**COMPUTER SYSTEM DESIGN**

**(ULC401)**

**LABORATORY FILE**

**B.E. Electrical and Computer Engineering**

(2ND YEAR)



**SUBMITTED BY: ISHAN GROVER (102219028)**

**SUBMITTED TO: DR. ASHISH SONI**

THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY, PATIALA

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SNO.** | **LAB ASSIGNMENT** | **DATE OF PERFORMANCE** | **SIGNATURE** | **REMARKS** |
| 1. | **I**ntroduction to MP lab and creating a file PIC microcontroller**.** | 02/02/2024 |  |  |
| 2. | Write a program to Add/Sub two 8-bit number in the MP lab. | 02/02/2024 |  |  |
| 3. | Write a program to Add your roll no (last 2 digit) by 8 times and store value in file register. | 09/02/2024 |  |  |
| 4. | Write a program to add 2, 8-bit number one stored in File Location 20H and other in working register. After addition store the result in File Location 40H (Upper Byte) and 41H (Lower Byte). | 09/02/2024 |  |  |
| 5. | Write a program to add two 8-bit numbers. One is stored in file register and other in working register. | 16/02/2024 |  |  |
| 6. | Write a program to add 5 hexadecimal Number (8-bit) stored at memory location 20H onward, consecutively. Check the status of the carry flag and store the result in 40H (Upper byte) and 41H (lower byte). | 23/02/2024 |  |  |
| 7. | Write a program to perform Boolean operation between two number stored in location 00 and 01. Also store the result in 20h onward. | 05/04/2024 |  |  |
| 8. | Write a program to compare two number stored in the location 00 and 01. | 05/04/2024 |  |  |

