Обзор возможностей LLVM для создания трансляторов, компиляторов, оптимизаторов

Зиновий Нис

Intel Corp.

I. Обзорная экскурсия

Есть у меня шестерка слуг,
Проворных, удалых.

И все, что вижу я вокруг,
Все знаю я от них.

Они по знаку моему
Являются в нужде.

Зовут их: ***Как*** *и* ***Почему****,*
Кто*,* ***Что****,* ***Когда*** *и* ***Где****.*

(Р. Киплинг)

Optimization Notice
Copyright © 2014, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

Что

LLVM – low level virtual machine

- Программный продукт с открытым исходным кодом
- Система анализа, трансформаций и исполнения, использующая RISC-подобное внутреннее представление – LLVM IR
- LLVM – это группа проектов:
  - компилятор
  - оптимизатор
  - анализаторы
  - библиотеки для встраивания в свои проекты
  - тесты

Copyright © 2014, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

Почему

- Удобно расширять
  - LLVM написан на C++ самого последнего стандарта в отличие от gcc, написанного в основном на C
- BSD-подобная лицензия LLVM
  - позволяет использовать код и любые его модификации в любых проприетарных проектах без раскрытия изменений
- LLVM IR
  - удобное и интуитивно понятное внутреннее представление
- Изначально нацелен на интеграцию с IDE
  - контекстно-зависимые автодополнения, рефакторинг, предупреждения, советы, раскраска сообщений об ошибках
- Многоплатформенность
  - от микроконтроллеров до суперкомпьютеров, от bare-metal до Windows и *nix

Copyright © 2014, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

Кто

- Apple
  - OpenGL
  - XCode
  - Swift
- ARM
  - Оптимизирующий компилятор
- Google
  - PNaCl
  - Верификация кода
  - Верификация ABI
- Intel
  - Новые инструкции и архитектуры
  - Поддержка OpenMP
- Nvidia
  - CUDA
- Samsung
  - JS
- Sony
  - PlayStation
- Энтузиасты

Copyright © 2014, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

Когда

1985 г.	Ричард Столлман выпустил первую версию GCC
2000 г.	Крисом Латтнером и Викрамом Адве создана ранняя версия LLVM в университете штата Иллионойс
2003 г.	выход LLVM 1.0
2005 г.	Apple наняла Криса Латтнера для развития LLVM
2005-2015 г.	взрывной рост и взросление проекта: добавление множества новых целевых архитектур, оптимизаций, областей применения, участников
2014 г.	основание The LLVM Foundation с целью дальнейшего развития проекта, получения независимости от компаний, организации мероприятий, решению инфраструктурных проблем

Copyright © 2014, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

Где

- Исходные тексты и предсобранные бинарники для 10+ архитектур:
  - http://llvm.org/releases/download.html
- SVN-репозиторий
  - svn co http://llvm.org/svn/llvm-project/llvm/trunk llvm
  - svn co http://llvm.org/svn/llvm-project/cfe/trunk clang
- GIT-репозиторий
  - git clone http://llvm.org/git/llvm.git
  - git clone http://llvm.org/git/clang.git

