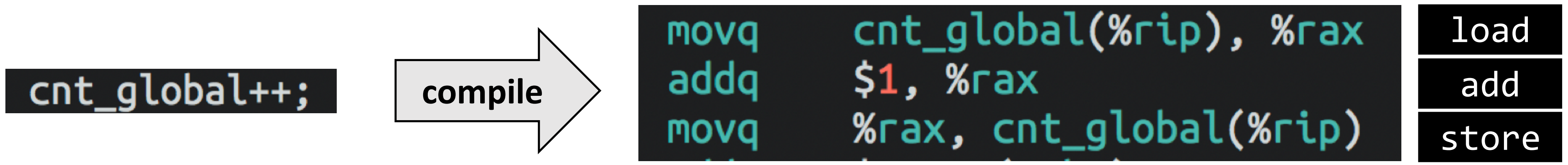
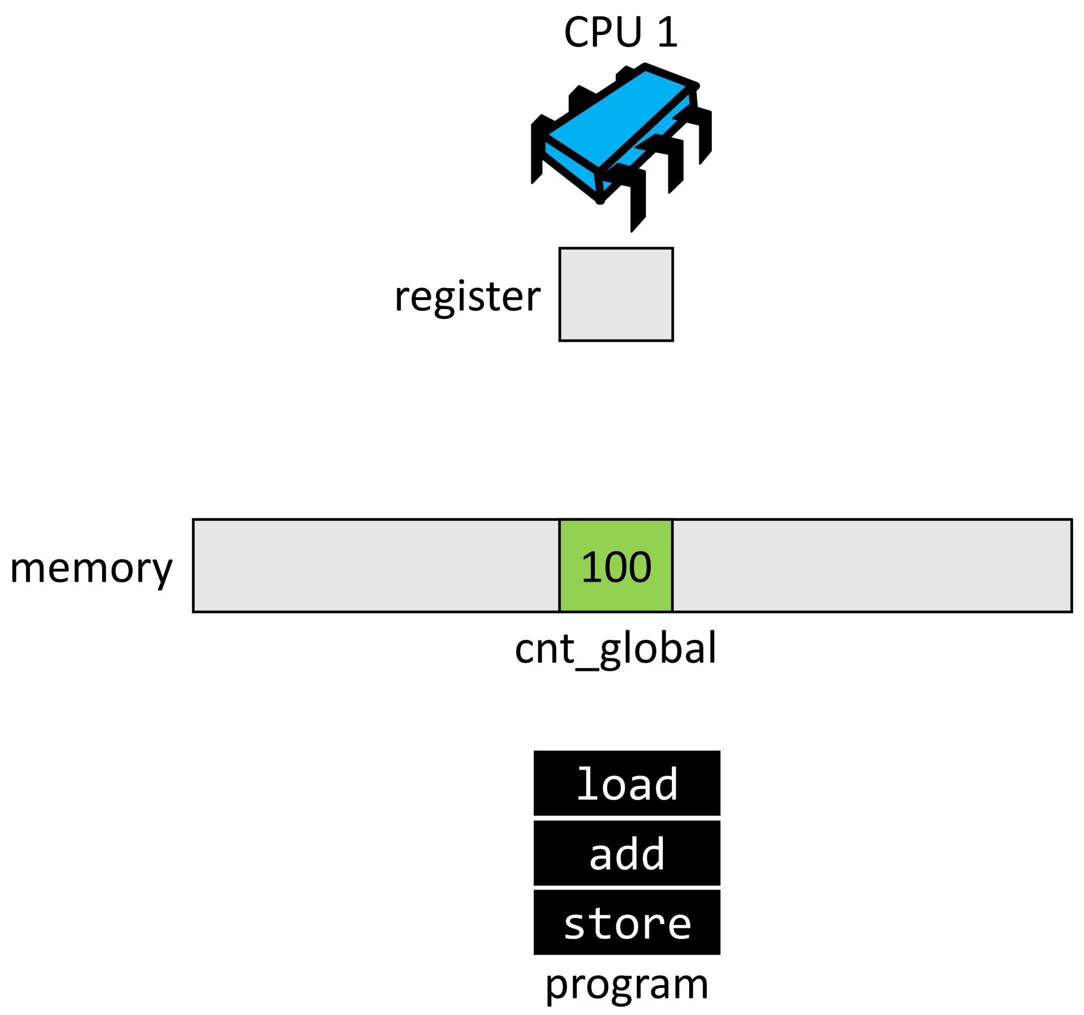


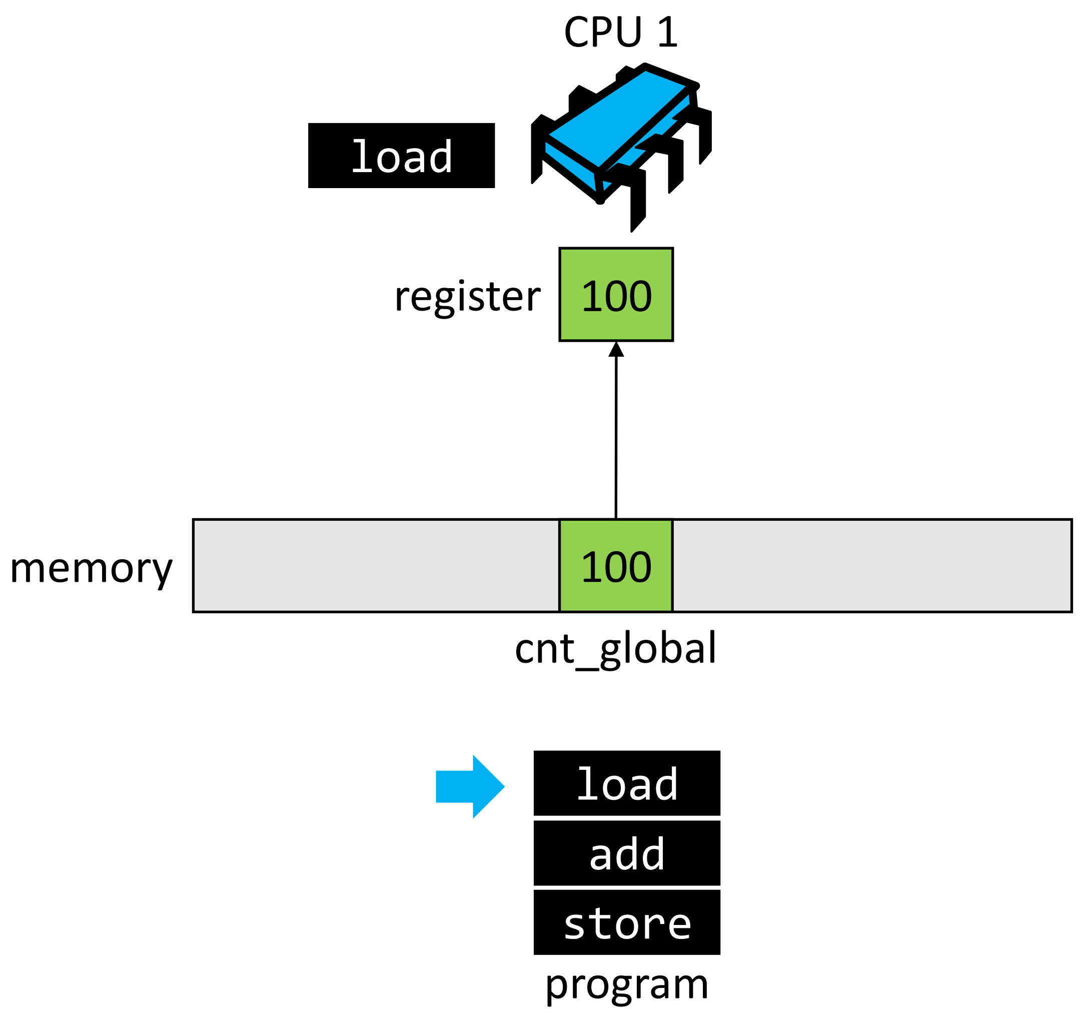
```
cnt_global++;
```

```
movq    cnt_global(%rip), %rax
addq    $1, %rax
movq    %rax, cnt_global(%rip)
```

