编译原理

第九章 运行时存储空间组织

第九章 运行时存储空间组织

- 目标程序运行时的活动
- 运行时存储器的划分
- 静态存储管理
- 一个简单栈式存储分配
- 嵌套过程语言的栈式实现

第九章 运行时存储空间组织

- 目标程序运行时的活动
- 运行时存储器的划分
- 静态存储管理
- 一个简单栈式存储分配
- 嵌套过程语言的栈式实现

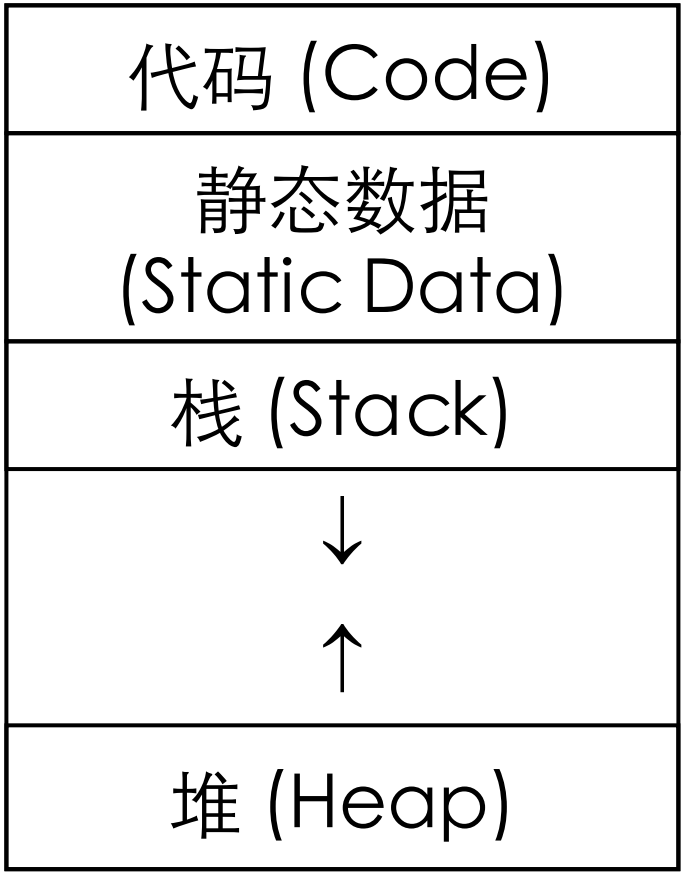
编译程序组织存储空间须考虑的问题

- 过程是否允许递归?
- 当控制从一个过程的活动返回时，对局部名称的值如何处理?
- 过程是否允许引用非局部名称?
- 过程调用时如何传递参数；过程是否可以做为参数被传递和做为结果被返回?
- 存储空间可否在程序控制下进行动态分配?
- 存储空间是否必须显式地释放?

