# 第一章：内存

## 程序运行为什么需要内存

- 程序运行为什么需要内存?
  - 计算机程序运行需要内存存数据（Flash存程序/代码段）
  - 计算机程序的本质：代码（动作）+数据（全局变量、局部变量）
  - 计算机运行的本质：多个程序相继运行，程序由多个函数组成，函数的本质是加工数据的动作
  - 程序运行的目的：得到结果（通过函数返回值或者形参返回）、得到过程（返回void，重要的是怎么执行）
- 冯诺依曼结构&哈佛结构
  - 冯诺依曼结构：数据、代码放在一起
  - 哈佛结构：数据、代码分开放（单片机中，程序烧写在NorFlash中，数据放在SRAM，也就是内存中）
- 不同角度下的内存管理
  - OS：有OS的程序，OS掌管所有硬件内存，以页面为单位管理，不需要了解细节，会调用API即可；裸机程序中，程序员直接操作内存
  - 语言：不同语言有不同操作内存的接口
    - 汇编：内存管理全靠程序员自己操作内存地址
    - C语言：通过变量名访问内存，大块空间使用malloc和free
    - C++：对内存进一步封装，可以使用new创建对象、分配内存，delete删除对象、释放内存（C、C++没有free，delete会内存泄露）
    - JAVA、C#：不直接操作内存，使用内存后不用管，虚拟机回收内存需要代价
    - 对程序性能在意用C、C++，快速开发应用使用JAVA/C#

## 位、字节、半字、字、内存位宽

- 内存可以随机访问，且可以读写，n位内存代表一个地址单元有n位（内存位宽为n）

- 内存单元大小的单位：位（一定是1 bit）、字节（一定是8 bit）、半字（一般16 bit）、字（一般32bit)，字和半字取决于系统的位数，ARM系统一般是32位
- 编程中一般用不到字这个概念，区分主要是为了文档阅读

# C语言操作内存

- 汇编直接操作内存地址，C语言对内存地址进行封装，用变量名访问内存（编译器将内存地址和变量名联系起来）
  - int a；//编译器申请一个内存格子，4个字节，地址只有编译器知道，并将符号a和内存格子首元素首地址绑定
  - int a[10]；//编译器申请10内存格子，40个字节，并将符号a和内存格子首元素首地址绑定
  - a=5；//编译器把5丢到a绑定的内存格子里
- 数据类型本质含义：指定**内存格子的长度**、**解析方式**（int表示将四个格子中存的二进制数连起来、并按照int解析）
- 函数名本质含义：函数是一段代码的封装，函数名是**一段代码的首地址**

# 栈（堆栈=栈）

- 自动分配小块内存（指针移动、内存分配、压栈出栈是自动的，不用程序员管），比如：局部变量
  - 弹栈只是修改指针，并读了一份数据出去，栈中还存有该值；局部变量不初始化，则值是随机的
  - int a=15和int a; a=15; 开销一致
- 不足之处：栈的大小固定（太大浪费空间、太小溢出）
  - C语言不检查溢出，定义局部变量时不能太多或者太大（a[10000]）
  - 递归一定要注意递归收敛

# 堆

- 适用场景：需要内存容量大（至少几十个字节）、需要反复使用、释放；实现数据结构
- 特点：容量没有限制；申请释放需要程序员进行（申请未释放则内存泄露）
  - 内存泄露/吃内存：堆管理器记录该内存仍属于进程，但进程需要使用又会重新申请
  - 内存泄露是C和C++最难处理的bug

- 堆内存申请指定大小，大小不变，要变化需要remalloc（重新申请、释放原空间、返回新的空间给用户，类似java的可变数组）

# 其它

- NorFlash和NandFlash
  - PC机里有一块BIOS，本质上是NorFlash，里面预先烧写了一段启动代码
  - 一般两种Flash配合使用

| NorFlash | NandFlash |
| --- | --- |
| 数据线、地址线分开 | 数据线、地址线共用 |
| 容量小、价格高、访问快 | 容量大、价格低、访问慢 |
| CPU上电直接读取，可用作启动介质 | 上电后，CPU运行一些软件，再通过时序接口（读写需要按照一定时序操作）读取 |

- 操作系统最复杂的部分就是内存管理和进程调度