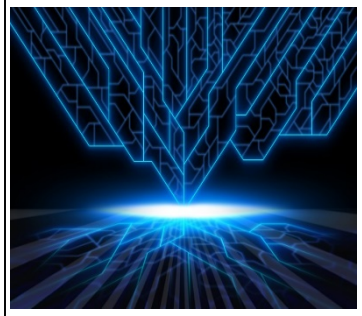
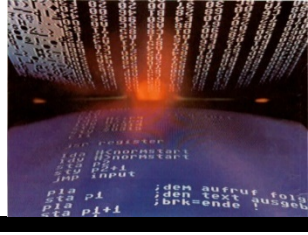
汇编程序设计
The Hardware/Software Interface



第1讲 引言

吴海军
南京大学计算机科学与技术系

主要内容



- 程序执行过程
- 计算机系统抽象
- 计算机系统现实
- 课程简介

一个c程序的转换处理过程

hello.c

```
#include <stdio.h>

int main()
{
printf("hello, world\n");
}
```

hello.c的ASCII文本表示

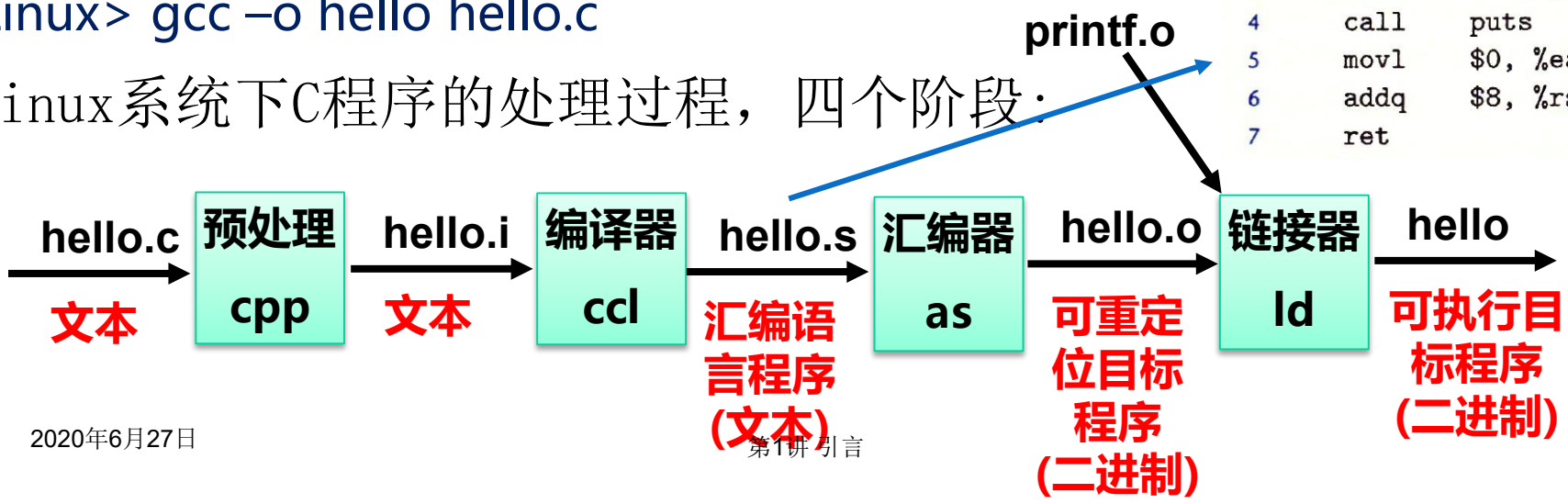
```
# i n c l u d e <sp> < s t d i o .
35 105 110 99 108 117 100 101 32 60 115 116 100 105 111 46
h > \n \n i n t <sp> m a i n ( ) \n {
104 62 10 10 105 110 116 32 109 97 105 110 40 41 10 123
\n <sp> <sp> <sp> <sp> p r i n t f ( " h e l
10 32 32 32 32 112 114 105 110 116 102 40 34 104 101 108
l o , <sp> w o r l d \ n " ) ; \n }
108 111 44 32 119 111 114 108 100 92 110 34 41 59 10 125
```

功能：输出“hello,world” **计算机不能直接执行hello.c!**

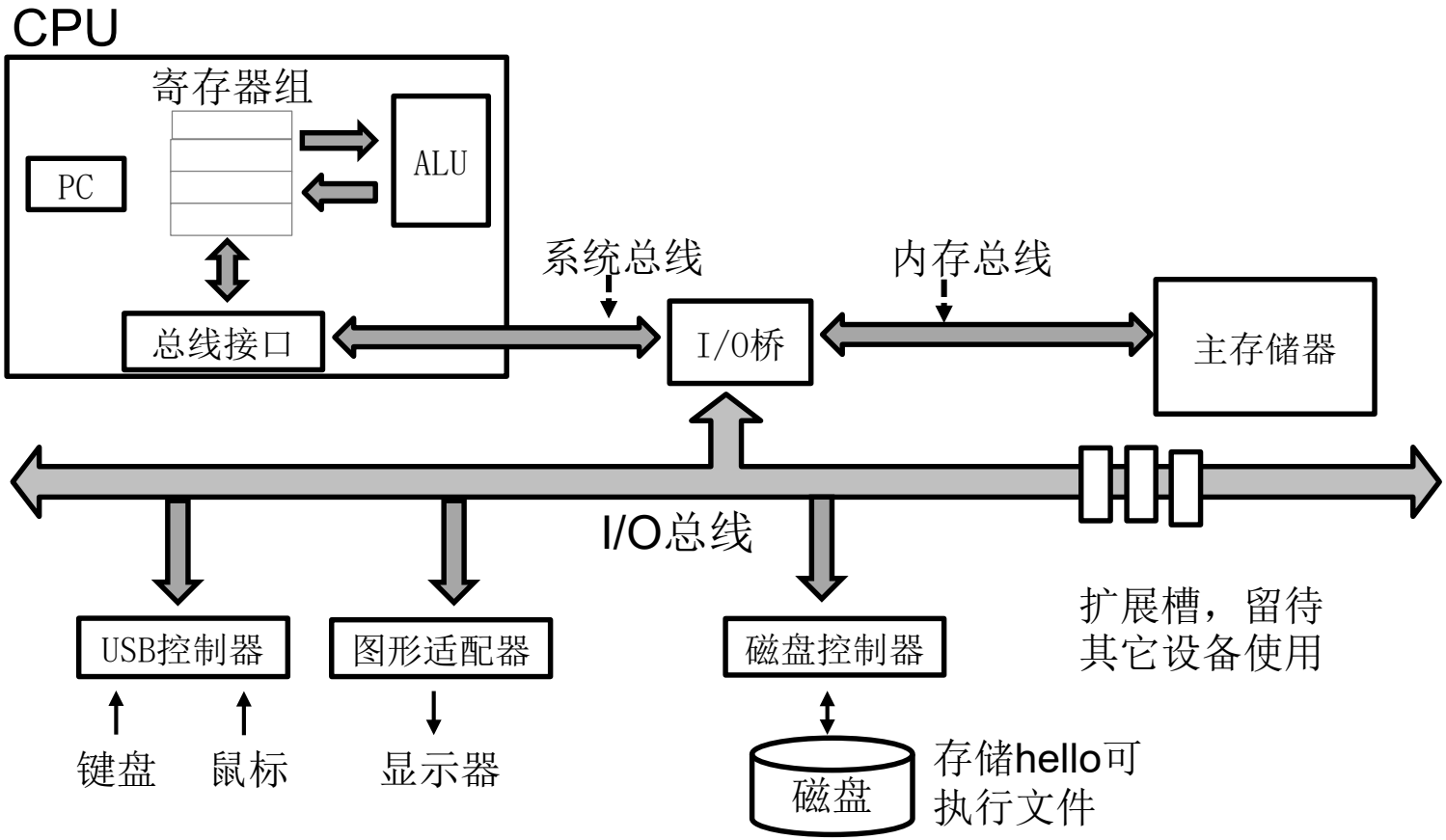
Linux> gcc –o hello hello.c

```
1   main:
2     subq    $8, %rsp
3     movl    $.LC0, %edi
4     call    puts
5     movl    $0, %eax
6     addq    $8, %rsp
7     ret
```

