
5Minds
LLVM
In den Untiefen des Compilers
Marco Artz, Bocholt, 16.10.2020

5Minds

Agenda

- LLVM, was ist das?
- Noch ein Compiler?
- Was ist das Besondere an LLVM?
- Wie funktioniert das?
- Was mache ich jetzt damit?

5Minds

LLVM

Was ist das?

5Minds

LLVM

- Werkzeugsammlung für Compiler von Programmiersprachen
- **Früher:** Low Level Virtual Machine, **jetzt:** Eigenständiger Begriff
- Keine virtuelle Maschine (!)
- Modulares, interoperables Konzept

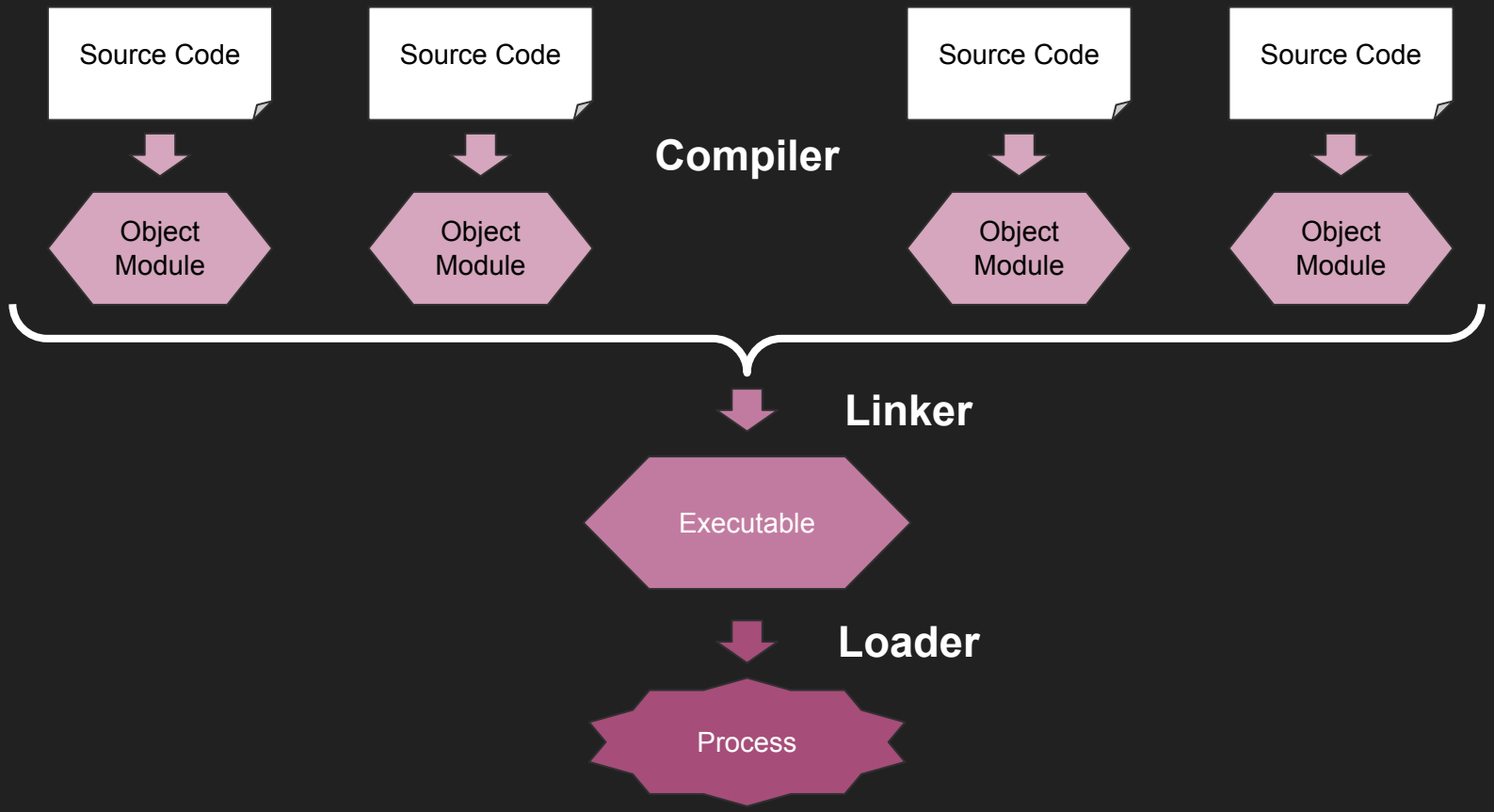
5Minds

Grundlagen: Compilerbau

Noch ein Compiler?

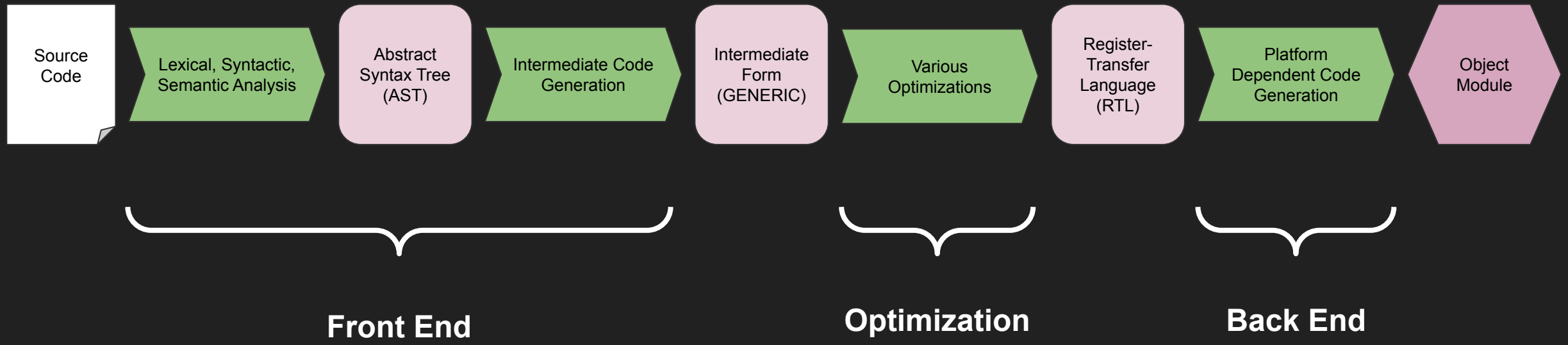
5Minds

Vom Quellcode zum Prozess



5Minds

Aufbau eines klassischen Compilers (GCC)



5Minds

Compiler-Architektur

3-Phasen Design:

- **Front End**: Mehrere Eingabesprachen
- **Optimizer**: Verschiedene Code-Optimierungen
- **Back End**: Unterstützung mehrer Zielarchitekturen

=> Austauschbarkeit (leider nur scheinbar)