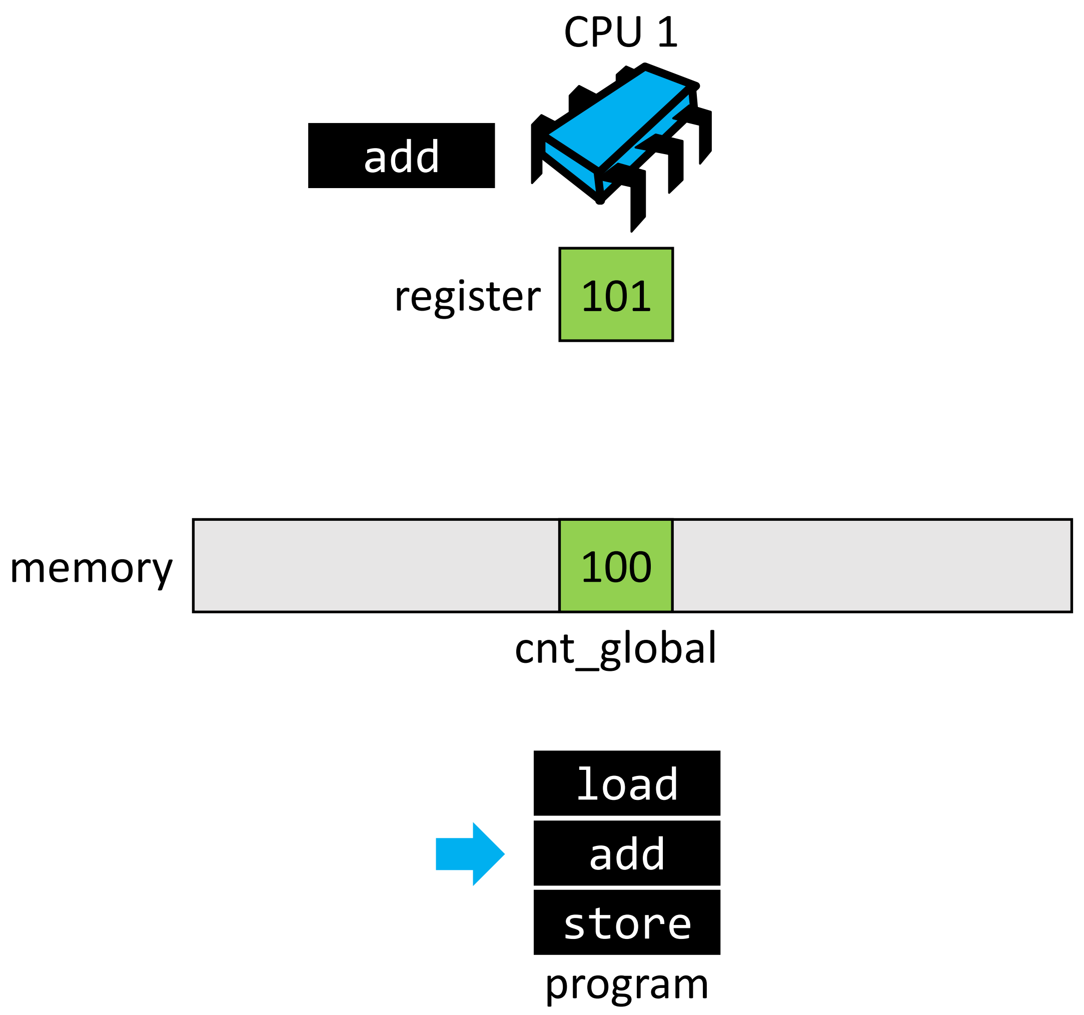
# Example: Single thread



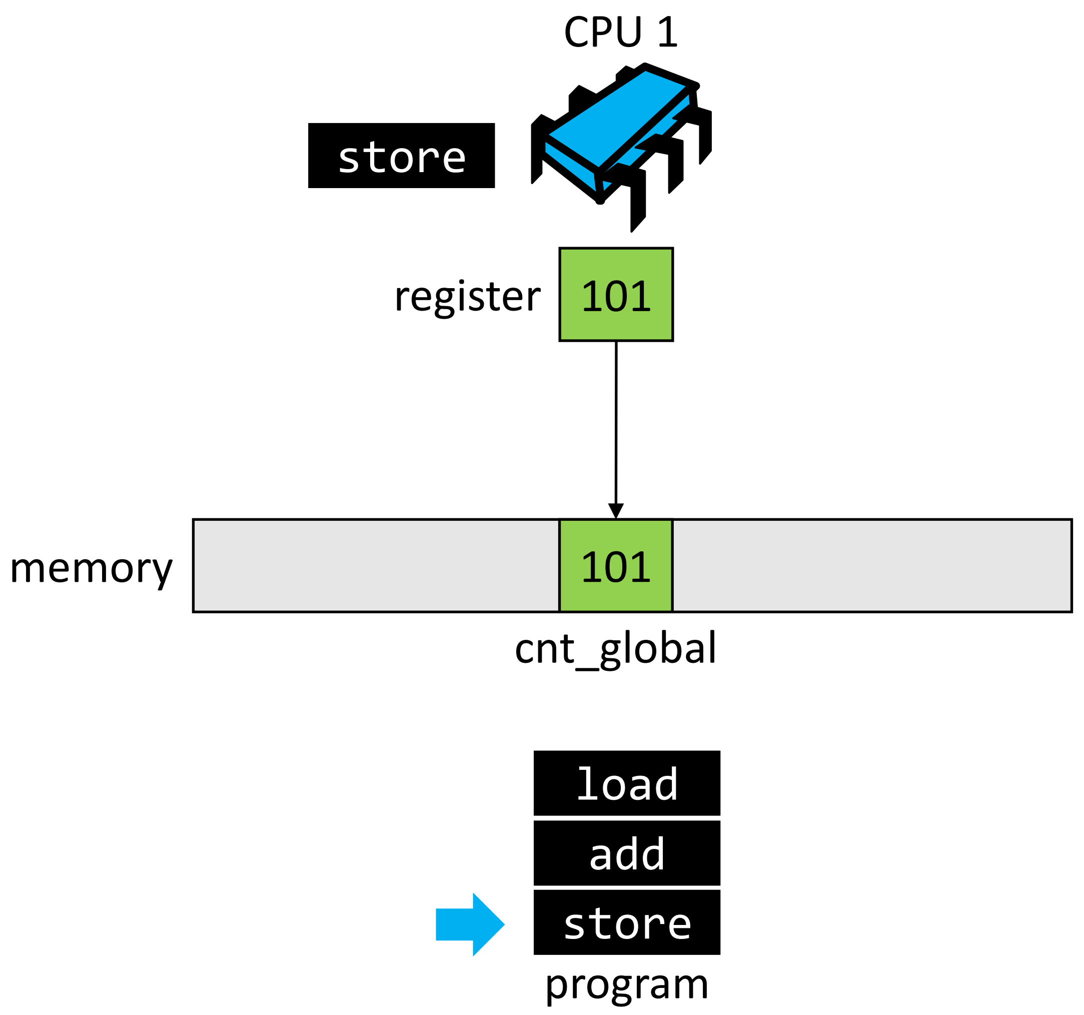
# Example: Single thread



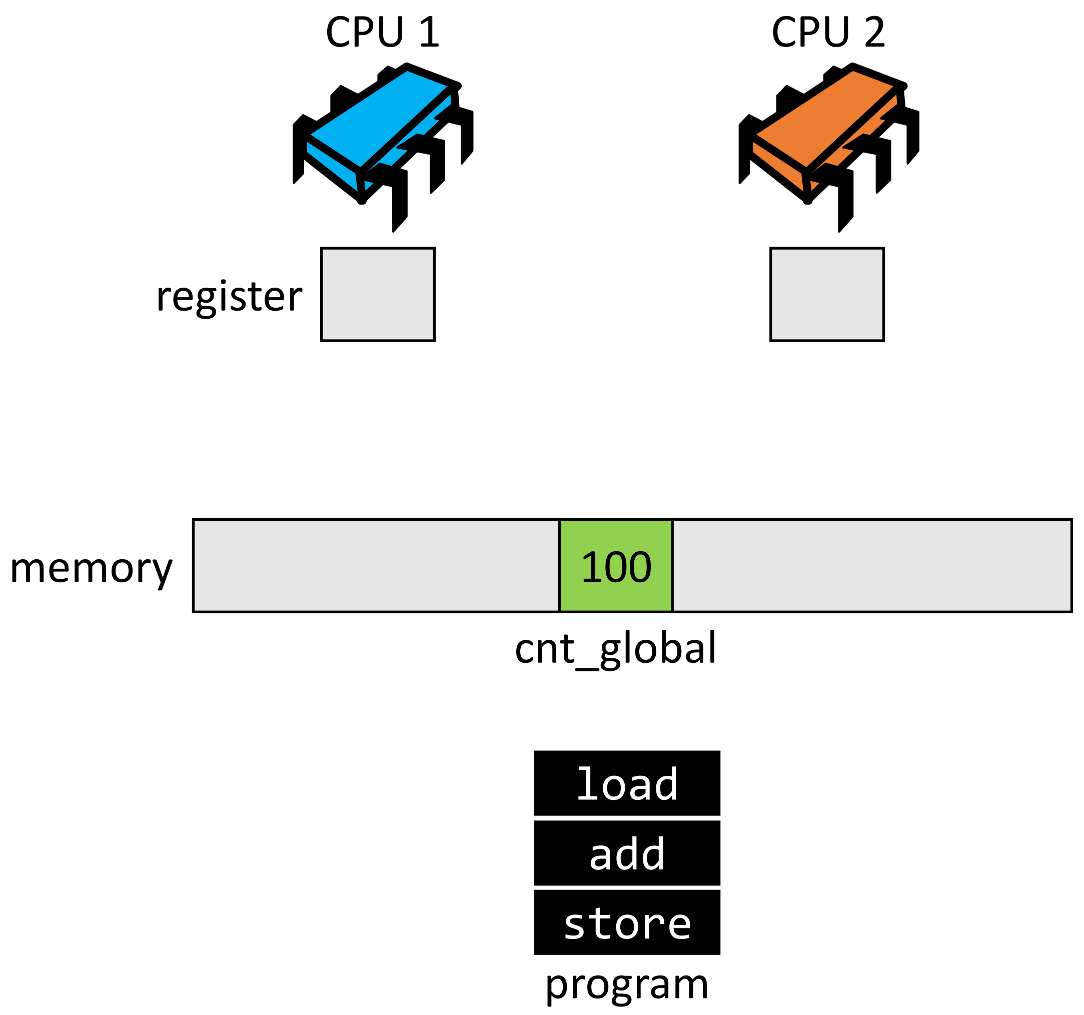
# Example: Single thread



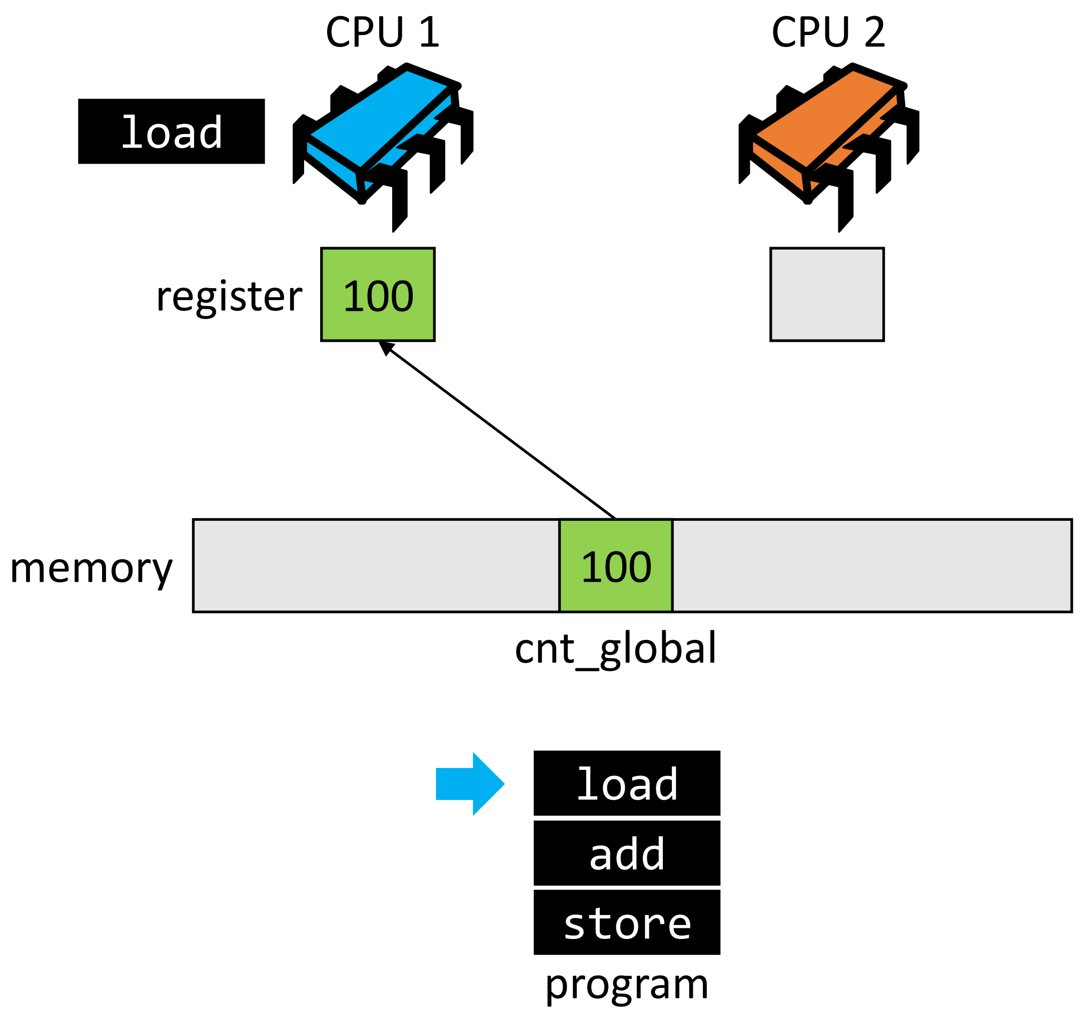
# Example: Single thread



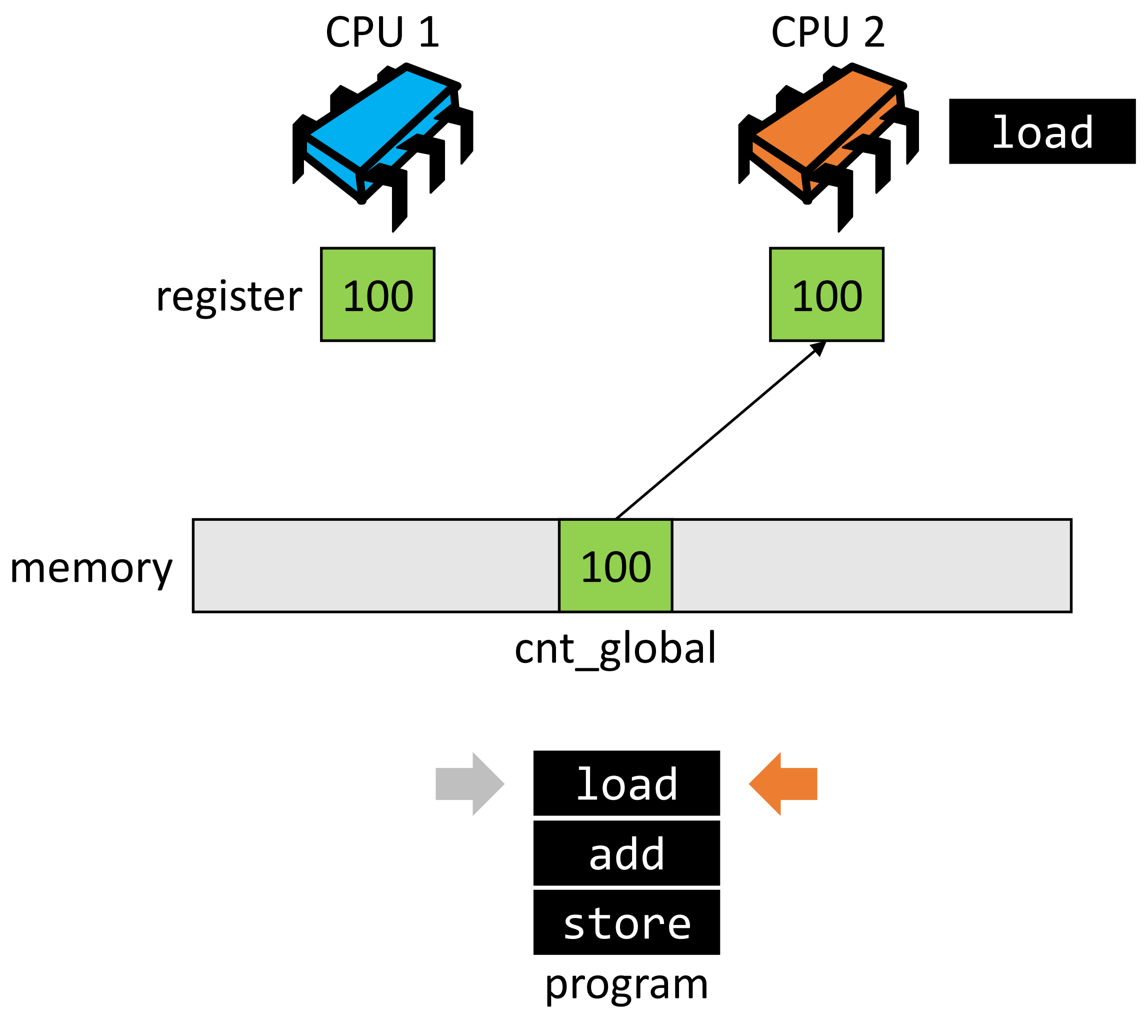
# Example 1: Two threads



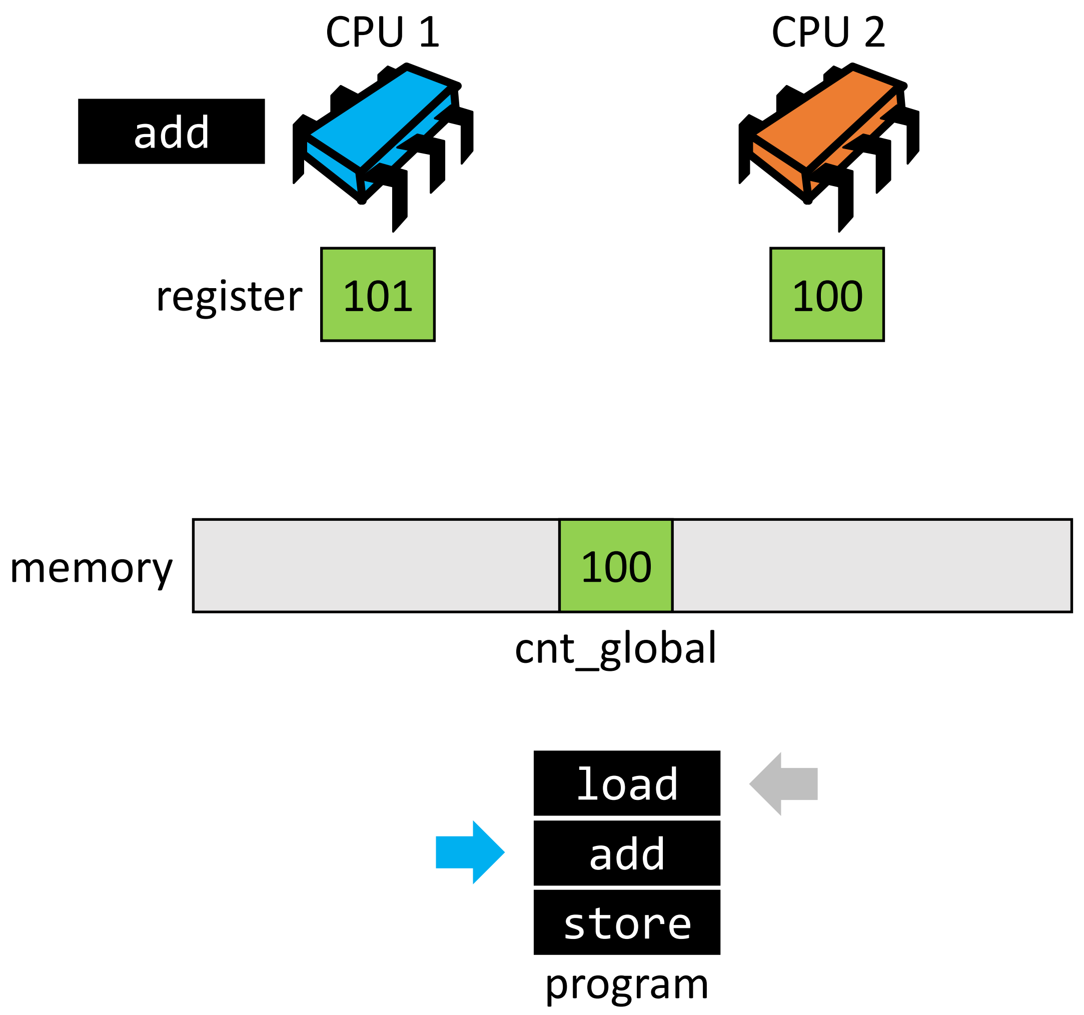
# Example 1: Two threads



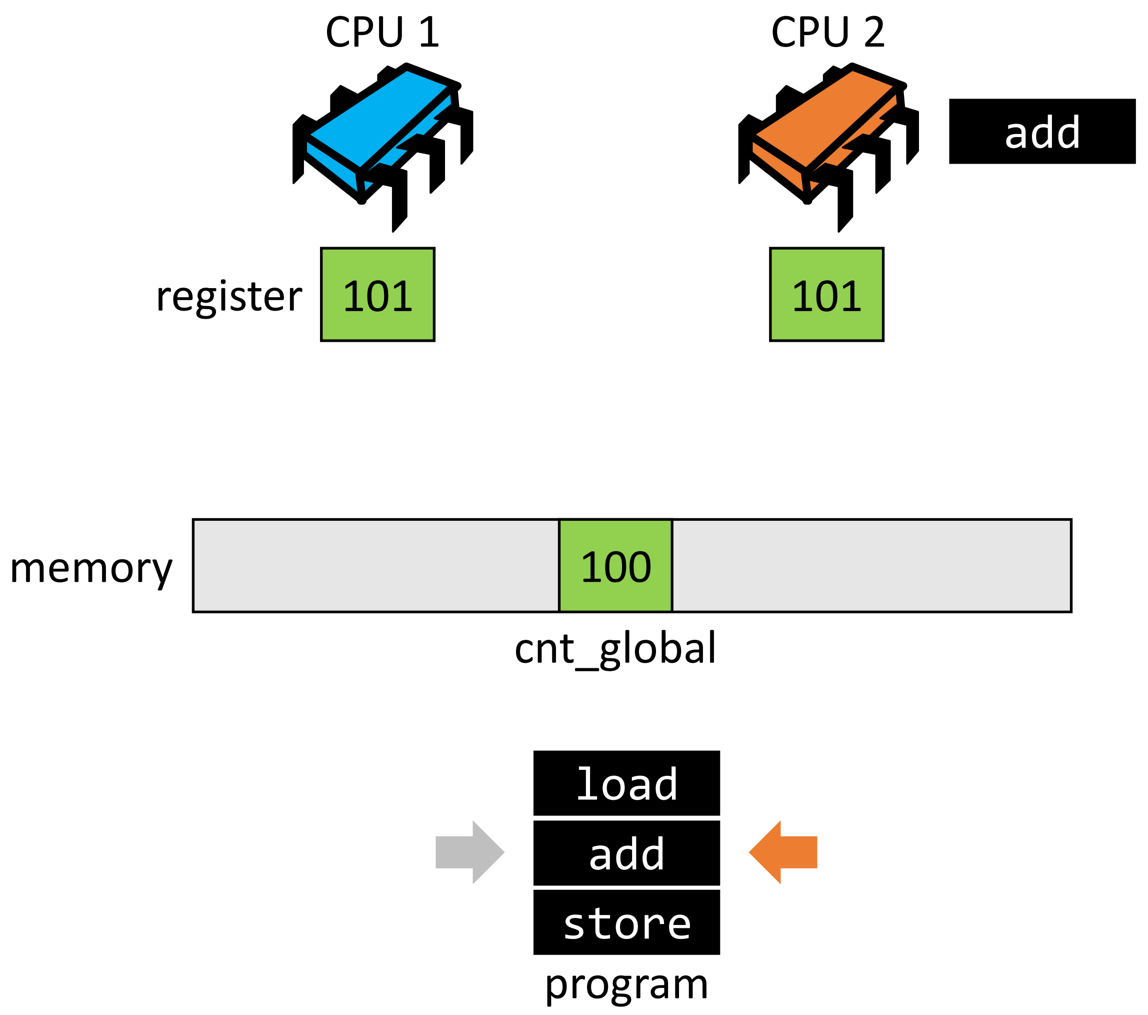
# Example 1: Two threads



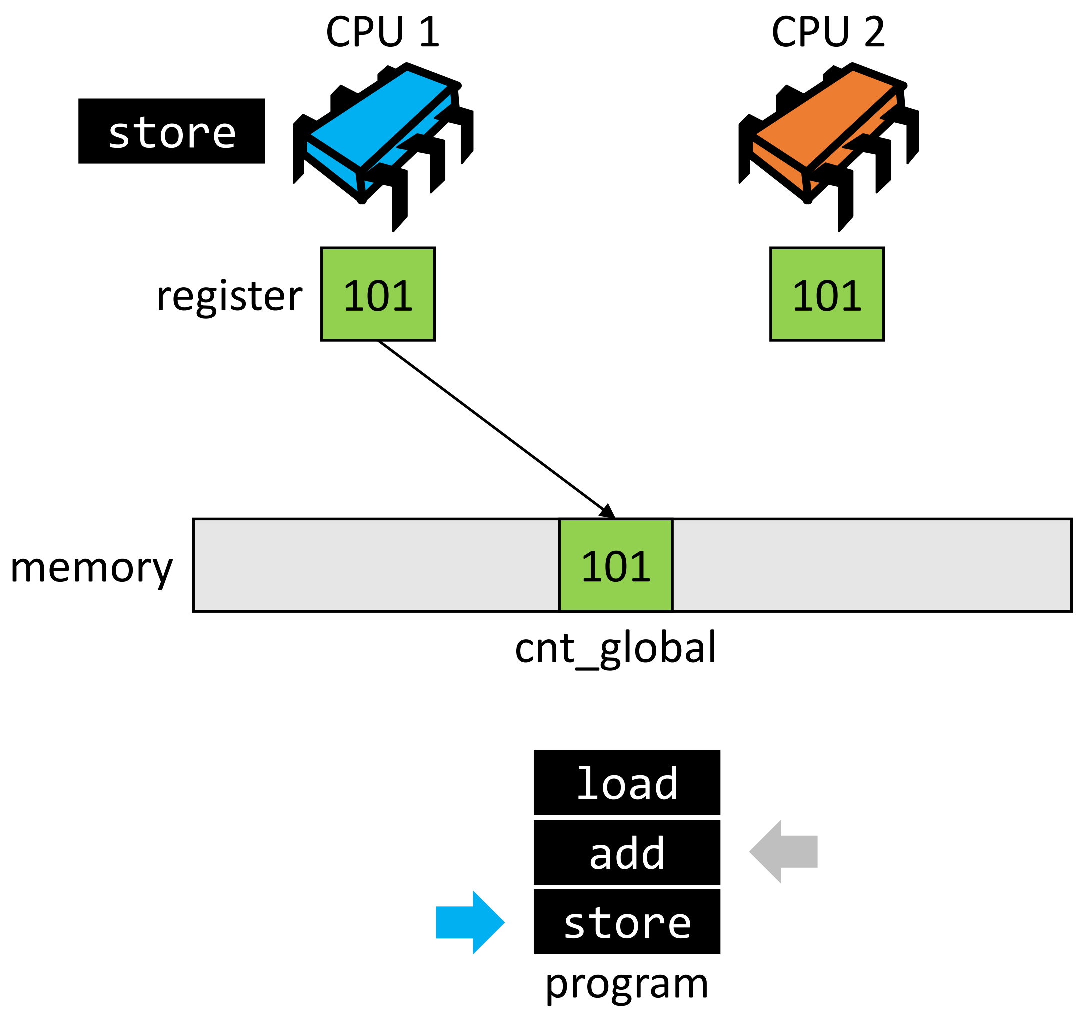
