操作系统
The Operating System on Linux Kernel

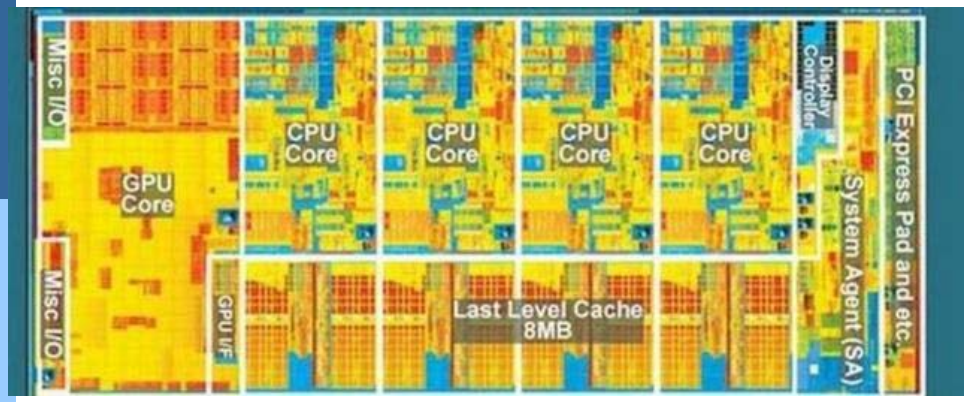
原仓周

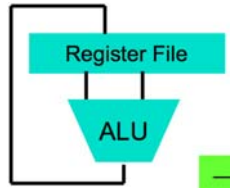
yuancz@buaa.edu.cn

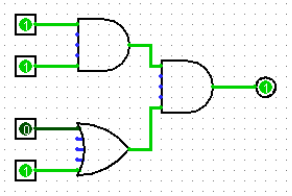
课程QQ群：805430126

软件需求的发展

硬件:多核CPU　GPU　谷歌TPU

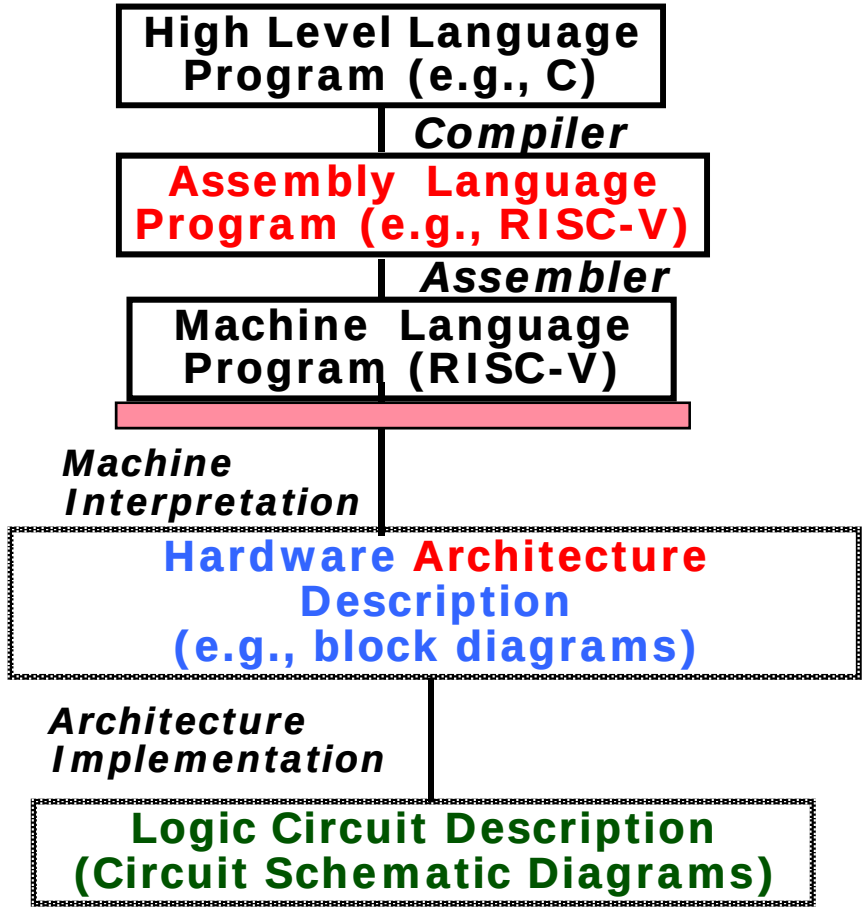






Tensor Processing Unit

One of the Great Ideas: Abstraction (Levels of Representation/Interpretation)

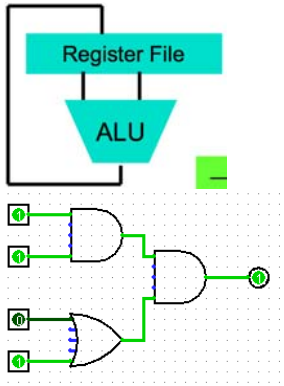



```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

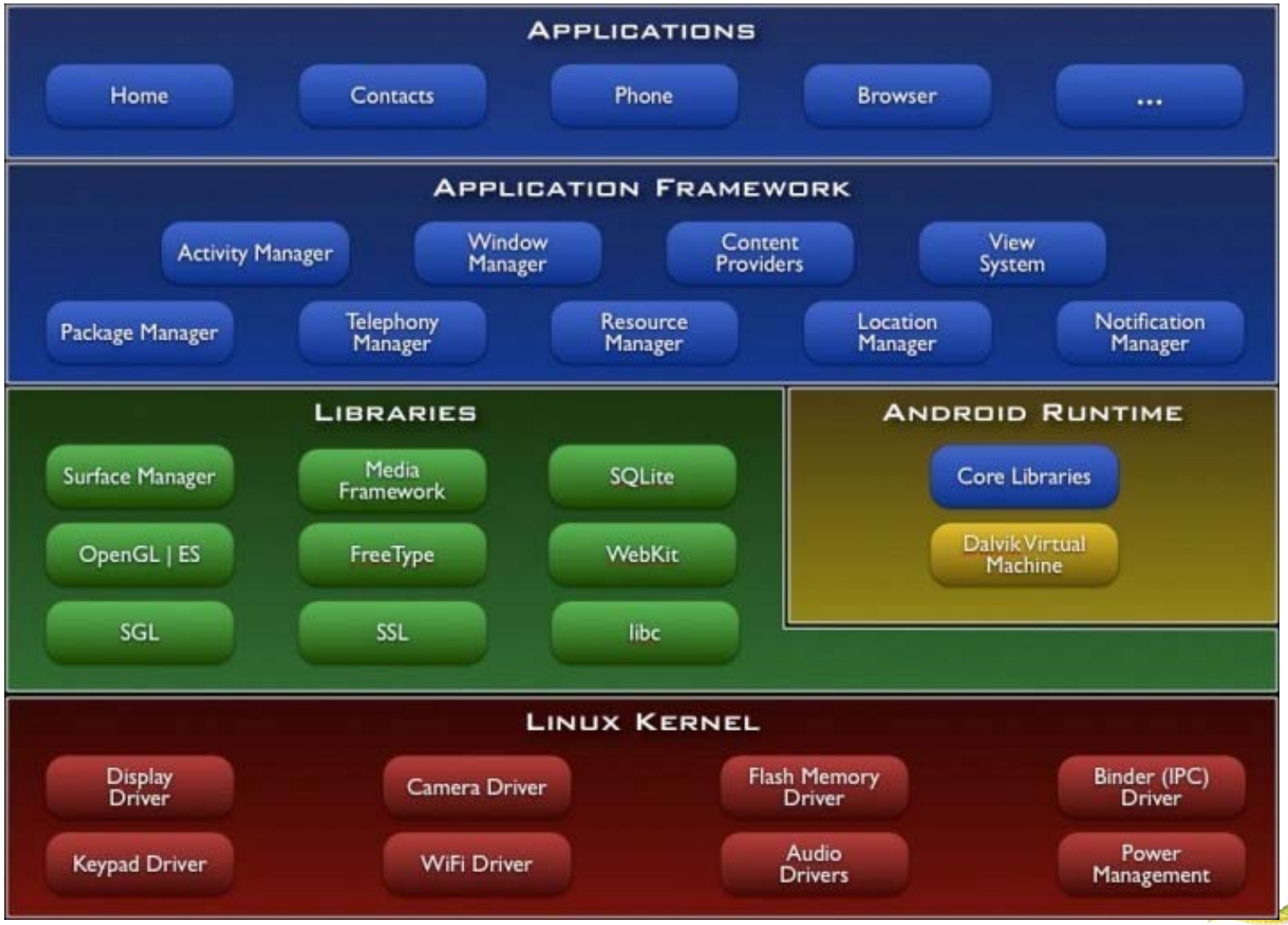
```
lw    $t0, 0($2)
lw    $t1, 4($2)
sw    $t1, 0($2)
sw    $t0, 4($2)
```

Anything can be represented as a *number*, i.e., data or instructions

