EXPERIMENT NO- 5

AIM: WAP to perform addition, subtraction, multiplication and division using assembly language in C++ (Mixed language programming)

Resource Required: P-IV and above RAM 128MB, Dot Matrix Printer, Emu 8086, MASM 611/ TASM, Turbo C/C++, Printer, Printout Stationary.

THEORY:

Assembly is useful for time-critical or real-time processes, because unlike with high-level languages, there is no ambiguity about how the code will be compiled**.**

Inline Assembly

One of the most common methods for using assembly code fragments in a C programming project is to use a technique called **inline assembly**. Inline assembly is invoked in different compilers in different ways. Also, the assembly language syntax used in the inline assembly depends entirely on the assembly engine used by the C compiler. Microsoft C++, for instance, only accepts inline assembly commands in MASM syntax, while GNU GCC only accepts inline assembly in GAS syntax(also known as AT&T syntax) .

- Microsoft C Compiler
- GNU GCC Compiler
- Borland C Compiler

a. Linked assembly

When an assembly source file is assembled by an assembler, and a C source file is compiled by a C compiler, those two **object files** can be linked together by a **linker** to form the final executable. The only disadvantages of mixing assembly and C in this way are that

a) both the assembler and the compiler need to be run, and

b) those files need to be manually linked together by the programmer.

These extra steps are comparatively easy, although it does mean that the programmer needs to learn the command-line syntax of the compiler, the assembler, and the linker.

- **Inline Assembly vs. linked assembly**

  Advantages of inline assembly:

  Short assembly routines can be embedded directly in C function in a C code file. The mixed-language file then can be completely compiled with a single command to the C compiler (as opposed to compiling the assembly code with an assembler, compiling the C code with the C Compiler, and then linking them together). This method is fast and easy.

  Advantages of linked assembly:

  If a new microprocessor is selected, all the assembly commands are isolated in a ".asm" file. The programmer can update just that one file -- there is no need to change any of the ".c" files (if they are portably written).

CONCLUSION: We have successfully performed arithmetic operation using mixed language programming in C/C++.

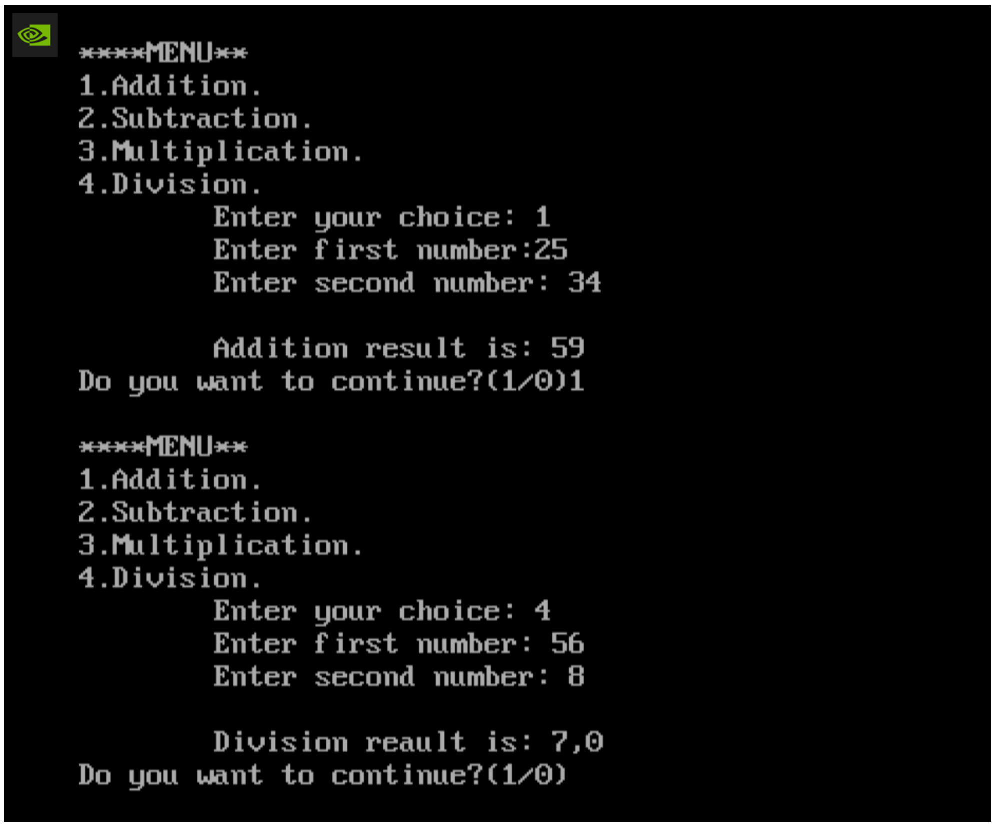
Code:

```
#include<stdio.h>

#include<conio.h>

void main()

{

int a, b, c, q, r;

int ch, ans;

clrscr();

do

{
```

