EXP NO: 36

CPU PERFORMANCE

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;
  int p,p1,i;
  float cpu[5];
  float cpi,ct,max;
  int n=1000;
  for(i=0;i<=4;i++)
  {
     cpu[5]=0;
```

```
  }
  printf("\n Enter the number of processors:");
  scanf("%d",&p);
  p1=p;
  for(i=0;i<p;i++)
{
    printf("\n Enter the Cycles per Instrcution of processor:");
  scanf("%f",&cpi);
  printf("\n Enter the clockrate in GHz:");
  scanf("%f",&cr);
  ct=1000*cpi/cr;
  printf("The CPU time is: %f",ct);
  cpu[i]=ct;
}
max=cpu[0];
for(i=0;i<p1;i++)
{
   if(cpu[i]<=max)
   max=cpu[i];
}
printf("\n The processor has lowest Execution time is: %f ", max);
   return 0;
}
```

INPUT:

OUTPUT:

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

EXP NO: 37

SINGLE PRECISION

AIM: To write a C program to implement SINGLE PRECISION.

PROCEDURE:

PROGRAM:

```
#include <stdio.h>
void printBinary(int n, int i)
{
 int k;
 for (k = i - 1; k >= 0; k--) {
  if ((n >> k) & 1)
   printf("1");
  else
   printf("0");
 }
}
typedef union {
 float f;
 struct
 {
  unsigned int mantissa : 23;
  unsigned int exponent : 8;
  unsigned int sign : 1;
 } raw;
} myfloat;
void printIEEE(myfloat var)
{
 printf("%d | ", var.raw.sign);
```

```
 printBinary(var.raw.exponent, 8);
 printf(" | ");
 printBinary(var.raw.mantissa, 23);
 printf("\n");
}
int main()
{
myfloat var;
var.f = 1259.125;
printf("IEEE 754 representation of %f is : \n",
  var.f);
 printIEEE(var);
 return 0;
}
```

INPUT:

OUTPUT:

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

EXP NO: 38

TWO STAGE PIPELINE

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:
12.1For choice 1: Perform addition (res = a + b). Increment the counter by 1.
12.2For 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:

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

EXP NO: 39

BOOTHS ALGORITHM

AIM: To write a C program to implement BOOTHS ALGORITHM.

PROCEDURE:

PROGRAM:

```
#include <stdio.h>
#include <math.h>

int a = 0,b = 0, c = 0, a1 = 0, b1 = 0, com[5] = { 1, 0, 0, 0, 0};
int anum[5] = {0}, anumcp[5] = {0}, bnum[5] = {0};
int acomp[5] = {0}, bcomp[5] = {0}, pro[5] = {0}, res[5] = {0};

void binary(){
    a1 = fabs(a);
    b1 = fabs(b);
    int r, r2, i, temp;
    for (i = 0; i < 5; i++){
        r = a1 % 2;
        a1 = a1 / 2;
        r2 = b1 % 2;
        b1 = b1 / 2;
        anum[i] = r;
        anumcp[i] = r;
        bnum[i] = r2;
        if(r2 == 0){
            bcomp[i] = 1;
        }
        if(r == 0){
            acomp[i] =1;
        }
    }
```

```
//part for two's complementing
c = 0;
for ( i = 0; i < 5; i++){
        res[i] = com[i]+ bcomp[i] + c;
        if(res[i] >= 2){
            c = 1;
        }
        else
            c = 0;
        res[i] = res[i] % 2;
  }
for (i = 4; i >= 0; i--){
  bcomp[i] = res[i];
}
//in case of negative inputs
if (a < 0){
   c = 0;
  for (i = 4; i >= 0; i--){
        res[i] = 0;
  }
  for ( i = 0; i < 5; i++){
        res[i] = com[i] + acomp[i] + c;
        if (res[i] >= 2){
            c = 1;
        }
        else
            c = 0;
        res[i] = res[i]%2;
  }
  for (i = 4; i >= 0; i--){
        anum[i] = res[i];
        anumcp[i] = res[i];
  }
}
if(b < 0){
  for (i = 0; i < 5; i++){
        temp = bnum[i];
        bnum[i] = bcomp[i];
        bcomp[i] = temp;
  }   }      }
```

