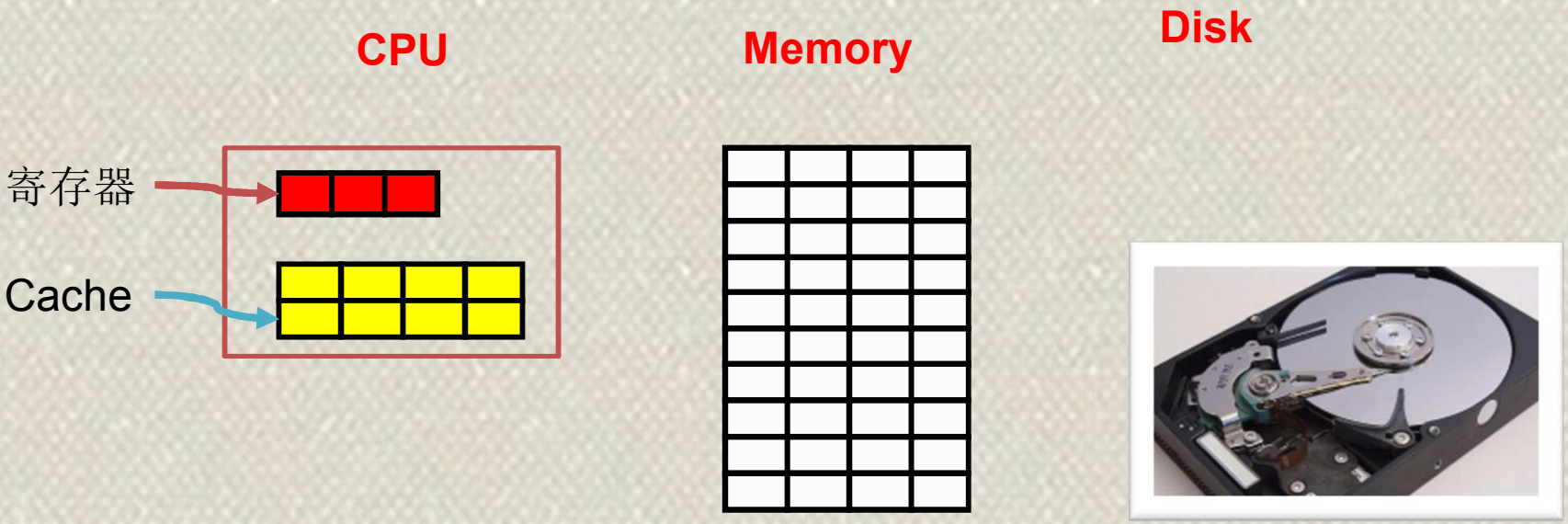
Ch.5 运算、存储

2018 / 10

都有哪些存储设备

- CPU寄存器
- 高速缓存cache
- 内存
- 外部存储设备（磁盘/光盘/SSD硬盘）