```
1000 1101 1110 0010 0000 0000 0000 0000
1000 1110 0001 0000 0000 0000 0000 0100
1010 1110 0001 0010 0000 0000 0000 0000
1010 1101 1110 0010 0000 0000 0000 0100
```

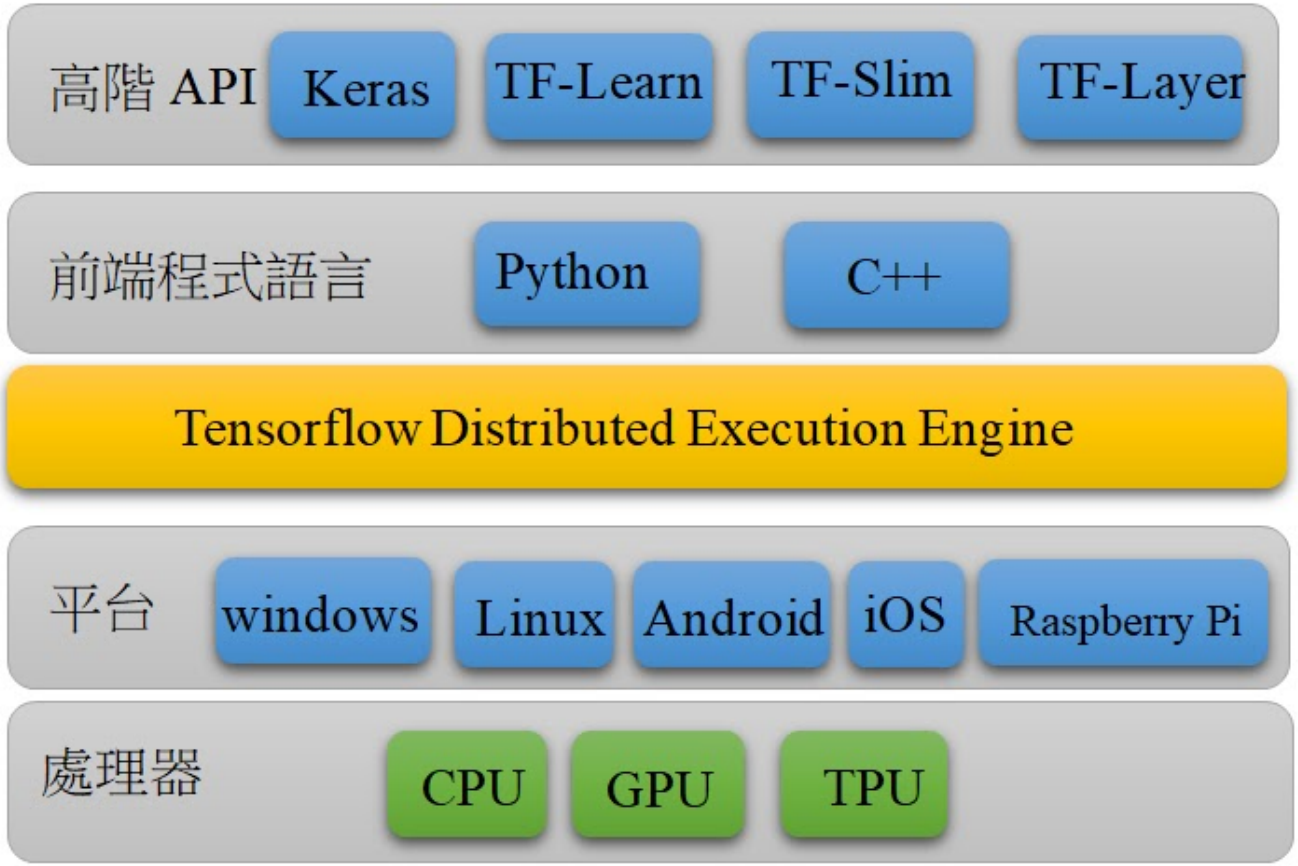


Android Architecture 分层架构



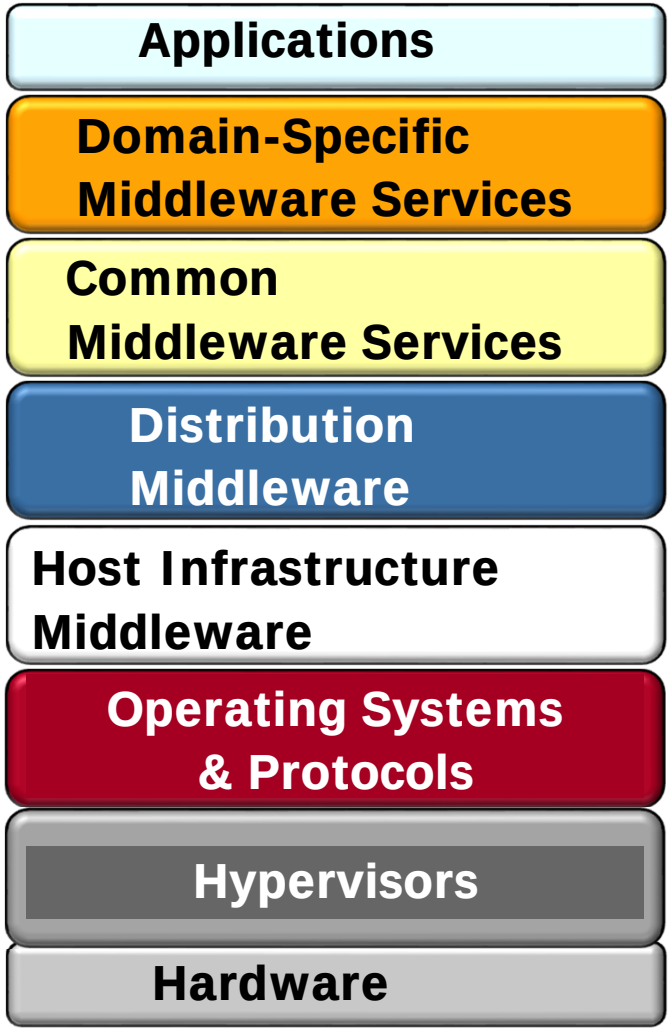
Tensorflow分层架构

Tensorflow 架構圖說明:

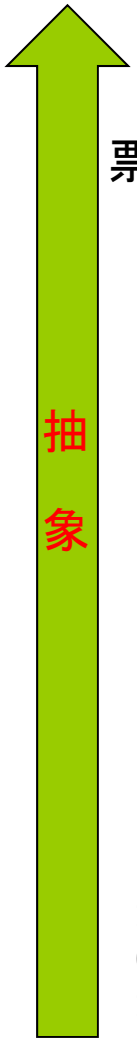


Software Stack 软件栈

领域模型

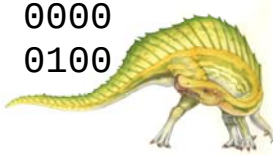


冯・诺依曼

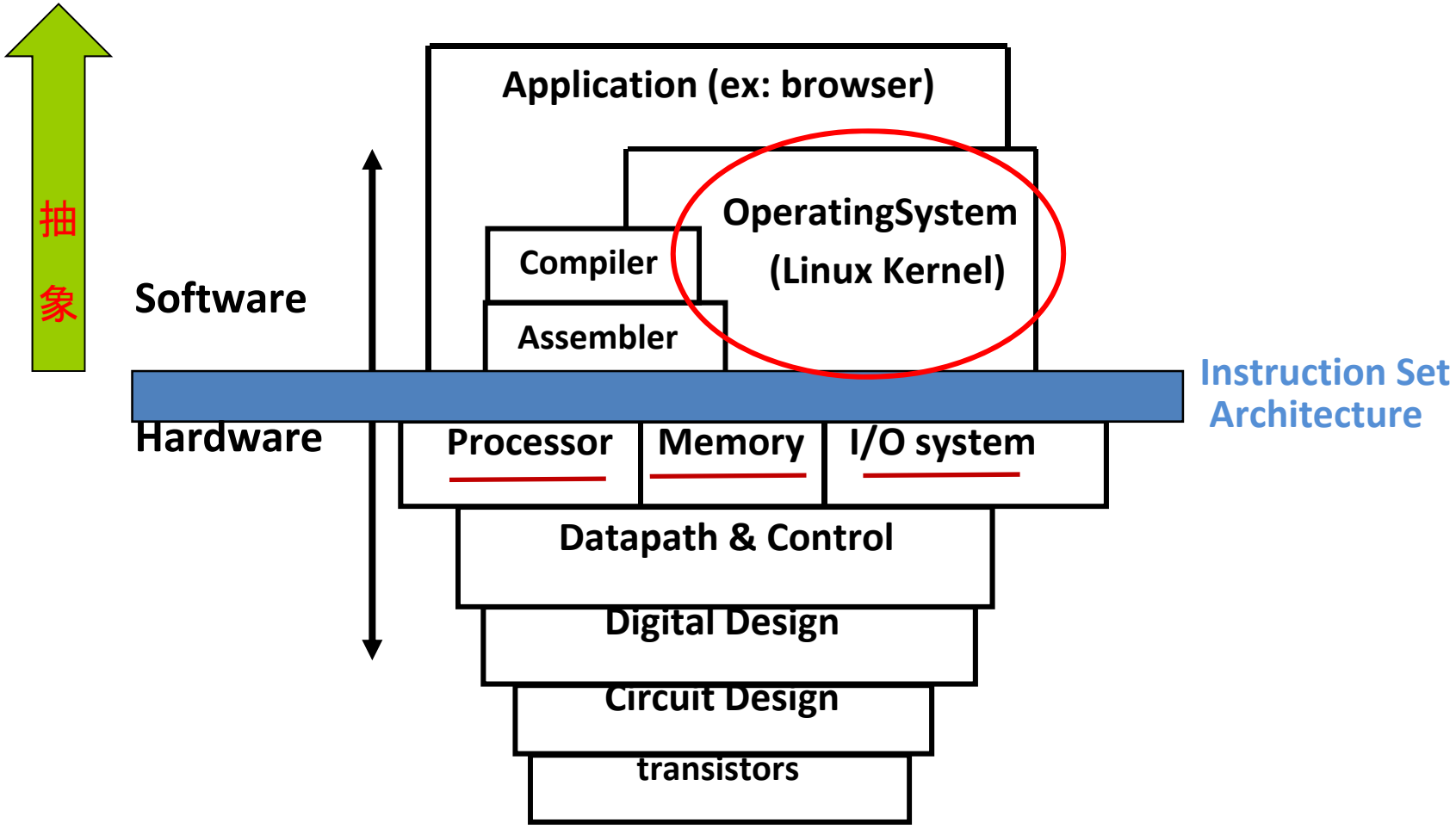


票据、账户、订单

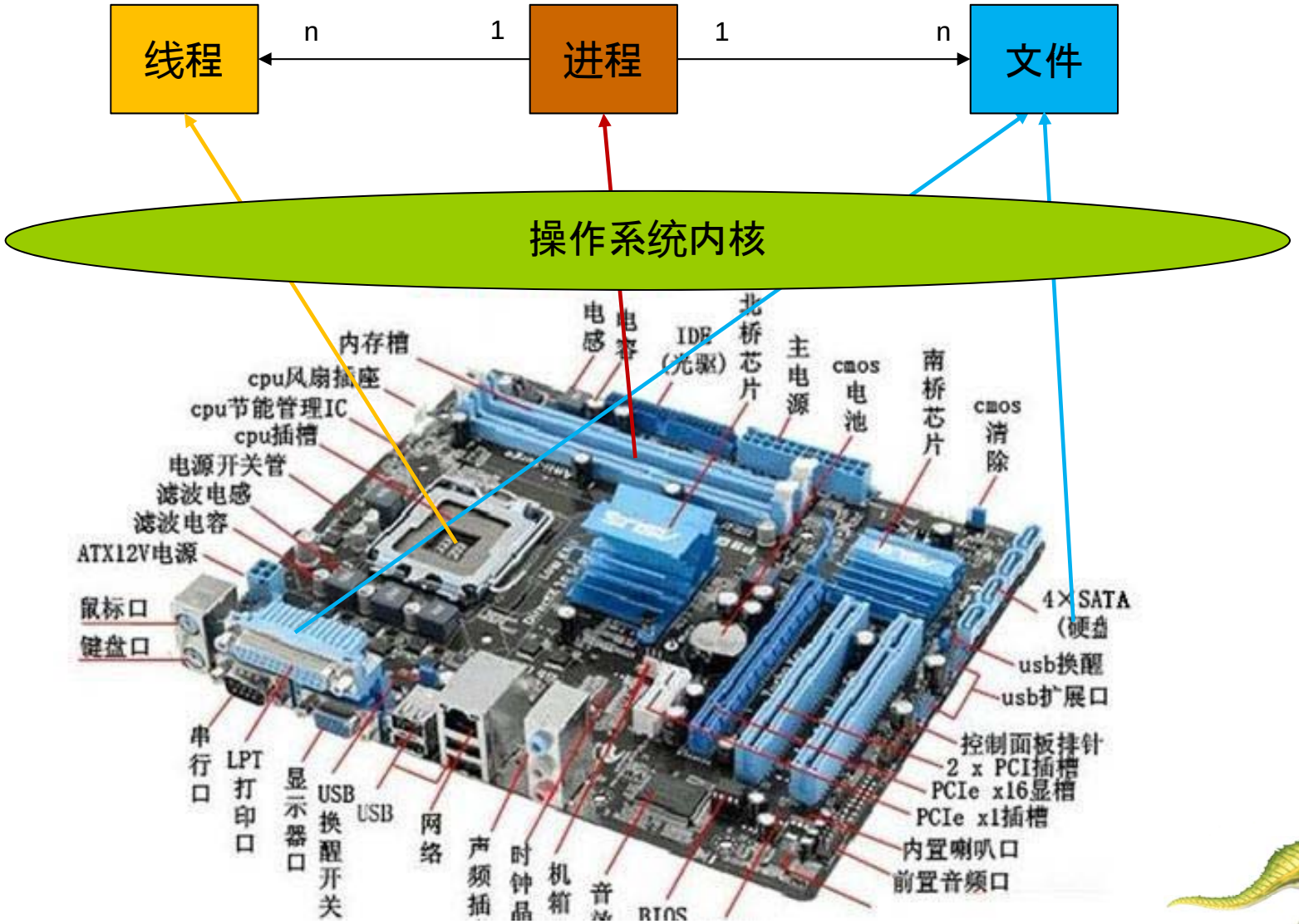
```
1000 1101 1110 0010
0000 0000 0000 0000
1000 1110 0001 0000
0000 0000 0000 0100
```



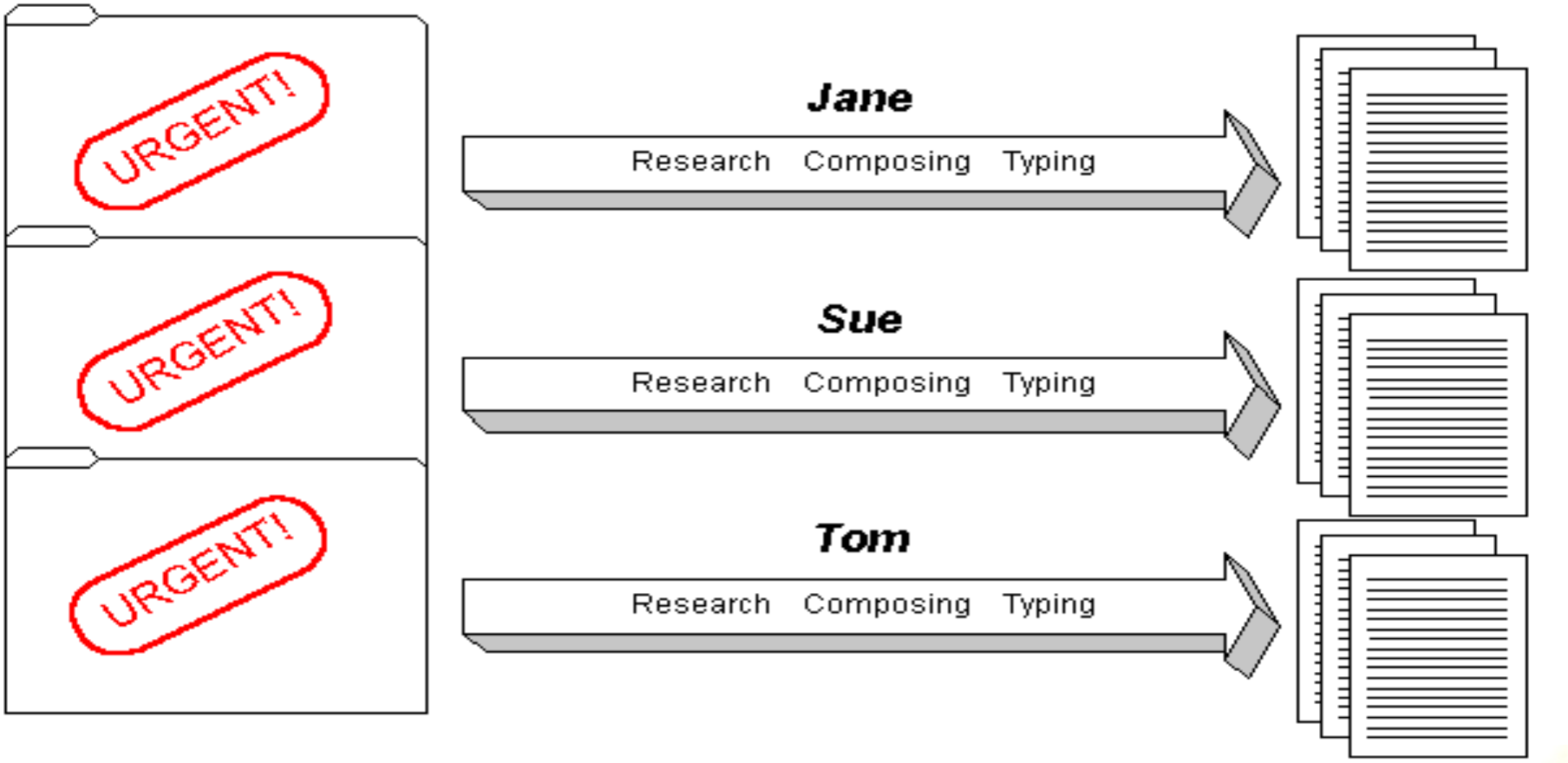
操作系统在计算机系统中的位置



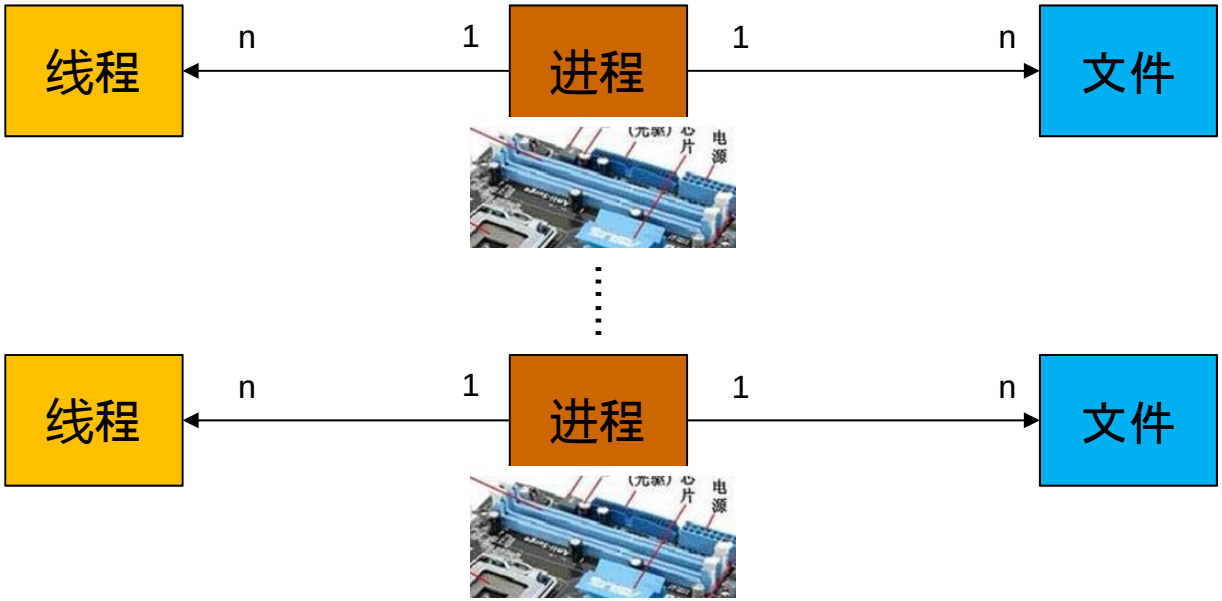
操作系统的抽象


线程
n
1
进程
1
n
文件
操作系统内核
内存槽
cpu风扇插座
cpu节能管理IC
cpu插槽
电源开关管
滤波电感
滤波电容
ATX12V电源
鼠标口
键盘口
串行口
LPT打印口
显示器口
USB换醒开关
USB
网络
声频插
时钟晶
机箱
音
BIOS
电感
电容
IDE(光驱)
北桥芯片
主电源
cmos电池
南桥芯片
cmos清除
4×SATA(硬盘
usb换醒
usb扩展口
控制面板排针
2 x PCI插槽
PCIe x16显槽
PCIe x1插槽
内置喇叭口
前置音频口

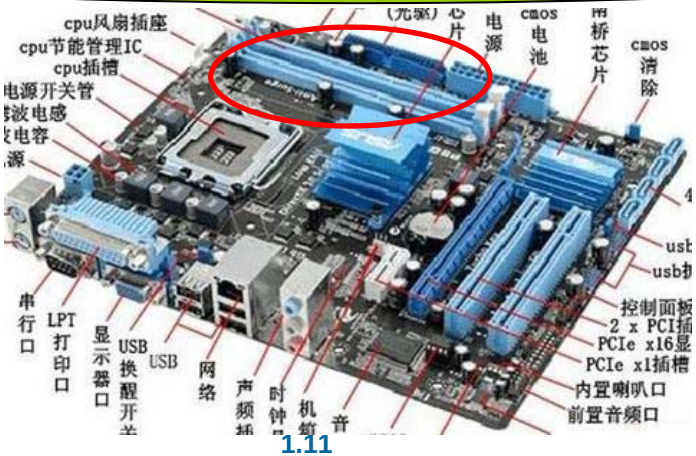
Another Great Idea: Parallelism(并行）



操作系统的虚拟(内存)

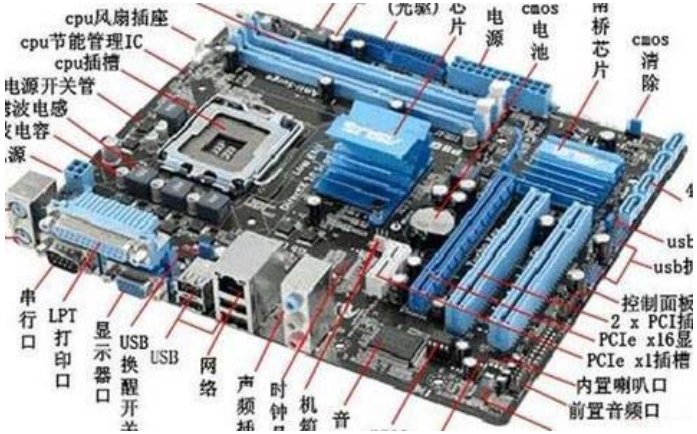


操作系统内核

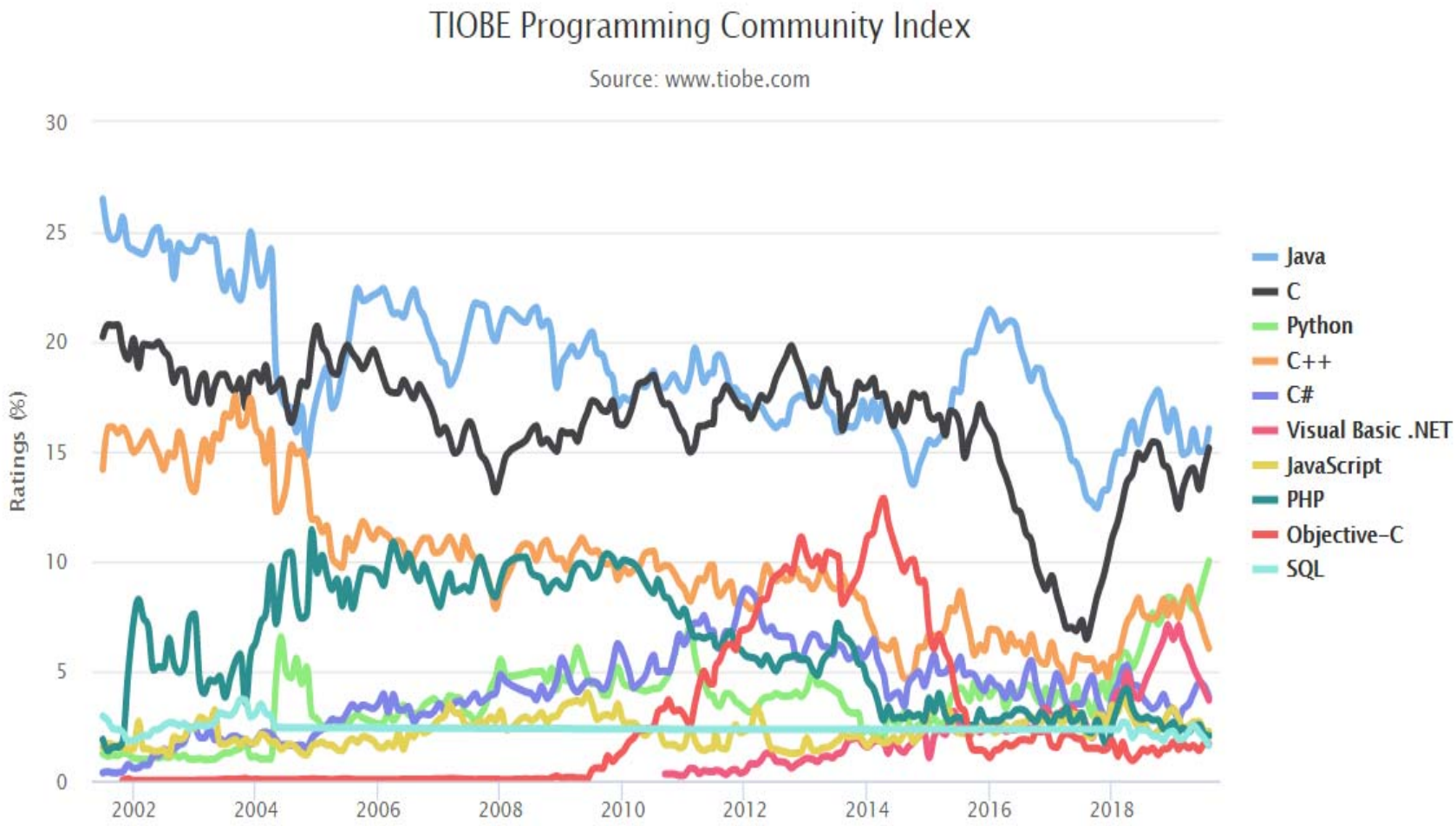


虚拟化技术

虚拟化软件

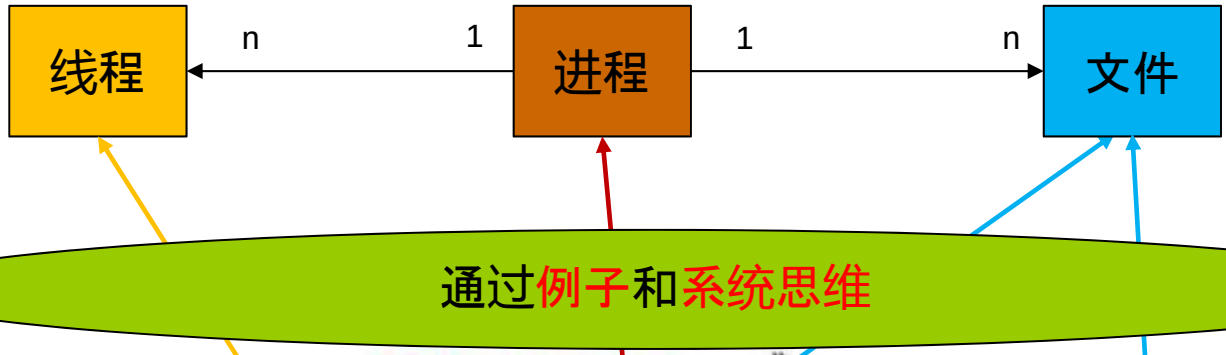

cpu风扇插座
cpu节能管理IC
cpu插槽
电源开关管
电源
CMOS
电池
CMOS
清除
串行口
LPT
打印口
显示器口
USB
USB
网络
声频
时钟
机箱
音
PCIe x16显
PCIe x1插槽
内置喇叭口
前置音频口

2019年9月3日前编程语言排名趋势



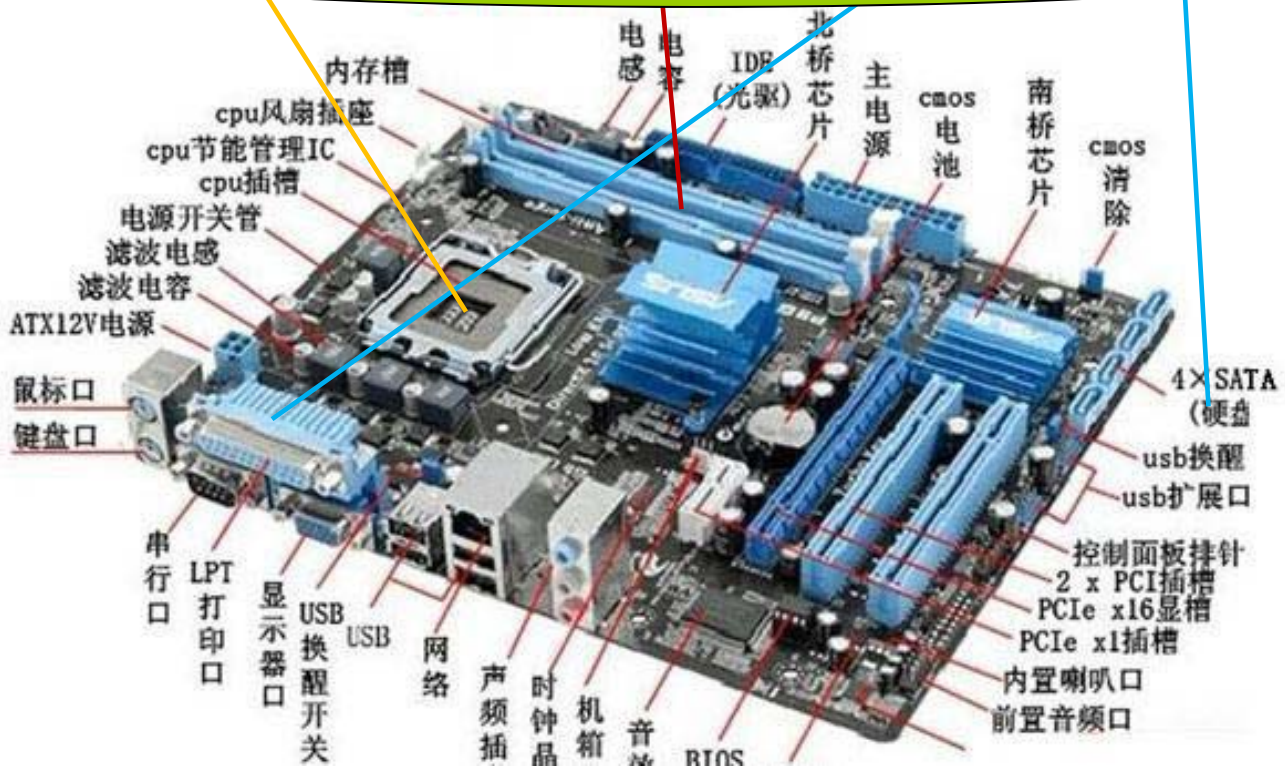
注意学习方法

新知识



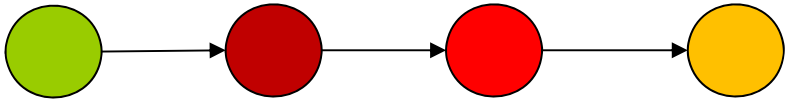
通过例子和系统思维

基础知识：程序在硬件各部分的运行机制

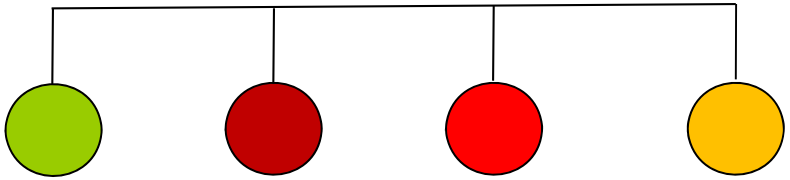


系统思维

- 逻辑思维（有理）

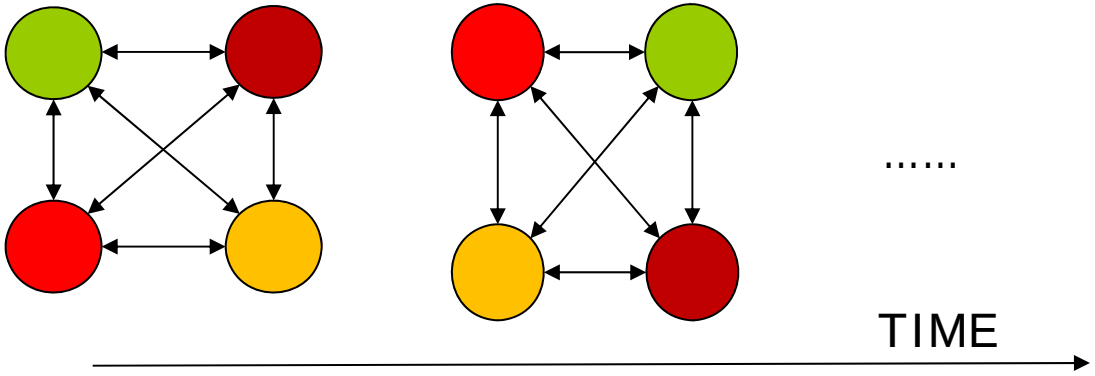


- 结构思维（有条理）

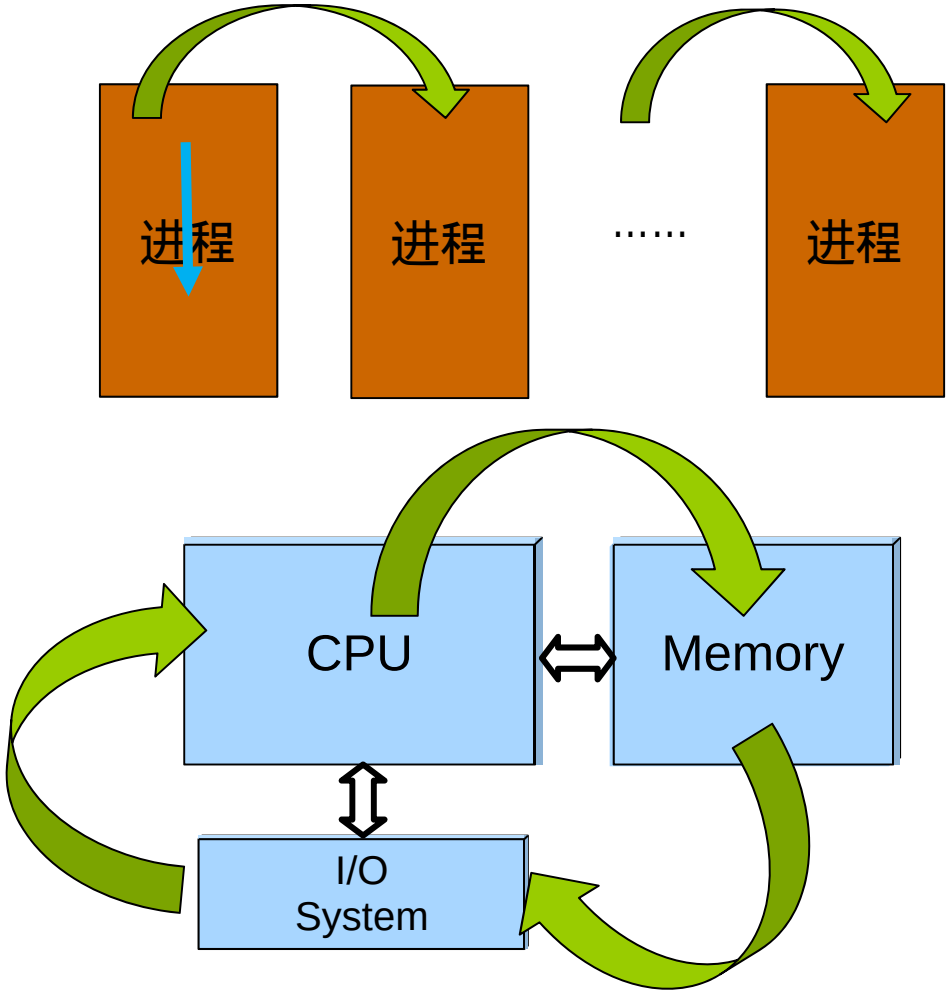


- 系统思维（重点在关系，动态变换）

计算思维之一



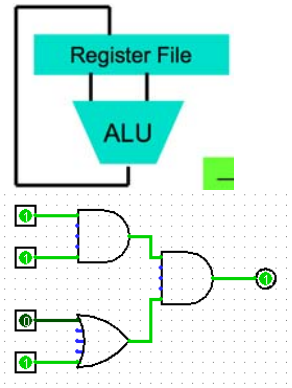
关系　与　动态变化




