北京郵電大學
BEIJING UNIVERSITY OF POSTS AND TELECOMMUNICATIONS

Computer Organization and Architecture

Chapter 8

Operating System Support

School of Computer Science（National Pilot Software Engineering School）
AO XIONG (熊翱)
xiongao@bupt.edu.cn

Preface

We have learned:

- Basic Concepts and Computer Evolution 基本概念和计算机发展历史
- Performance Issues 性能问题
- Top level view of computer function and interconnection 计算机功能和互联结构顶层视图
- Cache Memory cache存储器
- Internal Memory 内部存储器
- External Memory 外部存储器
- Input& Output 输入输出
  - External devices 外部设备
  - I/O modules IO模块
  - Programmed I/ 编程式IO
  - interrupt-driven I/O 中断式IO
  - DMA & DCA DMA 和DCA
  - I/O channels and processors IO通道和处理器
  - External Interconnection 外部接口

Preface

We will focus the following contents today:

- Operating System Support 操作系统支持
  - What is operating system? 什么是操作系统?
  - How does the operating system schedule processes or tasks? 操作系统如何调度处理器和任务?
  - How to manage memory in operating system? 操作系统如何管理内存?

Outline

- Operating System Overview 操作系统概览
- Scheduling 调度
- Memory Management 内存管理

Question

- 什么是操作系统？
- 有哪些常用的操作系统？

Question

- 什么是操作系统？
  - 操作系统是一组主管并控制计算机操作、运用和运行硬件、软件资源，提供公共服务来组织用户交互的相互关联的系统软件程序。根据运行的环境，操作系统可以分为桌面操作系统，手机操作系统，服务器操作系统，嵌入式操作系统等。
- 有哪些常用的操作系统？
  - DOS
  - windows系列
  - UNIX系列：HP-UX，Solaris，AIX
  - linux及各种变种
  - OS/2
  - iOS，Android
  - Mac OS

Operating system overview 操作系统概览

- Operating System Objectives and Functions 操作系统的目标和功能
- Types of Operating Systems 操作系统类型

Definition 定义

- Operating system, or OS for short, is a computer program that manages computer hardware and software resources. Operating system，简称OS，是管理计算机硬件与软件资源的计算机程序
- The operating system deals with such basic matters as 操作系统需要处理一些基本的事务
  - managing and configuring memory 管理与配置内存
  - determining the priority of system resource supply and demand 确定系统资源分配的优先次序
  - controlling input devices and output devices 控制输入与输出设备
  - operating networks and managing file systems 操作网络与管理文件系统
- The operating system also provides an operation interface for users to interact with the system 操作系统也提供一个让用户与系统交互的操作界面

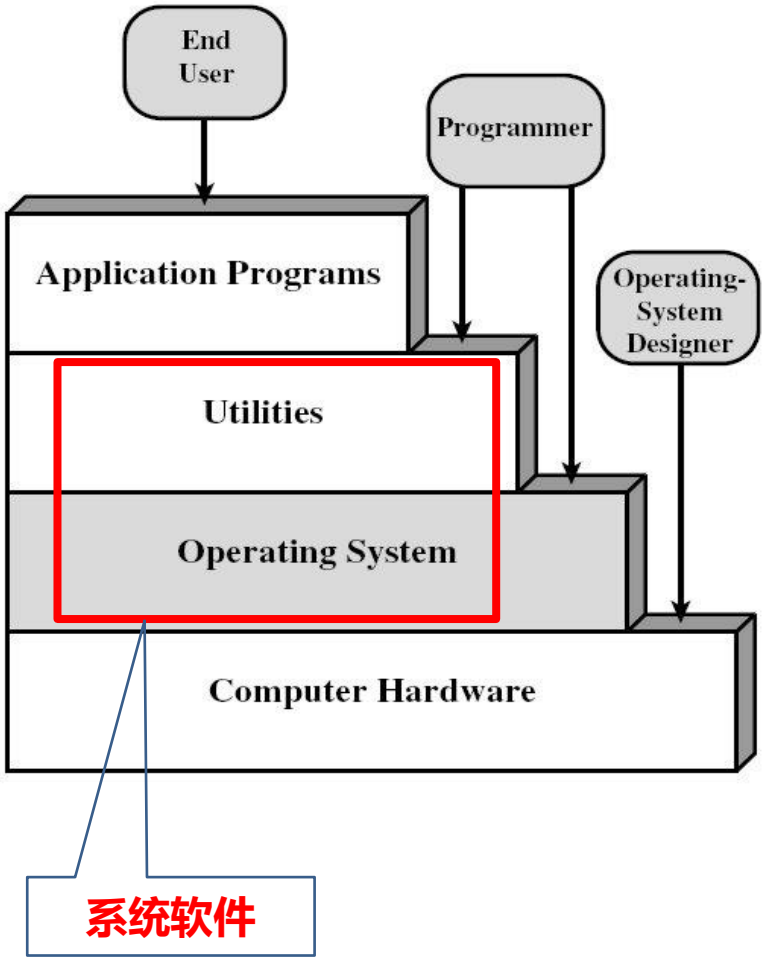
Objectives and Functions 目标和功能

- A system program developed to improve resource utilization and enhance computer system performance 为了满足提高资源利用率、增强计算机系统性能而开发的系统程序
- Convenience 便利
  - Shields the details of computer hardware 屏蔽硬件细节
  - Provides convenient interfaces for users 提供用户接口
  - Making the computer easier to use 使得计算机容易使用
- Efficiency 高效
  - Process scheduling 进程调度
  - Memory management 内存管理
  - Fully exploit the power of computer 充分发挥计算机的能力

Characteristic

- The most fundamental system program is the operating system 操作系统是最基本的系统程序
  - Provides an interface between the user and the computer 提供用户和计算机的接口
  - Manages the computer resources 管理计算机资源
  - Provides an environment for the application program execution 为应用程序执行提供环境
  - Improves the efficiency and stability of program execution 提高程序执行的效率和稳定性

Layers of a Computer System 计算机系统层次结构

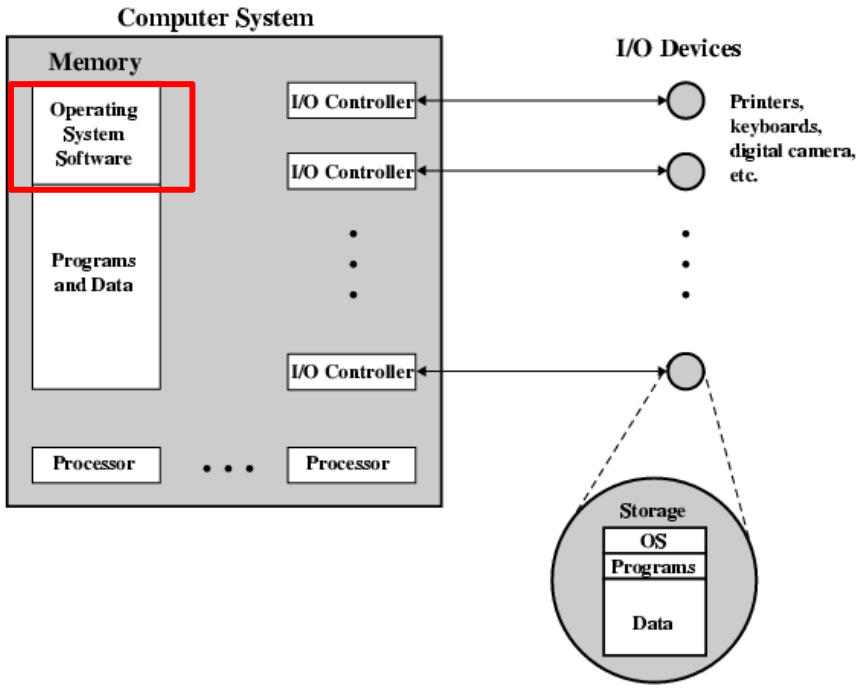


- 系统软件是指控制和协调计算机及外部设备，支持应用软件开发和运行的系统，主要功能是调度，监控和维护计算机系统
- 系统软件包括操作系统和一系列基本的工具，比如编译器，数据库管理，存储器格式化，文件系统管理等，支持计算机系统正常运行并实现用户操作
- 操作系统是最重要的系统软件，位于硬件和工具之间
- 使得上层方便地访问和使用计算机系统提供的资源和服务

Operating system services 操作系统的服务

- Program creation 程序创建
- Program execution 程序执行
- Access to I/O devices I/O设备访问
- Controlled access to files 文件访问控制
- System access 系统访问
- Error detection and response 错误检测和处理
- Accounting 统计

OS as a resource management　OS作为资源管理



- 操作系统是一个计算机程序，执行也是由处理器来进行，通过给处理器提供一系列指令，来执行操作系统程序
- 操作系统不能一直占用了处理器，而是需要放弃对处理器的控制，让处理器去执行应用程序。之后，它再次获得控制权，为下一个应用程序的执行进行准备
- 操作系统有一部分驻留在内存中，称为操作系统内核
- 操作系统还要对I/O的使用进行分配

Operating system overview 操作系统概览

- Operating System Objectives and Functions 操作系统的目标和功能
- Types of Operating Systems 操作系统类型

Types of operating system 操作系统类型

Operating system support batch processing or not?
是否支持批处理

- Interactive 交互式
  - User usually interacts with the computer by keyboard or display 用户通常用键盘或显示器与计算机交互
  - Request to execute jobs or process transactions 请求执行作业或处理事务
- Batch 批处理
  - Multiple user programs are packaged and submits to the computer in batches for processing 多个用户程序打包后，操作员成批提交给计算机进行处理
  - After processing, print the results 处理结束后，打印结果

Types of operating system 操作系统类型

Support multiple programs or not?
是否支持多道程序

- Single program (Uni-programming) 单道程序
  - Processor can only process one program at a time 处理器一次只能处理一个程序
- Multi-programming (Multi-tasking) 多道程序
  - Processor can process multiple programs at one time 处理器一次可以处理多个程序
  - Multiple programs loaded into memory at the same time 多个程序同时装入内存
  - processor switches between programs and processes 处理器在程序之间进行切换，同时处理

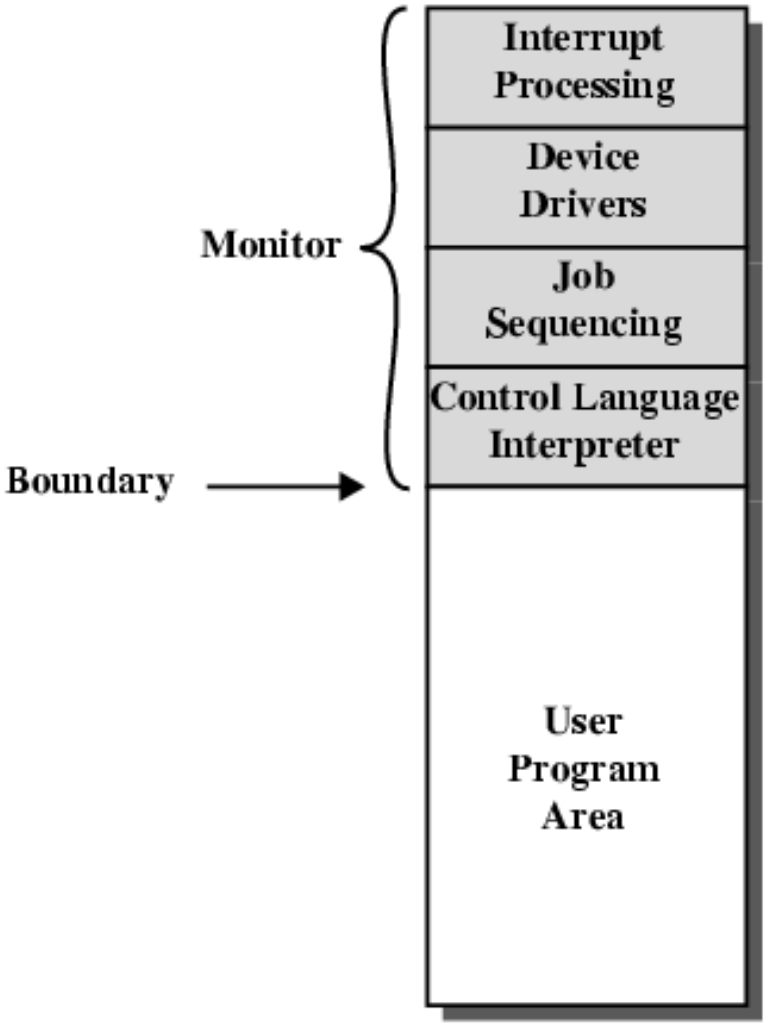
Early systems 早期的系统

- Late 1940s to mid 1950s 40年代末到50年代
  - No operating System 没有操作系统
  - Programmer interact directly with hardware 程序员直接和硬件交互
  - Two main problem: schedule, setup time 两个主要问题：调度和安装
- Scheduling 调度
  - programmer needs to determine how long to use 程序员要确定需要多长时间
  - Expected time is longer，a waste of computer resources预期时间超过实际时间，CPU浪费
  - Expected time is shorter, work cannot be completed预期时间少于实际时间，程序运行不完
- Setup time 程序安装
  - Scheduling takes a lot of time to install the program every time 每次都需要花时间安装程序
  - Waste a lot of time 浪费时间