5Minds

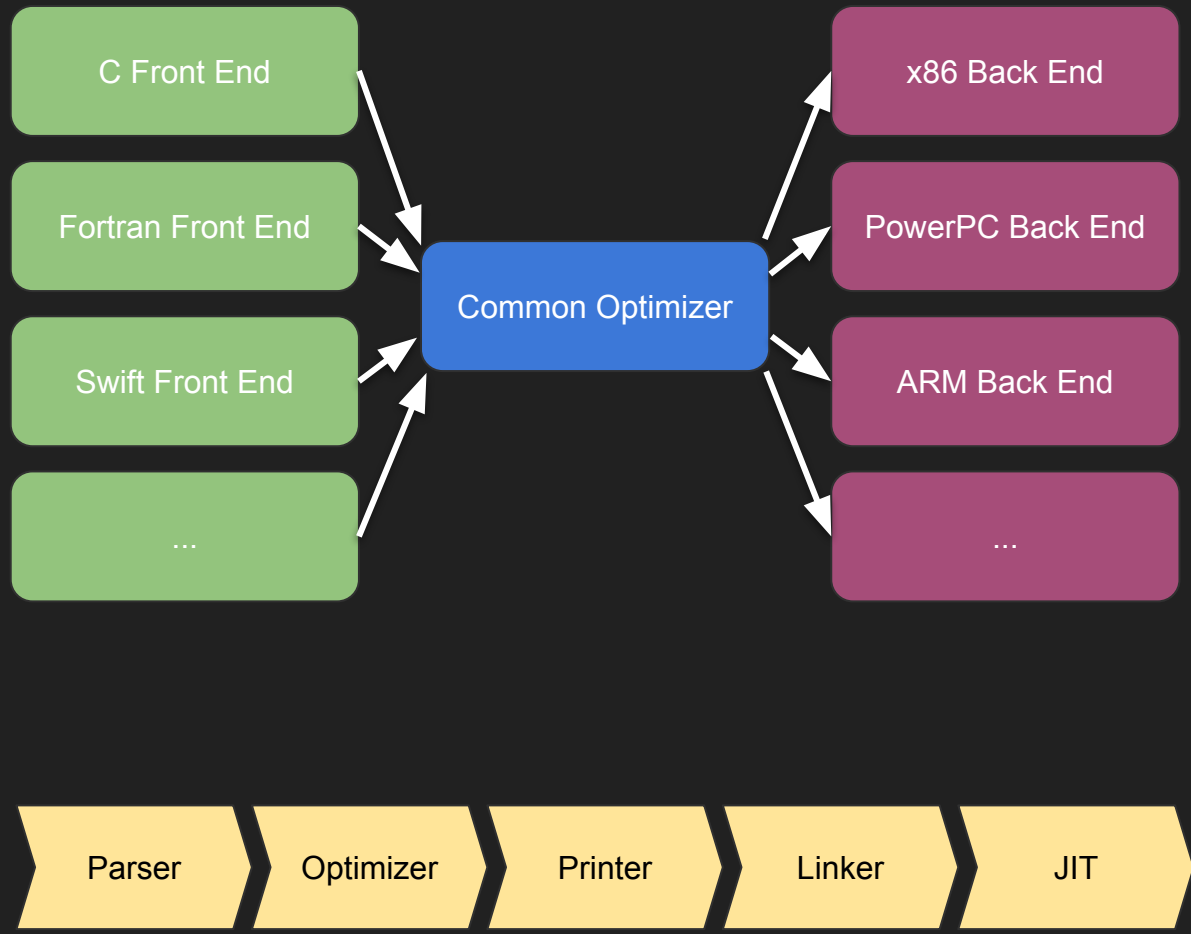
Ziele der LLVM-Architektur

Was ist das Besondere an LLVM?

5Minds

Ziele von LLVM

- 3-Phasen-Compiler
- “Echte” Modularität zwischen den Phasen
- Einziges Zwischenformat: **Intermediate Representation (IR)**
- Frei kombinierbarer Compiler Workflow



5Minds

Frontends (Sprachen) von LLVM

- Ada
- C / C++
- D
- Delphi
- Fortran
- Haskell
- Julia
- Objective-C
- Rust
- Swift
- ...

