Intel® Compilers Overview and Optimizations

Intel® C++ Compiler Classic (ICC), Intel® NextGen Compiler (ICX)

Alina Shadrina

Technical Consulting Engineer

alina.shadrina@intel.com

Agenda

- Введение
  - Архитектура компилятора
  - Проблема оптимизации
  - Компиляторы Intel
- Опции оптимизации, IPO, PGO
- Воспроизводимость результатов вычислений с плавающей точкой
- Автоматическая векторизация, отчеты об оптимизации
- Поддержка openMP
- Производительность

Введение

Введение

Исходный код

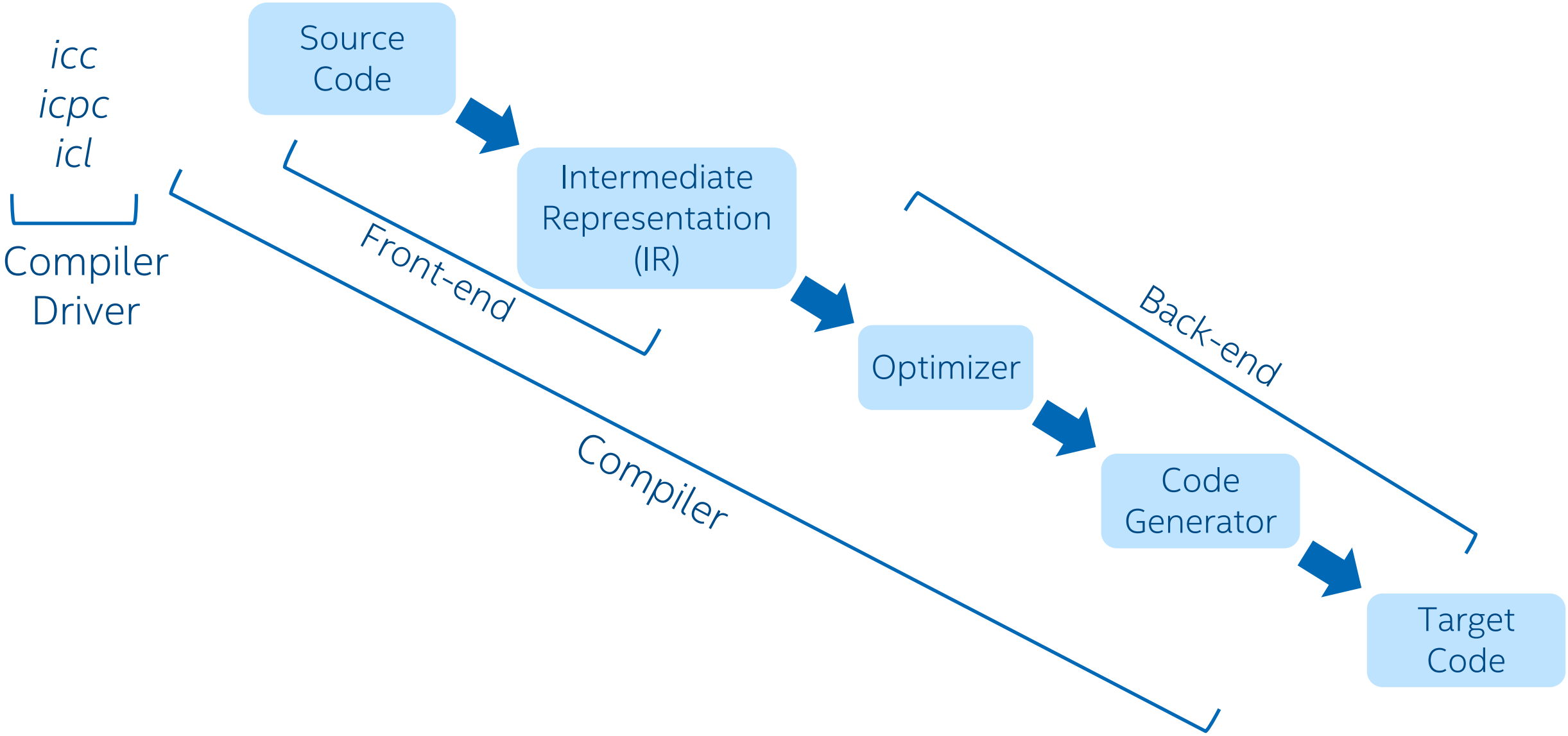
Высокоуровневый
Читаемый
...

Компилятор

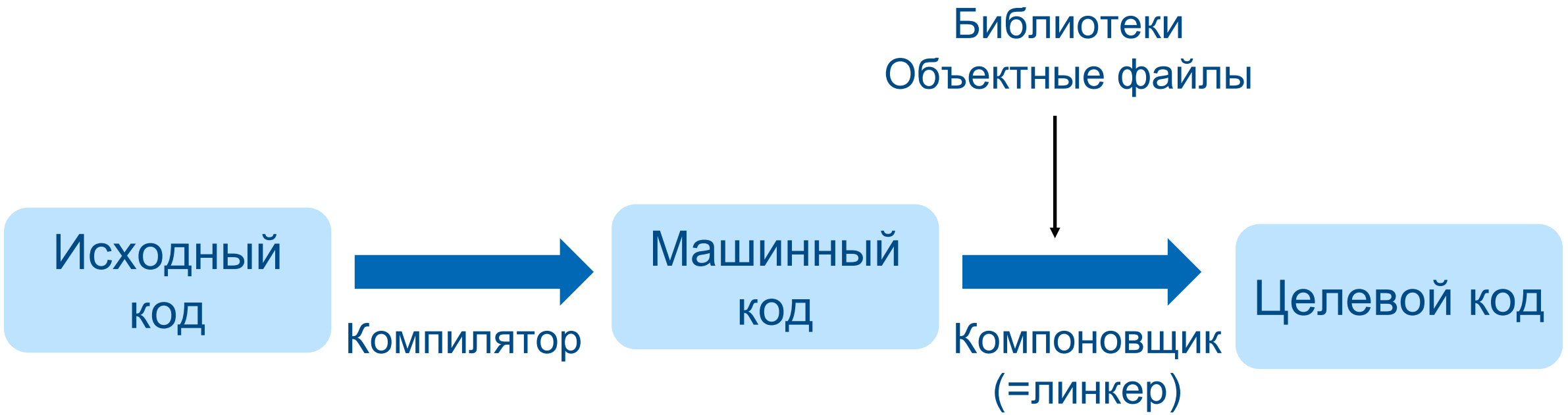
Целевой код

Низкоуровневый
Целевая архитектура
Сложен для чтения
...

