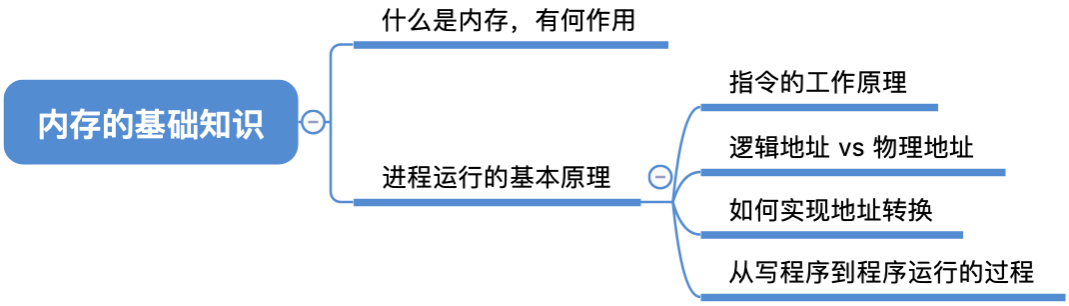
本节内容

内存的基础知识

知识总览

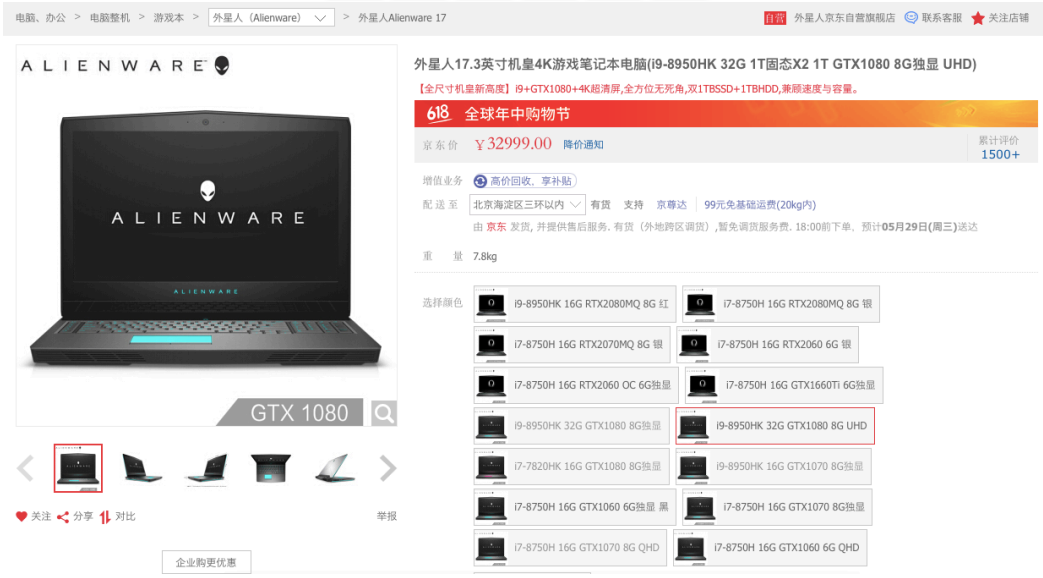


2019/6/26

什么是内存？有何作用？

王道考研/CSKAOYAN.COM

什么是内存？有何作用？



王道考研/CSKAOYAN.COM

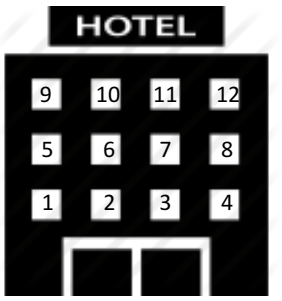
什么是内存？有何作用？

什么是内存？有何作用？

内存可存放数据。程序执行前需要先放到内存中才能被CPU处理——缓和CPU与硬盘之间的速度矛盾

思考：在多道程序环境下，系统中会有多个程序并发执行，也就是说会有多个程序的数据需要同时放到内存中。那么，如何区分各个程序的数据是放在什么地方的呢？

方案：给内存的存储单元编地址



内存地址从0开始，每个地址对应一个存储单元

地址	内存
0	“小房间”
1	“小房间”
2	……
3	
4	
5	
6	
……	

内存中也有一个一个的“小房间”，每个小房间就是一个“**存储单元**”

