

浙江大学实验报告

课程名称：_____ 计算机系统原理 _____

实验项目名称：_____ 移码运算 _____

学生姓名：_____ 翁罗轩 _____ 专业：_____ 软件工程 _____ 学号：_____ 3170103226 _____

一、实验目的

1. 给出移码的算术运算算法，并编程实现。
2. 分析移码的字位扩展（8-16，16-32 位）、运算溢出、大小比较等的方法。

二、实验原理

2.1 四则运算的证明

令 $H = 2^{N-1}$ ，则 $[x]_{\text{移码}} = x + H$ 。

1. 加法

$$\begin{aligned}[x + y]_{\text{移码}} &= x + y + H \\ &= [x]_{\text{移码}} - H + [y]_{\text{移码}} - H + H \\ &= [x]_{\text{移码}} + [y]_{\text{移码}} - H\end{aligned}$$

```
INT madd(INT code1, INT code2) //add
{
    INT result = (code1 + code2) - HALFINT;
    return result;
}
```

2. 减法

$$\begin{aligned}[x - y]_{\text{移码}} &= x - y + H \\ &= [x]_{\text{移码}} - H - [y]_{\text{移码}} + H + H \\ &= [x]_{\text{移码}} + [y]_{\text{移码}} + H\end{aligned}$$

```
INT msub(INT code1, INT code2) //subtract
{
    INT result = (code1 - code2) + HALFINT;
    return result;
}
```

3. 乘法

$$\begin{aligned}[x * y]_{\text{移码}} &= x * y + H \\ &= ([x]_{\text{移码}} - H) * ([y]_{\text{移码}} - H) + H \\ &= [x]_{\text{移码}} * [y]_{\text{移码}} - H * ([x]_{\text{移码}} + [y]_{\text{移码}}) + H^2 + H\end{aligned}$$

```

INT mmul(INT code1, INT code2) //multiply
{
    INT result = code1 * code2 -
        HALFINT * (code1 + code2) +
        HALFINT * HALFINT + HALFINT;
    return result;
}

```

4. 除法（只考虑整除的情况）

$$\begin{aligned}
 [x / y]_{\text{移码}} &= x / y + H \\
 &= ([x]_{\text{移码}} - H) / ([y]_{\text{移码}} - H) + H
 \end{aligned}$$

```

INT mdiv(INT code1, INT code2) //divide
{
    INT result = (code1 - HALFINT) / (code2 - HALFINT) + HALFINT;
    return result;
}

```

2.2 字位扩展

首先分析一般情况，即从 N 位扩展到 M 位。

$$[x]_N = 2^{N-1} + x, [x]_M = 2^{M-1} + x$$

$$\therefore [x]_M = 2^{M-1} + [x]_N - 2^{N-1}$$

$$= [x]_N + (2^{M-N} - 1) * 2^{N-1}$$

$$\text{故 } [x]_{16} = [x]_8 + (2^8 - 1) * 2^7, [x]_{32} = [x]_{16} + (2^{16} - 1) * 2^{15}.$$

2.3 溢出判断

有符号数运算后，结果超出所能表示的范围，即为溢出；无符号数不考虑溢出。对于 N 位数，若数值小于 -2^{N-1} 或大于 $2^{N-1}-1$ ，则为溢出。

为找到移码运算的溢出判断条件，我们先从简单的例子出发。考虑 4 位的情况：

$$[x]_{\text{移码}} = x + 8, \therefore 0 \leq [x]_{\text{移码}} \leq 15 \therefore -8 \leq x \leq 7$$

$$1. [2]_{\text{移码}} + [4]_{\text{移码}} = 1010 + 1100 = 10110, \text{未溢出}$$

$$2. [2]_{\text{移码}} + [6]_{\text{移码}} = 1010 + 1110 = 11000, \text{溢出}$$

$$3. [-2]_{\text{移码}} + [-4]_{\text{移码}} = 0110 + 0101 = 01011, \text{未溢出}$$

$$4. [-2]_{\text{移码}} + [-7]_{\text{移码}} = 0110 + 0001 = 00111, \text{溢出}$$

可见，是否溢出取决于移码相加后的前两位符号位。

一般地，移码运算的溢出判断条件如下：

溢出	上溢	11
	下溢	00
未溢出	正	10
	负	01

2.4 大小比较

首先需明确，移码与补码只是符号位相反，其他位数字均相同。

1. 同号

正数与正数比较时，移码符号位都是 1，对于其他位，均与它们的补码相同。由于大的正数的补码数值位大于小的正数的补码数值位，因此只需

比较移码本身的大小关系，即为真值的大小关系。

负数与负数比较时，移码符号位都是 0，对于其他位，均与它们的补码相同。由于大的负数的补码数值位大于小的负数的补码数值位，因此只需比较移码本身的大小关系，即为真值的大小关系。

