平时作业1

姓名：孟启轩 学号：2212452 专业：计算机科学与技术

问题1：浮点数有没有移位操作和扩展操作?为什么？fscale

浮点数本身没有直接的移位操作符，因为其存储格式由符号位、指数、尾数组成，直接移位会破坏数值结构。但可通过间接方法实现类似效果：将浮点数的内存解释为整型，再对整型进行移位操作，或使用汇编指令（如FSCLAE）通过调整指数实现乘以2^N的效果，等效于逻辑移位。

浮点数的扩展包括精度扩展和数值范围扩展。精度扩展通过类型转换或自动提升实现，利用更高位数的尾数提升精度；数值范围扩展则通过指数调整（如FSCLAE）或运算实现，扩大表示范围。

FSCLAE是x86汇编中的浮点运算指令，用于将浮点数乘以2^N，通过直接修改指数字段实现，等效于对浮点数的二进制位进行逻辑移位。

问题2：为什么整数除0会发生异常而浮点数除0不会?

整数除法基于数学规则，除以0在数学上无定义，因此直接触发运行时异常，强制程序停止以避免非法操作。浮点数运算遵循IEEE 754标准，规定除以0的结果为特殊值（如正无穷大或负无穷大，若0的符号不同；0/0则为NaN），这些值合法存在且可继续参与计算，从而避免程序崩溃，确保浮点运算的连续性。

问题3：为什么同一个实数赋值给float型变量和double型变量，输出结果会有所不同呢?

例：将同一实数分别赋值给单精度和双精度类型变量，然后打印输出。

```
#include <stdio.h>
main()
{
        float a;
        double b;
        a = 123456.789e4;
        b = 123456.789e4;
        printf( “%f/n%f/n” ,a,b);
}
```

运行结果如下：

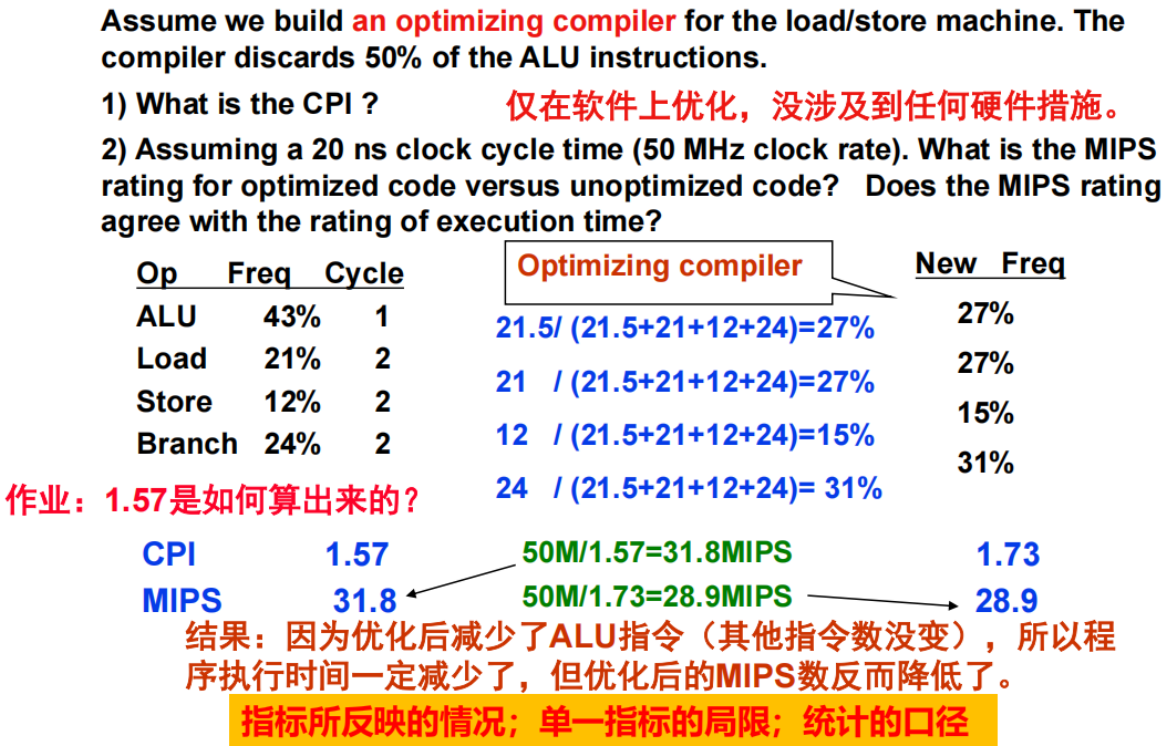
```
1234567936.000000
1234567890.000000
```

为什么float情况下输出的结果会比原来的大？这到底有没有根本性原因还是随机发生的？为什么会出现这样的情况？

问题：为什么同一个实数赋值给float型变量和double型变量，输出结果会有所不同呢?

根本原因在于float 和 double 类型的内存分配和精度限制不同。float 是 32 位（单精度），尾数部分占 23 位，有效数字为 6~7 位；而 double 是 64 位（双精度），尾数部分占 52 位，有效数字为 15~16 位。当实数的有效位数超过 float 的精度时，它会因存储空间不足而丢失部分精度（通过舍入或截断），导致存储值与原值存在偏差；而 double 因为尾数位数更多，能更精确地保留原值的细节。

问题4：1.57是如何算出来的?



要计算平均CPI，我们需要计算的是每种类型的指令的平均执行周期数，并加权求和。

根据题目给的信息，对每一种操作的占比和所占周期数进行加权求和： CPI = 43%x1 + 21%x2 + 12%x2 + 24%x2 = 1.57