

本节内容

内存的基础 知识

知识总览

内存的基础知识

什么是内存，有何作用

进程运行的基本原理

指令的工作原理

逻辑地址 vs 物理地址

如何实现地址转换

从写程序到程序运行的过程

什么是内存？有何作用？



LEICA
HUAWEI

新品 华为 HUAWEI P30 超感光徕卡三摄麒麟980AI智能芯片全面屏屏内指纹版手机8GB+64GB亮黑色全网通双4G手机双

现在购机享受价保至6月18日，618提前购，安心购买！屏内指纹，感光徕卡三摄mate20优惠200，到手价3299起，还有赠品

618 全球年中购物节

京东价 **¥3988.00** 降价通知

促销 **满送** 满100元即赠热销商品，赠完即止

满额返券 购买此商品满10元返配件品类优惠券（送完为止）详情 >>

增值业务 **高价回收-卖了换钱** **套 3元1G**

配送至 北京海淀区三环以内 有货

由 **京东** 发货，并提供售后服务。23:00前下单，预计明天(05月28日)送达

重量 0.48kg

服务支持 **自营放心购** 免举证退换货 原厂维修 ①

京尊达 99元免基础运费(20kg内) 京准达 自提

选择颜色  天空之境  **亮黑色**  极光色  赤茶橘  珠光贝母

选择版本 **8GB+64GB** 8GB+128GB 8GB+256GB

选择版本 **标准版** 碎屏险套装版 京享无忧版 特惠版

套 装 优惠套装1 优惠套装2 优惠套装3 优惠套装4 优惠套装5 优惠套装6


举报

关注 分享 对比

什么是内存？有何作用？

电脑、办公 > 电脑整机 > 游戏本 > 外星人 (Alienware) > 外星人Alienware 17

自营 外星人京东自营旗舰店 联系客服 关注店铺



ALIENWARE

GTX 1080

外星人17.3英寸机皇4K游戏笔记本电脑(i9-8950HK 32G 1T固态X2 1T GTX1080 8G独显 UHD)

【全尺寸机皇新高度】i9+GTX1080+4K超清屏,全方位无死角,双1TBSSD+1TBHDD,兼顾速度与容量。

618 全球年中购物节

京东价 ¥32999.00 降价通知

累计评价 1500+















增值业务 高价回收,享补贴






配送至 北京海淀区三环以内 有货 支持 京尊达 99元免基础运费(20kg内)

由 京东 发货,并提供售后服务,有货(外地跨区调货),暂免调货服务费,18:00前下单,预计05月29日(周三)送达

重量 7.8kg

选择颜色

 i9-8950HK 16G RTX2080MQ 8G 红	 i7-8750H 16G RTX2080MQ 8G 银
 i7-8750H 16G RTX2070MQ 8G 银	 i7-8750H 16G RTX2060 6G 银
 i7-8750H 16G RTX2060 OC 6G独显	 i7-8750H 16G GTX1660Ti 6G独显
 i9-8950HK 32G GTX1080 8G独显	 i9-8950HK 32G GTX1080 8G UHD
 i7-7820HK 16G GTX1080 8G独显	 i9-8950HK 16G GTX1070 8G独显
 i7-8750H 16G GTX1060 6G独显 黑	 i7-8750H 16G GTX1070 8G独显
 i7-8750H 16G GTX1070 8G QHD	 i7-8750H 16G GTX1060 6G QHD



关注 分享 对比

举报

企业购更优惠

什么是内存？有何作用？



❤ 关注 ➦ 分享 1️⃣ 对比

举报

京东物流 三星 (SAMSUNG) DDR4 2400 2133 4G 8G 四代笔记本内存条 三星
原厂正品 DDR4 2133 4GB

品牌机原厂内存供应商 买就买个真的内存 京东自营仓直发 原厂正品内存 不烧机 吃鸡不蓝屏

京东价 **¥355.00** 降价通知

累计评
3700

促销 **赠品** x 1 x 1 (赠完即止)