Simple batch systems 简单批处理系统

- Simple batch system 简单批处理系统
  - Users submit jobs to operator 用户提交作业给操作员
  - Operator batches jobs 操作员批量提交作业
  - Monitor programs to perform these tasks 监控程序执行任务
- Resident Monitor program 驻留监控程序
  - Controls sequence of events to process batch 控制批量作业的执行顺序
  - When one job is finished, control returns to Monitor which reads next job 一个作业完成后，监控程序读下一个需要执行的作业
  - Monitor handles scheduling 监控程序承担了调度工作
- For processor
  - First executes the monitor program 开始执行监控程序
  - Monitor program execute job scheduling 监控程序进行作业调度
  - Execute the user program 执行用户程序

Memory layout for resident monitor 驻留监控程序的内存分布



- 多程序批量处理系统中，内存划分为监控程序和用户程序两个区域
- 监控程序包括：中断处理，设备驱动程序，作业排序，控制语言解释器
- 通过监控程序，实现了作业的调度。
- 用户程序提交之后，就可以按照顺序处理，不会浪费处理器

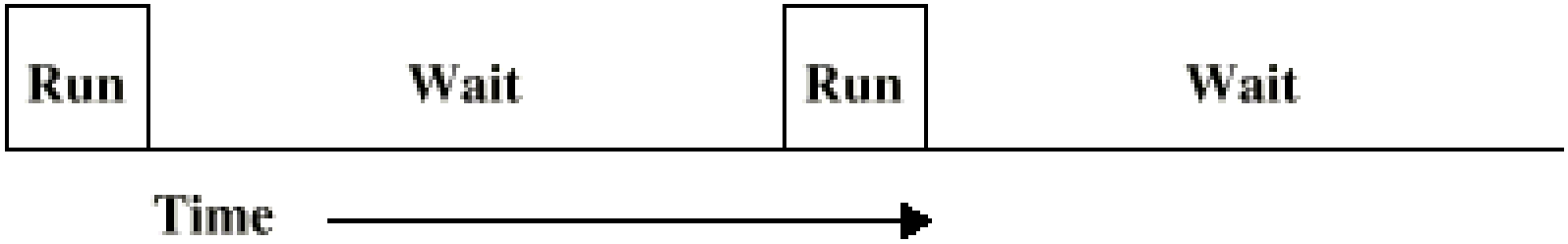
Required hardware support 需要的硬件支持

- Memory protection 内存保护
  - To protect the Monitor 保护监控程序
- Timer 定时器
  - To prevent a job monopolizing the system 防止一个作业独占处理器
- Privileged instructions 特权指令
  - Only executed by Monitor 只能由监控程序运行
  - e.g. I/O 比如I/O
- Interrupts 中断
  - Allows operation system for relinquishing and regaining control 允许操作系统放弃和获取控制权

Multi-programmed batch system 多程序批处理系统

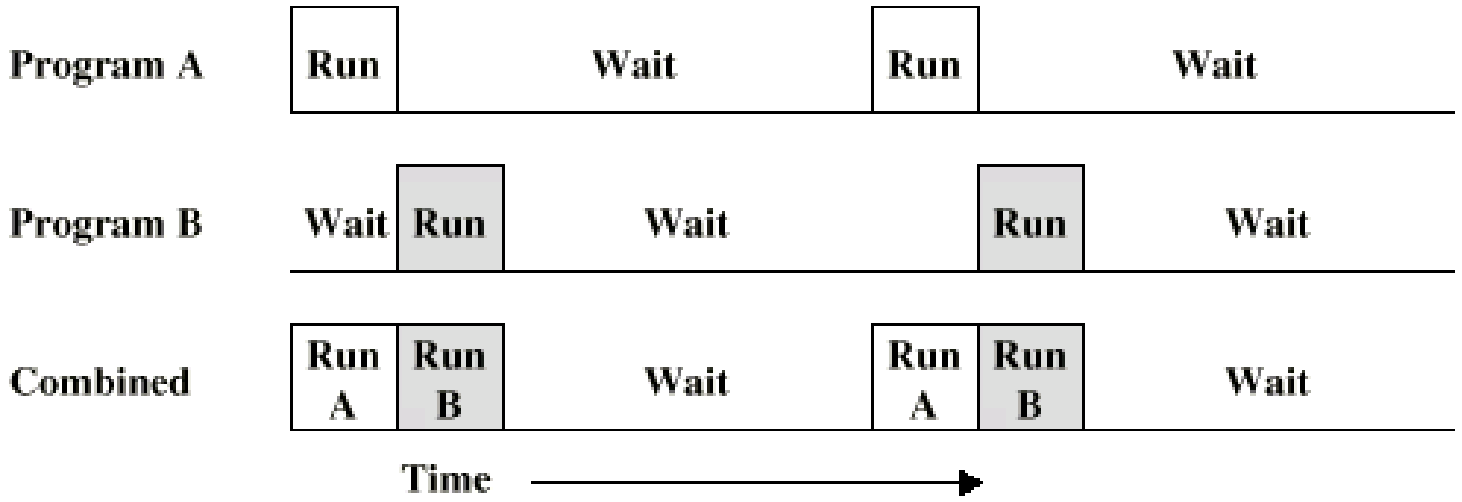
- Batch system performs monitoring and user programs in turn, which improves the utilization of the whole system 批处理系统轮流执行监控程序和用户程序，一定程度上提高了整个系统的利用率
- I/O devices very slow I/O设备慢
- Processors are also often idle 处理器还会经常处于空闲状态
- Multi-programmed batch systems 多程序批处理系统
  - Memory load multiple user programs 内存加载多个用户程序
  - When one program is waiting for I/O, another can use the CPU 一个程序等待I/O的时候，其他程序可以使用CPU
  - CPU keeps working CPU持续工作
  - Core idea of modern operation system 现代操作系统的核心

Single program 单个程序



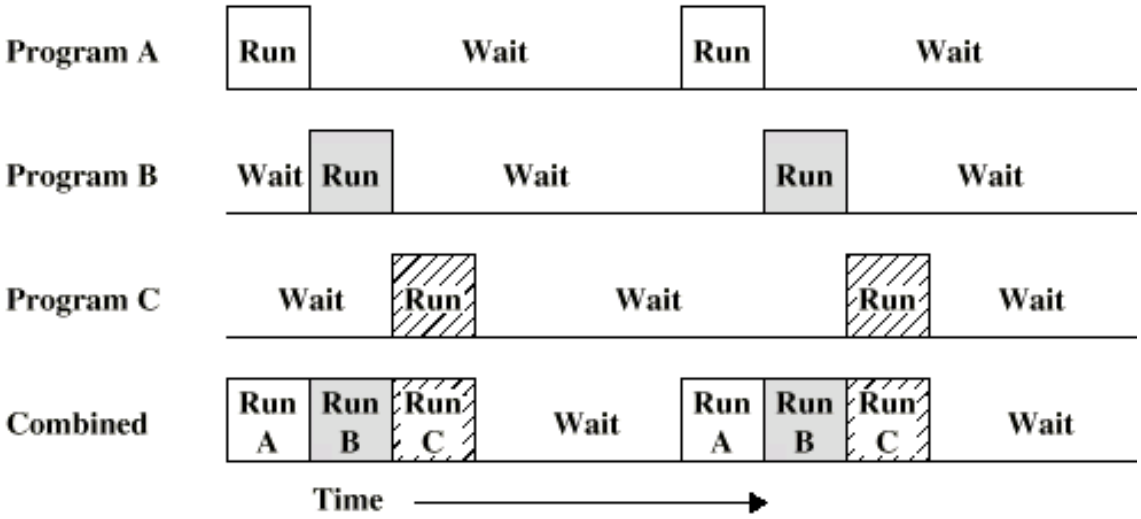
- 只有一个程序处于执行状态
- CPU大部分时间都处于等待状态
- CPU效率很低

Multi-Programming with two programs 2个程序



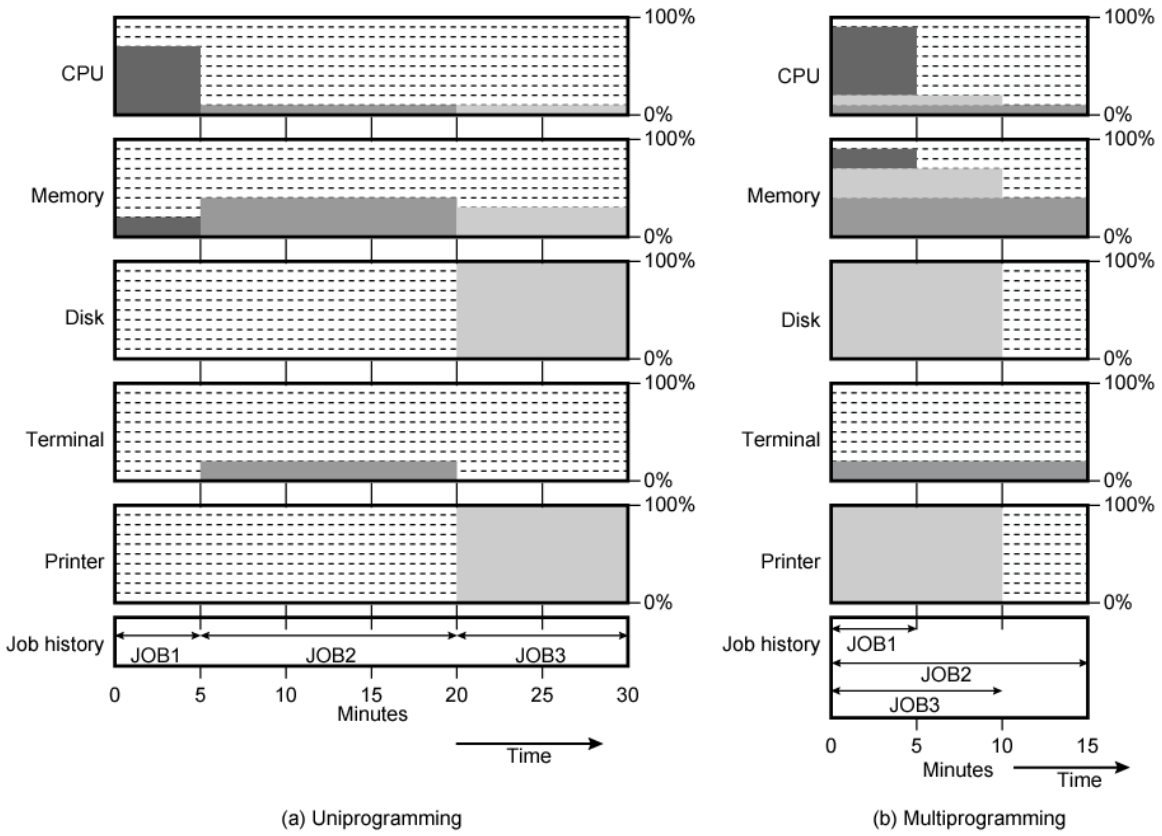
- 同时有两个程序处于执行状态
- CPU等待状态的时间少了一些
- CPU效率有一定的提高

Multi-Programming with three programs 3个程序



- 同时有三个程序处于执行状态
- CPU等待状态的时间更少
- CPU效率更高

Utilization 利用率



- 单个程序逐个执行，总共需要30个时间单位
- 多道程序同时执行，只需要15个时间单位
- 节省了一半时间

Time sharing systems 分时系统

- Multi-programmed batch system 多道程序批处理系统
  - Improve the efficiency of system operation 提高了系统效率
  - Disadvantage is that the response to users is not timely 缺点是对用户的响应不及时
- For multiple interactive job scenarios, a time-sharing system is proposed 对于多个交互式作业，提出了分时系统
  - Multiple users to share processor time and interact with the processor 多个用户共享处理器，和处理器交互
  - Operating system responds to each user's request in time slices computer 操作系统按照时间片来响应每个用户的请求
  - For each user, he thinks the processor only serves him 对于每个用户来说，他以为他是独享处理器

Summary

- 操作系统的目标
  - 为了满足提高资源利用率、增强计算机系统性能而开发的系统程序
- 操作系统的功能
  - 便利：屏蔽硬件细节，提供用户接口，使得计算机容易使用
  - 高效：进程调度，内存管理
- 操作系统分类
  - 是否支持批处理：交互式操作系统，批处理操作系统
  - 是否支持多道程序：单道程序，多道程序
  - 分时操作系统：按时间片响应用户需求