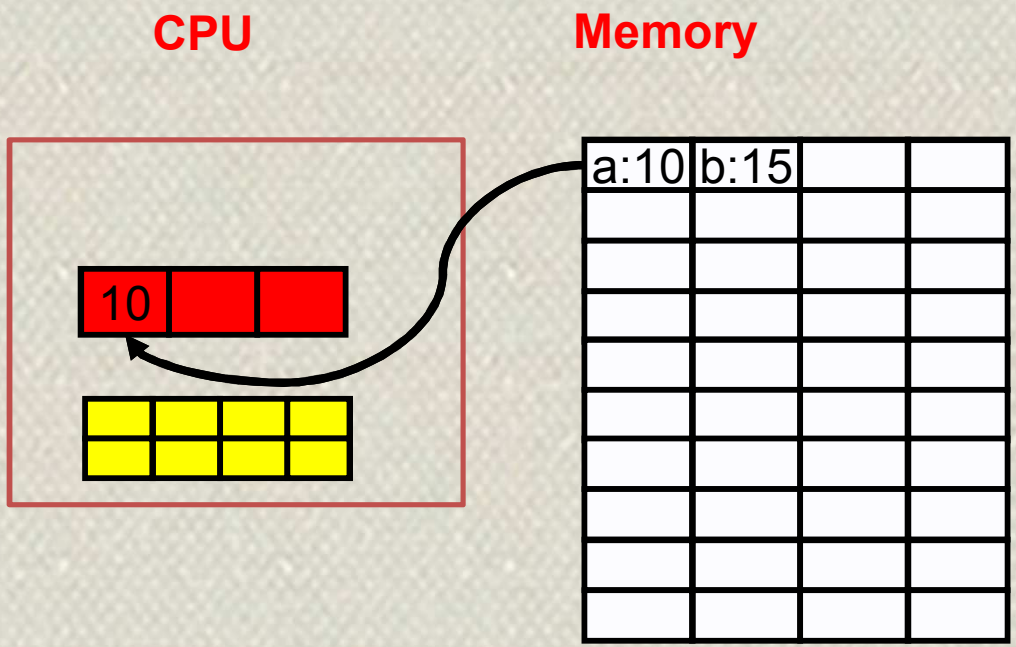
设备示意



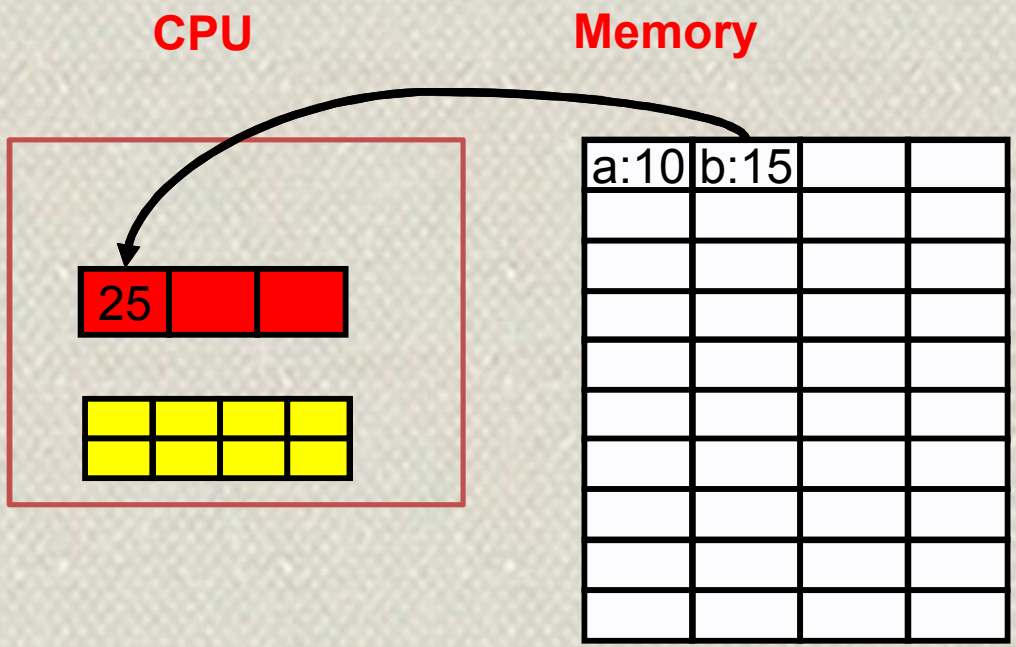
一个简单的程序

```
int a = 10, b = 15;
int sum = a + b;
```

