

0117401: Operating System

计算机原理与设计

Chapter 1-2: OS Structure

陈香兰

xlanchen@ustc.edu.cn

<http://staff.ustc.edu.cn/~xlanchen>

Computer Application Laboratory, CS, USTC @ Hefei

Embedded System Laboratory, CS, USTC @ Suzhou

February 26, 2024

温馨提示：



为了您和他人的工作学习，
请在课堂上**关机或静音**。

不要在课堂上接打电话。

outline

Computer System Operation

- A modern computer system

- System boot

- Interrupt

I/O Structure

- I/O Structure

- I/O operation

Storage Structure and Storage Hierarchy

- Storage Structure

- Storage hierarchy

Hardware Protection

- Hardware Protection

General System Architecture

- General System Architecture

- system call

Computing Environments

小结

CS & Von Neumann architecture

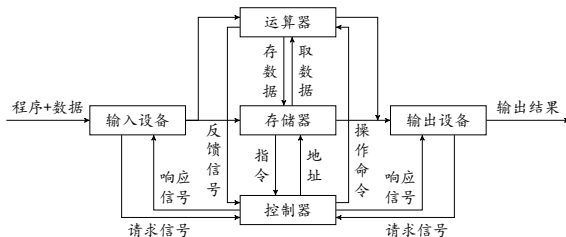
► 计算机

1. 不可编程的：强定制，高效
2. 可编程的：灵活

► 提供指令集，程序就是一个指令序列

冯·诺伊曼体系结构

- **五大部件**：运算器、控制器、存储器、I/O设备
- **存储器与CPU相分离；指令存储与数据存储共享存储器**



Outline

Computer System Operation

- A modern computer system

- System boot

- Interrupt

I/O Structure

- I/O Structure

- I/O operation

Storage Structure and Storage Hierarchy

- Storage Structure

- Storage hierarchy

Hardware Protection

- Hardware Protection

General System Architecture

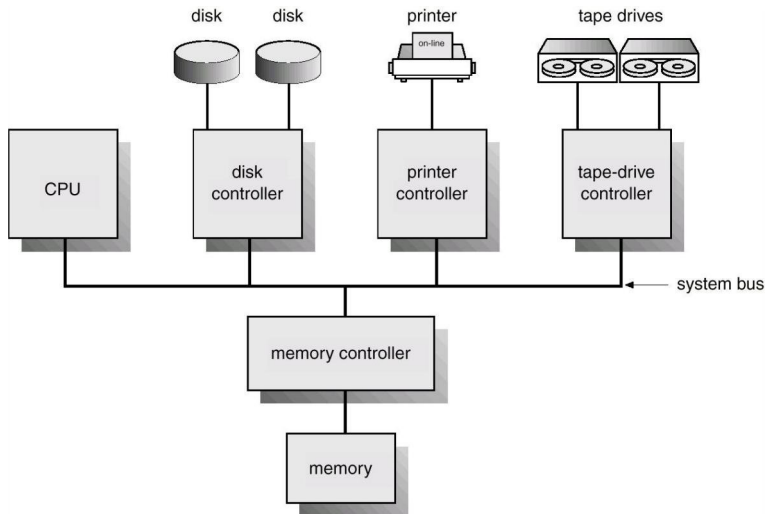
- General System Architecture

- system call

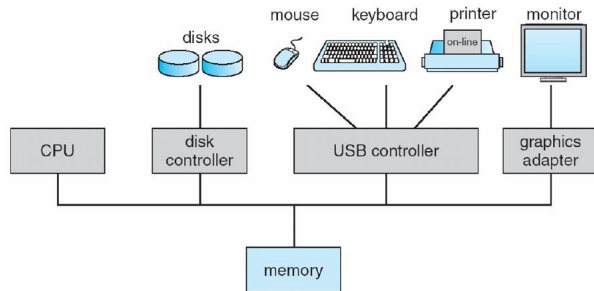
Computing Environments

- 小结

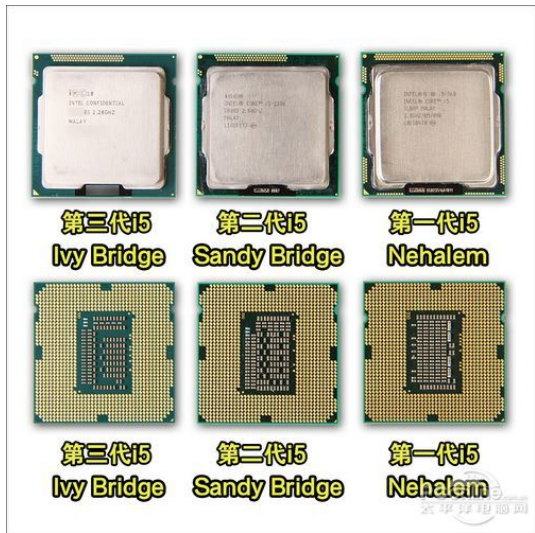
A modern computer system I



A modern computer system II

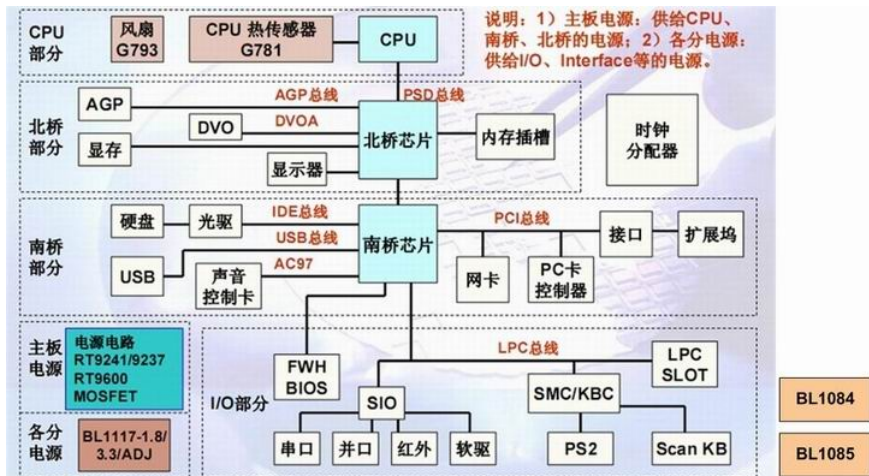


参考：三款core i5 CPU芯片外观比较



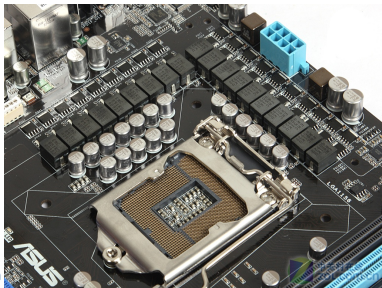
From: VB先睹為快！Intel三代Core i5搶先評測

参考：一个电脑主板芯片应用方案



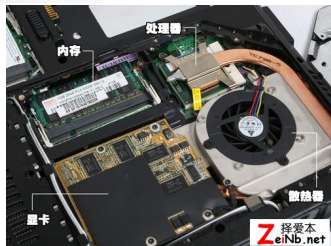
仅用作参考示意。From: http://www.chinesechip.com/news_4/ec3c0dbbed7f458cb7d06c012c86cc44.html

参考：华硕的一款主板



From: 中关村在线 华硕P7P55D Deluxe

参考：华硕F8H笔记本拆解



From: 华硕F8H系列笔记本评测

