南开大学

Nankai University

南　　开　　大　　学

计　算　机　学　院

语言处理系统的完整工作过程实验报告

语言处理系统探究

杨乔钦

年级：2023 级

专业：计算机科学与技术

指导教师：王刚

2025 年 9 月 25 日

摘要

123

关键字：Parallel

目录

一、 预处理器

< 数字 > < 文件名 > < 标志 > 这是编译器预处理器产生的注释（注释行），用于追踪源代码中每一行的来源，便于调试、错误追踪和调试信息生成。例如：

$$\#1\text{”}test.cpp\text{”}$$

表示接下来要处理的代码来源于”test.cpp”，且在原始源文件 test.cpp 中，从第 1 行开始。

$$\#37\text{”}/usr/include/c++/13/iostream\text{”}3$$

表示接下来的代码来自于 <iostream> 文件；原始 iostream 文件的第 37 行；3 表示这是一个系统头文件。

之后文件都是处理头文件中的内容，将各个头文件中包含的头文件进行处理。直到 test.i 文件的末尾，即文件的第 36583 行，我们才开始处理源文件 test.cpp 中的第二行内容，之后便全是源文件 test.cpp 中的内容了。

预处理器的主要任务包括：

- 宏定义和展开：处理宏定义（如 #define）并在代码中展开宏。
- 文件包含：处理文件包含指令（如 #include），将包含的文件内容插入到当前文件中。
- 条件编译：处理条件编译指令（如 #ifdef、#ifndef、#if、#else、#elif、#endif），根据条件决定哪些代码被编译。
- 其他预处理指令：处理其他预处理指令，如 #undef、#pragma 等。

预处理器的输出是一个纯文本文件，包含了所有预处理指令处理后的代码，这个文件随后会被编译器用来生成目标代码。

二、 编译器

(一) 词法分析

词法分析器的主要任务是将源代码转换为一系列的记号（tokens）。这些记号是源代码中有意义的基本单位，如关键字、标识符、操作符、字面值等。词法分析器通过扫描源代码，识别出这些记号，并为每个记号分配一个类型和属性。

词法分析器的工作流程通常包括以下几个步骤：

- 输入读取：词法分析器从源代码文件中逐字符读取输入。
- 模式匹配：使用正则表达式或有限状态机（FSM）来识别不同类型的记号。
- 记号生成：当识别出一个记号时，词法分析器会创建一个记号对象，包含记号的类型、值和位置等信息。
- 忽略空白和注释：词法分析器通常会忽略源代码中的空白字符和注释，因为它们对程序的语义没有影响。
- 错误处理：如果遇到无法识别的字符或模式，词法分析器会报告词法错误。
- 输出记号流：词法分析器将生成的记号按顺序输出，供后续的语法分析器使用。

这些格式统一为 <token_type> '<literal>' [属性...] Loc=< 文件名: 行号: 列号 > 从前到后依次是词法单元类型、实际文本内容、可选属性、源文件中位置。

在输入命令 $clang - E - Xclang - dump - tokens main.c$ 后，词法分析器会输出源代码中的所有记号及其类型。例如，输出中的 $identifier$ 表示标识符，$keyword$ 表示关键字，$semi$ 表示分号等。并且所有内容都是 test.i 中的内容，但是 # 后面的语句不会被识别。

(二) 语法分析

语法分析器的主要任务是将词法分析器生成的记号流转换为抽象语法树（AST）。抽象语法树是一种树状结构，表示源代码的语法结构和层次关系。

语法分析器的工作流程通常包括以下几个步骤：

- 输入读取：语法分析器从词法分析器生成的记号流中读取输入。
- 语法规则：语法分析器使用一系列语法规则来解析记号流，生成抽象语法树。
- 错误处理：如果遇到无法识别的语法结构，语法分析器会报告语法错误。
- 输出抽象语法树：语法分析器将生成的抽象语法树输出，供后续的语义分析器使用。
- 抽象语法树的节点类型：抽象语法树的节点类型通常包括表达式节点、语句节点、声明节点等，每种节点类型表示不同的语法结构。

Clang AST 每行节点的通用格式为：< 树形符号 >< 节点类型 > < 地址 > < 源码位置 > < 其他属性...>

其中不同类型节点的具体格式如下：

Decl 类节点（声明）

字段名	示例/说明
节点类型	`FunctionDecl`, `VarDecl`, `ParmVarDecl`, `UsingDirectiveDecl`
名字 + 类型	`'int'`, `'int ()'`, `'ostream'`
使用状态	`used`, `referenced`
初始化信息	`cinit`, `callinit`
存储类	`extern`, `static`（视情况而定）

Stmt 类节点（语句）

字段名	示例/说明
节点类型	`CompoundStmt`, `IfStmt`, `ReturnStmt`, `WhileStmt`
地址	有时省略
位置信息	主要包含代码在源码中的位置
名称字段	无（语句通常不带名称）
结构作用	用于建立语法树嵌套结构

Expr 类节点（表达式）

字段名	示例/说明
节点类型	`BinaryOperator`, `DeclRefExpr`, `CallExpr`, `CXXOperatorCallExpr`
值类别	`lvalue`, `xvalue`, `prvalue`
类型	表达式求值结果，如 `'int'`
操作符	如 `'+'`, `'*'`, `'='`
字面值	如 `IntegerLiteral 'int' 0`
隐式转换信息	如 `<FunctionToPointerDecay>`, `<LValueToRValue>`

举例说明：`|-UsingDirectiveDecl 0x617d83d7d7d8 <test.cpp:2:1, col:17> col:17 Namespace 0x617d82f2e528 'std'`

为例，UsingDirectiveDecl 表示一个 using directive，即 using 指令；0x617d83d7d7d8 这是编译器给这个 AST 节点分配的内存地址（或唯一 id）；<test.cpp:2:1, col:17> col:17 表示代码原位置；Namespace 0x617d82f2e528 'std' 表示这个 using 指令引入的命名空间是 std ，其节点位置在 0x617d82f2e528，这里只是引用。类似的，其他节点类型也有各自的格式。

(三) 语义分析

语义分析器的主要任务是检查抽象语法树（AST）是否符合语言的语义规则，并进行类型检查、作用域管理和符号表维护等工作。

语义分析器的工作流程通常包括以下几个步骤:

- 输入读取：语义分析器从语法分析器生成的抽象语法树中读取输入。
- 语义规则：语义分析器使用一系列语义规则来检查抽象语法树，并进行类型检查、作用域管理和符号表维护。
- 错误处理：如果遇到语义错误，语义分析器会报告错误。
- 输出符号表：语义分析器将生成的符号表输出，供后续的中间代码生成器使用。
- 符号表的内容：符号表通常包含变量、函数、类等符号的信息，如名称、类型、作用域等。

(四) 中间代码生成

以 gcc -fdump-tree-all-graph test.cpp -o test gcc -fdump-rtl-all-graph test.cpp -o test 命令，我们生成了中间代码的多阶段输出，其中.dot 文件即文件的控制流图。对于 tree 阶段和 rtl 阶段，我们得到了不同层次的中间表示，每次优化 pass 都会生成一个新的.dot 文件。例如生成图如下??、??所示，图 1 图 2 分别是 rtl 和 tree 的控制流图，图 3 图 4 分别是 rtl 和 tree 的-O2 优化控制流图。