```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

```
lw      $t0, 0($2)
lw      $t1, 4($2)
sw      $t1, 0($2)
sw      $t0, 4($2)
```

```
1000 1101 1110 0010
0000 0000 0000 0000
1000 1110 0001 0000
0000 0000 0000 0100
```



课程形式

- 课堂
  - 主课
  - 习题和演讲课
  - 讨论课
- 课程实验
  - 上机实验(报告)
  - 论文分析

教　材

《操作系统概念　第9版》

Abraham Silberschatz等著，郑扣根等译

机械工业出版社，2018年7月

ISBN: 978-7-111-60436-5

三遍

Operating System Concepts (9th Edition)

Abraham Silberschatz

John Wiley & Sons; October 10, 2012

参考书：理论部分

- “Operating Systems Principles and Practice”2nd edition, Thomas Anderson，Mike Dahlin, 2014年8月, ISBN-13: 978-0985673529
- 《现代操作系统第4版》塔嫩鲍姆等著，陈向群等译. 机械工业出版社，2017年7月. ISBN: 978-7-111-57369-2
- Operating Systems: Three Easy Pieces. Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau, http://pages.cs.wisc.edu/~remzi/OSTEP/#book-chapters,
- “Operating System:Internals and Design Principles”中文版：电子工业出版社 英文版：清华大学出版社

实验参考书：具体操作系统

- 《Linux内核完全注释》，赵炯，V3.0，2007年9月。
- 《 X86汇编语言：从实模式到保护模式》，李忠，王晓波、余洁，电子工业出版社，2013年1月
- 《Understanding the Linux Kernel(3rd Edition)》，Daniel P. Bovet, Marco Cesati. O'Reilly, 2004

学习资料

- MIT 6.828 Operating Systems Engineering
  - https://pdos.csail.mit.edu/6.828/2017/schedule.html
- RTOS, Linux
- James M's kernel development tutorials
  - x86架构操作系统内核的实现
  - http://wiki.0xffffff.org/ 在线文档
  - http://hurlex.0xffffff.org/ 项目介绍
  - https://github.com/jmolloy/JMTK 源代码

考核方式和评分标准

- 平时成绩（15%）
  - 平时作业和课堂表现(15%)
- 期末考试（65%）
  - 笔试（含上机实验内容5-10%）
- 课程实验（20%）
  - 实验报告（15%）
  - 主题论文（5%）
    - 新技术综述或OS发展历程总结

实验与实验指导材料

- 基本要求：(约30小时左右)
  - https://www.shiyanlou.com
  - 注册，并加入私课，邀请码： 7U489Q53
- 个性化实验(需要投入较多时间)
  - 自行设计或将已有的研究扩展为新的复杂实验
  - 需有明确的预期产出和工作计划
  - 自由结组，人数不超过3人
  - 需与老师讨论并经老师同意
  - 每完成一个达到要求的实验设计，共可获得最终成绩加5分，由组长分配。每位同学累计获得加分不超过5分

主题论文分析报告

- 根据兴趣，自行搜索三年内的操作系统相关主题的论文1篇（长度不少于8页）
- 阅读论文并写1篇不少于2页的A4（小四字体）篇幅的读后感（WORD格式化）。
  - 对论文中技术与所学操作系统关系的评述和展望。（第13周－16周完成）

课程讨论PIAZZA

- piazza.com/buaa.edu.cn/fall2019/os2019
- ACESS CODE： softOS2019
- 选择Operating System on Linux Kernel

复习相关基础知识

- 冯诺依曼体系结构
- CPU的结构，取指与执行过程
- X86的基本架构
- 子程序调用过程（栈的使用）
- 链接与加载

冯诺依曼体系结构图

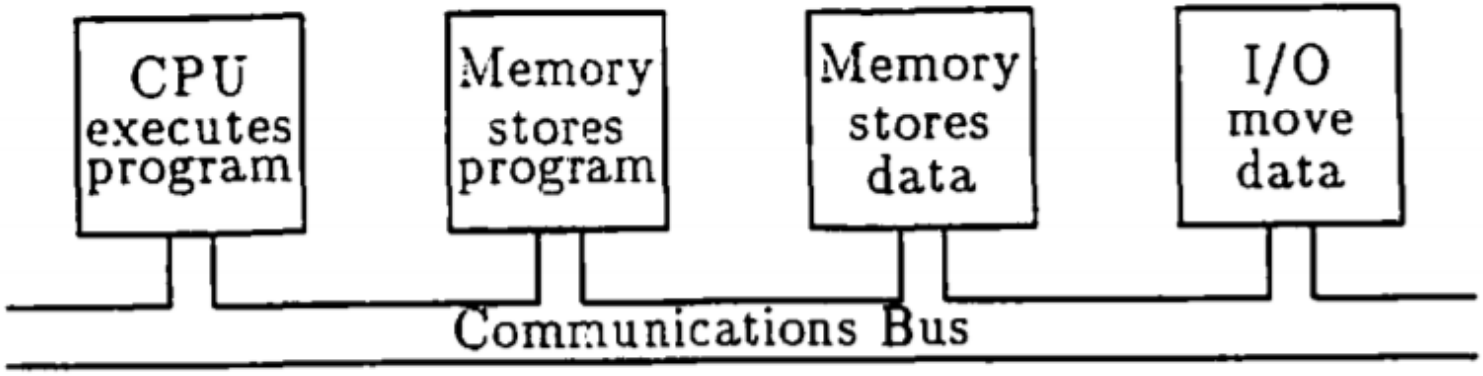
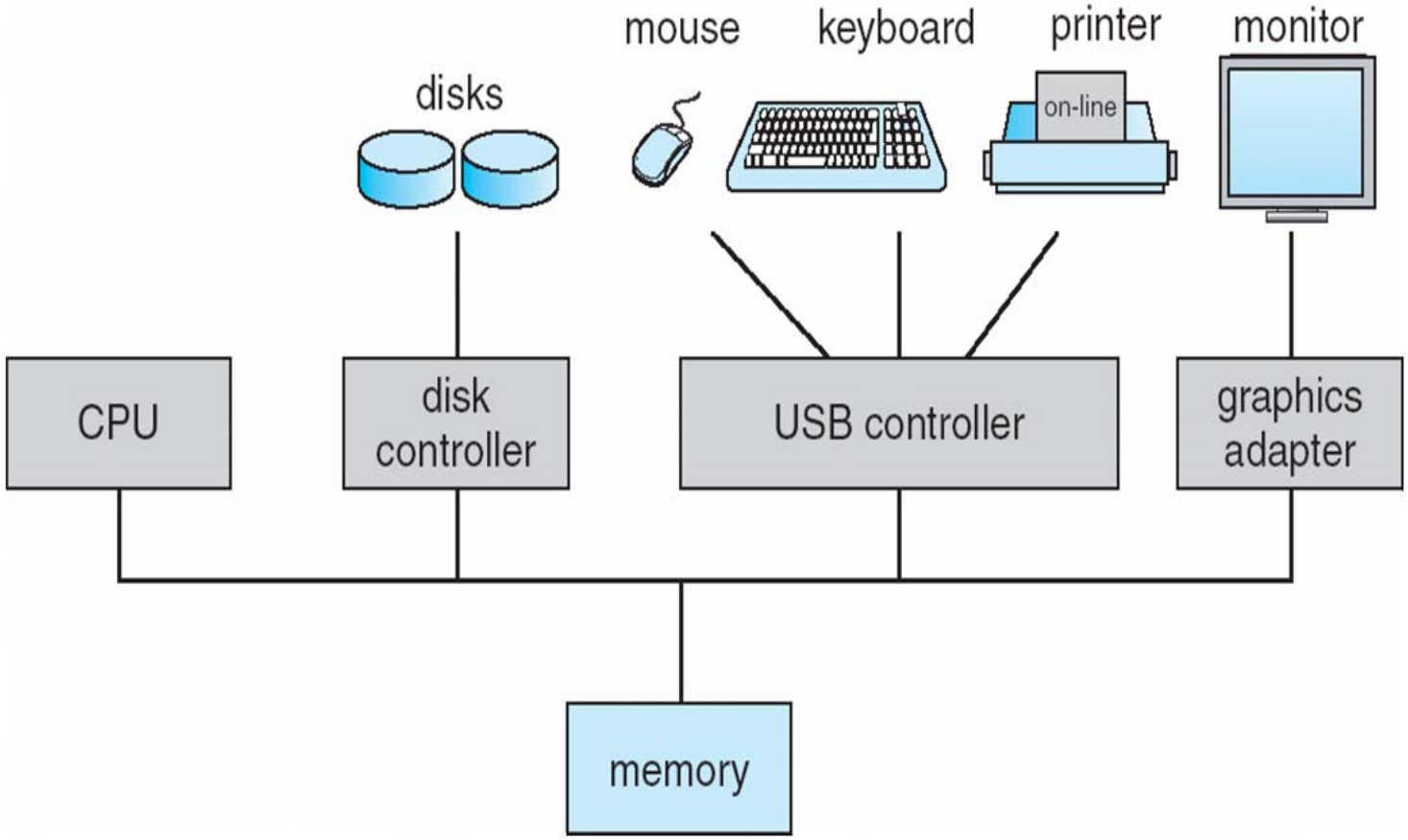
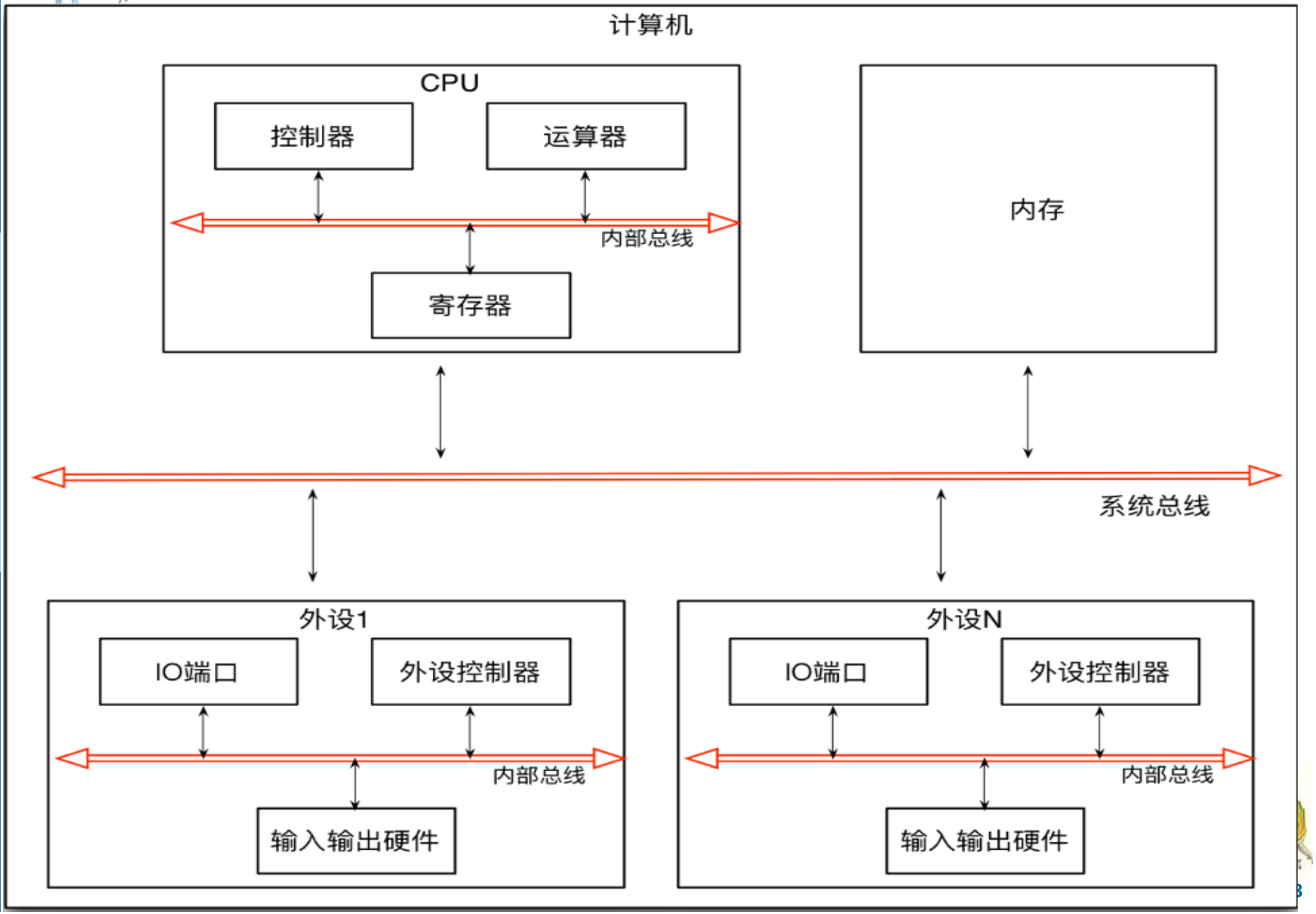


Figure 1: A von Neumann Computer

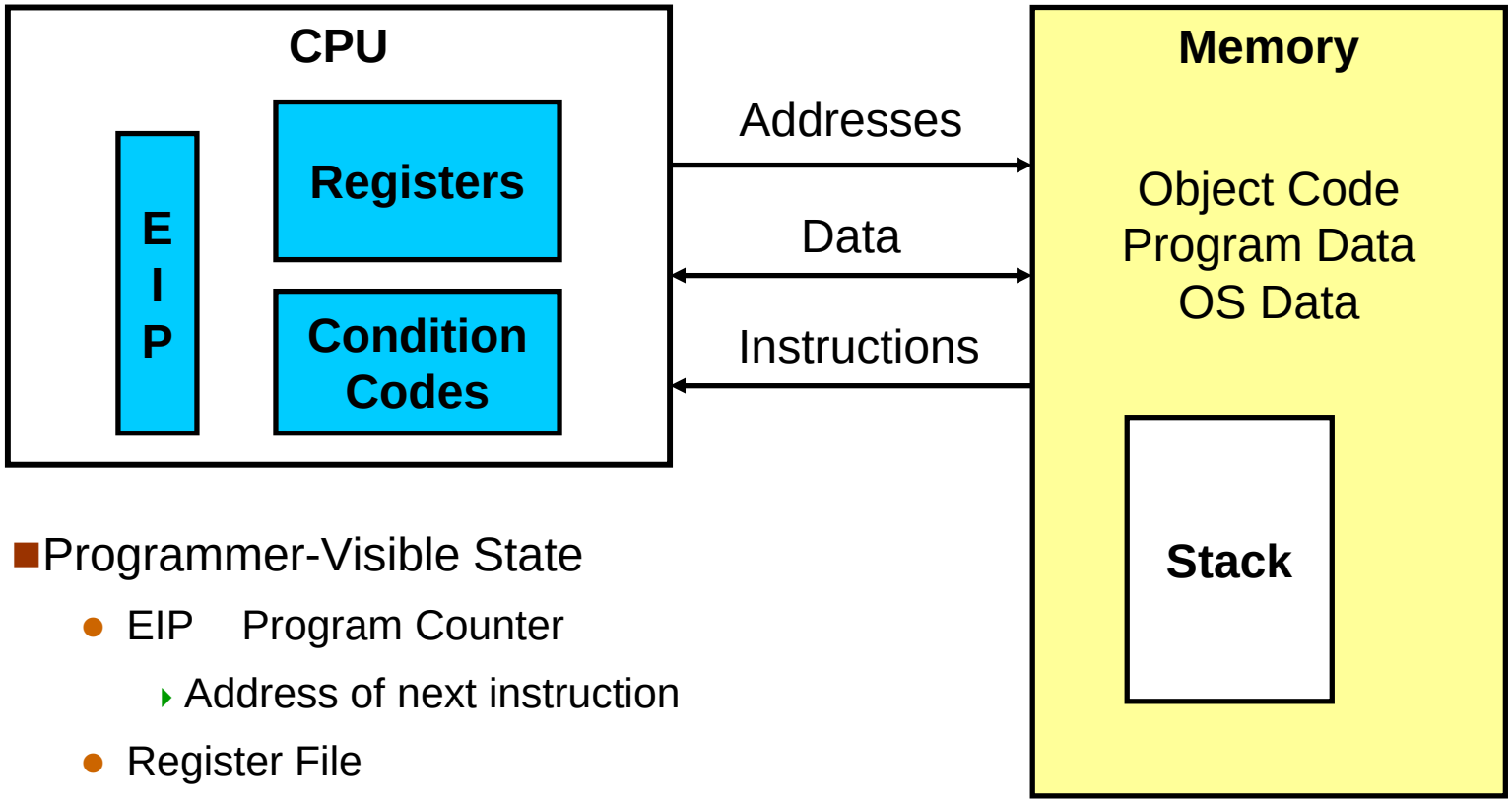
Modern Computer System



硬件组件间的数据“流动”

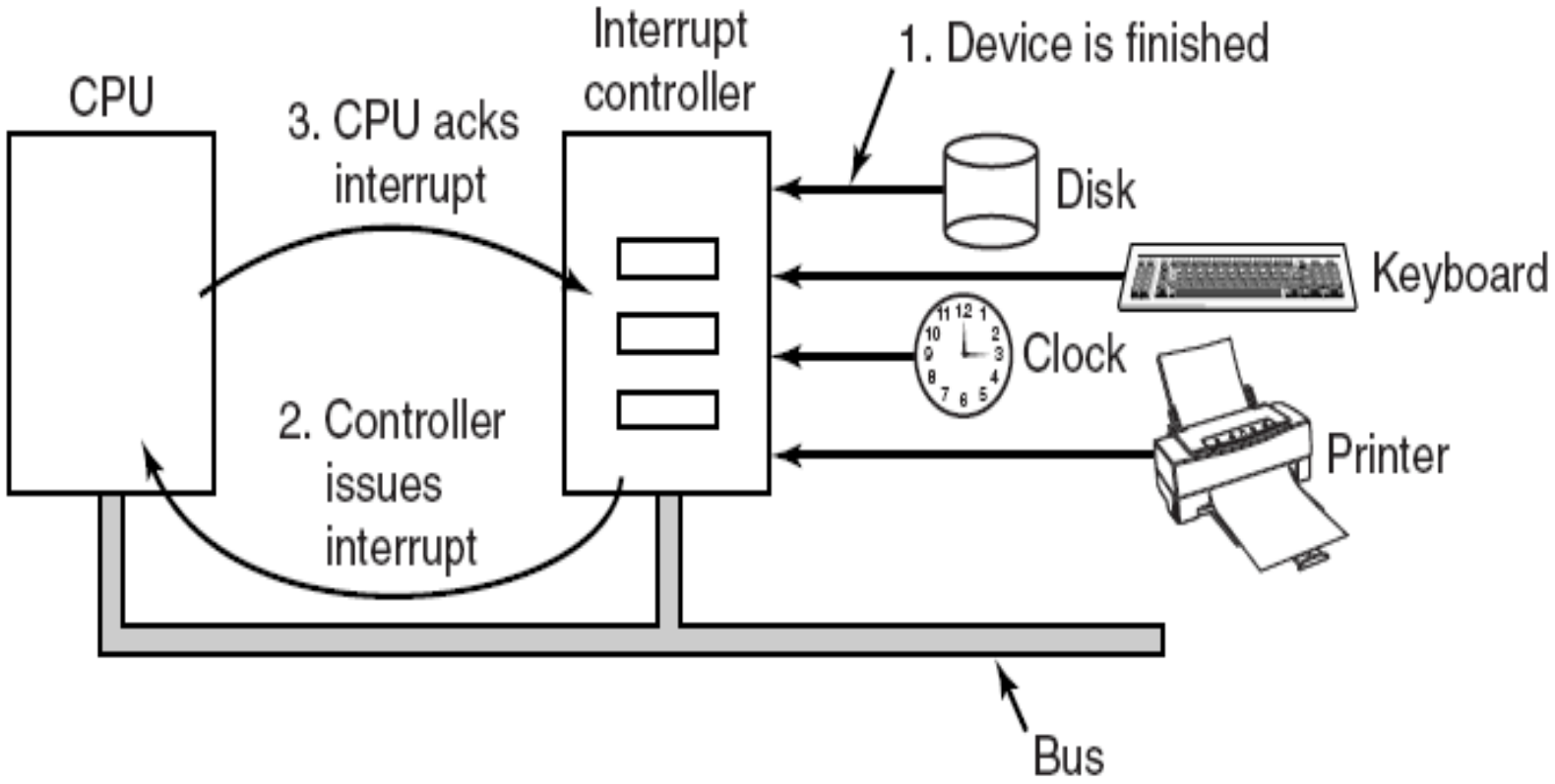

计算机
CPU
控制器
运算器
内部总线
寄存器
内存
系统总线
外设1
IO端口
外设控制器
内部总线
输入输出硬件
外设N
IO端口
外设控制器
内部总线
输入输出硬件

