**NISHITH SHAH ( SJSU ID:011478948)** **DATE:28TH SEP’16**

# 5 – STAGE PIPELINED (IEEE 754) FLOATING POINT ADDER

## CODE:

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

void Compare(int i);
void Shift(int i);
void Align(int i);
void Add(int i);
void Normalize(int i);

        int A_sign_48[6]={ 0 , 0 , 1 , 1 , 0 , 0 },                         //Array of Signs of A
            B_sign_48[6]={ 0 , 1 , 0 , 1 , 0 , 1 },                         //Array of Signs of B
            Sum_sign_48[6],                                                 //Array of Signs of Sum
            A_exp_48[6]={ 0x85 , 0x85 , 0x84 , 0x87 , 0x00 , 0x00 },        //Array of Exponents of A
            B_exp_48[6]={ 0x86 , 0x84 , 0x84 , 0x85 , 0x00 , 0x86 },        //Array of Exponents of B
            Sum_exp_48[6],                                                  //Array of Exponents of Sum
            shift;                                                          //Shifts required to align Mantissa

        short int clock;                                                    //To simulate a Sequential execution

                            //Array of Mantissa’s of A
        unsigned long int A_mantissa_48[6]={ 0x480000 , 0x480000 , 0x540000 , 0x160000 , 0x000000 ,
                                             0x000000 },
                            //Array of Mantissa’s of B
                            B_mantissa_48[6]={ 0x480000 , 0x480000 , 0x0c0000 , 0x460000 , 0x000000 ,
                                               0x2B0000 },
                            //Array of Mantissa’s of Sum
                            Sum_mantissa_48[6],
                            //Masks for Normalization
                            mask1,
                            mask2,
                            //Array of Normalized Sum
                            x[6];

void main()
{
        clrscr();
        for(clock=0;clock<10;clock++)
                {
                        switch(clock)
                        {
                                case 0:
                                {
                                        Compare(0);                         //Comparison of Pair 1
                                        printf("\nSTAGE 0 END");
                                        break;
                                }
```

```
case 1:
{
        Shift(0);                       //Shifting of Pair 1
        Compare(1);                     //Comparison of Pair 2
        printf("\nSTAGE 1 END");
        break;
}
case 2:
{
        Align(0);                       //Alignment of Pair 1
        Shift(1);                       //Shifting of Pair 2
        Compare(2);                     //Comparison of Pair 3
        printf("\nSTAGE 2 END");
        break;
}
case 3:
{
        Add(0);                         //Addition of Pair 1
        Align(1);                       //Alignment of Pair 2
        Shift(2);                       //Shifting of Pair 3
        Compare(3);                     //Comparison of Pair 4
        printf("\nSTAGE 3 END");
        break;
}
case 4:
{
        Normalize(0);                   //Normalization of Pair 1
        Add(1);                         //Addition of Pair 2
        Align(2);                       //Alignment of Pair 3
        Shift(3);                       //Shifting of Pair 4
        Compare(4);                     //Comparison of Pair 5
        printf("\nSTAGE 4 END");
        break;
}
case 5:
{
        Normalize(1);                   //Normalization of Pair 2
        Add(2);                         //Addition of Pair 3
        Align(3);                       //Alignment of Pair 4
        Shift(4);                       //Shifting of Pair 5
        Compare(5);                     //Comparison of Pair 6
        printf("\nSTAGE 5 END");
        break;
}
```

```
                case 6:
                {
                    Normalize(2);                   //Normalization of Pair 3
                    Add(3);                         //Addition of Pair 4
                    Align(4);                       //Alignment of Pair 5
                    Shift(5);                       //Shifting of Pair 6
                    printf("\nSTAGE 6 END");
                    break;
                }
                case 7:
                {
                    Normalize(3);                   //Normalization of Pair 4
                    Add(4);                         //Addition of Pair 5
                    Align(5);                       //Alignment of Pair 6
                    printf("\nSTAGE 7 END");
                    break;
                }
                case 8:
                {
                    Normalize(4);                   //Normalization of Pair 5
                    Add(5);                         //Addition of Pair 6
                    printf("\nSTAGE 8 END");
                    break;
                }
                case 9:
                {
                    Normalize(5);                   //Normalization of Pair 6
                    printf("\nSTAGE 9 END");
                    break;
                }
            }
            getch();
            clrscr();
     }
 }
void Compare(int i)
{
    printf("\n----------------------------STAGE - 1-------------------------------");