Linux系统下C程序的处理过程，四个阶段：



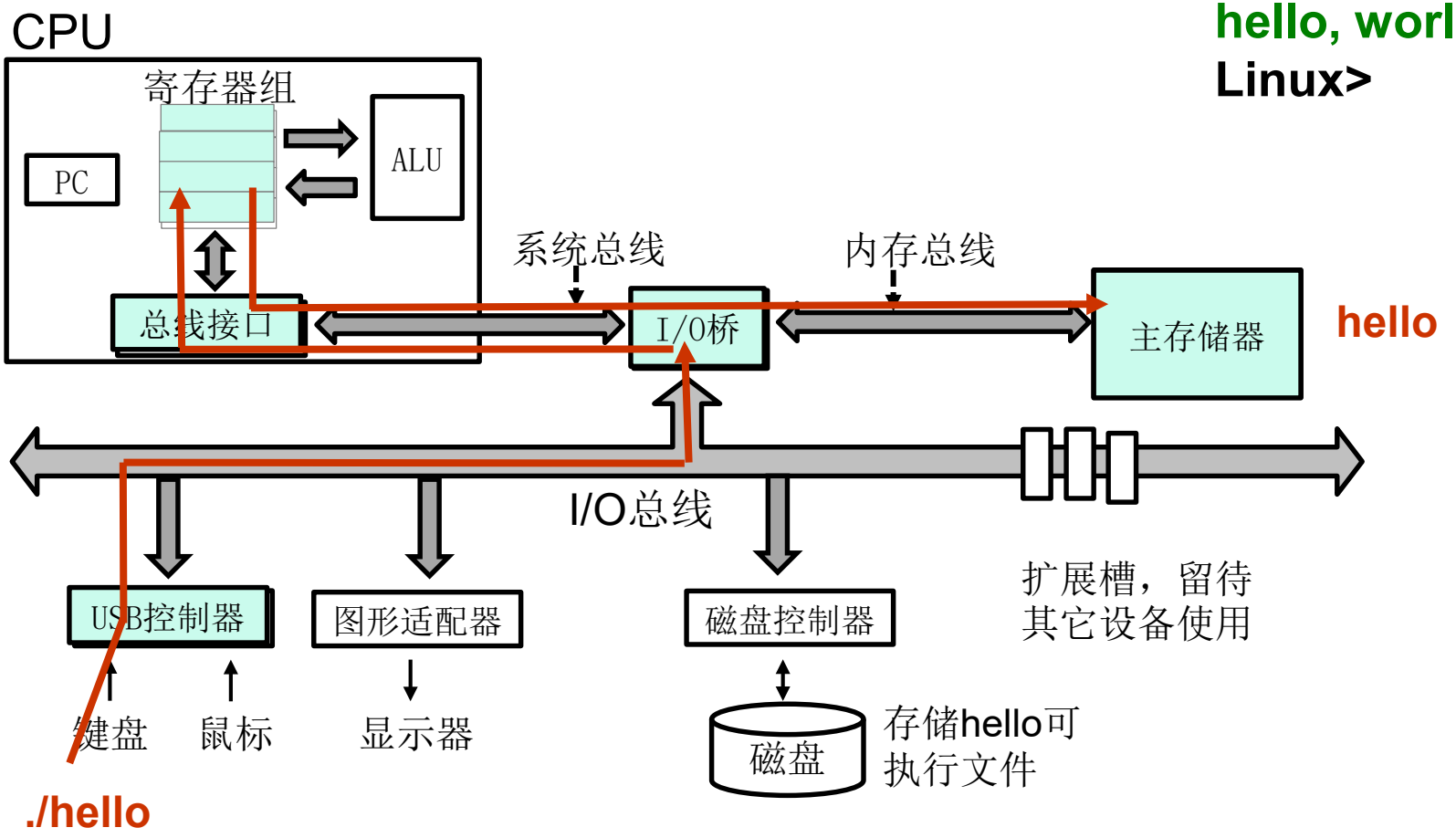
系统硬件组成


CPU
寄存器组
PC
ALU
总线接口
系统总线
内存总线
I/O桥
主存储器
I/O总线
USB控制器
图形适配器
磁盘控制器
扩展槽，留待
其它设备使用
键盘
鼠标
显示器
磁盘
存储hello可
执行文件

