



Test delle istruzioni atomiche per processori riscv



Abbiamo testato le operazioni atomiche:


- atomic swap
- load reserve, store conditional

rispettivamente grazie alle funzioni C:

- `pthread_spin_lock()`
- `atomic_compare_exchange()`



Spin lock




Lo spin lock grazie ad un'operazione di atomic-swap (marchiata come acquire, ergo tutte le operazioni successive alla swap non potevano essere eseguite prima dallo scheduler). Successivamente se l'atomic swap dovesse aver fallito, grazie alla "bnez" invece che concludere la funzione con una ret, passiamo ad un loop, che precede la logica di load reserve-store conditional

```
000000000001a940 <__pthread_spin_lock>:
1a940: 4785          li a5,1
1a942: 0cf527af      amoswap.w.aq a5,a5,(a0)
1a946: 2781          sext.w a5,a5
1a948: e399          bnez a5,1a94e <__pthread_spin_lock+0xe>
1a94a: 4501          li a0,0 PRESO LOCK
1a94c: 8082          ret
1a94e: 4705          li a4,1 NON PRESO LOCK
1a950: 411c          lw a5,0(a0)
1a952: fffd          bnez a5,1a950 <__pthread_spin_lock+0x10>
1a954: 100527af      lr.w a5,(a0)
1a958: e781          bnez a5,1a960 <__pthread_spin_lock+0x20>
1a95a: 1ce526af      sc.w.aq a3,a4,(a0)
1a95e: fafd          bnez a3,1a954 <__pthread_spin_lock+0x14>
1a960: 2781          sext.w a5,a5
1a962: f7fd          bnez a5,1a950 <__pthread_spin_lock+0x10>
1a964: 4501          li a0,0 PRESO LOCK
1a966: 8082          ret

000000000001a968 <__pthread_spin_unlock>:
1a968: 0f50000f      fence iorw,ow
1a96c: 0805202f      amoswap.w zero,zero,(a0)
1a970: 4501          li a0,0
1a972: 8082          ret
```

Il core switch1 (a riga 618) è il primo a concludere correttamente l'atomic swap, quindi tutte le successive atomic swap non ritorneranno più 0, ma 1 (l'indirizzo 0x8a8b0 si vede dal file di object dump che è l'indirizzo della variabile globale spin lock).

```
612 290489370000: board.cache_hierarchy.ruby_system.l1_controllers1.L1Dcache: No tag match for address: 0x101268940
613 290489371000: board.processor.switch1.core.mmu.dtb: lookup(vpn=0x8a8b0, asid=0): hit ppn 0x10107d
614 290489371000: board.processor.switch1.core.mmu.dtb: translate(vpn=0x8a8b0, asid=0): 0x10107d8b0
615 290489372000: board.cache_hierarchy.ruby_system.l1_controllers1.L1Dcache: address: 0x10107d880 found
616 290489372000: board.cache_hierarchy.ruby_system.l1_controllers1.L1Icache: No tag match for address: 0x10107d880
617 290489372000: board.cache_hierarchy.ruby_system.l1_controllers1: MESI_Two_Level-L1cache.sm:860: [ 0x1 0x0 0x0 0x0 0x0 0x0
618 290489372000: global: Testing Lock for addr: 0x10107d880 cur -1 con 1
619 290489372000: board.cache_hierarchy.ruby_system.l1_controllers1.sequencer: Cache hit at [0x10107d8b0, line 0x10107d880]
620 290489372000: board.cache_hierarchy.ruby_system.l1_controllers1.L1Dcache: address: 0x10107d880 found
621 290489372000: board.cache_hierarchy.ruby_system.l1_controllers1.L1Icache: No tag match for address: 0x10107d880
622 290489373000: board.processor.switch1.core.mmu.itb: lookup(vpn=0x1a944, asid=0): hit ppn 0x101268
623 290489373000: board.processor.switch1.core.mmu.itb: translate(vpn=0x1a944, asid=0): 0x101268944
624 290489373000: board.cache_hierarchy.ruby_system.l2_controllers0: MESI_Two_Level-L2cache.sm:373: Addr: 0x10107d880 State: M
625 290489374000: board.cache_hierarchy.ruby_system.l1_controllers1.L1Dcache: No tag match for address: 0x101268940
626 290489374000: board.cache_hierarchy.ruby_system.l1_controllers1: MESI_Two_Level-L1cache.sm:843: [ 0x85 0x47 0xaf 0x27 0xf5
627 290489374000: board.cache_hierarchy.ruby_system.l1_controllers1.sequencer: Cache hit at [0x101268944, line 0x101268940]
628 290489374000: board.cache_hierarchy.ruby_system.l1_controllers1.L1Dcache: No tag match for address: 0x101268940
629 290489375000: board.processor.switch1.core.mmu.itb: lookup(vpn=0x1a948, asid=0): hit ppn 0x101268 bnez
630 290489375000: board.processor.switch1.core.mmu.itb: translate(vpn=0x1a948, asid=0): 0x101268948
631 290489376000: board.cache_hierarchy.ruby_system.l1_controllers1.L1Dcache: No tag match for address: 0x101268940
632 290489376000: board.cache_hierarchy.ruby_system.l1_controllers1: MESI_Two_Level-L1cache.sm:843: [ 0x85 0x47 0xaf 0x27 0xf5
633 290489376000: board.cache_hierarchy.ruby_system.l1_controllers1.sequencer: Cache hit at [0x101268948, line 0x101268940]
634 290489376000: board.cache_hierarchy.ruby_system.l1_controllers1.L1Dcache: No tag match for address: 0x101268940
635 290489377000: board.processor.switch1.core.mmu.itb: lookup(vpn=0x1a948, asid=0): hit ppn 0x101268
636 290489377000: board.processor.switch1.core.mmu.itb: translate(vpn=0x1a948, asid=0): 0x101268948 ← li
637 290489378000: board.cache_hierarchy.ruby_system.l1_controllers1.L1Dcache: address: 0x10107d880 found
638 290489378000: board.cache_hierarchy.ruby_system.l1_controllers1.L1Icache: No tag match for address: 0x10107d880
639 290489378000: board.cache_hierarchy.ruby_system.l1_controllers1.L1Dcache: address: 0x10107d880 found
```



