# Course Overview 课程概述

Introduction to Computer Systems
1st Lecture, Sep 12, 2016

计算机系统导论
第一讲，2016年9月12日

**Instructors:**
Xiangqun Chen , Junlin Lu
Guangyu Sun, Xuetao Guan
Shiliang Zhang

**教师：**
陈向群，陆俊林
孙广宇，管雪涛
张史梁

# 提纲

- 课程起源
- 课程规划
- 五个有趣的现实问题
- 注意事项

# 课程起源

- 创立：
  - 卡耐基梅隆大学计算机科学学院创立
  - 全球超过180所大学采用了该课程教材、设立了相同或类似的课程
  - 特点：注重实践（程序员视角）、强调对系统的理解


- 发展
  - 2012年，北大信息科学技术学院与卡耐基梅隆大学计算机科学学院联合对该课程进行升级，并正式引入国内

# 合作建设课程

**课程特点：**
**影响面广，关注度高**

# 北京大学的课程规模

- **2010-2011 学年，本科班级规模的初步统计**
  - 20 人以下的班级占所有本科课程的比例仅为3.8%，100 人以上的课程约占27.2%（进一步统计表明，200 人以上的班级占4%）
  - 在全校153 个20 人以下的小班中，大部分是外国语学院课程，大约占67%，其他为公共英语课程
  - 这表明，当时在绝大多数院系中小班教学很少开展

| | 20人以下 | 20~39人 | 40~49人 | 50~99人 | 100人以上 | 合计 |
|---|---|---|---|---|---|---|
| 课程数 | 153 | 1387 | 186 | 1205 | 1093 | 4024 |
| 百分比 | 3.8% | 34.5% | 4.6% | 29.9% | 27.2% | 100% |

