SYRMIA

Projekat LLVM i informacije za debagovanje

Đorđe Todorović

Predavanje na kursu Konstrukcija kompilatora, Matematički fakultet, Beograd, 31. mart 2021.

Syrmia LLC

SYRMIA

Pregled predavanja

SYRMIA

O kompaniji

- https://www.syrmia.com/
- Beograd, Novi Sad, Niš, Banja Luka
  - C/C++ (C++14, C++17)
  - Linux, Windows i Android OS (internals)
  - Embeded development (Gdb, Git, Linux kernel...)
  - Kompilatori
  - ...
- Praksa? Pošalji CV na jobs@syrmia.com

SYRMIA

O meni

- Diplomirao na osnovnim i master strudijama Matematičkog fakulteta u Beogradu
- Radi za kompaniju SYRMIA
- Aktivno učestvuje u razvoju projekta LLVM
- Autor
  - llvm-locstats (https://llvm.org/docs/CommandGuide/llvm-locstats.html)
  - Alat za verifikaciju LLVM optimizacionih prolaza u pogledu čuvanja informacija za debagovanje (https://llvm.org/docs/HowToUpdateDebugInfo.html#test-original-debug-info-preservation-in-optimizations)
  - ...
- djordje.todorovic@syrmia.com

SYRMIA

Programski prevodioci i infrastruktura LLVM

SYRMIA

Uvod

- Programski prevodioci (kompilatori) prevode izvorni kod u ciljani (asembler/binarni) kod, koji mašina razume
- Izvorni kod i izvršni program

```
#include <iostream>

int main() {
  std::cout << "Hello world!\n";
  return 0;
}
```


```
./a.out
```

- Preprocesiranje
- Faze prevodjenja: prednji deo (eng. frontend), srednji deo (eng. middleend), zadnji deo (eng. backend), štampanje asemblera
- Linkovanje

SYRMIA

Uvod

- Projekat LLVM (https://llvm.org/)
  - llvm/ clang/ flang/ libc++/ lldb/ mlir/ ...
  - LLVM vs GCC
- Faze prevođenja kod programskih prevodilaca baziranim na ekosistemu LLVM

- LLVM IR – troadresni kod (SSA form) - svakoj promenljivoj tačno jednom dodeljujemo vrednost
- asembler za konkretan hardver (CPU, GPU, Hibridna arhitektura, itd.)

SYRMIA

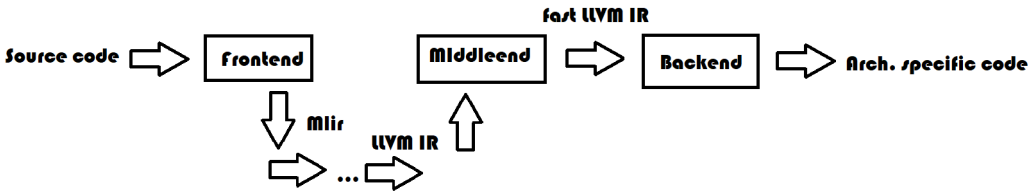
Uvod

- Optimizacije koje su specifične za konkretan programski jezik?
- Programski jezici kao sto su Swift i Rust uvode novi nivo apstrakcije koda (novi međukod) koji je dosta bliže izvornom kodu, gde vršimo razne optimizacije koje su specifične za taj programski jezik
  - Swift SIL
- Primer: Dvostruko transponovanje matrice
- MLIR
  - Dijalekti

SYRMIA

Uvod

- Faze prevođenja kod programskih prevodilaca baziranim na ekosistemu LLVM + MLIR



SYRMIA

Prednji deo

- Clang/Flang/rustc/Swift
- Leksička analiza (eng. Lexical analysis or tokenization)
  - Tekst pretvara u tokene

Example

studenti += mika + zika
ID (studenti)
PLUS_EQ
...

