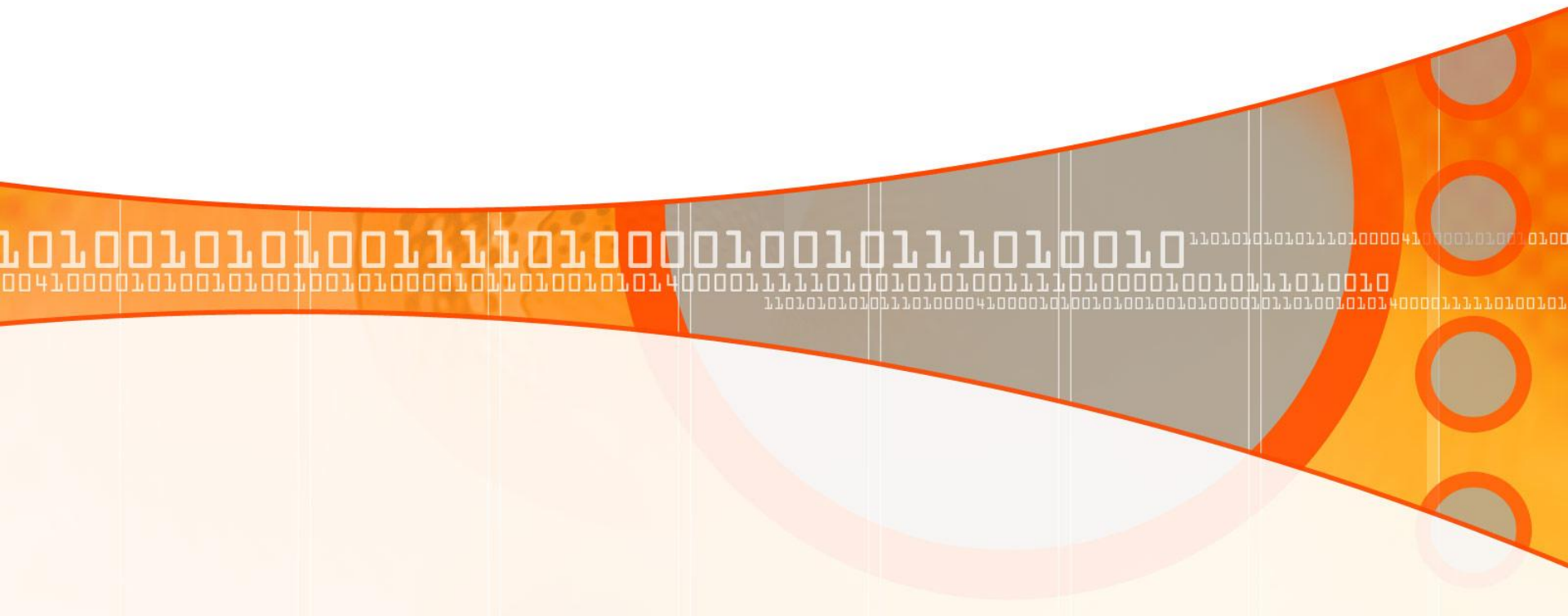
第4章 存储器管理



- 用户程序的主要处理阶段
- 连续分配方式
- 离散分配方式（分页式、分段式、页段式）
- 虚拟存储器的基本特征

手机通讯 > 手机 > 手机 > 华为（HUAWEI） > 华为（HUAWEI）Ma...　五星店铺　京联通达旗舰店　联系客服　关注店铺

新品

关注　对比　　举报

新品 华为Mate 70 旗舰手机 mate70新品上市【现货当天发】 红枫原色影像 超可靠玄武架构华为鸿蒙智能手机 曜石黑 12G+512GB 官方标配

【现货当天发，重磅新品，咨询客服享豪礼，晒单领50元红包】鸿蒙AI、红枫原色影像、超可靠玄武架构

出游季 又好又便宜

京 东 价　¥ 5999.00　降价通知　　累计评价 5000+

促　　销　赠品　× 1　× 1　× 1　（条件：购买1件及以上，赠完即止）

增值业务　高价回收，极速到账

配 送 至　江苏南京市高淳区淳溪街道　有货　支持 晚发赔 | 全国联保 | 一年质保 | 7天价保

包邮（请以提交订单时为准）

由 京联通达旗舰店 发货，并提供售后服务. 承诺今天（3月22日）24:00前发货，预计3月25日24:00前送达

选择外观　曜石黑　雪域白　云杉绿　风信紫（无货）

选择版本　12G+256GB（无货）　12G+512GB　12G+1TB　12GB+512GB 鸿蒙 NEXT 先锋版

套餐　官方标配　12期分期免息　24期分期免息

京选服务　2年碎屏4年电池 ¥304.00　2年碎屏保 ¥279.00　2年电池换新 ¥49.00　1年延长保 ¥67.00　学到老保到老 ¥345.00　全家贴膜无忧服务 ¥59.00

白条分期　不分期　¥2047.67 x 3期　¥1047.84 x 6期　¥547.98 x 12期　¥298.14 x 24期

1　加入购物车　立即购买

温馨提示　·支持7天无理由退货(激活后不支持)

承　　诺　"公益好物"公益捐赠





关注 对比 举报

企业购更优惠

自营 **ThinkPad【国家补贴20%】T14p AI PC 酷睿Ultra9 高性能工程师本笔记本电脑 32G 1TB 3K 商务办公本**

【AI高性能专业工程师本】英特尔酷睿Ultra9-185H！3K+120Hz高刷屏！高色彩高色准屏幕！内存硬盘可扩展【点击查看抢购】查看>

国家补贴 再减¥1890 立即领

价 格 ¥ **9451.51** PLUS到手价 ¥ 9499.00 京东价 降价通知

累计评价 5万+

促 销 PLUS专享立减 可与PLUS价、满减、券等优惠叠加使用 详情>>

满额返券 实付满1元，收货后返满1打9折券，先到先得。若点击跳转后的页面展示券面额则为用券商品页，如无券面额则为凑单商品页或活动页。 详情>>

增值业务 高价回收，极速到账

配 送 至 江苏南京市高淳区淳溪街道 有货 支持 59元免基础运费 | 可配送全球 | PLUS 180天只换不修 | 7天价保

京东物流 明日达 | 预约送货 | 部分收货

由 京东 发货，ThinkPad京东自营旗舰店 提供售后服务. 23:10前付款，预计明天(03月23日)送达

选择颜色 高性能AI工程师本 | 高性能经典工程师本

选择版本 Ultra5-125H 32G 1TB 2.5K 90Hz | Ultra7-155H 32G 1TB 3K RTX4050 | 推荐 Ultra9-185H 32G 1TB 3K | Ultra9 64G 2TB 3K 120Hz | Ultra9-185H 32G 1TB 3K RTX4050 | Ultra9 32G 2TB 3K RTX4050 | Ultra9 64G 2TB 3K RTX4050 | 推荐 i5-13500H 32G 1TB | 推荐 i9-13900H 32G 1TB | i9-13900H 32G 1TB RTX3050

京选服务 4年全面保修 ¥459.00 | 2年免费换新 ¥429.00 | 延长3年保修 ¥149.00 | 替换至64G ¥1975.00 | 服务送背包 ¥75.00

白条分期 不分期 | ¥3226.12 x 3期 | ¥1650.87 x 6期 | ¥863.30 x 12期 | ¥469.49 x 24期

1 + - 加入购物车 立即购买



关注 对比 举报

企业购更优惠

自营 三星 SAMSUNG 台式机内存条 32G DDR4 3200频率

出游季 又好又便宜

价　格 ¥ 576.11 PLUS到手价 ¥ 579.00 京东价 降价通知

累计评价 5万+

促　销 PLUS专享立减 可与PLUS价、满减、券等优惠叠加使用 详情>>

增值业务 高价回收，极速到账

配 送 至 江苏南京市高淳区淳溪街道 有货 支持 59元免基础运费 | 可配送全球 | PLUS 180天只换不修 | 7天价保

京东物流 京准达 | 211限时达 | 明日达

由 京东 发货，并提供售后服务. 23:10前付款，预计明天(03月23日)送达

选择内存频率

- 台式机内存,8GB,2666MHz 无货
- 台式机内存,8GB,3200MHz
- 台式机内存,16GB,3200MHz
- 台式机内存,32GB,3200MHz
- 台式机内存,8GB*2片装,3200MHz
- 台式机内存,16GB*2片装,3200MHz

京选服务

- 4年全保换新 ¥66.00
- 3年免费换新 ¥55.00
- 5年性能保修 ¥18.00
- 30天试用 ¥69.00
- 3年复购补贴 ¥41.80
- 1年数据恢复 ¥49.00
- 基础版装机 ¥148.00

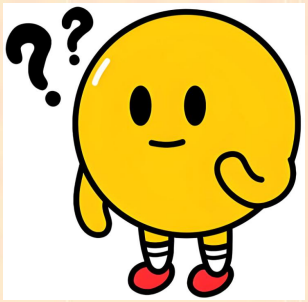
白条分期 不分期 | ¥ 196.66 x 3期 | ¥ 100.67 x 6期 | ¥ 52.72 x 12期 | ¥ 28.72 x 24期

1 + -

加入购物车 立即购买

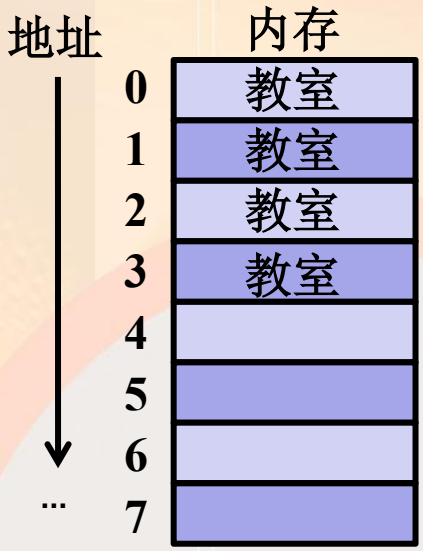
什么是内存，有何作用？

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
--缓冲CPU和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？

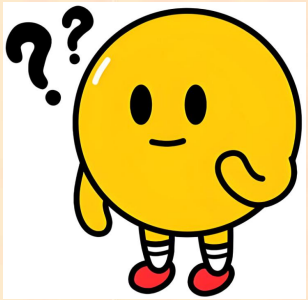
给内存的存储单元编地址



内存中也有一间一间的“教室”，每个教室就是一个“存储单元”

内存地址从0开始，每个地址对应一个存储单元。如果计算机按照字节编址，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 $4*2^{30} = 2^{32}$ 间教室

2^{32}间教室，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$(0\sim2^{32}-1)$

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）

存储器管理的功能

1. 存储分配和回收：是存储管理的主要内容。讨论其算法和相应的数据结构。
2. 地址变换：可执行文件生成中的链接技术、程序加载时的重定位技术，进程运行时硬件和软件的地址变换技术和机构。
3. 存储共享和保护：代码和数据共享，对地址空间的访问权限（读、写、执行）。
4. 存储器扩充：它涉及存储器的逻辑组织和物理组织。
   - 由应用程序控制：覆盖；
   - 由OS控制：交换（整个进程空间），请求调入和预调入（部分进程空间）

