# **LDC:** A Dragon on Mars

David Nadlinger (@klickverbot)

DConf 2013

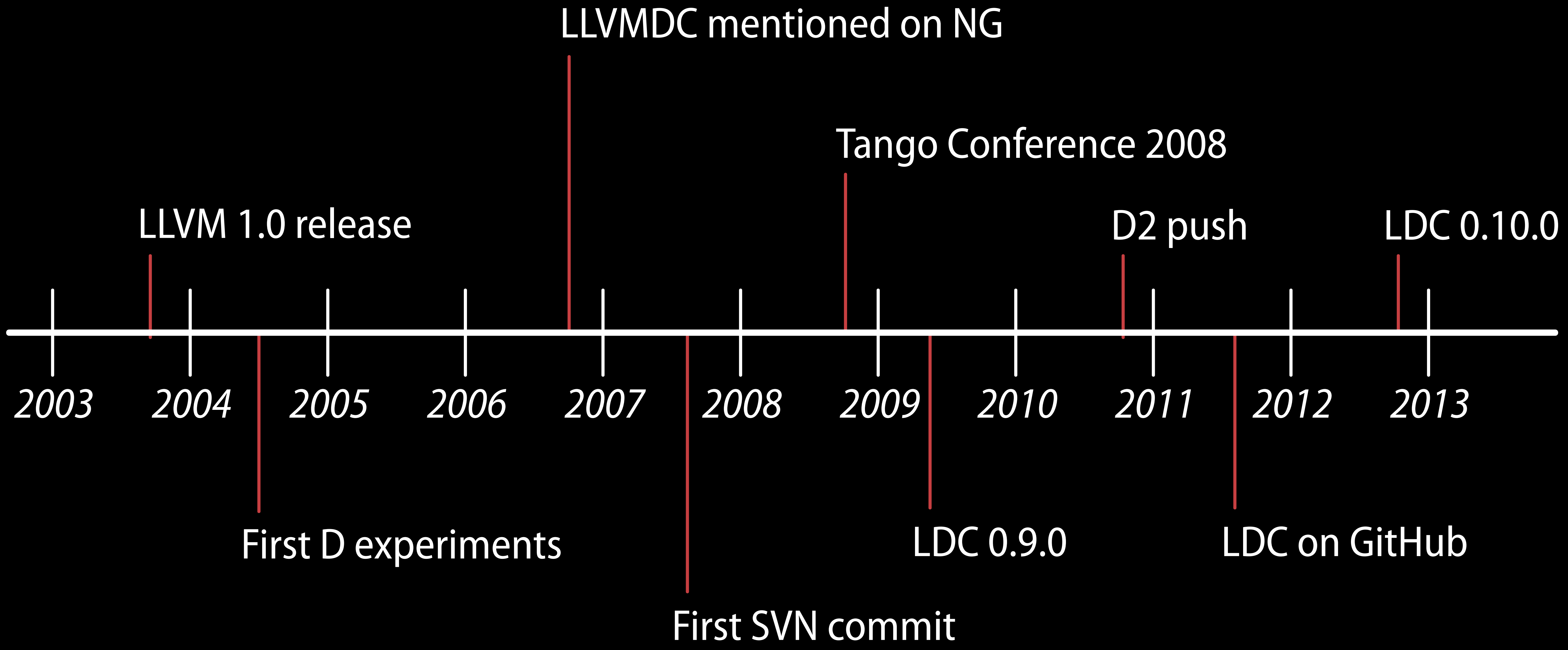
**L**LVM-based **D** **C**ompiler

# LLVM

Looking just at the core:

- »Compiler Infrastructure«
- x86, ARM, PPC, Mips, NVPTX, R600, …
- Advanced optimizer
- Apple, Intel, NVidia, AMD, Google, …
- Open Source! (BSD-like)

\+ ?


LLVMDC mentioned on NG
Tango Conference 2008
LLVM 1.0 release
D2 push
LDC 0.10.0
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
First D experiments
LDC 0.9.0
LDC on GitHub
First SVN commit

# **Status:** Overview

- DMD 2.062
- Linux and OS X, x86/x86_64: Just works™
- Passes (almost) the full DMD/druntime/Phobos test suites

# **Status:** Target Architectures

- x86/x86_64: Main target, solid support
- PPC64: Standard library compiles, does not pass all tests
- ARM: Painfully close to Hello World

# **Status:** Target Operating Systems

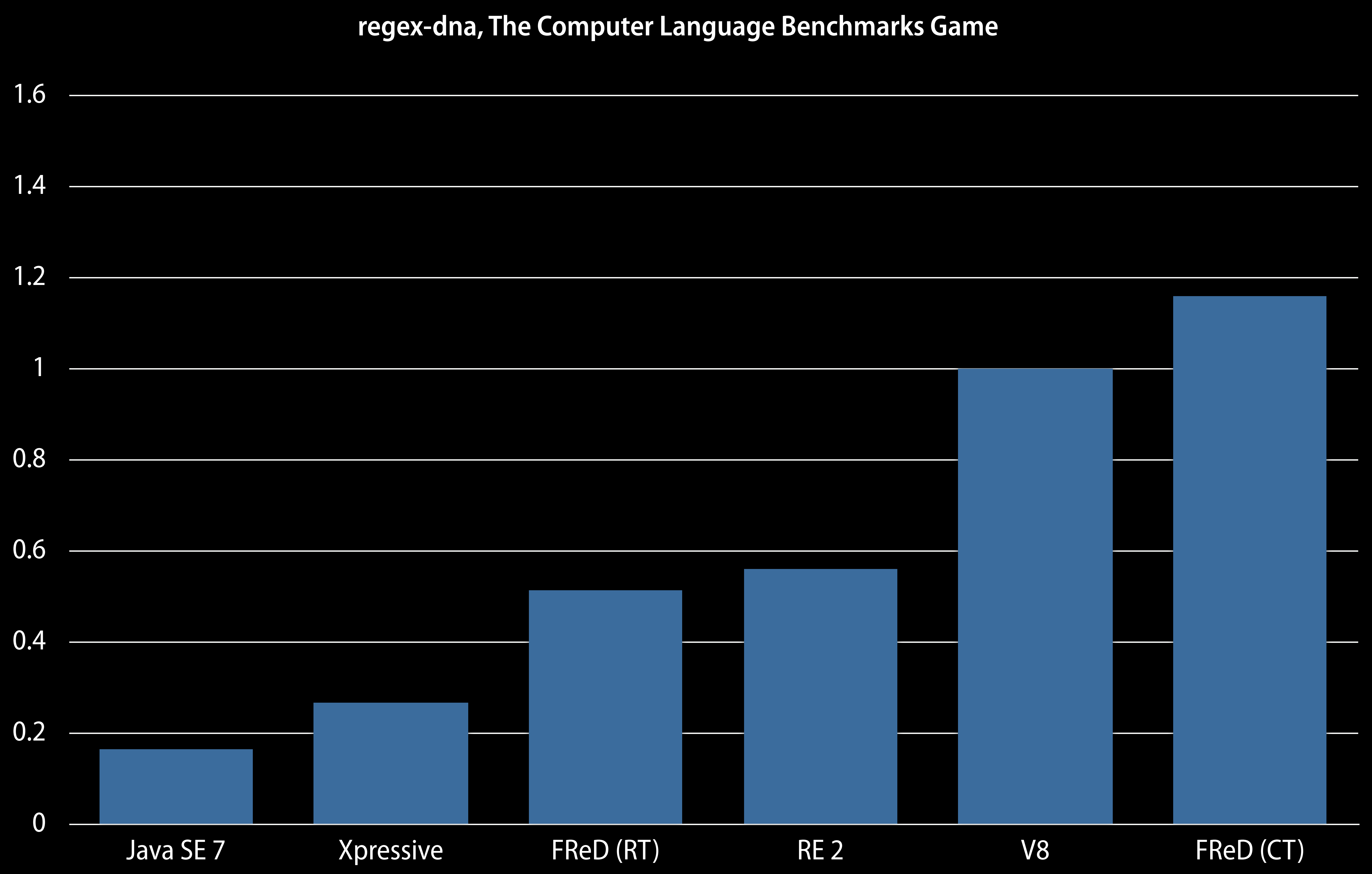
- (GNU) Linux: Main development platform
- OS X: Stable (10.7+)
- Windows: x86/MinGW, x64/MSVC
- *BSD: Not regularly tested
- Android: Bionic specifics need work
- AIX, Haiku, …

