计算机系统基础
Programming Assignment

PA 1 数据的表示、存取和运算

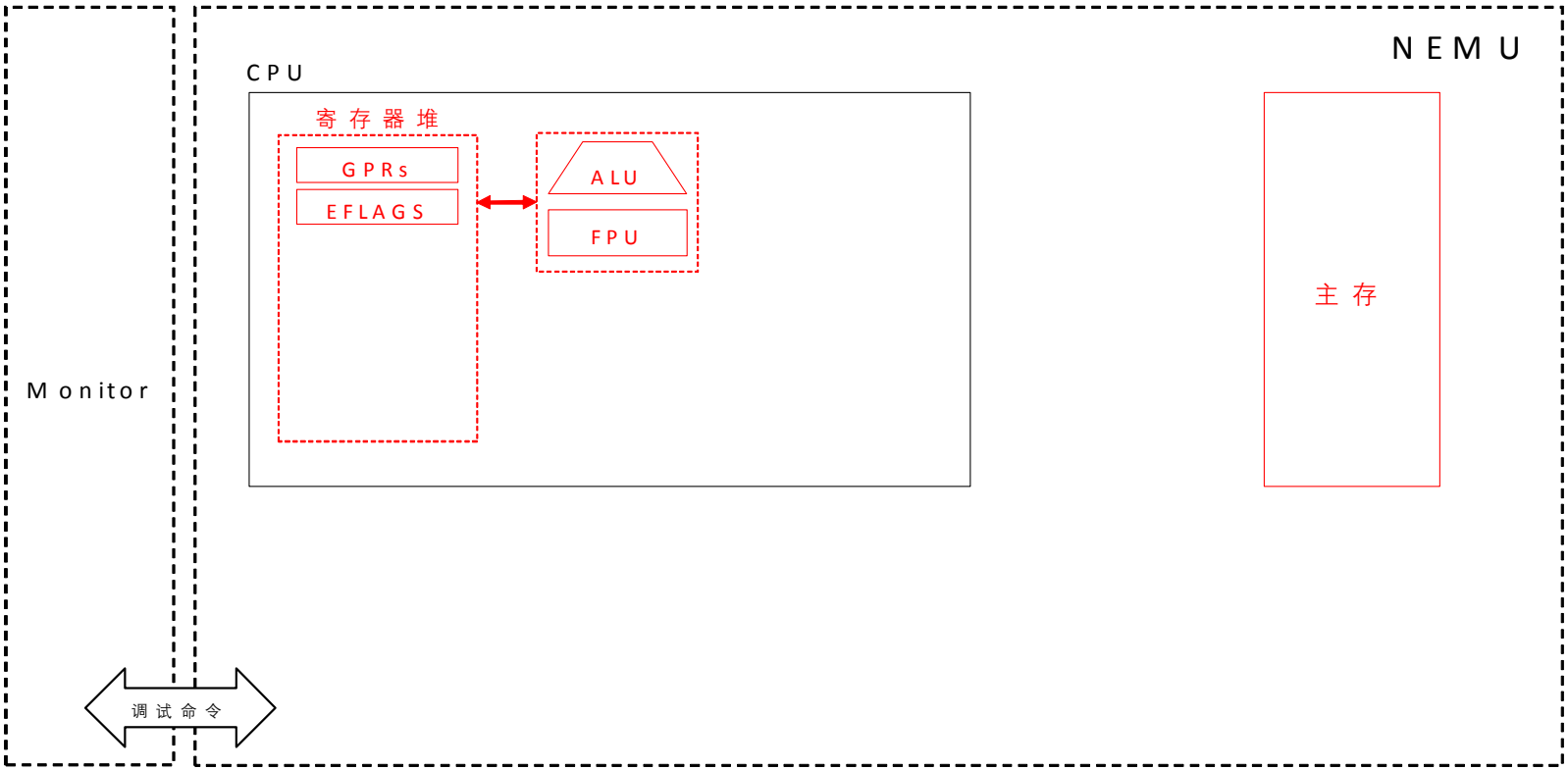
2019年9月11日

PA 1 – 目录

- PA 1-1 数据的类型和存取
- PA 1-2 整数的表示和运算
- PA 1-3 浮点数的表示和运算

PA 1-2 整数的表示和运算

PA 1 – 路线图



PA 1-2 整数的表示和运算

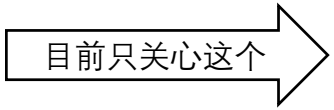
- 整数
  - 无符号整数
    - 32位整数：0x0 ~ 0xFFFFFFFF（32个1）
  - 带符号整数（归于某种无符号整数的表示方法）
    - 原码表示法：最高位为符号位
    - 补码表示法（普遍采用）
      - 各位取反末位加一
      - 用加法来实现减法
      - 如何方便地理解补码地运算方式？

