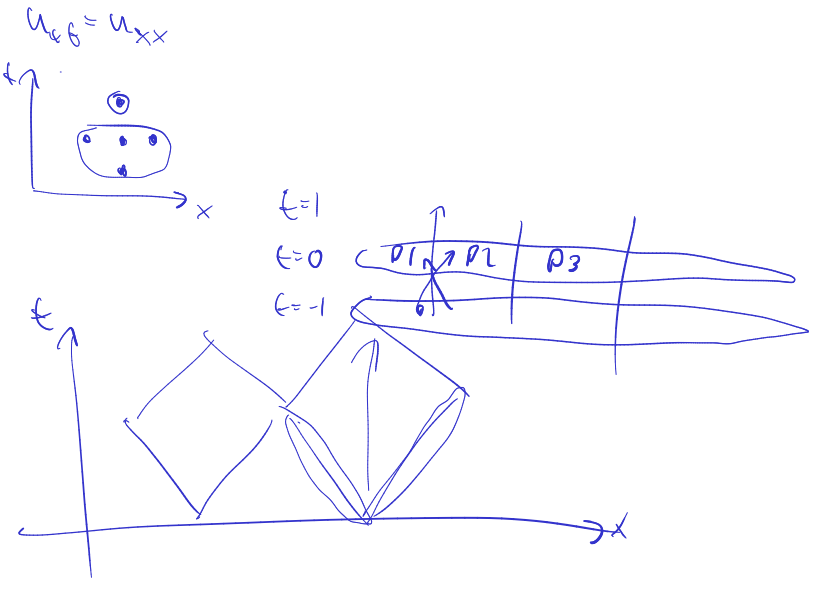
March 11, 2025

Announcements

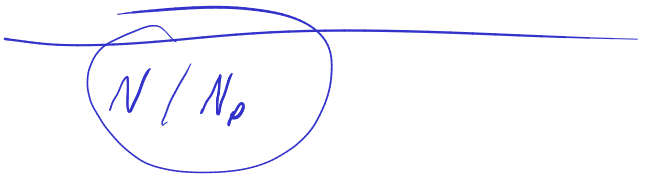
Project Proposals
MK contest.

Goals

Review


$u_{tt} = u_{xx}$
t
x
t=1
t=0
t=-1
ρ_1
ρ_2
ρ_3
t
x

$S(i,j)$:



$5 \cdot n$ $\lceil N/3 \rceil$

Memory Ordering

Why is Memory Ordering a Problem?

reordering (processor
compiler

What's the purpose of different memory orders?

"and then"

Memory ordering semantics

atomic non-atomic

consume: no r/w reorder before in this thread
data-dependent releasing writes

acquire: no r/w reorder before in this thread,
all releasing writes elsewhere

release: no r/w reorder after in this thread
all writes here are visible to acquirer.

seq_cst

Example: A Semaphore With Atomics

```
#include <stdatomic.h> // mo_->memory_order, a_->atomic
typedef struct { atomic_int v;} naive_sem_t;
void sem_down(naive_sem_t *s)
{
  while (1) {
    while (a_load_explicit(&(s->v), mo_acq___) < 1)
        spinloop_body();
    int tmp=a_fetch_add_explicit(&(s->v), -1, mo_acq___rel);
    if (tmp >= 1)
        break; // we got the lock
    else // undo our attempt
      a_fetch_add_explicit(&(s->v), 1, mo_relaxed);
  }
}
void sem_up(naive_s_t *s) {
  a_fetch_add_explicit(&(s->v), 1, mo_release);
}
```

[Cordes '16] — Hardware implementation: how?

C: What is 'order'?

C11 Committee Draft, December '10, Sec. 5.1.2.3, "Program execution":

- (3) *Sequenced before* is an asymmetric, transitive, pair-wise relation between evaluations executed by a single thread, which induces a partial order among those evaluations. Given any two evaluations A and B, if A is sequenced before B, then the execution of A shall precede the execution of B. (Conversely, if A is sequenced before B, then B is sequenced after A.) If A is not sequenced before or after B, then A and B are unsequenced. Evaluations A and B are *indeterminately sequenced* when A is sequenced either before or after B, but it is unspecified which. The presence of a sequence point between the evaluation of expressions A and B implies that every value computation and side effect associated with A is sequenced before every value computation and side effect associated with B. (A summary of the *sequence points* is given in annex C.)

Q: Where is this definition used (in the standard document)?

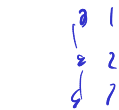
atomic ops.

C: What is 'order'? (Encore)

C11 Draft, 5.1.2.4 "Multi-threaded executions and data races":

- All modifications to a particular atomic object M occur in some particular total order, called the *modification order* of M.
- An evaluation A *carries a dependency* to an evaluation B if ...
- An evaluation A is *dependency-ordered* before an evaluation B if...
- An evaluation A *inter-thread happens before* an evaluation B if...
- An evaluation A *happens before* an evaluation B if...

Why is this so subtle?



Want to avoid ruining opt. by overly strict ordering cond.

C: How Much Lying is OK?

C11 Committee Draft, December '10, Sec. 5.1.2.3, "Program execution":

- (1) The semantic descriptions in this International Standard describe the behavior of an abstract machine in which issues of optimization are irrelevant.
- (2) Accessing a volatile object, modifying an object, modifying a file, or calling a function that does any of those operations are all *side effects*, which are changes in the state of the execution environment. [...]

C: How Much Lying is OK?

- (4) In the abstract machine, all expressions are evaluated as specified by the semantics. An actual implementation need not evaluate part of an expression if it can deduce that its value is not used and that no needed side effects are produced (including any caused by calling a function or accessing a volatile object).
- (6) The least requirements on a conforming implementation are:
  - Accesses to volatile objects are evaluated strictly according to the rules of the abstract machine.
  - At program termination, all data written into files shall be identical to the result that execution of the program according to the abstract semantics would have produced.
  - The input and output dynamics of interactive devices shall take place as specified in 7.21.3. The intent of these requirements is that unbuffered or line-buffered output appear as soon as possible, to ensure that prompting messages actually appear prior to a program waiting for input.

  This is the observable behavior of the program.

Arrays

Why are arrays the dominant data structure in high-performance code?

Any comments on C's arrays?