操作系统

L13 主存管理（上）

胡燕
大连理工大学 软件学院

过渡到内存管理模块

进程管理模块

主要进行任务执行过程的控制

内存管理模块

任务执行期间相关数据的存储

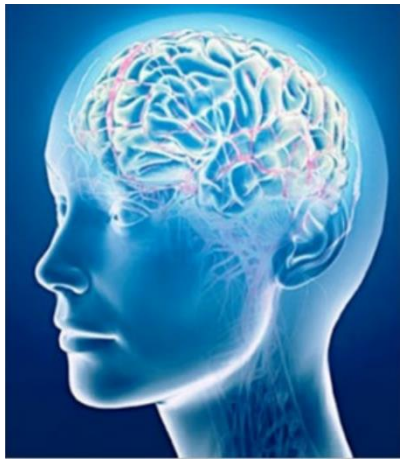
主存管理概念

Physical Memory Management

01

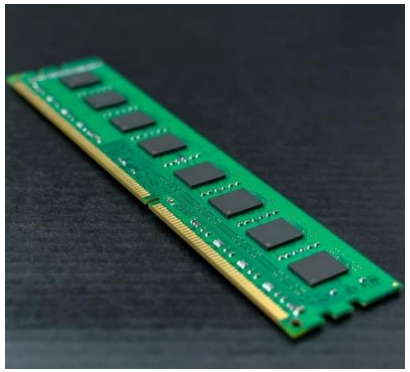
1-主存管理基本概念 What is main memory

主存 v.s. 人类记忆

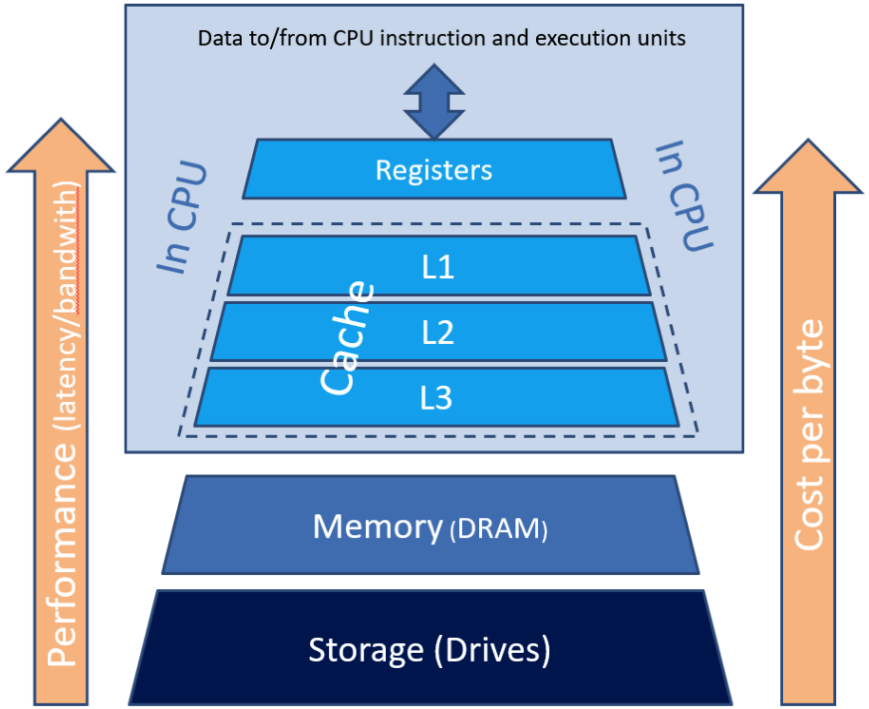


人类大脑：有一个memory

大脑具有短期和长期记忆



Computer Memory

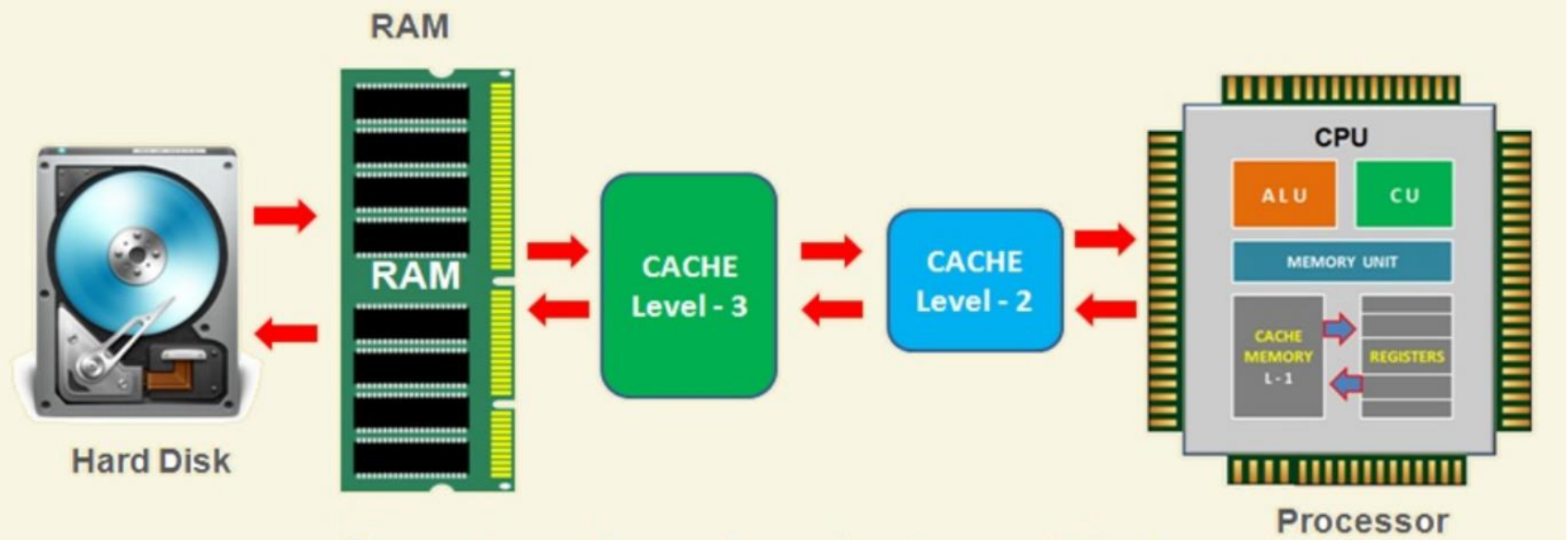


1-主存管理基本概念

What is main memory

主存 v.s. 人类记忆

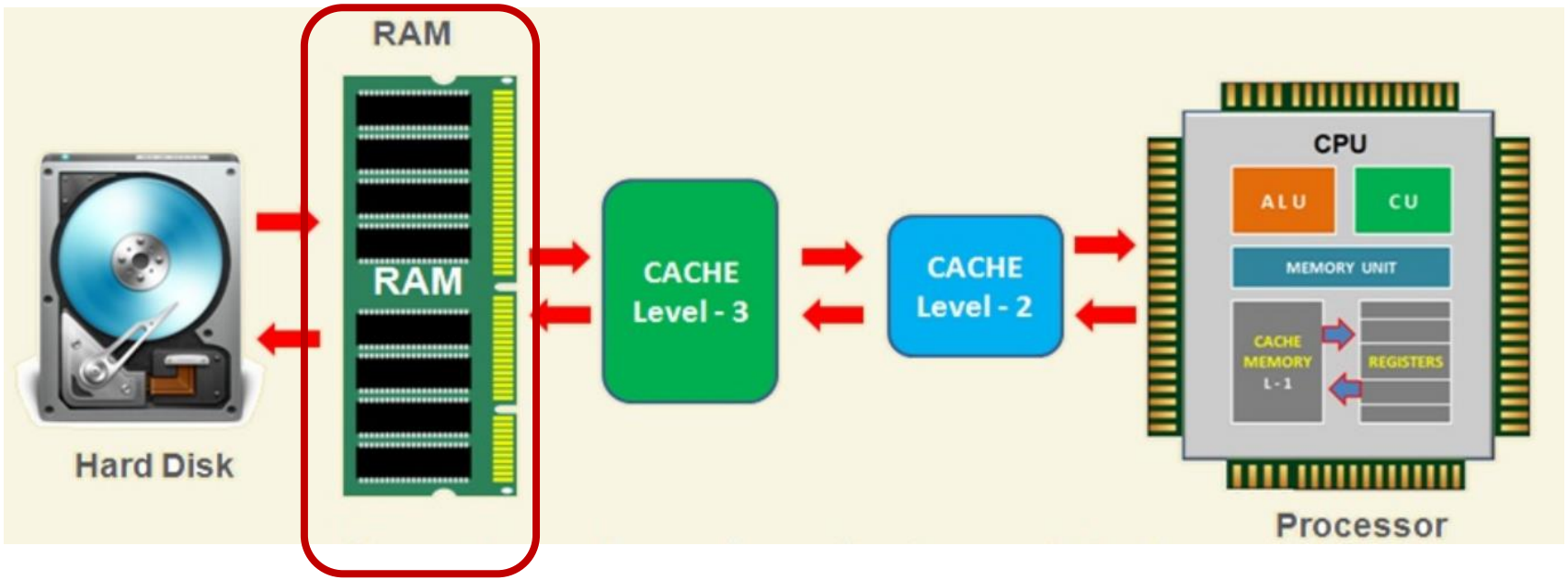
Computer Hierarchy



Brain memory counterparts	Neocortex、Amygdala	Hippocampus (main memory)	Prefrontal cortex (short-term working memory)

1-主存管理基本概念 What is main memory

主存（Main Memory）



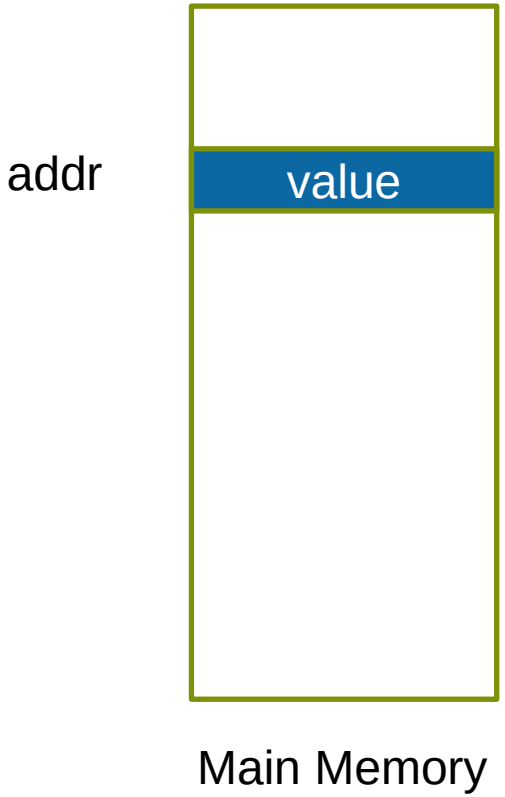
主存管理的对象：
RAM (Main Memory)

主存管理核心问题

主存管理中的主要问题

```
#include <stdio.h>
main(){
  int i = 55;
  printf("i=%d\n", i);
}
```

Where to put variable i?