```
void add(int num[]){
    int i;
    c = 0;
    for ( i = 0; i < 5; i++){
          res[i] = pro[i] + num[i] + c;
          if (res[i] >= 2){
              c = 1;
          }
          else{
              c = 0;
          }
          res[i] = res[i]%2;
     }
     for (i = 4; i >= 0; i--){
         pro[i] = res[i];
         printf("%d",pro[i]);
     }
   printf(":");
   for (i = 4; i >= 0; i--){
          printf("%d", anumcp[i]);
     }
}
void arshift(){//for arithmetic shift right
    int temp = pro[4], temp2 = pro[0], i;
    for (i = 1; i < 5  ; i++){//shift the MSB of product
       pro[i-1] = pro[i];
    }
    pro[4] = temp;
    for (i = 1; i < 5  ; i++){//shift the LSB of product
        anumcp[i-1] = anumcp[i];
    }
    anumcp[4] = temp2;
    printf("\nAR-SHIFT: ");//display together
    for (i = 4; i >= 0; i--){
        printf("%d",pro[i]);
    }
    printf(":");
    for(i = 4; i >= 0; i--){
        printf("%d", anumcp[i]);
    }        }
```

```
void main(){
  int i, q = 0;
  printf("\t\tBOOTH'S MULTIPLICATION ALGORITHM");
  printf("\nEnter two numbers to multiply: ");
  printf("\nBoth must be less than 16");
  //simulating for two numbers each below 16
  do{
      printf("\nEnter A: ");
      scanf("%d",&a);
      printf("Enter B: ");
      scanf("%d", &b);
    }while(a >=16 || b >=16);
    printf("\nExpected product = %d", a * b);
   binary();
   printf("\n\nBinary Equivalents are: ");
   printf("\nA = ");
   for (i = 4; i >= 0; i--){
      printf("%d", anum[i]);
   }
   printf("\nB = ");
   for (i = 4; i >= 0; i--){
      printf("%d", bnum[i]);
   }
   printf("\nB'+ 1 = ");
   for (i = 4; i >= 0; i--){
      printf("%d", bcomp[i]);
   }
   printf("\n\n");
   for (i = 0;i < 5; i++){
         if (anum[i] == q){//just shift for 00 or 11
             printf("\n-->");
             arshift();
             q = anum[i];
         }
         else if(anum[i] == 1 && q == 0){//subtract and shift for 10
            printf("\n-->");
            printf("\nSUB B: ");
            add(bcomp);//add two's complement to implement subtraction
            arshift();
            q = anum[i];
```

```
        }
        else{                           //add ans shift for 01
          printf("\n-->");
          printf("\nADD B: ");
          add(bnum);
          arshift();
          q = anum[i];
        }
    }
     printf("\nProduct is = ");
    for (i = 4; i >= 0; i--){
        printf("%d", pro[i]);
    }
    for (i = 4; i >= 0; i--){
        printf("%d", anumcp[i]);
    }
}
```

INPUT:

OUTPUT:

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

EXP NO: 40

INTEGER RESTORATION DIVISION

AIM: To write a C program to implement INTEGER RESTORATION DIVISION.

PROCEDURE:

PROGRAM:

```
#include<stdlib.h>
#include<stdio.h>
int acum[100]={0}     ;
void add(int acum[],int b[],int n);
int q[100],b[100];
int main()
{
int x,y;
printf("Enter the Number :");
scanf("%d%d",&x,&y);
int i=0;
while(x>0||y>0)
{
if(x>0)
{
q[i]=x%2;
x=x/2;
}
else
{
q[i]=0;
}
if(y>0)
{
b[i]=y%2;
y=y/2;
}
```

```
else
{
b[i]=0;
}
i++;
}

int n=i;
int bc[50];
printf("\n");
for(i=0;i<n;i++)
{
if(b[i]==0)
{
bc[i]=1;
}
else
{
bc[i]=0;
}
}
bc[n]=1;
for(i=0;i<=n;i++)
{
if(bc[i]==0)
{
bc[i]=1;
i=n+2;
}
else
{
bc[i]=0;
}
}
int l;
 b[n]=0;
int k=n;
int n1=n+n-1;
int j,mi=n-1;
for(i=n;i!=0;i--)
```

```
{
for(j=n;j>0;j--)
{
acum[j]=acum[j-1];
}
acum[0]=q[n-1];
for(j=n-1;j>0;j--)
{
q[j]=q[j-1];
}

add(acum,bc,n+1);
if(acum[n]==1)
{
q[0]=0;
add(acum,b,n+1);
}
else
{
q[0]=1;
}
}
printf("\nQuoient   : ");
for(  l=n-1;l>=0;l--)
{
printf("%d",q[l]);
}
printf("\nRemainder : ");
for( l=n;l>=0;l--)
{
printf("%d",acum[l]);
}
return 0;
}
void add(int acum[],int bo[],int n)
{
int i=0,temp=0,sum=0;
for(i=0;i<n;i++)
{
sum=0;
```

```
sum=acum[i]+bo[i]+temp;
if(sum==0)
{
acum[i]=0;
temp=0;
}
else if (sum==2)
{
acum[i]=0;
temp=1;
}
else if(sum==1)
{
acum[i]=1;
temp=0;
}
else if(sum==3)
{
acum[i]=1;
temp=1;
}
}
}
```

INPUT:

OUTPUT:

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