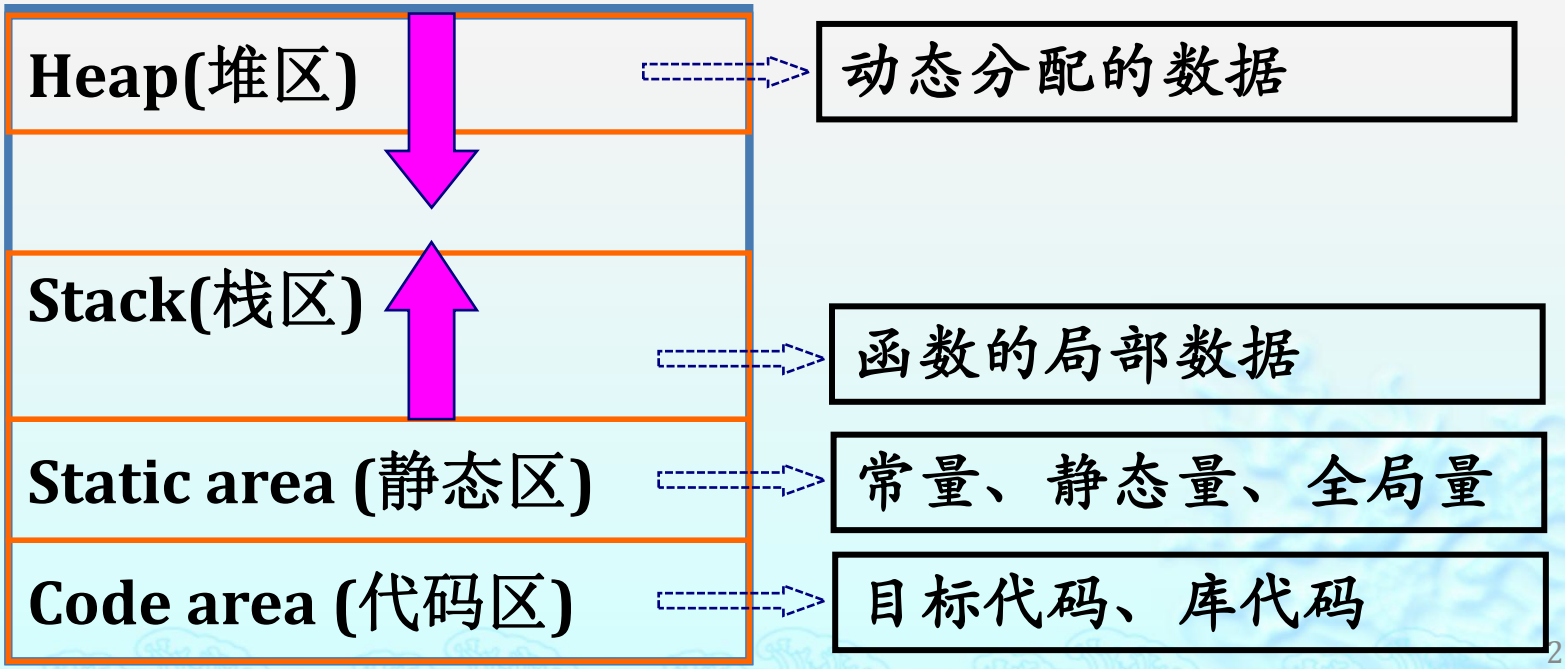
第9章 运行时存储空间管理

- 9.1 目标程序运行时的存储结构
- 9.2 数据空间的三种不同管理方法
- 9.3 栈式存储分配的实现

9.1 目标程序运行时的存储结构

- 在代码生成前,编译程序要向操作系统申请一块存储区,用意容纳生成的目标代码和目标代码运行时需要的数据空间.数据空间包括:用户定义的各种类型的数据对象(变量和常量),用于保留计算的中间结果和传递参数的临时工作单元,调用函数时所需的连续单元,以及组织输入输出所需的缓冲区.