Il core switch0 (a riga 670) ritorna dall'atomic swap con un risultato diverso da 0 dopo aver fatto una miss sulla variabile di spinlock poichè la entry dello spinlock (presente nella cache L1 del core) è stata invalidata dalla prima atomic swap.


```
668 290489384000: board.cache_hierarchy.ruby_system.l1_controllers0.L1Icache: No tag match for address: 0x10107d880
669 290489384000: board.cache_hierarchy.ruby_system.l1_controllers0: MESI_Two_Level-L1cache.sm:869: [ 0x1 0x0 0x0 0x0 0x0 0x0
670 290489384000: global: Testing Lock for addr: 0x10107d880 cur -1 con 0
671 290489384000: board.cache_hierarchy.ruby_system.l1_controllers0.sequencer: Cache miss at [0x10107d8b0, line 0x10107d880]
672 290489384000: board.cache_hierarchy.ruby_system.l1_controllers0: MESI_Two_Level-L1cache.sm:826: 0x10107d880
673 290489384000: board.cache_hierarchy.ruby_system.l1_controllers0.L1Dcache: address: 0x10107d880 found
674 290489384000: board.cache_hierarchy.ruby_system.l1_controllers0.L1Icache: No tag match for address: 0x10107d880
675 290489384000: board.cache_hierarchy.ruby_system.l1_controllers1.L1Dcache: address: 0x10107d880 found
676 290489384000: board.cache_hierarchy.ruby_system.l1_controllers1.L1Icache: No tag match for address: 0x10107d880
677 290489384000: board.cache_hierarchy.ruby_system.l1_controllers1: MESI_Two_Level-L1cache.sm:552: address: 0x10107d880, dest
678 290489384000: board.cache_hierarchy.ruby_system.l1_controllers1.L1Dcache: address: 0x10107d880 found
679 290489384000: board.cache_hierarchy.ruby_system.l1_controllers1.L1Icache: No tag match for address: 0x10107d880
```


