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

01_Kompajleri

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

```c
#include <stdio.h>

int main()
{
    printf("Hello world!\n");
    return 0;
}

```

```
0000000 facf feed 000c 0100 0000 0000 0002 0000
0000010 0011 0000 0420 0000 0085 0020 0000 0000
0000020 0019 0000 0048 0000 5f5f 4150 4547 455a
0000030 4f52 0000 0000 0000 0000 0000 0000 0000
0000040 0000 0000 0001 0000 0000 0000 0000 0000
0000050 0000 0000 0000 0000 0000 0000 0000 0000
0000060 0000 0000 0000 0000 0019 0000 0188 0000
0000070 5f5f 4554 5458 0000 0000 0000 0000 0000
0000080 0000 0000 0001 0000 4000 0000 0000 0000
0000090 0000 0000 0000 0000 4000 0000 0000 0000
00000a0 0005 0000 0005 0000 0004 0000 0000 0000
00000b0 5f5f 6574 7478 0000 0000 0000 0000 0000
00000c0 5f5f 4554 5458 0000 0000 0000 0000 0000
```

02_Uvod u LLVM

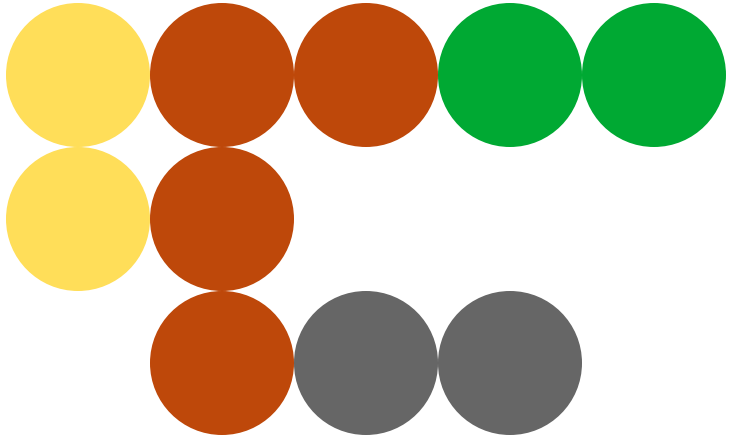
02_Uvod u LLVM_O projektu

- 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

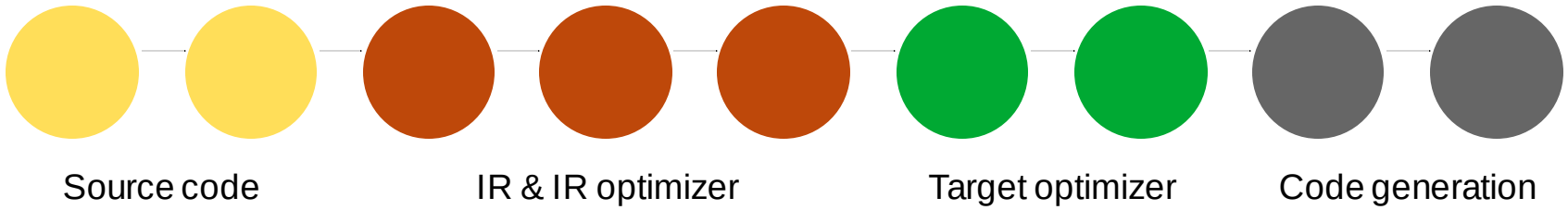
02_Uvod u LLVM

Zašto je LLVM *promenio igru*?

