CS32124: 计算机系统
课程概述

郑贵滨
计算机学院，听觉智能研究中心
综合楼603室

要点

- 课程主题
- 五个事实/现实
- 可执行程序的生成与执行
- 计算机系统层次模型
- 本课程在CS/CE课程体系中的地位

课程主题: (系统) 知识就是力量!

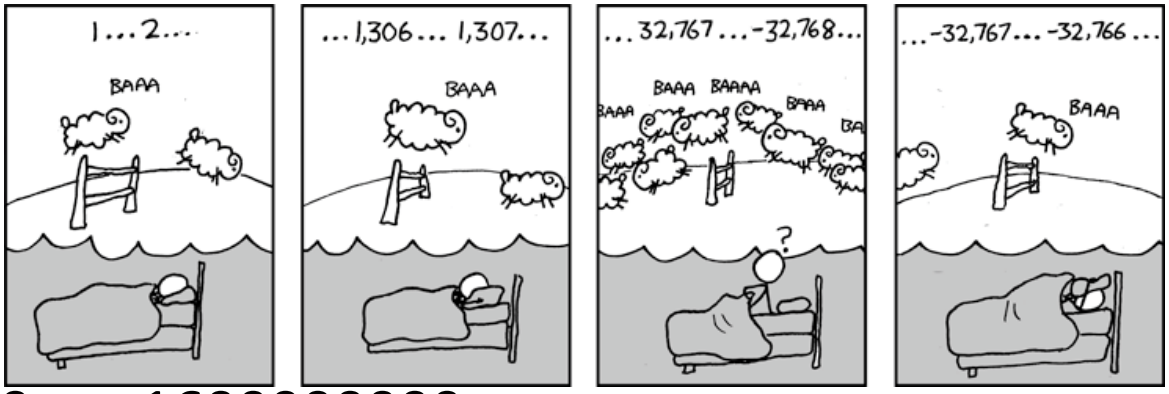
- **(Systems) Knowledge is Power!**
- **系统知识**
  - 系统：硬件(处理器、内存、磁盘、网络)加上 软件(操作系统、编译器、库、网络协议)，共同支持应用程序的运行。
  - 如何充分利用这些资源?
- **学完本课程的有用收获**
  - 成为更高效的程序员
    - 能够发现并有效地排除bug
    - 能理解并调整程序性能
  - 为CS/SE的后续系统课程打基础
    - 编译、操作系统、计算机网络、计算机体系结构、嵌入式系统、存储系统等。

理解系统如何工作很重要！！！

- **为何要学？**
  - 抽象虽好，勿忘现实！
- **多数计算机科学与计算机工程的课程强调抽象**
  - 抽象数据（类）型
  - 渐进分析Asymptotic analysis
- **抽象有局限性！**
  - 特别是在bug（程序缺陷-故障/错误）面前
  - 需要理解底层实现的细节
  - 抽象接口无法提供我们所需层级的控制或行为 (performance)

现实1: int不是整数, float不是实数

- **例 1: $x^2 \geq 0$?**
  - Float's: Yes!
  - Int's:
    - 40000 * 40000 → 1600000000
    - 50000 * 50000 → ??



- **例 2: (x + y) + z = x + (y + z)?**
  - 有/无符号 Int: Yes!
  - 浮点数Float:
    - (1e20 + -1e20) + 3.14 --> 3.14
    - 1e20 + (-1e20 + 3.14) --> ??

Source: xkcd.com/571

计算机的算术运算

- **不生成随机值**
  - 算术运算有重要的数学特性
- **不要假设所有的 “通常”数学特性都成立**
  - 原因：数据表示的有限性
  - 整数操作满足 “环”(ring)特性
    - 交换律, 结合律 , 分配律
  - 浮点操作满足“排序”(ordering)特性
    - 单调性, 符号值
- **观察**
  - 要理解哪一种抽象在哪些上下文中成立
  - 对于编译器开发人员和认真的应用程序员，这些都是重要事项

现实2: 你不得不懂汇编

- **有可能是, 你永远不用汇编语言写程序**
  - 编译器比你更好、更耐心
- **但是: 汇编知识是理解机器级执行模型的关键**
  - 程序有Bug时的行为
    - 高级语言模型会失效
  - 调优程序性能
    - 理解由/不由编译器实现的优化
    - 理解程序低效的根源
  - 实现系统软件
    - 编译器把机器代码作为目标
    - 操作系统要管理进程状态
  - 创造/对抗恶意软件（malware）
    - x86 汇编语言是首选！

