Memory, an Elusive Abstraction

Peter Sewell
University of Cambridge
http://www.cl.cam.ac.uk/users/pes20/weakmemory

Abstract

Multiprocessors are now ubiquitous. They provide an abstraction of shared memory, accessible by concurrently executing threads, which supports a wide range of software. However, exactly what this key abstraction is —what the hardware designers implement, and what programmers can depend on— is surprisingly elusive. In 1979, when articulating the notion of sequential consistency (SC), Lamport wrote "For some applications, achieving sequential consistency may not be worth the price of slowing down the processors." [7], and indeed most major multiprocessor families, including Alpha, ARM, Itanium, Power, Sparc, and x86, do not provide the abstraction of SC memory. Internally, they incorporate a range of sophisticated optimisations which have various programmer-visible effects. For some (such as Sparc) these effects are captured in a well-defined relaxed memory model, making it possible (if challenging) to reason with confidence about the behaviour of concurrent programs. For others, however, it has been very unclear what a reasonable model is, despite extensive research over the last three decades. In this talk, I will reflect on the experience of my colleagues and I in trying to establish usable models for x86 multiprocessors, where it appears that our x86-TSO model suffices for common-case code [1–4], and for Power and ARM multiprocessors, where we have models that capture some but not all aspects of their behaviour [5, 6]. The underlying causes of these difficulties are complex, including:

- The programmer-observable relaxed-memory behaviour of a multiprocessor is a whole-system property that arises from the interaction between many complex aspects of the processor implementation: speculative execution, store buffering, cache protocol, and so forth.
- Programs are executed (and tested) on specific multiprocessor implementations, but processor vendors attempt to document loose specifications to cover a range of possible (past and future) implementations.
- Multiprocessor implementation details are typically confidential and may change radically from one implementation to another.
- Vendor specifications suffer from the tension between the need for loose specification, to preserve freedom for such changes, and the need for tight specification, to give strong properties to properties.
- All too often, loose specification has been achieved by vague specification, using informal prose. When it comes to subtle concurrent properties this is almost inevitably ambiguous; it also makes it impossible (even in principle) to test conformance between a processor implementation and such a specification, let alone to verify such a correspondence or to reason about concurrent programs.

Categories and Subject Descriptors C.1.2 [*Multiple Data Stream Architectures (Multiprocessors)*]: Parallel processors; D.1.3 [*Concurrent Programming*]: Parallel programming; F.3.1 [*Specifying and Verifying and Reasoning about Programs*]

General Terms Documentation, Reliability, Standardization, Theory, Verification

Keywords Relaxed Memory Models, Semantics

Acknowledgements This is based on joint work with colleagues in Cambridge and INRIA Rocquencourt. I acknowledge funding from EPSRC grants EP/F036345 and EP/H005633.

References

[1] S. Sarkar, P. Sewell, F. Zappa Nardelli, S. Owens, T. Ridge, T. Braibant, M. Myreen, and J. Alglave. The semantics of x86-CC multiprocessor machine code. In *Proc. POPL 2009*, January 2009.

[2] Scott Owens, Susmit Sarkar, and Peter Sewell. A better x86 memory model: x86-TSO. In *Proc. TPHOLs, LNCS 5674*, pages 391–407, 2009.

[3] Scott Owens. Reasoning about the implementation of concurrency abstractions on x86-TSO. In *Proc. ECOOP*, 2010. To appear.

[4] Peter Sewell, Susmit Sarkar, Scott Owens, Francesco Zappa Nardelli, and Magnus Myreen. x86-TSO: A rigorous and usable programmer's model for x86 multiprocessors. *Communications of the ACM*, 2010. To appear.

[5] J. Alglave, A. Fox, S. Ishtiaq, M. Myreen, S. Sarkar, P. Sewell, and F. Zappa Nardelli. The semantics of Power and ARM multiprocessor machine code. In *Proc. DAMP 2009*, January 2009.

[6] Jade Alglave, Luc Maranget, Susmit Sarkar, and Peter Sewell. Fences in weak memory models. In *Proceedings of CAV*, 2010. To appear.

[7] L. Lamport. How to make a multiprocessor computer that correctly executes multiprocess programs. *IEEE Trans. Comput.*, C-28(9):690–691, 1979.

ISMM'10, June 5–6, 2010, Toronto, Ontario, Canada.
ACM 978-1-4503-0054-4/10/06.