# Example 1: Two threads



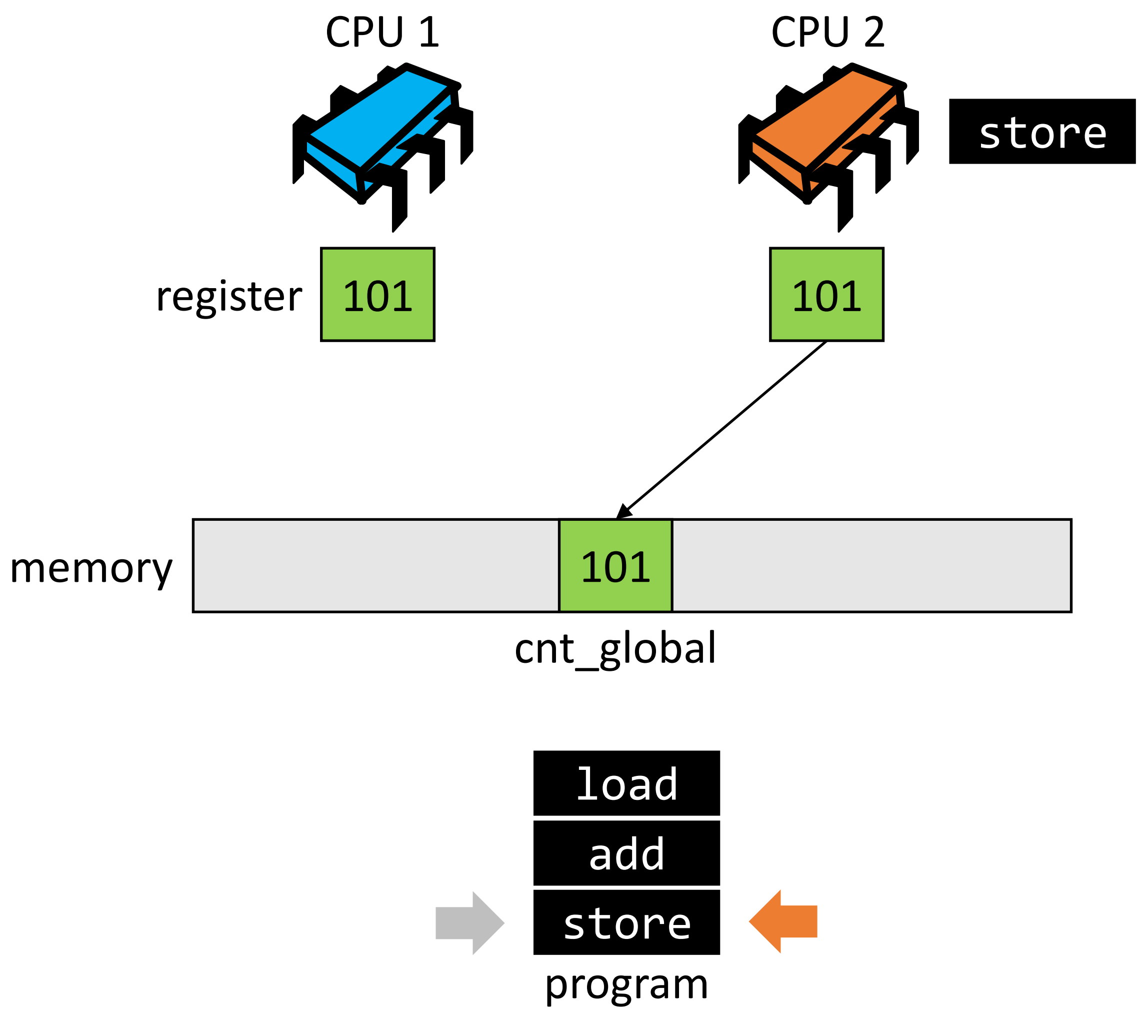
## Example 1: Two threads



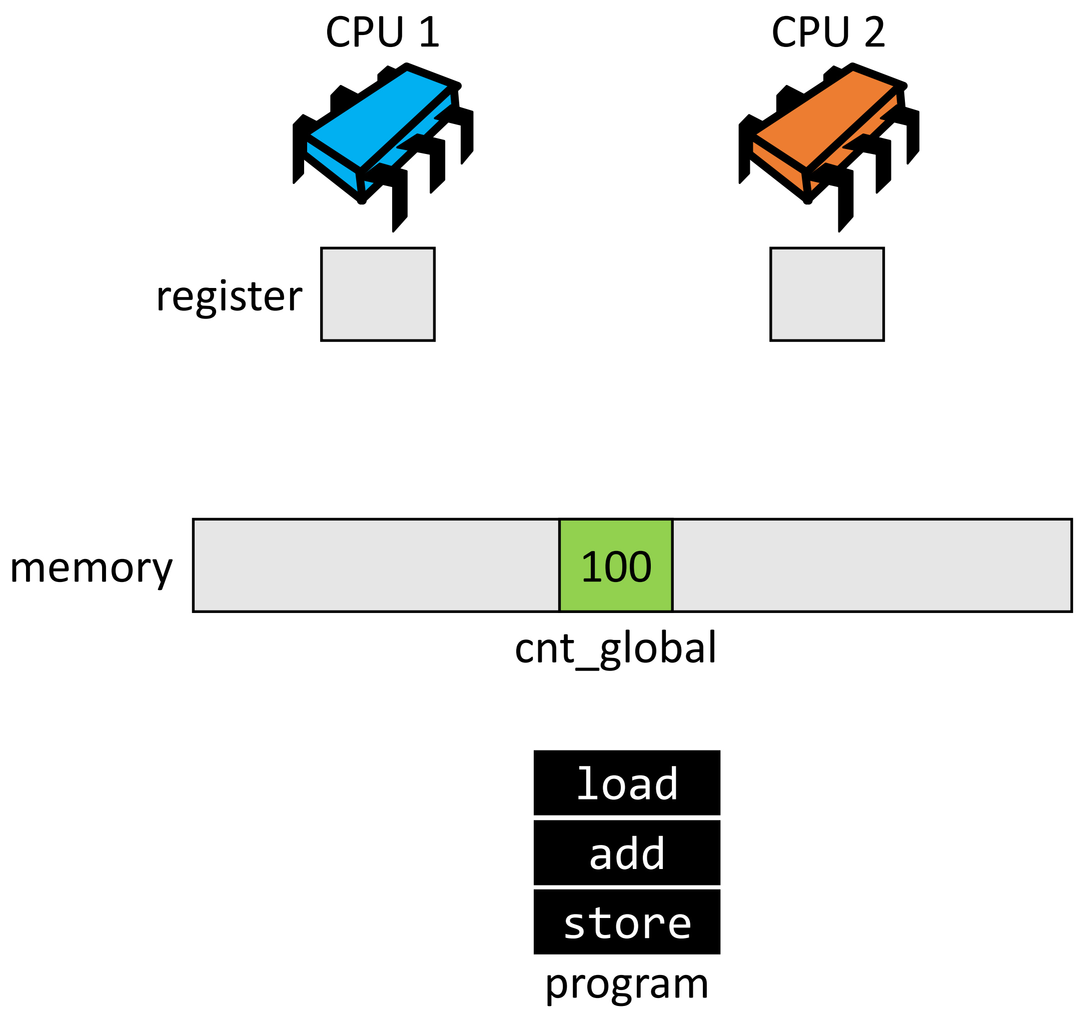
# Example 1: Two threads



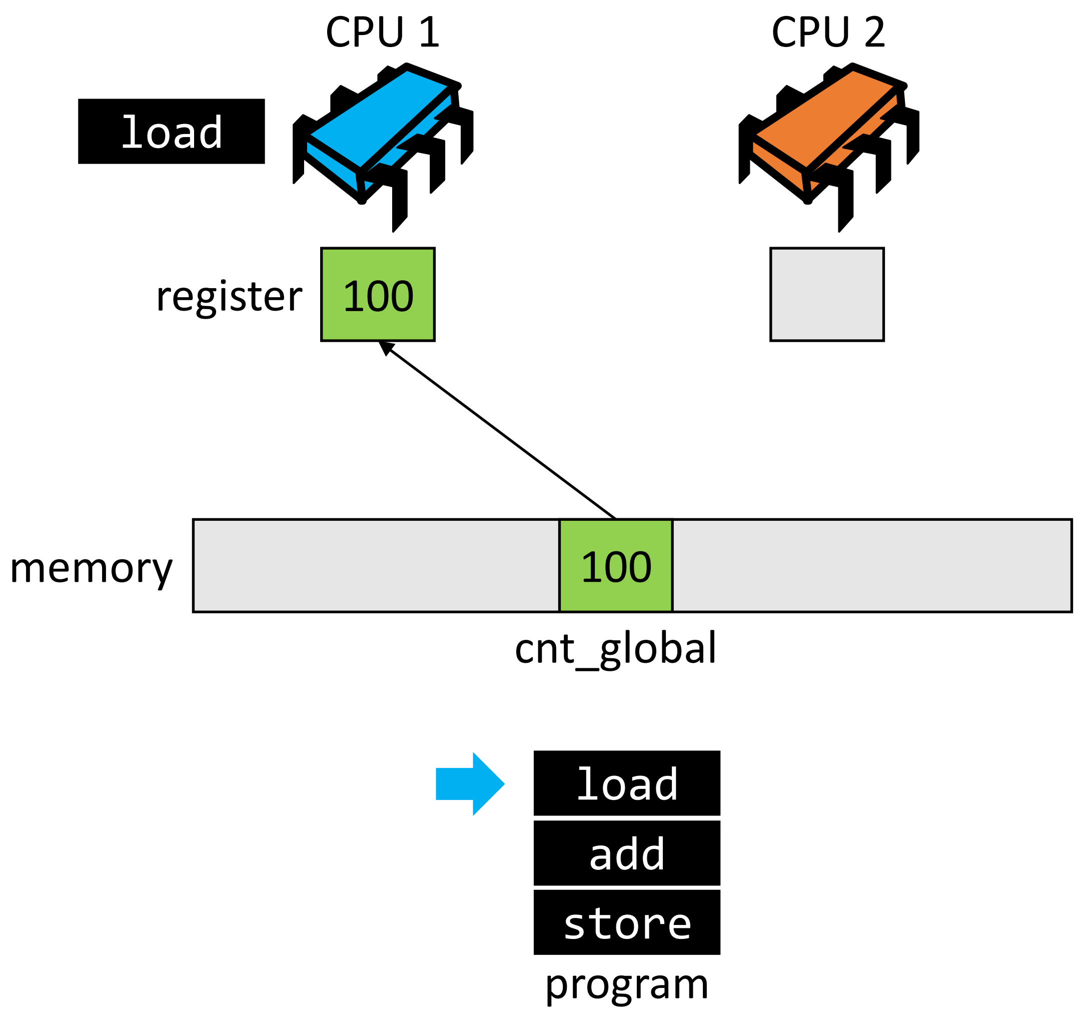
# Example 1: Two threads



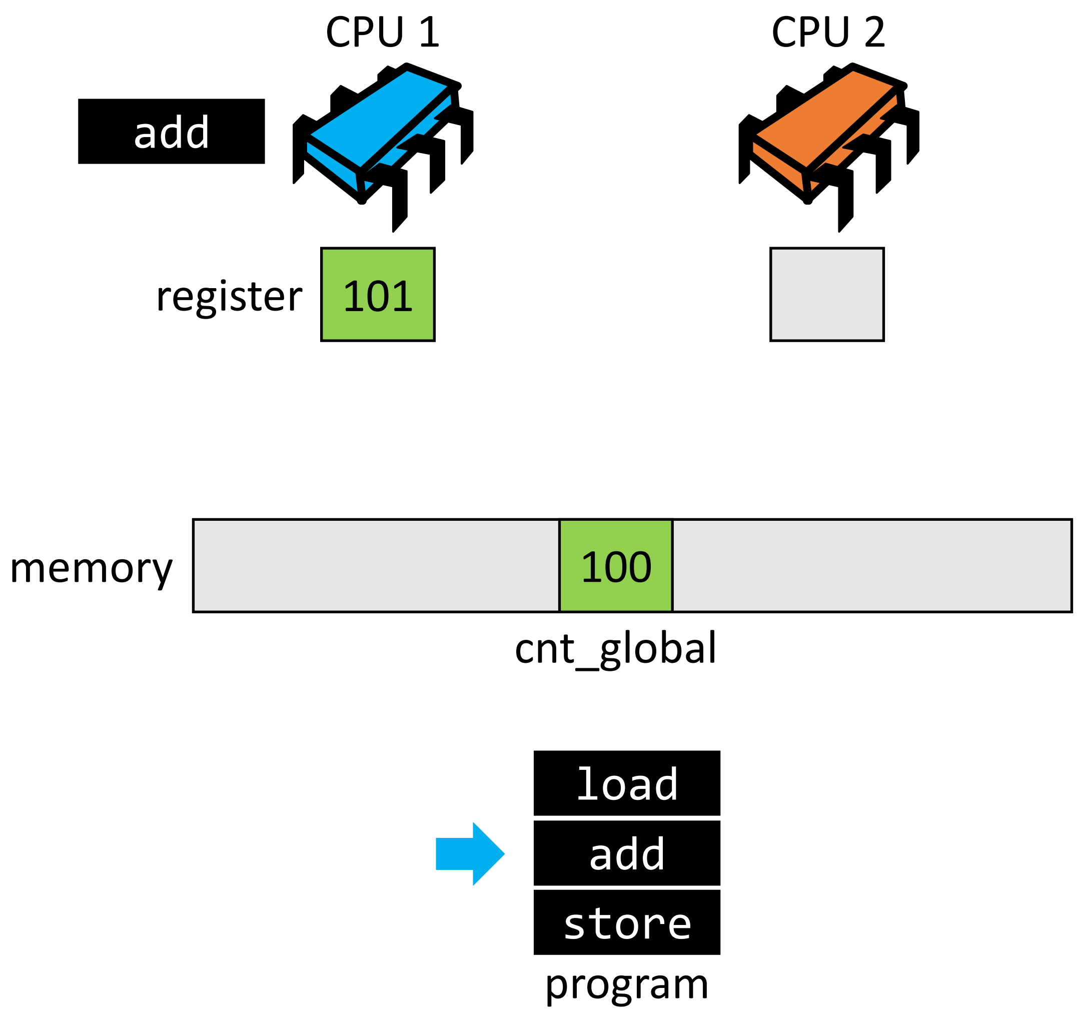
# Example 2: Two threads



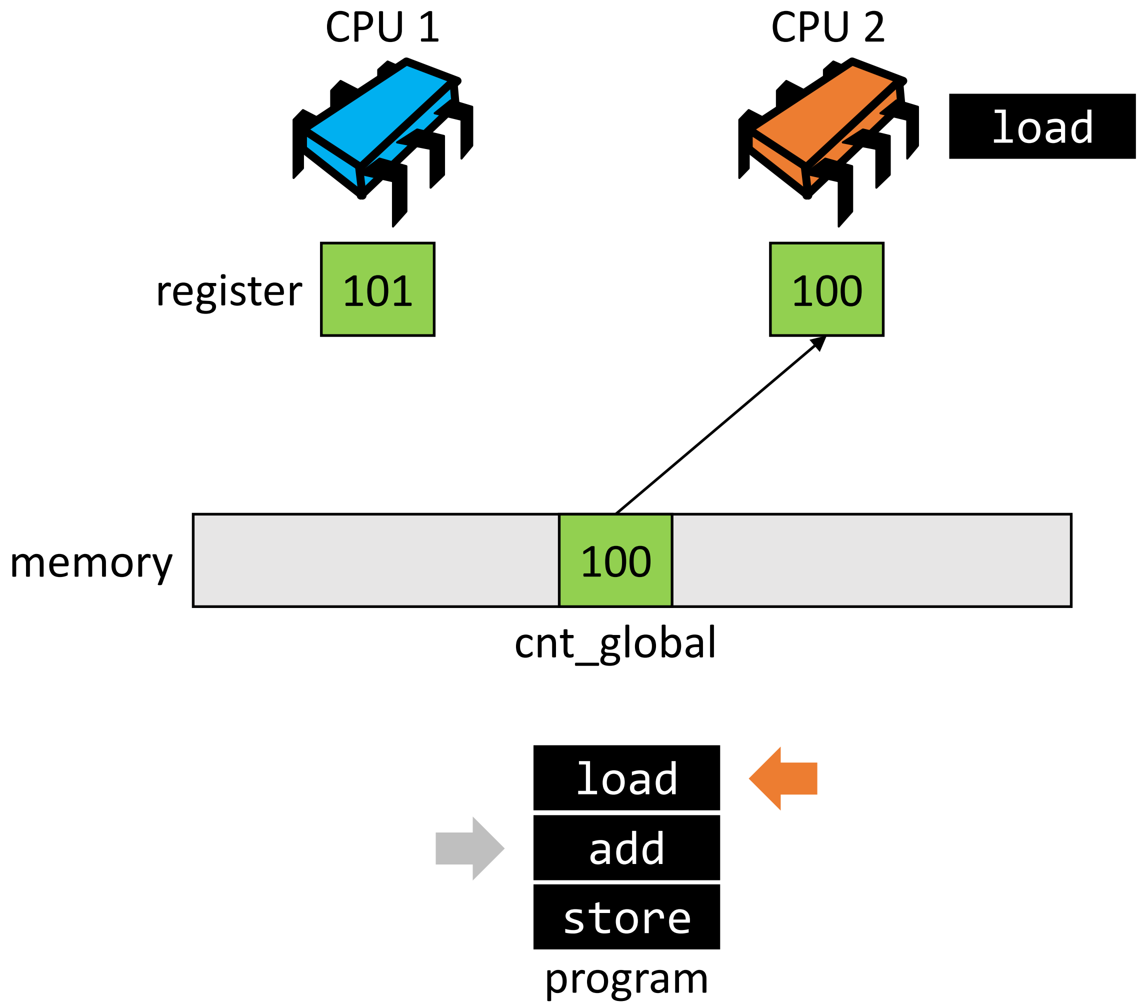
# Example 2: Two threads



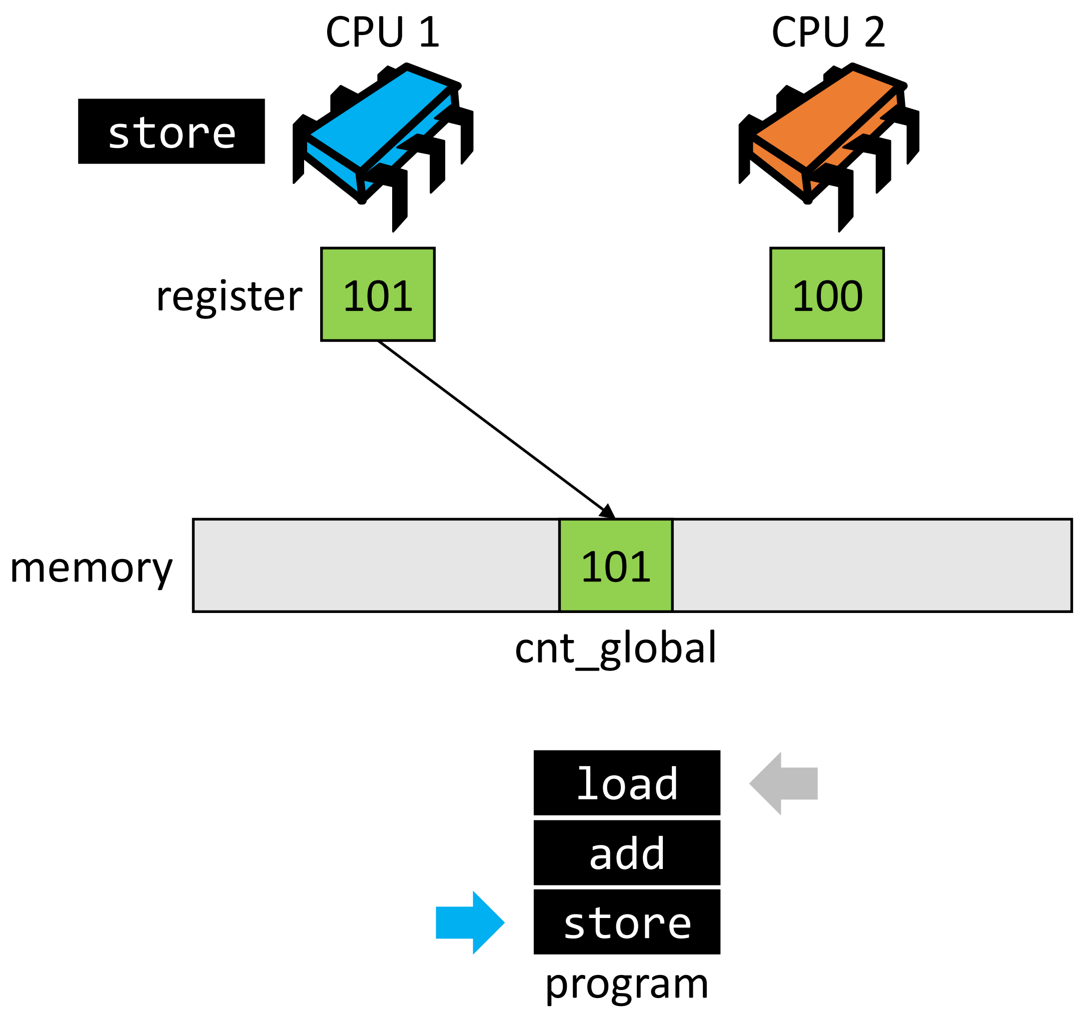
# Example 2: Two threads



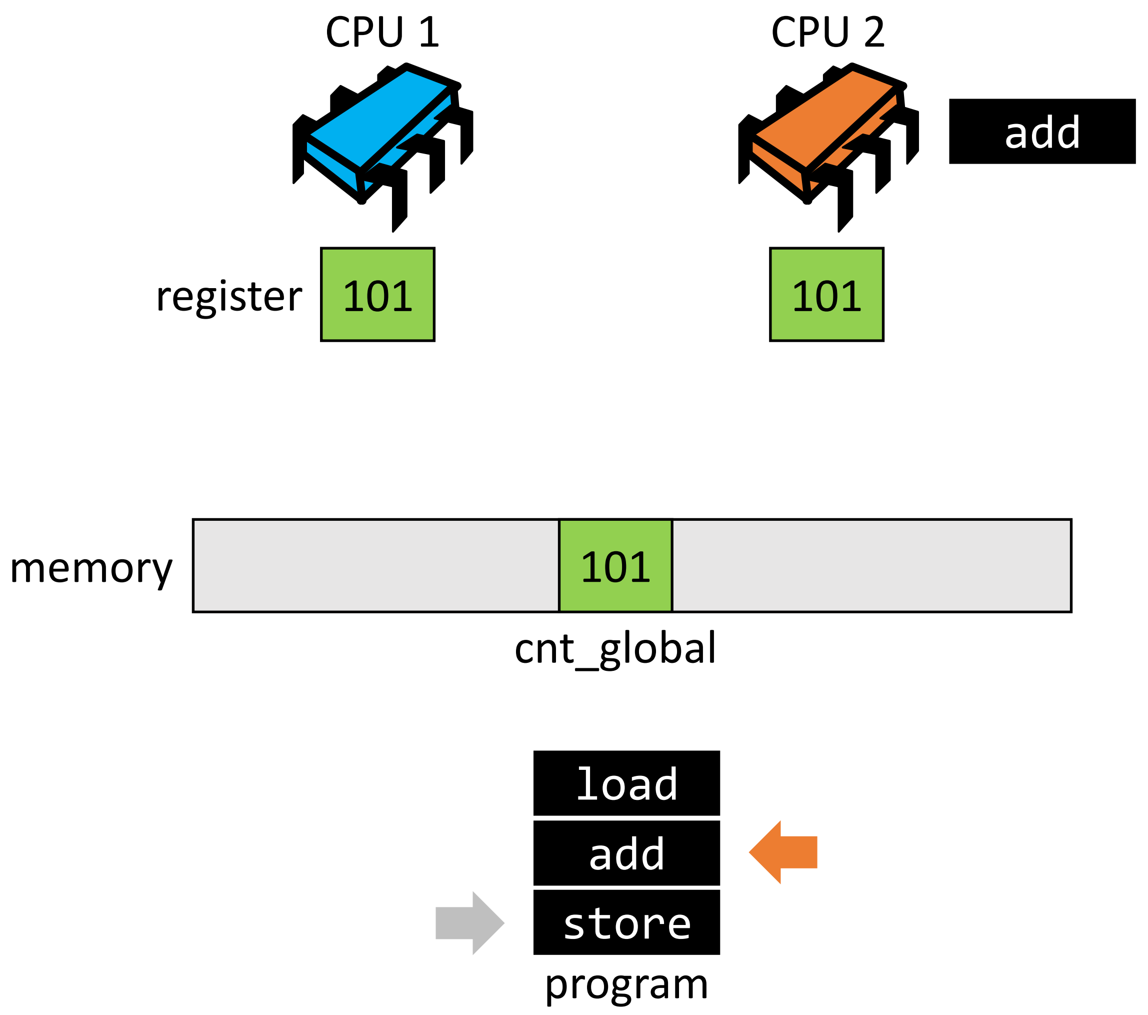