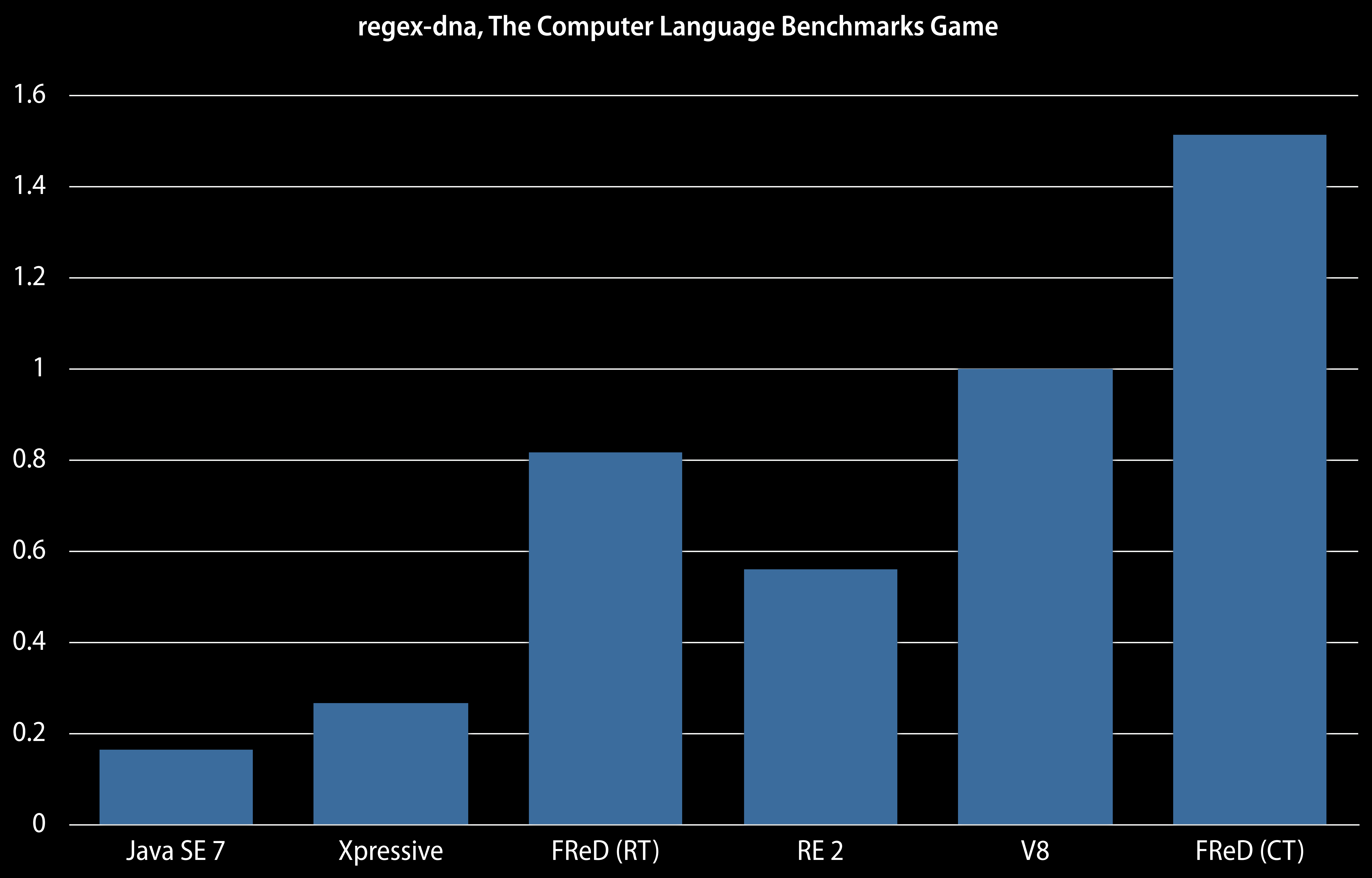
# **Status:** Language support

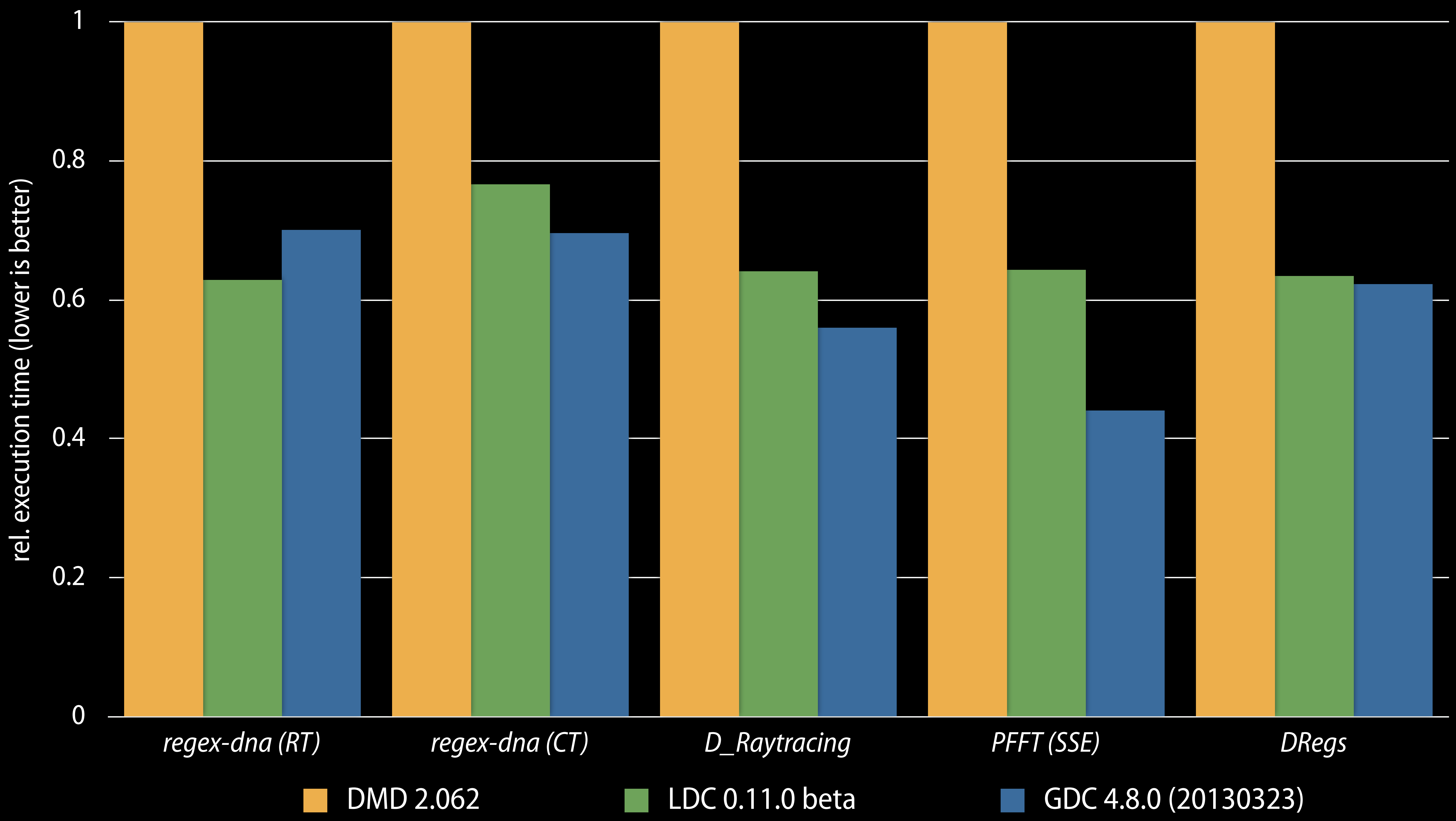
- Exception chaining
- Inline assembler restrictions
- Multiple `extern(C)` declaration hacks
- LDC-specific pragmas/intrinsics