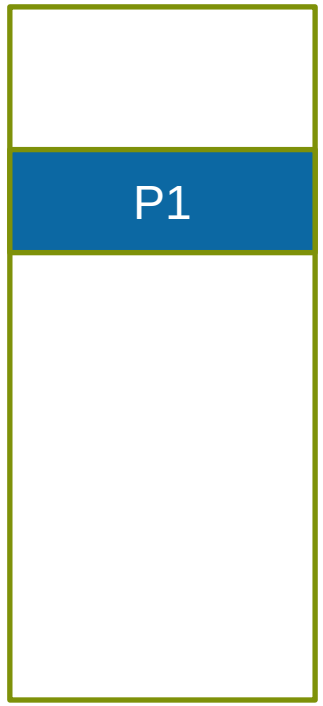


Q1: 地址绑定
(Address binding)

主存管理核心问题

主存管理中的主要问题

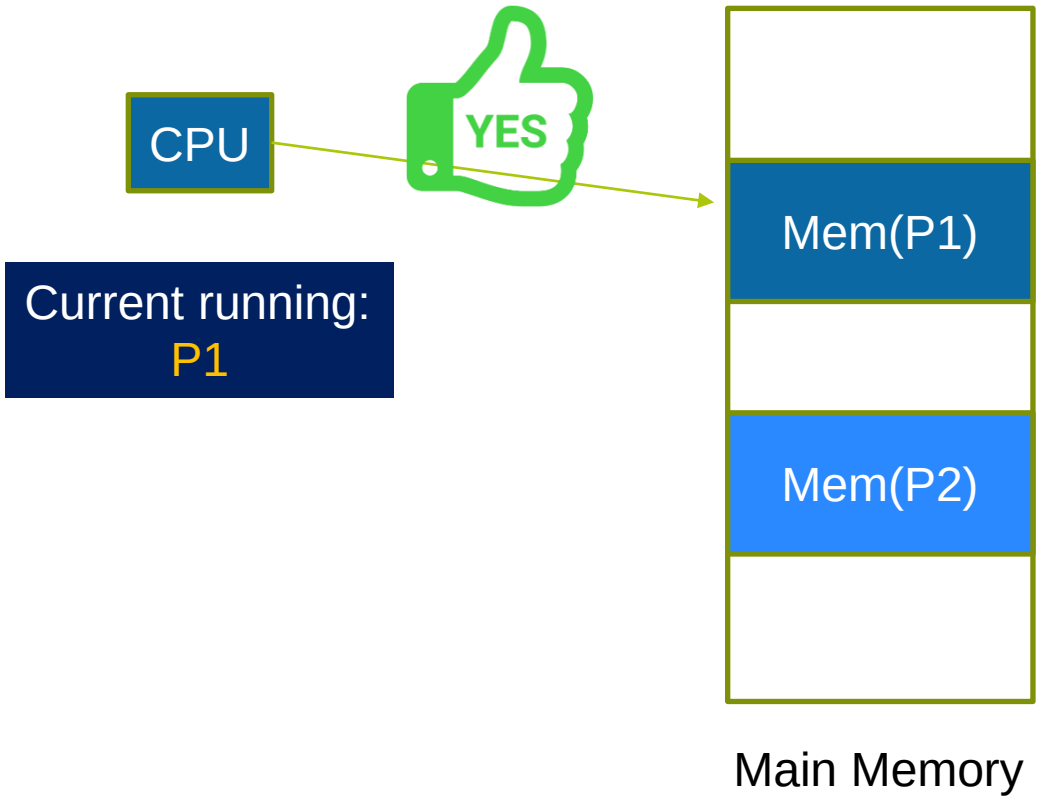
P1申请1M内存



Q2: 内存分配
(Memory Allocation)

主存管理核心问题

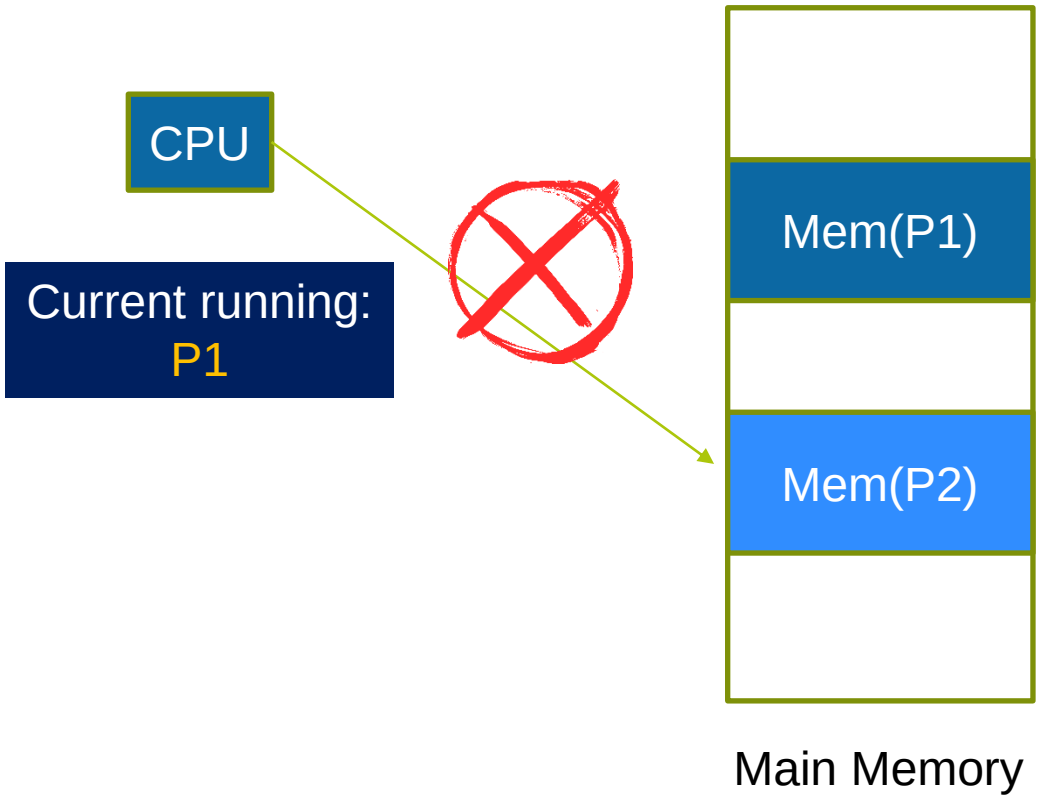
主存管理中的主要问题



进程可以访问自己地址空间内的内存地址

主存管理核心问题

主存管理中的主要问题

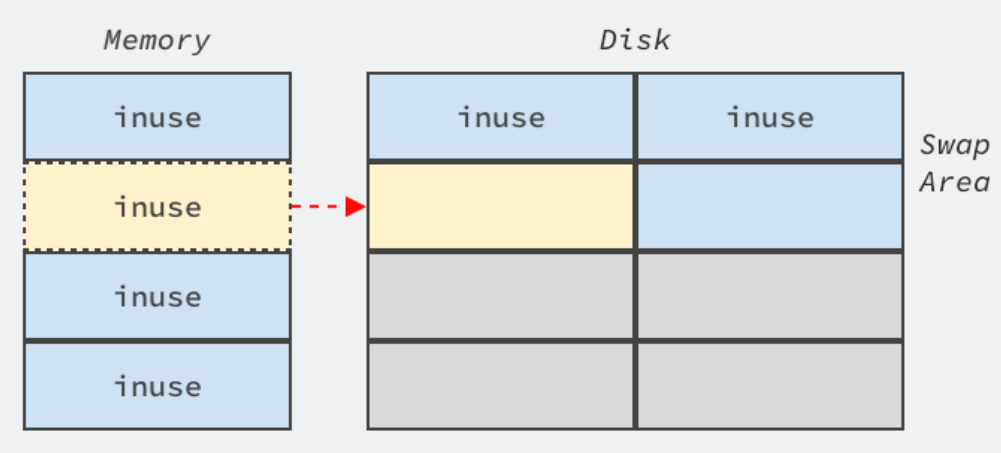


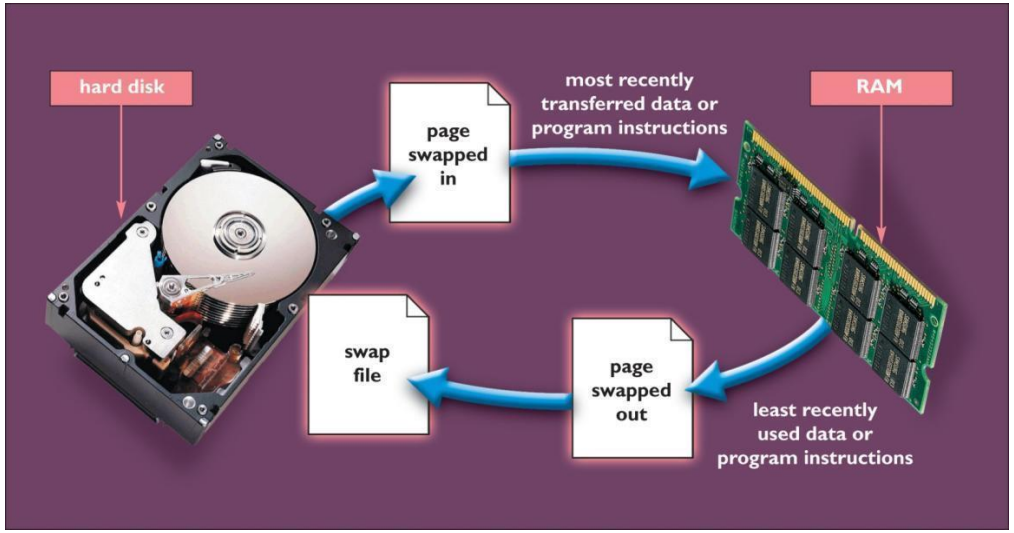
进程不可访问其他进程地址空间内的内存地址

Q3: 内存保护
(Memory Protection)

主存管理核心问题

主存管理中的主要问题





Q4: 内存扩充

地址绑定

主存管理中主要问题的处理

Q1：地址绑定
（Address Binding）

?