地址空间

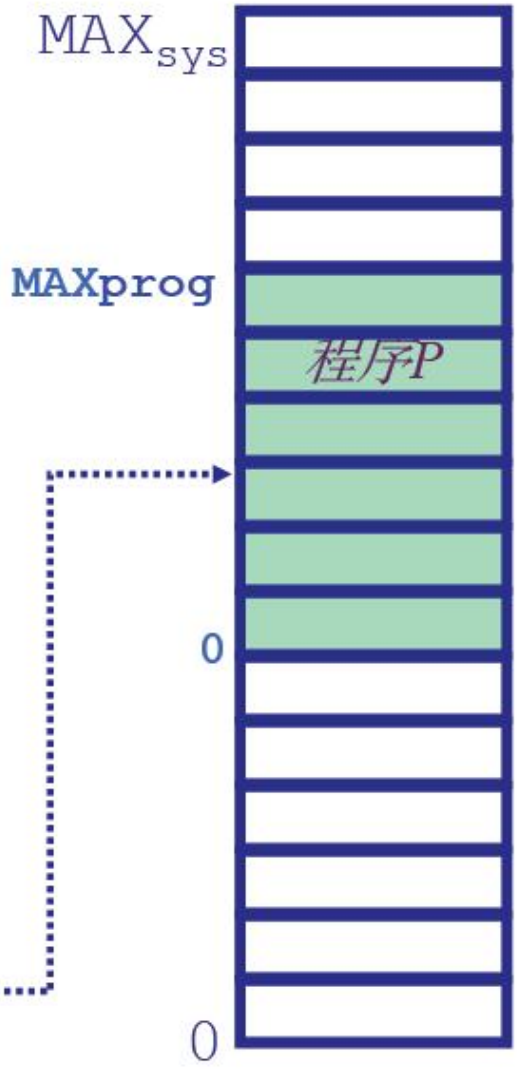
- *物理地址空间* — 硬件支持的地址空间
  - 起始地址0，直到 MAX_{sys}

Main memory　　Disk

- *逻辑地址空间* — 运行中的应用程序能观察到的地址
  - 起始地址0， 直到 MAX_{prog}

但是地址(address)是从哪里来的?

```
movl %eax, $0xffa620e
```

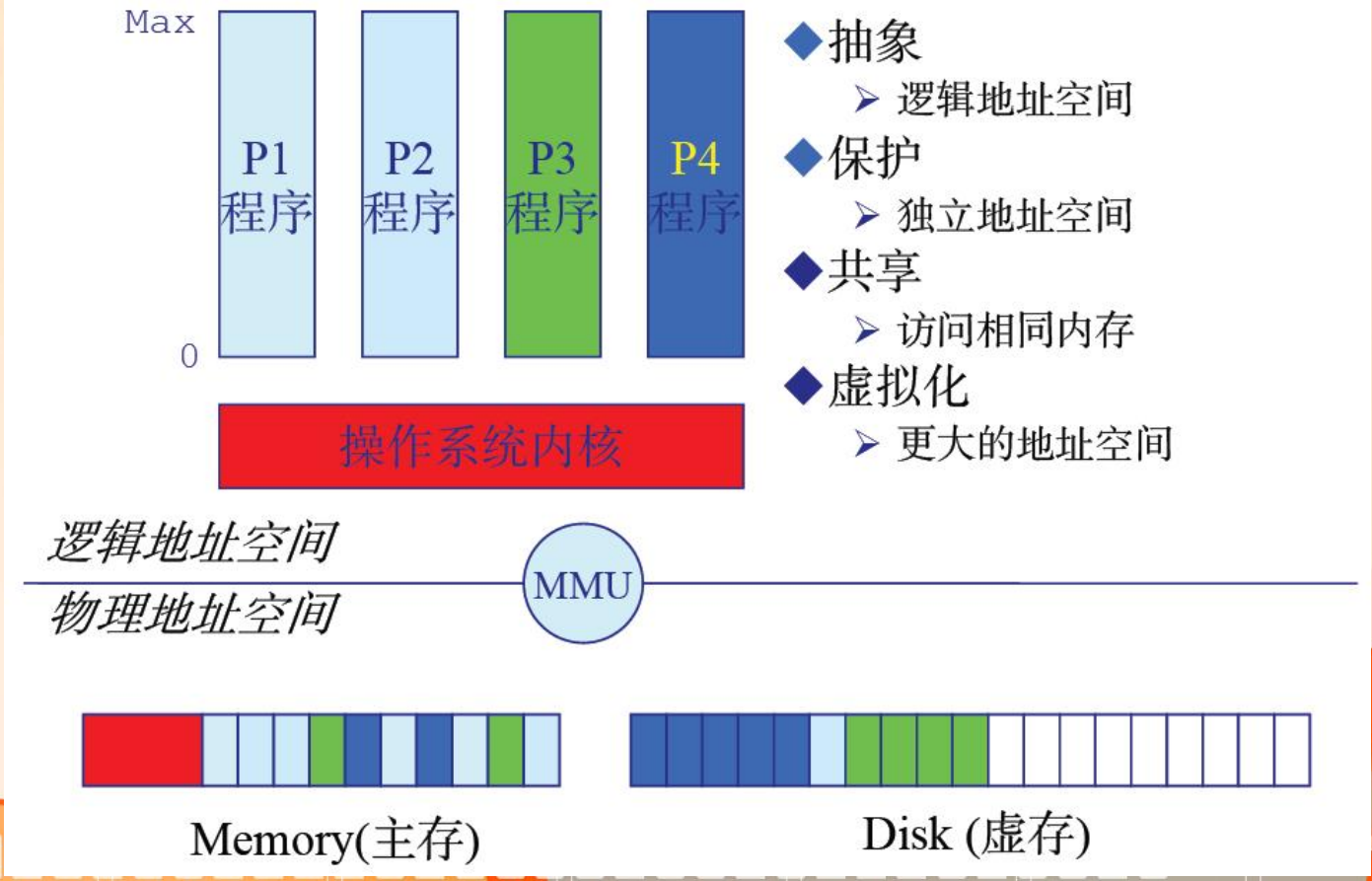


逻辑地址和物理地址

- 将逻辑地址空间与物理地址空间相分离，是内存管理的核心。
- 逻辑地址（虚地址）与物理地址是相同：
  - 地址映像工作在编译阶段或加载阶段完成
- 重定位
  - 进程的逻辑地址空间不同于物理地址空间，所以存储管理模块（MMU）要解决逻辑地址到物理地址的映射问题
  - 也称地址映射、地址映像
  - 在执行阶段完成

逻辑地址空间与物理地址空间

- 逻辑地址，也称虚地址（virtual address）、相对地址
  - 由CPU执行指令时生成的地址（本条指令所需数据的地址或下一条指令的地址）
- 物理地址，也称绝对地址、实地址
  - 实际的内存单元地址



指令的工作原理

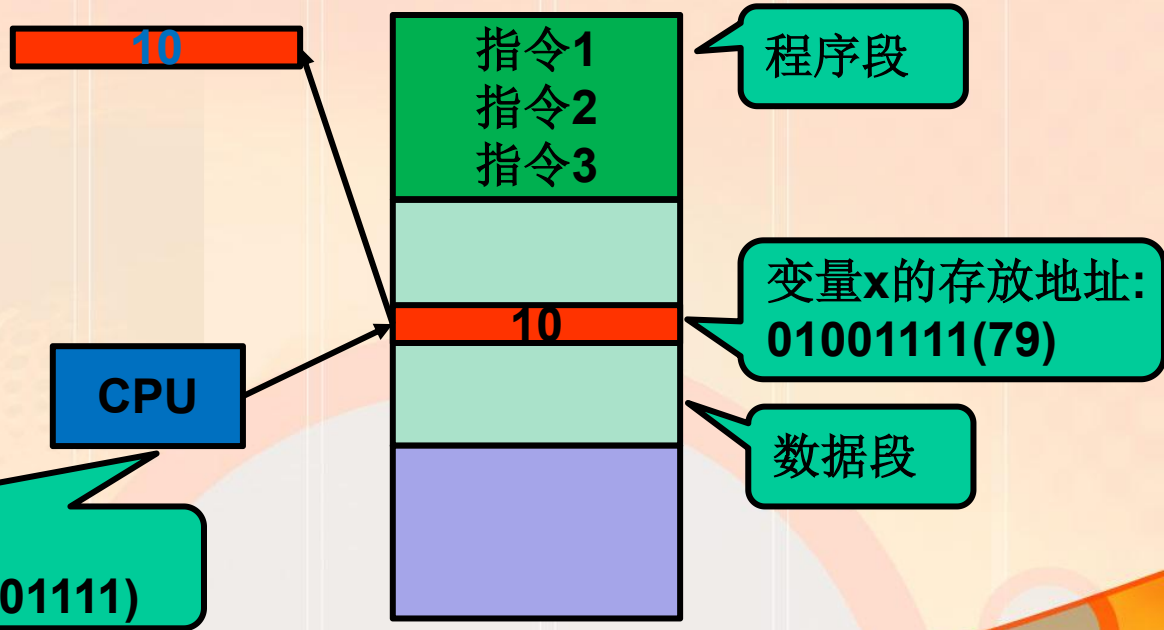
指令的工作基于“地址”。每个地址对应一个数据的存储单元

x=x+1

编译

指令1(00101100, 00000011, 01001111) 数据传送指令
指令2(10010010, 00000011, 01001111) 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。



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

操作码	目的寄存器	源寄存器

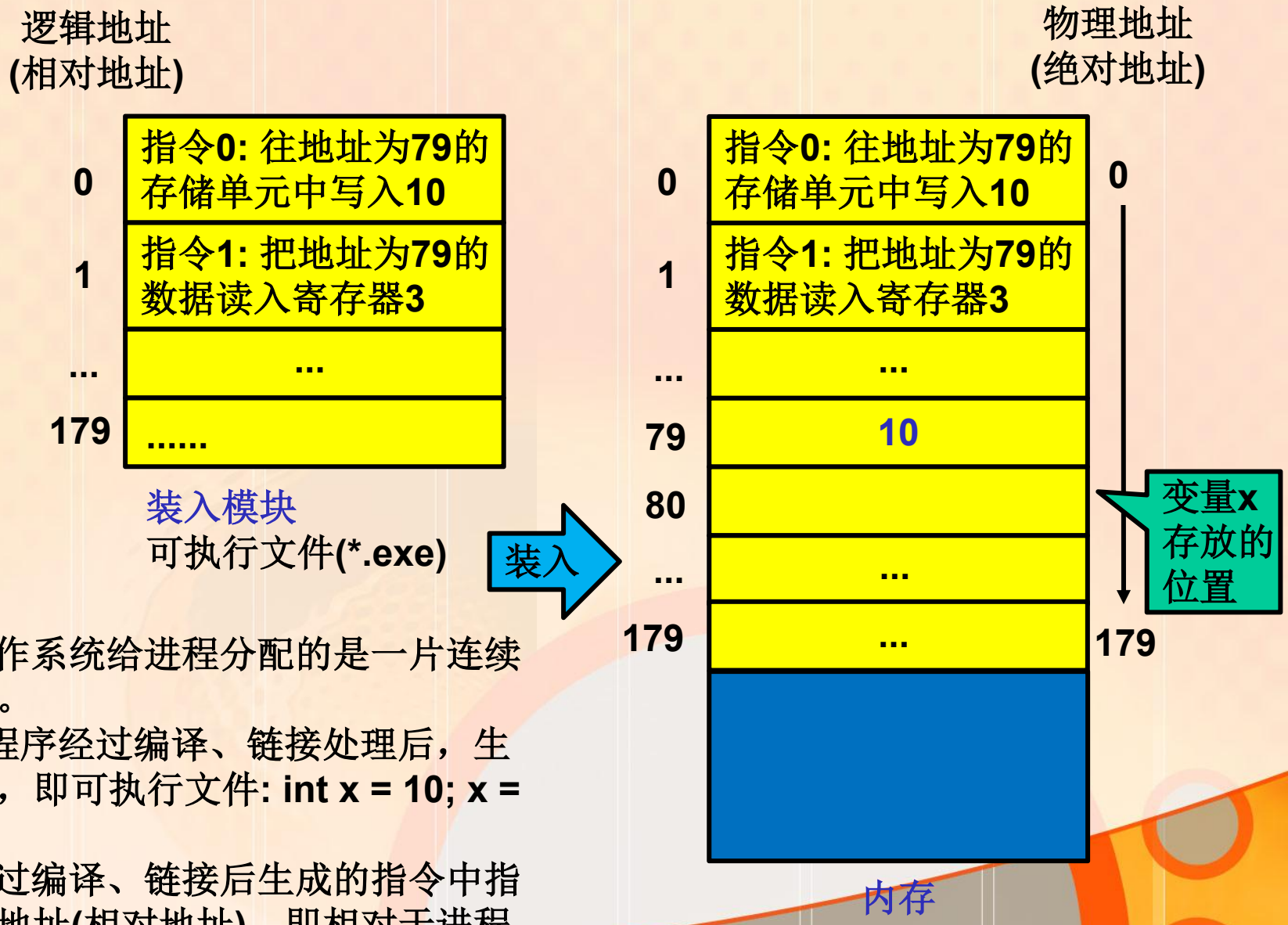
指令的工作原理

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，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？