- SIMD

# **Status:** DMD compatibility

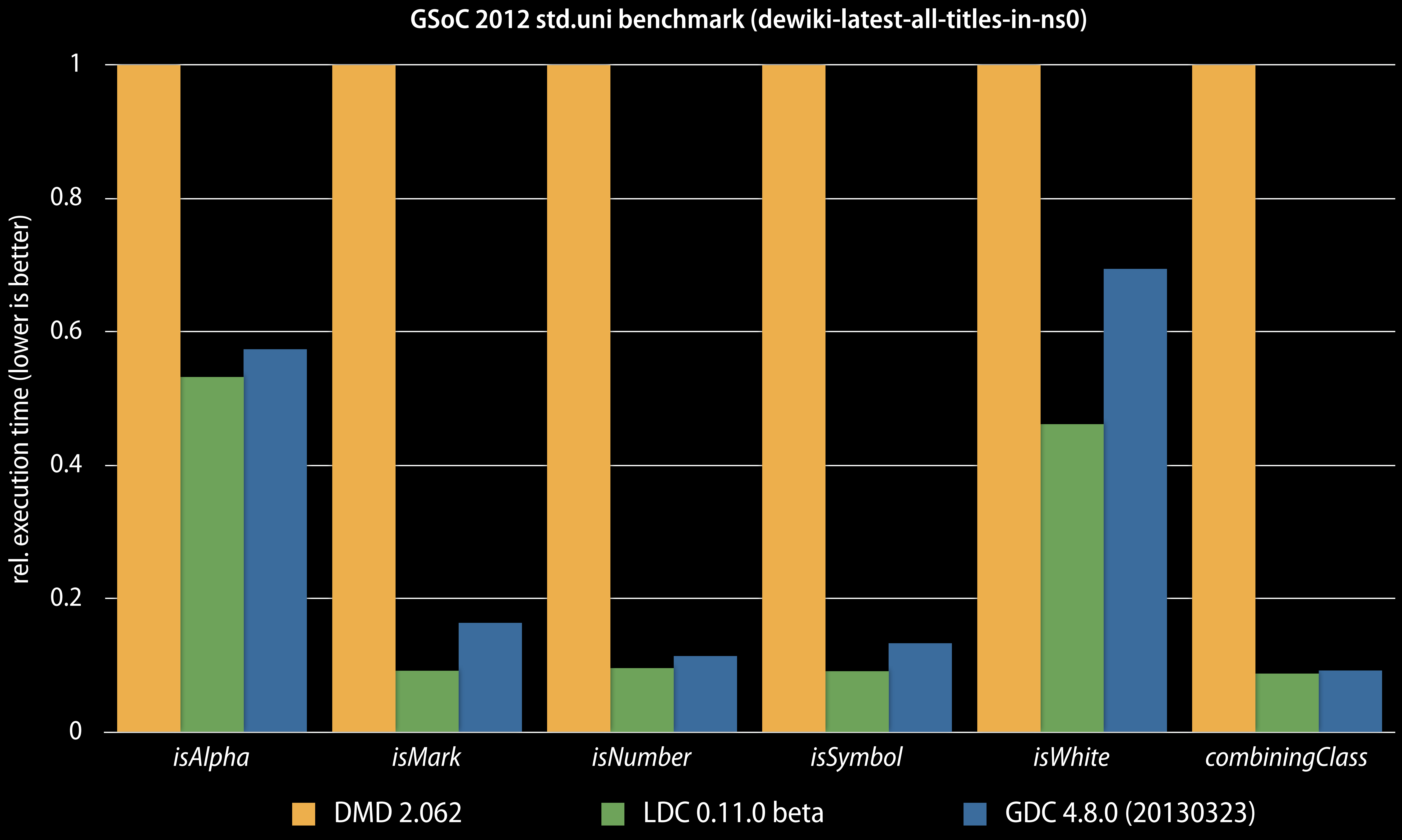
- `ldmd2`: Drop-in replacement for DMD
- Supports `-deps`, … (RDMD)
- `-cov`, `-profile` not available
- Goal for ABI, etc.: As compatible as reasonably possible