Архитектура Компилятора



Архитектура Компилятора



Проблема оптимизации

- *Оптимизация* – попытка получить более эффективный код

НО?

Проблема оптимизации

- *Оптимизация* – попытка получить более эффективный код?

Нет **гарантированного** способа получить столь же быстрый код, выполняющий такие же задачи	НО	Должны сохранять смысл кода
Усложнение архитектуры процессоров => усложение оптимизации	НО	Должны решать разные задачи
NP-полные задачи => эвристики	НО	Время компиляции
Разные исходные данные	НО	Стоимость разработки и поддержки

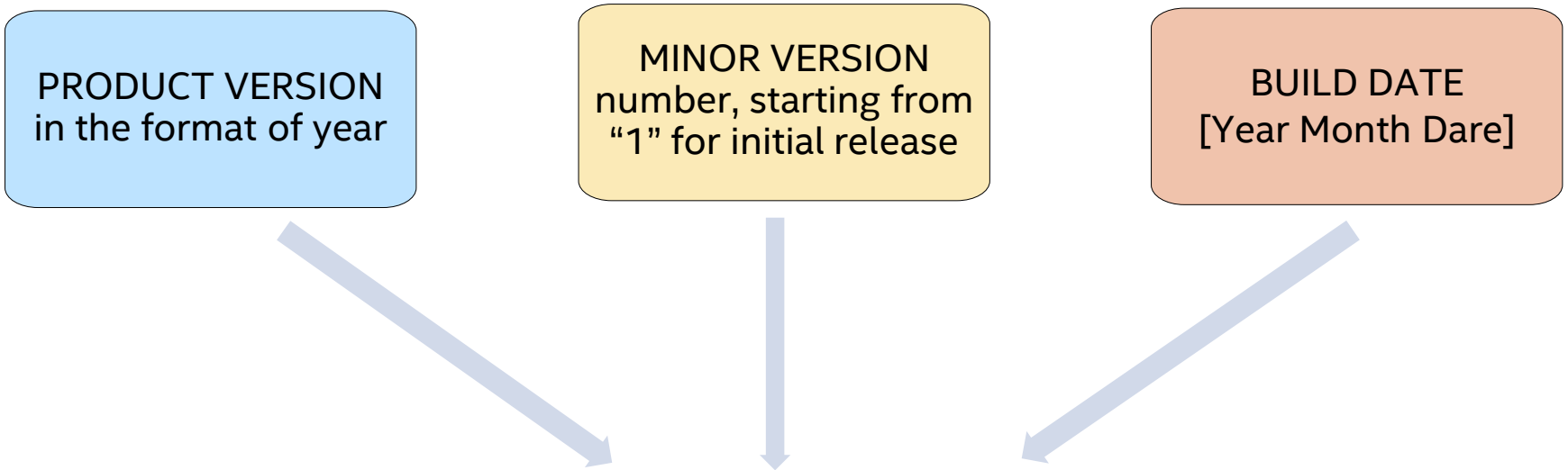
Intel® C++ Compilers

Intel Compiler	Target	OpenMP Support	OpenMP Offload Support	Included in oneAPI Toolkit
Intel® C++ Compiler, IL0 ***icc/icpc/icl***	CPU	Yes	No	HPC
Intel® oneAPI DPC++/C++ Compiler, LLVM ***dpcpp***	CPU, GPU, FPGA*	Yes	Yes	Base
Intel® oneAPI DPC++/C++ Compiler, LLVM ***icx/icpx***	CPU GPU*	Yes	Yes	Base

Бинарно совместимы

tinyurl.com/oneapi-standalone-components

Соглашение о версиях



icx --version

Intel(R) oneAPI DPC++/C++ Compiler 2021.3.0 (2021.3.0.20210619)

Опции Компилятора

Опции оптимизациии

	-Linux* /Windows* icx (icc)
Выключить оптимизации	-O0 /O0
Оптимизации скорости (без увеличения размера кода)	-O1 /O1
Оптимизации скорости (по умолчанию)	-O2 /O2
Высокоуровневые оптимизации циклов	-O3 /O3
Генерация информации для отладки (debug info)	-g /Zi
Многофайловые межпроцедурные оптимизации (Interprocedural optimizations, IPO)	-ipo /Qipo
Оптимизации основанные на профилировке (Profile-Guided Optimizations, PGO)	-fprofile-generate /Qprof-gen (-prof-gen) -fprofile-use /Qprof-use (-prof-use)
Оптимизации скорости	-fast /fast same as “-ipo -O3 -static -fp-model fast” (-ipo -O3 -no-prec-div –static -fp-model fast=2 -xHost)
Поддержка openMP	-fiopenmp –qopenmp /Qopenmp (-qopenmp)

Высокоуровневые оптимизации

-O0 нет оптимизаций; включает **–g (/Zi)** для отладки

-O1 скалярные оптимизации
не включаются оптимизации, которые увеличивают размер кода

-O2 **по умолчанию** (за исключением случая, когда используется -g)
включает автовекторизацию, некоторые оптимизации циклов (unrolling, interchange), инлайнинг;
после начальной отладки с O0, рекомендуется начинать оптимизации с этой опции