9.2 运行时存储器的划分

- 一个目标程序运行所需的存储空间包括
  - 存放目标代码的空间
  - 存放数据项目的空间
  - 存放程序运行的控制或连接数据所需单元（控制栈）

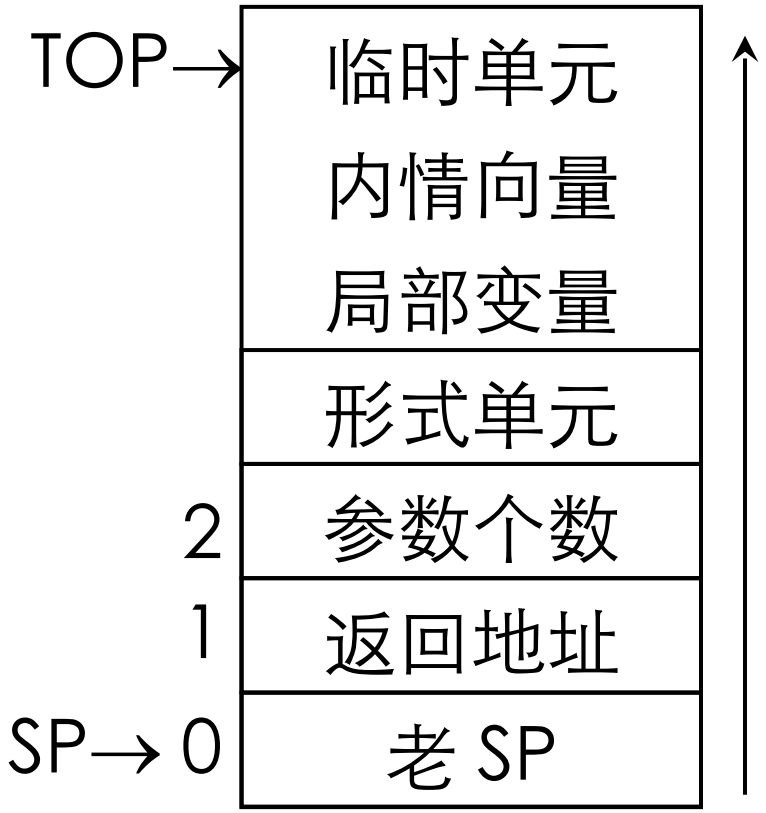
程序在运行时刻的内存划分



9.2.2 活动记录

- 假定语言的特点
  - 允许过程递归调用、允许过程含有可变数组
  - 过程定义不允许嵌套，如 C 语言
- 采用栈式存储分配机制
- 活动记录
  - 运行时，每当进入一个过程就有一个相应的活动记录累筑于栈顶
  - 此记录含有连接数据、形式单元、局部变量、局部数组的内情向量和临时工作单元等

每个过程的活动记录内容（非嵌套语言）

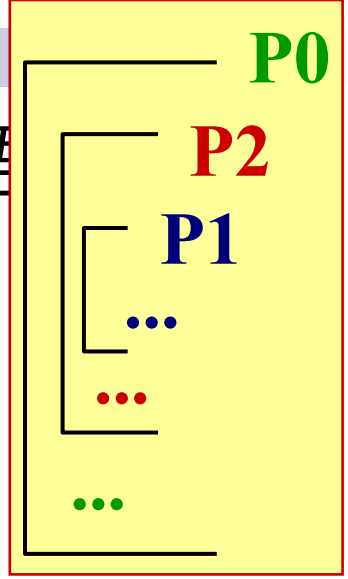


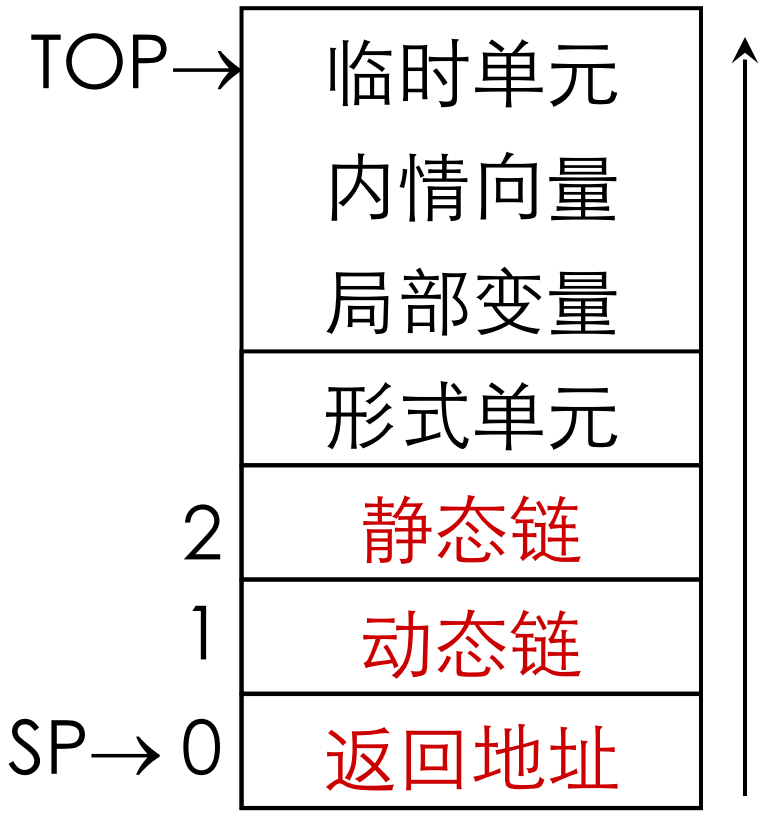
对任何局部变量 X 的引用可表示为变址访问：

dx[SP]