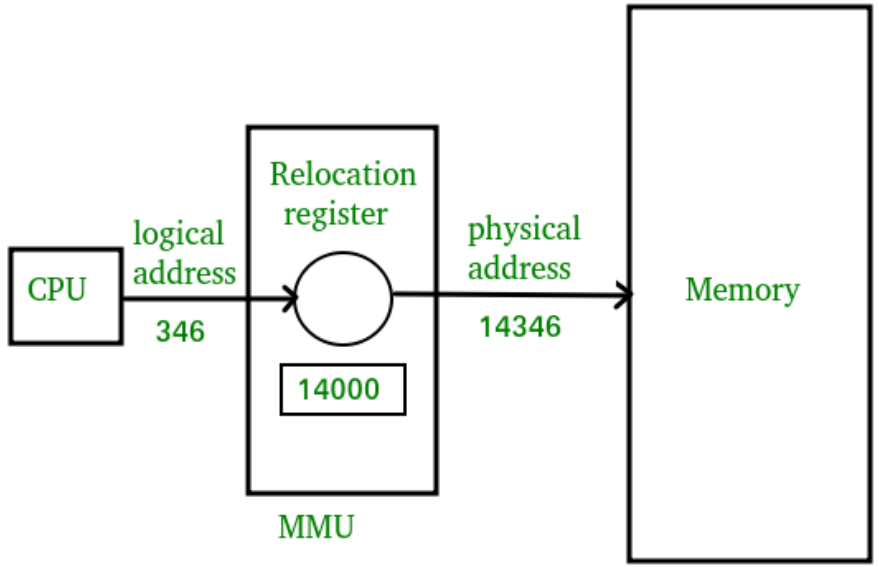
Q2：内存分配
（Memory Allocation）

Q3：内存保护
（Memory Protection）

Q4：内存扩充

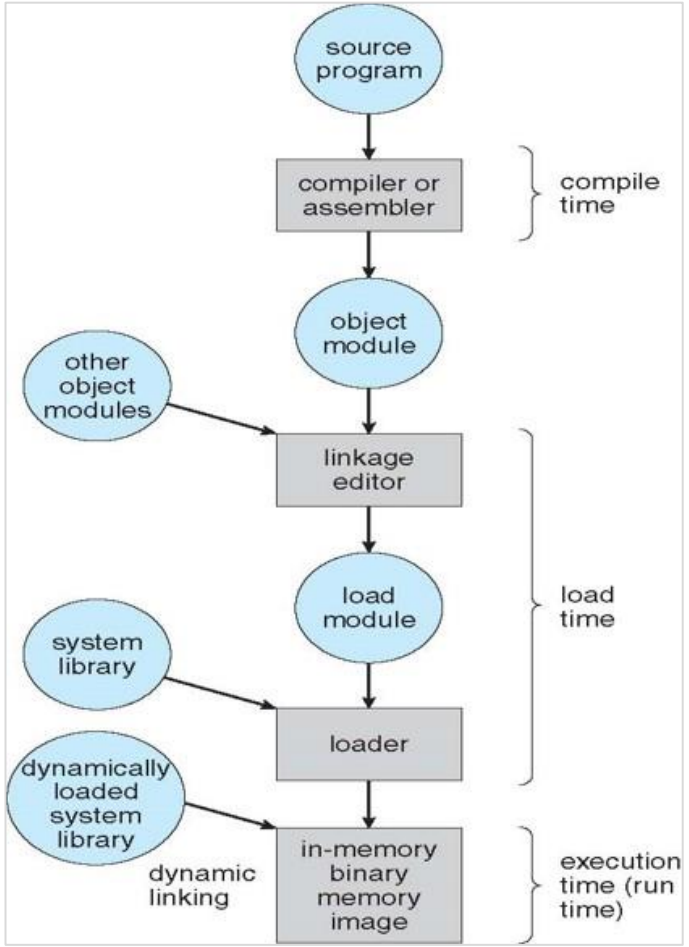
地址绑定

主存管理中主要问题的处理：Address Binding



地址绑定（Address Binding）

逻辑地址 ⟷ 物理地址



符号地址

目标文件内的可重定位地址

可执行文件内的可重定位地址

加载到内存的逻辑地址

地址绑定

主存管理中主要问题的处理：Address Binding

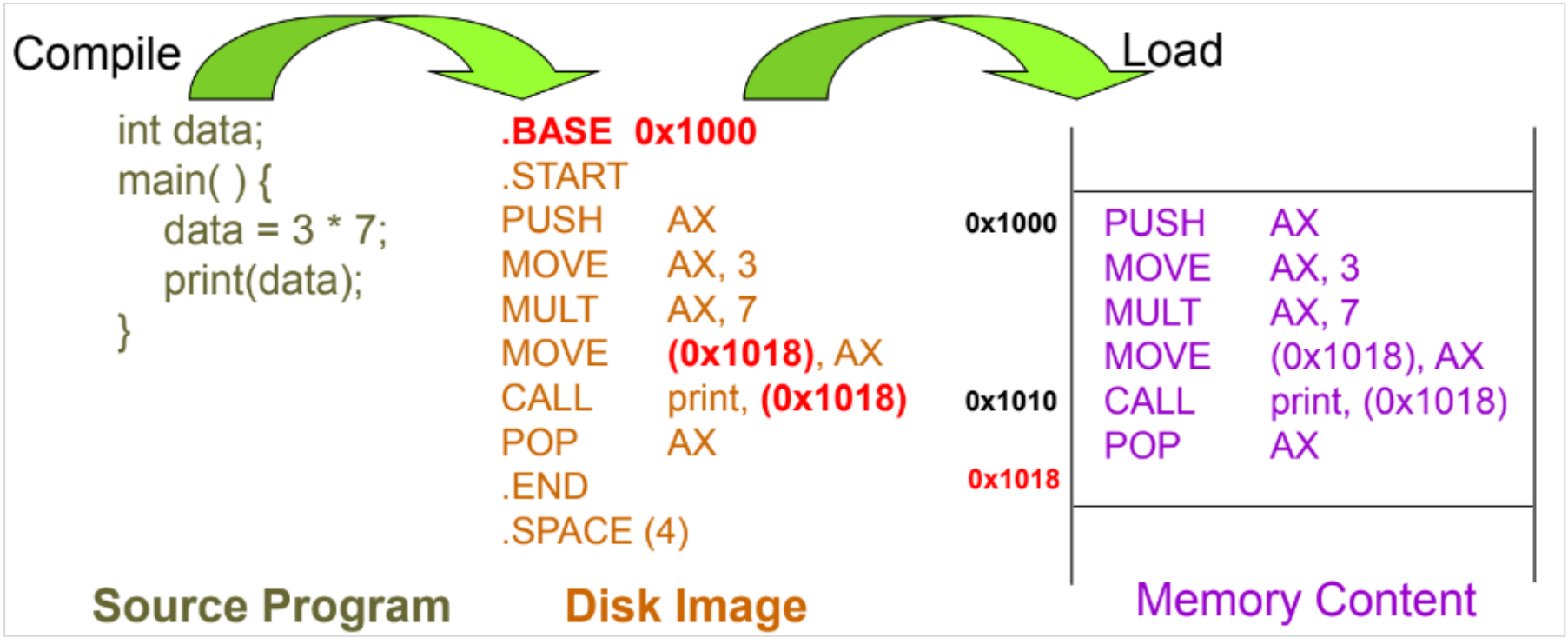
When?

逻辑地址 ⟷ 物理地址

- Compile time
- load time
- run time

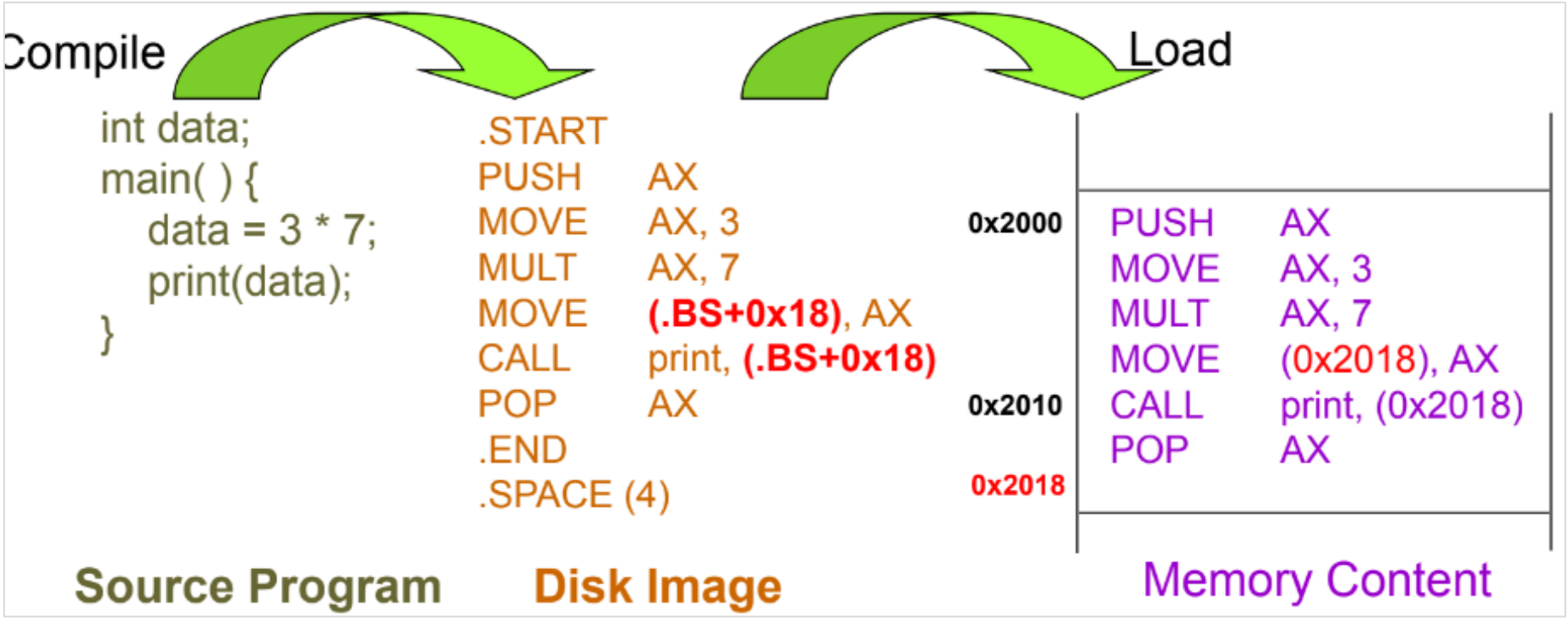
地址绑定

编译时绑定：示例



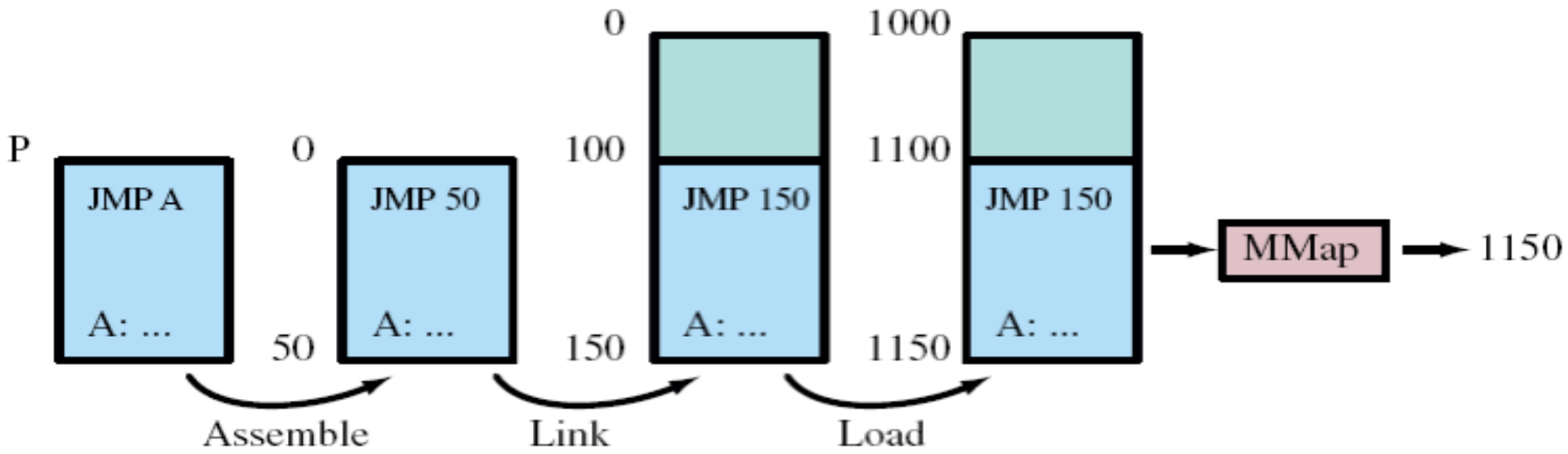
地址绑定

加载时绑定：示例



地址绑定

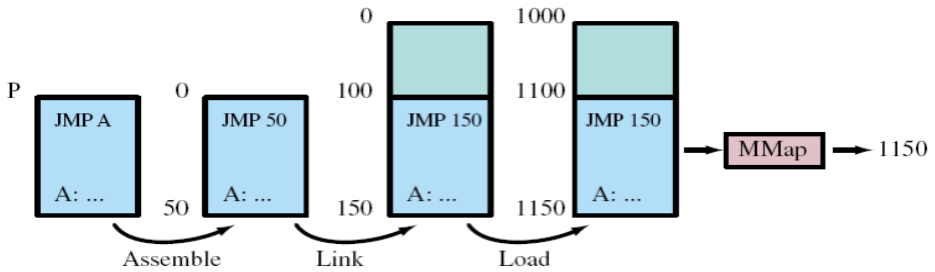
运行时绑定：示例



地址绑定

静态重定位 v.s. 动态重定位

- ✓ Static relocation
  - bind instruction or data to absolute address at load time
- ✓ Dynamic relocation 动态重定位
  - bind instruction or data to absolute address at run time



内存分配

主存管理中主要问题的处理

Q1：地址绑定
(Address Binding)

Q2：内存分配
(Memory Allocation)

Q3：内存保护
(Memory Protection)

Q4：内存扩充

?