(a riga 701) si vede che il core switch0 che ha eseguito la jump iniziare a ciclare sulla variabile di spin lock prima della load reserve, (a riga 707) switch0 carica lo spin lock per ciclarci sopra, che questa volta fa hit

```
686 290489387000: board.processor.switch0.core.mmu.itb: lookup(vpn=0x1a948, asid=0): hit ppn 0x101268 bnez
687 290489387000: board.processor.switch0.core.mmu.itb: translate(vpn=0x1a948, asid=0): 0x101268948
688 290489388000: board.cache_hierarchy.ruby_system.l1_controllers0.L1Dcache: No tag match for address: 0x101268940
689 290489388000: board.cache_hierarchy.ruby_system.l1_controllers0.MESI_Two_Level-L1cache.sm:843: [ 0x85 0x47 0xaf 0x27 0xf5 0xc0
690 290489388000: board.cache_hierarchy.ruby_system.l1_controllers0.sequencer: Cache hit at [0x101268948, line 0x101268940]
691 290489388000: board.cache_hierarchy.ruby_system.l1_controllers0.L1Dcache: No tag match for address: 0x101268940
692 290489389000: board.processor.switch0.core.mmu.itb: lookup(vpn=0x1a94c, asid=0): hit ppn 0x101268 li
693 290489389000: board.processor.switch0.core.mmu.itb: translate(vpn=0x1a94c, asid=0): 0x10126894c
694 290489389000: board.cache_hierarchy.ruby_system.l2_controllers0.MESI_Two_Level-L2cache.sm:309: Addr: 0x10107d880 State: MT_M
695 290489389000: board.cache_hierarchy.ruby_system.l2_controllers0.MESI_Two_Level-L2cache.sm:190: machineID: L2Cache-0, request
696 290489390000: board.cache_hierarchy.ruby_system.l2_controllers0.MESI_Two_Level-L2cache.sm:373: Addr: 0x10107d880 State: MT_R
697 290489390000: board.cache_hierarchy.ruby_system.l1_controllers0.L1Dcache: No tag match for address: 0x101268940
698 290489390000: board.cache_hierarchy.ruby_system.l1_controllers0.MESI_Two_Level-L1cache.sm:843: [ 0x85 0x47 0xaf 0x27 0xf5 0xc0
699 290489390000: board.cache_hierarchy.ruby_system.l1_controllers0.sequencer: Cache hit at [0x10126894c, line 0x101268940]
700 290489390000: board.cache_hierarchy.ruby_system.l1_controllers0.L1Dcache: No tag match for address: 0x101268940
701 290489391000: board.processor.switch0.core.mmu.itb: lookup(vpn=0x1a950, asid=0): hit ppn 0x101268 lw
702 290489391000: board.processor.switch0.core.mmu.itb: translate(vpn=0x1a950, asid=0): 0x101268950
703 290489392000: board.cache_hierarchy.ruby_system.l1_controllers0.L1Dcache: No tag match for address: 0x101268940
704 290489392000: board.cache_hierarchy.ruby_system.l1_controllers0.MESI_Two_Level-L1cache.sm:843: [ 0x85 0x47 0xaf 0x27 0xf5 0xc0
705 290489394000: board.cache_hierarchy.ruby_system.l1_controllers0.sequencer: Cache hit at [0x101268950, line 0x101268940]
706 290489392000: board.cache_hierarchy.ruby_system.l1_controllers0.L1Dcache: No tag match for address: 0x101268940
707 290489393000: board.processor.switch0.core.mmu.dtb: lookup(vpn=0x8a8b0, asid=0): hit ppn 0x10107d
708 290489393000: board.processor.switch0.core.mmu.dtb: translate(vpn=0x8a8b0, asid=0): 0x10107d8b0
709 290489394000: board.cache_hierarchy.ruby_system.l1_controllers0.L1Dcache: address: 0x10107d880 found
710 290489394000: board.cache_hierarchy.ruby_system.l1_controllers0.L1Icache: No tag match for address: 0x10107d880
711 290489394000: board.cache_hierarchy.ruby_system.l1_controllers0.MESI_Two_Level-L1cache.sm:835: [ 0x1 0x0 0x0 0x0 0x0 0x0
712 290489394000: board.cache_hierarchy.ruby_system.l1_controllers0.sequencer: Cache hit at [0x10107d8b0, line 0x10107d880]
713 290489394000: board.cache_hierarchy.ruby_system.l1_controllers0.L1Dcache: address: 0x10107d880 found
714 290489394000: board.cache_hierarchy.ruby_system.l1_controllers0.L1Icache: No tag match for address: 0x10107d880
715 290489395000: board.processor.switch0.core.mmu.itb: lookup(vpn=0x1a950, asid=0): hit ppn 0x101268
716 290489395000: board.processor.switch0.core.mmu.itb: translate(vpn=0x1a950, asid=0): 0x101268950 bnez
717 290489395000: board.cache_hierarchy.ruby_system.l1_controllers0.L1Dcache: address: 0x10107d880 found
```

