ICS课程报告

本次课程报告有三个可选题目，具体见下文。最终提交的报告不得超过8页A4纸，正文最小字体不得小于五号，建议包含设计、实验、评价标准，并阐述清楚背后的逻辑，以本学期课程内学到的知识为主要论述基础，同时不应有大段代码的复制粘贴（如有代码等内容，可以附件形式提交，但不以附件作为打分依据），严格禁止抄袭。

如有余力也欢迎谈谈对本学期课程感想（单独附页，不在篇幅限制内，特别欢迎各种建议）。

报告提交截止：2023年1月15日

报告提交方式：eLearning平台

报告提交格式：正文以PDF格式提交，附件以ZIP方式压缩提交

选题1：

请自选一个**C/C++程序**，综合运用**这学期所学的知识**，尝试优化这个程序的性能。

建议：

1. 鼓励同学使用自己之前开发或在其他课程中所写的程序，最好选择**算法类程序**以便评估性能提升。我们也会提供一个备选程序（elearning上的exampe1.cpp）。
2. CSAPP第五章中所说的部分优化技术可由编译器自动实现，在gcc/g++命令行中添加`-Ofast -march=native`可以指示编译器在当前系统的指令集架构下最大限度优化程序的性能。
3. 编译器优化无法改变代码逻辑，一般不能优化缓存性能，也不能消除所有的冗余代码，需要同学自行考虑这些问题。

> 比如之前datalab写过的bitReverse，编译器并不总是能将循环版本优化为位运算版本或CPU的专用指令。
>
> 有些函数的输入输出有额外限制，如果编译器不知道限制条件，就没有办法优化。

4. **评分主要关注课内知识的运用**，过多偏离课内知识不会为你带来额外加分。鼓励同学结合所选程序的特点，基于课程内知识提出创新设计。**如果性能不进反退，详细分析失败原因也是一个不错的做法。**

> gcc还支持一个自动进行多线程计算的扩展openmp，可以自行尝试一下（不会加分，但是真的很有用）。

5. 不建议同学使用GPU、分布式计算等课程范围外的技术，也不建议手动以破坏封装或使用goto 等方式优化程序性能。

评分不会考虑性能提升的具体数值。

选题2：

任务目标：分析不同类型cache机制设计(直接映射与组相联、不同替换策略LRU/LFU/…、写策略cache aside/write back/write through等)的优缺点与适用场景，并给出缓存友好的编程建议。

建议：

1.缓存友好的编程可以结合具体样例分析，比如针对一个特定的cache，矩阵乘不同的实现(循环的次序、分块)会如何影响性能

2.报告中可辅以一些图示对比展示

选题3：

任务目标：针对所提供的样例程序（或**自选一个C/C++程序**），请结合所学知识分析该程序在编写、编译、链接以及在命令行里运行时分别经历了什么样的过程。

建议：下面这些点可供参考

1. 编译和链接
2. 命令行执行
3. 运行main函数
4. 调用库函数
5. 程序退出

样例程序：

```
#include <stdio.h>
int main()
{
	char s[16];
	scanf("%15s", s);
	printf("Hello %s\n", s);
	return 0;
}
```