汇编程序员眼中的抽象机器

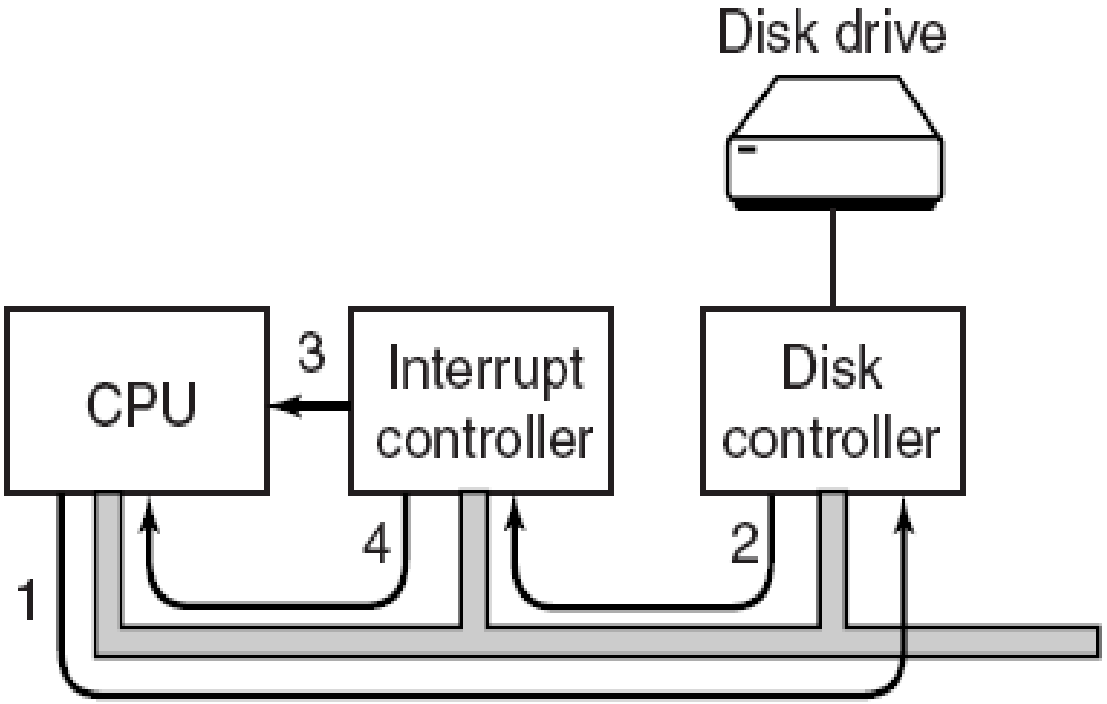


- Programmer-Visible State
  - EIP Program Counter
    - Address of next instruction
  - Register File
    - Heavily used program data
  - Condition Codes
    - Store status information about most recent arithmetic operation
    - Used for conditional branching
- Memory
  - Byte addressable array
  - Code, user data, (some) OS data
  - Includes stack used to support procedures

How an interrupt happens



The steps in an interrupt processing



中断和异常的“跳转”

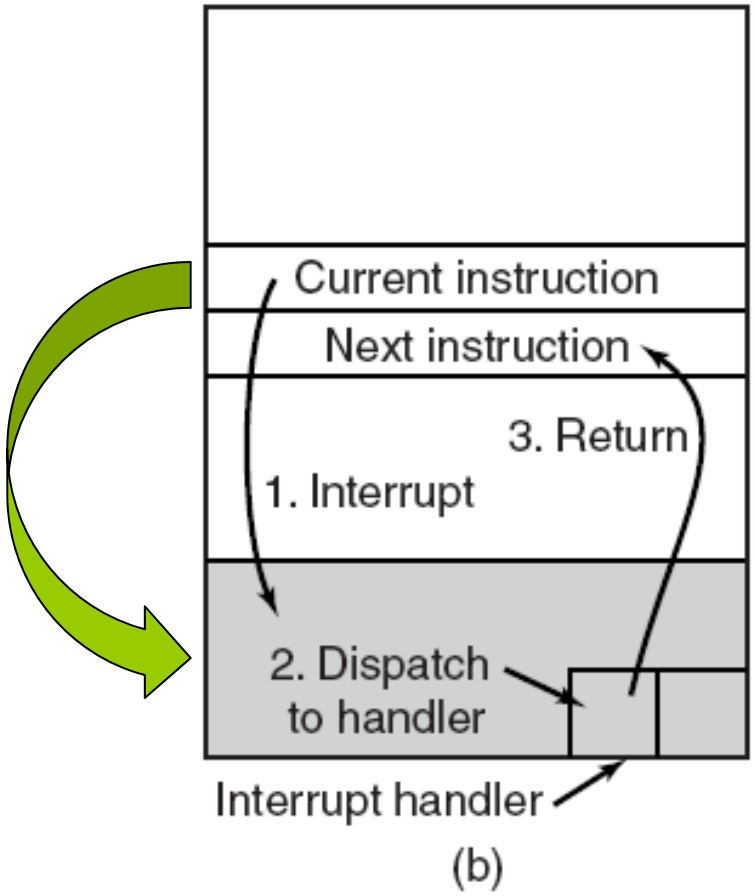
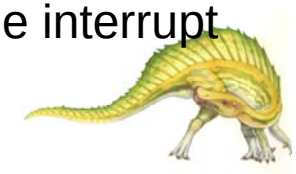
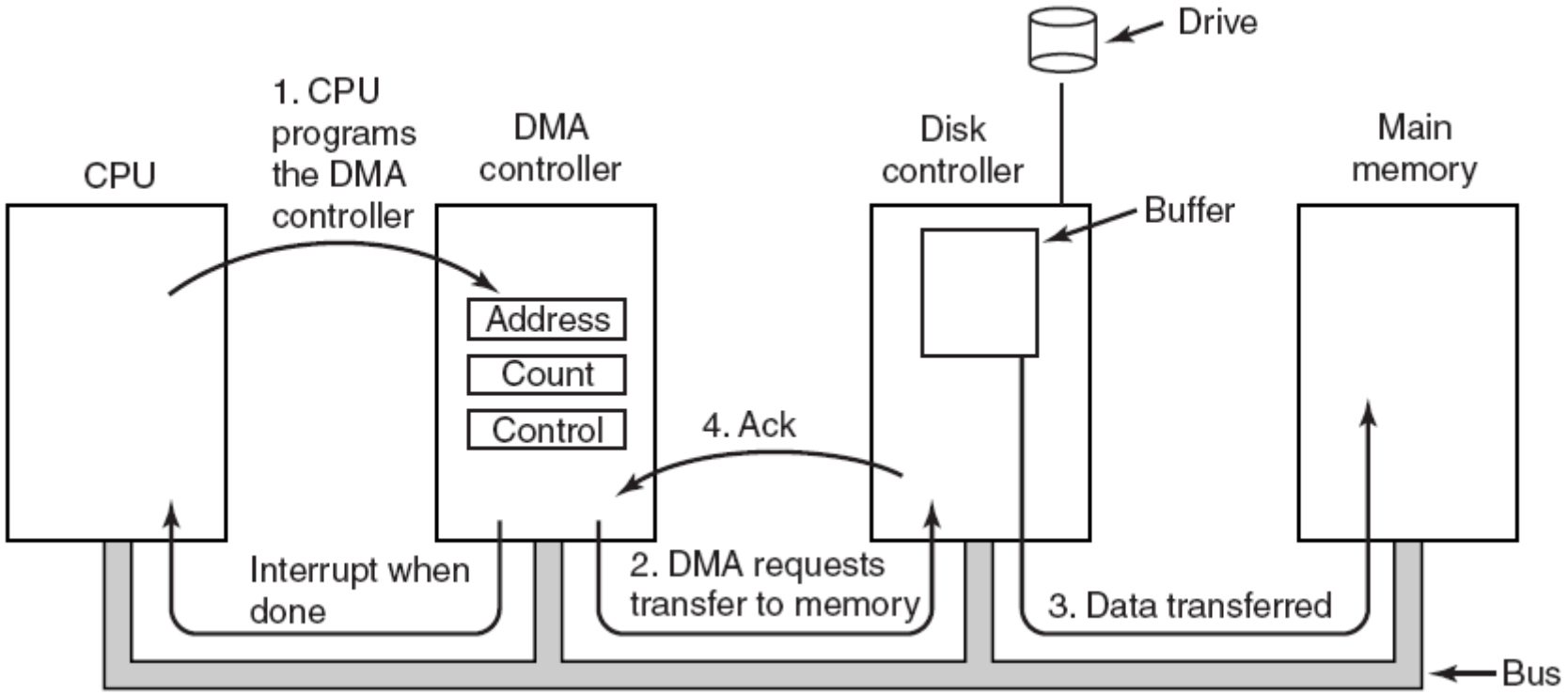


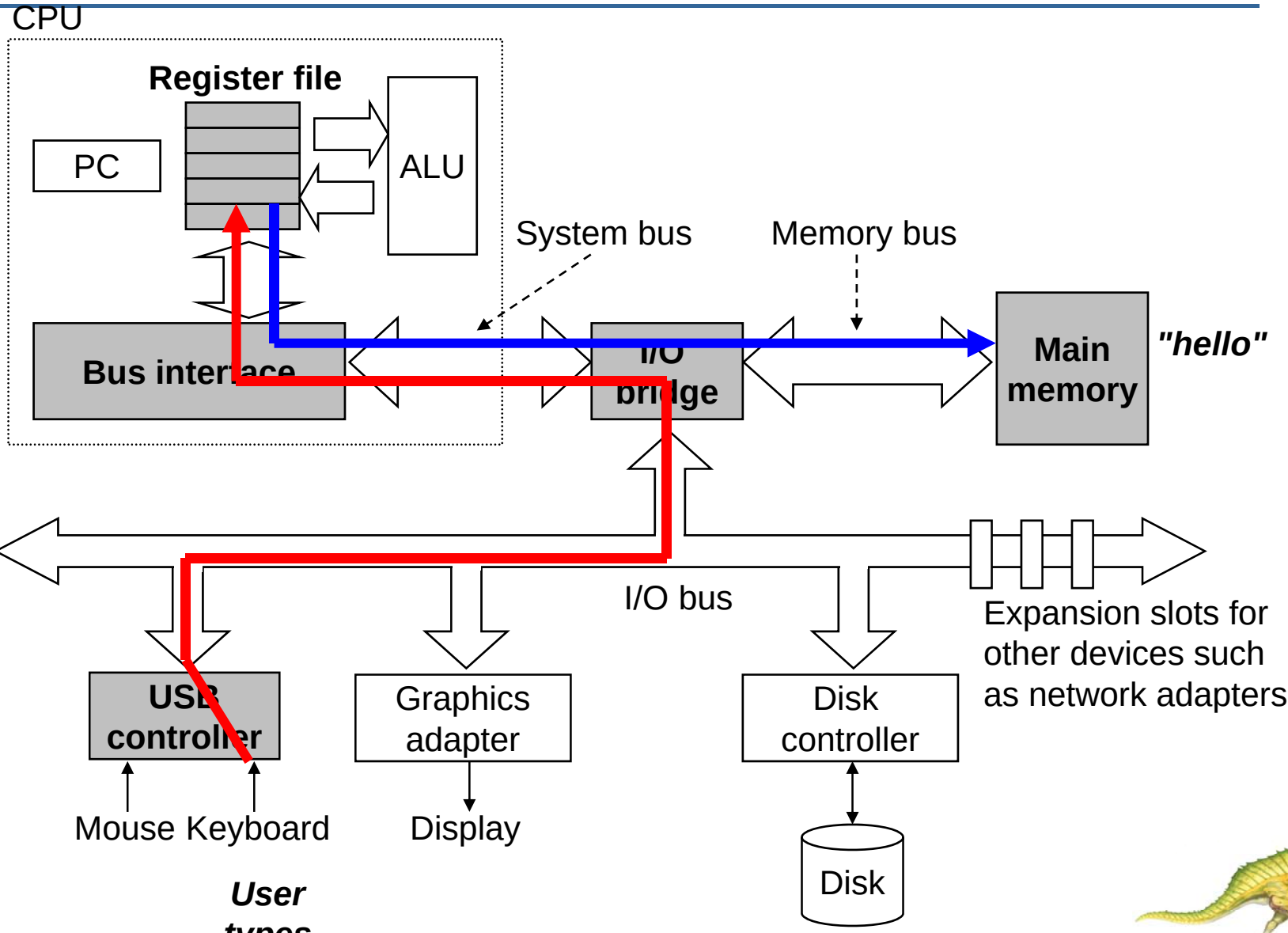
Figure 1-11.(b) Interrupt processing involves taking the interrupt, running the interrupt handler, returning to the user program.



Direct Memory Access (DMA)

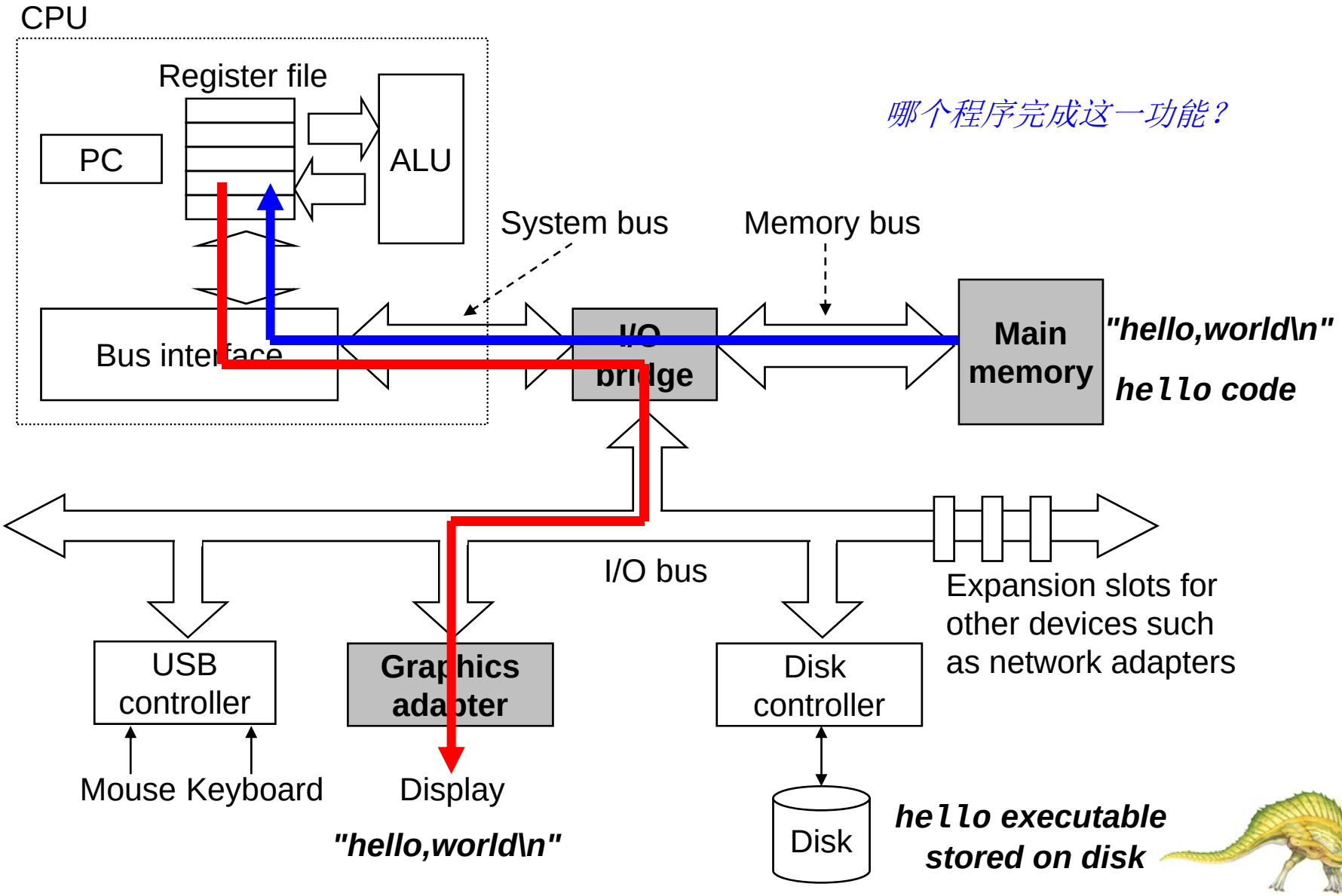


程序的运行过程—输入可执行文件名

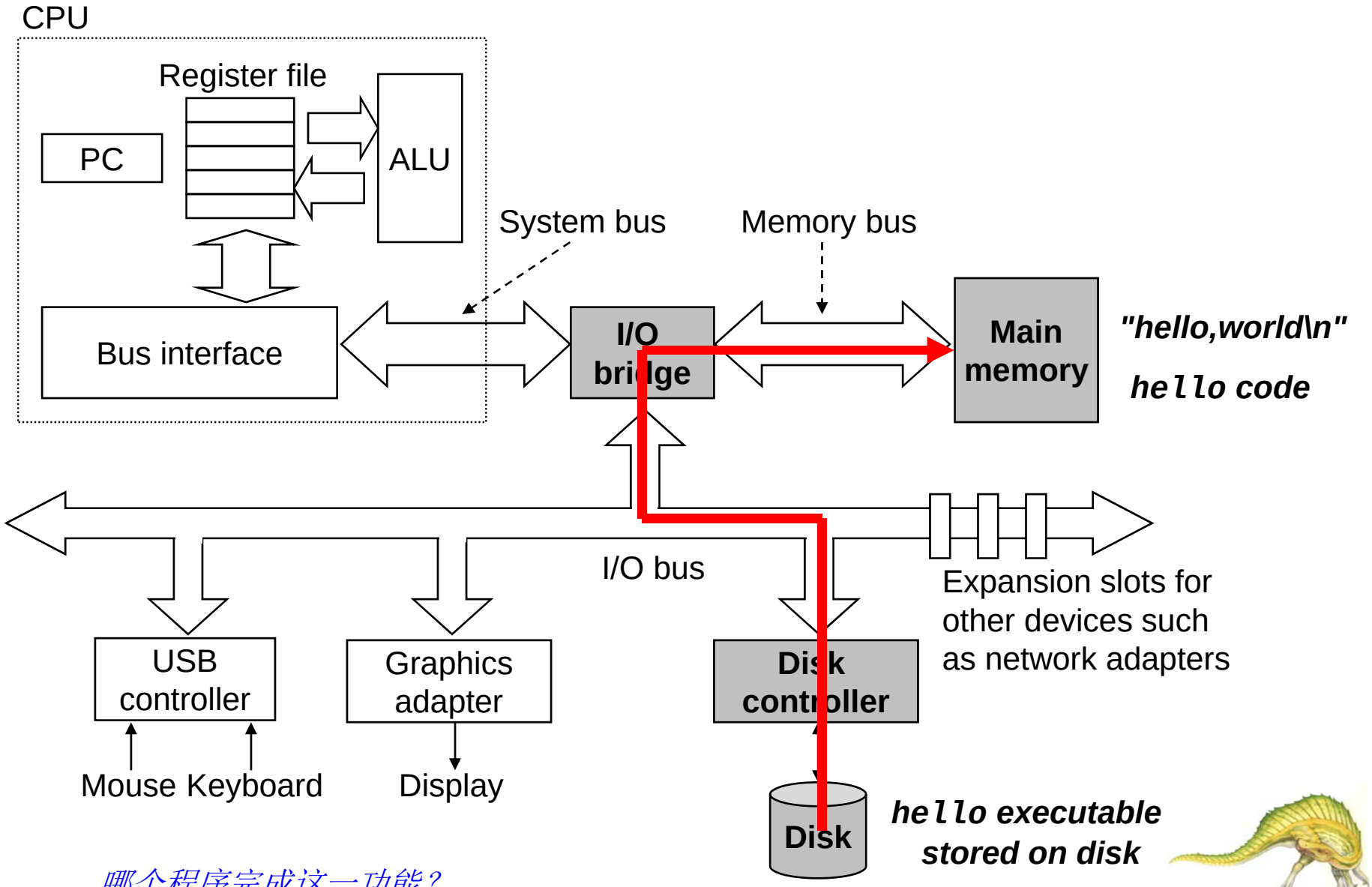


哪个程序完成这一功能？

程序的运行过程—显示“Hello world\n”

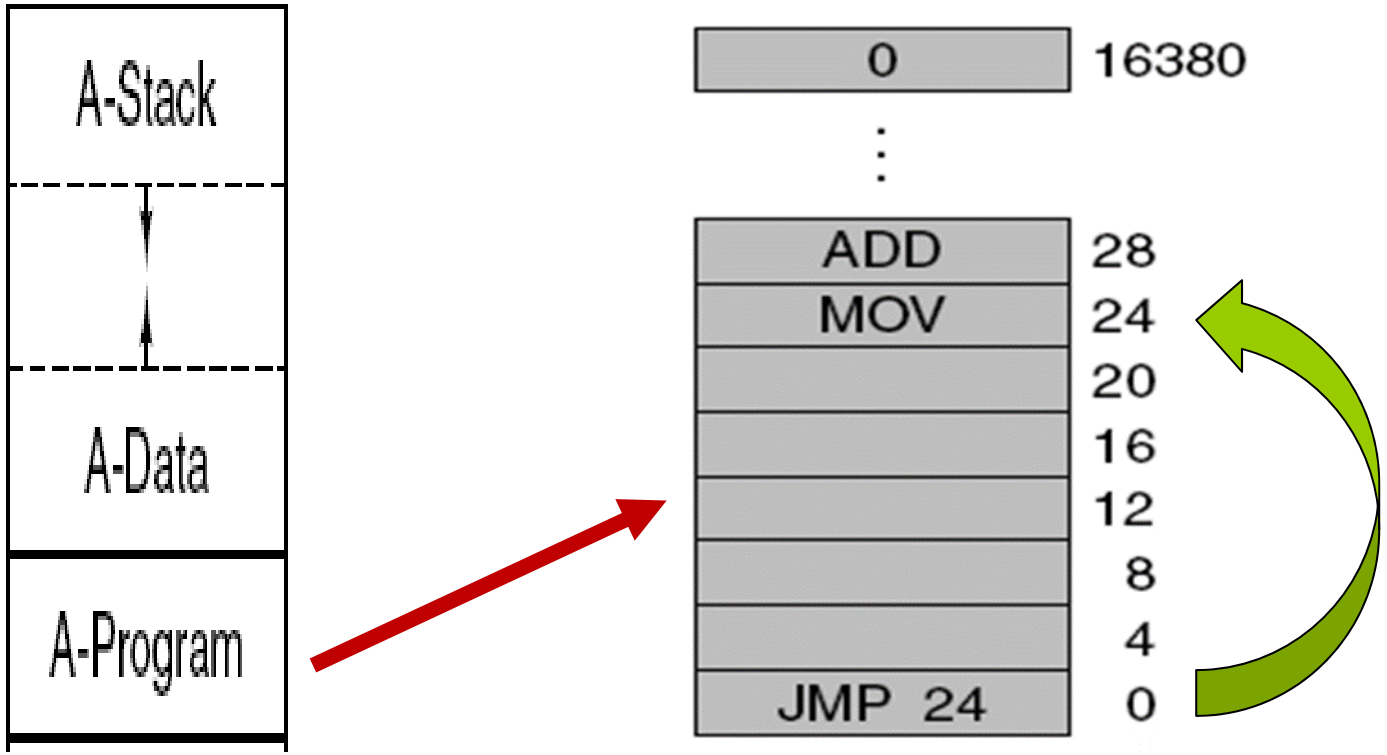


程序的运行过程—载入Hello



哪个程序完成这一功能？

程序与地址



Compiling Into Assembly

- C Code

```
int sum(int x, int y)
{
  int t = x+y;
  return t;
}
```

Generated Assembly

```
_sum:
    pushl %ebp
    movl %esp,%ebp
    movl 12(%ebp),%eax
    addl 8(%ebp),%eax
    movl %ebp,%esp
    popl %ebp
    ret
```