增值业务 以旧换新，卖了换钱

配送至 北京海淀区五环到六环之间 有货 支持 货到付款 免运费

由 京东 发货，本尚网来数码专营店 提供售后服务。11:10前下单，预计今天(08月03日)送达

选择颜色



增值保障



1

+

加入购物车

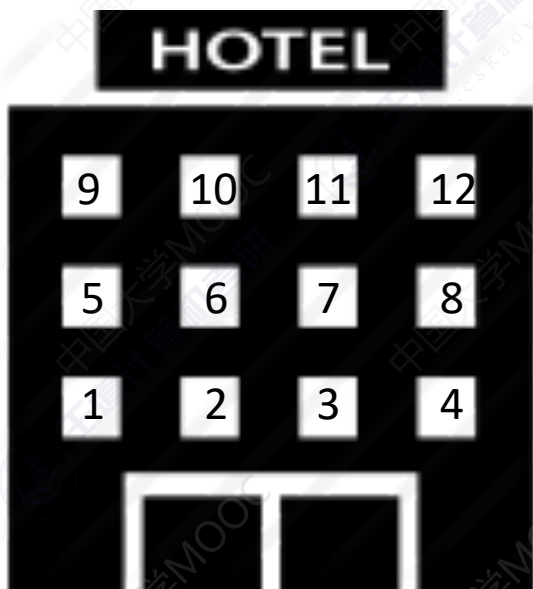
什么是内存？有何作用？

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

0
1
2
3
4
5
6
.....

内存

“小房间”

“小房间”

.....

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 $4 \times 2^{30} = 2^{32}$ 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（ $0 \sim 2^{32} - 1$ ）