# Example 2: Two threads



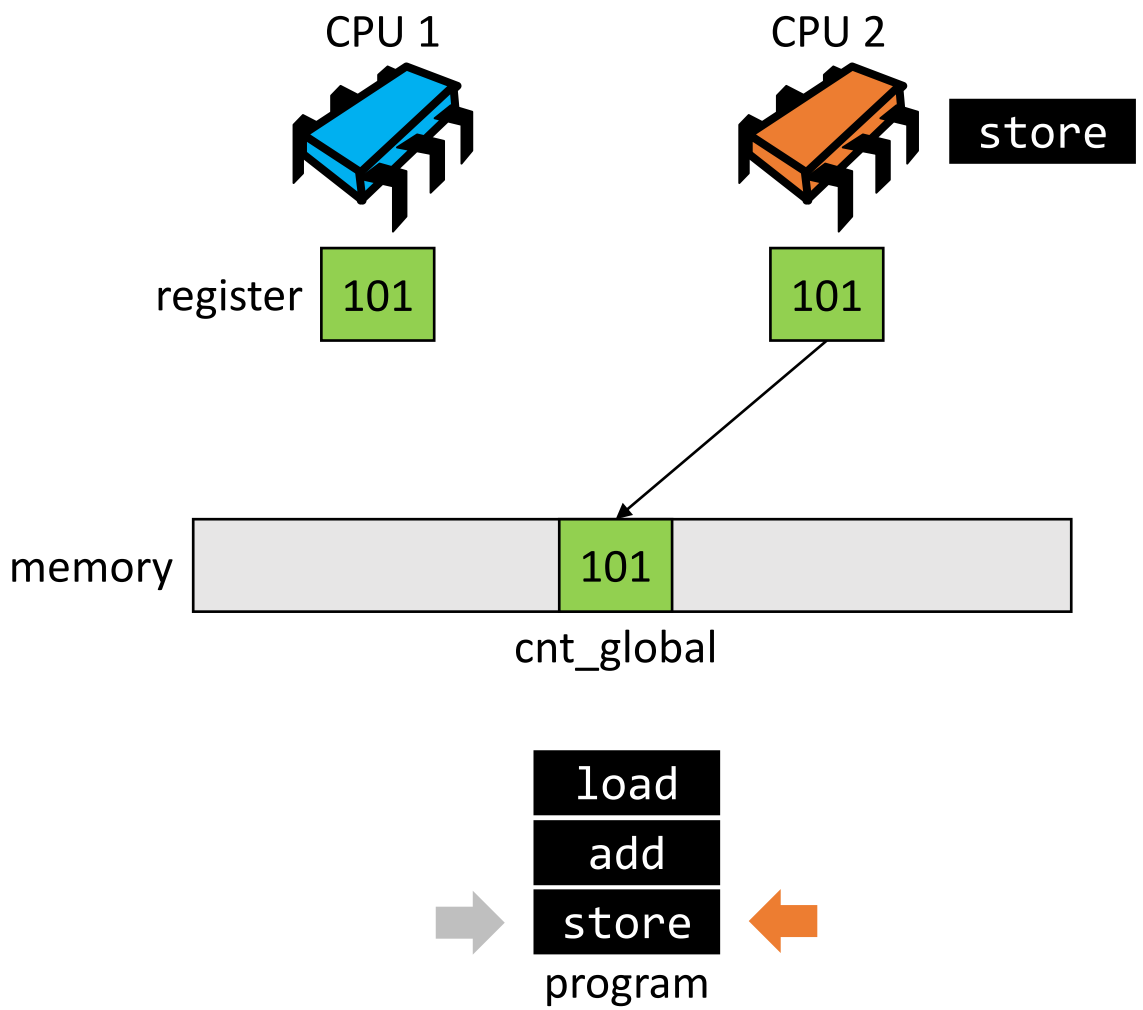
# Example 2: Two threads



# Example 2: Two threads



# Example 2: Two threads



## Mutex
## (MUTual EXclusion)

Concurrent Programming

# Introduction

- What is Mutex?
- Pthread Mutex API
- Example

# What is Mutex?

- Synchronization mechanism for enforcing limits on access to a resource in an environment where there are many threads of execution

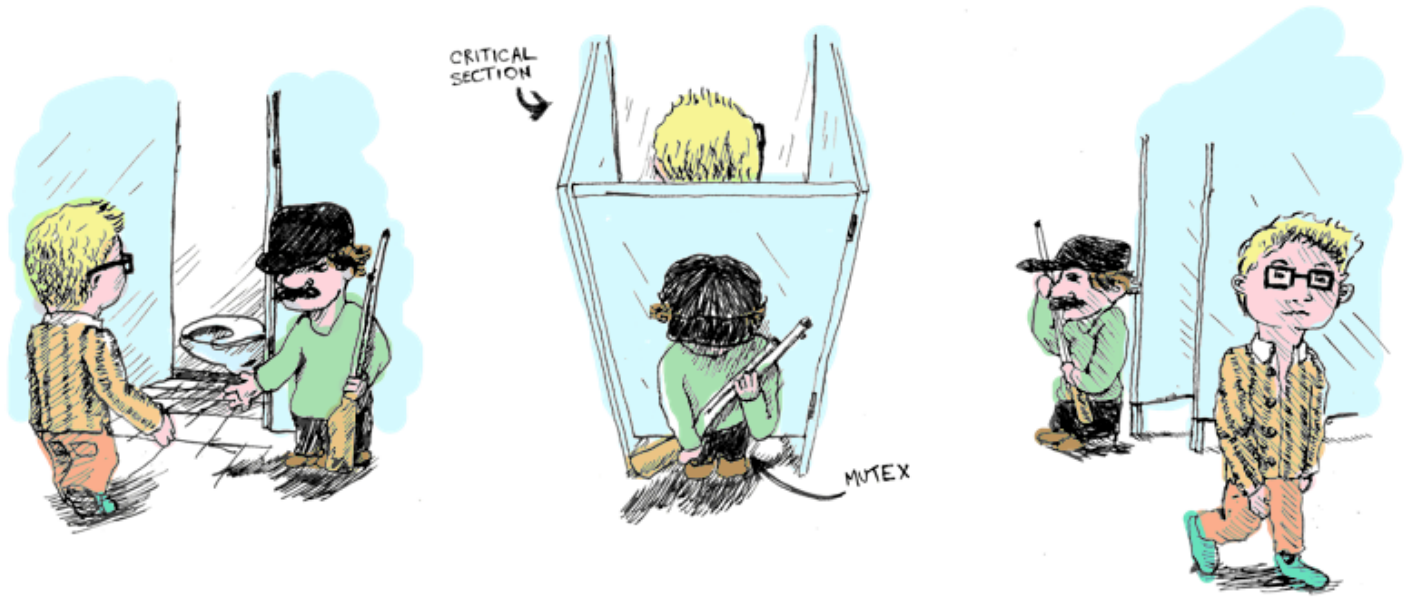