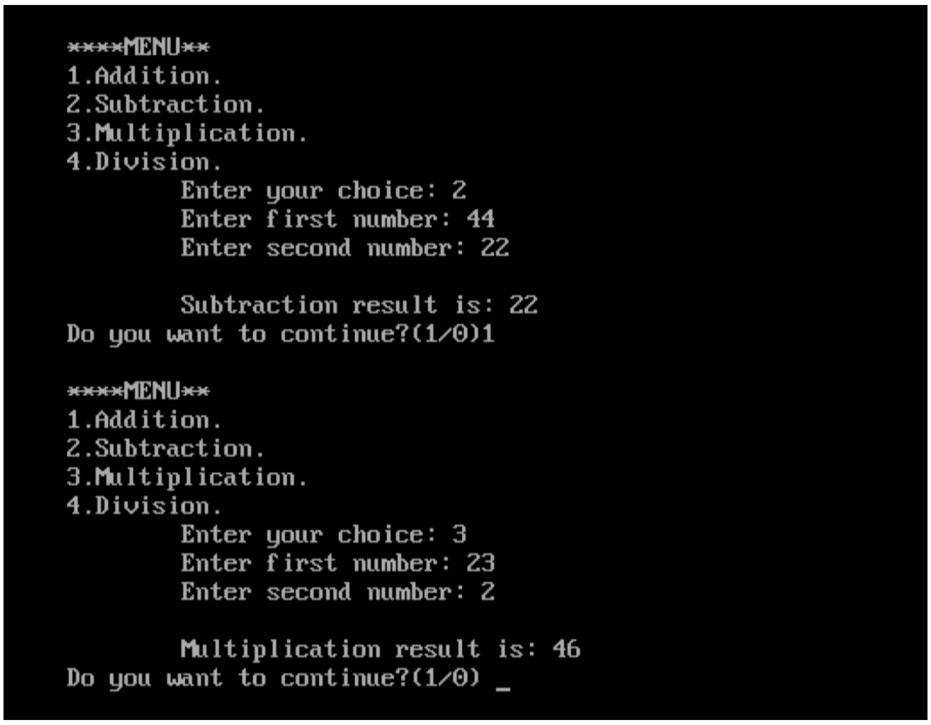
```
printf("\n****MENU**");
printf("\n1.Addition.\n2.Subtraction.\n3.Multiplication.\n4.Division.\n");
printf("\tEnter your choice: ");
scanf("%d", &ch);
printf("\tEnter first number: ");
scanf("%d", &a);
printf("\tEnter second number: ");
scanf("%d", &b);
switch(ch)
{
        case 1:
                asm mov ax, a;
                asm mov bx, b;
                asm add ax, bx;
                asm mov c, ax;
                printf("\n\tAddition result is: %d", c);
                break;
        case 2:
                asm mov ax, a;
                asm mov bx, b;
                asm sub ax, bx;
                asm mov c, ax;
                printf("\n\tSubtraction result is: %d", c);
                break;

        case 3:

                asm mov ax, a;

                asm mov bx, b;

                asm mul bx;

                asm mov c, ax;

                printf("\n\tMultiplication result is: %d", c);

                break;

        case 4:

                asm mov ax, a;

                asm mov bx ,b;

                asm div bx;

                asm mov q, ax;

                asm mov r, dx;

                printf("\n\tDivision reault is: %d,%d", q, r);

                break;

        default:

                printf("\nINVALID INPUT!!");

}

printf("\nDo you want to continue?(1/0)");

scanf("%d", &ans);

}while(ans == 1);

getch();
```

}

Output:



```
****MENU**
1.Addition.
2.Subtraction.
3.Multiplication.
4.Division.
        Enter your choice: 1
        Enter first number:25
        Enter second number: 34

        Addition result is: 59
Do you want to continue?(1/0)1

****MENU**
1.Addition.
2.Subtraction.
3.Multiplication.
4.Division.
        Enter your choice: 4
        Enter first number: 56
        Enter second number: 8

        Division reault is: 7,0
Do you want to continue?(1/0)
```



```
****MENU**
1.Addition.
2.Subtraction.
3.Multiplication.
4.Division.
        Enter your choice: 2
        Enter first number: 44
        Enter second number: 22

        Subtraction result is: 22
Do you want to continue?(1/0)1

****MENU**
1.Addition.
2.Subtraction.
3.Multiplication.
4.Division.
        Enter your choice: 3
        Enter first number: 23
        Enter second number: 2

        Multiplication result is: 46
Do you want to continue?(1/0) _
```