# 北京大学本科生“研讨型小班教学”试点

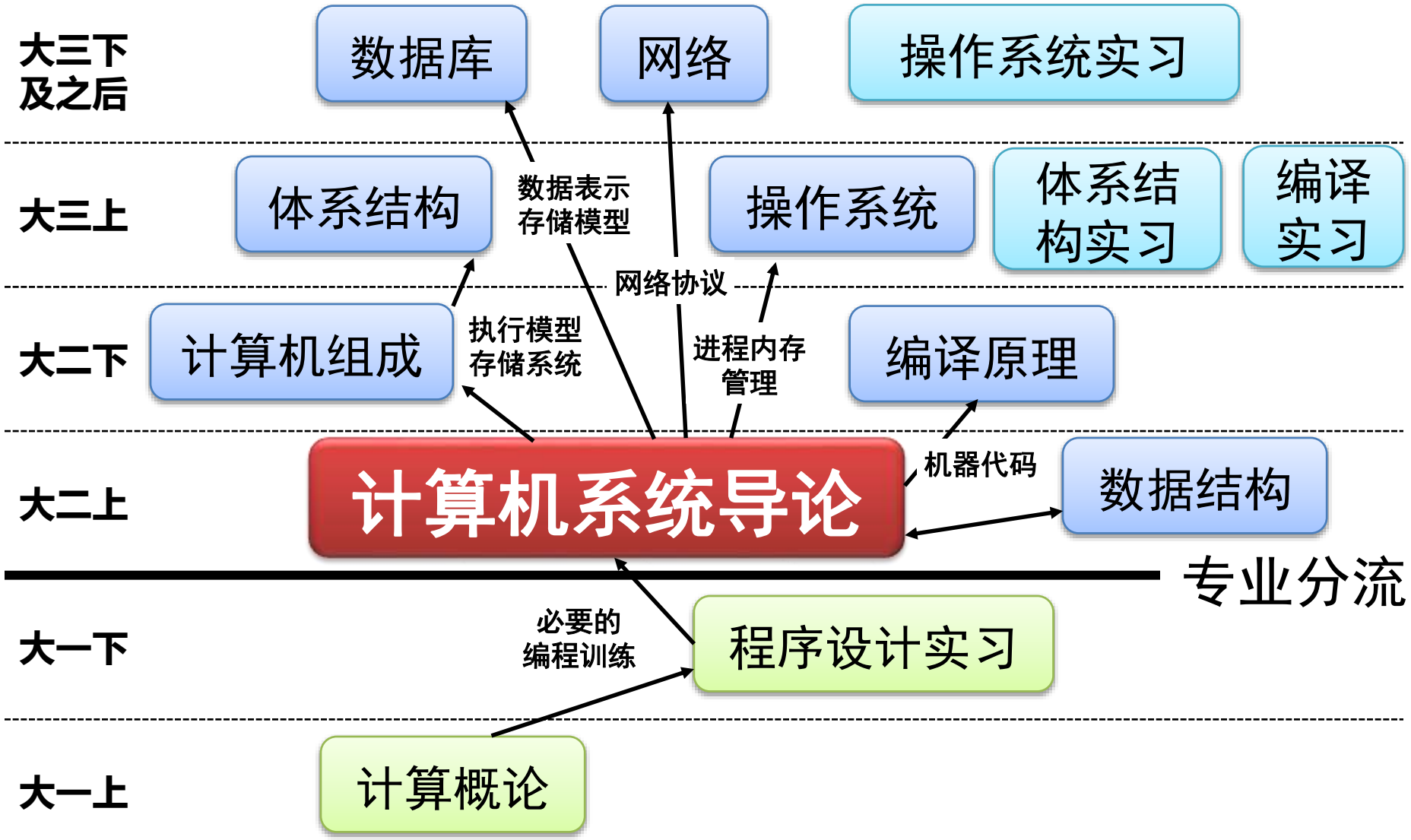
- 2012年秋开展第一批试点：五个学院，六门必修基础课

| | |
|---|---|
| 信息科学技术学院 | **《计算机系统导论》** |
| 数学科学学院 | 《数学分析》、《抽象代数》 |
| 物理学院 | 《量子力学》 |
| 化学与分子工程学院 | 《无机化学》 |
| 生命科学学院 | 《生物化学》 |

# 提纲

- 课程体系
- 课程规划
- 五个有趣的现实问题
- 注意事项

# 本课程在课程体系中的位置



# 本课程的教学方式

- **研讨型教学的两种主要方式**
  - 第一种，一学期由一个教师面对一个小班的学生
  - **第二种，大班讲授课教学同时辅以小班研讨课（本课程的方式）**

每周两次大班授课
- 周一，5~6节
- 周三，1~2节

**共30次大班课**

每周一次小班研讨
- 周四，10~11节

**共14次小班课**

# 重要的时间点

| 周次 | 日期 | 说明 |
| --- | --- | --- |
| 一 | 周一9.12 | 大班第一次课，两班合上 |
| 一 | 周三9.14 | 大班第二次课起，两班分上 |
| 一 | 周四9.15 | 中秋节放假，无小班课 |
| … | … | …… |
| 二 | 周四9.22 | 小班第一次课 |
| … | … | …… |
| 九 | 周一11.7 | 期中考试，两班统考 |
| … | … | …… |
| 十六 | 周三12.28 | 大班最后一次课 |
| 十六 | 周四12.29 | 小班最后一次课 |
|  |  | 期末考试，两班统考 |

# 大班课程安排

- **上半学期**
  - 1班-陆俊林（张史梁），2班-孙广宇
  - 大致覆盖教材第一部分（Part I），即第2~6章
- **下半学期**
  - 1班-陈向群（张史梁），2班-管雪涛
  - 大致覆盖教材第二、三部分（Part II/III），即第7~12章