- Sintaksna analiza
  - Apstraktno sintaksno stablo (AST)
- Semantička analiza
  - Provera onoga što linije koda rade (npr. provera tipova)
- Generator međukoda

SYRMIA

Prednji deo

- Primer (test.c):

```
extern int fn2();
int fn() {
  int x = fn2();
  if (x % 2)
    return 0;
  return 1;
}
```

SYRMIA

Prednji deo

```
$ clang -Xclang -ast-dump test.c
```

```
TranslationUnitDecl 0x653f1f8 <<invalid sloc>> <invalid sloc>
|-TypedefDecl 0x653faf0 <<invalid sloc>> <invalid sloc> implicit __int128_t '__int128'
| `-BuiltinType 0x653f790 '__int128'
|-TypedefDecl 0x653fb60 <<invalid sloc>> <invalid sloc> implicit __uint128_t 'unsigned __int128'
| `-BuiltinType 0x653f7b0 'unsigned __int128'
|-TypedefDecl 0x653fed8 <<invalid sloc>> <invalid sloc> implicit __NSConstantString '__NSConstantString_tag'
| `-RecordType 0x653fc50 '__NSConstantString_tag'
|   `-CXXRecord 0x653fbb8 '__NSConstantString_tag'
|-TypedefDecl 0x653ff70 <<invalid sloc>> <invalid sloc> implicit __builtin_ms_va_list 'char *'
| `-PointerType 0x653ff30 'char *'
|   `-BuiltinType 0x653f290 'char'
|-TypedefDecl 0x6580db8 <<invalid sloc>> <invalid sloc> implicit __builtin_va_list '__va_list_tag [1]'
| `-ConstantArrayType 0x6580d60 '__va_list_tag [1]' 1
|   `-RecordType 0x6540060 '__va_list_tag'
|     `-CXXRecord 0x653ffc8 '__va_list_tag'
|-FunctionDecl 0x6580e60 <test.cpp:1:1, col:16> col:12 used fn2 'int ()' extern
`-FunctionDecl 0x6580f70 <line:3:1, line:8:1> line:3:5 fn 'int ()'
  `-CompoundStmt 0x6581288 <col:10, line:8:1>
    |-DeclStmt 0x6581160 <line:4:3, col:16>
    | `-VarDecl 0x6581028 <col:3, col:15> col:7 used x 'int' cinit
    |   `-CallExpr 0x6581140 <col:11, col:15> 'int'
    |     `-ImplicitCastExpr 0x6581128 <col:11> 'int (*)()' <FunctionToPointerDecay>
    |       `-DeclRefExpr 0x65810d8 <col:11> 'int ()' lvalue Function 0x6580e60 'fn2' 'int ()'
    |-IfStmt 0x6581238 <line:5:3, line:6:12>
    | |-ImplicitCastExpr 0x65811f0 <line:5:7, col:11> 'bool' <IntegralToBoolean>
    | | `-BinaryOperator 0x65811d0 <col:7, col:11> 'int' '%'
    | |   |-ImplicitCastExpr 0x65811b8 <col:7> 'int' <LValueToRValue>
    | |   | `-DeclRefExpr 0x6581178 <col:7> 'int' lvalue Var 0x6581028 'x' 'int'
    | |   `-IntegerLiteral 0x6581198 <col:11> 'int' 2
    | `-ReturnStmt 0x6581228 <line:6:5, col:12>
    |   `-IntegerLiteral 0x6581208 <col:12> 'int' 0
    `-ReturnStmt 0x6581278 <line:7:3, col:10>
      `-IntegerLiteral 0x6581258 <col:10> 'int' 1
```

SYRMIA

Prednji deo

- Primer LLVM IR:

```
$ clang -S -emit-llvm test.c
```

```
; ModuleID = 'test.c'
source_filename = "test.c"
target datalayout = "e-m:e-p270:32:32-p271:32:32-p272:64:64-i64:64-f80:128-n8:16:32:64-S128"
target triple = "x86_64-unknown-linux-gnu"

; Function Attrs: noinline nounwind optnone uwtable
define dso_local i32 @fn() #0 {
entry:
  %retval = alloca i32, align 4
  %x = alloca i32, align 4
  %call = call i32 (...) @fn2()
  store i32 %call, i32* %x, align 4
  %0 = load i32, i32* %x, align 4
  %rem = srem i32 %0, 2
  %tobool = icmp ne i32 %rem, 0
  br i1 %tobool, label %if.then, label %if.end

if.then:                                          ; preds = %entry
  store i32 0, i32* %retval, align 4
  br label %return

if.end:                                           ; preds = %entry
  store i32 1, i32* %retval, align 4
  br label %return

return:                                           ; preds = %if.end, %if.then
  %1 = load i32, i32* %retval, align 4
  ret i32 %1
}
```

SYRMIA

Srednji deo

- Pretvaranje međukoda u brži međukod
- Optimizacije (-O1, -O2, -O3)
- Optimizacije u praksi
  - Bliže prednjem delu
    - Alias (pointer) analysis
  - Bliže ciljanom kodu (specifično za konkretne procesore)
    - X86 CPU:

```
mov %eax, 0
xor %eax, %eax
```

