The first seminar of ICS, 2021 Fall

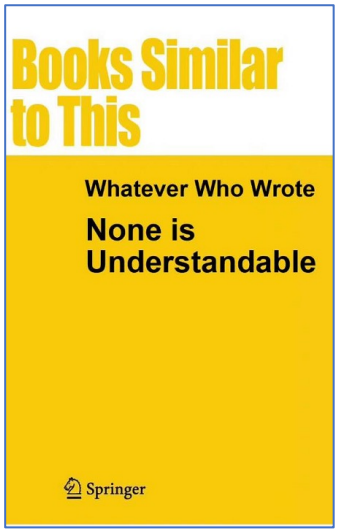
Class 07 (Experimental), TA: Rui Ding
2021-09-19

自我介绍

- **18级计算机系本科生 丁睿**

- **联系方式**
  - WeChat_id　　abcisnotabc
  - E-mail　　dr_abc@pku.edu.cn
  - QQ　　2104843292

- **爱好**



日系本格推理
e.g. 东野圭吾系列 + 早期柯南漫画/TV/特典

吃货
就是太瘦hhh

划水
你想多了，我说的是游泳

Introduction to

Introduction to Computer System

漫游 #1/2

- **第二章：信息表示与处理**

  整数/实数怎样表示？ 代码与数据的底层储存?

- **第三章：汇编语言**

  C语言到可读机器语言的映射?

- **第四章：处理器结构**

  电路和电器元件怎样实现指令？

- **第五章：程序优化**

  利用现代编译器的特点优化程序效率

- **第六章：存储器层次结构**

  磁盘 / 内存 / 缓存 / 寄存器…… 数据究竟被存储在哪？

漫游 #2/2

- **第七章：链接**

  多个源代码怎么被整合成一个可执行文件？

- **第八章：异常控制流**

  程序真的在不间断地执行吗？ 为什么可以边刷树洞边听歌？

- **第九章：虚拟内存**

  为什么需要虚拟的内存空间？谁在使用？怎么映射到物理内存？

- **第十章：文件、输入/输出**

- **第十一章：网络入门**

- **第十二章：并发**

Lab #1/2

- **DataLab: 数据位运算实验 (2)**

  难度：★★★★★　　代码量：★★

- **BombLab: 二进制炸弹实验 (3)**

  难度：★　　代码量：★

- **AttackLab: 可执行文件攻击实验(3)**

  难度：★★　　代码量：★

- **ArchLab: 处理器实验 (4, 5)**

  难度：★★★　　代码量：★★★

Lab #2/2

- **CacheLab: 内存访问性能优化实验 (6)**

  难度：★★★★　　代码量：★★★

- **TshLab: 简易shell实验 (8,10)**

  难度：★★★★★☆　　代码量：★★★★★

- **MallocLab: 动态内存管理实验 (9)**

  难度：★★★★★　　代码量：★★★★★☆

- **ProxyLab：网络代理实验 (10, 11, 12)**

  难度：★★★　　代码量：★★★★★

More About Labs #1/3

- 几个时间的含义
  - Handout Due Deadline
  - Due之前尽量完成! Due之后的副作用稍后说！
  - Graceday

周次	周一周四大班	节次	主题	周二小班	节次	LAB日程
一	9月13日	1	Overview	9月14日	1	
二	9月16日 9月20日	2 3	Bits and Bytes/Integers Floating Point	9月21日	中秋	L1 (datalab) out
	9月23日	4	Machine Prog: Basics			
三	9月27日	5	Machine Prog: Control	9月28日	2	L2 (bomblab) out
	9月30日	6	Machine Prog: Procedures			
四	10月4日		国庆节放假	10月5日		
五	10月7日 10月11日	 7	国庆节放假 Machine Prog: Data	10月12日	3	L3(attacklab) out
	10月14日	8	Machine Prog: Advanced			
六	10月18日	9	Processor Arch: ISA&Logic	10月19日	4	L4 (archlab) out
	10月21日	10	Processor Arch: Sequential			
七	10月25日	11	Processor Arch: Pipelined	10月26日	5	
八	10月28日 11月1日	12 13	The Memory Hierarchy Cache Memories	11月2日	6	
	11月4日	14	Program optimization			
九	11月8日	15	期中考试	11月9日	7	L5 (cachelab) out