- 连续内存分配
- 页式分配与管理
- 段式分配与管理

内存保护

主存管理中主要问题的处理

Q1：地址绑定
(Address Binding)

Q2：内存分配
(Memory Allocation)

Q3：内存保护
(Memory Protection)

?

Q4：内存扩充

内存保护

主存管理中主要问题的处理: memory protection

Q1：地址绑定
(Address Binding)

Q2：内存分配
(Memory Allocation)

Q3：内存保护
(Memory Protection)

Q4：内存扩充

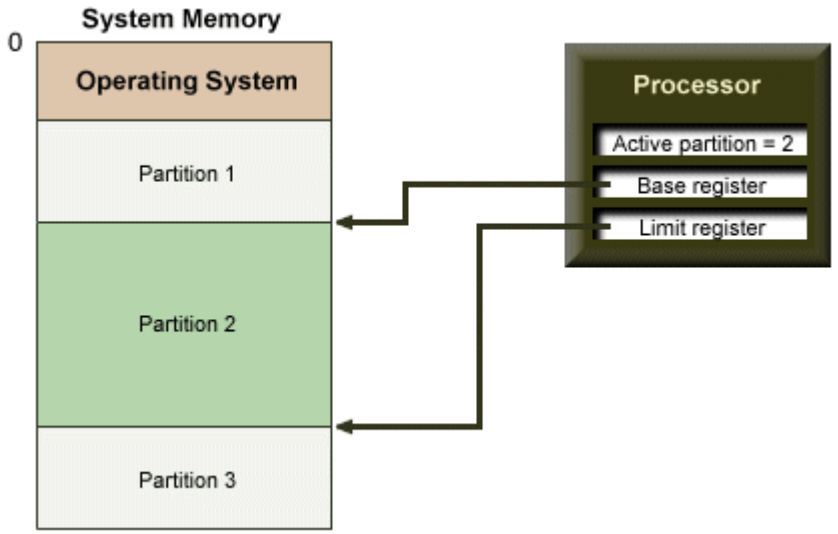
并发环境下，存在多个任务可能同时操作相同的内存区域，因此，需要对内存加以保护，避免内存访问冲突

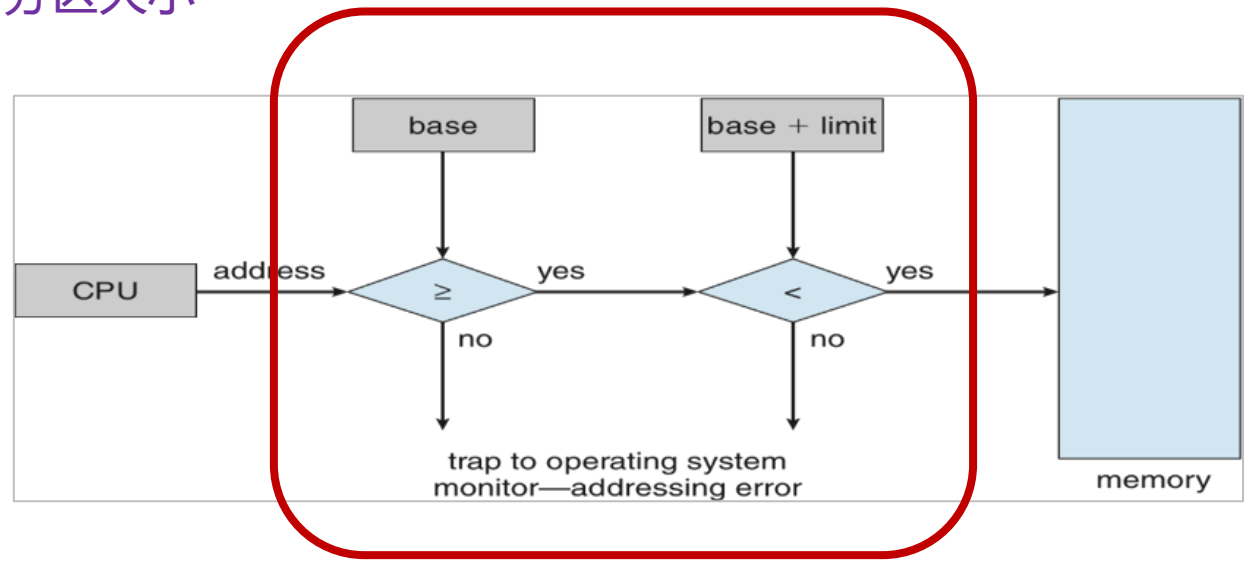
内存保护

主存管理中主要问题的处理: memory protection

示例：固定分区分配下的内存保护

为每个进程分配一块内存分区，
设置Base register = 分区基地址，Limit register = 分区大小





借助硬件保护机制，进行内存保护

问题：为何需要硬件辅助？纯软件方式实现这种内存保护行不行？

内存扩充

主存管理中主要问题的处理

Q1：地址绑定
(Address Binding)

Q2：内存分配
(Memory Allocation)

Q3：内存保护
(Memory Protection)

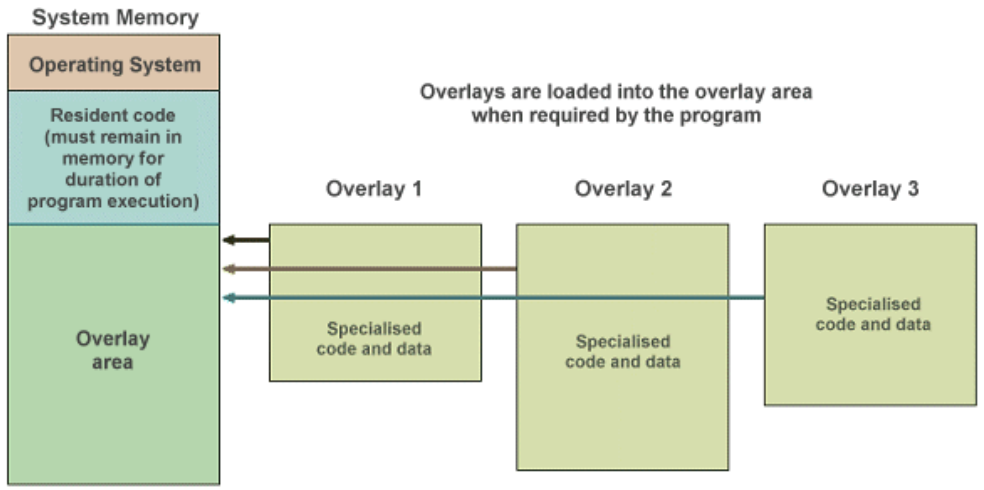
Q4：内存扩充

?

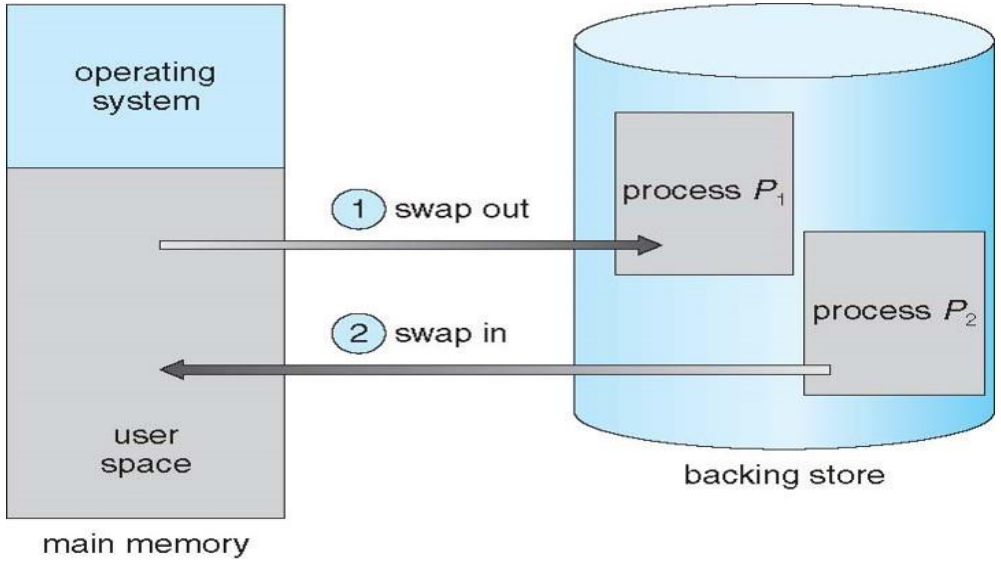
内存扩充

主存管理中主要问题的处理：内存扩充

Overlay vs. Swap



Overlay(覆盖)