补充知识：

$$2^{10} = 1K$$

（千）

$$2^{20} = 1M$$

（兆，百万）

$$2^{30} = 1G$$

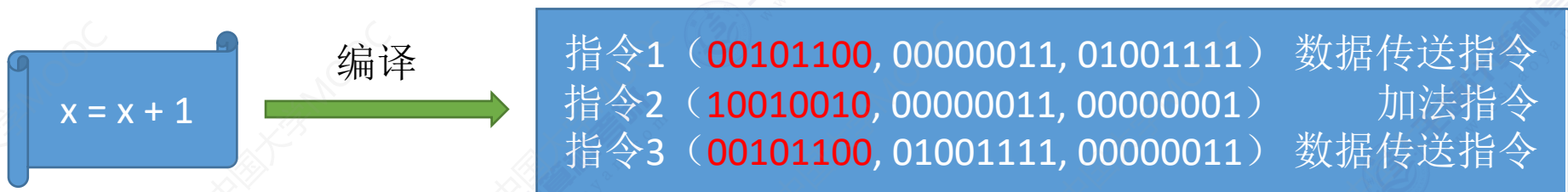
（十亿，千兆）

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）

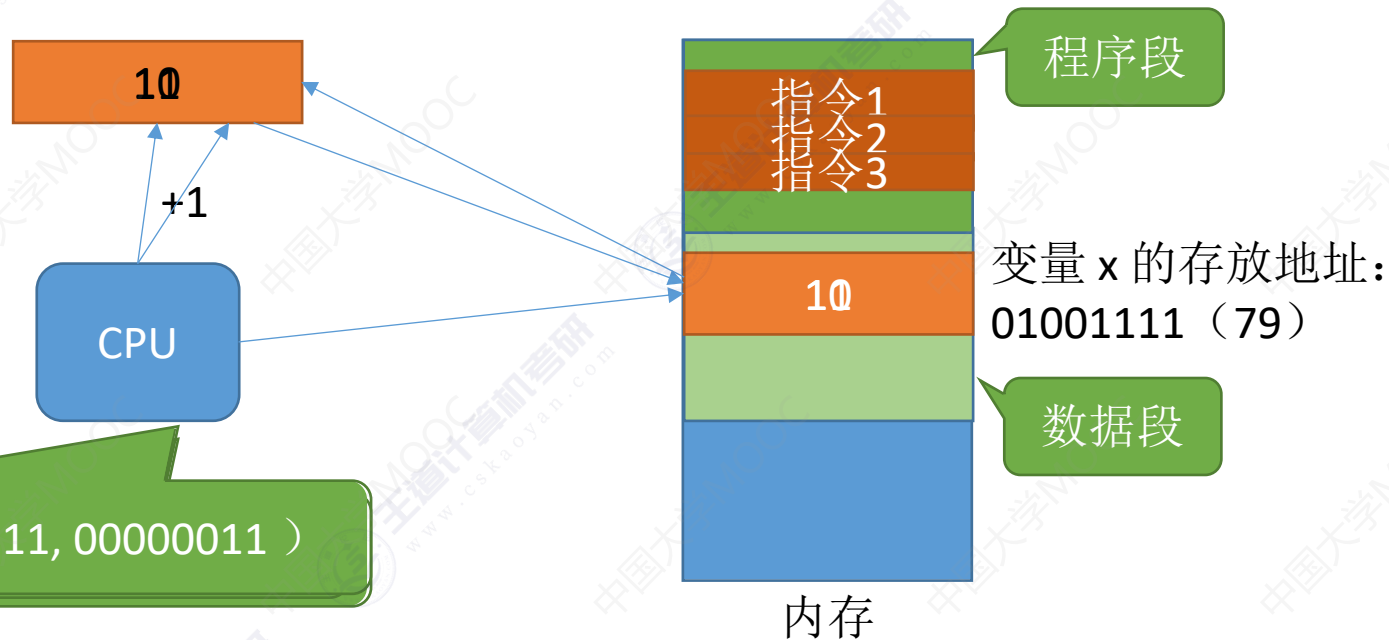
Tips: 想深入了解可以学习汇编原理、计算机组成原理

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



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)



执行 数据传送指令 (00101100, 01001111, 00000011)

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？

知识滚雪球：指令的工作原理

程序经过编译、链接后生成的指令中指明的是逻辑地址（相对地址），即：相对于进程的起始地址而言的地址

C语言程序经过编译、链接处理后，生成装入模块，即可执行文件：
`int x = 10;`
`x = x+1;`

逻辑地址
(相对地址)

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 (*.exe)

装入

物理地址
(绝对地址)

0	指令0: 往地址为 79 的存储单元中写入 10	0
1	指令1: 把地址 79 中的数据读入寄存器3	
...	...	
79	10	
80		
...	...	
179	...	179

变量 x 存放的位置

内存

Tip: 为了简化理解，本节中我们默认操作系统给进程分配的是一片连续的内存空间

知识滚雪球：指令的工作原理

程序经过编译、链接后生成的指令中指明的是逻辑地址（相对地址），即：相对于进程的起始地址而言的地址

C语言程序经过编译、链接处理后，生成装入模块，即可执行文件：
`int x = 10;`
`x = x+1;`

逻辑地址
(相对地址)

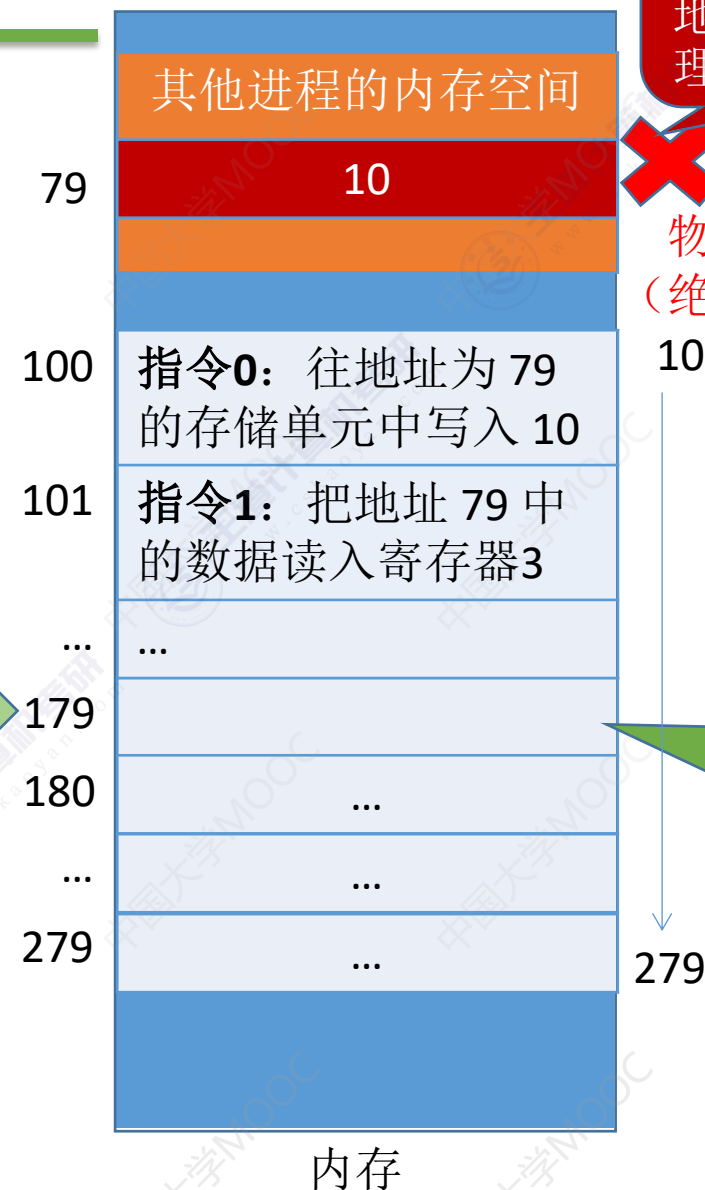
0	指令0: 往地址为 79 的存储单元中写入 10
1	指令1: 把地址 79 中的数据读入寄存器3
...	...
179

