Weak Ordering:
A New Definition
(presenting: Matthew Moskewicz)

CS258, UCB, 2002.
(2002.02.28)

Definition 1
• Definition 1 is a little interesting…
  – It appears to be a (re)formulation of the visible behaviors of a particular flavor of multiprocessor memory system that ‘defined’ the idea of weak consistency
  – It is hinted that the original ‘definition’ was simply a description of the hardware itself. But since we’re going to compare it against another concept entirely, we better reformulate.
  – Semantics: sync accesses are strongly ordered. for the rest, finish them before sync, and wait until after sync to do more.

Definition 2
*In its full glory:* “Hardware is weakly ordered with respect to a synchronization model if and only if it appears sequentially consistent to all software that obey the synchronization model.” Bam.

• Programmer wants to think about sequential consistency
  – So he can, if he obeys the synchronization model
• Hardware can do anything it wants
  – As long as it meets its contractual obligation
• But?
  – ‘What’s a synchronization model?’
  – Is this even comparable with definition 1? Does that matter?

Truth in Advertising
• This is paper is exactly what its title says it is: a new definition of ‘weak ordering’
  – “Definition 1” – (the old definition) acts as the whipping boy throughout the paper.
  – “Definition 2” – (the new definition) illustrates the key goals of the work.
  – “Definition 3” – (the example) tries to show that “Definition 2” is just as cool as it looks.
Definition 3

- Well, okay, but clearly we need a good synch model for this to fly.
- Here’s a simple one:
  - No data races (DRF0)
- Simple concept
  - If no conflicting accesses not separated by synchronization allowed.
- What does this mean to the programmer?
  - Must make synchronization explicit in a certain way
  - Not so bad overall, “most” programs want to avoid data races anyway.

Definitions 4&5?

- What does this mean for the hardware?
  - We’ll describe some hardware, pretty much like the hardware we reverse engineered “Definition 1” from.
  - But we won’t actually formulate “Definition 4” and compare that against its counterpart: “Definition 1”
  - Instead, we’ll show that DRF0 is ‘weaker’ than whatever model is it that “Definition 1” supports – “Definition 5”.
    - Which really isn’t much of a surprise
    - But then again, all we had before was Definition 1.

And?

- And now we can make up as many synchronization models as we want.
  - And hopefully, adding restrictions to the models will allow more efficient hardware
    - Without restricting the programmer too much
  - And the contract is ‘tight’ from both sides
  - And look at DRF0
    - It’s clean
    - It’s ‘almost’ as good as whatever Definition 1 supports.
    - And it allows more freedom in hardware
    - And the program can execute more efficiently

To Sum Up

- Concept of an Interface: Good.
- Previous definition of ‘weak ordering’ didn’t use an interface model: Bad.
- Programmer wants to think about sequential consistency of constrained class of programs.
- So that’s our Interface.
- And it seems to work pretty well.
  - But we couldn’t quite characterize the old definition under the new Interface
  - But we probably could
Fin