Rust Utrecht

Programme

19:30	Welcome by Channable
19:35	Talk: *Optimising Claxon*
20:00	Workshop
21:30	Fin

channable

channable.com/jobs · tech.channable.com

Optimising Claxon

Claxon

Flac: Free Lossless Audio Codec.

Pure-Rust decoder for the Flac codec.

Used to be 6 times slower than reference.

Now only 13% slower.

github.com/ruuda/claxon

Perf

```
33.87%  bench_decode <claxon::input::Bitstream<R>>::read_leq_u8
27.35%  bench_decode claxon::subframe::decode_partitioned_rice
15.14%  bench_decode <claxon::input::Bitstream<R>>::read_leq_u32
 8.11%  bench_decode claxon::subframe::predict_lpc
 5.54%  bench_decode claxon::input::shift_left
 3.16%  bench_decode claxon::input::shift_right
 2.49%  libc-2.24.so __memmove_avx_unaligned_erms
 1.75%  bench_decode claxon::subframe::rice_to_signed
 0.97%  bench_decode claxon::frame::decode_mid_side
 0.76%  bench_decode bench_decode::main
 0.47%  bench_decode memcpy@plt
 0.09%  bench_decode claxon::frame::decode_right_side
```

Shift left?!

```
fn shift_left(x: u8, shift: usize) -> u8 {
    debug_assert!(shift <= 8);

    // Rust panics when shifting by the integer width,
    // so we have to treat that case separately.
    if shift >= 8 { 0 } else { x << shift }
}
```

Surprise!

Compiler bug (rust-lang/rust#37538)

Solution: `#[inline]`

Decode time went down by 49–53%.

Decoding unary

`read_leq_u8`: single bit 99.8% of the time.

Bit reader buffers one byte anyway.
`u8::leading_zeros` compiles to lzcnt.
Shaved 19–27% off of decoding time.

Rice decoding

```
fn rice_to_signed(val: i64) -> i64 {
    if val & 1 == 1 {
        - 1 - val / 2
    } else {
        val / 2
    }
}
```

Rice decoding

```
push %rbp
mov  %rsp,%rbp
mov  %rdi,%rax
shr  $0x3f,%rax // Add sign bit of %rax to itself.
add  %rdi,%rax  // Also here.
sar  %rax       // Divide %rax by two.
shl  $0x3f,%rdi // Extend bit 0 of %rdi to entire register.
sar  $0x3f,%rdi // Also here.
xor  %rax,%rdi  // Xor %rax/2 with either 0 or 0xffff...
mov  %rdi,%rax
pop  %rbp
retq
```

Zero-cost abstractions

```
buffer: &mut [i32];      // A mutable array slice.
coefficients: [i64; 12]; // A fixed-size array of 12 elements.
qlp_shift: i16;
for i in 12..buffer.len() {
    let prediction = coefficients.iter()
                                 .zip(&buffer[i - 12..i])
                                 .map(|(&c, &s)| c * s as i64)
                                 .sum::<i64>() >> qlp_shift;
    let delta = buffer[i] as i64;
    buffer[i] = (prediction + delta) as i32;
}
```

Zero-cost abstractions

```
movslq %r14d,%r11
movslq -0x2c(%r8,%rdi,4),%rsi
imul   %r10,%rsi
movslq -0x30(%r8,%rdi,4),%r14
imul   %rbp,%r14
add    %rsi,%r14
// This 12 times.
sar    %cl,%r14
add    (%r8,%rdi,4),%r14d
mov    %r14d,(%r8,%rdi,4)
inc    %rdi
cmp    %r9,%rdi
jb     10c00 <claxon::subframe::predict_lpc::...>
```

Questions?

Workshop

WiFi

Rust @ Channable

Assignment

Write a cut-like program.

- Read UTF-8 from file, print to stdout.
- May assume separator is a comma.
- May assume cutting a single column.
- No allocations allowed after startup.

```
lang,generics        lang        generics
Rust,yes             Rust        yes
C#,yes               C#          yes
Go,no                Go          no
```

Pointers

- API reference at doc.rust-lang.org/std.
- `std::env::args`
- `std::fs::File`
- `std::io::stdout`
- `std::io::Read`, `std::io::Write`
- Example at github.com/ruuda/rust-utrecht.

Thanks for attending

Want to speak or sponsor? Get in touch.

Ruud van Asseldonk
ruud@veniogames.com

Adolfo Ochagavía
aochagavia92@gmail.com