指令的工作原理

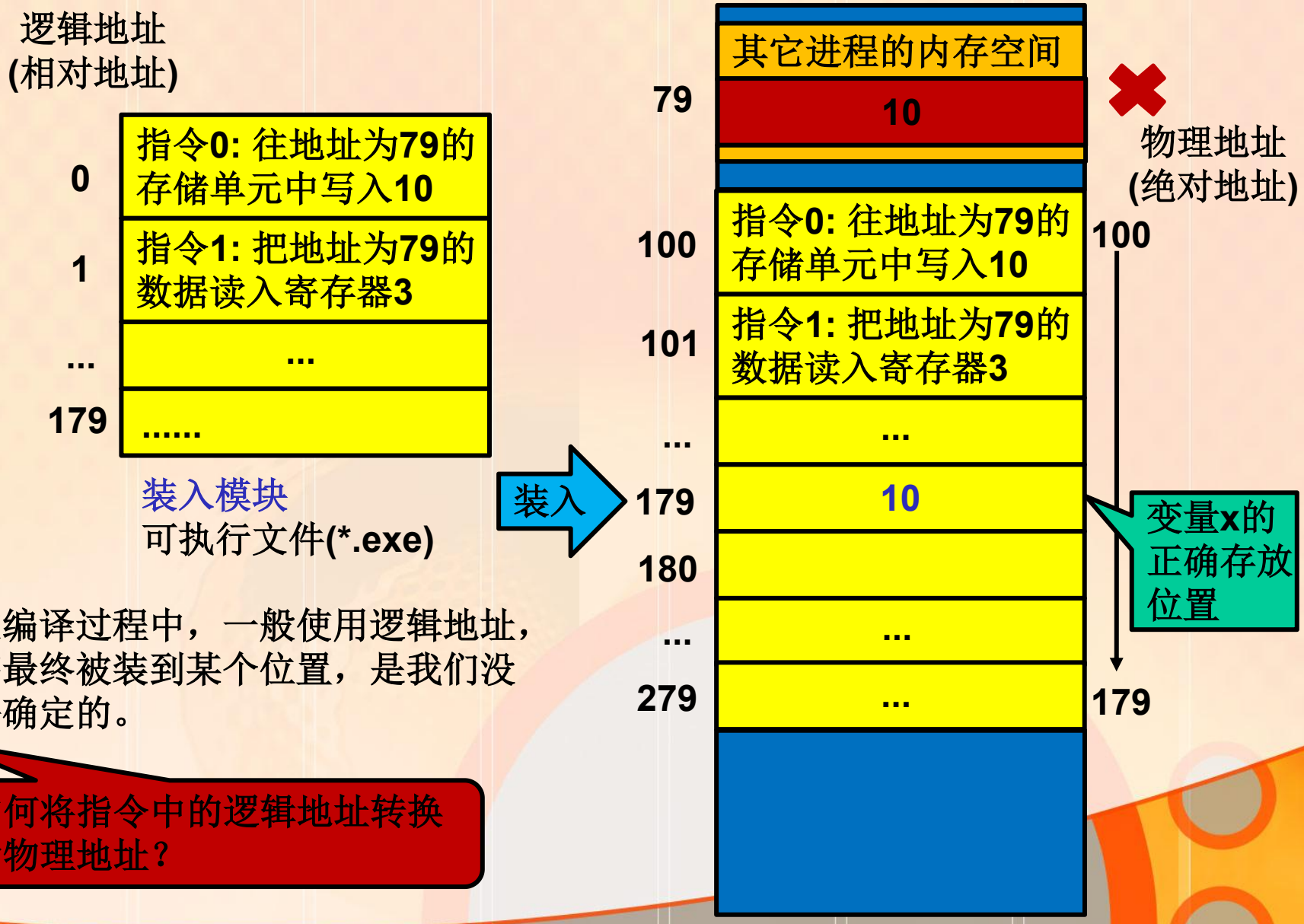


默认操作系统给进程分配的是一片连续的内存空间。

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。

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。

指令的工作原理



程序在编译过程中，一般使用逻辑地址，而程序最终被装到某个位置，是我们没有办法确定的。

如何将指令中的逻辑地址转换为物理地址？

指令的工作原理

为了解决地址转换，有三种装入方式：

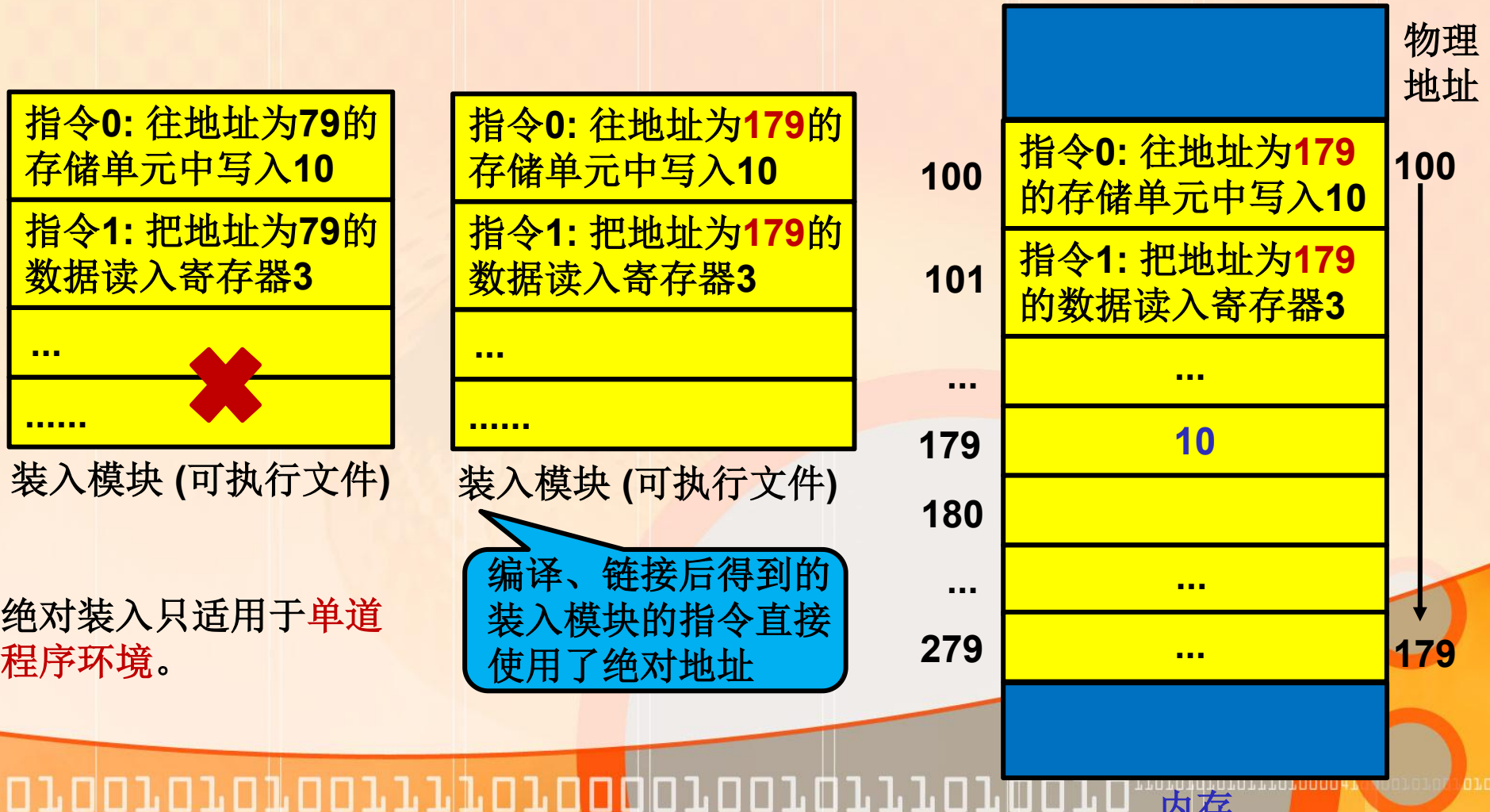
1. 绝对装入
2. 可重定位装入(静态重定位)
3. 动态运行时装入(动态重定位)

装入的三种方式——绝对装入

绝对装入: 在编译时，如果知道程序将放入到内存的哪个位置，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。

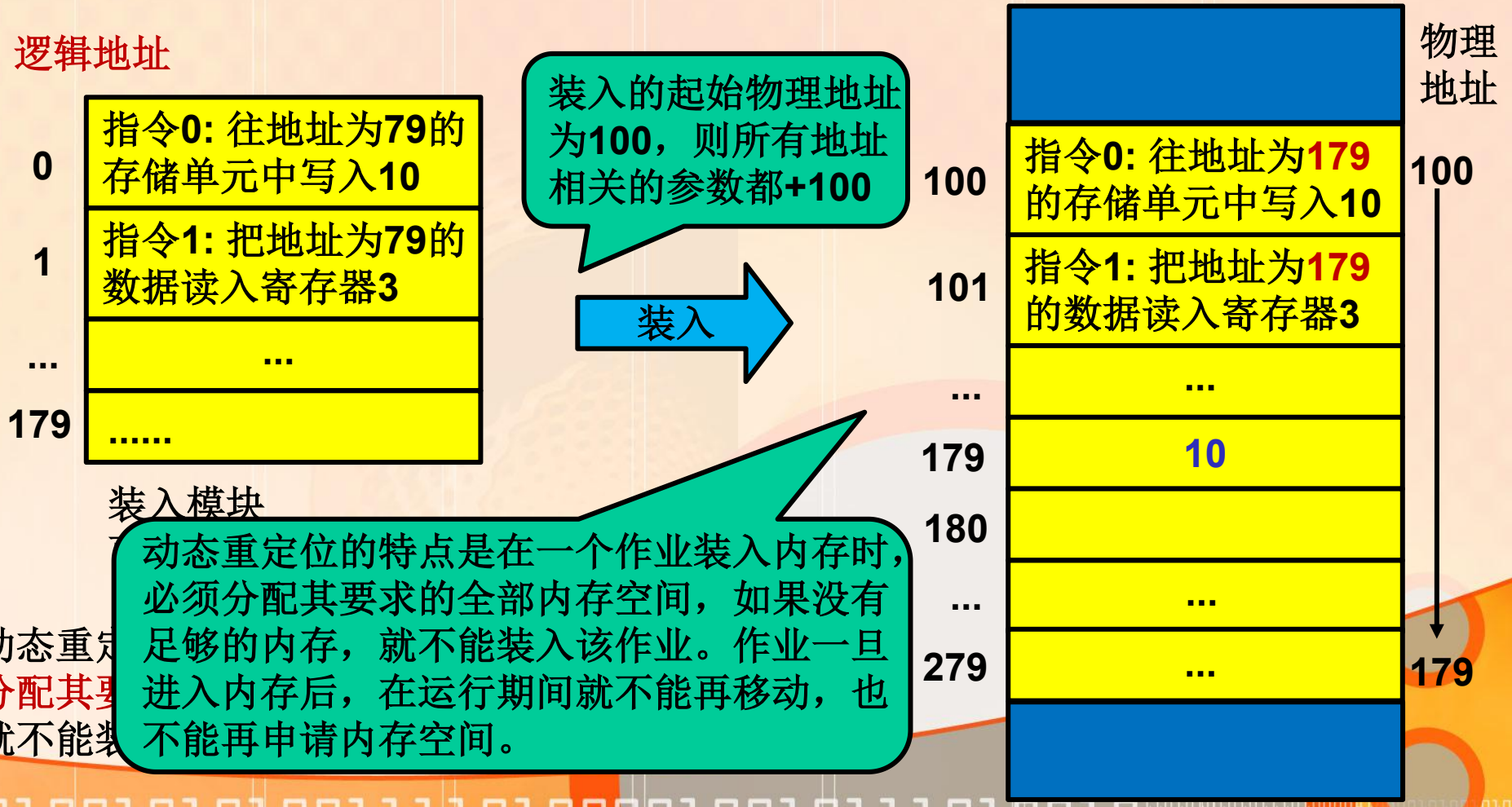
Eg: 如果知道装入模块要从地址为100的地方开始存放...