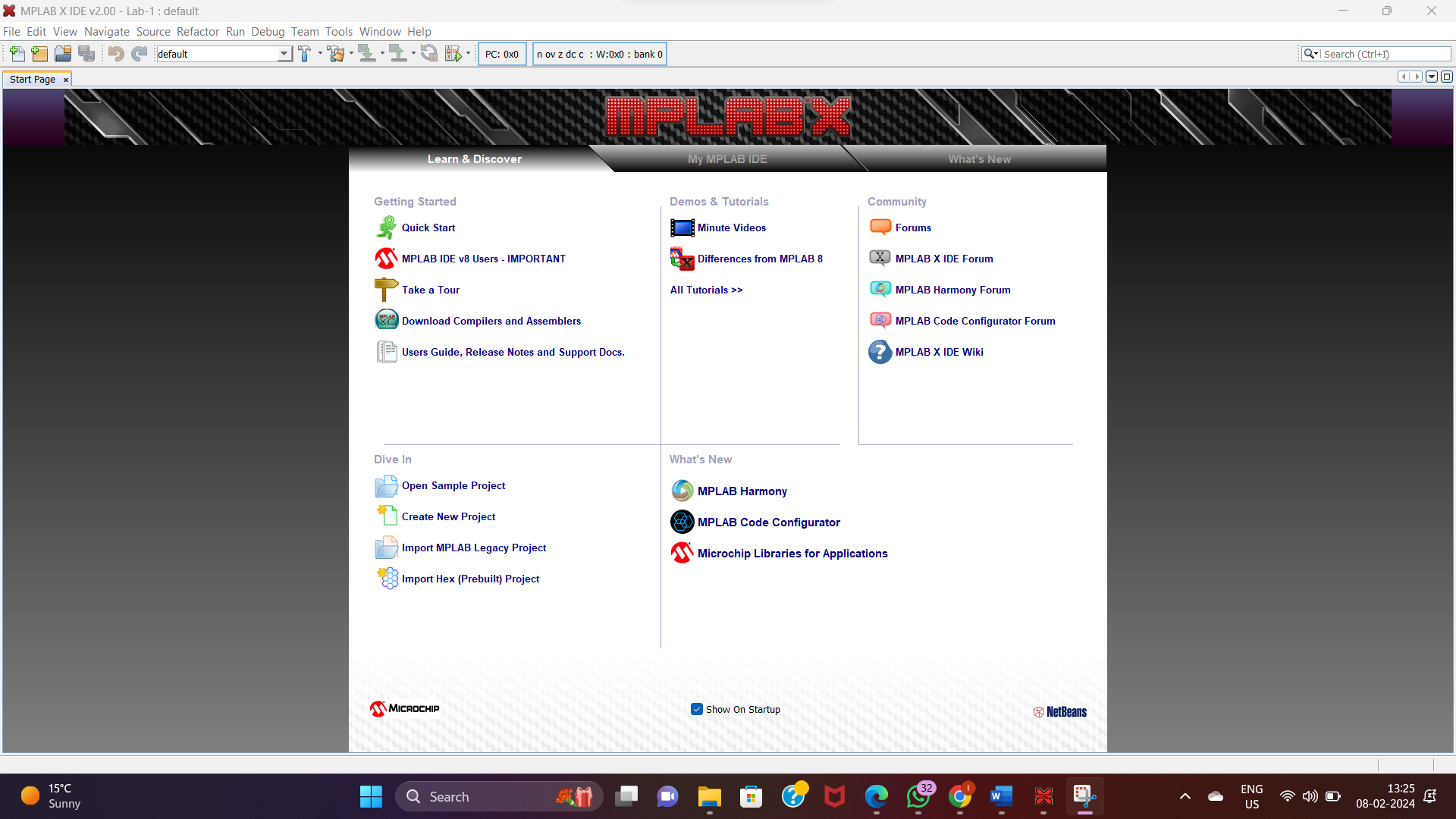
**LAB ASSIGNMENT - 1**

**AIM: I**ntroduction to MP lab and creating a file PIC microcontroller**.**

**Software Used:** MPLABX IDE v2.00

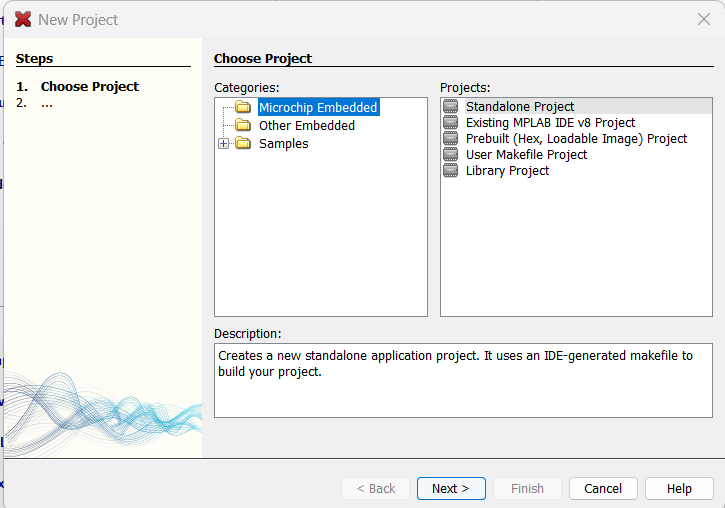
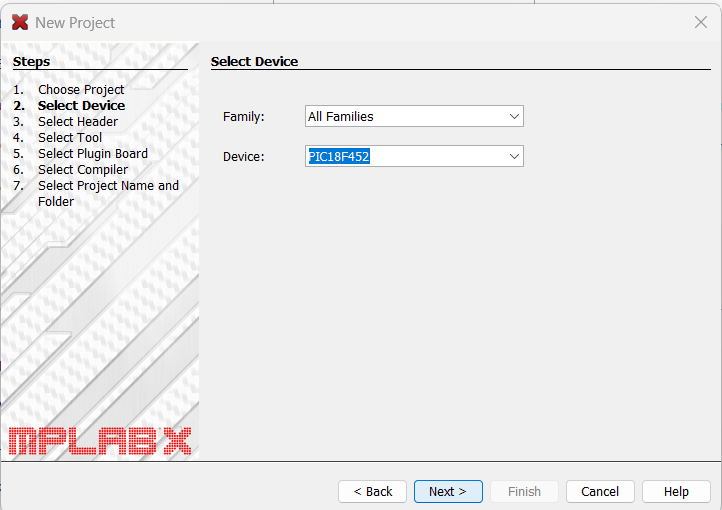
**MP-Lab:**

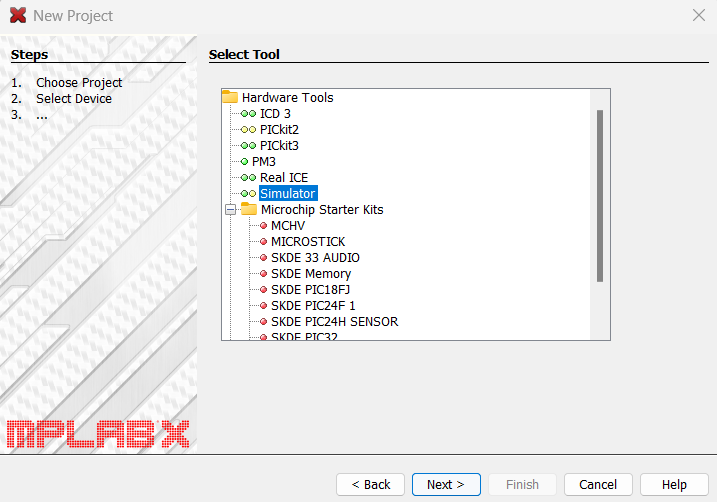
MP-lab is a public domain software and is developed for the embedded applications on PIC and microcontroller and this is developed by multinational company named as Microchip technology. MP lab supports the coding, debugging, and programming of microchips, microcontrollers of 8bit, 16bits and DS-PIC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



