

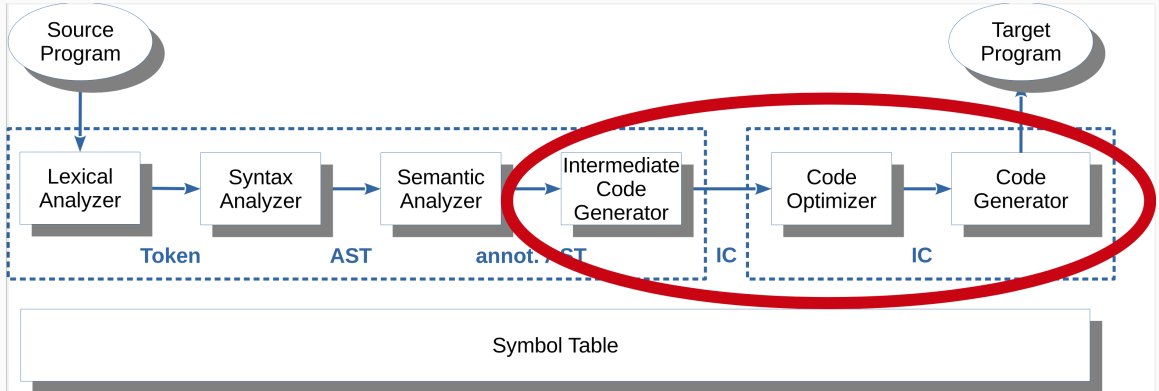
# Überblick Zwischencode

---

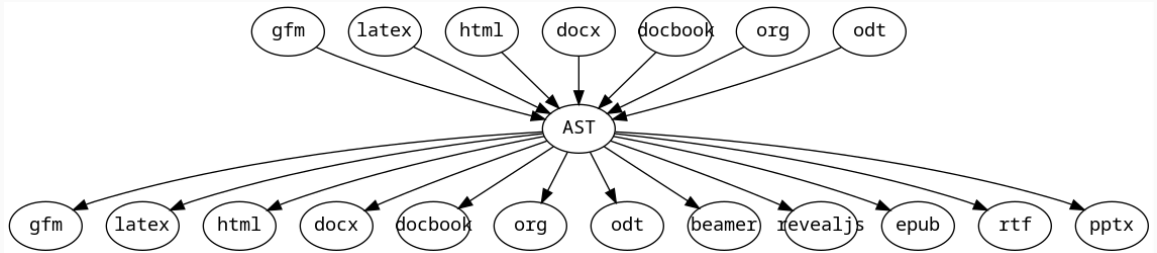
Carsten Gips (FH Bielefeld)

Unless otherwise noted, this work is licensed under CC BY-SA 4.0.

# Einordnung



# AST als Zwischencode (Beispiel Pandoc)



Konsole: `pandoc hello.md -s -t native`

## Zwischenformat: Drei-Adressen-Code

```
i = i+1;  
if (a[i] >= v) {  
    i = 0;  
}
```

```
t1 = i + 1  
i  = t1  
t2 = i * 8  
t3 = a + t2  
if t3 >= v goto L1  
goto L2  
L1: i  = 0  
L2: ...
```

# LLVM IR

## Low Level Virtual Machine

```
int main() {  
    int x = 7;  
    int y = x + 35;  
  
    return 0;  
}
```

```
define i32 @main() #0 {  
    %1 = alloca i32, align 4  
    %2 = alloca i32, align 4  
    %3 = alloca i32, align 4  
    store i32 0, i32* %1, align 4  
    store i32 7, i32* %2, align 4  
    %4 = load i32, i32* %2, align 4  
    %5 = add nsw i32 %4, 35  
    store i32 %5, i32* %3, align 4  
    ret i32 0  
}
```

Beispiel: `clang -emit-llvm -S -o - hello.c`

# Bytecode (Beispiel Python)

```
x = 7  
y = x + 35
```

```
1  0 LOAD_CONST      0 (7)  
   3 STORE_NAME       0 (x)  
  
2  6 LOAD_NAME        0 (x)  
   9 LOAD_CONST       1 (35)  
  12 BINARY_ADD  
  13 STORE_NAME       1 (y)  
  16 LOAD_CONST       2 (None)  
  19 RETURN_VALUE
```

Beispiel: `python -m dis hello.py`

# Bytecode (Beispiel Java)

```
public class Hello {  
  
    void wuppie() {  
        int x = 7;  
        int y = x + 35;  
    }  
  
}
```

Compiled from "Hello.java"

```
public class Hello {  
    public Hello();
```

Code:

0: aload\_0

1: invokespecial #1 // Method java/lang/Object."<init>":()V

4: return

```
void wuppie();
```

Code:

0: bipush 7

2: istore\_1

3: iload\_1

4: bipush 35

6: iadd

7: istore\_2

8: return

```
}
```

# Assembler

```
int main() {  
    int x = 7;  
    int y = x + 35;  
  
    return 0;  
}
```

```
.file    "hello.c"  
.text  
.globl  main  
.type   main, @function  
main:  
.LFBO:  
    .cfi_startproc  
    pushq   %rbp  
    .cfi_def_cfa_offset 16  
    .cfi_offset 6, -16  
    movq    %rsp, %rbp  
    .cfi_def_cfa_register 6  
    movl    $7, -8(%rbp)  
    movl    -8(%rbp), %eax  
    addl    $35, %eax  
    movl    %eax, -4(%rbp)  
    movl    $0, %eax  
    popq    %rbp  
    .cfi_def_cfa 7, 8  
    ret  
    .cfi_endproc  
.LFE0:  
    .size   main, .-main  
    .ident  "GCC: (Ubuntu 7.3.0-27ubuntu1~18.04) 7.3.0"  
    .section .note.GNU-stack,"",@progbits
```



- Compiler generieren aus AST Zwischencode (“*IC*” oder “*IR*”)
- Kein allgemein definiertes Format, große Bandbreite:
  - AST als IR
  - LLVM IR
  - Drei-Adressen-Code
  - Diverse Arten von Bytecode
  - Assemblercode

# LICENSE



Unless otherwise noted, this work is licensed under CC BY-SA 4.0.