如果计算机“**按字节编址**”，则每个存储单元大小为1字节，即1B，即8个二进制位

如果字长为16位的计算机“**按字编址**”，则每个存储单元大小为1个字；每个字的大小为16个二进制位

补充知识：几个常用的数量单位

一台手机/电脑 有 4GB 内存，是什么意思？

是指该内存中可以存放 $4*2^{30}$ 个字节。如果是按字节编址的话，也就是有 $4*2^{30} = 2^{32}$ 个“小房间”

这么多“小房间”，需要 2^{32} 个地址才能一一标识，所以地址需要用 32 个二进制位来表示（0~ $2^{32}-1$）

补充知识：

2^{10} = 1K （千）

2^{20} = 1M （兆，百万）

2^{30} = 1G （十亿，千兆）

注：有的题目会告诉我们内存的大小，让我们确定地址长度应该是多少（即要多少个二进制位才能表示相应数目的存储单元）

王道考研/CSKAOYAN.COM

Tips：想深入了解可以学习汇编原理、计算机组成原理

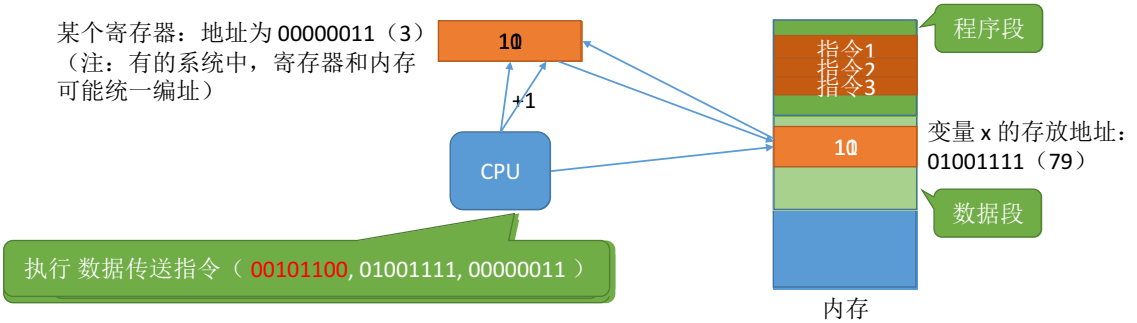
知识滚雪球：指令的工作原理

指令的工作基于“地址”。每个地址对应一个数据的存储单元

x = x + 1 → 编译 →

指令1（00101100, 00000011, 01001111） 数据传送指令
指令2（10010010, 00000011, 00000001） 加法指令
指令3（00101100, 01001111, 00000011） 数据传送指令

