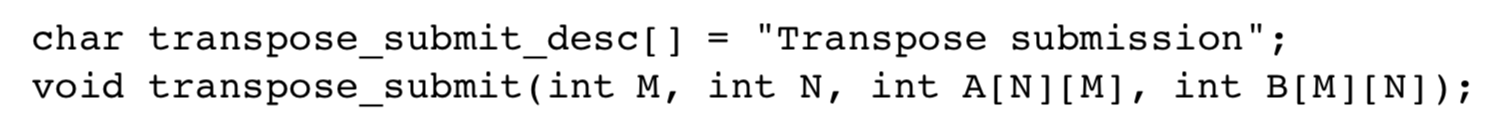
6.5日实验课 查看partb要求

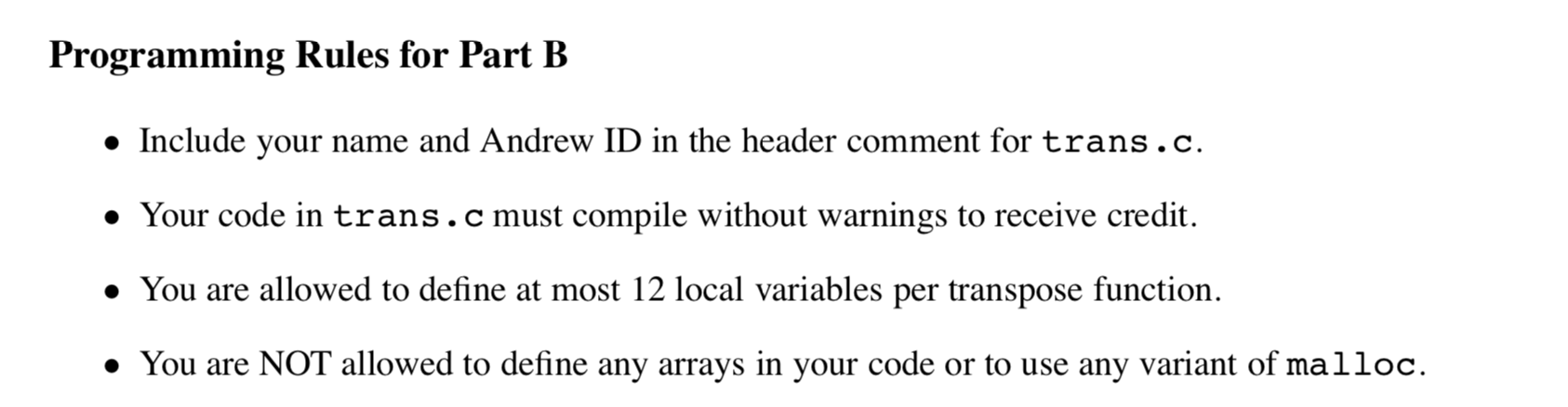
要求：write a transpose function in trans.c that causes as few cache misses as possible

The example transpose function is correct, but it is inefficient because the access pattern results in relatively many **cache misses.**

我的任务：Your job in Part B is to write a similar function, called transpose\_submit, that minimizes the number of cache misses across different sized matrices:



rule：



规格：

* 32×32(M =32,N =32)
* 64×64(M =64,N =64)
* 61×67(M =61,N =67)

首先尝试的代码是：

char transpose\_submit\_desc[] = "Transpose submission";

void transpose\_submit(int M, int N, int A[N][M], int B[M][N])

{

for (int i = 0; i < N; i += 8)

{

for (int j = 0; j < M; j += 8)

{

for (int k = i; k < i + 8; ++k)

{

for(int l = j; l < j + 8; l++)

{

int aa = A[k][l];

B[l][k] = aa;

}

}

}

}

}

然而……

miss是343次

6.8日实验课分析：

题目给的cache大小：块大小为5字节，5个组，1路组相连

那么，在32\*32矩阵里，要求miss次数在300以下cache一行正好可以放下8个数据。所以把矩阵分成8\*8。

对于在对角线上的块，A中每读一行，因为第一个数据就找不到，所以会有一次miss，然后就会把同一行的都放到cache里面。

miss次数=1/8 \*读取的操作次数。

对于B数组的话，第一次读取这行会产生一次miss，之后对于第i行，只有A中读到第i行的时候，会被移除出Cache，然后存的时候会产生一次miss。粗略计算为miss次数是读取次数的1/4。