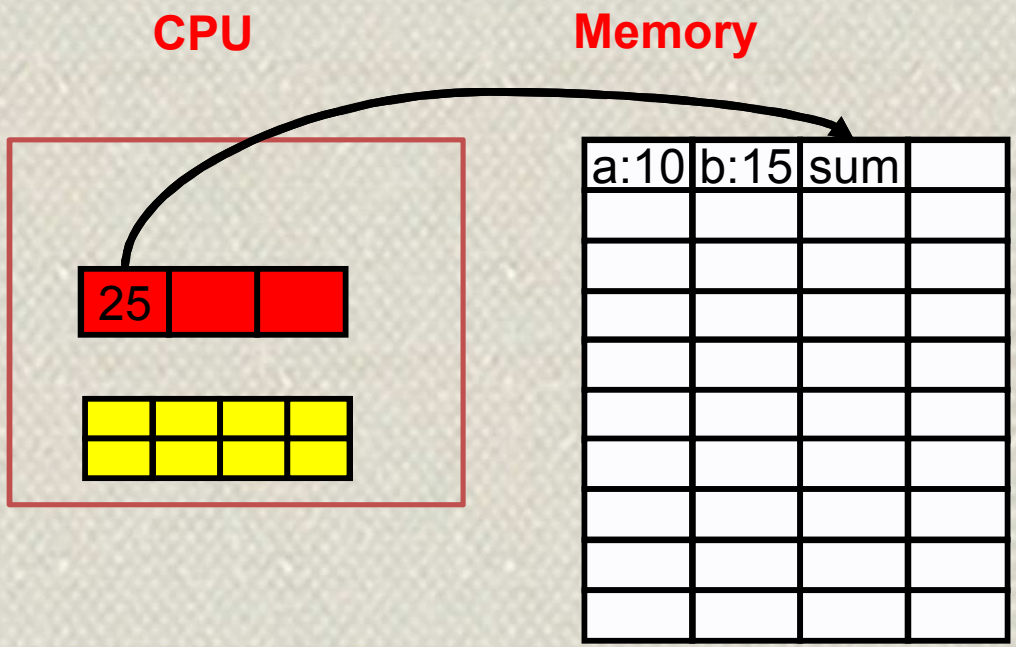
计算示意



计算示意



计算示意



如何进行效率比较？

- 多次执行看总体的时间消耗

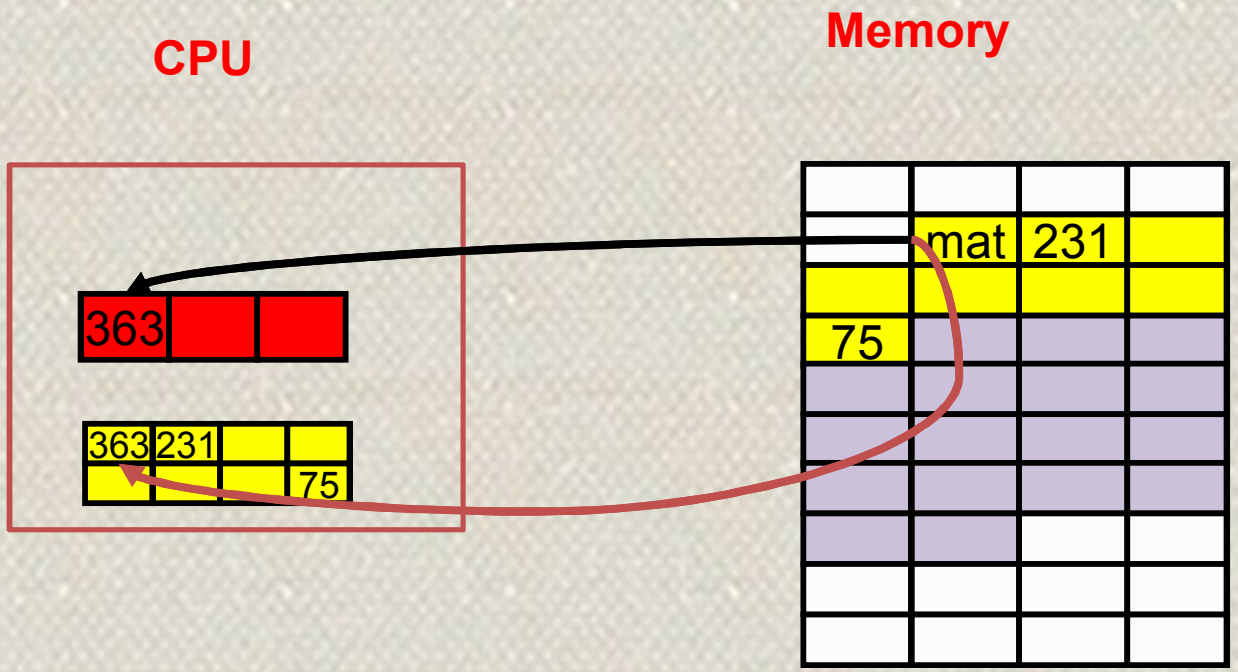