Outline

Computer System Operation

A modern computer system

System boot

Interrupt

I/O Structure

I/O Structure

I/O operation

Storage Structure and Storage Hierarchy

Storage Structure

Storage hierarchy

Hardware Protection

Hardware Protection

General System Architecture

General System Architecture

system call

Computing Environments

小结

System boot

- ▶ **Booting the system:** the procedure of starting a computer by loading the OS
- ▶ **Bootstrap program or bootstrap loader:** a small piece of code
 - ▶ Loaded at power-up or reboot
 - ▶ Typically stored in ROM or EPROM, generally known as firmware(固件)
 - ▶ initializes hardware
 - ▶ CPU registers, device controllers, memory content
 - ▶ Locate the OS, load at least a part of the OS into main memory & start executing it
- ▶ Platform dependent(平台相关/体系结构相关)
- ▶ Some use **two-step process**: a simple bootstrap loader fetches a more complex boot program from disk, which in turn loads the OS
- ▶ Some systems store the entire OS in ROM

Example: Linux system startup

typical operating systems startup course

Power-on→Bootstrap: BIOS→BootLoader: GRUB→OS: Linux

Linux (Intel i386)

Refer to appendix A of 《Understanding Linux Kernel》

- ▶ →RESET pin of the CPU
- ▶ cs:ip= 0xFFFF FFF0
- ▶ ROM BIOS (基本输入输出系统)

Example: Linux system startup (cont.)

BIOS (基本输入输出系统)

Basic I/O System (BIOS) : A set of programs stored in ROM, including

- ▶ Several interrupt-driven low-level procedures
- ▶ A bootstrap procedure, who
 - ▶ POST (Power On Self-Test)
 - ▶ Initializes hardware device
 - ▶ Searches for an OS to boot
 - ▶ Master Boot Record(MBR) on Hard drive, Boot Sector on floppy disk, network
- ▶ Copies the first sector of the OS into RAM 0x00007C00, and jumps & executes

Example: Linux system startup (cont.)

