3-bit Waymarking

(a.k.a. Son of Use -Diet)

Gabor Greif
Weekend LLVM-hobbyist
The Situation Before 2008

Use has 4 pointers
- drop pointer to `User`
- allocate `Use`'s before `User` in memory
- make `Prev` pointer tagged (2-bits, since always 4-byte aligned)
- seen 12% space savings on big C++ programs
- landed in the LLVM codebase: May 2008
How it Works

Employ a framed serial code in consecutive Use s

- S → full stop
- s → stop
- 0, 1 → binary digits

Read off binary digits to obtain distance to User
Interlude

:-)
The Sacrifice

2.5% runtime increase

(but it was worth it!)
 Solution

When two feet permit just so much speed, then you have to upgrade to three feet!

I really did not mean to do something cruel as this!

But no earthly life-form provides this feature, so...
Clearly I was in need of some alien technology!

…then I took a page from the book of space exploration and found this gem:

Alien tricks from Mars! :-)
Son of Use - Diet: 3-bits Encoding

On today’s predominantly 64-bit platforms, pointers are 8-byte aligned

We have 8 distinct tags for disposal

- double digits: 00, 01, 10, 11
- 3 stop tags: q, r, s (always in this order)
- full stop: S

Originally modelled in Haskell (+ QuickCheck)

Now in LLVM repo (on a branch), with automatic algorithm selection
Benefits

- stop tags allow longer hops while hunting down the framed digits
- any stop tag encodes the distance to the framed payload
- harvesting 2-bits at a time
## Comparison

<table>
<thead>
<tr>
<th>tag-bits</th>
<th>frames</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>...1s100000s11010s10100s1111s1010s110s11s1S</td>
</tr>
<tr>
<td>accesses</td>
<td>...87CBA9876BA9876A987659876587654654343221</td>
</tr>
<tr>
<td>3</td>
<td>...rs203qrs131qrs113qrs101qrs30qrs13qrs3rsS</td>
</tr>
<tr>
<td>accesses</td>
<td>...55666555665556655555445544443332221</td>
</tr>
<tr>
<td>Δ</td>
<td>...3265443205443204332104332132210211011000</td>
</tr>
</tbody>
</table>
Further Opportunities

- unroll tag initialisation loops
- distance relative to stopped frame (microoptimization)
- `rol` (rotate) instructions with condition flags
- examining resultant assembly (on all archs!)
Questions? — or just ask me later

Credits:

- NASA (image)
- Wikipedia (image)
- W3C Slidy