寄存器

- Register
  - register int a = 220;
  - register int b = 100;
  - a + b

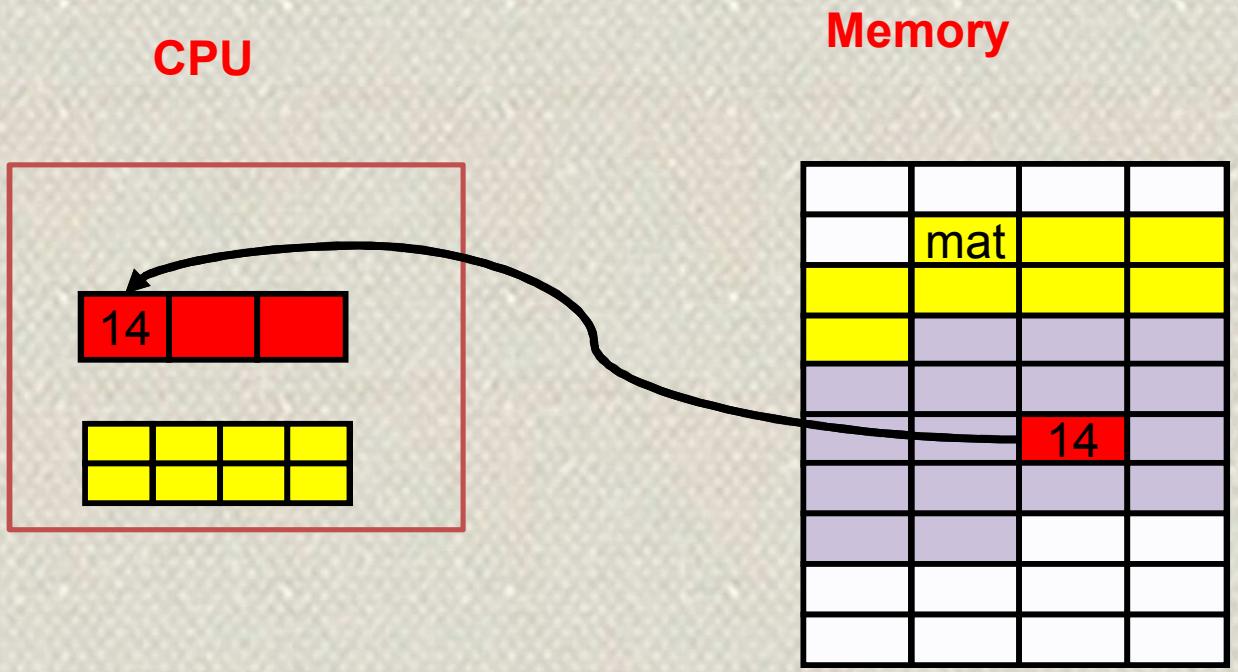
Cache