More About Labs #2/3

- **怎样不算抄袭？**
  - 课本代码随便用
  - 除了课本代码，其余代码不要看，包括同学代码、网上代码、往届学生代码
  - 可以问同学某个方法有没有用
- **实在做不出来怎么办？**
  - 找助教要提示
  - 千万不要面向CSDN/Github/StackOverflow/…编程!
- **早点开始做，意想不到的事情很多!**
- **多数 lab 提交次数有限制，取最后一次提交为lab成绩**
- **用好每个lab的Writeup，最好写之前能有规划!**
- **未必每个lab都要肝到满分，MallocLab满分极难！偶尔有个90/100不影响总评！**

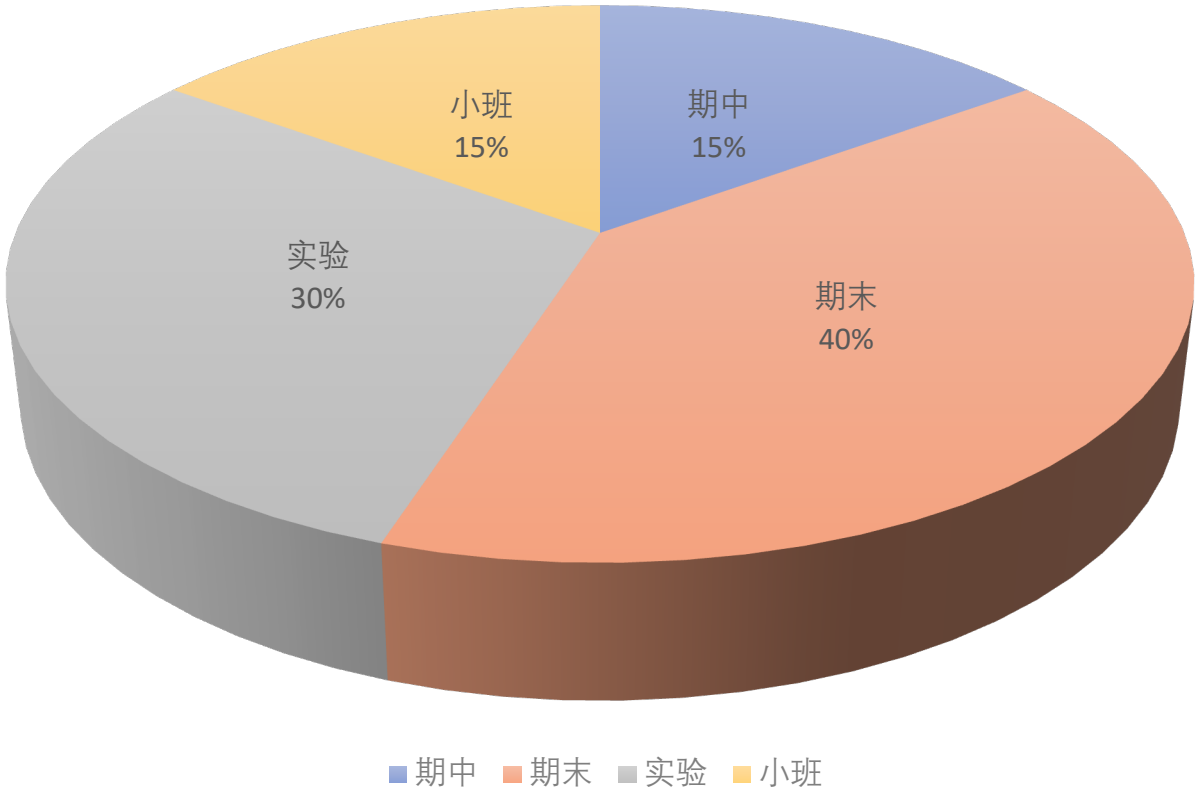
  -> 自己可以算成绩组成合理取舍，**每个lab占总评的3%**

More About Labs #3/3

- **测试环境 autolab.pku.edu.cn**
  - 服务器环境是Ubuntu18.04
  - 部分测试性能的lab都以autolab的最后测试结果为准。Performance可能轻微受到服务器压力影响。
  - 提交结果有问题请联系助教查看后台日志。
  - 请勿恶意攻击，如有发现后果极其严重！
  - 每个lab刚发布时可能不稳定，建议稍后再下载lab作业(例如半天)，不必争抢。
- **如有可能，建议肝lab期间及时行乐。有时深陷bug不能自拔，放松后再调试却柳暗花明。**
- **Lab也没有那么可怕。用好课本和Writeup，大部分的问题即刻破解!**
- **Lab同学讲评:**

  简单的lab可以讲讲基本原理和实现，复杂的lab可以讲自己的设计和实现过程中踩过的坑。非常自由! 只要反映出自己确实认真做了lab并准备了讲评即可。

分数组成



- 期末大约30%的内容是前半学期的

小班要求

- **专题讨论**
  - 小班课得分评定的**大头**。具体占比待定。提出好的讨论课题和积极参与讨论同等重要。助教每周也会针对初学者的疑难杂症列出论题。助教会在当周周六前给出下一周的论题。同学们提出的课题也请尽量在此时间之前。
  - 仅按个人评分
  - 问题导向型的讨论——不再针对课本过每一个细节，而且着重其中难以理解的、容易被忽略的或者有前后联系的部分。
  - 助教仅提供必要的补充解释。每节课都会有一位同学成为讨论的主持人。请主持人在周日前形成讨论草稿（可以非常粗糙和high-level），周日或周一约时间与助教讨论草稿。草稿往往需要经过多次修改。请主动联系助教而不是等待被催促。欢迎主讲人点名同学来回答。
- ~~回课~~
  - 最后全员 ppt / csapp reader
  - 你们也不用挤破脑袋胡 ppt 了

小班要求

- **延伸阅读**
  - 因属拓展部分，**不计分**。
  - 结合课程进度和负担程度适当安插相对前沿的论文和教材阅读
  - 主要来源：caaqa (量化研究方法) 和 系统&网络顶会高引论文
  - 不会每周开展。如需开展，助教会提前一周在周一时布置下来。请大家务必在对课本有基本的了解后阅读。
- **出勤**
  - 小班得分**小头**。无特殊理由迟到十五分钟以上一次扣0.5分；无特殊理由缺勤扣1分。扣完本部分分为止。