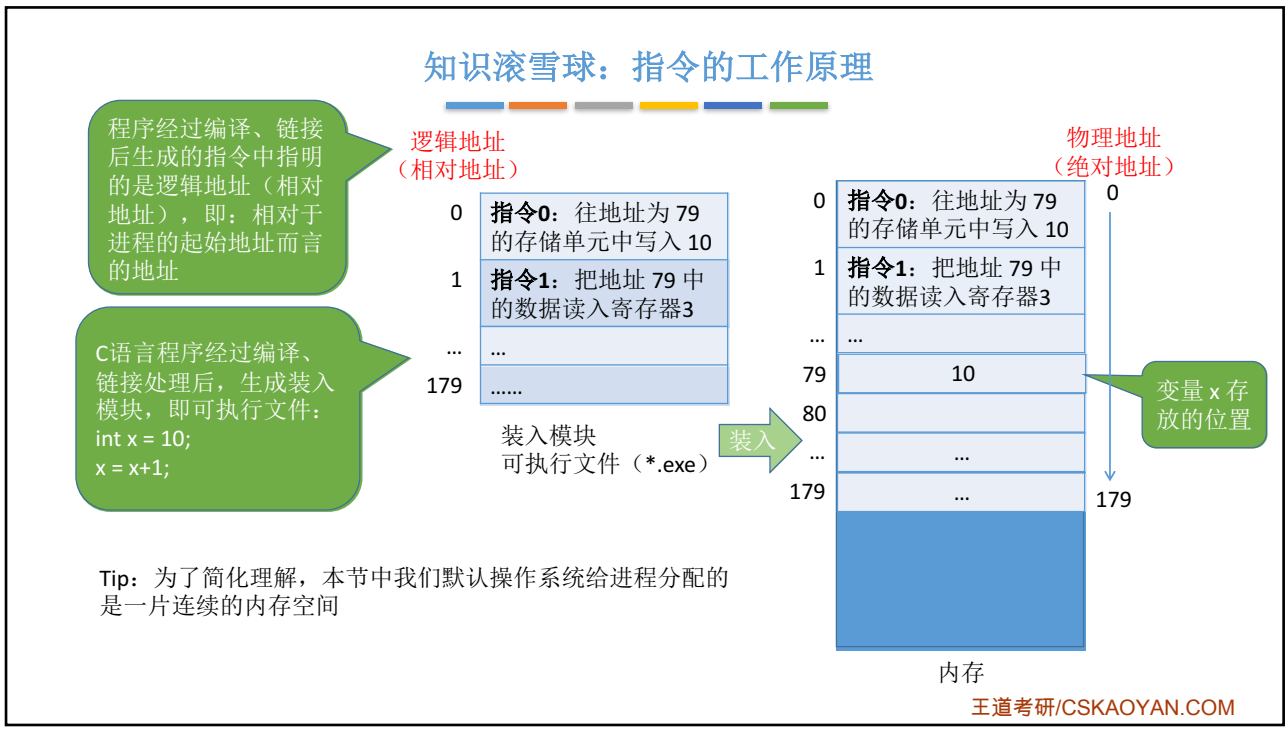
某个寄存器：地址为 00000011（3）（注：有的系统中，寄存器和内存可能统一编址）