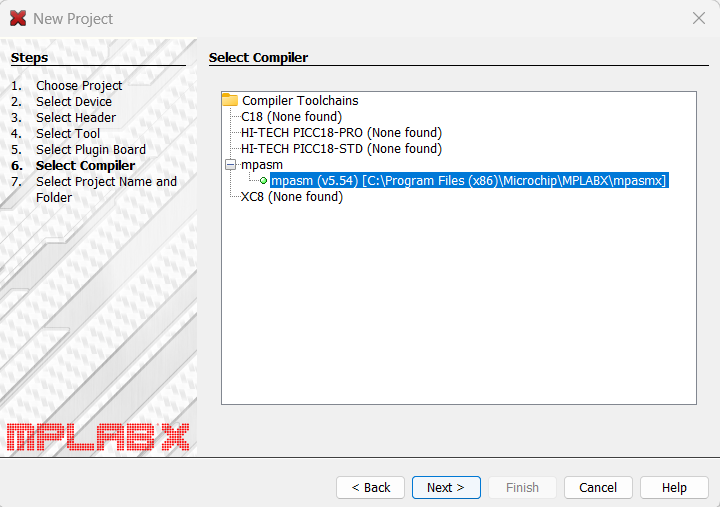
**MPLAB MAIN SCREEN**

**PROCEDURE:**

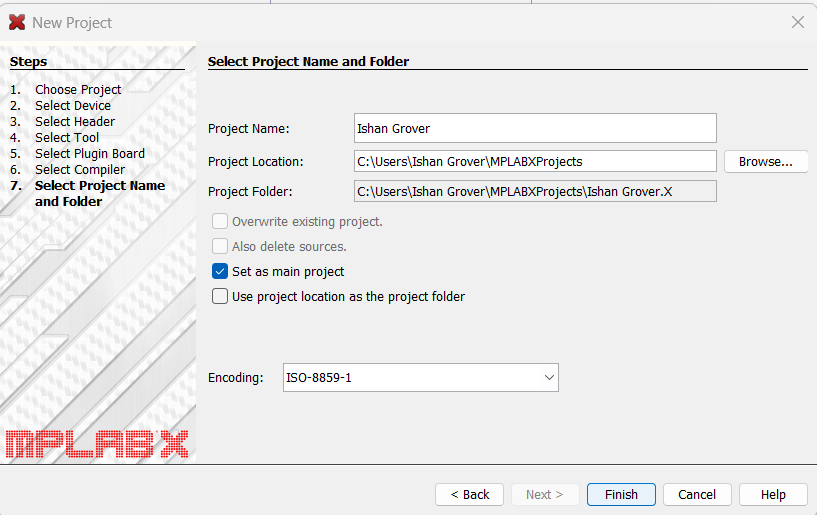
* First, we must open the MP-Lab Software and to start the project we must go on the left top corner where the file bar is written and select the New-Project. The pop-up screen will be shown as follow: 
* Then we must select the Standalone Project.
* After that under Family, user must choose all families and under device the user must choose PIC-18F-452. Here PIC stands for Peripheral Interface Controller and the 18 stands for the generation.
* Now after selecting device, we will move towards hardware tools where we select Simulator.



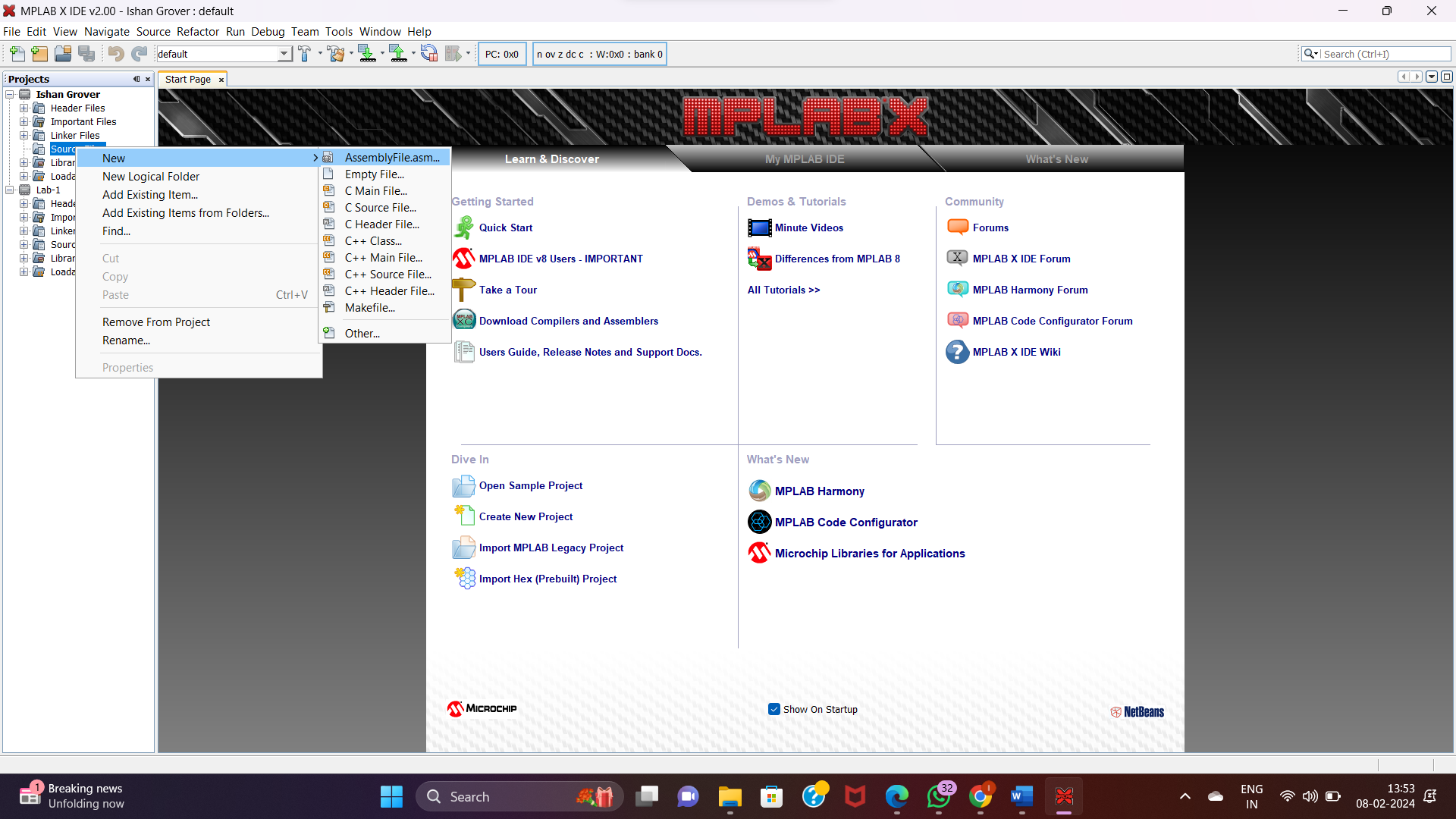
* Now we move on to choosing the compiler, in which we select the mpasm(v5.54) option.