(a riga 2015) inizia la fase di unlock del core switch1, che recupera la variabile sp_lock facendo anche lui hit (entrambi i core fanno hit poiché la variabile spinlock è usata da entrambi in sola lettura) e (a riga 2055) lo switch1 libera con un'altra atomic-swap lo spin lock.

```
2014 290489632000: board.cache_hierarchy.ruby_system.l1_controllers0.L1Dcache: No tag match for address: 0x101268940
2015 290489633000: board.processor.switch1.core.mmu.itb: lookup(vpn=0x105e4, asid=0): hit ppn 0x10125e4 jal->spin_unlock
2016 290489633000: board.processor.switch1.core.mmu.itb: translate(vpn=0x105e4, asid=0): 0x10125e5e4
2017 290489633000: board.processor.switch0.core.mmu.dtb: lookup(vpn=0x8a8b0, asid=0): hit ppn 0x10107d
2018 290489633000: board.processor.switch0.core.mmu.dtb: translate(vpn=0x8a8b0, asid=0): 0x10107d8b0
2019 290489634000: board.cache_hierarchy.ruby_system.l1_controllers0.L1Dcache: address: 0x10107d880 found
2020 290489634000: board.cache_hierarchy.ruby_system.l1_controllers0.L1Dcache: No tag match for address: 0x10107d880
2021 290489634000: board.cache_hierarchy.ruby_system.l1_controllers0: MESI_Two_Level-L1cache.sm:835: [ 0x1 0x0 0x0 0x0 0x0 0x0 0
2022 290489634000: board.cache_hierarchy.ruby_system.l1_controllers0.sequencer: Cache hit at [0x10107d8b0, line 0x10107d880]
2023 290489634000: board.cache_hierarchy.ruby_system.l1_controllers0.L1Dcache: address: 0x10107d880 found
2024 290489634000: board.cache_hierarchy.ruby_system.l1_controllers0.L1Dcache: No tag match for address: 0x10107d880
2025 290489634000: board.cache_hierarchy.ruby_system.l1_controllers1.L1Dcache: No tag match for address: 0x10125e5c0
2026 290489634000: board.cache_hierarchy.ruby_system.l1_controllers1: MESI_Two_Level-L1cache.sm:843: [ 0x85 0x27 0x23 0x20 0xf4
2027 290489634000: board.cache_hierarchy.ruby_system.l1_controllers1.sequencer: Cache hit at [0x10125e5e4, line 0x10125e5c0]
2028 290489634000: board.cache_hierarchy.ruby_system.l1_controllers1.L1Dcache: No tag match for address: 0x10125e5c0
2029 290489635000: board.processor.switch0.core.mmu.itb: lookup(vpn=0x1a950, asid=0): hit ppn 0x101268
2030 290489635000: board.processor.switch0.core.mmu.itb: translate(vpn=0x1a950, asid=0): 0x101268950
2031 290489635000: board.processor.switch1.core.mmu.itb: lookup(vpn=0x1a968, asid=0): hit ppn 0x101268
2032 290489635000: board.processor.switch1.core.mmu.itb: translate(vpn=0x1a968, asid=0): 0x101268968 fence
2033 290489636000: board.cache_hierarchy.ruby_system.l1_controllers1.L1Dcache: No tag match for address: 0x101268940
2034 290489636000: board.cache_hierarchy.ruby_system.l1_controllers1: MESI_Two_Level-L1cache.sm:843: [ 0x85 0x47 0xaf 0x27 0xf5
2035 290489636000: board.cache_hierarchy.ruby_system.l1_controllers1.sequencer: Cache hit at [0x101268968, line 0x101268940]
2036 290489636000: board.cache_hierarchy.ruby_system.l1_controllers1.L1Dcache: No tag match for address: 0x101268940
2037 290489636000: board.cache_hierarchy.ruby_system.l1_controllers0.L1Dcache: No tag match for address: 0x101268940
2038 290489636000: board.cache_hierarchy.ruby_system.l1_controllers0: MESI_Two_Level-L1cache.sm:843: [ 0x85 0x47 0xaf 0x27 0xf5
2039 290489636000: board.cache_hierarchy.ruby_system.l1_controllers0.sequencer: Cache hit at [0x101268950, line 0x101268940]
2040 290489636000: board.cache_hierarchy.ruby_system.l1_controllers0.L1Dcache: No tag match for address: 0x101268940
2041 290489637000: board.processor.switch1.core.mmu.itb: lookup(vpn=0x1a96c, asid=0): hit ppn 0x101268
2042 290489637000: board.processor.switch1.core.mmu.itb: translate(vpn=0x1a96c, asid=0): 0x10126896c amoswap
2043 290489637000: board.processor.switch0.core.mmu.itb: lookup(vpn=0x1a950, asid=0): hit ppn 0x101268
2044 290489637000: board.processor.switch0.core.mmu.itb: translate(vpn=0x1a950, asid=0): 0x101268950
2045 290489638000: board.cache_hierarchy.ruby_system.l1_controllers0.L1Dcache: No tag match for address: 0x101268940
2046 290489638000: board.cache_hierarchy.ruby_system.l1_controllers0: MESI_Two_Level-L1cache.sm:843: [ 0x85 0x47 0xaf 0x27 0xf5
2047 290489638000: board.cache_hierarchy.ruby_system.l1_controllers0.sequencer: Cache hit at [0x101268950, line 0x101268940]
2048 290489638000: board.cache_hierarchy.ruby_system.l1_controllers0.L1Dcache: No tag match for address: 0x101268940
2049 290489638000: board.cache_hierarchy.ruby_system.l1_controllers1.L1Dcache: No tag match for address: 0x101268940
2050 290489638000: board.cache_hierarchy.ruby_system.l1_controllers1: MESI_Two_Level-L1cache.sm:843: [ 0x85 0x47 0xaf 0x27 0xf5
2051 290489638000: board.cache_hierarchy.ruby_system.l1_controllers1.sequencer: Cache hit at [0x10126896c, line 0x101268940]
2052 290489638000: board.cache_hierarchy.ruby_system.l1_controllers1.L1Dcache: No tag match for address: 0x101268940
2053 290489639000: board.processor.switch0.core.mmu.dtb: lookup(vpn=0x8a8b0, asid=0): hit ppn 0x10107d
2054 290489639000: board.processor.switch0.core.mmu.dtb: translate(vpn=0x8a8b0, asid=0): 0x10107d8b0
2055 290489639000: board.processor.switch1.core.mmu.dtb: lookup(vpn=0x8a8b0, asid=0): hit ppn 0x10107d
2056 290489639000: board.processor.switch1.core.mmu.dtb: translate(vpn=0x8a8b0, asid=0): 0x10107d8b0
2057 290489640000: board.cache_hierarchy.ruby_system.l1_controllers1.L1Dcache: address: 0x10107d880 found
```