9.2 数据空间的三种不同管理方法

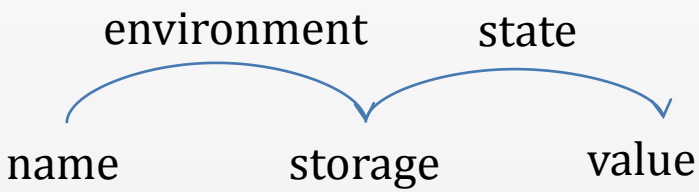
- 完全静态存储管理
  - 编译时能够确定目标程序运行时所需的全部数据空间的大小，编译时可安排好目标程序运行时的全部数据空间，确定每个数据对象的存储位置.
- 动态存储分配
  - 如果一个程序设计语言允许递归过程、可变数组或允许用户自由申请和释放空间，那么就需要采用动态存储管理技术.
  - 栈式动态存储管理
    - 目标程序运行时,把存储器作为一个栈管理
    - 每当调用一个函数,它所需要的存储空间就会被分配在栈顶,每当退出一个函数,它所占用的空间就收回.
  - 堆式动态存储管理
    - 目标程序运行时,把存储器作为一个堆管理
    - 随用随分配

9.2 数据空间的三种不同管理方法

- 几种存储管理方式的比较
- 管理的复杂程度
  - 静态：简单，但易造成空间的浪费
  - 堆式：复杂，代价高，但节省空间
  - 栈式：节省空间，需一定的管理
- 适合的分配对象
  - 静态：全局变量、静态变量、常量、表
  - 堆式：指针
  - 栈式：递归的过程/函数