装入模块
可执行文件 (*.exe)

装入

策略：三种
装入方式

1. 绝对装入
2. 可重定位装入（静态重定位）
3. 动态运行时装入（动态重定位）



问题：如何将指令中的逻辑地址转换为物理地址？

物理地址
(绝对地址)

变量 x 的
正确存放
位置

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

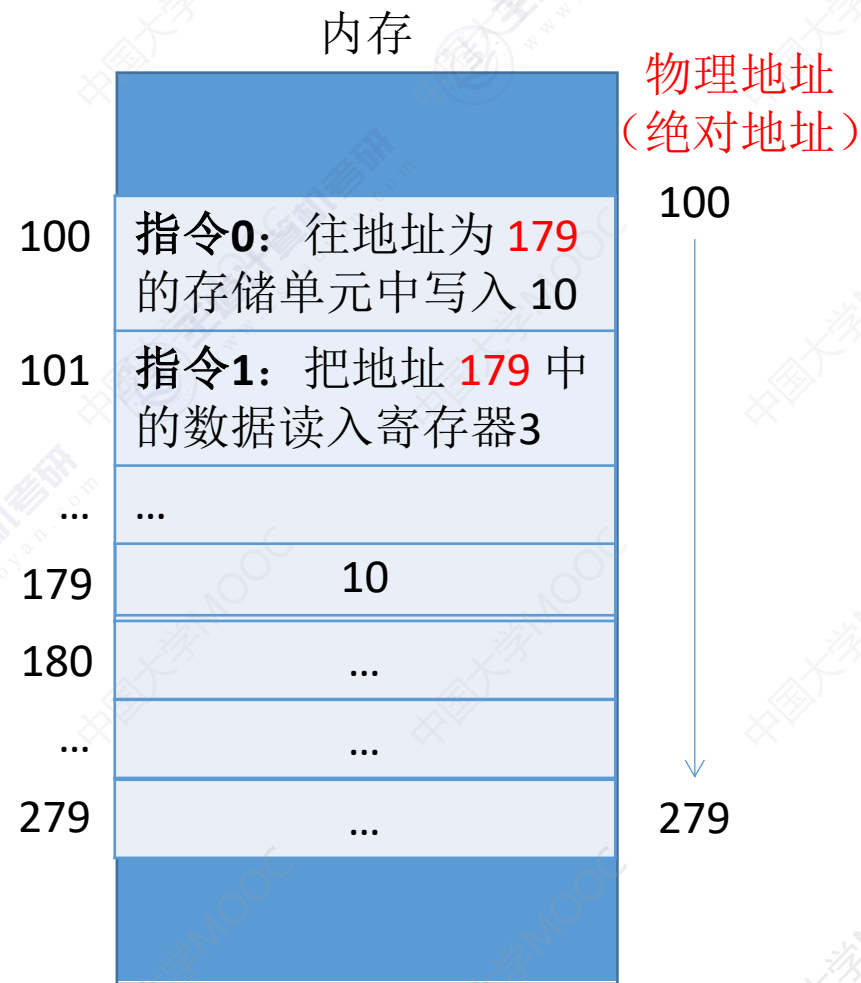
...

.....

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。



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

内存

物理地址
(绝对地址)

100	指令0: 往地址为 179 的存储单元中写入 10	100
101	指令1: 把地址 179 中的数据读入寄存器3	
...	...	
179	10	
180	...	
...	...	
279	...	279

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。

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。

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）

装入时依然保持使用逻辑地址

装入

内存

物理地址
(绝对地址)

100	指令0: 往地址为 79 的存储单元中写入 10	100
101	指令1: 把地址 79 中的数据读入寄存器3	
...	...	
179		
180		
...	...	
279		279