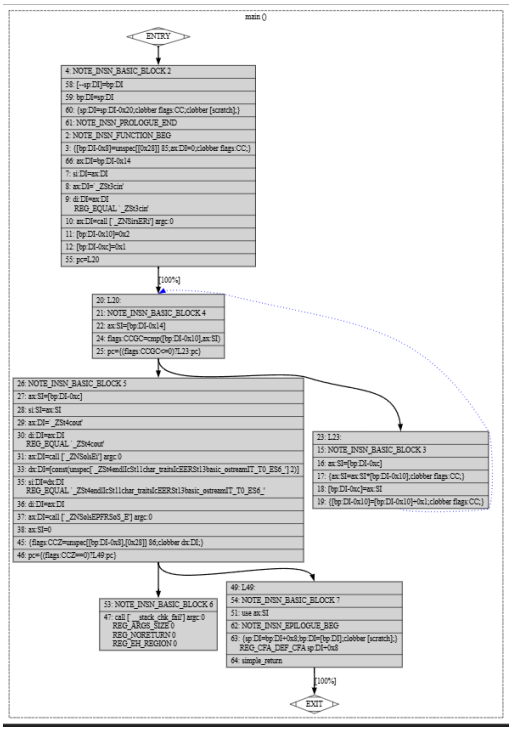

图 1: 图 1

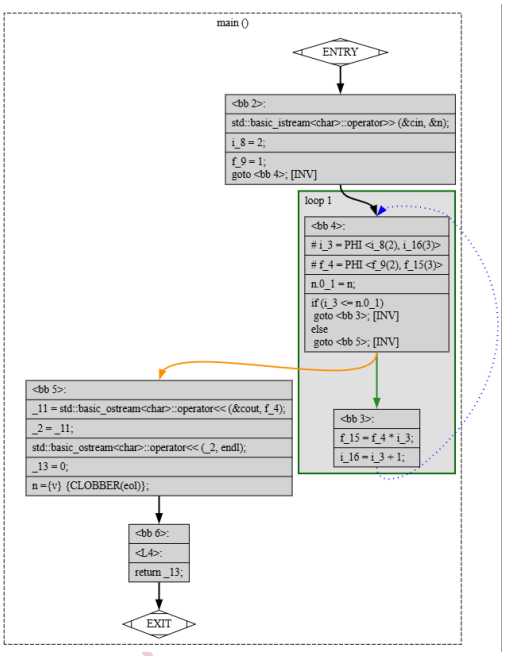

图 2: 图 2

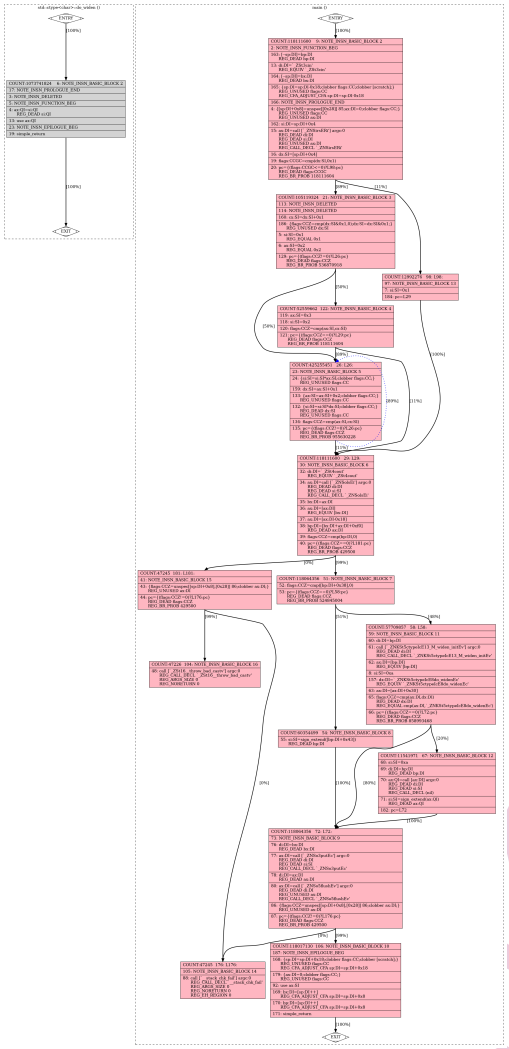

图 3: 图 3

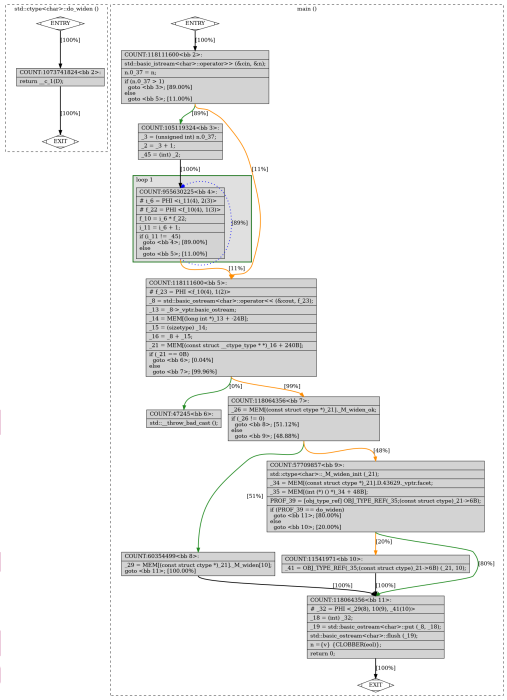

图 4: 图 4

（五） 代码优化

代码优化是编译器中的一个重要阶段，旨在提高生成代码的性能和效率。代码优化可以分为两类：机器无关优化和机器相关优化。对于中间代码生成，利用 clang -S -emit-llvm main.c 指令，我们可以通过-O1、-O2、-O3 等不同的优化级别来生成不同优化程度的 LLVM 中间代码，优化后代码如下。

O1 优化

```
; ModuleID = 'test.cpp'
source_filename = "test.cpp"
target datalayout = "e-m:e-p270:32:32-p271:32:32-p272:64:64-i64:64-i128:128-
    f80:128-n8:16:32:64-S128"
target triple = "x86_64-pc-linux-gnu"

module asm ".globl _ZSt21ios_base_library_initv"

%"class.std::basic_istream" = type { ptr, i64, %"class.std::basic_ios" }
```

```
%"class.std::basic_ios" = type { %"class.std::ios_base", ptr, i8, i8, ptr, ptr, ptr, ptr }
%"class.std::ios_base" = type { ptr, i64, i64, i32, i32, i32, ptr, %"struct.std::ios_base::_Words", [8 x %"struct.std::ios_base::_Words"], i32, ptr, %"class.std::locale" }
%"struct.std::ios_base::_Words" = type { ptr, i64 }
%"class.std::locale" = type { ptr }
%"class.std::basic_ostream" = type { ptr, %"class.std::basic_ios" }
%"class.std::ctype" = type <{ %"class.std::locale::facet.base", [4 x i8], ptr, i8, [7 x i8], ptr, ptr, ptr, i8, [256 x i8], [256 x i8], i8, [6 x i8] }>
%"class.std::locale::facet.base" = type <{ ptr, i32 }>

@_ZSt3cin = external global %"class.std::basic_istream", align 8
@_ZSt4cout = external global %"class.std::basic_ostream", align 8

; Function Attrs: mustprogress norecurse uwtable
define dso_local noundef i32 @main() local_unnamed_addr #0 {
  %1 = alloca i32, align 4
  call void @llvm.lifetime.start.p0(i64 4, ptr nonnull %1) #4
  %2 = call noundef nonnull align 8 dereferenceable(16) ptr @_ZNSirsERi(ptr noundef nonnull align 8 dereferenceable(16) @_ZSt3cin, ptr noundef nonnull align 4 dereferenceable(4) %1)
  %3 = load i32, ptr %1, align 4, !tbaa !5
  %4 = icmp slt i32 %3, 2
  br i1 %4, label %11, label %5

5:                                                ; preds = %0, %5
  %6 = phi i32 [ %8, %5 ], [ 1, %0 ]
  %7 = phi i32 [ %9, %5 ], [ 2, %0 ]
  %8 = mul nsw i32 %6, %7
  %9 = add nuw i32 %7, 1
  %10 = icmp eq i32 %7, %3
  br i1 %10, label %11, label %5, !llvm.loop !9

11:                                               ; preds = %5, %0
  %12 = phi i32 [ 1, %0 ], [ %8, %5 ]
  %13 = call noundef nonnull align 8 dereferenceable(8) ptr @_ZNSolsEi(ptr noundef nonnull align 8 dereferenceable(8) @_ZSt4cout, i32 noundef %12)
  %14 = load ptr, ptr %13, align 8, !tbaa !12
  %15 = getelementptr i8, ptr %14, i64 -24
  %16 = load i64, ptr %15, align 8
  %17 = getelementptr inbounds i8, ptr %13, i64 %16
  %18 = getelementptr inbounds %"class.std::basic_ios", ptr %17, i64 0, i32 5
  %19 = load ptr, ptr %18, align 8, !tbaa !14
  %20 = icmp eq ptr %19, null
  br i1 %20, label %21, label %22