(a riga 2121) Quando si conclude l'unlock, il core 1 fa miss sull'indirizzo della variabile spinlock (poichè la modifica fatta dall'atomic-swap invalida tutti i blocchi della cache)

```
2118 290489657000: board.cache_hierarchy.ruby_system.l1_controllers1: MESI_Two_Level-L1cache.sm:826: 0x10107d880
2119 290489657000: board.cache_hierarchy.ruby_system.l1_controllers1: MESI_Two_Level-L1cache.sm:869: [ 0x1 0x0 0x0 0x0 0x0 0x0 0x0 0
2120 290489657000: global: Testing Lock for addr: 0x10107d880 cur -1 con 1
2121 290489657000: board.cache_hierarchy.ruby_system.l1_controllers1.sequencer: Cache miss at [0x10107d8b0, line 0x10107d880]
2122 290489657000: board.cache_hierarchy.ruby_system.l1_controllers1.L1Dcache: address: 0x10107d880 Round
2123 290489657000: board.cache_hierarchy.ruby_system.l1_controllers1.L1Icache: No tag match for address: 0x10107d880
2124 290489658000: board.processor.switch1.core.mmu.itb: lookup(vpn=0x1a970, asid=0): hit ppn 0x101268
2125 290489658000: board.processor.switch1.core.mmu.itb: translate(vpn=0x1a970, asid=0): 0x101268970
2126 290489658000: board.cache_hierarchy.ruby_system.l2_controllers0: MESI_Two_Level-L2cache.sm:373: Addr: 0x10107d880 State: SS_MB
2127 290489659000: board.cache_hierarchy.ruby_system.l1_controllers1.L1Dcache: No tag match for address: 0x101268940
2128 290489659000: board.cache_hierarchy.ruby_system.l1_controllers1: MESI_Two_Level-L1cache.sm:843: [ 0x85 0x47 0xaf 0x27 0xf5 0xc
2129 290489659000: board.cache_hierarchy.ruby_system.l1_controllers1.sequencer: Cache hit at [0x101268970, line 0x101268940]
2130 290489659000: board.cache_hierarchy.ruby_system.l1_controllers1.L1Dcache: No tag match for address: 0x101268940
2131 290489659000: board.processor.switch1.core.mmu.itb: lookup(vpn=0x1a970, asid=0): hit ppn 0x101268
```



