

# LLVM IR 简介

魏恒峰

hfwei@nju.edu.cn

2022 年 12 月 19 日



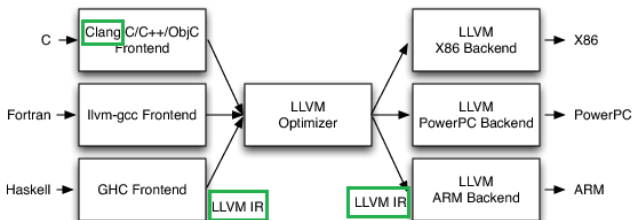
## LLVM Overview

## Latest LLVM Release!

**1 November 2022:** [LLVM 15.0.4](#) is now [available for download](#)! LLVM is publicly available under an open source [License](#). Also, you might want to check



Chris Lattner (1978)



[Home](#)

[Info](#)

[Road](#)

[Tools](#)

[FAQs](#)

[Projects](#)

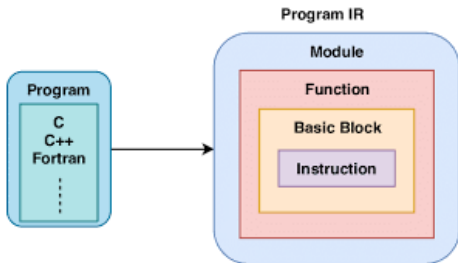
# Clang: a C language family frontend for LLVM

The Clang project provides a language **front-end** and tooling infrastructure for languages in the C language family (C, C++, Objective C/C++, OpenCL, CUDA, and RenderScript) for the [LLVM](#) project. Both a GCC-compatible compiler driver (`clang`) and an MSVC-compatible compiler driver (`clang-cl.exe`) are provided. You can [get and build](#) the source today.

```
clang hello.c -o hello
```

hello @ CompilerExplorer

```
clang -Xclang -ast-dump -c hello.c
```



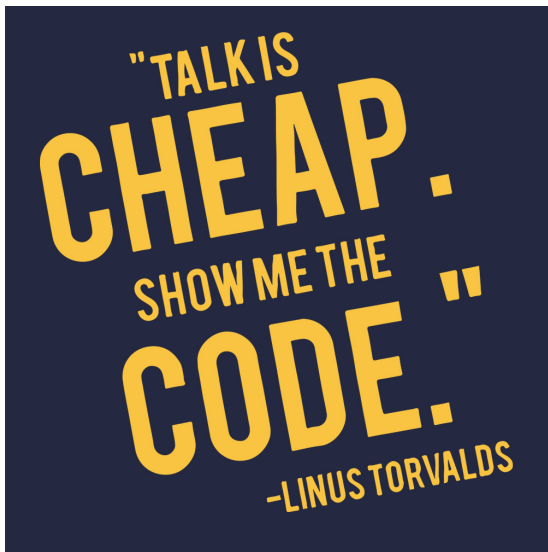
IR: Intermediate Representation

LLVM IR: 带有类型的、介于高级程序设计语言与汇编语言之间



8 章技术内容, 其中 4 章介绍 Maple IR, 另外 4 章基于 Maple IR





```
int factorial(int val);
```

```
int main(int argc, char **argv) {  
    return factorial(val: 2) * 7 == 42;  
}
```

factorial0.c

注意: SysY 中没有函数声明语句。

```

; Function Attrs: noinline nounwind optnone uwtable
define dso_local i32 @main(i32 %0, i8** %1) #0 {
    %3 = alloca i32, align 4
    %4 = alloca i32, align 4
    %5 = alloca i8**, align 8
    store i32 0, i32* %3, align 4
    store i32 %0, i32* %4, align 4
    store i8** %1, i8*** %5, align 8
    %6 = call i32 @factorial(i32 2)
    %7 = mul nsw i32 %6, 7
    %8 = icmp eq i32 %7, 42
    %9 = zext i1 %8 to i32
    ret i32 %9
}