hello命令行处理

Red：shell命令行处理

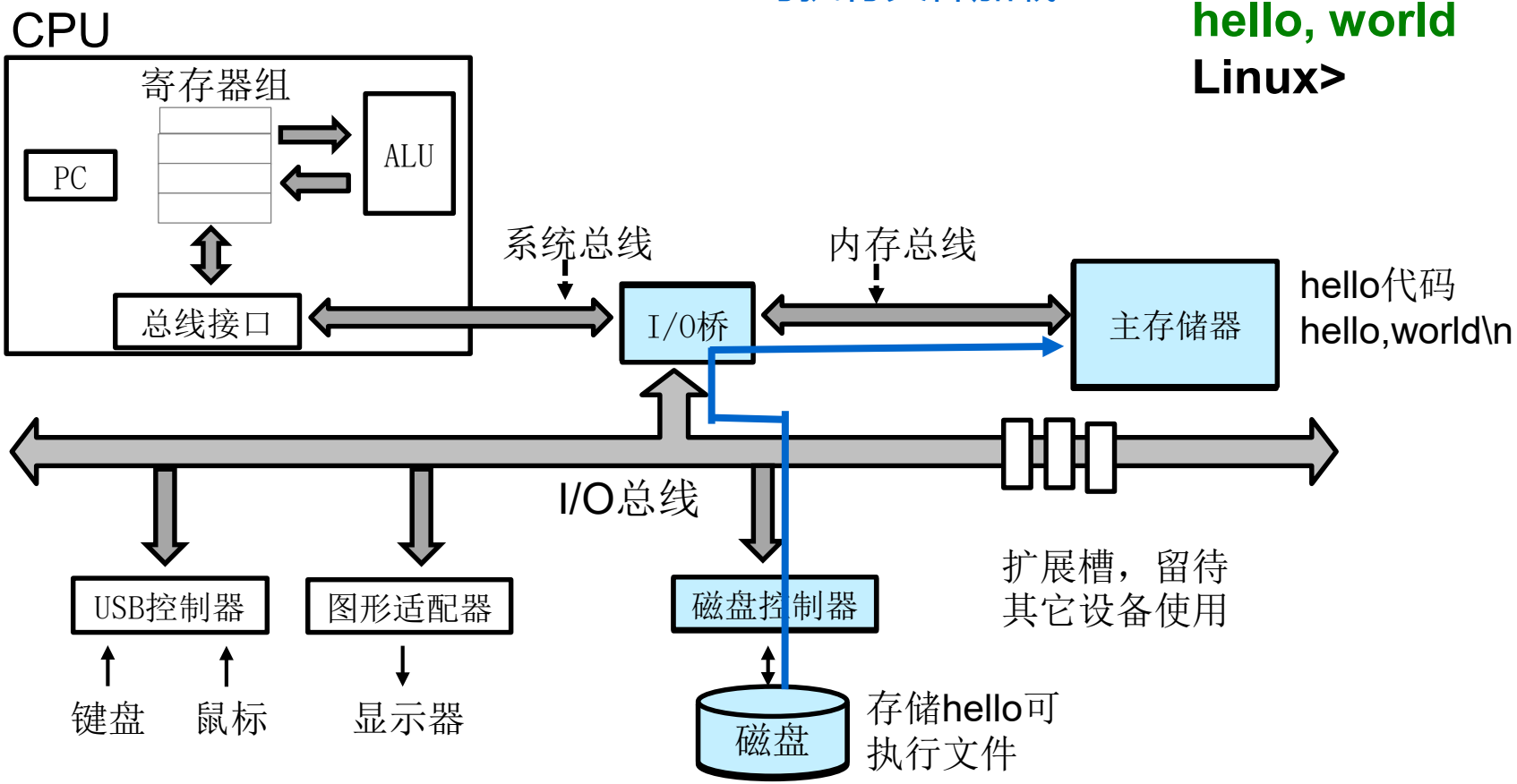
Linux> ./hello
hello, world
Linux>



hello可执行文件装载

Blue：可执行文件加载

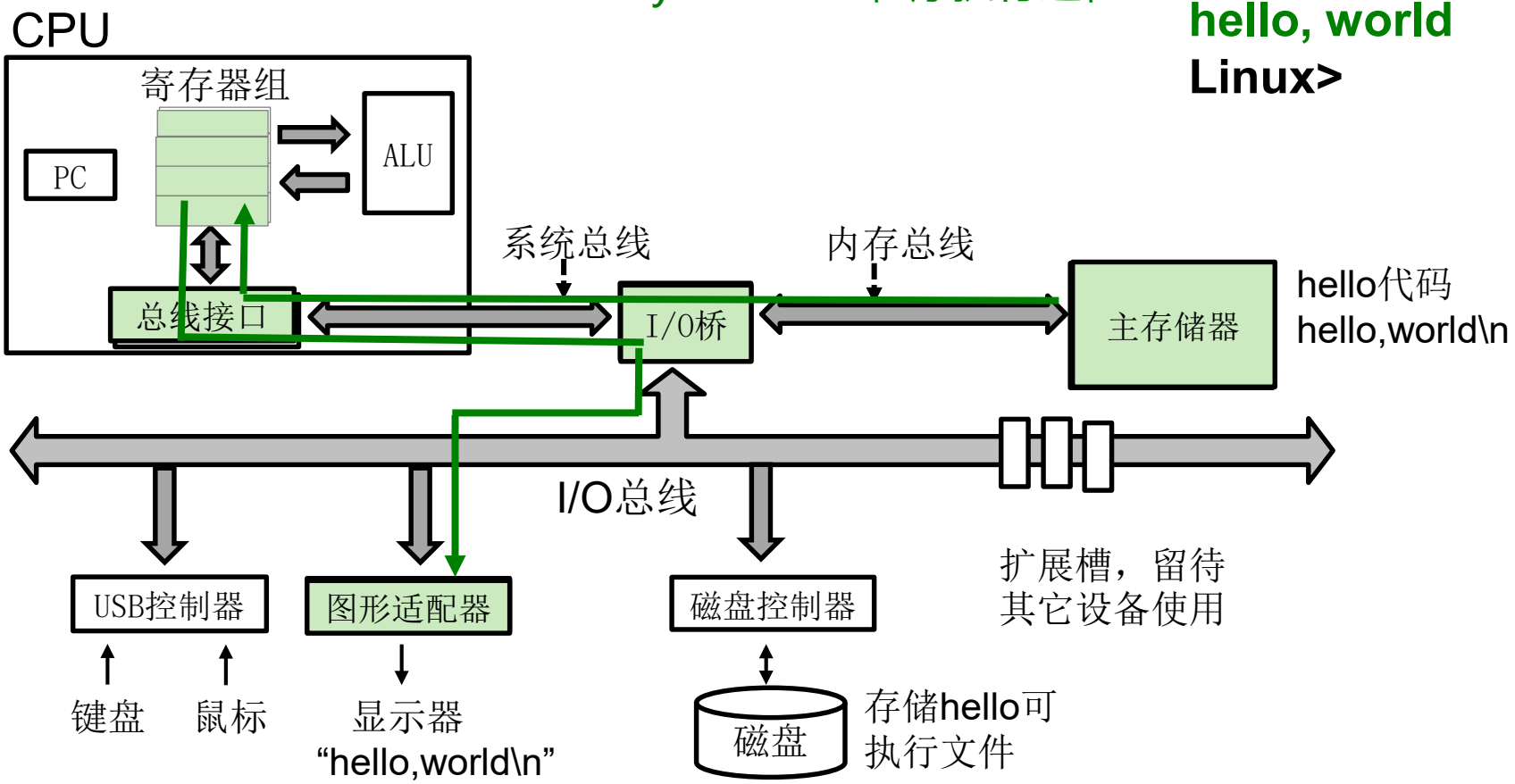
Linux> ./hello
hello, world
Linux>



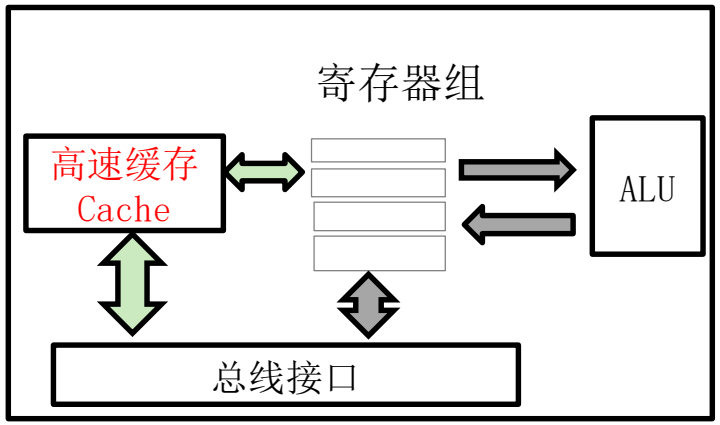
hello文件运行

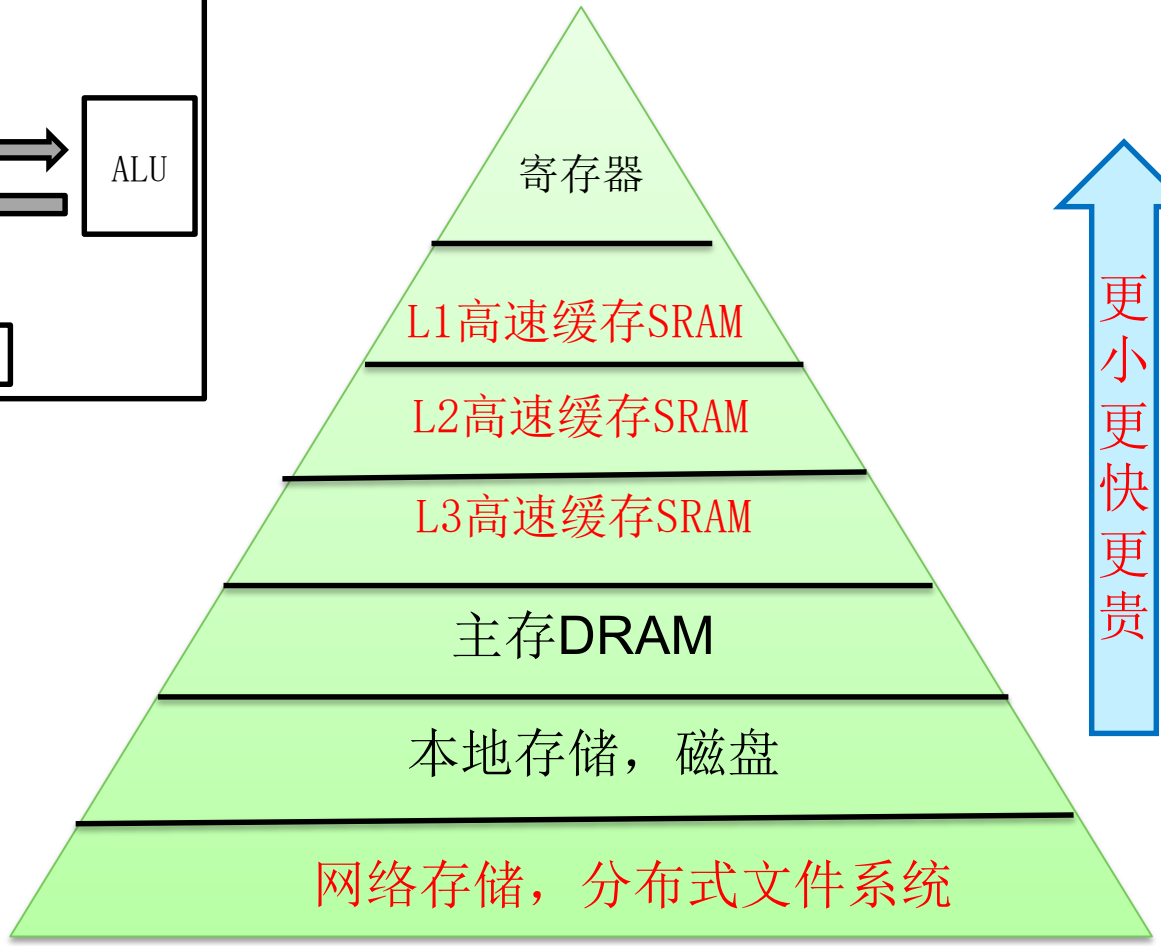
Cyan：hello程序执行过程

```
Linux> ./hello