```

```
21:                                               ; preds = %11
  call void @_ZSt16__throw_bad_castv() #5
  unreachable

22:                                               ; preds = %11
  %23 = getelementptr inbounds %"class.std::ctype", ptr %19, i64 0, i32 8
  %24 = load i8, ptr %23, align 8, !tbaa !24
  %25 = icmp eq i8 %24, 0
  br i1 %25, label %29, label %26

26:                                               ; preds = %22
  %27 = getelementptr inbounds %"class.std::ctype", ptr %19, i64 0, i32 9,
      i64 10
  %28 = load i8, ptr %27, align 1, !tbaa !27
  br label %34

29:                                               ; preds = %22
  call void @_ZNKSt5ctypeIcE13_M_widen_initEv(ptr noundef nonnull align 8
      dereferenceable(570) %19)
  %30 = load ptr, ptr %19, align 8, !tbaa !12
  %31 = getelementptr inbounds ptr, ptr %30, i64 6
  %32 = load ptr, ptr %31, align 8
  %33 = call noundef signext i8 %32(ptr noundef nonnull align 8
      dereferenceable(570) %19, i8 noundef signext 10)
  br label %34

34:                                               ; preds = %26, %29
  %35 = phi i8 [ %28, %26 ], [ %33, %29 ]
  %36 = call noundef nonnull align 8 dereferenceable(8) ptr @_ZNSo3putEc(ptr
      noundef nonnull align 8 dereferenceable(8) %13, i8 noundef signext %35)
  %37 = call noundef nonnull align 8 dereferenceable(8) ptr @_ZNSo5flushEv(
      ptr noundef nonnull align 8 dereferenceable(8) %36)
  call void @llvm.lifetime.end.p0(i64 4, ptr nonnull %1) #4
  ret i32 0
}

; Function Attrs: mustprogress nocallback nofree nosync nounwind willreturn
    memory(argmem: readwrite)
declare void @llvm.lifetime.start.p0(i64 immarg, ptr nocapture) #1

declare noundef nonnull align 8 dereferenceable(16) ptr @_ZNSirsERi(ptr
    noundef nonnull align 8 dereferenceable(16), ptr noundef nonnull align 4
    dereferenceable(4)) local_unnamed_addr #2

declare noundef nonnull align 8 dereferenceable(8) ptr @_ZNSolsEi(ptr noundef
     nonnull align 8 dereferenceable(8), i32 noundef) local_unnamed_addr #2

; Function Attrs: mustprogress nocallback nofree nosync nounwind willreturn
```

```
    memory(argmem: readwrite)
declare void @llvm.lifetime.end.p0(i64 immarg, ptr nocapture) #1

declare noundef nonnull align 8 dereferenceable(8) ptr @_ZNSo3putEc(ptr
    noundef nonnull align 8 dereferenceable(8), i8 noundef signext)
    local_unnamed_addr #2

declare noundef nonnull align 8 dereferenceable(8) ptr @_ZNSo5flushEv(ptr
    noundef nonnull align 8 dereferenceable(8)) local_unnamed_addr #2

; Function Attrs: noreturn
declare void @_ZSt16__throw_bad_castv() local_unnamed_addr #3

declare void @_ZNKSt5ctypeIcE13_M_widen_initEv(ptr noundef nonnull align 8
    dereferenceable(570)) local_unnamed_addr #2

attributes #0 = { mustprogress norecurse uwtable "min-legal-vector-width"="0"
     "no-trapping-math"="true" "stack-protector-buffer-size"="8" "target-cpu
    "="x86-64" "target-features"="+cmov,+cx8,+fxsr,+mmx,+sse,+sse2,+x87" "
    tune-cpu"="generic" }
attributes #1 = { mustprogress nocallback nofree nosync nounwind willreturn
    memory(argmem: readwrite) }
attributes #2 = { "no-trapping-math"="true" "stack-protector-buffer-size"="8"
     "target-cpu"="x86-64" "target-features"="+cmov,+cx8,+fxsr,+mmx,+sse,+
    sse2,+x87" "tune-cpu"="generic" }
attributes #3 = { noreturn "no-trapping-math"="true" "stack-protector-buffer-
    size"="8" "target-cpu"="x86-64" "target-features"="+cmov,+cx8,+fxsr,+mmx
    ,+sse,+sse2,+x87" "tune-cpu"="generic" }
attributes #4 = { nounwind }
attributes #5 = { noreturn }

!llvm.module.flags = !{!0, !1, !2, !3}
!llvm.ident = !{!4}

!0 = !{i32 1, !"wchar_size", i32 4}
!1 = !{i32 8, !"PIC Level", i32 2}
!2 = !{i32 7, !"PIE Level", i32 2}
!3 = !{i32 7, !"uwtable", i32 2}
!4 = !{!"Ubuntu clang version 18.1.3 (1ubuntu1)"}
!5 = !{!6, !6, i64 0}
!6 = !{!"int", !7, i64 0}
!7 = !{!"omnipotent char", !8, i64 0}
!8 = !{!"Simple C++ TBAA"}
!9 = distinct !{!9, !10, !11}
!10 = !{!"llvm.loop.mustprogress"}
!11 = !{!"llvm.loop.unroll.disable"}
!12 = !{!13, !13, i64 0}
!13 = !{!"vtable pointer", !8, i64 0}
```

```
!14 = !{!15, !20, i64 240}
!15 = !{!"_ZTSSt9basic_iosIcSt11char_traitsIcEE", !16, i64 0, !20, i64 216, !7, i64 224, !23, i64 225, !20, i64 232, !20, i64 240, !20, i64 248, !20, i64 256}
!16 = !{!"_ZTSSt8ios_base", !17, i64 8, !17, i64 16, !18, i64 24, !19, i64 28, !19, i64 32, !20, i64 40, !21, i64 48, !7, i64 64, !6, i64 192, !20, i64 200, !22, i64 208}
!17 = !{!"long", !7, i64 0}
!18 = !{!"_ZTSSt13_Ios_Fmtflags", !7, i64 0}
!19 = !{!"_ZTSSt12_Ios_Iostate", !7, i64 0}
!20 = !{!"any pointer", !7, i64 0}
!21 = !{!"_ZTSNSt8ios_base6_WordsE", !20, i64 0, !17, i64 8}
!22 = !{!"_ZTSSt6locale", !20, i64 0}
!23 = !{!"bool", !7, i64 0}
!24 = !{!25, !7, i64 56}
!25 = !{!"_ZTSSt5ctypeIcE", !26, i64 0, !20, i64 16, !23, i64 24, !20, i64 32, !20, i64 40, !20, i64 48, !7, i64 56, !7, i64 57, !7, i64 313, !7, i64 569}
!26 = !{!"_ZTSNSt6locale5facetE", !6, i64 8}
!27 = !{!7, !7, i64 0}
```

