March 25, 2025

Announcements

- Project proposal Fri
- Pres. submission.
- Remember

Goals

- Machine check.
  - SIMD
  - Compiler go poof
  - GPUs

Review

SIMD

Name language mechanisms for SIMD:

- Intrinsics - m128
- Inline assembly
- SIMD v4f
- #pragma
- ispc / Mapping of scalar programs / what GPUs do

Demo: machabstr/Ways to SIMD

Outer-Loop/inner-Loop Vectorization

Contrast *outer-loop* vs *inner-loop vectorization*.

Outer-loop:
- control flow
- comm.

Side q: Would you consider GPUs outer- or inner-loop-vectorizing?

Alignment: How?

The old way:

```
int __attribute__ ((aligned (8))) a_int;
```

Difference between these two?

```
int __attribute__ ((aligned (8))) * ptr_t_1;
int *__attribute__ ((aligned (8))) ptr_t_2;
```

The 'new' way (C/C++11):

```
struct alignas(64) somestruct_t { /* ... */ };
struct alignas(alignof(other_t))
  somestruct_t { /* ... */ };
struct
  alignas(
    std::hardware_destructive_interference_size)
    somestruct_t { /* ... */ };
```

What is *constructive interference*?

Alignment: Why?

What is the concrete impact of the constructs on the previous slide?

Pointers and Aliasing

Demo: machabstr/Pointer Aliasing

Register Pressure

What if the register working set gets larger than the registers can hold?
What is the performance impact?

Demo: machabstr/Register Pressure

Object-Oriented Programming

Object-oriented programming: The weapon of choice for encapsulation and separation of concerns!
Performance perspective on OOP?

- AoS disaster
- gobs of memory traffic for temporaries
  - array a;
  - ((a + a) + a) + a;
  - "expression templates"
- more

Demo: machabstr/Object Orientation vs Performance

Being Nice to Your Compiler

Some rules of thumb:

- Use indices rather than pointers
- Extract common subexpressions
- Make functions static
- Use `const`
- Avoid store-to-load dependencies

What are the concrete impacts of doing these things?