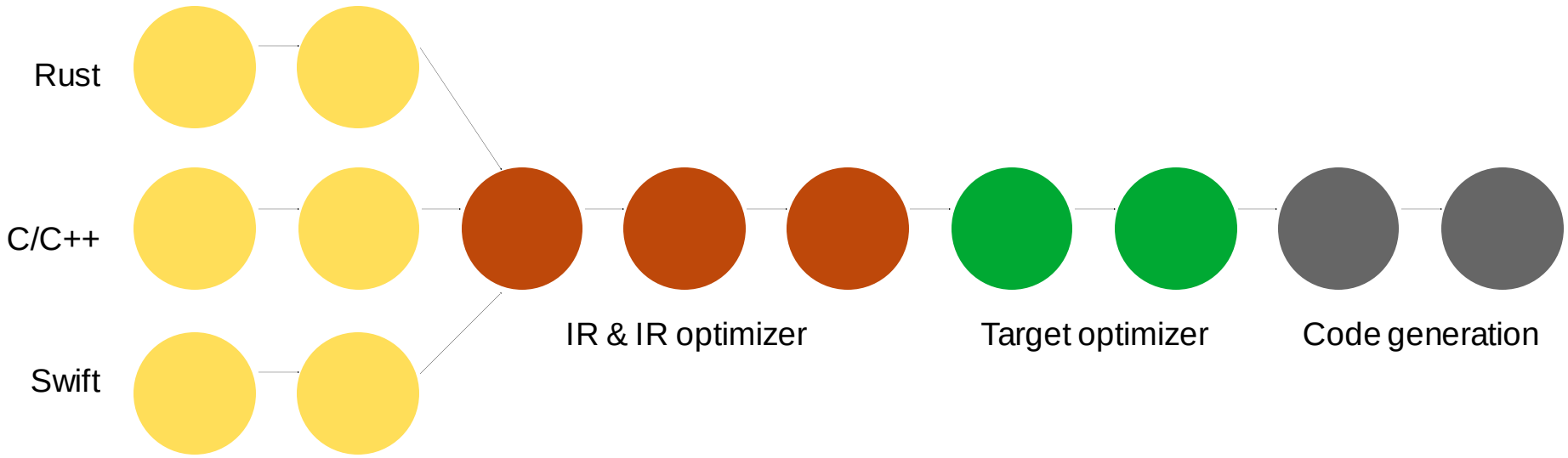
02_Uvod u LLVM_Dizajn



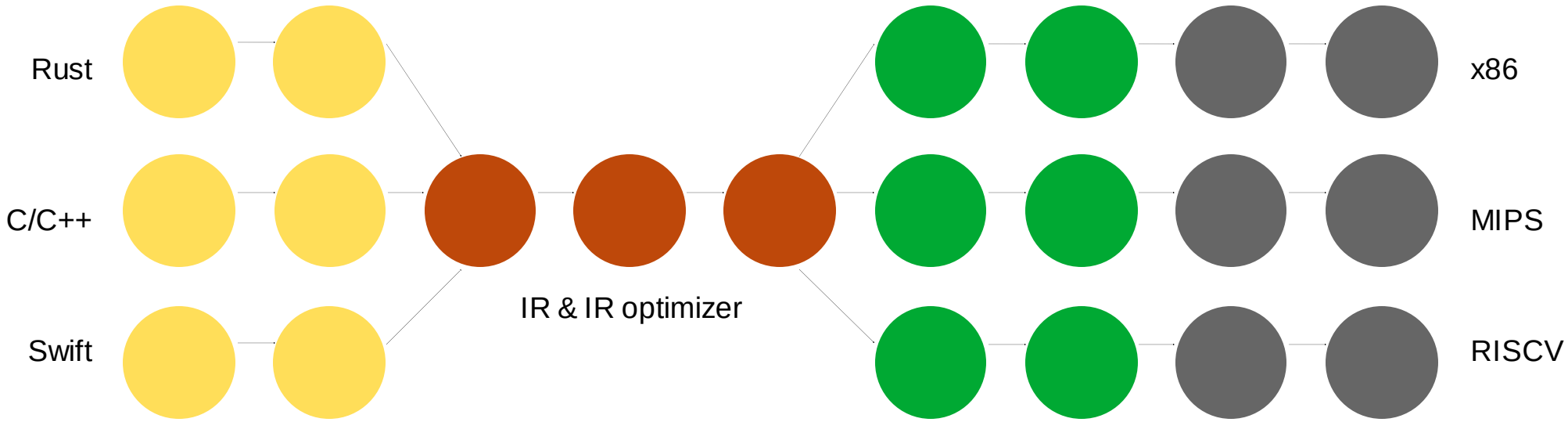
02_Uvod u LLVM_Dizajn



02_Uvod u LLVM_Dizajn