现实3: 存储事宜

RAM随机存储器是一个非物理抽象

- **存储器不是无限的**
  - 存储器需要分配与管理
  - 很多应用是存储支配/控制的
- **存储引用错特别要命(有害)**
  - 在时间和空间方面影响深远
- **存储器性能并非一致**
  - Cache与虚拟存储器的效应能大大影响程序性能
  - 针对存储系统的特点，调整程序，能大幅提升速度

例：存储引用Bug

```
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
```

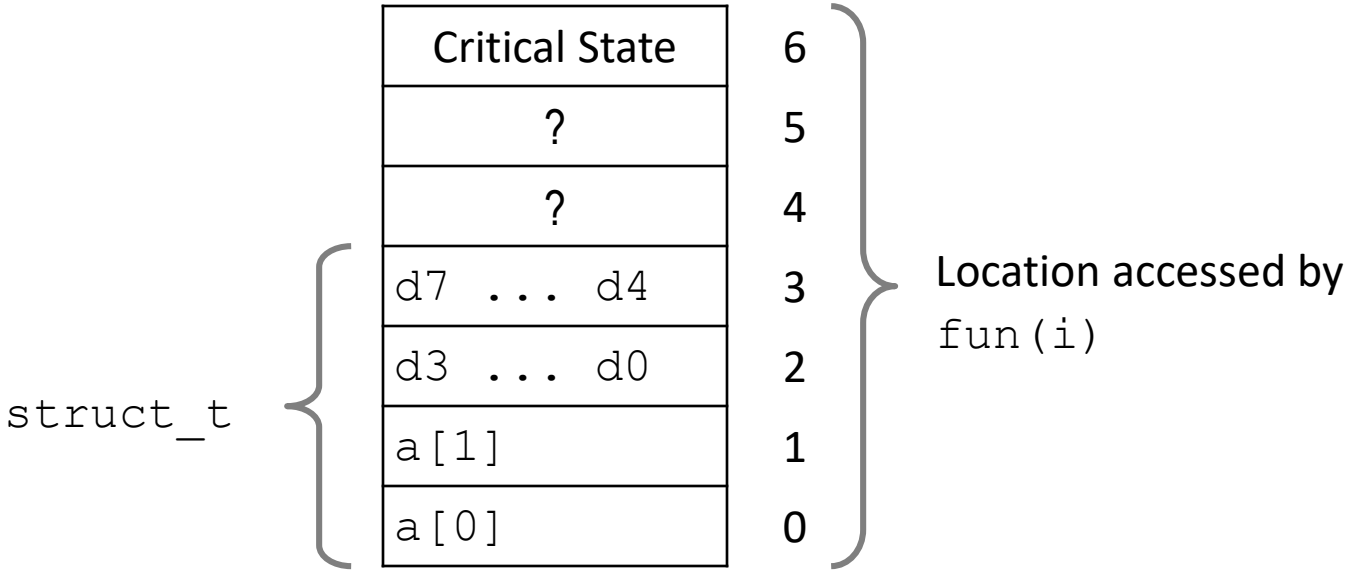
运行结果与机器有关

例：存储引用Bug

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

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

注释:



存储引用错误

- **C and C++ 不提供任何存储保护**
  - 数组访问的越界
  - 无效指针值
  - 滥用 malloc/free
- **导致可恶的bug**
  - Bug是否产生效果，依赖于系统或编译器
  - 远距离的行为(Action at a distance)
    - 崩溃的对象逻辑上与你正访问的不相干
    - 可能在bug生成很久后，才观察到Bug的影响

存储引用错误

- **该如何应对?**
  - 用 Java, Ruby, Python, ML, …编程
  - 理解可能出现的交互(interactions)
  - 使用或自己开发工具来检测引用错 (e.g. Valgrind)

现实4: 性能比渐进复杂性更重要

- （算法速度的）常数因子也重要!
- 即使是精确的操作数量，也无法预测性能
  - 很容易看到，代码编写不同，会引起10:1 性能变化
  - 要多层次优化: 算法、数据表示、 过程、 循环
- 优化性能一定要理解系统
  - 程序是怎么编译和执行的
  - 怎样测量系统性能和定位瓶颈
  - 如何在不破坏代码模块性和通用性的前提下提高性能