绝对装入只适用于单道程序环境。

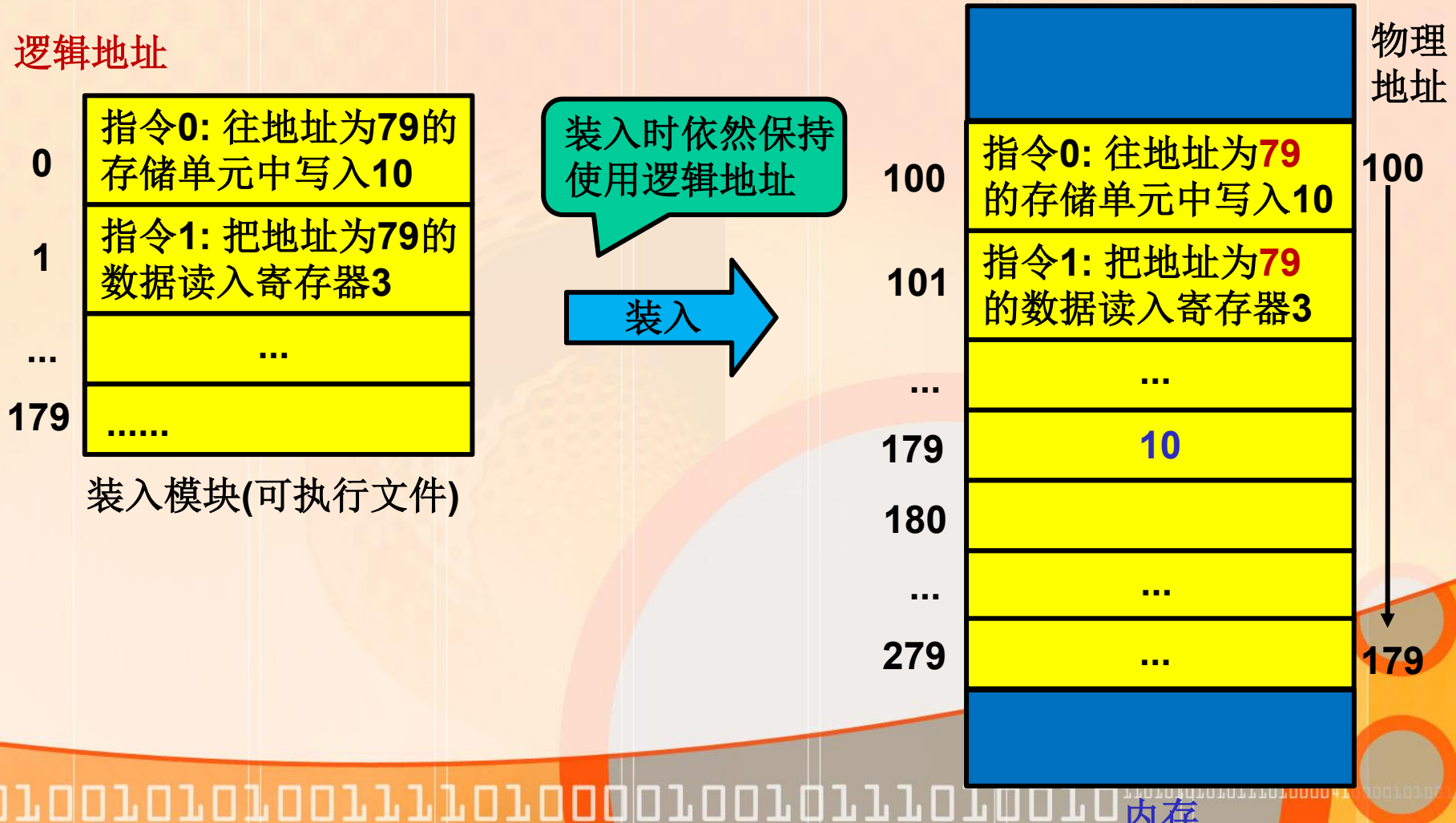
装入的三种方式——可重定位装入

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）。



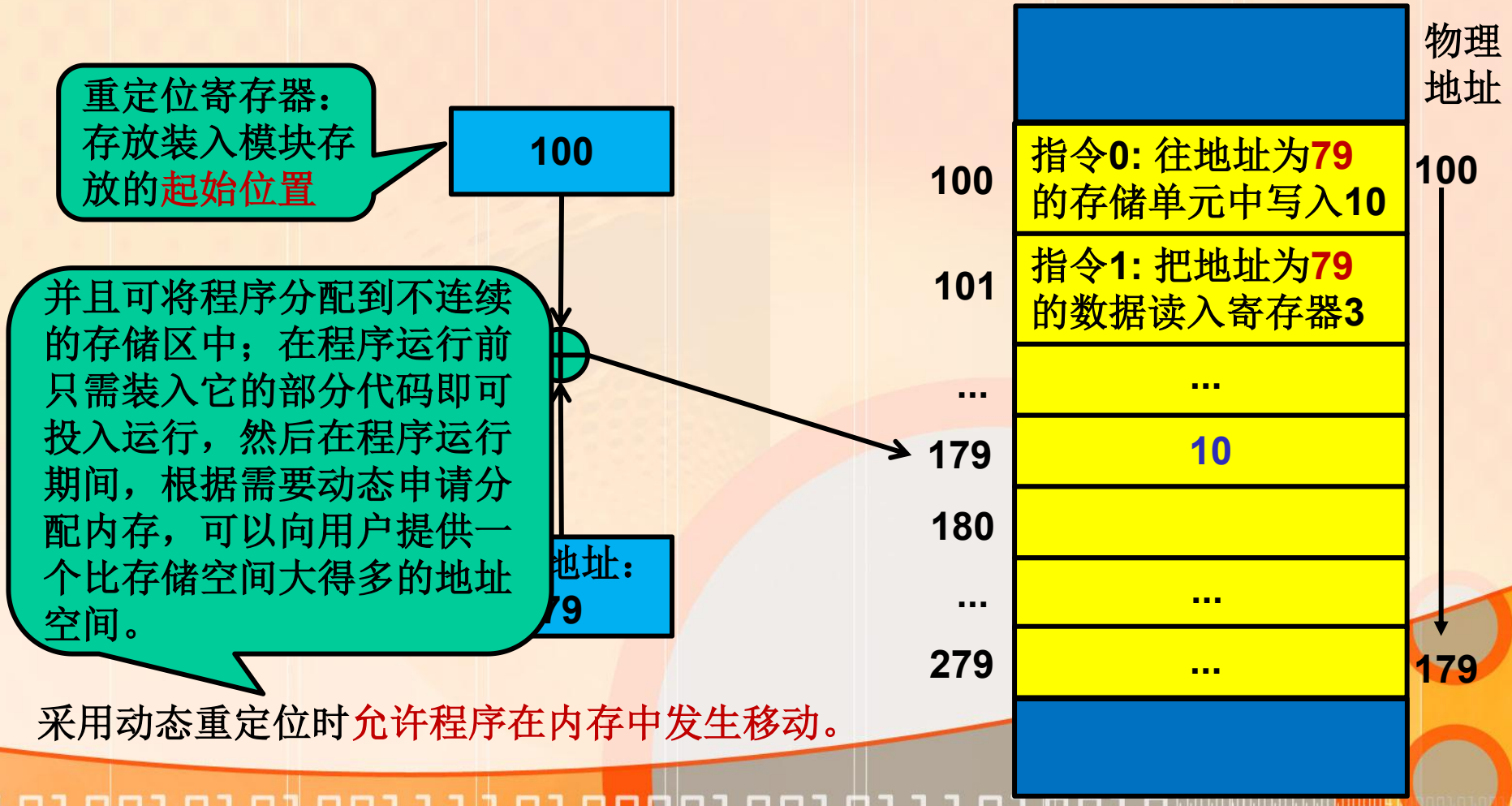
装入的三种方式——动态运行时装入

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。



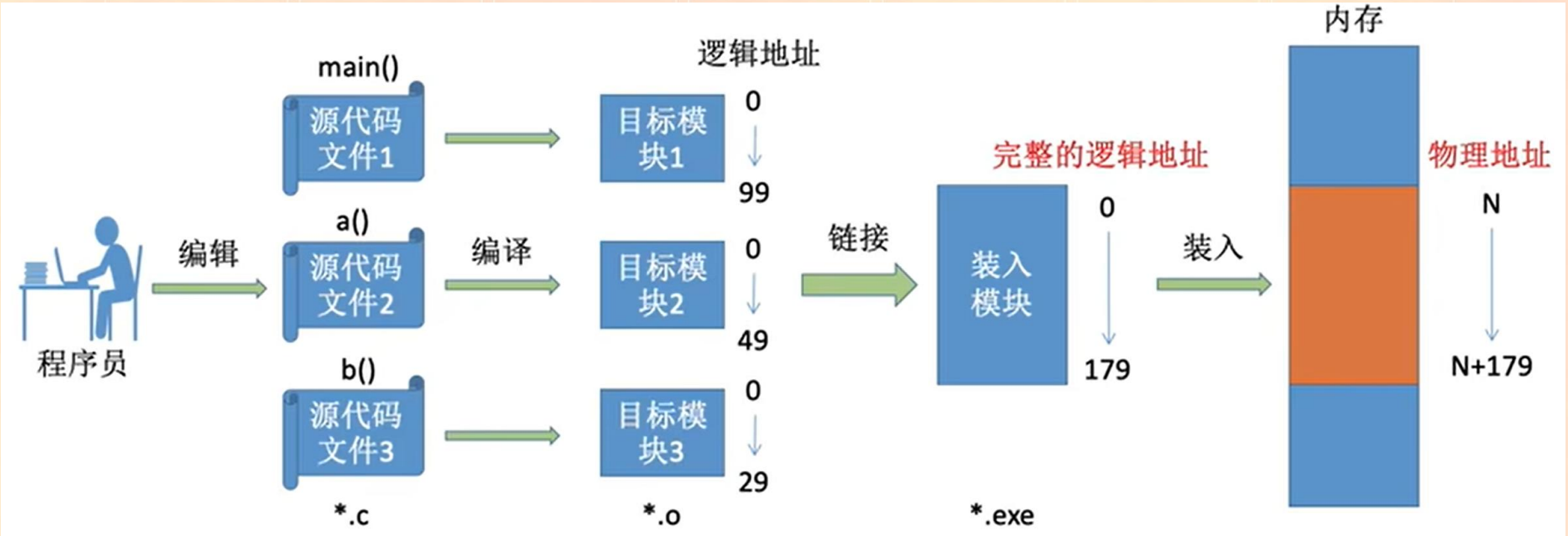
装入的三种方式——动态运行时装入

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。