02_Uvod u LLVM_Dizajn



02_Uvod u LLVM_Međukod

Međukod(ovi)/LLVM IRs:

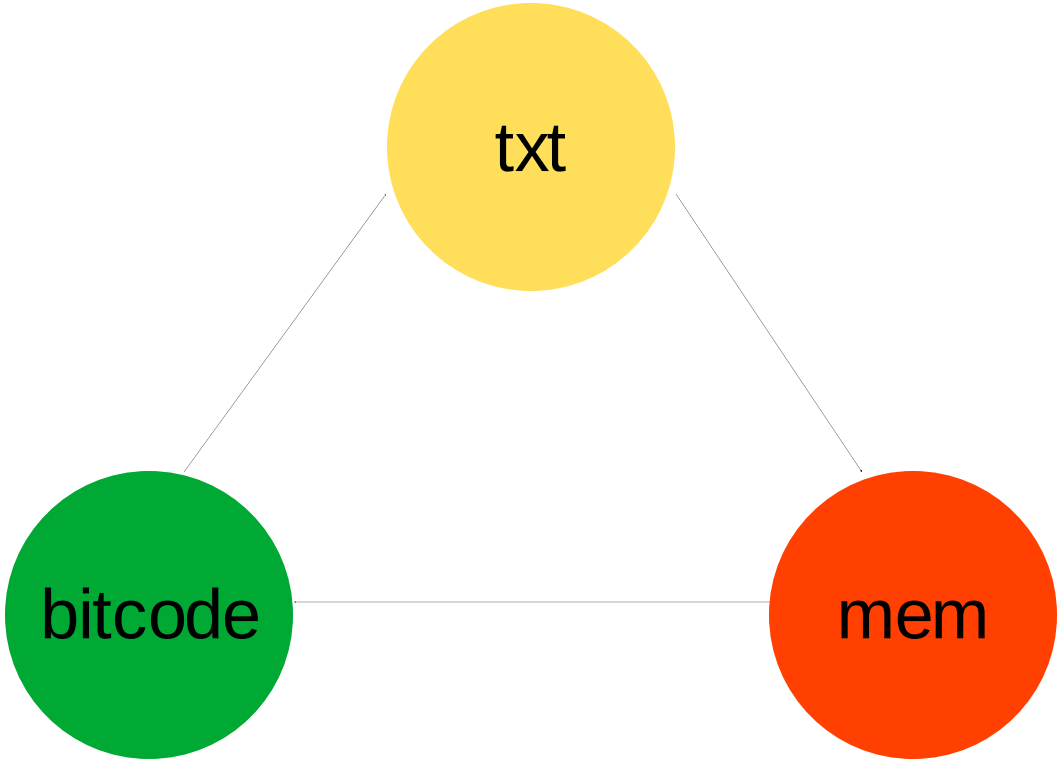
- LLVM IR
- LLVM Machine IR
- MLIR

LLVM IR:

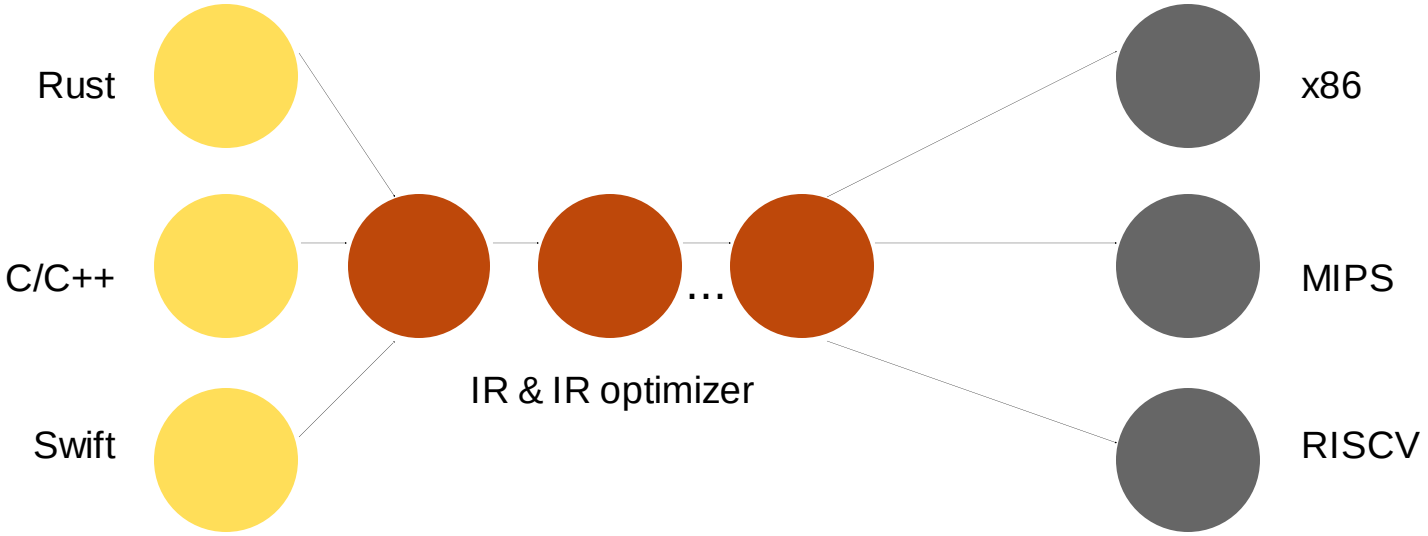
- Jezik niskog nivoa
- Jako tipiziran
- Virtualni registri (%1, %2, …)
- *SSA* forma
- Nezavistan od polaznog izvornog programskog jezika
- Nezavistan od krajnje procesorkse arhitekture

