X86-TSO

Sequential consistency (SC)

Shared memory

Atomic actions:
- Mem read(a) \rightarrow v
- Mem write(a,v) \rightarrow v

- SC used in real HW.

★ - Clean abstraction for devs.
Example

**Results:**

\[
\begin{align*}
&\text{OK in sc.} \\
&\begin{cases}
(1, 1) \\
(0, 1) \\
(1, 0) \\
&\times (0, 0)
\end{cases}
\end{align*}
\]

main:

\[
\begin{align*}
x &= 0 \\
y &= 0
\end{align*}
\]

Spawn T1, T2
wait
print \((x', y')\).
Real HW for shared mem.

CPU/socket

Core

L3 cache

Shared DRAM

Out-of-order exec

store(x, 1)

t := fetch(y)

store(y', t)

Speculative exec

T1

*p = x

p = shared

shared = p

print(*p)

if (a) {

... <=

3

else {

...

3

}
Indep reads of indep writes (IRIW)

$\begin{align*}
T1 & \quad x &= 1 \rightarrow SB \\
T2 & \quad y &= 1 \rightarrow SB \\
T3 & \quad x3 &= x \leftarrow 1 \text{ from } SB \\
& \quad y3 &= y \leftarrow 0 \in \text{mem} \\
T4 & \quad y4 &= y \leftarrow 1 \in SB \\
& \quad x4 &= x \leftarrow 0 \in \text{mem}
\end{align*}$

| 0,0 | 0,0 |
| 1,1 | 1,1 |
| 1,0 | 1,0 |
| 0,1 | 0,1 |

How to program on weak mem?

Simulate SC: locks, etc.

General: lock init?

lock-free: Linux RCU

x86-TSO.

1,0 \quad ? \quad 0,1

not on x86 (not on x86-TSO).
Intel/AMD specs

Principles: “causality”
Litmus tests.

Incomplete
Imprecise (prose)
"Axiomatic": predicates on traces.

Locked?
release in mem or later?

On entry the address of spinlock is in register EAX
and the spinlock is unlocked iff its value is 1

acquire: LOCK, DEC [EAX] ; LOCK'd decrement of [EAX]
         JNS enter ; branch if [EAX] was ≥ 1
         spin: CMP [EAX], 0 ; test [EAX]
         JLE spin ; branch if [EAX] was ≤ 0
         JMP acquire ; try again

enter: ; the critical section starts here

release: MOV [EAX] → 1