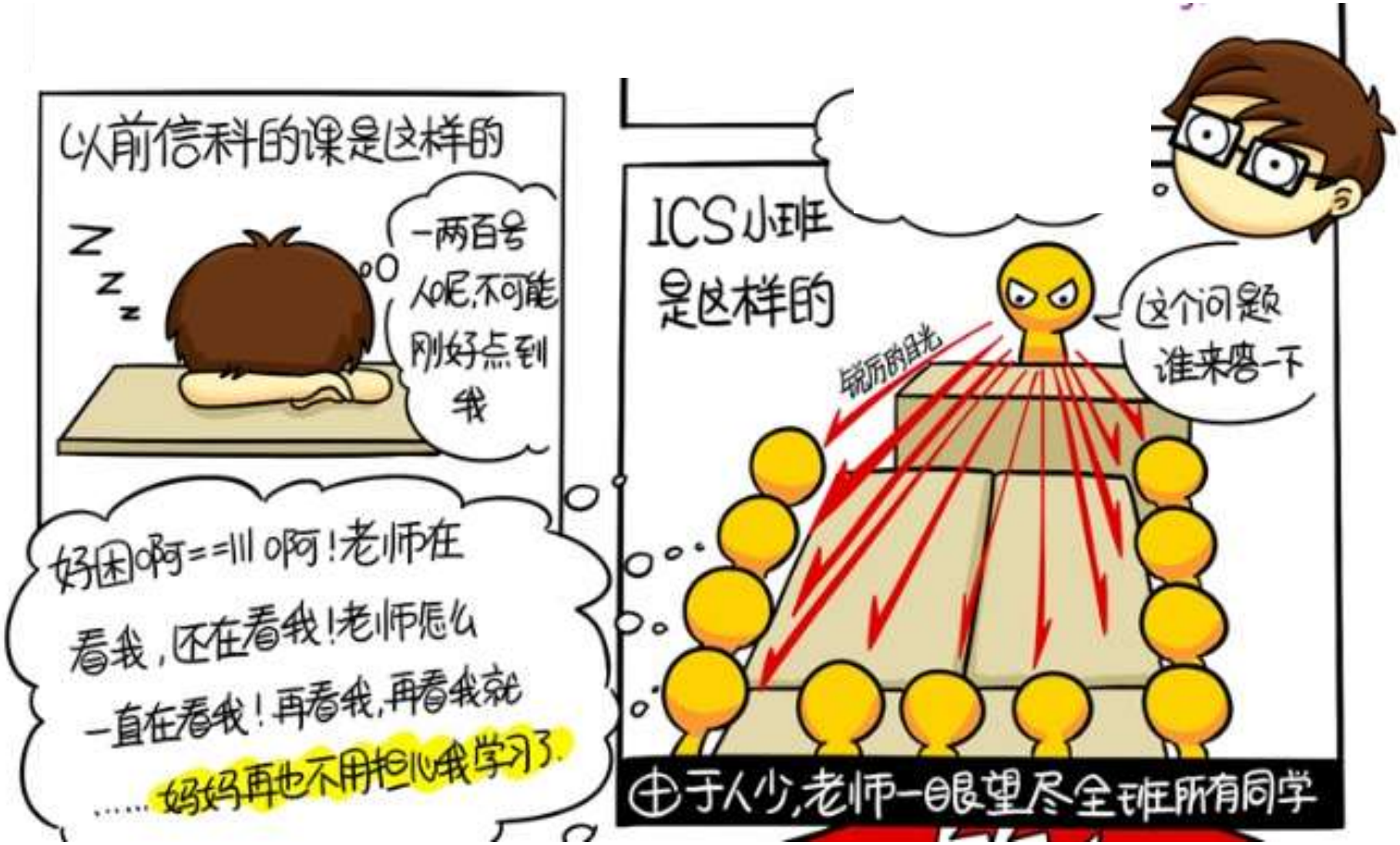

课程特点：
课时多，教学内容多

# 小班课的安排

- 保证教学效果，严格控制小班人数
  - 2012年设立14个小班（每班约13人）
  - 2013年设立16个小班（每班约14人）
  - 2014年设立18个小班（每班约13人）
  - 2015年设立18个小班（每班约13人）
  - 2016年设立19个小班
- 投入大量优秀教师

**2016年秋季小班课教师**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 陈向群 | 6 | 金芝 | 11 | 张铭 | 16 | 英向华 |
| 2 | 陈一峯 | 7 | 李文新 | 12 | 周明辉 | 17 | 黄铁军 |
| 3 | 陈钟 | 8 | 汪国平 | 13 | 易江芳 | 18 | 曹东刚 |
| 4 | 郭耀 | 9 | 汪小林 | 14 | 边凯归 | 19 | 许辰人 |
| 5 | 焦文品 | 10 | 熊英飞 | 15 | 王亚沙 | | |


以前信科的课是这样的
Z z z
一两百号人呢,不可能刚好点到我
好困啊==!!! 啊!老师在看我,还在看我!老师怎么一直在看我!再看我,再看我就……妈妈再也不用担心我学习了.
ICS小班是这样的
这个问题谁来答一下
锐历的目光
由于人少,老师一眼望尽全班所有同学

# 课程特点：
# 大班教学和小班研讨结合

# 实验题系统

**课程特点：**
**学生在指定系统上完成实验题**

- **大型特色实验题**
  - 从实际问题出发
  - 具有很强的趣味性
  - 平均每两周完成一个

- **新颖的“实验题智能评价系统”**
  - 自动根据性能、时间、提交次数等对学生提交的实验题进行评分
  - 实时公开发布所有同学完成情况并分步分题进行比对，鼓励学生对实验的钻研

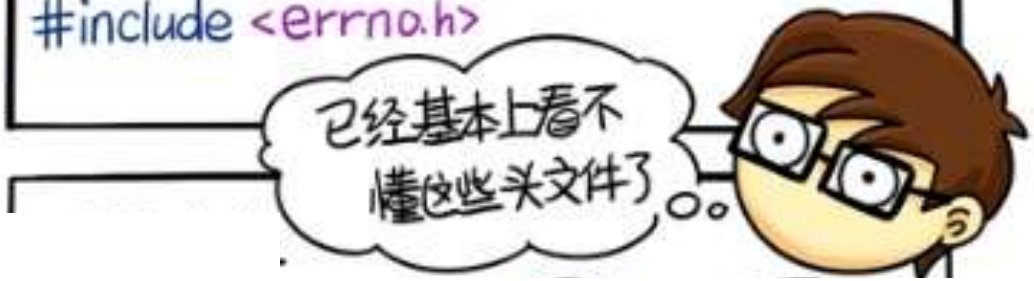
以前作业的头文件是这样的

```
#include <iostream>
#include <string>
#include <iomanip>
#include <cstdio>
```

这是"魔兽世界"大作业的头文件

上了ICS后头文件是这样的

```
#include <assert.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <ctype.h>
#include <signal.h>
#include <sys/types.h>
#include <fcntl.h>
#include <sys/wait.h>
#include <errno.h>
```




以前周末相遇是这样的
吃了吗？
杏锅走起
没呢
果断的!!!