- **Lab总结**
  - 指派到人，此部分**不算分**。纯粹自愿。希望大家参与讨论。
  - 除了对 lab 涉及到的技术性、技巧性、工程性的关注，也可以针对做lab的个人体验提出建议。不限于代码本身，包括 writeup 里误导性的或者不全面的地方。
  - Talk is cheap, show us the code.

Master the tools: Do as a CSer

- To fully exploit the facility and efficiency provided by modern computers, you still have to pick up these tools as in the old days which cannot be interplayed with a graphical button or dictated by your voice.
- To save you from repetitive tasks. Many of them can be automated. E.g.
  - Run sth. for 10 times.
  - Sequentially execute several commands from a text document without copy-pasting line by line.
  - Maintain historical copies of a file without creating xx1.cpp xx2.cpp xx3.cpp…
- **Selected topics:**
  - Command-line tool: **Shell (2 weeks)**
  - ~~Command-line text editing: **Vim (2 weeks)**~~
  - ~~Version Control: **Git (2 weeks)**~~
  - Metaprogramming: **Automake & Make** (Highly effective, though it is not likely to be used in this course) **(2 weeks)**
  - A full version can be found on **The Missing Semester of Your CS Education**

Shell #1/2: Basic Commands

- What is a shell?
- Basic shell commands
  - Navigating in the shell
  - Program Execution
  - I/O redirection; Pipes;
  - Aliasing;

- Shell 是什么？

一个文字界面，可以执行程序、代码，甚至控制或监测计算机的运行状态。图形界面、声音控制、手势控制等等易用而且更现代的接口使得计算机能够被大众所用。但大多数底层的功能和高效的执行还是需要基于shell。

Shell顾名思义是一层壳，外面是用户，里面是内核或者宽泛地说是操作系统的一部分。

Get your hands dirty!