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

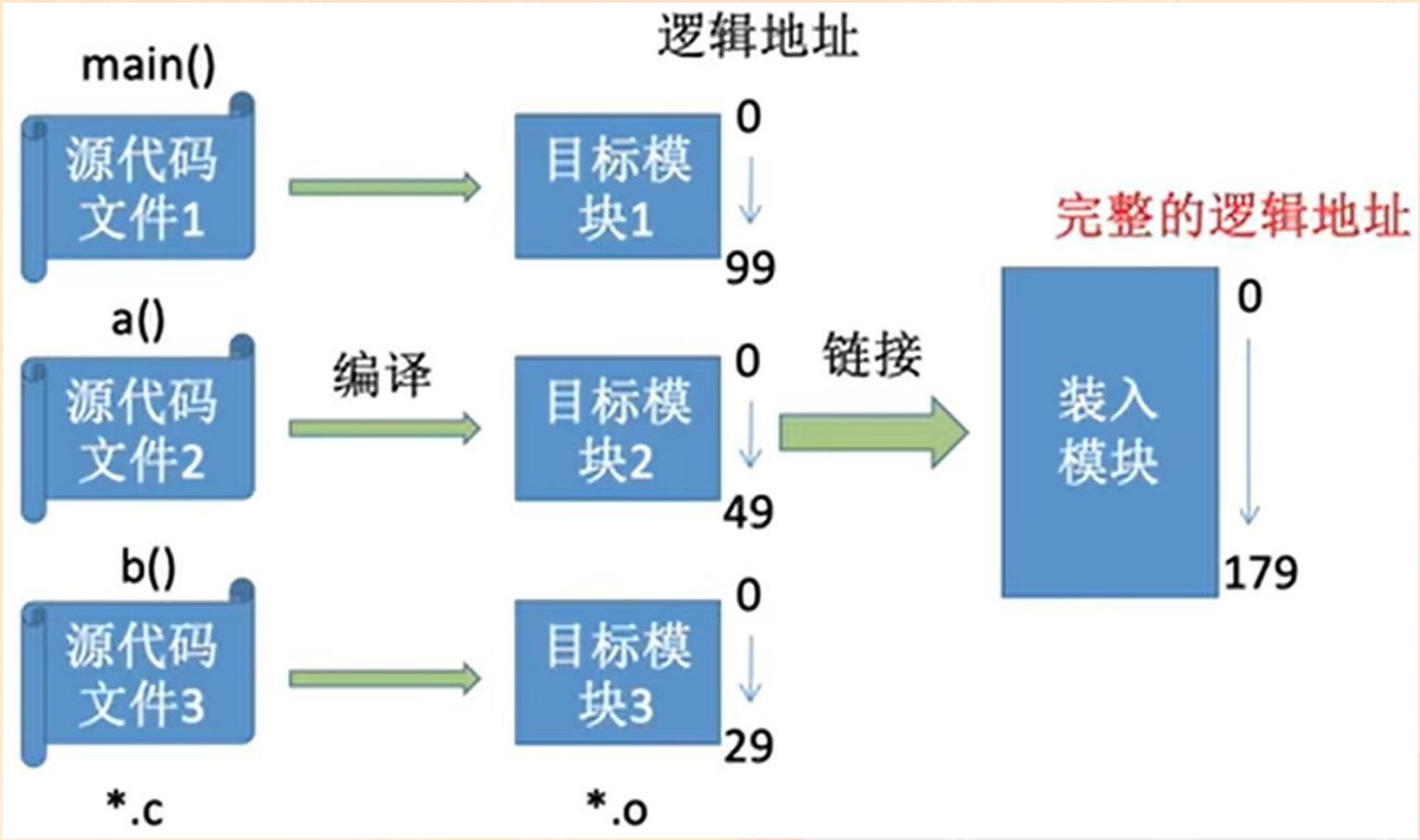


编译：由编译程序将用户源代码编译成若干个目标模块（编译就是把高级语言翻译为机器语言）

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。

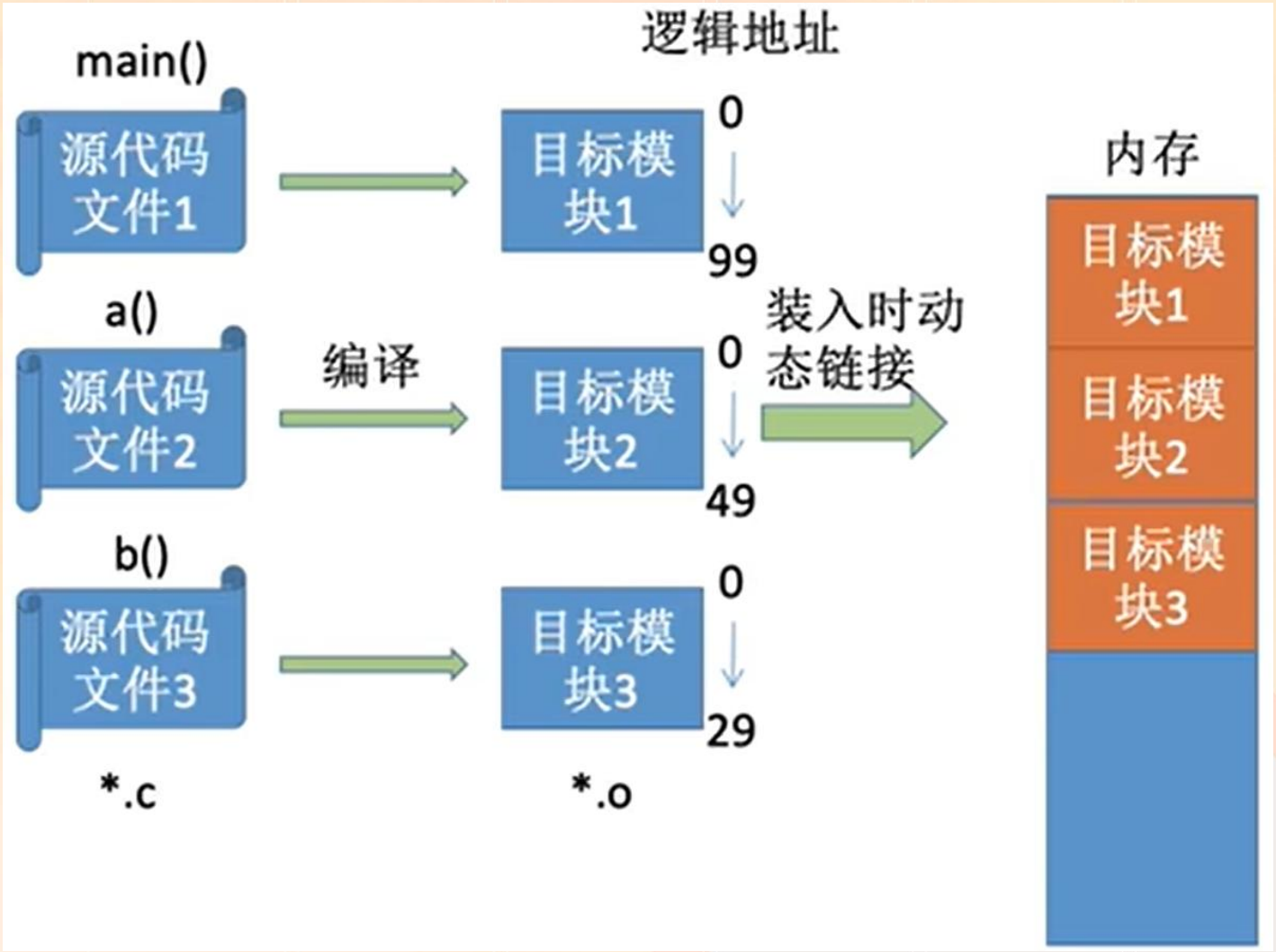
装入（装载）：由装入程序将装入模块装入内存运行

链接的三种方式



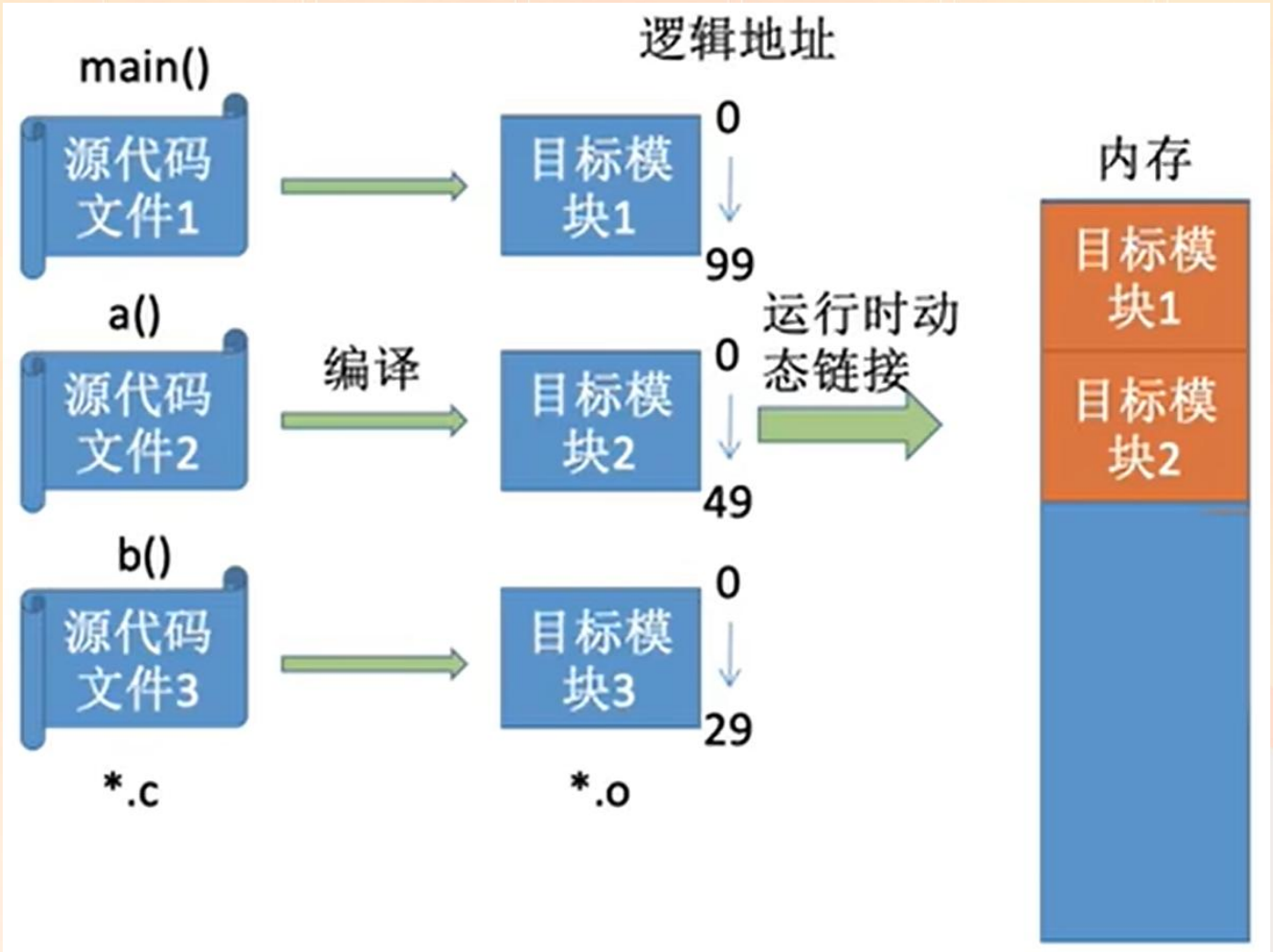
静态链接：在程序之前，先将各目标模块及它们所需的库函数连接成一个完整的可执行文件(装入模块)，之后不再拆开。

