

TWO STAGE PIPELINE

EXP NO: 25

AIM: To write a C program to implement two stage pipelining.

PROCEDURE:

Step1:Start

Step 2: Initialize the counter variable to 1.

Step 3: Prompt the user to enter the first number (a).

Step 4: Read the first number (a) from the user.

Step 5: Increment the counter by 1.

Step 6: Prompt the user to enter the second number (b).

Step 7: Read the second number (b) from the user.

Step 8: Increment the counter by 1.

Step 9: Display the menu of operations: Addition, Subtraction, Multiplication, and Division.

Step 10: Prompt the user to select an operation (choice).

Step 11: Read the choice from the user.

Step 12: Use a switch statement to perform the operation based on the selected choice:

12.1 For choice 1: Perform addition ($res = a + b$). Increment the counter by 1.

12.2 For choice 2: Perform subtraction ($res = a - b$). Increment the counter by 1.

12.3. For choice 3: Perform multiplication ($res = a * b$). Increment the counter by 1.

12.4 For choice 4: Perform division ($res = a / b$). Increment the counter by 1.

12.5. For any other choice: Display "Wrong input".

Step 13: Display the value of the counter (the number of cycles taken).

Step 14: Prompt the user to enter the number of instructions (ins).

Step 15: Read the number of instructions (ins) from the user.

Step 16: Calculate the performance measure by dividing the number of instructions (ins) by the counter and store it in the performance measure variable.

Step 17: Display the performance measure

Step 18: End

PROGRAM:

```
#include<stdio.h>
int main()
{
    int counter =1,a,b,choice,res,ins;
    printf("Enter number 1:");
    scanf("%d",&a);
    counter = counter+1;
    printf("Enter number 2:");
    scanf("%d",&b);
    counter = counter +1;
    printf("1-Addition:\n2-Subtraction:\n3-Multiplication:\n4-Division:");
    scanf("%d",&choice);
    switch(choice)
    {
        case 1: printf("Performing addition\n");
                res = a+b;
                counter = counter+1;
                break;
        case 2: printf("Performing subtraction\n");
                res = a-b;
                counter = counter+1;
```

```

        break;
    case 3: printf("Performing Multiplication\n");
        res = a*b;
        counter = counter+1;
        break;
    case 4: printf("Performing Division\n");
        res = a/b;
        counter = counter+1;
        break;
    default: printf("Wrong input");
        break;
}
printf("The cycle value is:%d\n",counter);
printf("Enter the number of instructions:");
scanf("%d",&ins);
int performance_measure = ins/counter;
printf("The performance measure is:%d\n",performance_measure);
return 0;

}

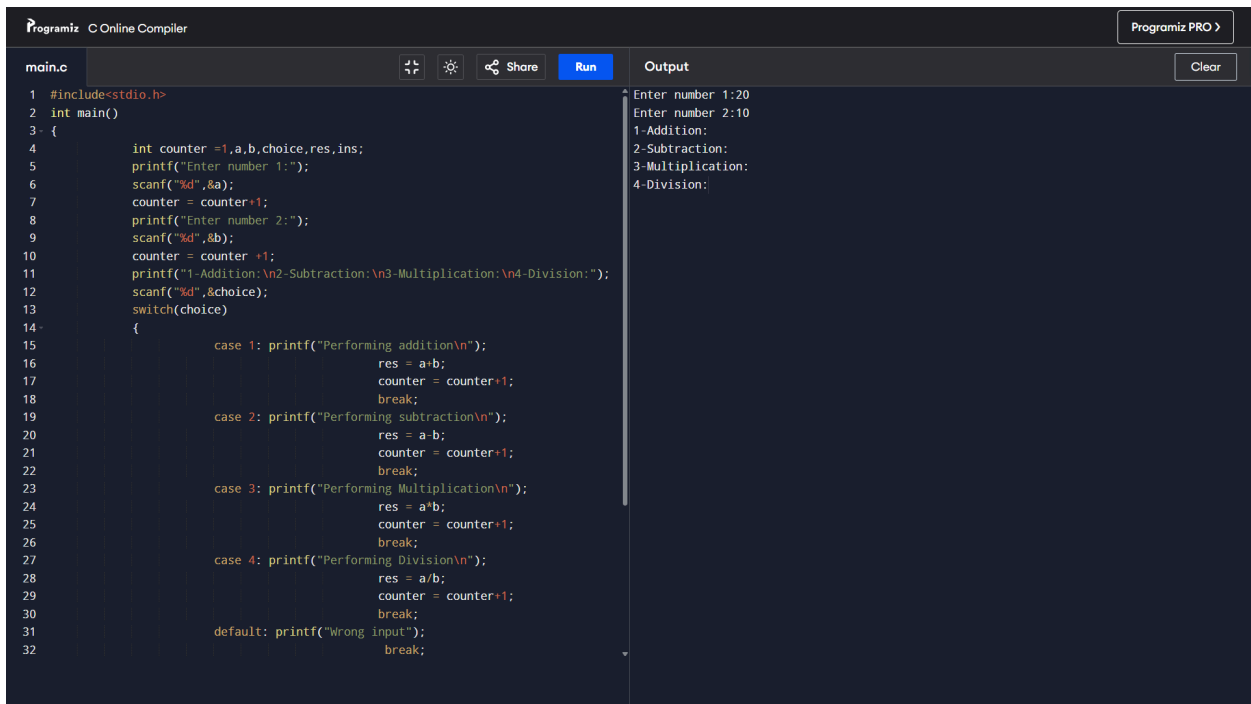
```