dx：变量 X 相对于活动记录起点的地址，在编译时可确定。

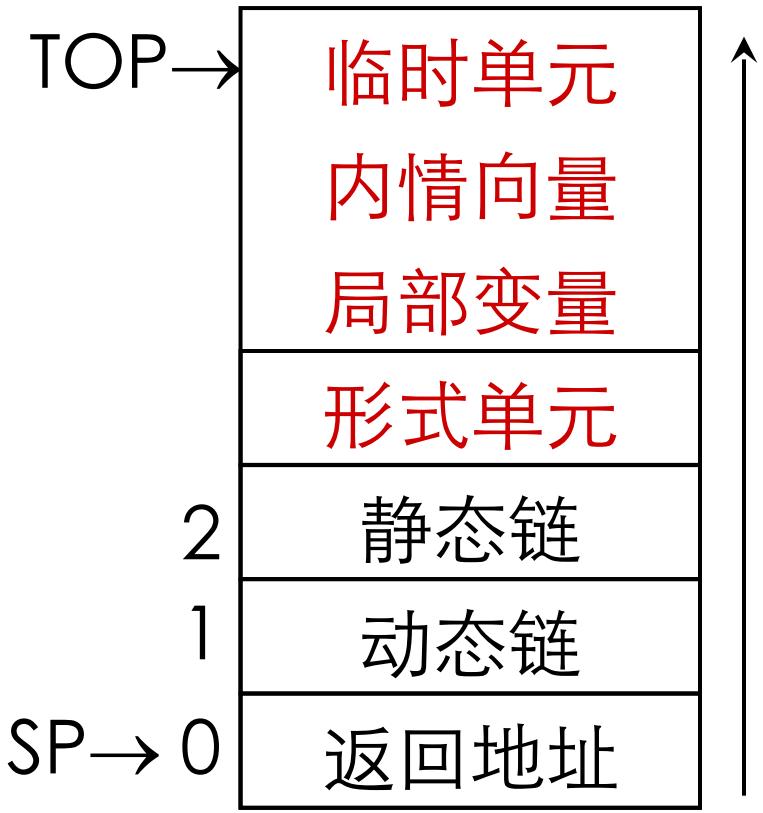
每个过程的活动记录内容（嵌套语





- 连接数据
  - 返回地址
  - 动态链：指向调用该过程前的最新活动记录地址的指针。
  - 静态链：指向静态直接外层最新活动记录地址的指针，用来访问非局部数据。

每个过程的活动记录内容



- 形式单元：存放相应的实在参数的地址或值。
- 局部数据区：局部变量、内情向量、临时工作单元（如存放对表达式求值的结果）。

9.2.3 存储分配策略

- 静态分配策略 (FORTRAN)

  如果在编译时能确定数据空间的大小，则可采用静态分配方法：在编译时刻为每个数据项目确定出在运行时刻的存储空间中的位置。

- 动态分配策略 (PASCAL)

  如果在编译时不能确定运行时数据空间的大小，则必须采用动态分配方法。允许递归过程和动态申请释放内存。

  - 栈式动态分配
  - 堆式动态分配

第九章 运行时存储空间组织

- 目标程序运行时的活动
- 运行时存储器的划分
- 静态存储管理
- 一个简单栈式存储分配
- 嵌套过程语言的栈式实现

9.3 静态存储管理

```
PROGRAM   MAIN
  …
integer X ,  Y
COMMON /C/A ,  B
 …
END
SUBROUTINE SUB1
  …
real X ,  Z
COMMON /C/A1 ,  B1
  …
END
```

9.3 静态存储管理

- FORTRAN 程序的特点
  - 整个程序所需数据空间的总量在编译时完全确定
  - 每个数据名的地址可以静态地进行分配
- 每个 FORTRAN 程序段可以独立编译，运行前由装入程序把各段连成可运行的整体
- 按数据区组织存储
  - 每个程序段定义一个局部数据区，用来存放程序段中未出现在 COMMON 里的局部名的值
  - 每个公用块定义一个公用数据区，用来存放公用块里各个名字的值