例：内存系统性能

```
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];
}
```

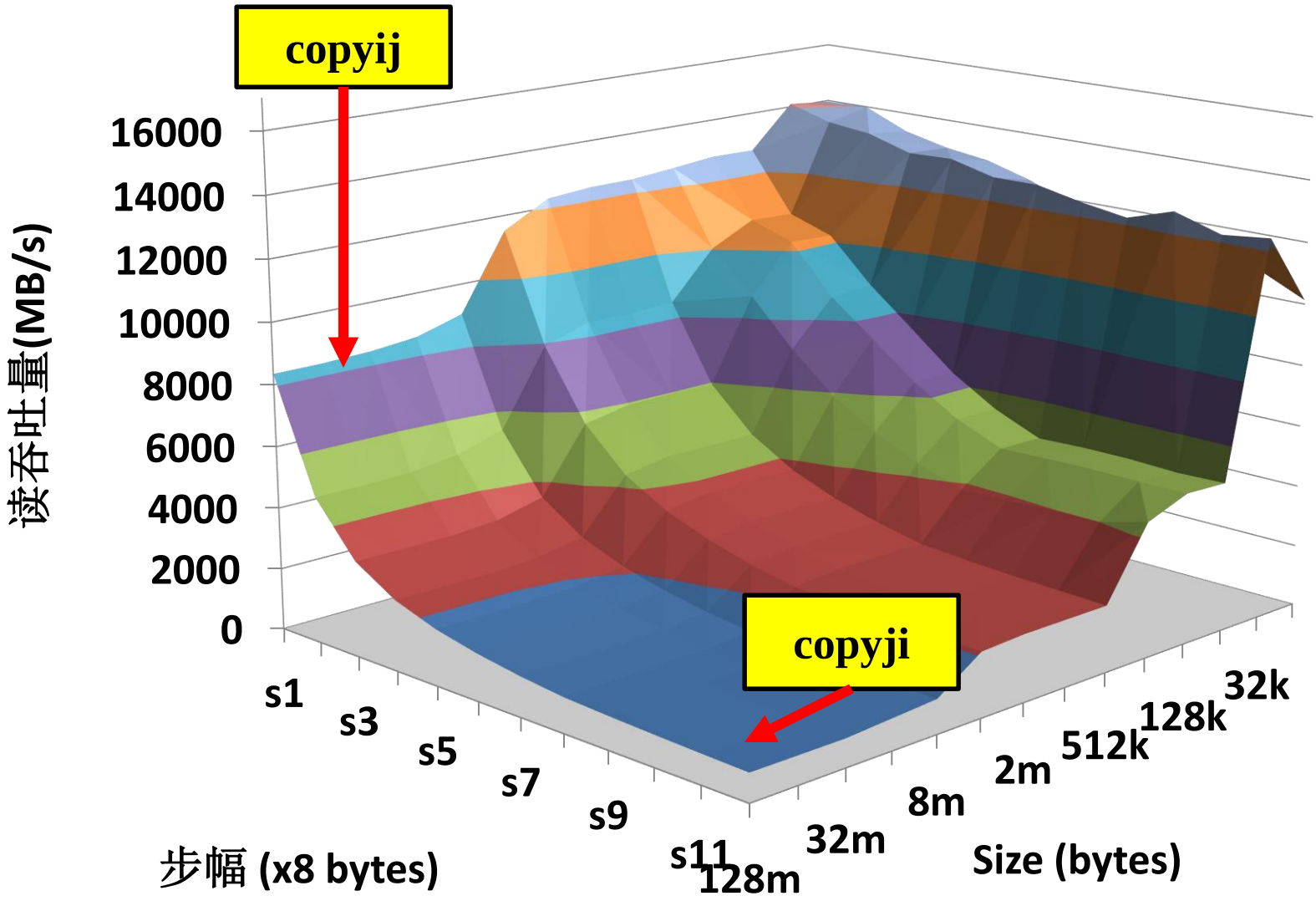
```
void copyji(int src[2048][2048],
            int dst[2048][2048])
{
  int i,j;
  for (j = 0; j < 2048; j++)
    for (i = 0; i < 2048; i++)
      dst[i][j] = src[i][j];
}
```

