第十章 运算方法和运算器

第 12 周-课后思考题： 第十章 运算方法和运算器（1—后小半节课）

（该章很多内容较繁琐，课时关系删减了很多，大家复习时只需要关注讲过的部分）

说明：学习本章时重点是从运算器（硬件角度理解运算），而不是我们自己用笔和纸去做运算，注意两者之间的联系和区别。

1、 掌握逻辑移位、算术移位（重点是补码移位，**符号位要一起参与移位**）和循环移位的区别，分别在什么情况下使用，算术移位什么情况下会发生**溢出**（Overflow）？

课后在 C 语言中 char s = -1 和 unsigned char=255，分别进行右移运算，观察有什么区别，从而体会什么时候用逻辑移位，什么时候用算术移位。

例如：unsigned char s = 255; //s=0xFF
s >>4; //s=0x0F

例如： char s = -1; //s=0xFF
s >>4; //s=0xFF

Java 有不同的右移符号>>>和>>，python 怎么处理的？python 数据类型宽度可变，不存在溢出，默认都是算数移位。左移为什么不需要区别算数和逻辑移位？

解释下面运行结果：

Java：

```
-1>>1:-1
-1>>>1:2147483647
255>>1:127
255>>>1:127
```

Python：

```
>>> 4>>2
1
>>> -1>>2
-1
>>> 0b11001100>>1
102
>>> format(102,'08b')
'01100110'
```

第 13 周-课后思考题： 第十章 运算方法和运算器（2）

1、 什么时候关注**溢出**？掌握定点数加（减）法运算，为什么会有溢出？如何判断溢出，教材中介绍的 3 中判溢出方法的优缺点是什么？为什么实际中常用**变形补码**法？

例 2.28 已知 32 位寄存器 R1 中存放的变量 x 的机器数为 8000 0004H，请回答下列问题。

① 当 x 是 unsigned int 类型时，x 的值是多少？$x/2$ 的值是多少？$x/2$ 的机器数是什么？$2x$ 的值是多少？$2x$ 的机器数是什么？

② 当 x 是 int 类型时，x 的值是多少？$x/2$ 的值是多少？$x/2$ 的机器数是什么？$2x$ 的值是多少？$2x$ 的机器数是什么？

2、 有符号数运算发生溢出时，硬件会置 OF 为 1，C、java、python 等语言的处理方式分别是什么？

C语言（静默环绕）

```c
#include <stdio.h>
#include <limits.h>

int main() {
    int max = INT_MAX;  // 通常 2147483647
    printf("max = %d\n", max);
    printf("max + 1 = %d\n", max + 1);  // 静默环绕到 INT_MIN

    int min = INT_MIN;  // 通常 -2147483648
    printf("min = %d\n", min);
    printf("min - 1 = %d\n", min - 1);  // 静默环绕到 INT_MAX

    return 0;
}
```

Java（静默环绕但可检测）

```java
public class WrapExample {
    public static void main(String[] args) {
        int max = Integer.MAX_VALUE;  // 2147483647
        System.out.println("max = " + max);
        System.out.println("max + 1 = " + (max + 1));  // 静默环绕到最小值

        // 但有安全的方法可以检测
        try {
            int safe = Math.addExact(max, 1);  // 抛出异常
        } catch (ArithmeticException e) {
            System.out.println("溢出被捕获了！");
        }
    }
}
```

3、 掌握原码位乘法（串行乘）采用加法器和移位器、计数器的运算过程，理解为什么做了两个修正后就便于硬件实现了？具体是哪两个修正？能根据图 10.8 完成硬件的乘运算过程；

4、 图 10.9 了解它是完成乘法的**硬件串行乘法器**即可，不用深究内部细节。

5、 有了按位相乘乘法器，如何提高乘法器的效率？

提示：回忆加法器是怎么提高效率的？多位数据相加，为了提高计算速度在硬件上就由串行加法器（将全加器串联）改进成并行加法器（也叫超前加法器，加法器作为典型的组合电路，有什么输入组合就会直接得到输出结果，天生的并行性）。

6、 了解定点运算器是由 ALU、寄存器组和数据总线和辅助电路构成；

7、 了解 ALU 的设计思路；

8、 了解多地址多数据端口寄存器组的优点；

9、 了解多总线实现运算器的优点，体会什么时候需要增加**缓冲器**（也叫缓存器）；

10、 多端口寄存器组往往和多总线结构的 ALU 相对应。

11、 将图 10.25（b）的多端口寄存器与第 8 章图 8.5 中的（d）通用寄存器组做比较，我们前面用的就是这种多端口的寄存器组。

12、 计算机为什么要采用浮点方式记录小数？浮点计数中，阶码位数和尾数位数都有什么含义？（回顾第二章相关内容）浮点加（减）法运算包括哪五步？会具体计算（判 0、对阶、尾数运算、规格化和判溢出（是对阶码做判断）、舍入）

13、 了解常用的四种舍入方法的不同。（0 舍 1 入（类似于十进制的四舍五入）、截断、向+∞、向-∞）

14、 例 10.13 中阶码为什么采用变形补码？判断溢出是针对阶码还是尾数？既然是对阶码判断溢出，为什么对尾数也采用变型补码（为了记录最高位数值运算的进位位）？规格化的同时要验证阶码溢不溢出。

课后有同学问：如果 10.13 浮点用 IEEE754 标准怎么做？浮点加减运算的步骤还是一样的，只是细节要根据 754 标准记录方式变通，感兴趣的同学可想想具体怎么做的？

作业： 10.1、10.2-10.5（都只做（1））、10.8、10.12（只做（1））

编程练习（不用提交）：

用任何你熟悉的语言实现两位十六进制数的交换，如 0xAB，变为 0xBA，可以尝试用 x86 的循环移位指令（如 ROL 或 ROR），比较用低级语言和高级语言实现的区别和效能。

课外作业（不用交）：

学完算数逻辑运算器后尝试用 4 位 ALU 芯片 74LS181 构建 8 位、16 位或 32 位 ALU。