Green University of Bangladesh
Department of Computer Science and Engineering (CSE)

Faculty of Sciences and Engineering
Semester: (Spring, Year:2023), B.Sc. in CSE (Day)

Course Title: Computer Architecture
Course Code: CSE 211 **Section:** DA

Lab Project Name: Arithmetic Operations (Restoring, Left Shift, Subtraction)

Student Details

	Name	ID
1.	Md. Rahul Islam Joy	212902070
2.	Sajid Rahman Rifan	212902017

Submission Date: 23/06/2023
Course Teacher's Name: Mahbubur Rahman

[For Teachers use only: Don't Write Anything inside this box]

Lab Project Status

Marks: ………………………………… **Signature: ....**

Comments: .... **Date: ....**

Table of Contents

Chapter 1 Introduction 3
1.1 Introduction 3
Chapter 2 4
Design/Development/Implementation of the Project 4
2.1 Tools & Technology 4
2.2 Algorithm 4
2.3 Flowchart 4
2.4 Implementation 5
Chapter 3 Performance Evaluation 9
3.1 Results and Discussions 9
Chapter 4 Conclusion 10
4.1 Introduction 10
References 10

Chapter 1

Introduction

1.1 Introduction

Arithmetic operations are fundamental mathematical operations that involve manipulating numbers to perform calculations. They form the basis of mathematical computations and are used extensively in various fields such as science, engineering, finance, and everyday life.

There are several basic arithmetic operations, including addition, subtraction, multiplication, and division. These operations allow us to perform calculations and solve problems involving quantities, measurements, and values. These arithmetic operations are fundamental building blocks for more complex mathematical calculations and algorithms. Understanding how they work and when to use them is essential for performing accurate calculations and solving problems efficiently.

1.2 Design Goals/Objective

1. The objective you mentioned, "Arithmetic Operations (Restoring, L-R Shift, Subtraction)," seems to indicate that you would like information or assistance related to these specific arithmetic operations.
2. I can provide an explanation of each operation and how they are typically performed.
3. Please let me know which operation you would like to learn about or if you have any specific questions related to them.

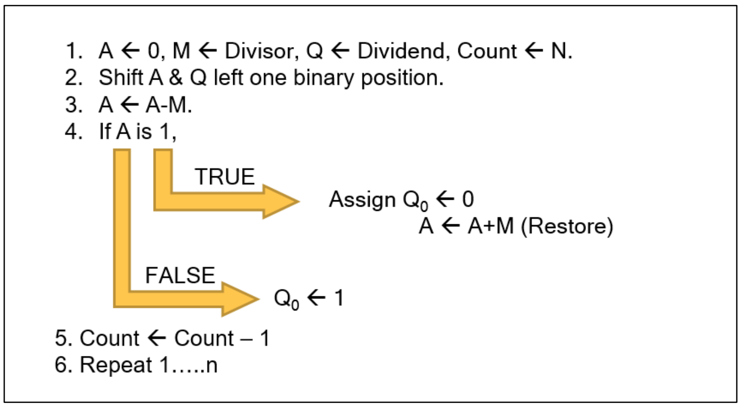
Chapter 2

Design/Development/Implementation of the Project

2.1 Tools & Technology

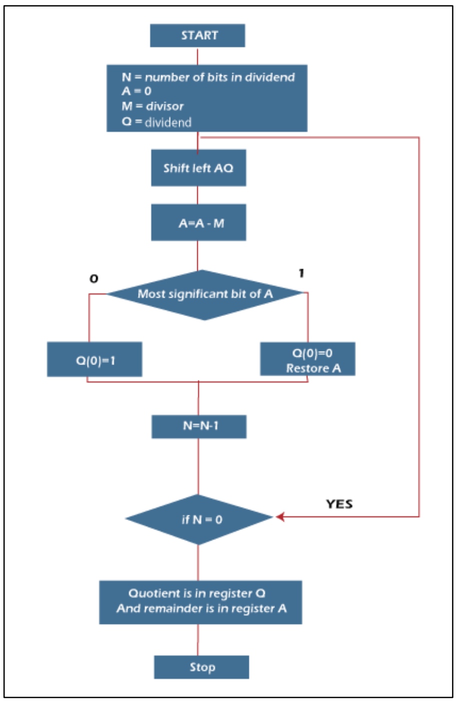
- Emulator: Codeblocks – Windows
- Language: C Programming

2.2 Algorithm



```
1. A ← 0, M ← Divisor, Q ← Dividend, Count ← N.
2. Shift A & Q left one binary position.
3. A ← A-M.
4. If A is 1,
       TRUE  → Assign $Q_0$ ← 0
                      A ← A+M (Restore)
       FALSE → $Q_0$ ← 1
5. Count ← Count – 1
6. Repeat 1…..n
```