(a riga 2208) il core 0 esegue la load reserve

```
2206 290489676000: board.processor.switch1.core.mmu.itb: lookup(vpn=0x105f4, asid=0): hit ppn 0x10125e
2207 290489676000: board.processor.switch1.core.mmu.itb: translate(vpn=0x105f4, asid=0): 0x10125e5f4
2208 290489676000: board.processor.switch0.core.mmu.itb: lookup(vpn=0x1a954, asid=0): hit ppn 0x101268
2209 290489676000: board.processor.switch0.core.mmu.itb: translate(vpn=0x1a954, asid=0): 0x101268954
2210 290489677000: board.cache_hierarchy.ruby_system.l1_controllers0.L1Dcache: No tag match for address: 0x101268940
2211 290489677000: board.cache_hierarchy.ruby_system.l1_controllers0.MESI_Two_Level-L1cache.sm:843: [ 0x85 0x47 0xaf 0x27 0xf5 0xc
2212 290489677000: board.cache_hierarchy.ruby_system.l1_controllers0.sequencer: Cache hit at [0x101268954, line 0x101268940]
2213 290489677000: board.cache_hierarchy.ruby_system.l1_controllers0.L1Dcache: No tag match for address: 0x101268940
2214 290489677000: board.cache_hierarchy.ruby_system.l1_controllers1.L1Dcache: No tag match for address: 0x10125e5c0
2215 290489677000: board.cache_hierarchy.ruby_system.l1_controllers1.MESI_Two_Level-L1cache.sm:843: [ 0x85 0x27 0x23 0x20 0xf4 0xfe
2216 290489677000: board.cache_hierarchy.ruby_system.l1_controllers1.sequencer: Cache hit at [0x10125e5f4, line 0x10125e5c0]
2217 290489677000: board.cache_hierarchy.ruby_system.l1_controllers1.L1Dcache: No tag match for address: 0x10125e5c0
2218 290489678000: board.processor.switch0.core.mmu.dtb: lookup(vpn=0x8a8b0, asid=0): hit ppn 0x10107d
2219 290489678000: board.processor.switch0.core.mmu.dtb: translate(vpn=0x8a8b0, asid=0): 0x10107d8b0
2220 290489678000: board.processor.switch1.core.mmu.dtb: lookup(vpn=0x3fd4660c14, asid=0): hit ppn 0x2781c4
2221 290489678000: board.processor.switch1.core.mmu.dtb: translate(vpn=0x3fd4660c14, asid=0): 0x2781c4c14
2222 290489678000: board.cache_hierarchy.ruby_system.l2_controllers0: MESI_Two_Level-L2cache.sm:309: Addr: 0x10107d880 State: MT_SB
2223 290489678000: board.cache_hierarchy.ruby_system.l2_controllers0: MESI_Two_Level-L2cache.sm:190: machineID: L2Cache-0, requestor
```



(a riga 2233) grazie al messaggio:

```
2232 290489679000: board.cache_hierarchy.ruby_system.l1_controllers0: MESI_Two_Level-L1cache.sm:835: [ 0x1 0x0 0x0 0x0 0x0 0x0 0x0 0
2233 290489679000: global: Setting Lock for addr: 0x10107d880 to 0
2234 290489679000: board.cache_hierarchy.ruby_system.l1_controllers0.sequencer: Cache hit at [0x10107d8b0, line 0x10107d880]
2235 290489679000: board.cache_hierarchy.ruby_system.l1_controllers0.L1Dcache: address: 0x10107d880 found
```


la load reserve setta il lock globale con un valore univoco corrispondente al suo contesto e (a riga 2343-44) con i messaggi:

```
2341 290489704000: board.cache_hierarchy.ruby_system.l1_controllers0: MESI_Two_Level-L1cache.sm:826: 0x10107d880
2342 290489704000: board.cache_hierarchy.ruby_system.l1_controllers0: MESI_Two_Level-L1cache.sm:869: [ 0x1 0x0 0x0 0x0 0x0 0x0 0
2343 290489704000: global: Testing Lock for addr: 0x10107d880 cur 0 con 0
2344 290489704000: global: Clear Lock for addr: 0x10107d880
2345 290489704000: board.cache_hierarchy.ruby_system.l1_controllers0.sequencer: Cache miss at [0x10107d8b0, line 0x10107d880]
2346 290489704000: board.cache_hierarchy.ruby_system.l1_controllers0.L1Dcache: address: 0x10107d880 found
2347 290489704000: board.cache_hierarchy.ruby_system.l1_controllers0.L1Icache: No tag match for address: 0x10107d880
2348 290489705000: board.processor.switch0.core.mmu.itb: lookup(vpn=0x1a95c, asid=0): hit ppn 0x101268
2349 290489705000: board.processor.switch0.core.mmu.itb: translate(vpn=0x1a95c, asid=0): 0x10126895c
```

la store conditional si conclude correttamente e avviene una miss (poichè il blocco di memoria viene invalidato in tutte le cache



Compare exchange



Per quanto riguarda l'atomic_compare_exchange() prima di tutto si cerca di fare una load reserve sulla variabile globale, successivamente si fa la store del valore desiderato, se la reserve presa precedentemente fosse ancora valida.

