

Projekat LLVM: Pregled

Matematički fakultet,
Univerzitet u Beogradu
Mart, 2024.

Đorđe Todorović,
Head of System Software at
Syrmia, an HTEC company

Syrmia

- 2017
- Beograd, Niš, Novi Sad, Banja Luka
- Projekti niskog nivoa
- Open Source
- *An HTEC Company*
- Stipendirane prakse
 - hr@syrmia.com
 - <https://www.syrmia.com/>

Content



01_Kompajleri

02_Uvod u LLVM

03_LLVM IR(s) i optimizacije

04_Alati bazirani na projektu LLVM

05_Autocheck



01_Kompajleri

01_Kompajleri



- Specijalni alati
- input/output
- Linkeri
- Faze
- Optimizacije

```
● ● ●  
1  #include <stdio.h>  
2  
3  int main()  
4  {  
5      printf("Hello world!\n");  
6      return 0;  
7  }  
8
```

```
● ● ●  
1  00000000 facf feed 000c 0100 0000 0000 0002 0000  
2  00000100 0011 0000 0420 0000 0085 0020 0000 0000  
3  00000200 0019 0000 0048 0000 5f5f 4150 4547 455a  
4  00000300 4f52 0000 0000 0000 0000 0000 0000 0000  
5  00000400 0000 0000 0001 0000 0000 0000 0000 0000  
6  00000500 0000 0000 0000 0000 0000 0000 0000 0000  
7  00000600 0000 0000 0000 0000 0019 0000 0188 0000  
8  00000700 5f5f 4554 5458 0000 0000 0000 0000 0000  
9  00000800 0000 0000 0001 0000 4000 0000 0000 0000  
10 00000900 0000 0000 0000 0000 4000 0000 0000 0000  
11 00000a00 0005 0000 0005 0000 0004 0000 0000 0000  
12 00000b00 5f5f 6574 7478 0000 0000 0000 0000 0000  
13 00000c00 5f5f 4554 5458 0000 0000 0000 0000 0000
```



02_Uvod u LLVM

02_Uvod u LLVM_O projektu



- Chris Lattner
 - 2000, 2005, 2006
 - danas
- Apple
- *Low-level ecosystem*
- Glavni kompajleri za Swift, Rust, C/C++ su bazirani na projektu LLVM
- Open-source
 - <https://github.com/llvm/llvm-project>

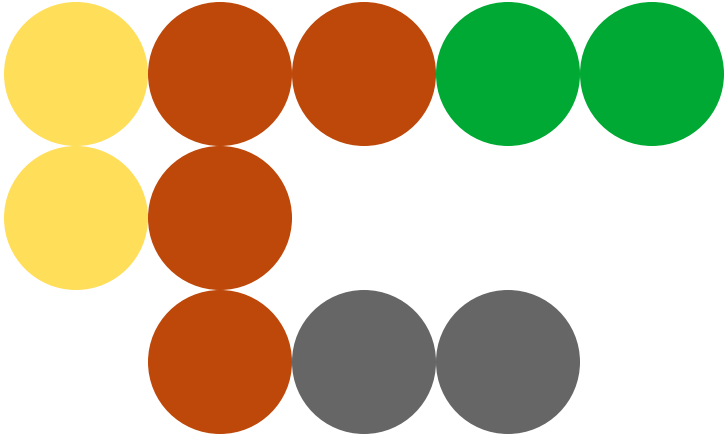


02_Uvod u LLVM

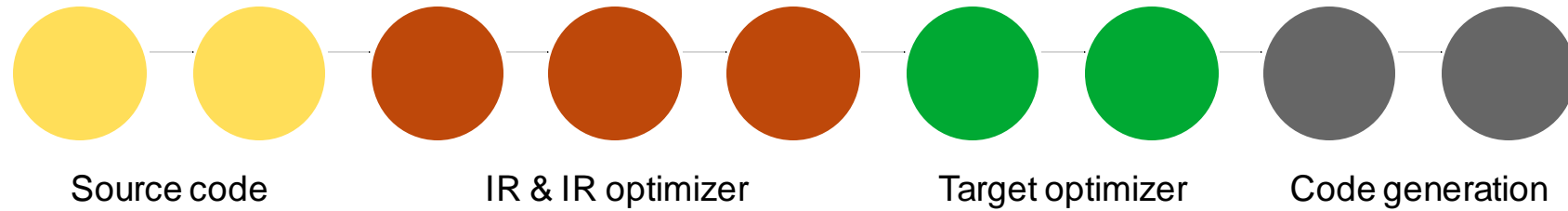


Zašto je LLVM *promenio igru*?