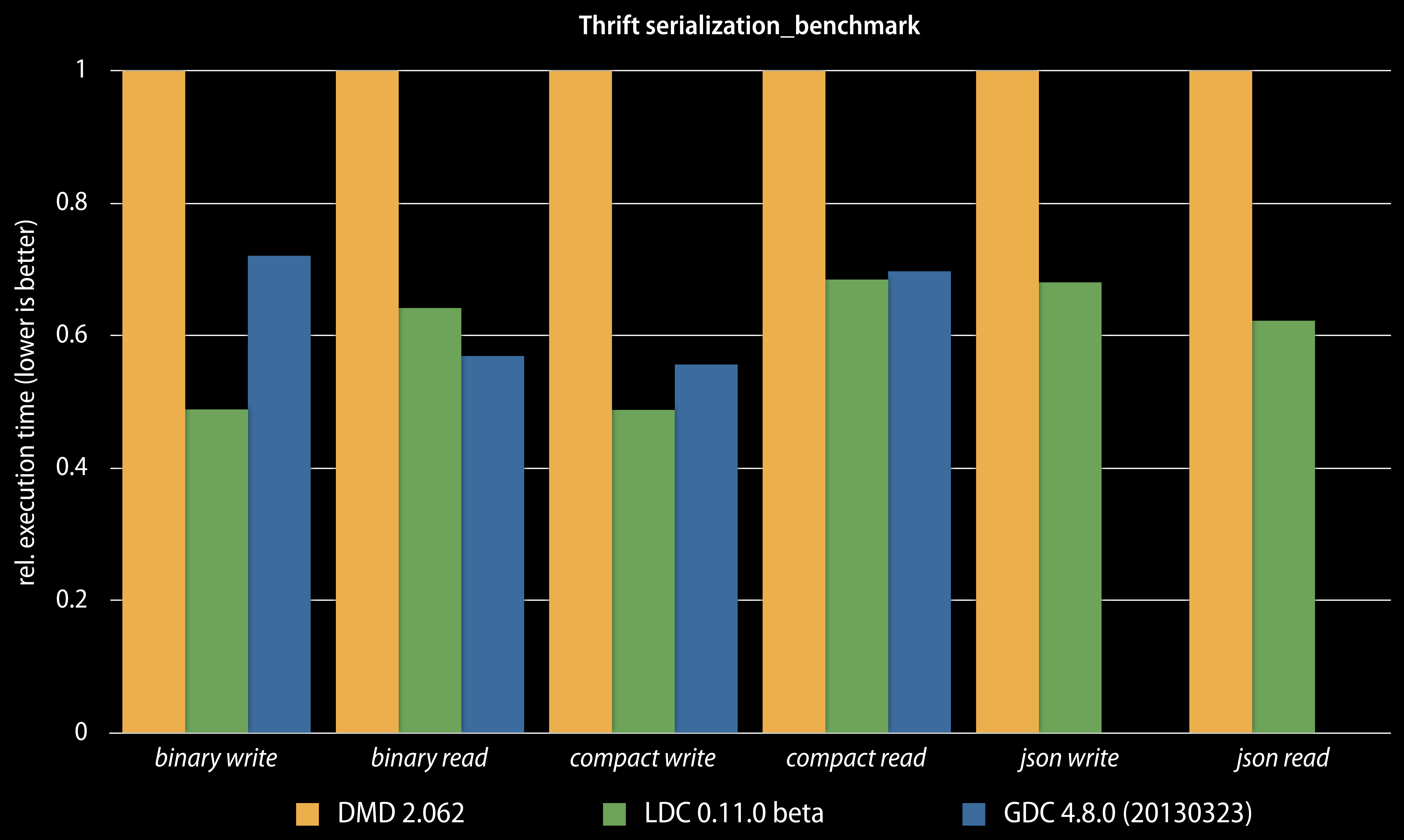

regex-dna, The Computer Language Benchmarks Game
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
Java SE 7
Xpressive
FReD (RT)
RE 2
V8
FReD (CT)


regex-dna, The Computer Language Benchmarks Game
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
Java SE 7
Xpressive
FReD (RT)
RE 2
V8
FReD (CT)



rel. execution time (lower is better)
1
0.8
0.6
0.4
0.2
0
regex-dna (RT)
regex-dna (CT)
D_Raytracing
PFFT (SSE)
DRegs
DMD 2.062
LDC 0.11.0 beta
GDC 4.8.0 (20130323)


GSoC 2012 std.uni benchmark (dewiki-latest-all-titles-in-ns0)
rel. execution time (lower is better)
1
0.8
0.6
0.4
0.2
0
isAlpha
isMark
isNumber
isSymbol
isWhite
combiningClass
DMD 2.062
LDC 0.11.0 beta
GDC 4.8.0 (20130323)


Thrift serialization_benchmark
rel. execution time (lower is better)
1
0.8
0.6
0.4
0.2
0
binary write
binary read
compact write
compact read
json write
json read
DMD 2.062
LDC 0.11.0 beta
GDC 4.8.0 (20130323)

Hypothesis 1:

*Control is the unique offering of systems languages.*

Hypothesis 1:

