计算机系统基础
Programming Assignment

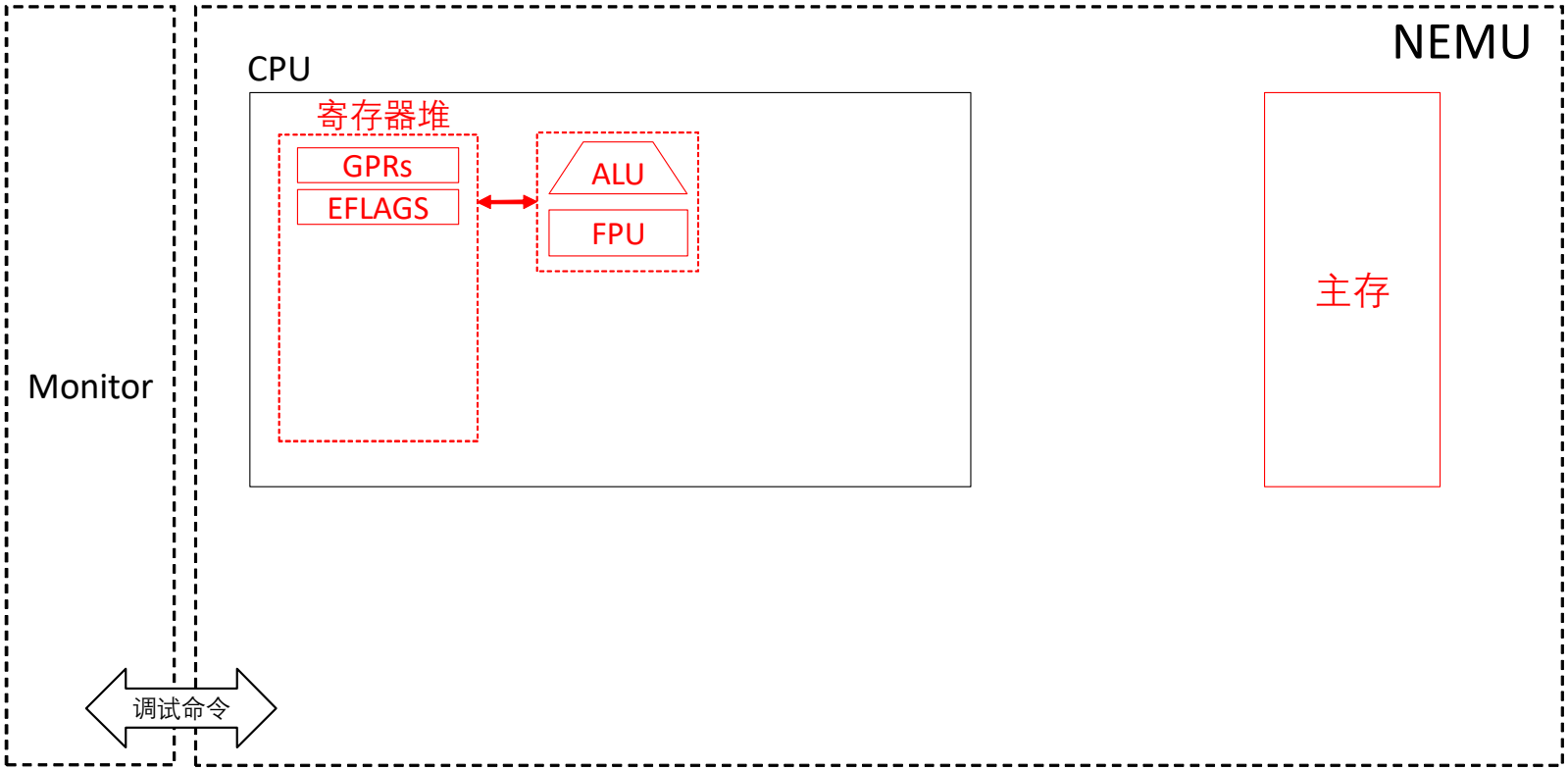
PA 1 数据的表示、存取和运算

2018年9月5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 – 路线图



PA 1-1 数据的类型和存取

PA 1-1 数据的类型和存取

- 数据基本可以分为两个大类
  - 非数值型数据
    - 各类文字的编码
    - 指令的操作码
  - 数值型数据（重点关注）
    - 整数
    - 浮点数
  - 在计算机内，所有类型的数据（包括代码）都表现为01串

PA 1-1 数据的类型和存取

- NEMU模拟的是i386体系结构
  - 数据存储的最小单位是比特（bit）
  - 数据存储的基本单位是字节（byte）
    - 典型长度为：1字节、2字节、4字节
  - 内存中采用小端方式存储
    - 对于超过一个字节的数据
    - 低有效位的字节在前（低地址），高有效位的字节在后（高地址）

想表达的数字	内存中的数字
0x12345678	0x78 0x56 0x34 0x12

低地址　　高地址

PA 1-1 数据的类型和存取

“巧妇难为无米之炊，没有锅她也不行。”

