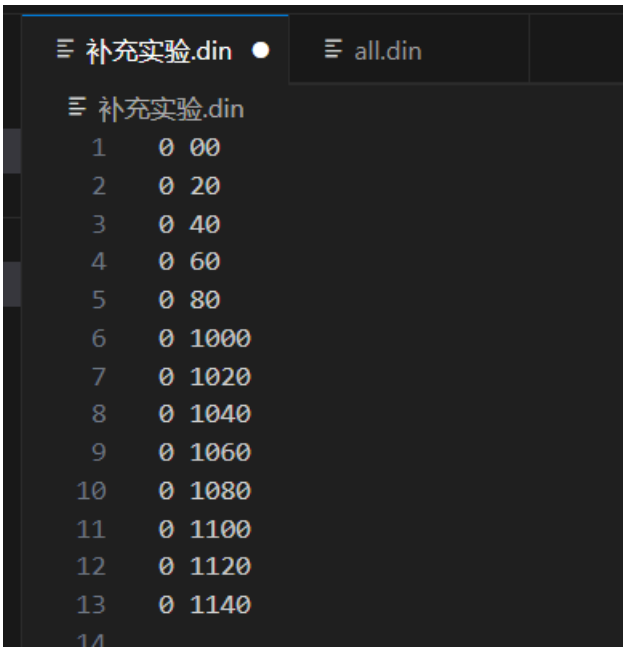
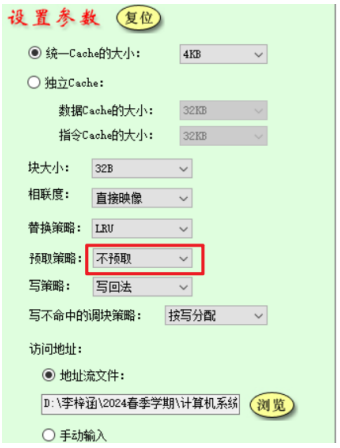
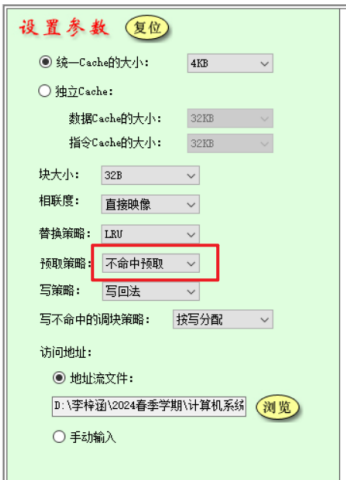
计算机系统结构实验报告

<table>
<tr><td>班级</td><td>计算机 1 班</td><td>实验日期</td><td>2024.4.16</td><td>实验成绩</td><td></td></tr>
<tr><td>姓名</td><td>李梓涵</td><td>学号</td><td colspan="3">34520212201574</td></tr>
<tr><td>实验名称</td><td colspan="5">实验 4 Cache 性能分析</td></tr>
<tr><td>实验目的、要求</td><td colspan="5">1. 加深对 Cache 的基本概念、基本组织结构以及基本工作原理的理解。
2. 掌握 Cache 容量、相联度、块大小对 Cache 性能的影响。
3. 掌握降低 Cache 不命中率的各种方法以及这些方法对提高 Cache 性能的好处。
4. 理解 LRU 与随机法的基本思想以及它们对 Cache 性能的影响。</td></tr>
<tr><td>实验内容、步骤及结果</td><td colspan="5">一、 补充实验
1. Cache 预取的思想：在处理器未发出访问请求，且存储器由空闲带宽的情况下，将指令和数据提前放入 Cache 或缓冲器中
2. 设计实验
1.1 首先确定 Cache 容量大小为 4KB，块大小为 32B，其余默认。这样在编写地址流文件时，采用相隔 32 位地址的取法，在不预取时，不命中率应为 100%，预取时不命中率下降。
1.2 编写地址流文件，统一采用读数据
 
