《计算机系统基础实验》任务书

# 实验二 程序优化

# 实验目的与要求

1. 了解程序计时的方法以及运行环境对程序执行情况的影响。
2. 深刻理解CPU执行指令的过程，不同特点的编程技巧和指令序列组合对程序长度及执行效率的影响，掌握代码优化的基本方法。

# 实验内容

（上机实验环境说明：本次实验使用使用Linux环境，需要调用系统函数，例如printf和clock函数。)

## 任务2.1 对任务1.4的程序进行优化

在任务1.4描述的背景下，假设在输入缓冲区中已经存放了N组采集到的状态信息，需要对这N组数据分别计算对应的f并依据分组原则将N组数据复制到对应的LOWF、MIDF、HIGHF存储区中。

**优化工作包括代码长度的优化和执行效率的优化**等等（本次以执行效率/性能的优化为主）。请尝试对f的计算过程以及数据的复制过程的代码进行优化，通过对这些代码执行时间的计时来判断优化的效果。

为体现程序的优化效果，对任务1.4的程序主体，重复执行m次。m的大小可以自行设定，直到总的执行时间和优化时间有明显的效果。

在程序主体执行前使用clock函数计时，计时结果保存在变量start\_t中。在程序主体执行结束后再次调用clock计时，结果保存在end\_t中。调用printf输出(end\_t - start\_t)，即程序主体的执行时间。

分别实现未优化和优化的程序，对比两者的执行时间。要求优化效果提高10%以上。

* 进度提示：

本实验使用一次课内上机的4学时完成。

* 自行设定需要探究学习的问题提示：

1. 度量程序执行效率/性能的常用方法是程序计时。对一段代码的执行时间进行测量的方法有多种，对本实验而言，选取的计时方法需要达到一个要求：应能观察到程序中的指令发生修改时，程序完成同样功能时的执行时间的变化。
2. 比较直观的方法是在待测程序执行前获取一个时间标签，执行之后再获取一个时间标签，然后通过计算这两个时间标签的差值来得到程序的执行时间。
3. 常用的与计时相关的功能有clock()，毫秒级
4. 本次实验是研究性实验，要明确：（1）所采用的计时方法（有精度问题）；（2）主要从哪几个方面优化（由于本课程关注的是**指令级别的影响**，因此，请大家不要只考虑纯算法级别的优化）。
5. 优化实验都需要对优化前后的相关数据进行对比，因此，首先要记录优化前的性能等数据，然后按照优化思路逐步进行优化，每实施了一种优化思路，就需要记录该次优化后相关的性能等数据。最后再对结果进行分析说明。
6. 常见的优化思路：首先是通过选择执行速度较快的指令来提高性能，比如，**把乘除指令转换成移位指令、加法指令**等；其次，循环体中（尤其是多重循环中的内循环体中）每减少一条指令，就相当于减少了“外循环次数\*内循环次数”条指令的执行时间，需要仔细斟酌；第三，灵活利用机器指令的特点、寻址方式的特点等，提高程序的执行效率（比如，利用“LEA指令、基址+变址寻址方式”实现一些表达式的计算）。第四，利用顺序结构代替循环结构，将循环次数较少的循环结构展开，可以充分利用CPU的流水线特性，提高程序的执行效率。
7. 可尝试用C语言模拟实现任务1.4，并和汇编语言的实现版本进行对比。

# 提示

汇编语言中调用C语言库函数的方法，和调用普通函数的方法类似。例如，调用printf函数的过程如下：

...

num: .long 1234

msg: .ascii “the number is %d\n\0”

...

# printf(“the number is %d\n”, num);

pushl num # 将参数num压栈

pushl $msg # 将格式串首址压栈

call printf

...

调用clock函数的过程如下：

...

start\_time: .long 0

...

call clock

mov %eax, start\_time # 将clock函数的返回结果送入start\_time变量中

使用C语言库函数后，在链接时，需要使用动态链接方式，并链接C函数库。链接命令如下：

ld -m elf\_i386 -dynamic-linker /lib/ld-linux.so.2 -o helloworld helloworld.o -lc