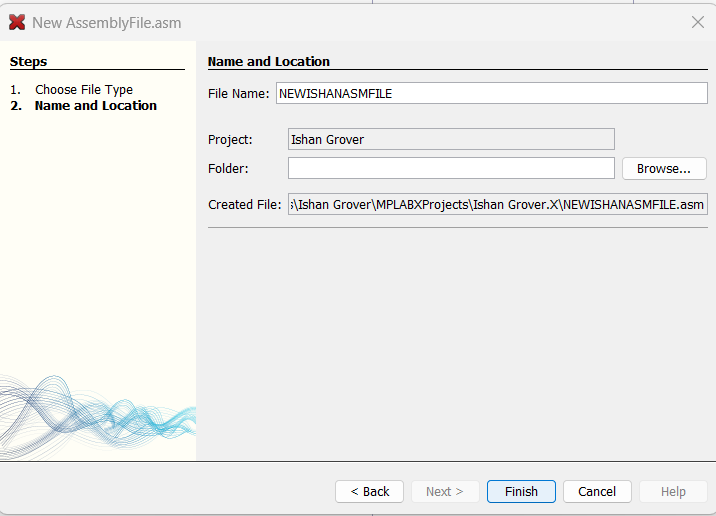
* Now we must name the project. Here I’ve name it as Ishan Grover but you can name it whatever you like. After naming the project the project is ready to go.



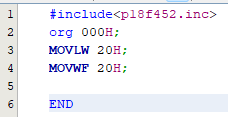
* After the project has been created, we will click on project and some new sub files will be shown. From those sub files we must select source files. After that right click on Source files and select New, from New select AssemblyFile.asm.



* Now name the new .asm file. Here I’ve named the file as NEWISHANASMFILE. You can choose any name and proceed.



**TEMPLATE PROGRAM:**



* Include<p18f452.ic>
* Start the program with org 000H;
* MOVLW loads the 20H value in WR.
* MOVWF moves the value from WR and stores the result in 20H register.

**Result:**

MPLAB software was installed and the basic template program was executed.

**LAB ASSIGNMENT - 2**

**AIM: -** Write a program to Add/Sub two 8-bit number in the MP lab.

**Software Used:** MPLABX IDE v2.00

**Theory:**

* The # include directive tells the PIC Assembler to use the libraries associated with the specific chip for which the program is compiled.
* The ORG (origin) directive indicate the starting position in the memory where the program block is to be stored.
* The MOVLW (Move the literal value to working register) instruction tells the processor to move the 8-bit literal value with the content of working register.
* The ADDWF instruction is a fundamental command in assembly language used in microcontroller programming, particularly in architectures like PIC (Peripheral Interface Controller). This instruction performs addition between the contents of a specified file register and the Working Register (WREG) within the microcontroller's CPU. The result of the addition is stored back either in the specified file register or in the WREG, depending on the addressing mode used.
* SUBWF command is an instruction used in PIC microcontroller assembly language programming. It stands for "Subtract W from f" where "f" represents a register or memory location and "W" is the Working register. This command performs a subtraction operation between the value in the specified register/memory location and the contents of the Working register (W register), and then stores the result back into the specified register/memory location.
* The END directive indicates to the assembler the end of the source file meaning that anything after the END directive in the source code is ignored by the assembler.

**Code:**

Addition of two numbers:

#include<p18f452.inc>

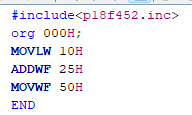
org 000H;

MOVLW 10H

ADDWF 25H

MOVWF 50H

END



Subtraction of two numbers:

#include<p18f452.inc>

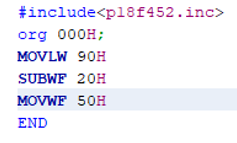
org 000H;

MOVLW 90H

SUBWF 20H

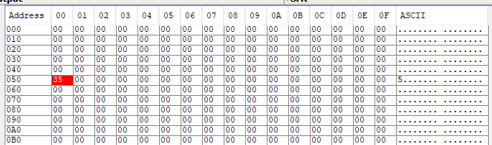
MOVWF 50H

END

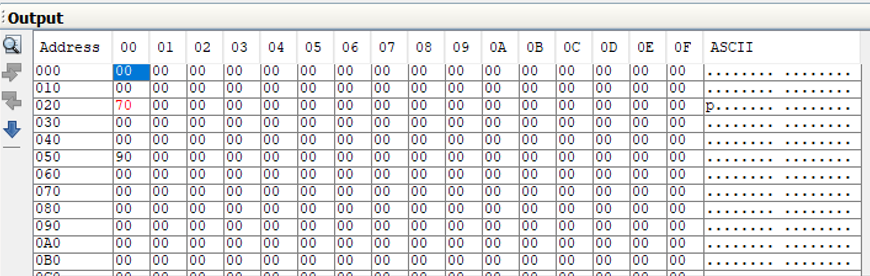


**Output:**

Addition of two numbers:

****

Subtraction of two numbers:



**Result:**

Addition and Subtraction of two 8-bit numbers were carried out.

**LAB ASSIGNMENT - 3**

**AIM: -** Write a program to Add your roll no (last 2 digit) by 8 times and store value in file register.

**Software Used:** MPLABX IDE v2.00

**Theory:**

* The # include directive tells the PIC Assembler to use the libraries associated with the specific chip for which the program is compiled.
* The ORG (origin) directive indicate the starting position in the memory where the program block is to be stored.
* The MOVLW (Move the literal value to working register) instruction tells the processor to move the 8-bit literal value with the content of working register.
* The ADDLW (Add the literal value to working register) instruction tells the processor to add the literal value with the content of working register.
* The MOVWF (Move contents of wreg to specified file register in the selected bank if a=1)
* The END directive indicates to the assembler the end of the source file meaning that anything after the END directive in the source code is ignored by the assembler.