链接的三种方式



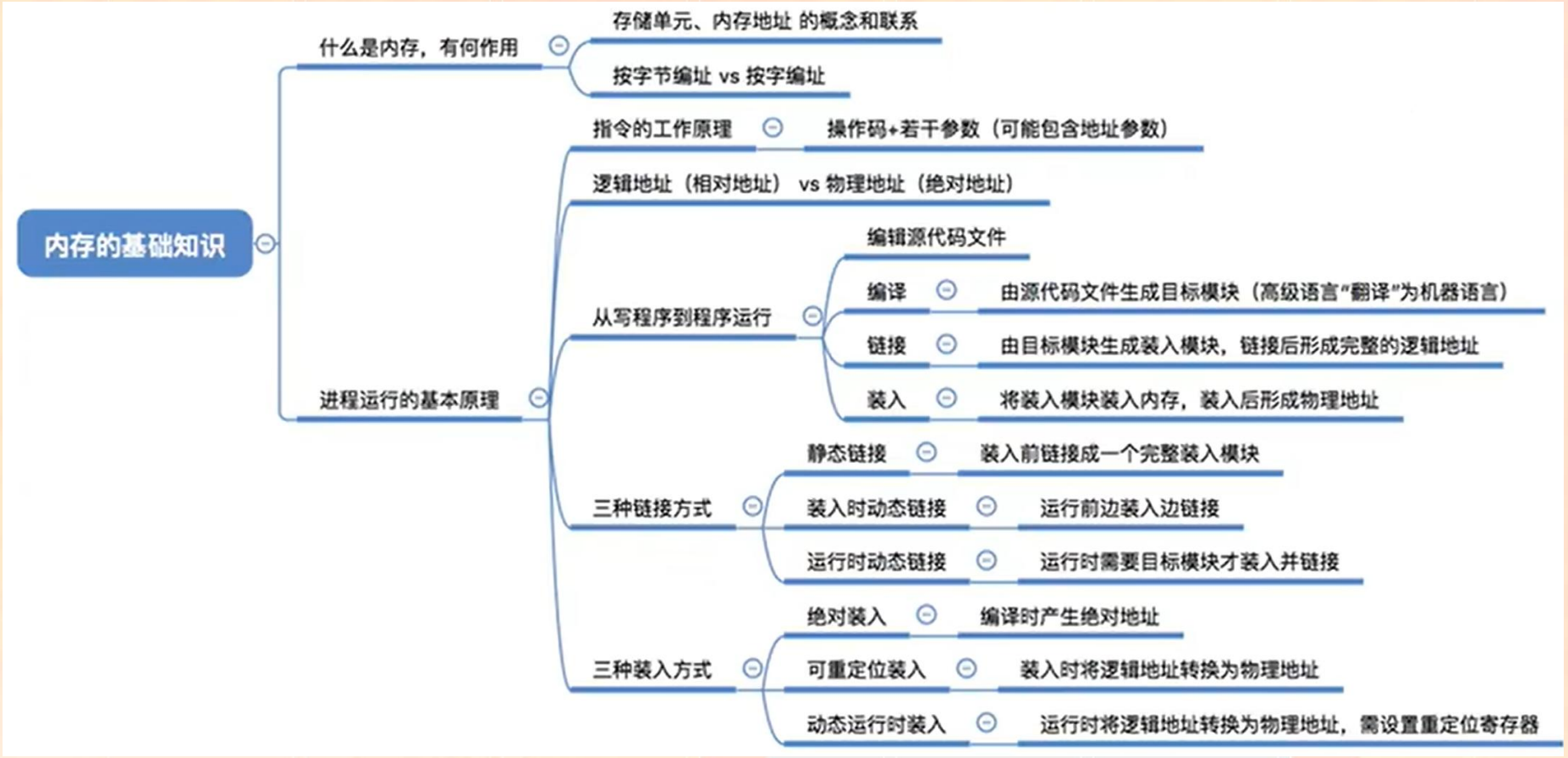
2. 装入时动态链接：将各目标模块装入内存时，边装入边链接的链接方式。

链接的三种方式



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。

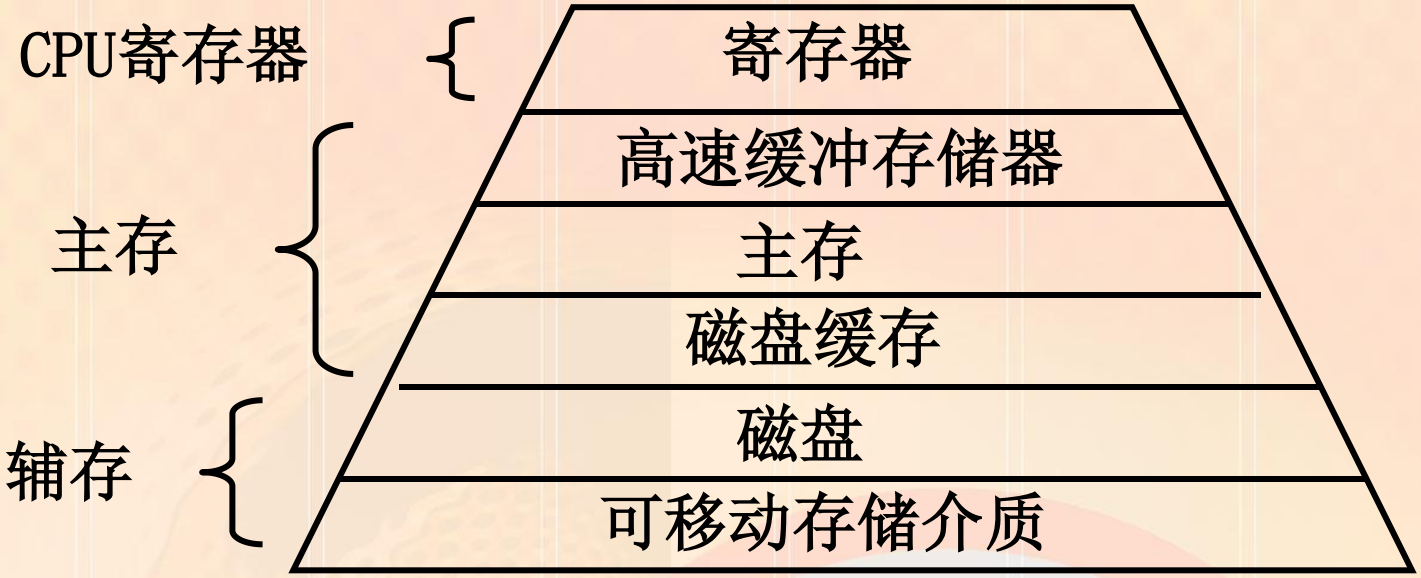
总结


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

4.1 存储器的层次结构

存储器是计算机系统的重要组成部分之一。对存储器加以有效管理，不仅直接影响存储器的利用率，而且对系统性能有重大影响。存储器管理的主要对象是内存，对外存的管理在文件管理中。

存储器的层次结构



4.1 存储器的层次结构

1、主存储器

CPU只能从主存储器中取得指令和数据。但运行速度远低于CPU执行指令的速度。

2、寄存器

访问速度最快，但价格昂贵。

3、高速缓存

容量远大于寄存器，但比主存小两三个数量级。

4、磁盘缓存

利用主存中的存储空间。

4.2 程序的装入和链接

- 编程得到可执行文件的步骤：编译、链接、装入。

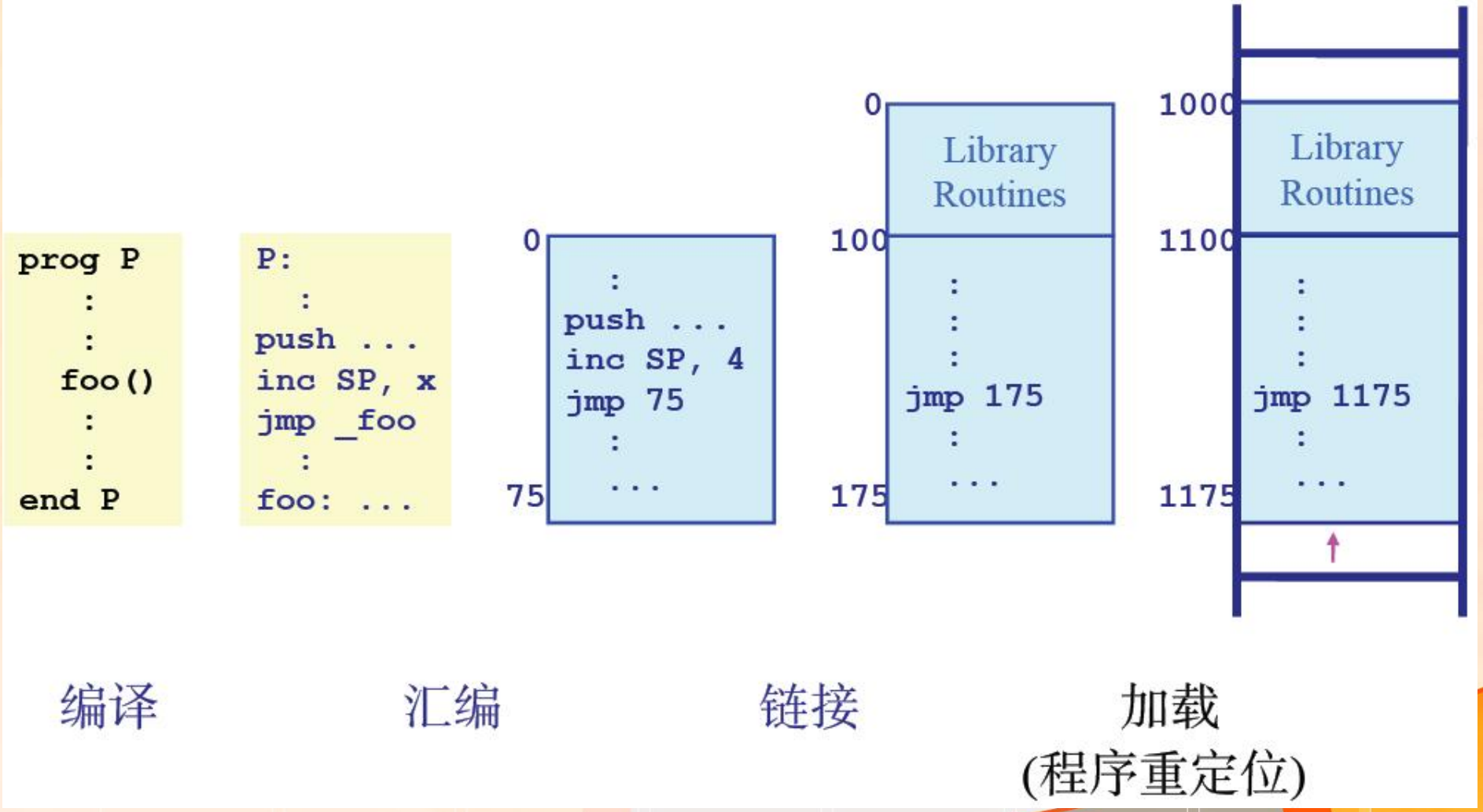
  编辑：得到如test.cpp ，a.asm等源文件

编译：从每个源文件得到对应的目标文件（PC机系统后缀为OBJ的文件）

链接：将若干有关目标文件（在VC++环境中为一个workspace中的文件）及有关系统库目标文件进行链接，得到相应的可执行文件（PC机系统后缀为EXE的文件或动态连接文件DLL），即装入模块。

装入：装入模块再由OS装入内存，成为进程。

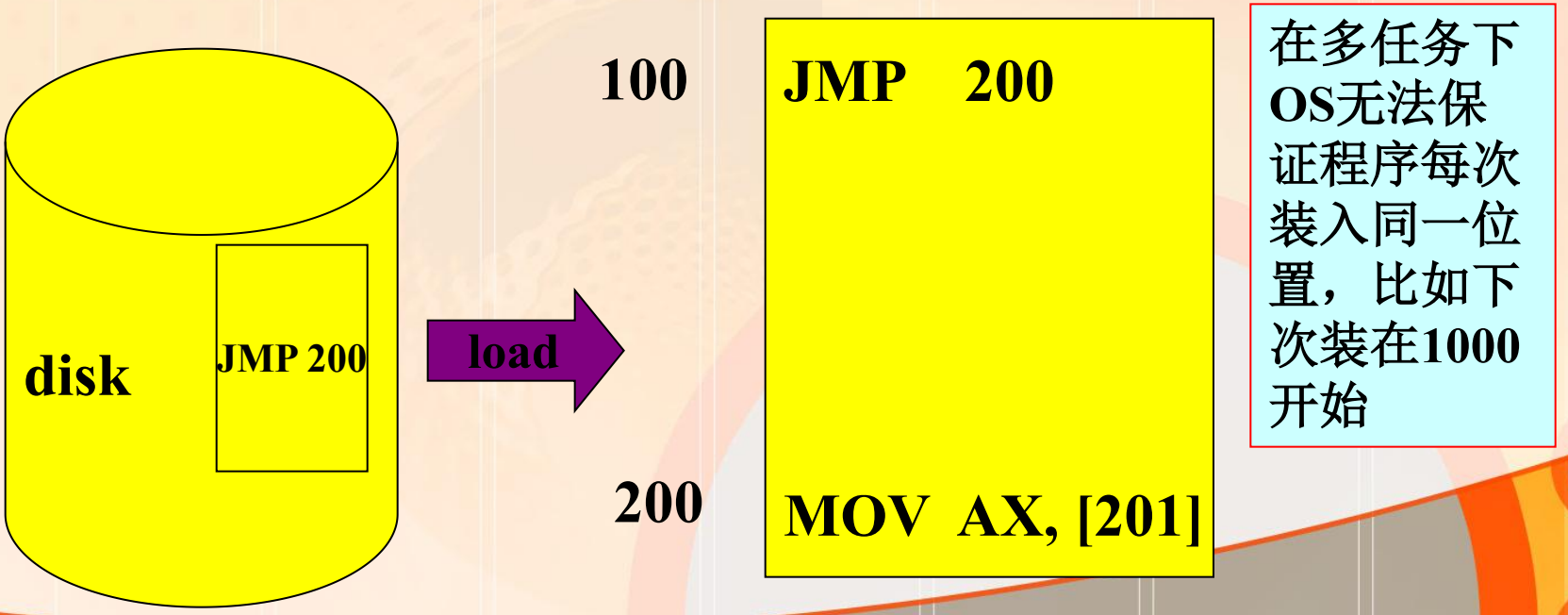
4.2 程序的装入和链接


prog P
:
:
foo()
:
:
end P
P:
:
push ...
inc SP, x
jmp _foo
:
foo: ...
0
:
push ...
inc SP, 4
jmp 75
:
75 ...
0
Library Routines
100
:
:
:
jmp 175
:
175 ...
1000
Library Routines
1100
:
:
:
jmp 1175
:
1175 ...
编译
汇编
链接
加载
(程序重定位)