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。

重定位寄存器：
存放装入模块存放的**起始位置**

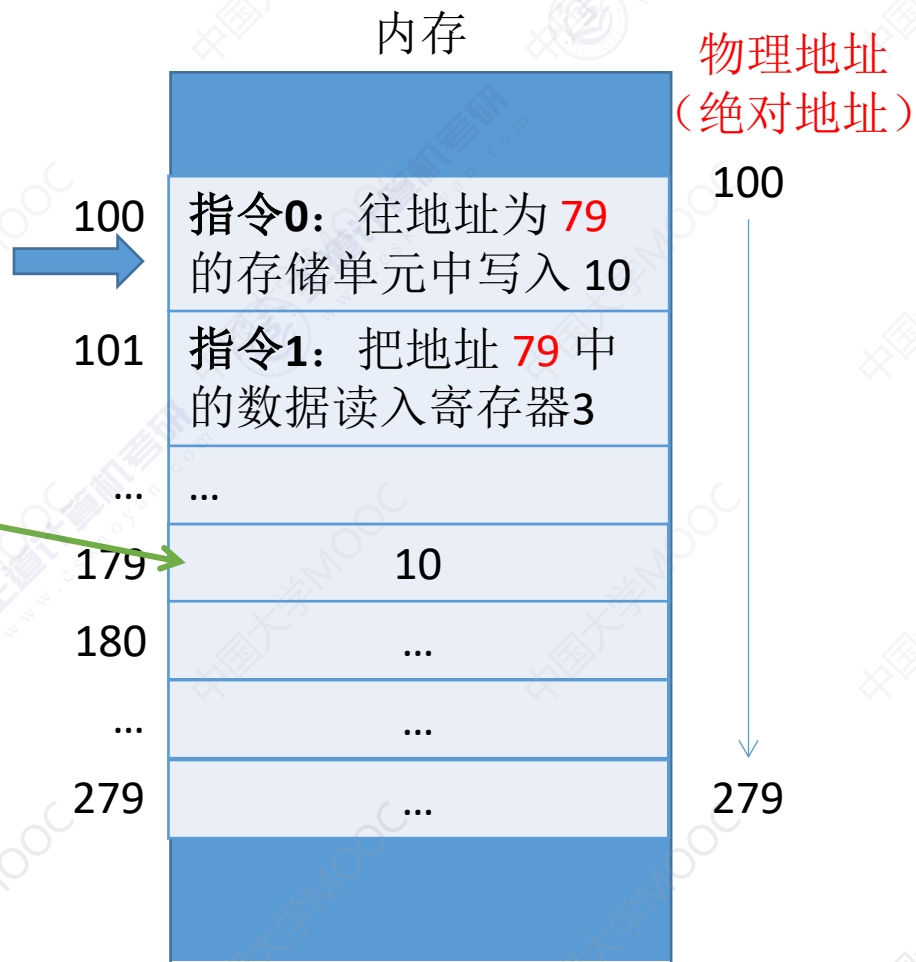
并且可将程序分配到不连续的存储区中；在程序运行前只需装入它的部分代码即可投入运行，然后在程序运行期间，根据需要动态申请分配内存；便于程序段的共享，可以向用户提供一个比存储空间大得多的地址空间。

