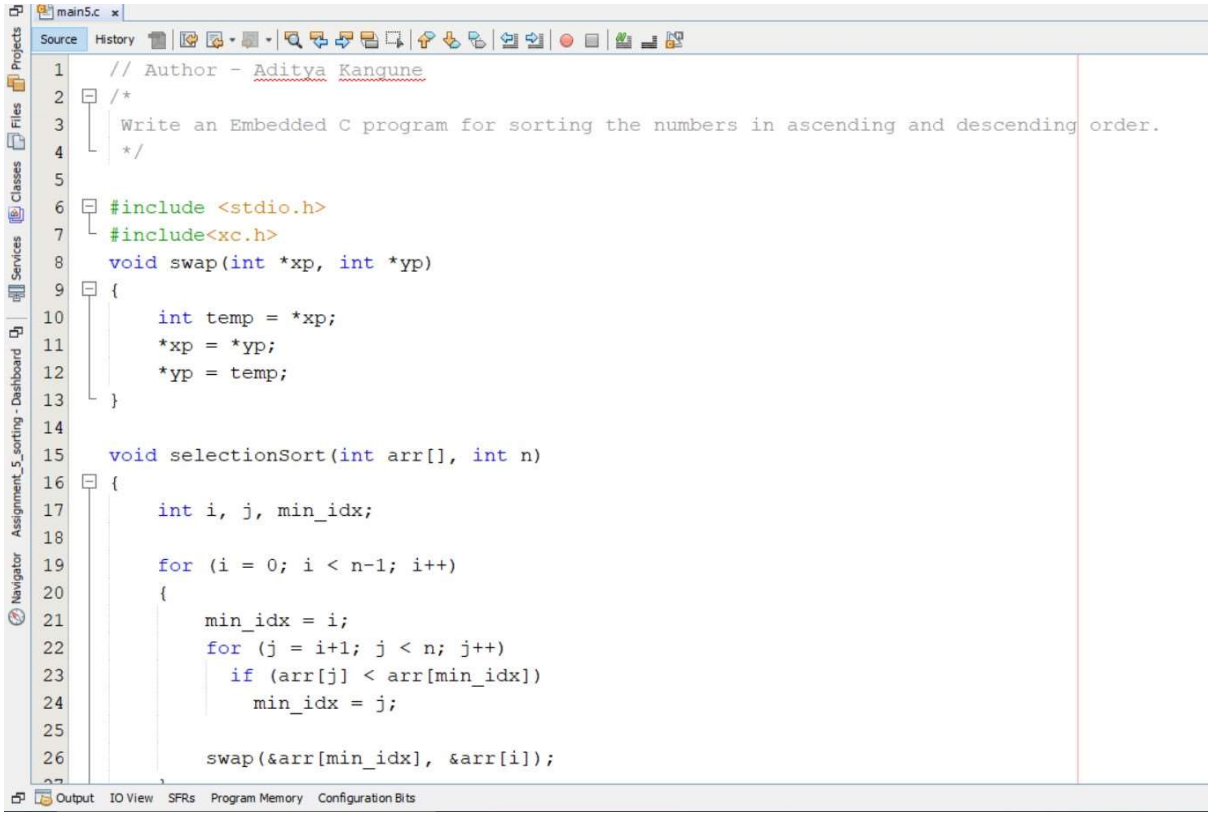
Program code: (Attached below)

Output: (Attached below).

Conclusion: Using MPLAB, an unsorted array was sorted in ascending as well as descending order successfully.