*Control is the unique offering of systems languages.*

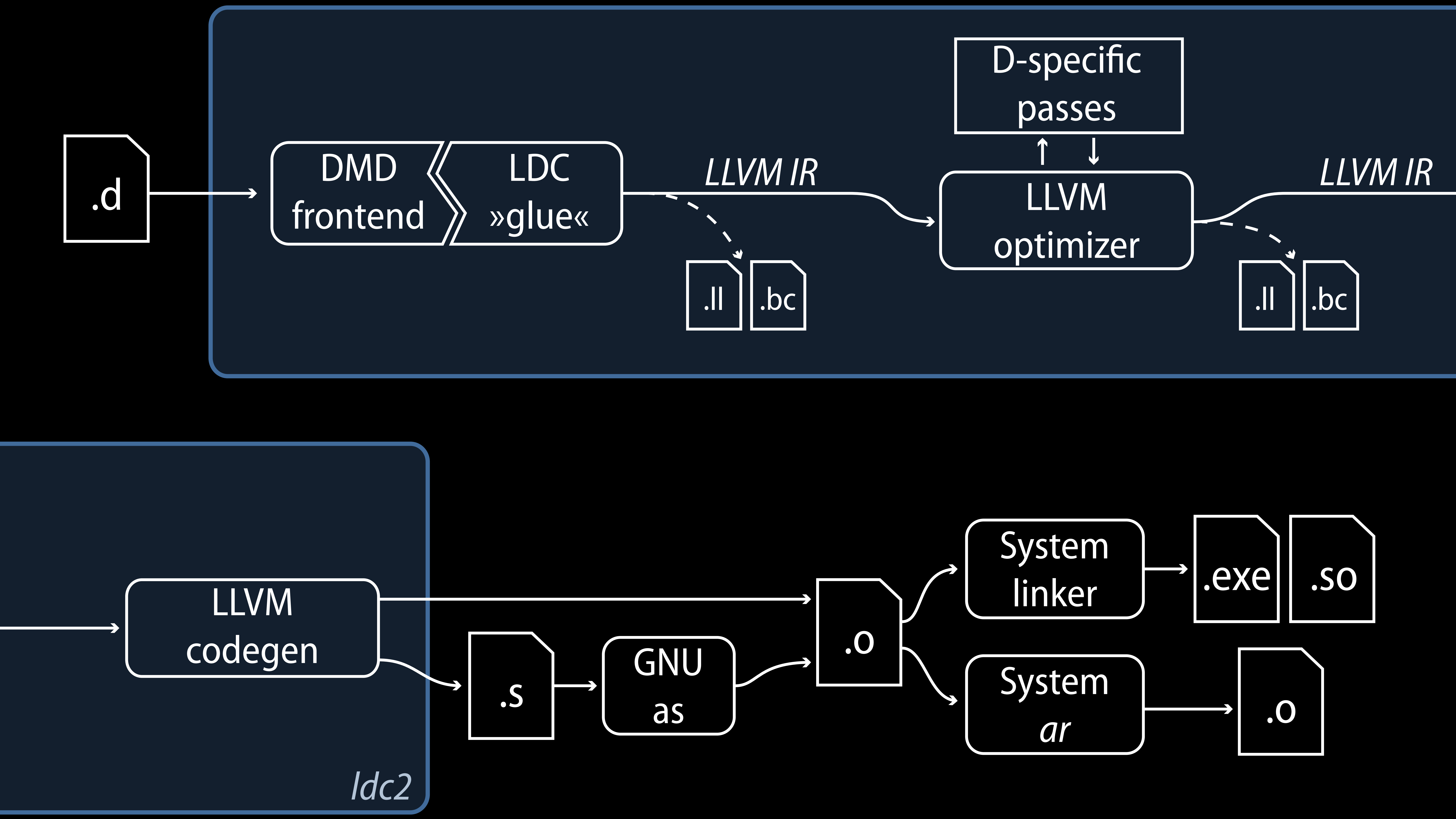
Hypothesis 2:

*Performance requirements drive the demand for control.*

Hypothesis 1:
*Control is the unique offering of systems languages.*

Hypothesis 2:
*Performance requirements drive the demand for control.*

Hypothesis 3:
*Zero-cost abstractions matter.*



.d
DMD frontend
LDC »glue«
LLVM IR
.ll
.bc
D-specific passes
LLVM optimizer
LLVM IR
.ll
.bc
LLVM codegen
ldc2
.s
GNU as
.o
System linker
.exe
.so
System ar
.o

# LLVM IR

```
module test;
int add(int a, int b) {
    return a + b;
}
```

```
define i32 @_D4test3addFiiZi(
  i32 %b_arg, i32 %a_arg) {
entry:
  %a = alloca i32
  %b = alloca i32
  store i32 %a_arg, i32* %a
  store i32 %b_arg, i32* %b
  %tmp = load i32* %a
  %tmp1 = load i32* %b
  %tmp2 = add i32 %tmp, %tmp1
  ret i32 %tmp2
}
```

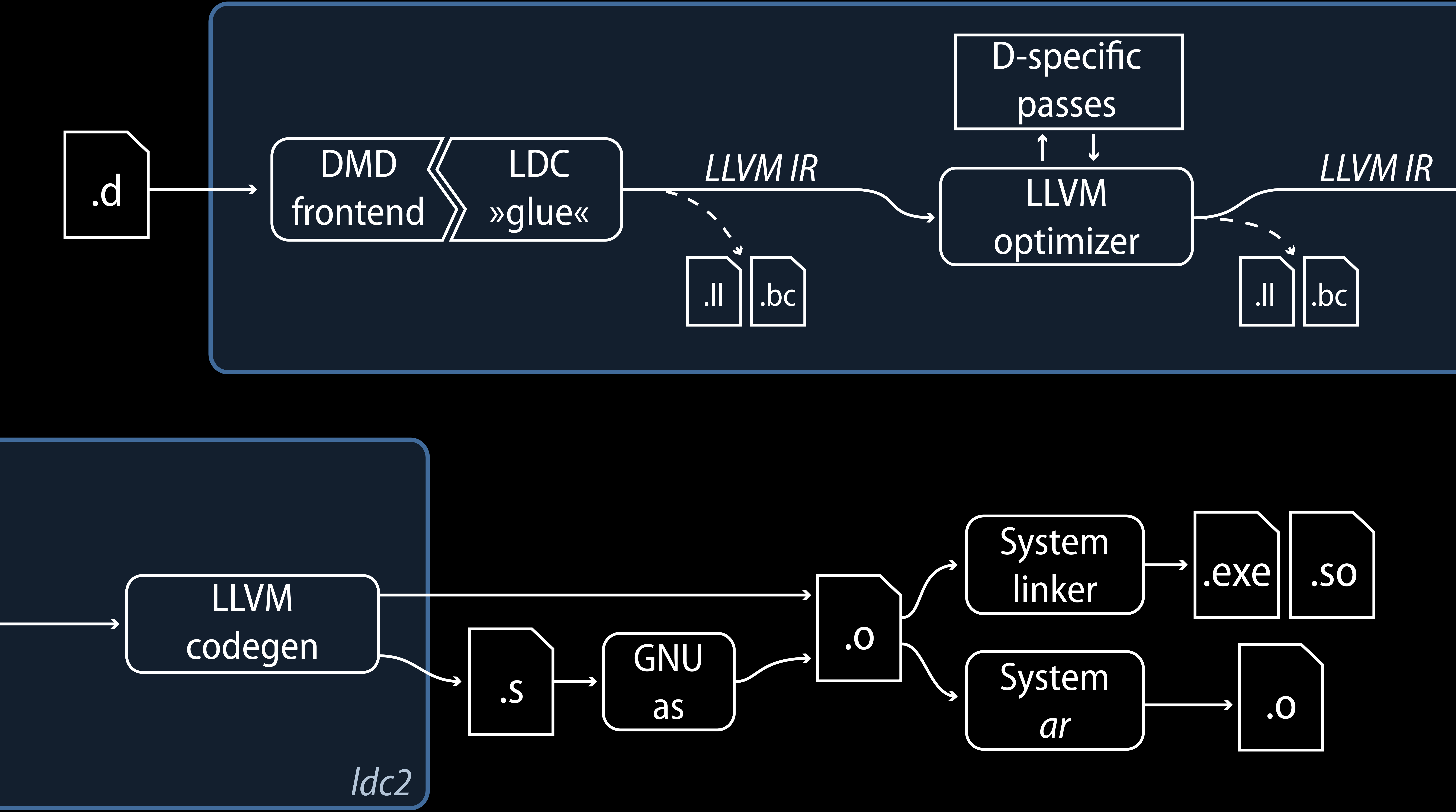
```
define i32 @_D4test3addFiiZi(
  i32 %b_arg, i32 %a_arg
) nounwind readnone {
entry:
  %tmp2 = add i32 %a_arg, %b_arg
  ret i32 %tmp2
}
```