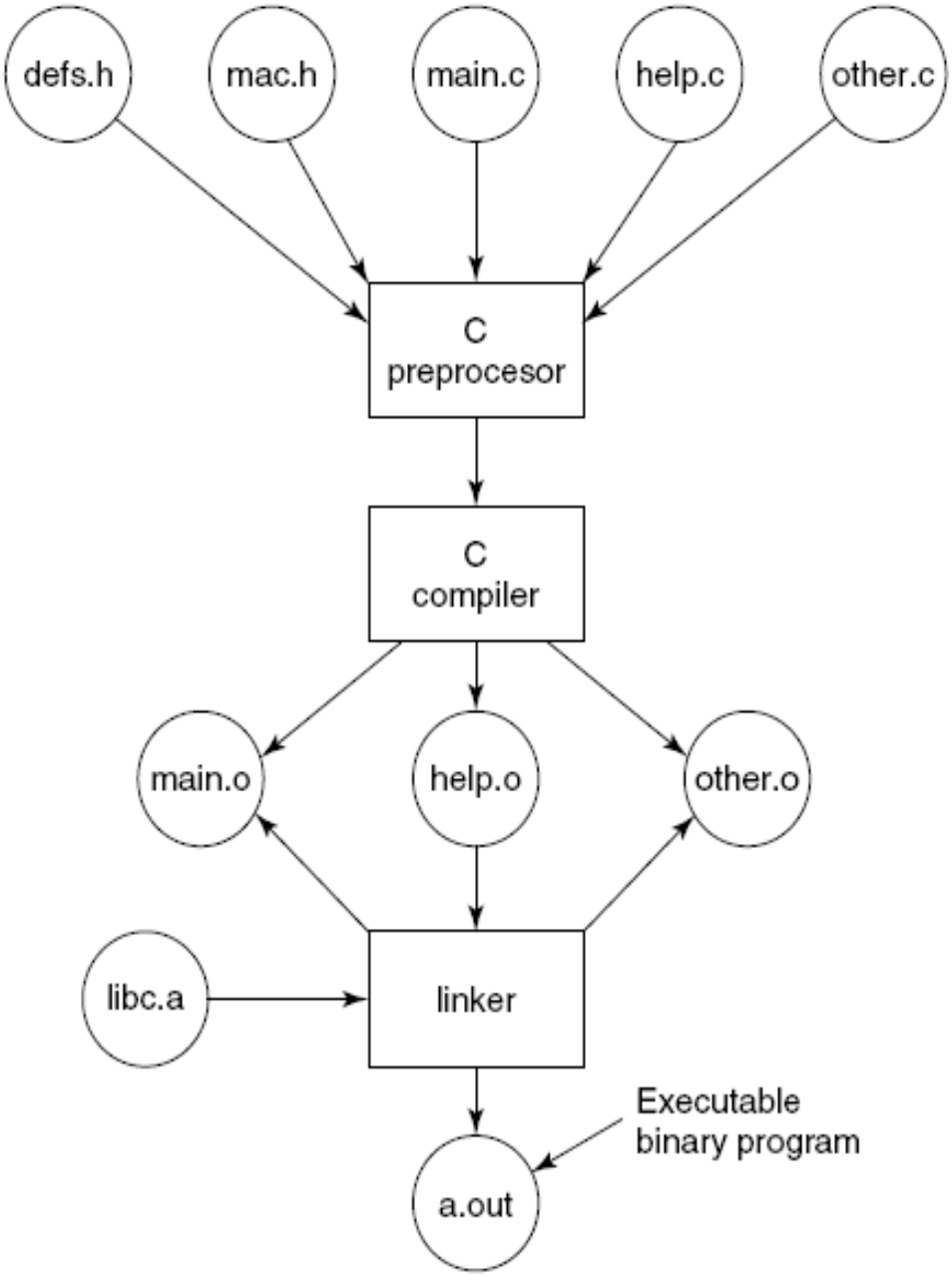
Obtain with command

```
gcc -O -S code.c
```

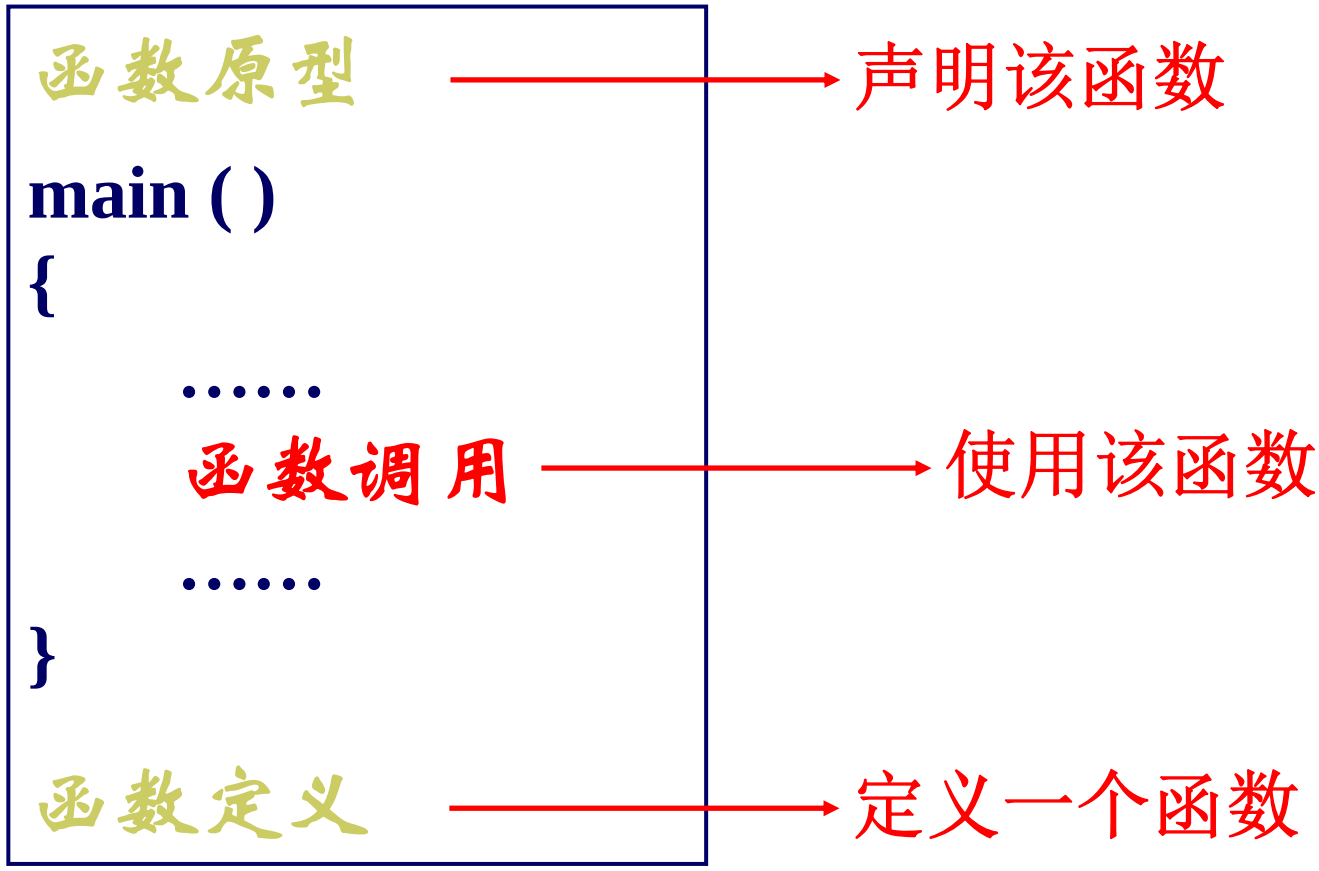
Produces file `code.s`

The World According to C

- The C language
- Header files
- Large programming projects
- The model of run time



C语言的函数

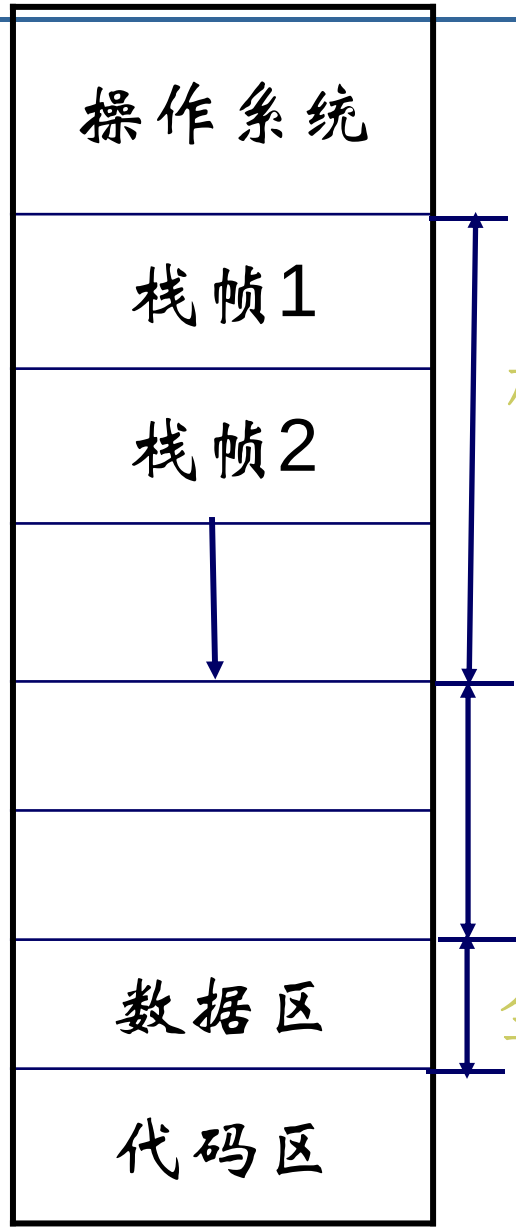


函数的使用模式

函数调用过程

- 当一个函数被调用时:
  - 在内存的栈空间当中为其分配一个栈帧，用来存放该函数的形参和局部变量;
  - 把实参变量的值复制到相应的形参变量;
  - 控制转移到该函数的起始位置;
  - 该函数开始执行;
  - 控制流和返回值返回到函数调用点。

C语言的内存分布




```
int m; //未初始化全局变量
int n=5; //初始化全局变量
char *p = “buaa”;//常量
main()
{
        int l = 5;
        char *q = “buaa”;
        static int i;
        char *pp;

        pp = malloc(l);
}
```

这种栈堆结构的优缺点?

举例

```
#include <stdio.h>
#include <malloc.h>
int a = 0;
char *p1;
main()
{
        int b;
        char s[] = “abc”;
        char *p2 = "123456";
        char *p3;
        static int c = 0;
        p1 = (char *)malloc(20);
        p3 = (char *)malloc(0);
        strcpy(p1, "123456");
}
```

p, & p, * p的区别?

Virtual Machines

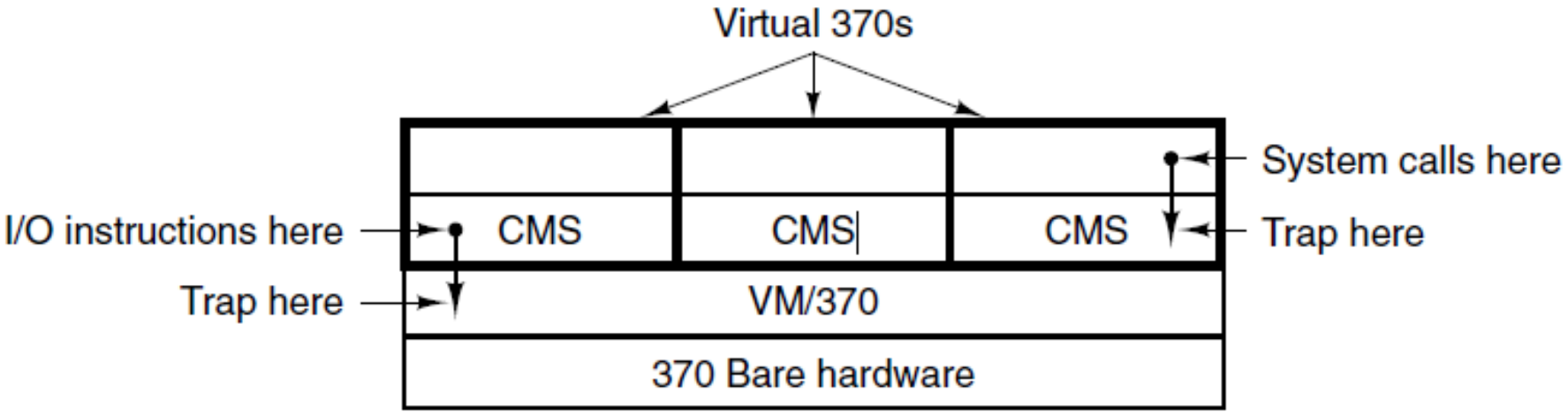


Figure 1-28. The structure of VM/370 with CMS.

Tanenbaum & Bo, Modern Operating Systems:4th ed., (c) 2013 Prentice

Virtual Machines Rediscovered

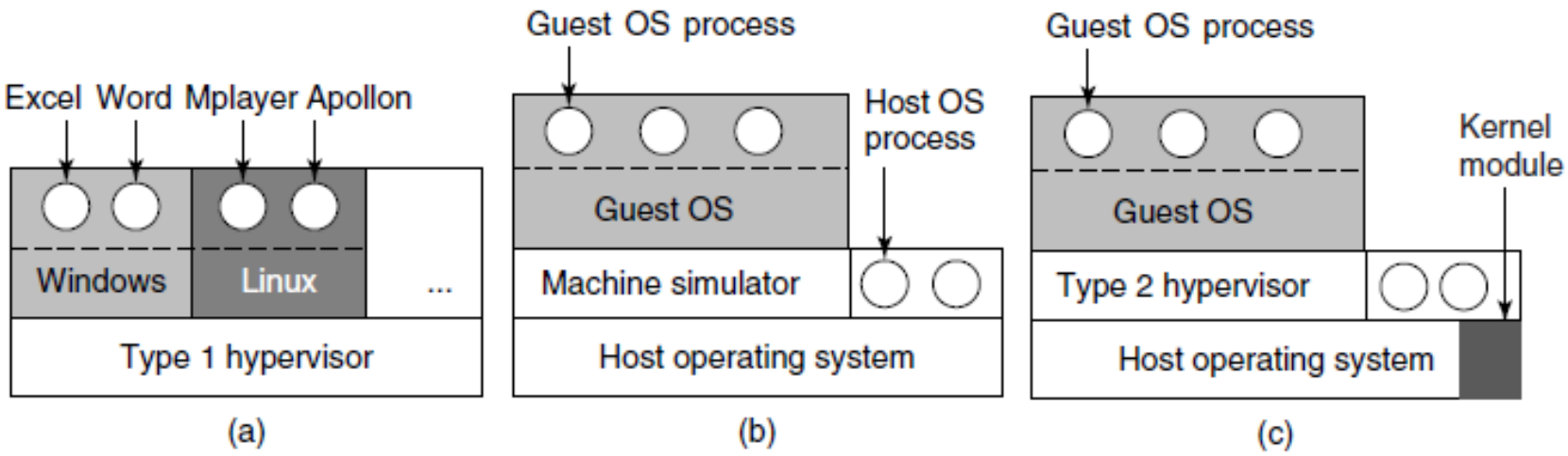
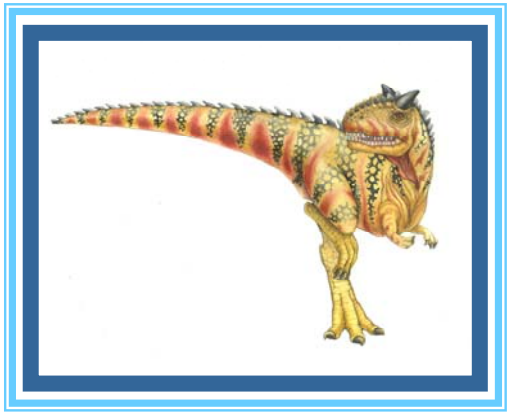


Figure 1-29. (a) A type 1 hypervisor. (b) A pure type 2 hypervisor. (c) A practical type 2 hypervisor.

Tanenbaum & Bo, Modern Operating Systems:4th ed., (c) 2013 Prentice

Chapter 1: Introduction



Chapter 1: Introduction

- What is an Operating System?
- Computer-System Organization
- Operating-System Structure
- Operating-System Operations
- Process Management
- Memory Management
- Storage Management
- Protection and Security
- Kernel Data Structures
- Computing Environments
- Open-Source Operating Systems
- Computer-System Architecture

Objectives

- To describe the basic organization of computer systems
- To provide a grand tour of the major components of operating systems
- To give an overview of the many types of computing environments
- To explore several open-source operating systems

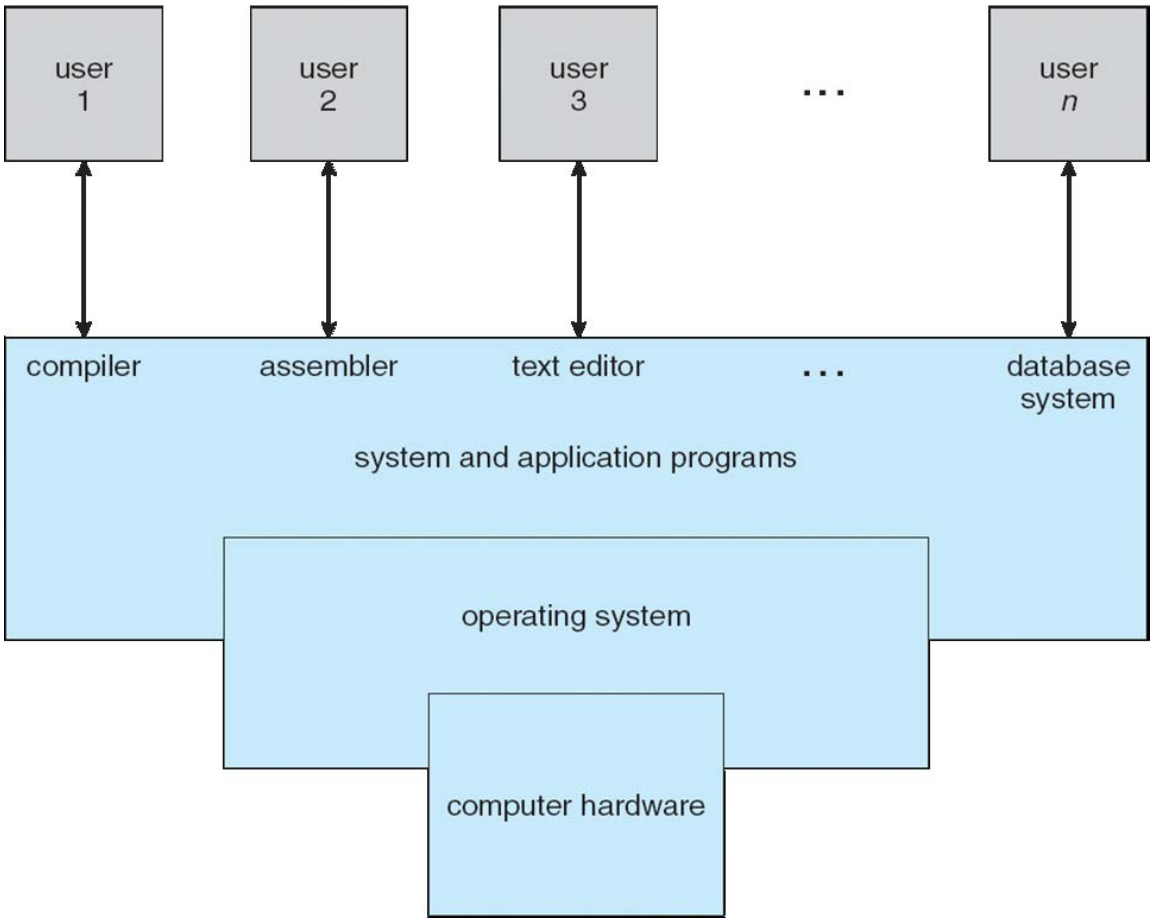
What is an Operating System?

- A program that acts as an intermediary between a user of a computer and the computer hardware
- Operating system goals:
  - Execute user programs and make solving user problems easier
  - Make the computer system convenient to use
  - Use the computer hardware in an efficient manner

Computer System Structure

- Computer system can be divided into four components:
  - Hardware – provides basic computing resources
    - CPU, memory, I/O devices
  - Operating system
    - Controls and coordinates use of hardware among various applications and users
  - Application programs – define the ways in which the system resources are used to solve the computing problems of the users
    - Word processors, compilers, web browsers, database systems, video games
  - Users
    - People, machines, other computers

Four Components of a Computer System



What Operating Systems Do

- The operating system controls the hardware and coordinates its use among the various application programs for the various users.
- We can also view a computer system as consisting of hardware, software, and data.
- The operating system provides the means for proper use of these resources in the operation of the computer system.
- An operating system is similar to a government. Like a government, it performs no useful function by itself. It simply provides an **environment** within which other programs can do useful work.
- To understand more fully the operating system's role, we explore operating systems from two viewpoints:
  - The user
  - The system.

User View

The user's view of the computer varies according to the interface being used

- **Single user computers** (e.g., PC, workstations). Such systems are designed for one user to monopolize its resources. The goal is to maximize the work (or play) that the user is performing. the operating system is designed mostly for **ease of use** and **good performance.**
- **Multi user computers** (e.g., mainframes, computing servers). These users share resources and may exchange information. The operating system in such cases is designed to maximize resource utilization -- to assure that all available CPU time, memory, and I/O are used efficiently and that no individual users takes more than their air share.

User View (Cont.)

- **Handheld computers** (e.g., smartphones and tablets). The user interface for mobile computers generally features a **touch screen.** The systems are resource poor, optimized for usability and battery life.
- **Embedded computers** (e.g., computers in home devices and automobiles) The user interface may have numeric keypads and may turn indicator lights on or off to show status. The operating systems are designed primarily to run without user intervention.

System View

From the computer's point of view, the operating system is the program most intimately involved with the hardware. There are two different views:

- The operating system is a **resource allocator**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use
- The operating systems is a **control program**
  - Controls execution of programs to prevent errors and improper use of the computer

Defining Operating System

No universally accepted definition of **what** an OS:

- Operating systems exist to offer a reasonable way to solve the problem of creating a usable computing system.
- The fundamental goal of computer systems is to execute user programs and to make solving user problems easier.
- Since bare hardware alone is not particularly easy to use, application programs are developed.
  - These programs require certain common operations, such as those controlling the I/O devices.
  - The common functions of controlling and allocating resources are brought together into one piece of software: the **operating system**.

Defining Operating System (Cont.)