INPUT:

20

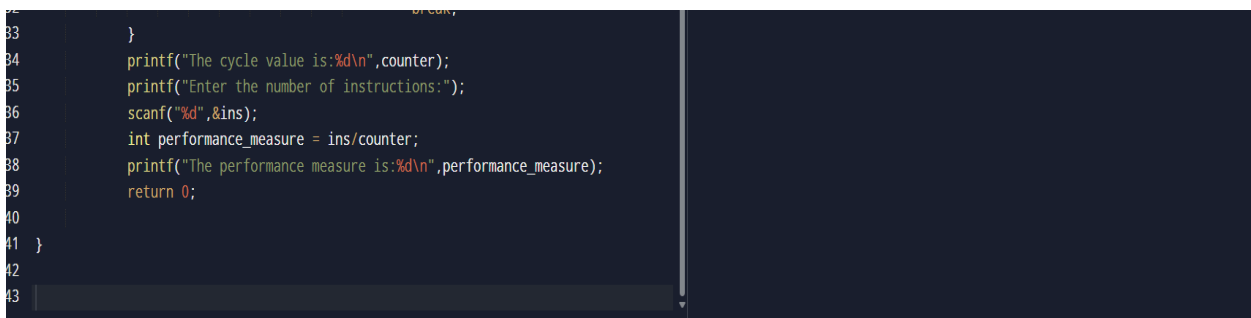
10

OUTPUT:



The screenshot shows the Programiz Online Compiler interface. The code editor on the left contains a C program that prompts the user for two numbers and a choice of operation (Addition, Subtraction, Multiplication, or Division). The output window on the right shows the program's execution with the inputs 20 and 10, and choice 1, resulting in the output '1-Addition:'. The code is as follows:

```
1 #include<stdio.h>
2 int main()
3 {
4     int counter =1,a,b,choice,res,ins;
5     printf("Enter number 1:");
6     scanf("%d",&a);
7     counter = counter+1;
8     printf("Enter number 2:");
9     scanf("%d",&b);
10    counter = counter +1;
11    printf("1-Addition:\n2-Subtraction:\n3-Multiplication:\n4-Division:");
12    scanf("%d",&choice);
13    switch(choice)
14    {
15        case 1: printf("Performing addition\n");
16                res = a+b;
17                counter = counter+1;
18                break;
19        case 2: printf("Performing subtraction\n");
20                res = a-b;
21                counter = counter+1;
22                break;
23        case 3: printf("Performing Multiplication\n");
24                res = a*b;
25                counter = counter+1;
26                break;
27        case 4: printf("Performing Division\n");
28                res = a/b;
29                counter = counter+1;
30                break;
31        default: printf("Wrong input");
32                break;
```



The continuation of the C code from the previous block, showing the calculation of the performance measure and the return statement.

```
33    }
34    printf("The cycle value is:%d\n",counter);
35    printf("Enter the number of instructions:");
36    scanf("%d",&ins);
37    int performance_measure = ins/counter;
38    printf("The performance measure is:%d\n",performance_measure);
39    return 0;
40
41 }
42
43
```

RESULT: Thus the program was executed successfully using DevC++.