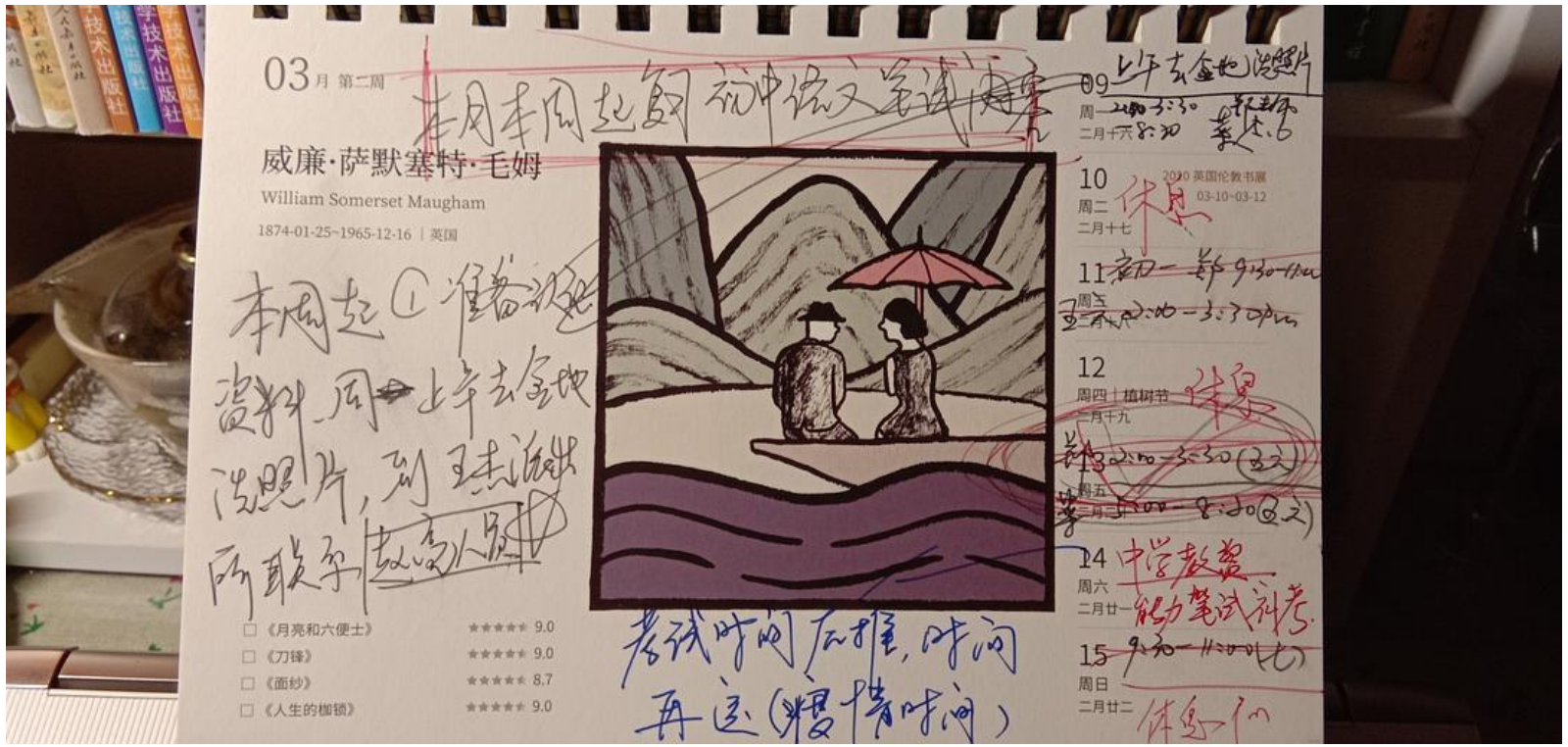


Swap(交换)