—— L. Wang

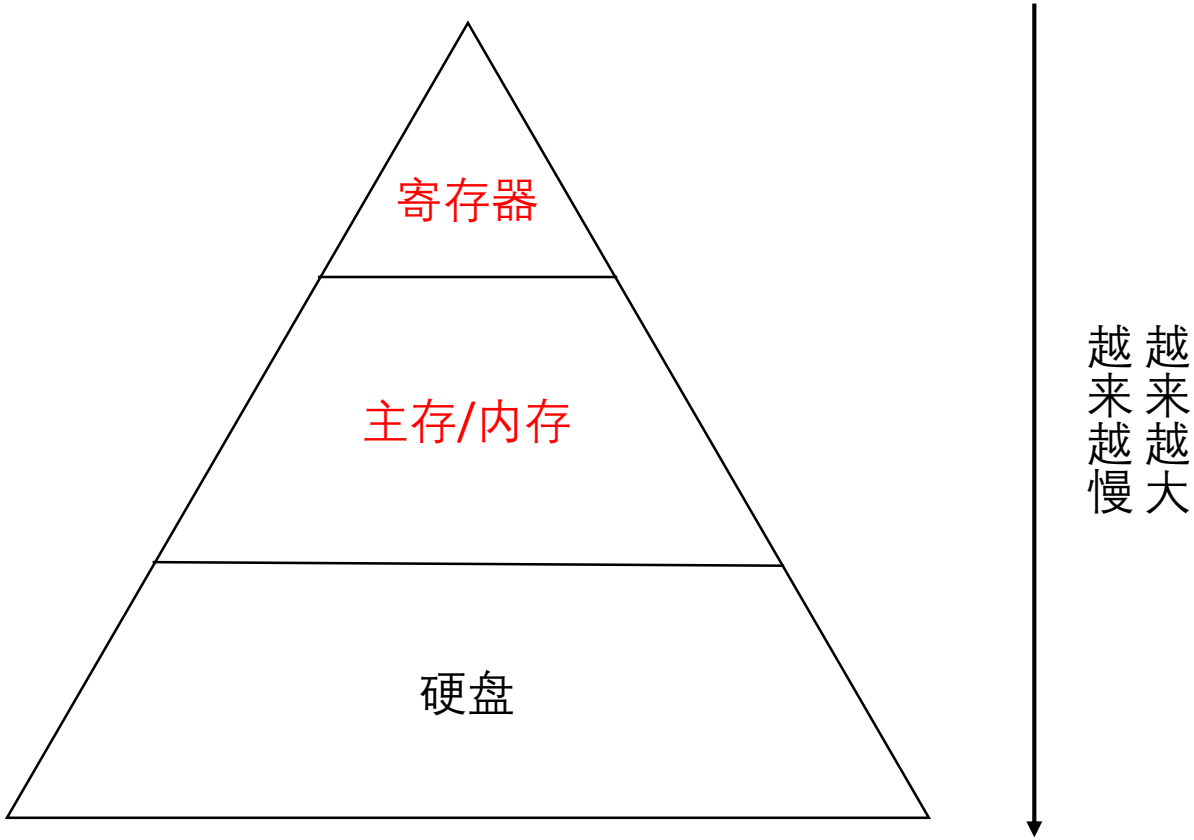
我们首先要为待处理的各类数据找到摆放它们的场所，在计算机中，这些场所就是各种类型的存储器

PA 1-1 数据的类型和存取

	计算机	餐厅	
解决计算问题的步骤	程序	菜谱	做菜的步骤
程序处理的对象	数据	食材	做菜加工的对象
执行程序的器件	CPU	大厨	执行菜谱的人
储存正在处理的数据	CPU内部的寄存器	灶台上的锅	放置正在加工的食材
储存马上要用的数据	主存	厨房里的冰箱	储存马上用的菜谱和食材
存储大量的数据	硬盘	仓库	啥都放这里