```

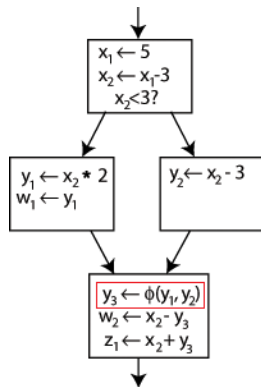
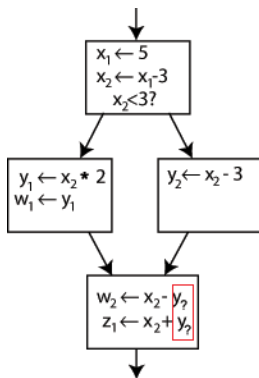
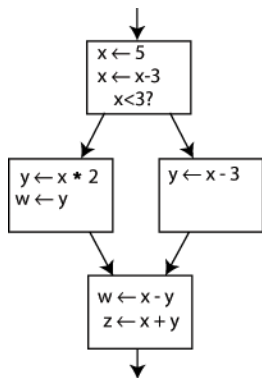
*SSA: Static Single Assignment*

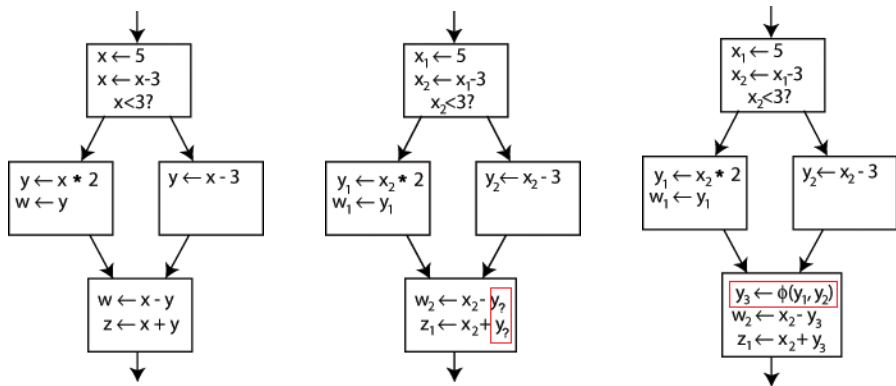
*TAC: Three-Address Code*

`clang -S -emit-llvm factorial0.c -o f0-opt0.ll`

```
; Function Attrs: nounwind uwtable
define dso_local i32 @main(i32 %0, i8** nocapture readnone %1)
{
    %3 = call i32 @factorial(i32 2) #2
    %4 = mul nsw i32 %3, 7
    %5 = icmp eq i32 %4, 42
    %6 = zext i1 %5 to i32
    ret i32 %6
}
```

```
clang -S -emit-llvm factorial0.c -o f0-opt1.ll -O1 -g0
```





$\phi$  函数根据控制流决定选择  $y_1$  还是  $y_2$

```

int factorial(int val);

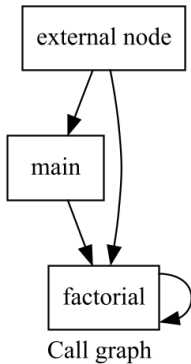
int main(int argc, char **argv) {
    return factorial(val: 2) * 7 == 42;
}

// precondition: val is non-negative
int factorial(int val) {
    if (val == 0) {
        return 1;
    }

    return val * factorial(val: val - 1);
}

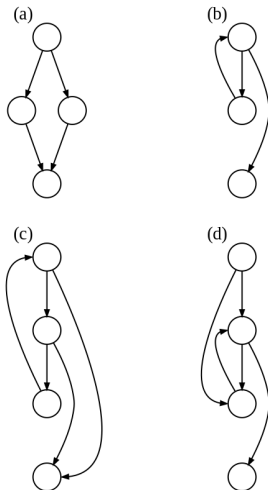
```

factorial1.c





Frances Elizabeth Allen  
(1932 ~ 2020; 2006 Turing Award)

