

Сравнение эффективности компиляторов

Побенчмаркаем?



Проблема

- Компиляторов много
- Все они интересны, имеют те или иные особенности

А выбрать что?

Варианты решения

- Пробовать самому все компиляторы на целевом проекте
- Читать про особенности каждого компилятора
- Пролистать ВСЕ опции компилятора, которые могут влиять на результат
- А после пролистывания — понять их :-)

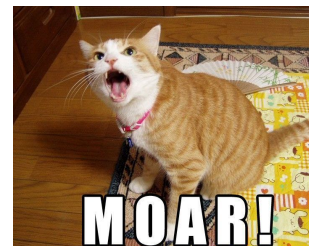


Проблема

- У разных компиляторов — разные опции
- Они далеко не всегда соответствуют друг другу
- А если и описание примерно одинаковое, то реализация может кардинально отличаться

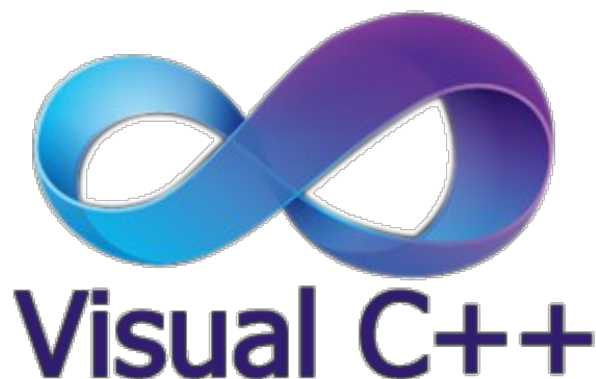
Участники соревнования

- GCC
- Clang
- ICC
- MSVC



Вести с фронта: MSVC

- Приятно радуют скоростью разработки
- Совместимость с новыми Стандартами улучшилась
- Внедряются всё новые и новые оптимизации (например, новый SSA)
- *Приятность* Модули



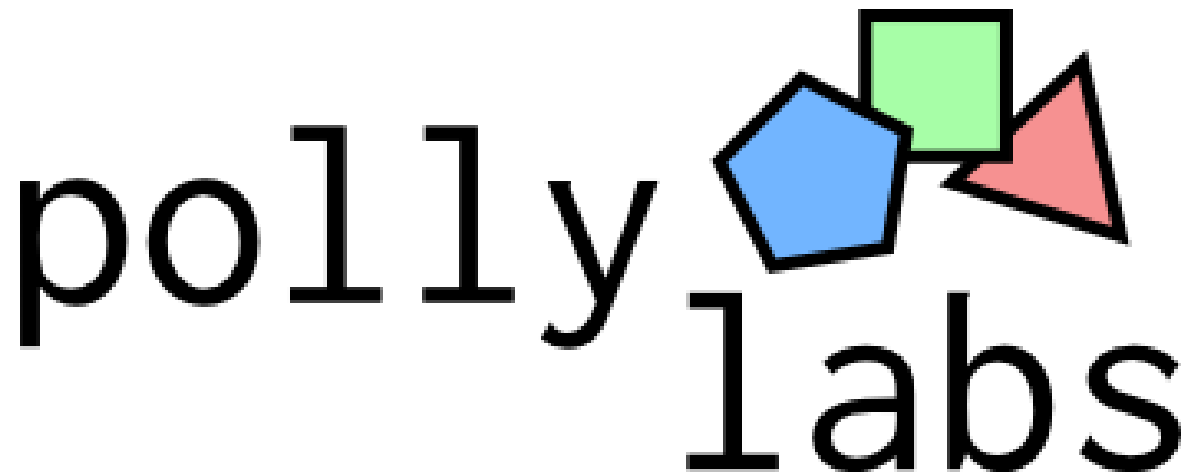
Вести с фронта: Clang

- В последние годы сильно догнал GCC в оптимизациях (спасибо, LLVM)
- Поддержка Стандарта превосходна
- Местами может потягаться с GCC во оптимизациях (привет, Polly)



Что такое Polly?

- Высокоуровневый оптимизатор кода для Clang
- Хочет попасть в компилятор
- Сайт: polly.llvm.org



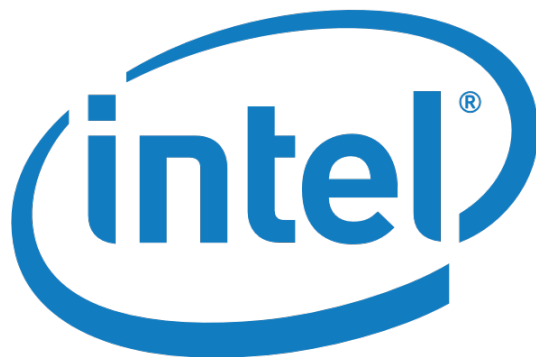
Вести с фронта: GCC

- В последнее время пушат много оптимизаций в компилятор
- В целом — стабильность :-)



Вести с фронта: ICC

- С каждой версией реализует кучу крутых оптимизаций(например, очень хороший LTO и векторные оптимизации)
- Помимо компилятора улучшаются сопутствующие библиотеки
- Отправили свою реализацию параллельной STL в libstdc++ && libc++




Тестовая система

- CPU: i7-3630QM, (4+4 потоков, 2.40 GHz, Turbo Boost: 3.40 GHz, 6 MiB кэш).
Поддерживает: SSE, SSE3, SSE4.1, AVX. **Не поддерживает: AVX2 и выше.**
- RAM: 16 GiB,
- SSD: Samsung 850 EVO
- OS: Fedora 26 (Linux kernel 4.13)

Опции компиляции

- -O2
- -O3
- -ffast-math (ICC + MSVC аналоги)
- -march=native (msvc аналог)
- Polly (Clang only)
- И может ещё чуть-чуть :-)



Почему только такие флаги

- Перебрать все флаги — очень и очень сложно
- Нам не хватит времени обо всём поговорить
- Большинство людей (и я тоже) на работе используем просто O2/O3 (+ чуть-чуть ещё :) или аналоги и хотим получить быстрый бинарь



На чём будем тестить?

- Числодробилки (e.g. ray-tracers)
- [Openbenchmarking.org](https://openbenchmarking.org)
- Более-менее реальные приложения :-)



Что будем мерять?

- Скорость выполнения скомпилированного файла
- Заодно посмотрим немного на то, что компиляторы нагенерировали на конкретных кейсах (спасибо godbolt.org)



Что мы НЕ будем мерять?