- 每个数据区有一个编号，地址分配时，在符号表中，对每个数据名登记其所属数据区编号及在该区中的相对位置
- 每个局部数据区的内容及存放次序

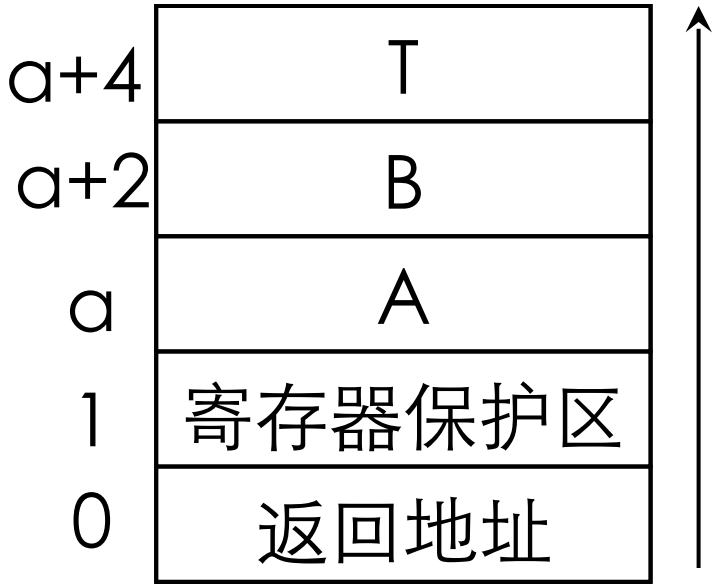
- 考虑子程序段：

```
SUBROUTINE SWAP(A,B)
T=A
A=B
B=T
RETURN
END
```

To understand a program you must become both the machine and the program.

Alan J. Perlis

名字 NAME	性质 ATTRIBUTE	地址 DA	地址 ADDR
SWAP	子程序，二目		
A	哑，实变量	k	a
B	哑，实变量	k	a+2
T	实变量	k	a+4