PA 1-2 整数的表示和运算

- 整数
  - 无符号整数
    - 32位整数：0x0 ~ 0xFFFFFFFF（32个1）
  - 带符号整数（归于某种无符号整数的表示方法）
    - 原码表示法：最高位为符号位
    - 补码表示法（普遍采用）
      - 各位取反末位加一
      - 用加法来实现减法
      - 可以试试X+(-X)等于多少，其中X为某一32位正整数，-X为其补码表示，运算结果截取低32位

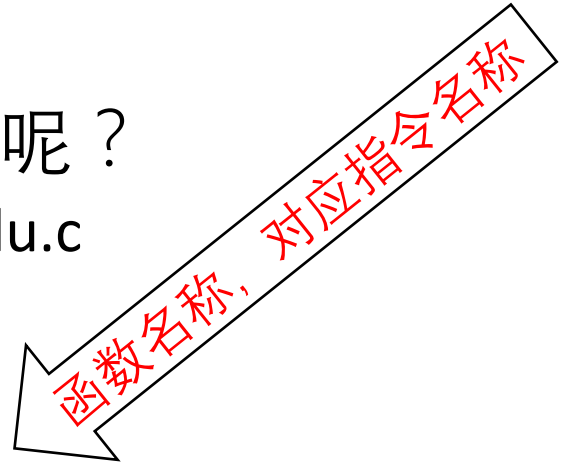
PA 1-2 整数的表示和运算

- 在NEMU中模拟各类整数运算的部件
  - ALU – 算术逻辑单元
    - 能进行各类算术运算：加减乘除、移位
    - 能进行各种逻辑运算：与或非
    - 对应代码：nemu/src/cpu/alu.c
  - 得到运算结果的同时设置标志位
    - EFLAGS
      - 指示运算结果的标志位：CF, PF, AF, ZF, SF, OF, …
      - 指示机器状态的标志位：IF, …
      - 对应代码：nemu/include/cpu/reg.h

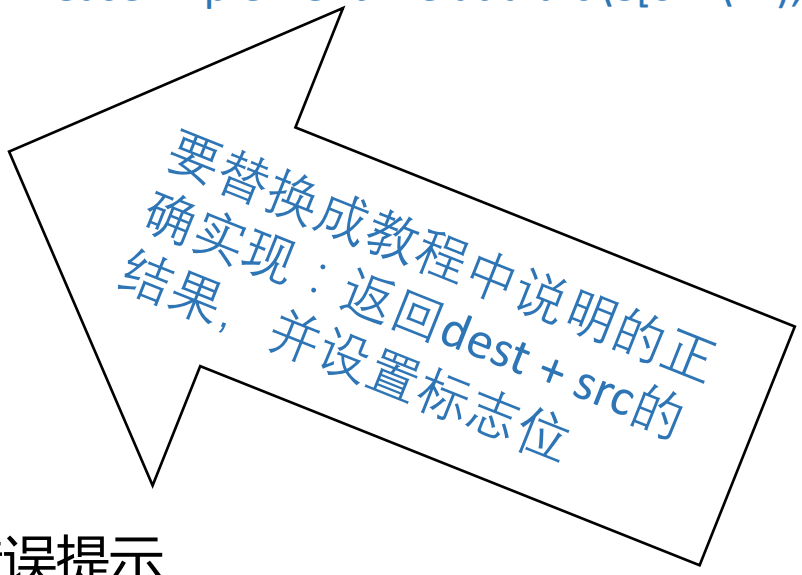


PA 1-2 整数的表示和运算

- 怎么来完成模拟呢？
  - nemu/src/cpu/alu.c
  - 分四步




```
uint32_t alu_add(uint32_t src, uint32_t dest, size_t data_size) {
        printf("\e[0;31mPlease implement me at alu.c\e[0m\n");
        assert(0);
        return 0;
}
```