如果将一个计算机比作是一个餐厅，
那么所有概念大概都能找到一个对应

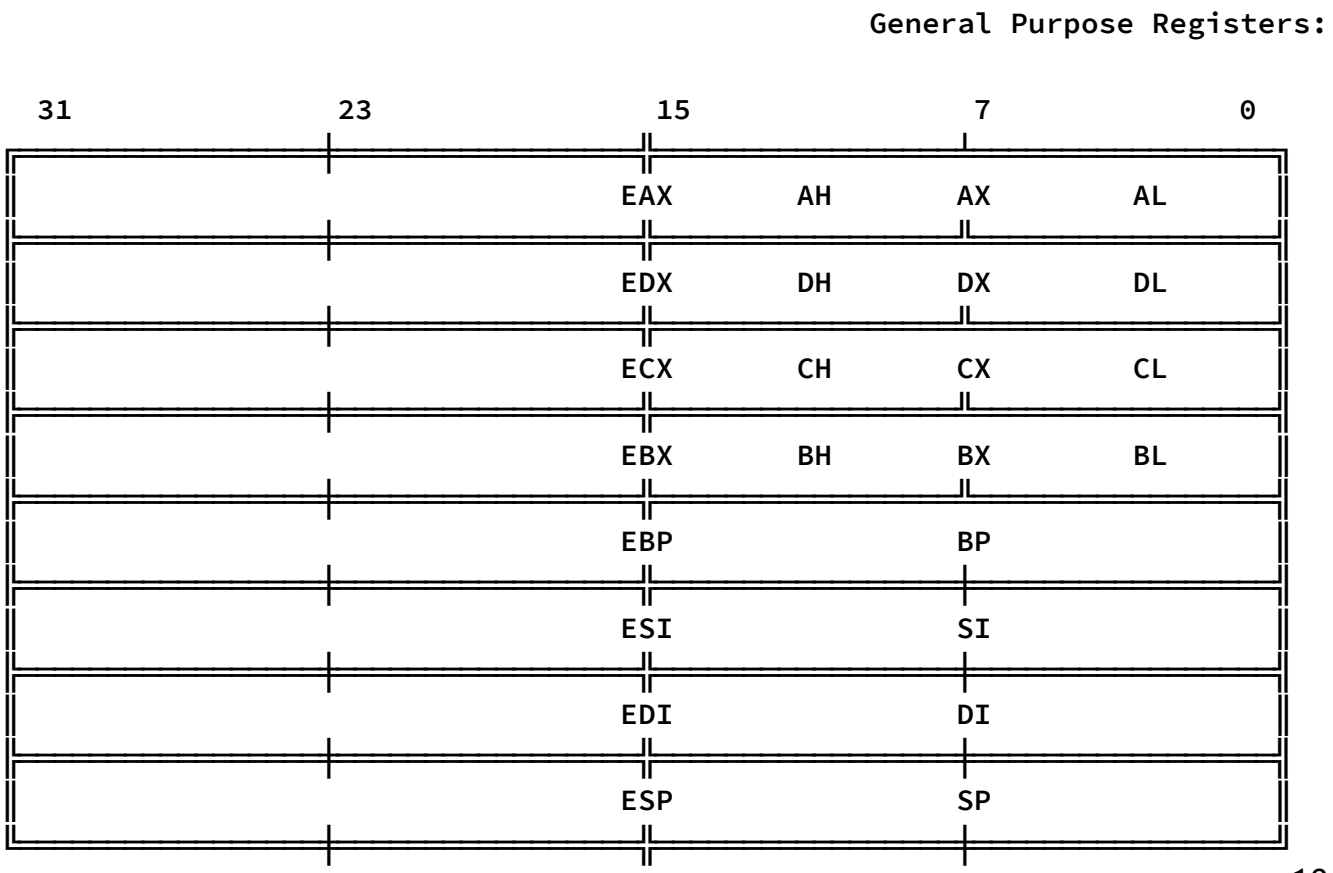
PA 1-1 数据的类型和存取



存储器的层次结构

PA 1-1 数据的类型和存取

- NEMU模拟的是i386体系结构
  - 8个32位通用寄存器



用C语言怎么模拟?

```
// define the structure of registers
typedef struct {

        // general purpose registers
        struct {
                struct {
                        struct {
                                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;
                };
        };

        // EFLAGS ···
} CPU_STATE;
```

框架代码中
nemu/include/cpu/reg.h
要实现和前面图中一样的结构

```
// define the structure of registers
typedef struct {

        // general purpose registers
        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;
                };
        };

        // EFLAGS ···
} CPU_STATE;
```

框架代码中
nemu/include/cpu/reg.h
要实现和前面图中一样的结构

PA 1-1 数据的类型和存取

在教程§1-1.3中提出的实验过程及要求:

1. 修改 CPU_STATUS 结构体中的通用寄存器结构体;
2. 使用 make 编译项目;
3. 在项目根目录通过执行下面两个命令之一来进行测试

- make test_pa-1
- ./nemu/nemu --test-reg

~PA 1-1顺利完成~

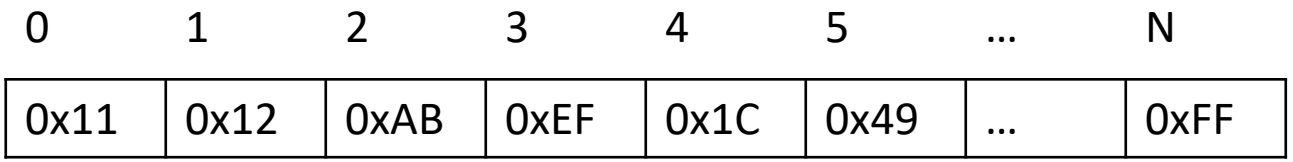
reg_test() pass

在实验报告中,简要叙述:

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?

PA 1-1 数据的类型和存取

- 除了寄存器之外，还要熟悉主存的模拟方式及其接口
  - 以字节为基本编址单位
  - 通过地址直接访问某个字节



- 怎么来模拟？数组！nemu/src/memory/memory.c中hw_mem[]
- 模拟内存多大？nemu/include/memory/memory.h中MEM_SIZE_B
- 模拟内的对外提供的读写接口？memory.h声明memory.c中实现

~要熟悉这些接口~

本周的任务

- 完成PA实验环境的搭建
- 完成PA 1-1
- 熟悉框架代码
- 不需要提交任何代码或答卷