-O3 более агрессивные оптимизации цикла
Включает блокирование кэшей, префетчинг, …
подходит для приложений, в которых основные вычисления происходят в циклах, много вычислений с плавающей точкой, большие объемы данных

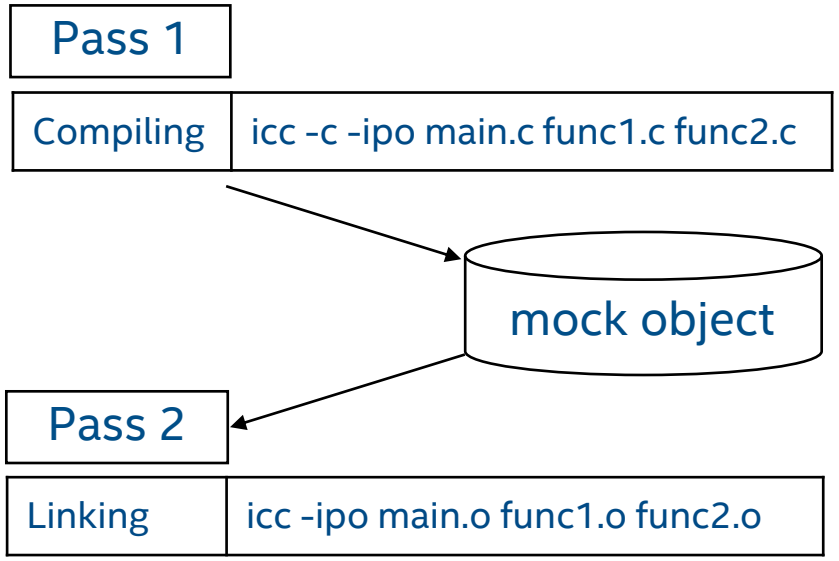
Межпроцедурные Оптимизации
Interprocedural optimizations (IPO)

Межпроцедурные оптимизации (IPO)

Interprocedural optimizations, Multi-pass Optimization

- Межпроцедурная оптимизация выполняет статический анализ нашего приложения
- Включает:
  - Встраивание (inlining) процедур (уменьшение накладных расходов на вызовы функций)
  - Удаление мертвого кода (dead code elimination), подстановка констант (constant propagation) и изменение порядка процедур (procedure reordering)
  - Улучшает оптимизацию в сочетании с другими оптимизациями
  - Большая часть ip (включая встраивание) включена по умолчанию в опции O2.

Отчет об inlining посмотреть с помощью **-qopt-report-phase=ipo**

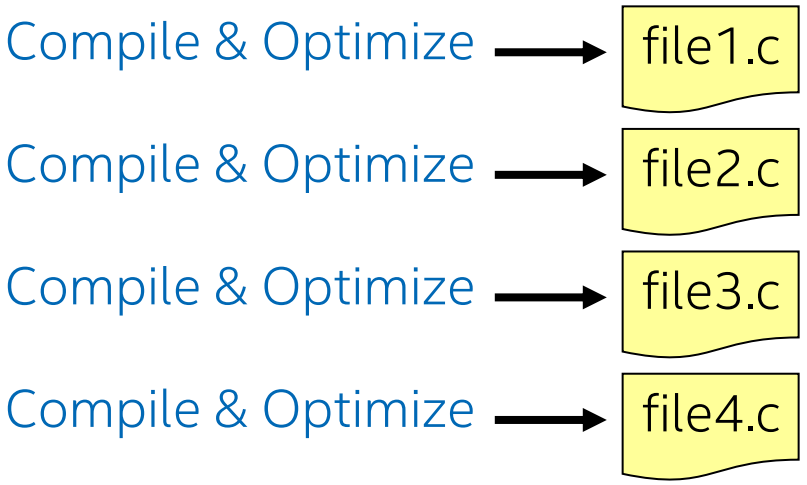


Межпроцелдурные оптимизации (IPO)

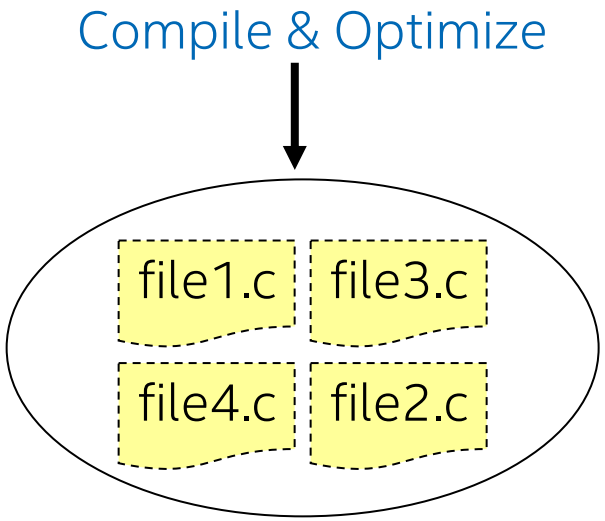
Расширение границ оптимизаций

icc	-ip	Только между модулями одного файла исходного кода
	-ipo	Модули многих файлов
icx	-ipo (отображается в -flto)	Оптимизация во время линковки (Link Time Optimization)

Без IPO



С IPO



Link Time Optimization (LTO): tinyurl.com/clang-lto

Оптимизации основанные на профилировке
Profile-Guided Optimizations (PGO)

Profile-Guided Optimizations (PGO)

- Статический анализ оставляет много вопросов, например:
  - Как часто x > y?
  - Каков размер count?
  - Какой код исполняется чаще?