Tuttavia l'output del test è uguale alla seconda parte del test con spin lock.

```
105a8: 1e042783      lw a5,-32(s0)
105ac: 9fb9          addw a5,a5,a4
105ae: 2781          sext.w a5,a5
105b0: fcf42423      sw a5,-56(s0)
105b4: fc842783      lw a5,-56(s0)
105b8: 0007859b      sext.w a1,a5
105bc: fd043603      ld a2,-48(s0)
105c0: fcc40793      addi a5,s0,-52
105c4: 4398          lw a4,0(a5)
105c6: 86ba          mv a3,a4
105c8: 1006272f      lr.w a4,(a2) CERCO DI FARE RESERVE
105cc: 00d71563      bne a4,a3,105d6 <func+0xc2>
105d0: 18b6252f      sc.w a0,a1,(a2) STORE SE RESERVE VALIDA
105d4: f975          bnez a0,105c8 <func+0xb4>
105d6: 40d706bb      subw a3,a4,a3
105da: 2681          sext.w a3,a3
105dc: 0016b613      seqz a2,a3
105e0: 0006069b      sext.w a3,a2
105e4: e291          bnez a3,105e8 <func+0xd4>
105e6: c398          sw a4,0(a5)
```


Transizioni della macchina a stati che modella la cache di secondo livello:


Stati	Richieste	Core
MT	(getx)	switch0
MT_MB	(exclusive unblock)	switch0
MT	(gets)	switch1
poi:		
MT_SB	(unblock)	switch1
SS	(upgrade)	switch1
SS_MB	(gets)	switch0
SS_MB	(exclusive unblock)	switch1
MT	(gets)	switch0
MT_SB	(unblock)	switch0
SS	(upgrade)	switch1

Il core 0 si trova ancora nel while precedente al ciclo for (di nostro interesse), riesce a prendere un lock in scrittura (getx) poiché il core 1 non ne possiede né uno in scrittura né in lettura. Successivamente libera il lock e il core 1 cerca di prendere (con spin-lock) in lettura.

<----- LOOP ----->

<----- FINE LOOP ----->

A questo punto il core 1, fa l'upgrade del lock, da lettura a scrittura (per l'atomic-swap) e inizia il ciclo dello spin lock per il core 0



I loop termina a riga 2300 con il nuovo stato
SS (upgrade) switch0

e poi si torna nel loop, con un transitorio:

SS_MB (getx) switch1
SS_MB (exclusive unblock) switch0
MT (getx) switch1

MT_MB (gets) switch0
MT_MB (exclusive unblock) switch1
MT (gets) switch0
MT_SB (unblock) switch0
SS (upgrade) switch0

SS_MB (gets) switch1 <----- LOOP ----->
SS_MB (exclusive unblock) switch0
MT (gets) switch1
MT_SB (unblock) switch1
SS (upgrade) switch0 <----- FINE LOOP ----->

Descrizione degli stati

States	Invariants and Semantic/Purpose of the state
NP	The cache blocks is not present in the on-chip cache hierarchy.
SS	The cache block is present in potentially multiple private caches in only readable mode (i.e.in "S" state in private caches). Corresponding "Sharers" vector with the block should give the identity of the private caches which possibly have the cache block in its cache. The cache block in the L2 cache is valid and readable .
M	The cache block is present ONLY in the L2 cache and has exclusive permission. L1 Cache's read/write requests (GETS/GETX) can be satisfied directly from the L2 cache.
MT	The cache block is in ONE of the private L1 caches with exclusive permission. The data in the L2 cache is potentially stale. The identity of the L1 cache which has the block can be found in the "Owner" field associated with the cache block. Any request for read/write (GETS/GETX) from other cores/private L1 caches need to be forwarded to the owner of the cache block. L2 can not service requests itself.
M_I	Its a transient state. This state indicates that the cache is trying to replace the cache block from its cache and the write-back (PUTX/PUTS) to the Directory controller (which act as interface to Main memory) has been issued but awaiting write-back acknowledgement. The data is neither readable nor writable.
MT_I	Its a transient state. This state indicates that the cache is trying to replace a cache block in MT state from its cache. Invalidation to the current owner (private L1 cache) of the cache block has been issued and awaiting write-back from the Owner L1 cache. Note that the this Invalidation (called back-invalidation) is instrumental in making sure that the inclusion is maintained between L1 and L2 caches. The data is neither readable nor writable.
MCT_I	Its a transient state.This state is same as MT_I , except that it is known that the data in the L2 cache is in <i>clean</i> state. The data is neither readable nor writable.
I_I	Its a transient state. The L2 cache is trying to replace a cache block in the SS state and the cache block in the L2 is in <i>clean</i> state. Invalidations has been sent to all potential sharers (L1 caches) of the cache block. The L2 cache's directory is waiting for all the required Acknowledgements to arrive from the L1 caches. Note that the this Invalidation (called back-invalidation) is instrumental in making sure that the inclusion is maintained between L1 and L2 caches. The data is neither readable nor writable.