对于不在对角线上的块，做转置的时候，A还是1/8的miss率，B的每行在Cache中和A的行不冲突 ，所以也是1/8的miss率

miss次数：4（对角线）\*64\*（1/4 + 1/8）+12（非对角线）\* 64 \* 1/4=288

char transpose\_submit\_desc[] = "Transpose submission";

void transpose\_submit(int M, int N, int A[N][M], int B[M][N])

{

int i, j, aa, bb, cc, dd, ee, ff, gg, hh;

int n = N / 8 \* 8;

int m = M / 8 \* 8;

for (j = 0; j < m; j += 8)

{

for (i = 0; i < n; i++)

{

aa = A[i][j];

bb = A[i][j+1];

cc = A[i][j+2];

dd = A[i][j+3];

ee = A[i][j+4];

ff = A[i][j+5];

gg = A[i][j+6];

hh = A[i][j+7];

B[j][i] = aa;

B[j+1][i] = bb;

B[j+2][i] = cc;

B[j+3][i] = dd;

B[j+4][i] = ee;

B[j+5][i] = ff;

B[j+6][i] = gg;

B[j+7][i] = hh;

}

}

for(i = n;i < N;i++)//处理小尾巴～

for(j = m;j < M;j++)

{

aa = A[i][j];

B[j][i] = aa;

}

for(i = 0;i < N;i++)

for(j = m;j < M;j++)

{

aa = A[i][j];

B[j][i] = aa;

}

for(i = n;i < N;i++)

for(j = 0;j < M;j++)

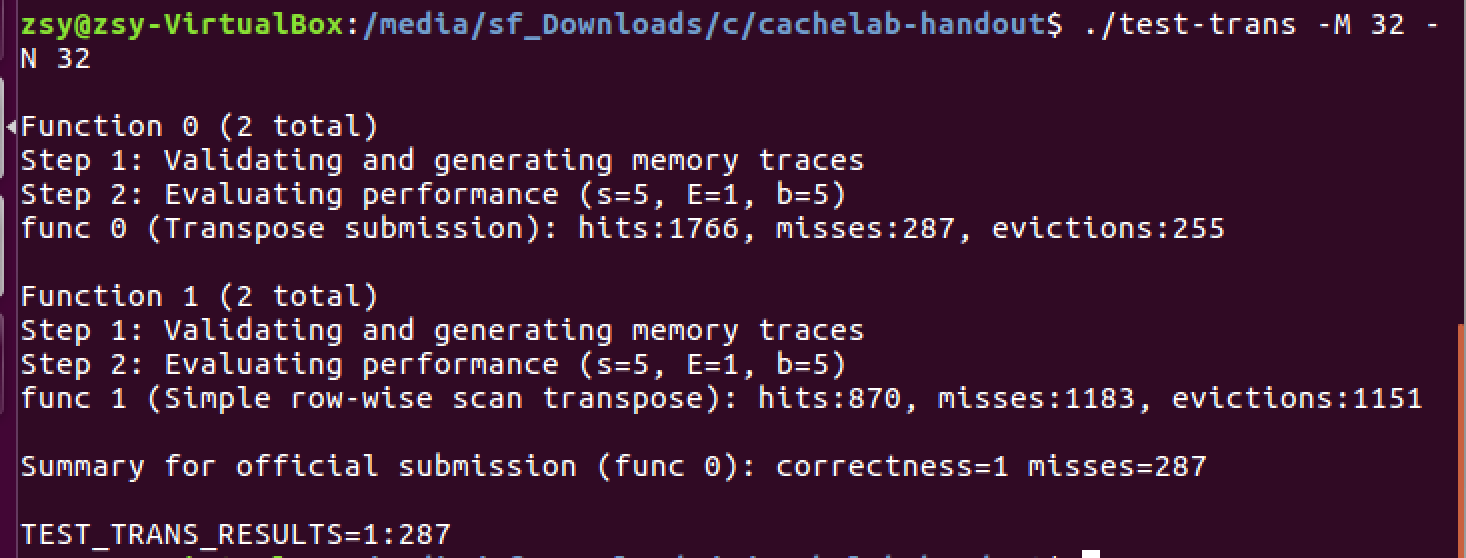
{

aa = A[i][j];