- Размер скомпилированного файла
- Скорость компиляции
- Различия в эффективности различных реализаций стандартной библиотеки



Что не будем трогать

- PGO
- LTO
- Использование всевозможных флагов компиляторов

Примеры

Compiler Explorer

C++ Editor Diff View More

Share Other

C++ source #1

```
1 #include <algorithm>
2 #include <cmath>
3
4 float foo(float a, float b)
5 {
6     return a / (1.0f / b);
7 }
8
```

x86-64 gcc 7.2 (Editor #1, Compiler #2) x

x86-64 gcc 7.2 --std=c++17 -O3 -march=native -fno-math-errno

```
1 foo(float, float):
2     vmulss    xmm0, xmm0, xmm1
3     ret
```

gcc (GCC-Explorer-Build) 7.2.0 - 941ms

x86-64 clang 5.0.0 (Editor #1, Compiler #3) x

x86-64 clang 5.0.0 --std=c++17 -O3 -march=native -fno-math-errno

```
1 foo(float, float): # @foo(float, float)
2     vmulss    xmm0, xmm1, xmm0
3     ret
```

clang version 5.0.0 (tags/RELEASE_500/final 312636) - 1936ms

x86-64 MSVC 19 2017 RTW (Editor #1, Compiler #4) x

x86-64 MSVC 19 2017 RTW /O2 /std:c++latest /fp:fast

```
1 __real@3f800000 DD 03f80000r ; 1
2 foo, COMDAT PROC
3     movss    xmm2, DWORD PTR __real@3f800000
4     divss    xmm2, xmm1
5     divss    xmm0, xmm2
6     ret     0
7 foo ENDP
```

Microsoft (R) C/C++ Optimizing Compiler Version 19.10.25017 for x64 - 1154ms

Примеры

Compiler Explorer

C++ Editor Diff View More

Share Other

C++ source #1

```
1 #include <algorithm>
2 #include <cmath>
3
4 float foo(float a, float b, float c, float d)
5 {
6     return a / b / c / d;
7 }
8 //a / b / c / d -> a / (b * c * d)
```

x86-64 icc 18 (Editor #1, Compiler #1)

x86-64 icc 18 -O3 -march=native -fp-model fast=2

```
11010 .LX0: .text // vs+ Intel Demangle
1 foo(float, float, float, float):
2     vdivss xmm4, xmm0, xmm1 #6.16
3     vdivss xmm5, xmm4, xmm2 #6.20
4     vdivss xmm0, xmm5, xmm3 #6.24
5     ret #6.24
```

icc (ICC) 18.0.0 20170811 - 1296ms

x86-64 clang 5.0.0 (Editor #1, Compiler #3)

x86-64 clang 5.0.0 --std=c++17 -O3 -march=native -fast-math

```
11010 .LX0: .text // vs+ Intel Demangle
1 foo(float, float, float, float): # @foo(float, float, flo
2     vmulss xmm1, xmm2, xmm1
3     vmulss xmm1, xmm1, xmm3
4     vdivss xmm0, xmm0, xmm1
5     ret
```

clang version 5.0.0 (tags/RELEASE_500/final 312636) - 1957ms

x86-64 gcc 7.2 (Editor #1, Compiler #2)

x86-64 gcc 7.2 --std=c++17 -O3 -march=native -fast-math

```
11010 .LX0: .text // vs+ Intel Demangle
1 foo(float, float, float, float):
2     vmulss xmm2, xmm2, xmm3
3     vmulss xmm1, xmm2, xmm1
4     vdivss xmm0, xmm0, xmm1
5     ret
```

gcc (GCC-Explorer-Build) 7.2.0 - 1478ms

x86-64 MSVC 19 2017 RTW (Editor #1, Compiler #4)

x86-64 MSVC 19 2017 RTW /O2 /std:c++latest /fp:fast

```
11010 .LX0: .text // vs+ Intel Demangle
1 foo, COMDAT PROC
2     divss xmm0, xmm1
3     divss xmm0, xmm2
4     divss xmm0, xmm3
5     ret 0
6 foo ENDP
```

Microsoft (R) C/C++ Optimizing Compiler Version 19.10.25017 for x64 - 1193ms

Примеры

Compiler Explorer

C++ source #1 x

```
1 #include <cmath>
2
3 float foo(float a, float b)
4 {
5     return std::sqrt(a) * std::sqrt(b);
6 }
7
8
```

x86-64 icc 18 (Editor #1, Compiler #1) x

x86-64 icc 18 -O3 -march=native -fp-model fast=2 -fma -prec-div -prec-sqrt

```
11010
1 foo(float, float):
2     vsqrtss xmm0, xmm0, xmm0 #487.12
3     vsqrtss xmm1, xmm1, xmm1 #487.12
4     vmulss xmm0, xmm0, xmm1 #5.27
5     ret #5.27
```

icc (ICC) 18.0.0 20170811 - 1216ms

x86-64 gcc (trunk) (Editor #1, Compiler #2) x

x86-64 gcc (trunk) --std=c++17 -O3 -march=native -ffast-math

```
11010
1 foo(float, float):
2     vmulss xmm0, xmm0, xmm1
3     vsqrtss xmm0, xmm0, xmm0
4     ret
```

gcc (GCC-Explorer-Build) 8.0.0 20171206 (experimental) - cached

x86-64 clang (trunk) (Editor #1, Compiler #3) x

x86-64 clang (trunk) --std=c++17 -O3 -march=native -ffast-math

```
11010
1 foo(float, float): # @foo(float, float)
2     vsqrtss xmm0, xmm0, xmm0
3     vsqrtss xmm1, xmm1, xmm1
4     vmulss xmm0, xmm0, xmm1
5     ret
```

clang version 6.0.0 (trunk 319878) - cached

x86-64 MSVC 19 2017 RTW (Editor #1, Compiler #4) x

x86-64 MSVC 19 2017 RTW /Ox /std:c++latest -fp:fast

```
11010
1 foo PROC
2     sqrtss    xmm0, xmm0
3     sqrtss    xmm1, xmm1
4     mulss     xmm0, xmm1
5     ret       0
6 foo ENDP
7 sqrt, COMDAT PROC
8     sqrtss    xmm0, xmm0
9     ret       0
10 sqrt ENDP
```