- **Everything is a file in Linux!**
- **FHS:** Filesystem Hierarchy Standard

  文件系统层次化标准，规范了不少一级和二级目录的用途。希望各用户和软件开发商遵守该标准以管理自己的文件系统层次。

  -> 请看课本图10-1　　Keys: 根目录/　当前目录.　上级目录 ..

  相对路径　绝对路径

- **Features**
  - 在各层次上大小写敏感。
  - 不能靠扩展名区分文件类型，但应当遵循规范添加拓展名以供程序猿和用户直观地了解文件类型。
- **Prompt**

  `user @ localhost: ~ #`

  登录用户 分隔符 系统的简写主机名

  ~ 当前目录(当前是家目录)

  # 超级用户(普通用户 $)

Navigating in the shell

- **`cd`** Change Directory
  - **`cd ~`** 转到当前用户家目录(相当于cd)
  - **`cd -`** 转到上次所在目录
- **`pwd`** Print Working Directory
- **`ls [options] [Directory]`** List
  - **`-l`** Long list, 长格式列出文件信息
  - **`-a`** 显示全部文件(包括隐藏文件)
- **`mkdir`** Make Directory
- **`mv [options] Source_file Target_file`** Move
  - **`-f`** 强制覆盖

  常用于文件重命名

该部分中**`file`**表示一切文件，例如目录也看作是特殊的文件

Navigating in the shell: Lead your Life on FHS

- **`touch`** 修改文件时间戳

  副作用: 文件不存在时创建该文件

- **`cp [Options] Source_file Target_file`** Copy

  `-r` 递归复制

- **`rm [options] File`** Remove

  `-r` 递归删除

  `-f` 强制删除

  `sudo rm -rf /` 删库跑路

Aside:

- **`man`** Manual Page

Get your hands dirty!

- 打包(归档) 和 压缩　Packaging & Compression

  tar　Tape Archive

  -x　解压缩操作

  -c　将多个目录或文件打包

  -f Package_name　指定打包形成的文件名

  -z　压缩/解压缩 .tar.gz 格式

  -C Directory　指定解打包的位置

  -v　verbose, 打印执行过程的详细信息

  tar cvf xxx.tar Source_file1 Source_file2 …

  tar xvf xxx.tar -C directory

  tar zxvf xxx.tar.gz -C directory

  tar zcvf xxx.tar.gz …

  - 处理autolab下载的lab压缩文件时很管用！虽然Writeup里也会写hhh

Get your hands dirty!

- **自动命令行补全**

  补全文件名称　　Tab 1次　　确定性匹配

  2次　　不确定时给出待选列表

  -> 常在 **cd** 等需要列出文件名的命令时使用

  Shell通配符　　* 匹配任意数量字符

  ? 匹配任意单字符

  []匹配括号中任意字符

  -> 常在 **ls** 命令时使用

- **Shell执行命令行的简单模型**
  - **判断路径：** 若是则路径下文件则默认为可执行文件，找不到则报 No such file or directory
  - **检查别名**
  - **内部命令：** 直接执行
  - **外部命令：** PATH 变量下查找有无相应的可执行文件
  - **报错：** command not found

I/O Redirection: Fluid R/W in command line

- Shell中的程序默认有三个流：input stream 0, output stream 1, error stream 2. (Normally they are directed all at the terminal)
  - `< file` 重定向输入 `cat < hello.txt`
  - `> file` 重定向输出 `echo hello > hello.txt`
  - `2> file` 重定向错误 `grep a b 2> log.txt`
  - `&> file` 重定向输出和错误 `grep a b &> log.txt`
  - 追加文件（从已有内容后面开始读/写）`>` 换成 `>>`
  - `|` 管道：前一个程序的输出会成为后一个程序的输入