X

Assignment 5 Input-



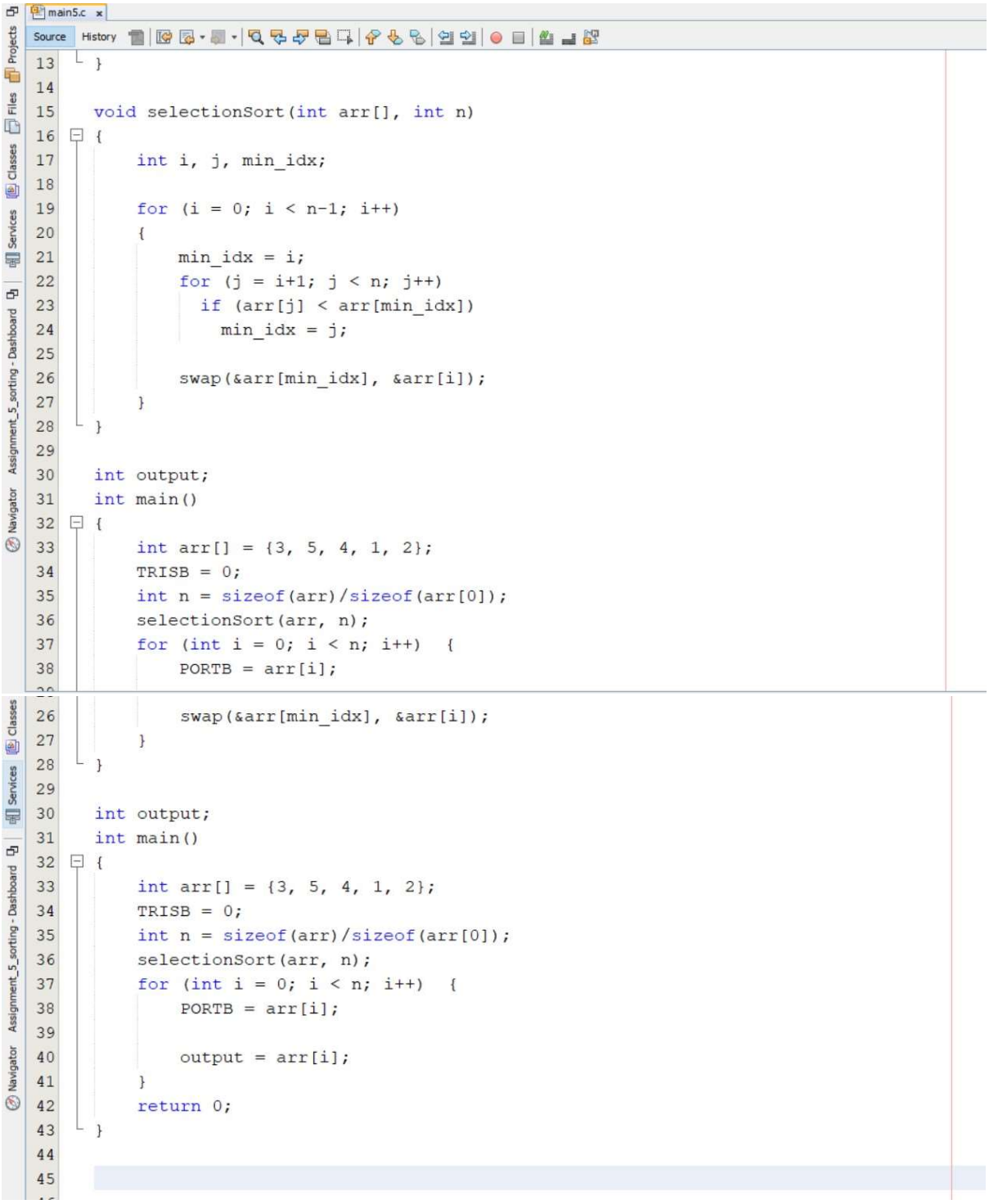

```c
// Author - Aditya Kangune
/*
 Write an Embedded C program for sorting the numbers in ascending and descending order.
 */

#include <stdio.h>
#include<xc.h>
void swap(int *xp, int *yp)
{
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}

void selectionSort(int arr[], int n)
{
    int i, j, min_idx;

    for (i = 0; i < n-1; i++)
    {
        min_idx = i;
        for (j = i+1; j < n; j++)
          if (arr[j] < arr[min_idx])
            min_idx = j;

        swap(&arr[min_idx], &arr[i]);
```



```
}

void selectionSort(int arr[], int n)
{
    int i, j, min_idx;

    for (i = 0; i < n-1; i++)
    {
        min_idx = i;
        for (j = i+1; j < n; j++)
          if (arr[j] < arr[min_idx])
            min_idx = j;

        swap(&arr[min_idx], &arr[i]);
    }
}

int output;
int main()
{
    int arr[] = {3, 5, 4, 1, 2};
    TRISB = 0;
    int n = sizeof(arr)/sizeof(arr[0]);
    selectionSort(arr, n);
    for (int i = 0; i < n; i++)  {
        PORTB = arr[i];
```

```
        swap(&arr[min_idx], &arr[i]);
    }
}

int output;
int main()
{
    int arr[] = {3, 5, 4, 1, 2};
    TRISB = 0;
    int n = sizeof(arr)/sizeof(arr[0]);
    selectionSort(arr, n);
    for (int i = 0; i < n; i++)  {
        PORTB = arr[i];

        output = arr[i];
    }
    return 0;
}
```

