Introduction to qbe
A lightweight compiler backend

Drew DeVault
SourceHut

January 16, 2022
What is qbe?

qbe is an optimizing compiler backend which provides ”70% of the performance of advanced compilers in 10% of the code.”

- Similar to LLVM in purpose
- Compiles an intermediate representation (IR) to machine code
- Supports x86_64, aarch64, and riscv64 today
- About 14,000 lines of C99 code
function w $add(w %a, w %b) {
  @start
    %c =w add %a, %b
    ret %c
  }
export function w $main() {
  @start
    %r =w call $add(w 1, w 1)
    call $printf(l $fmt, w %r, ...)
    ret 0
  }
data $fmt = { b "1 + 1 = %d!\n", b 0 }
qbe IR

.text
add:
  pushq %rbp
  movq %rsp, %rbp
  movl %edi, %eax
  addl %esi, %eax
  leave
  ret

.data
.balign 8
fmt:
  .ascii "1 + 1 = %d!\n"
  .byte 0

.text
.globl main
main:
  pushq %rbp
  movq %rsp, %rbp
  movl $1, %esi
  movl $1, %edi
  callq add
  movl %eax, %esi
  leaq fmt(%rip), %rdi
  movl $0, %eax
  callq printf
  movl $0, %eax
  leave
  ret
qbe usage

$ qbe test.ssa > test.s
$ cc -o test test.s
$ ./test
1 + 1 = 2!
https://sr.ht/~mcf/cproc/

- Self-hosting C11 compiler based on qbe
- 8,000 lines of C
- Builds GCC 4.7, binutils, util-linux, BearSSL, git, u-Boot, and much more*
- Does not have: VLAs, TLS, PIC, inline assembly

*https://github.com/oasislinux/oasis/issues/13
cproc

$ cat test.c
#include <stdio.h>

int main() {
    printf("Hello world!\n");
}
$cproc -emit-qbe -o - test.c
data $.Lstring.2 = align 1 { b "Hello world!\012\000", } export function w $main() {
@start.1
@body.2
%.1 =w call $printf(1 $.Lstring.2, ...)
ret 0
}
Performance?

“70% of the performance of advanced compilers in 10% of the code.”
Performance?

“70% of the performance of advanced compilers in 10% of the code.”

<table>
<thead>
<tr>
<th>Compiler</th>
<th>Lines of code</th>
<th>Number of files</th>
</tr>
</thead>
<tbody>
<tr>
<td>LLVM</td>
<td>10,000,000</td>
<td>87,000</td>
</tr>
<tr>
<td>GCC†</td>
<td>9,000,000</td>
<td>100,000</td>
</tr>
<tr>
<td>qbe</td>
<td>14,000</td>
<td>55</td>
</tr>
</tbody>
</table>

†Not including its frontends
“70% of the performance of advanced compilers is in **10% 0.1%** of the code.”

<table>
<thead>
<tr>
<th>Compiler</th>
<th>Lines of code</th>
<th>Number of files</th>
</tr>
</thead>
<tbody>
<tr>
<td>LLVM</td>
<td>10,000,000</td>
<td>87,000</td>
</tr>
<tr>
<td>GCC‡</td>
<td>9,000,000</td>
<td>100,000</td>
</tr>
<tr>
<td>qbe</td>
<td>14,000</td>
<td>55</td>
</tr>
</tbody>
</table>

‡Not including its frontends
Performance?

Let’s compile BearSSL with each compiler and compare the results.
Let’s compile BearSSL with cproc and gcc and compare the results.

<table>
<thead>
<tr>
<th>Compiler</th>
<th>Build (seconds)</th>
<th>Tests (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clang 12.0.1 with -O2</td>
<td>5.05</td>
<td>45.91</td>
</tr>
<tr>
<td>GCC 1.11.2 with -O2</td>
<td>4.24</td>
<td>43.79</td>
</tr>
<tr>
<td>cproc 67aee986</td>
<td>1.30</td>
<td>62.87</td>
</tr>
</tbody>
</table>

Lower numbers are better. Run on an AMD Ryzen 7 3700X on Alpine Linux edge.
Performance!

“70% of the performance of advanced compilers in 10% of the code.”

73% of the runtime performance and 380% the compile performance of advanced compilers in 0.1% of the code.

...based on building & testing BearSSL. Yay!
BearSSL speed tests

Selected results from BearSSL testspeed (all values in MB/s):

<table>
<thead>
<tr>
<th>Test</th>
<th>GCC</th>
<th>cproc</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHA-256</td>
<td>295 MB/s</td>
<td>159 MB/s</td>
</tr>
<tr>
<td>SHA-512</td>
<td>463 MB/s</td>
<td>225 MB/s</td>
</tr>
<tr>
<td>AES-128</td>
<td>266 MB/s</td>
<td>69 MB/s</td>
</tr>
<tr>
<td>ChaCha20</td>
<td>545 MB/s</td>
<td>109 MB/s</td>
</tr>
<tr>
<td>Poly1305</td>
<td>1593 MB/s</td>
<td>481 MB/s</td>
</tr>
<tr>
<td>SHAKE256</td>
<td>526 MB/s</td>
<td>230 MB/s</td>
</tr>
</tbody>
</table>

Higher numbers are better. Run on an AMD Ryzen 7 3700X on Alpine Linux edge.

Full results:
https://mirror.drewdevault.com/bearssl-gcc-11.2.1.log
https://mirror.drewdevault.com/bearssl-cproc-67ae986.log
Performance!

“70% of the performance of advanced compilers in 10% of the code.”

73% 25-75% of the runtime performance and 380% the compile performance of advanced compilers in 0.1% of the code.

...based BearSSL testspeed. But: is it worth it?
Ports

qbe ports today:

- x86_64: 2,118 lines of code
- aarch64: 1,665 lines of code
- riscv64: 1,458 lines of code; 341 commits by one (talented) author over 8 months

qbe ports tomorrow?

- ppc64 (big endian?)
- 32-bit: i486 et al, armhf et al, riscv32
- Others?
- Plus: Plan 9
"QBE aims to be a pure C embeddable backend that provides 70% of the performance of advanced compilers in 10% of the code. Its small size serves both its aspirations of correctness and our ability to understand, fix, and improve it. It also serves its users by providing trivial integration and great flexibility."

qbe: https://c9x.me/compile

cproc: https://sr.ht/~mcf/cproc