第一步：找到需要实现的函数
执行./nemu/nemu遇到错误提示

PA 1-2 整数的表示和运算

- 怎么来完成模拟呢？
  - nemu/src/cpu/alu.c
  - 找到i386手册Sec. 17.2.2.11
    - 有关ADD指令的描述 p.g. 261 of 421
    - 看Flags Affected: OF, SF, ZF, AF, CF, and PF as described in Appendix C
    - 找到Appendix C并仔细体会（AF不模拟）

第二步：掏出i386手册

PA 1-2 整数的表示和运算

- 怎么来完成模拟呢？

```
uint32_t alu_add(uint32_t src, uint32_t dest, size_t data_size) {
        uint32_t res = 0;
        res = dest + src;

        set_CF_add(res, src, data_size);
        set_PF(res);
        // set_AF(); 我们不模拟AF
        set_ZF(res, data_size);
        set_SF(res, data_size);
        set_OF_add(res, src, dest, data_size);

        return res & (0xFFFFFFFF >> (32 - data_size));
}
```

nemu/src/cpu/alu.c
add的参考实现方案

第三步：按照手册规定的操作进行实现

```
// CF contains information relevant to unsigned integers
void set_CF_add(uint32_t result, uint32_t src, size_t data_size) {
        result = sign_ext(result & (0xFFFFFFFF >> (32 - data_size)), data_size);
        src = sign_ext(src & (0xFFFFFFFF >> (32 - data_size)), data_size);
        cpu.eflags.CF = result < src;
}

void set_ZF(uint32_t result, size_t data_size) {
        result = result & (0xFFFFFFFF >> (32 - data_size));
        cpu.eflags.ZF = (result == 0);
}

// SF and OF contain information relevant to signed integers
void set_SF(uint32_t result, size_t data_size) {
        result = sign_ext(result & (0xFFFFFFFF >> (32 - data_size)), data_size);
        cpu.eflags.SF = sign(result);
}

void set_PF(uint32_t result) { … } // 简单暴力穷举也行
```

```
void set_OF_add(uint32_t result, uint32_t src, uint32_t dest, size_t data_size) {
        switch(data_size) {
                case 8:
                        result = sign_ext(result & 0xFF, 8);
                        src = sign_ext(src & 0xFF, 8);
                        dest = sign_ext(dest & 0xFF, 8);
                        break;
                case 16:
                        result = sign_ext(result & 0xFFFF, 16);
                        src = sign_ext(src & 0xFFFF, 16);
                        dest = sign_ext(dest & 0xFFFF, 16);
                        break;
                default: break;// do nothing
        }
        if(sign(src) == sign(dest)) {
                if(sign(src) != sign(result))
                        cpu.eflags.OF = 1;
                else
                        cpu.eflags.OF = 0;
        } else {
                cpu.eflags.OF = 0;
        }
}
```

注意事项

- ALU函数传入的参数都是无符号整型，是机器数
- CF和OF在设置时，分别将传入的机器数看作无符号数与有符号进行处理，无需关注其实际想表达什么值（真值）

PA 1-2 整数的表示和运算

- 怎么来完成模拟呢？
  - nemu/src/cpu/alu.c
  - 实现完，保存alu.c
  - 执行make编译
  - 执行./nemu/nemu --test-alu add （每次替换add为其它）
  - 或执行 make test_pa-1
  - 测试用例位于nemu/src/cpu/test/alu_test.c

```
alu_test_add()  pass
```

第四步：执行测试用例

框架代码的执行过程

- 从make test_pa-1开始讲解
- 课堂实际演示

PA 1-2 整数的表示和运算

~PA 1-2顺利完成~

- 实验的要求
  - 把教程中提到的所有运算函数都实现，通过测试用例
  - 参见教程§1-2.3 实验过程及要求

```
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
```

注意：移位操作不测试OF位
imul操作所有标志位都不测试

注意：禁止采用测试用例里面使用内联汇编进行实现的方法，但是可以学习这种交叉验证的思想

第四步：执行测试用例

PA 1-2 整数的表示和运算

- 实现ALU的目的
  - 复习课本第二章内容
  - 在alu.c中实现的这些函数，到了PA 2实现对应指令的时候，就可以直接调用了

截止时间

- PA 1不设置小的阶段截止
- 建议PA 1-2在一周内完成

祝大家学习快乐，身心健康！

欢迎大家踊跃参加问卷调查