  - Negde između (middleend level)
    - Loop invariant hoisting
    - Combine redundant instructions

SYRMIA

Srednji deo

- Combine redundant instructions

```
12  //
13  // This pass combines things like:
14  //     %Y = add i32 %X, 1
15  //     %Z = add i32 %Y, 1
16  // into:
17  //     %Z = add i32 %X, 2
```

SYRMIA

Zadnji deo

- Optimizacije tipa:

```
mov %eax, 0
xor %eax, %eax
```

- Biranje instrukcija (Instruction selection) – IR instrukcije pretvaramo u mašinske – LLVM SelectionDAG, GlobalIsel, FastIsel
- Raspoređivanje instrukcija (Instruction scheduler) – rasporediti instrukcije tako da iskoristimo maksimum kapaciteta hardvera
- Alokacija registara (Register allocation) – imamo promenljive i određen broj registara, kako napraviti najbolju iskorišćenost?
- Pažnja: Heuristike – na osnovu analiza, kompilator generiše/raspoređuje instrukcije, ali... nekad i greši, jer ne postoji univerzalan odgovor koji važi za sve arhitekture
- Štampanje asemblera (AsmPrinter)

SYRMIA

Informacije za debagovanje

SYRMIA

Debagovanje

- Proces analize i otklanjanje grešaka u programima
- Kako debagovati?
  - printf()
  - debager
  - ...
  - drugi alati

SYRMIA

Debagovanje – printf()

```c
#include <stdio.h>
int fn() {
  int x = 4;
  if (x % 2) {
    printf("neparan broj\n");
    return 0;
  }
  printf("paran broj\n");
  return 1;
}
int main() {
  return fn();
}
```

```
$ clang -g -O2 test.c -o test
$ ./test
paran broj
```

SYRMIA

Debagovanje – printf() - realnost :)

```c
#include <stdio.h>
int fn() {
  int x = 4;
  if (x % 2) {
    printf("AAAAAAAAAAAAAAA\n");
    return 0;
  }
  printf("BBBBBBBBBBBBB\n");
  return 1;
}
int main() {
  return fn();
}
```

```
$ clang -g -O2 test.c -o test
$ ./test
BBBBBBBBBBBBB
```

SYRMIA

Debageri

- GNU GDB (https://www.gnu.org/software/gdb/)
  - Prilično stabilan i zreo alat
  - Prevođenje programa sa opcijom -g
  - Pomoćne informacije za debagovanje
    - DWARF (za Unix-like operativne sisteme)
    - CodeView (za Windows operativne sisteme)
    - ...
- LLDB (https://lldb.llvm.org/)

Debageri

SYRMIA

```
djtodorovic@djtodorovic:~/projects/predavanje-matf-examples$ gdb a.out
GNU gdb (Ubuntu 9.1-0ubuntu1) 9.1
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from a.out...
(No debugging symbols found in a.out)
(gdb)
```

Debageri

SYRMIA

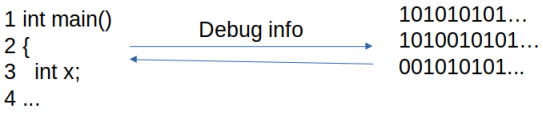
```
djtodorovic@djtodorovic:~/projects/predavanje-matf-examples$ gcc -g test2.c
djtodorovic@djtodorovic:~/projects/predavanje-matf-examples$ gdb ./a.out
GNU gdb (Ubuntu 9.1-0ubuntu1) 9.1
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./a.out...
(gdb) b 5
Breakpoint 1 at 0x115c: file test2.c, line 5.
(gdb) r
Starting program: /home/djtodorovic/projects/predavanje-matf-examples/a.out

Breakpoint 1, fn () at test2.c:5
5           if (x % 2) {
(gdb)
```

SYRMIA

DWARF



- DWARF - standard za zapisivanje informacija za debagovanje za Unix-like operativne sisteme
- drvolika struktura
- DIE - osnovna jedinica
- DWARF tagovi i atributi (DW_TAG_subprogram, DW_TAG_variable, DW_TAG_formal_parameter, DW_AT_name, DW_AT_location)
- Sekcije: .debug_info, .debug_line, .debug_loc ...

