Terna Engineering College
Computer Engineering Department

Program: Sem V

Course: Microprocessor Lab

Faculty: ARATHI BOYANAPALLI

LAB Manual

PART A

(PART A: TO BE REFERRED BY STUDENTS)

Experiment No. 10

A.1 Aim:

Write & execute a mixed program to shift a number given no of times

A.2 Prerequisite:

Knowledge of c language and DOS

A.3 Outcome:

After successful completion of this experiment, students will be able to shift a number given no of times

A.4 Theory:

- Mixed-language programming allows you to combine the unique strengths of C++ with your assembly-language routines. There are some cases where you want to achieve things using inline assembly, such as improving speed, reducing memory needs and getting more efficiency. However, an inline assembler is not as powerful as TASM, it does not support macros or directives.
- A bit shift moves each digit in a number's binary representation left or right. There are three main types of shifts:
  Left Shifts
  When shifting left, the most-significant bit is lost, and a 00 bit is inserted on the other end.
  The left shift operator is usually written as "<<".
  0010 << 1 → 0100
  0010 << 2 → 1000
  A single left shift multiplies a binary number by 2:
  0010 << 1 → 0100
  0010 is 2
  0100 is 4

Logical Right Shifts

When shifting right with a logical right shift, the least-significant bit is lost and a 00 is inserted on the other end.

1011 >>> 1 → 0101

1011 >>> 3 → 0001

For positive numbers, a single logical right shift divides a number by 2, throwing out any remainders.

0101 >>> 1 → 0010

0101 is 5

0010 is 2

Arithmetic Right Shifts

When shifting right with an arithmetic right shift, the least-significant bit is lost and the most-significant bit is copied.

Languages handle arithmetic and logical right shifting in different ways. Java provides two right shift operators: >> does an arithmetic right shift and >>> does a logical right shift.

1011 >> 1 → 1101

1011 >> 3 → 1111

0011 >> 1 → 0001

0011 >> 2 → 0000

The first two numbers had an 11 as the most significant bit, so more 11's were inserted during the shift. The last two numbers had a 00 as the most significant bit, so the shift inserted more 00's.

If a number is encoded using two's complement, then an arithmetic right shift preserves the number's sign, while a logical right shift makes the number positive.

// Arithmetic shift

1011 >> 1 → 1101

1011 is -5

1101 is -3

// Logical shift

1111 >>> 1 → 0111

1111 is -1

0111 is 7

Algorithm:

Procedure rotateLeft(int,int):

1. Start
2. Calculate c=(n << d) | (n >> (tot_bits - d))
3. Return the value of c
4. End

Procedure rotateRight(int,int):
1. Start
2. Calculate c=(n >> d) | (n << (tot_bits - d))
3. Return the value of c
4. End

PART B

(PART B: TO BE COMPLETED BY STUDENTS)

(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the ERP or emailed to the concerned lab in charge faculties at the end of the practical in case the there is no ERP access available)

Roll No.: 50	Name: Amey Thakur
Class: TE-Comps B	Batch: B3
Date of Experiment: 08/10/2020	Date of Submission: 08/10/2020
Grade:	

B.1 Observations and learning:

(Software Code written by a student and output of the program)

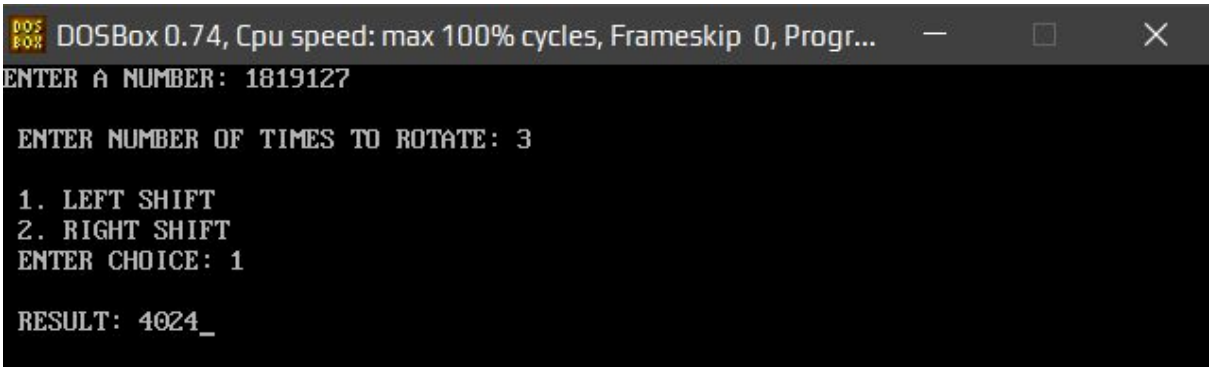
- **Mixed program to shift a number given no of times**