O2 优化

```
; ModuleID = 'test.cpp'
source_filename = "test.cpp"
target datalayout = "e-m:e-p270:32:32-p271:32:32-p272:64:64-i64:64-i128:128-f80:128-n8:16:32:64-S128"
target triple = "x86_64-pc-linux-gnu"

module asm ".globl _ZSt21ios_base_library_initv"

%"class.std::basic_istream" = type { ptr, i64, %"class.std::basic_ios" }
%"class.std::basic_ios" = type { %"class.std::ios_base", ptr, i8, i8, ptr, ptr, ptr, ptr }
%"class.std::ios_base" = type { ptr, i64, i64, i32, i32, i32, ptr, %"struct.std::ios_base::_Words", [8 x %"struct.std::ios_base::_Words"], i32, ptr, %"class.std::locale" }
%"struct.std::ios_base::_Words" = type { ptr, i64 }
%"class.std::locale" = type { ptr }
%"class.std::basic_ostream" = type { ptr, %"class.std::basic_ios" }
%"class.std::ctype" = type <{ %"class.std::locale::facet.base", [4 x i8], ptr, i8, [7 x i8], ptr, ptr, ptr, i8, [256 x i8], [256 x i8], i8, [6 x i8] }>
%"class.std::locale::facet.base" = type <{ ptr, i32 }>

@_ZSt3cin = external global %"class.std::basic_istream", align 8
@_ZSt4cout = external global %"class.std::basic_ostream", align 8

; Function Attrs: mustprogress norecurse uwtable
```

```
define dso_local noundef i32 @main() local_unnamed_addr #0 {
  %1 = alloca i32, align 4
  call void @llvm.lifetime.start.p0(i64 4, ptr nonnull %1) #5
  %2 = call noundef nonnull align 8 dereferenceable(16) ptr @_ZNSirsERi(ptr
      noundef nonnull align 8 dereferenceable(16) @_ZSt3cin, ptr noundef
      nonnull align 4 dereferenceable(4) %1)
  %3 = load i32, ptr %1, align 4, !tbaa !5
  %4 = icmp slt i32 %3, 2
  br i1 %4, label %35, label %5

5:                                                ; preds = %0
  %6 = add nsw i32 %3, -1
  %7 = icmp ult i32 %3, 9
  br i1 %7, label %26, label %8

8:                                                ; preds = %5
  %9 = and i32 %6, -8
  %10 = or disjoint i32 %9, 2
  br label %11

11:                                               ; preds = %11, %8
  %12 = phi i32 [ 0, %8 ], [ %19, %11 ]
  %13 = phi <4 x i32> [ <i32 1, i32 1, i32 1, i32 1>, %8 ], [ %17, %11 ]
  %14 = phi <4 x i32> [ <i32 1, i32 1, i32 1, i32 1>, %8 ], [ %18, %11 ]
  %15 = phi <4 x i32> [ <i32 2, i32 3, i32 4, i32 5>, %8 ], [ %20, %11 ]
  %16 = add <4 x i32> %15, <i32 4, i32 4, i32 4, i32 4>
  %17 = mul <4 x i32> %13, %15
  %18 = mul <4 x i32> %14, %16
  %19 = add nuw i32 %12, 8
  %20 = add <4 x i32> %15, <i32 8, i32 8, i32 8, i32 8>
  %21 = icmp eq i32 %19, %9
  br i1 %21, label %22, label %11, !llvm.loop !9

22:                                               ; preds = %11
  %23 = mul <4 x i32> %18, %17
  %24 = call i32 @llvm.vector.reduce.mul.v4i32(<4 x i32> %23)
  %25 = icmp eq i32 %6, %9
  br i1 %25, label %35, label %26

26:                                               ; preds = %5, %22
  %27 = phi i32 [ 1, %5 ], [ %24, %22 ]
  %28 = phi i32 [ 2, %5 ], [ %10, %22 ]
  br label %29

29:                                               ; preds = %26, %29
  %30 = phi i32 [ %32, %29 ], [ %27, %26 ]
  %31 = phi i32 [ %33, %29 ], [ %28, %26 ]
  %32 = mul nsw i32 %30, %31
```

```
  %33 = add nuw i32 %31, 1
  %34 = icmp eq i32 %31, %3
  br i1 %34, label %35, label %29, !llvm.loop !13

35:                                               ; preds = %29, %22, %0
  %36 = phi i32 [ 1, %0 ], [ %24, %22 ], [ %32, %29 ]
  %37 = call noundef nonnull align 8 dereferenceable(8) ptr @_ZNSolsEi(ptr
      noundef nonnull align 8 dereferenceable(8) @_ZSt4cout, i32 noundef %36)
  %38 = load ptr, ptr %37, align 8, !tbaa !14
  %39 = getelementptr i8, ptr %38, i64 -24
  %40 = load i64, ptr %39, align 8
  %41 = getelementptr inbounds i8, ptr %37, i64 %40
  %42 = getelementptr inbounds %"class.std::basic_ios", ptr %41, i64 0, i32 5
  %43 = load ptr, ptr %42, align 8, !tbaa !16
  %44 = icmp eq ptr %43, null
  br i1 %44, label %45, label %46

45:                                               ; preds = %35
  call void @_ZSt16__throw_bad_castv() #6
  unreachable

46:                                               ; preds = %35
  %47 = getelementptr inbounds %"class.std::ctype", ptr %43, i64 0, i32 8
  %48 = load i8, ptr %47, align 8, !tbaa !26
  %49 = icmp eq i8 %48, 0
  br i1 %49, label %53, label %50

50:                                               ; preds = %46
  %51 = getelementptr inbounds %"class.std::ctype", ptr %43, i64 0, i32 9,
      i64 10
  %52 = load i8, ptr %51, align 1, !tbaa !29
  br label %58

53:                                               ; preds = %46
  call void @_ZNKSt5ctypeIcE13_M_widen_initEv(ptr noundef nonnull align 8
      dereferenceable(570) %43)
  %54 = load ptr, ptr %43, align 8, !tbaa !14
  %55 = getelementptr inbounds ptr, ptr %54, i64 6
  %56 = load ptr, ptr %55, align 8
  %57 = call noundef signext i8 %56(ptr noundef nonnull align 8
      dereferenceable(570) %43, i8 noundef signext 10)
  br label %58

58:                                               ; preds = %50, %53
  %59 = phi i8 [ %52, %50 ], [ %57, %53 ]
  %60 = call noundef nonnull align 8 dereferenceable(8) ptr @_ZNSo3putEc(ptr
      noundef nonnull align 8 dereferenceable(8) %37, i8 noundef signext %59)
  %61 = call noundef nonnull align 8 dereferenceable(8) ptr @_ZNSo5flushEv(
```

```
      ptr noundef nonnull align 8 dereferenceable(8) %60)
  call void @llvm.lifetime.end.p0(i64 4, ptr nonnull %1) #5
  ret i32 0
}

; Function Attrs: mustprogress nocallback nofree nosync nounwind willreturn memory(argmem: readwrite)
declare void @llvm.lifetime.start.p0(i64 immarg, ptr nocapture) #1

declare noundef nonnull align 8 dereferenceable(16) ptr @_ZNSirsERi(ptr noundef nonnull align 8 dereferenceable(16), ptr noundef nonnull align 4 dereferenceable(4)) local_unnamed_addr #2

declare noundef nonnull align 8 dereferenceable(8) ptr @_ZNSolsEi(ptr noundef nonnull align 8 dereferenceable(8), i32 noundef) local_unnamed_addr #2

; Function Attrs: mustprogress nocallback nofree nosync nounwind willreturn memory(argmem: readwrite)
declare void @llvm.lifetime.end.p0(i64 immarg, ptr nocapture) #1

declare noundef nonnull align 8 dereferenceable(8) ptr @_ZNSo3putEc(ptr noundef nonnull align 8 dereferenceable(8), i8 noundef signext) local_unnamed_addr #2

declare noundef nonnull align 8 dereferenceable(8) ptr @_ZNSo5flushEv(ptr noundef nonnull align 8 dereferenceable(8)) local_unnamed_addr #2

; Function Attrs: noreturn
declare void @_ZSt16__throw_bad_castv() local_unnamed_addr #3

declare void @_ZNKSt5ctypeIcE13_M_widen_initEv(ptr noundef nonnull align 8 dereferenceable(570)) local_unnamed_addr #2

; Function Attrs: nocallback nofree nosync nounwind speculatable willreturn memory(none)
declare i32 @llvm.vector.reduce.mul.v4i32(<4 x i32>) #4

attributes #0 = { mustprogress norecurse uwtable "min-legal-vector-width"="0" "no-trapping-math"="true" "stack-protector-buffer-size"="8" "target-cpu"="x86-64" "target-features"="+cmov,+cx8,+fxsr,+mmx,+sse,+sse2,+x87" "tune-cpu"="generic" }
attributes #1 = { mustprogress nocallback nofree nosync nounwind willreturn memory(argmem: readwrite) }
attributes #2 = { "no-trapping-math"="true" "stack-protector-buffer-size"="8" "target-cpu"="x86-64" "target-features"="+cmov,+cx8,+fxsr,+mmx,+sse,+sse2,+x87" "tune-cpu"="generic" }
attributes #3 = { noreturn "no-trapping-math"="true" "stack-protector-buffer-size"="8" "target-cpu"="x86-64" "target-features"="+cmov,+cx8,+fxsr,+mmx
```

```
    ,+sse,+sse2,+x87" "tune-cpu"="generic" }
attributes #4 = { nocallback nofree nosync nounwind speculatable willreturn
    memory(none) }
attributes #5 = { nounwind }
attributes #6 = { noreturn }

!llvm.module.flags = !{!0, !1, !2, !3}
!llvm.ident = !{!4}

!0 = !{i32 1, !"wchar_size", i32 4}
!1 = !{i32 8, !"PIC Level", i32 2}
!2 = !{i32 7, !"PIE Level", i32 2}
!3 = !{i32 7, !"uwtable", i32 2}
!4 = !{!"Ubuntu clang version 18.1.3 (1ubuntu1)"}
!5 = !{!6, !6, i64 0}
!6 = !{!"int", !7, i64 0}
!7 = !{!"omnipotent char", !8, i64 0}
!8 = !{!"Simple C++ TBAA"}
!9 = distinct !{!9, !10, !11, !12}
!10 = !{!"llvm.loop.mustprogress"}
!11 = !{!"llvm.loop.isvectorized", i32 1}
!12 = !{!"llvm.loop.unroll.runtime.disable"}
!13 = distinct !{!13, !10, !12, !11}
!14 = !{!15, !15, i64 0}
!15 = !{!"vtable pointer", !8, i64 0}
!16 = !{!17, !22, i64 240}
!17 = !{!"_ZTSSt9basic_iosIcSt11char_traitsIcEE", !18, i64 0, !22, i64 216,
    !7, i64 224, !25, i64 225, !22, i64 232, !22, i64 240, !22, i64 248, !22,
     i64 256}
!18 = !{!"_ZTSSt8ios_base", !19, i64 8, !19, i64 16, !20, i64 24, !21, i64
    28, !21, i64 32, !22, i64 40, !23, i64 48, !7, i64 64, !6, i64 192, !22,
    i64 200, !24, i64 208}
!19 = !{!"long", !7, i64 0}
!20 = !{!"_ZTSSt13_Ios_Fmtflags", !7, i64 0}
!21 = !{!"_ZTSSt12_Ios_Iostate", !7, i64 0}
!22 = !{!"any pointer", !7, i64 0}
!23 = !{!"_ZTSNSt8ios_base6_WordsE", !22, i64 0, !19, i64 8}
!24 = !{!"_ZTSSt6locale", !22, i64 0}
!25 = !{!"bool", !7, i64 0}
!26 = !{!27, !7, i64 56}
!27 = !{!"_ZTSSt5ctypeIcE", !28, i64 0, !22, i64 16, !25, i64 24, !22, i64
    32, !22, i64 40, !22, i64 48, !7, i64 56, !7, i64 57, !7, i64 313, !7,
    i64 569}
!28 = !{!"_ZTSNSt6locale5facetE", !6, i64 8}
!29 = !{!7, !7, i64 0}
```

