**矩阵乘法的优化**

姓名：郁万祥 学号：2013852

1. 问题描述

分别实现四种矩阵乘法程序，并对比运行效率

1. 串行算法
2. Catch优化
3. SSE版本
4. 分片策略
5. 算法设计与实现
6. 串行算法：
7. 算法概述：

A,B为待乘矩阵，C为结果矩阵，遍历A的第i个行向量与B的第i个列向量，乘积之和作为C的第i个分量，依次遍历，得到结果矩阵C。

1. 时间复杂度分析：

对矩阵A的每一行的元素，矩阵B的每一列的元素都要与之相乘并相加，所以需要先遍历矩阵A一行中的元素的同时遍历矩阵B一列中的元素，再遍历矩阵B中的每一列，再遍历矩阵A中的每一行，三层遍历嵌套，时间复杂度O(n) = n^3。

1. Catch优化：
2. 算法概述：

考虑到C++中的数组在内存中的物理地址是连续的，而按照穿行算法，每次都会进行跳跃寻址，又因为，在程序运行的过程中，会先访问速度更快的cache，因此，我们通过先将b转置，从而实现增大程序访存过程中cache的命中率，从而提高程序运行的速度。

1. 时间复杂度分析：

因为此算法只是通过提高了cache的命中率优化了运算过程中访存的时间，而对于程序本省而言，依然通过了三层for循环，因此时间复杂度依旧是O(n) = n^3。

1. SSE版本：
2. 算法概述：

基于SIMD编程，将多个元素同时进行寄存器加载和存储器存值等操作能够提高并行效率。使用SSE指令能够实现该功能。同样对矩阵B进行转置。

1. 时间复杂度分析：

时间复杂度为O(n) = (n^3)/4。

1. 分片策略：
2. 算法概述：

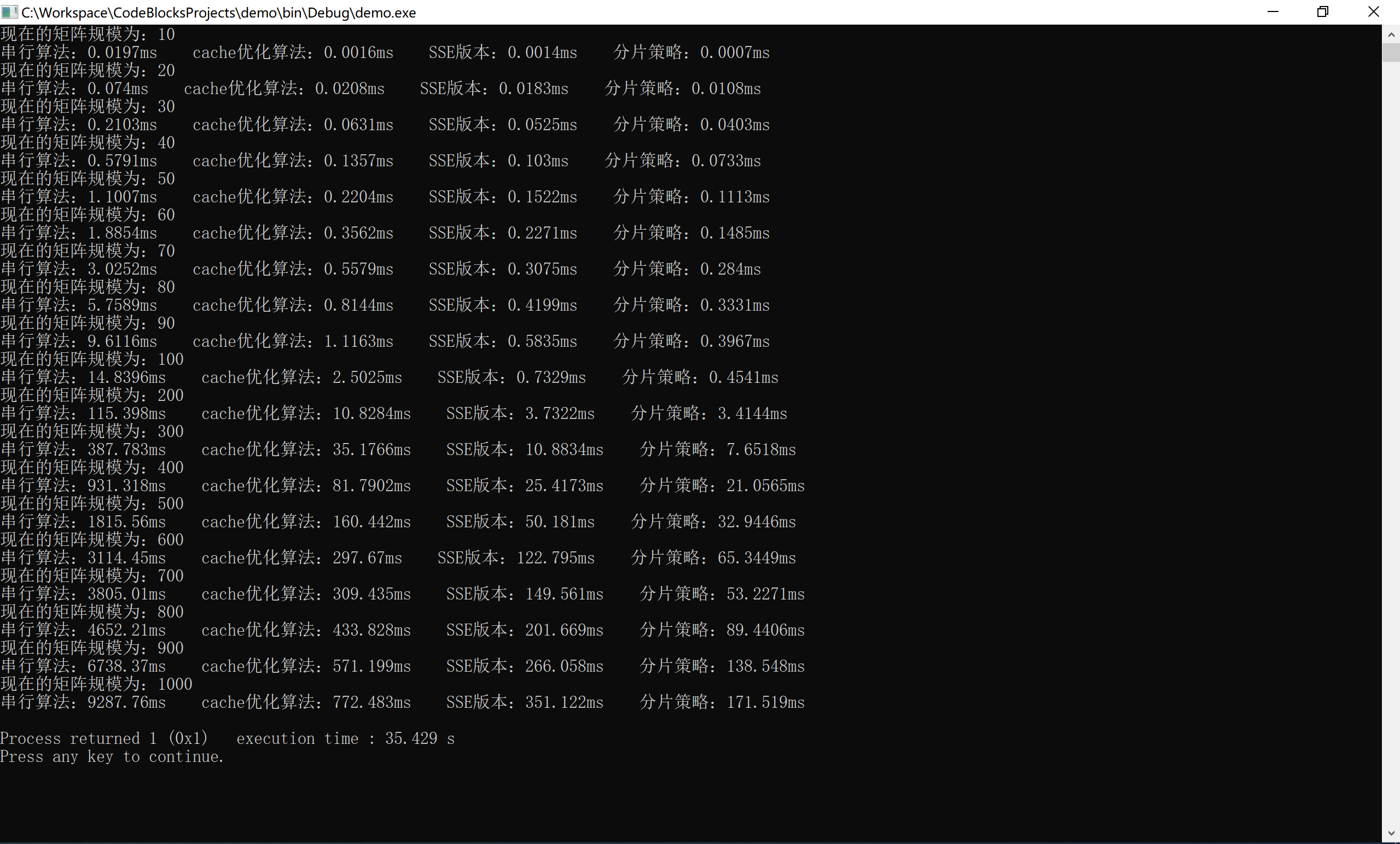
先针对cpu线程数设置分片数，将矩阵分为多个小矩阵，对每个小矩阵进行SSE指令计算，同样对矩阵B进行转置。