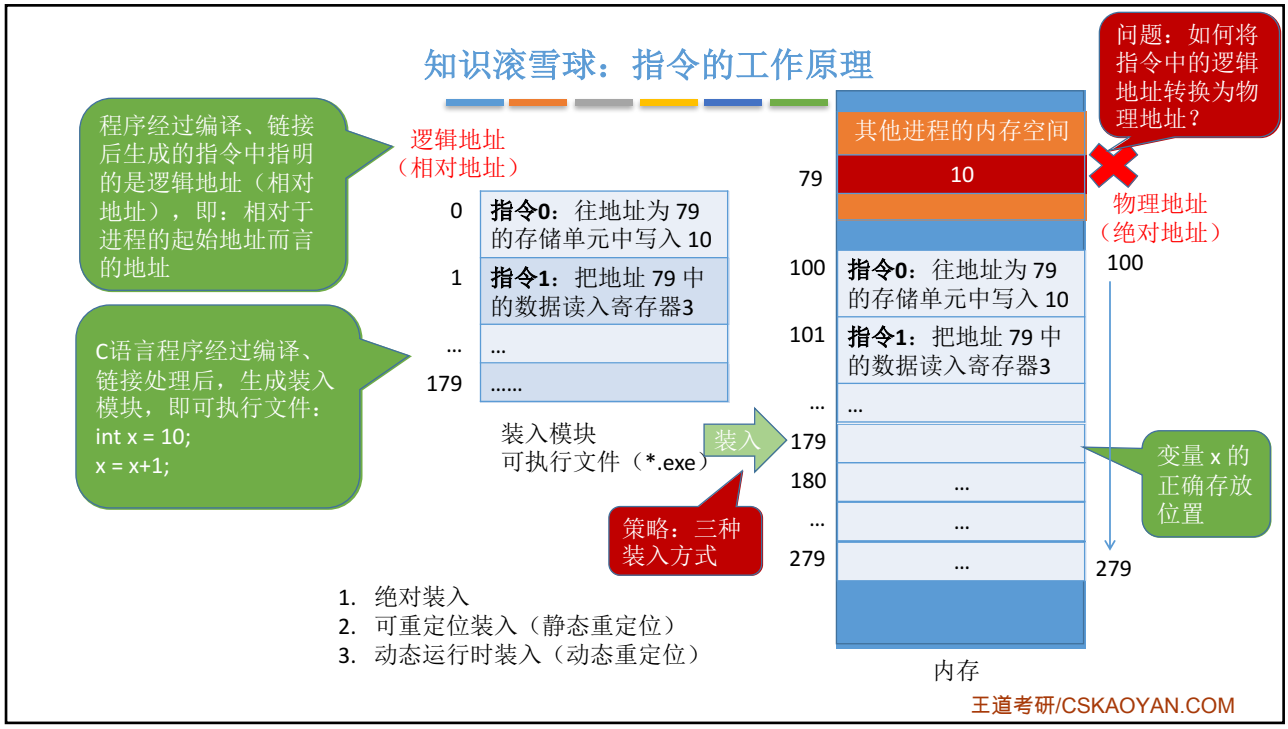


可见，我们写的代码要翻译成CPU能识别的指令。这些指令会告诉CPU应该去内存的哪个地址读/写数据，这个数据应该做什么样的处理。在这个例子中，我们默认让这个进程的相关内容从地址#0开始连续存放，指令中的地址参数直接给出了变量 x 的实际存放地址（物理地址）。

思考：如果这个进程不是从地址#0 开始存放的，会影响指令的正常执行吗？

王道考研/CSKAOYAN.COM





装入的三种方式——绝对装入

绝对装入：在编译时，如果知道程序将放到内存中的哪个位置，编译程序将产生绝对地址的目标代码。装入程序按照装入模块中的地址，将程序和数据装入内存。

Eg：如果知道装入模块要从地址为 100 的地方开始存放…

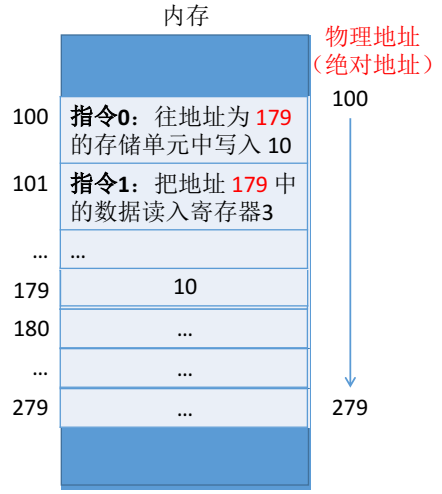
编译、链接后得到的装入模块的指令直接就使用了绝对地址

指令0：往地址为 79 的存储单元中写入 10
指令1：把地址 79 中的数据读入寄存器3
…
……

装入模块（可执行文件）

指令0：往地址为 179 的存储单元中写入 10
指令1：把地址 179 中的数据读入寄存器3
…
……

装入模块（可执行文件）



绝对装入只适用于单道程序环境。

程序中使用的绝对地址，可在编译或汇编时给出，也可由程序员直接赋予。通常情况下都是编译或汇编时再转换为绝对地址。

王道考研/CSKAOYAN.COM

装入的三种方式——可重定位装入

静态重定位：又称可重定位装入。编译、链接后的装入模块的地址都是从0开始的，指令中使用的地址、数据存放的地址都是相对于起始地址而言的逻辑地址。可根据内存的当前情况，将装入模块装入到内存的适当位置。装入时对地址进行“重定位”，将逻辑地址变换为物理地址（地址变换是在装入时一次完成的）。