B[j][i] = aa;

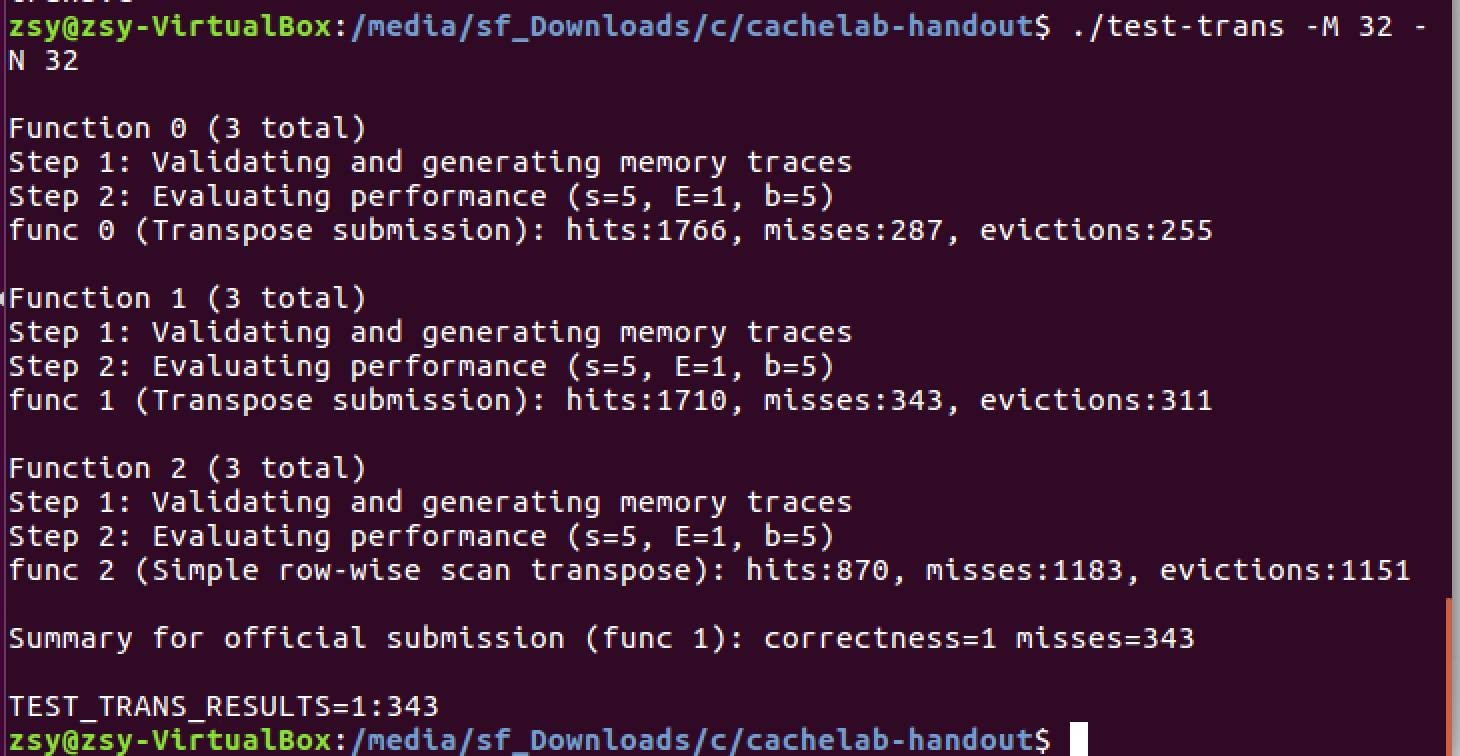
}

}



成功啦

测试时是287，和我计算的288很接近。



三种对比