2. 异号

正数与负数比较时，由于正数移码的符号位是 1，负数移码的符号位是 0，因此移码的大小关系即为真值的大小关系。

综上所述，移码的大小关系与真值的大小关系相同。

三、程序使用方法

1. 运行程序，根据提示选择相应操作。

```
Choose the operation you want to take:
1.Change a string to frame shift(移码).
2.Change a frame shift into string.
3.Add.
4.Subtract.
5.Multiply.
6.Divide.
0.Exit.
```

2. 选择操作 1 时，输入一个数字，程序会自动将其转换为移码。

3. 选择操作 2 时，输入一个移码，程序会自动将其转换为原数字。

4. 选择操作 3 时，根据提示先后输入两个数字，程序会模拟移码下的加法运算。

5. 选择操作 4、5、6 时，与加法相同，先后输入两个数字，程序会模拟移码下的减法\乘法\除法运算。

四、测试结果

4.1 移码转换

```
Enter a string.
1250
84E2
```

 (正数)

```
Enter a string.
-125
7F83
```

 (负数)

4.2 移码还原

```
Enter a code.
84E2
1250
```

 (正数)

```
Enter a code.  
7F83  
-125
```

 (负数)

4.3 加法

```
Enter the first number.  
125  
Enter the second number.  
15  
125(807D) + 15(800F) = 140(808C)
```

 (正数加法)

```
Enter the first number.  
-125  
Enter the second number.  
-15  
-125(7F83) + -15(7FF1) = -140(7F74)
```

 (负数加法)

```
Enter the first number.  
125  
Enter the second number.  
-50  
125(807D) + -50(7FCE) = 75(804B)
```

 (正负数加法, 结果为正数)

```
Enter the first number.  
13  
Enter the second number.  
-45  
13(800D) + -45(7FD3) = -32(7FE0)
```

 (正负数加法, 结果为负数)

4.4 减法

```
Enter the first number.  
18  
Enter the second number.  
14  
18(8012) - 14(800E) = 4(8004)
```

 (正数减法, 结果为正数)

```
Enter the first number.  
25  
Enter the second number.  
32  
25(8019) - 32(8020) = -7(7FF9)
```

 (正数减法, 结果为负数)

```
Enter the first number.  
-16  
Enter the second number.  
-6  
-16(7FF0) - -6(7FFA) = -10(7FF6)
```

 (负数减法, 结果为负数)

```
Enter the first number.  
-56  
Enter the second number.  
-78  
-56(7FC8) - -78(7FB2) = 22(8016)
```

(负数减法, 结果为正数)

```
Enter the first number.  
15  
Enter the second number.  
-6  
15(800F) - -6(7FFA) = 21(8015)
```

(正负数减法, 结果为正数)

```
Enter the first number.  
-5  
Enter the second number.  
25  
-5(7FFB) - 25(8019) = -30(7FE2)
```

(正负数减法, 结果为负数)

4.5 乘法

```
Enter the first number.  
16  
Enter the second number.  
7  
16(8010) * 7(8007) = 112(8070)
```

(正数乘法)

```
Enter the first number.  
-3  
Enter the second number.  
-16  
-3(7FFD) * -16(7FF0) = 48(8030)
```

(负数乘法)

```
Enter the first number.  
23  
Enter the second number.  
-3  
23(8017) * -3(7FFD) = -69(7FBB)
```

(正负数乘法)

4.6 除法 (只考虑整除)

```
Enter the first number.  
48  
Enter the second number.  
3  
48(8030) / 3(8003) = 16(8010)
```

(正数除法)

```
Enter the first number.  
-45  
Enter the second number.  
-9  
-45(7FD3) / -9(7FF7) = 5(8005)
```

(负数除法)

```
Enter the first number.  
-72  
Enter the second number.  
8  
-72(7FB8) / 8(8008) = -9(7FF7)
```

（正负数除法）