**Code:**

#include<p18f452.inc>

org 000H;

MOVLW 28H

ADDLW 01H

ADDLW 01H

ADDLW 01H

ADDLW 01H

ADDLW 01H

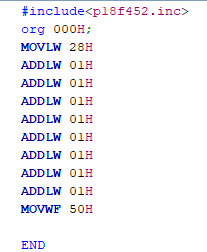
ADDLW 01H

ADDLW 01H

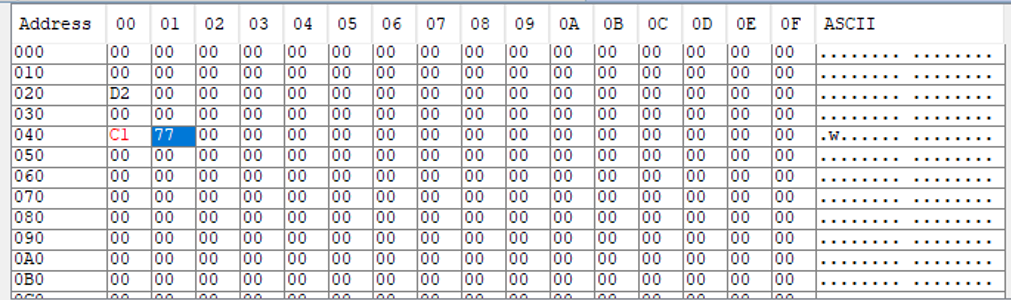
ADDLW 01H

MOVWF 50H

END



**Output:**

****

**Result:**

The last 2 digits of my roll number was added 8 times and stored in file register.

**LAB ASSIGNMENT - 4**

**AIM: -** Write a program to add 2, 8-bit number one stored in File Location 20H and other in working register. After addition store the result in File Location 40H (Upper Byte) and 41H (Lower Byte).

**Software Used:** MPLABX IDE v2.00

**Theory:**

* The # include directive tells the PIC Assembler to use the libraries associated with the specific chip for which the program is compiled.
* The ORG (origin) directive indicate the starting position in the memory where the program block is to be stored.
* The MOVLW (Move the literal value to working register) instruction tells the processor to move the 8-bit literal value with the content of working register.
* The MOVWF (Move contents of wreg to specified file register in the selected bank if a=1).
* The ADDWF instruction is a fundamental command in assembly language used in microcontroller programming, particularly in architectures like PIC (Peripheral Interface Controller). This instruction performs addition between the contents of a specified file register and the Working Register (WREG) within the microcontroller's CPU. The result of the addition is stored back either in the specified file register or in the WREG, depending on the addressing mode used.
* BNC instruction is used in PIC microcontroller assembly language programming. It stands for "Branch if Not Carry Set". This instruction is a conditional branch instruction, meaning it alters the flow of program execution based on the status of the Carry flag.
* INCF (Increment File Register) instruction increments the file register.
* BRA (Branch Unconditionally) instruction adds 2’s complement number to program counter
* The END directive indicates to the assembler the end of the source file meaning that anything after the END directive in the source code is ignored by the assembler.

**Code:**

#include<p18f452.inc>

ORG 0000H

MOVLW 0D2H

MOVWF 20H

MOVLW 0A5H

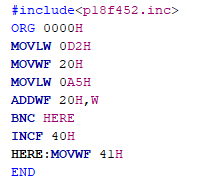
ADDWF 20H,W

BNC HERE

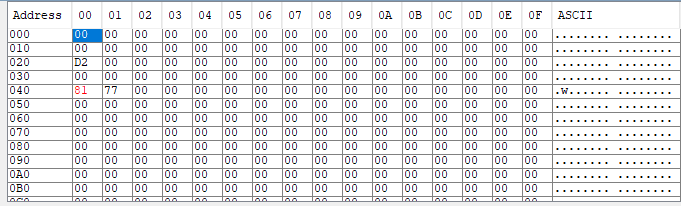
INCF 40H

HERE:MOVWF 41H

END



**Output:**



**Result:**

The 2 numbers are added and the result is stored in file location 40H (upper byte) and 41H (Lower Byte).

**LAB ASSIGNMENT - 5**

**AIM:** Write a program to add two 8-bit numbers. One is stored in file register and other in working register.

**Software Used:** MPLABX IDE v2.00

**Theory:**

* The # include directive tells the PIC Assembler to use the libraries associated with the specific chip for which the program is compiled.
* The ORG (origin) directive indicate the starting position in the memory where the program block is to be stored.
* CLRF is a straightforward instruction in microcontroller programming used to clear the contents of a file register, providing a simple and efficient way to initialize variables and ensure predictable behaviour in embedded systems.
* The MOVLW (Move the literal value to working register) instruction tells the processor to move the 8-bit literal value with the content of working register.
* The MOVWF (Move contents of wreg to specified file register in the selected bank if a=1).
* The ADDWF instruction is a fundamental command in assembly language used in microcontroller programming, particularly in architectures like PIC (Peripheral Interface Controller). This instruction performs addition between the contents of a specified file register and the Working Register (WREG) within the microcontroller's CPU. The result of the addition is stored back either in the specified file register or in the WREG, depending on the addressing mode used.
* BC (Branch if Carry) instruction reads the carry flag in status register, if its set then branching is done.
* INCF (Increment File Register) instruction increments the file register.
* BRA (Branch Unconditionally) instruction adds 2’s complement number to program counter
* NOP (No Operation)
* The END directive indicates to the assembler the end of the source file meaning that anything after the END directive in the source code is ignored by the assembler.

**Code:**

#include<P18F452.INC>

ORG 0010H

CLRF WREG

MOVLW 0XFE

MOVWF 0X20

MOVLW 0XEF

ADDWF 0X20,0

MOVWF 0X41

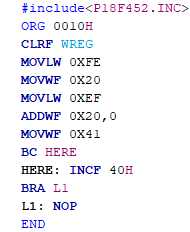
BC HERE

HERE: INCF 40H

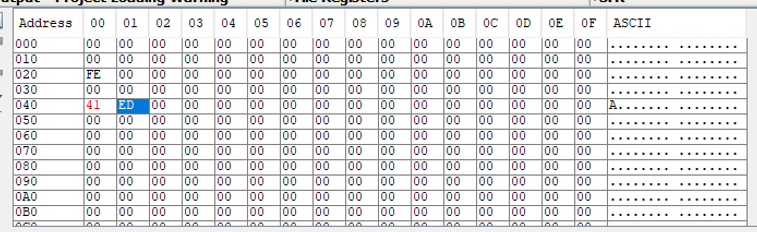
BRA L1

L1: NOP

END



**Output:**

****

**Result:**

Addition of two 8 bit numbers was done where one was stored in file register and other in working register.