4.2.1　程序的装入

1. 绝对装入(absolute loading)

- 编译程序知道程序将驻留在内存的地址，产生绝对地址的目标代码；
- 绝对装入模块装入时直接定位在上述内存地址，不需修改程序和数据的地址。
- 优点：装入过程简单。
- 缺点：过于依赖于硬件结构，不适于多道程序系统。

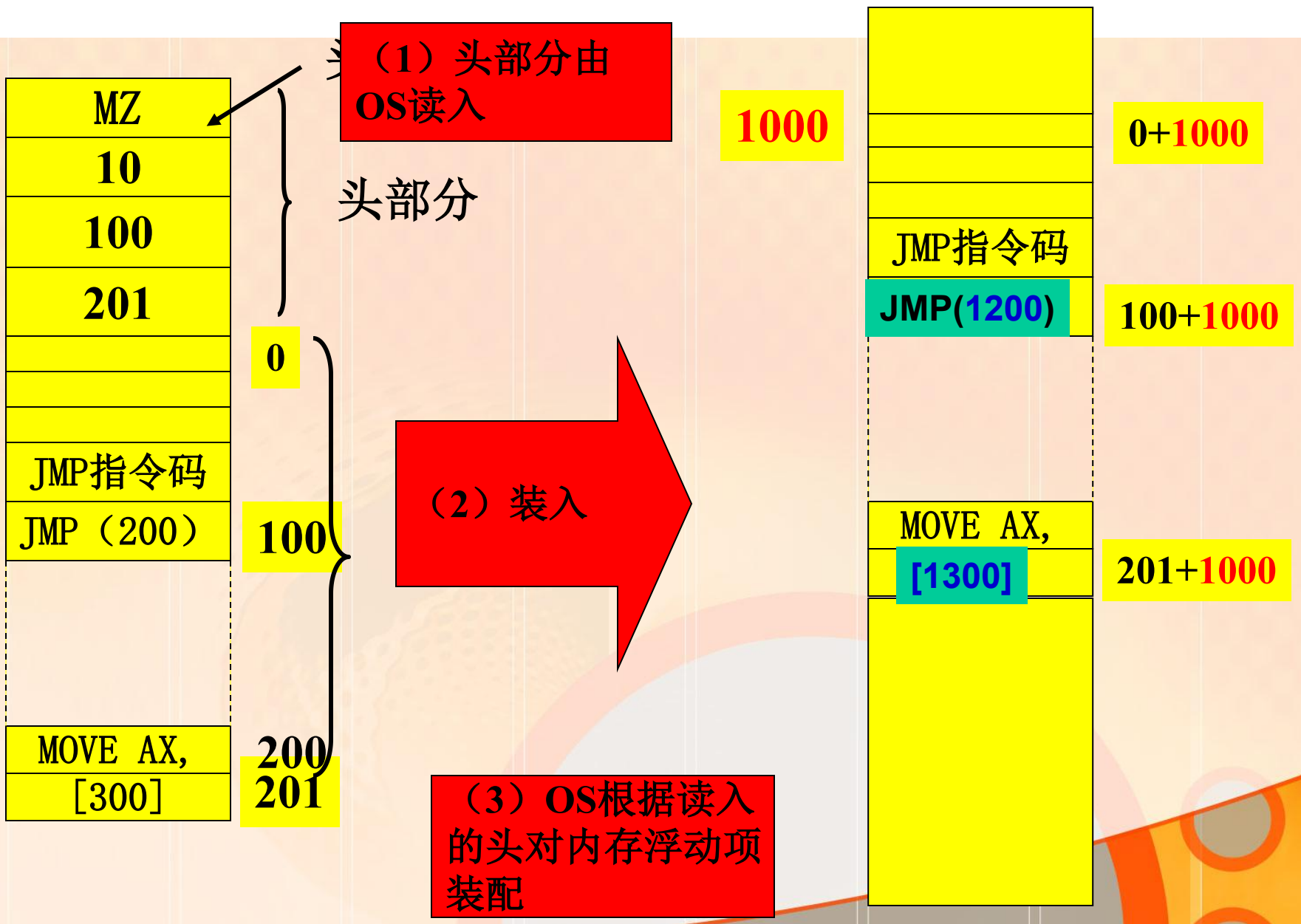


在多任务下OS无法保证程序每次装入同一位置，比如下次装在1000开始

2. 可重定位装入(relocatable loading)

- 在多道程序环境下，目标模块的起始地址通常从0开始，程序的其它地址也都是相对于起始地址计算的。装入时采用可重定位装入。
- 在可执行文件中，列出各个需要重定位的地址单元和相对地址值（可重定位表），装入时再根据所定位的内存地址去修改每个重定位地址项，添加相应偏移量。

例：


（1）头部分由OS读入
头部分
MZ
10
100
201
JMP指令码
JMP（200）
MOVE AX,
[300]
0
100
200
201
（2）装入
（3）OS根据读入的头对内存浮动项装配
1000
0+1000
JMP指令码
JMP(1200)
100+1000
MOVE AX,
[1300]
201+1000

4.2.1　程序的装入

- 优点：
  - 不需硬件支持，可以装入有限多道程序。
- 缺点：
  - 一个程序通常需要占用连续的内存空间，程序装入内存后不能移动。不易实现共享。
- 地址变换是由装入程序在装入目标模块时一次完成，进程装入内存后不能移动，故称为静态重定位。

4.2.1 程序的装入

- 3. 动态运行时装入(dynamic run-time loading)
  - 进程开始执行时，未全部装入内存，而是部分装入，运行时，需要哪个模块再装入哪个模块。
  - 程序装入内存后，并不立即将相对地址转换为绝对地址，地址转换推迟到程序真正要执行时才进行，即动态重定位。
  - 装入内存中进程的所有地址都是相对的。
  - 优点：
    - OS可以将一个程序分散存放于不连续的内存空间，可以移动程序，有利于实现共享。
    - 能够支持程序执行中产生的地址引用，如指针变量（而不仅是生成可执行文件时的地址引用）
  - 缺点：需要硬件支持（通常是CPU），OS实现较复杂——是虚拟存储的基础

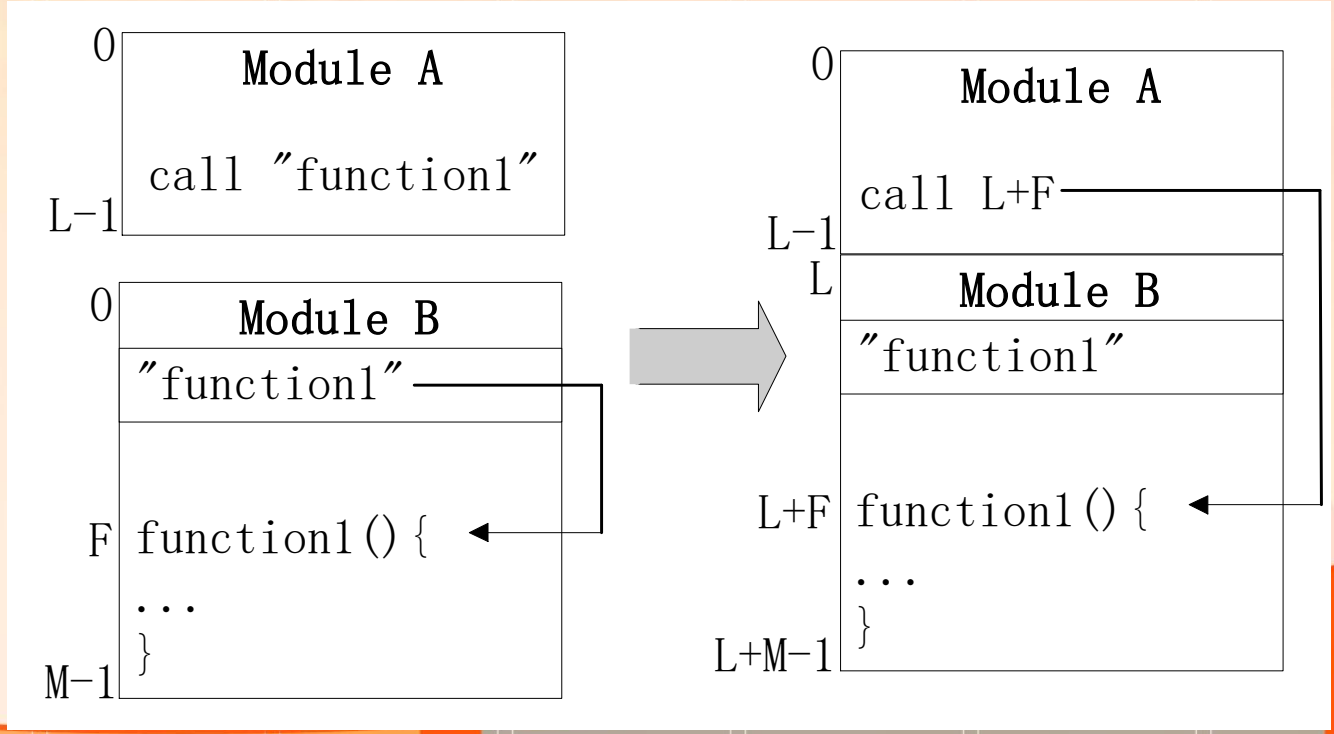
4.2.2 程序的链接

- 源程序经过编译后，得到一组目标模块，再利用链接程序将这组目标模块链接，形成装入模块，根据链接时间的不同，链接分为：
  - 静态链接
  - 装入时动态链接
  - 运行时动态链接

4.2.2 程序的链接

1. 静态链接(static-linking)

- 在程序运行前，先将各目标模块及它们所需的库函数，链接成一个完整的装入模块，以后不再拆开。要解决两个问题：
  - 修改相对地址
  - 变换外部调用符号
- 对多用户、多任务系统显然有冗余，比如多个用户调用了sin(x)，则每个目标代码中都有这部分代码，装入到内存则也都有这部分代码。



2. 装入时动态链接(dynamic-linking)

- 源程序编译得到的目标模块是在装入内存时，边装入边链接的，即在装入一个目标模块时，若发现一个外部模块调用事件，装入程序去找出相应的外部目标模块，并将它装入内存，同时修改相对地址。
- 优点
  - 共享：多个进程可以共用一个目标模块，节省内存，减少文件交换。
  - 便于修改和更新。各目标模块是分开存放的，便于修改。