上了ICS后变成了这样
你lab做了吗?
我也没做.呵呵
果断没做
呵呵ㄇㄇ

囧囧有神的北大生活

# 提纲

- 课程体系
- 课程规划
- 五个有趣的现实问题
- 注意事项

# 本课程关注的问题和目标

- **本课程关注的问题：**
  - 计算机抽象概念与实际计算机系统之间的差异
  - 计算机抽象概念在实际计算机系统上的实现方式

- **本课程的目标：**
  - 为初入计算机专业的学生建立计算机系统的整体知识框架
  - 训练学生养成良好的编程习惯，进而具备更为高效的编程能力，尤其是提高程序的性能、可移植性和健壮性等方面
  - 为学生后续学习编译、网络、操作系统、计算机体系结构等专业课程奠定基础

# 本课程独特的视角

- **本课程是从编程者角度出发，描述计算机系统如何执行程序、存储信息和通信**

- **涵盖计算机系统从上到下的多个层次，包括：**
  - 机器语言及其如何通过编译器优化生成
  - 程序性能评估和优化
  - 存储结构组织和管理
  - 网络技术和协议
  - 并行计算的相关知识

# 问题1：整型不是整数，浮点型不是实数
# Ints are not Integers, Floats are not Reals

- **例1.1: $x^2 \geq 0$永远成立吗?**
  - 如果 x 是浮点型，成立
  - 如果 x 是整型
    - 40000 * 40000 → 1600000000
    - 50000 * 50000 → **负数**，因为整型有上界溢出

- **例1.2: 是否满足结合律 (x + y) + z = x + (y + z)?**
  - 如果x, y, z是整型，满足结合律
  - 如果x, y, z是浮点型
    - (1e20 + -1e20) + 3.14 → 3.14
    - 1e20 + (-1e20 + 3.14) → **0**，因为浮点数精度不同不满足结合律

# 计算机系统中的算术 ≠ 数学中的算术(1/2)

- **整数性质**
  - 交换律: a+b = b+a
  - 结合律: (a+b)+c=a+(b+c)
  - 分配律: a·(b+c)=a·b + a·c
  - 整型运算满足以上性质

- **实数性质**
  - 单调性：if a ≥ b, c ≥ 0, then (a+c) ≥ (b+c)
  - 浮点型运算满足单调性

# 计算机系统中的算术 ≠ 数学中的算术(2/2)

- **有些性质在计算机系统中并不成立**
  - 计算机系统只能表示“**有限大小的数**”：溢出问题（例1.1）
  - 浮点型不满足结合律：**舍入操作会造成精度误差**（例1.2）
  - 需要记住计算机中不同数据类型所满足的数学性质
  - 对编译器和科学计算程序员尤为重要：因为缺少一些数学性质会使得解决某些简单问题变得麻烦。
- **例1.3：**
  - 两个整型a和b是否相等：a == b 
  - 两个浮点型a和b是否相等：a == b 
    - 因为两个数精度可能不同
    - 正确方法——作差取绝对值

      fabs (a-b)<= epsilon, (epsilon是很小的数，如0.00001)

# 问题2：了解汇编 (1/4)

## You've Got to Know Assembly

**可能你永远都不会去写汇编程序，但是……**

**有助于了解机器层面的程序执行模型**

- **帮助查找底层实现相关的程序错误（bug）**
  - **例2.1**：比较整型(int)、无符号整型(unsigned int)
  - d = −1 < TOTAL=12，理应输出small，但结果却是large
    - sizeof()的返回值是unsigned int；
    - if语句作比较时，编译器认为−1是unsigned int (很大的整数)
  - 通过底层汇编代码/目标程序文件(二进制文件)查看 d 的数值

```
int array[] = {1,2,3};
#define TOTAL sizeof(array) /* unsigned int */
void main() {
    int d = -1;
    if (d <= TOTAL)
        printf("small\n");
    else printf("large\n");
}
```

# 问题2：了解汇编 (2/4)
# You've Got to Know Assembly

- 程序性能调优
  - **例2.2**：尝试不同代码写法，分析比较不同的底层汇编代码效率
  - 两个程序似乎有相同的行为。但是fun2的效率会更高
  - 通过底层代码可以看出，fun1需要6次存储器引用，而fun2只需3次

```
void fun1(int *x, int *y)
{
    *x += *y;
    *x += *y;
}
```

```
void fun2(int *x, int *y)
{
    *x += 2* (*y);
}
```