4.3ms **81.8ms**

2.0 GHz Intel Core i7 Haswell

- 存储器的层次化组织
- 性能依赖于访问模式
  - 包括怎样遍历多维数组

为什么性能不同


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

现实5: 计算机做的事情远比执行程序多

- **要进行数据的输入输出**
  - I/O系统对程序可靠性与性能很关键
- **要通过网络与其他计算机互相通讯**
  - 网络环境下，有很多系统级的问题要解决
    - 自主进程的并发操作
    - 拷贝不可靠的媒体
    - 跨平台的兼容性
    - 复杂的性能问题

课程愿景

- **多数系统课程以"建设"为中心**
  - 计算机体系结构
    - 用Verilog设计流水线处理器
  - OS
    - 实现OS的示例部分
  - 编译器
    - 编写简单语言的编译器
  - 网络
    - 实现并模拟网络协议

课程愿景

- **本门课：以程序员为核心—程序员的视角**
  - 目标：通过更多地理解底层系统，成为更高效的程序员
  - 使你能
    - 编写更加可靠、有效的程序
    - 将需要钩子的特性合并到操作系统中
      - 如, 并发, 信号句柄
  - 这门课包括你们不会在其他地方看到的内容
  - 不是仅仅针对专门黑客的课程
    - 要把隐藏的黑客带到每个人的面前!