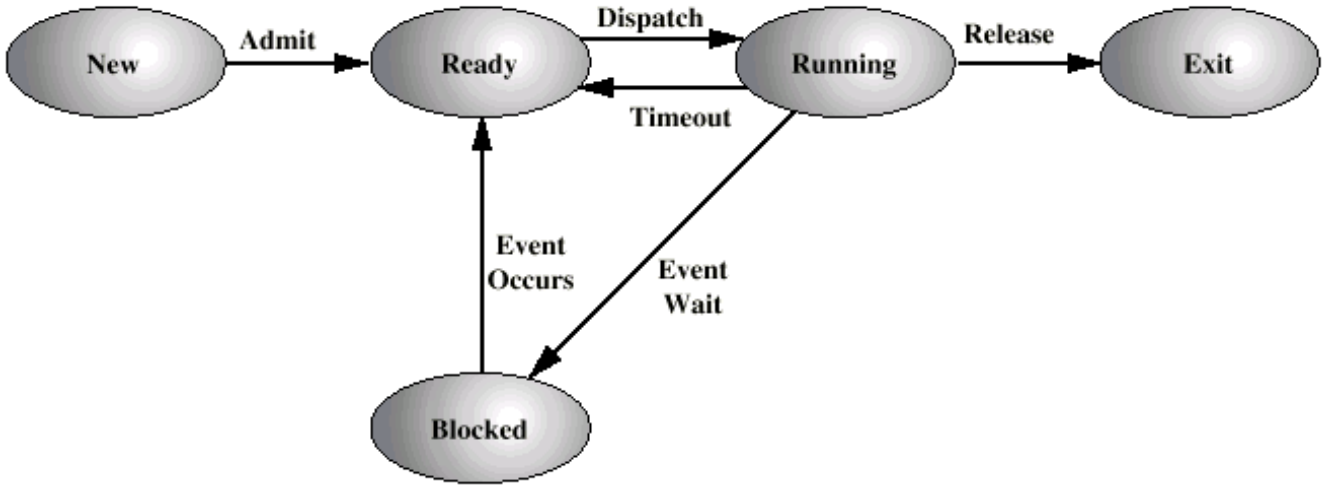
Outline

- Operating System Overview 操作系统概览
- Scheduling 调度
- Memory Management 内存管理

Scheduling 调度功能

- Key to multi-programming 是多程序的关键
- Determines the efficiency of multiprogramming 决定多道程序运行效率
- Includes long term, medium term, short term
  - Short term：decide which process the processor will execute 短期调度：决定哪个进程下一个处理
  - Medium term：determines the number of processes added to main memory 中期调度：决定哪些进程能加载到内存中
  - Long term：determines the number of processes added to the pending process pool 长期调度：决定哪些程序需要加到待处理的进程池

Five state process model 进程处理的五个状态模型



- 新建状态：调度程序提交一个程序，操作系统为这个程序创建一个进程，并将进程移入就绪状态
- 就绪状态：进程已经准备就绪，等待处理器的执行
- 运行状态：进程正在由处理器执行
- 等待状态：进程在等待资源，处于挂起状态
- 终止状态：进程运行结束

What is process? 什么是进程?

- A program in execution 处于执行中的程序
  - An instance of a program running on a computer 运行在计算机中的程序的一个实例
  - The entity that can be assigned to and executed on a processor 能够分派给处理并且执行的一个实体
- A unit of activity characterized by the execution of a sequence of instructions, a current state, and an associated set of system instructions 进程是一种活动单元，特征是一系列指令的执行、当前状态和一组相关的系统指令

Process elements 进程的要素

- A process is comprised of 进程由下面几个部分组成
  - Program code (possibly shared) 程序代码，可能是共享的
  - A set of data 一组数据
  - A number of attributes describing the state of the process 描述进程状态的若干属性
- Process elements 进程的元素
  - Identifier 标识符
  - State 状态
  - Priority 优先级
  - Program counter 程序计数器
  - Memory pointers 存储器指针
  - Context data 上下文数据
  - I/O status information I/O状态信息
  - Accounting information 统计信息

Process control block 进程控制块

- Contains the process elements 包含进程的元素
- Created and manage by the operating system 由操作系统创建并管理
- Allows support for multiple processes 提供了对多进程的支持

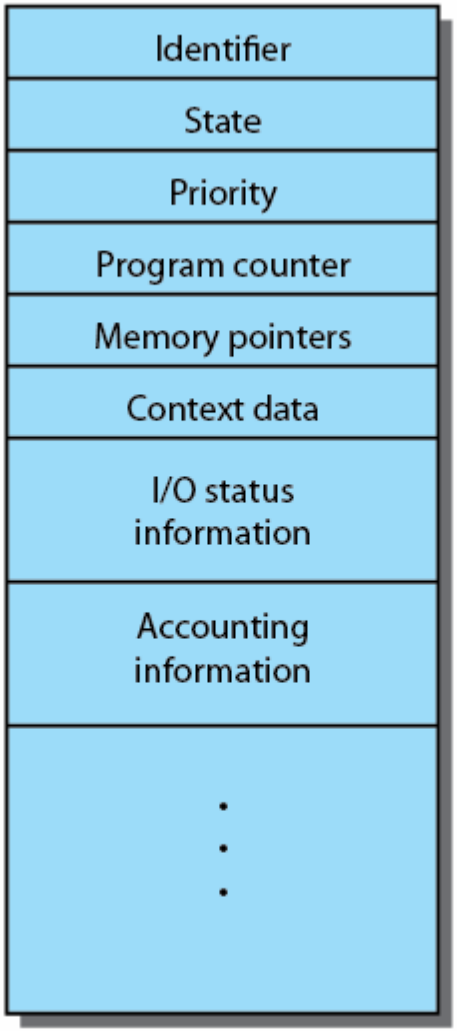
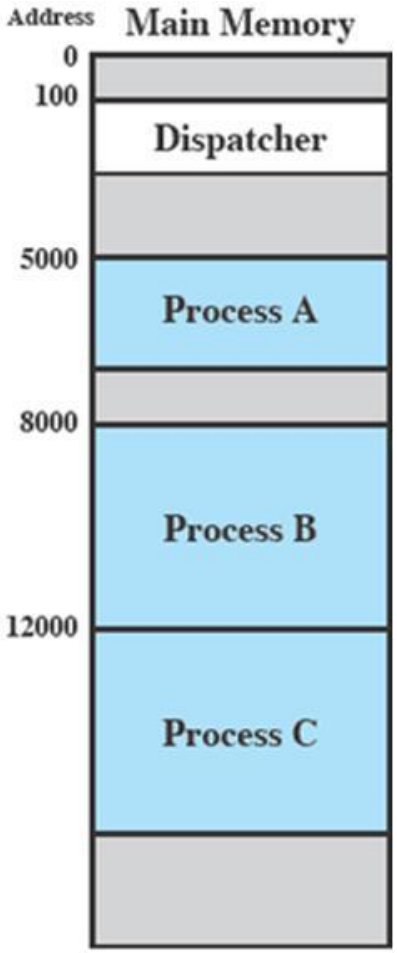


Figure Simplified Process Control Block

Trace of the process 进程跟踪

- The behavior of an individual process is shown by listing the sequence of instructions that are executed 一个进程的行为可以由该进程执行的一系列指令来表示
  - This list is called a Trace 这个指令执行的序列称为跟踪
- Processor needs to execute multiple processes one by one 处理器需要逐个执行多个进程
  - Dispatcher is responsible for scheduling the process 进程的调度由调度器来完成
  - Dispatcher is a small program which switches the processor from one process to another 调度器是一个小程序，它负责将处理器在不同进程之间进行切换

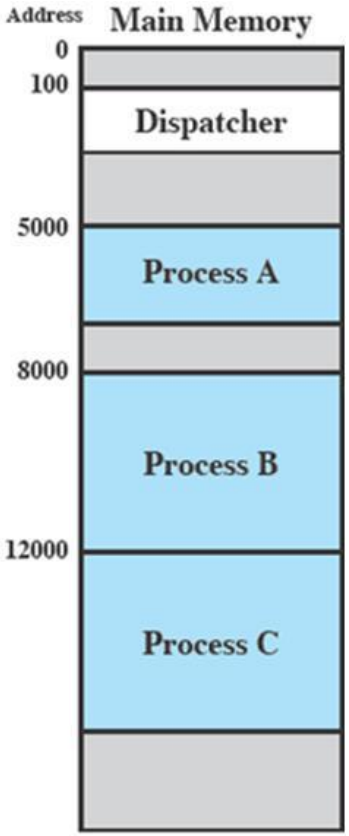
Example of process execution



(a) Trace of Process A	(b) Trace of Process B	(c) Trace of Process C
5000	8000	12000
5001	8001	12001
5002	8002	12002
5003	8003	12003
5004		12004
5005		12005
5006		12006
5007		12007
5008		12008
5009		12009
5010		12010
5011		12011

- 考虑三个进程在执行
- 调度器和进程都在内存中
- 不考虑虚拟内存
- 每个进程运行到结束

Trace from processors points 从处理的角度



1	5000	
2	5001	
3	5002	
4	5003	
5	5004	
6	5005	
---------- Timeout		
7	100	
8	101	
9	102	
10	103	
11	104	
12	105	
13	8000	
14	8001	
15	8002	
16	8003	
---------- I/O Request		
17	100	
18	101	
19	102	
20	103	
21	104	
22	105	
23	12000	
24	12001	
25	12002	
26	12003	
27	12004	
28	12005	
---------- Timeout		
29	100	
30	101	
31	102	
32	103	
33	104	
34	105	
35	5006	
36	5007	
37	5008	
38	5009	
39	5010	
40	5011	
---------- Timeout		
41	100	
42	101	
43	102	
44	103	
45	104	
46	105	
47	12006	
48	12007	
49	12008	
50	12009	
51	12010	
52	12011	
---------- Timeout		

100 = Starting address of dispatcher program

Shaded areas indicate execution of dispatcher process;
first and third columns count instruction cycles;
second and fourth columns show address of instruction being executed

Figure 3.4 Combined Trace of Processes of Figure 3.2

- 处理器开始执行进程A，执行到5005的时候，到时间了，于是切换到调度器
- 调度器确定下面执行进程B，执行到8003，需要请求IO，于是挂起，回到调度器
- 调度器指示执行进程C。进程C执行到12005，到时间了，回到调度器
- 继续执行进程A到5011，进程A执行完毕。然后继续执行进程C