    //Loop to determine the shift and the greater number
    if(A_exp_48[i]>B_exp_48[i])
    {
        shift=A_exp_48[i]-B_exp_48[i];
     // printf("\nA is greater\n");
    }
    else if(A_exp_48[i]<B_exp_48[i])
    {
        shift=B_exp_48[i]-A_exp_48[i];
      //printf("\nB is greater\n");
    }
    else
    {
        shift=0;
    }

    //Including the 1 from "1.m" into the Mantissa
    mask1=0x400000;
    printf("\nTestbench %d",i+1);
    A_mantissa_48[i]=A_mantissa_48[i]>>1;
    B_mantissa_48[i]=B_mantissa_48[i]>>1;
    A_mantissa_48[i]=A_mantissa_48[i]|mask1;
    B_mantissa_48[i]=B_mantissa_48[i]|mask1;
    A_exp_48[i]=A_exp_48[i]+1;
    B_exp_48[i]=B_exp_48[i]+1;
}

void Shift(int i)
{
    printf("\n----------------------------STAGE - 2------------------------------");

        //Aligning the Mantissa based on the number of shifts required and making the exponents same
        printf("\nTestbench %d",i+1);
        if(A_exp_48[i]<B_exp_48[i])
        {
            A_mantissa_48[i]=A_mantissa_48[i]>>shift;
            A_exp_48[i]=A_exp_48[i]+shift;
            Sum_sign_48[i]=B_sign_48[i];
        }
        else if(A_exp_48[i]>B_exp_48[i])
        {
            B_mantissa_48[i]=B_mantissa_48[i]>>shift;
            B_exp_48[i]=B_exp_48[i]+shift;
            Sum_sign_48[i]=A_sign_48[i];
        }
        else
        {
            A_mantissa_48[i]=A_mantissa_48[i];
            B_mantissa_48[i]=B_mantissa_48[i];
            Sum_sign_48[i]=A_sign_48[i];
        }
        Sum_exp_48[i]=A_exp_48[i];
//  printf("\nThe alligned mantissa of A is %lx\n",A_mantissa_48[i]);
//  printf("\nThe alligned mantissa of B is %lx\n",B_mantissa_48[i]);
}
```

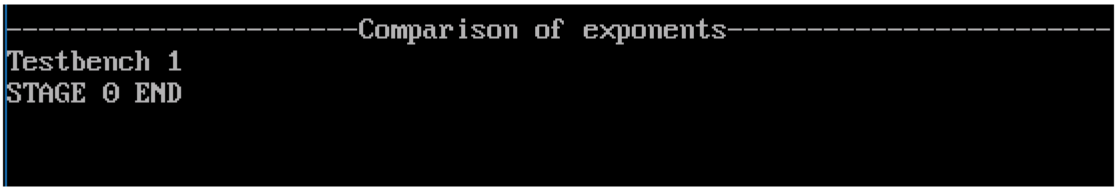
```
void Align(int i)
{
        printf("\n---------------------------STAGE - 3 ----------------------------");

        //Determining the smaller mantissa and taking it’s 2’s complement
        printf("\nTestbench %d",i+1);
        if(A_sign_48[i]!=B_sign_48[i])
        {
                if(A_mantissa_48[i]<B_mantissa_48[i])
                {
                        A_mantissa_48[i]=(~A_mantissa_48[i])+1;
                }
                else
                {
                        B_mantissa_48[i]=(~B_mantissa_48[i])+1;
                }
   //   printf("\nThe mantissa's after 2's complement\n\n %lx \n\n %lx\n", A_mantissa_48[i],
                 B_mantissa_48[i]);
        }
        else
        {
                A_mantissa_48[i]=A_mantissa_48[i];
                B_mantissa_48[i]=B_mantissa_48[i];
    //  printf("\n2's complement is not required as signs are same\n");
        }
}

void Add(int i)
{
        printf("\n---------------------------STAGE - 4-----------------------------");

        //Addition of the Mantissa’s
        printf("\nTestbench %d",i+1);
        Sum_mantissa_48[i]=A_mantissa_48[i]+B_mantissa_48[i];
        Sum_exp_48[i]=A_exp_48[i];
   //   printf("\nThe Sum is %d %x %lx\n",Sum_sign_48[i],Sum_exp_48[i],Sum_mantissa_48[i]);
}

void Normalize(int i)
{
        printf("\n---------------------------STAGE - 5-----------------------------");