**LAB ASSIGNMENT - 6**

**AIM:** Write a program to add 5 hexadecimal Number (8-bit) stored at memory location 20H onward, consecutively. Check the status of the carry flag and store the result in 40H (Upper byte) and 41H (lower byte).

**Software Used:** MPLABX IDE v2.00

**Theory:**

* The # include directive tells the PIC Assembler to use the libraries associated with the specific chip for which the program is compiled.
* The ORG (origin) directive indicate the starting position in the memory where the program block is to be stored.
* CLRF is a straightforward instruction in microcontroller programming used to clear the contents of a file register, providing a simple and efficient way to initialize variables and ensure predictable behaviour in embedded systems.
* The MOVLW (Move the literal value to working register) instruction tells the processor to move the 8-bit literal value with the content of working register.
* The MOVWF (Move contents of wreg to specified file register in the selected bank if a=1).
* The ADDWF instruction is a fundamental command in assembly language used in microcontroller programming, particularly in architectures like PIC (Peripheral Interface Controller). This instruction performs addition between the contents of a specified file register and the Working Register (WREG) within the microcontroller's CPU. The result of the addition is stored back either in the specified file register or in the WREG, depending on the addressing mode used.
* LFSR (Move literal to FSR): The 12-bit literal ‘k’ is loaded into the file select register pointed to by ‘f’. This command is used for indirect addressing. It copies a value from one place to multiple memory locations. Here using 0, we have selected the file select register 0, although there are three of them i.e. FSR0, FSR1, FSR2. The value 20 in decimal is loaded into the FSR.
* BTFSC (Bit tert File, Skip if Clear): If bit ‘b’ in register F is zero then the next instruction is skipped. If bit B is zero then the next instruction fetched during the current instruction execution is discarded and NOP is executed instead making this a 2-cycle instruction. If a is zero the Access Bank will be selected overriding the BSR value if a is one then the bank will be selected as per the BSR value.
* DECFSZ (Decrement F, Skip if Zero): The contents of a register F are decremented. If D is zero the result is placed in W. If D is one, the results are placed in register F. If the result is 0, the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it a two-cycle instruction. If a is 0 the access bank will be selected, overriding the BSR value. If a=1 then the bank will be selected as per the BSR value.
* GOTO (Go to address): This command allows the unconditional branch anywhere within the entire range of 2Mbyte memory. The 20-bit value ‘k’ is loaded into PC<20:1>. This is a two-cycle instruction.
* BNC instruction is used in PIC microcontroller assembly language programming. It stands for "Branch if Not Carry Set". This instruction is a conditional branch instruction, meaning it alters the flow of program execution based on the status of the Carry flag.
* INCF (Increment File Register) instruction increments the file register.
* The END directive indicates to the assembler the end of the source file meaning that anything after the END directive in the source code is ignored by the assembler.

**Code:**

#include<p18f452.inc>

ORG 0000H

MOVLW 0X05

MOVWF 0X10, 1

LFSR 0, 0X20

CLRF WREG

H2: ADDWF POSTINC0, 0

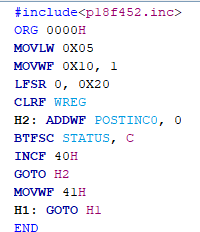
BTFSC STATUS, C

INCF 40H

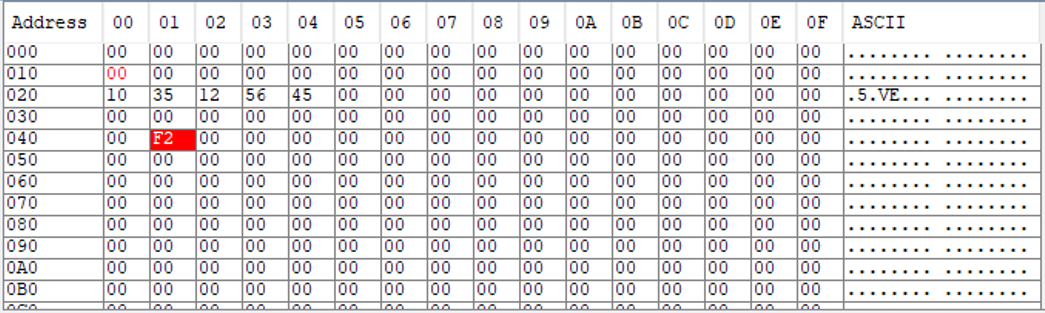
GOTO H2

MOVWF 41H

H1: GOTO H1

END

**Output:**

****

**Result:**

5 hexadecimal numbers were added and the result was stored in 40H (Upper Byte) and 41H (Lower Byte).

**LAB ASSIGNMENT - 7**

**AIM:** Write a program to perform Boolean operation between two number stored in location 00 and 01. Also store the result in 20h onward.

**Software Used:** MPLABX IDE v2.00

**Theory:**

* The # include directive tells the PIC Assembler to use the libraries associated with the specific chip for which the program is compiled.
* The ORG (origin) directive indicate the starting position in the memory where the program block is to be stored.
* CLRF is a straightforward instruction in microcontroller programming used to clear the contents of a file register, providing a simple and efficient way to initialize variables and ensure predictable behaviour in embedded systems.
* The MOVLW (Move the literal value to working register) instruction tells the processor to move the 8-bit literal value with the content of working register.
* The MOVWF (Move contents of wreg to specified file register in the selected bank if a=1).
* ANDWF instruction is used in PIC microcontroller assembly language programming. It stands for "AND W with f" where "f" represents a register or memory location and "W" is the Working register. This instruction performs a bitwise AND operation between the value in the specified register/memory location and the contents of the Working register (W register), and then stores the result back into the specified register/memory location.
* The IORWF instruction performs an inclusive OR operation between the contents of a register or memory location (f) and the contents of the Working register (WREG). The result is then stored either back into the specified register/memory location (f) or into the Working register (WREG), depending on the destination (d) bit.
* The XORWF instruction performs an exclusive OR operation between the contents of a register or memory location (f) and the contents of the Working register (WREG). The result is then stored either back into the specified register/memory location (f) or into the Working register (WREG), depending on the destination (d) bit.
* The END directive indicates to the assembler the end of the source file meaning that anything after the END directive in the source code is ignored by the assembler.