No universally accepted definition of what is **part** of the OS:

- A simple viewpoint is that it includes everything a vendor ships when you order the operating system. The features that are included vary greatly across systems:
  - Some systems take up less than a megabyte of space and lack even a full-screen editor,
  - Some systems require gigabytes of space and are based entirely on graphical windowing systems.

Defining Operating System (Cont.)

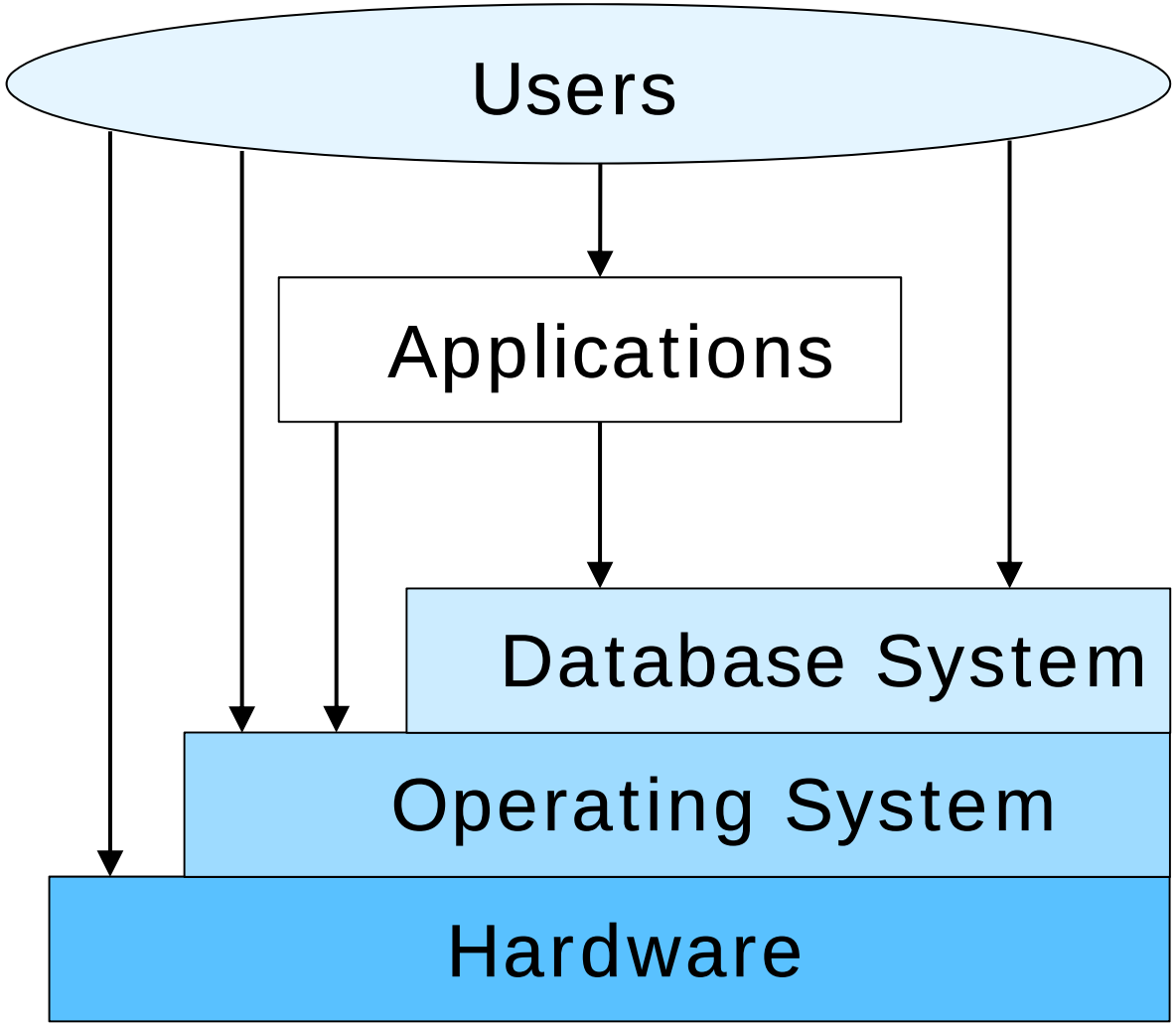
No universally accepted definition of what is **part** of the OS:

- A more common definition, and the one that we usually follow, is that the operating system is the one program running at all times on the computer -- usually called the **kernel**.
- Along with the kernel, there are two other types of programs:
  - System programs, which are associated with the operating system but are not necessarily part of the kernel.
  - Application programs, which include all programs not associated with the operation of the system.

Defining Operating System (Cont.)

- The emergence of mobile devices, have resulted in an increase in the number of features that constituting the operating system.
- Mobile operating systems often include not only a core kernel but also **middleware** -- a set of software frameworks that provide additional services to application developers.
- For example, each of the two most prominent mobile operating systems -- Apple's iOS and Google's Android -- feature a core kernel along with middleware that supports databases, multimedia, and graphics (to name only a few).

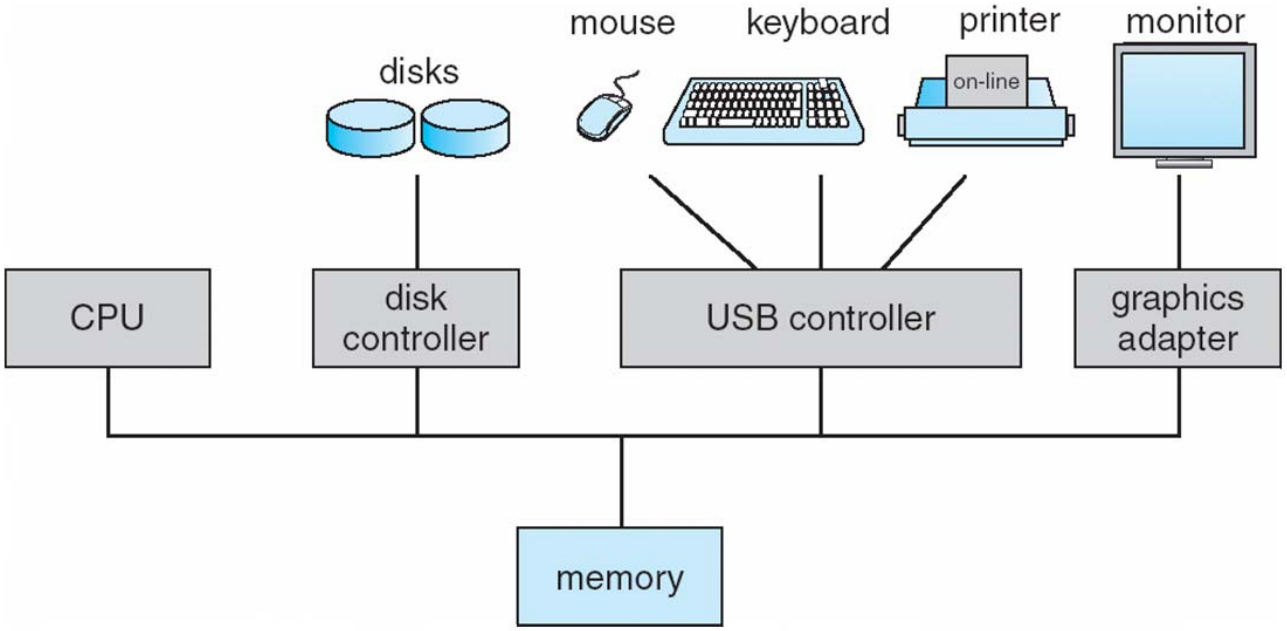
Evolution of Computer Systems



Computer-System Organization

- A modern general-purpose computer system consists of one or more CPUs and a number of device controllers connected through a common bus that provides access to shared memory.
- Each device controller is in charge of a specific type of device (for example, disk drives, audio devices, or video displays). Each device controller has a local buffer.
- CPU moves data from/to main memory to/from local buffers.
- The CPU and the device controllers can execute in parallel, competing for memory cycles. To ensure orderly access to the shared memory, a memory controller synchronizes access to the memory.

Modern Computer System



Computer Startup

- **Bootstrap program** is loaded at power-up or reboot
  - Typically stored in ROM or EPROM, generally known as **firmware**
  - Initializes all aspects of system
  - Loads operating system kernel and starts execution

Computer-System Operation

- Once the kernel is loaded and executing, it can start providing services to the system and its users.
- Some services are provided outside of the kernel, by system programs that are loaded into memory at boot time to become **system processes**, or **system daemons** that run the entire time the kernel is running.
- On UNIX, the first system process is **init** and it starts many other daemons. Once this phase is complete, the system is fully booted, and the system waits for some event to occur.
- The occurrence of an event is usually signaled by an **interrupt.**

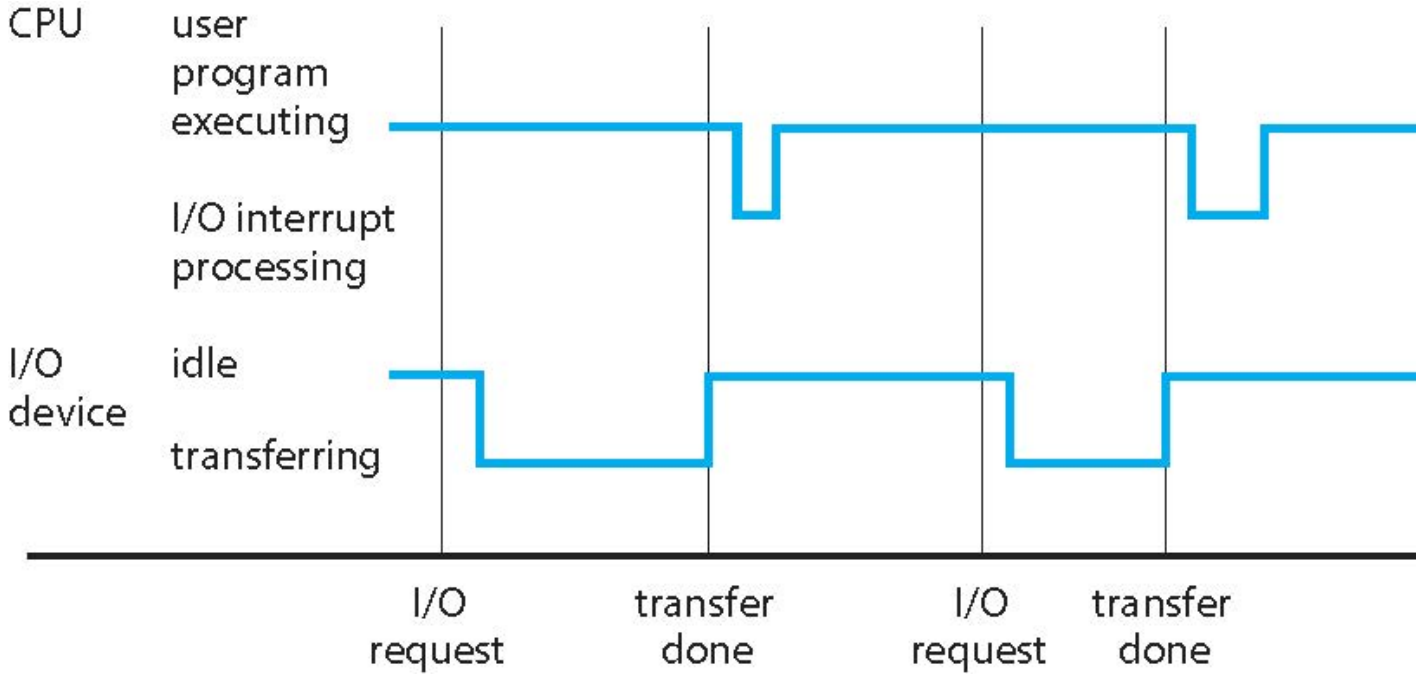
Interrupts

- There are two types of interrupts:
  - **Hardware** -- a device may trigger an interrupt by sending a signal to the CPU, usually by way of the system bus.
  - **Software** -- a program may trigger an interrupt by executing a special operation called a **system call.**
- A software-generated interrupt (sometimes called **trap** or **exception**) is caused either by an error (e.g., divide by zero) or a user request (e.g., an I/O request).
- An operating system is **interrupt driven.**

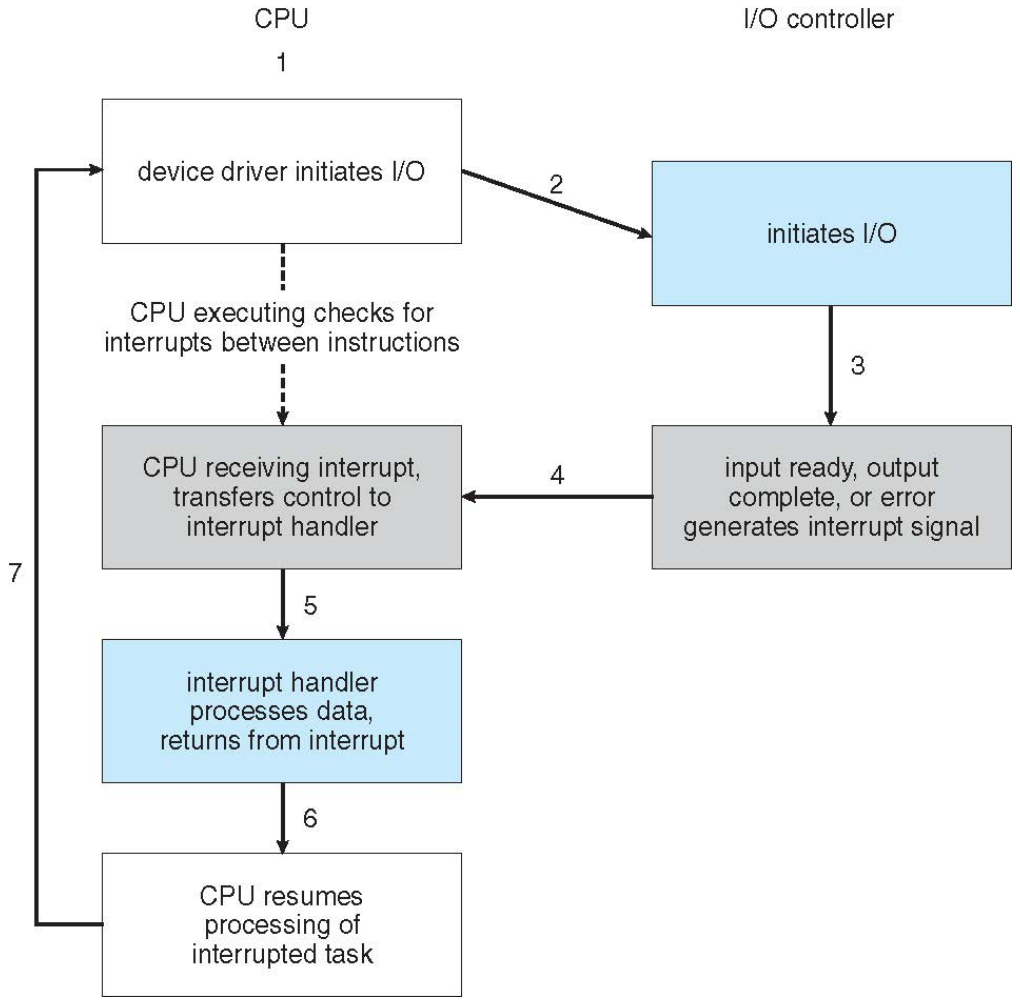
Common Functions of Interrupts

- When an interrupt occurs, the operating system preserves the state of the CPU by storing the registers and the program counter
- Determines which type of interrupt has occurred and transfers control to the interrupt-service routine.
- An interrupt-service routine is a collection of routines (modules), each of which is responsible for handling one particular interrupt (e.g., from a printer, from a disk)
- The transfer is generally through the **interrupt vector**, which contains the addresses of all the service routines
- Interrupt architecture must save the address of the interrupted instruction.

Interrupt Timeline



Interrupt-driven I/O cycle.



Intel Pentium processor event-vector table

vector number	description
0	divide error
1	debug exception
2	null interrupt
3	breakpoint
4	INTO-detected overflow
5	bound range exception
6	invalid opcode
7	device not available
8	double fault
9	coprocessor segment overrun (reserved)
10	invalid task state segment
11	segment not present
12	stack fault
13	general protection
14	page fault
15	(Intel reserved, do not use)
16	floating-point error
17	alignment check
18	machine check
19–31	(Intel reserved, do not use)
32–255	maskable interrupts

Storage Structure

- Main memory – the only large storage media that the CPU can access directly
  - **Random access**
  - Typically **volatile**
- Secondary storage – extension of main memory that provides large **nonvolatile** storage capacity
  - Hard disks – rigid metal or glass platters covered with magnetic recording material
    - Disk surface is logically divided into **tracks**, which are subdivided into **sectors**
    - The **disk controller** determines the logical interaction between the device and the computer
  - **Solid-state disks** – faster than hard disks, nonvolatile
    - Various technologies
    - Becoming more popular
- Tertiary storage

Storage Definition

- The basic unit of computer storage is the **bit**. A bit can contain one of two values, 0 and 1. All other storage in a computer is based on collections of bits.
- A **byte** is 8 bits, and on most computers it is the smallest convenient chunk of storage.
- A less common term is **word**, which is a given computer architecture's native unit of data. A word is made up of one or more bytes.

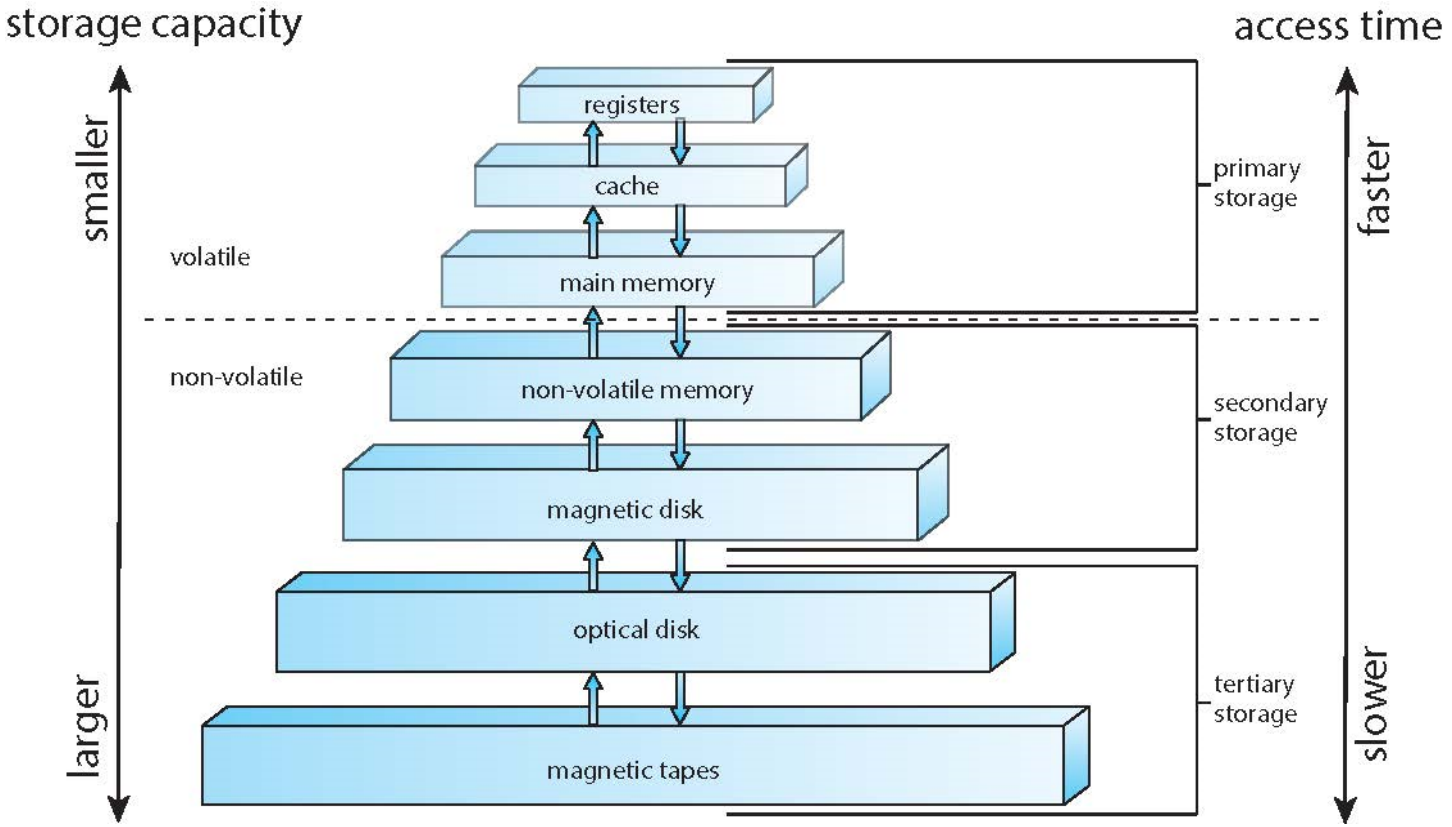
Storage Definition (Cont.)

- Computer storage, along with most computer throughput, is generally measured and manipulated in bytes and collections of bytes.
  - A **kilobyte**, or **KB**, is 1,024 bytes
  - a **megabyte**, or **MB**, is $1,024^2$ bytes
  - a **gigabyte**, or **GB**, is $1,024^3$ bytes
  - a **terabyte**, or **TB**, is $1,024^4$ bytes
  - a **petabyte**, or **PB**, is $1,024^5$ bytes
  - exabyte, zettabyte, yottabyte
- Computer manufacturers often round off these numbers and say that a megabyte is 1 million bytes and a gigabyte is 1 billion bytes. Networking measurements are an exception to this general rule; they are given in bits (because networks move data a bit at a time).

Storage Hierarchy

- Storage systems organized in hierarchy
  - Speed
  - Cost
  - Volatility
- **Caching** – copying information from “slow” storage into faster storage system;
  - Main memory can be viewed as a cache for secondary storage
- **Device Driver** for each device controller to manage I/O
  - Provides uniform interface between controller and kernel

Storage-device hierarchy



I/O Structure

- A general-purpose computer system consists of CPUs and multiple device controllers that are connected through a common bus.
- Each device controller is in charge of a specific type of device. More than one device may be attached. For instance, seven or more devices can be attached to the **small computer-systems interface (**SCSI) controller.
- A device controller maintains some local buffer storage and a set of special-purpose registers.
- The device controller is responsible for moving the data between the peripheral devices that it controls and its local buffer storage.
- Typically, operating systems have a **device driver** for each device controller. This device driver understands the device controller and provides the rest of the operating system with a uniform interface to the device.

I/O Structure (Cont.)

