Cache Lab Report

姓名：骆炳君

班级：软件71

学号：2017013573

Part A

initCache()

为缓存池开辟内存空间并初始化为0.

showHelp()

显示帮助界面.

searchCache(unsigned long long addr)

在缓存中搜索指定地址addr处的数据，如果未命中则加载它.

sim()

逐行读取.trace文件并模拟缓存的读写过程.

main(int argc, char *argv[])

程序入口.

Part B

因为s=5, E=1, b=5, 整个缓存有$2^5 = 32$个块且每个块存有$2^5 = 32$字节，也就是8个int型变量的数据。

32*32

32*32矩阵的一行有4个缓存块，相邻的8行刚好能占据整个缓存空间，所以把矩阵分成8*8的子块进行转置。但是这样依然不能达到要求，主要是因为A、B对角线上对应位置的块映射到了缓存上的同一区域，在读A、写B的过程中会出现交替的未命中情况，从而增加miss数量。针对对角线上的元素进行特殊处理，在每一行结束时才更新对角线元素，可以明显减少miss数量，达到题目要求。

缓存未命中的原因是第一次访问某一块数据时的加载和对角线子块交替读写时的未命中情况。

64*64

32*32矩阵相邻的4行就能占据整个缓存空间，但为了充分利用缓存空间，还是将矩阵分为8*8的子块。为了减少频繁的未命中读写，在访问一个子块时，要尽可能地减少上半块与下半块的交替访问。在对A和B的对应子块A‘、B'进行访问时，首先逐行读取A'的上半部分，转置后存储在B'的上半部分中（左上部分存储正常的转置，右上部分存储本应该在左下的数据），然后按列从两侧向中间读取A'的下半部分，将B'右上部分的数据转移到左下部分的正确位置，并将A'的左下部分的转置写入到B'的右上部分，再对右下部分进行转置。通过对B'右上部分临时数据进行优化，将其在竖直方向反向，可以使得在访问A'的下半部分时，避免交替访问B'的上、下部分映射到同一缓存块的数据行，从而明显减少miss的数量，达到题目要求。

缓存未命中的主要原因是对角线子块的交替访问和访问B'下半部分时上、下半部分的交替切换。

61*67

由于该矩阵不是方阵，针对A优化的算法可能并不适用于B，反之亦然. 所以使用与32*32类似的分块方法进行优化，并对分块的尺寸进行测试，发现在分块大小为17时的miss数量最小且能达到题目要求。

和32*32的情况类似，除了第一次访问数据块时的加载，缓存未命中的主要原因是A、B对角线对应位置的子块映射到缓存的同一区域，交替读写导致大量miss。

48*48

48*48的矩阵介于32*32和64*64之间，一行6个缓存块，相邻不到6行就会占据整个缓存空间，但为了充分利用缓存空间，还是选择8*8的分块方法。对A和B的对应子块A‘、B'，逐行读取A'中的数据，并直接（不加转置）复制到B‘中，然后再对B'中关于对角线对称的元素进行交换。

除了数据块第一次加载和对角线子块的交替读写以外，缓存未命中的主要原因是一个8*8子块并不能被同时加载在缓存中所导致的miss。