Copyright © 2014, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

Как

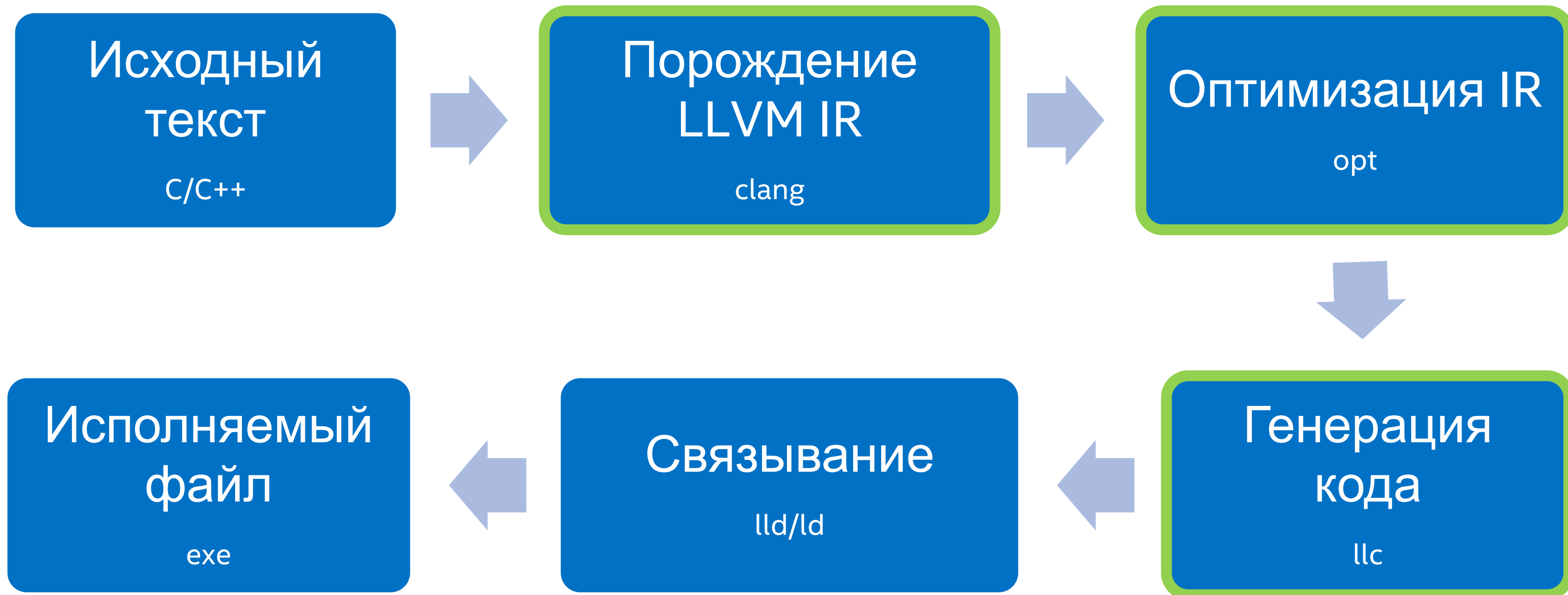
1. Загрузка исходных текстов LLVM из SVN- или GIT-репозитория
2. Сборка одним из двух способов
   - ./configure <опции конфигурирования> && make && make install
   - или cmake <опции конфигурирования> && make
   - для MS Windows + Visual Studio удобнее использовать CMake
3. Тестирование собранного проекта:
   - make check-all
4. Использование
   - ./clang –O3 main.c utils.c –lm –Wall

Copyright © 2014, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

II. Под капотом

Copyright © 2014, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

Схема работы LLVM



Copyright © 2014, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

Основные утилиты и библиотеки

- clang/clang++
  - фронтэнд, порождает LLVM IR из программы на C/C++
- opt
  - оптимизатор LLVM IR в LLVM IR
- llc
  - компилятор (бэкенд) LLVM IR в объектные файлы целевой архитектуры
- lld
  - линкер, связывание объектных файлов в исполняемый

Optimization Notice
Copyright © 2014, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

LLVM IR

- Все оптимизации проводятся в LLVM на уровне внутреннего представления – LLVM IR
- LLVM IR
  - низкоуровневое представление программы
  - трехадресный код с низкоуровневыми RISC-операциями – выглядит как ассемблер
  - SSA-форма
  - неограниченное количество виртуальных регистров («переменных»)
  - строгое разбиение тела функции на базовые блоки
  - типобезопасное
  - полно для представления C/C++
  - *во многом* не зависит от платформы

Copyright © 2014, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

SSA-представление

- Static Single Assignment – форма однократного статического присваивания
- Активно используется в компиляторах уже 30 лет
- Каждой переменной можно присвоить значение лишь один раз
- Последующие присвоения – уже следующим версиям переменной

```
x = 1; y = 2; x = y  →
x_1=1; y_1 = 2; x_2 = y_1
```

- Выбор версии осуществляется через phi-функцию

```
if(cond) x = 5; else x = 6; z = x; →
if(cond) x_1=5; else x_2=6; z_1=phi(x_1, x_2);
```