Example of execution

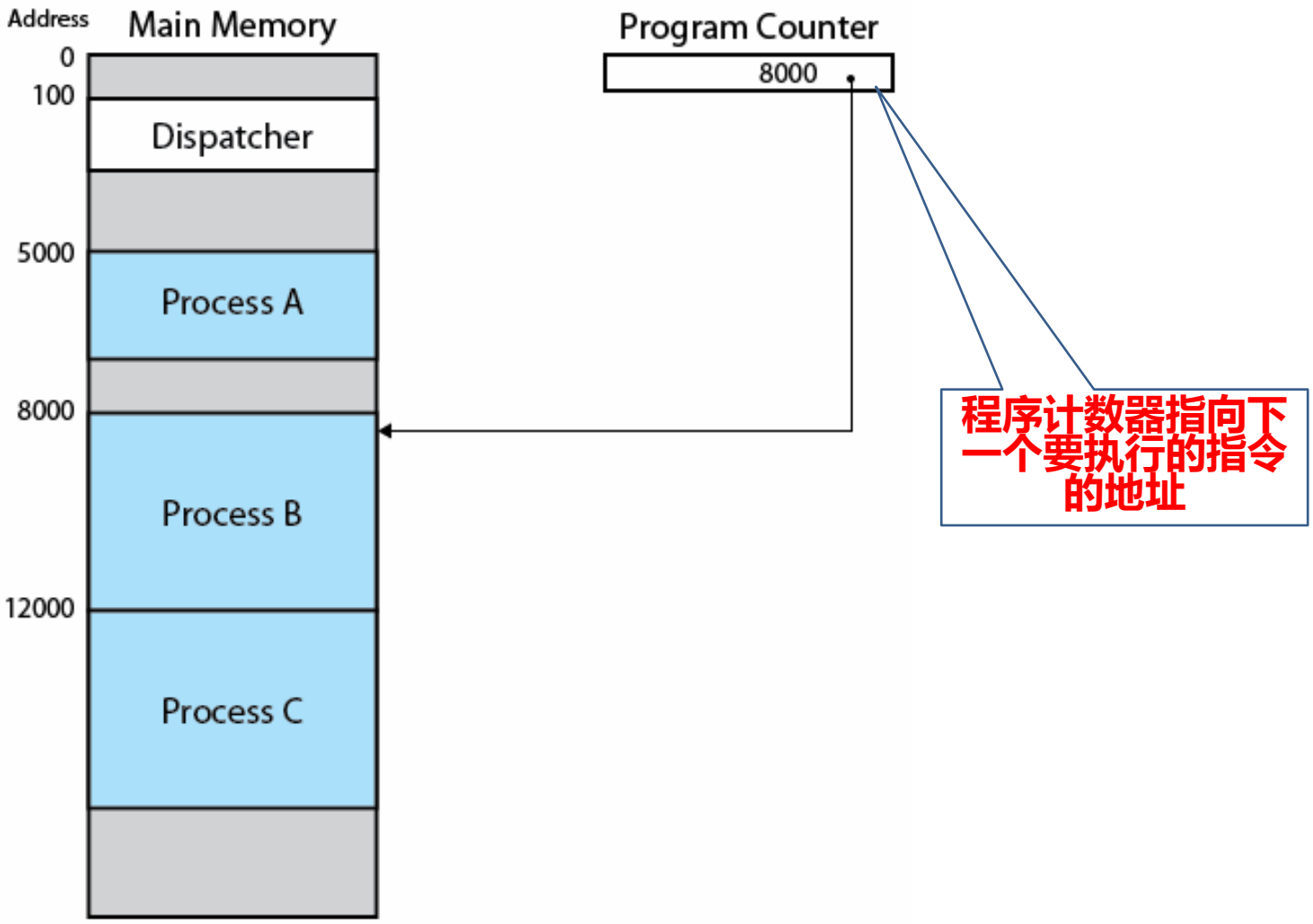
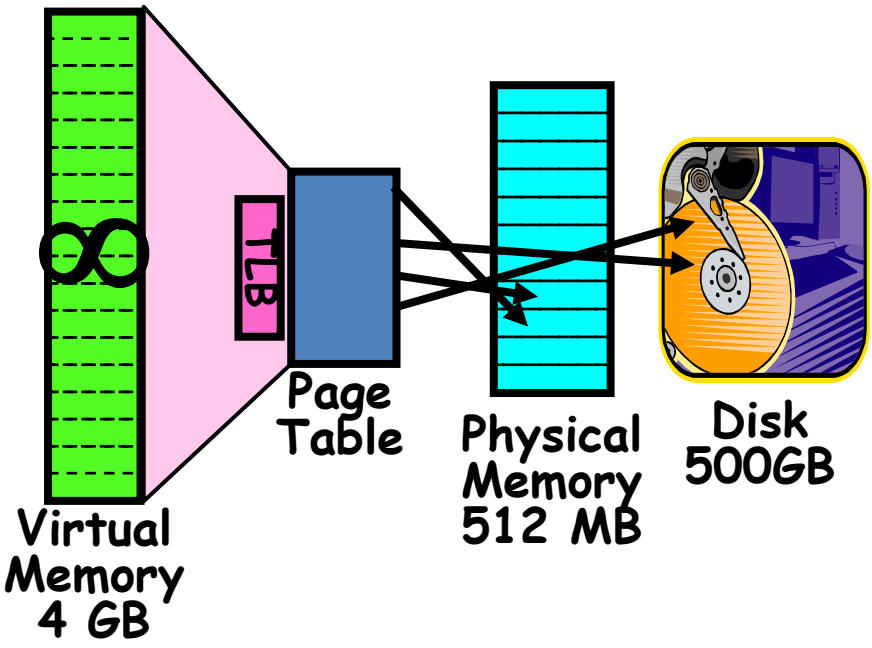


Figure Snapshot of Example Execution

Virtual memory 虚拟内存



- 虚拟存储器是内存管理的一种技术，允许程序从逻辑的角度对内存进行寻址，而不考虑物理上可用的内存空间
- 本质上是一种内存扩充技术，不会使内存的物理空间大小发生改变，但是能够使程序访问内存的方式更灵活
- 由于虚拟内存的存在，所以程序中可以只将其中的一部分加载到物理内存中。只有当需要的程序段不在内存中的时候，才通过换页的方式将它加载到内存中

Execution of a process -1 进程执行1

- Entire program does not need to be loaded to memory 整个程序不需要都加载到内存中
  - Operating system brings into main memory a few pieces of the program 操作系统将程序的一些片段加载到内存中
  - Resident set - portion of process that is in main memory 驻留部分保留在内存中
- An interrupt is generated when an address is needed that is not in main memory 当需要的地址不在内存中，产生一个中断
  - Operating system places the process in a blocking state 操作系统设置进程为阻塞状态

Execution of a process -2 进程执行2

- Piece of process that contains the logical address is brought into main memory 包含逻辑地址的进程片段加载到内存中
  - Operating system issues a disk I/O Read request 操作系统发起磁盘读操作请求
  - Another process is dispatched to run while the disk I/O takes place 另一个进程来完成磁盘I/O
  - An interrupt is issued when disk I/O complete which causes the operating system to place the affected process in the Ready state 当磁盘I/O完成后，会发出中断。操作系统根据中断信号，将受影响的进程置于就绪状态

Implications of this new strategy 新策略的影响

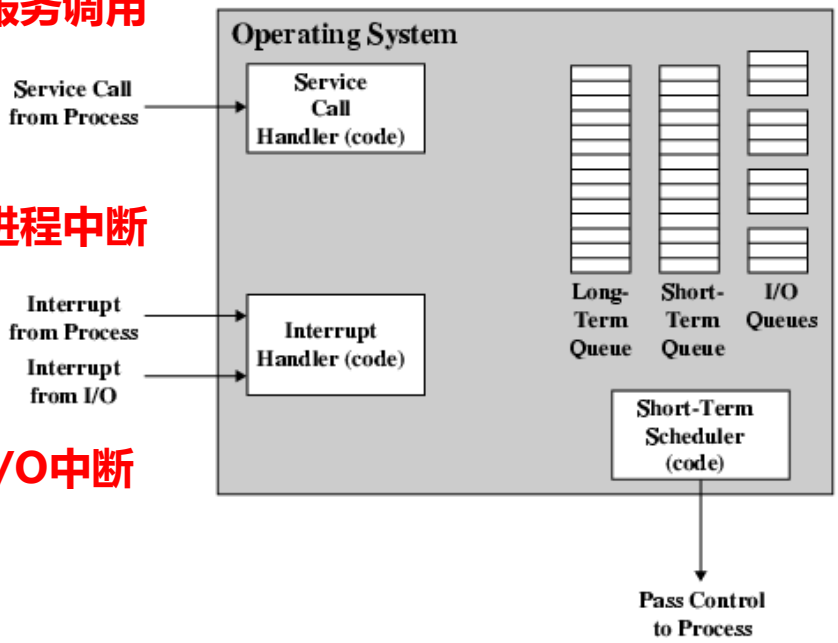
- More processes may be maintained in main memory 内存中可以维护更多的进程
  - Only load in some of the pieces of each process 每个进程只需要加载一些片段
  - With so many processes in main memory, it is very likely a process will be in the Ready state at any particular time 由于有多个进程在主存中，所以在任意时刻都很可能会有进程处于就绪状态
- A process may be larger than all of main memory 一个进程本身就可能比整个物理内存要大

Wording mode of OS 操作系统工作方式

服务调用

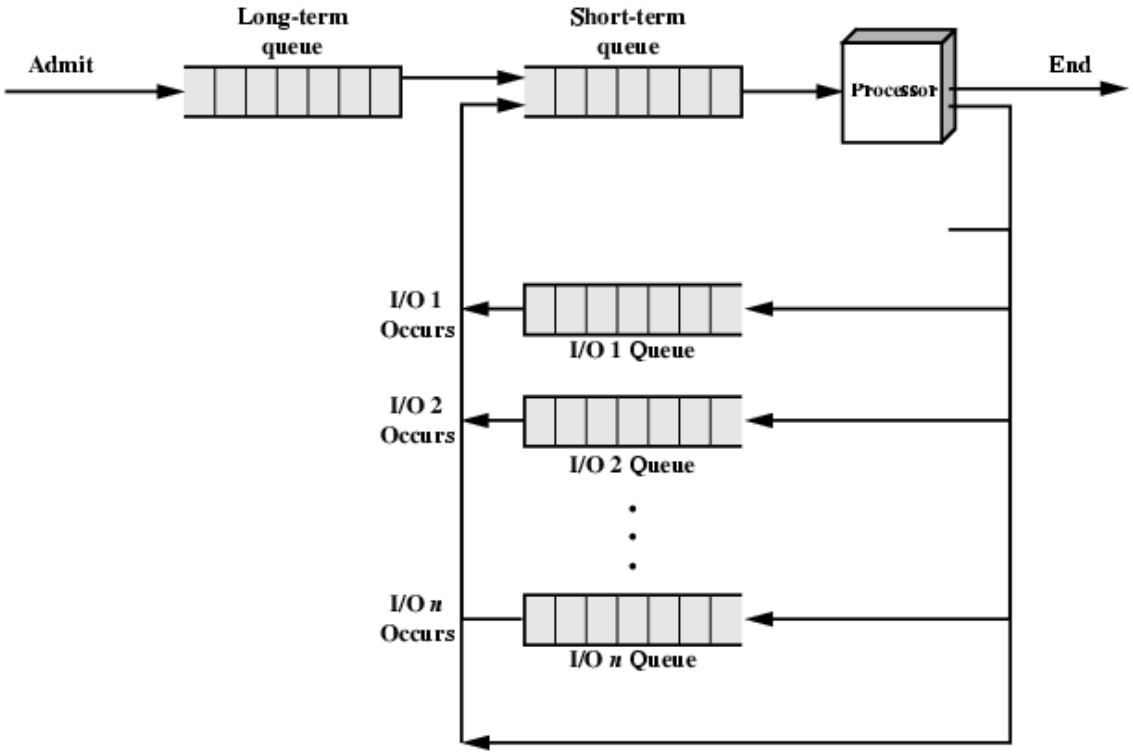
进程中断

I/O中断



- 操作系统中维护一个长期队列和一个短期队列
- 长期队列是等待系统资源的作业列表，短期队列包含所有就绪状态的进程
- 中断发生时，操作系统接管处理器的控制权，执行中断处理程序
- 进程调用服务时，操作系统执行服务调用处理程序
- 中断或服务调用程序处理完成后，短期调度会调度某个进程进入到运行状态
- 每个I/O设备有一个I/O队列

Process scheduling 进程调度



- 进程请求进入长期队列
- 进程的请求满足后，进程进入就绪状态并进入短期队列
- 如果进程请求I/O，则进入I/O队列
- I/O完成后，继续进入短期队列，完成进程的执行

Outline

- Operating System Overview
- Scheduling
- Memory Management

Memory management 内存管理

- Uni-program 单一程序
  - Memory split into two 内存分为两个部分
  - One for Operating System (monitor) 一个给操作系统，监控程序
  - One for currently executing program 一个给当前执行的用户程序
- Multi-program 多个程序
  - “User” part is sub-divided and shared among active processes 用户程序部分由多个活跃进程共享
- Memory management 存储器管理
  - Make full use of the memory space 充分利用存储器空间
  - keep the CPU busy 保持CPU忙
  - avoid idle CPU due to IO waiting 避免CPU因为等待导致空闲

Methods of memory management 存储器管理方法

- Swapping 交换
- Partitioning 分区
- Paging 分页
- Virtual Memory 虚拟存储器
- Translation Lookaside Buffer 快表
- Segmentation 分段

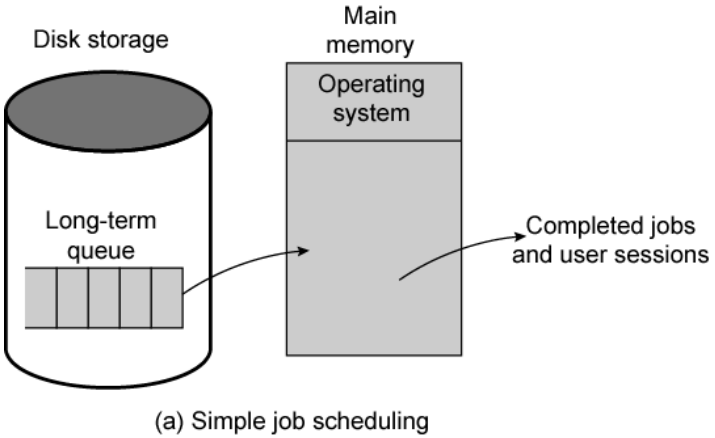
Swapping 交换

- Problem:
  - I/O is so slow compared with CPU I/O相对于CPU太慢
  - Even in multi-programming system，CPU is idle most of the time 即使在多道程序系统中，CPU在大多数时候空闲
- Solutions 解决方法
  - Increase main memory，more process in memory 增加内存，内存中有更多的进程
    - Expensive 贵
    - Leads to larger programs 导致更大的程序
  - Swapping 交换

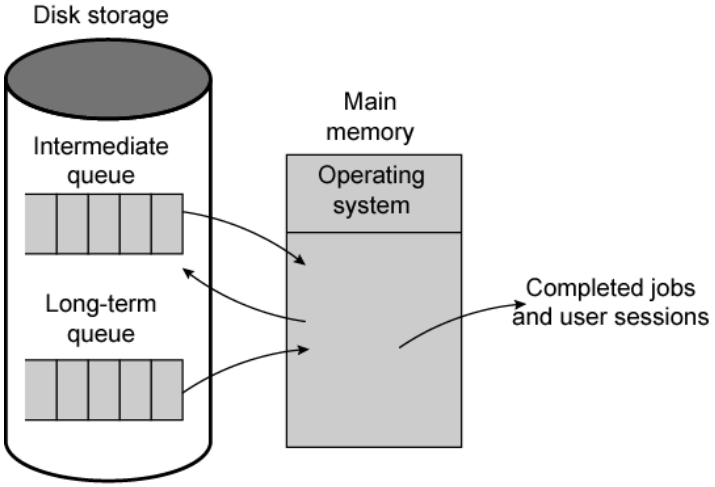
What is swapping? 什么是交换?