Photo reference: http://www.rudyhuyn.com/blog/2015/12/31/synchroniser-ses-agents-avec-lapplication/mutex/

## Pthread Mutex API

- pthread_mutex_init
- pthread_mutex_lock
- pthread_mutex_trylock
- pthread_mutex_unlock
- more APIs, but not today

## Pthread Mutex API – pthread_mutex_init

```
int pthread_mutex_init(pthread_mutex_t *mutex,
                       const pthread_mutexattr_t *mutexattr);
```

- Initialize the mutex object

```
@param[in]  mutex         Mutex to be initialized
@param[in]  mutexattr     Used for setting attributes of a mutex.(e.g.,Deadlock Checking)
                          Default 0
@return                   Always 0
```

## Pthread Mutex API – pthread_mutex_lock

```
int pthread_mutex_lock(pthread_mutex_t *mutex);
```

- Lock the mutex object. If the mutex is already locked, the calling thread shall block until the mutex becomes available.

```
@param[in]  mutex          Mutex to be locked
@return                    0 if acquired. Error number related to the mutexattr if failed.
```

# Pthread Mutex API – pthread_mutex_trylock

```
int pthread_mutex_trylock(pthread_mutex_t *mutex);
```

- Lock the mutex object. If the mutex is already locked, return immediately.

```
@param[in]  mutex         Mutex to be locked
@return                   0 if acquired. Error number related to the mutexattr if failed.
```

# Pthread Mutex API – pthread_mutex_unlock

```
int pthread_mutex_unlock(pthread_mutex_t *mutex);
```

- Release the mutex object.