03_LLVM IR(s) & Optimizacije

03_LLVM IR



03_LLVM IR_Optimizator



03_LLVM IR_Optimizator

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

03_LLVM IR_Optimizator

```
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 ====> 0
    - X – 0 =====> X
    - (X*2)-X =====> 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

03_LLVM_MIR

LLVM Machine IR

- Nakon pranja isntrukcija (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 LLVM IR prilikom generisanja Informacija za Debagovanje
- https://llvm.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_CrashAnayzer

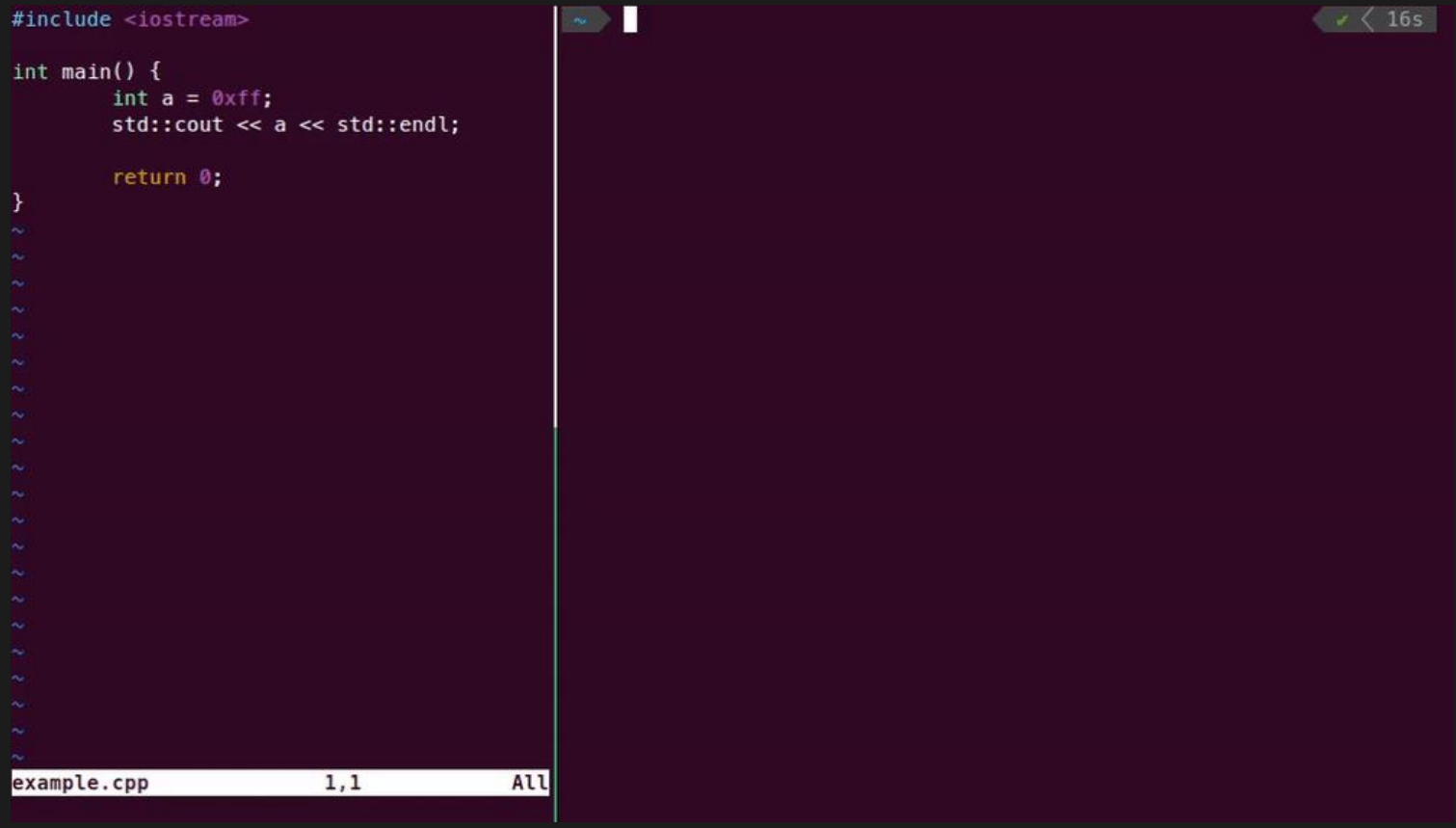
```
$ llvm-crash-analyzer --core-file=core.a.out.9595 ./a.out
Crash Analyzer -- crash analyzer utility