内存扩充

主存管理中主要问题的处理：内存扩充

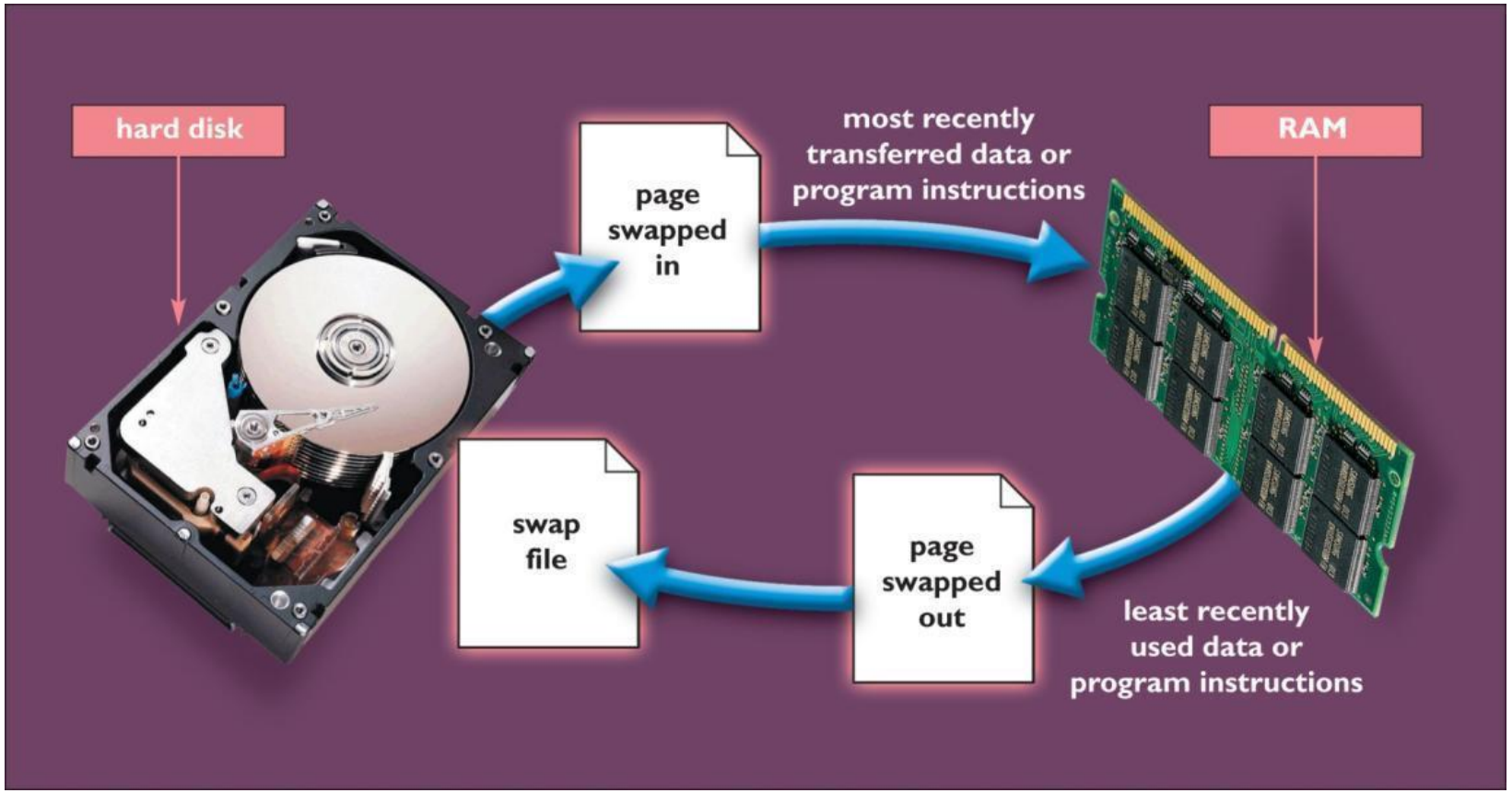
内存扩充的意义之一：好记性不如烂笔头



内存扩充

主存管理中主要问题的处理：内存扩充

Swap：虚存机制的重要基础



内存扩充

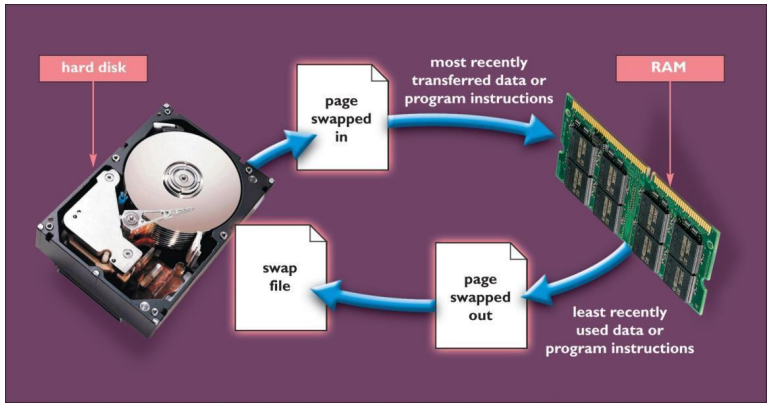
主存管理中主要问题的处理：内存扩充

Swap：虚存机制的重要基础

通过交换控制系统中的多道程序度（Degree of Multiprogramming）

系统中同时并发运行任务过多时：换出（Swap Out）

系统中同时并发运行任务很少时：换入（Swap In）



连续内存分配

Contiguous Memory Allocation

02

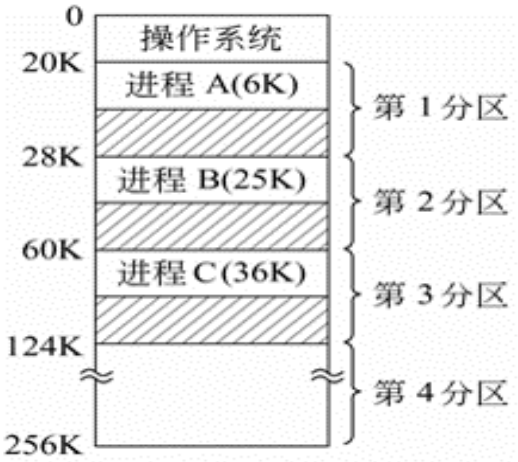
2-连续内存分配 Fix-sized v.s. Variable-Sized

为每位进程分配连续的内存：分区分配

Fixed-sized Partition Allocation

variable-sized Partition Allocation

示例：



思考：如何进行数据结构设计，支持这种连续内存分配方式？如何进行分配？

使用一个数组来记录各个连续内存块的分配情况
分配：将某个内存块分配给进程，把这个内存块的状态设为已分配
释放：将进程占用的内存块释放，并将内存块的状态设为空闲

思考：这种分配方式存在什么问题？

内存块内存在内部碎片（Internal Fragmentation）

2-连续内存分配 Fix-sized v.s. Variable-Sized

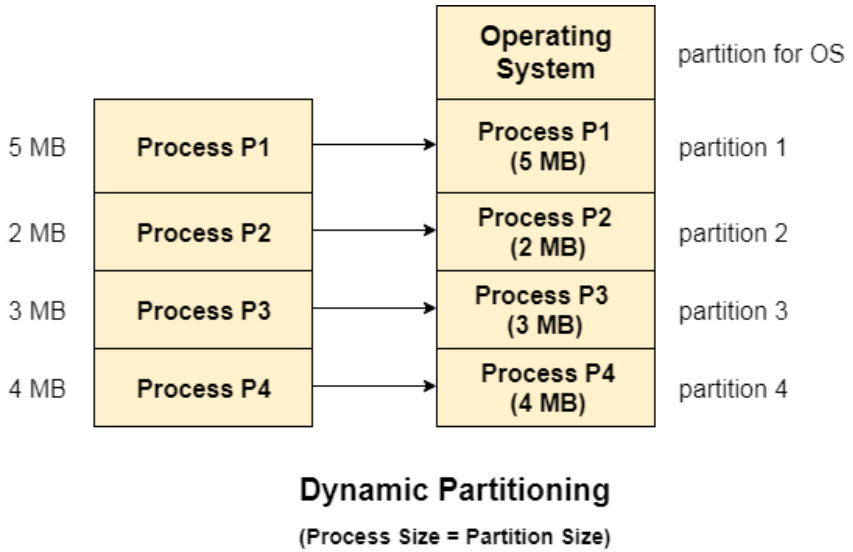
Fixed-sized Partition Allocation

思考：这种分配方式存在什么问题？

存在外部碎片（External Fragmentation）

variable-sized Partition Allocation

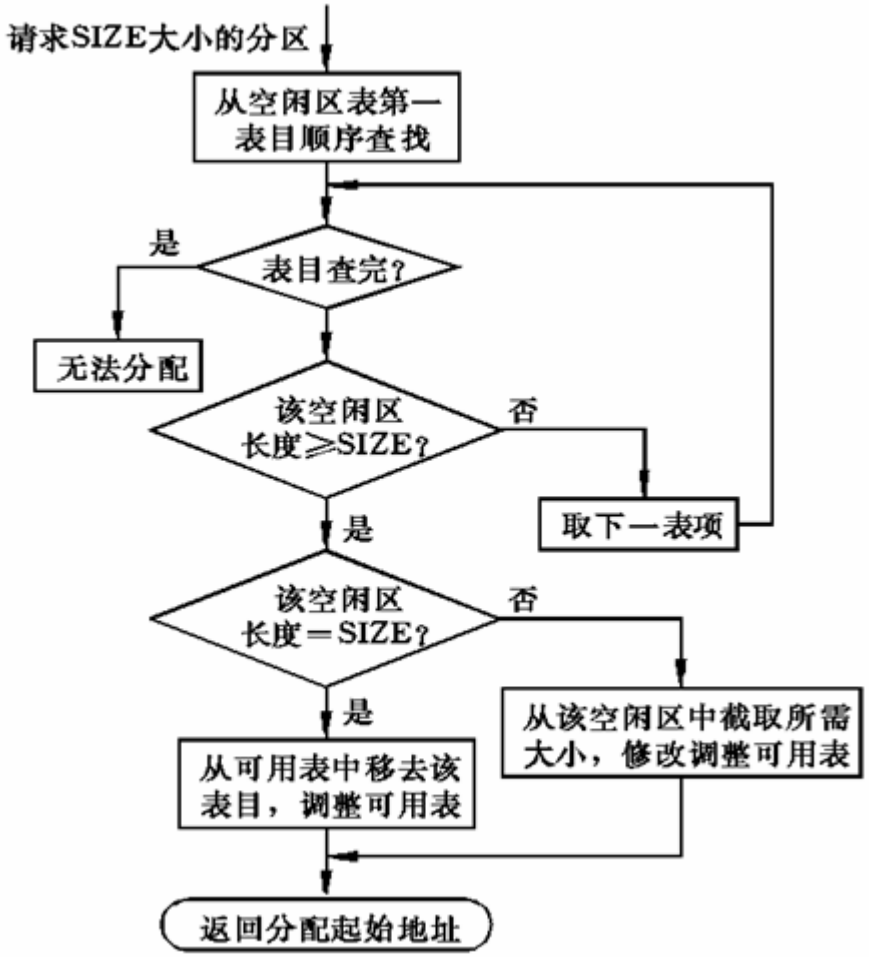