9.2 数据空间的三种不同管理方法

- 存储分配的本质是将变量名字与存储位置关联起来，该存储位置用于容纳该变量的值



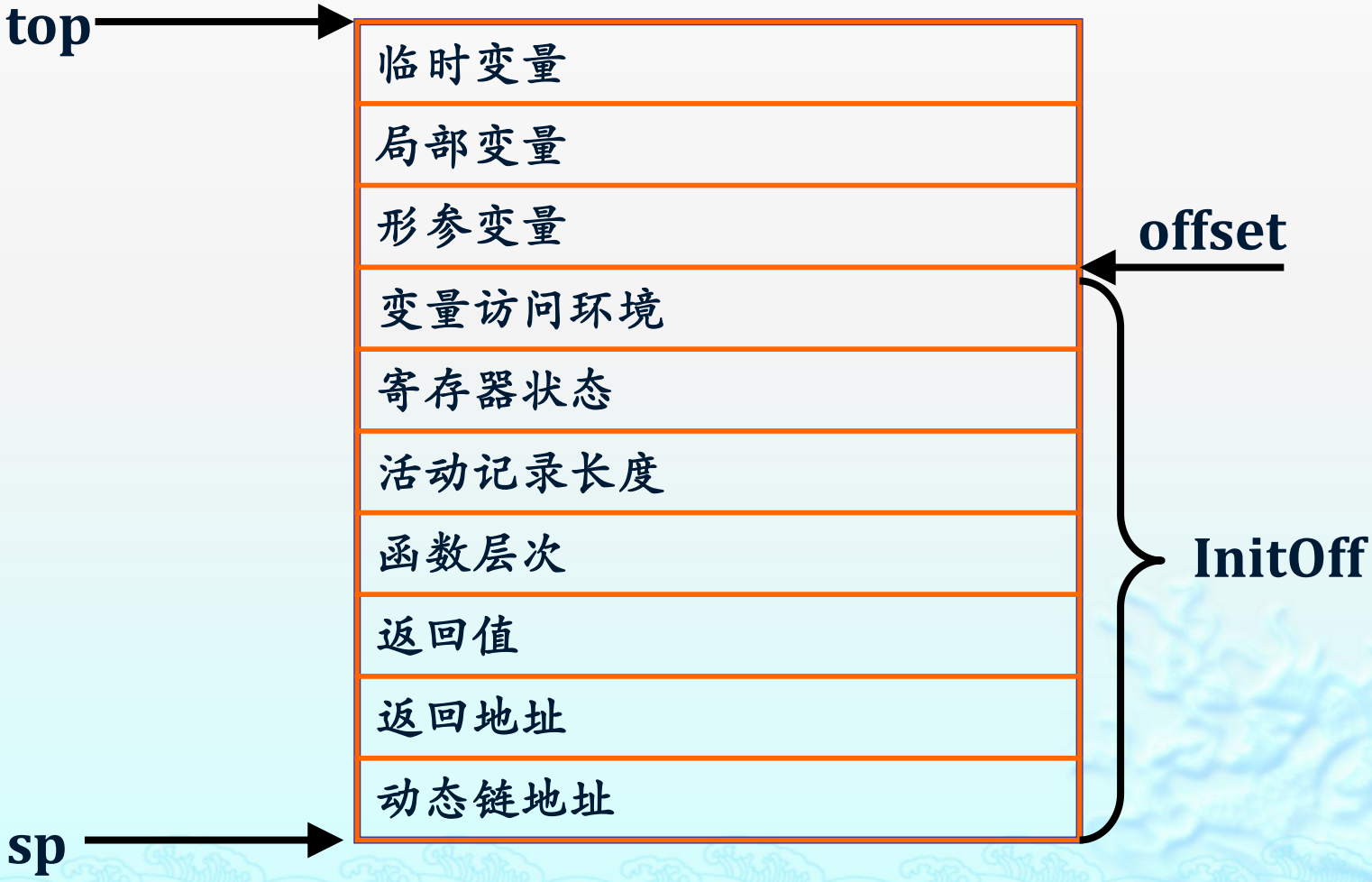
- 源语言的结构特点、数据类型、名字作用域规则等因素影响存储空间管理和组织的复杂程度，决定存储空间分配的基本策略
- 常见的影响因素：过程能否递归？当控制从过程活动返回时，局部变量的值是否要保留？过程能否访问非局部量？过程调用的参数传递方式(传值、传地址、传引用)。存储块能否在程序控制下动态地分配？存储块是否必须显式地释放？

9.3 栈式存储分配的实现

- 9.3.1 过程活动记录
- 9.3.2 简单的栈式存储分配的实现
- 9.3.3 过程活动记录的管理

9.3.1 过程活动记录

- 过程活动记录AR(Activation Record):是一段连续的存储区,用以存放过程的一次执行所需要的信息,这些信息包括:



9.3.1 过程活动记录

- 调用链: (M)是调用链，若(M,...,R)是调用链，并且R中有S的调用，则(M,...,R,S)也是调用链.
  - M是C语言中的main函数或Pascal语言中的主程序
  - S是当前正在执行的过程，M,...,R是已经开始执行，但被中断了的过程. CallChains(S) = (M,...,R,S)表示S的调用链
- 每个调用链对应一个动态的过程调用序列