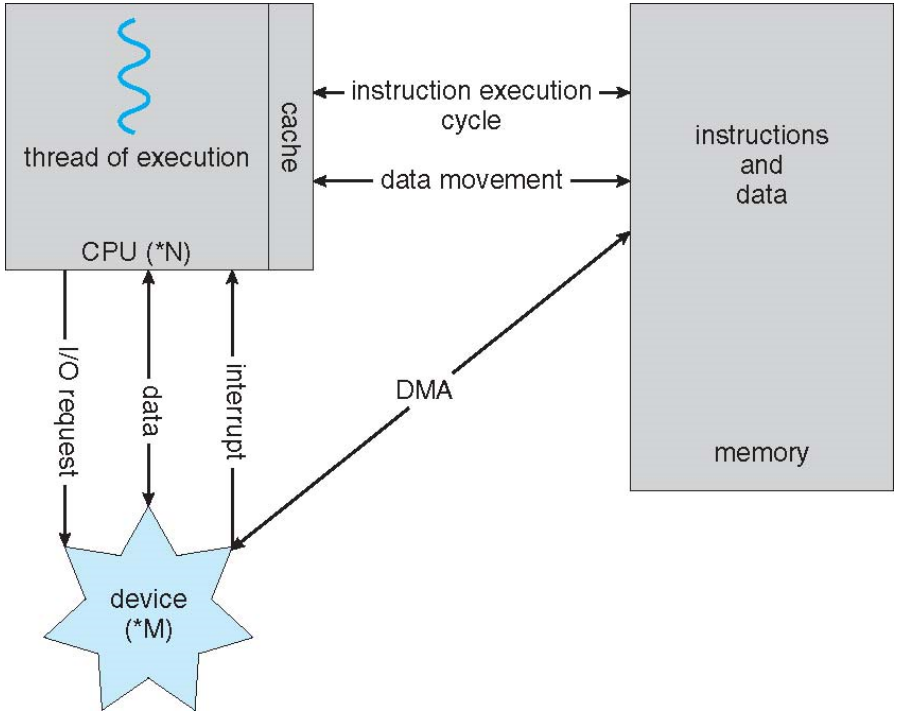
- To start an I/O operation, the device driver loads the appropriate registers within the device controller.
- The device controller, in turn, examines the contents of these registers to determine what action to take (such as “read” a character from the keyboard).
- The controller starts the transfer of data from the device to its local buffer. Once the transfer of data is complete, the device controller informs the device driver via an interrupt that it has finished its operation.
- The device driver then returns control to the operating system, possibly returning the data or a pointer to the data if the operation was a read.
- For other operations, the device driver returns status information.

Direct Memory Access Structure

- Interrupt-driven I/O is fine for moving small amounts of data but can produce high overhead when used for bulk data movement such as disk I/O.
- To solve this problem, **direct memory access** (DMA) is used.
  - After setting up buffers, pointers, and counters for the I/O device, the device controller transfers an entire block of data directly to or from its own buffer storage to memory, with no intervention by the CPU.
  - Only one interrupt is generated per block, to tell the device driver that the operation has completed. While the device controller s performing these operations, the CPU is available to accomplish other work.
- Some high-end systems use switch rather than bus architecture. On these systems, multiple components can talk to other components concurrently, rather than competing for cycles on a shared bus. In this case, DMA is even more effective. The figure in next slide shows the interplay of all components of a computer system.

How a Modern Computer Works

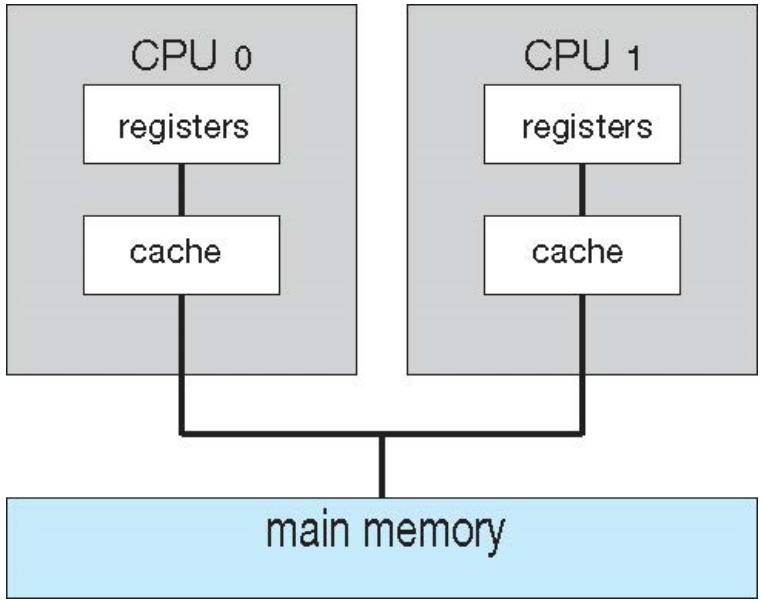
A von Neumann architecture and a depiction of the interplay of all components of a computer system.



Computer-System Architecture

- **Single general-purpose processor**
  - Most systems have special-purpose processors as well
- **Multiprocessors** systems growing in use and importance
  - Also known as **parallel systems**, **tightly-coupled systems**
  - Advantages include:
    - **Increased throughput**
    - **Economy of scale**
    - **Increased reliability** – graceful-degradation/fault-tolerance
  - Two types:
    - **Symmetric Multiprocessing** – each processor performs all tasks
    - **Asymmetric Multiprocessing** – each processor is assigned a specific task.

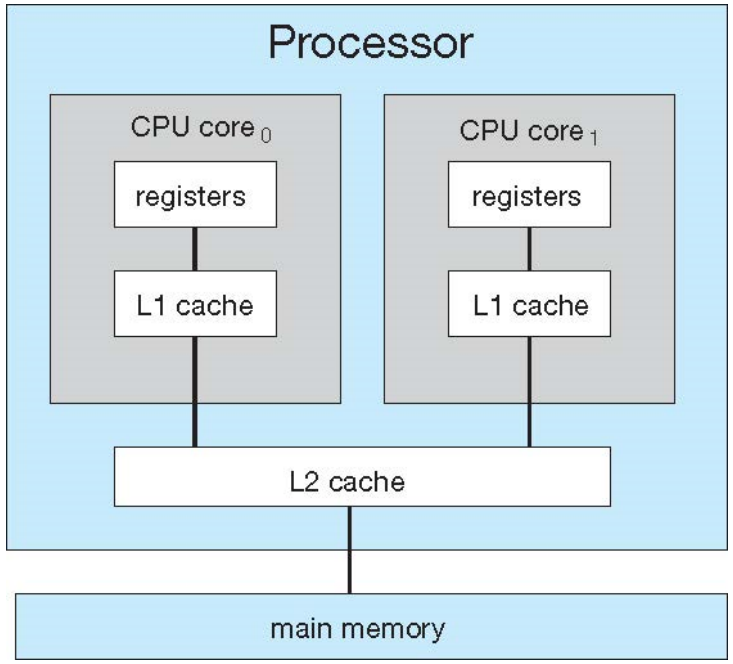
Symmetric Multiprocessing Architecture



Multicore Systems

- Most CPU design now includes multiple computing cores on a single chip. Such multiprocessor systems are termed **multicore**.
- Multicore systems can be more efficient than multiple chips with single cores because:
  - On-chip communication is faster than between-chip communication.
  - One chip with multiple cores uses significantly less power than multiple single-core chips, an important issue for laptops as well as mobile devices.
- Note -- while multicore systems are multiprocessor systems, not all multiprocessor systems are multicore.

A dual-core with two cores placed on the same chip

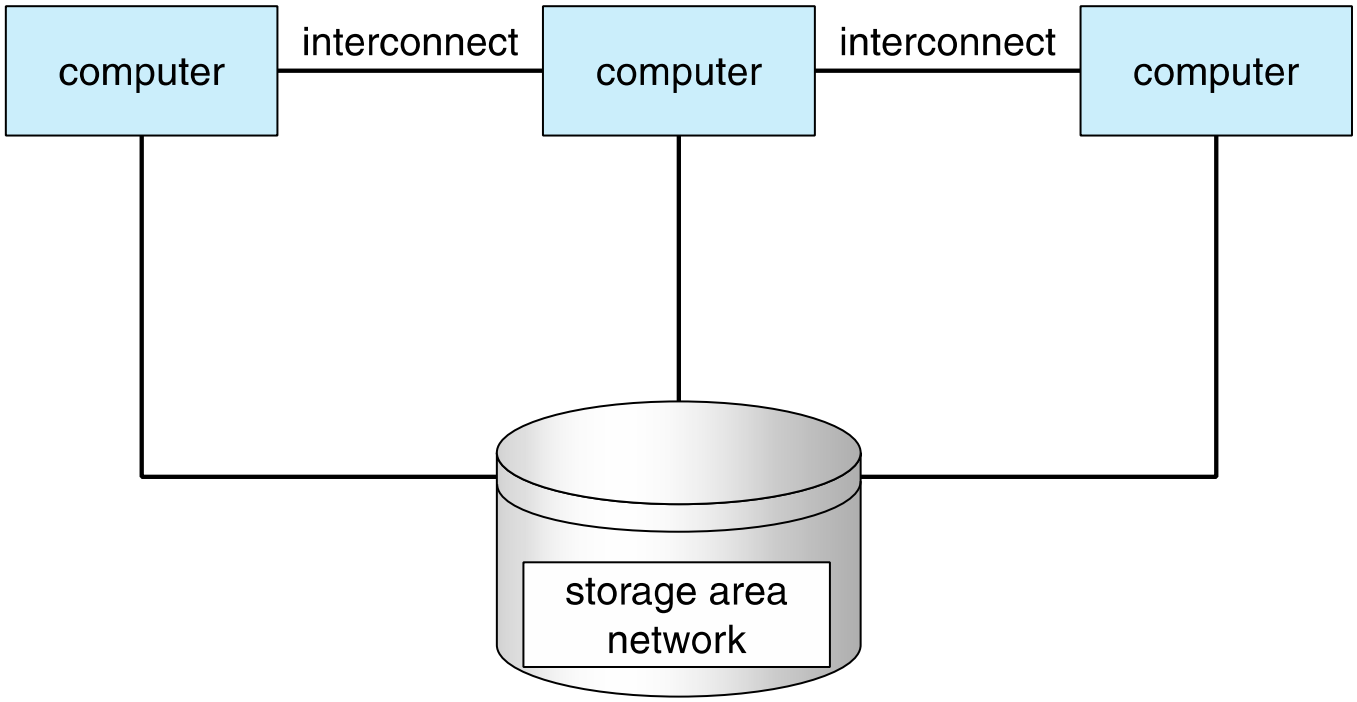


Clustered Systems

Like multiprocessor systems, but multiple systems working together

- Usually sharing storage via a **storage-area network (SAN)**
- Provides a **high-availability** service which survives failures
  - **Asymmetric clustering** has one machine in hot-standby mode
  - **Symmetric clustering** has multiple nodes running applications, monitoring each other
- Some clusters are for **high-performance computing (HPC)**
  - Applications must be written to use **parallelization**
- Some have **distributed lock manager** (**DLM**) to avoid conflicting operations

Clustered Systems



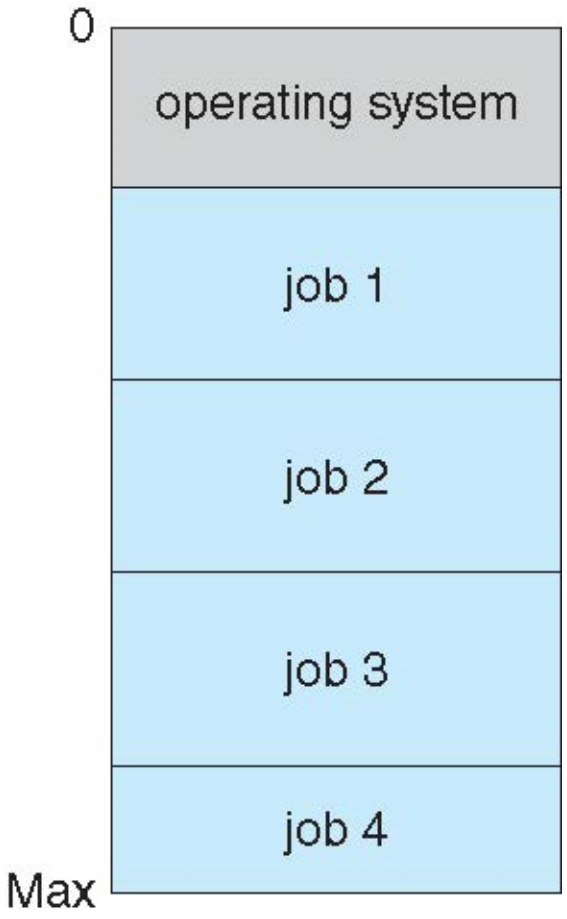
Multiprogrammed System

- Single user cannot keep CPU and I/O devices busy at all times
- **Multiprogramming** organizes jobs (code and data) so CPU always has one to execute
- A subset of total jobs in system is kept in memory
- Batch systems:
  - One job selected and run via **job scheduling**
  - When it has to wait (for I/O for example), OS switches to another job
- Timesharing systems:
  - Logical extension of batch systems -- CPU switches jobs so frequently that users can interact with each job while it is running, creating **interactive** computing

Timesharing Systems

- Timesharing is also referred to as **multitasking.**
- **Response time** should be < 1 second
- Each user has at least one program executing in memory. Such a program is referred to as a **process**
- If several processes are ready to run at the same time, we need to have **CPU scheduling.**
- If processes do not fit in memory, **swapping** moves them in and out to run
- **Virtual memory** allows execution of processes not completely in memory

Memory Layout for Multiprogrammed System

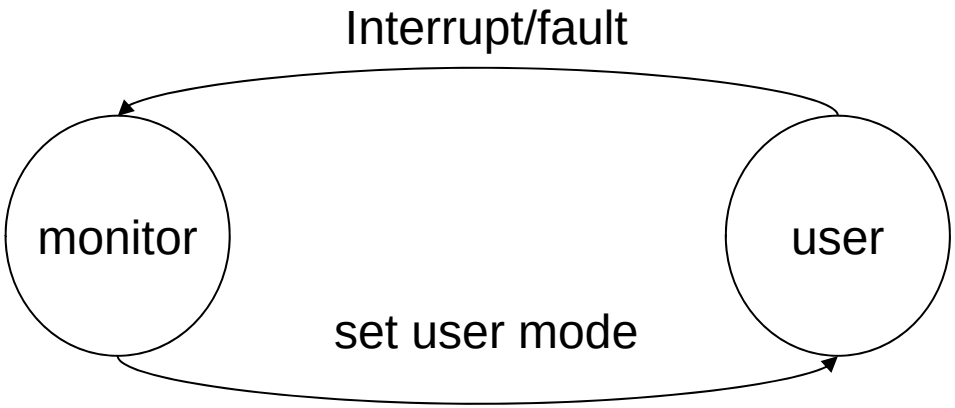


Modes of Operation

- A mechanism that allows the OS to protect itself and other system components
- Two modes:
  - **User mode**
  - **Kernel mode**
- **Mode bit (0 or 1)** provided by hardware
  - Provides ability to distinguish when system is running user code or kernel code
  - Some instructions designated as **privileged**, only executable in kernel mode
  - Systems call by a user asking the OS to perform some function changes from user mode to kernel mode.
  - Return from a system call resets the mode to user mode.

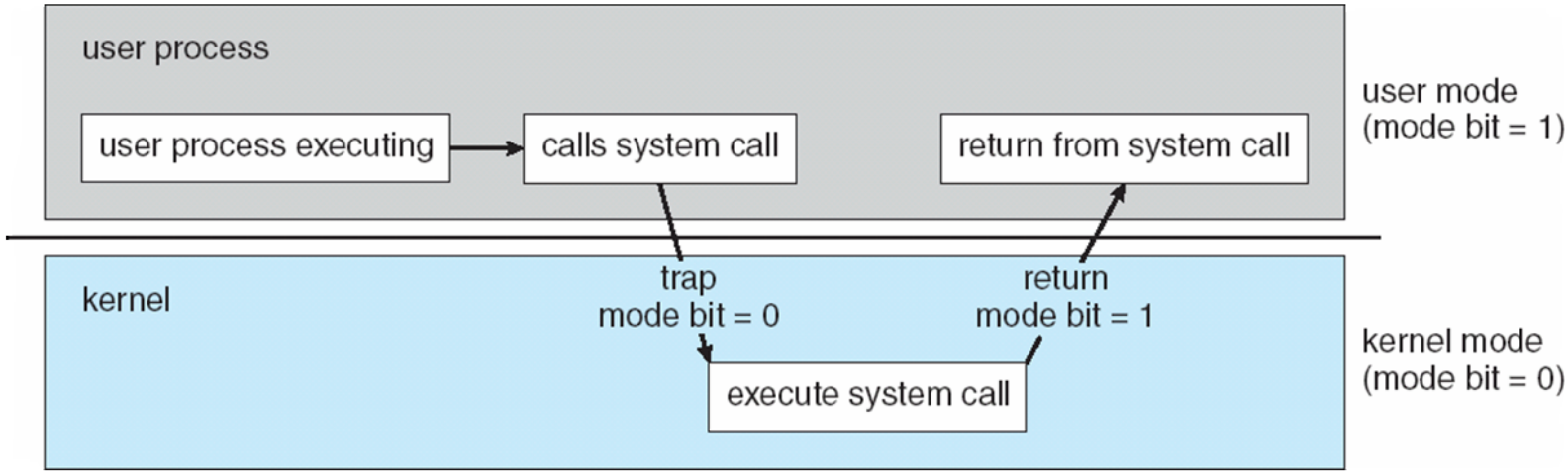
Dual-Mode Operation

- *Mode bit*（*模式位*） added to computer hardware to indicate the current mode: monitor (0) or user (1).
- When an interrupt or fault occurs hardware switches to monitor mode.当中断和错误发生时，切换到内核态



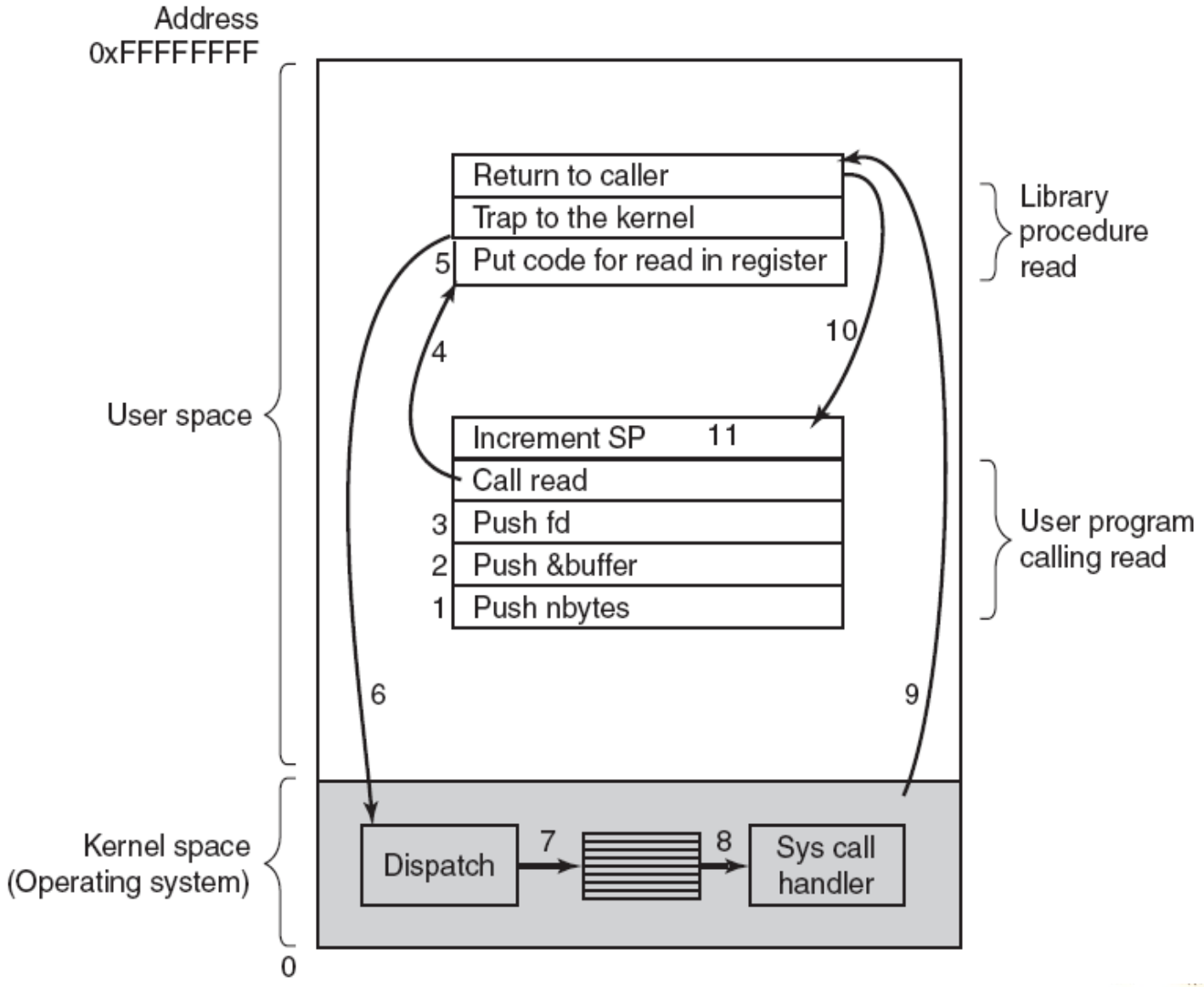
Privileged instructions （*特权指令*） can be issued only in monitor mode.*特权指令只能在内内核态执行。*

Transition from User to Kernel Mode



System Calls

- The 11 steps in making the system call read(fd, buffer, nbytes).



Timer

To prevent process to be in infinite loop (process hogging resources), a **timer** is used, which is a hardware device.