# 问题2：了解汇编 (3/4)
# You've Got to Know Assembly

- 系统软件或嵌入式软件开发
  - 例如系统软件工程师往往会要求写小段汇编代码
  - **例2.3**：把小段汇编代码加入C代码，来访问硬件（处理器）上的周期计数器（cycle counter）。

```
static unsigned cyc_hi = 0;
static unsigned cyc_lo = 0;

/* Set *hi and *lo to the high and low order bits
   of the cycle counter.
*/
void access_counter(unsigned *hi, unsigned *lo)
{
    asm("rdtsc; movl %%edx,%0; movl %%eax,%1"
        : "=r" (*hi), "=r" (*lo)
        :
        : "%edx", "%eax");
}
```

# 问题2：了解汇编 (4/4)
# You've Got to Know Assembly

- 防范恶意软件或分析第三方软件的安全性
  - 分析没有源代码的软件时，需要进行**反汇编**
  - 常见的安全漏洞包括：缓冲区溢出、内存泄露、非授权内存写入等
  - 对反汇编得到的代码进行**静态分析**，是一种找到已知安全漏洞代码的有效手段
  - **例2.4**：定位 gets() 这样不安全函数对应的汇编代码

```
void main{}
{
    char buf[1024];
      gets(buf);
    /*用户输入不做限制，缓冲区溢出*/
}
```

```
#define BUFSIZE 1024
void main{}
{
    char buf[BUFSIZE];
      fgets(buf, BUFSIZE, stdin);
    /*限制输入大小的参数*/
}
```

# 问题3：内存对程序性能的影响至关重要

## Memory Matters

## Random Access Memory Is an Unphysical Abstraction

- **内存是有限的**
  - 必须合理地分配和管理内存
  - 很多程序受限于内存
- **内存引用错误尤为严重**
  - 错误的危害因时间、空间而异
- **内存性能并不是始终如一的**
  - 高速缓存和虚拟内存极大地影响程序性能
  - 根据存储系统的特点，可以对程序进行调优(见问题4)

# 内存引用错误 (1/3)

```
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
```

- Result is system specific

# 内存引用错误 (2/3)

```
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
```

**Explanation:**

# 内存引用错误 (3/3)

- **C 和 C++ 并没有提供对此类错误的防范机制，比如：**
  - 数组越界错误
  - 指针错误
  - 滥用 malloc/free 函数

- **应对措施**
  - 用其他语言编程，例如 Java, Ruby, Python, ML
  - 使用工具来检测此类内存错误

# 问题4：算法性能分析结果 ≠ 实际程序性能

## There's more to performance than asymptotic complexity

- 代码写的好坏与否，可能导致程序性能的数量级差别
- 程序性能优化有多个层面
  - 算法，数据表达，过程，循环
- 只有理解了系统实现才能做到有效优化
  - 衡量程序性能的指标：执行时间、内存占用、能耗等。
  - 了解程序的编译、执行过程中的细节，如内存访问模式
    - 例：内存访问模式影响程序性能