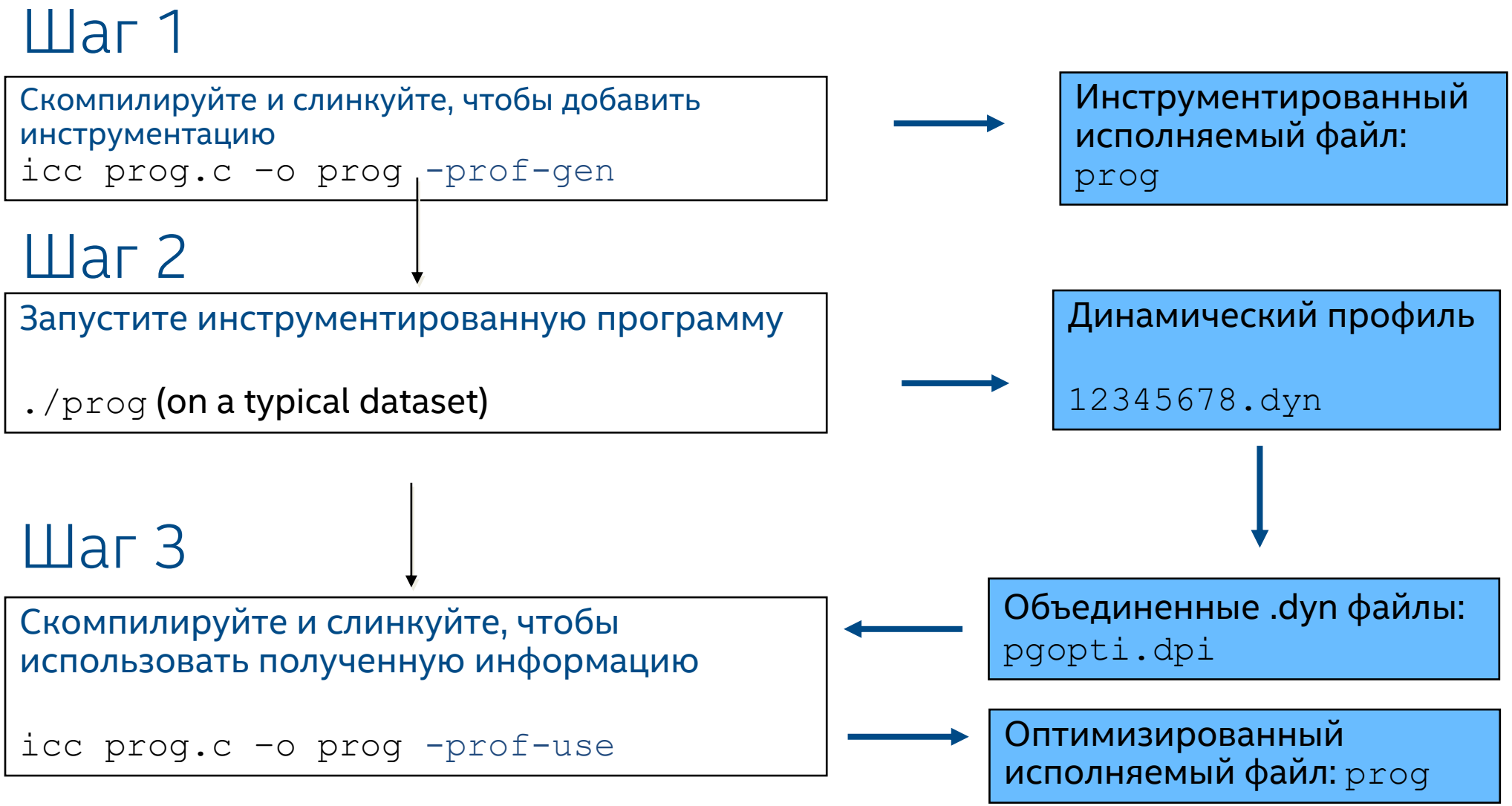
```
if (x > y)
     do_this();
 else
     do that();
```

```
for(i=0; i<count; ++i)
do_work();
```

- Можно использовать информацию времени исполнения (runtime)
- Улучшения с PGO:
  - Более точное предсказание переходов (branch prediction)
  - Базовое перемещение блоков для улучшения поведения кэша инструкций
  - Подстановка (inline) функций (помощь IPO)
  - Может оптимизировать порядок функций (function ordering)
  - Оптимизация оператора switch
  - Улучшенная векторизация

Использование PGO в ICC

Профилировка с инструментацией



Воспроизводимость результатов вычислений с плавающей точкой Floating-Point (FP)

Floating-Point (FP) Programming Objectives

- Точность
  - Результаты, близкие к верному ответу
    - Измеряется относительная ошибка
- Производительность
  - Наиболее эффективный код
- Воспроизводимость
  - Стабильный результат
    - От запуска к запуску
    - С разным набором опций
    - С разными компиляторами
    - На разных платформах

Эти целы обычно конфликтуют!
Есть опции компилятора, которые позволяют смещать приоритеты
Разные компиляторы имеют разное поведение по умолчанию

Контроль воспроизводимости результатов

-fp-model

- fast [=1] оптимизации, влияющие на точность (default)
- fast=2 некоторые дополнительные оптимизации
- precise только оптимизации, НЕ влияющие на точность
- strict precise + except (строгое следование семантике FP-исключений) + отключить fma

Исключения

Floating Point Exceptions

Когда возникает (**raised**) **исключение**

- Ничего не делать (Default)
- Обработать (trap) (`SIGFPE`)

Типы исключений:

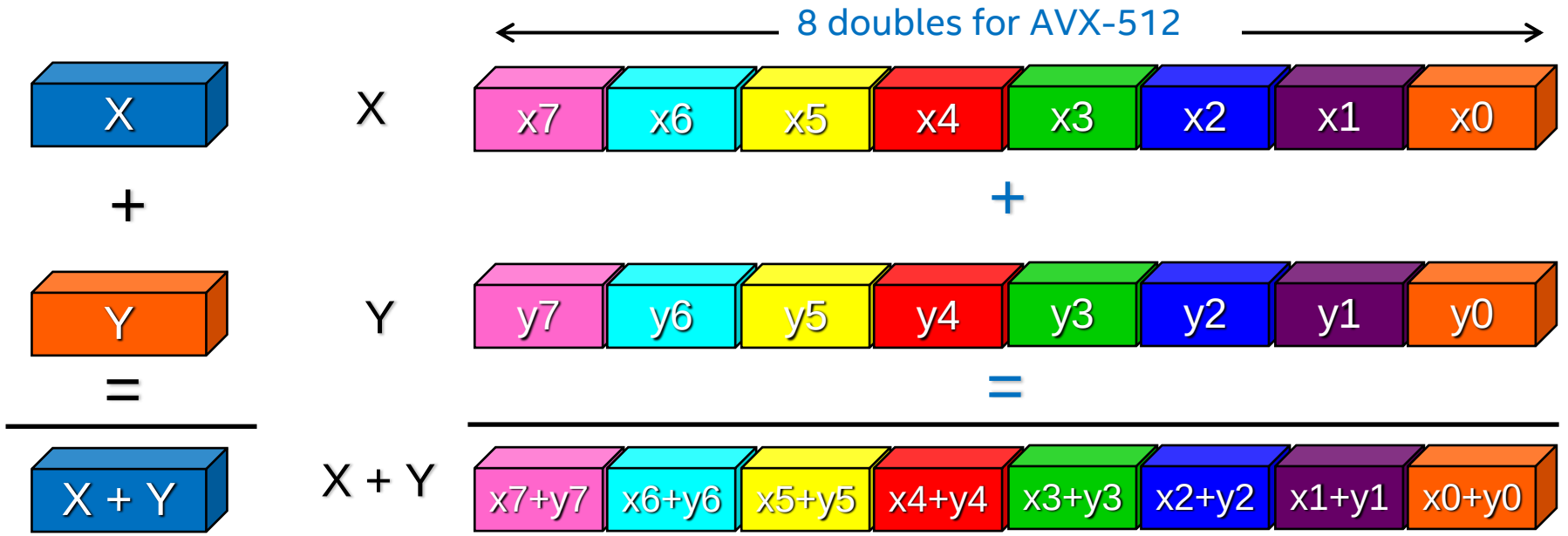
- Invalid Operation — Для заданных операндов невозможно выполнить операцию NaN
- Division by Zero — Деление конечного ненулевого числа на ноль *+Inf , -Inf*
- Overflow — Результат нельзя представить как конечное число заданной *0, +Inf, -Inf*
- Underflow — Промежуточный результат операции или результат округления слишком мал — Ноль или Неточное маленькое значение
- Inexact — Результат округления не точен или было исключение Overflow

Автоматическая векторизация
Auto-vectorization

SIMD: Single Instruction, Multiple Data

```
for (i=0; i<n; i++) z[i] = x[i] + y[i];
```

- Скалярный режим
  - Одна инструкция – один результат
  - E.g. vadd**ss**, (vadd**sd**)
- Векторный режим (SIMD)
  - Одна инструкция – множественный результат
  - E.g. vadd**ps**, (vadd**pd**)

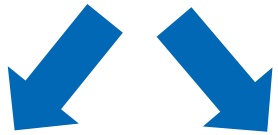


Автоматическая векторизация

```
#include <math.h>
void foo (float * theta, float * sth)  {
  int i;
  for (i = 0; i < 512; i++)
       sth[i] = sin(theta[i]+3.1415927);
}
```

-vec

-no-vec



```
vmovups   zmm1, ZMMWORD PTR [r12+rdi*4]                 #5.21
vextractf64x4 ymm2, zmm1, 1                             #5.21
vcvtps2pd zmm3, ymm1                                    #5.21
vaddpd    zmm0, zmm16, zmm3                             #5.30
vcvtps2pd zmm4, ymm2                                    #5.21
vaddpd    zmm17, zmm16, zmm4                            #5.30
call      QWORD PTR [__svml_sin8@GOTPCREL+rip]          #5.17
vmovaps   zmm18, zmm0                                   #5.17
vmovaps   zmm0, zmm17                                   #5.17
call      QWORD PTR [__svml_sin8@GOTPCREL+rip]          #5.17
vcvtpd2ps ymm2, zmm18                                   #5.17
vcvtpd2ps ymm1, zmm0                                    #5.17
vinsertf64x4 zmm3, zmm2, ymm1, 1                        #5.17
vmovups   ZMMWORD PTR [rsi+rdi*4], zmm3                 #5.8
```

```
vcvtss2sd xmm16, xmm16, DWORD PTR [r12+r14*4]           #5.17
vaddsd    xmm0, xmm16, QWORD PTR .L_2il0floatpacket.1[rip] #5.17
call      sin                                           #5.17
vcvtsd2ss xmm0, xmm0, xmm0                              #5.8
vmovss    DWORD PTR [r13+r14*4], xmm0
```

godbolt.org/z/dWvEh7GGc

Базовые опции векторизации

-x<code>

- Может включать оптимизаци, спецфичные для Intel CPU
- Добавляется проверка CPU:
  Приложение выдаст ошибку в случае, если нет поддержки указанного набора инструкций, или если приложение исполняется на процессоре другого вендора

- <code> - целевой набор инструкций, в виде:

- Имени микроархитектуры: BROADWELL, HASWELL, IVYBRIDGE, KNL, KNM, SANDYBRIDGE, SILVERMONT, SKYLAKE, SKYLAKE-AVX512
- SIMD-расширений: CORE-AVX512, CORE-AVX2, CORE-AVX-I, AVX, SSE4.2, etc.
- Например: icx -xCORE-AVX2 test.c

  icx -xSKYLAKE test.c

Базовые опции векторизации

-ax<code>

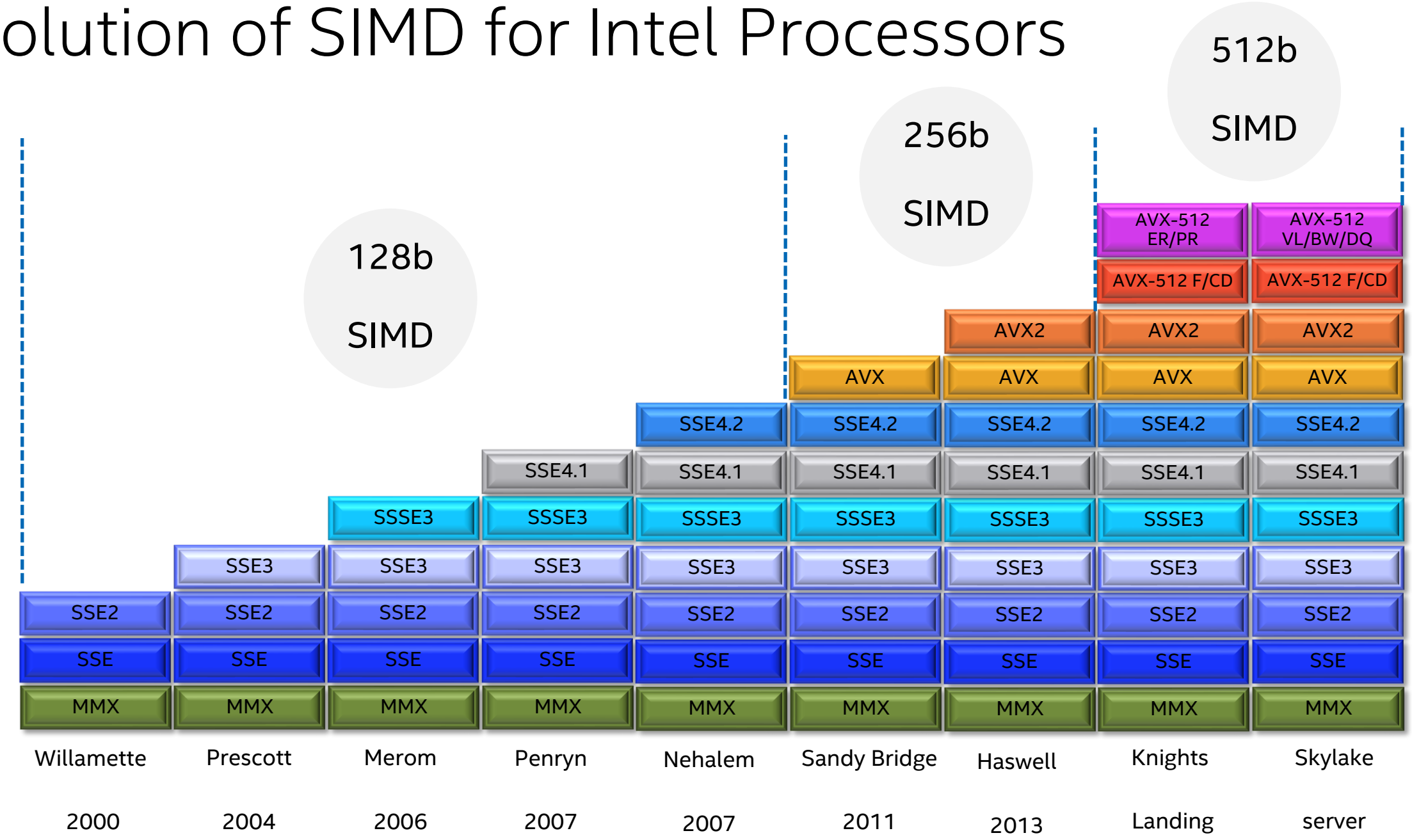
- Две ветки кода: ветка по умолчанию (baseline) и оптимизированная под заданный процессор
- Оптимизация под процессоры Intel <code>

-m<code>

- Нет проверки исполнения на процессорах Intel
- Нет оптимизаций под процессоры Intel
- Приложение оптимизируется одинаково как для процессоров Intel, так и других вендоров в рамках заданных SIMD-наборов инструкций
- Отсутствие проверки может приводить к падению в том случае, если заданный набор инструкций не поддерживается

- `-xHost`

Evolution of SIMD for Intel Processors



Отчёт об оптимизации
Optimization Report

Отчёт об оптимизации

- -qopt-report[=n]: сгенерировать отчёт
  - n: (опционально) уровень детализации отчёта
  - **ICC:** (0-5) Уровень 5 даёт наибольшую детализацию
  - **ICX:** (0-3) Уровень 3 включает Loop Optimizations, OpenMP, Register Allocation

- -qopt-report-phase[=list] – выбрать одну или несколько фаз оптимизации для отчёта
  - loop - оптимизация вложенных циклов
  - vec - векторизация
  - par – автоматическое распараллеливание
  - all – все фазы
- -qopt-report-filter=string выбрать отдельную функцию или модуль приложения для отчёта

Пока нет Поддержки В ICX

OpenMP* Support

OpenMP

- Полная поддержка OpenMP* 5.0 TR4 + частично OpenMP 5.1
- Опции

-fiopenmp

- OMP оптимизированный Intel
- -fopenmp - Clang* OMP
- -qopenmp отображается в -fiopenmp

-fopenmp-targets=spir64

- Опция для поддержки OMP Offload (исполнение кода на акселераторах)
- Генерирует SPIRV для кернелов

Get Started with OpenMP* Offload Feature to GPU: tinyurl.com/intel-openmp-offload

Производительность
Performance

Идеальный набор опций компилятора?