课程的关键主题

- Topic 1: 程序与数据
- Topic 2: 存储器层次
- Topic 3: 异常控制流
- Topic 4: 虚拟存储器

可执行程序是怎么生成的？

经典的“ hello.c ”C-源程序

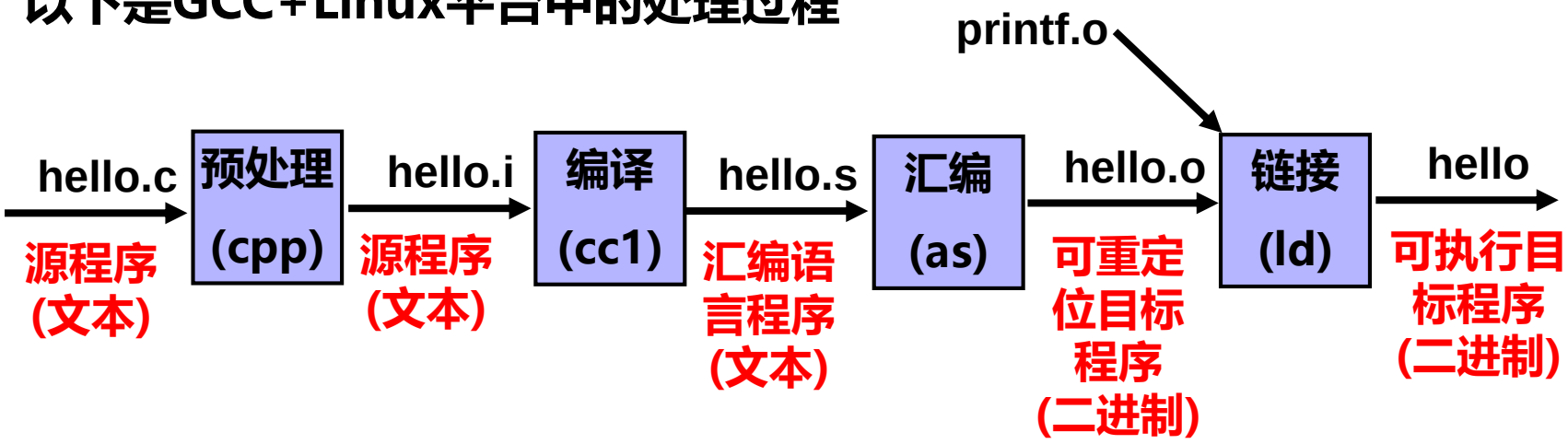
```
#include <stdio.h>

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!

以下是GCC+Linux平台中的处理过程

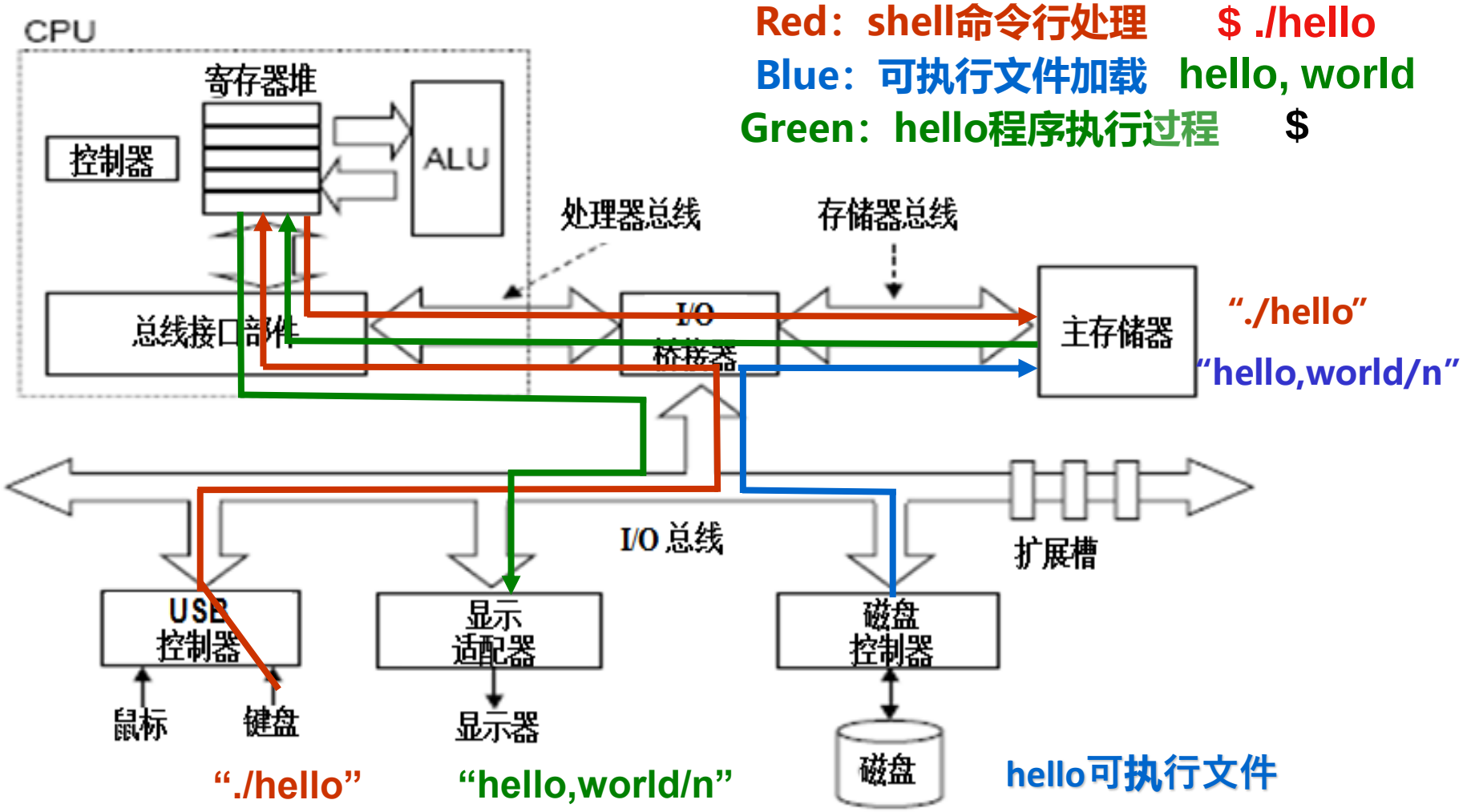


可执行程序是怎么执行的？

Red：shell命令行处理 $ ./hello

Blue：可执行文件加载 hello, world

Green：hello程序执行过程 $



数据经常在各存储部件间传送。故现代计算机大多采用“缓存”技术！

所有过程都是在CPU执行指令所产生的控制信号的作用下进行的。

怎么优化程序？

1. 更快（本课程重点！）
2. 更省（存储空间、运行空间）
3. 更美（UI 交互）
4. 更正确（本课程重点！各种条件下）
5. 更可靠
6. 可移植
7. 更强大（功能）
8. 更方便（使用）
9. 更规范（格式符合编程规范、接口规范 ）
10. 更易懂（能读明白、有注释、模块化）

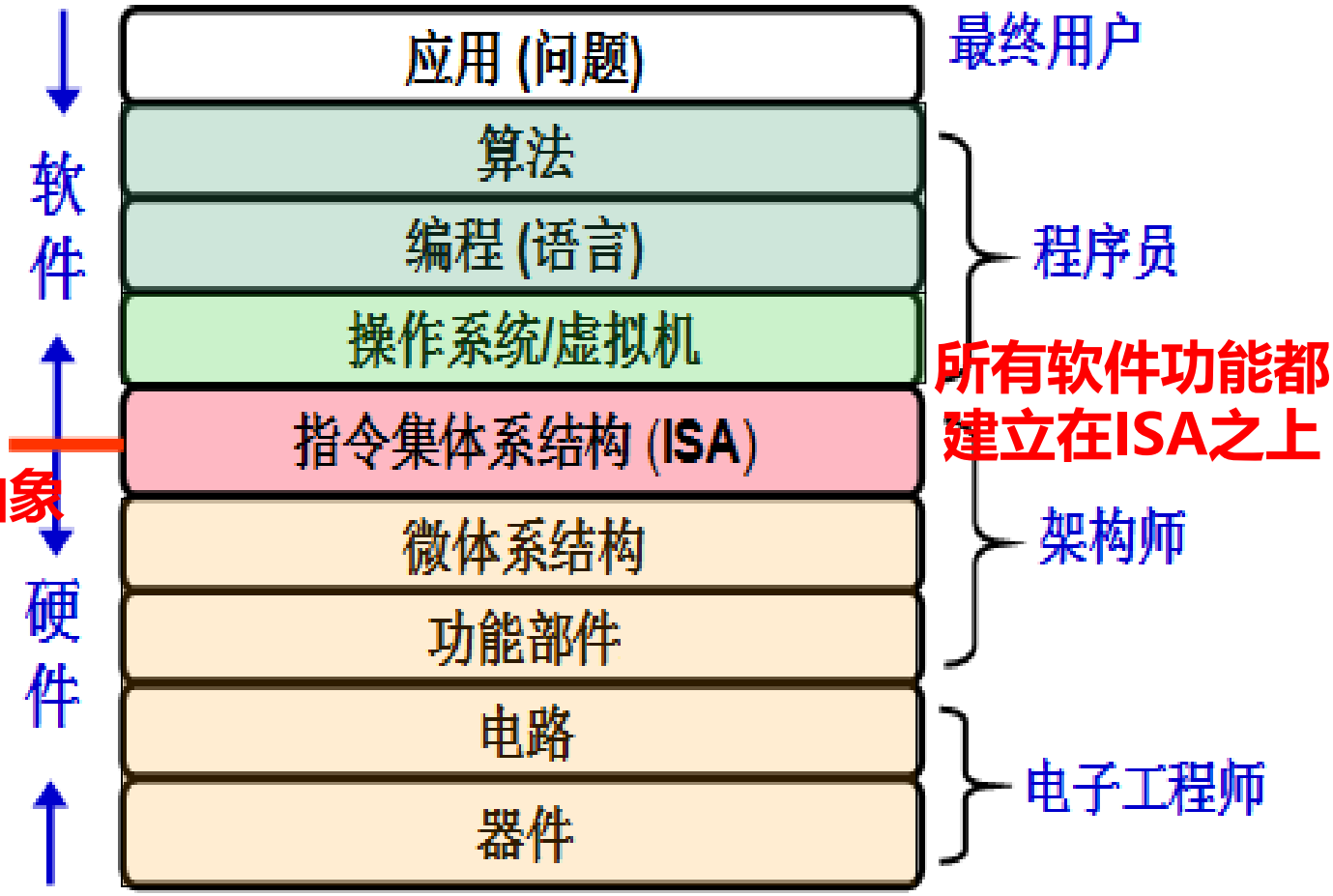
计算机系统层次模型

程序执行结果
不仅取决于
算法、程序编写
而且取决于
语言处理系统
操作系统
ISA-机器语言
微体系结构
ISA是对硬件的抽象

不同计算机课程
处于不同层次

必须将各层次关
联起来解决问题

功能**转换**：上层是下层的**抽象**，下层是上层的**实现**
底层为上层提供支撑环境！



最高层抽象就是点点鼠标、拖拖图标、敲敲键盘，但这背后有多少层转化啊！

