运行时环境与绑定

那AR和运行期的栈有什么关系？

AR（Procedure Active Record，过程活动记录）和**运行期栈（runtime stack）**有着密切的关系。它们共同工作，以管理程序在运行时的函数调用、局部变量、返回值等信息。具体来说，AR是栈的一部分，每次程序调用一个过程时，都会在栈中为该过程分配一个AR。以下是它们之间关系的详细解释：

1. 栈的作用：

栈是计算机内存的一部分，用来存储程序的**活动数据**，尤其是函数调用的相关信息。栈的特点是**后进先出**（LIFO，Last In, First Out），意味着最近的函数调用（或过程调用）会在栈的顶部，而已完成的调用会被从栈中移除。

2. AR与栈：

- 每当程序**进入一个过程**（即调用一个函数），程序会在栈的顶部分配一个新的**AR**。AR会包含与当前函数调用相关的所有信息，比如：
  - 局部变量
  - 临时变量
  - 返回地址
  - 参数
  - 返回值
  - 静态链和动态链等（用于管理作用域和函数调用的关系）。

  这个AR的目的是为了解决函数调用时需要存储的各类数据，确保函数调用完毕后能够正确返回。
- **栈的增长**：每次进入一个新过程时，AR被"推入"栈中，栈的大小随着函数调用的深度增长。
- **栈的收缩**：当一个过程**返回**时，栈顶的AR会被**弹出**，并释放其所占的空间。此时，程序会根据AR中的返回地址继续执行调用过程后的代码。

3. AR和栈的交互：

- **函数调用**：当程序调用一个函数时，栈顶会为该函数分配一个AR，AR存储了函数的局部信息。
- **函数返回**：当函数执行完毕并返回时，栈顶的AR会被释放，栈指针回到调用该函数之前的位置。返回地址存储在AR中，程序就能从这个地址继续执行。

4. 栈和递归：

在递归调用中，栈会不断增长，每次递归调用都会为每个函数调用分配一个新的AR，直到递归的终止条件满足。每当递归函数返回时，栈顶的AR会被释放，递归过程逐步回退。

总结：

- **AR**是栈中存储的一个数据结构，用来保存与过程调用相关的各种信息。
- **栈**是存储所有活动记录（包括AR）的地方，它帮助管理函数调用的顺序和作用域。
- 每当程序调用一个函数时，AR会被压入栈中；当函数执行完毕时，AR会被弹出栈，程序会继续执行返回后的代码。