**Code:**

ANDWF:

#include<p18f452.inc>

ORG 0000H

CLRF WREG

MOVLW 60

MOVWF 00H

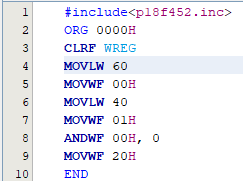
MOVLW 40

MOVWF 01H

ANDWF 00H, 0

MOVWF 20H

END

  
IORWF:

#include <p18f452.inc>

ORG 0000H

CLRF WREG ; Clear WREG

MOVLW 60 ; Load value 60 into WREG

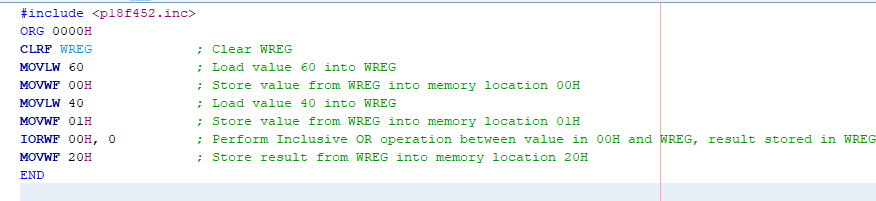
MOVWF 00H ; Store value from WREG into memory location 00H

MOVLW 40 ; Load value 40 into WREG

MOVWF 01H ; Store value from WREG into memory location 01H

IORWF 00H, 0 ; Perform Inclusive OR operation between value in 00H and WREG, result stored in WREG

MOVWF 20H ; Store result from WREG into memory location 20H

END  


XORWF:

#include <p18f452.inc>

ORG 0000H

CLRF WREG ; Clear WREG

MOVLW 60 ; Load value 60 into WREG

MOVWF 00H ; Store value from WREG into memory location 00H

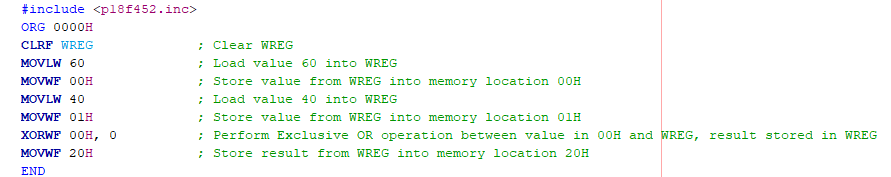
MOVLW 40 ; Load value 40 into WREG

MOVWF 01H ; Store value from WREG into memory location 01H

XORWF 00H, 0 ; Perform Exclusive OR operation between value in 00H and WREG, result stored in WREG

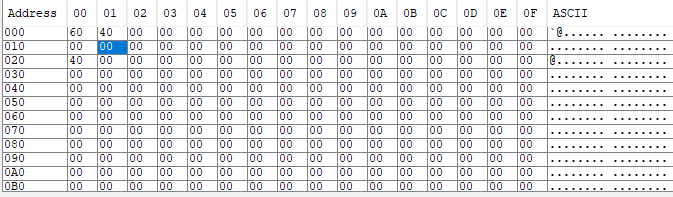
MOVWF 20H ; Store result from WREG into memory location 20H

END

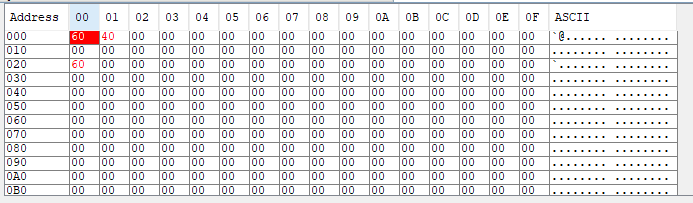


**Output:**

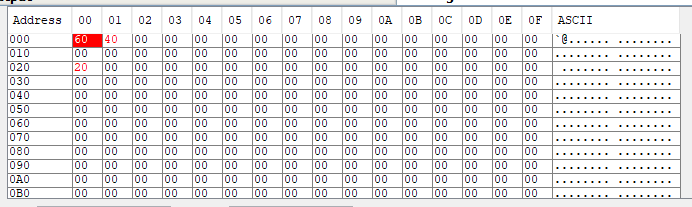
ANDWF:



IORWF:

****

XORWF:

****

**Result:**

Boolean operation between two number stored in location 00 and 01 was performed and the result was stored from 20H onwards.

**LAB ASSIGNMENT - 8**

**AIM:** Write a program to compare two number stored in the location 00 and 01.

**Software Used:** MPLABX IDE v2.00

**Theory:**

* The # include directive tells the PIC Assembler to use the libraries associated with the specific chip for which the program is compiled.
* The ORG (origin) directive indicate the starting position in the memory where the program block is to be stored.
* CLRF is a straightforward instruction in microcontroller programming used to clear the contents of a file register, providing a simple and efficient way to initialize variables and ensure predictable behaviour in embedded systems.
* The MOVLW (Move the literal value to working register) instruction tells the processor to move the 8-bit literal value with the content of working register.
* The MOVWF (Move contents of wreg to specified file register in the selected bank if a=1).
* CPFSGT instruction stands for "Compare File with Signed Greater Than". It is used to compare the value in the WREG (Working register) with the value in the specified register or memory location. This comparison is done considering the values as signed numbers.
* BRA (Branch Unconditionally) instruction adds 2’s complement number to program counter
* The END directive indicates to the assembler the end of the source file meaning that anything after the END directive in the source code is ignored by the assembler.