Зависит от

- Вашей задачи, железа, операционной системы, версии компилятора и тд
- Пример бенчмарков:

ICC:

SPECint®_rate_base_2017: *-xCORE-AVX512 -ipo -O3 -no-prec-div -qopt-mem-layout-trans=4*

SPECfp®_rate_base_2017: *-xCORE-AVX512 -ipo -O3 -no-prec-div -qopt-prefetch -ffinite-math-only -qopt-mem-layout-trans=4*

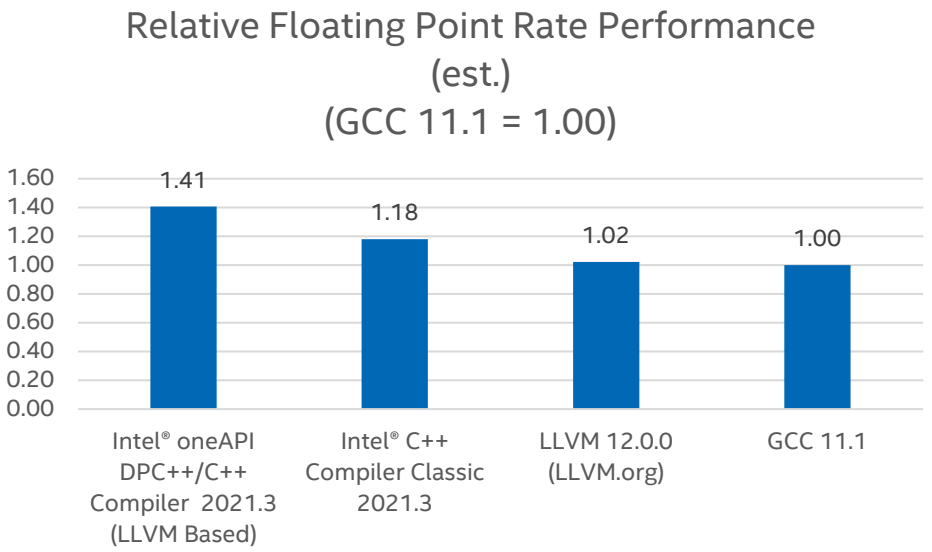
SPEC HPC2021: -qopt-zmm-usage=high -Ofast -xCORE-AVX512 -qopenmp -ipo -qopt-multiple-gather-scatter-by-shuffles -fimf-precision=low:sin,sqrt [for IFORT: -align array64byte -nostandard-realloc-lhs]

ICX:

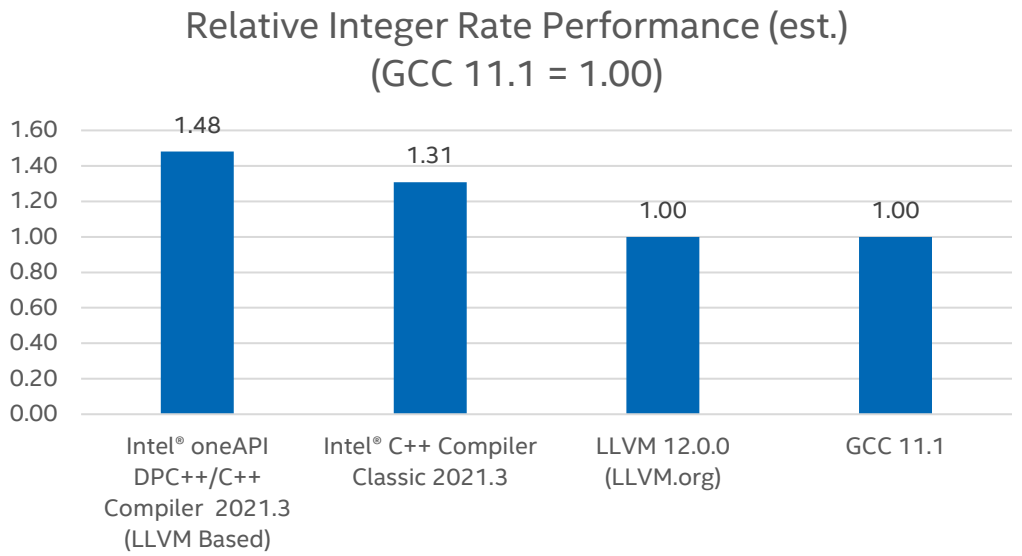
SPEC HPC2021: -mprefer-vector-width=512 -Ofast -xCORE-AVX512 -ffast-math -fiopenmp -flto -fimf-precision=low:sin,sqrt -funroll-loops [for IFX: -align array64byte -nostandard-realloc-lhs]

Intel® Compilers Boost C++ Application Performance on Linux*

Performance advantage relative to other compilers on Intel® Xeon Platinum 8280 Processor



Estimated: internal measurement of the geometric mean of the C/C++ workloads from the SPECrate* 2017 Floating Point suite



Estimated: internal measurement of the geometric mean of the C/C++ workloads from the SPECrate* 2017 Integer suite