我们通过 diff 工具发现-O3 与-O2 代码并无区别，但是—O2 与-O1 区别较大，我们试分析其主要区别:

1. 循环优化相关元数据差异

- 旧文件:

```
!11 = !{!"llvm.loop.unroll.disable"}
```

表示禁止循环展开。

- 新文件:

```
!11 = !{!"llvm.loop.isvectorized", i32 1}
!12 = !{!"llvm.loop.unroll.runtime.disable"}
!13 = distinct !{!13, !10, !12, !11}
```

表示循环已向量化，运行时循环展开被禁用，并使用 `distinct` 元数据保证节点唯一性。

2. 类型与 vtable 元数据差异

- 旧文件:

```
!13 = !{!"vtable pointer", !8, i64 0}
!15 = !{!"_ZTSSt9basic_iosIcSt11char_traitsIcEE", ...}
```

- 新文件:

```
!15 = !{!"vtable pointer", !8, i64 0}
!17 = !{!"_ZTSSt9basic_iosIcSt11char_traitsIcEE", ...}
```

- 说明:
  - 描述 C++ 类型布局、vtable 偏移和成员偏移。
  - 新文件调整了节点编号和顺序，但类型信息逻辑不变。

3. 指针与基本类型元数据差异

- 旧文件:

```
!20 = !{!"any pointer", !7, i64 0}
!21 = !{!"_ZTSNSt8ios_base6_WordsE", !20, i64 0, !17, i64 8}
```

- 新文件:

```
!22 = !{!"any pointer", !7, i64 0}
!23 = !{!"_ZTSNSt8ios_base6_WordsE", !22, i64 0, !19, i64 8}
```

- 说明:
  - 元数据表示指针类型和基本类型的偏移信息。
  - 差异仅在节点编号或顺序，逻辑不变。

4. 总结

- 两个 diff 的差异主要体现在:
  1. 循环属性变化，如从 `unroll.disable` 到 `isvectorized`。
  2. metadata 节点编号和顺序调整。
  3. C++ 类型布局和 vtable 信息重新组织。
- 这些差异不会影响程序逻辑，仅用于 LLVM 优化器做:
  - 循环优化决策
  - 类型和别名分析
  - 调试信息生成

对于 opt 指定使用某个 pass 优化.ll 文件，我们发现其对于 pass 选择很重要，我试了几个 pass 都未发现显著的差异，因此并不具体比较。

(六) 代码生成

我们通过 gcc，clang+llc，和 aarch64-linux-gnu-gc 一共生成了三段汇编代码。其中 gcc 生成的汇编代码与 clang+llc 生成的 x86 汇编代码几乎一致，因此我们只展示其中两段代码，gcc 其实就是 clang+llc 的一个封装。

arm 汇编

```
        .arch armv8-a
        .file   "test.cpp"
        .text
#APP
        .globl _ZSt21ios_base_library_initv
#NO_APP
        .align  2
        .global main
        .type   main, %function
main:
.LFB1986:
        .cfi_startproc
        sub     sp, sp, #48
        .cfi_def_cfa_offset 48
        stp     x29, x30, [sp, 32]
        .cfi_offset 29, -16
        .cfi_offset 30, -8
        add     x29, sp, 32
        adrp    x0, :got:__stack_chk_guard
        ldr     x0, [x0, :got_lo12:__stack_chk_guard]
        ldr     x1, [x0]
        str     x1, [sp, 24]
        mov     x1, 0
        add     x0, sp, 12
        mov     x1, x0
```

```
26          adrp    x0, :got:_ZSt3cin
27          ldr     x0, [x0, :got_lo12:_ZSt3cin]
28          bl      _ZNSirsERi
29          mov     w0, 2
30          str     w0, [sp, 16]
31          mov     w0, 1
32          str     w0, [sp, 20]
33          b       .L2
34  .L3:
35          ldr     w1, [sp, 20]
36          ldr     w0, [sp, 16]
37          mul     w0, w1, w0
38          str     w0, [sp, 20]
39          ldr     w0, [sp, 16]
40          add     w0, w0, 1
41          str     w0, [sp, 16]
42  .L2:
43          ldr     w0, [sp, 12]
44          ldr     w1, [sp, 16]
45          cmp     w1, w0
46          ble     .L3
47          ldr     w1, [sp, 20]
48          adrp    x0, :got:_ZSt4cout
49          ldr     x0, [x0, :got_lo12:_ZSt4cout]
50          bl      _ZNSolsEi
51          adrp    x1, :got:
                _ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_
52          ldr     x1, [x1, :got_lo12:
                _ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_]
53          bl      _ZNSolsEPFRSoS_E
54          mov     w0, 0
55          mov     w1, w0
56          adrp    x0, :got:__stack_chk_guard
57          ldr     x0, [x0, :got_lo12:__stack_chk_guard]
58          ldr     x3, [sp, 24]
59          ldr     x2, [x0]
60          subs    x3, x3, x2
61          mov     x2, 0
62          beq     .L5
63          bl      __stack_chk_fail
64  .L5:
65          mov     w0, w1
66          ldp     x29, x30, [sp, 32]
67          add     sp, sp, 48
68          .cfi_restore 29
69          .cfi_restore 30
70          .cfi_def_cfa_offset 0
71          ret
```

```
        .cfi_endproc
.LFE1986:
        .size   main, .-main
        .section        .rodata
        .type   _ZNSt8__detail30__integer_to_chars_is_unsignedIjEE, %object
        .size   _ZNSt8__detail30__integer_to_chars_is_unsignedIjEE, 1
_ZNSt8__detail30__integer_to_chars_is_unsignedIjEE:
        .byte   1
        .type   _ZNSt8__detail30__integer_to_chars_is_unsignedImEE, %object
        .size   _ZNSt8__detail30__integer_to_chars_is_unsignedImEE, 1
_ZNSt8__detail30__integer_to_chars_is_unsignedImEE:
        .byte   1
        .type   _ZNSt8__detail30__integer_to_chars_is_unsignedIyEE, %object
        .size   _ZNSt8__detail30__integer_to_chars_is_unsignedIyEE, 1
_ZNSt8__detail30__integer_to_chars_is_unsignedIyEE:
        .byte   1
        .ident  "GCC: (Ubuntu 13.3.0-6ubuntu2~24.04) 13.3.0"
        .section        .note.GNU-stack,"",@progbits
```