100

100

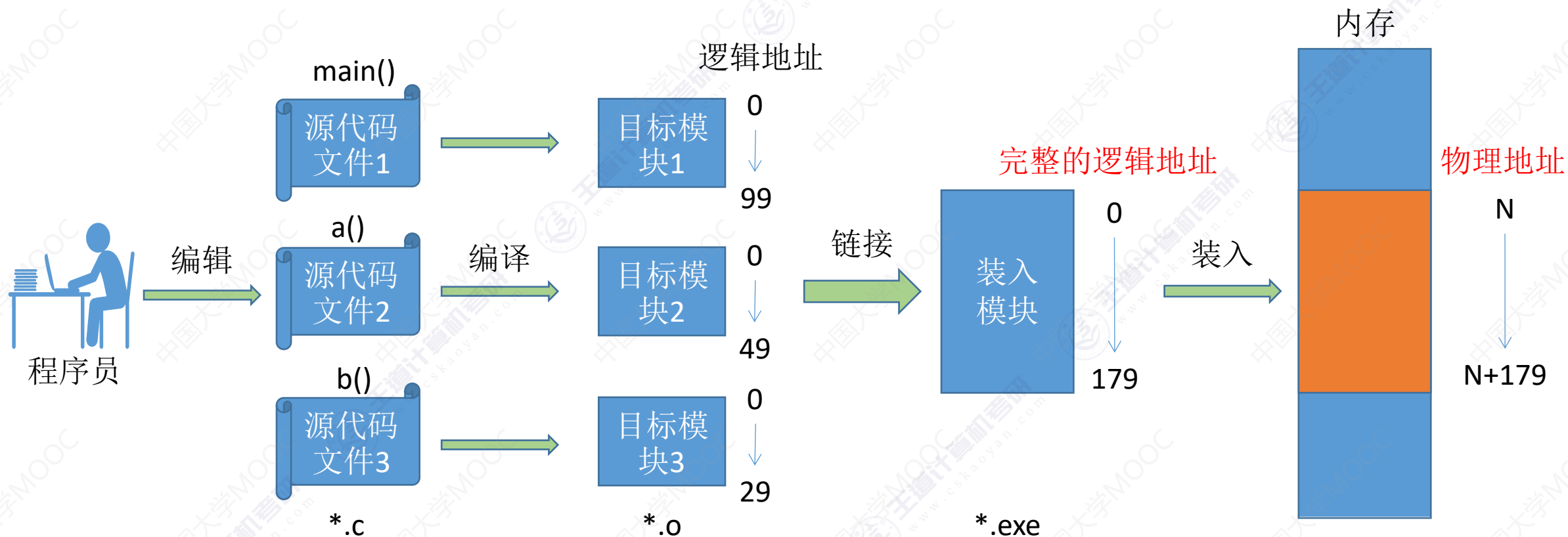
+

逻辑地址：
79



采用动态重定位时**允许程序在内存中发生移动**。

从写程序到程序运行

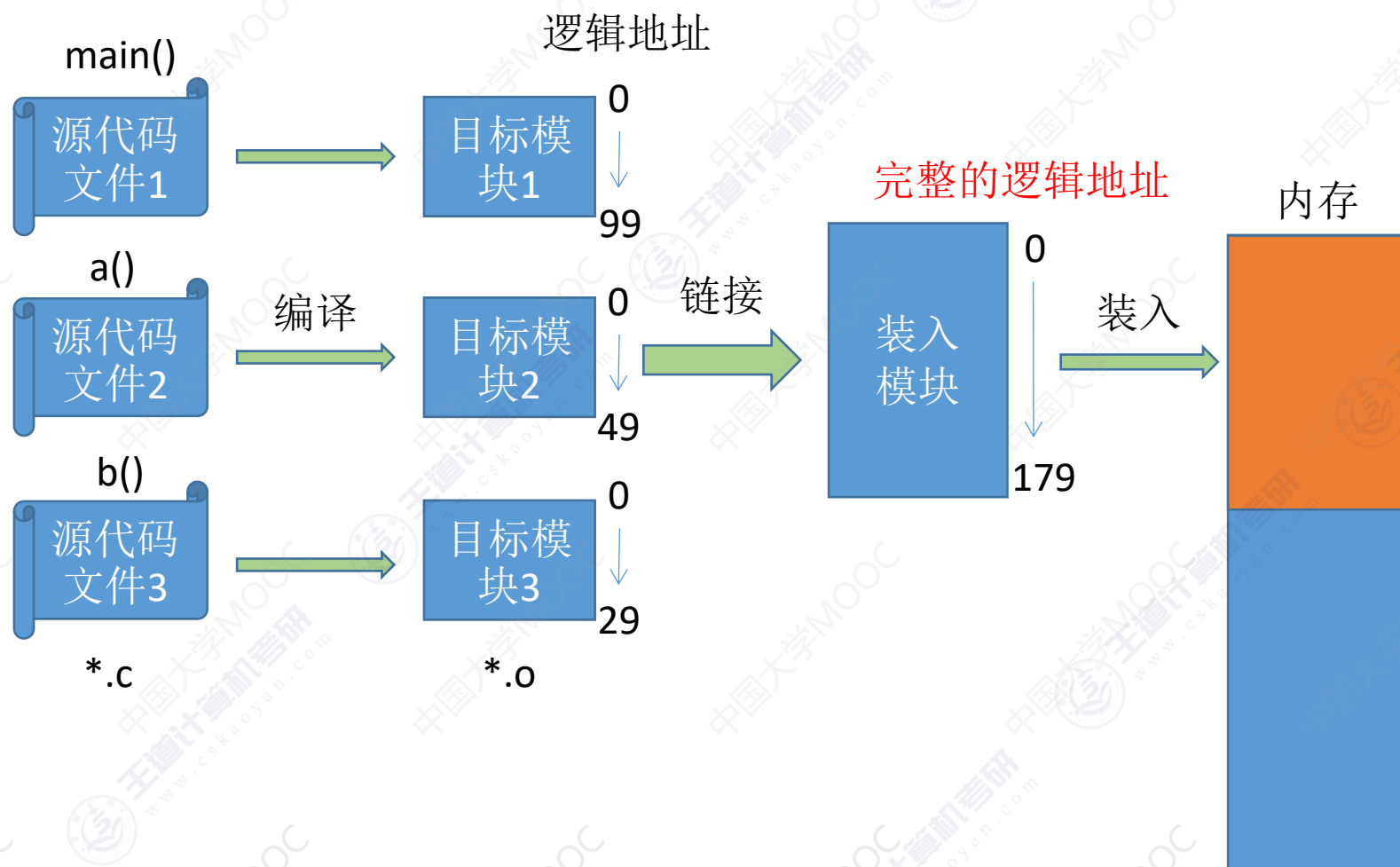


编译：由编译程序将用户源代码编译成若干个目标模块（编译就是把高级语言**翻译为机器语言**）