Master Boot Record, MBR, 主引导记录

- ▶ the first sector on a hard drive, a special type of boot sector
- ▶ MBR = MBR code (also called boot loader) + partition table
 - ▶ MBR code: code necessary to startup the OS
 - ▶ typical boot loader: GRUB

Example: Linux system startup (cont.)

Master Boot Record, MBR, 主引导记录

Structure of a classical generic MBR

| Address | | Description | | Size in bytes |
|---------------------------|------|---------------------|---|---------------|
| Hex | Dec | | | |
| +000h | +0 | Bootstrap code area | | 446 |
| +1BEh | +446 | Partition entry #1 | Partition table (for primary partitions) | 16 |
| +1CEh | +462 | Partition entry #2 | | 16 |
| +1DEh | +478 | Partition entry #3 | | 16 |
| +1EEh | +494 | Partition entry #4 | | 16 |
| +1FEh | +510 | 55h | Boot signature | 2 |
| +1FFh | +511 | AAh | | |
| Total size: 446 + 4*6 + 2 | | | | 512 |

??? After starts up

- ▶ Executes prearranged process, or
- ▶ Waits for interrupt

Modern OSs are interrupt-driven (中断驱动的) .

Outline

Computer System Operation

A modern computer system

System boot

Interrupt

I/O Structure

I/O Structure

I/O operation

Storage Structure and Storage Hierarchy

Storage Structure

Storage hierarchy

Hardware Protection

Hardware Protection

General System Architecture

General System Architecture

system call

Computing Environments

小结

Interrupt I

Interrupt represents an event to be handled

For hardware: Device interrupt

- ▶ The completion of an I/O operation
- ▶ a key stroke or a mouse move
- ▶ timer
- ▶ ...

For error (also hardware): exception

1. Trap for debug
2. Fault
 - ▶ example: page fault, division by zero, invalid memory access
3. Abort, a serious error

For software: System call

Interrupt II

- ▶ To request for some operating-system service
 - ▶ Linux: INT 0x80
 - ▶ MS/DOS, windows: INT 0x21

Modern OSs are interrupt-driven (中断驱动的)

Interrupt handling I

When the CPU is interrupted

1. Stops what it is doing
2. Incoming interrupts are disabled to prevent a lost interrupt
3. Transfers control to the **ISR (Interrupt Service Routine, 中断服务例程)**
 - ▶ **ISR**: A generic routine in fixed location and then call the interrupt-specific handler
 - ▶ **interrupt vector table (中断向量表)**

When the ISR completed,
Back to interrupted program

Interrupt handling II

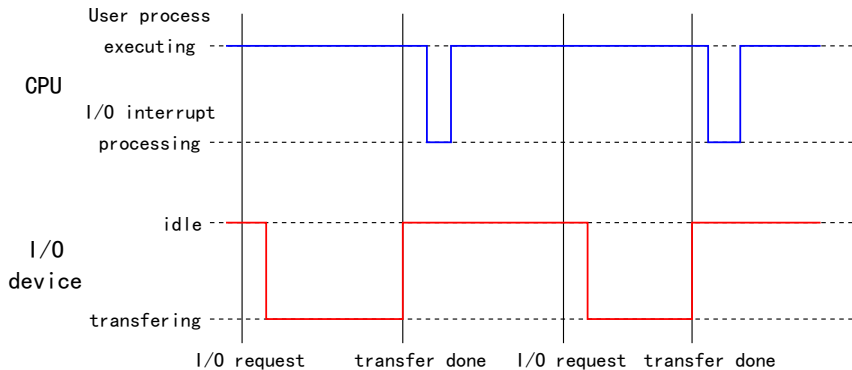
► HOW ?

—— OS preserves the state of the CPU by storing registers and the program counter.

also called context (上下文, 硬件上下文) .