- Cache命中

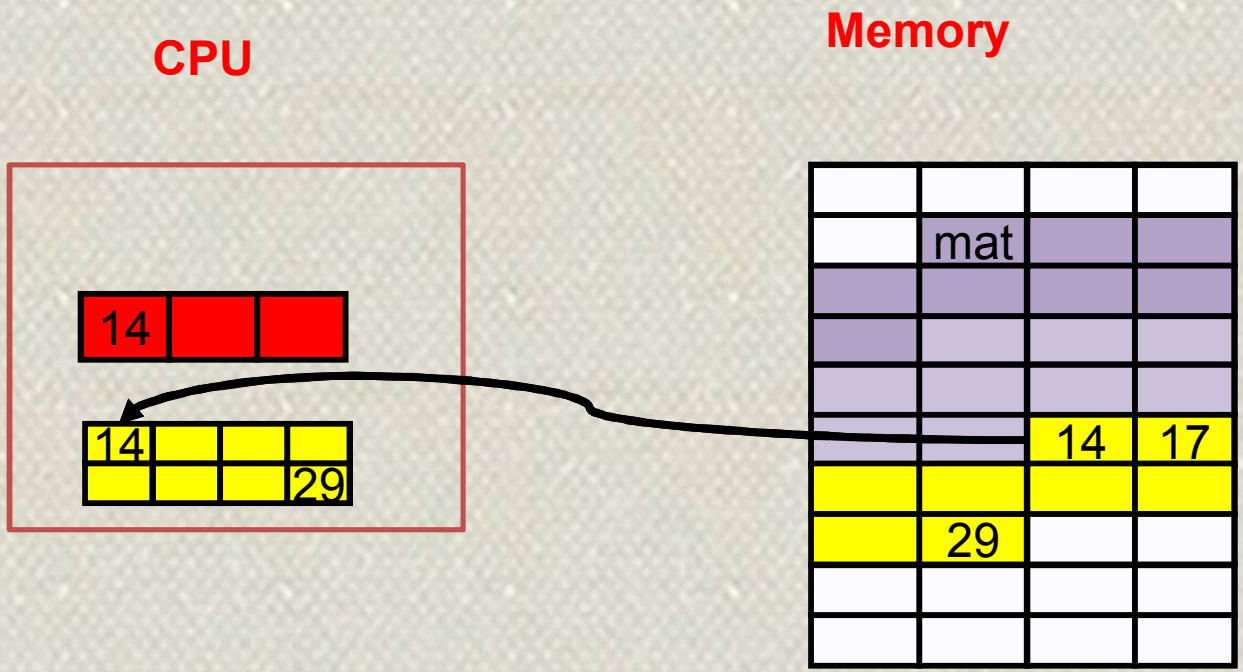
cache示意



cache示意



cache示意



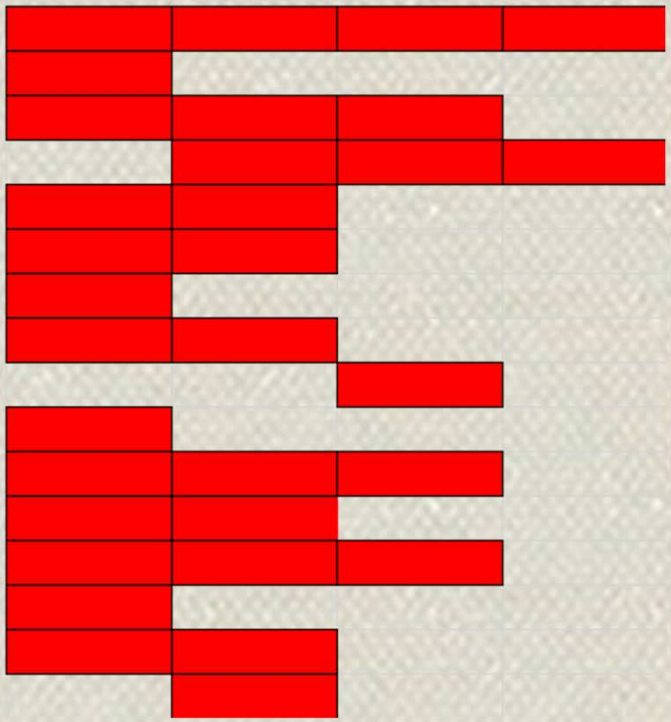
sizeof()

