第七章 运行时环境

在前几章中，我们已研究了实现源语言静态分析的编译程序各阶段。该内容包括了扫描、分析和静态语义分析。这个分析仅仅取决于源语言的特性，它与目标 (汇编或机器)语言及目标机器和它的操作系统的特性完全无关。

运行时环境指的是目标计算机的寄存器以及存储器的结构，用来管理存储器并保存指导执行过程所需的信息。

三种运行时环境：

- 完全静态环境（FORTRAN77）
- 基于栈的环境（C, C++, PASCAL）
- 完全动态环境（LISP）

三种类型的混合形式也是可能的

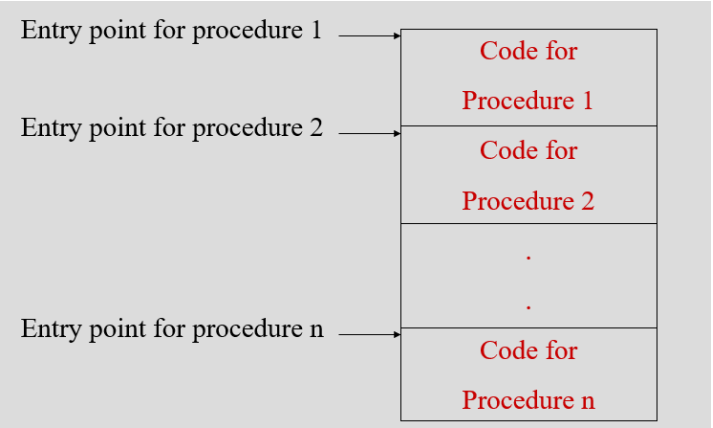
程序执行时的存储器组织

存储器

> - 寄存器
> - RAM
>   - 代码区
>   - 数据区

在绝大多数的语言中，执行时不可能改变代码区，且在概念上可将代码和数据区看作是独立的。另外由于代码区在执行之前是固定，所以在编译时所有代码的地址都是可计算的。

代码区：



编译时还可以知道每个过程的入口点和函数

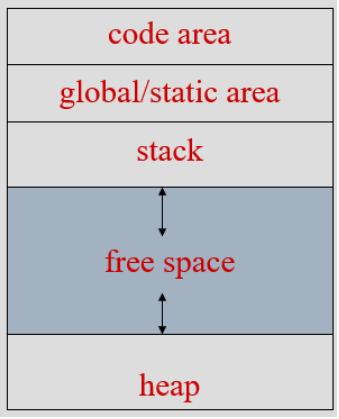
数据区：执行前，只有一小部分数据可以被固定在存储器中

- 程序的全局 和/或 静态数据
- 编译时常量
  - 大型的整型值
  - 浮点值
- 串文字

动态存储器

用作动态数据分配的存储区何以按照多种方式组织，典型的组织是将这个存储器分为栈区域和堆区域。

存储器组织图：【代码区域】【全程、静态区域】【栈】【自由空间】【堆】



过程活动记录（Procedure Activation Record）（考点）

它是存储器分配中的一个重要单元，当调用或激活过程或函数时，它包含了为其局部数据分配的存储器

活动记录
自变量（参数）空间
用作簿记信息的空间，包括了返回地址
用作局部数据的空间
用作局部临时变量的空间

图示仅表示活动记录的一般组织，特定细节依赖于目标机器的体系结构、被编译的语言特性。当将活动记录保存在栈中时，它们有时指的是栈框架。

寄存器可用来保存临时变量、局部变量甚至是全局变量。当处理器含有多个寄存器的时候，整个静态区域和整个活动记录都可完整地保存在寄存器中。处理器还具有特殊用途的寄存器以记录执行，如在 大多数的体系结构中的程序计数器 (pc)、栈指针(sp)(stack pointer)。