    ```
    echo . | xargs ls
    ```

- **`xargs`** 把管道或者标准输入转化成后面指令的命令行参数

Aliasing: Customize your own mnemonics

- 仅当前登录期生效
  - `alias [name]=literal`

    `alias ll='ls -l'`
- 长久生效(针对本用户登录)
  - 把alias指令写在shell的配置文件里。E.g. MacOS上

    `~/.zshrc`
  - 再使用source指令(或.)使配置不必重新登录立即生效。以后每次登陆shell，都会利用此文件初始化从而不必再source。

    `source ~/.zshrc` 或

    `. ~/.zshrc`

Next week: Shell #2/2: Advanced Features

- Shell as a scripting language
  - Spacing `foo=bar vs foo = bar`
  - Literals & non literals `'$?' vs "$?"`
  - Special arguments `!! $? $0 $1`
  - Control flow `if for`
  - Regex, wildcards & curly braces `foo* bar?`
  - Return value `echo $?`
- Finding files
- Finding code

A Brief Review On
Bits Bytes Integers

常见术语

- 抽象
- 协议 / 标准 / 规范
- 向前兼容 / 向后兼容

  Forward Compatibility vs. Backward Compatibility

Keys

- **二进制文件 / 文本文件 ？ 信息到底是什么？**

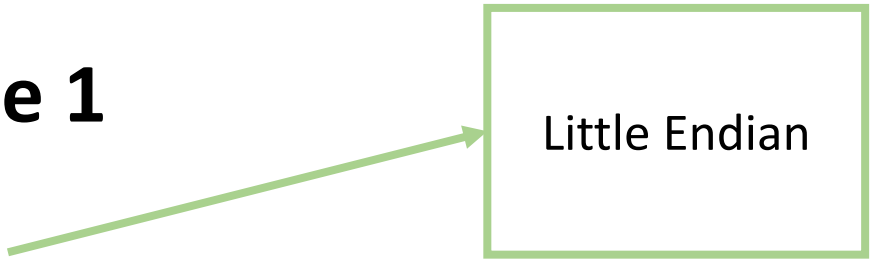
  -> 每个位所在的上下文表明了它的含义，这种含义的具体体现是在解释信息时。

- **信息是怎样排列在内存中的？**

  -> 大端 / 小端 Big Endian vs. Little Endian

- **C语言中的整数**
  - 不同长度 / 符号的整数转换
  - 逻辑运算
  - 算术运算
    - 补码的加减乘运算与无符号数的相应运算具有位级等价性
    - 简单全面地理解 C 整数的算术运算: 构成剩余类环
  - 常见的错误来源: 操作数类型的隐式转换

Exercise 1



- 在 x86-64 机器上，定义

```
unsigned int A = 0x123456;
unsigned short B[2] = {0x1234, 0x5678};
```

- 画出A和B在内存中的存储方式

...	低地址	A				高地址	...
...		0x56	0x34	0x12	0x00	...	

...	低地址	B				高地址	...
...		0x34	0x12	0x78	0x56	...	

Exercise 2

- 在x86-64机器上，有如下代码

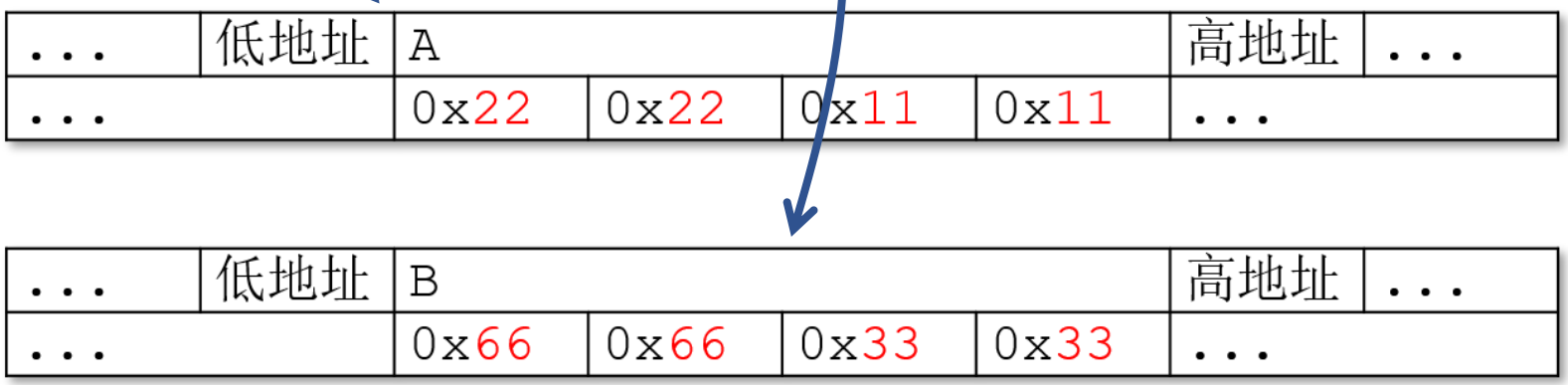
```
int main() {
    unsigned int A = 0x11112222;
    unsigned int B = 0x33336666;
    void *x = (void *)&A;
    void *y = 2 + (void *)&B;
    unsigned short P = *(unsigned short *)x;
    unsigned short Q = *(unsigned short *)y;
    printf("0x%04x", P + Q);
    return 0;
}
```