- SSA-представление упрощает проведение многих оптимизаций

Copyright © 2014, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

Создание IR фронтендом

- Стандартный фронтенд в LLVM – clang (/ˈklæŋ/)
- Это на самом деле драйвер, то есть фронтэнд, оптимизатор и компилятор в одной программе
- Понимает C и C++ (включая встроенный ассемблер)
- Поддерживает почти на 100% все новые возможности последних стандартов языков
- Порождает корректный LLVM IR, готовый для дальнейших оптимизаций
- Сам оптимизацией не занимается

Optimization Notice
Copyright © 2014, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

Создание IR фронтендом. Продолжение

```c
#include <stdio.h>
int foo(int i, int j)
{
	if (j > 0)
		return i * (1 << j);
	printf("j<=0\n");
	return 0;
}
```

```
$> clang test.c -emit-llvm –S –O0
```


Copyright © 2014, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

Создание IR фронтендом. Окончание

$> clang test.c **-emit-llvm** –S –O0

```
@.str = private unnamed_addr constant [6 x i8] c"j<=0\0A\00", align 1
define i32 @foo(i32 %i, i32 %j) #0 {
entry:
  %retval = alloca i32, align 4
  %i.addr = alloca i32, align 4
  %j.addr = alloca i32, align 4
  store i32 %i, i32* %i.addr, align 4
  store i32 %j, i32* %j.addr, align 4
  %0 = load i32, i32* %j.addr, align 4
  %cmp = icmp sgt i32 %0, 0
  br i1 %cmp, label %if.then, label %if.end
if.then:                                          ; preds = %entry
  %1 = load i32, i32* %i.addr, align 4
  %2 = load i32, i32* %j.addr, align 4
  %shl = shl i32 1, %2
  %mul = mul nsw i32 %1, %shl
  store i32 %mul, i32* %retval
  br label %return
if.end:                                           ; preds = %entry
  %call = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([6 x i8], [6 x i8]* @.str, i32 0, i32 0))
  store i32 0, i32* %retval
  br label %return
return:                                           ; preds = %if.end, %if.then
  %3 = load i32, i32* %retval
  ret i32 %3
}
declare i32 @printf(i8*, ...) #1
```

```
#include <stdio.h>
int foo(int i, int j)
{
        if (j > 0)
                return i * (1 << j);
        printf("j<=0\n");
        return 0;
}
```


Optimization Notice
Copyright © 2014, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

Оптимизация IR. Проходы

- Оптимизатор работает проходами (pass)
  - «проход» - это обход сущностей IR программы
  - например, обход модулей, функций, циклов, базовых блоков
- Проход
  - аналитический: собирает информацию о сущности
  - трансформирующий: преобразует сущность
- $> opt test.bc –o test.out.bc –dce –constprop
  - можно явно указывать необходимые проходы
  - либо задавать преопределенные группы проходов: -O0, -O1, -O2, -O3
- Управляет списком проходов объект класса PassManager
  - хранит список проходов и вызывает их (объекты классов xxxPass, например LoopPass)
  - не ищет оптимальных последовательностей проходов

Optimization Notice
Copyright © 2014, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

Оптимизация IR. Пример

```
#include <stdio.h>
int foo(int i, int j)
{
        if (j > 0)
                return i * (1 << j);
        printf("j<=0\n");
        return 0;
}
```

$> opt test.ll –O3 –o - -S

```
@str = private unnamed_addr constant [5 x i8] c"j<=0\00"
define i32 @foo(i32 %i, i32 %j) #0 {
entry:
  %cmp = icmp sgt i32 %j, 0
  br i1 %cmp, label %if.then, label %if.end
if.then:                                          ; preds = %entry
  %mul = shl i32 %i, %j
  br label %return
if.end:                                           ; preds = %entry
  %puts = tail call i32
                @puts(i8* getelementptr inbounds ([5 x i8], [5 x i8]* @str, i64 0, i64 0))
  br label %return
return:                                           ; preds = %if.end, %if.then
  %retval.0 = phi i32 [ %mul, %if.then ], [ 0, %if.end ]
  ret i32 %retval.0
}
; Function Attrs: nounwind
declare i32 @puts(i8* nocapture readonly) #1
```