完全静态运行时环境

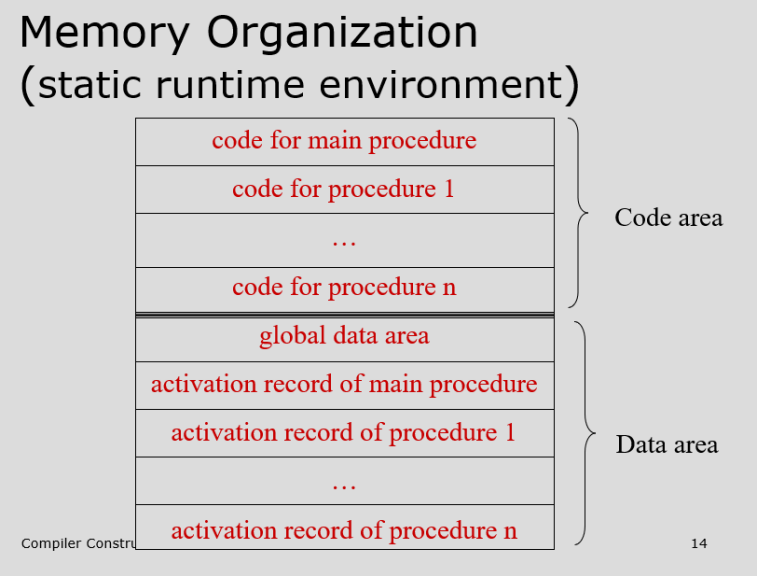
例子：FORTRAN77

最简单的运行时环境类型是所有数据都是静态的，且执行程序期间在存储器中保持固定。我们都可通过固定的地址直接访问所有的变量。

每个过程只有一个在执行之前被静态分配的活动记录。

这样的环境可用来实现没有指针或动态分配，且过程不可递归调用的语言

存储器组织如下：



基于栈的运行时环境（考点）

在允许递归调用以及每一个调用中都重新分配局部变量的语言中，不能静态地分配活动记录。相反地，必须以一个基于栈的风格来分配活动记录。这个栈被称为活动记录的栈(stack of activation record)(也指运行时栈(runtime stack) 或调用栈(call stack))每个过程每次在调用栈上可以有若干个不同的活动记录，每个都代表了一个不同的调用。

没有局部过程的基于栈的环境

在一个所有过程都是全局的语言 (例如C语言)中，基于栈的环境有两个要求：

- 指向当前活动记录的指针的维护允许访问局部变量
  - 指向当前活动的指针通常称为 框架指针(frame pointer) 或(fp)，且通常保存在寄存器中(通常也称作fp)
- 先前活动记录的位置或大小
  - 作为一个指向先前活动记录的指针，有关先前活动的信息一般是放在当前活动中，并被认为是控制链 (control link) 或动态链(dynamic link)
  - 有时将这个指针称为旧fp (old fp)，这是因为它代表了fp的先前值
- 此外，还有一个栈指针(stack pointer)或sp
  - 它通常指向调用栈上的最后位置

eg：一定要先读懂程序的输入和输出

右边的表格可以辅助理解但没必要写在卷子上

Example 7.2

```
#include <stdio.h>
int x,y;
int gcd(int u,int v)
{
    if (v == 0) return u;
    else return gcd(v, u%v);
}
main()
{
    scanf("%d%d", &x, &y);
    printf("%d\n", gcd(x,y));
    return 0;
}
```