x86 汇编

```
        .file   "test.cpp"
        .text
#APP
        .globl _ZSt21ios_base_library_initv
#NO_APP
        .globl  main
        .type   main, @function
main:
.LFB1988:
        .cfi_startproc
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        subq    $32, %rsp
        movq    %fs:40, %rax
        movq    %rax, -8(%rbp)
        xorl    %eax, %eax
        leaq    -20(%rbp), %rax
        movq    %rax, %rsi
        leaq    _ZSt3cin(%rip), %rax
        movq    %rax, %rdi
        call    _ZNSirsERi@PLT
        movl    $2, -16(%rbp)
        movl    $1, -12(%rbp)
        jmp     .L2
```

```
.L3:
        movl    -12(%rbp), %eax
        imull   -16(%rbp), %eax
        movl    %eax, -12(%rbp)
        addl    $1, -16(%rbp)
.L2:
        movl    -20(%rbp), %eax
        cmpl    %eax, -16(%rbp)
        jle     .L3
        movl    -12(%rbp), %eax
        movl    %eax, %esi
        leaq    _ZSt4cout(%rip), %rax
        movq    %rax, %rdi
        call    _ZNSolsEi@PLT
        movq
            _ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_@GOTPCREL
            (%rip), %rdx
        movq    %rdx, %rsi
        movq    %rax, %rdi
        call    _ZNSolsEPFRSoS_E@PLT
        movl    $0, %eax
        movq    -8(%rbp), %rdx
        subq    %fs:40, %rdx
        je      .L5
        call    __stack_chk_fail@PLT
.L5:
        leave
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE1988:
        .size   main, .-main
        .section        .rodata
        .type   _ZNSt8__detail30__integer_to_chars_is_unsignedIjEE, @object
        .size   _ZNSt8__detail30__integer_to_chars_is_unsignedIjEE, 1
_ZNSt8__detail30__integer_to_chars_is_unsignedIjEE:
        .byte   1
        .type   _ZNSt8__detail30__integer_to_chars_is_unsignedImEE, @object
        .size   _ZNSt8__detail30__integer_to_chars_is_unsignedImEE, 1
_ZNSt8__detail30__integer_to_chars_is_unsignedImEE:
        .byte   1
        .type   _ZNSt8__detail30__integer_to_chars_is_unsignedIyEE, @object
        .size   _ZNSt8__detail30__integer_to_chars_is_unsignedIyEE, 1
_ZNSt8__detail30__integer_to_chars_is_unsignedIyEE:
        .byte   1
        .ident  "GCC: (Ubuntu 13.3.0-6ubuntu2~24.04) 13.3.0"
        .section        .note.GNU-stack,"",@progbits
        .section        .note.gnu.property,"a"
```

```
75          .align 8
76          .long   1f - 0f
77          .long   4f - 1f
78          .long   5
79  0:
80          .string "GNU"
81  1:
82          .align 8
83          .long   0xc0000002
84          .long   3f - 2f
85  2:
86          .long   0x3
87  3:
88          .align 8
89  4:
```

三、 汇编器处理分析与编译流程对比

(一) 汇编器功能与处理结果分析

汇编器（Assembler）是编译工具链中的重要组成部分，其主要功能是将高级语言或中间表示（IR）转换为目标架构的机器码。汇编器的处理结果通常表现为汇编代码文件（.s 文件），其内容包括：

- **指令序列**：对应目标 CPU 指令集的操作码（opcode）和操作数，最终会生成机器码。
- **符号表和重定位信息**：标识全局变量、函数地址等，供链接器使用。
- **伪指令**：如栈帧设置、段声明、调试信息（.file、.section、.cfi 等）。
- **优化信息**（可选）：部分优化后的 IR 对应的汇编指令可能带有循环展开、向量化等标记。

汇编器具体功能分析如下：

1. **指令选择（Instruction Selection）**：将 IR 或中间代码映射为目标架构的指令。
2. **寄存器分配（Register Allocation）**：决定 IR 中虚拟寄存器对应目标寄存器。
3. **指令调度（Instruction Scheduling）**：优化指令顺序以减少流水线冲突。
4. **生成符号和重定位信息**：生成全局变量、函数符号及其偏移，供链接器处理。
5. **伪指令处理**：如栈帧、调试信息、段信息的生成。

汇编器输出的汇编代码是最终生成机器码前的中间结果，对于调试和性能分析非常重要。

(二) GCC 与 Clang+LLC 汇编处理流程对比

1. GCC 编译生成汇编

GCC 直接从 C/C++ 源码生成汇编，前端完成语法分析、类型检查和中间优化，后端将中间表示转换为目标指令。GCC 汇编结果中包含完整的伪指令和调试信息。

工具链	输入	处理阶段	输出
GCC	C/C++ 源代码	前端解析 + 中间优化 + 汇编后端	汇编代码（.s）
Clang + LLC	C/C++ 源代码 → LLVM IR	Clang 前端生成 LLVM IR → LLVM 优化 → LLC 后端生成汇编	汇编代码（.s）
aarch64-linux-gnu-gcc	C/C++ 源代码	与 GCC 类似，但后端针对 ARM64 架构优化	汇编代码（.s，ARM64 指令）

表 1: 不同工具链的汇编生成对比

2. Clang + LLC 编译生成汇编

Clang 将 C/C++ 源码生成 LLVM IR，然后使用 LLVM 的优化 passes 对 IR 进行各种优化（如循环展开、向量化、死代码消除），最后由 `llc` 将优化后的 IR 转换为目标架构汇编。此方法可显式控制优化阶段，适合研究 LLVM IR 对汇编的影响。

3. aarch64-linux-gnu-gcc 编译生成汇编

aarch64-linux-gnu-gcc 是交叉编译工具链，用于生成 ARM64 架构汇编。其流程与 GCC 类似，但指令选择、寄存器分配和调度针对 AArch64 架构进行优化，生成的汇编文件适合 ARM64 处理器直接使用。

(三) 总结

- 汇编器主要负责将中间表示或高层语言映射到目标机器指令，生成符号和伪指令。
- GCC 集成了前端、优化和后端，生成汇编快速方便。
- Clang+LLC 将前端与后端解耦，可对 IR 做详细优化分析。
- aarch64-linux-gnu-gcc 是针对 ARM64 的 GCC 变体，适合交叉编译 ARM 平台程序。