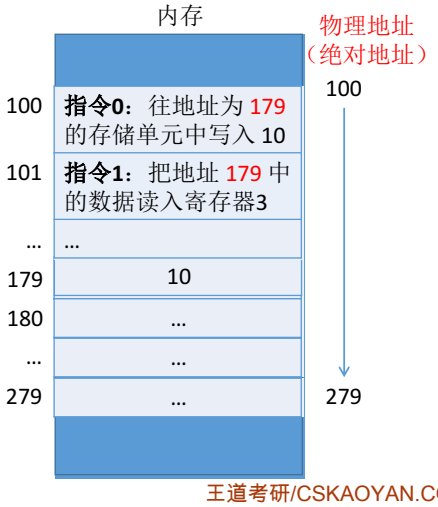
逻辑地址

0	**指令0**：往地址为 79 的存储单元中写入 10
1	**指令1**：把地址 79 中的数据读入寄存器3
…	…
179	……

装入模块（可执行文件）

装入的起始物理地址为100，则所有地址相关的参数都 +100

装入

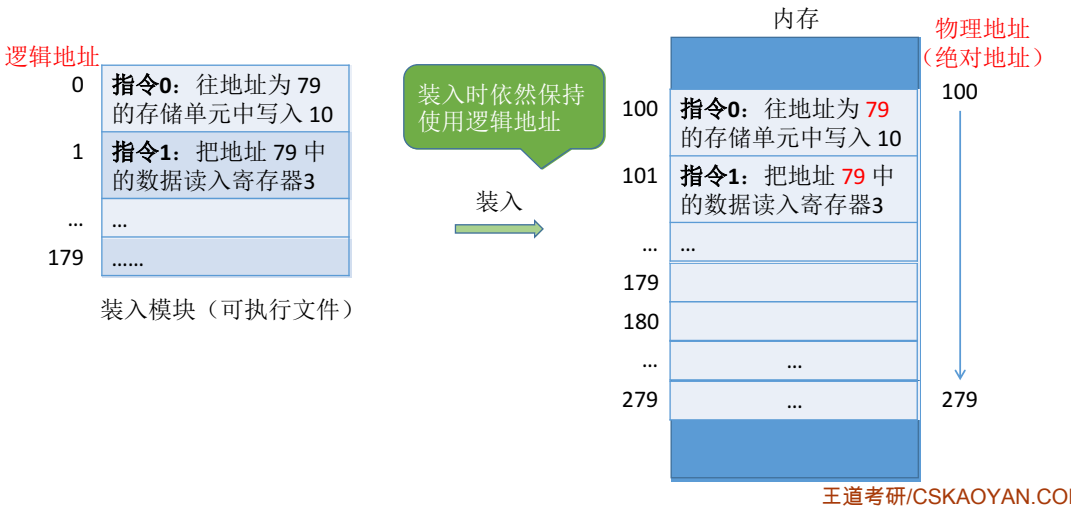


静态重定位的特点是在一个作业装入内存时，必须分配其要求的全部内存空间，如果没有足够的内存，就不能装入该作业。作业一旦进入内存后，在运行期间就不能再移动，也不能再申请内存空间。

王道考研/CSKAOYAN.COM

装入的三种方式——动态运行时装入

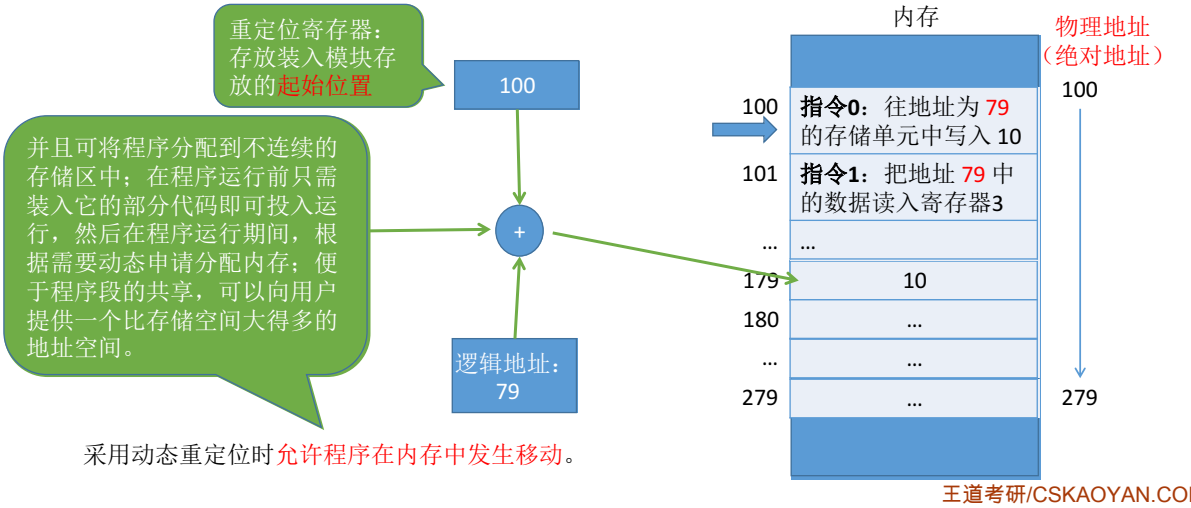
动态重定位：又称动态运行时装入。编译、链接后的装入模块的地址都是从0开始的。装入程序把装入模块装入内存后，并不会立即把逻辑地址转换为物理地址，而是把地址转换推迟到程序真正要执行时才进行。因此装入内存后所有的地址依然是逻辑地址。这种方式需要一个重定位寄存器的支持。