- Long term queue of processes stored on disk 长周期队列中的进程保存在磁盘
  - Processes "swapped" in as space becomes available 当内存有空的时候，把进程交换进去
  - As a process completes it is moved out of main memory 进程处理完之后，移出内存
- If none of the processes in memory are ready (i.e. all I/O blocked) 如果因为I/O阻塞等原因，导致所有进程都处于非就绪状态
  - Swap out a blocked process to intermediate queue 把阻塞的进程交换到中间队列
  - Swap in a ready process or a new process 交换就绪进程，或者新进程
  - Swapping is an I/O process 交换本身也是一个I/O进程

Use of swapping 交换的使用



(a) Simple job scheduling



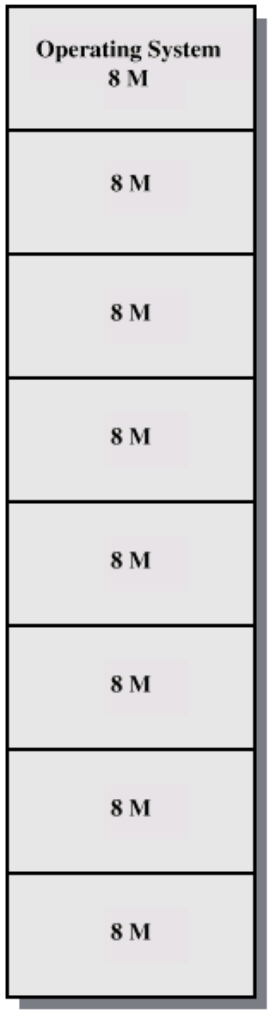
(b) Swapping

- 简单的进程调度中，磁盘维护一个长周期队列。当内存中有作业完成了，会将它从内存中去掉，同时将长周期队列中的一个作业放到内存中
- 交换调度中，磁盘中维护了一个中间队列。阻塞进程将会被“交换”到中间队列中，并从中间队列中调入一个已经就绪的进程。如果中间队列没有就绪的进程，就从长期队列中调度一个新的进程到内存中。
- 通过中间队列，可以在进程均处于阻塞的时候进行进程的有效调度，提高CPU效率

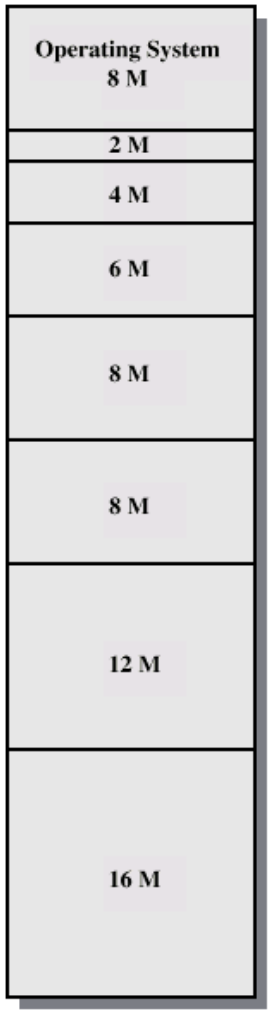
Partitioning 分区

- Swapping does not explain how memory is managed 交换没有说明内存如何管理
- Memory management mode
  - Partitioning 分区
  - Paging 分页
  - Segmentation 分段
- Partitioning：Splitting memory into sections to allocate to processes (including Operating System) 分区：将内存分为几个部分，分配给包括操作系统在内的进程

Fixed partitioning 固定大小分区



(a) Equal-size partitions



(b) Unequal-size partitions

- 分区大小固定，但不一定都是一样大
- 进程分配内存时，分配到能容纳它的最小分区
- 存在内存浪费的现象
- 采取非固定大小分区可能会更好

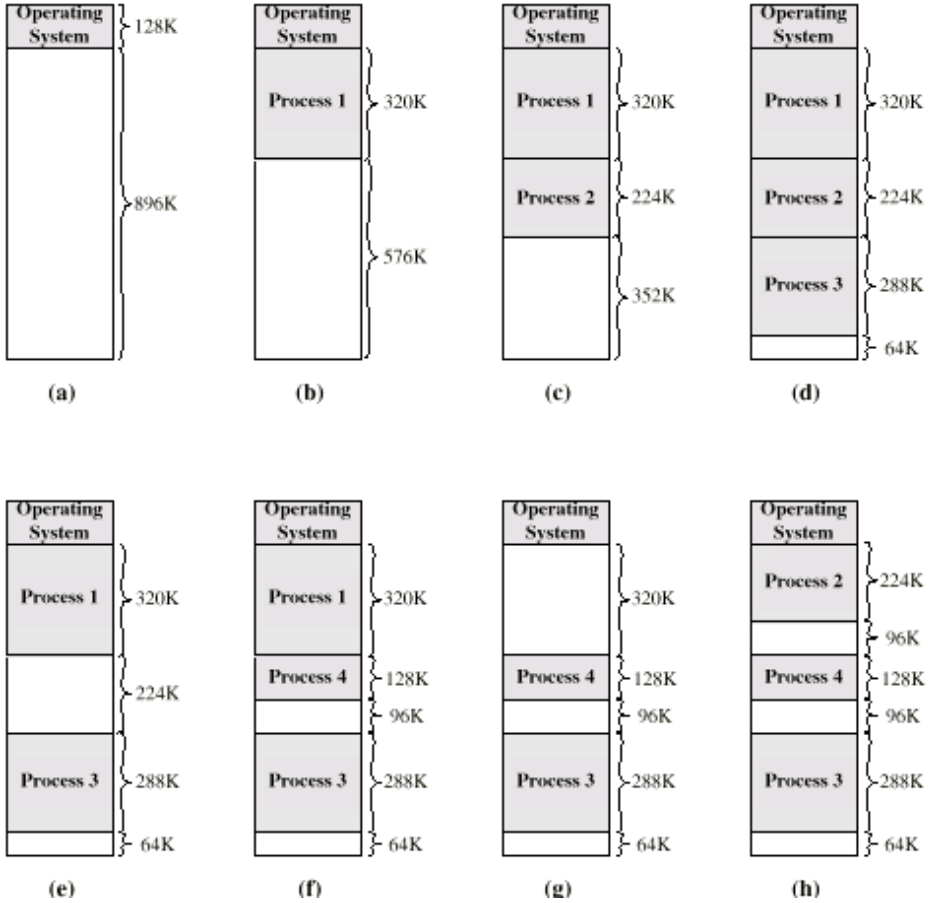
Variable sized partitions -1 可变分区1

- Allocate exactly the required memory to a process 按进程的要求分配内存
- This leads to a hole at the end of memory, too small to use 在内存的最后会留下一个空块，太小不能使用
  - Only one small hole - less waste 只有一个空块，浪费不大
- When all processes are blocked, swap out a process and bring in another process，leads to another hole 当所有进程都阻塞了，进程交换出去，会产生另一个空块
- A process is completed and swapped out，new process is swapped in. May leads to a hole. 当进程执行完交换出去后，新的进程进来了，可能会产生另一个空块

Variable sized partitions -2 可变分区2

- Eventually have lots of holes (called fragmentation) 到最后，内存里有大量的空块，称为碎片
- Solutions 解决方法
  - Coalesce - Join adjacent holes into one large hole 合并法：将相邻的空块合并成一个大的空块
  - Compaction- From time to time go through memory and move all hole into one free block (c.f. disk de-fragmentation) 紧缩法：检索内存，将所有的空块移动形成一个大的空块。类似磁盘的去碎片化

Effect of Dynamic Partitioning 动态分区的效果



- 分配进程1~3的时候，按需求分配，剩下64k的空间
- 进程2交换出去了，进程4进来了。进程4只有128k，在进程4和进程3之间，形成了一个96k的空块
- 进程1交换出去了，进程2进来。进程1有320k，进程2只有224k，在进程2和进程4之间，又形成了一个96k的空块
- 有2个96k空块和1个64k空块

Problem about relocation 重载问题

- No guarantee that process will load into the same place in memory 没有机制保证进程重新加载到内存的同一个位置
- Instructions contain addresses information 指令包括地址信息
  - Locations of data 数据位置
  - Addresses for instructions (e.g. branching) 指令地址（分支）
- If a fixed address is used, memory address of process changes after the process is swapped out and then swapped in 如果采用固定地址的话，那么进程被交换出去后，再交换进来，进程的内存地址发生了变化
- Data and instructions cannot be addressed 数据和指令无法寻址

Problem about relocation 重载问题

- Using logical address and physical address 逻辑地址&物理地址
  - Logical address - relative to beginning of program 逻辑地址-程序开始的相对位置
  - Physical address - actual location in memory (this time) 物理地址-在内存中当前的实际位置
- Starting unit address of the current process is set as the base address 当前进程的起始单元地址设置为基址
- Addressing data and instructions using logical addresses in programs 程序中使用逻辑地址进行数据和指令的寻址
- Physical address in memory can be obtained through the base address and logical address 通过基址和逻辑地址，就可以得到在内存中的物理地址

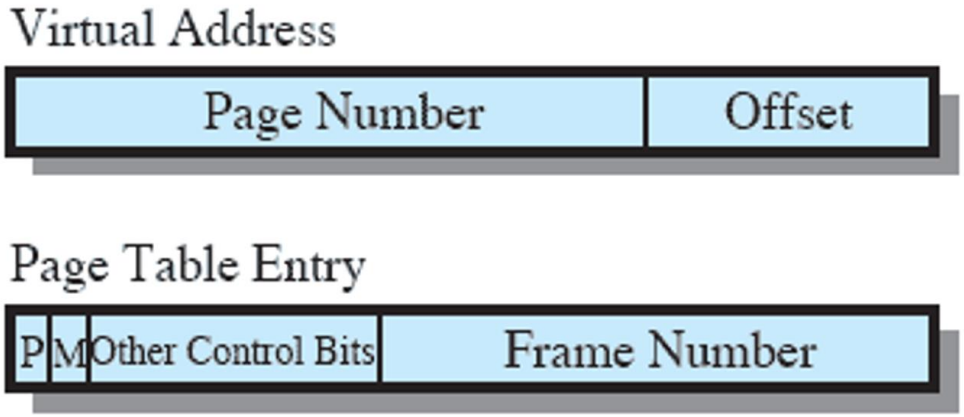
Paging 分页

- Partitioning will lead to waste of memory 分区会导致内存浪费
- Paging Management
  - Split memory into equal sized, small chunks -page frames 将内存分为小的，固定长度的块-页帧
  - Split programs (processes) into equal sized small chunks – pages 将进程分为固定大小的小块-页
  - Allocate the required number page frames to a process 给进程分配它需要的页帧数量
- Operating System completes paging management 操作系统进行分页管理
  - A process does not require contiguous page frames 进程不一定需要要求连续的页帧
  - Use page table to keep track 使用页表来跟踪

Using of Page 页的使用

- Each process has its own page table 每个进程有它自己的页表
- Each page table entry contains the frame number of the corresponding page in main memory 每个页表包含页在内存中对应的帧号
- Two extra bits are needed to indicate 两个额外的比特
  - whether the page is in main memory or not 页是否在内存中
  - Whether the contents of the page has been altered since it was last loaded 页的内容在最后一次加载之后，是否被修改过

Paging table 页表



- 寻址的虚拟地址为“页号+偏移量”
- 页表中指明了这个页对应的内存的帧序号
- 通过“页——》帧”的转换，以及页内的偏移量，就可以得到逻辑地址在内存中的实际地址

