计算机组成原理II —— 第一次作业

计科一班 王昊恩 3018216021

本次作业源代码已经上传至 `github` 。

`github` 链接：https://github.com/FireSSang/19_20_1_Computer_Organization

1. 本章书后习题 `2.63` 。

> 将下面的C函数代码补充完整。函数 `srl` 用算术右移（由值 `xsra` 给出）来完成逻辑右移，后面的其他操作不包括右移或者除法。函数 `sra` 用逻辑右移（由值 `xsrl` 给出）来完成算术右移，后面的其他操作不包括右移或者除法。可以通过计算 `8*sizeof(int)` 来确定数据类型 `int` 中的位数 w 。位移量 k 的取值范围为 $0 \sim w-1$ 。

```
unsigned srl(unsigned x,int k) {
    /*Perform shift arithmetically*/
    unsigned xsra = (int) x >> k;

    int w = 8 * sizeof(int);
    int mask = (1 << (w - k)) - 1; //前k位为0，后w-k位为1
    return xsra & mask; //前k位置为0，后k位不变
}

int sra(int x, int k) {
    /*Perform shift logically*/
    int xsrl = (unsigned) x >> k;

    int w = 8 * sizeof(int);
    int sign_mask = xsrl & (1 << (w - k - 1));
    //若符号位为0，该值为0
    //否则为000...001000...000形式
    int mask = ~((sign_mask << 1) - 1);
    //如果符号位为1，mask为111...1100000...000000
    //如果符号位为0，mask为000...0000000...000000
    return xsrl | mask; //前k为改为mask前k位，实现符号位扩展
}
```

代码实现如下（源代码在 `github` 仓库中）：

```
#include <iostream>
using namespace std;

unsigned srl(unsigned x,int k) {
    /*Perform shift arithmetically*/
    unsigned xsra = (int) x >> k;

    int w = 8 * sizeof(int);
    int mask = (1 << (w - k)) - 1; //前k位为0，后w-k位为1
    return xsra & mask; //前k位置为0，后k位不变
}

int sra(int x, int k) {
    /*Perform shift logically*/
    int xsrl = (unsigned) x >> k;

    int w = 8 * sizeof(int);
    int sign_mask = xsrl & (1 << (w - k - 1)); //若符号位为0，该值为0
    //否则为000...001000...000形式
    int mask = ~((sign_mask << 1) - 1);
    //如果符号位为1，mask为111...1100000...000000
    //如果符号位为0，mask为000...0000000...000000
    return xsrl | mask; //前k为改为mask前k位，实现符号位扩展
}

int main()
{
    int date = 19200000 + 60817; //一个比较大的常用素数
    cout << (date >> 7) << endl;
    cout << srl(date, 7) << endl;
    cout << sra(date, 7) << endl;

```

```
    int datee = -1 * date;
    cout << (datee >> 7) << endl;
    cout << srl(datee, 7) << endl;
    cout << sra(datee, 7) << endl;

    return 0;
}
```

```
"C:\MyFiles\19-20 1\19_20_1_Computer_Organization\homeworks\homework1_1\cmake-build-debug\homework1_1.exe"
150475
150475
150475
-150476
33403956
-150476

Process finished with exit code 0
```

2. 假设尾数部分位宽为 `n` ，阶码位宽为 `k` 。请给出最小/最大非规格化负数、最小/最大规格化整数、 `-1` 的数值表达式。

名称	数值表达式
最小非规格化负数	$(-1) \times (1 - 2^{-n}) \times 2^{-2^{k-1}+2}$
最大非规格化负数	$(-1) \times 2^{-n-2^{k-1}+2}$
最小规格化整数	$(-1) \times (1 - 2^{-n-1}) \times 2^{2^{k-1}}$
最大规格化整数	$(1 - 2^{-n-1}) \times 2^{2^{k-1}}$
-1	$(-1) \times (2^0) \times 1$

3. 本章书后习题 `2.87` 。

> 2008版 `IEEE` 浮点标准，即 `IEEE 754-2008` ，包含了一种 16 位的"半精度"浮点格式。他最初是由计算机图形公司设计的，其存储的数据所需的动态范围要高于 16 位整数可获得的范围。这种格式具有 1 个符号位、 5 个阶码位 $(k = 5)$ 和 10 个小数位 $(n = 10)$ 。阶码偏置量是 $2^{5-1} - 1 = 15$ 。
>
> 对于每个给定的数，填写下表，其中，每一列具有如下指示说明：
>
> Hex：描述编码形式的 4 个十六进制数字。
>
> M：尾数的值。这应该是一个形如 x 或 $\frac{x}{y}$ 的数，其中 x 是一个整数，而 y 是2的整数幂。例如： 0 、 $\frac{67}{64}$ 和 $\frac{1}{256}$ 。
>
> E：阶码的整数值。
>
> V：所表示的数字值。使用 x 或 $x \times 2^z$ 表示，其中 x 和 z 都是整数。
>
> D：（可能近似的）数值，用 `printf` 的格式规范 `%f` 打印。
>
> 举一个例子，为了表示数 $\frac{7}{8}$ ，我们有 $s = 0$ ， $M = \frac{7}{4}$ 和 $E = -1$ 。因此这个数的阶码字段为 01110_2（十进制值 $15 - 1 = 14$ ），尾数字段为 1100000000_2 ，得到一个十六进制的表示 $3B00$ 。其数值为 0.875 。
>
> 标记为"—"的条目不用填写。

以下 D 输出已经过验证，浮点数默认保留六位小数。

描述	Hex	M	E	V	D
-0	8000	0	-14	-0	-0.0
最小的>2的值	4001	$\frac{1025}{1024}$	1	$\frac{1025}{512}$	2.001953
512	6000	0	9	512	512.0
最大的非规格化数	$07FF$	$\frac{1023}{1024}$	-14	$\frac{1023}{2^{24}}$	0.000061
$-\infty$	$FC00$	—	—	$-\infty$	$-\infty$
十六进制表示为 $3BB0$ 的数	$3BB0$	$\frac{123}{64}$	-1	$\frac{123}{128}$	0.960938

4. 请解释为什么在C程序中 $0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1 \neq 1$ ，而 $0.2+0.2+0.2+0.2+0.2=1$ 。

在C程序中， 0.1 和 0.2 都无法用二进制精确表示。

在 `IEEE 754` 标准下， 0.1 的二进制存储值为：$0.0001100110011001100110011001100110011001100110011001100110011001$

0.2 的二进制存储值为： $0.0011001100110011001100110011001100110011001100110011001100110011$

所以

$$10*0.1=0.1111111111111111111111111111111111111111111111111111111111111010$$
$$5*0.2=0.1111111111111111111111111111111111111111111111111111111111111111$$

可以发现两式结果在最后三位有着微小差异，正是这一点差异使结果转换为十进制后结果不同。

在程序中实测如下：

```cpp
#include <bits/stdc++.h>
#define ll long long
#define INIT(a, b) memset(a, b, sizeof(a))
using namespace std;
const int maxn = 1e9 + 7;

int main()
{
    printf("%.18lf\n", 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1+ 0.1);
    printf("%.18lf\n", 0.2 + 0.2 + 0.2 + 0.2 + 0.2);
    return 0;
}
```

```
"C:\Users\79377\Desktop\TJU 2019\TJU 2019 Autumn\2019 [National Day] Contest 2\B\cmake-build-debug\B.exe"
0.999999999999999889
1.000000000000000000

Process finished with exit code 0
```