- Old: Fixed location, or a location indexed by the device number
- Recent: system stack (Linux: 内核态栈)

Interrupt time line for a single process doing output



Example: interrupts in I386

- ▶ protect mode （保护模式）
 - ▶ IDT （Interrupt Descriptor Table, 中断描述符表）
 - ▶ OS填写IDT表，包括每个中断处理例程的入口地址等信息
 - ▶ 中断发生的时候，CPU根据从中断控制器获得的中断向量号在IDT表中索引到对应的中断处理例程（ISR）入口地址，并跳转过去运行
 - ▶ 保存上下文
 - ▶ 处理中断
 - ▶ 恢复上下文

Outline

Computer System Operation

- A modern computer system

- System boot

- Interrupt

I/O Structure

- I/O Structure**

- I/O operation

Storage Structure and Storage Hierarchy

- Storage Structure

- Storage hierarchy

Hardware Protection

- Hardware Protection

General System Architecture

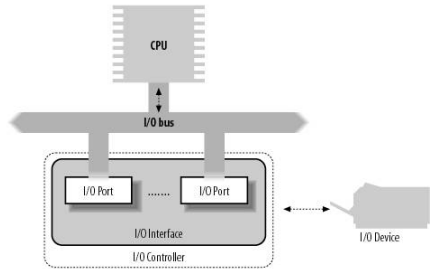
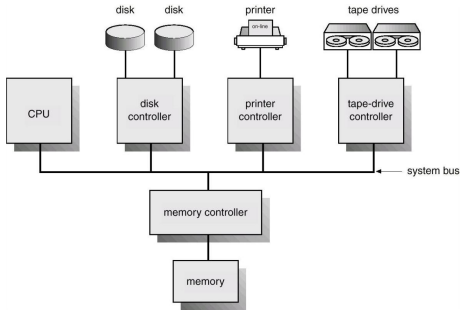
- General System Architecture

- system call

Computing Environments

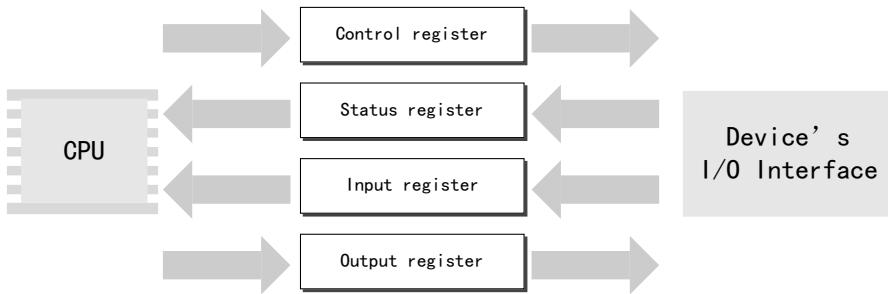
- 小结

I/O structure



I/O structure

- ▶ Each device controller is in charge of a particular device type
- ▶ Each device controller has
 - ▶ a local buffer & a set of special-purpose registers
- ▶ Data transfer, two phrase
 - ▶ Main memory \leftarrow (CPU) \rightarrow local buffer of controller
 - ▶ device \leftarrow (device controller) \rightarrow local buffer
- ▶ I/O devices & CPU can execute **concurrently** (并发地)
 - ▶ Share/compete memory cycle
 - ▶ Memory controller



Outline

Computer System Operation

- A modern computer system

- System boot

- Interrupt

I/O Structure

- I/O Structure

- I/O operation**

Storage Structure and Storage Hierarchy

- Storage Structure

- Storage hierarchy

Hardware Protection

- Hardware Protection

General System Architecture

- General System Architecture

- system call

Computing Environments

- 小结

I/O operation

- ▶ CPU start an I/O operation by
 - ▶ Loading the appropriate registers within the device controller
 - ▶ When complete, device controller informs CPU by
 - ▶ Triggering an interrupt, or
 - ▶ Simply set a flag in one of their registers
- ▶ Two I/O methods
 - ▶ synchronous VS. asynchronous

Outline

Computer System Operation

- A modern computer system

- System boot

- Interrupt

I/O Structure

- I/O Structure

- I/O operation

Storage Structure and Storage Hierarchy

- Storage Structure

- Storage hierarchy

Hardware Protection

- Hardware Protection

General System Architecture

- General System Architecture

- system call

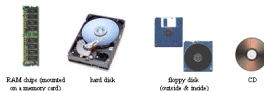
Computing Environments

- 小结

Storage structure

- ▶ **Von Neumann architecture VS. Harvard architecture**
 - ▶ Separated data & code in different memory???
- ▶ **Main memory** (RAM) is the only large storage media that the CPU can access directly
 - ▶ Small, Volatile
- ▶ **Secondary storage** is an extension of main memory that provides large nonvolatile storage capacity

- ▶ Magnetic disk, 磁盘
- ▶ Optical disk, 光盘
- ▶ Magnetic tape, 磁带



Memory vs. registers

Memory VS. registers

- ▶ **Same:** Access directly for CPU
 - ▶ Register name
 - ▶ Memory address
- ▶ **Different:** access speed
 - ▶ Register, one cycle of the CPU clock
 - ▶ Memory, Many cycles (2 or more)
- ▶ **Disadvantage:**
 - ▶ CPU needs to stall frequently & this is intolerable
- ▶ **Remedy :** cache, 高速缓存