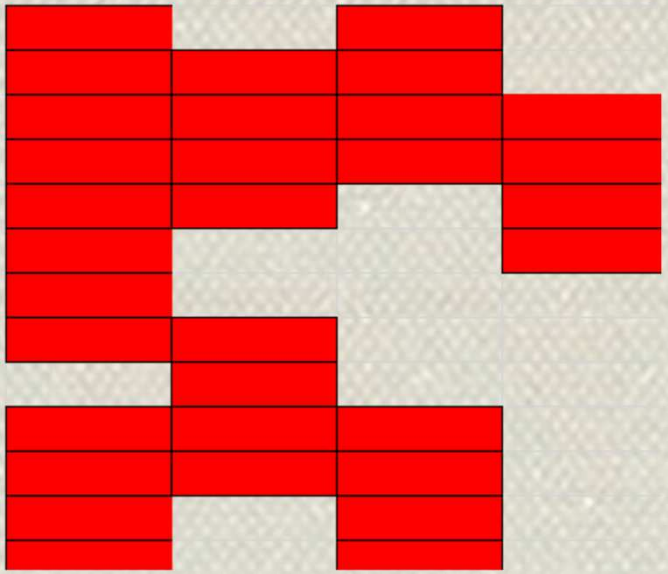
```
struct CCI
{
    char a;
    char b;
    int i;
};
```

```
struct CIC
{
    char a;
    int i;
    char b;
};
```