王道考研/CSKAOYAN.COM

装入的三种方式——动态重定位

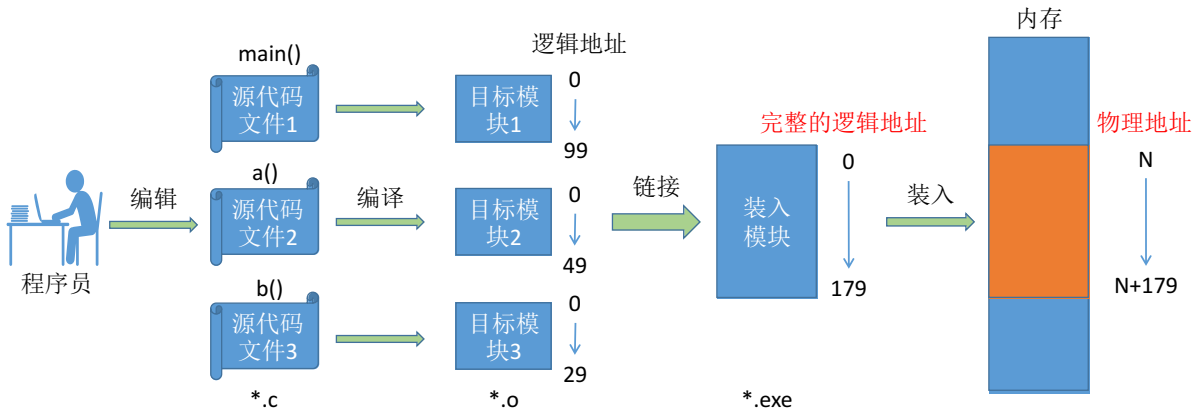
动态重定位：又称动态运行时装入。编译、链接后的装入模块的地址都是从0开始的。装入程序把装入模块装入内存后，并不会立即把逻辑地址转换为物理地址，而是把地址转换推迟到程序真正要执行时才进行。因此装入内存后所有的地址依然是逻辑地址。这种方式需要一个重定位寄存器的支持。