Copyright © 2014, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

Оптимизация IR. Пример реализации: DCE.cpp

```
bool DCE::runOnFunction(Function &F) {
  if (skipOptnoneFunction(F))
    return false;

  auto *TLIP =
getAnalysisIfAvailable<TargetLibraryInfoWrapperPass>();
  TargetLibraryInfo *TLI = TLIP ? &TLIP->getTLI() : nullptr;

  // Start out with all of the instructions in the worklist...
  std::vector<Instruction*> WorkList;
  for (inst_iterator i = inst_begin(F), e = inst_end(F); i != e; ++i)
    WorkList.push_back(&*i);

  // Loop over the worklist finding instructions that are dead.  If they are
  // dead make them drop all of their uses, making other instructions
  // potentially dead, and work until the worklist is empty.
  //
  bool MadeChange = false;
  while (!WorkList.empty()) {
    Instruction *I = WorkList.back();
    WorkList.pop_back();

    if (isInstructionTriviallyDead(I, TLI)) { // If the instruction is dead.
     // Loop over all of the values that the instruction uses, if there are
     // instructions being used, add them to the worklist, because they might
     // go dead after this one is removed.
     //
      for (User::op_iterator OI = I->op_begin(), E = I->op_end();
           OI != E; ++OI)
        if (Instruction *Used = dyn_cast<Instruction>(*OI))
          WorkList.push_back(Used);

      // Remove the instruction.
      I->eraseFromParent();

      // Remove the instruction from the worklist if it still exists in it.
      WorkList.erase(std::remove(WorkList.begin(),
            WorkList.end(), I),  WorkList.end());

      MadeChange = true;
      ++DCEEliminated;
    }
  }
  return MadeChange;
```


Copyright © 2014, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

Генерация инструкций архитектуры

```
#include <stdio.h>
int foo(int i, int j)
{
        if (j > 0)
                return i * (1 << j);
        printf("j<=0\n");
        return 0;
}
```

```
foo:
        pushl   %ebp
        movl    %esp, %ebp
        pushl   %eax
        movl    12(%ebp), %ecx
        testl   %ecx, %ecx
        jle     LBB0_2
# BB#1:                         # %if.then
        movl    8(%ebp), %eax
        shll    %cl, %eax
        jmp     LBB0_3
LBB0_2:                         # %if.end
        movl    $L_str, (%esp)
        calll   _puts
        xorl    %eax, %eax
LBB0_3:                         # %return
        addl    $4, %esp
        popl    %ebp
        retl
        .section .rdata,"rd"
L_str:                          # @str
        .asciz  "j<=0"
```

$> llc test.bc –march=x86

или сразу

$> clang test.c -march=x86

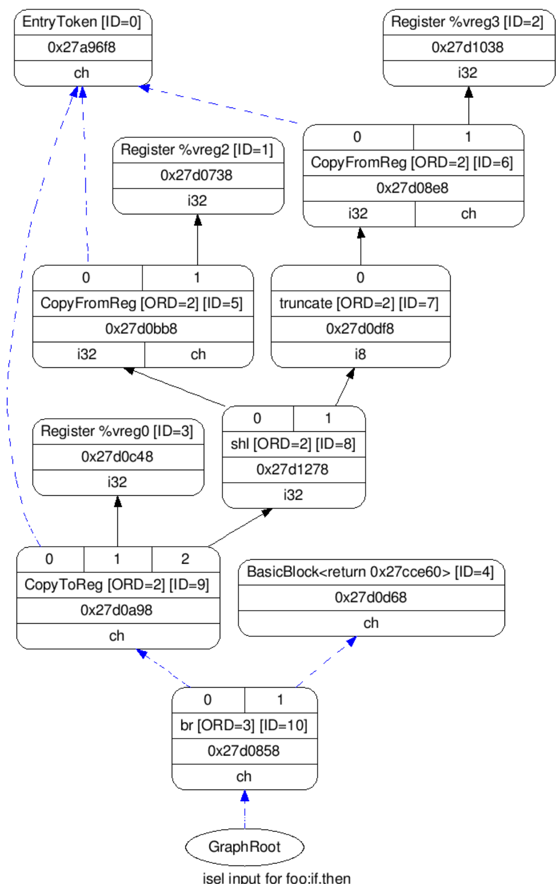
Copyright © 2014, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