2.3 Flowchart



START
N = number of bits in dividend
A = 0
M = divisor
Q = dividend
Shift left AQ
A=A - M
0
1
Most significant bit of A
Q(0)=1
Q(0)=0
Restore A
N=N-1
YES
if N = 0
Quotient is in register Q
And remainder is in register A
Stop

2.4 Theoretical Implementation

M	A	Q	Operation
00011	00000	1011	Initialize
00011	00001	011_	Shift left AQ
00011	11110	011_	A = A - M
00011	00001	0110	Q[0] = 0 And restore A
00011	00010	110_	Shift left AQ
00011	11111	110_	A = A - M
00011	00010	1100	Q[0] = 0
00011	00101	100_	Shift left AQ
00011	00010	100_	A = A - M
00011	00010	1001	Q[0] = 1
00011	00101	001_	Shift left AQ
00011	00010	001_	A = A - M
00011	00010	0011	Q[0] = 1

2.5 Implementation

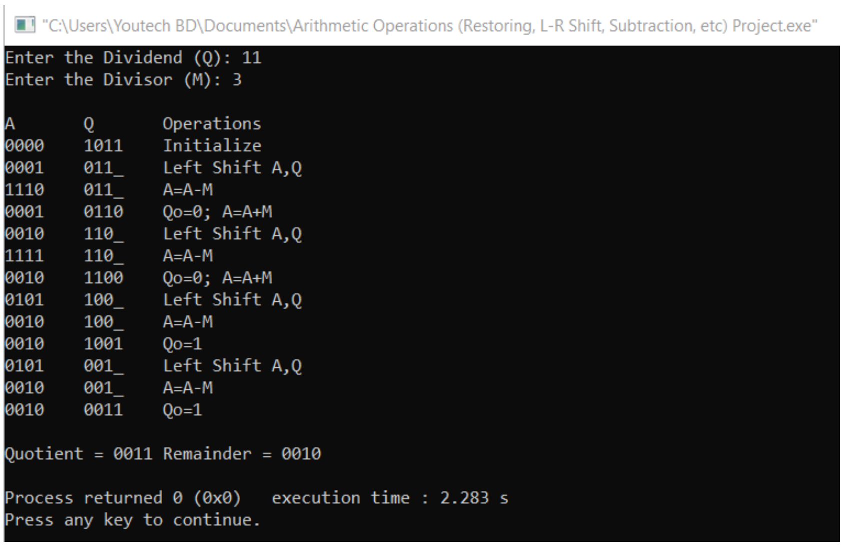
```
#include <stdio.h>
#include <stdlib.h>

int dec_bin(int, int []);
int twos(int [], int []);
int left(int [], int []);
int add(int [], int []);

int main()
{
    int a, b, m[4]={0,0,0,0}, q[4]={0,0,0,0}, acc[4]={0,0,0,0}, m2[4], i, n=4;
    printf("Enter the Dividend: ");
    scanf("%d", &a);
    printf("Enter the Divisor: ");
    scanf("%d", &b);
    dec_bin(a, q);
    dec_bin(b, m);
    twos(m, m2);
    printf("\nA\tQ\tQ\n");
    for(i=3; i>=0; i--)
    {
        printf("%d", acc[i]);
    }
    printf("\t");
    for(i=3; i>=0; i--)
    {
        printf("%d", q[i]);
    }
    printf("\tStart\n");
    while(n>0)
    {
        left(acc, q);
        for(i=3; i>=0; i--)
        {
            printf("%d", acc[i]);
        }
        printf("\t");
        for(i=3; i>=1; i--)
        {
            printf("%d", q[i]);
        }
        printf("_\tLeft Shift A,Q\n");
        add(acc, m2);
        for(i=3; i>=0; i--)
        {
            printf("%d", acc[i]);
        }
        printf("\t");
        for(i=3; i>=1; i--)
        {
            printf("%d", q[i]);
        }
        printf("_\tA=A-M\n");
        if(acc[3]==0)
        {
            q[0]=1;
            for(i=3; i>=0; i--)
            {
                printf("%d", acc[i]);
            }
            printf("\t");
            for(i=3; i>=0; i--)
            {
                printf("%d", q[i]);
            }
            printf("\tQo=1\n");
        }
        else
        {
            q[0]=0;
            add(acc, m);
            for(i=3; i>=0; i--)
            {
                printf("%d", acc[i]);
            }
            printf("\t");
            for(i=3; i>=0; i--)
            {
                printf("%d", q[i]);
            }
            printf("\tQo=0; A=A+M\n");
        }
        n--;
    }
    printf("\nQuotient = ");
    for(i=3; i>=0; i--)
    {
            printf("%d", q[i]);
    }
    printf("\tRemainder = ");
    for(i=3; i>=0; i--)
    {
            printf("%d", acc[i]);
    }
    printf("\n");
    return 0;
}

int dec_bin(int d, int m[])
{
    int b=0, i=0;
    for(i=0; i<4; i++)
    {
        m[i]=d%2;
        d=d/2;
    }
    return 0;
}

int twos(int m[], int m2[])
{
    int i, m1[4];
    for(i=0; i<4; i++)
    {
        if(m[i]==0)
        {
            m1[i]=1;
        }
        else
        {
            m1[i]=0;
        }
    }
    for(i=0; i<4; i++)
    {
        m2[i]=m1[i];
    }
    if(m2[0]==0)
    {
        m2[0]=1;
    }
    else
    {
        m2[0]=0;
        if(m2[1]==0)
        {
            m2[1]=1;
        }
        else
        {
            m2[1]=0;
            if(m2[2]==0)
            {
                m2[2]=1;
            }
            else
            {
                m2[2]=0;
                if(m2[3]==0)
                {
                  m2[3]=1;
                }
                else
                {
                  m2[3]=0;
                }
            }
        }
    }
    return 0;
}

int left(int acc[], int q[])
{
    int i;
    for(i=3; i>0; i--)
    {
        acc[i]=acc[i-1];
    }
    acc[0]=q[3];
    for(i=3; i>0; i--)
    {
        q[i]=q[i-1];
    }
}

int add(int acc[], int m[])
{
  int i, carry=0;
  for(i=0; i<4; i++)
  {
    if(acc[i]+m[i]+carry==0)
    {
      acc[i]=0;
      carry=0;
    }
    else if(acc[i]+m[i]+carry==1)
    {
      acc[i]=1;
      carry=0;
    }
    else if(acc[i]+m[i]+carry==2)
    {
      acc[i]=0;
      carry=1;
    }
    else if(acc[i]+m[i]+carry==3)
    {
      acc[i]=1;
      carry=1;
    }
  }
  return 0;
}
```