四、 链接器

链接器（Linker）是编译工具链中的最后一个阶段，负责将多个目标文件（.o 文件）和库文件链接成最终的可执行文件或共享库。链接器的主要功能包括：

- **符号解析（Symbol Resolution）**：链接器会解析目标文件中的符号引用，确定每个符号（如函数名、变量名）对应的内存地址。如果一个符号在多个目标文件中定义，链接器会根据链接规则选择一个定义。
- **地址重定位（Relocation）**：链接器会调整目标文件中的地址引用，使其指向正确的内存位置。由于每个目标文件可能被加载到不同的内存地址，链接器需要更新所有相关的地址引用。

- **合并节（Section Merging）**：链接器会将多个目标文件中的相同类型的节（如代码节、数据节）合并成一个节，以便生成最终的可执行文件。
- **生成可执行文件格式**：链接器会根据目标平台生成符合该平台要求的可执行文件格式，如 ELF（Linux）、PE（Windows）等。
- **处理库文件**：链接器可以链接静态库（.a 文件）和动态库（.so 文件），将库中的符号和代码包含到最终的可执行文件中。

我们通过执行代码 g++ test_x86.o -o test_x86 得到了最终的可执行文件。我们通过反汇编的命令 objdump -d test_x86 和 objdump -d test_x86.o 分别得到了没有链接的时候和链接结束后的汇编代码。代码比较如下，可见链接前，只有程序的本地代码，没有任何外部符号或库的引用。主要是函数和控制流的实现。程序中的所有操作都是静态的，指向常量或固定内存位置。链接后，包含了动态链接信息（如.plt 和.plt.got）。调用标准库函数和外部符号会通过全局偏移表（GOT）进行解析。在程序执行期间，动态链接器会解析这些符号并将它们与实际库中的符号链接起来。