hello, world
Linux>
```

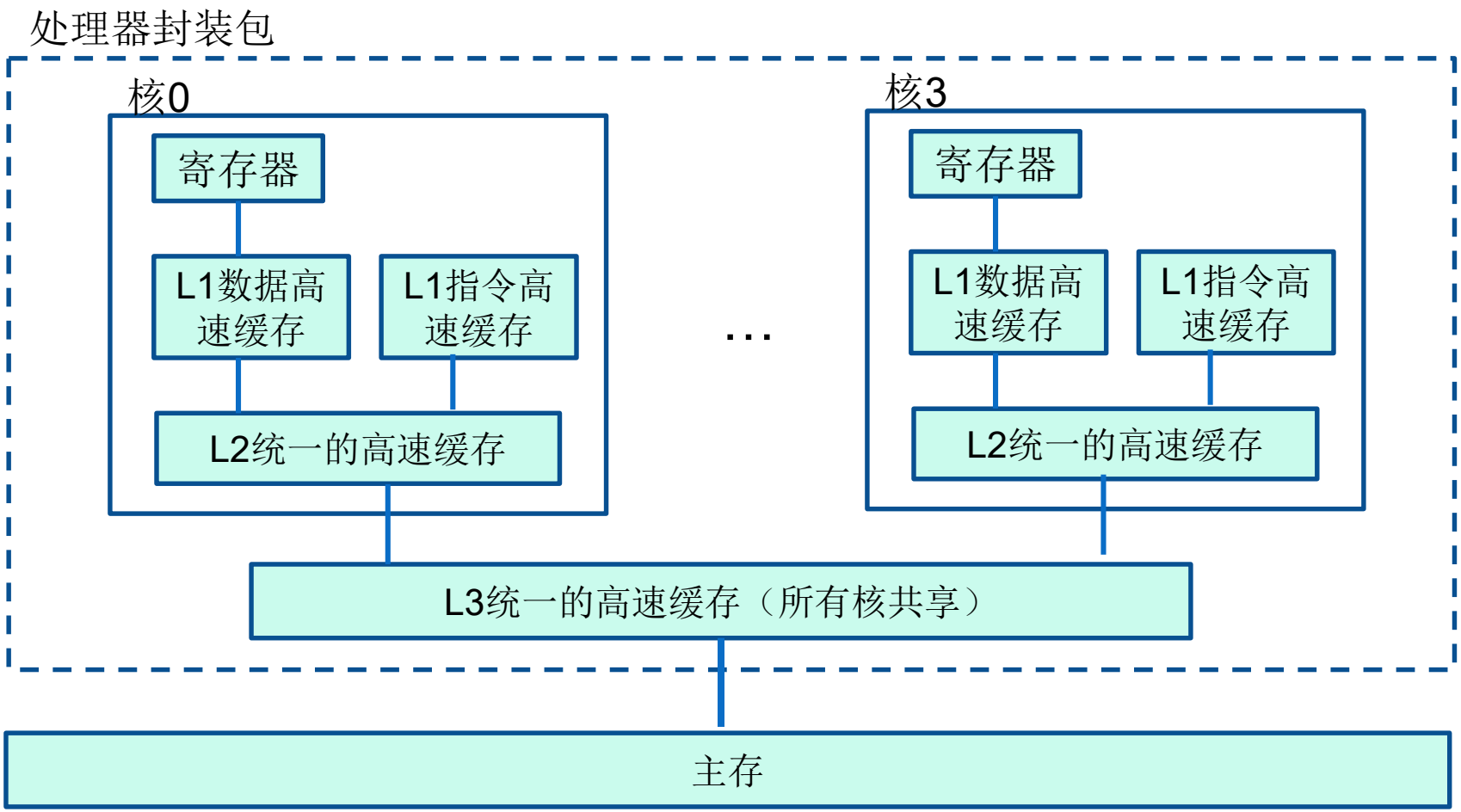


存储体系

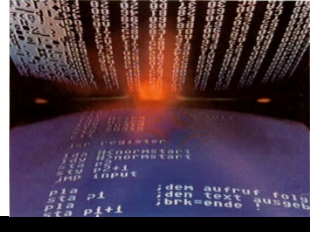

CPU
寄存器组
高速缓存
Cache
ALU
总线接口


寄存器
L1高速缓存SRAM
L2高速缓存SRAM
L3高速缓存SRAM
主存DRAM
本地存储，磁盘
网络存储，分布式文件系统
更小更快更贵
更大更慢更便宜

多核处理器


处理器封装包
核0
寄存器
L1数据高速缓存
L1指令高速缓存
L2统一的高速缓存
...
核3
寄存器
L1数据高速缓存
L1指令高速缓存
L2统一的高速缓存
L3统一的高速缓存（所有核共享）
主存