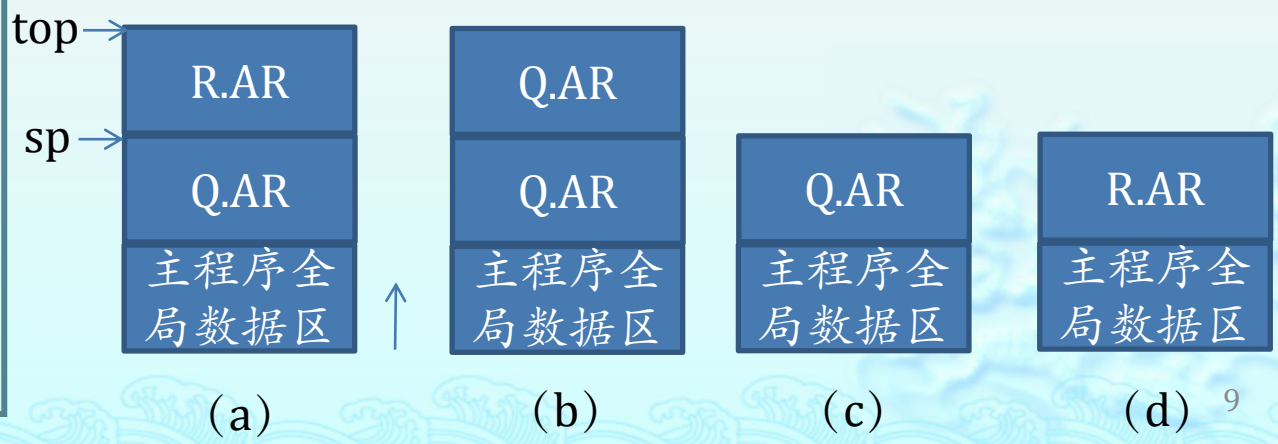
- 动态链: 若S是当前正在执行的过程，CallChains(S) = (M, ..., R, S)，则栈的内容可表示为：[AR(M),...,AR(R),AR(S)],称它为对应调用链(M, ..., R, S)的动态链
- 由于每个过程的AR的大小可能都是不同的，因此在每个过程的AR中都要保留它的调用者的AR 的起始地址

9.3.2 简单的栈式存储分配的实现

- 语言特点: 没有分程序, 过程定义不嵌套, 但允许过程递归调用

若主程序调用了Q，Q又调用了R，在R进入运行后的存储结构如图a所示:
若主程序调用了过程Q，Q递归调用自己，在Q第二次进入运行后的存储结构如图b所示:
若主程序先调用了过程Q，然后调用过程R, 且过程Q没有调用过程R，这时Q和R进入运行后的存储结构分别如图c和图d所示.

```
program main
全局变量说明
proc R;
...
end(R);
proc Q;
...
end(Q);
主程序执行语句
end.(main)
```



9.3.2 简单的栈式存储分配的实现

- 例9.1: 写出下列程序运行时存储空间的变化情况.

```
#define  n 2
int  sum = 0;
int fac(int i)
{ if (i==0) return 1;
   if (i<0) return -1;
   return (i*fac(i-1));
 }

void main ()
{
 sum = fac(n);
}
```

9.3.3 过程活动记录的管理

- Who?
  - 编译程序要生成相应的目标代码来管理AR
- What?
  - 分配和释放AR的空间,传递必要的信息
- When?
  - 传参四元式
  - 函数调用四元式
  - 函数入口四元式
  - 函数出口四元式
  - Return四元式

9.3.3 过程活动记录的管理

- 传参四元式(VALACT/VARACT/FUNCACT,result, offset, size)：
  - VALACT: 将result对应内存单元的值存储到offset所指向的内存单元
  - VARACT:将result的地址存储到offset所指向的内存单元
  - FUNCACT:将result对应单元中存放的函数入口地址和Display表信息传递到offset所指向的内存单元

9.3.3 过程活动记录的管理

- 函数调用四元式(CALL, Lf, true/false, t):
  - 将t的地址存到返回值地址单元中
  - 将sp的当前值存入动态链单元
  - 保存返回地址
  - 保存寄存器的当前状态
  - 保存变量访问环境，层数，size等信息
- 函数入口四元式(ENTRY, Lf, size, level):
  - 调整sp指针
  - 调整top指针

9.3.3 过程活动记录的管理

- Return四元式(RETURN, -,-,t)
  - 将t的值存储到返回值单元记录的地址
- 函数出口四元式(ENDFUNC, -，-，-):
  - 恢复寄存器的内容
  - 调整top指针
  - 调整sp指针
  - 按照返回地址返回