PA1 实验报告

PA1-1 数据在计算机内的存储：

C语言中的`struct`和`union`关键字都是什么含义，寄存器结构体的参考实现为什么把部分`struct`改成了`union`？

答：struct是定义一个结构体的关键字，在C语言中，可以使用**结构体（Struct）**来存放一组不同类型的数据。

结构体的定义形式为：

```
struct 结构体名{
    结构体所包含的变量或数组
};
```

结构体是一种集合，它里面包含了多个变量或数组，它们的类型可以相同，也可以不同，每个这样的变量或数组都称为结构体的**成员（Member）**。

在C语言中，还有另外一种和结构体非常类似的语法，叫做**联合（Union）**，它的定义格式为：

```
union 联合体名{
    成员列表
};
```

结构体和共用体的区别在于：结构体的各个成员会占用不同的内存，互相之间没有影响；而共用体的所有成员占用同一段内存，修改一个成员会影响其余所有成员。

在reg.h中，我们利用数据结构来模拟计算机中寄存器的表示。为了能够访问寄存器的32位，16位，高8位和低8位，我们定义了变量 _32 ， _16 和数组 _8[2] 。显然，为了使每次能够正确访问到寄存器中的对应内容，这些变量应该是共享一块内存空间的，如果使用struct来定义，他们会占用不同的空间。所以使用联合体才正确。

```
union
{
    union
    {
        union
        {
            uint32_t _32;
            uint16_t _16;
            uint8_t _8[2];
        };
        uint32_t val;
    } gpr[8];
    struct
    { // do not change the order of the registers
        uint32_t eax, ecx, edx, ebx, esp, ebp, esi, edi;
    };
};
```

PA1-3 浮点数的表示和运算：

为浮点数加法和乘法各找两个例子：

1）对应输入是规格化或非规格化数，而输出产生了阶码上溢结果为正（负）无穷的情况；

2）对应输入是规格化或非规格化数，而输出产生了阶码下溢结果为正（负）零的情况。

是否都能找到？若找不到，说出理由。

加法：

1）能找到：比如：

7F000000（2^{127}）+ 7F000000（2^{127}）阶码上溢为正无穷

FF000000（-2^{127}）+ FF000000（-2^{127}）阶码上溢为负无穷

2）不能找到。

因为无论输入是规格化数还是非规格化数，其加法运算的最小尺度都是$2^{-126} * 2^{-23} = 2^{-149}$

在加减过程中，最后的结果都可以以 2^{-149} 为单位表示，所以不会发生阶码下溢结果为正（负）零的情况

乘法：

1）能找到：比如：

7F000000（2^{127}）* 7F000000（2^{127}）阶码上溢为正无穷

7F000000（2^{127}）* FF000000（-2^{127}）阶码上溢为负无穷

2）能找到：比如：

00400000（2^{-127}）* 00400000（2^{-127}）阶码下溢为正0

00400000（2^{-127}）* 80400000（-2^{-127}）阶码下溢为负0

实验中遇到的问题：

（1）adc的CF进位：出现边界情况res == src 时，进位输入CF = 1和CF = 0是两个结果

（2）sub：可以用减数取补之后加上被减数来实现，但是减法与加法的CF相反

（3）整数比较时，逻辑上dest <= src 与 dest - 1 < src 等价，但是做运算dest - 1可能会得到无法预料的结果，所以用前者更能降低出错概率

（4）浮点数尾数规格化时，尾数为0可能进入死循环，需要单独处理