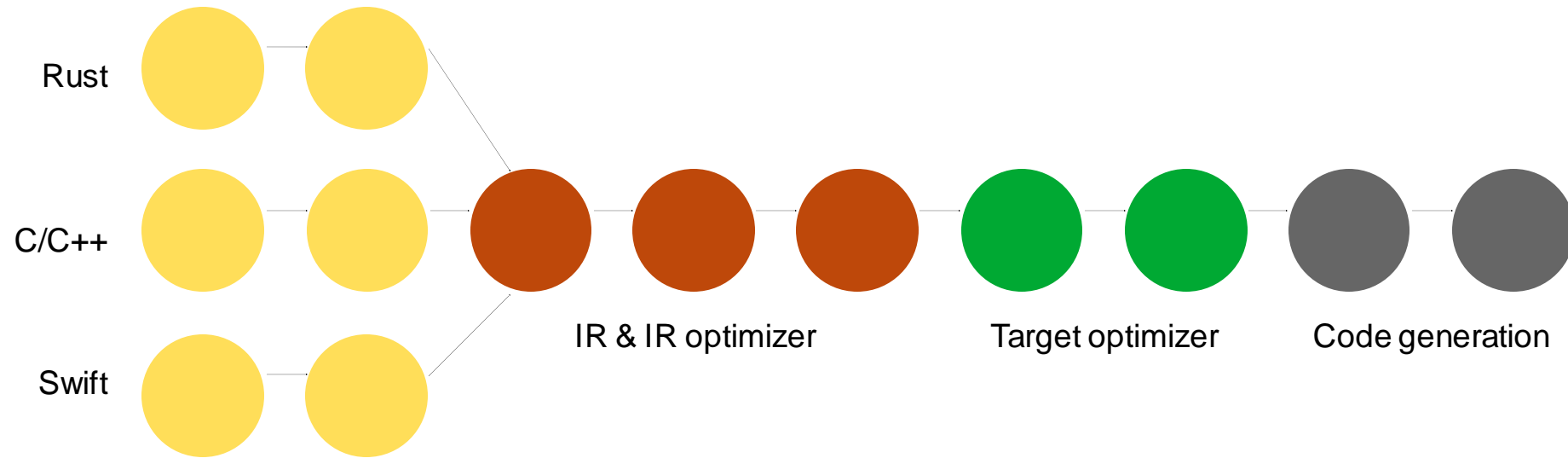
02_Uvod u LLVM_Dizajn



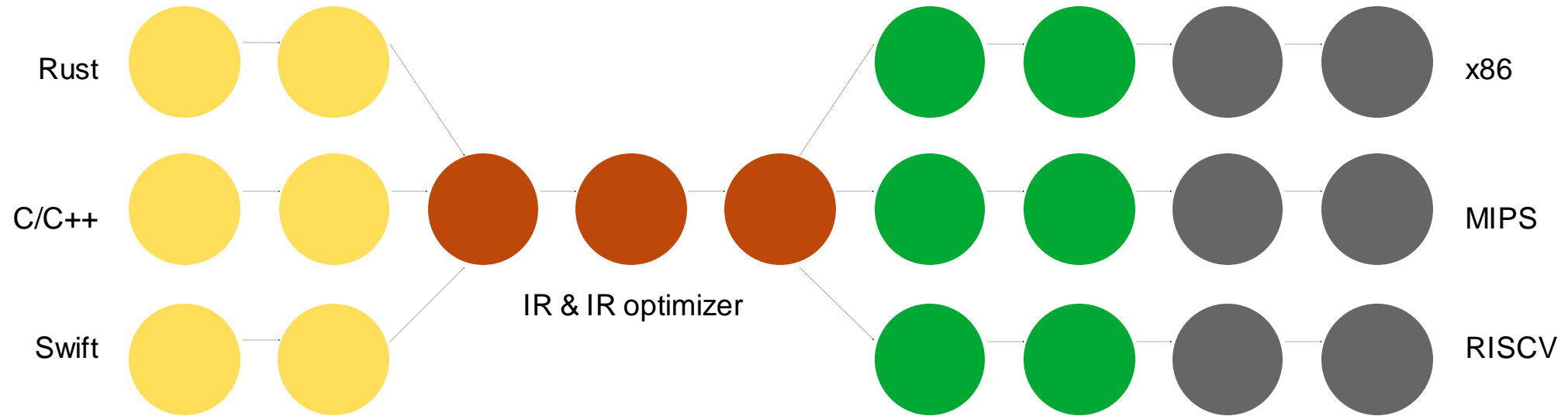
02_Uvod u LLVM_Dizajn



02_Uvod u LLVM_Dizajn



02_Uvod u LLVM_Dizajn



02_Uvod u LLVM_Međukod



Međukod(ovi)/LLVMIRs:

- LLVMIR
- LLVMMachine IR
- MLIR

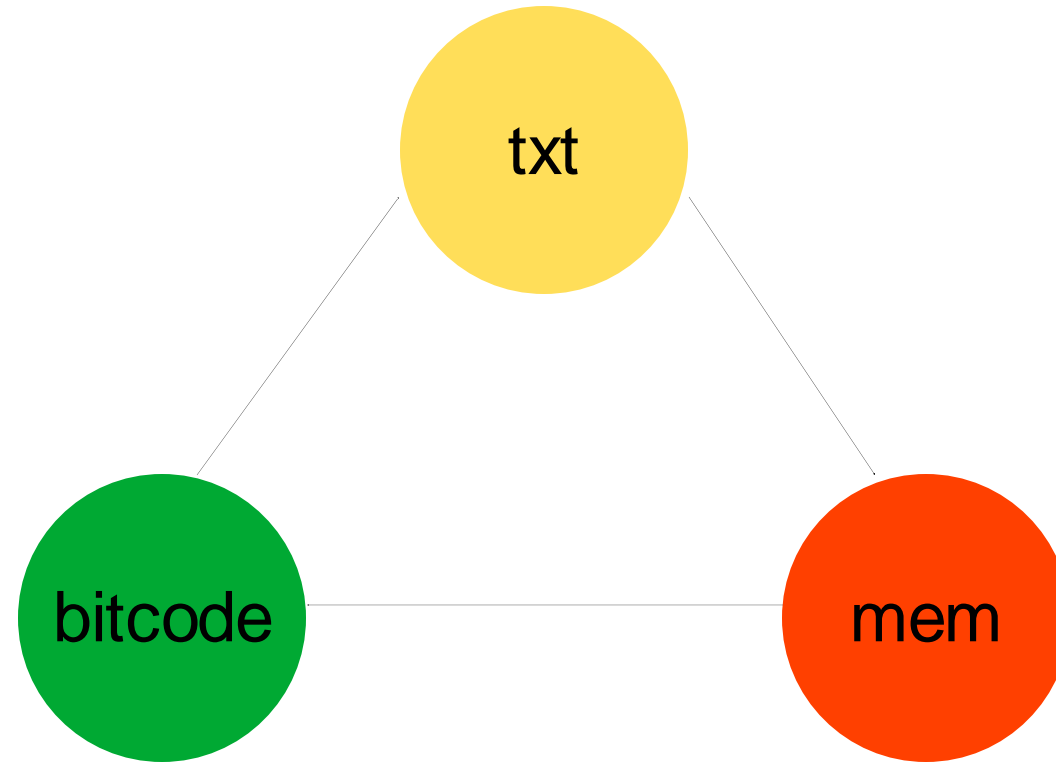
LLVMIR:

- Jezik niskog nivoa
- Jako tipiziran
- Virtualni registri (%1, %2, ...)
- SSA forma
- Nezavistan od polaznog izvornog programskog jezika
- Nezavistan od krajnje procesorske arhitekture