Magnetic disks

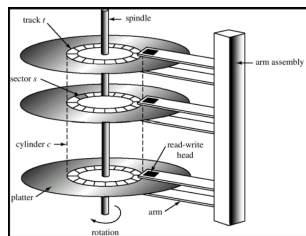
- ▶ Magnetic disks - rigid metal or glass platters covered with magnetic recording material
 - ▶ Disk surface is logically divided into **tracks** (磁道), which are subdivided into **sectors** (扇区).
 - ▶ The **disk controller** determines the logical interaction between the device and the computer.

▶ Position (定位) time T_p

▶ Transfer (传输) time T_T

▶ T_T VS. T_p

▶ Please Store data closely



Magnetic disks

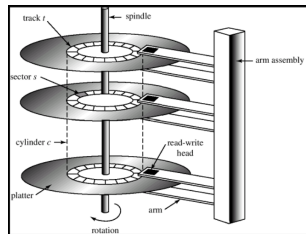
- ▶ Magnetic disks - rigid metal or glass platters covered with magnetic recording material
 - ▶ Disk surface is logically divided into **tracks** (磁道), which are subdivided into **sectors** (扇区).
 - ▶ The **disk controller** determines the logical interaction between the device and the computer.

- ▶ Position (定位) time T_p
 - ▶ $T_p \approx T_s + T_R \approx \text{mms}$
 - ▶ Seek time T_s
 - ▶ Rotational latency T_R

- ▶ Transfer (传输) time T_T

- ▶ T_T VS. T_p

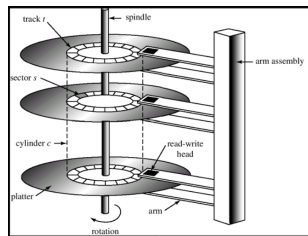
- ▶ Please Store data closely



Magnetic disks

- ▶ Magnetic disks - rigid metal or glass platters covered with magnetic recording material
 - ▶ Disk surface is logically divided into **tracks** (磁道), which are subdivided into **sectors** (扇区).
 - ▶ The **disk controller** determines the logical interaction between the device and the computer.

- ▶ Position (定位) time T_p
- ▶ Transfer (传输) time T_T
 - ▶ T_T
 $\approx \text{data size} \times \text{Transfer rate}$
 - ▶ Transfer rate $\approx (\text{nM/s})^{-1}$
 $\approx (\text{n Byte/us})^{-1}$
 $\approx 1/\text{n us/Byte}$

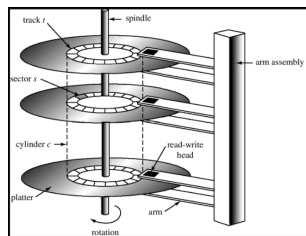


- ▶ T_T VS. T_p
 - ▶ Please **Store data closely**

Magnetic disks

- ▶ Magnetic disks - rigid metal or glass platters covered with magnetic recording material
 - ▶ Disk surface is logically divided into **tracks** (磁道), which are subdivided into **sectors** (扇区).
 - ▶ The **disk controller** determines the logical interaction between the device and the computer.

- ▶ Position (定位) time T_p
- ▶ Transfer (传输) time T_T
- ▶ T_T VS. T_p
 - ▶ Please **Store data closely**



Outline

Computer System Operation

A modern computer system

System boot

Interrupt

I/O Structure

I/O Structure

I/O operation

Storage Structure and Storage Hierarchy

Storage Structure

Storage hierarchy

Hardware Protection

Hardware Protection

General System Architecture

General System Architecture

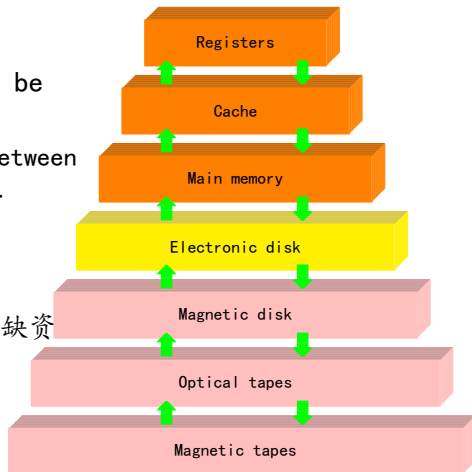
system call

Computing Environments

小结

Storage hierarchy, 存储的层次

- ▶ Storage systems in a CS can be organized in a **hierarchy**.
 1. The **contradiction** (矛盾) between COST, SPEED, and CAPACITY.
 2. COST per bit.
 3. Volatility (易失性) VS. persistence (持久性).
- ▶ MM is a scarce resource (稀缺资源).