Генерация инструкций архитектуры. Продолжение

1. Выбор инструкций

- для каждой IR-инструкции подставляется последовательность машинных инструкций данной архитектуры (см td-файлы) – формируются DAG’и:

```
let Constraints = "$src1 = $dst” in {
let Uses = [CL] in {
def SHL32rCL : I<0xD3, MRM4r,
     (outs GR32:$dst), (ins GR32:$src1),
     "shl{l}\t{%cl, $dst|$dst, cl}",
     [(set GR32:$dst, (shl GR32:$src1, CL))], IIC_SR>,
     OpSize32;
}}
```



Copyright © 2014, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

Генерация инструкций архитектуры. Продолжение

2. Планирование
   - линеаризация DAG’ов (т.е задание обхода графа) как можно более оптимальным способом
   - конвертация DAG’ов в списки MachineInstr; DAG’и больше не нужны

```
BB#1: derived from LLVM BB %if.then
    Predecessors according to CFG: BB#0
        %vreg7<def> = COPY %vreg3; GR32_ABCD:%vreg7 GR32:%vreg3
        %vreg8<def> = COPY %vreg7:sub_8bit; GR8:%vreg8 GR32_ABCD:%vreg7
        %CL<def> = COPY %vreg8; GR8:%vreg8
        %vreg0<def,tied1> = SHL32rCL %vreg2<tied0>, %EFLAGS<imp-def,dead>, %CL<imp-use>; GR32:%vreg0,%vreg2
        JMP_4 <BB#3>
```

Optimization Notice

Copyright © 2014, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

Генерация инструкций архитектуры. Продолжение

3. SSA-оптимизации машинного кода
   - оптимизации, специфичные для архитектуры
4. Распределение регистров
   - назначение виртуальным регистрам физических
   - число виртуальных регистров в LLVM неограниченно
   - число физических регистров в любой архитектуре конечно
   - некоторые инструкции требуют, чтобы операнды лежали в фиксированных регистрах
   - после этой фазы SSA больше нет

Copyright © 2014, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

Генерация инструкций архитектуры. Окончание

5. Вставка прологов/эпилогов функций
   - например, выделение/освобождение стека функциями
6. Заключительные оптимизации машинного кода
   - например, оптимизация spill/fill кода
7. Порождение машинных инструкций
   - вывод машинного кода или текстового ассемблера

Optimization Notice
Copyright © 2014, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

Источники

1. llvm.org официальный сайт проекта
2. Eli Bendersky blog статьи про кодогенерацию в LLVM
3. Tutorial: Creating an LLVM Backend for the Cpu0 Architecture
4. LLVM mailing lists
5. http://ellcc.org/demo/index.cgi онлайн-компилятор LLVM в IR и ассемблер
6. http://llvm.org/devmtg/ слайды и видео с конференций
7. http://llvm.org/docs/CodeGenerator.html генерация кода
8. clang --help-hidden

Copyright © 2014, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

Спасибо за внимание

Вопросы?

Ссылка на эту презентацию

http://bit.ly/1TUN3U7

Copyright © 2014, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

Legal Disclaimer & Optimization Notice

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.

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.

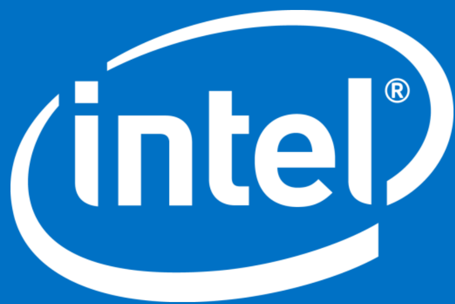
Copyright © 2014, Intel Corporation. All rights reserved. Intel, Pentium, Xeon, Xeon Phi, Core, VTune, Cilk, and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

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

Copyright © 2014, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.


intel®