高级语言程序的运行层次



高级语言源程序

```
int t = x+y;
```

高级语言到汇编语言到机器语言的转换。

编译程序

汇编语言源程序

```
movl %esp,%ebp
movl 12(%ebp),%eax
addl 8(%ebp),%eax
```

数据的表示和操作

汇编程序

机器语言目标程序

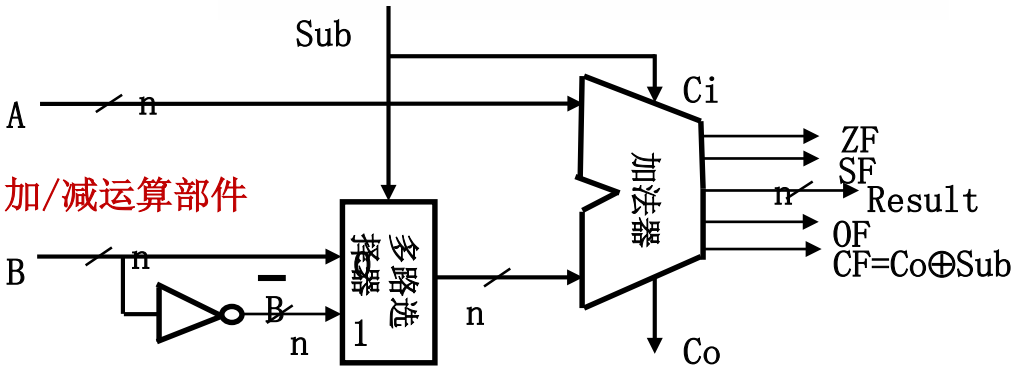
```
100011 00010 01000 0000000000000000
100011 00010 01001 0000000000000100
101011 00010 01001 0000000000000000
101011 00010 01000 0000000000000100
```