# 内存性能影响程序性能

```
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];
}
```

**4.3ms**

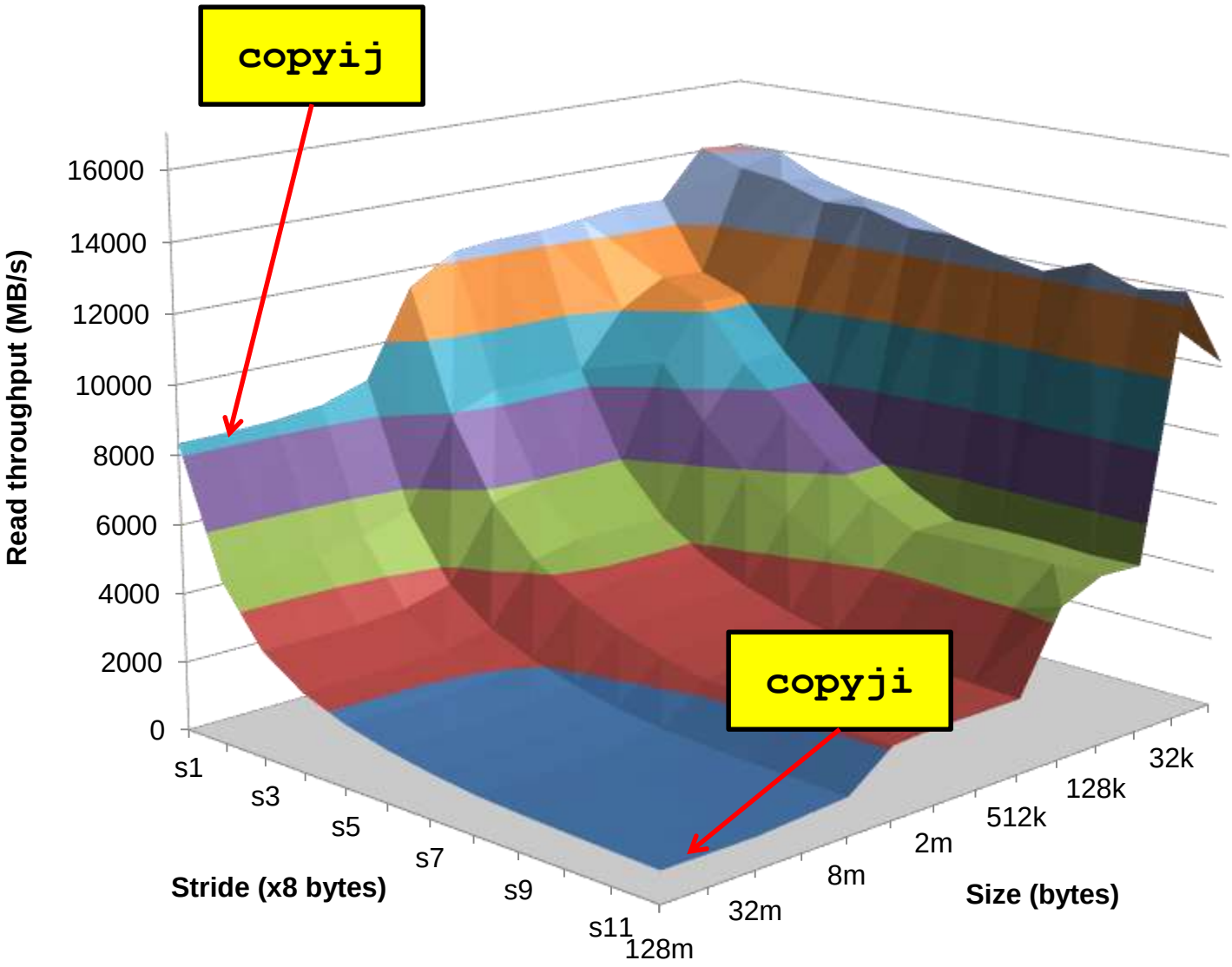
```
void copyji (int src[2048][2048],
             int dst[2048][2048])
{
  int i,j;
  for (j = 0; j < 2048; j++)
    for (i = 0; i < 2048; i++)
      dst[i][j] = src[i][j];
}
```

**81.8ms**

2.0 GHz Intel Core i7 Haswell

- 内存是分层组织的
- 程序性能取决于内存访问模式
  - 例如：如何访问内存中的二维数组、多维数组

# 为什么性能有这些差别

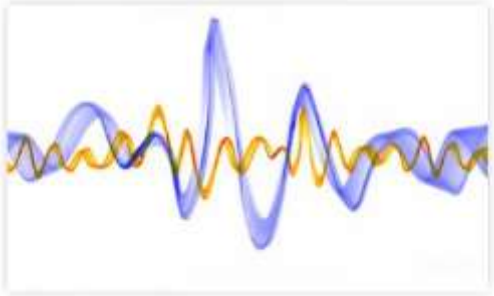

copyij
copyji
Read throughput (MB/s)
16000
14000
12000
10000
8000
6000
4000
2000
0
s1
s3
s5
s7
s9
s11
Stride (x8 bytes)
32k
128k
512k
2m
8m
32m
128m
Size (bytes)

# 问题5：计算机网络环境下的新问题
# Computers do more than execute programs

- **计算机需要输入和输出数据**
  - 程序执行前，需要输入数据
  - 程序执行后，需要输出结果
  - 在**网络环境**下，数据输入来源
    - 本地磁盘
    - 网络中别的计算机。例如，利用上传数据到服务器，利用服务器的超强计算能力做仿真实验
- **I/O 系统对程序稳定性和性能至关重要**
  - 如果缺少I/O异常处理能力，就会出现程序运行错误

# 课程主体内容

① 程序与数据 Programs and Data
② 处理器体系结构 Processor Architecture
③ 程序性能 Performance
④ 分级存储器体系 The Memory Hierarchy
⑤ 异常控制流 Exceptional Control Flow
⑥ 虚拟内存 Virtual Memory
⑦ 网络、并发 Networking, and Concurrency



