```
#include "mthread_internal.h"
#include <sched.h>

#if defined(i686_ARCH) || defined(x86_64_ARCH)

static inline int __mthread_test_and_set(mthread_tst_t *atomic)
{
    int ret;

    __asm__ __volatile__("lock; xchgl %0, %1":"=r"(ret), "=m"(*atomic)
                         :"0"(1), "m"(*atomic)
                         :"memory");
    return ret;
}

#elif defined(sparc_ARCH)
static inline int __mthread_test_and_set(mthread_tst_t *spinlock)
{
  char ret = 0;

  __asm__ __volatile__("ldstub [%0], %1"
        : "=r"(spinlock), "=r"(ret)
        : "0"(spinlock), "1" (ret) : "memory");

  return (unsigned)ret;
}

#elif defined(ia64_ARCH)
static __inline__ int __mthread_test_and_set(mthread_tst_t *atomic)
{
    int ret;

    __asm__ __volatile__("xchg4 %0=%1, %2":"=r"(ret), "=m"(*atomic)
                         :"0"(1), "m"(*atomic)
                         :"memory");
    return ret;
}
#else
#define USE_GENERIC_ASM
#warning "Using generic test and set using pthread"
#include <pthread.h>
static pthread_mutex_t tst_mutex = PTHREAD_MUTEX_INITIALIZER;

static inline int __mthread_test_and_set(mthread_tst_t *atomic)
{
  int res;
  pthread_mutex_lock(&tst_mutex);
  res = *atomic;
  if(*atomic == 0){
    *atomic = 1;
  }
  pthread_mutex_unlock(&tst_mutex);
  return res;
}
```

```
#endif

int mthread_test_and_set(mthread_tst_t *atomic){
  return __mthread_test_and_set(atomic);
}

void mthread_spinlock_lock(mthread_tst_t *atomic){
#ifdef USE_GENERIC_ASM
  static pthread_mutex_t spin_tst_mutex = PTHREAD_MUTEX_INITIALIZER;
  pthread_mutex_lock(&spin_tst_mutex);
#endif
  while(mthread_test_and_set(atomic)){
    sched_yield();
  }
#ifdef USE_GENERIC_ASM
  pthread_mutex_unlock(&spin_tst_mutex);
#endif
}

void mthread_spinlock_unlock(mthread_tst_t *atomic){
  *atomic = 0;
}
```