Topic1: 程序与数据

- **主题内容**
  - 位操作,算术预算, 汇编语言程序
  - C控制与数据结构的表示
  - 包括体系结构与编译的方面

- **实验**
  - L1:现代计算机系统漫游
    Linux下这种工具的使用、Linux与Windows的对比。
  - L2:数据与代码的机器表示
    二进制炸弹、 代码注入攻击等的基础知识等。

Topic2: 存储器层次

- **主题内容**
  - 存储技术,存储层次，高速缓冲器, 磁盘, 局部性
  - 包括体系结构与编译的方面

- **实验**
  - L3 ： 程序的性能优化
    - 程序的优化方法（使用累积量、循环展开等）
    - 建立一个 cache模拟器，利用局部性优化程序

Topic3: 异常控制流

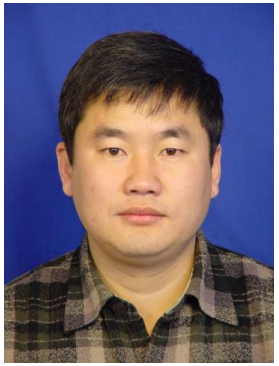
- **主题内容**
  - 硬件异常，进程，进程控制，Unix信号，非局部跳转
  - 包括体系结构、OS与编译的方面

- **实验**
  - L7 (tiny shell lab): 编写自己的 Unix 外壳（引入并发）

Topic4: 虚拟存储器

- **主题内容**
  - 虚拟存储器, 地址翻译, *动态存储器分配*
  - 包括体系结构、OS的方面

教师



刘宏伟

史先俊

郑贵滨

吴锐

教材

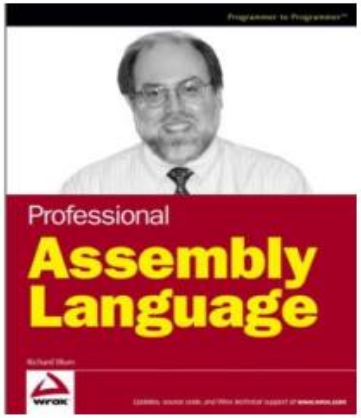
- Randal E. Bryant. 深入理解计算机系统, 机械工业出版社

《*Computer Systems: A Programmer's Perspective*, Third Edition (CS:APP3e)》, Pearson, 2016

  - http://csapp.cs.cmu.edu （教材网站）