Suppose the user inputs the values 15 and 10 to this program

u	v	u%v
15	10	5
10	5	0
5	0	End

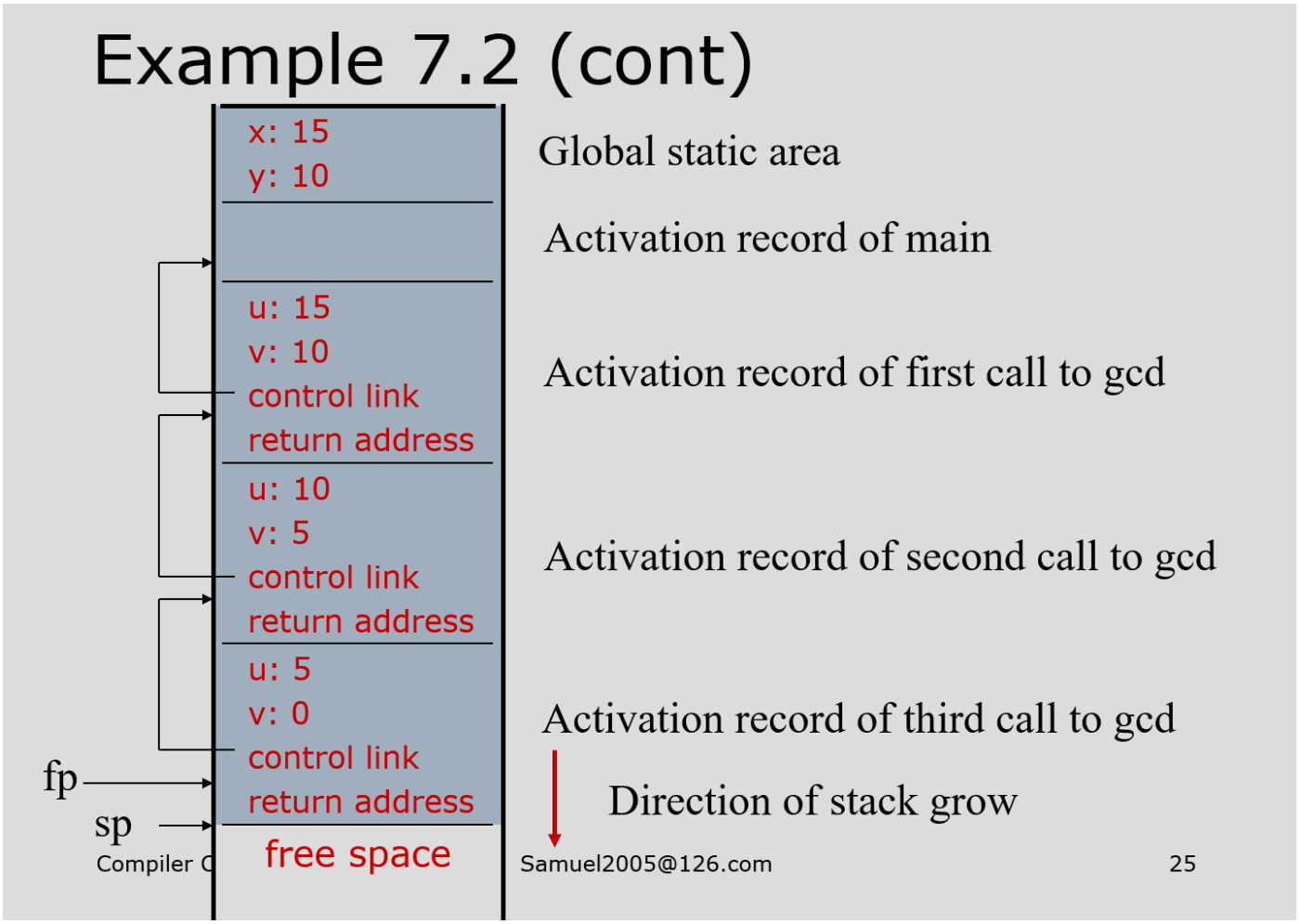
6.com 21

活动记录表格式：

活动记录
自变量（参数）空间
用作簿记信息的空间，包括了返回地址
用作局部数据的空间
用作局部临时变量的空间

这里一定要看动画演示，怎么做出来的：

【【哈尔滨工程大学】 编译原理（Compilation Principle）】
https://www.bilibili.com/video/BV19h4y1u7JL/?p=49&share_source=copy_web&vd_source=d80f309439bdd49df0b0808ccb80749a



eg:

【【哈尔滨工程大学】 编译原理（Compilation Principle）】
https://www.bilibili.com/video/BV19h4y1u7JL/?p=50&share_source=copy_web&vd_source=d80f309439bdd49df0b0808ccb80749a

活动树：

活动树是分析程序中复杂调用结构的有用工具，每个活动记录都成为该树的一个节点，而每个节点的子孙则代表了与该节点相对应的调用时进行的调用

上一个例题的活动树如下：