

## 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;
```

```
    // Input first operand
```

```
    printf("Enter number 1: ");
```

```
    scanf("%d", &a);
```

```
    counter += 1;
```

```
    // Input second operand
```

```
    printf("Enter number 2: ");
```

```
    scanf("%d", &b);
```

```
    counter += 1;
```

```
// Input operation choice

printf("1-Addition:\n2-Subtraction:\n3-Multiplication:\n4-Division: ");
scanf("%d", &choice);

switch(choice) {
    case 1:
        printf("Performing addition\n");
        res = a + b;
        counter += 1;
        break;
    case 2:
        printf("Performing subtraction\n");
        res = a - b;
        counter += 1;
        break;
    case 3:
        printf("Performing multiplication\n");
        res = a * b;
        counter += 1;
        break;
    case 4:
        if (b == 0) {
            printf("Division by zero is not allowed\n");
        } else {
            printf("Performing division\n");
            res = a / b;
            counter += 1;
        }
        break;
    default:
        printf("Wrong input\n");
        // No counter increment for invalid input
        break;
}
```

```

}

printf("The cycle value is: %d\n", counter);

// Input number of instructions
printf("Enter the number of instructions: ");
scanf("%d", &ins);

// Calculate performance measure
int performance_measure = ins / counter;

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

return 0;
}

```

## INPUT & OUTPUT:

The screenshot displays the DevC++ IDE with a C program named `dummy.c` open. The code includes `<stdio.h>` and defines a `main` function. It prompts the user for two numbers and an operation choice. A `switch` statement handles the operation choice (1 for addition, 2 for subtraction, 3 for multiplication, 4 for division). The program calculates a performance measure as the number of instructions divided by the cycle value. The console output shows the program running with inputs 10 and 20, performing addition, subtraction, multiplication, and division, and finally exiting after 81.24 seconds.

```

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

```

```

Enter number 1: 10
Enter number 2: 20
1-Addition:
2-Subtraction:
3-Multiplication:
4-Division: 4
Performing division
The cycle value is: 4
Enter the number of instructions: 3
The performance measure is: 0

-----
Process exited after 81.24 seconds with return value 0
Press any key to continue . . .

```

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