Microsoft (R) C/C++ Optimizing Compiler Version 19.10.25017 for x64 - 1472ms

Примеры

Compiler Explorer

C++ source #1 x

```
1 #include <cmath>
2
3 float foo(float a, float x, float y)
4 {
5     return std::pow(a, x) * std::pow(a, y);
6 }
7
8
```

x86-64 gcc (trunk) (Editor #1, Compiler #1) x

x86-64 gcc (trunk) -O3 -march=native -fp-model fast=2 -fma -prec-div -prec-sqrt

```
11010 LX0: .text // \s+ Intel Demangle
1 foo(float, float, float):
2     sub rsp, 24 #4.1
3     vmovss DWORD PTR [rsp], xmm2 #4.1[spill]
4     vmovss DWORD PTR [8+rsp], xmm0 #4.1[spill]
5     call powf #412.12
6     vmovss DWORD PTR [16+rsp], xmm0 #412.12[spill]
7     vmovss xmm0, DWORD PTR [8+rsp] #412.12[spill]
8     vmovss xmm1, DWORD PTR [rsp] #412.12[spill]
9     call powf #412.12
10    vmulss xmm0, xmm0, DWORD PTR [16+rsp] #5.29[spill]
11    add rsp, 24 #5.29
12    ret #5.29
```

icc (ICC) 18.0.0 20170811 - 763ms

x86-64 gcc (trunk) (Editor #1, Compiler #2) x

x86-64 gcc (trunk) --std=c++17 -O3 -march=native -fast-math

```
11010 LX0: .text // \s+ Intel Demangle
1 foo(float, float, float):
2     vaddss xmm1, xmm1, xmm2
3     jmp __powf_finite
```

g++ (GCC-Explorer-Build) 8.0.0 20171206 (experimental) - 984ms

x86-64 clang (trunk) (Editor #1, Compiler #3) x

x86-64 clang (trunk) --std=c++17 -O3 -march=native -fast-math

```
11010 LX0: .text // \s+ Intel Demangle
1 foo(float, float, float): # @foo(float, float, float)
2     sub rsp, 24
3     vmovss dword ptr [rsp + 20], xmm2 # 4-byte Spill
4     vmovss dword ptr [rsp + 12], xmm0 # 4-byte Spill
5     call powf
6     vmovss dword ptr [rsp + 16], xmm0 # 4-byte Spill
7     vmovss xmm0, dword ptr [rsp + 12] # 4-byte Reload
8     vmovss xmm1, dword ptr [rsp + 20] # 4-byte Reload
9     call powf
10    vmulss xmm0, xmm0, dword ptr [rsp + 16] # 4-byte Folded
11    add rsp, 24
12    ret
```

clang version 6.0.0 (trunk 319878) - 1662ms

x86-64 MSVC 19 2017 RTW (Editor #1, Compiler #4) x

x86-64 MSVC 19 2017 RTW /Ox /std:c++latest -fp:fast

```
11010 LX0: .text // \s+ Intel Demangle
1 EXTRN powf:PROC
2 foo PROC
3     sub     rsp, 88             ; 00000058H
4     movaps  XMMWORD PTR [rsp+64], xmm6
5     movaps  XMMWORD PTR [rsp+48], xmm7
6     movaps  xmm7, xmm0
7     movaps  XMMWORD PTR [rsp+32], xmm8
8     movaps  xmm8, xmm2
9     call    powf
10    movaps  xmm6, xmm0
11    movaps  xmm1, xmm8
12    movaps  xmm0, xmm7
13    call    powf
14    movaps  xmm7, XMMWORD PTR [rsp+48]
15    movaps  xmm8, XMMWORD PTR [rsp+32]
16    mulss   xmm6, xmm0
17    movaps  xmm0, xmm6
18    movaps  xmm6, XMMWORD PTR [rsp+64]
19    add     rsp, 88             ; 00000058H
20    ret     0
21 foo ENDP
22 pow, COMDAT PROC
```

Microsoft (R) C/C++ Optimizing Compiler Version 19.10.25017 for x64 - 569ms

Примеры

Compiler Explorer

C++ Editor Diff View More

Share Other

C++ source #1 x

```
1 #include <cmath>
2
3 float foo(float x)
4 {
5     return std::sin(x) / std::cos(x);
6 }
7
8
```

x86-64 gcc 18 (Editor #1, Compiler #1) x

x86-64 gcc 18 -O3 -march=native -fp-model fast=2 -fma -prec-div -prec-sqrt

```
11010 LX0: .text // \s+ Intel Demangle
1 foo(float):
2     push rsi #4.1
3     vzeroupper #5.26
4     call __libm_sse2_sincosf #5.26
5     vdivss xmm0, xmm0, xmm1 #5.26
6     pop rcx #5.26
7     ret #5.26
```

clang version 6.0.0 (trunk 319878) - 1425ms

x86-64 clang (trunk) (Editor #1, Compiler #3) x

x86-64 clang (trunk) --std=c++17 -O3 -march=native -ffast-math

```
11010 LX0: .text // \s+ Intel Demangle
1 foo(float): # @foo(float)
2     push rax
3     lea rdi, [rsp + 4]
4     mov rsi, rsp
5     call sincosf
6     vmovss xmm0, dword ptr [rsp + 4] # xmm0 = mem[0],zero,z
7     vdivss xmm0, xmm0, dword ptr [rsp]
8     pop rax
9     ret
```

x86-64 MSVC 19 2017 RTW (Editor #1, Compiler #4) x

x86-64 MSVC 19 2017 RTW /Ox /std:c++latest -fp:fast

```
11010 LX0: .text // \s+ Intel Demangle
1 EXTRN cosf:PROC
2 EXTRN sinf:PROC
3 foo PROC
4     sub     rsp, 72                ; 00000048H
5     movaps  XMMWORD PTR [rsp+48], xmm6
6     movaps  xmm6, xmm0
7     movaps  XMMWORD PTR [rsp+32], xmm7
8     call    sinf
9     movaps  xmm7, xmm0
10    movaps  xmm0, xmm6
11    call    cosf
12    movaps  xmm6, XMMWORD PTR [rsp+48]
13    divss   xmm7, xmm0
14    movaps  xmm0, xmm7
15    movaps  xmm7, XMMWORD PTR [rsp+32]
16    add     rsp, 72                ; 00000048H
17    ret     0
18 foo ENDP
19 sin, COMDAT PROC
20     jmp     sinf
21 sin ENDP
22 cos, COMDAT PROC
```

gcc (GCC-Explorer-Build) 8.0.0 20171206 (experimental) - 980ms

x86-64 gcc (trunk) (Editor #1, Compiler #2) x

x86-64 gcc (trunk) --std=c++17 -O3 -march=native -ffast-math

```
11010 LX0: .text // \s+ Intel Demangle
1 foo(float):
2     jmp     tanf
```