# 提纲

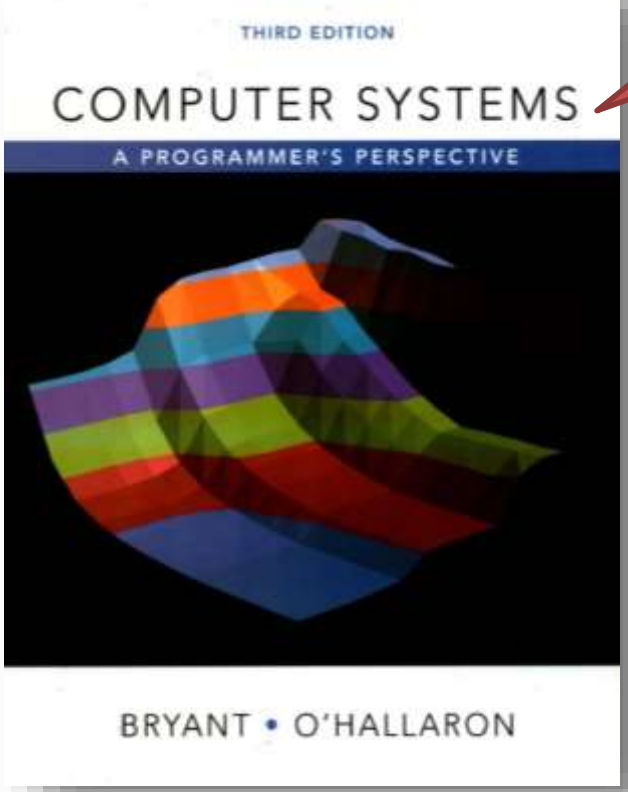
- 课程体系
- 课程规划
- 五个有趣的现实问题
- 注意事项

# 课程主页

http://course.pku.edu.cn

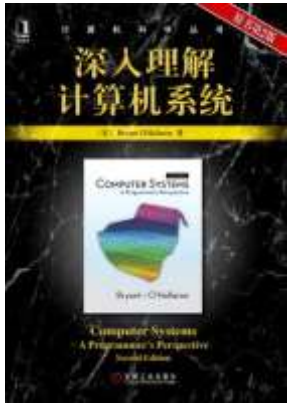
# 课程教材

- Computer Systems: A Programmer's Perspective（3rd Edition）
  深入理解计算机系统（英文版·第3版）
- 英文版作者：（美）Randal E. Bryant / David O'Hallaron