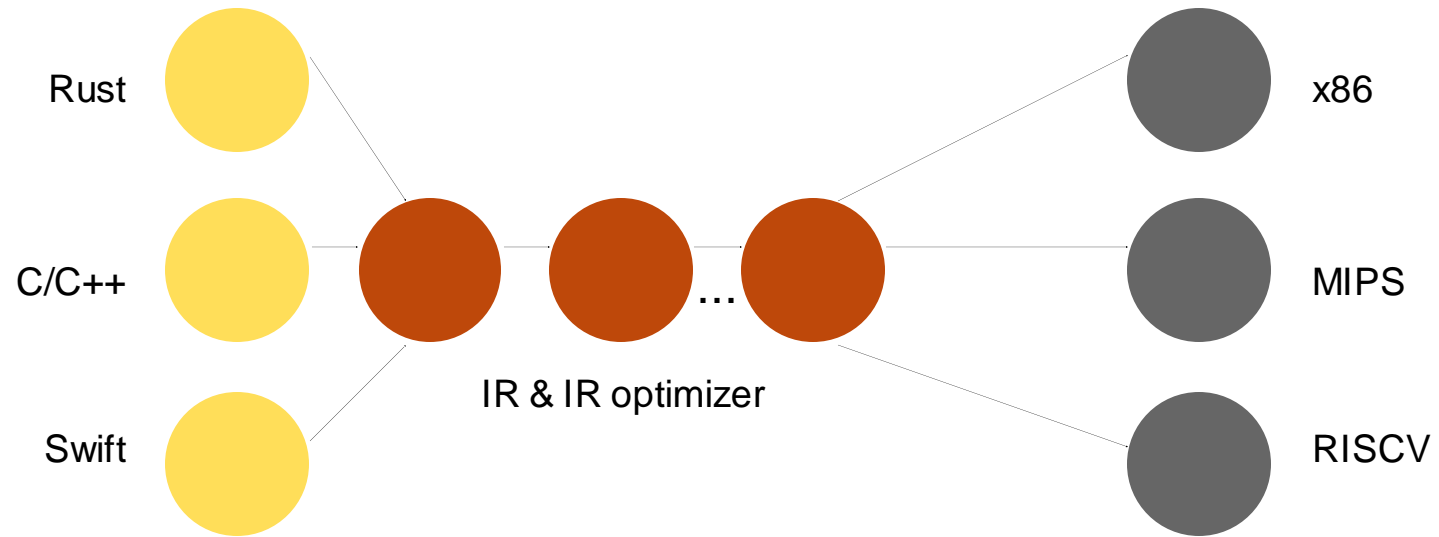


03_LLVM IR(s) & Optimizacije

03_LLVM IR



03_LLVM IR_Optimizer



03_LLVM

IR_Optimizer



```
int add(int x, int y) {  
    return x + y;  
}
```

03_LLVM

IR_Optimizer



```
define i32 @add(i32 %x, i32 %y) {  
entry:  
    %res = add i32 %x, %y  
    ret i32 %res  
}
```



03_Kako napisati optimizaciju na LLVM IR nivou

- Detekcija paterna
- Dokaz korektnosti!
- Pisanje optimizacije
- Paterni:
 - $X - X \implies 0$
 - $X - 0 \implies X$
 - $(X*2)-X \implies X$



03_Kako napisati optimizaciju na LLVM IR nivou

1. `%pattern1 = sub i32 %x, %x`
2. `%pattern2 = sub i32 %x, 0`
3. `%tmp = mul i32 %x, 2`
`%pattern3 = sub i32 %tmp, %x`



03_Kako napisati optimizaciju na LLVM IR nivou

```
llvm::Module, llvm::Function, llvm::BasicBlock, llvm::Instruction
```

```
for (BasicBlock &BB : Function.blocks())  
  for (Instruction &I : BB) {  
    if (isRedundant(I)) {  
      Value *V = simplifyInstruction(I);  
      I.replaceAllUsesWith(V);  
    }  
  }  
}
```

- SimplifyInstruction:

```
if (I->getOpcode == sub)  
  if (match(I->getOp1(), 0))  
    return Op0;
```

- <https://llvm.org/docs/WritingAnLLVMNewPMPass.html>



LLVMMachine IR

- Nakon pranja istrukcija (SelectionDAG)
- Zavisno od arhitekture
- Razlučujemo o pojmovima kao što su memorija i registri
- Primer: optimizacija na x86:
 - `mov $x, 0 => xor $x, $x`



04_Alati bazirani na projektu LLVM

04_Alati bazirani na projektu LLVM



Debugify

- Alati za detekciju grešaka u projektu LLVMIR prilikom generisanja Informacija za Debagovanje
- <https://lvm.org/docs/HowToUpdateDebugInfo.html#test-original-debug-info-preservation-in-optimizations>

Optimizacije

- <https://www.phoronix.com/news/LLVM-Clang-14-Hoist-Load> (PostgreSQL optimized for 12%)
- ...