        //Normalization of the Mantissa’s
        printf("\nTestbench %d",i+1);
        mask2=0x7ffff;
                if(A_sign_48[i]!=B_sign_48[i])                  //For Sign(A) != Sign(B)
                {
                        if(A_sign_48[i]==0&B_sign_48[i]==1)     //For A=+ve & B=-ve
                        {
```

```
        if(A_mantissa_48[i]==0)                  //For A=0
        {
            Sum_mantissa_48[i]=Sum_mantissa_48[i]<<2;
             x[i]=Sum_mantissa_48[i]&mask2;
             x[i]=x[i]>>1;
             Sum_exp_48[i]=Sum_exp_48[i]-1;
        printf("\nThe Normalized Value of Sum is %d %x %lx\n" , Sum_sign_48[i],
              Sum_exp_48[i], x[i]);
        }
        Else                                     //For A !=0
        {
            Sum_mantissa_48[i]=Sum_mantissa_48[i]<<2;
            x[i]=Sum_mantissa_48[i]&mask2;
            Sum_exp_48[i]=Sum_exp_48[i]-2;
            printf("\nThe Normalized Value of Sum is %d %x %lx\n",
                   Sum_sign_48[i], Sum_exp_48[i],x[i]);
        }
    }

    Else                                         //For Sign(A) != Sign(B)
    {
        Sum_mantissa_48[i]=Sum_mantissa_48[i]<<2;
        x[i]=Sum_mantissa_48[i]&mask2;
        Sum_exp_48[i]=Sum_exp_48[i]-2;
        printf("\nThe Normalized Value of Sum is %d %x %lx\n", Sum_sign_48[i],
              Sum_exp_48[i] ,x[i]);
    }
}
Else                                             //For Sign(A) = Sign(B)
{
    if(A_sign_48[i]==0&B_sign_48[i]==0)          //Both A & B = +ve
    {
        if(A_mantissa_48[i]==0)                  //For A=0
        {
            x[i]=Sum_mantissa_48[i]&mask2;
            Sum_exp_48[i]=Sum_exp_48[i]-1;
        }


        Else                                     //For A != 0
        {
            x[i]=Sum_mantissa_48[i]&mask2;
        }

    }
    else if(A_sign_48[i]==1&B_sign_48[i]==1)     //Both A & B = -ve
    {
        Sum_mantissa_48[i]=Sum_mantissa_48[i]<<1;
        Sum_exp_48[i]=Sum_exp_48[i]-1;
        x[i]=Sum_mantissa_48[i]&mask2;
    }
```