- 运行结果是

Exercise 2

```
int main() {
    unsigned int A = 0x11112222;
    unsigned int B = 0x33336666;
    void *x = (void *)&A;
    void *y = 2 + (void *)&B;
    unsigned short P = *(unsigned short *)x;
    unsigned short Q = *(unsigned short *)y;
    printf("0x%04x", P + Q);
    return 0;
}
```

0x5555



Exercise 3

- 在x86-64机器上，有如下代码

```
int main() {
    char A[12] = "11224455";
    char B[12] = "11445577";
    void *x = (void *)&A;
    void *y = 2 + (void *)&B;
    unsigned short P = *(unsigned short *)x;
    unsigned short Q = *(unsigned short *)y;
    printf("0x%04x", Q - P);
    return 0;
}
```

- 运行结果是

Exercise 3

```
int main() {
    char A[12] = "11224455";
    char B[12] = "11445577";
    void *x = (void *)&A;
    void *y = 2 + (void *)&B;
    unsigned short P = *(unsigned short *)x;
    unsigned short Q = *(unsigned short *)y;
    printf("0x%04x", Q - P);
    return 0;
}
```

x86: 小端法！
字符串？

0x0303

...	低地址	A				高地址	...
...		0x31	0x31	0x32	0x32	...	

...	低地址	B				高地址	...
...		0x31	0x31	0x34	0x34	...	

Exercise 4

- 在x86-64机器上，定义

```
int x = _______;
int y = _______;
unsigned int ux = x;
unsigned int uy = y;
```

- 判断下列表达式是否等价：

`x > y`	`ux > uy`	N	x = 0, y = -1
`(x > 0) || (x < ux)`	`1`	N	x <= 0
`x ^ y ^ x ^ y ^ x`	`x`	Y	交换律 & 结合律
`((x >> 1) << 1) <= x`	`1`	Y	LHS为x除以2的下取整的两倍
`((x / 2) * 2) <= x`	`1`	N	负奇数
`x ^ y ^ (~x) - y`	`y ^ x ^ (~y) - x`	Y	优先级
`(x == 1) && (ux - 2 < 2)`	`(x == 1) && ((!!ux) - 2) < 2`	N	!!ux 是有符号数

Exercise 5

- 下列代码的目的是将字符串A的内容复制到字符串B，覆盖B原有的内容，并输出”Hello World”；但实际运行输出是”Buggy Codes”。尝试找到代码中的错误。

```
int main() {
    char A[12] = "Hello World";
    char B[12] = "Buggy Codes";
    int pos;
    for (pos = 0; pos - sizeof(B) < 0; pos++)
        B[pos] = A[pos];
    printf("%s\n", B);
}
```

- 答：sizeof的结果是无符号数，因此循环成立条件恒为假。

Supplementary Materials… (Not in Exams)