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int num=0, times=0, choice=0;
        clrscr();
        printf("ENTER A NUMBER: ");
        scanf("%d", &num);
        printf("\nENTER NUMBER OF TIMES TO ROTATE: ");
        scanf("%d", &times);
        printf("\n1.LEFT SHIFT");
        printf("\n2.RIGHT SHIFT");
        printf("\nENTER CHOICE: ");
        scanf("%d", &choice);
```

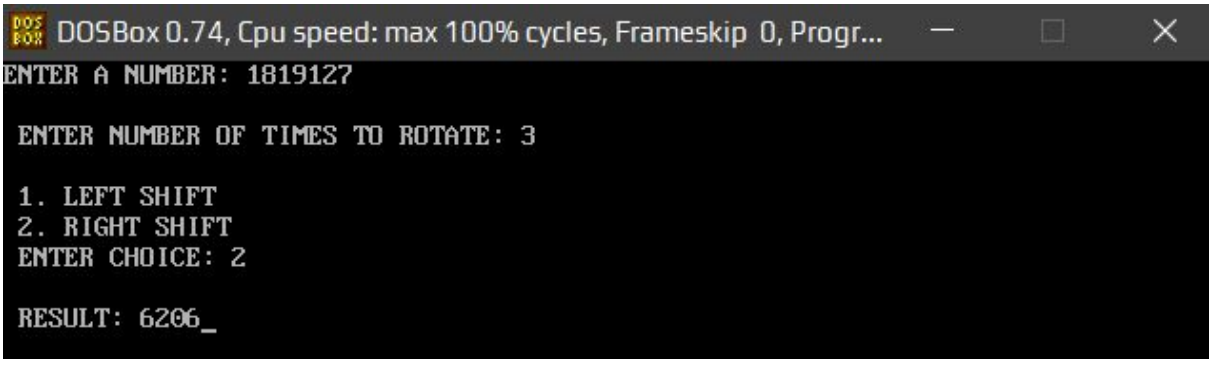
```
        asm mov ax, num
        asm mov cx, times
        if (choice == 1)
        {
                asm shl ax, cl
        }
        else if (choice == 2)
        {
                asm shr ax, cl
        }
        else
        {
                printf("\nINVALID CHOICE");
        }
        asm mov num, ax
        printf("\nRESULT: %d", num);
        getch();
}
```

- **Output -**

➔ LEFT SHIFT



➔ RIGHT SHIFT



B.2 Conclusion:

(Students must write the conclusion as per the attainment of individual outcome listed above and learning/observation noted in section B.1)

- A bit shift moves each digit in a number's binary representation left or right.
- We learned a mixed program to shift a number given a number of times.

B.5 Question of Curiosity

Q1. Explain basic rules to write a mixed-language program to Separate shifting a number

Ans:

➔ A shift operator performs bit manipulation on data by shifting the bits of its first operand right or left.

➔ Left Shift: Left shift operator is a binary operator which shift some number of bits, in the given bit pattern, to the left and append 0 at the end. Left shift is equivalent to multiplying the bit pattern with 2k (if we are shifting k bits).

➔ Right Shift: Right shift operator is a binary operator which shift some number of bits, in the given bit pattern, to the right and append 1 at the end. The right shift is equivalent to dividing the bit pattern with 2k (if we are shifting k bits).

Q2. Write a short note for shifting bits towards the right and left side

Ans:

A bit shift moves each digit in a number's binary representation left or right. There are three main types of shifts:

1. Left Shifts

When shifting left, the most-significant bit is lost, and a 0 bit is inserted on the other end.

The left shift operator is usually written as "<<".

0010 << 1 → 0100

0010 << 2 → 1000

A single left shift multiplies a binary number by 2:

0010 << 1 → 0100

0010 is 2

0100 is 4

2. Logical Right Shifts

When shifting right with a logical right shift, the least-significant bit is lost and a 0

0 is inserted on the other end.

1011 >>> 1 → 0101

1011 >>> 3 → 0001

For positive numbers, a single logical right shift divides a number by 2, throwing out any remainders.
0101 >>> 1 → 0010

0101 is 5
0010 is 2

3. Arithmetic Right Shifts

When shifting right with an arithmetic right shift, the least-significant bit is lost and the most-significant bit is copied.
Languages handle arithmetic and logical right shifting in different ways. Java provides two right shift operators: >> does an arithmetic right shift and >>> does a logical right shift.
1011 >> 1 → 1101
1011 >> 3 → 1111

0011 >> 1 → 0001
0011 >> 2 → 0000

The first two numbers had a 1 as the most significant bit, so more 1's were inserted during the shift. The last two numbers had a 0 as the most significant bit, so the shift inserted more 0's.
If a number is encoded using two's complement, then an arithmetic right shift preserves the number's sign, while a logical right shift makes the number positive.

// Arithmetic shift
1011 >> 1 → 1101
1011 is -5
1101 is -3

// Logical shift
1111 >>> 1 → 0111
1111 is -1
0111 is 7