# LLVM IR

```
define i32 @_D4test3addFiiZi(
  i32 %b_arg, i32 %a_arg
) nounwind readnone {
entry:
  %tmp2 = add i32 %a_arg, %b_arg
  ret i32 %tmp2
}
```

→

```
_D4test3addFiiZi:
    add EDI, ESI
    mov EAX, EDI
    ret
```


.d
DMD frontend
LDC »glue«
LLVM IR
.ll
.bc
D-specific passes
LLVM optimizer
LLVM IR
.ll
.bc
LLVM codegen
.s
GNU as
.o
System linker
.exe
.so
System ar
.o
ldc2

# DMD frontend

- Copy in LDC tree, merged on per-release basis
- Inliner, etc. not used
- Minimize changes to »shared« DMD source

# DMD frontend

- Lack of documentation
- Implicit invariants often hard to track down
- Layering violations
- “Impedance mismatch”: LLVM IR strictly typed

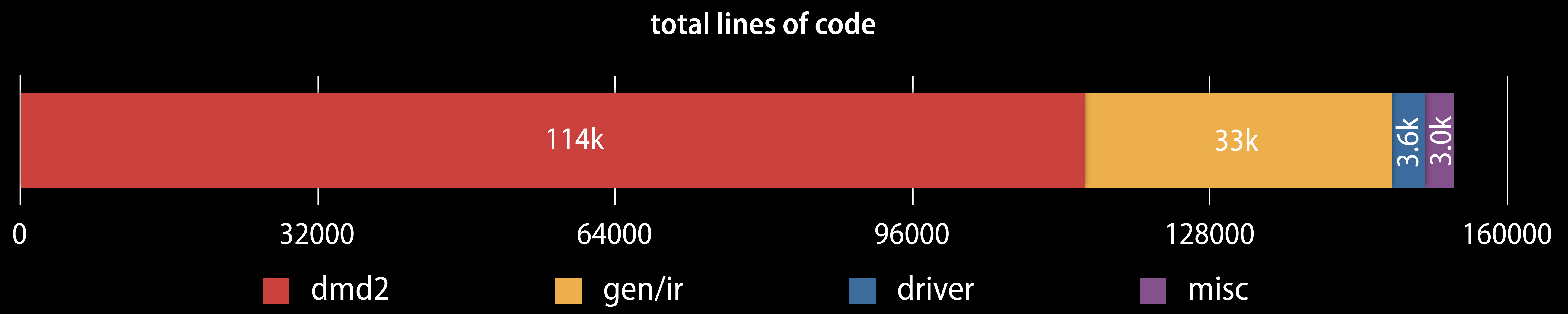
# **LLVM:** Advantages

- Easy to approach
- Defensively written code
- Effective debugging facilities (graphs, `dump()`, `bugpoint`, …)
- Modular, easy to extend
- Active, helpful community
- Patches easy to upstream

# **LLVM:** Challenges

- LLVM IR is target dependent (varargs, struct ABI, …)
- Liberal in C++ API changes
- No emulation for missing OS functionality
- Windows support not needed by big corporate clients

# Source size

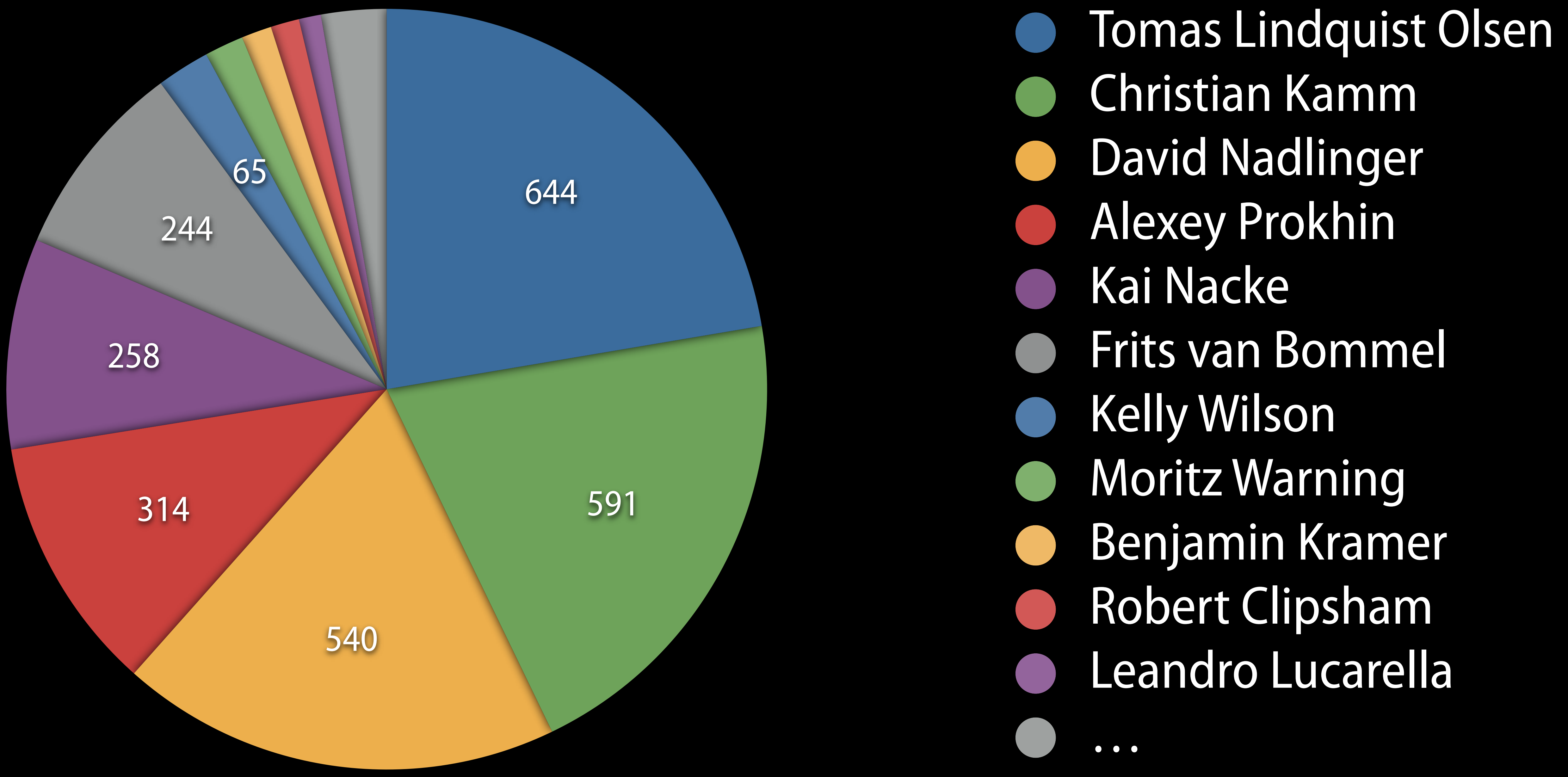


total lines of code
114k
33k
3.6k
3.0k
0
32000
64000
96000
128000
160000
dmd2
gen/ir
driver
misc