1. 时间复杂度分析：

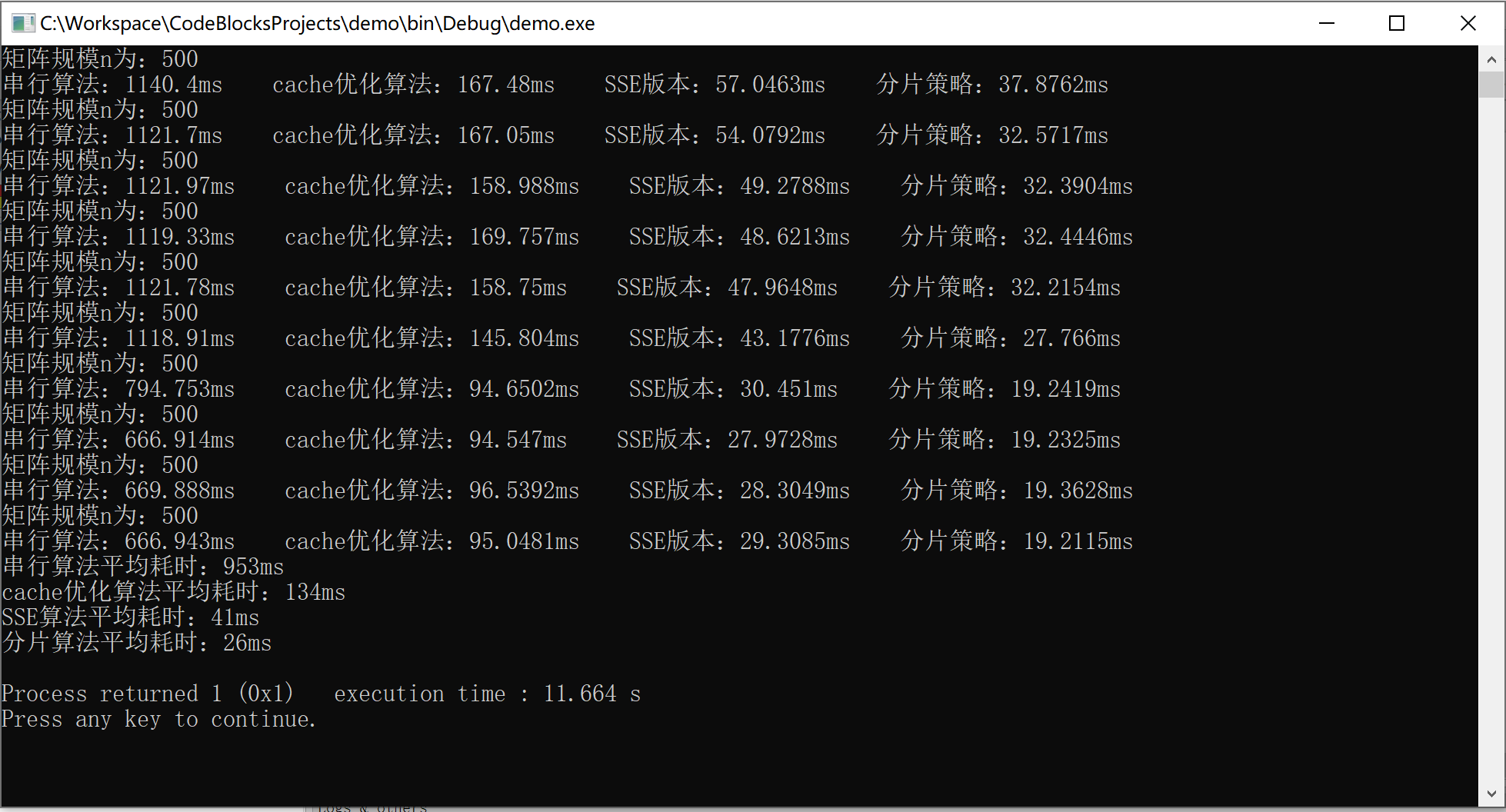
时间复杂度为O(n) = n^3

1. 实验及结果分析
2. 测试数据：
3. 、对于矩阵规模n，从10到1000采取递增策略，100以内每次递增10，100~1000每次递增100。
4. 、同时设定固定矩阵规模500，采取10次取平均值的方法，观察每种算法的运行时间。
5. 测试结果：

1）、



2）、



1. 结果分析：

1）、

可以看到，随着规模逐渐增大，cache算法，SSE算法和分片策略相对于串行算法的优化效果越来越明显。

2）、

可以看到，四种算法优化程度比较：分片策略>SSE版本>cache优化算法>串行算法。

**高斯消元法SSE并行化**

1. 算法分析设计：

算法伪码：

1. **procedure** LU (*A*)

2. **begin**

3. **for** *k* := 1 **to** *n* **do**

4. **for** *j* := *k*+1**to** *n* **do**

5. *A*[*k, j*] := *A*[*k, j*]*/A*[*k, k*];

6 *A*[*k, k*] := 1.0;

7. **endfor**;

8. **for** *i* := *k* + 1 **to** *n* **do**

9. **for** *j* := *k* + 1 **to** *n* **do**

10. *A*[*i, j*] := *A*[*i, j*] - *A*[*i, k*] × *A*[*k, j* ];

11. **endfor**;

12. *A*[*i, k*] := 0;

13. **endfor**;

14. **endfor**;

15. **end** LU

通过分析高斯的程序可以发现，高斯消去法有两部分可以实现并行，分别是第一部分的除法和第二部分的减法。即:

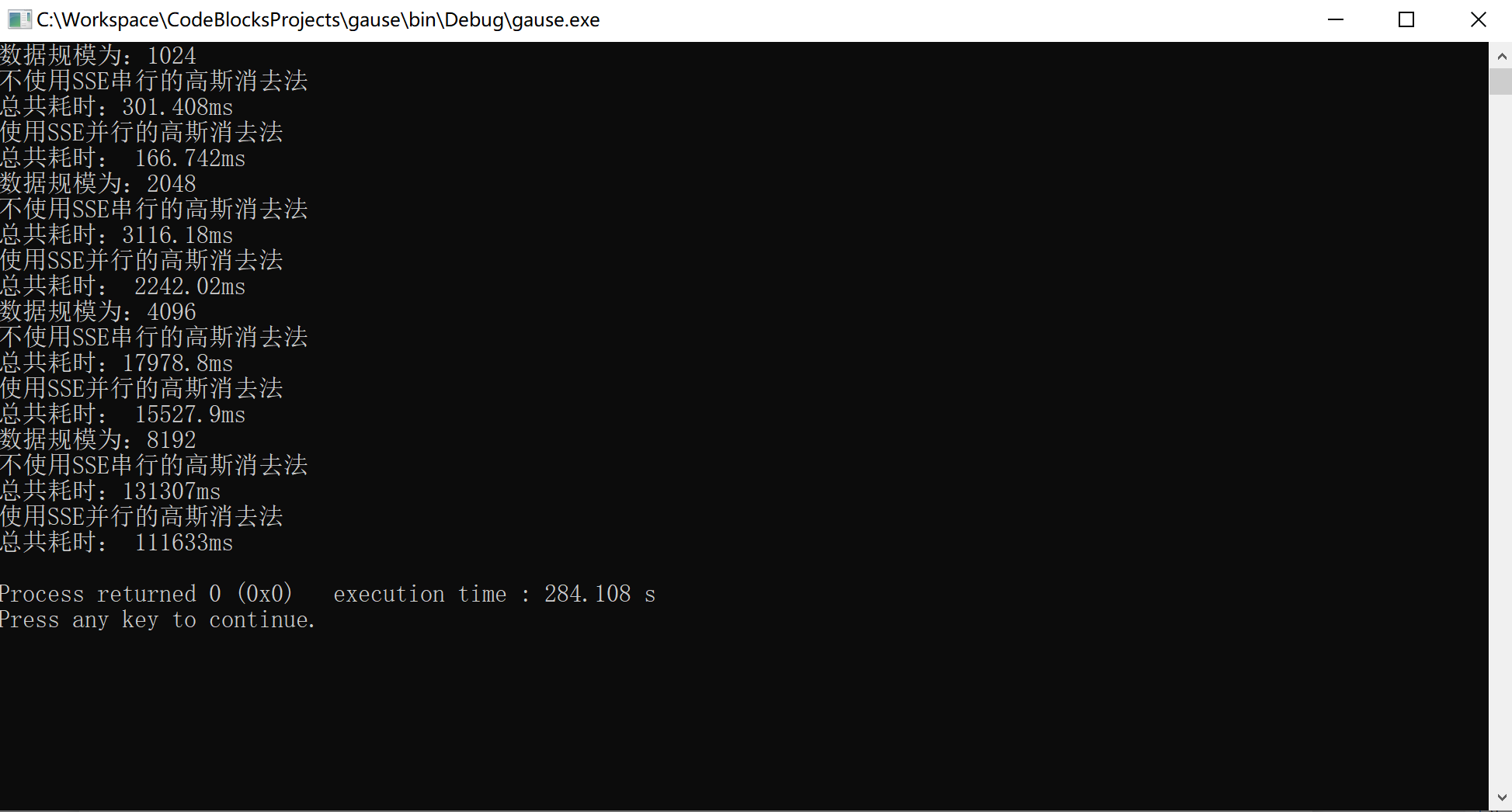
1.第一个内嵌的for循环里的A[k, j]:= A[k, j]/A[k, k]; 我们可以做除法并行。

2.第二个双层for循环里的A[i, j] := A[i, j] - A[i, k] × A[k, j ];我们可以做减法并行。

1. 实验结果：
2. 数据选取：

为了尽可能地贴合并行算法的实际使用情况，此次实验没有选取规模较小的数据进行测试，而是从比较大的数据开始测试，因此，此次实验所选区的数据规模从小到大依次取：1024，2048，4096，8192。

1. 实验运行结果：



1. 讨论：
2. 相同算法对于不同问题规模的性能提升是否有影响，影响情况如何：

我们根据实验结果将串行消元和SSE消元作比较：

我们可以看到，相同算法对于不同问题规模的性能提升有一定的影响，但是影响情况不大。

1. 消元过程中采用向量编程的性能提升情况如何：

消元过程中，如果数据比较多，比较普遍，经过计算，向量编程的性能提升大约在14%左右，因此在消元过程中,向量编程的性能提升情况一般。

1. 回代过程可否向量化，有的话性能提升情况如何；

回代过程可以向量化，但是对性能提升效果不佳。

1. 数据对齐与不对齐对计算性能有怎样的影响

对于串行算法，不需要数据对齐，因此数据对齐与否对时间复杂度与性能影响不大，对于SSE并行算法，因为在并行计算前，数据对齐是固定的步骤，如果数据本身对对齐性较差，则需要额外的时间开销进行数据对齐，因此对性能影响比较大。

正是可能因为SSE有一些固定的步骤，才导致了SSE对于高斯消元优化的效果并不是特别明显。