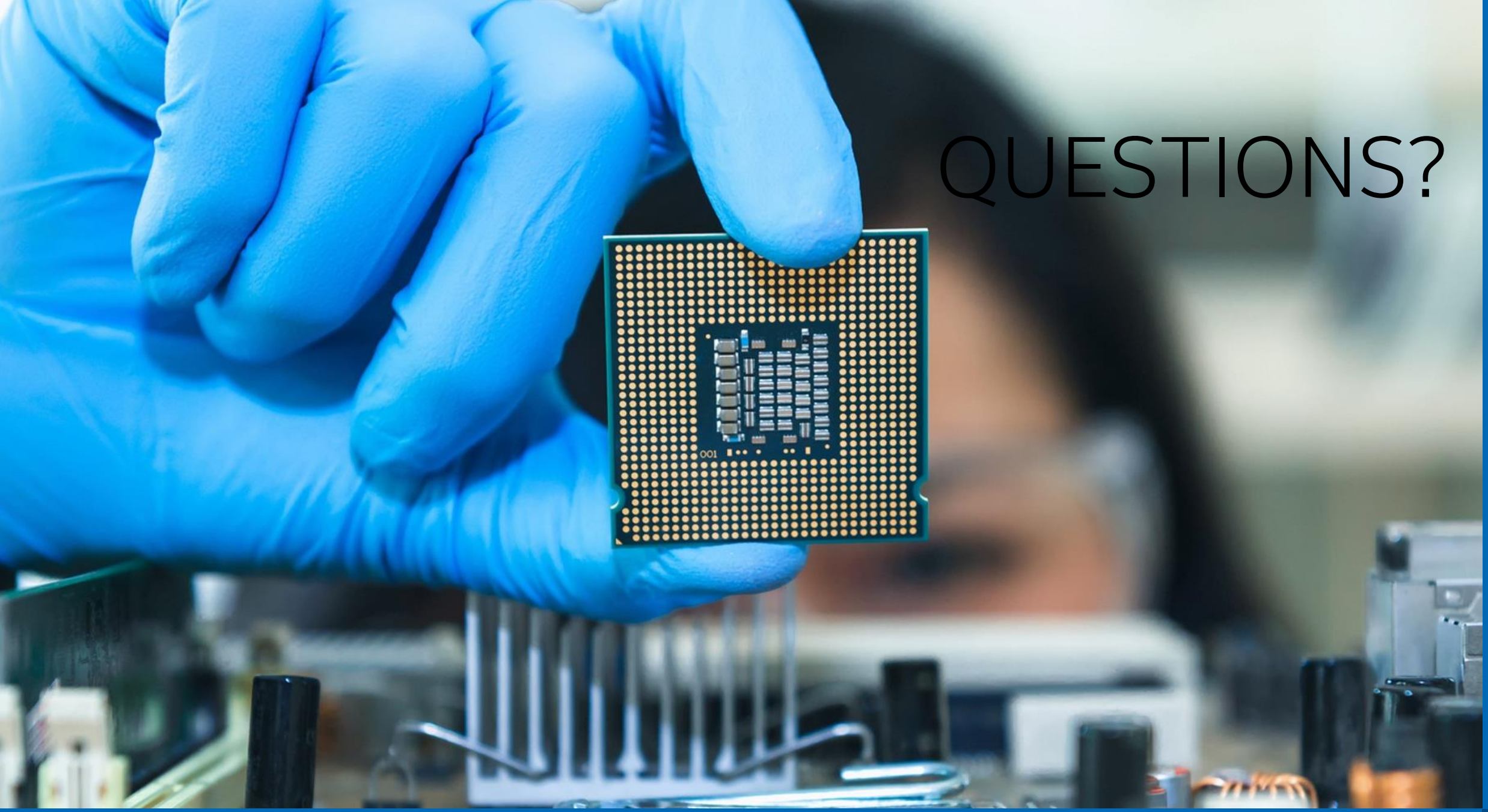
Performance varies by use, configuration, and other factors. Learn more at www.intel.com/PerformanceIndex.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See configuration disclosure for details. No product or component can be absolutely secure.

Your costs and results may vary. Intel technologies may require enabled hardware, software, or service activation.

More information on the SPEC benchmarks can be found at: http://www.spec.org

Configuration: Testing by Intel as of Jun 10, 2021 -Intel(R) Xeon(R) Platinum 8380 CPU @ 2.30GHz, 2 socket, Hyper Thread on, Turbo on, 32G x12 DDR4 3200 (1DPC). Software: Intel(R) oneAPI DPC++/C++ Compiler for applications running on Intel(R) 64, Version 2021.2.0 Build 20210317, Intel(R) C++ Compiler Classic for applications running on Intel(R) 64, Version 2021.3.0 Build 20210604_000000, GCC 11.1, Clang/LLVM 12.0.0. Red Hat Enterprise Linux release 8.2 (Ootpa), 4.18.0-193.el8.x86_64. SPECint®_rate_base_2017 compiler switches: Intel(R) oneAPI DPC++/C++ Compiler for applications running on Intel(R) 64, Version 2021.2.0 Build 20210317: -xCORE-AVX512 -O3 -ffast-math -flto -mfpmath=sse -funroll-loops -qopt-mem-layout-trans=4. qkmalloc used. Intel(R) C++ Compiler Classic for applications running on Intel(R) 64, Version 2021.3.0 Build 20210604_000000: -xCORE-AVX512 -ipo -O3 -no-prec-div -qopt-mem-layout-trans=4 -qopt-multiple-gather-scatter-by-shuffles. qkmalloc used. GCC 11.1 -march=znver1 -mfpmath=sse -Ofast -funroll-loops -flto. LLVM 12.0.0: -march=core-avx2 -mfpmath=sse -Ofast -funroll-loops –flto. SPECfp®_rate_base_2017 compiler switches: Intel(R) oneAPI DPC++/C++ Compiler for applications running on Intel(R) 64, Version 2021.1 Build 20201113: -xCORE-AVX512 -Ofast -ffast-math -flto -mfpmath=sse -funroll-loops -qopt-mem-layout-trans=4. jemalloc 5.0.1 used.Intel(R) C++ Compiler Classic for applications running on Intel(R) 64, Version 2021.3.0 Build 20210604_000000:-xCORE-AVX512 -ipo -O3 -no-prec-div -qopt-prefetch -ffinite-math-only -qopt-multiple-gather-scatter-by-shuffles -qopt-mem-layout-trans=4. jemalloc 5.0.1 used. GCC 11.1: -march=skylake-avx512 -mfpmath=sse -Ofast -fno-associative-math -funroll-loops -flto. jemalloc 5.0.1 used. LLVM 12.0.0: - march=znver1 -mfpmath=sse -Ofast -funroll-loops –flto. jemalloc 5.0.1 used.


QUESTIONS?

Notices & Disclaimers

- This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice.
- Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at intel.com, or from the OEM or retailer.
- Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit www.intel.com/benchmarks.
- INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS". NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.
-

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.
Notice revision #20110804


intel