// Author - Aditya Kangune – 23365 – Assignment 5

/*

Write an Embedded C program for sorting the numbers in ascending and descending order.

*/

#include <stdio.h>

```
#include<xc.h>
void swap(int *xp, int *yp)
{
  int temp = *xp;
  *xp = *yp;
  *yp = temp;
}

void selectionSort(int arr[], int n)
{
  int i, j, min_idx;

  for (i = 0; i < n-1; i++)
  {
    min_idx = i;
    for (j = i+1; j < n; j++)
      if (arr[j] < arr[min_idx])
        min_idx = j;

    swap(&arr[min_idx], &arr[i]);
  }
}

int output;
int main()
{
  int arr[] = {3, 5, 4, 1, 2};
  TRISB = 0;
  int n = sizeof(arr)/sizeof(arr[0]);
  selectionSort(arr, n);
  for (int i = 0; i < n; i++)  {
```

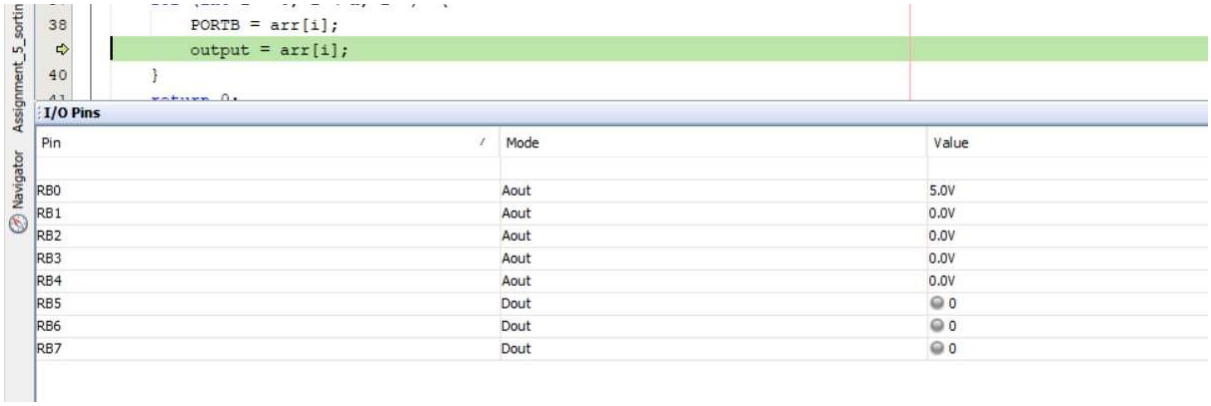
```
    PORTB = arr[i];


    output = arr[i];
  }
  return 0;
}
```