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

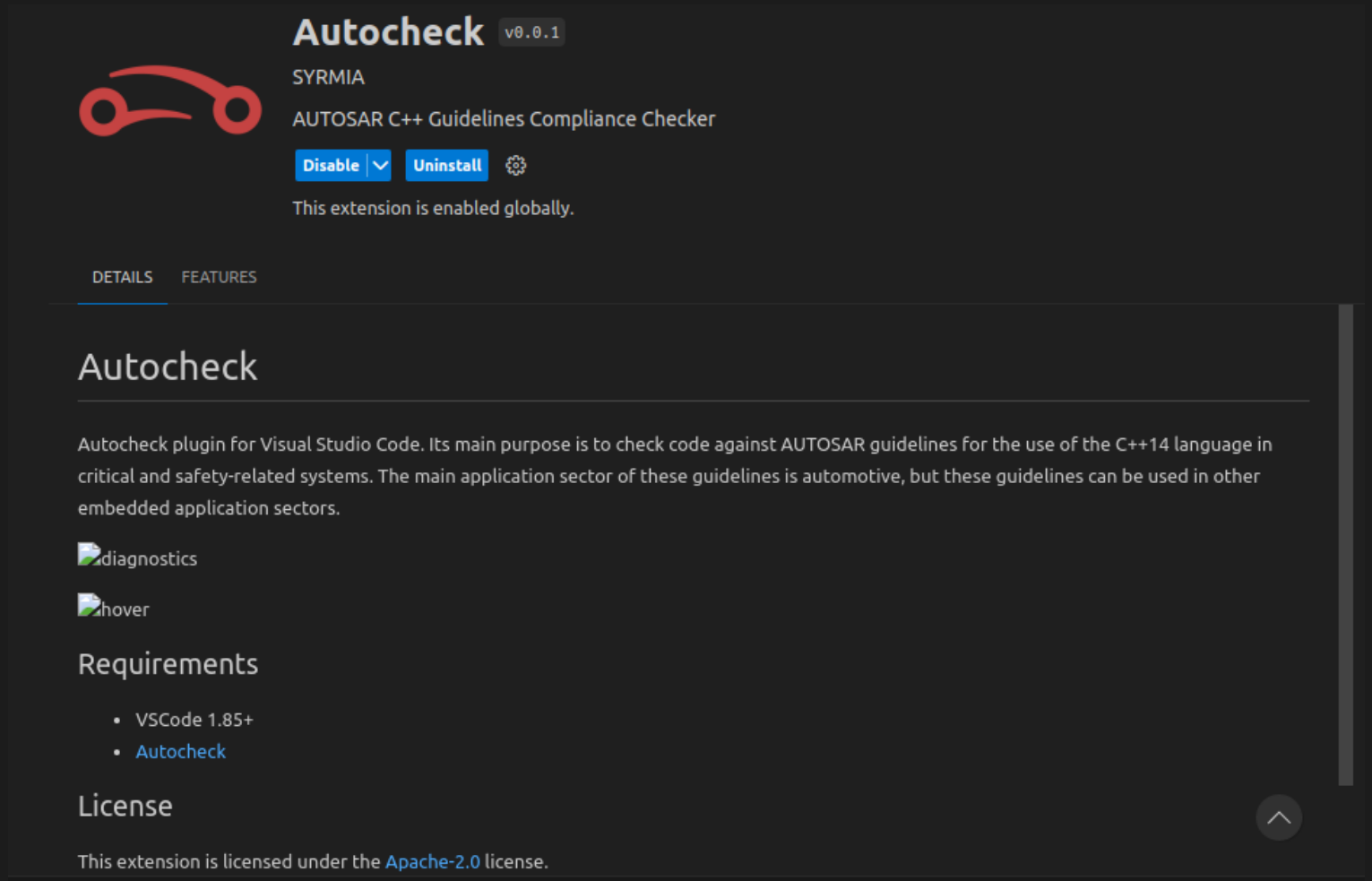
05_Autocheck

- 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 privatan repozitorijum, uskoro javan
  - https://www.phoronix.com/news/Autocheck-LLVM-Safety-Critical
- IDE plugins
  - vscode