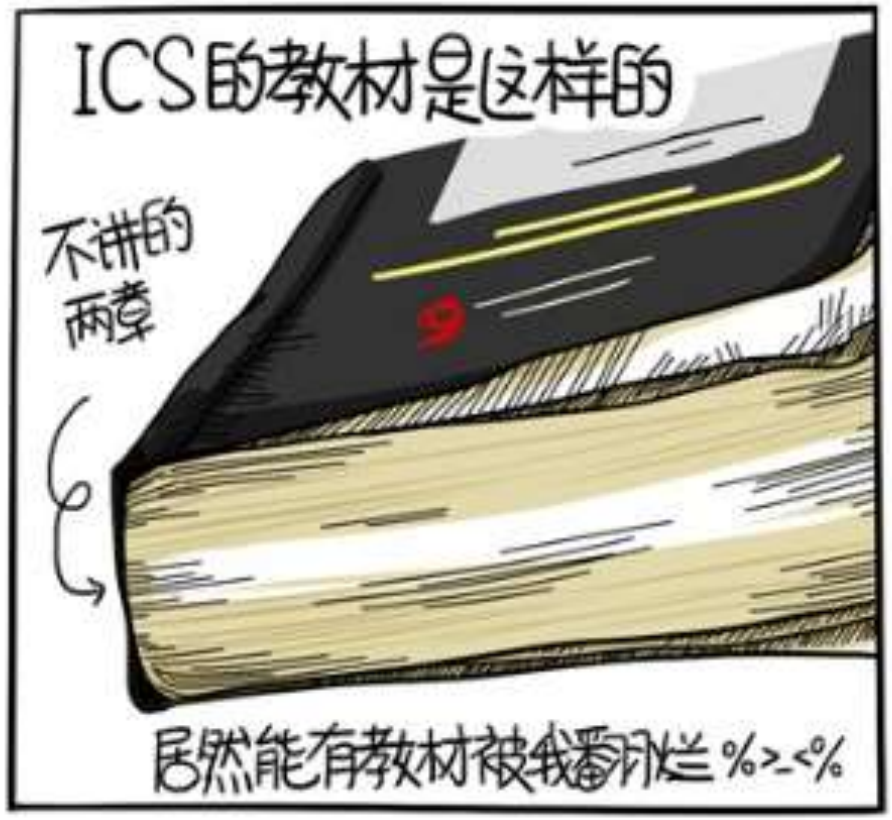



从本学期开始，教材升级到第3版

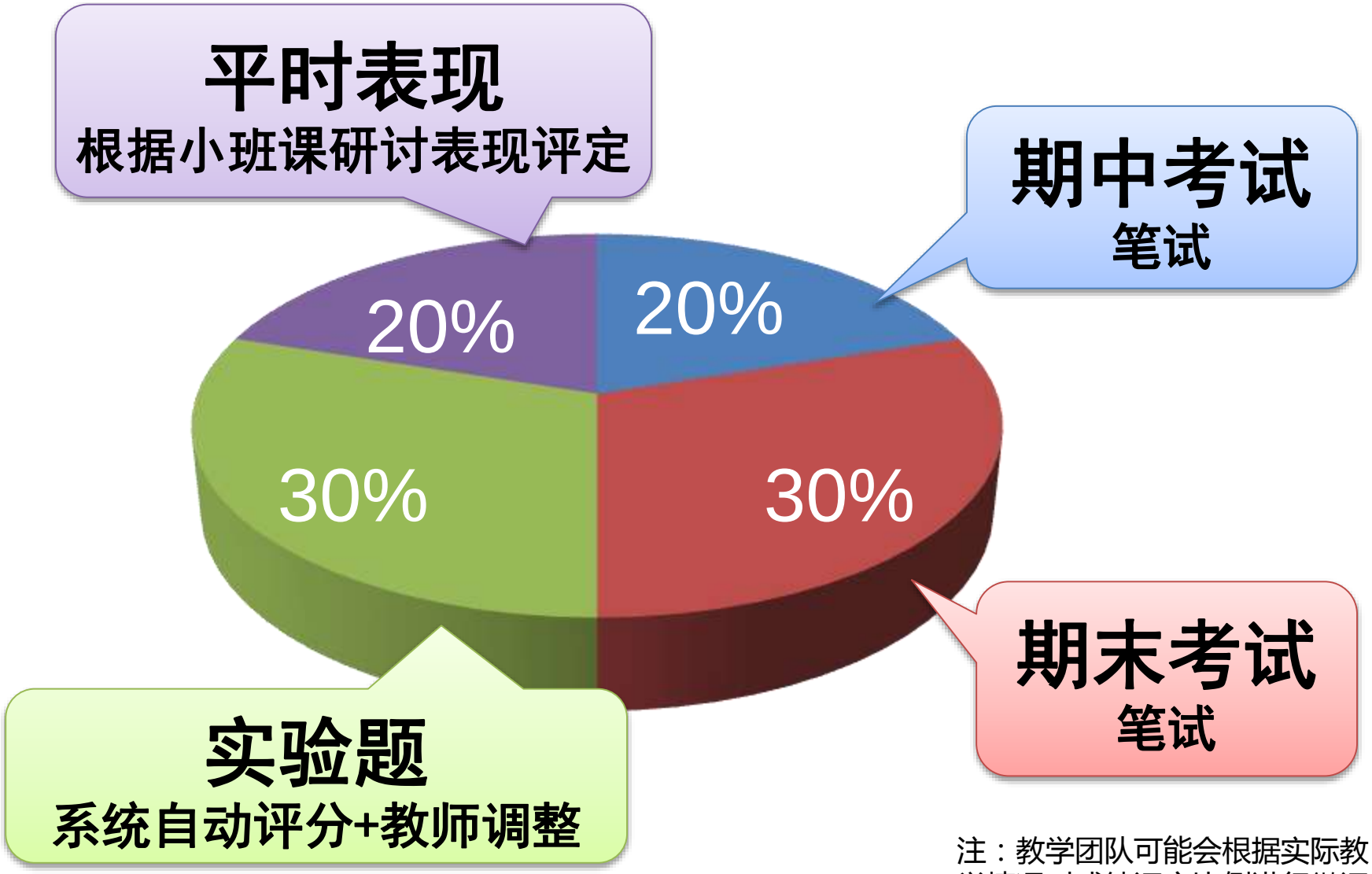
这些是第2版


以前教材是这样的
这不是我开学时
买的《xxxx》吗
我期末都考完
了居然一
页都没看
崭新


ICS的教材是这样的
不讲的
两章
居然能有教材被我看到烂%>_<%

# 成绩评定




注：教学团队可能会根据实际教学情况对成绩评定比例进行微调


以前期末考完是这样的
跪了
挂科了挂科了
我要挂科了
学渣
学霸


ICS期末考完是这样的
等会跪着爬出去
吃香锅好了


考试时间还
延长了半小时
我一个小时就
没有会做的题
了,后一个半小时
就一直在对着子
干瞪眼
完全不用担心考得差
反正几乎都不会做
真是考得神清气爽
今天真是
人逢喜事精神爽
一定要好好吃一顿
真实的故(cǎn)事(jù)!!!

囧囧有神的北大生活


囧囧有神的北大生活

2013-08-21


WELCOME