计算机软件

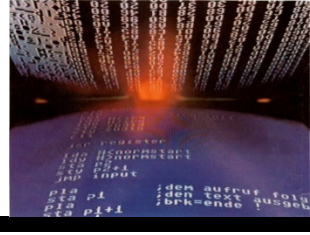
计算机硬件

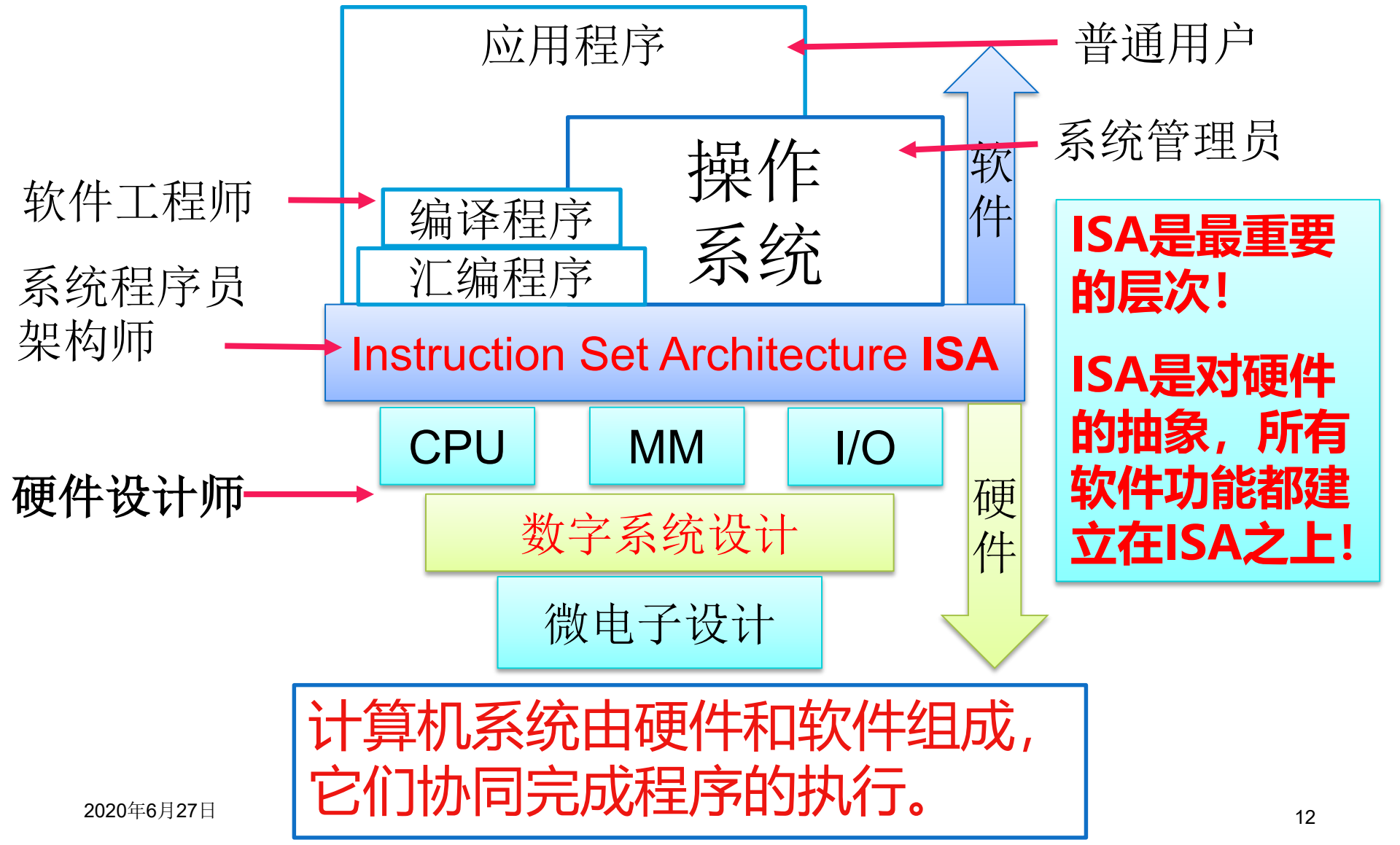
指令译码器

控制信号



计算机系统抽象层次

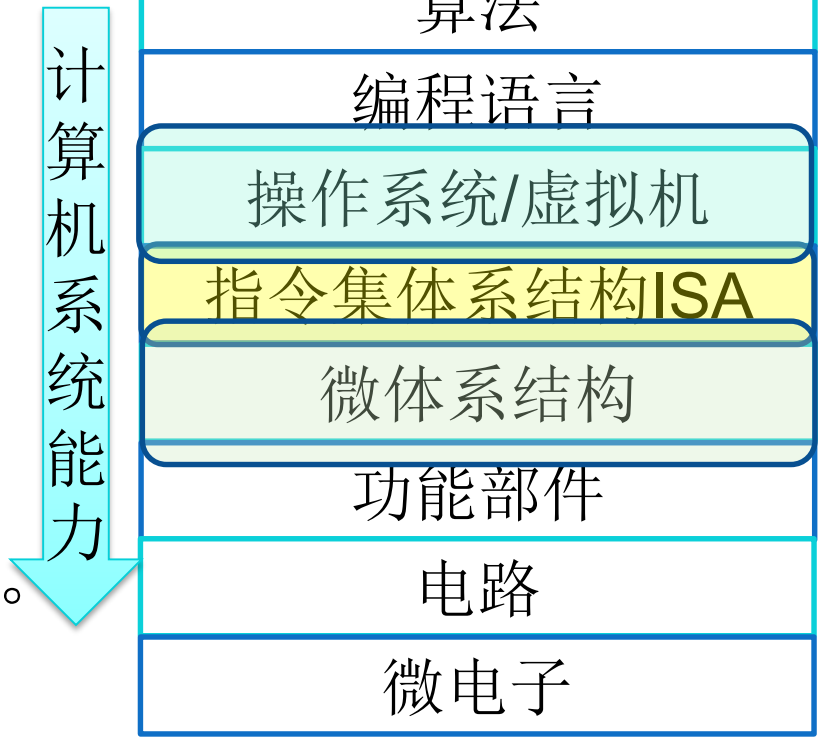



应用程序
普通用户
操作系统
系统管理员
软件
软件工程师
编译程序
汇编程序
系统程序员
架构师
Instruction Set Architecture ISA
CPU
MM
I/O
硬件设计师
数字系统设计
微电子设计
硬件
ISA是最重要的层次!
ISA是对硬件的抽象，所有软件功能都建立在ISA之上!
计算机系统由硬件和软件组成，它们协同完成程序的执行。

计算机系统抽象层次

上层是下层的抽象，下层是上层的实现，下层为上层提供支撑环境！

计算机系统能力：将分布在不同抽象层次的计算机课程串联起来解决问题。

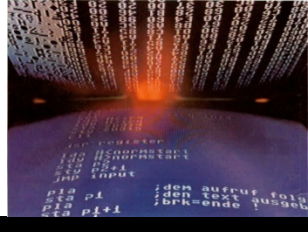


C：数据的机器级表示、运算语句和过程调用的机器级表示

汇编：指令系统、机器代码、汇编语言程序设计。

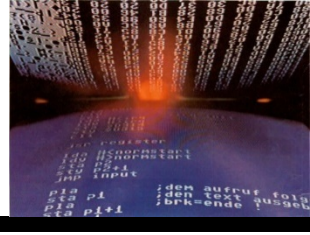
计算机底层软硬件是如何协同工作的？

抽象与现实



- **Abstraction Is Good But Don't Forget Reality**
- 大多数CS课程都强调抽象
  - 抽象数据类型
  - 逐步逼近的算法分析等
- 这些抽象都是有限制的，特别是存在bug的情况下
- 我们需要了解底层实现的细节

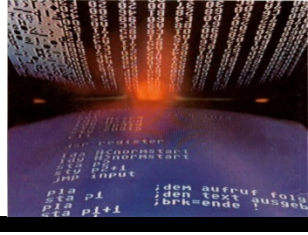
现实1：数据表示的制约



- 例1：$x^2 \geq 0$？
  - Float's: 是!
  - Int's: 不一定！
    - 40000*40000=1600000000
    - 50000*50000= -1794967296
- (x + y) + z = x + (y + z)？
  - Unsigned & Signed Int's: 是!
  - Float's: 不一定！
    - (1e20 - 1e20) + 3.14 --> 3.14
    - 1e20 + (-1e20 + 3.14) --> **0**

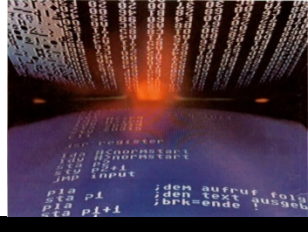
Ints are not Integers, Floats are not Reals

现实1：数据表示的制约



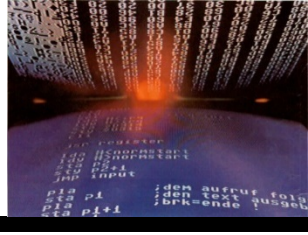
- Int类型数据：
  - 由于数据长度的限制，并不能表示所有的整数
  - 在运算的时候可能会造成的上下溢出，使得符号和值会跳变。
- Folat类型数据：
  - 因为数据长度和精度有限，并不能表示所有的实数
  - 在运算的时候可能会出现上下溢出，数值会跳变
- 他们都是整数集合和实数集合的子集。

算术运算的制约



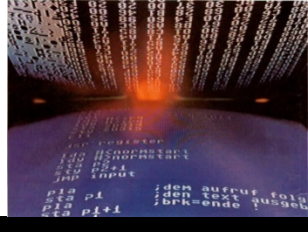
- 不能产生真正的随机值
  - 算术运算具有重要的数学特性
- 不能假设所有“通常”的数学属性都成立
  - 由于表示范围的有限：
    - 整数运算满足“环”属性
      - 交换性，相关性，分配性
    - 浮点运算满足“排序”属性
      - 单调性，符号位的值
- 对于程序员来说
  - 需要了解哪些抽象适用于哪些上下文

现实2: 需要了解汇编语言



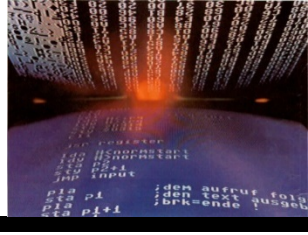
- 你**可能**永远不会使用汇编语言来写程序
  - 编译器做得比你更好，更耐心
- 但是：**汇编语言是理解机器级执行模型的关键**
  - 程序存在bug，导致系统崩溃
  - 调整程序性能：
    - 理解哪些优化是编译器能够实现或不能实现
    - 了解哪些源程序是没有效率的
  - 开发系统软件
    - 编译器需要研究机器代码
    - 操作系统需要管理进程状态
  - 创建/对抗恶意软件
    - 汇编语言依然是首选语言！

现实3: 存储器访问的非物理性



- 存储器大小不是无限的
  - 必须进行分配和管理
  - 应用程序都会受到存储限制
- 内存引用错误危害极大
  - 在时间和空间上效率相差极大
- 存储器性能不一致
  - Cache和虚拟内存访问效果极大地影响程序性能
  - 存储系统的自适应特性可以大大提高运行速度

内存引用错误实例



```
typedef struct {
  int a[2];
  double d;
} struct_t;

double fun(int i) {
  volatile struct_t s;
  s.d = 3.14;
  s.a[i] = 1073741824; /* Possibly out of bounds */
  return s.d;
}
```

```
fun(0)  ↬  3.14
fun(1)  ↬  3.14
fun(2)  ↬  3.1399998664856
fun(3)  ↬  2.00000061035156
fun(4)  ↬  3.14
fun(6)  ↬  Segmentation fault
```