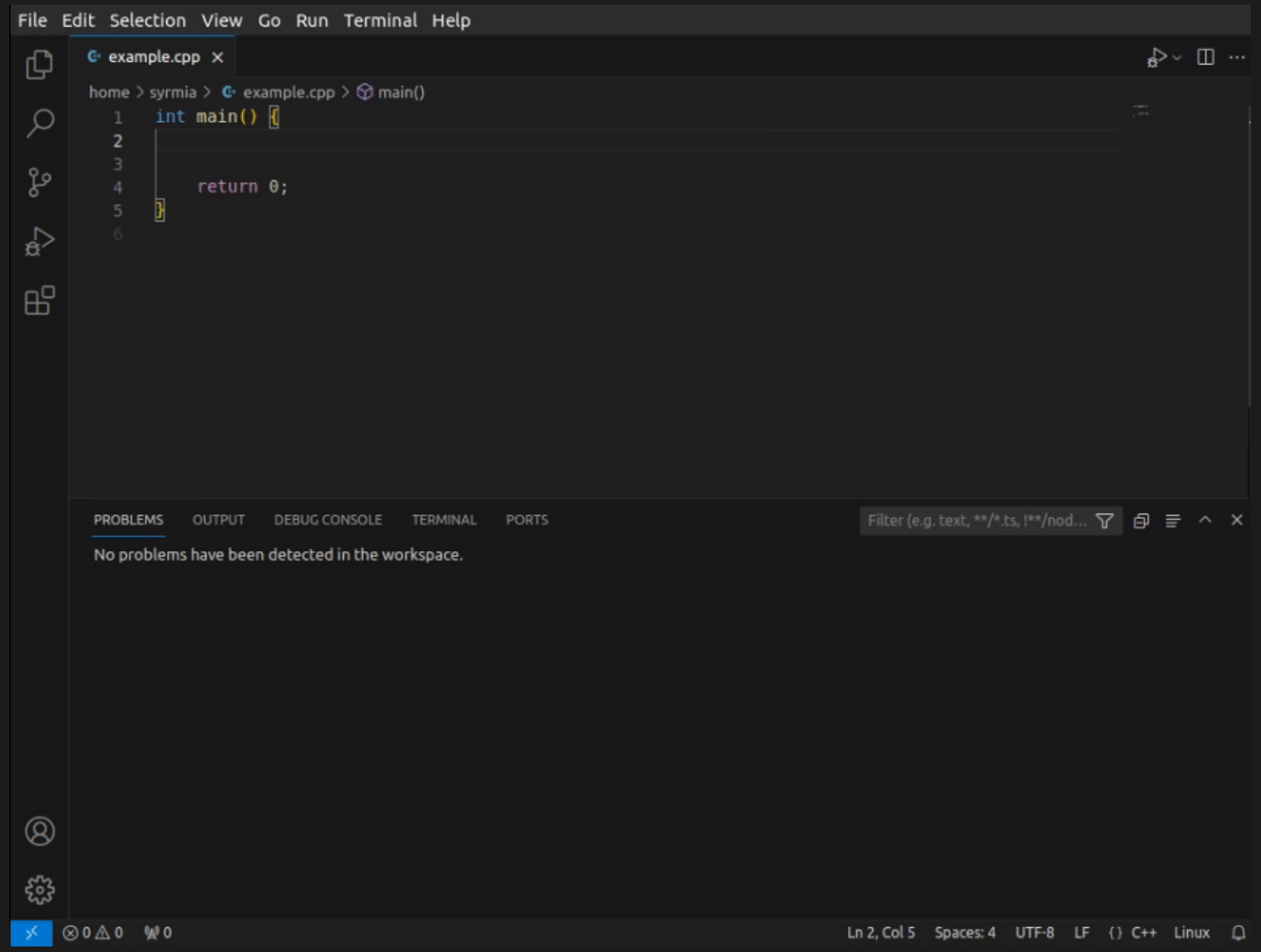
05_Autocheck (Primer)



05_Autocheck (Primer)



05_Autocheck (Primer)



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:

```
CI.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 <cstdio> shall not be used

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

Code:

```
void AutocheckPPCallbacks::InclusionDirective(...) {
  llvm::StringRef FileName = File->getName();

  if (FileName == "cstdio" || FileName == "stdio.h")
    AD.reportWarning(Loc, AutocheckWarnings::clibHeaderUsed);
}
```

Pitanja

thanks

learn ■ share ■ experience
technology