Microsoft (R) C/C++ Optimizing Compiler Version 19.10.25017 for x64 - 1132ms

Примеры

Compiler Explorer

C++ Editor Diff View More

Share Other

C++ source #1

```
1 #include <cmath>
2
3 float foo(float a, float x)
4 {
5     return std::pow(a, x) * a;
6 }
7
8
```

x86-64 gcc 18 (Editor #1, Compiler #1)

x86-64 clang (trunk) (Editor #1, Compiler #3)

clang version 6.0.0 (trunk 319878) - 2015ms

x86-64 MSVC 19 2017 RTW (Editor #1, Compiler #4)

x86-64 gcc (trunk) (Editor #1, Compiler #2)

g++ (GCC-Explorer-Build) 8.0.0 20171206 (experimental) - 1800ms

Microsoft (R) C/C++ Optimizing Compiler Version 19.10.25017 for x64 - 563ms

11010 .text .r .s+ Intel Demangle

```
1 foo(float, float):
2   push rsi #4.1
3   vmovss DWORD PTR [rsp], xmm0 #4.1[spill]
4   call powf #412.12
5   vmulss xmm0, xmm0, DWORD PTR [rsp] #5.29[spill]
6   pop rcx #5.29
7   ret #5.29
```

11010 .text .r .s+ Intel Demangle

```
1 foo(float, float): # @foo(float, float)
2   push rax
3   vmovss dword ptr [rsp + 4], xmm0 # 4-byte Spill
4   call powf
5   vmulss xmm0, xmm0, dword ptr [rsp + 4] # 4-byte Folded
6   pop rax
7   ret
```

11010 .text .r .s+ Intel Demangle

```
1 foo(float, float):
2   sub     rsp, 24
3   vmovss  DWORD PTR [rsp+12], xmm0
4   call    __powf_finite
5   vmovss  xmm2, DWORD PTR [rsp+12]
6   vmulss  xmm0, xmm2, xmm0
7   add     rsp, 24
8   ret
```

11010 .text .r .s+ Intel Demangle

```
1 EXTRN  powf:PROC
2 foo PROC
3   sub     rsp, 56             ; 00000038H
4   movaps  XMMWORD PTR [rsp+32], xmm6
5   movaps  xmm6, xmm0
6   call    powf
7   mulss   xmm0, xmm6
8   movaps  xmm6, XMMWORD PTR [rsp+32]
9   add     rsp, 56             ; 00000038H
10  ret     0
11 foo ENDP
12 pow, COMDAT PROC
13 | jmp     powf
14 pow ENDP
```

Примеры

Compiler Explorer

C++ Editor Diff View More

Share Other

C++ source #1

```
1 #include <cmath>
2
3 float foo(float a, float b)
4 {
5     return exp(a) * exp(b);
6 }
7
8
```

x86-64 gcc (trunk) (Editor #1, Compiler #1)

x86-64 gcc (trunk) --std=c++17 -O3 -march=native -fpmath=fast -fma -prec-div -prec-sqrt

```
1 foo(float, float):
2     sub     rsp, 24 #4.1
3     vcvts2sd xmm0, xmm0, xmm0 #5.12
4     vmovss  DWORD PTR [8+rsp], xmm1 #4.1[spill]
5     call    exp #5.12
6     vmovsd  QWORD PTR [rsp], xmm0 #5.12[spill]
7     vxorpd  xmm0, xmm0, xmm0 #5.21
8     vcvts2sd xmm0, xmm0, DWORD PTR [8+rsp] #5.21[spill]
9     call    exp #5.21
10    vmulsd  xmm0, xmm0, QWORD PTR [rsp] #5.21[spill]
11    vcvtsd2ss xmm0, xmm0, xmm0 #5.21
12    add     rsp, 24 #5.21
13    ret     #5.21
```

x86-64 clang (trunk) (Editor #1, Compiler #3)

x86-64 clang (trunk) --std=c++17 -O3 -march=native -ffast-math

```
1 foo(float, float): # @foo(float, float)
2     sub     rsp, 24
3     vmovss  dword ptr [rsp + 12], xmm1 # 4-byte Spill
4     vcvts2sd xmm0, xmm0, xmm0
5     call    exp
6     vmovsd  qword ptr [rsp + 16], xmm0 # 8-byte Spill
7     vmovss  dword ptr [rsp + 12] # 4-byte Reload
8     vcvts2sd xmm0, xmm0, xmm0
9     call    exp
10    vmulsd  xmm0, xmm0, qword ptr [rsp + 16] # 8-byte Folded
11    vcvtsd2ss xmm0, xmm0, xmm0
12    add     rsp, 24
13    ret
```

clang version 6.0.0 (trunk 319878) - 1727ms

x86-64 MSVC 19 2017 RTW (Editor #1, Compiler #4)

x86-64 MSVC 19 2017 RTW /Ox /std:c++latest -fp:fast

```
1 EXTRN  expf:PROC
2 foo PROC
3     sub     rsp, 72 ; 00000048H
4     movaps  XMMWORD PTR [rsp+48], xmm6
5     movaps  XMMWORD PTR [rsp+32], xmm7
6     movaps  xmm7, xmm1
7     call    expf
8     movaps  xmm6, xmm0
9     movaps  xmm0, xmm7
10    call    expf
11    movaps  xmm7, XMMWORD PTR [rsp+32]
12    mulss   xmm6, xmm0
13    movaps  xmm0, xmm6
14    movaps  xmm6, XMMWORD PTR [rsp+48]
15    add     rsp, 72 ; 00000048H
16    ret     0
17 foo ENDP
18 exp, COMDAT PROC
19 |         jmp     expf
20 exp ENDP
```

gcc (GCC-Explorer-Build) 8.0.0 20171206 (experimental) - 1103ms

Microsoft (R) C/C++ Optimizing Compiler Version 19.10.25017 for x64 - 947ms

Примеры

Compiler Explorer

C++ Editor Diff View More

Share Other

C++ source #1

```
1 #include <cmath>
2
3 unsigned int foo(unsigned int a, unsigned int x,
4                 unsigned int y)
5 {
6     x = a | 3; // Sets lowest 2 bits, useless.
7     y = x >> 4;
8     return y;
9 }
10
11
```

x86-64 gcc (trunk) (Editor #1, Compiler #1)

x86-64 gcc (trunk) -O3 -march=native -fp-model fast=2 -fma -prec-div -prec-sqrt

```
1 foo(unsigned int, unsigned int, unsigned int):
2     or     edi, 3 #6.13
3     shr     edi, 4 #7.14
4     mov     eax, edi #8.12
5     ret     #8.12
```

x86-64 clang (trunk) (Editor #1, Compiler #3)

x86-64 clang (trunk) --std=c++17 -O3 -march=native -fast-math

```
1 foo(unsigned int, unsigned int, unsigned int): # @foo(unsigned int, unsigned int, unsigned int)
2     shr     edi, 4
3     mov     eax, edi
4     ret
```

clang version 6.0.0 (trunk 319878) - 1450ms

x86-64 MSVC 19 2017 RTW (Editor #1, Compiler #4)