# Runtime and Tests

- druntime, Phobos, dmd-testsuite are Git submodules
- Pro: consistent state; Contra: merging harder
- druntime/Phobos changes: atomic operations, intrinsics (math/bitops), platform support, x86_64 varargs

# Committers



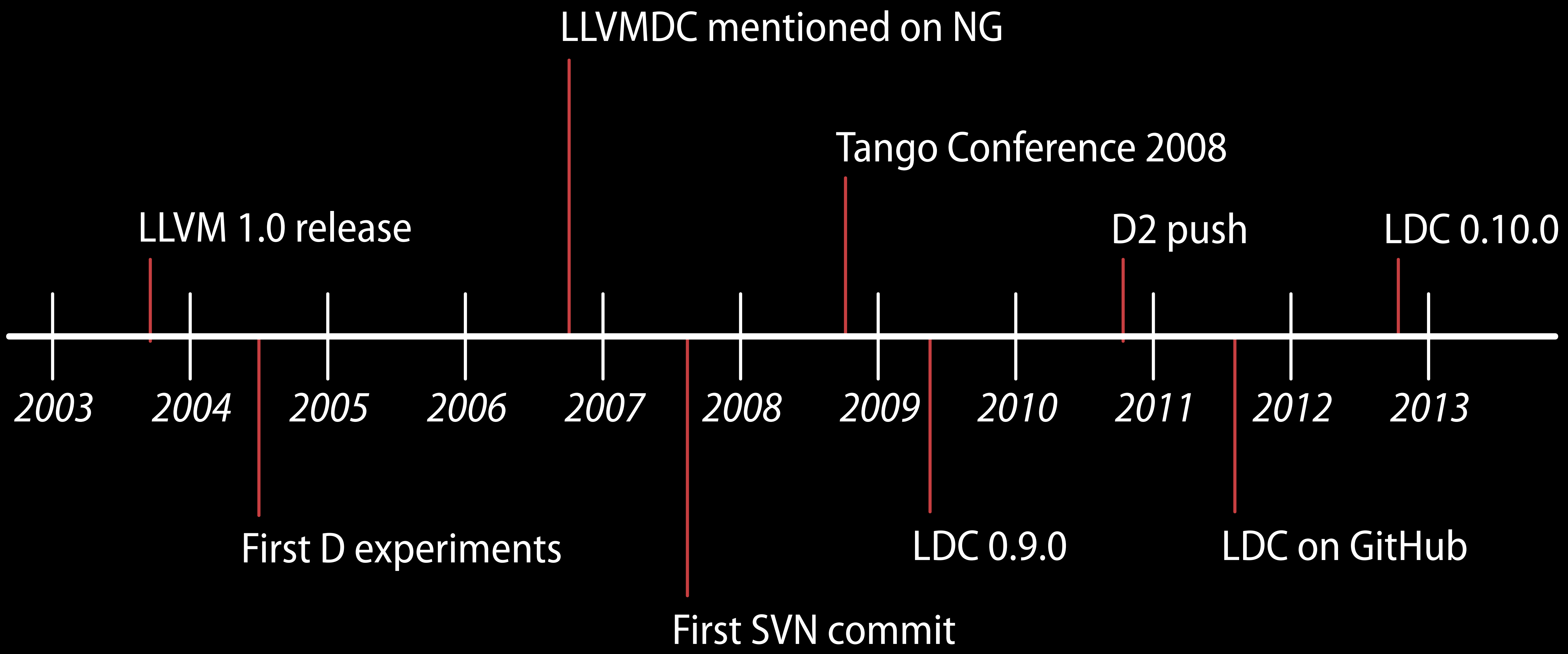
644
591
540
314
258
244
65
Tomas Lindquist Olsen
Christian Kamm
David Nadlinger
Alexey Prokhin
Kai Nacke
Frits van Bommel
Kelly Wilson
Moritz Warning
Benjamin Kramer
Robert Clipsham
Leandro Lucarella
…

# Development

- Source code, issue tracker: http://github.com/ldc-developers
- CI systems:
  - Travis (http://travis-ci.org, pull requests)
  - Lycus Foundation (http://ci.lycus.org, thanks Alex!)
- Forum/mailing list: http://forum.dlang.org
- Developer docs: http://wiki.dlang.org/LDC

# Next steps

- Shared library support (DMD 2.063)
- Exception chaining
- x86_64 vararg ABI
- Minimize differences to DMD upstream source, translation to D (?)


LLVMDC mentioned on NG
Tango Conference 2008
LLVM 1.0 release
D2 push
LDC 0.10.0
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
First D experiments
LDC 0.9.0
LDC on GitHub
First SVN commit

# Future directions

- Compiler performance work
- Leverage D attributes; further D-specific optimizations?
- Integration of  LLVM-based tools
  e.g.: AddressSanitizer/ThreadSanitizer
- Link-time optimization
- PNaCl/Emscripten/…
- Scripts for setting up cross toolchains

# Summary

LDC …

… is a D2 compiler.

… is »ready« on Linux/OS X, Windows coming.

… produces significantly faster code than DMD.

… provides many promising opportunities.

… is Open Source, and easy to hack on!

# LDC needs your help!

You could…

… fix bugs: tracker on GitHub, has »junior jobs«

… champion new platforms (ARM!)

… package LDC for your favorite distro *(Debian!)*

…help keeping the wiki/docs up to date

…try LDC on your own projects (DustMite!)

