**Name:**

**Roll no :**

**Group A Lab Assignment:** 2

**Subject:** PSDL

**Title:** Array Addition

**Assignment No: 2**

**Title :** Embedded C programming for Array addition of n numbers

**Aim:** Write an Embedded C Program to add an Array of n numbers

**Objective:** To develop and execute embedded C program to perform array addition for PIC18FXXX microcontrollers**.**

**Theory:**

**Programming of PIC microcontroller in Embedded C**

1. **Preprocessor Directives in Embedded C programming**

Preprocessor directives are not normal code statements. In Embedded C, Preprocessor Directives are usually represented using # symbol like #include… or #define….. In Embedded C Programming, we usually use the preprocessor directive to indicate a header file specific to the microcontroller, which contains all the SFRs and the bits in those SFRs.

They are lines of code that begin with the character "#" and appear in Embedded C programming before the main function. At runtime, the compiler looks for preprocessor directives within the code and resolves them completely before resolving any of the functions within the code itself. Many types of preprocessor directives are available in the Embedded C language. The **stdio**.**h** header defines three variable types, several macros, and various functions for performing input and output.

In case of PIC18F4550, XC8 Compiler has the header file “#include<pic18f4550.h>”, which must be written at the beginning of every Embedded C Program.

**1.1).Global Variable Declaration**

Global declarations happen before the main function of the source code. Engineers can declare global variables that may be called by the main program or any additional functions or sub-programs within the code. Engineers may also define functions here that will be accessible anywhere in the code.

**1.2).** **Main Program**

The main part of the program begins with main(). If the main function is expected to return an integer value, we would write int main(). If no return is expected, convention dictates that we should write void main(void).

**1.3).** **Declaration of local variables**

Unlike global variables, these ones can only be called by the function in which they are declared.

**1.4).** **Initializing variables/devices**

A portion of code that includes instructions for initializating variables, I/O ports, devices, function registers, and anything else needed for the program to execute

**1.5).Program body**

Includes the functions, structures, and operations needed to do something useful with our embedded system

**2). Keywords, Identifiers, variables and datatypes in Embedded C**

**2.1). Keywords:**

Keywords are predefined, reserved words used in programming that have special meanings to the compiler. Keywords are part of the syntax and they cannot be used as an identifier. For example int money; Here, int is a keyword that indicates money is a variable of type int (integer).

**2.2). Identifiers:**

In embedded C identifiers are names given to different user defined things like variables constants, functions, classes, objects, structures etc. An identifier is a sequence of digits, underscores, lowercase and uppercase Latin letters, and most Unicode characters. The first character of an identifier name must be a non-digit (including the underscore \_ , the lowercase and uppercase Latin letters, and other characters). You cannot use keywords like int, while etc. as identifiers.There is no rule on how long an identifier can be. However, you may run into problems in some compilers if the identifier is longer than 31 characters

You create an identifier by specifying it in the declaration of a variable, type, or function. In the below example, newworld is an identifier for an integer variable, and main is identifier names for function.

**#include <stdio.h>**

**int** main()

{

**int** newworld;

**2.3). Variables:**

Variables are simple keywords that are defined by the values. Values can be changed. Variables are like a box with some size where values like apples can be put in. So variables can be various forms and sizes, so-called variable types.

Variable type is defined by a reserved word which indicates the type and size of variable identifier:

*unsigned char my\_char;*

*long int all\_my\_numbers;*

*int number****;***

*2***.4). Data types:**

Embedded C supports three different data types for integers: int, short, and long. On 8-bit architectures, the default size of int values is typically set to 16 bits but Embedded C allows for int sizes to be switched between 8 and 16 bits to reduce memory consumption.

The short int data type allows embedded engineers to specify and integer value that is just one or two bytes in size.

Using the long int data type allocates twice the standard amount of memory as the int data type - usually 16 bits on most 8-bit platforms. The following are the extra data types in Embedded C associated with the Keil’s Cx51 Compiler.