示例：



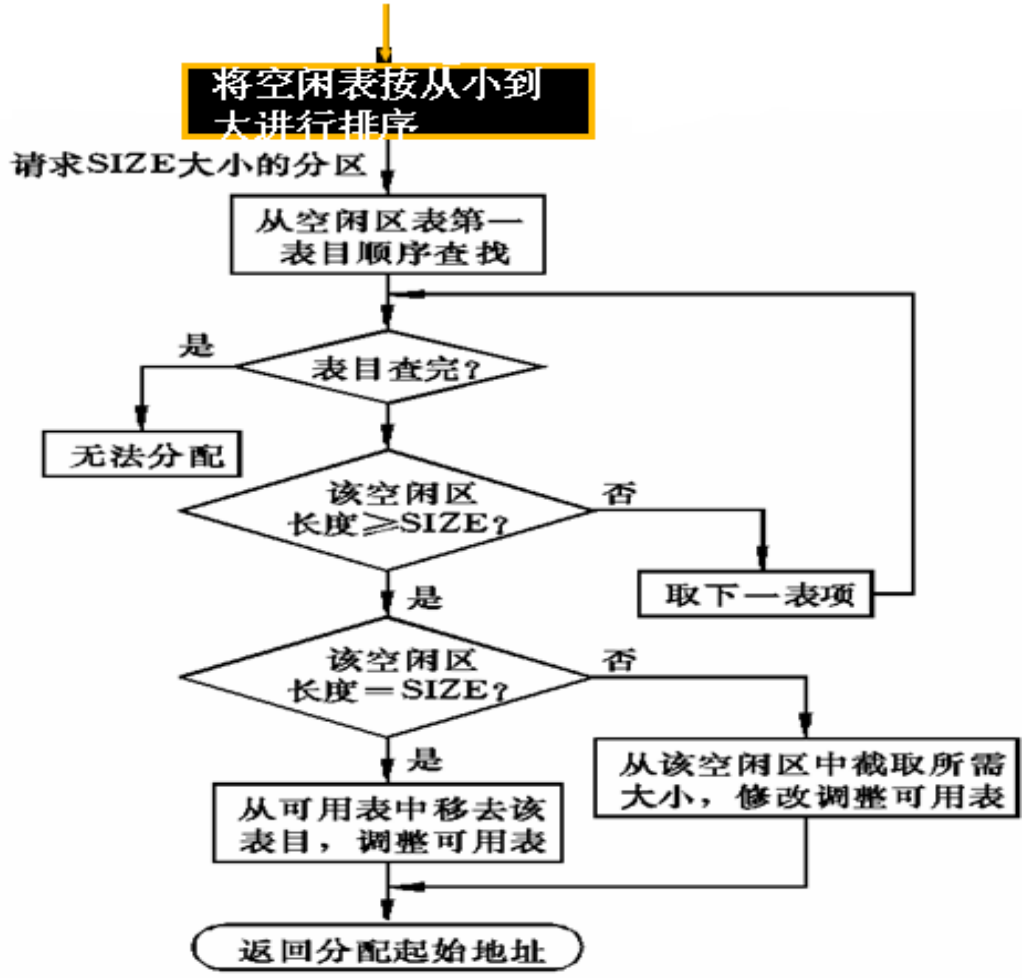
思考：如何进行数据结构设计，支持这种连续内存分配方式？如何进行分配？

链表 （或动态数组）

首次适配分配算法（First-Fit）



最佳适配分配算法（Best-Fit）

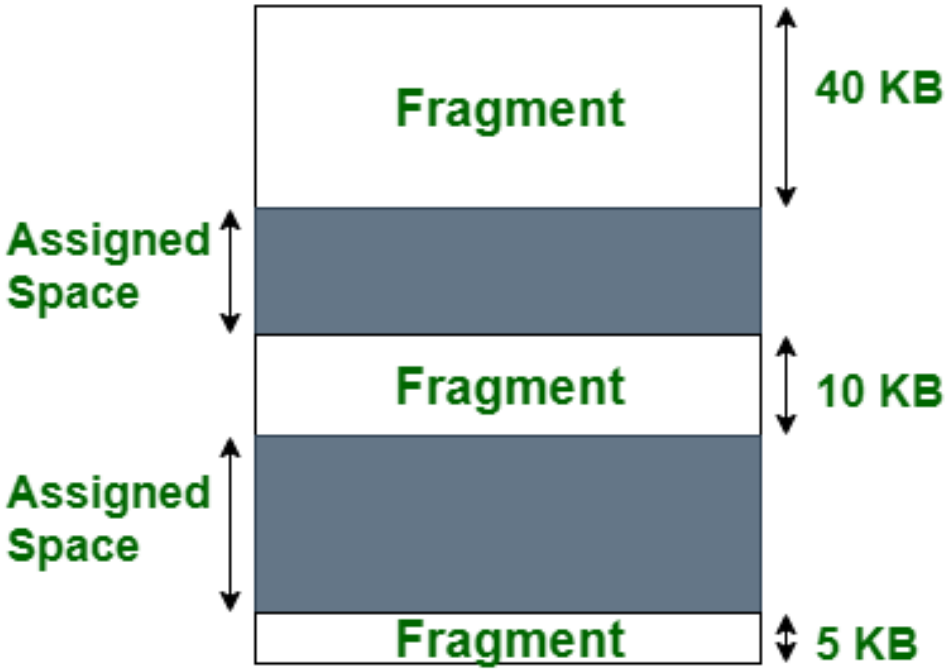


最差适配分配算法（Worst-Fit）

从空闲区块中最大的那个中，为进程分配所需内存

Variable-Sized Partition Allocation

External Fragmentation



Process 07 needs 50KB memory space

2-连续内存分配 Variable-Sized Partition Allocation

Variable-sized Partiton Allocation (三种典型算法)

First-Fit

首次适应分配算法

- Q: 如何合理地组织数据结构来实现这种分配方式? → 链表，空闲块按起始地址升序
- Q: 算法的关键弱点是什么? → 外部碎片问题

Best-Fit

最佳适应分配算法

- Q: 如何合理地组织数据结构来实现这种分配方式? → 链表，空闲块按大小升序
- Q: 算法的关键弱点是什么? → 容易产生很多细小外部碎片

Worst-Fit

最差适应分配算法

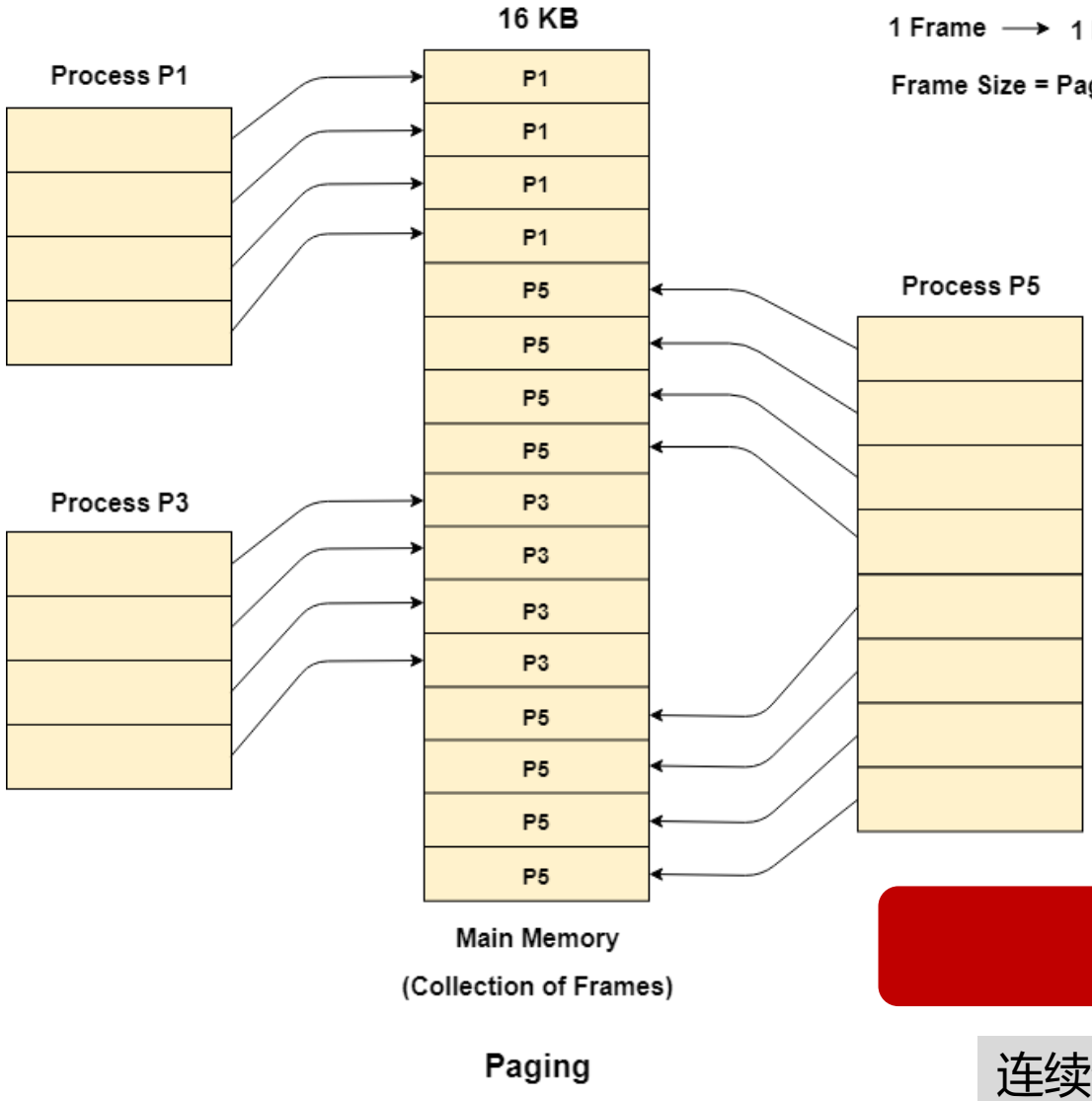
- Q: 如何合理地组织数据结构来实现这种分配方式? → 链表，空闲块按大小降序
- Q: 算法的关键弱点是什么? → 外部碎片问题

分页机制

Paging Mechanism

03

什么是分页



将进程的逻辑地址空间按页划分

将物理内存的地址也按页划分

每个页代表一组连续的地址

进程被分配的页在物理内存中不一定连续（如P5）

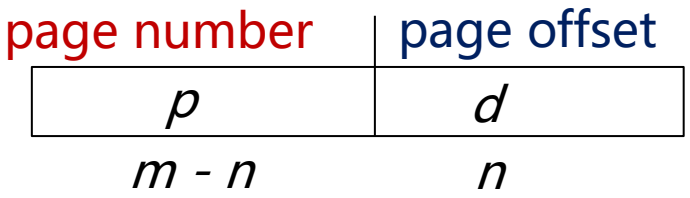
通过一个页表来维持进程中逻辑页与物理页之间的映射关系

Q:为何引入分页?