**Code:**

INCLUDE <P18F452.INC>

ORG 0x00

START:

; Load value 10 into memory location 00

MOVLW 0x0A ; Load value 10 into WREG

MOVWF 0x20 ; Store value from WREG into memory location 00

; Load value 20 into memory location 01

MOVLW 0x14 ; Load value 20 into WREG

MOVWF 0x21 ; Store value from WREG into memory location 01

; Compare the numbers

MOVLW 0x00 ; Load WREG with 0 for comparison

CPFSGT 0x20, 0 ; Compare WREG with data in location 00

BRA NEXT ; If the number in 01 is greater than the number in 00, skip the next instruction

; Code to execute if the number in 00 is greater than or equal to the number in 01

; Add your code here, for example:

; Display a message or perform some action

NEXT:

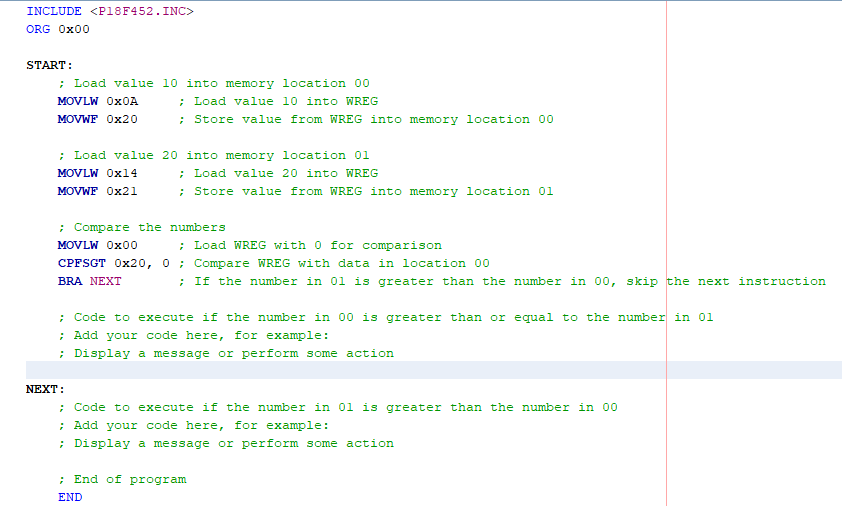
; Code to execute if the number in 01 is greater than the number in 00

; Add your code here, for example:

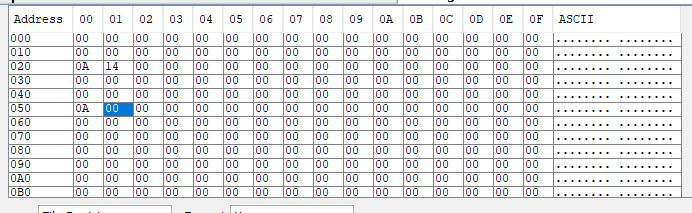
; Display a message or perform some action

; End of program

END



**Output:**

****

**Result:**

Comparison of two numbers stored in 00 and 01 was done.

**LAB ASSIGNMENT – 9**

**AIM:** Write a program to add a number 20 times in working register using Loop.

**Software Used:** MPLABX IDE v2.00

**Theory:**

* The # include directive tells the PIC Assembler to use the libraries associated with the specific chip for which the program is compiled.
* The ORG (origin) directive indicate the starting position in the memory where the program block is to be stored.
* The MOVLW (Move the literal value to working register) instruction tells the processor to move the 8-bit literal value with the content of working register.
* The MOVWF (Move contents of wreg to specified file register in the selected bank if a=1).
* INCF (Increment File Register) instruction increments the file register.
* The ADDWF instruction is a fundamental command in assembly language used in microcontroller programming, particularly in architectures like PIC (Peripheral Interface Controller). This instruction performs addition between the contents of a specified file register and the Working Register (WREG) within the microcontroller's CPU. The result of the addition is stored back either in the specified file register or in the WREG, depending on the addressing mode used.
* DECFSZ (Decrement F, Skip if Zero): The contents of a register F are decremented. If D is zero the result is placed in W. If D is one, the results are placed in register F. If the result is 0, the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it a two-cycle instruction. If a is 0 the access bank will be selected, overriding the BSR value. If a=1 then the bank will be selected as per the BSR value.
* BNC instruction is used in PIC microcontroller assembly language programming. It stands for "Branch if Not Carry Set". This instruction is a conditional branch instruction, meaning it alters the flow of program execution based on the status of the Carry flag.
* GOTO (Go to address): This command allows the unconditional branch anywhere within the entire range of 2Mbyte memory. The 20-bit value ‘k’ is loaded into PC<20:1>. This is a two-cycle instruction.
* The END directive indicates to the assembler the end of the source file meaning that anything after the END directive in the source code is ignored by the assembler.

**Code:**

#include<p18f452.inc>

ORG 0x0000

MOVLW D'20'

MOVWF 0x05

MOVLW 0xC8

MOVWF 0x00

MOVLW 0x00

LOOP:

ADDWF 0x00,F

BNC THERE

INCF 0x10,F

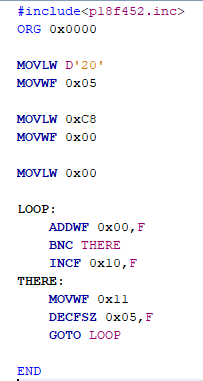
THERE:

MOVWF 0x11

DECFSZ 0x05,F

GOTO LOOP

END



**Output:**