Chapter 3

Performance Evaluation

3.1 Results and Discussions

3.1.1 Results


"C:\Users\Youtech BD\Documents\Arithmetic Operations (Restoring, L-R Shift, Subtraction, etc) Project.exe"

```
Enter the Dividend (Q): 11
Enter the Divisor (M): 3

A       Q       Operations
0000    1011    Initialize
0001    011_    Left Shift A,Q
1110    011_    A=A-M
0001    0110    Qo=0; A=A+M
0010    110_    Left Shift A,Q
1111    110_    A=A-M
0010    1100    Qo=0; A=A+M
0101    100_    Left Shift A,Q
0010    100_    A=A-M
0010    1001    Qo=1
0101    001_    Left Shift A,Q
0010    001_    A=A-M
0010    0011    Qo=1

Quotient = 0011 Remainder = 0010

Process returned 0 (0x0)   execution time : 2.283 s
Press any key to continue.
```

Fig:01

3.1.2 Analysis and Discussion

In Fig:01 we take dividend (Q) is 11 and Divisor (Q) is 3. In the program first of all it Will initialize first. Then it will do left shift of A and Q and then it will do A-M□A. Then if A=1 than Q(0)=1 or Q(0). In this process it will continue.

Chapter 4

Conclusion

4.1 Introduction

In this discussion, we explored three fundamental arithmetic operations: restoring, left-right shift, and subtraction. These operations are essential in various mathematical computations and play a significant role in many fields, including computer science and engineering.
Restoring is a method used to perform addition or subtraction on binary numbers. It involves comparing the magnitudes of the numbers and determining the result based on the comparison. By applying the restoring algorithm, we can accurately perform arithmetic operations on binary data.

4.2 Scope of Future Work

The scope of future work in arithmetic operations can include various areas of improvement and advancement. Here are a few potential directions for future research and development:
Efficient algorithms: There is always a need for more efficient algorithms for arithmetic operations. Researchers can explore new algorithms that reduce computational complexity or improve speed and accuracy. For example, developing faster multiplication or division algorithms can have significant implications in various domains. Parallel processing: With the increasing availability of parallel processing architectures, there is a scope for optimizing arithmetic operations to leverage the power of parallel computing. Future work can focus on designing algorithms that effectively utilize parallel processing units such as GPUs or distributed computing systems to perform arithmetic operations more efficiently. Error analysis and precision: The accuracy and precision of arithmetic operations are crucial in many scientific and computational applications. Future work can focus on analyzing the sources of error in arithmetic operations and developing techniques to minimize or compensate for those errors. This can involve investigating numerical stability, error propagation, and rounding errors.

References

[1] Youtube
[2] Geek for Geeks
[3] Github