连续分区分配导致较大碎片问题，引入分页加以控制

- **分页的三个重要环节**

地址划分 → 建立地址映射（页表） → 地址翻译

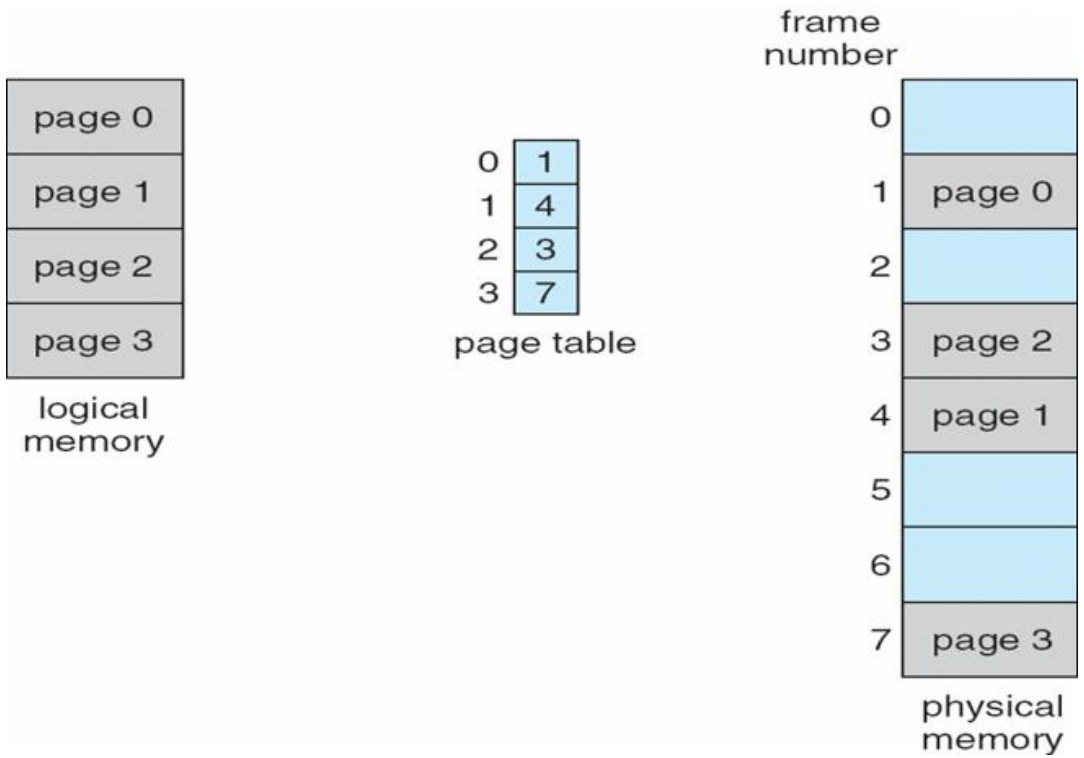


地址被按位拆分成页号和页内偏移这2部分

什么是分页

- **分页的三个重要环节**

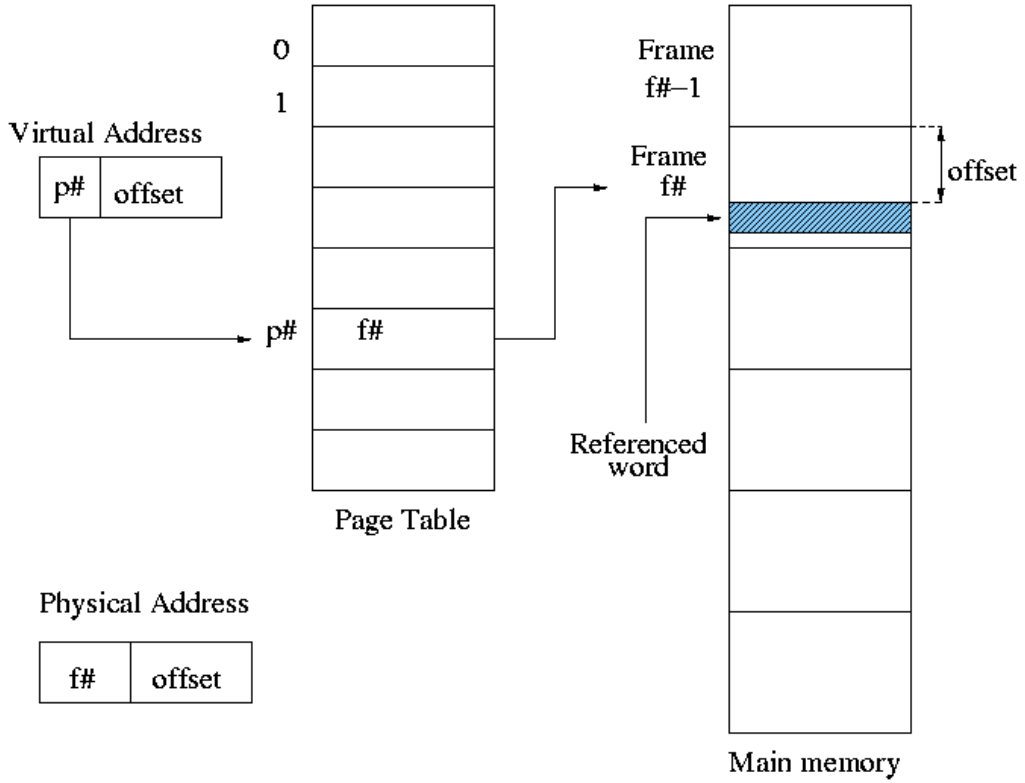
地址划分 → 建立地址映射（页表） → 地址翻译



什么是分页

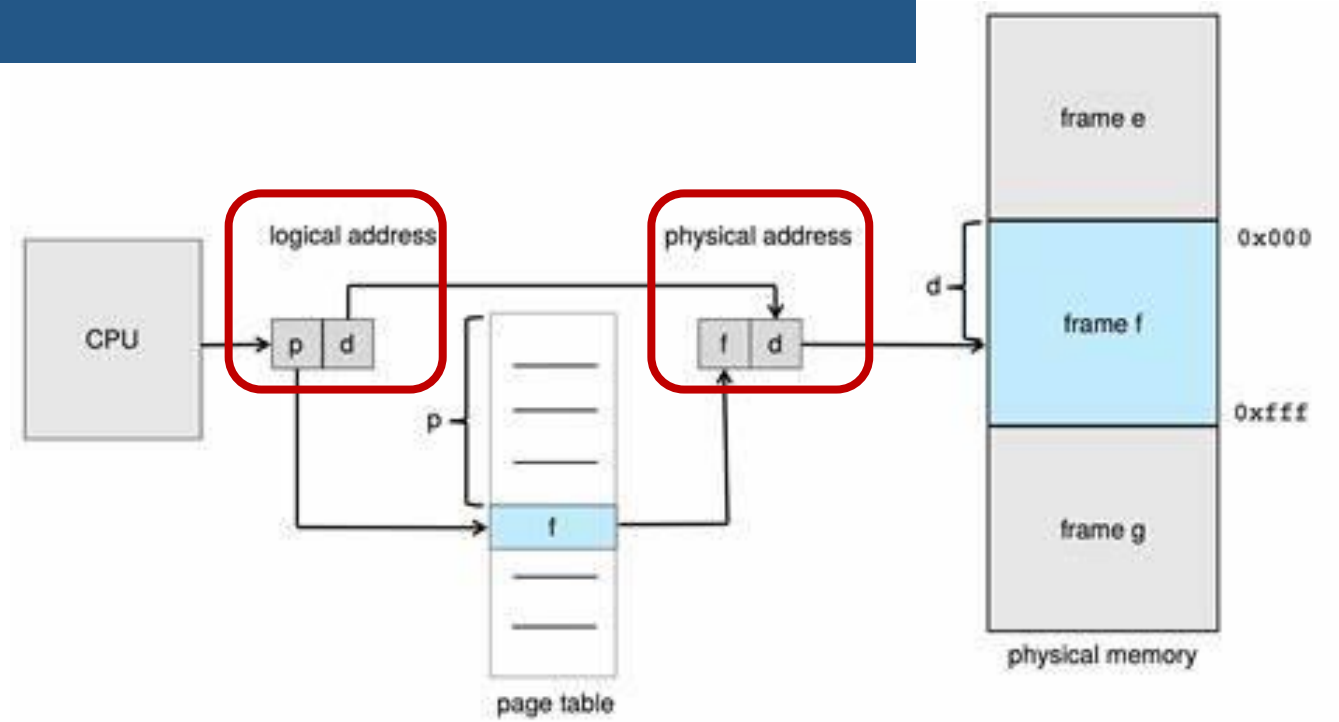
- **分页的三个重要环节**

地址划分 → 建立地址映射（页表） → 地址翻译



页地址翻译

分页下的地址翻译过程

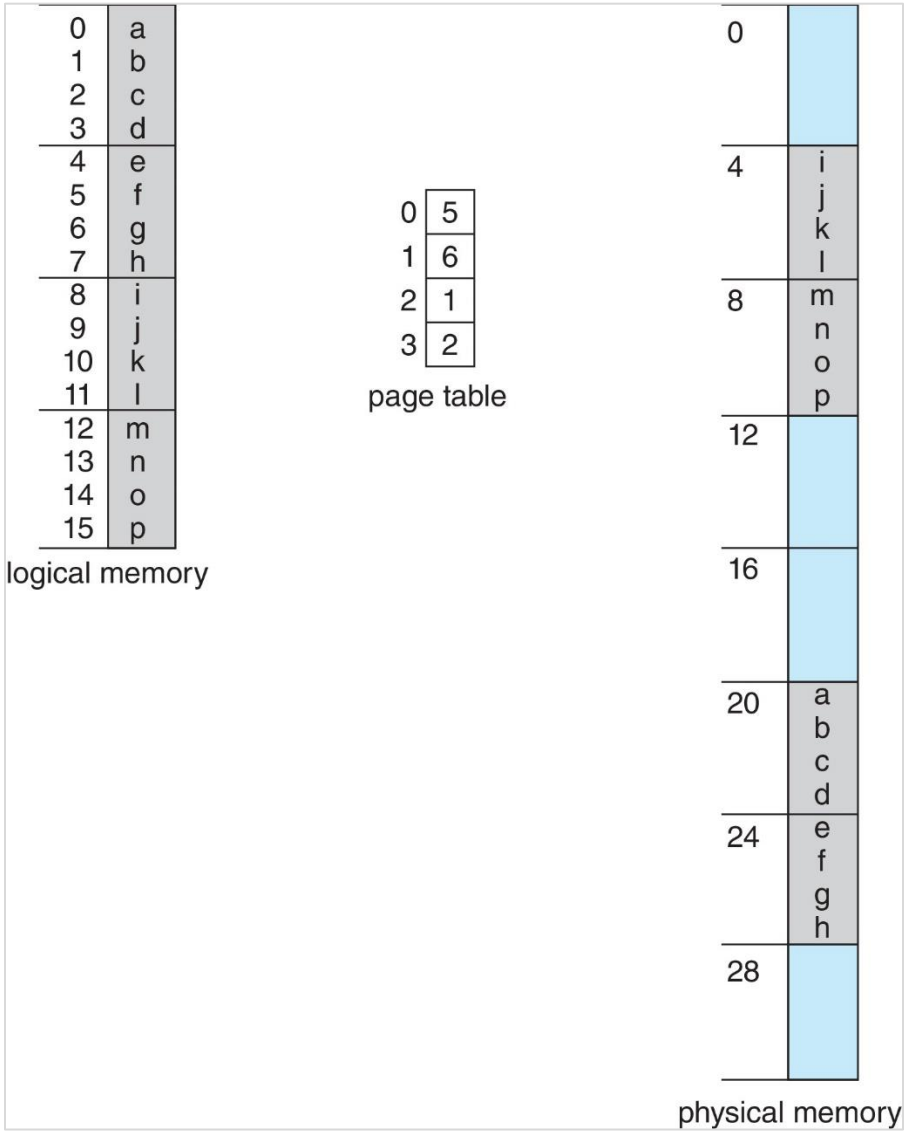


逻辑地址 —翻译→ 物理地址

Q：这个页式地址翻译的过程可不可以用纯软件的形式实现？为什么？

页地址翻译

分页下的地址翻译练习



假设左边的示例中，对应的是8位处理器下的系统

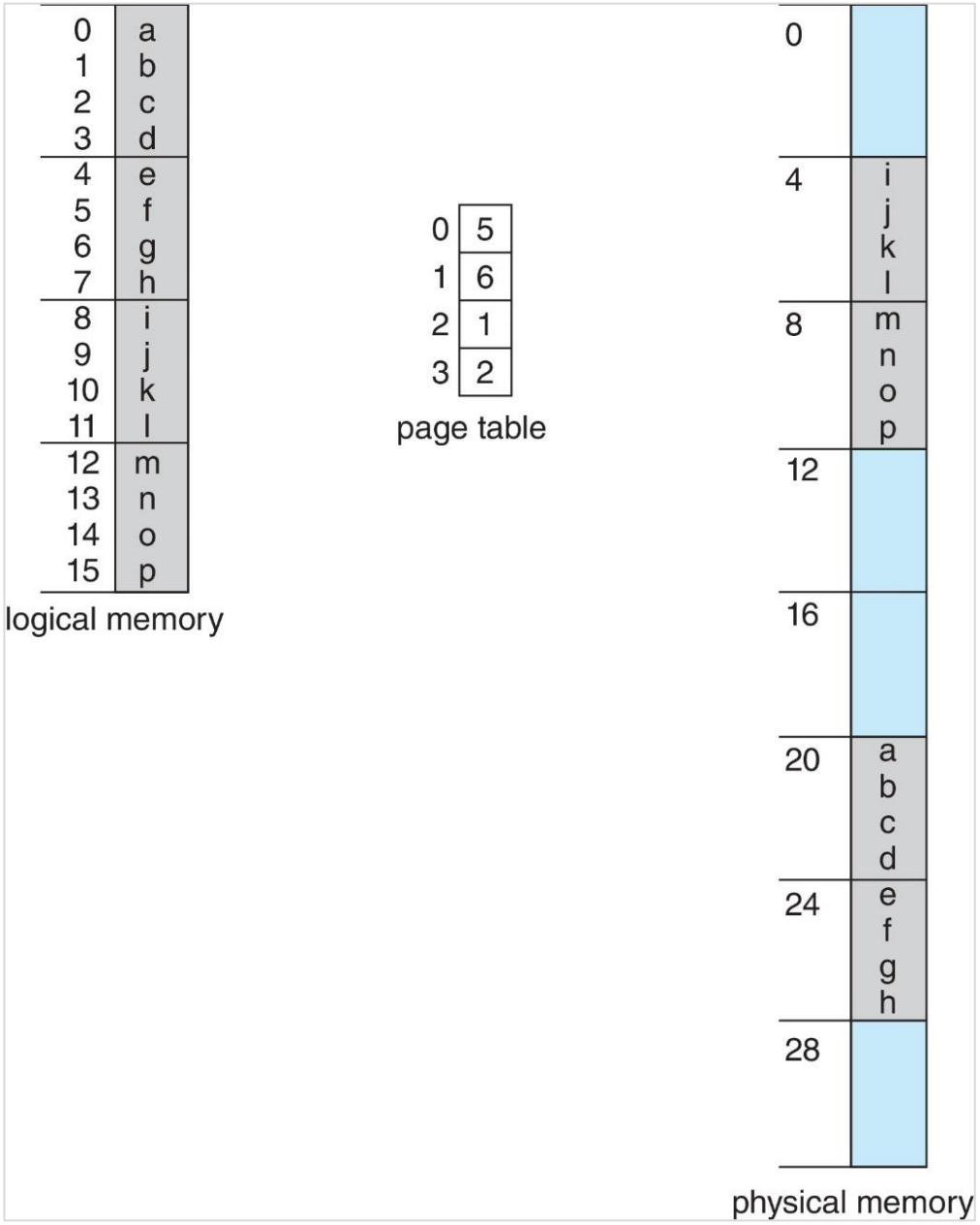
8位地址：

页号 (6bit)　　页内偏移 (2bit)

问题：用二进制标识的逻辑地址addr=1001对应的字母是？其物理地址是？

填空题 10分

设置

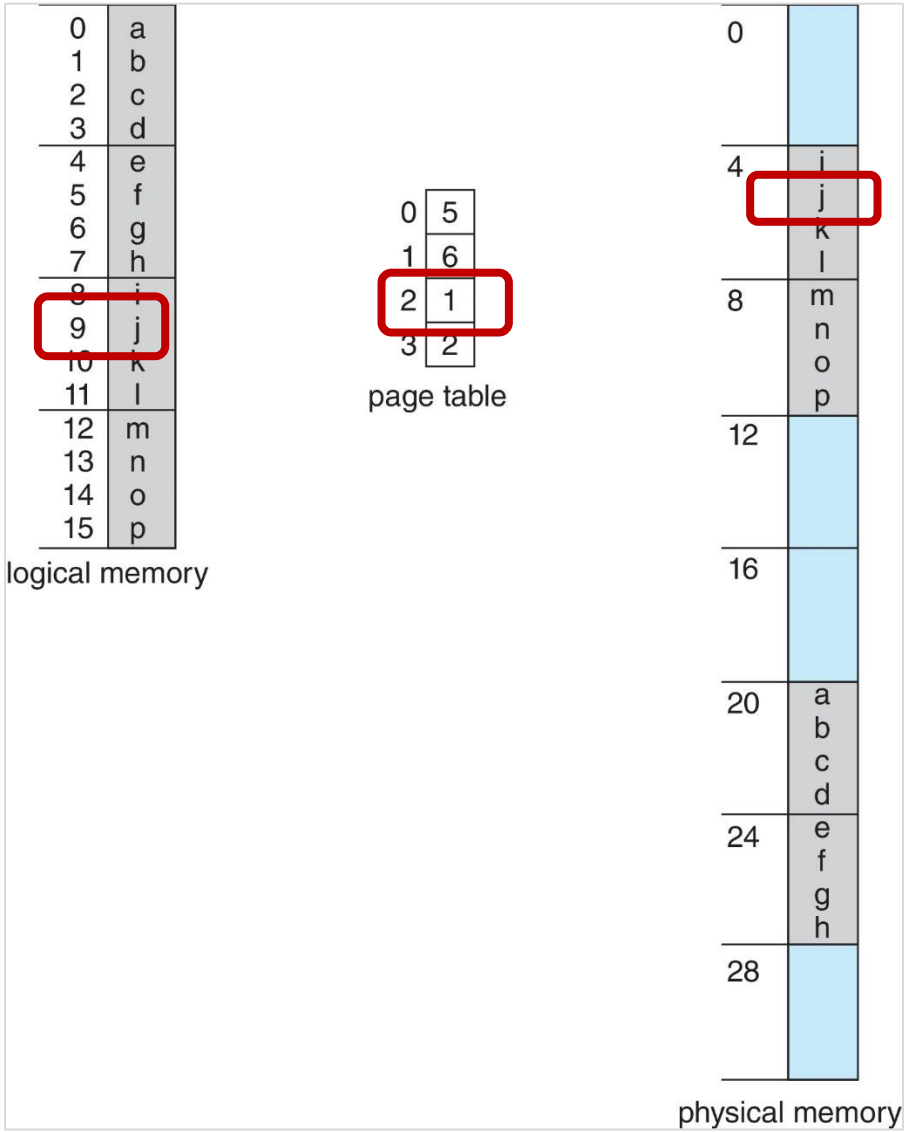


用二进制标识的逻辑地址addr=1001对应的字母是（ [填空1] ），其物理地址是（ [填空2] ）。

作答

页地址翻译

分页下的地址翻译练习

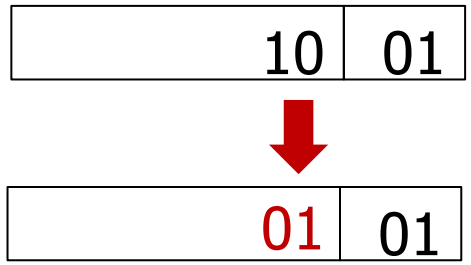


假设左边的示例中，对应的是8位处理器下的系统

8位地址：

页号 (6bit)　　页内偏移 (2bit)

问题：用二进制标识的逻辑地址addr=1001对应的字母是？其物理地址是？



Summary

小结：

- 主存管理概念
- 连续内存分配
- 分页机制

连续内存分配的局限性（缺点）

动态分区分配运作过程中的内存格局变化特点

1. 操作系统内有1个执行CPU周期长度为40秒的进程，若系统采用多级反馈队列调度，每级队列均采用RR算法，最高优先级队列的时间片=2秒，向下每一级的时间片增加5秒。请问进程会被中断 _____ 次，最终在第 _____ 级队列执行结束?

2.操作系统调度的基本对象是什么?

2.操作系统调度的基本对象是什么?

在未引入线程的操作系统中，调度的基本单位是进程；
在引入了线程概念的操作系统中，调度的基本单位是线程（内核级线程）

死锁练习

3.有三类资源A(17)、B(5)、C(20)。有5个进程P1-P5.T0时刻系统状态分配如下

	最大需求	已分配
P1	5 5 9	2 1 2
P2	5 3 6	4 0 2
P3	4 0 11	4 0 5
P4	4 2 5	2 0 4
P5	4 2 4	3 1 4

问基于银行家算法:

(1)T0时刻是否为安全状态，若安全，请给出安全系列。

(2)T0时刻，P2:Request(0,3,4),能否分配，为什么?

(3)在(2)的基础上P4:Request(2,0,1),能否分配，为什么?

(4)在(3)的基础上P1:Request(0,2,0),能否分配，为什么?

谢 谢!
Thank you!