#include<iostream>

using namespace std;

#include<pthread.h>

#include <sys/time.h>

#include<time.h>

#include <immintrin.h>//AVX

#include <xmmintrin.h>

#include<semaphore.h>

typedef struct{

int t\_id;

float\*\* A;

int n;

}threadParam\_t;

const int NUM\_THREADS=7;

//信号量

sem\_t sem\_leader;

sem\_t sem\_Division[NUM\_THREADS-1];

sem\_t sem\_Elimination[NUM\_THREADS-1];

//线程函数定义

void \*threadFunc(void \*param){

threadParam\_t \*p = (threadParam\_t\*)param;

int t\_id = p->t\_id;

float\*\* A=p->A;

int n=p->n;

for(int k=0;k<n;k++){

if(t\_id == 0){

//先由0号线程统一完成除法工作

for(int j=k+1;j<n;j++){

A[k][j]=A[k][j]/A[k][k];

}

A[k][k]=1.0;

}

else{

sem\_wait(&sem\_Division[t\_id-1]);

}

//t\_id=0的线程唤醒其他工作线程，进行消去

if(t\_id==0){

for(int i=0;i<NUM\_THREADS-1;i++){

sem\_post(&sem\_Division[i]);

}

}

//划分循环任务 消去工作

for(int i=k+1+t\_id;i<n;i=i+NUM\_THREADS){

\_\_m256 t1, t2, t3, t4;

float tmp[8] = {A[i][k], A[i][k], A[i][k], A[i][k],A[i][k], A[i][k], A[i][k], A[i][k]};

t1 = \_mm256\_loadu\_ps(tmp);

int j;

for(j=n-8;j>=k;j=j-8){

//A[i][j] = A[i][j]-A[i][k]\*A[k][j];

t2 = \_mm256\_loadu\_ps(A[i] + j);

t3 = \_mm256\_loadu\_ps(A[k] + j);

t4 = \_mm256\_sub\_ps(t2, \_mm256\_mul\_ps(t1, t3));

\_mm256\_storeu\_ps(A[i]+j, t4);

}

if((j+8)!=k)

{

for(int s=k; s<(j+8);s++)

A[i][s] = A[i][s]-A[i][k]\*A[k][s];

}

/\*for(int j=k+1;j<n;j++){

A[i][j]=A[i][j]-A[i][j]\*A[k][j];

}

A[i][k]=0.0;\*/

}

//所有线程一起进入下一轮

if(t\_id==0){

for(int i=0;i<NUM\_THREADS-1;i++)

sem\_wait(&sem\_leader);

for(int i=0;i<NUM\_THREADS-1;i++)

sem\_post(&sem\_Elimination[i]);

}

else{

sem\_post(&sem\_leader);

sem\_wait(&sem\_Elimination[t\_id-1]);

}

}

pthread\_exit(NULL);

}

void display(float \*\*A, int n){

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

for(int j=0; j<n; j++)

{cout<<A[i][j]<<' ';}

cout<<endl;}

}

int main(){

//初始化矩阵

int n=4;

for(n=4;n<=1024;n=n\*2){

cout<<n<<' ';

struct timeval start;

struct timeval end;

gettimeofday(&start,NULL);

for(int cy=0;cy<10;cy++){

float\*\* A = new float\* [n];

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

A[i]= new float[n];

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

{ for(int j=0;j<i;j++)

A[i][j]=0;

A[i][i]=1.0;

for(int j=i+1;j<n;j++)

A[i][j]=rand();

}

for(int k=0;k<n;k++)

for(int i=k+1;i<n;i++)

for(int j=0;j<n;j++)

A[i][j]+=A[k][j];

//初始化信号量

sem\_init(&sem\_leader,0,0);

for(int i=0;i<NUM\_THREADS-1;i++){

sem\_init(&sem\_Division[i],0,0);

sem\_init(&sem\_Elimination[i],0,0);

}

//创建静态线程

pthread\_t handles[NUM\_THREADS];//创建对应的handle参数

threadParam\_t param[NUM\_THREADS]; //创建对应的线程数

for(int t\_id=0;t\_id<NUM\_THREADS;t\_id++){

param[t\_id].t\_id=t\_id;

param[t\_id].A=A;

param[t\_id].n=n;

pthread\_create(&handles[t\_id],NULL,threadFunc,(void \*)&param[t\_id]);

}

for(int t\_id=0;t\_id<NUM\_THREADS;t\_id++){

pthread\_join(handles[t\_id], NULL);

}

sem\_destroy(&sem\_leader);

for(int i=0;i<NUM\_THREADS-1;i++){

sem\_destroy(&sem\_Division[i]);

sem\_destroy(&sem\_Elimination[i]);

}

}

gettimeofday(&end,NULL);

cout<<((long long)end.tv\_sec-(long long)start.tv\_sec)\*1000000+((long long)end.tv\_usec-(long long)start.tv\_usec)<<endl;//微秒

}

return 0;}