x86-64 MSVC 19 2017 RTW /Ox /std:c++latest -fp:fast

```
1 foo PROC
2     shr     ecx, 4
3     mov     eax, ecx
4     ret     0
5 foo ENDP
```

gcc (GCC-Explorer-Build) 8.0.0 20171206 (experimental) - 1197ms

Microsoft (R) C/C++ Optimizing Compiler Version 19.10.25017 for x64 - 632ms

Примеры

Compiler Explorer

C++ Editor Diff View More

Share Other

C++ source #1

```
1 #include <cmath>
2
3 unsigned int foo(unsigned int a, unsigned int x,
4                 unsigned int y)
5 {
6     x = a & 0x0FFFFFFF; //useless
7     y = x | 0xFFFF0000;
8     return y;
9 }
10
11
```

x86-64 gcc 18 (Editor #1, Compiler #1)

x86-64 clang (trunk) (Editor #1, Compiler #3)

x86-64 gcc (trunk) (Editor #1, Compiler #2)

x86-64 MSVC 19 2017 RTW (Editor #1, Compiler #4)

gcc (GCC-Explorer-Build) 8.0.0 20171206 (experimental) - 1047ms

Microsoft (R) C/C++ Optimizing Compiler Version 19.10.25017 for x64 - 632ms

11010 .LX0: .text // %s+ Intel Demangle

```
1 foo(unsigned int, unsigned int, unsigned int):
2     or edi, -65536 #7.13
3     mov eax, edi #8.12
4     ret #8.12
```

11010 .LX0: .text // %s+ Intel Demangle

```
1 foo(unsigned int, unsigned int, unsigned int): # @foo(uns
2     or edi, -65536
3     mov eax, edi
4     ret
```

clang version 6.0.0 (trunk 319878) - 1750ms

x86-64 MSVC 19 2017 RTW (Editor #1, Compiler #4)

x86-64 MSVC 19 2017 RTW /Ox /std:c++latest -fp:fast

11010 .LX0: .text // %s+ Intel Demangle

```
1 foo PROC
2     or     ecx, -65536          ; ffff0000H
3     mov    eax, ecx
4     ret    0
5 foo ENDP
```

Примеры

Compiler Explorer

C++ Editor Diff View More

Share Other

C++ source #1

```
1 #include <cmath>
2
3 unsigned foo(unsigned x)
4 {
5     return x * 4;
6 }
7
8
```

x86-64 gcc 18 (Editor #1, Compiler #1)

x86-64 gcc 18 -O3 -march=native -fp-model fast=2 -fma -prec-div -prec-sqrt

```
11010 .LX0: .text // %s+ Intel Demangle
1 foo(unsigned int):
2 shl edi, 2 #5.16
3 mov eax, edi #5.16
4 ret #5.16
```

gcc (GCC) 18.0.0 20170811 - cached

x86-64 gcc (trunk) (Editor #1, Compiler #2)

x86-64 gcc (trunk) --std=c++17 -O3 -march=native -fasmath

```
11010 .LX0: .text // %s+ Intel Demangle
1 foo(unsigned int):
2 lea eax, [0+rdi*4]
3 ret
```

g++ (GCC-Explorer-Build) 8.0.0 20171206 (experimental) - cached

x86-64 clang (trunk) (Editor #1, Compiler #3)

x86-64 clang (trunk) --std=c++17 -O3 -march=native -fasmath

```
11010 .LX0: .text // %s+ Intel Demangle
1 foo(unsigned int): # @foo(unsigned int)
2 lea eax, [4+rdi]
3 ret
```

clang version 6.0.0 (trunk 319878) - cached

x86-64 MSVC 19.2017 RTW (Editor #1, Compiler #4)

x86-64 MSVC 19.2017 RTW /Ox /std:c++latest -fp:fast

```
11010 .LX0: .text // %s+ Intel Demangle
1 foo PROC
2 lea eax, DWORD PTR [rcx*4]
3 ret 0
4 foo ENDP
```

Microsoft (R) C/C++ Optimizing Compiler Version 19.10.25017 for x64 - cached

Примеры

Compiler Explorer

C++ Editor Diff View More

Share Other

C++ source #1

```
1 #include <cmath>
2
3 unsigned foo(unsigned x)
4 {
5     return x / 16;
6 }
7
8
```

x86-64 gcc 18 (Editor #1, Compiler #1)

x86-64 gcc 18 -O3 -march=native -fp-model fast=2 -fma -prec-div -prec-sqrt

```
11010
1 foo(unsigned int):
2     mov     edx, -858993459 #5.16
3     mulx   eax, ecx, edi #5.16
4     shr     eax, 3 #5.16
5     ret     #5.16
```

x86-64 clang (trunk) (Editor #1, Compiler #3)

x86-64 clang (trunk) --std=c++17 -O3 -march=native -fast-math

```
11010
1 foo(unsigned int): # @foo(unsigned int)
2     mov     ecx, edi
3     mov     eax, 3435973837
4     imul    rax, rcx
5     shr     rax, 35
6     ret
```

clang version 6.0.0 (trunk 319878) - cached

x86-64 MSVC 19 2017 RTW (Editor #1, Compiler #4)

x86-64 MSVC 19 2017 RTW /Ox /std:c++latest -fp:fast

```
11010
1 foo PROC
2     mov     eax, -858993459 ; ccccccdH
3     mul     ecx
4     shr     edx, 3
5     mov     eax, edx
6     ret     0
7 foo ENDP
```

gcc (GCC-Explorer-Build) 8.0.0 20171206 (experimental) - cached

Microsoft (R) C/C++ Optimizing Compiler Version 19.10.25017 for x64 - cached

Примеры

Compiler Explorer

C++ Editor Diff View More

Share Other

C++ source #1

```
1 #include <cmath>
2
3 unsigned foo(unsigned x)
4 {
5     unsigned int res = 0;
6     for(int i=0; i<1000; ++i)
7     {
8         res += i;
9     }
10    return res;
11 }
12
13
```

x86-64 gcc (trunk) (Editor #1, Compiler #1)

x86-64 gcc (trunk) --std=c++17 -O3 -march=native -fp-model fast=2 -fma -prec-div -prec-sqrt