Crash Analyzer

- Alati za automatsku detekciju grešaka
- Dekompilacija: binary --> LLVM MIR --> analiza
- <https://github.com/syrmia/crash-analyzer>

04_Alati bazirani na projektu LLVM_Debugify



```
[djtodorovic@Djordjes-MacBook-Pro predavanje_matf % clang -Xclang -fverify-debuginfo-preserve -O2 -g add.c
Annotation2Metadata: 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
Dead Argument Elimination: 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
```

04_Alati bazirani na projektu LLVM_CrashAnalyzer



```
$ llvm-crash-analyzer --core-file=core.a.out.9595 ./a.out
Crash Analyzer -- crash analyzer utility

Loading core-file core.a.out.9595
core-file processed.

Decompiling...
Decompiling b(int*)
Decompiling main
Decompiled.

Analyzing...

Blame Function is b(int*)
From File test.cpp
```



05_Autocheck

05_Autocheck



- Alat baziran na projektu LLVM za proveru koda pisanog u programskom jeziku C++ prema Standardu AUTOSAR
 - AUTOSAR - *Standard for C++ Automotive Software*
 - Provera sigurnosti u kritičnom softveru
 - Sigurnost, Bezbednost, Kvalitet
- Clang API
- Jezgro (*Core* funkcionalnost)
 - <https://github.com/syrmia/autocheck>
 - I dalje privatn repozitorijum, uskoro javan
 - <https://www.phoronix.com/news/Autocheck-LLVM-Safety-Critical>
- IDE plugins
 - vscode

05_Autocheck (Primer)



```
#include <iostream>

int main() {
    int a = 0xff;
    std::cout << a << std::endl;

    return 0;
}

~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
example.cpp      1,1      All
```



05_Autocheck (Primer)



Autocheck v0.0.1

SYRMIA

AUTOSAR C++ Guidelines Compliance Checker

[Disable](#) [Uninstall](#)

This extension is enabled globally.

[DETAILS](#) [FEATURES](#)

Autocheck

Autocheck plugin for Visual Studio Code. Its main purpose is to check code against AUTOSAR guidelines for the use of the C++14 language in critical and safety-related systems. The main application sector of these guidelines is automotive, but these guidelines can be used in other embedded application sectors.

diagnostics

hover

Requirements

- VSCode 1.85+
- [Autocheck](#)

License

This extension is licensed under the [Apache-2.0](#) license.

05_Autocheck (Primer)



The screenshot shows the Visual Studio Code editor interface. The main editor window displays a file named `example.cpp` with the following code:

```
home > syrmia > example.cpp > main()
1  int main()
2
3
4      return 0;
5
6
```

The editor has a dark theme. The menu bar at the top includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The left sidebar contains icons for Explorer, Search, Source Control, Run and Debug, and Extensions. The bottom status bar shows the current cursor position as `Ln 2, Col 5`, `Spaces: 4`, `UTF-8`, `LF`, `() C++`, and `Linux`. The bottom right corner of the editor area displays the message: `No problems have been detected in the workspace.`

05_Autocheck_Implementacija



- Clang API
 - Visitors
 - Matchers
 - Lexer/Parser/Preprocessor
 - clang::RecursiveASTVisitor<T>
 - clang::ast_matchers::StatementMatcher
 - clang::PPCallbacks
 - clang::Preprocessor
- Clang IR (CIR) - C/C++ MLIR Dialect
- Zasebsan projekat
 - LLVM project git submodule

05_Autocheck_Implementacija



clang::Preprocessor

Example: The goto statement shall not be used

```
void test() {  
  pong:  
  asm goto("ping");  
  ping:  
  goto pong;  
}
```

Implementation:

```
Cl.getPreprocessor().setTokenWatcher(  
  [this](const clang::Token &Tok) {  
    if (Tok.is(clang::tok::kw_goto)) {  
      // Goto keyword used.  
      AD.reportWarning(Loc, AutocheckWarnings::gotoUsed);  
    }  
  }  
);
```

05_Autocheck_Implementacija



clang::PPCallbacks

Example: The stream input/output library <stdio> shall not be used

```
#include <stdio>
```

```
#include <stdio.h>
```

Code:

```
void AutocheckPPCallbacks::InclusionDirective(...) {  
    llvm::StringRef FileName = File->getName();  
  
    if (FileName == "stdio" || FileName == "stdio.h")  
        AD.reportWarning(Loc, AutocheckWarnings::clibHeaderUsed);  
}
```



Pitanja



thanks ■ learn ■ share ■ experience
technology