# Wish list

- ~~OS X and Win32/MinGW: Need a build slave~~ … Thanks, Brad!
- *Idea:* D compiler performance tracker
- "Real-world" benchmarks

http://wiki.dlang.org/LDC

http://forum.dlang.org

irc://irc.freenode.net/ldc

*David Nadlinger*

*<david@klickverbot.at>*

# Backup slides

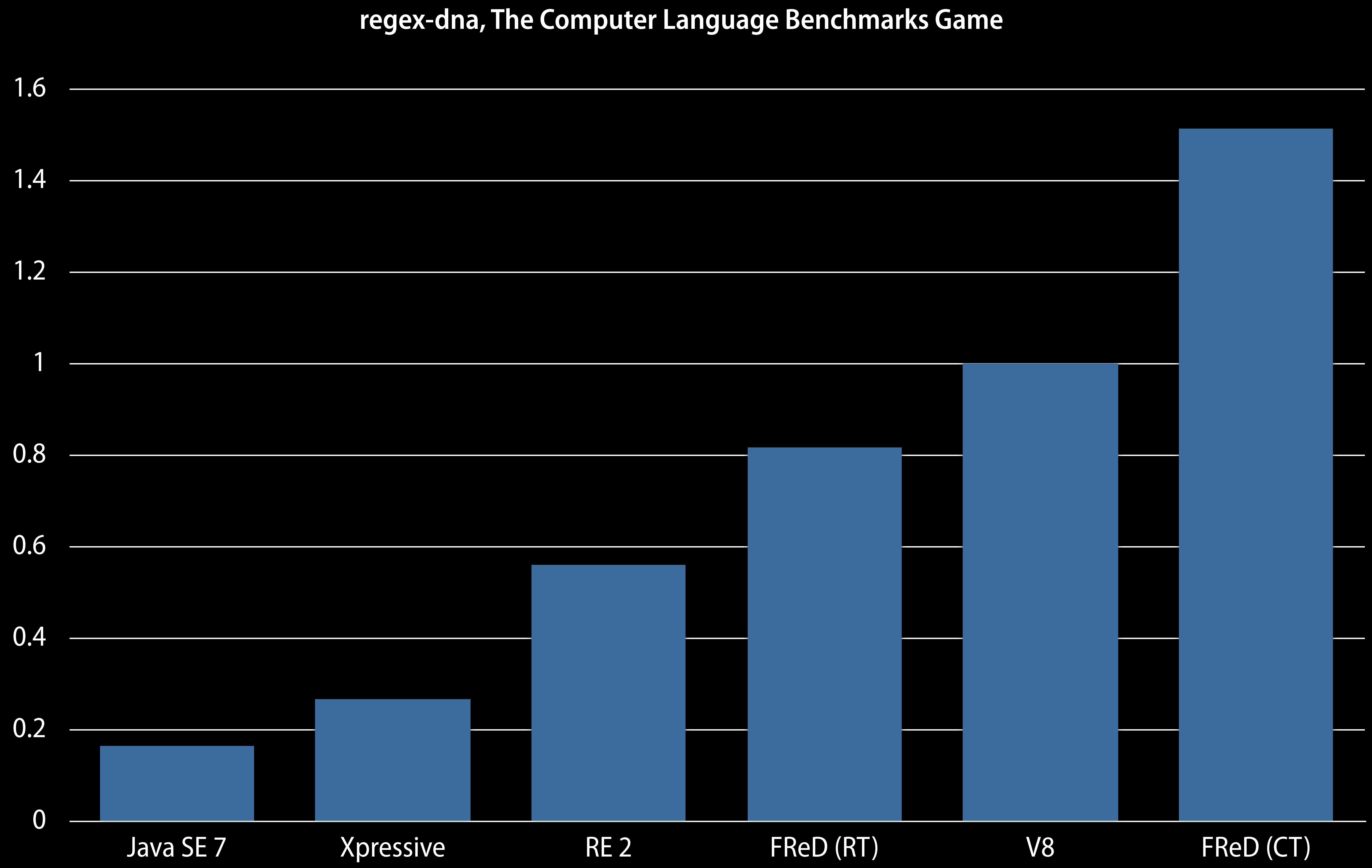
# **Status:** Performance

Optimized micro-benchmarks:

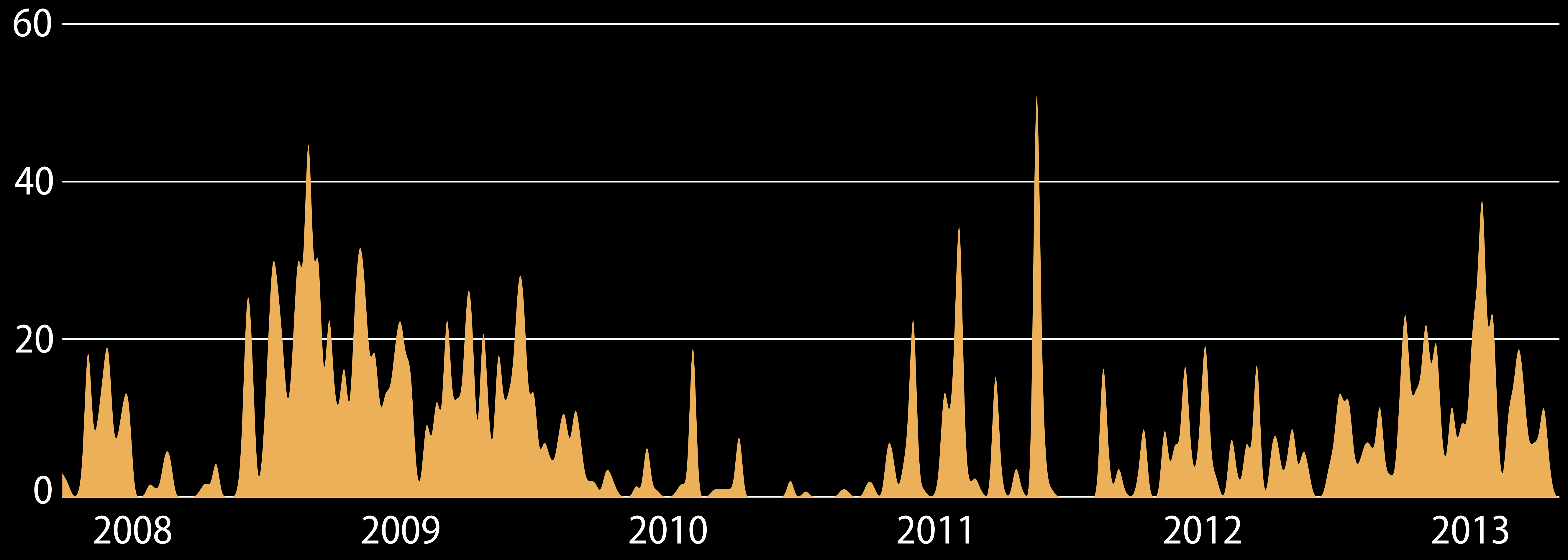
- Consistently ~30% faster than DMD
- On par with GDC

Further potential:

- D-specific optimizations
- Leverage attributes (`scope`, `pure`, …)


regex-dna, The Computer Language Benchmarks Game
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
Java SE 7
Xpressive
RE 2
FReD (RT)
V8
FReD (CT)

# Commit activity



# ABI compatibility

- Shared libraries!
- Documentation severely lacking
- GDC/LDC use default EH/… mechanisms
- Implications for packaging tools