- Timer is a counter that is decremented by the physical clock.
- Timer is set to interrupt the computer after some time period
- Operating system sets the counter (privileged instruction)
- When counter reaches the value zero, and interrupt is generated.
- The OS sets up the value of the counter before scheduling a process to regain control or terminate program that exceeds allotted time

Process Management

- A process is a program in execution. It is a unit of work within the system. Program is a ***passive entity***, process is an ***active entity***.
- Process needs resources to accomplish its task
  - CPU, memory, I/O, files, etc.
  - Initialization data
- Process termination requires reclaim of any reusable resources
- A thread is a basic unit of CPU utilization within a process.
  - Single-threaded process. Instructions are executed sequentially, one at a time, until completion
  - Process has one **program counter** specifying location of next instruction to execute
- Multi-threaded process has one program counter per thread
- Typically, a system has many processes, some user, some operating system running concurrently on one or more CPUs
  - Concurrency by multiplexing the CPUs among the threads

Process Management Activities

The operating system is responsible for the following activities in connection with process management:

- Creating and deleting both user and system processes
- Suspending and resuming processes
- Providing mechanisms for process synchronization
- Providing mechanisms for process communication
- Providing mechanisms for deadlock handling

Memory Management

- To execute a program all (or part) of the instructions must be in memory
- All (or part) of the data that is needed by the program must be in memory.
- Memory management determines what is in memory and when
  - Optimizing CPU utilization and computer response to users
- Memory management activities
  - Keeping track of which parts of memory are currently being used and by whom
  - Deciding which processes (or parts thereof) and data to move into and out of memory
  - Allocating and deallocating memory space as needed

Storage Management

- OS provides uniform, logical view of information storage
- Abstracts physical properties to logical storage unit - **file**
- Files are stored in a number of different storage medium.
  - Disk
  - Flash Memory
  - Tape
- Each medium is controlled by device drivers (i.e., disk drive, tape drive)
  - Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)

File System Management

- Files usually organized into directories
- Access control on most systems to determine who can access what
- OS activities include
  - Creating and deleting files and directories
  - Primitives to manipulate files and directories
  - Mapping files onto secondary storage
  - Backup files onto stable (non-volatile) storage media

Secondary-Storage Management

- Usually disks used to store data that does not fit in main memory or data that must be kept for a “long” period of time
- Proper management is of central importance
- Entire speed of computer operation hinges on disk subsystem and its algorithms
- OS activities
  - Free-space management
  - Storage allocation
  - Disk scheduling
- Some storage need not be fast
  - Tertiary storage includes optical storage, magnetic tape
  - Still must be managed – by OS or applications

Caching

- Important principle, performed at many levels in a computer (in hardware, operating system, software)
- Information in use copied from slower to faster storage temporarily
- Faster storage (cache) checked first to determine if information is there
  - If it is, information used directly from the cache (fast)
  - If not, data copied to cache and used there
- Cache are smaller (size-wise) than storage being cached
  - Cache management important design problem
  - Cache size and replacement policy

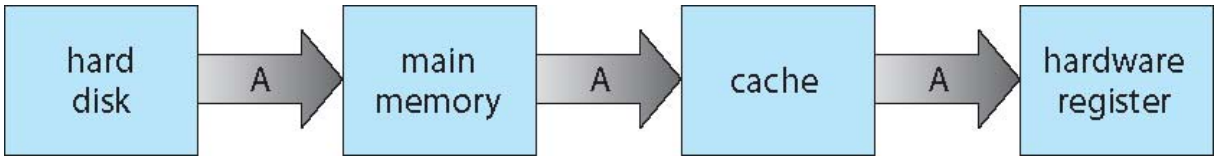
Performance of Various Levels of Storage

Level	1	2	3	4	5
Name	registers	cache	main memory	solid state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25 - 0.5	0.5 - 25	80 - 250	25,000 - 50,000	5,000,000
Bandwidth (MB/sec)	20,000 - 100,000	5,000 - 10,000	1,000 - 5,000	500	20 - 150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape

Movement between levels of storage hierarchy can be explicit or implicit

Migration of data “A” from Disk to Register

- Multitasking environments must be careful to use most recent value, no matter where it is stored in the storage hierarchy



- Multiprocessor environment must provide **cache coherency** in hardware such that all CPUs have the most recent value in their cache
- Distributed environment situation even more complex
  - Several copies of a datum can exist
  - Various solutions covered in Chapter 17

I/O Subsystem

- One purpose of an operating system is to hide peculiarities of hardware devices from the user
- I/O subsystem responsible for
  - Memory management of I/O including buffering (storing data temporarily while it is being transferred), caching (storing parts of data in faster storage for performance), spooling (the overlapping of output of one job with input of other jobs)
  - General device-driver interface
  - Drivers for specific hardware devices

Protection and Security

- **Protection** – A mechanism for controlling access of processes (or users) to resources defined by the OS
- **Security** – A defense of the system against internal and external attacks
  - Huge range, including denial-of-service, worms, viruses, identity theft, theft of service
- Systems generally first distinguish among users, to determine who can do what
  - User identities (**user IDs**, security IDs) include name and associated number, one per user
  - User ID is associated with all files and processes of that user to determine access control
  - Group identifier (**group ID**) allows set of users to be defined and controls managed, then also associated with each process, file
  - **Privilege escalation** allows user to change to effective ID with more rights

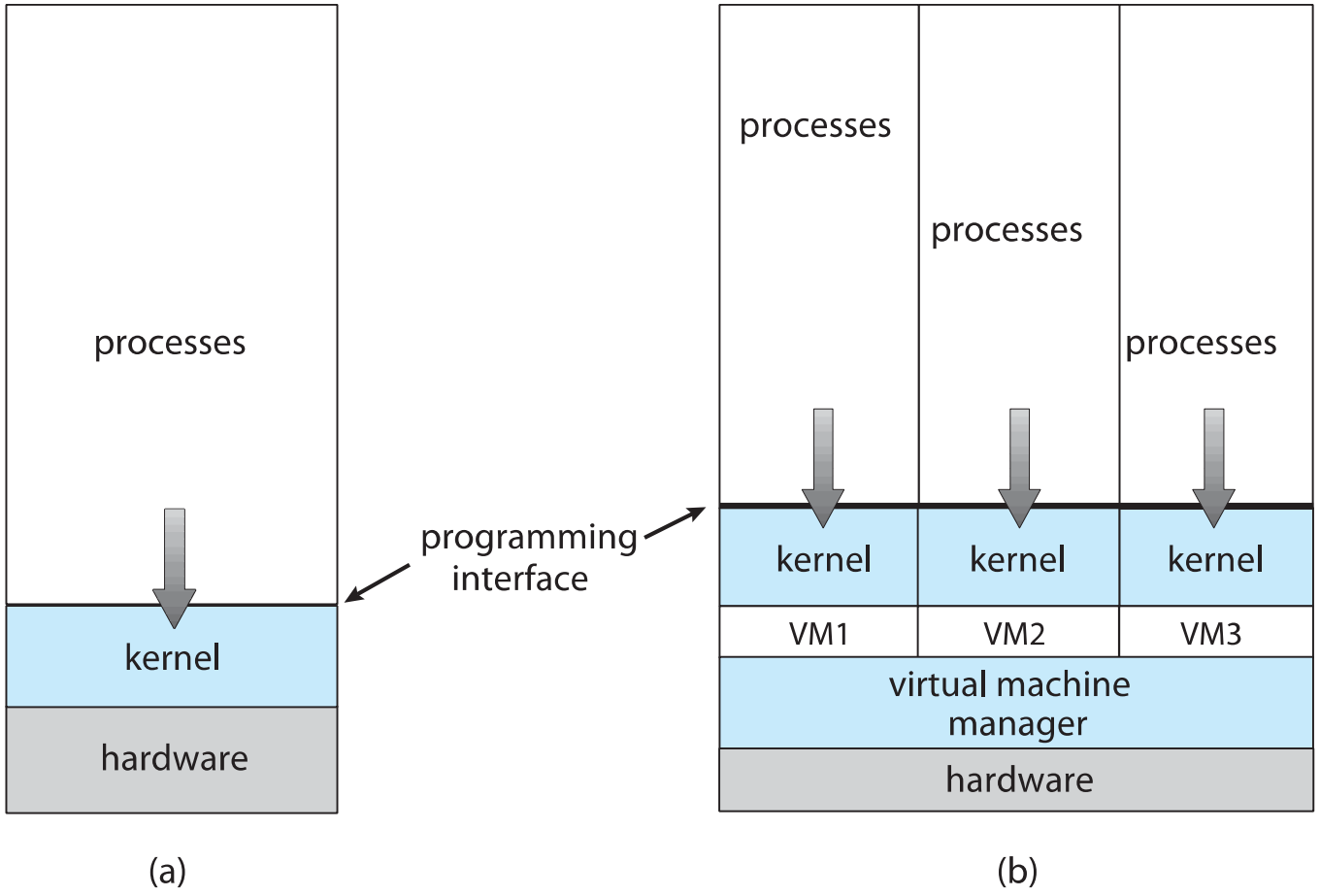
Virtualization

- Allows operating systems to run applications within other OSes
  - Vast and growing industry
- **Emulation** used when the source CPU type is different from the target type (i.e., PowerPC to Intel x86)
  - Generally slowest method
  - When computer language not compiled to native code – **Interpretation**
- **Virtualization** – OS natively compiled for CPU, running **guest** OSes also natively compiled
  - Consider VMware running WinXP guests, each running applications, all on native WinXP **host** OS
  - **VMM** (virtual machine Manager) provides virtualization services

Virtualization on Laptops and Destops

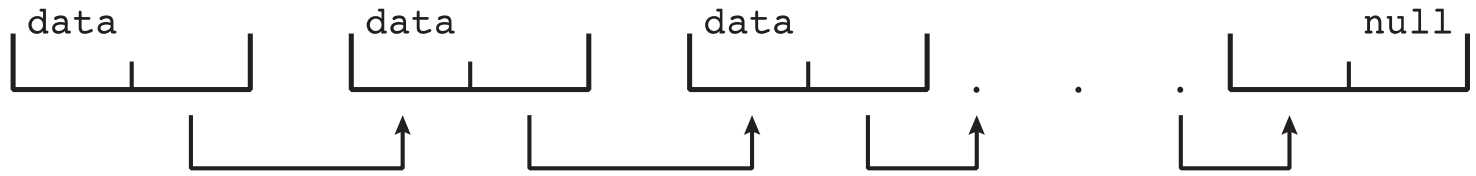
- A VMM allow the user to install multiple operating systems to run application written for operating systems other than the native host.
  - Apple laptop running Mac OS X host Windows as a guest
  - Developing apps for multiple OSes without having multiple systems
  - Testing applications without having multiple systems
  - Executing and managing compute environments within data centers

Virtualization Architecture Structure

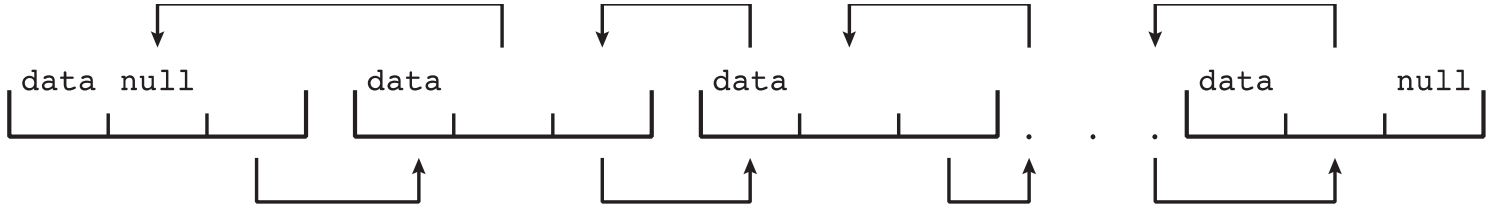


Kernel Data Structures

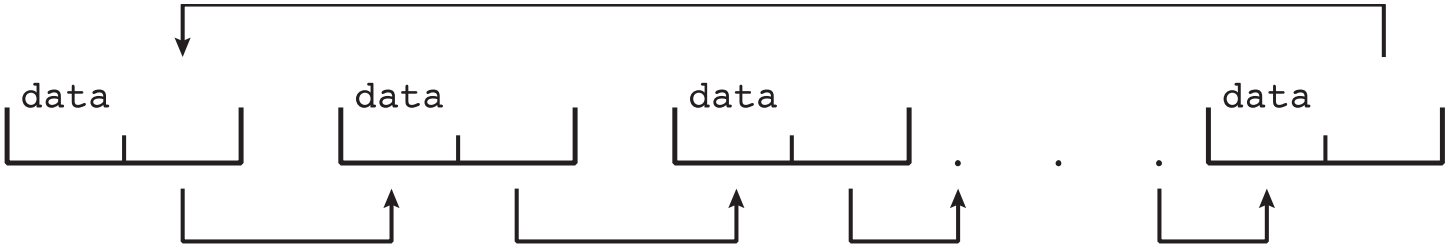
- Many -- similar to standard programming data structures
- ***Singly linked list***



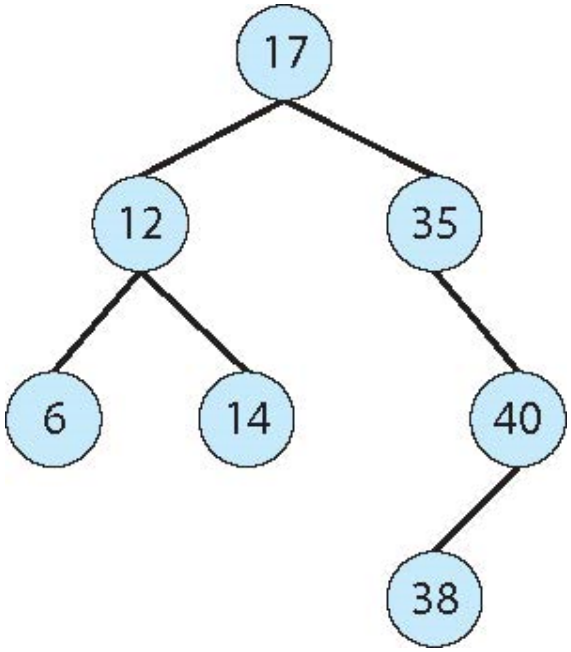
- ***Doubly linked list***



- ***Circular linked list***

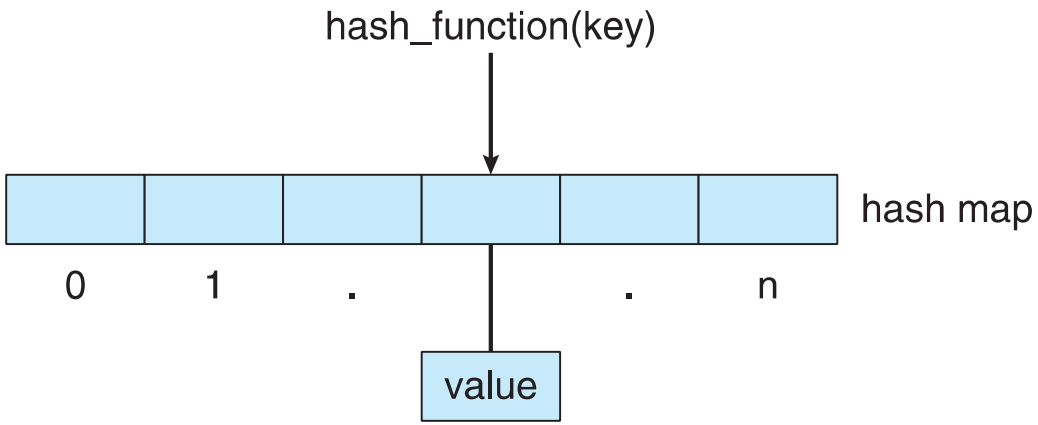


Binary search tree



Kernel Data Structures

- **Hash function** can create a **hash map**



- **Bitmap** – string of *n* binary digits representing the status of *n* items
- Linux data structures defined in

  include files `<linux/list.h>`, `<linux/kfifo.h>`, `<linux/rbtree.h>`

Computing Environments - Traditional

- Stand-alone general purpose machines
- But blurred as most systems interconnect with others (i.e., the Internet)
- **Portals** provide web access to internal systems
- **Network computers** (**thin clients**) are like Web terminals
- Mobile computers interconnect via **wireless networks**
- Networking becoming ubiquitous – even home systems use **firewalls** to protect home computers from Internet attacks

Computing Environments - Mobile

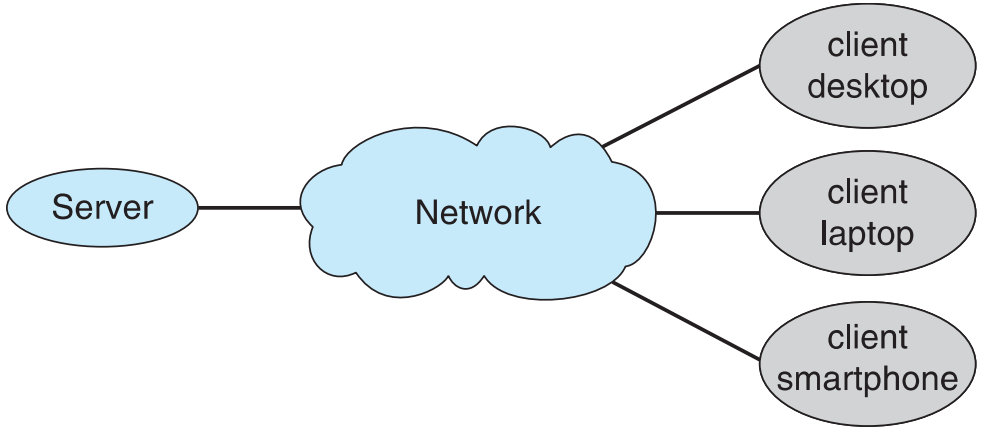
- Handheld smartphones, tablets, etc
- What is the functional difference between them and a "traditional" laptop?
- Extra features – more OS features (GPS -- Waze)
- Allows new types of apps like ***augmented reality***
- Use IEEE 802.11 wireless, or cellular data networks for connectivity
- Leaders are **Apple iOS** and **Google Android**

Computing Environments – Distributed

- Collection of separate, possibly heterogeneous, systems networked together
  - **Network** is a communications path, **TCP/IP** most common
    - **Local Area Network** (**LAN**)
    - **Wide Area Network** (**WAN**)
    - **Metropolitan Area Network** (**MAN**)
    - **Personal Area Network** (**PAN**)
- **Network Operating System** provides features to allow sharing of data between systems across a network.
  - Communication scheme allows systems to exchange messages
  - Illusion of a single system

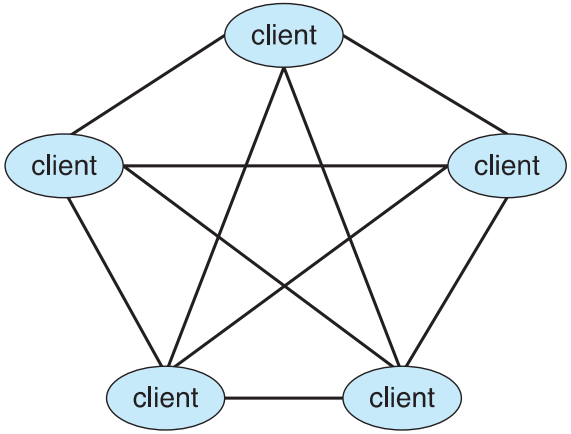
Computing Environments – Client-Server

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



Computing Environments - Peer-to-Peer

- Another model of distributed system. P2P does not distinguish clients and servers
  - Instead all nodes are considered peers
  - Each node may act as client, server, or both
  - Node must join P2P network
    - Registers its service with central lookup service on network, or
    - Broadcast request for service and respond to requests for service via ***discovery protocol***
  - Examples include Napster and Gnutella, **Voice over IP** (**VoIP**) such as Skype

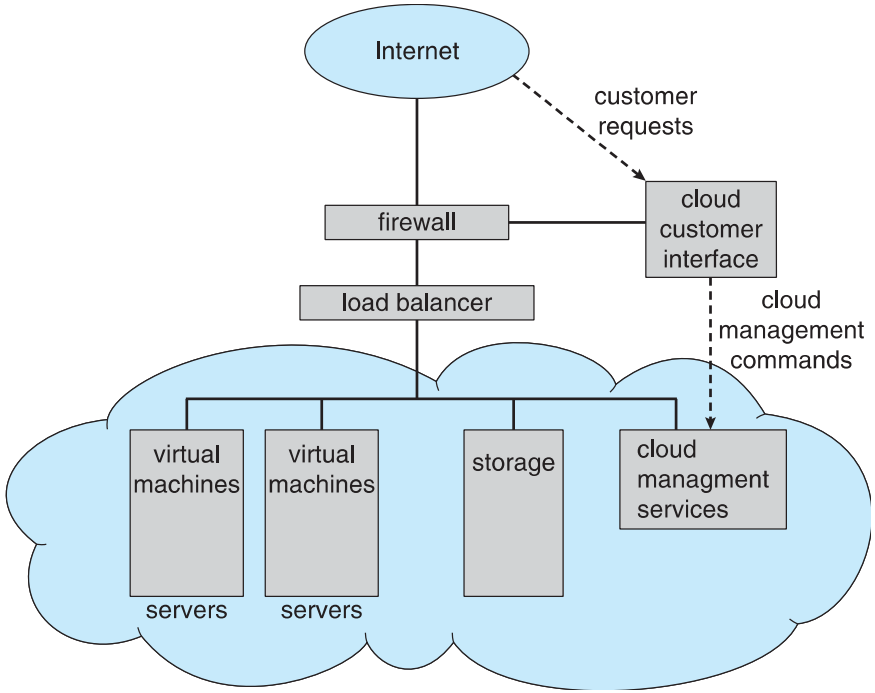


Computing Environments – Cloud Computing

- Delivers computing, storage, even apps as a service across a network
- Logical extension of virtualization because it uses virtualization as the base for it functionality.
  - Amazon **EC2** has thousands of servers, millions of virtual machines, petabytes of storage available across the Internet, pay based on usage
- Many types
  - **Public cloud** – available via Internet to anyone willing to pay
  - **Private cloud** – run by a company for the company's own use
  - **Hybrid cloud** – includes both public and private cloud components
  - Software as a Service (**SaaS**) – one or more applications available via the Internet (i.e., word processor)
  - Platform as a Service (**PaaS**) – software stack ready for application use via the Internet (i.e., a database server)
  - Infrastructure as a Service (**IaaS**) – servers or storage available over Internet (i.e., storage available for backup use)

Computing Environments – Cloud Computing

- Cloud computing environments composed of traditional OSes, plus VMMs, plus cloud management tools
  - Internet connectivity requires security like firewalls
  - Load balancers spread traffic across multiple applications



Computing Environments – Real-Time Systems

- Real-time embedded systems most prevalent form of computers
  - Vary considerable, special purpose, limited purpose OS, **real-time OS**
  - Use expanding
- Many other special computing environments as well
  - Some have OSes, some perform tasks without an OS
- Real-time OS has well-defined fixed time constraints
  - Processing ***must*** be done within constraint
  - Correct operation only if constraints met

Open-Source Operating Systems

- Operating systems made available in source-code format rather than just binary **closed-source**
- Counter to the **copy protection** and **Digital Rights Management (DRM)** movement
- Started by **Free Software Foundation (FSF)**, which has "copyleft" **GNU Public License (GPL)**
- Examples include **GNU/Linux** and **BSD UNIX** (including core of **Mac OS X**), and many more
- Can use VMM like VMware Player (Free on Windows), Virtualbox (open source and free on many platforms - http://www.virtualbox.com)
  - Use to run guest operating systems for exploration

第01章 课后习题作业

以下为OSC第9版英文版的章节题目号
第01章 习题
1.1，1.2， 1.5 ， 1.6， 1.5 ， 1.12 ， 1.13，
1.15， 1.16， 1.18， 1.19， 1.20， 1.22， 1.23
， 1.26， 1.28

End of Chapter 1