```
                        else{}
                        printf("\nThe Normalized Value of Sum is %d %x %lx\n", Sum_sign_48[i],
                                Sum_exp_48[i], x[i]);
                }
}
```

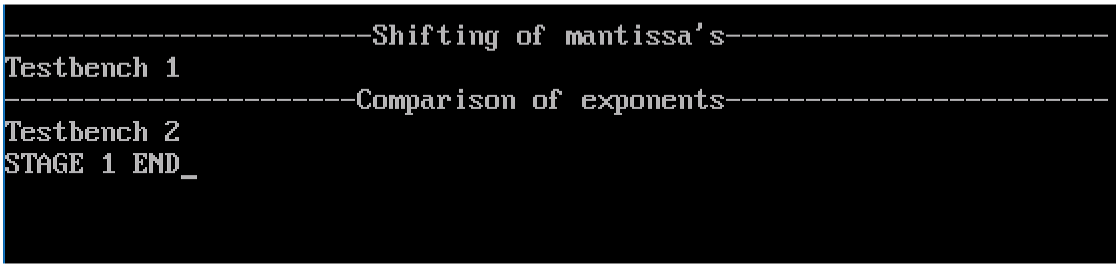
## OUTPUT:

### STAGE 0 :



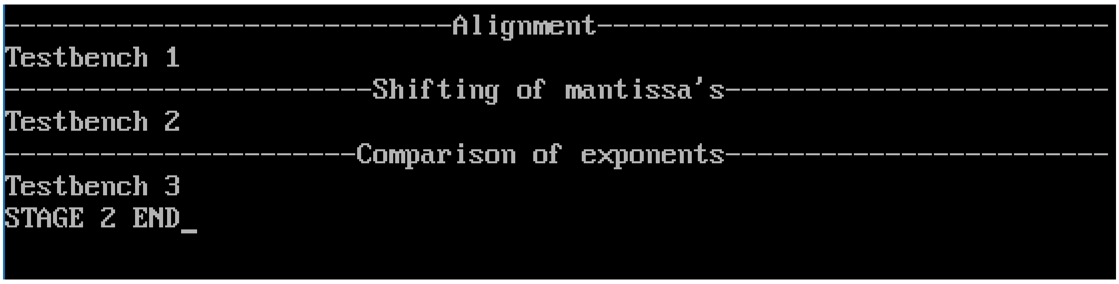
```
-----------------------Comparison of exponents-------------------------
Testbench 1
STAGE 0 END
```

### STAGE 1 :



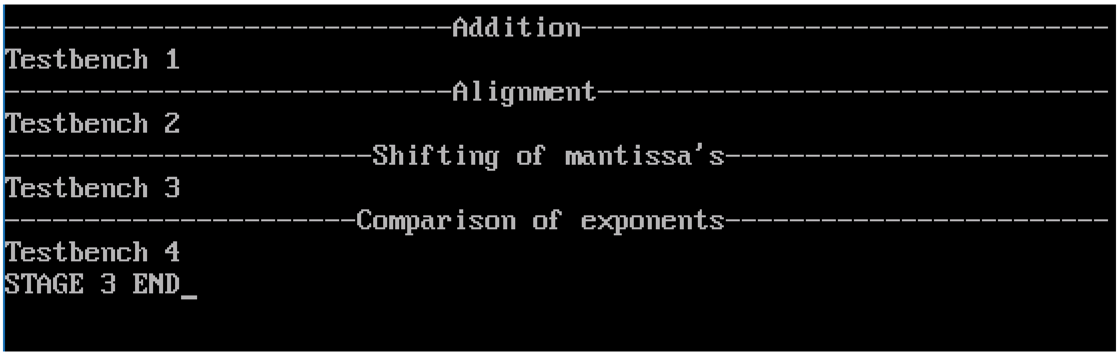
```
------------------------Shifting of mantissa's-------------------------
Testbench 1
-----------------------Comparison of exponents-------------------------
Testbench 2
STAGE 1 END_
```

### STAGE 2 :



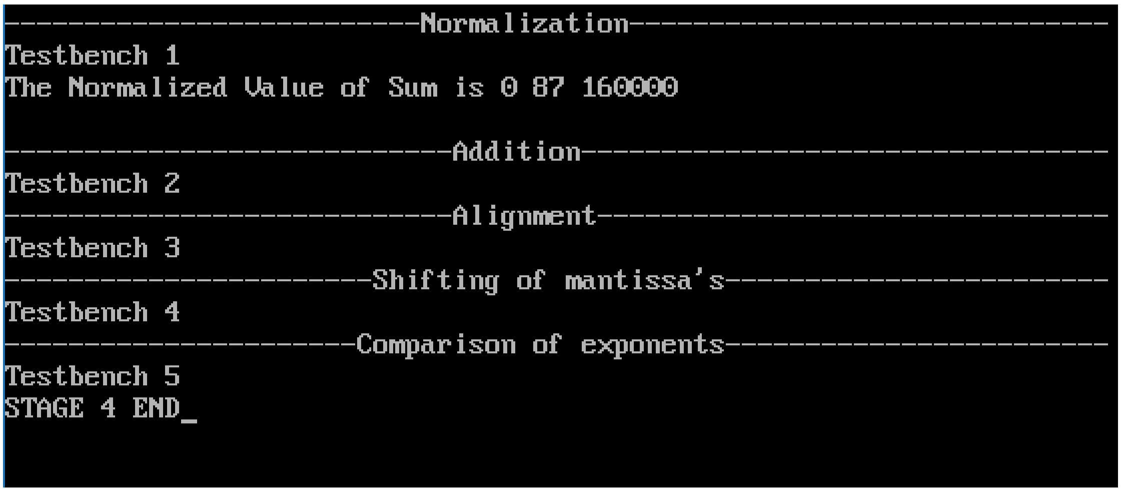
```
-----------------------------Alignment---------------------------------
Testbench 1
------------------------Shifting of mantissa's-------------------------
Testbench 2
-----------------------Comparison of exponents-------------------------
Testbench 3
STAGE 2 END_
```

### STAGE 3 :



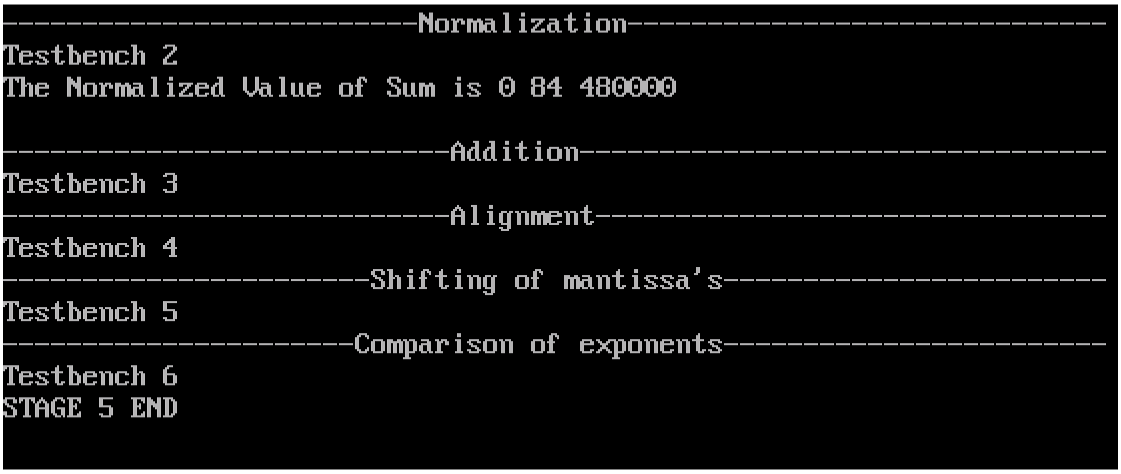
```
-----------------------------Addition----------------------------------
Testbench 1
-----------------------------Alignment---------------------------------
Testbench 2
------------------------Shifting of mantissa's-------------------------
Testbench 3
-----------------------Comparison of exponents-------------------------
Testbench 4
STAGE 3 END_
```

## STAGE 4 :



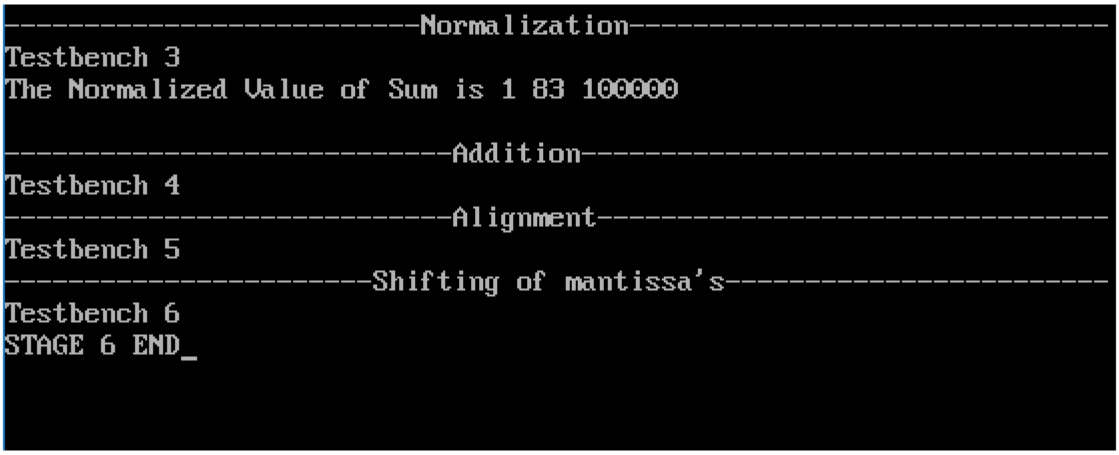
```
--------------------------Normalization------------------------------
Testbench 1
The Normalized Value of Sum is 0 87 160000