4.2.2 程序的链接

- 3. 运行时动态链接（Run-time Dynamic Linking）
  - 应用程序运行时，每次运行的模块可能不同。但事先又无法知道，在前两种链接方式中，只能将所有模块都装入内存，并在装入时都链接在一起。显然低效。
  - 运行时动态链接是将某些模块的链接推迟到执行时。即，执行时发现调用的模块未被装入，由OS找到该模块并装入，并将其链接到调用者模块上。
  - 优点：
    - 部分装入：一个进程可以将多种操作分散在不同的DLL中实现，而只将与当前操作相对应的DLL装入内存。
    - 便于局部代码修改：即便于代码升级和代码重用；只要函数的接口参数（输入和输出）不变，则修改函数及其DLL，无需对可执行文件重新编译或链接。
    - 便于适应运行环境：调用不同的DLL，就可以适应多种使用环境和提供不同功能。如：不同的显示卡只需厂商为其提供特定的DLL，而OS和应用程序不必修改。
  - 缺点：
    - 链接开销：增加了程序执行时的链接开销；
    - 管理开销：程序由多个文件组成，增加管理复杂度。

地址生成

CPU

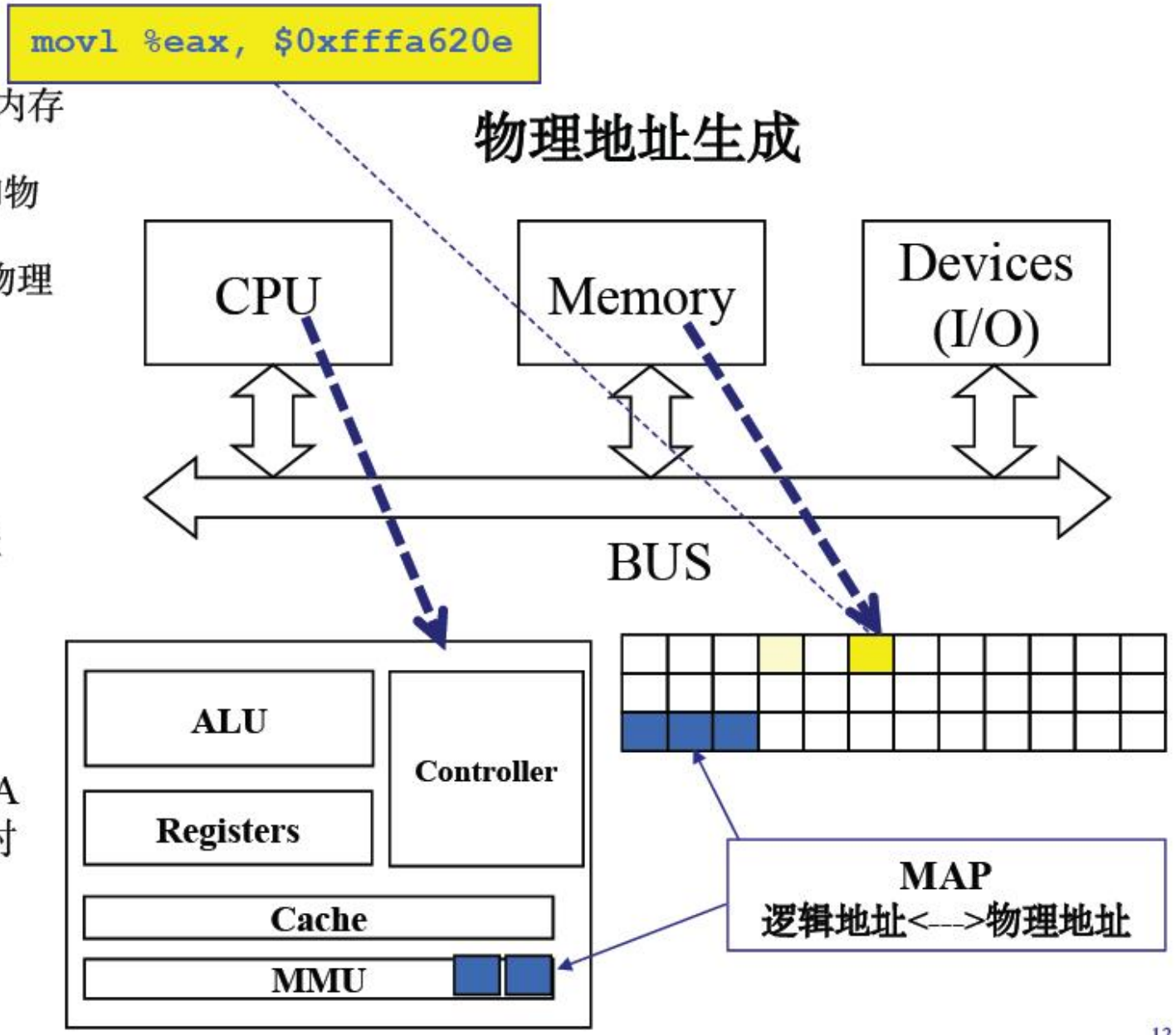
1. ALU 需要逻辑地址的内存内容
2. MMU 进行逻辑地址和物理地址的转换
3.Controller 给总线发送物理地址请求

Memory

4.Memory发送物理地址的内容给CPU

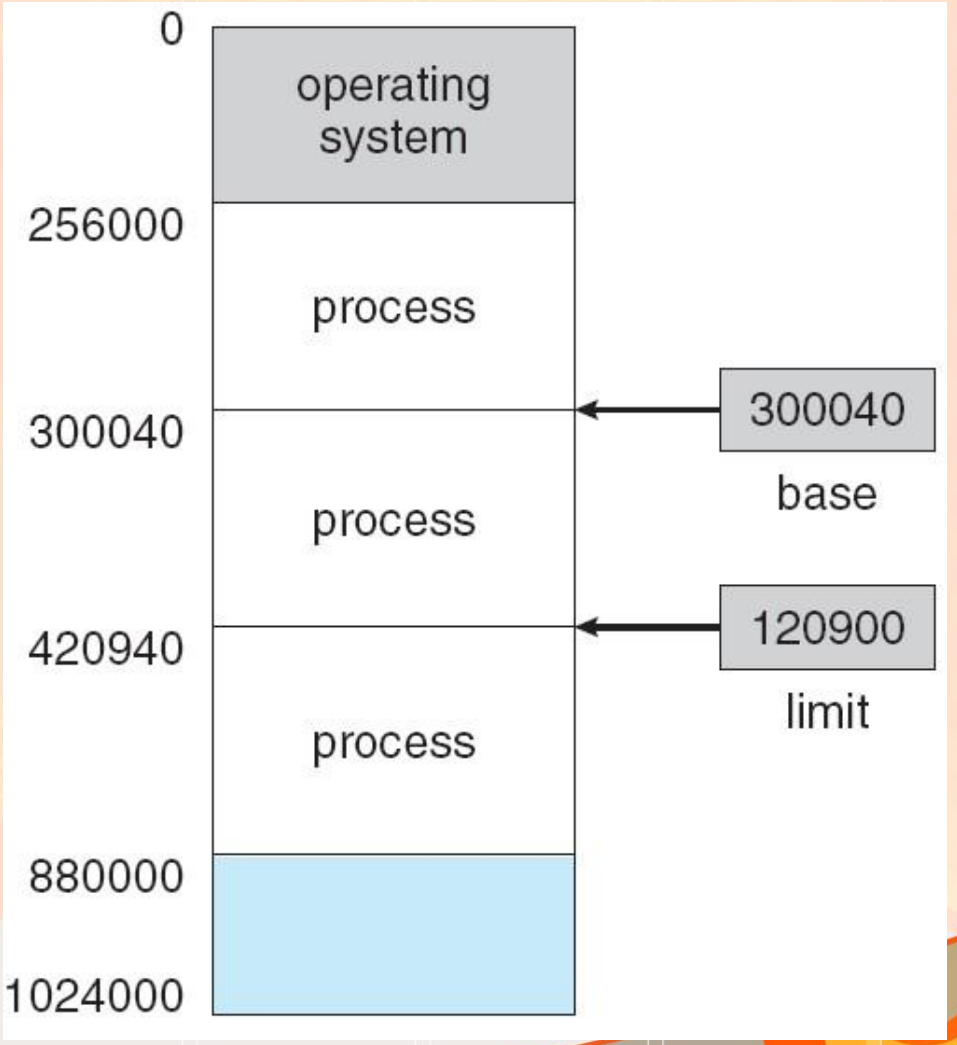
操作系统

负责建立逻辑地址LA和物理地址PA的映射



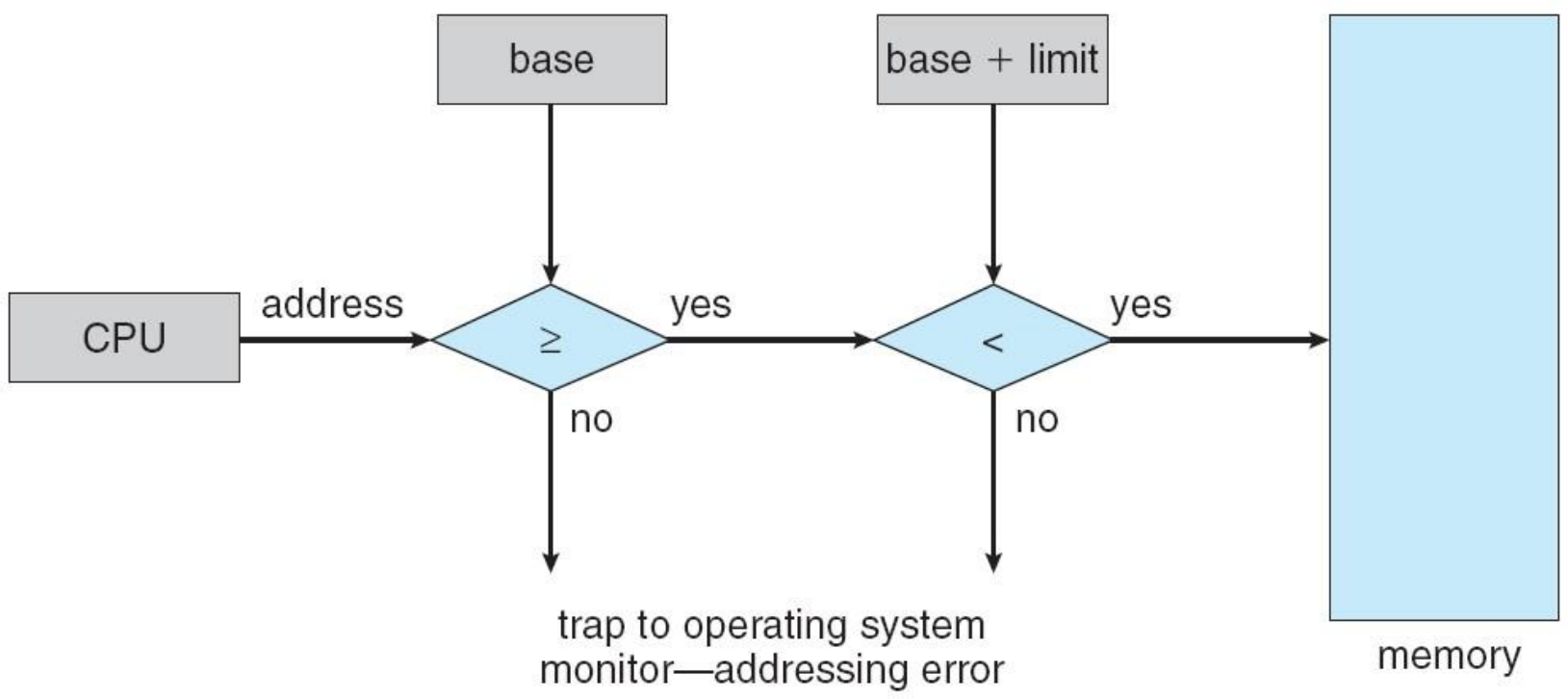
地址安全检查

- 逻辑地址空间由base基址寄存器和limit界址寄存器定义
- 仅OS能设置base和limit
  - 特权指令



地址安全检查

- 在用户模式中验证所产生的地址
- 如果发现不好的地址，中断进入内核



- 例：在虚拟内存管理中，地址变换机构将逻辑地址变换为物理地址，形成该逻辑地址的阶段是
- A 编辑　　B 编译　C 链接　　D装载

- 答 C