- 袁春风 计算机系统基础，机械工业出版社，2019.12

- Richard Blum. Professional Assembly Language



CMU课程 网站

- CMU课程 网站: http://www.cs.cmu.edu/~213
- https://www.cs.cmu.edu/~213/schedule.html
  - 完整的课程计划安排表, 考试， 作业
  - 讲授、作业、测验、答案的拷贝

教学与考核

- **大班讲授**
  - 高层次的概念
- **复习-练习-习题**
  - 应用概念、重要的工具和实验技巧，澄清讲授，考试覆盖相关内容
- **实验: 4个**
  - 课程的关键
  - 每次1-2 周
  - 提供对系统的某方面的深入理解
  - 编程和测试
- **考试**
  - 测试对概念和原理的理解

考核

考核环节	分值比例	考核/评价细则
平时作业	必要条件	5次。学生须完成教师指定的教材中家庭作业内容。教师TA根据每一题标准答案给出每次作业的等级A-D，并按比例汇总本项分值。
大作业	10	按要求完成的各部分的正确性占75分，格式规范10分、条理清晰并重点突出5分、图文并茂10分
实验	10	按照每次实验预习10%、实验操作10%、实验报告80%的比例，汇总每次实验的百分制成绩，并按各实验分值占比，最终统计形成实验总成绩。
考试	80	采用一纸开卷模式，采用填空、选择、判断、简答、分析、综合设计等题型，按学时比例分配各章节的考核知识点与分值，客观题占60%，主观题占40%，并可设置附加题10分。

课后支持...

- **QQ群**

  名称：HIT-CS-2022-ZGB

  群 号：927602994

  密码：hit-cs

群名称:HIT-CS-2022-ZGB

群　号:927602994

- **答疑:**

  周五, 14:00-15:00 综合楼603