Address translation 地址翻译

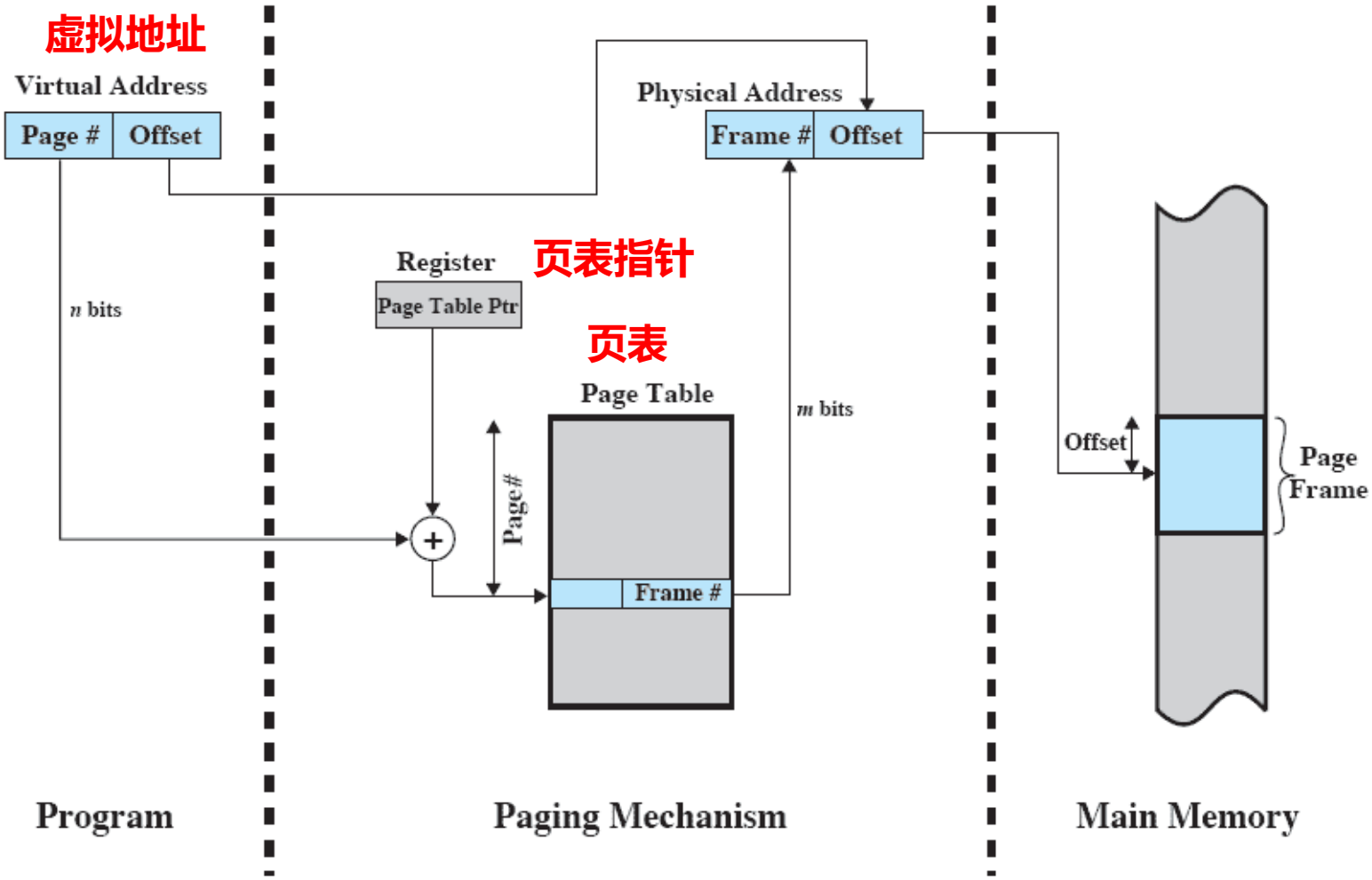
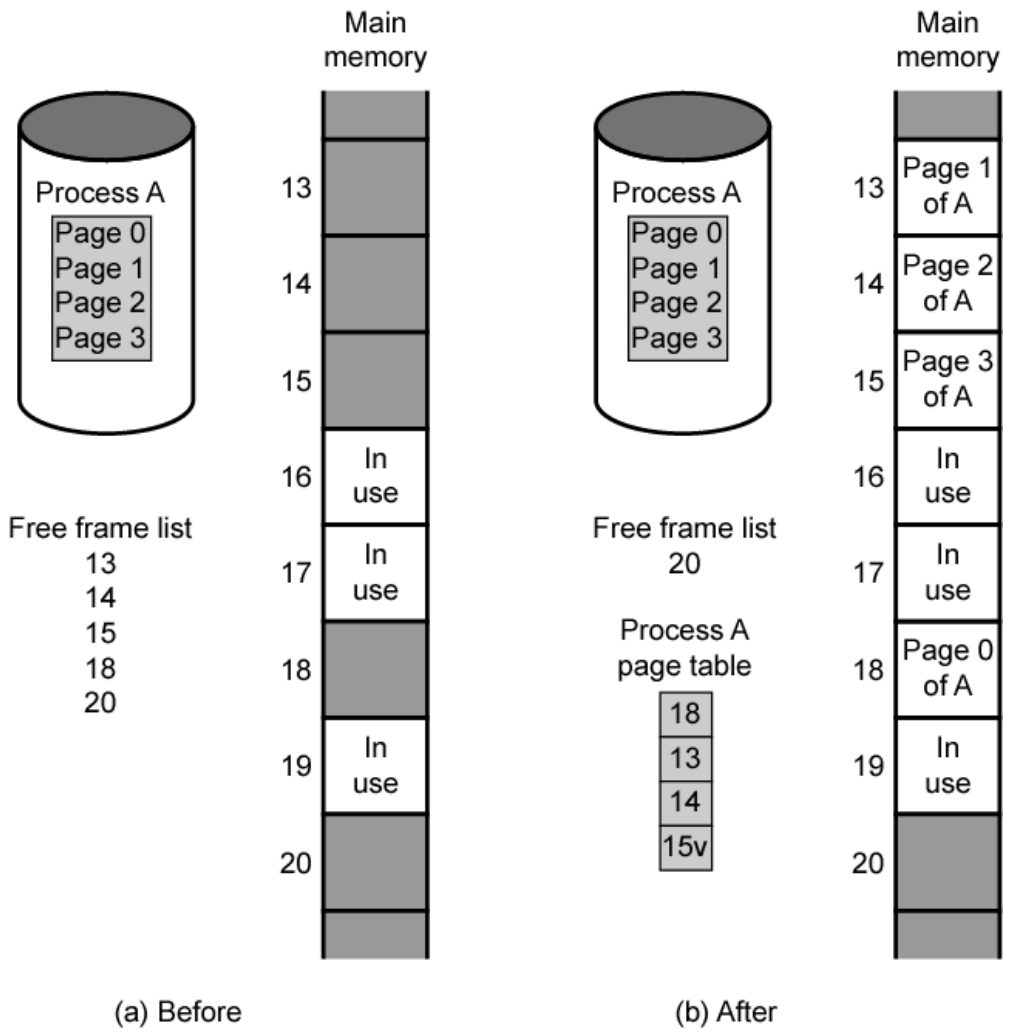


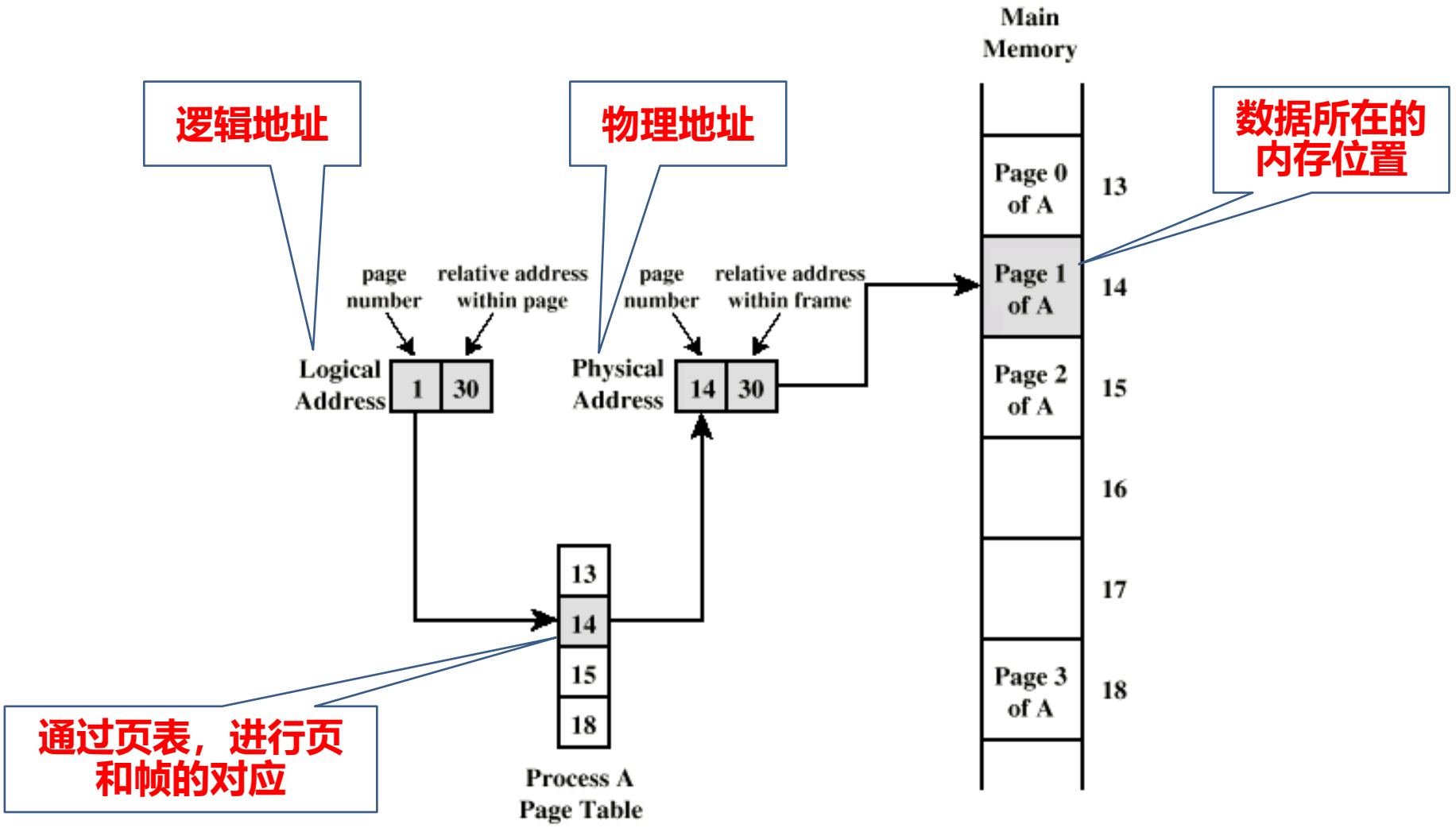
Figure 8.3 Address Translation in a Paging System

Allocation of free frames 空闲帧的分配



- 内存中帧13~15是空闲的，分配给了进程A地第1~3页
- 帧16、17其他进程占用
- 帧18分配给进程A的0页
- 分配帧的时候，既不要求是连续的，也不要求前后顺序

Logical and Physical Addresses 逻辑和物理地址



Real and virtual memory 实际存储和虚拟存储

- On the basis of logical address, expand the memory, which leads to the concept of virtual memory 在逻辑地址的基础上，进行存储器的扩充，这就引出了虚拟存储器的概念
- Real memory 真实存储器
  - Main memory, the actual RAM 内存，实际的RAM
- Virtual memory 虚拟存储器
  - Memory on disk 磁盘上的存储
- Allows for effective multiprogramming and relieves the user of tight constraints of main memory 允许更有效的多道程序，减轻用户受到存储器的约束

Support deeded for virtual memory 虚拟存储的要求

- Hardware must support paging and segmentation 要求硬件支持分页和分段
- Operating system must be able to manage the movement of pages and/or segments between secondary memory and main memory 操作系统能够管理页或者段在辅存（硬盘）和主存之间的移动，以进行换出和换入

Virtual memory

- Page load-demand paging 页加载方式-请求分页
  - Do not require all pages of a process in memory 进程的所有页不一定都需要在内存中
  - Bring in pages as required 根据需要加载页到内存
- Page fault 页失效
  - Required page is not in memory 需要的页不在内存中
  - Operating System must swap in required page 操作系统需要将需要的页交换进来
  - May need to swap out a page to make space 可能需要将页交换出去
  - Select page to throw out based on recent history 基于历史交换页