Caching

- ▶ **Caching** (高速缓存技术)
 - ▶ Copying information into faster storage system
 - ▶ When accessing, first check in the **cache**,
 - ▶ if **In**: use it directly
 - ▶ **Not in**: get from upper storage system, and leave a copy in the cache
- ▶ Using of caching
 - ▶ Registers provide a high-speed cache for main memory
 - ▶ **Instruction cache & data cache**
 - ▶ Main memory can be viewed as a fast cache for secondary storage
 - ▶ ...

Cache management

- ▶ Design problem
 - ▶ Hardware or software?
 - ▶ Cache size & Replacement policy is important
 - ▶ Hit rate $\approx 80\% \sim 99\%$ is OK!

Memory Wall

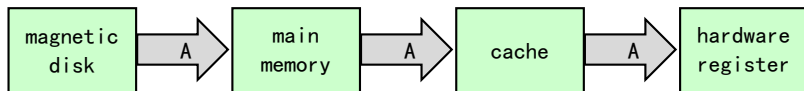
Memory Wall, 内存墙

- ▶ the growing disparity of speed between CPU and memory outside the CPU chip¹.
 - ▶ From 1986 to 2000, CPU speed improved at an annual rate of 55% while memory speed only improved at 10%.
 - ▶ Trend: memory latency would become an overwhelming **bottleneck** in computer performance

¹From Wikipedia: Random-access memory

Coherency and consistency

- ▶ Multitasking environments must be careful to use most recent value, no matter where it is stored in the storage hierarchy
- ▶ Migration of Integer A from Disk to Register



- ▶ The same data may appear in different level of the storage system
- ▶ When
 - ▶ Simple batch system, no problem
 - ▶ Multitasking, always obtain the most recently updated value
 - ▶ Multiprocessor, cache coherency (always implicit to OS)
 - ▶ Distributed system?

Performance of Various Levels of Storage

- Movement between levels of storage hierarchy can be explicit or implicit

| Level | 1 | 2 | 3 | 4 |
|---------------------------|---|-------------------------------|-------------|---------------|
| Name | registers | cache | main memory | disk storage |
| Typical size | <1KB | >16MB | >16GB | >100GB |
| Implementation technology | custom memory with multiple ports, CMOS | on-chip or off-chip CMOS SRAM | CMOS DRAM | magnetic disk |
| Access time (ns) | 0.25 - 0.5 | 0.5 - 25 | 80 - 250 | 5,000.000 |
| Bandwidth (MB/sec) | 20,000 - 100,000 | 5000 - 10,000 | 1000 - 5000 | 20 - 150 |
| Managed by | compiler | hardware | OS | OS |
| Backed by | cache | main memory | disk | CD or tape |

Outline

Computer System Operation

- A modern computer system

- System boot

- Interrupt

I/O Structure

- I/O Structure

- I/O operation

Storage Structure and Storage Hierarchy

- Storage Structure

- Storage hierarchy

Hardware Protection

- Hardware Protection

General System Architecture

- General System Architecture

- system call

Computing Environments

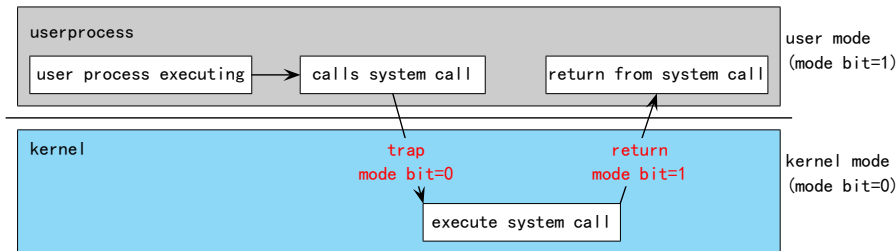
- 小结

Hardware protection

- ▶ A properly designed OS must ensure that an incorrect (or malicious) **program cannot cause other programs to execute incorrectly.**
 1. When in dead loop
 2. When sharing recourses
 3. When one erroneous program might modify the program or data of another program, or even the OS
- ▶ Hardware must provide protection
 1. **Dual-Mode Operation**
 2. **I/O protection**
 3. **Memory protection**
 4. **CPU protection**