S_I	Its a transient state.Same as I_I , except the data in L2 cache for the cache block is <i>dirty</i> . This means unlike in the case of I_I , the data needs to be sent to the Main memory. The cache block is neither readable nor writable..
ISS	Its a transient state. L2 has received a GETS (read) request from one of the private L1 caches, for a cache block that it not present in the on-chip caches. A read request has been sent to the Main Memory (Directory controller) and waiting for the response from the memory. This state is reached only when the request is for data cache block (not instruction cache block). The purpose of this state is that if it is found that only one L1 cache has requested the cache block then the block is returned to the requester with exclusive permission (although it was requested for reading permission). The cache block is neither readable nor writable.
IS	Its a transient state. The state is similar to ISS , except the fact that if the requested cache block is Instruction cache block or more than one core request the same cache block while waiting for the response from the memory, this state is reached instead of ISS . Once the requested cache block arrives from the Main Memory, the block is sent to the requester(s) with read-only permission. The cache block is neither readable nor writable at this state.
IM	Its a transient state. This state is reached when a L1 GETX (write) request is received by the L2 cache for a cache blocks that is not present in the on-chip cache hierarchy. The request for the cache block in exclusive mode has been issued to the main memory but response is yet to arrive.The cache block is neither readable nor writable at this state.
SS_MB	Its a transient state. In general any state whose name ends with "B" (like this one) also means that it is a <i>blocking</i> coherence state. This means the directory awaiting for some response from the private L1 cache ans until it receives the desired response any other request is not entertained (i.e. request are effectively serialized). This particular state is reached when a L1 cache requests a cache block with exclusive permission (i.e. GETX or UPGRADE) and the coherence state of the cache blocks was in SS state. This means that the requested cache blocks potentially has readable copies in the private L1 caches. Thus before giving the exclusive permission to the requester, all the readable copies in the L1 caches need to be invalidated. This state indicate that the required invalidations has been sent to the potential sharers (L1 caches) and the requester has been informed about the required number of Invalidation Acknowledgement it needs before it can have the exclusive permission for the cache block. Once the requester L1 cache gets the required number of Invalidation Acknowledgement it informs the director about this by UNBLOCK message which allows the directory to move out of this blocking coherence state and thereafter it can resume entertaining other request for the given cache block. The cache block is neither readable nor writable at this state.

MT_MB	Its a transient state and also a <i>blocking</i> state. This state is reached when L2 cache's directory has sent out a cache block with exclusive permission to a requester L1 cache but yet to receive <i>UNBLOCK</i> from the requester L1 cache acknowledging the receipt of exclusive permission. The cache block is neither readable nor writable at this state.
MT_IIB	Its a transient state and also a <i>blocking</i> state. This state is reached when a read request (GETS) request is received for a cache blocks which is currently held with exclusive permission in another private L1 cache (i.e. directory state is MT). On such requests the L2 cache's directory forwards the request to the current owner L1 cache and transitions to this state. Two events need to happen before this cache block can be unblocked (and thus start entertaining further request for this cache block). The current owner cache block need to send a write-back to the L2 cache to update the L2's copy with latest value. The requester L1 cache also needs to send <i>UNBLOCK</i> to the L2 cache indicating that it has got the requested cache block with desired coherence permissions. The cache block is neither readable nor writable at this state in the L2 cache.
MT_IB	Its a transient state and also a <i>blocking</i> state. This state is reached when at MT_IIB state the L2 cache controller receives the <i>UNBLOCK</i> from the requester L1 cache but yet to receive the write-back from the previous owner L1 cache of the block. The cache block is neither readable nor writable at this state in the L2 cache.
MT_SB	Its a transient state and also a <i>blocking</i> state. This state is reached when at MT_IIB state the L2 cache controller receives write-back from the previous owner L1 cache for the blocks, while yet to receive the <i>UNBLOCK</i> from the current requester for the cache block. The cache block is neither readable nor writable at this state in the L2 cache.