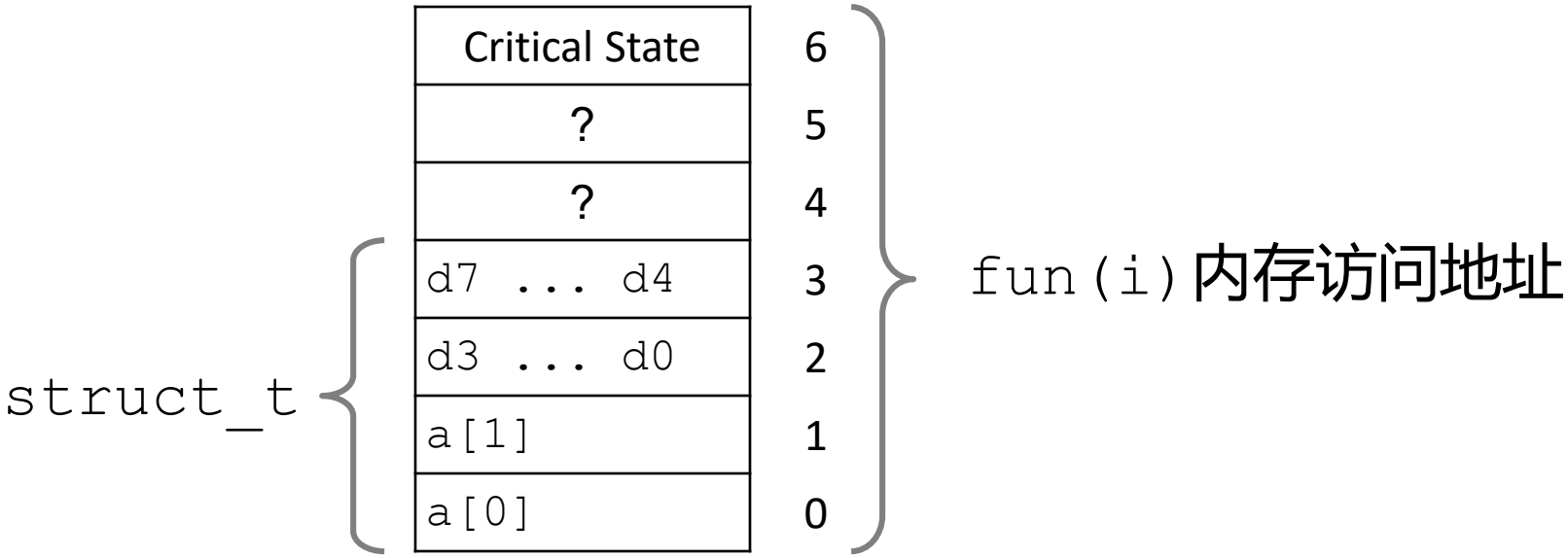
结果依赖于特定系统

内存引用错误实例

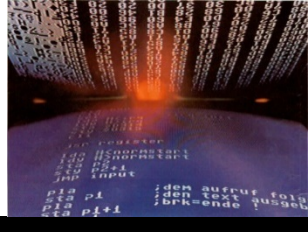
```
typedef struct {
  int a[2];
  double d;
} struct_t;
```

```
fun(0)  ↝  3.14
fun(1)  ↝  3.14
fun(2)  ↝  3.1399998664856
fun(3)  ↝  2.0000061035156
fun(4)  ↝  3.14
fun(6)  ↝  Segmentation fault
```

Explanation:

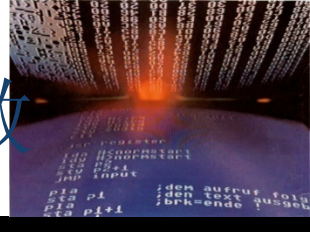


内存引用错误



- C和C ++不提供任何内存保护
  - 超出范围的数组引用
  - 无效的指针值
  - 滥用malloc /free
- 可能导致Bug
  - bug是否有影响取决于操作系统和编译器
  - 远距离访问
    - 访问的对象与执行的对象没有逻辑上相关性
    - Bug的影响在生成之后很久才能观察到

现实4：优化比降低算法复杂度更有效



- 常量也很重要！
- 根据代码的编写方式可以看出10倍差异性能
- 必须在多个级别进行优化：算法、数据表示、过程和循环
- 优化性能必须了解操作系统
  - 程序如何编译和执行
  - 如何衡量程序性能和识别其中的瓶颈
  - 如何在不破坏代码模块化和通用性的情况下提高性能

内存访问优化实例

```
void copyij (int src[2048][2048],
             int dst[2048][2048])
{
  int i,j;
  for (i = 0; i < 2048; i++)
    for (j = 0; j < 2048; j++)
      dst[i][j] = src[i][j];
}
```

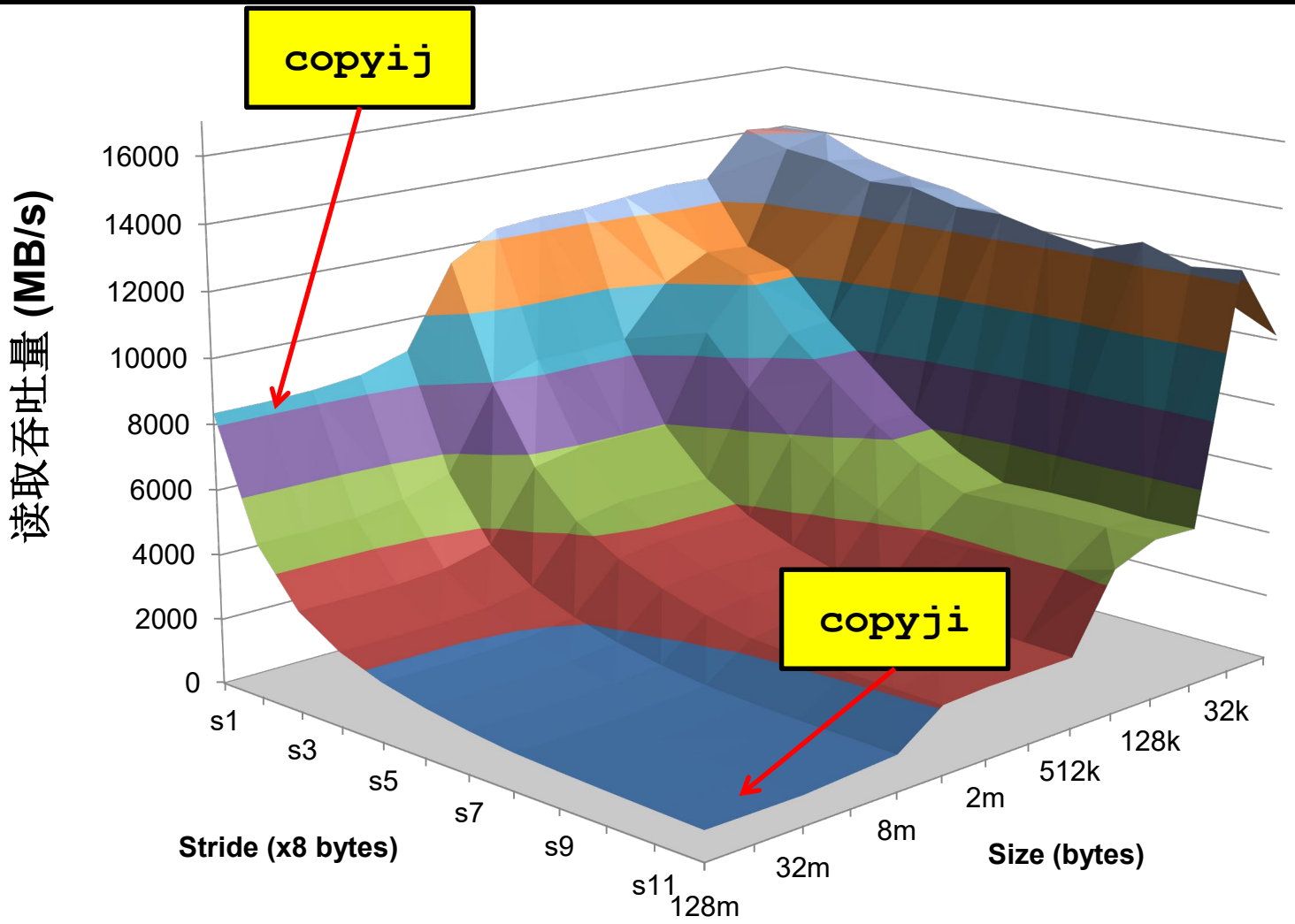
```
void copyji (int src[2048][2048],
             int dst[2048][2048])
{
  int i,j;
  for (j = 0; j < 2048; j++)
    for (i = 0; i < 2048; i++)
      dst[i][j] = src[i][j];
}
```