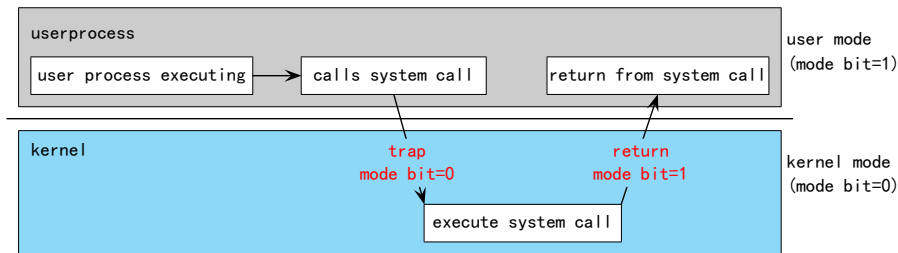
Hardware protection 1: Dual-Mode Operation (双操作模式)

- ▶ Using **mode bit** to provide different modes of execution
 - ▶ **mode bit=1**≡**User mode (用户模式)** :
execution done on behalf of user
 - ▶ **mode bit=0**≡**privileged mode (特权模式)** ,
also called **monitor mode (监督程序模式) /supervisor mode (管理模式) /system mode (系统模式)** :
execution done on behalf of OS
 - ▶ Privileged instructions



Hardware protection 1: Dual-Mode Operation (双操作模式)

- ▶ User program VS. OS (or Kernel)
 - ▶ Switch between user mode (1) and privileged mode(0)
 - ▶ Boot: from privileged mode.
 - ▶ User program: user mode.
 - ▶ Interrupt (include system call): switch to privileged mode.
 - ▶ OS: privileged mode



► Example: i386

- 4 modes (2 mode bits)
- Linux uses 2 mode (00b & 11b)

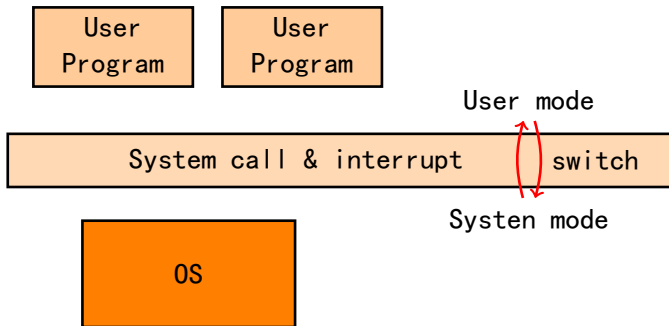
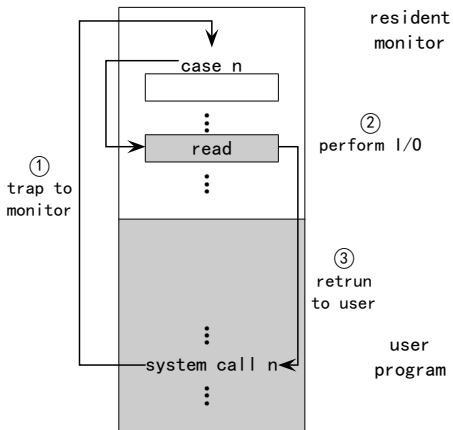


Figure: Linux uses two modes

Hardware protection 2: I/O protection

- ▶ Preventing the users from issuing illegal I/O instructions
- ▶ All I/O instructions are privileged instructions
 - ▶ instead of performing I/O operation directly, user program must make a system call
 - ▶ OS, executing in monitor mode, checks validity of request and does the I/O
 - ▶ input is returned to the program by the OS
- ▶ Smart hacker may...
 - ▶ Stores in the interrupt vector a new address, which points to a malicious routine
 - ▶ The I/O protection is compromised
 - ▶ We need some more protection...

Use of a system call to perform I/O



Hardware protection 3: Memory protection

- ▶ At least for interrupt vector and the ISR

- ▶ **Base register protection scheme**

- ▶ Base register + Limit register
- ▶ Memory outside is protected
- ▶ OS has unrestricted access to both monitor and user's memory
- ▶ Load instructions for the base/limit registers

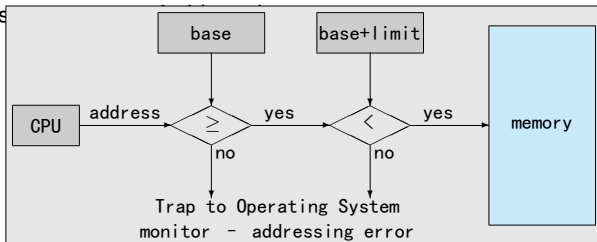
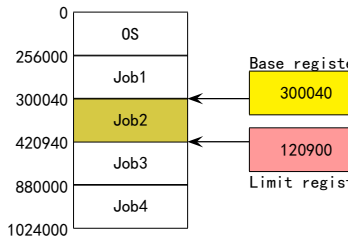