SYRMIA

LLVM DI Metadata

- Apstrakcija informacija za debagovanje
- Reprezentacija na LLVM IR nivou

```
!llvm.dbg.cu = !{!0}
!llvm.module.flags = !{!3, !4, !5}
!llvm.ident = !{!6}

!0 = distinct !DICompileUnit(language: DW_LANG_C99, file: !1, producer: "clang version 12.0.0", isOptimized: false, runtimeVersion: 0, emissionKind: FullDebug, enums: !2, splitDebugInlining: false, nameT
ableKind: None)
!1 = !DIFile(filename: "test.c", directory: "/home/djtodorovic/projects/predavanje-matf-examples")
!2 = !{}
!3 = !{i32 7, !"Dwarf Version", i32 4}
!4 = !{i32 2, !"Debug Info Version", i32 3}
!5 = !{i32 1, !"wchar_size", i32 4}
!6 = !{!"clang version 12.0.0"}
!7 = distinct !DISubprogram(name: "fn", scope: !1, file: !1, line: 3, type: !8, scopeLine: 3, spFlags: DISPFlagDefinition, unit: !0, retainedNodes: !2)
!8 = !DISubroutineType(types: !9)
!9 = !{!10}
!10 = !DIBasicType(name: "int", size: 32, encoding: DW_ATE_signed)
!11 = !DILocalVariable(name: "x", scope: !7, file: !1, line: 4, type: !10)
!12 = !DILocation(line: 4, column: 7, scope: !7)
!13 = !DILocation(line: 4, column: 11, scope: !7)
!14 = !DILocation(line: 5, column: 7, scope: !15)
!15 = distinct !DILexicalBlock(scope: !7, file: !1, line: 5, column: 7)
!16 = !DILocation(line: 5, column: 9, scope: !15)
!17 = !DILocation(line: 5, column: 7, scope: !7)
!18 = !DILocation(line: 6, column: 5, scope: !15)
!19 = !DILocation(line: 7, column: 3, scope: !7)
!20 = !DILocation(line: 8, column: 1, scope: !7)
                                                                                                                                                                    65,1          Bot
```

SYRMIA

Informacije za debagovanje u optimizovanom kodu

- Release mode vs Debug mode
- Gubljenje informacija za debagovanje tokom optimizacija
- Posledice – $< optimized\_out >$, nemogućnost postavljanja tačaka prekida koristeći debager, itd.

SYRMIA

Informacije za debagovanje u optimizovanom kodu

- Optimizovana promenljiva

```
(gdb) f 3
#3  0x0000000000004011f6 in f () at test.c:8
8           f1(local);
(gdb) p local
$1 = <optimized out>
```

- Spašena promenljiva

```
(gdb) f 3
#3  0x0000000000401216 in f () at test2.c:8
8           f1(local);
(gdb) p local
$1 = 0
```

SYRMIA

Informacije za debagovanje u optimizovanom kodu

- Alat za verifikaciju LLVM optimizacionih prolaza u pogledu čuvanja informacija za debagovanje (https://llvm.org/docs/HowToUpdateDebugInfo.html#test-original-debug-info-preservation-in-optimizations)

```
bash-4.2$ ../build/bin/clang -g -O2 -Xclang -fverify-debuginfo-preserve -Xclang -fverify-debuginfo-preserve-export=sample.json simple.c -c
AnnotationToMetadata: PASS
Force set function attributes: PASS
Infer set function attributes: PASS
Interprocedural Sparse Conditional Constant Propagation: PASS
Called Value Propagation: PASS
Global Variable Optimizer: PASS
Promote Memory to Register: PASS
Promote Memory to Register: PASS
Dead Argument Elimination: PASS
Combine redundant instructions: PASS
Simplify the CFG: PASS
Combine redundant instructions: PASS
Simplify the CFG: PASS
Globals Alias Analysis: PASS
SROA: PASS
Early CSE w/ MemorySSA: PASS
Speculatively execute instructions if target has divergent branches: PASS
Jump Threading: PASS
Value Propagation: PASS
Simplify the CFG: PASS
Combine redundant instructions: PASS
```

- llvm-locstats (https://llvm.org/docs/CommandGuide/llvm-locstats.html)

SYRMIA

Najbitnije...

- LLVM projekat je ekosistem za razne ‘low-level‘ alate, ne samo za kompilatore
- Clang, Rust, Swift su najbrži kompilatori današnjice
- Informacije za debagovanje generišu kompilatori, debageri ih koriste za debagovanje
- Neke informacije za debagovanje se gube tokom optimizacija, te je veliki izazov sačuvati iste

SYRMIA

Hvala

SYRMIA

Contact us

SYRMIA
Belgrade Office Park
Đorđa Stanojevića 12
Belgrade 11070
Serbia