- **取模运算 / 求余运算　Modulus Operation vs. Remainder Operation**
  - C语言的整数除法向零舍入(求余运算方式)，相应地 % 表示求余运算。求余导致如下性质:
  - `a % b` 中不论 `b` 的符号如何，该表达式的结果为零或者与a同号。
  - 构造程序，通过随机测试检查以上性质。
  - 在python / Java等高级语言中，`%` 的运算语义是取模还是求余，它与 / 舍入的方式适配吗？
- **整型提升 Integral promotion**

https://en.cppreference.com/w/cpp/language/implicit_conversion

有时会在程序中引入奇怪的现象… 来个例子***(可以跳过它以避免世界观崩塌)***

Integral Promotion (Not in Exams)

Standard Output

```
int main() {
    unsigned char a = 0xFF;
    char b = 0xFF;
    printf("%d\n", a == b);
}
```

0

```
int main() {
    unsigned a = 0xFFFFFFFF;
    int b = 0xFFFFFFFF;
    printf("%d\n", a == b);
}
```

1

A few suggestions…

- 助教的职责是解决大家学习过程中的任何困难。欢迎提问！特别基础的问题如果难以理解请不必自卑，本来要求大家在如此有限的时间内掌握这么多内容就是有挑战性的。只是我希望在提问前自己试着检索资料或者有所思考，确实困难或拿不准再提问。
- 欢迎提供小班教学的建议！不限于讨论形式或者讨论内容。例如，别的班提到了很重要的内容但我们没有，我会视情况做调整或者补充。你们的建议很重要（比心@v@）！
- **怎么学ICS？**
  - 首要的任务是阅读课本以及自主实践。课堂内容高度概括，硬核技术和细节都需要自己课下理解掌握！
  - 有些章节初读比较困难，适合反复阅读。往往读到第二、第三遍其义自见。
  - 不同章节的内容差别较大，例如第二章是ICS中唯一有数学性的章节。我会在每节小班课的最后做一个简单的preview，并建议一些方法。但最终请以自己喜欢的方式为准！

Next class...

IEEE 754 浮点标准

- **基础目标：**理解浮点表示与存储的形式，浮点舍入，浮点运算与实数运算的相似与差别，浮点运算律及其成立的原因
- **要求目标：**完成Lab的浮点部分。
- **兴趣拓展(Not in exams)：** 浮点位级表示的精巧应用：Fast inverse square root

  可以试着推导自己的常数。0x5f3759df 并非唯一解，实际的可行解很多！

  Carmack, QuakeIII (雷神之锤3) Lomont. http://www.lomont.org/papers/2003/InvSqrt.pdf

x86汇编

- **基础目标:** AT&T汇编格式规范，识别伪指令。理解数据传送、数据运算、条件码寄存器、测试和分支指令。理解C语言条件判断、循环到汇编代码结构的映射，理解过程的实现和调用惯例。
- **要求目标：**熟读简单的汇编代码。
- **兴趣拓展(Not in exams)：**试着写一些简单的汇编代码，比如枚举。然后可以尝试数组的二分查找或者简单的二叉树搜索。伪指令可以通过将目标函数置空先编译取得，然后直接用汇编语言在.s文件中书写函数体。

Next class...

阅读建议

- 至少完成课本正文部分的每道习题（不包括家庭作业）。
- 目前课程进度相对平缓。有空的话可以自己构思并实现有趣的功能来测试自己是不是真的理解了。比如利用浮点的位级表示来设计一个对浮点数求以2为底数的对数的下取整函数。
- 大量练习巩固，例如汇编代码的阅读能力。这个可以从往年题或者课本的家庭作业里找。不过建议最近两到三年的题目可以跳过，因为如果你想在期中期末前通过往年题巩固的话最好能保持存在两三年的题没有提前看过（实际上我个人认为这是准备ICS考试非常有效的方式）。
- 往年题稍后就会放在北大网盘共享。请注意，往年题的答案仅供参考，存在很多错误。如存疑请联系助教！
- 其他资料都会放在北大网盘中。course上的内容只有提交作业！
  - 同学回课 / lab 的课件（如果有）
  - 往年题 & 错误很多的参考答案
  - 推荐的其他参考教材、论文、题目集锦 etc.
  - 大班往年的课件（不是今年的，但预习时可以使用）

Next class…

Any questions?

Hope to see you in two weeks!
Enjoy the Mid-Autumn Festival & the National Holiday!
Labs are awaiting us! :)