第九章 运行时存储空间组织

- 目标程序运行时的活动
- 运行时存储器的划分
- 静态存储管理
- 一个简单栈式存储分配
- 嵌套过程语言的栈式实现

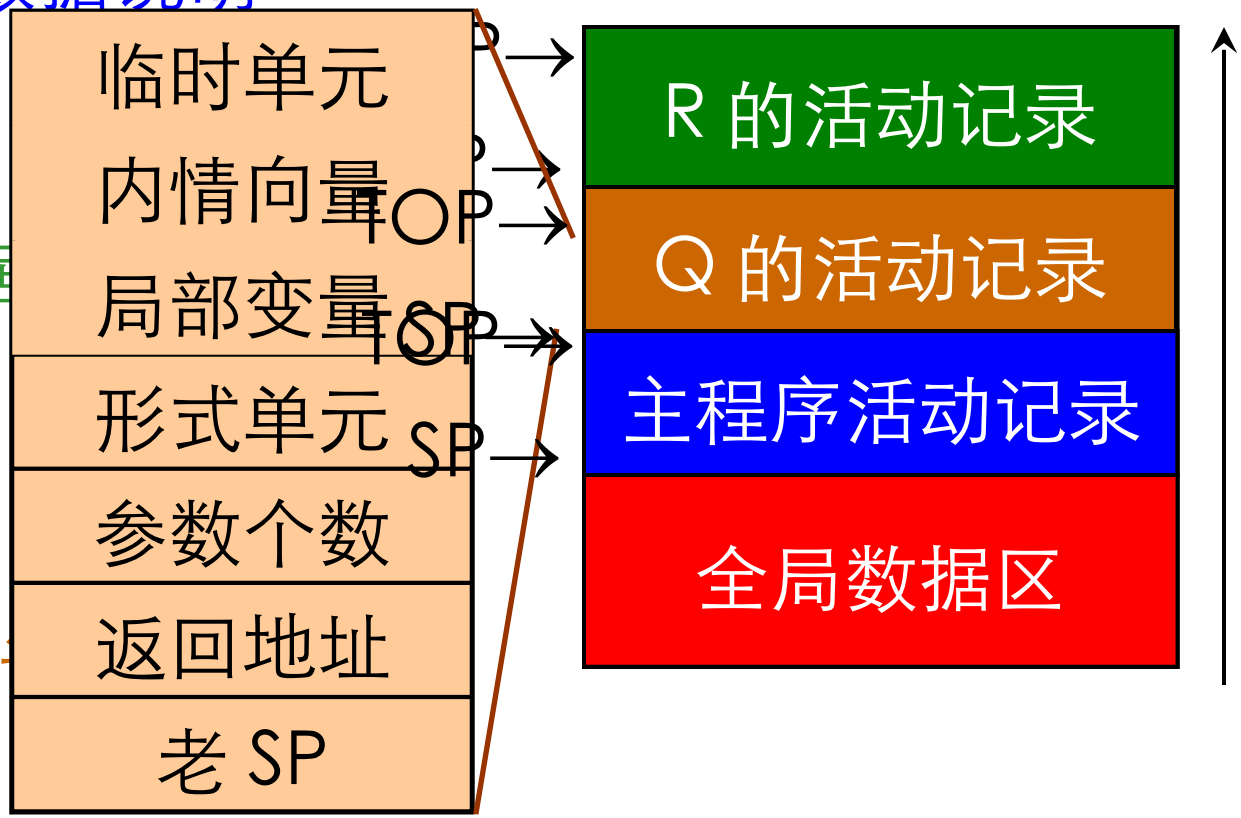
9.4 一个简单栈式存储分配

- 假定语言的特点为
  - 允许过程递归调用、允许过程含有可变数组
  - 过程定义不允许嵌套，如 C 语言
- 采用栈式存储分配机制
- 活动记录
  - 运行时，每当进入一个过程就有一个相应的活动记录累筑于栈顶
  - 此记录含有连接数据、形式单元、局部变量、局部数组的内情向量和临时工作单元等

全局数据说明

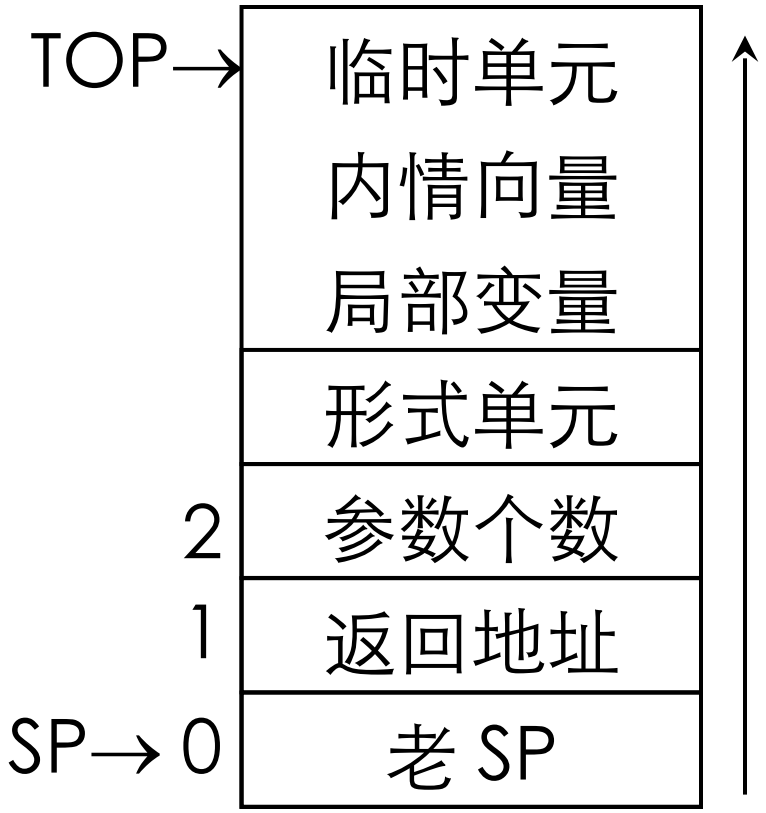
```
Main( ) {
      Main 中的数据说明
}
void R( ) {
      R 中的数据
}
…
void Q( ){
      Q 中的数据
}
```