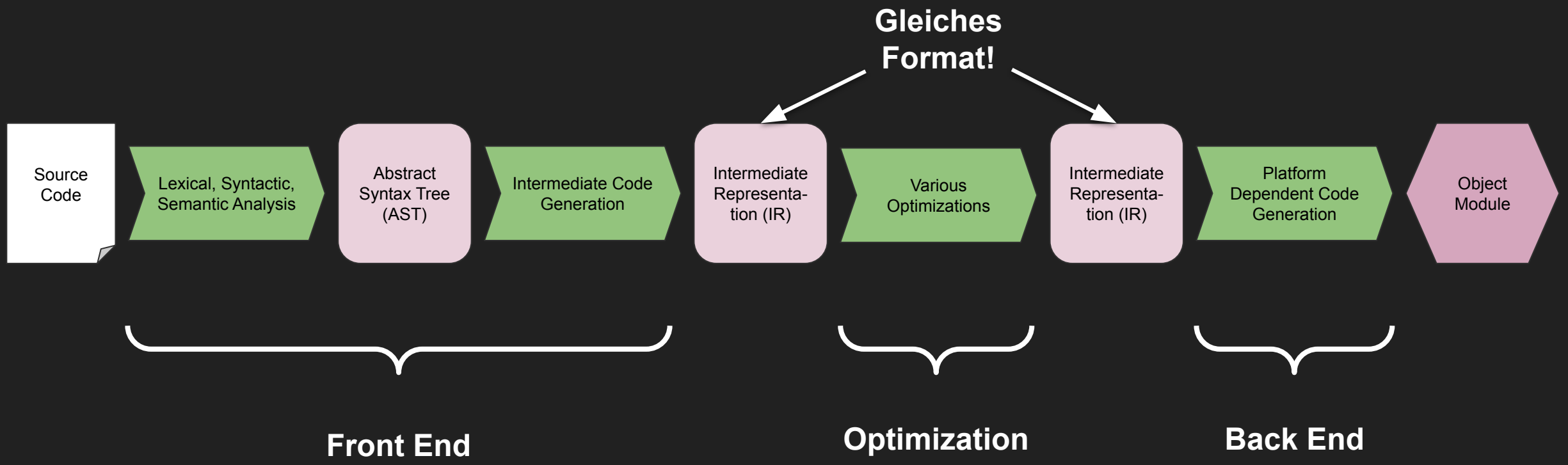
5Minds

Backends (Plattformen) von LLVM

- ***ARM***
- Qualcomm Hexagon
- MIPS
- Nvidia Parallel Thread Execution
- ***PowerPC***
- AMD TeraScale
- AMD Graphics Core Next
- SPARC
- ***z/architecture***
- ***x86 / x86-64***
- XCore
- WASM (WebAssembly)

5Minds

Aufbau von LLVM



5Minds

Intermediate Representation (IR)

- Low-Level Programmiersprache
- Ähnlich zu Assembler
- Abstrakter, streng typisierter RISC Befehlssatz
- Drei äquivalente Formate:
  - Textuell
  - Binär
  - In-Memory

```c
#include <stdio.h>

int main(int argc, char** argv)
{
    printf("Hello, World!\n");
    return 0;
}
```

```
@.str = internal constant [15 x i8] c"Hello, World!\0A\00"

declare i32 @printf(i8*, ...)

define i32 @main(i32 %argc, i8** %argv) nounwind {
entry:
    %tmp1 = getelementptr [15 x i8], [15 x i8]* @.str, i32 0, i32 0
    %tmp2 = call i32 (i8*, ...) @printf( i8* %tmp1 ) nounwind
    ret i32 0
}
```

5Minds

Live Demo

Wie funktioniert das?

5Minds

Anwendungen

Was mache ich damit?

5Minds

Neue Programmiersprache

- Fokus liegt ausschließlich auf den Compiler (Quellcode => IR)
- Algorithmen zur Optimierung können wiederverwendet werden
- Als Ziel kann jedes Backend verwendet werden (theoretisch)

5Minds

Neue Optimierungen

- Anwendbar auf alle Sprachen und Zielsysteme

5Minds

Weitere Zielsysteme

- Erweiterung um weitere Prozessorarchitekturen
- Alle Sprachen des LLVM-Systems werden unterstützt
- Algorithmen zur Optimierung können wiederverwendet werden

5Minds

Verwendung des LLVM APIs

- Code-Generatoren

- Dynamische Übersetzung in die Zielarchitektur:
  - Auslieferung des Zwischencodes (IR)
  - Übersetzung bei Installation
  - Übersetzung Just-In-Time (JIT)

5Minds