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

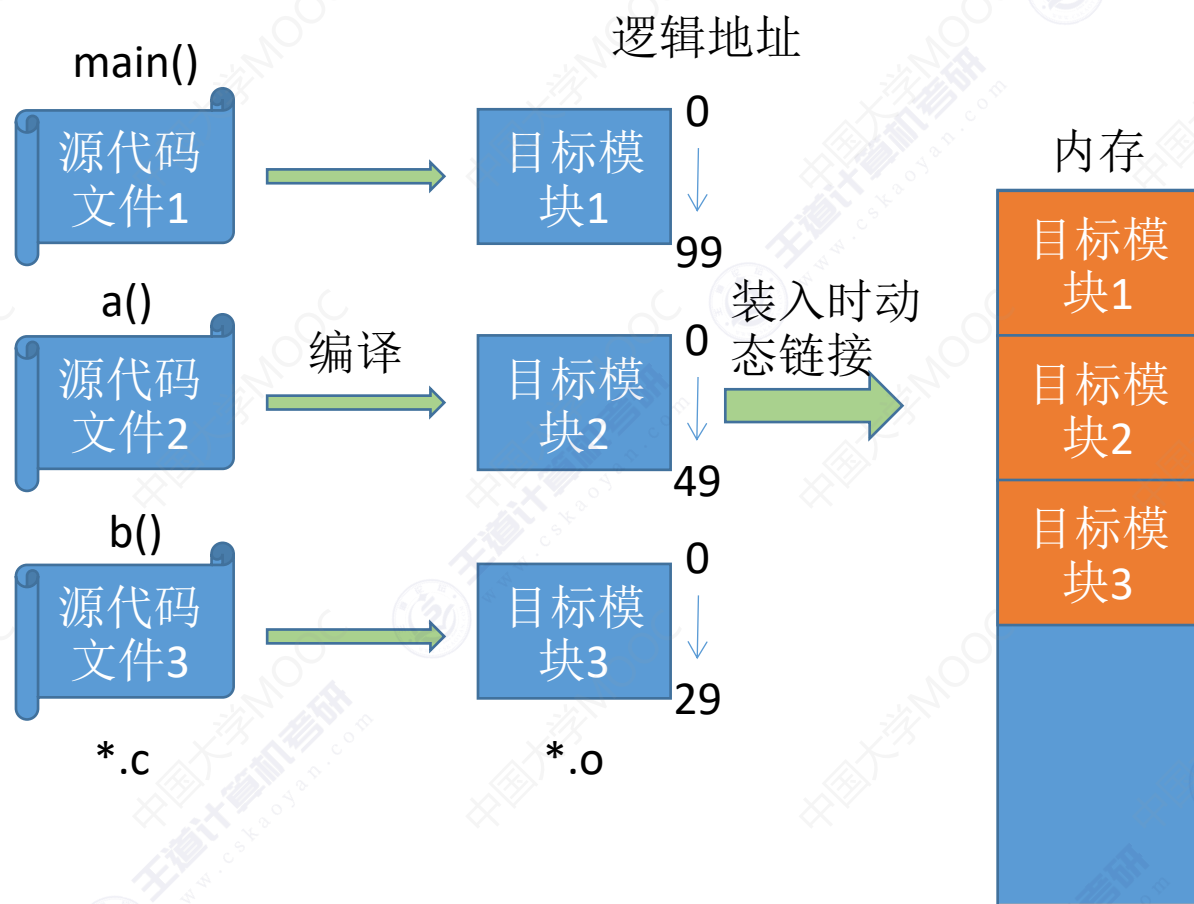
链接的三种方式



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。

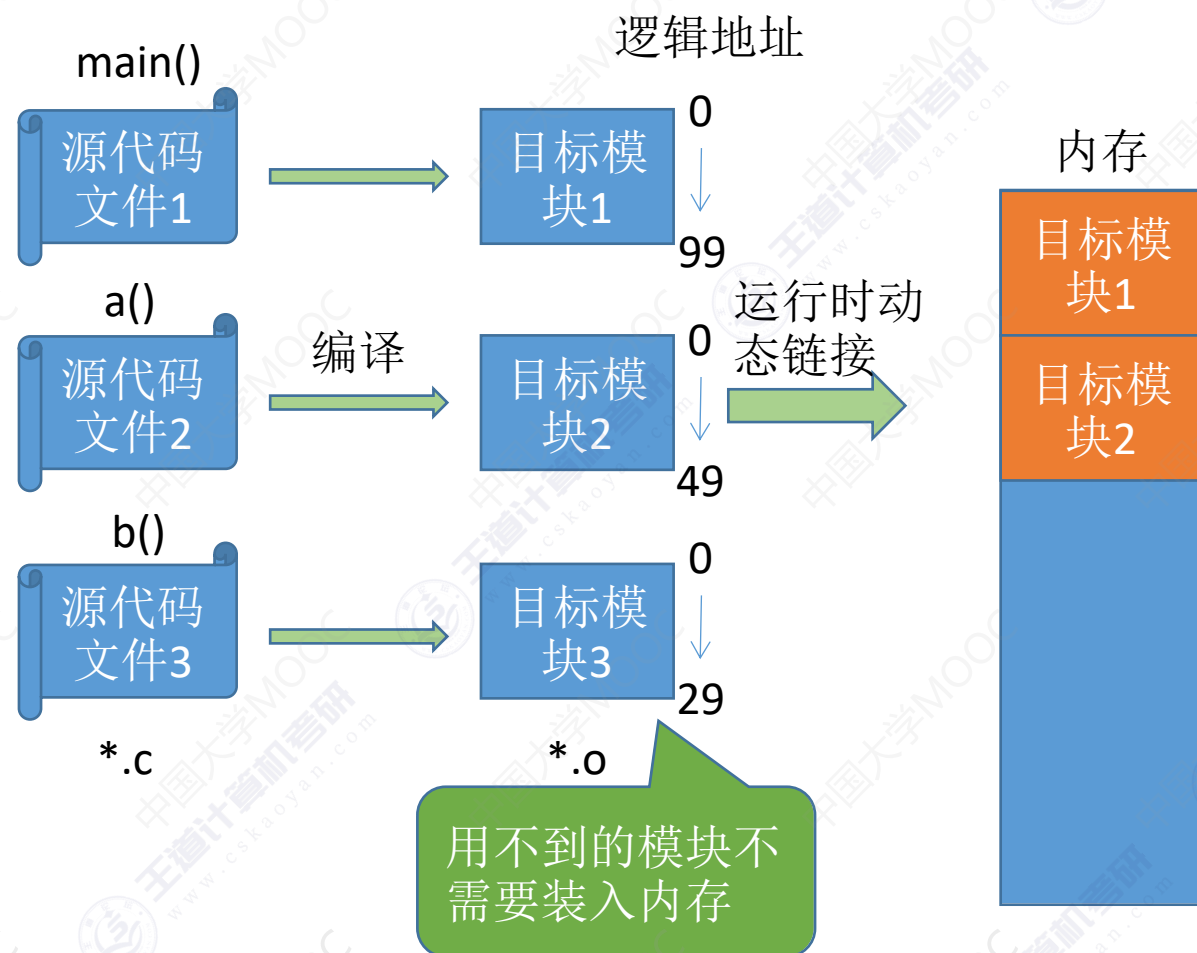
链接的三种方式



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, 边装入边链接的连接方式。

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, 边装入边链接的连接方式。
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。

知识回顾与重要考点





公众号：王道在线



b站：王道计算机教育



抖音：王道计算机考研