----------------------------Addition---------------------------------
Testbench 2
----------------------------Alignment--------------------------------
Testbench 3
-----------------------Shifting of mantissa's------------------------
Testbench 4
----------------------Comparison of exponents------------------------
Testbench 5
STAGE 4 END_
```

## STAGE 5:



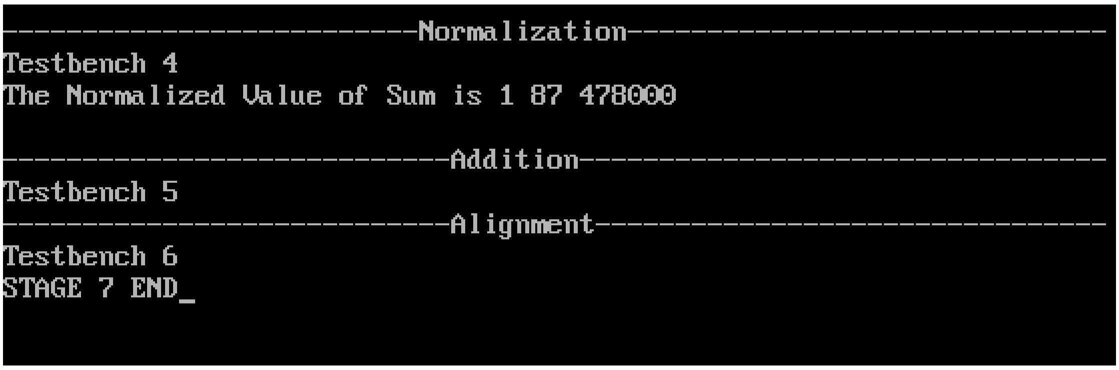
```
--------------------------Normalization------------------------------
Testbench 2
The Normalized Value of Sum is 0 84 480000