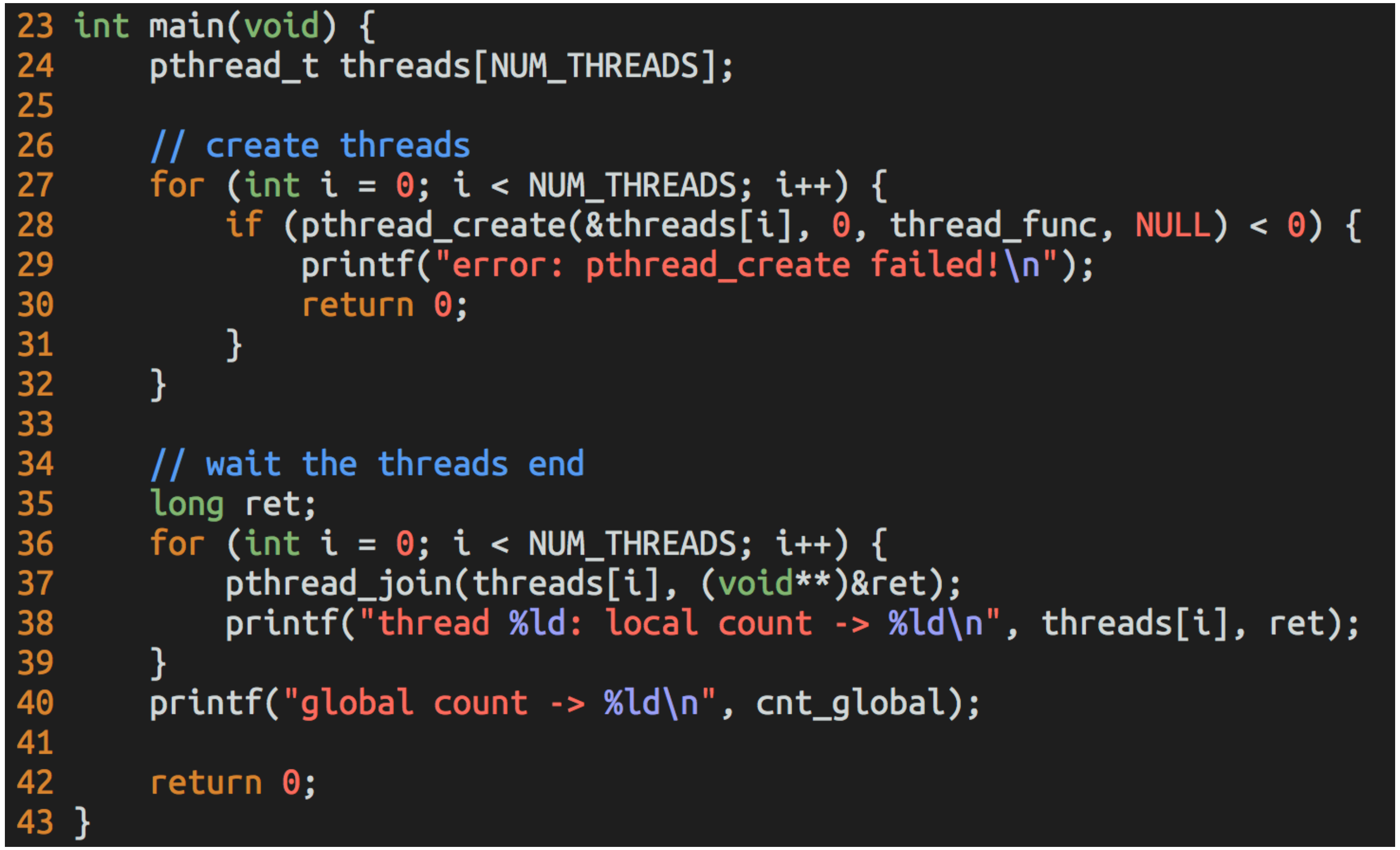
```
@param[in]  mutex          Mutex to be released
@return                    0 if released. Error number related to the mutexattr if failed.
```

# Example

< prac_mutex.cpp >

```c
#include <stdio.h>
#include <pthread.h>

#define NUM_THREADS     10
#define NUM_INCREMENT   1000000

long cnt_global = 0;
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;

void* thread_func(void* arg) {
    long cnt_local = 0;

    for (int i = 0; i < NUM_INCREMENT; i++) {
        pthread_mutex_lock(&mutex);
        cnt_global++;   // increase global value
        pthread_mutex_unlock(&mutex);
        cnt_local++;    // increase local value
    }

    return (void*)cnt_local;
}
```

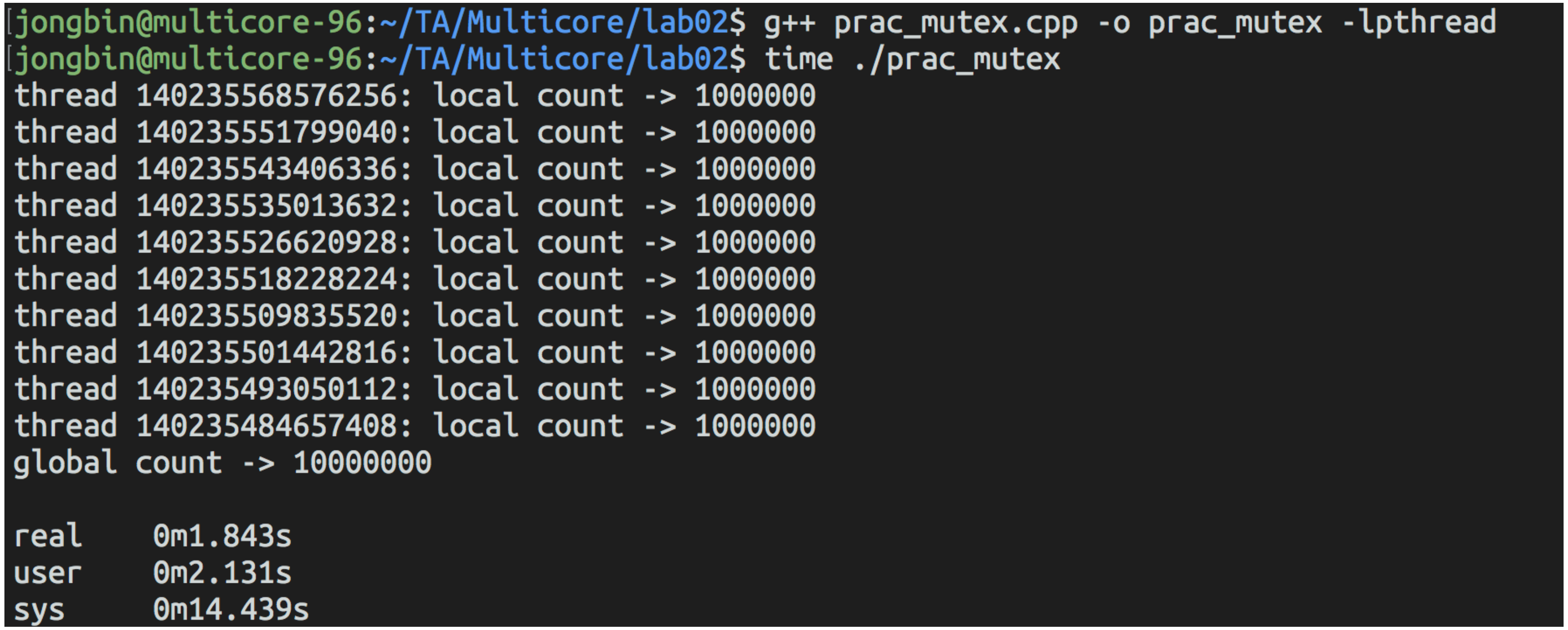
# Example (continue..)



```c
int main(void) {
    pthread_t threads[NUM_THREADS];

    // create threads
    for (int i = 0; i < NUM_THREADS; i++) {
        if (pthread_create(&threads[i], 0, thread_func, NULL) < 0) {
            printf("error: pthread_create failed!\n");
            return 0;
        }
    }

    // wait the threads end
    long ret;
    for (int i = 0; i < NUM_THREADS; i++) {
        pthread_join(threads[i], (void**)&ret);
        printf("thread %ld: local count -> %ld\n", threads[i], ret);
    }
    printf("global count -> %ld\n", cnt_global);

    return 0;
}
```

# Example (continue..)

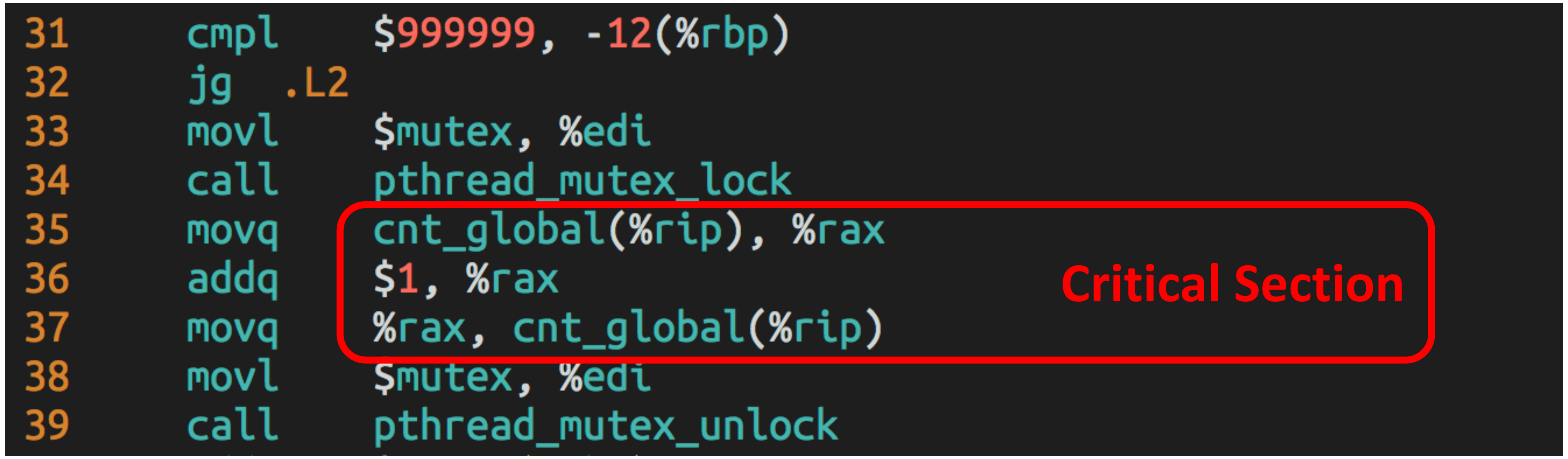
< Result >



```
[jongbin@multicore-96:~/TA/Multicore/lab02$ g++ prac_mutex.cpp -o prac_mutex -lpthread
[jongbin@multicore-96:~/TA/Multicore/lab02$ time ./prac_mutex
thread 140235568576256: local count -> 1000000
thread 140235551799040: local count -> 1000000
thread 140235543406336: local count -> 1000000
thread 140235535013632: local count -> 1000000
thread 140235526620928: local count -> 1000000
thread 140235518228224: local count -> 1000000
thread 140235509835520: local count -> 1000000
thread 140235501442816: local count -> 1000000
thread 140235493050112: local count -> 1000000
thread 140235484657408: local count -> 1000000
global count -> 10000000

real    0m1.843s
user    0m2.131s
sys     0m14.439s
```

# Example (continue..)

< Assembly instructions for *cnt_global++* in the C code >



```
        cmpl    $999999, -12(%rbp)
        jg   .L2
        movl    $mutex, %edi
        call    pthread_mutex_lock
        movq    cnt_global(%rip), %rax
        addq    $1, %rax
        movq    %rax, cnt_global(%rip)
        movl    $mutex, %edi
        call    pthread_mutex_unlock
```

Thank You