11010

LX0: .text // /s+ Intel Demangle

```
3 vpxor ymm5, ymm5, ymm5 #5.22
4 vmovdqu ymm1, YMMWORD PTR .L_2il0floatpacket.0[rip] #8.9
5 vmovdqu ymm0, YMMWORD PTR .L_2il0floatpacket.1[rip] #8.9
6 vmovdqa ymm4, ymm5 #5.22
7 vmovdqa ymm3, ymm4 #5.22
8 vmovdqa ymm2, ymm3 #5.22
9 ..B1.2: # Preds ..B1.2 ..B1.1
10 vpaddq ymm2, ymm2, ymm0 #8.9
11 add eax, 32 #6.5
12 vpaddq ymm0, ymm0, ymm1 #8.9
13 vpaddq ymm6, ymm0, ymm1 #8.9
14 vpaddq ymm5, ymm5, ymm0 #8.9
15 vpaddq ymm7, ymm6, ymm1 #8.9
16 vpaddq ymm4, ymm4, ymm6 #8.9
17 vpaddq ymm3, ymm3, ymm7 #8.9
18 vpaddq ymm0, ymm7, ymm1 #8.9
19 cmp eax, 992 #6.5
20 jb ..B1.2 # Prob 99% #6.5
21 vpaddq ymm0, ymm2, ymm5 #5.22
22 vpaddq ymm1, ymm4, ymm3 #5.22
23 vpaddq ymm2, ymm0, ymm1 #5.22
24 vextractf128 xmm3, ymm2, 1 #5.22
25 vpaddq xmm4, xmm2, xmm3 #5.22
26 vpsrlq xmm5, xmm4, 8 #5.22
27 vpaddq xmm6, xmm4, xmm5 #5.22
28 vpsrlq xmm7, xmm6, 32 #5.22
29 vpaddq xmm8, xmm6, xmm7 #5.22
30 vmovd eax, xmm8 #5.22
31 add eax, 7964 #8.9
32 vzeroupper #10.12
33 ret #10.12
34 .L_2il0floatpacket.0:
35 .long 0x00000000,0x00000000,0x00000000,0x00000000,0x00000000
```

icc (ICC) 18.0.0 20170811 - 530ms

x86-64 gcc (trunk) (Editor #1, Compiler #2)

x86-64 gcc (trunk) --std=c++17 -O3 -march=native -fma -prec-div -prec-sqrt

11010

LX0: .text // /s+ Intel Demangle

```
1 foo(unsigned int):
2     mov     eax, 499500
3     ret
```

gcc (GCC-Explorer-Build) 8.0.0 20171206 (experimental) - 937ms

x86-64 clang (trunk) (Editor #1, Compiler #3)

x86-64 clang (trunk) --std=c++17 -O3 -march=native -fma -prec-div -prec-sqrt

11010

LX0: .text // /s+ Intel Demangle

```
1 foo(unsigned int): # @foo(unsigned int)
2     mov     eax, 499500
3     ret
```

clang version 6.0.0 (trunk 319878) - 1753ms

x86-64 MSVC 19 2017 RTW (Editor #1, Compiler #4)

x86-64 MSVC 19 2017 RTW /Ox /std:c++latest -fp:fast

11010

LX0: .text // /s+ Intel Demangle

```
1 foo PROC
2     xor     eax, eax
3     mov     ecx, eax
4     mov     edx, eax
5     mov     r8d, eax
6     mov     r9d, eax
7     npad    4
8 $LL4@foo:
9     inc     r9d
10    add     r8d, 2
11    add     edx, 3
12    add     r9d, ecx
13    add     r8d, ecx
14    add     edx, ecx
15    add     eax, ecx
16    add     ecx, 4
17    cmp     ecx, 1000 ; 000003e8H
18    jl      SHORT $LL4@foo
19    lea     ecx, DWORD PTR [rdx+r8]
20    add     ecx, r9d
21    add     eax, ecx
22    ret     0
23 foo ENDP
```

Microsoft (R) C/C++ Optimizing Compiler Version 19.10.25017 for x64 - 583ms

Примеры

Compiler Explorer

C++ Editor Diff View More

Share Other

C++ source #1

```
1 #include <cmath>
2
3 unsigned foo(unsigned x)
4 {
5     return (x << 4) - (x << 2) - (x << 1);
6 }
7
8
```

x86-64 gcc 18 (Editor #1, Compiler #1)

x86-64 gcc 18 -O3 -march=native -fp-model fast=2 -fma -prec-div -prec-sqrt

```
11010 .LX0: .text // /s+ Intel Demangle
1 foo(unsigned int):
2     mov     eax, edi
3     shl     eax, 4
4     lea     edx, DWORD PTR [rdi*4]
5     sub     eax, edx
6     add     edi, edi
7     sub     eax, edi
8     ret     5.40
```

x86-64 clang (trunk) (Editor #1, Compiler #3)

x86-64 clang (trunk) --std=c++17 -O3 -march=native -fast-math

```
11010 .LX0: .text // /s+ Intel Demangle
1 foo(unsigned int): # @foo(unsigned int)
2     add     edi, edi
3     lea     eax, [rdi + 4*rdi]
4     ret
```

clang version 6.0.0 (trunk 320133) -1731ms

x86-64 MSVC 19 2017 RTW (Editor #1, Compiler #4)

x86-64 MSVC 19 2017 RTW /Ox /std:c++latest -fp:fast

```
11010 .LX0: .text // /s+ Intel Demangle
1 foo PROC
2     mov     eax, ecx
3     lea     edx, DWORD PTR [rcx*4]
4     shl     eax, 4
5     add     ecx, ecx
6     sub     eax, edx
7     sub     eax, ecx
8     ret     0
9 foo ENDP
```

gcc (GCC-Explorer-Build) 8.0.0 20171208 (experimental) -1104ms

Microsoft (R) C/C++ Optimizing Compiler Version 19.10.25017 for x64 -623ms

11010 .LX0: .text // /s+ Intel Demangle

```
1 foo(unsigned int):
2     imul    eax, edi, 10
3     ret
```

Примеры

Compiler Explorer

C++ Editor Diff View More

Share Other

C++ source #1

```
1 #include <algorithm>
2
3 int foo(int a, int b)
4 {
5     return std::max(std::min(a,b), a);
6 }
7
8 //max(min(a, b), a) -> a
9
```

x86-64 gcc (trunk) (Editor #1, Compiler #1)

x86-64 gcc (trunk) --std=c++17 -O3 -march=native -fpmath=fast -fma -prec-div -prec-sqrt

```
1 foo(int, int):
2     cmp esi, edi #5.12
3     cmovge esi, edi #5.12
4     cmp esi, edi #5.12
5     cmovge edi, esi #5.12
6     mov eax, edi #5.12
7     ret #5.12
```

icc (ICC) 18.0.0 20170811 - 778ms

x86-64 gcc (trunk) (Editor #1, Compiler #2)

x86-64 gcc (trunk) --std=c++17 -O3 -march=native -fpmath=fast

```
1 foo(int, int):
2     mov eax, edi
3     ret
```

