Homework 3

Threads

Deadline: February 24 2016
What is a barrier?

• Barriers are a synchronization primitive, which trigger once a certain number of participants reached the synchronization point
• The POSIX standard propose a simple interface

```c
int pthread_barrier_init ( pthread_barrier_t *barrier,
                          const pthread_barrierattr_t *attr,
                          unsigned count);

int pthread_barrier_destroy ( pthread_barrier_t *barrier);

int pthread_barrier_wait ( pthread_barrier_t *barrier);
```
What is a barrier?

- We need
  - A counter: to count how many threads are waiting
  - A condition to make them wait until we release the barrier
A possible implementation

```c
int barrier_wait ( barrier_t *barrier) {
    pthread_mutex_lock( &(barrier->mutex));
    barrier->curcount++;
    if( barrier->curcount == barrier->count) {
        barrier->curcount = 0;
        pthread_cond_broadcast( &(barrier->cond));
        pthread_mutex_unlock( &(barrier->mutex));
        return 1;
    }
    pthread_cond_wait( &(barrier->cond),
                      &(barrier->mutex));
    pthread_mutex_unlock( &(barrier->mutex));
    return 0;
}
```
Barrier analysis

• We need to develop a correct implementation.
  – Is the previous implementation correct?
  – If no provide a correct implementation.
  – Or simply a faster version
Threaded DGEMM

• DGEMM stands for double precision general matrix-matrix multiplication
  – A 3 affine loops algorithm available in any math or CS book

• The goal is to provide a version of this operations that works efficiently in a multi-core environment.

• Analyze the behavior of your function when the number of threads varies.
  – Compare with a well-known implementation such as MKL (Linux) or the Accelerate framework (OS X)