* bit
* sbit
* sfr
* sfr16

**Procedure:**

**Step 1:** Open MPLABX IDE on the PC for program development and create a new project and save it in a new folder.

**Step 2:** Write the program in C language to add an Array of n numbers

**Step 3:** Build the program and create hex file. In case of errors correct program and rebuild to create hex file.

**Step 4:** Select Window->Target memory views->**Configuration bits** from tool bar. Select appropriate settings, generate source code and paste the configurations in the main program.

**Step 5:** Select debug project and then Finish debugger session from the tool bar.

**Step 6:** Select Window->Target memory views->**SFRs to view the output.**

**Source code :**

#include <stdio.h>

#include <stdlib.h>

#include <pic18f4550.h>

void main(void)

{

int i,sum;

int number[]={1,2,3,4,5,6,7,8,9,10};

sum = 0;

for(i=0;i<=9;i++)

{

sum = sum+number[i];

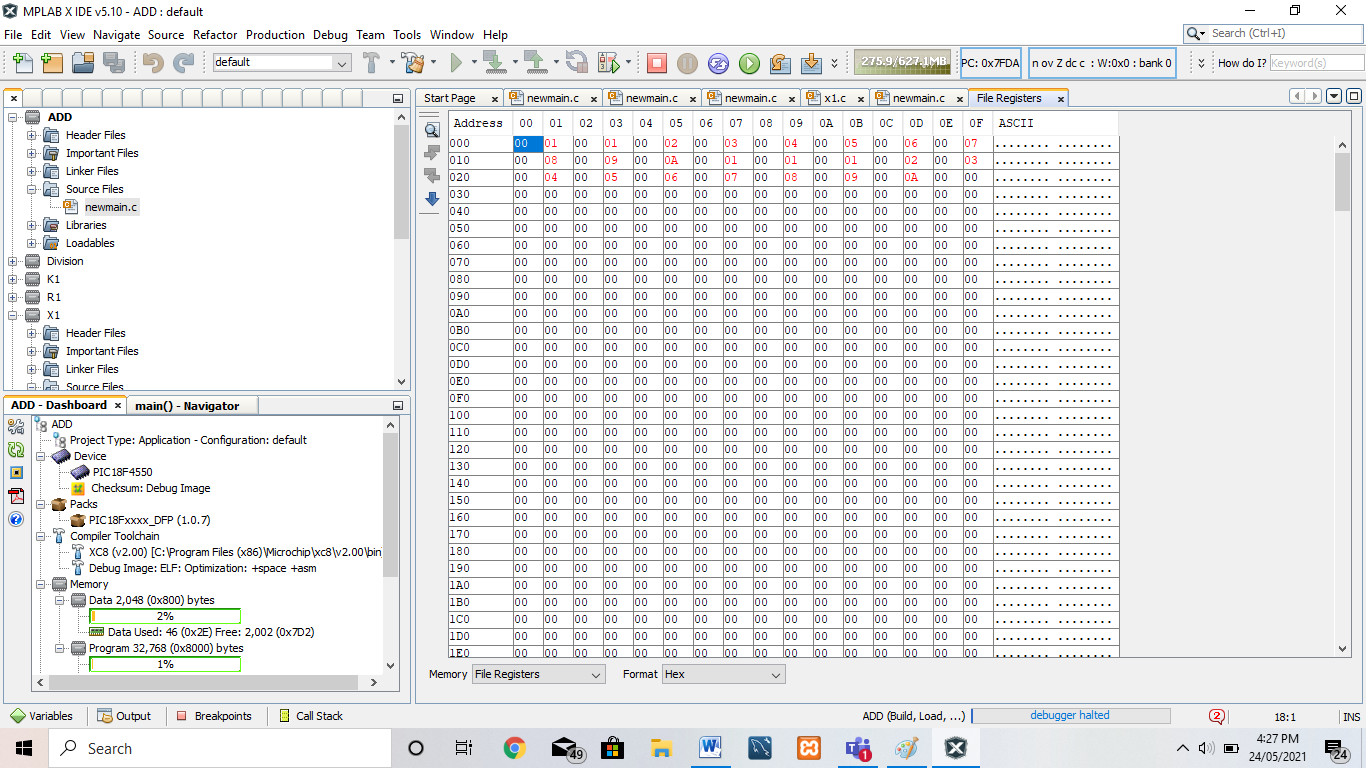
}

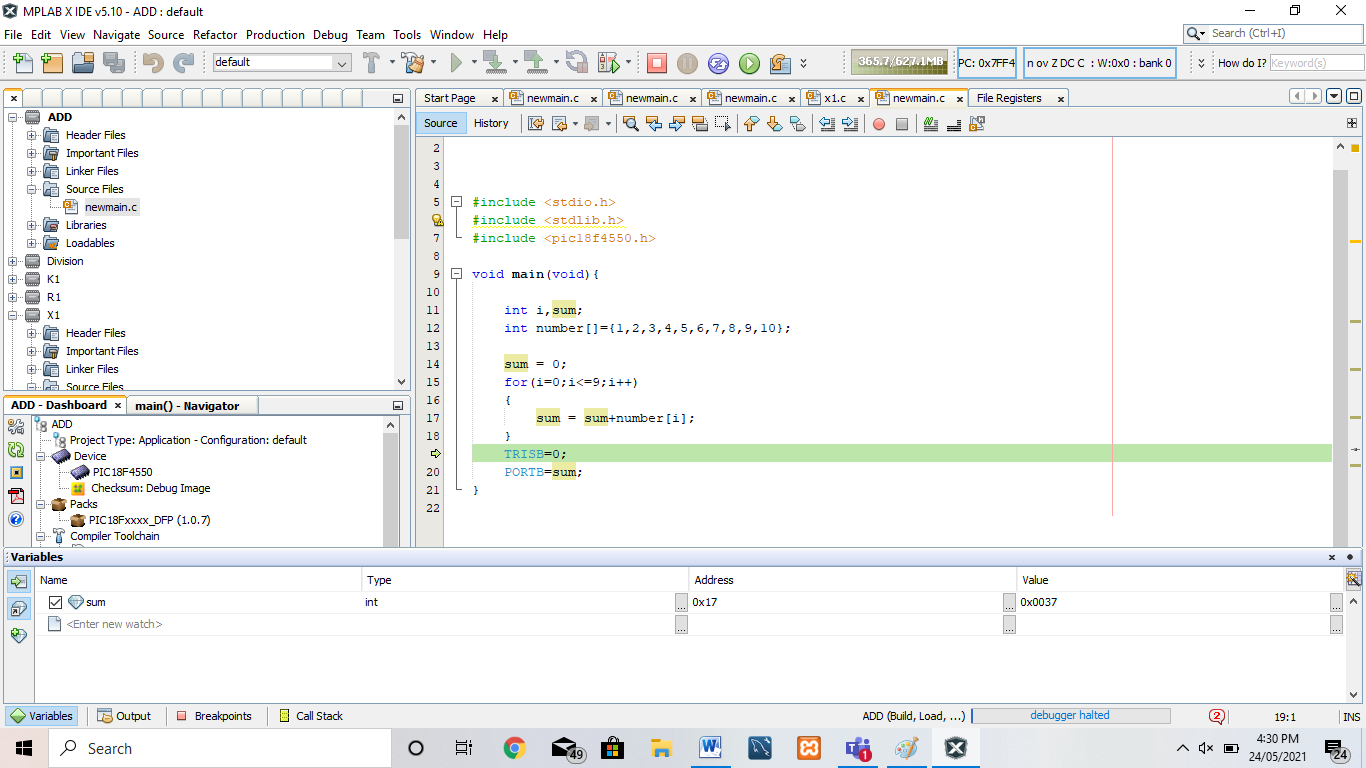
TRISB=0;

PORTB=sum;

}

**OUTPUT:**



****

**Conclusion:** Thus, we have studied embedded C programming for PIC18FXXX microcontrollers and perform array addition of n numbers