g++ (GCC-Explorer-Build) 8.0.0 20171208 (experimental) - 1022ms

x86-64 clang (trunk) (Editor #1, Compiler #3)

x86-64 clang (trunk) --std=c++17 -O3 -march=native -fpmath=fast

```
1 foo(int, int): # @foo(int, int)
2     mov eax, edi
3     ret
```

clang version 6.0.0 (trunk 320133) - 1042ms

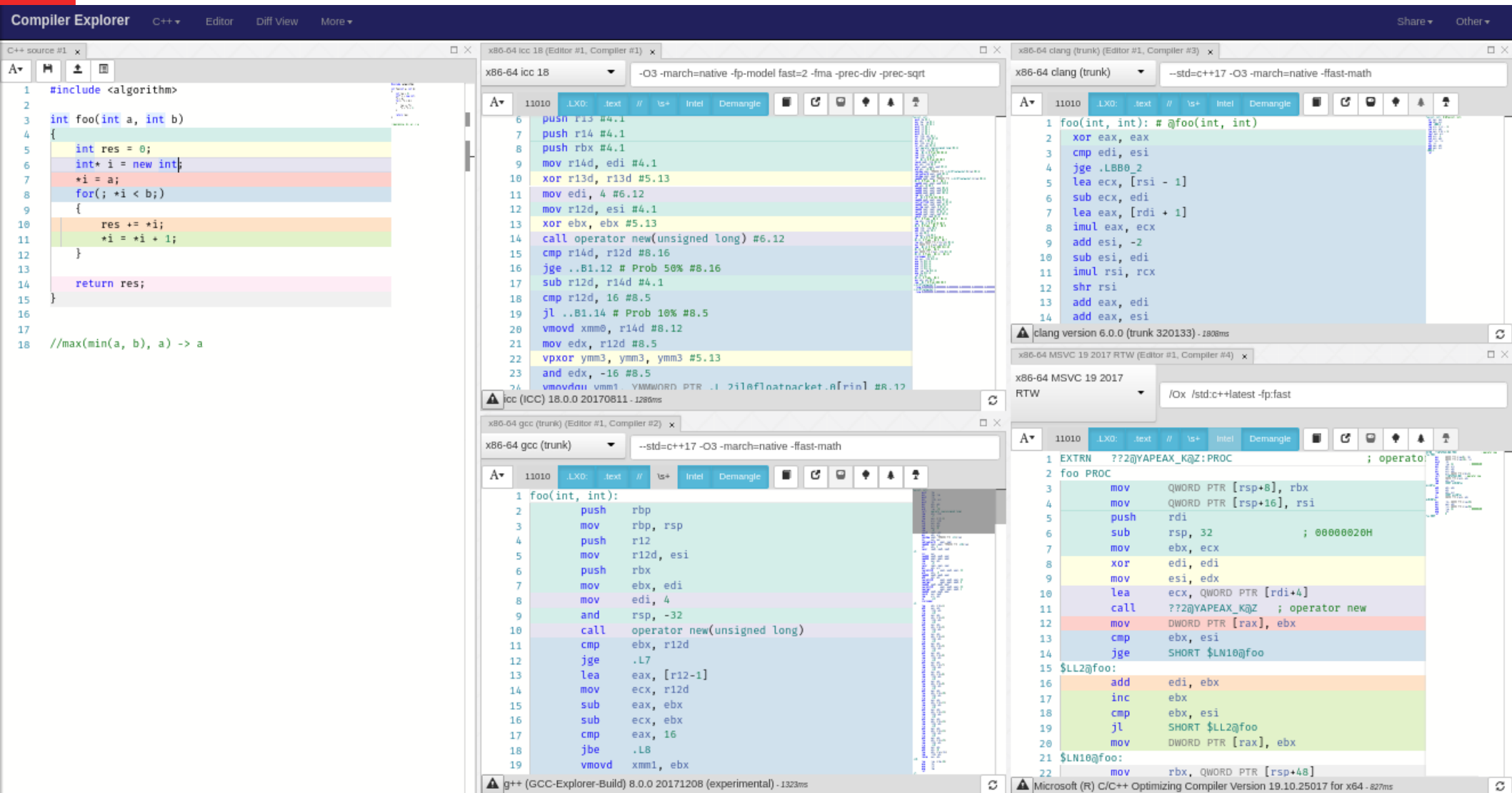
x86-64 MSVC 19.2017 RTW (Editor #1, Compiler #4)

x86-64 MSVC 19.2017 RTW /Ox /std:c++latest -fpmath=fast

```
1 std::max<int>, COMDAT PROC
2     mov     eax, DWORD PTR [rdx]
3     cmp     DWORD PTR [rcx], eax
4     cmovl   rcx, rdx
5     mov     rax, rcx
6     ret     0
7 std::max<int> ENDP
8 std::min<int>, COMDAT PROC
9     mov     eax, DWORD PTR [rcx]
10    cmp     DWORD PTR [rdx], eax
11    cmovl   rcx, rdx
12    mov     rax, rcx
13    ret     0
14 std::min<int> ENDP
15 a$ = 8
16 b$ = 16
17 foo PROC
18     cmp     edx, ecx
19     mov     DWORD PTR [rsp+16], edx
20     lea     rax, QWORD PTR b$[rsp]
21     mov     DWORD PTR [rsp+8], ecx
22     lea     r8, QWORD PTR a$[rsp]
23     cmovge  rax, r8
24     mov     eax, DWORD PTR [rax]
25     cmp     eax, ecx
26     cmovl   eax, ecx
27     ret     0
28 foo ENDP
```