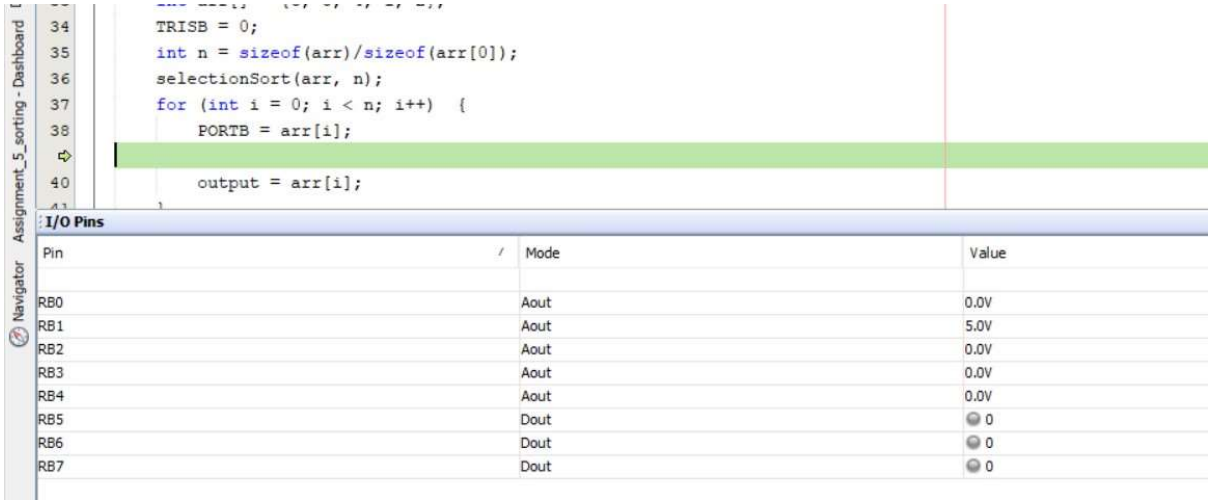
Assignment 5 Output -

arr = {3, 5, 4, 1, 2} // Unsorted

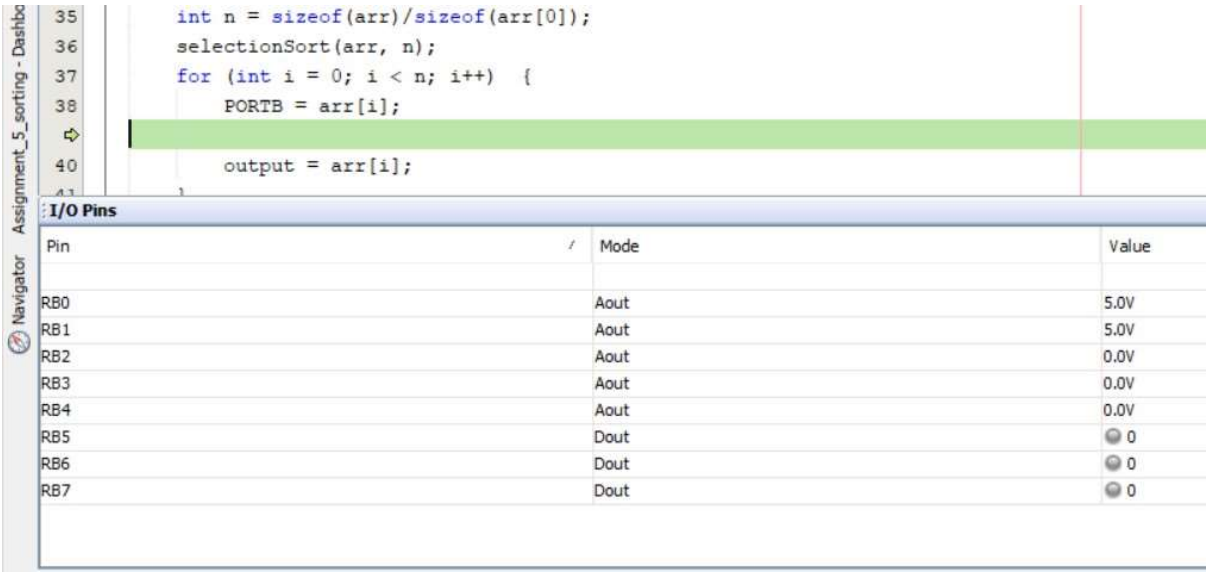
1-



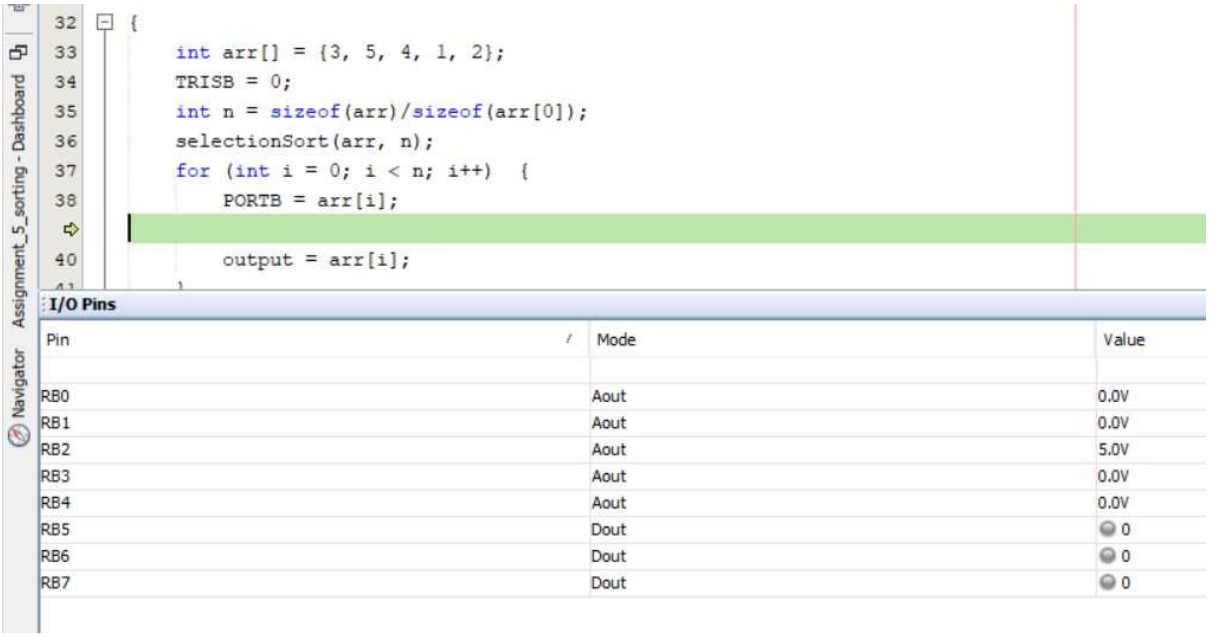
2-



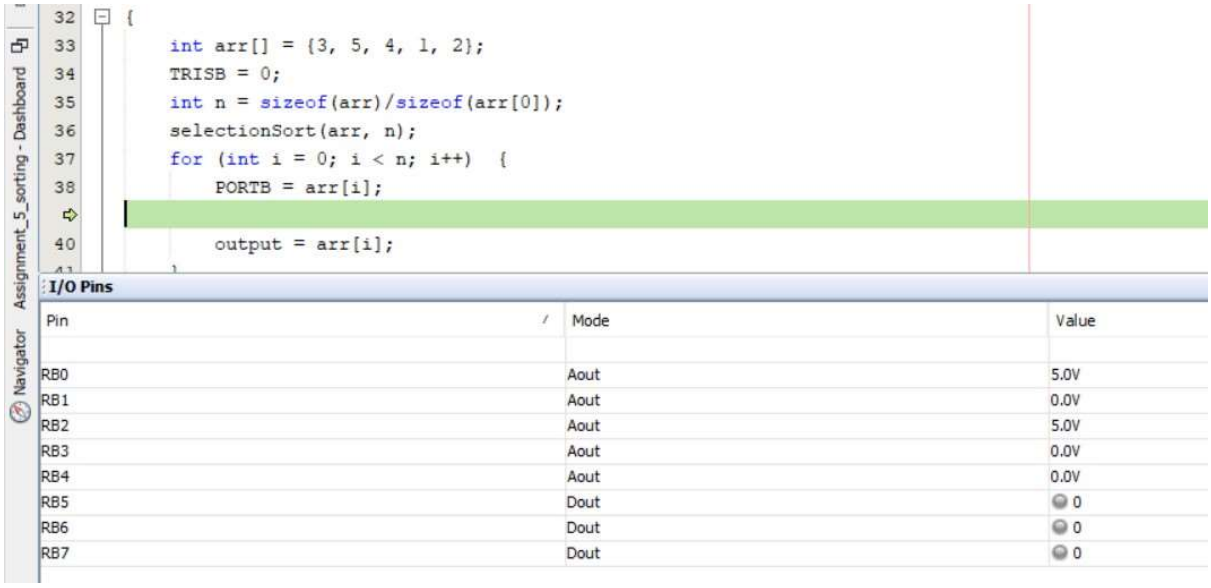
3-


Assignment_5_sorting - Dashbo
Navigator
35 int n = sizeof(arr)/sizeof(arr[0]);
36 selectionSort(arr, n);
37 for (int i = 0; i < n; i++) {
38 PORTB = arr[i];
40 output = arr[i];
I/O Pins
Pin
Mode
Value
RB0 Aout 5.0V
RB1 Aout 5.0V
RB2 Aout 0.0V
RB3 Aout 0.0V
RB4 Aout 0.0V
RB5 Dout 0
RB6 Dout 0
RB7 Dout 0