Thrashing 抖动

- Too many processes in too little memory 小内存中有太多的进程
- Operating System spends all its time swapping 操作系统花了大量的时间进行交换
- Little or no real work is done 几乎没有实际的工作在做
- Disk light is on all the time 磁盘一直处于工作状态
- Solutions 解决方案
  - Good page replacement algorithms 好的页替换策略
  - Reduce number of processes running 减少在运行的进程数量
  - Fit more memory 配置更多的内存

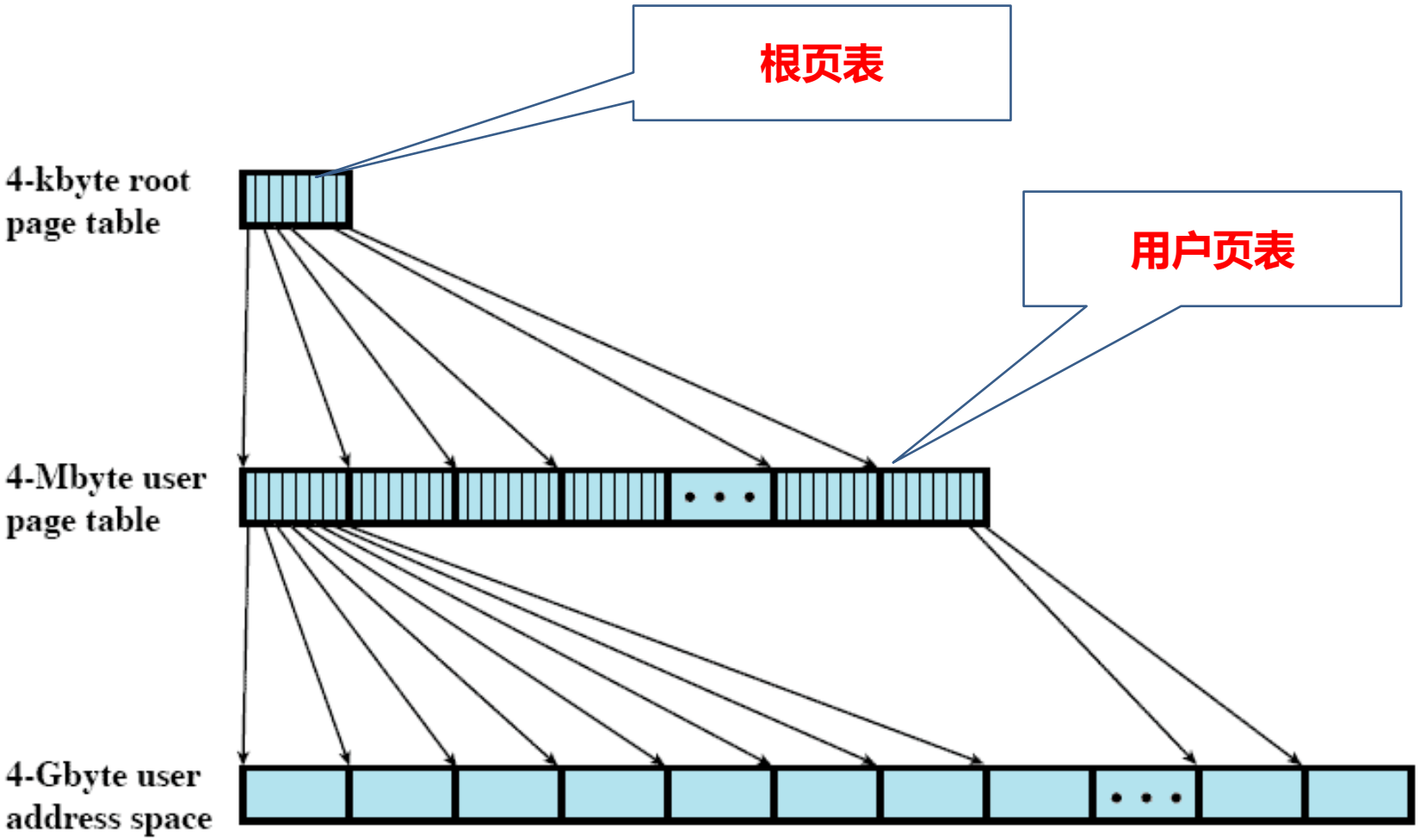
Advantage of virtual memory

- Do not need all of a process in memory for it to run 不需要将进程全部加载到内存中
- Swap in pages as required 能够根据需要交换页
- So can now run processes that are bigger than total memory available! 能够运行的进程大于实际可用的内存
- Main memory is called real memory 主存称为实际内存
- User/programmer sees much bigger memory - virtual memory 用户程序能够看到更多的内存-虚拟内存

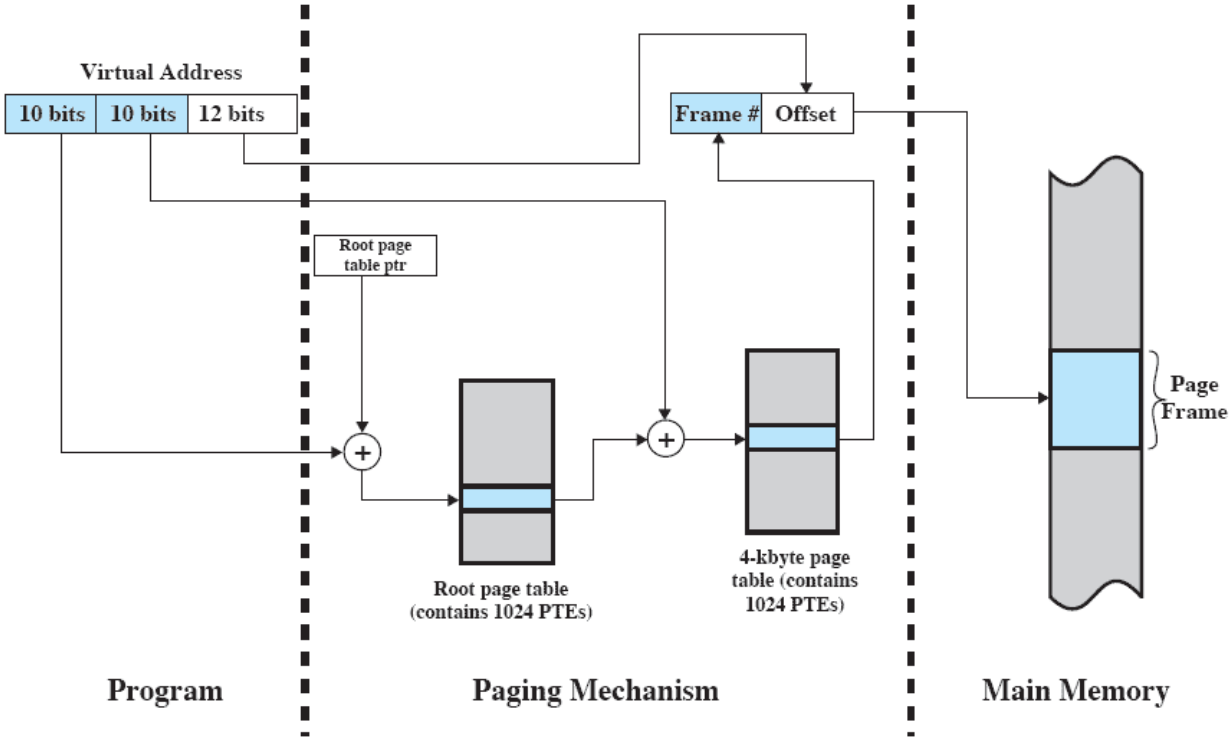
Page tables 页表

- Each process must have a page table to convert logical address or virtual address to physical address 每个进程都要有一个页表，用来进行逻辑地址或虚拟地址和物理地址的转换
- Some processes are very large, and their page tables themselves are very large 有些进程非常大，它的页表本身就非常大
  - Page tables are also stored in virtual memory 页表也是存储在虚拟存储器中
  - When a process is running, part of its page table is in main memory 进程运行时，部分页表在实际存储器中
  - Load the required page table through exchange 通过交换的方式来载入需要的页表

Two-Level Hierarchical Page Table 两级页表结构



Address Translation for Hierarchical page table
多级页表的地址翻译



- 虚拟地址包括根页表段、用户页表段和页内偏移量
- 根据根页表指针和根页表段，得到用户页表段的基准地址
- 加上用户页表段地址，得到用户页表地址
- 再加上页内偏移量，得到在实际内存中的地址

Problem about page tables 页表大小问题

- A drawback of the type of page tables is that their size is proportional to that of the virtual address space. 页表的缺点是页表大小和虚拟地址空间的大小成比例
  - Each virtual space page needs a row in the page table 每个虚拟空间的页都需要在页表中有一行
  - If the virtual space of the user program is large, the page table will also be very large 如果用户程序的虚拟空间很大，那么页表也会非常庞大
- An alternative is Inverted Page Tables 解决办法是反向页表

Inverted page table 反向页表

- Used on PowerPC, UltraSPARC, and IA-64 architecture 在PowerPC， UltraSPARC和IA-64 架构中使用
- Page number portion of a virtual address is mapped into a hash value 进程中的虚拟地址的页号部分通过哈希函数生成一个哈希值
- Hash value points to inverted page table 哈希值指向反向页表
- Fixed proportion of real memory is required for the tables regardless of the number of processes 反向页表的行数是物理内存的帧数，和进程数无关

Inverted page table 反向页表

Each entry in the page table includes 页表中的每一项包括

- Page number 页号
- Process identifier 进程ID
  - The process that owns this page 拥有这个页的进程ID
- Control bits 控制位
  - includes flags, such as valid, referenced, etc 包括标志位，比如有效，引用等
- Chain pointer 链式指针
  - the index value of the next entry in the chain 链中下一个条目的索引值