Figure: Hardware address protection with base and limit registers

Hardware protection 4: CPU protection

- ▶ OS should be always take control of everything
 - ▶ What if a user program is in dead loop?
- ▶ Timer
 - ▶ Interrupts computer after specified period
 - ▶ Periodically or one-shot
 - ▶ Load-timer is also a privileged instruction
- ▶ Usage
 - ▶ Time sharing
 - ▶ Compute current time
 - ▶ Alarm or timer

Timer to prevent infinite loop / process hogging resources

- ▶ Set interrupt after specific period
- ▶ Operating system decrements counter
- ▶ When counter zero generate an interrupt
- ▶ Set up before scheduling process to regain control or terminate program that exceeds allotted time

Outline

Computer System Operation

- A modern computer system

- System boot

- Interrupt

I/O Structure

- I/O Structure

- I/O operation

Storage Structure and Storage Hierarchy

- Storage Structure

- Storage hierarchy

Hardware Protection

- Hardware Protection

General System Architecture

- General System Architecture

- system call

Computing Environments

- 小结

General system architecture

- ▶ multiprogramming
- ▶ time sharing
- ▶ OS: in kernel (privileged) mode
 - ▶ control hardware & software resource
 - ▶ execute privileged instruction
 - ▶ system call

Outline

Computer System Operation

- A modern computer system

- System boot

- Interrupt

I/O Structure

- I/O Structure

- I/O operation

Storage Structure and Storage Hierarchy

- Storage Structure

- Storage hierarchy

Hardware Protection

- Hardware Protection

General System Architecture

- General System Architecture

- system call**

Computing Environments

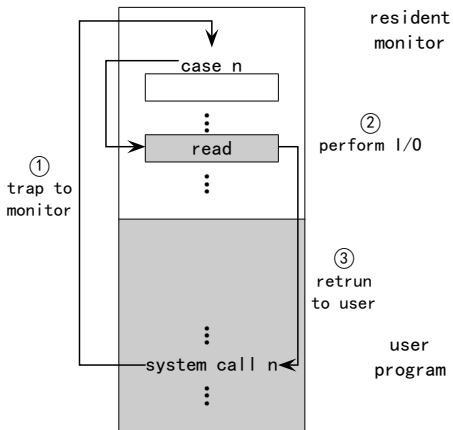
- 小结

system call

System call—like a common function call, but totally different!

- ▶ **Trap** to a specific location in interrupt vector
 - ▶ `int` (i386)
 - ▶ `trap` (SUN SPARC)
 - ▶ `syscall` (MIPS R2000)
- ▶ Control passes to a **service routine** in the OS, and the mode bit is set to **monitor mode**
- ▶ The kernel
 - ▶ Verifies that the parameters are correct and legal
 - ▶ Executes the request
 - ▶ Returns control to the instruction following the system call

Use of a system call to perform I/O



Computing Environments

- ▶ Traditional computer

- ▶ 随计算机的发展而变化

- ▶ Office environment

- ▶ PCs connected to a network, terminals attached to mainframe or minicomputers providing batch and timesharing

- ▶ Now portals allowing networked and remote systems access to same resources

- ▶ Home networks

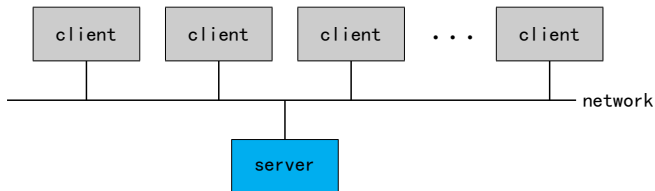
- ▶ Used to be single system, then modems

- ▶ Now firewalled, networked

Computing Environments

▶ Client-Server Computing

- ▶ Dumb terminals supplanted by smart PCs
- ▶ Many systems now servers, responding to requests generated by clients
 - ▶ Compute-server provides an interface to client to request services (i.e. database)
 - ▶ File-server provides interface for clients to store and retrieve files



Computing Environments

- ▶ 其他
 - ▶ Peer-to-Peer Computing
 - ▶ Web-Based Computing
 - ▶ Grid Computing
 - ▶ Cloud Computing
 - ▶ 普适计算Pervasive/Ubiquitous Computing

小结

Computer System Operation

- A modern computer system

- System boot

- Interrupt

I/O Structure

- I/O Structure

- I/O operation

Storage Structure and Storage Hierarchy

- Storage Structure

- Storage hierarchy

Hardware Protection

- Hardware Protection

General System Architecture

- General System Architecture

- system call

Computing Environments

小结

Thank you! Any question?