PA 1 实验报告

- **Author:** 张昱东
- **SID:** 141120159
- **Email:** 141120159@smail.nju.edu.cn

- **Time:** 4月12日

一、实验进度

完成了所有必做内容:

- PA 1-1 数据在计算机内的存储
  - [x] 通用寄存器模拟
- PA 1-2 整数的表示、存储和运算
  - [x] 加/减操作
  - [x] 移位操作
  - [x] 逻辑运算
  - [x] 乘除运算
- PA 1-3 浮点数的表示和运算
  - [x] 加/减运算
  - [x] 乘/除运算
  - [x] 尾数规格化

二、实验结果

```
daniel@ruby:~/Documents/pa2018_spring/nemu$ ./nemu

======== reg test ========
reg_test()      pass

======== alu test ========
alu_test_add()  pass
alu_test_adc()  pass
alu_test_sub()  pass
alu_test_sbb()  pass
alu_test_and()  pass
alu_test_or()   pass
alu_test_xor()  pass
alu_test_shl()  pass
alu_test_shr()  pass
alu_test_sal()  pass
alu_test_sar()  pass
alu_test_mul()  pass
alu_test_div()  pass
alu_test_imul() pass
alu_test_idiv() pass

======== fpu test ========
fpu_test_add()  pass
fpu_test_sub()  pass
fpu_test_mul()  pass
fpu_test_div()  pass
```

三、必答题

1. 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 ?

    - struct 和 union 都是数据结构，可以包含其他数据结构和变量作为成员
    - 二者区别是：union 中的所有成员都共用一个内存空间，在不同的时间可以保存不同的数据类型和不同长度的变量；而 struct 中的每个成员则拥有不同的内存空间
    - 寄存器结构体使用 union 的原因是：在 cpu 中，实际上 32 位的通用寄存器 EAX 等和 16 位通用寄存器 AX 等共享同样的存储空间，只是在处理不同长度的变量时使用相应的寄存器。因此我们需要使用 union 而不是 struct 来模拟这

一机制

2. 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:
        a. 阶码上溢: $1.11 \times 2^{255-127} + 1.01 \times 2^{255-127}$, 此时两个数的阶码已达到最大 (255)，不需要对阶, 而尾数 (加上隐藏位后) 相加得到 $1.11 + 1.01 = 11.00 = 1.10 \times 2^1$, 因此结果需要右规，导致阶码上溢
        b. 阶码下溢：$1.111 \times 2^{0-127} + (-1)1.110 \times 2^{0-127}$, 输入的两个数为非规格化数，因此需要右规一次，结果为 $0.1111 - 0.1110 = 0.0001 = 1.0 \times 2^{-4}$, 需要左规，导致阶码下溢
            - 如果严格来说的话，这个算是减法。如果是两个正数的加法，则不会产生阶码下溢。因为正数相加结果肯定都比两个加数大，不可能因为结果太小而超出浮点数表示范围。
    - 乘法:
        a. 阶码上溢: $1.10 \times 2^{255-127} * 1.01 \times 2^{255-127} = 1.111 \times 2^{\exp}$, 结果的阶码为$\exp = 255 + 255 - 127 - 20 > 255$，因此阶码上溢
        b. 阶码下溢: $1.10 \times 2^{3-127} * 1.01 \times 2^{3-127} = 1.111 \times 2^{\exp}$, 结果的阶码为 $\exp = 3 + 3 - 127 - 20 < -126$，因此导致阶码下溢

四、个人收获

- 通过代码实践，对于寄存器的结构、整数和浮点数的运算规则以及注意事项有了更深的理解。

- 实验过程中不少 bug 出现在标志位的设置上。

    - 有的标志位设置错误是因为没有理解清楚定义，比如 PF，一开始的理解是整个数里1的个数，直到后来找不出bug去查阅手册才找到真正的定义 （手册还是要多翻！）
    - 还有的标志位设置出错是因为实现方式存在的一些缺陷，比如在实现 ADC 的时候，src+dest+CF 是作为一次来计算的，我的实现方式调用了之前实现的 ADD，把这个过程拆成两步来计算，但是这样每一步都会改变标志位。后来是通过把每一步的标志位记录下来，最后一起判断来解决的

- 另外要给汪老师的 ppt 点个赞，讲得非常清楚。一开始写的时候无从下手，强行开始写导致结构很混乱。后来卡在某个 bug 上苦思不得其解的时候去翻了各种课程资料，发现了 ppt 里的讲解，于是给后来改进代码组织方式提供了很多帮助。