补充实验.din
all.din
补充实验.din
0 00
0 20
0 40
0 60
0 80
0 1000
0 1020
0 1040
0 1060
0 1080
0 1100
0 1120
0 1140
1.3 实验结果</td></tr>
</table>





3. 根据实验结果比较，采用 Cache 预取提升了命中率。但是在使用其他地址流文件时，发现采用 Cache 预取反而会降低命中率，这是因为如果下次访问不落在预取的内容中，程序预取的就是无用的数据，增加了额外开销。因此，Cache 预取是否能提升性能需要结合程序进行分析。

二、 探究性实验

1. 实验目的

使用内外循环交换的方法优化下面的代码

```
for(i=0;i<n;i++)
    for(j=0;j<n;j++)
        for(k=0;l<n;k++)
            C[i][j]+=A[i][k]*B[k][j];
```

2. 设计实验

观察程序，每次循环对于数组 B 的读取都会跨越 n 个区域，这是导致程序运行较慢的一个因素。因此我将对 k 的 for 循环和对 j 的 for 循环交换，优化读取数据跨度大的问题。而对于 AC 两个数组，本来就是按行访问的，因此不需要调整 i 与 j 的相对位置。

在记录程序效率方面，采用记录程序运行时间的方法。使用 Windows

内置的 API 进行记录，使用 QueryPerformanceFrequency 函数获取性能计数器频率，QueryPerformanceCounter 函数进行计时操作。

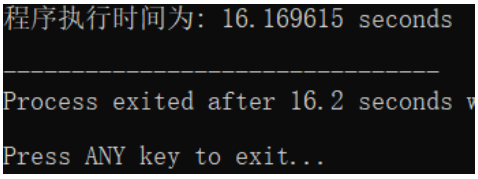
3. 进行实验

优化后的实验代码如下：

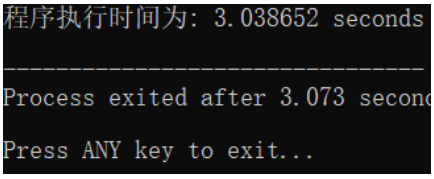
```
#include <iostream>
#include <windows.h>
#define N 1024
int A[N][N], B[N][N], C[N][N];
void init() {
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            A[i][j] = B[i][j] = i + j;
        }
    }
}
int main() {
    init();
    LARGE_INTEGER start, end, freq;
    double elapsed_time;
    QueryPerformanceFrequency(&freq);
    QueryPerformanceCounter(&start);
    for (int i = 0; i < N; i++) {
        for (int k = 0; k < N; k++) {
            for (int j = 0; j < N; j++) {
                C[i][j] += A[i][k] * B[k][j];
            }
        }
    }

    QueryPerformanceCounter(&end);
    elapsed_time = (end.QuadPart - start.QuadPart) / (double)freq.QuadPart;
    printf("程序执行时间为: %f seconds\n", elapsed_time);
    return 0;
}
```

4. 实验结果



优化前执行时间



优化后执行时间

5. 数据分析与结论

矩阵在内存中通常是以行优先的形式进行存储，在原始代码 ijk 的 for 循环中，矩阵 B 是按列访问的，Cache 不命中率较高，因此需要的时间较长。在

	优化代码时，将 j 的 for 循环与 k 的 for 循环互换位置，这样数组 B 也变为按行访问，AC 按行访问不变，Cache 命中率提高，运行时间自然变短。
总结	1. 本次实验我对于改变 Cache 容量、相连度、块大小的影响有了一个直观地理解和认识。同时我也认识到并不是 Cache 容量越大、相连度越高、块大小越大越好，需要在这三者中找到一个平衡值才能得到最好的命中效果。 2. 对于 Cache 预取，在不同的程序中会有不同的效果，如果程序访问的数据空间性不够好，反而会增加额外的开销。 3. 对于探究性实验，我选择使用内外循环交换的方法优化代码效率，主要是让代码执行时访问数据的顺序尽可能贴合数据存储的顺序，这样能增加 Cache 的命中率，提高效率。