

CPU PERFORMANCE

EXP NO: 32

AIM: To write a C program to implement CPU performance measures.

ALGORITHM:

Step 1: start

Step 2: Declare the necessary variables: cr

(clock rate), p (number of processors), p1 (a copy of the number of processors), i (loop variable), and cpu (array to store CPU times).

Step 3: Initialize the cpu array elements to 0.

Step 4: Prompt the user to enter the number of processors (p).

Step 5: Store the value of p in p1.

Step 6: Start a loop from 0 to p-1:

- a. Prompt the user to enter the cycles per instruction (cpi) for the current processor.
- b. Prompt the user to enter the clock rate (cr) in GHz for the current processor.
- c. Calculate the CPU time (ct) using the formula: $ct = 1000 * cpi / cr$.
- d. Display the CPU time for the current processor.
- e. Store the CPU time in the cpu array at index i.

Step 7: Set max as the first element of the cpu array.

Step 8: Start a loop from 0 to p1-1:

- a. If the CPU time at index i is less than or equal to max, update max to the current CPU time.

Step 9: Display the processor with the lowest execution time (max).

Step 10: Exit the program.

PROGRAM:

```
#include <stdio.h>

int main() {
    float cr, cpi, ct, max;

    int p, i;

    float cpu[5];

    // Initialize the CPU time array
    for (i = 0; i < 5; i++) {
        cpu[i] = 0;
    }

    // Get the number of processors
    printf("\nEnter the number of processors: ");
    scanf("%d", &p);

    // Get Cycles per Instruction (CPI) and clock rate for each processor
    for (i = 0; i < p; i++) {
```

```

printf("\nEnter the Cycles per Instruction of processor %d: ", i + 1);

scanf("%f", &cpi);


printf("\nEnter the clock rate in GHz for processor %d: ", i + 1);

scanf("%f", &cr);


// Calculate CPU time
ct = 1000 * cpi / cr;

printf("The CPU time for processor %d is: %f\n", i + 1, ct);


// Store the CPU time in the array
cpu[i] = ct;
}


// Find the processor with the lowest execution time
max = cpu[0];
for (i = 1; i < p; i++) {
    if (cpu[i] < max) {
        max = cpu[i];
    }
}


// Output the processor with the lowest execution time
printf("\nThe processor with the lowest execution time is: %f\n", max);


return 0;
}

```

INPUT & OUTPUT:

```
1 #include <stdio.h>
2
3 int main() {
4     float cr, cpi, ct, max;
5     int p, i;
6     float cpu[5];
7
8     // Initialize the CPU time array
9     for (i = 0; i < 5; i++) {
10         cpu[i] = 0;
11     }
12
13     // Get the number of processors
14     printf("\nEnter the number of processors: ");
15     scanf("%d", &p);
16
17     // Get Cycles per Instruction (CPI) and clock rate for each processor
18     for (i = 0; i < p; i++) {
19         printf("\nEnter the Cycles per Instruction of processor %d: ", i + 1);
20         scanf("%f", &cpi);
21
22         printf("\nEnter the clock rate in GHz for processor %d: ", i + 1);
23         scanf("%f", &cr);
24
25         // Calculate CPU time
26         ct = 1000 * cpi / cr;
27         printf("The CPU time for processor %d is: %f\n", i + 1, ct);
28
29         // Store the CPU time in the array
30         cpu[i] = ct;
31     }
32 }
```

Enter the number of processors: 3

Enter the Cycles per Instruction of processor 1: 1.2

Enter the clock rate in GHz for processor 1: 3.2
The CPU time for processor 1 is: 375.000000

Enter the Cycles per Instruction of processor 2: 0.37500000

Enter the clock rate in GHz for processor 2: 1.5
The CPU time for processor 2 is: 250.000000

Enter the Cycles per Instruction of processor 3: 3.0

Enter the clock rate in GHz for processor 3: 0.5
The CPU time for processor 3 is: 6000.000000

The processor with the lowest execution time is: 250.000000

Process exited after 15.43 seconds with return value 0
Press any key to continue . . .

Compiler: TDM-GCC 9.2.0 64-bit Release
Errors: 0
Warnings: 0
Output Filename: C:\Users\Ramachandras PS\OneDrive\Desktop\dummy.exe
Output Size: 323.2841796875 KiB
Compilation Time: 0.31s

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