采用动态重定位时允许程序在内存中发生移动。

王道考研/CSKAOYAN.COM

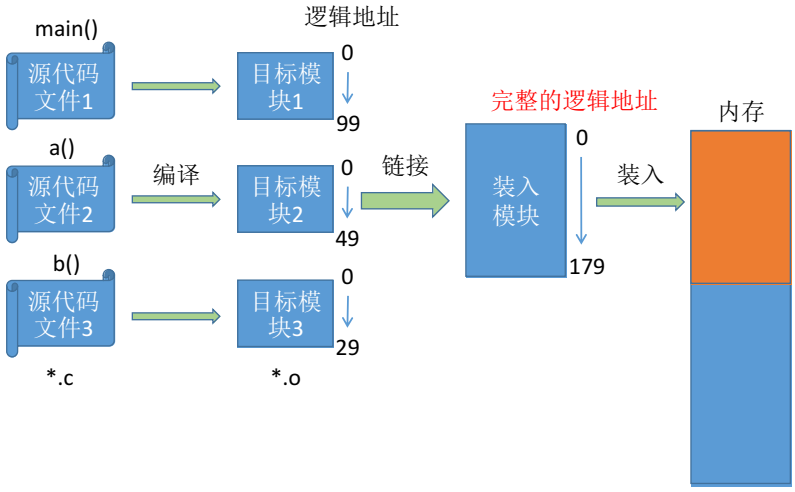
从写程序到程序运行



编译：由编译程序将用户源代码编译成若干个目标模块（编译就是把高级语言翻译为机器语言）
链接：由链接程序将编译后形成的一组目标模块，以及所需库函数链接在一起，形成一个完整的装入模块
装入（装载）：由装入程序将装入模块装入内存运行

王道考研/CSKAOYAN.COM

链接的三种方式

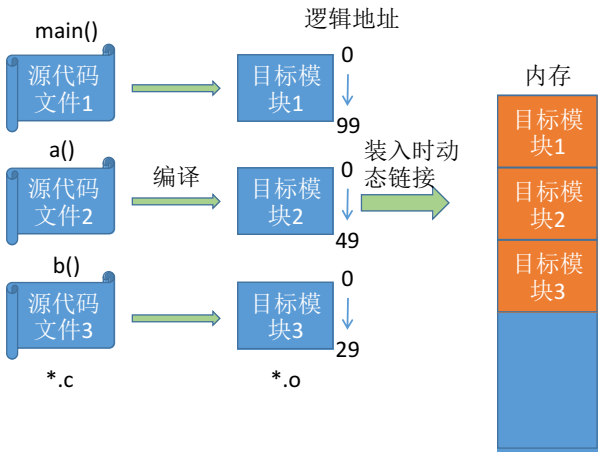


链接的三种方式：

1. 静态链接：在程序运行之前，先将各目标模块及它们所需的库函数连接成一个完整的可执行文件（装入模块），之后不再拆开。

王道考研/CSKAOYAN.COM

链接的三种方式

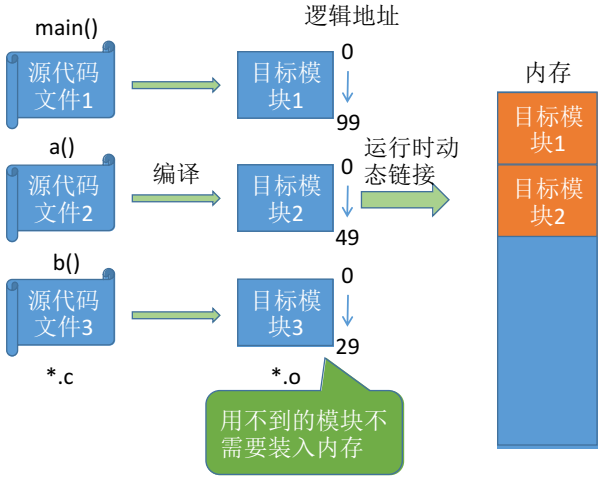


链接的三种方式：

1. 静态链接：在程序运行之前，先将各目标模块及它们所需的库函数连接成一个完整的可执行文件（装入模块），之后不再拆开。
2. 装入时动态链接：将各目标模块装入内存时，边装入边链接的链接方式。