----------------------------Addition---------------------------------
Testbench 3
----------------------------Alignment--------------------------------
Testbench 4
-----------------------Shifting of mantissa's------------------------
Testbench 5
----------------------Comparison of exponents------------------------
Testbench 6
STAGE 5 END
```

## STAGE 6:



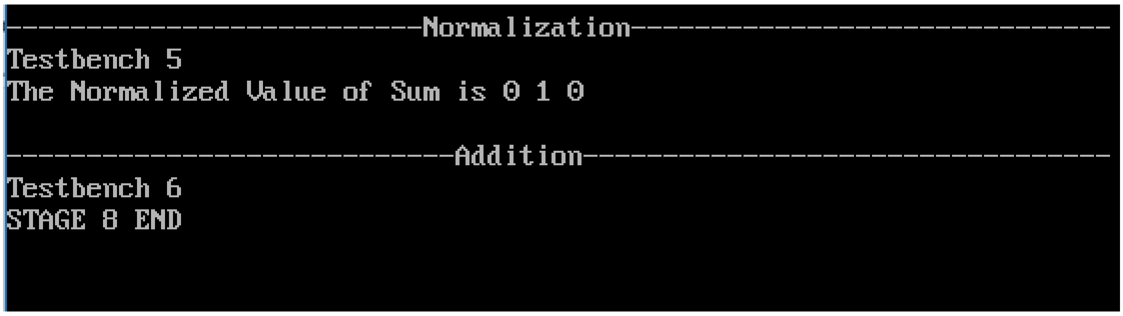
```
--------------------------Normalization------------------------------
Testbench 3
The Normalized Value of Sum is 1 83 100000

----------------------------Addition---------------------------------
Testbench 4
----------------------------Alignment--------------------------------
Testbench 5
-----------------------Shifting of mantissa's------------------------
Testbench 6
STAGE 6 END_
```

## STAGE 7:



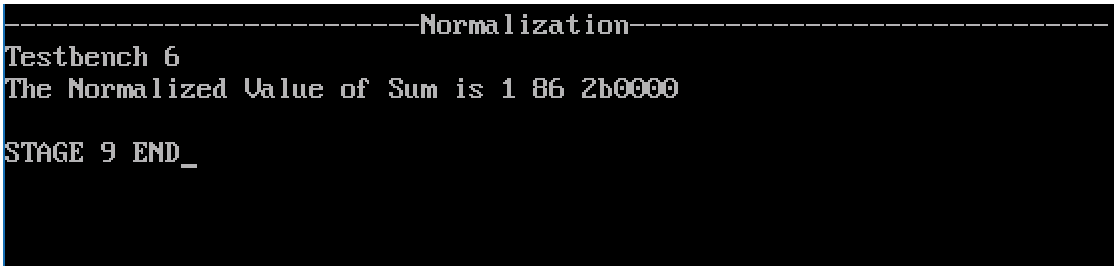
```
--------------------------Normalization------------------------------
Testbench 4
The Normalized Value of Sum is 1 87 478000

----------------------------Addition---------------------------------
Testbench 5
----------------------------Alignment--------------------------------
Testbench 6
STAGE 7 END_
```

## STAGE 8:



```
--------------------------Normalization------------------------------
Testbench 5
The Normalized Value of Sum is 0 1 0

----------------------------Addition---------------------------------
Testbench 6
STAGE 8 END
```

## STAGE 9:



```
--------------------------Normalization------------------------------
Testbench 6
The Normalized Value of Sum is 1 86 2b0000

STAGE 9 END_
```