Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

AY: 2024-25

Class:	**SE**	**Semester:**	**IV**
Course Code:	**CSL404**	**Course Name:**	**Microprocessor Lab**

Name of Student:	**Shravani Sandeep Raut**
Roll No. :	**48**
Experiment No.:	**8**
Title of the Experiment:	**Mixed language program for adding two numbers**
Date of Performance:	**20/02/2025**
Date of Submission:	**06/03/2025**

Evaluation

Performance Indicator	Max. Marks	Marks Obtained
Performance	5	
Understanding	5	
Journal work and timely submission	10	
Total	20	

Performance Indicator	Exceed Expectations (EE)	Meet Expectations (ME)	Below Expectations (BE)
Performance	4-5	2-3	1
Understanding	4-5	2-3	1
Journal work and timely submission	8-10	5-8	1-4

Checked by

Name of Faculty : Ms. Sweety Patil

Signature :

Date:

Aim: Mixed language program for adding two numbers.

Theory:

C generates an object code that is extremely fast and compact but it is not as fast as the object code generated by a good programmer using assembly language. The time needed to write a program in assembly language is much more than the time taken in higher level languages like C.

However, there are special cases wher a function is coded in assembly language to reduce the execution time.

Eg: The floating point math package must be loaded assembly language as it is used frequently and its execution speed will have great effect on the overall speed of the program that uses it.

There are also situations in which special hardware devices need exact timing and it is must to write a program in assembly language to meet this strict timing requirement. Certain instructions cannot be executed by a C program

Eg: There is no built in bit wise rotate operation in C. To efficiently perform this it is necessary to use assembly language routine.

Inspite of C being very powerful , routines must be written in assembly language to:

1.Increase the speed and efficiency of the routine

2.Perform machine specific function not available in Microsoft C or Turbo C.

3. Use third party routines

Combining C and assembly:

Built-In-Inline assembles is used to include assembly language routines in C program without any need for a specific assembler.

Such assembly language routines are called in-line assembly.

They are compiled right along with C routines rather than being assembled separately and then linked together using linker modules provided by the C compiler.

Turbo C has inline assembles.

In mixed language program, prefix the keyword asm for a function and write Assembly instruction in the curly braces in a C program

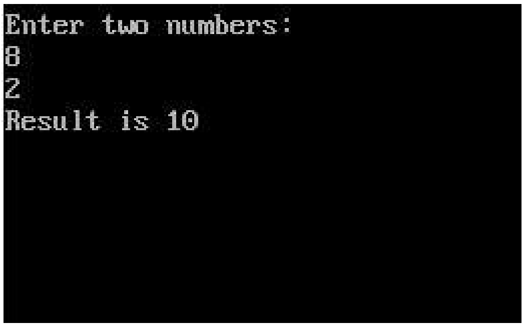
Code:

```
#include<stdio.h>

void main()
{
  int a,b,c;
  clrscr();
  printf("Enter two numbers:\n");
  scanf("%d %d", &a, &b);

  asm{
  mov ax,a
  mov bx,b
  add ax,bx
  mov c,ax
  }
  printf("Result is %d",c);
  getch();
}
```

Output:



```
Enter two numbers:
8
2
Result is 10
```

Conclusion:

In conclusion, a mixed-language program that adds two numbers demonstrates the ability to combine high-level language features with low-level assembly operations. By leveraging the strengths of both languages, the program can efficiently handle arithmetic operations while providing flexibility in system-level programming tasks. This approach highlights the power of using assembly for performance-critical operations in conjunction with high-level languages for ease of development.

1. Explain any 2 branch instructions.

   Branch instructions are used to alter the flow of control in a program, allowing execution to jump to a different part of the code. Here are two common branch instructions in assembly language:

   1. **JMP (Jump)**:
      Purpose: The `JMP` instruction unconditionally transfers control to a specified label or address.
      - **Syntax**: `JMP label`
      - **Example**:
        ```
        jmp skip      ; Jump to the 'skip' label
        ```
      - **Description**: The `JMP` instruction does not check any conditions; it always jumps to the given location.

2. **JE (Jump if Equal)**:

   - **Purpose**: The `JE` instruction causes a jump to a specified label if the Zero Flag (ZF) is set, indicating that two values were equal.
   - **Syntax**: `JE label`
   - **Example**:
     ```
     cmp ax, bx    ; Compare AX with BX

     je equal      ; Jump to 'equal' if AX == BX
     ```
   - **Description**: The `JE` instruction checks the Zero Flag after a comparison (e.g., `CMP`). If the values are equal, the Zero Flag is set, and the program jumps to the label.

2. Explain the syntax of loop.

   In assembly language, the `LOOP` instruction is used to create loops by automatically decrementing the `CX` (or `ECX` in 32-bit mode) register and jumping to a specified label if the value of `CX` is not zero. It is commonly used for repeating a set of instructions a specific number of times.

```
mov cx, 5      ; Initialize CX to 5 (number of iterations)

start_loop:

    ; Your code here (to be repeated)

    loop start_loop  ; Decrements CX and jumps if CX != 0
```