

Modelos de consistencia relajados

```
for (i=iproc ; i<n ; i=i+nproc) ①  
    sump = sump + a[i];  
    lock(k);  
    sum = sum + sump; /* SC */ ②  
    unlock(k);  
    ... ③
```

ADD R1,R1,R2 ; R1=R1+R2

ST (R3),R1 ; sum ← R1

ST (R5),#0 ; k ← 0

ST (R5),#0 ; k ← 0

ADD R1,R1,R2 ; R1=R1+R2

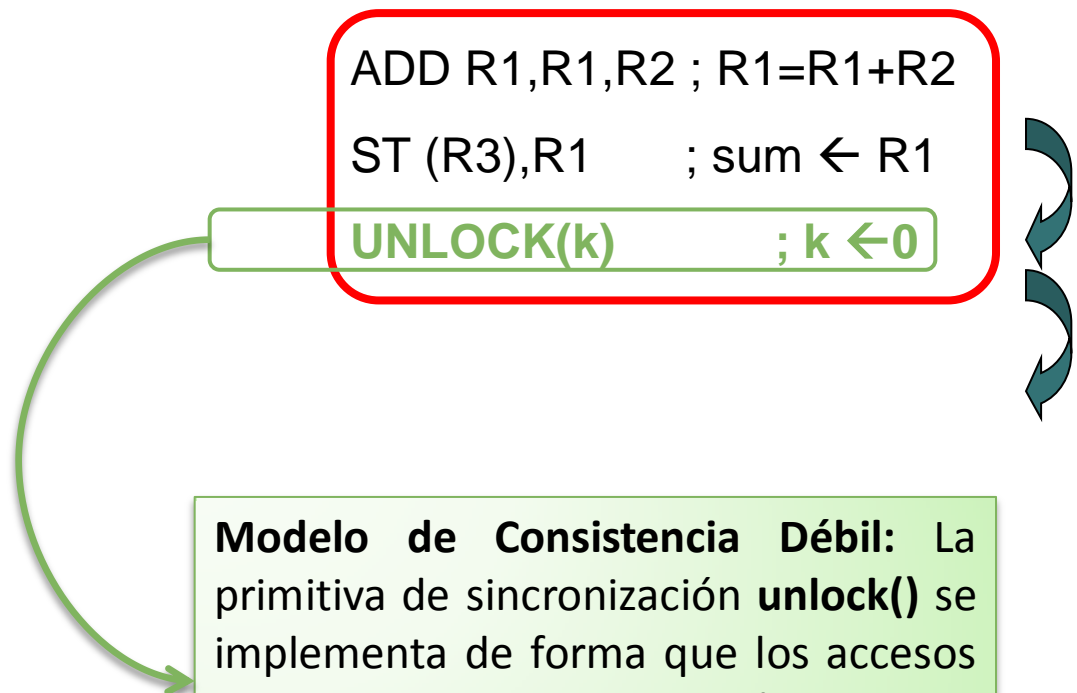
ST (R3),R1 ; sum ← R1

La primitiva de sincronización unlock() debe implementarse de otra forma: asegurando que los accesos a memoria previos se han completado

No se garantiza el funcionamiento correcto de la sección crítica

Modelo de consistencia débil

```
ADD R1,R1,R2 ; R1=R1+R2  
ST (R3),R1    ; sum ← R1  
UNLOCK(k)     ; k ← 0
```



Modelo de Consistencia Débil: La primitiva de sincronización **unlock()** se implementa de forma que los accesos a memoria previos se completen antes que ella y los que vienen detrás solo empiecen cuando termine **unlock()**

Modelo de consistencia de liberación

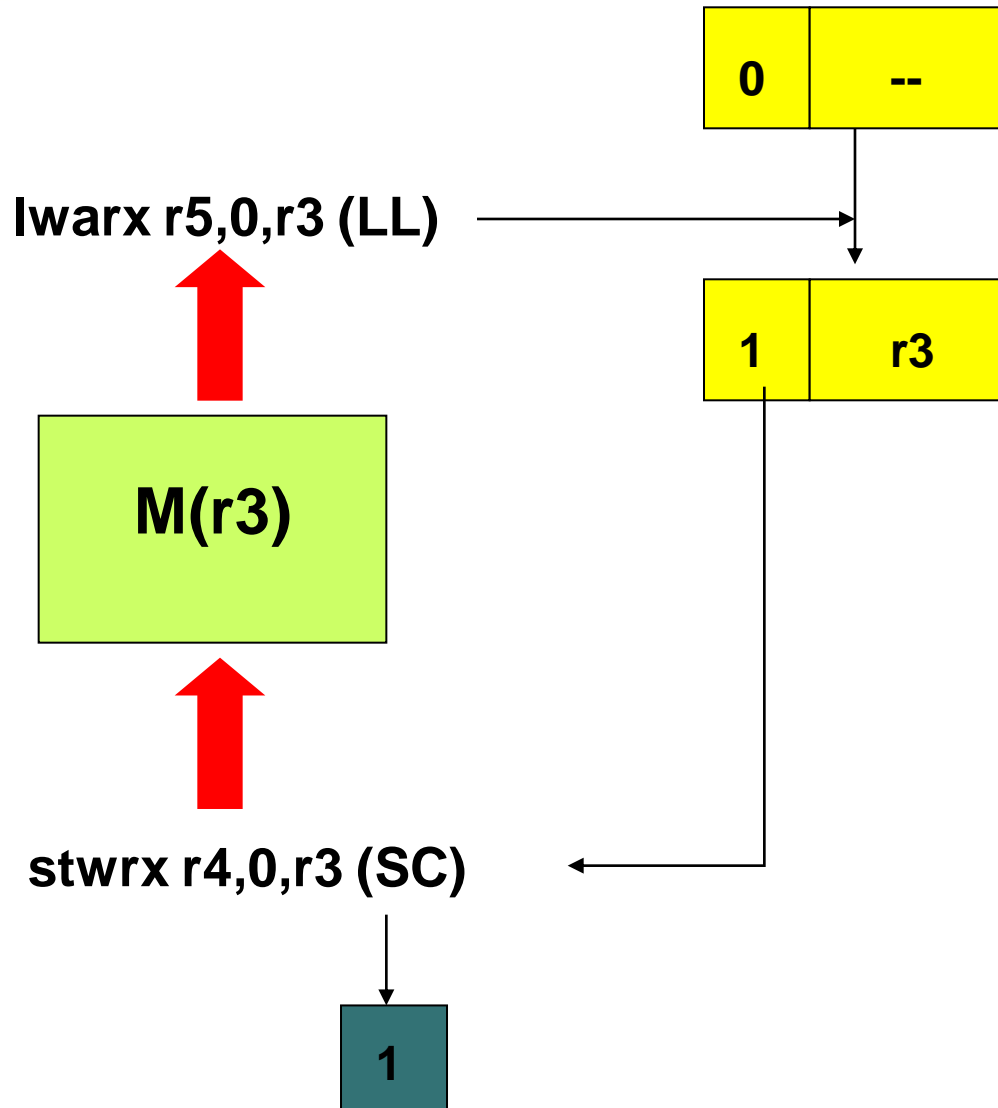
ADD R1,R1,R2 ; R1=R1+R2

ST (R3),R1 ; sum \leftarrow R1

UNLOCK(k) ; k \leftarrow 0

Modelo de Consistencia de liberación: La primitiva de sincronización **unlock()** se implementa de forma que los accesos a memoria previos se completen antes que ella

Sincronización LL/SC



Sincronización LL/SC

