

TWO STAGE PIPELINE

EXP NO: 37

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);
```

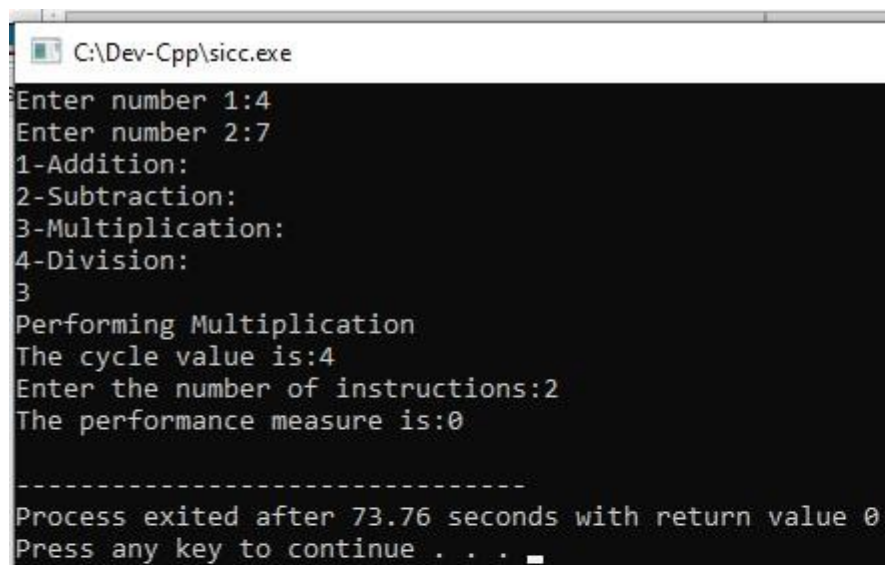
```
        int performance_measure =
ins/counter;
```

```
        printf("The performance measure
is:%d\n",performance_measure);
```

```
        return 0;
```

}

INPUT & OUTPUT:



```
C:\Dev-Cpp\sicc.exe
Enter number 1:4
Enter number 2:7
1-Addition:
2-Subtraction:
3-Multiplication:
4-Division:
3
Performing Multiplication
The cycle value is:4
Enter the number of instructions:2
The performance measure is:0

-----
Process exited after 73.76 seconds with return value 0
Press any key to continue . . .
```

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