4-


Assignment_5_sorting - Dashboard
Navigator
32 {
33 int arr[] = {3, 5, 4, 1, 2};
34 TRISB = 0;
35 int n = sizeof(arr)/sizeof(arr[0]);
36 selectionSort(arr, n);
37 for (int i = 0; i < n; i++) {
38 PORTB = arr[i];
40 output = arr[i];
I/O Pins
Pin
Mode
Value
RB0 Aout 0.0V
RB1 Aout 0.0V
RB2 Aout 5.0V
RB3 Aout 0.0V
RB4 Aout 0.0V
RB5 Dout 0
RB6 Dout 0
RB7 Dout 0

5-


Assignment_5_sorting - Dashboard
Navigator
32 {
33 int arr[] = {3, 5, 4, 1, 2};
34 TRISB = 0;
35 int n = sizeof(arr)/sizeof(arr[0]);
36 selectionSort(arr, n);
37 for (int i = 0; i < n; i++) {
38 PORTB = arr[i];
40 output = arr[i];
I/O Pins
Pin
Mode
Value
RB0 Aout 5.0V
RB1 Aout 0.0V
RB2 Aout 5.0V
RB3 Aout 0.0V
RB4 Aout 0.0V
RB5 Dout 0
RB6 Dout 0
RB7 Dout 0

23365

Assignment 7

Name : Aditya Kangune
Roll no: 23365

Batch : G11
Date : 5/6/2021.

Aim: Write an Embedded C program for Timer programming ISR based buzzer on/off.

Software used: MPLAB X IDE v5.45.

Theory: PIC Interrupt:

PIC μcontroller consists of both hardware and software interrupt.

- If the interrupts are generated by external hardware at certain pins of μcontroller or by default input devices like timer, they are called Hardware interrupts.
- While software interrupts are generated by a piece of code in the program.

IVT Table:

- It stands for Interrupt Vector Table.
- It holds the addresses of the interrupt device service routine, for various interrupts supported by the μcontroller.

IVT for PIC18:

Interrupt	Vector address (RAM location)
Poweron reset	0000H
High priority Interrupt	0008 H
low priority interrupt:	0018H

Programming of Timer using interrupt:

1.) 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.
2.) The timer interrupt can be masked by clearing the TMRxIE bit.
3.) Note: x may take value 0, 1, 2, 3.
4.) Table for the timers and their corresponding interrupt enable flag bits and interrupt flag bit.

Timer	Interrupt Enable		Interrupt Flag	
	Bit	Register	Bit	Register
Timer0	TMR0IE	INTCON	TMR0IF	INTCON
Timer1	TMR1IE	PIE1	TMR1IF	PIR1
Timer2	TMR2IE	PIE2	TMR2IF	PIR1
Timer3	TMR3IE	PIE2	TMR3IF	PIR2

External Hardware interrupt:

1.) PIC Microcontroller has 3 external hardware interrupts INT0, INT1, INT2.
2.) They are available on PORTB pins RB0, RB1 and RB2.
3.) These interrupts are edge triggered instructions, i.e. triggered by either a rising or a falling edge.

Procedure:

Step 1:- MPLAB X IDE → Go to project → Project Wizard → Select Device → PIC 18F4550 → next → Active toolkit → Microchip C18 Toolsuite → next.

- Create new project file → Create new folder → click on next → Finish.
- 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 any → Build successful will appear if no errors.

Step 3: After successful build, HEX file is generated. Burn the HEX file in PROTEUS 8.

Conclusion: Timer programming ISR based buzzer ON/OFF program has been successfully understood and studied using Embedded C.

X