Microsoft (R) C/C++ Optimizing Compiler Version 19.10.25017 for x64 - 980ms

Примеры



Бенчмарки

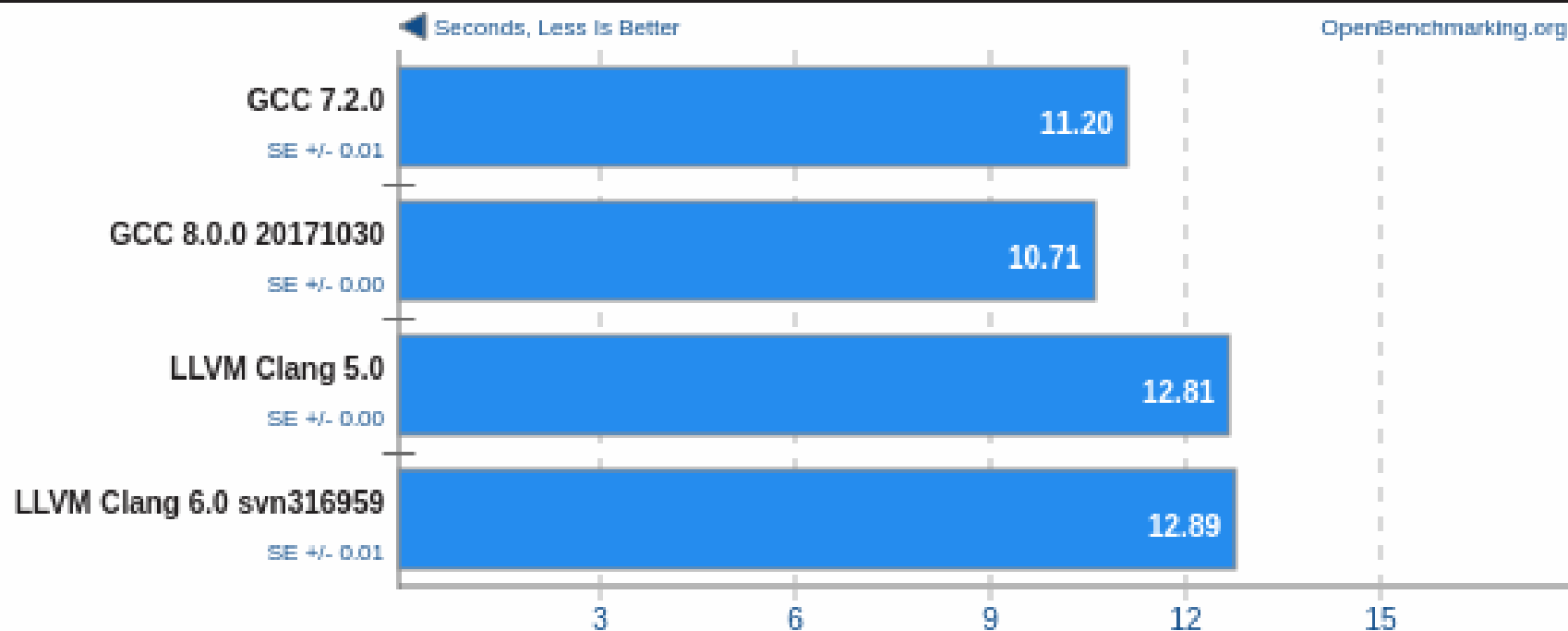
- Нет единого лидера
- Для каждого бенчмарка — свой лидер
- **Обычно** никто бенчмарки не гоняют с PGO и различными тонкими подстройками под конкретный компилятор



Немного бенчмарков

LAME MP3 Encoding v3.99.5

WAV To MP3



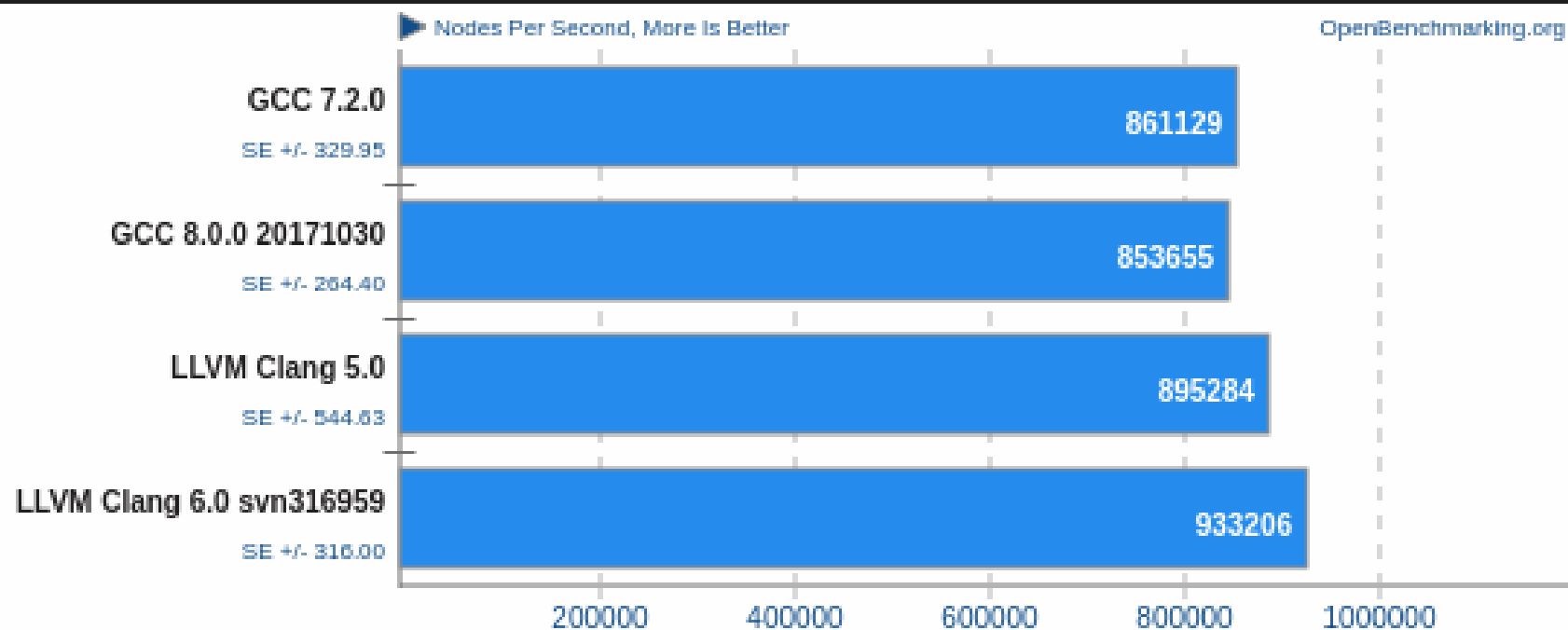
Phoronix Test Suite 7.4.0

1. (CC) gcc options: -O3 -fast-math -funroll-loops -fschedule-insns2 -fbranch-count-reg -fforce-addr -pipe -march=zincver1 -lm

Немного бенчмарков

TSCP v1.81

AI Chess Performance



Phoronix Test Suite 7.4.0

1. (CC) gcc options: -O3 -march=znver1 -march=native

Можно ли компиляторам верить?

- В повседневной жизни — безусловно
- Если в процессе профайлинга видите узкое место — смотрим, что там сгенерировалось конкретным компилятором под конкретной платформой
- Проверяем, оптимально ли с точки зрения C++ написан код
- Оптимизируем руками, если умеем (и репортируем performance issue в компиляторы)



Полезные ссылки

- openbenchmarking.org
- godbolt.org
- quick-bench.com
- CppCon 2017: Matt Godbolt “What Has My Compiler Done for Me Lately? Unbolting the Compiler's Lid”



Внимание!

Спасибо за внимание!