一种工具

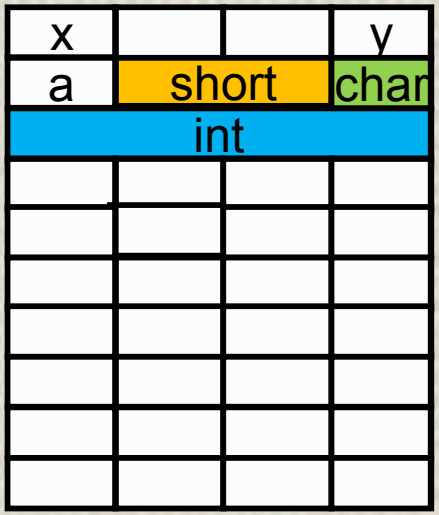
搬砖问题

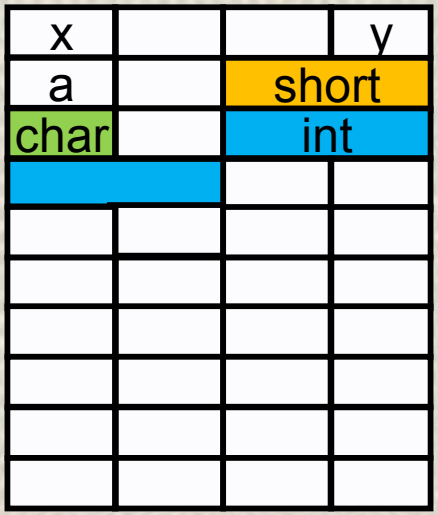


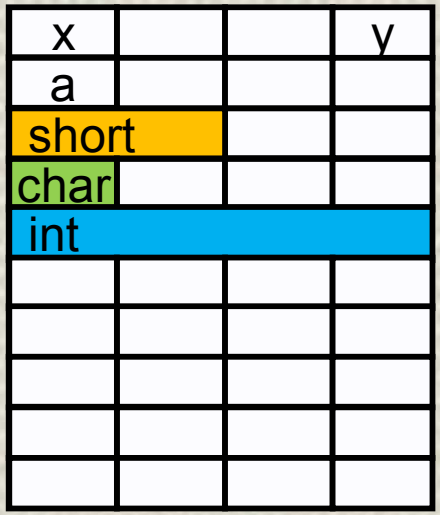


内存对齐

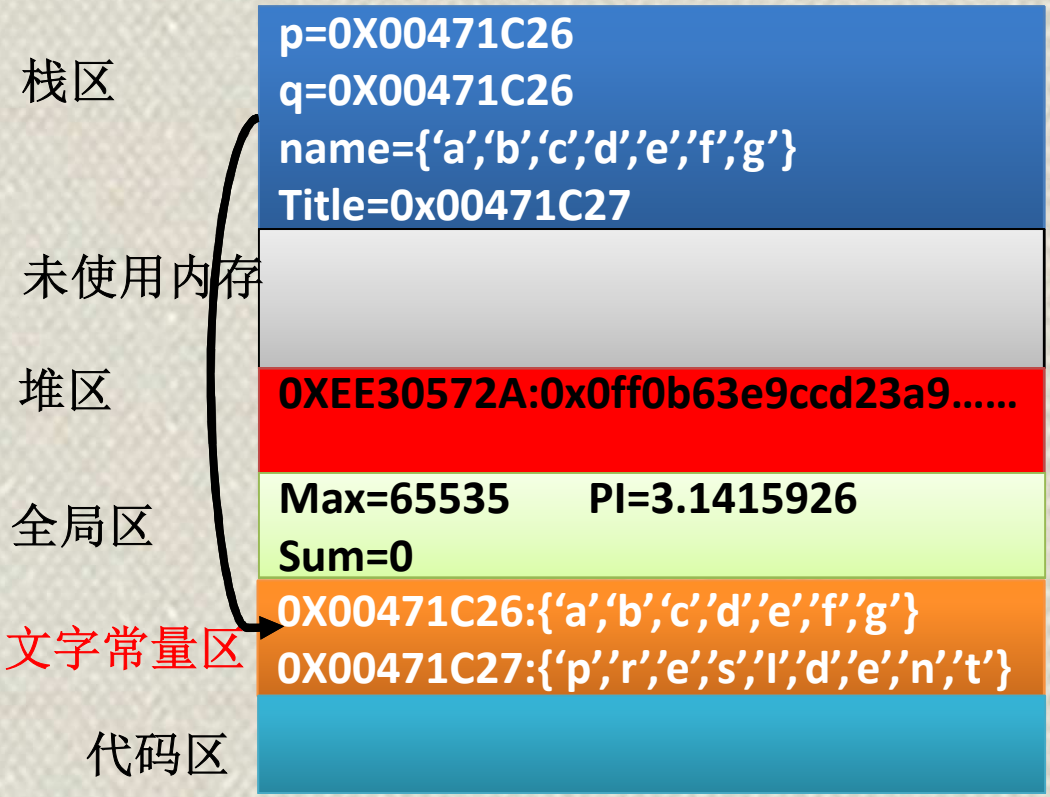
- 什么是内存字节对齐？
- 为什么要有对齐？







内存分配概览




```
int max = 65535;
const float PI = 3.1415926;
void main()
{
    char * p = "abcdefg";
    char * q = "abcdefg";
    char name [] = "abcdefg";
    char *title = "president";
    static int sum = 0;

    cbyte * dvec = new cbyte[128];
    delete[] dvec;
}
```

函数类比

- function = 吃饭（多人）
  - 内存空间 = 盘子
  - 数据 = 食物
- return = 吃完，盘子回收了

  void：吃完走人

  int：花了多少钱？

  bool：吃得好不好

  int * ：包间号

栈

- 吃完（reutrn）后，服务员把盘子拿走了
- 再想找你用过的盘子，可能已经分配给别人了

堆

- 餐厅有一种服务，可以外借盘子
- 借的盘子可以在餐厅就餐使用
- 你借了 10 个盘子回家吃饭，吃多少次饭都可以
- 直到你把盘子送回来，其他顾客才可以再使用

堆的分配方法

- malloc/free
- new/delete

几个概念

- 内存溢出：计算超过限定大小
- 访问越界：数组超出范围
- 栈溢出：栈空间不足
- 内存泄漏：没有回收，或是访问已经回收的位置

熟悉的代码

```
int fac (int n)
{
        if (n == 1)
                return 1;
        return fac (n - 1) * n;
}
```

递归与栈的关系

- 调用下一层时，先保存现场
- 返回后回到上一次的现场

内存泄漏

- 内存泄漏是申请的内存没有释放导致
- 已经释放的内存，再次释放

避免泄漏？

- 有借有还
- 是不是可以自动归还？
  - 设计什么样的结构，什么时候归还？

检测泄漏

- 怎么检测内存泄漏？

Q & A