Use of inverted page table 反向表的使用

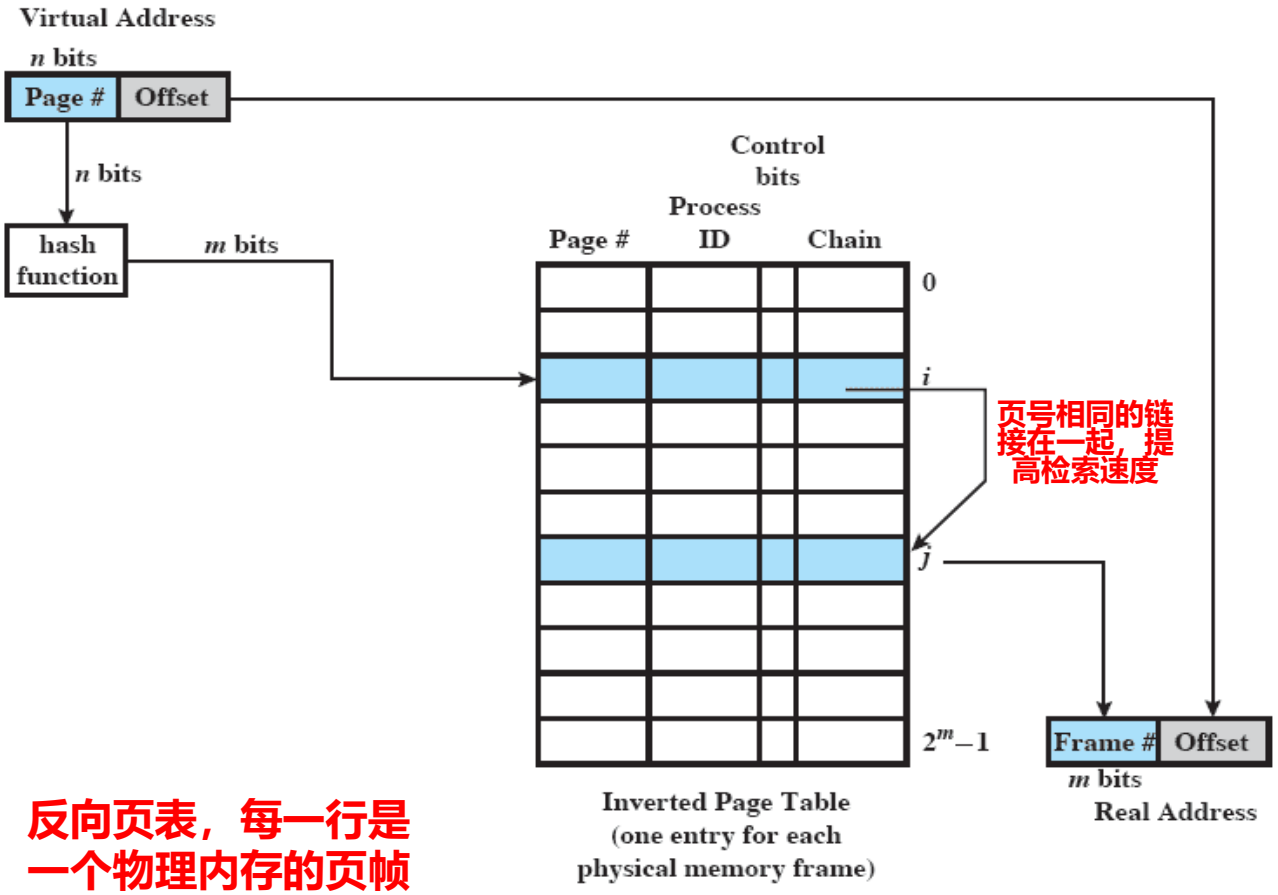


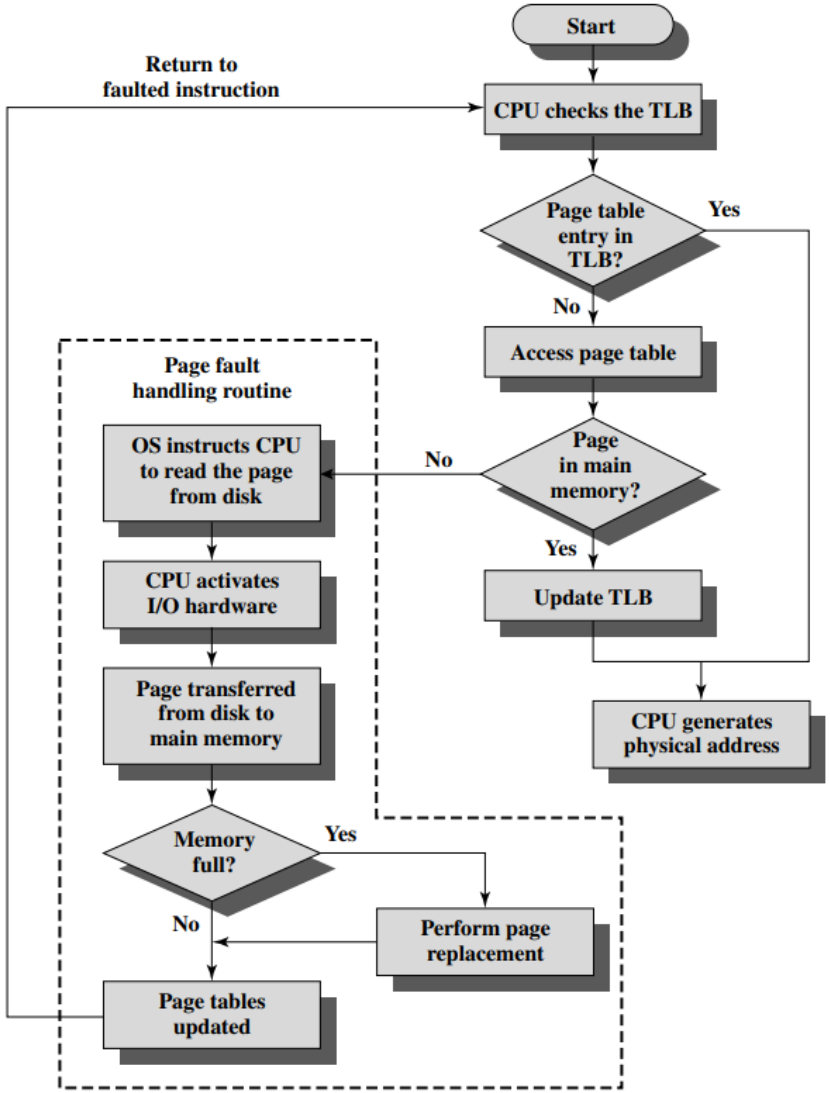
Figure 8.6 Inverted Page Table Structure

- 虚拟地址中的页号通过散列函数，形成一个散列值，映射到反向页表
- 通过页号的散列值和进程ID，确定虚拟地址的页是否在内存中
- 通过链技术来提高查找的速度

Translation lookaside buffer 快表

- Every virtual memory reference causes two physical memory access 每次虚拟地址引用需要两次物理内存访问
  - Fetch page table entry 获取页表
  - Fetch data 获取数据
- To overcome this problem a high-speed cache is set up for page table entries 为了克服这个问题，高速cache来作为页表
  - Called a Translation Lookaside Buffer (TLB) 称为快表
  - Contains page table entries that have been most recently used 包括最近访问的页表部分

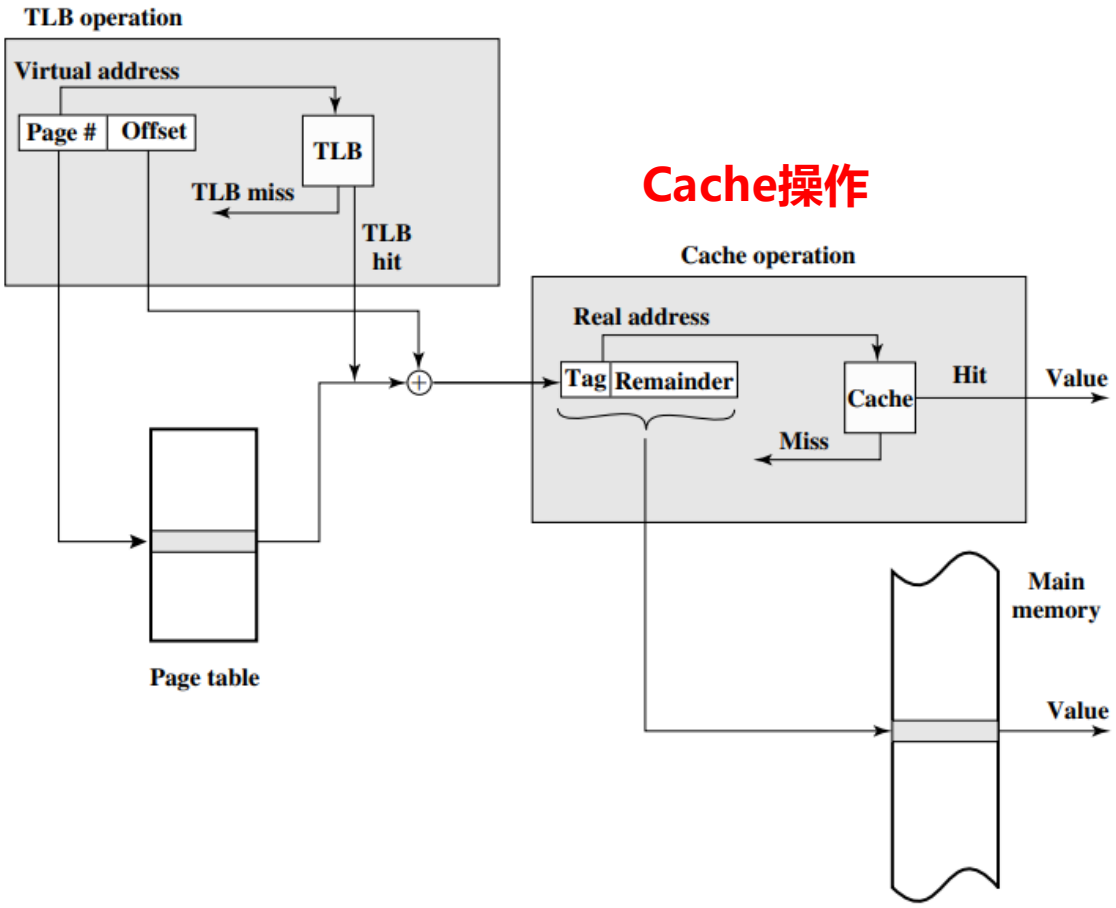
TLB operation 页表的操作



- 根据虚拟地址中的页地址，去快表中查找是否有页帧地址
- 如果有，直接就可以生成物理地址
- 如果没有，就去看内存中的页表中是否有这个页
- 如果在内存中的话，就先更新快表，然后生成物理地址
- 如果也没有，就要去读取I/O，然后更新内存页表

TLB and Cache operation 快表和cache的协调操作

快表操作



- 通过TLB得到物理地址，然后再到cache中看这个地址的内容是否在cache中，如果不在，就需要从主存中查询这个字
- 快表和Cache，一个是为了解决逻辑地址和物理地址的快速转换，一个是解决数据的快速存取

Segmentation 分段

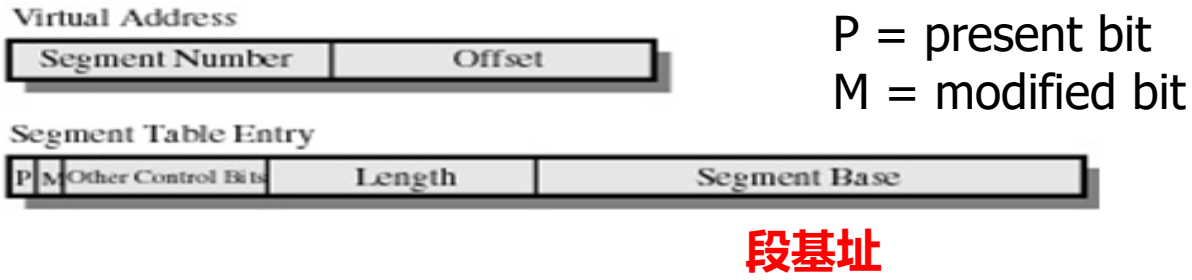
- Paging is not (usually) visible to the programmer 分页对程序员不可见
- Segmentation is visible to the programmer 分段对程序员可见
- Usually different segments allocated to program and data 通常对程序和数据分配不同的段
- May be a number of program and data segments 可能会有多个程序段和数据段

Advantage of segmentation 分段的好处

- Simplified management 简化管理
  - Programmers need not know how much space data needs 程序员可以不知道数据需要占用多大的空间
  - Data can be allocated to its own segment and can be dynamically expanded 数据可以分配自己的段，并且操作系统可以动态地扩充数据所在的段
- Allows programs to be altered and recompiled independently 允许程序独立修改和编译
- The segment can be shared among multiple processes 多个进程可以共享段
- Protections from assigning access privileges 给段赋予访问权，可以实现段保护

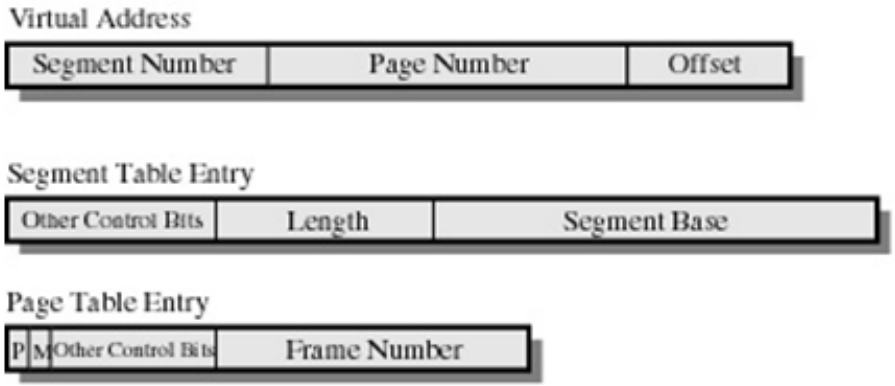
Segment tables 段表

- Corresponding segment in main memory 主存中的对应段



- 段需要有段表来实现地址转换
- 虚拟地址中包含段号，从段表中找到改段号对应的段基址，就可以根据偏移量到内存中得到这个虚拟地址对应的物理地址

Combined segmentation and paging 段页结合



- 程序员将进程分为若干个段
- 系统将每个段分为多个页
- 在进行地址转换的时候，根据段号，到段表中找到段基址，根据页号，到页表中找到对应的帧号，然后段基址和帧号结合，可以得到实际的物理地址

Summary

- 物理内存不够大，通过交换，将执行完或阻塞的进程交换出去，提高内存的利用率
- 交换的方式：分区，分页，分段
  - 分区：固定分区，不固定分区
  - 分页：逻辑地址到物理地址的转换——页表
  - 分段：逻辑地址到物理地址的转换——段表
- 虚拟地址和虚拟存储器：重载，物理内存不够大
- 段页结合：根据段号，到段表中找到段基址，根据页号，到页表中找到对应的帧号，然后段基址和帧号结合，可以得到实际的物理地址

Key Terms 术语

Operating system	Interactive	Batch	Resident monitor program
Schedule	Job control language	Time sharing system	Long term
Medium term	Short term	process	Process control block
Swapping	Partitioning	Page	Virtual Memory
Segmentation	Real memory	TLB	Inverted page talbe

Summary and Question

- 小结
  - 这节课我们对操作系统的功能和类型进行了介绍，并对操作系统的调度功能和内存管理功能做了详细的分析。
- 问题
  - 问题1：采用交换技术之后，为什么需要用虚拟地址？
  - 问题2：内存采用分段+分页后，如何得到存储的物理地址？
  - 问题3：简单描述一下普通页表和反向页表

Assignments 作业

- Review：8.5，8.7~8.10
- Problems 8.6，8.14

谢谢大家！