王道考研/CSKAOYAN.COM

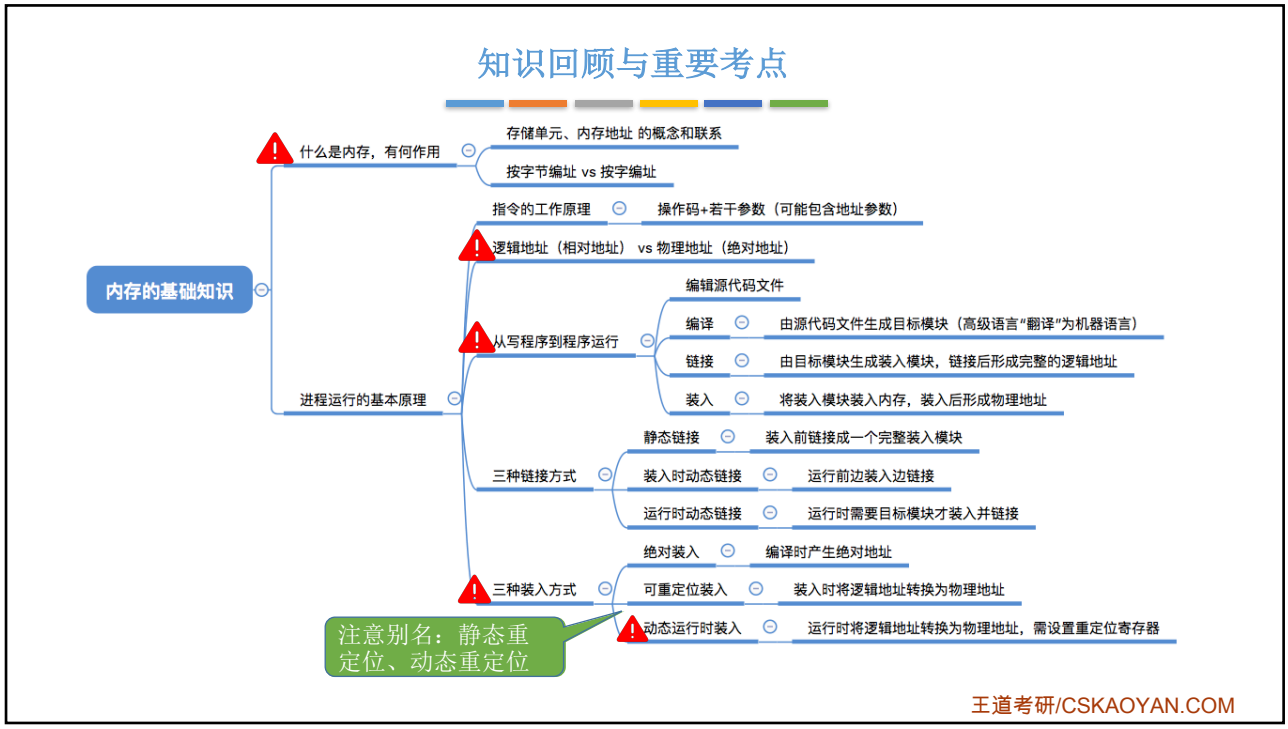
链接的三种方式



链接的三种方式：

1. 静态链接：在程序运行之前，先将各目标模块及它们所需的库函数连接成一个完整的可执行文件（装入模块），之后不再拆开。
2. 装入时动态链接：将各目标模块装入内存时，边装入边链接的链接方式。
3. 运行时动态链接：在程序执行中需要该目标模块时，才对它进行链接。其优点是便于修改和更新，便于实现对目标模块的共享。

王道考研/CSKAOYAN.COM


知识回顾与重要考点
内存的基础知识
什么是内存，有何作用
存储单元、内存地址 的概念和联系
按字节编址 vs 按字编址
进程运行的基本原理
指令的工作原理
操作码+若干参数（可能包含地址参数）
逻辑地址（相对地址） vs 物理地址（绝对地址）
从写程序到程序运行
编辑源代码文件
编译
由源代码文件生成目标模块（高级语言“翻译”为机器语言）
链接
由目标模块生成装入模块，链接后形成完整的逻辑地址
装入
将装入模块装入内存，装入后形成物理地址
三种链接方式
静态链接
装入前链接成一个完整装入模块
装入时动态链接
运行前边装入边链接
运行时动态链接
运行时需要目标模块才装入并链接
三种装入方式
绝对装入
编译时产生绝对地址
可重定位装入
装入时将逻辑地址转换为物理地址
动态运行时装入
运行时将逻辑地址转换为物理地址，需设置重定位寄存器
注意别名：静态重定位、动态重定位
王道考研/CSKAOYAN.COM