- 主程序
- → 过程 Q
- → 过程 R



9.4.1 C 的活动记录

- 每个过程的活动记录内容如图：



对任何局部变量 X 的引用可表示为变址访问：

dx[SP]

dx：变量 X 相对于活动记录起点的地址，在编译时可确定。

9.4.2 C 的过程调用、过程进入、数组空间分配和过程返回

- 过程调用的语句

$$P(T_1, T_2, \cdots, T_n)$$

- 翻译成四元式序列

par T_1
par T_2
…
par T_n
call P , n

对于 par 和 call 产生的目标代码如下：

1）每个 par T_i(i=1,2,…n) 可直接翻译成如下指令：

(i+3)[TOP]:= T_i　　（传值）

(i+3)[TOP]:=addr(T_i)　（传地址）

2) call P，　n　被翻译成：

1[TOP]:=SP（保护现行 SP)

3[TOP]:=n　（传递参数个数）

JSR P　　（转子指令）

3) 转进过程 P 后，首先执行下述指令

SP:=TOP+1　　（定义新的 SP）

1[SP]:= 返回地址　　（保护返回地址）

TOP:=TOP+L　　（新 TOP）

L：过程 P 的活动记录所需单元数，在编译时可确定。

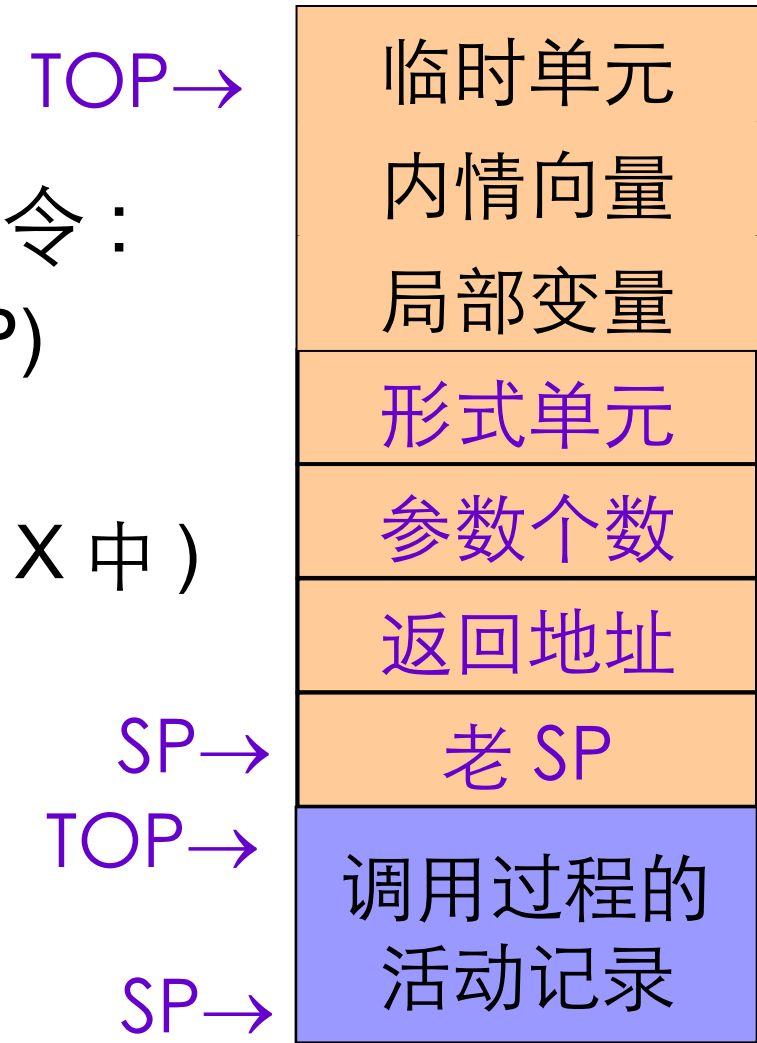
4）过程返回时，应执行下列指令：

TOP:=SP-1　　（恢复调用前 TOP）

SP:=0[SP]　　（恢复调用前 SP）

X:=2[TOP]　　（把返回地址取到 X 中）

UJ 0[X]　（按 X 返回）



小结

- 运行时存储器的划分
  - 活动记录
- 静态存储管理
- 一个简单栈式存储分配
  - C 的活动记录
  - C 的过程调用、过程进入、数组空间分配和过程返回