

x86 的 function call 當中

前六個 integer 或 pointer 的參數(arguments)會分別放在 rdi, rsi, rdx, rcx ,r8, r9  
因為 mutex 是 integer 且是第一個參數，所以 mutex 的記憶體位址會放在 rdi  
加括號變成(%rdi)可取得記憶體中 mutex 的值

spin\_lock:

movl \$1, %ecx	←將 ecx 設為 1(movl 是 32 位元)
loop:	
movl (%rdi), %eax	←將 eax 設為 mutex 的值
test %eax, %eax	←判斷 mutex 是否為 1
jnz loop	←mutex 為 1 則迴圈
lock cmpxchgl %ecx, (%rdi)	←mutex 為 0 則嘗試進入 critical section
jnz loop	←進入 critical section 失敗則迴圈
ret	←進入 critical section 成功則 return

spin\_unlock:

movl \$0, (%rdi)	←將 mutex 設為 0(release)
ret	←return

lock 確保了不會有兩個 threads 同時寫(%rdi)，又因為 comxchgl 是 atomic 指令  
所以不會有兩個 thread 同時執行 lock comxchgl %ecx, (%rdi)  
達成了 mutual exclusion

不直接嘗試進入 critical section

而是先用 test 指令判斷 mutex 是否為 1，mutex 為 0 才嘗試進入 critical section  
經過測試，可以跑得更快，見下頁附圖

## 附圖：執行時間測試

The image displays three terminal windows and their corresponding assembly code editors side-by-side, illustrating the execution times and assembly implementations of different spinlock algorithms.

**Terminal 1 (Top):**

```
os_project@os-VM:~/os_project/spinlock$ make
gcc -o spinlock main.c spinlock.s -lpthread
os_project@os-VM:~/os_project/spinlock$ time ./spinlock 10000000
Correct! cnt=20000000

real    0m1.393s
user    0m2.756s
sys     0m0.008s
os_project@os-VM:~/os_project/spinlock$ time ./spinlock 10000000
Correct! cnt=20000000

real    0m1.336s
user    0m2.647s
sys     0m0.004s
os_project@os-VM:~/os_project/spinlock$ time ./spinlock 10000000
Correct! cnt=20000000

real    0m1.291s
user    0m2.566s
sys     0m0.008s
os_project@os-VM:~/os_project/spinlock$
```

**Assembly Editor 1:**

```
.text
.global spin_lock
.global spin_unlock
spin_lock:
    movl $0, %eax
    movl $1, %ecx
    lock cmpxchgl %ecx, (%rdi)
    jnz spin_lock
    ret
spin_unlock:
    movl $0, (%rdi)
    ret
```

**Terminal 2 (Middle):**

```
os_project@os-VM:~/os_project/spinlock$ make
gcc -o spinlock main.c spinlock.s -lpthread
os_project@os-VM:~/os_project/spinlock$ time ./spinlock 10000000
Correct! cnt=20000000

real    0m1.332s
user    0m2.662s
sys     0m0.000s
os_project@os-VM:~/os_project/spinlock$ time ./spinlock 10000000
Correct! cnt=20000000

real    0m1.248s
user    0m2.463s
sys     0m0.000s
os_project@os-VM:~/os_project/spinlock$ time ./spinlock 10000000
Correct! cnt=20000000

real    0m1.318s
user    0m2.615s
sys     0m0.004s
os_project@os-VM:~/os_project/spinlock$
```

**Assembly Editor 2:**

```
.text
.global spin_lock
.global spin_unlock
spin_lock:
    xorl %ecx, %ecx
    incl %ecx
loop:
    xorl %eax, %eax
    lock cmpxchgl %ecx, (%rdi)
    jne loop
    ret
spin_unlock:
    movl $0, (%rdi)
    ret
```

**Terminal 3 (Bottom):**

```
os_project@os-VM:~/os_project/spinlock$ make
gcc -o spinlock main.c spinlock.s -lpthread
os_project@os-VM:~/os_project/spinlock$ time ./spinlock 10000000
Correct! cnt=20000000

real    0m0.728s
user    0m1.423s
sys     0m0.000s
os_project@os-VM:~/os_project/spinlock$ time ./spinlock 10000000
Correct! cnt=20000000

real    0m0.887s
user    0m1.769s
sys     0m0.004s
os_project@os-VM:~/os_project/spinlock$ time ./spinlock 10000000
Correct! cnt=20000000

real    0m0.620s
user    0m1.189s
sys     0m0.000s
os_project@os-VM:~/os_project/spinlock$
```

**Assembly Editor 3:**

```
.text
.global spin_lock
.global spin_unlock
spin_lock:
    movl $1, %ecx
loop:
    movl (%rdi), %eax
    test %eax, %eax
    jnz loop
    lock cmpxchgl %ecx, (%rdi)
    jnz loop
    ret
spin_unlock:
    movl $0, (%rdi)
    ret
```