以上两个程序功能完全一样，算法完全一样，因此，时间和空间复杂度完全一样，执行时间一样吗？

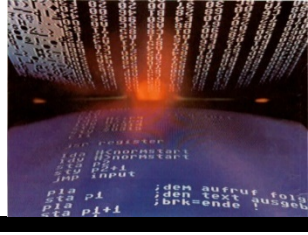
你们在实验中结果相差多少倍？

在奔腾4的计算机上相差21倍。

内存访问优化实例

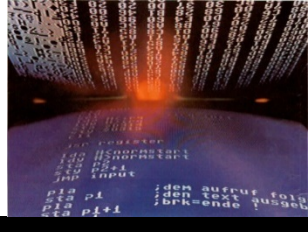

copyij
copyji
读取吞吐量 (MB/s)
16000
14000
12000
10000
8000
6000
4000
2000
0
s1
s3
s5
s7
s9
s11
Stride (x8 bytes)
32k
128k
512k
2m
8m
32m
128m
Size (bytes)

现实5:计算机不仅仅能执行程序



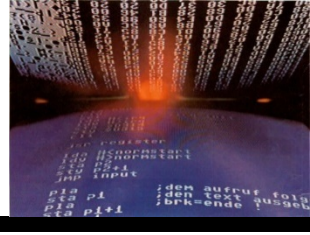
- 计算机需要输入输出数据
  - I/O系统对程序可靠性和性能至关重要
- 计算机间通过网络相互通信
  - 在当前网络上存在许多系统级别的问题
    - 自主处理过程中的并发操作
    - 来自不可靠媒体的复制
    - 跨平台兼容性
    - 复杂性能问题

抽象与现实



- 通过本课程的学习，你能
  - 成为更有效的程序员
  - 有效地发现和消除错误
  - 理解和调整程序性能
- 为后续“系统”课程做好准备
  - 编译器，操作系统，网络，计算机体系结构，嵌入式系统，存储系统等

课程介绍



- 教学目的：

  在软件和硬件间建立关联，在高级语言和机器级编码间建立关联，深入了解计算机底层系统架构及运行机制。

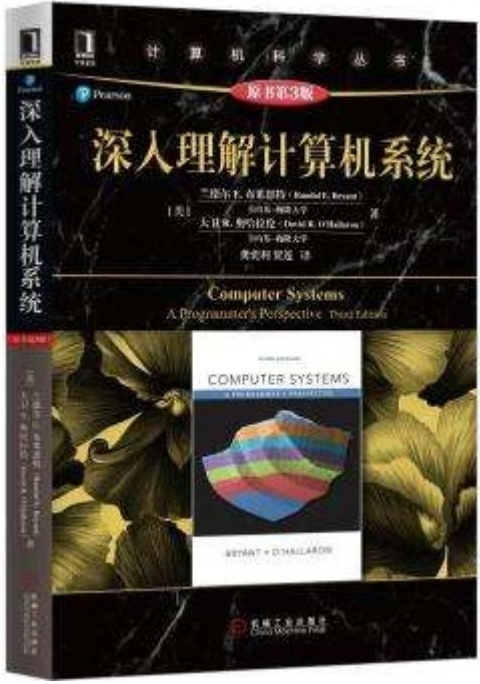
- 培养目标：

  培养学生的系统分析能力，使其成为一个“高效可靠”的程序员，在程序调试、性能提升、程序移植和健壮性等方面成为高手；建立扎实的计算机系统**底层实现基础**，为后续的计算机系统基础、OS、编译、体系结构等课程打下坚实基础。

- 前导课程：C语言程序设计、数字逻辑电路

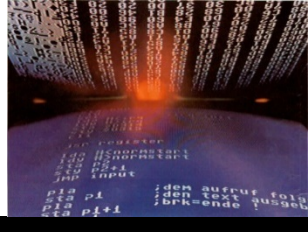
  没有学习前导课程的同学，很难学好该课程。
  时间短、内容多。

课程介绍



- TextBook:
  - 《深入理解计算机系统(原书第3版)》1-3章 作者：（美）Randal E.Bryant　David O'Hallaron 著,龚奕利 贺莲译，机械工业出版社，2016；
- Reference Book:
  - **《计算机系统基础》 1-3章，袁春风，机械工业出版社，2014.7**
  - 《汇编语言程序设计：Professional Assembly Languange》 Richard Blum著 马朝晖等译 机械工业出版社2006

课程介绍

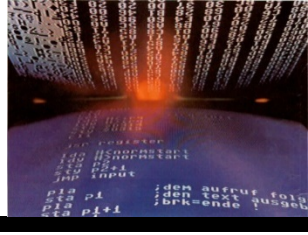


- 上课时间
  - 上午9:00-12:00：线上授课
  - 下午14:00-17:00 ：线上辅导
  - 晚上：课后作业+实验

时间短、任务重！

- 课程考核
  - 平时：20%，作业（按时在线提交）
  - 实验：30%，实验（按时在线提交）
  - 考试：50%，书面考试

课程介绍

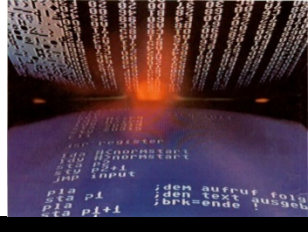


星期	一	二	三	四	五
上午	引言	数据表示与运算	传送、算术和逻辑操作、条件、循环、分支	堆栈、过程、递归和指针数组、结构、联合	内存分配、缓冲区溢出、x86-64
下午	Linux编程基础实验	数据表示与运算实验	分支控制实验	过程数组递归实验	内存分配溢出实验/二进制炸弹实验
晚上	作业1	作业2	作业3	作业4	作业5

暑期完成作业、实验、复习！

开学前考试！！！

在线资源



- 课程网站：http://cslabcms.nju.edu.cn/
- CMU课程：http://csapp.cs.cmu.edu/3e/students.html
- MOOC课程：
  - 中国大学MOOC平台《计算机系统基础（一）--程序的表示、转换与链接》袁春风老师

    https://www.icourse163.org/course/NJU-1001625001
  - **《The Hardware/Software Interface》：https://class.coursera.org/hwswinterface-002**