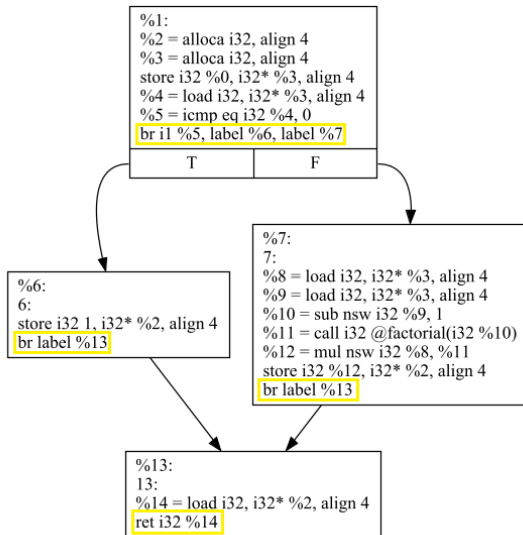


(Intra-procedure) Control Flow Graph (CFG)



```
int factorial(int val) {
    if (val == 0) {
        return 1;
    }

    return val * factorial(val - 1);
}
```



factorial1.c (opt0)

## Instruction Reference

### ○ Terminator Instructions

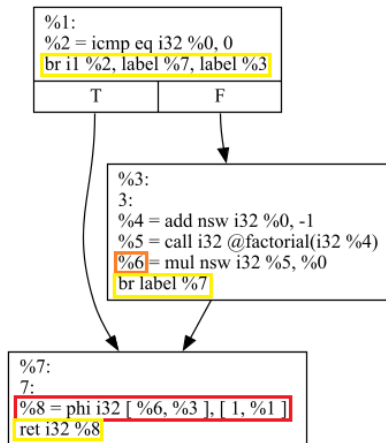
- **'ret'** Instruction
- **'br'** Instruction
- **'switch'** Instruction
- **'indirectbr'** Instruction
- **'invoke'** Instruction
- **'callbr'** Instruction
- **'resume'** Instruction
- **'catchswitch'** Instruction
- **'catchret'** Instruction
- **'cleanupret'** Instruction
- **'unreachable'** Instruction

```

int factorial(int val) {
    if (val == 0) {
        return 1;
    }

    return val * factorial(val: val - 1);
}

```

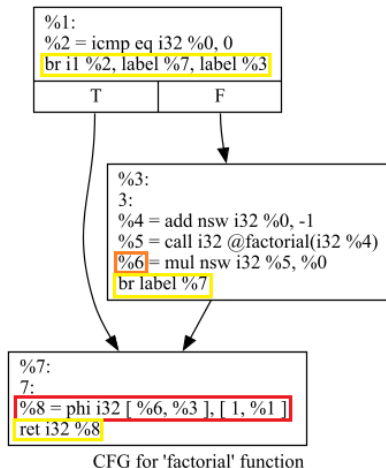


CFG for 'factorial' function

factorial1.c (opt1)

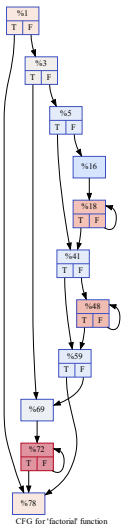
```
int factorial(int val) {
    if (val == 0) {
        return 1;
    }

    return val * factorial(val: val - 1);
}
```



factorial1.c (opt1)

Single-Static Assignment Form and PHI (How to implement it?)



factorial1 (opt3)

```

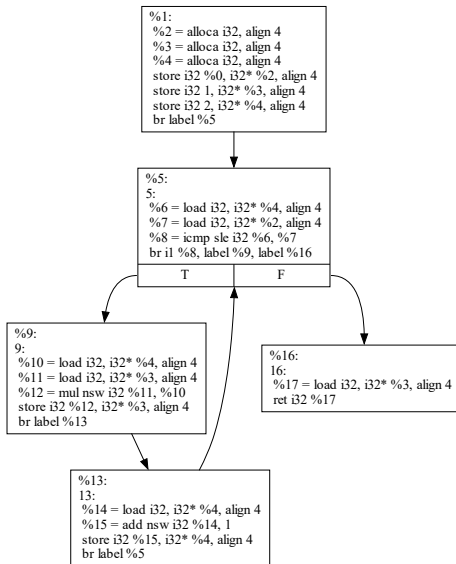
int factorial(int val) {
    int temp = 1;

    for (int i = 2; i <= val; i++) {
        temp *= i;
    }

    return temp;
}

```

factorial2.c (opt0)



CFG for 'factorial' function

```

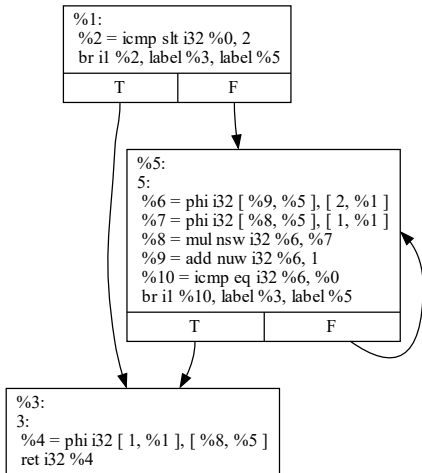
int factorial(int val) {
    int temp = 1;

    for (int i = 2; i <= val; i++) {
        temp *= i;
    }

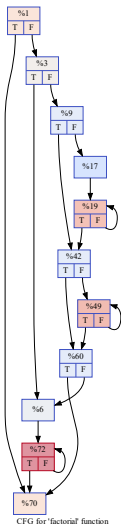
    return temp;
}

```

factorial2.c (opt1)



CFG for 'factorial' function



factorial2 (opt3)



 <https://llvm.org/docs/LangRef.html>



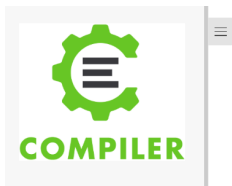
[LLVM Home](#) | [Documentation](#) » [Reference](#) »

# LLVM Language Reference Manual

如何用编程的方式生成 LLVM IR?

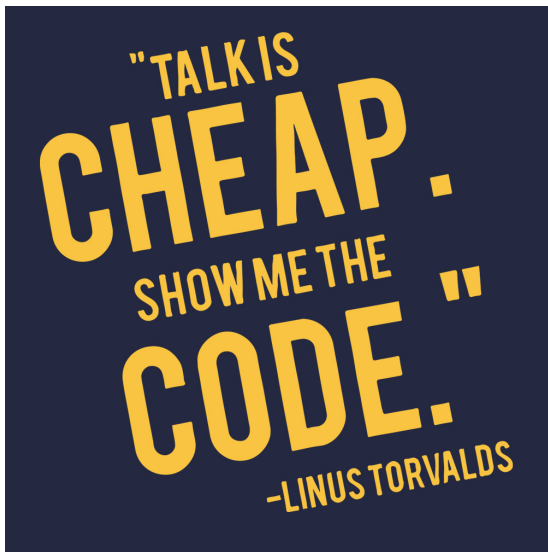
javacpp@github

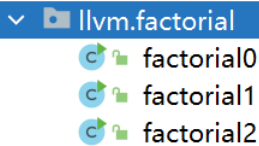
## JavaCPP Presets Platform For LLVM



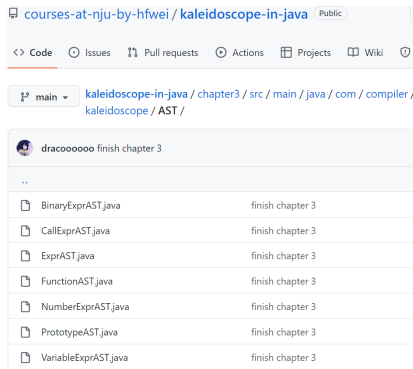
LLVM JAVA API使用手册

准备工作

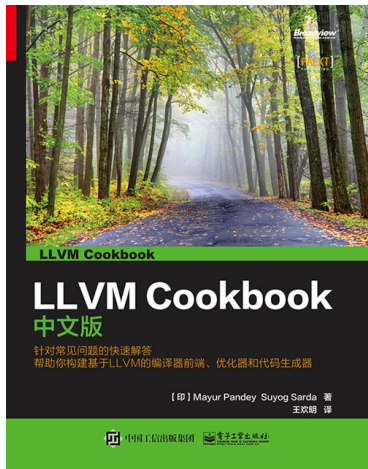




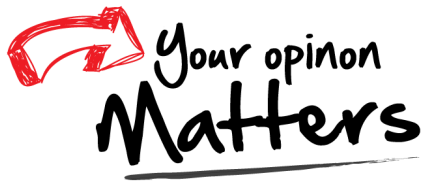
# Kaleidoscope: Implementing a Language with LLVM



kaleidoscope-in-java@github



Thank  
You!



Office 926

hfwei@nju.edu.cn