链接前后汇编代码对比

```
<链接前>
0000000000000000 <main>:
   0:   f3 0f 1e fa             endbr64
   4:   55                      push   %rbp
   5:   48 89 e5                mov    %rsp,%rbp
   8:   48 83 ec 20             sub    $0x20,%rsp
   c:   64 48 8b 04 25 28 00    mov    %fs:0x28,%rax
  13:   00 00
  15:   48 89 45 f8             mov    %rax,-0x8(%rbp)
  19:   31 c0                   xor    %eax,%eax
  1b:   48 8d 45 ec             lea    -0x14(%rbp),%rax
  1f:   48 89 c6                mov    %rax,%rsi
  22:   48 8d 05 00 00 00 00    lea    0x0(%rip),%rax        # 29 <main+0x29>
  29:   48 89 c7                mov    %rax,%rdi
  2c:   e8 00 00 00 00          call   31 <main+0x31>
  31:   c7 45 f0 02 00 00 00    movl   $0x2,-0x10(%rbp)
  38:   c7 45 f4 01 00 00 00    movl   $0x1,-0xc(%rbp)
  3f:   eb 0e                   jmp    4f <main+0x4f>
  41:   8b 45 f4                mov    -0xc(%rbp),%eax
  44:   0f af 45 f0             imul   -0x10(%rbp),%eax
  48:   89 45 f4                mov    %eax,-0xc(%rbp)
  4b:   83 45 f0 01             addl   $0x1,-0x10(%rbp)
  4f:   8b 45 ec                mov    -0x14(%rbp),%eax
  52:   39 45 f0                cmp    %eax,-0x10(%rbp)
  55:   7e ea                   jle    41 <main+0x41>
  57:   8b 45 f4                mov    -0xc(%rbp),%eax
  5a:   89 c6                   mov    %eax,%esi
  5c:   48 8d 05 00 00 00 00    lea    0x0(%rip),%rax        # 63 <main+0x63>
  63:   48 89 c7                mov    %rax,%rdi
  66:   e8 00 00 00 00          call   6b <main+0x6b>
  6b:   48 8b 15 00 00 00 00    mov    0x0(%rip),%rdx        # 72 <main+0x72>
  72:   48 89 d6                mov    %rdx,%rsi
```

```
  75:    48 89 c7                 mov    %rax,%rdi
  78:    e8 00 00 00 00           call   7d <main+0x7d>
  7d:    b8 00 00 00 00           mov    $0x0,%eax
  82:    48 8b 55 f8              mov    -0x8(%rbp),%rdx
  86:    64 48 2b 14 25 28 00     sub    %fs:0x28,%rdx
  8d:    00 00
  8f:    74 05                    je     96 <main+0x96>
  91:    e8 00 00 00 00           call   96 <main+0x96>
  96:    c9                       leave
  97:    c3                       ret
  <链接后>
0000000000001000 <_init>:
    1000:       f3 0f 1e fa             endbr64
    1004:       48 83 ec 08             sub    $0x8,%rsp
    1008:       48 8b 05 e1 2f 00 00    mov    0x2fe1(%rip),%rax        # 3
        ff0 <__gmon_start__@Base>
    100f:       48 85 c0                test   %rax,%rax
    1012:       74 02                   je     1016 <_init+0x16>
    1014:       ff d0                   call   *%rax
    1016:       48 83 c4 08             add    $0x8,%rsp
    101a:       c3                      ret

Disassembly of section .plt:

0000000000001020 <.plt>:
    1020:       ff 35 7a 2f 00 00       push   0x2f7a(%rip)        # 3fa0 <
        _GLOBAL_OFFSET_TABLE_+0x8>
    1026:       ff 25 7c 2f 00 00       jmp    *0x2f7c(%rip)        # 3fa8 <
        _GLOBAL_OFFSET_TABLE_+0x10>
    102c:       0f 1f 40 00             nopl   0x0(%rax)
    1030:       f3 0f 1e fa             endbr64
    1034:       68 00 00 00 00          push   $0x0
    1039:       e9 e2 ff ff ff          jmp    1020 <_init+0x20>
    103e:       66 90                   xchg   %ax,%ax
    1040:       f3 0f 1e fa             endbr64
    1044:       68 01 00 00 00          push   $0x1
    1049:       e9 d2 ff ff ff          jmp    1020 <_init+0x20>
    104e:       66 90                   xchg   %ax,%ax
    1050:       f3 0f 1e fa             endbr64
    1054:       68 02 00 00 00          push   $0x2
    1059:       e9 c2 ff ff ff          jmp    1020 <_init+0x20>
    105e:       66 90                   xchg   %ax,%ax
    1060:       f3 0f 1e fa             endbr64
    1064:       68 03 00 00 00          push   $0x3
    1069:       e9 b2 ff ff ff          jmp    1020 <_init+0x20>
    106e:       66 90                   xchg   %ax,%ax

Disassembly of section .plt.got:
```

```
0000000000001070 <__cxa_finalize@plt>:
    1070:       f3 0f 1e fa             endbr64
    1074:       ff 25 56 2f 00 00       jmp    *0x2f56(%rip)        # 3fd0 <
        __cxa_finalize@GLIBC_2.2.5>
    107a:       66 0f 1f 44 00 00       nopw   0x0(%rax,%rax,1)

Disassembly of section .plt.sec:

0000000000001080 <_ZNSirsERi@plt>:
    1080:       f3 0f 1e fa             endbr64
    1084:       ff 25 26 2f 00 00       jmp    *0x2f26(%rip)        # 3fb0 <
        _ZNSirsERi@GLIBCXX_3.4>
    108a:       66 0f 1f 44 00 00       nopw   0x0(%rax,%rax,1)

0000000000001090 <_ZNSolsEPFRSoS_E@plt>:
    1090:       f3 0f 1e fa             endbr64
    1094:       ff 25 1e 2f 00 00       jmp    *0x2f1e(%rip)        # 3fb8 <
        _ZNSolsEPFRSoS_E@GLIBCXX_3.4>
    109a:       66 0f 1f 44 00 00       nopw   0x0(%rax,%rax,1)

00000000000010a0 <__stack_chk_fail@plt>:
    10a0:       f3 0f 1e fa             endbr64
    10a4:       ff 25 16 2f 00 00       jmp    *0x2f16(%rip)        # 3fc0 <
        __stack_chk_fail@GLIBC_2.4>
    10aa:       66 0f 1f 44 00 00       nopw   0x0(%rax,%rax,1)

00000000000010b0 <_ZNSolsEi@plt>:
    10b0:       f3 0f 1e fa             endbr64
    10b4:       ff 25 0e 2f 00 00       jmp    *0x2f0e(%rip)        # 3fc8 <
        _ZNSolsEi@GLIBCXX_3.4>
    10ba:       66 0f 1f 44 00 00       nopw   0x0(%rax,%rax,1)

Disassembly of section .text:

00000000000010c0 <_start>:
    10c0:       f3 0f 1e fa             endbr64
    10c4:       31 ed                   xor    %ebp,%ebp
    10c6:       49 89 d1                mov    %rdx,%r9
    10c9:       5e                      pop    %rsi
    10ca:       48 89 e2                mov    %rsp,%rdx
    10cd:       48 83 e4 f0             and    $0xfffffffffffffff0,%rsp
    10d1:       50                      push   %rax
    10d2:       54                      push   %rsp
    10d3:       45 31 c0                xor    %r8d,%r8d
    10d6:       31 c9                   xor    %ecx,%ecx
    10d8:       48 8d 3d ca 00 00 00    lea    0xca(%rip),%rdi        # 11a9
        <main>
```

```
120     10df:       ff 15 fb 2e 00 00       call   *0x2efb(%rip)        # 3fe0 <
            __libc_start_main@GLIBC_2.34>
121     10e5:       f4                      hlt
122     10e6:       66 2e 0f 1f 84 00 00    cs nopw 0x0(%rax,%rax,1)
123     10ed:       00 00 00
124
125 00000000000010f0 <deregister_tm_clones>:
126     10f0:       48 8d 3d 19 2f 00 00    lea    0x2f19(%rip),%rdi        #
            4010 <__TMC_END__>
127     10f7:       48 8d 05 12 2f 00 00    lea    0x2f12(%rip),%rax        #
            4010 <__TMC_END__>
128     10fe:       48 39 f8                cmp    %rdi,%rax
129     1101:       74 15                   je     1118 <deregister_tm_clones+0
            x28>
130     1103:       48 8b 05 de 2e 00 00    mov    0x2ede(%rip),%rax        # 3
            fe8 <_ITM_deregisterTMCloneTable@Base>
131     110a:       48 85 c0                test   %rax,%rax
132     110d:       74 09                   je     1118 <deregister_tm_clones+0
            x28>
133     110f:       ff e0                   jmp    *%rax
134     1111:       0f 1f 80 00 00 00 00    nopl   0x0(%rax)
135     1118:       c3                      ret
136     1119:       0f 1f 80 00 00 00 00    nopl   0x0(%rax)
137
138 0000000000001120 <register_tm_clones>:
139     1120:       48 8d 3d e9 2e 00 00    lea    0x2ee9(%rip),%rdi        #
            4010 <__TMC_END__>
140     1127:       48 8d 35 e2 2e 00 00    lea    0x2ee2(%rip),%rsi        #
            4010 <__TMC_END__>
141     112e:       48 29 fe                sub    %rdi,%rsi
142     1131:       48 89 f0                mov    %rsi,%rax
143     1134:       48 c1 ee 3f             shr    $0x3f,%rsi
144     1138:       48 c1 f8 03             sar    $0x3,%rax
145     113c:       48 01 c6                add    %rax,%rsi
146     113f:       48 d1 fe                sar    $1,%rsi
147     1142:       74 14                   je     1158 <register_tm_clones+0x38>
148     1144:       48 8b 05 ad 2e 00 00    mov    0x2ead(%rip),%rax        # 3
            ff8 <_ITM_registerTMCloneTable@Base>
149     114b:       48 85 c0                test   %rax,%rax
150     114e:       74 08                   je     1158 <register_tm_clones+0x38>
151     1150:       ff e0                   jmp    *%rax
152     1152:       66 0f 1f 44 00 00       nopw   0x0(%rax,%rax,1)
153     1158:       c3                      ret
154     1159:       0f 1f 80 00 00 00 00    nopl   0x0(%rax)
155
156 0000000000001160 <__do_global_dtors_aux>:
157     1160:       f3 0f 1e fa             endbr64
158     1164:       80 3d 0d 31 00 00 00    cmpb   $0x0,0x310d(%rip)        #
```

```
        4278 <completed.0>
159     116b:       75 2b                   jne    1198 <__do_global_dtors_aux+0
        x38>
160     116d:       55                      push   %rbp
161     116e:       48 83 3d 5a 2e 00 00    cmpq   $0x0,0x2e5a(%rip)        # 3
        fd0 <__cxa_finalize@GLIBC_2.2.5>
162     1175:       00
163     1176:       48 89 e5                mov    %rsp,%rbp
164     1179:       74 0c                   je     1187 <__do_global_dtors_aux+0
        x27>
165     117b:       48 8b 3d 86 2e 00 00    mov    0x2e86(%rip),%rdi        #
        4008 <__dso_handle>
166     1182:       e8 e9 fe ff ff          call   1070 <__cxa_finalize@plt>
167     1187:       e8 64 ff ff ff          call   10f0 <deregister_tm_clones>
168     118c:       c6 05 e5 30 00 00 01    movb   $0x1,0x30e5(%rip)        #
        4278 <completed.0>
169     1193:       5d                      pop    %rbp
170     1194:       c3                      ret
171     1195:       0f 1f 00                nopl   (%rax)
172     1198:       c3                      ret
173     1199:       0f 1f 80 00 00 00 00    nopl   0x0(%rax)
174
175 00000000000011a0 <frame_dummy>:
176     11a0:       f3 0f 1e fa             endbr64
177     11a4:       e9 77 ff ff ff          jmp    1120 <register_tm_clones>
178
179 00000000000011a9 <main>:
180     11a9:       f3 0f 1e fa             endbr64
181     11ad:       55                      push   %rbp
182     11ae:       48 89 e5                mov    %rsp,%rbp
183     11b1:       48 83 ec 20             sub    $0x20,%rsp
184     11b5:       64 48 8b 04 25 28 00    mov    %fs:0x28,%rax
185     11bc:       00 00
186     11be:       48 89 45 f8             mov    %rax,-0x8(%rbp)
187     11c2:       31 c0                   xor    %eax,%eax
188     11c4:       48 8d 45 ec             lea    -0x14(%rbp),%rax
189     11c8:       48 89 c6                mov    %rax,%rsi
190     11cb:       48 8d 05 8e 2f 00 00    lea    0x2f8e(%rip),%rax        #
        4160 <_ZSt3cin@GLIBCXX_3.4>
191     11d2:       48 89 c7                mov    %rax,%rdi
192     11d5:       e8 a6 fe ff ff          call   1080 <_ZNSirsERi@plt>
193     11da:       c7 45 f0 02 00 00 00    movl   $0x2,-0x10(%rbp)
194     11e1:       c7 45 f4 01 00 00 00    movl   $0x1,-0xc(%rbp)
195     11e8:       eb 0e                   jmp    11f8 <main+0x4f>
196     11ea:       8b 45 f4                mov    -0xc(%rbp),%eax
197     11ed:       0f af 45 f0             imul   -0x10(%rbp),%eax
198     11f1:       89 45 f4                mov    %eax,-0xc(%rbp)
199     11f4:       83 45 f0 01             addl   $0x1,-0x10(%rbp)
```

```
    11f8:       8b 45 ec                mov    -0x14(%rbp),%eax
    11fb:       39 45 f0                cmp    %eax,-0x10(%rbp)
    11fe:       7e ea                   jle    11ea <main+0x41>
    1200:       8b 45 f4                mov    -0xc(%rbp),%eax
    1203:       89 c6                   mov    %eax,%esi
    1205:       48 8d 05 34 2e 00 00    lea    0x2e34(%rip),%rax        #
        4040 <_ZSt4cout@GLIBCXX_3.4>
    120c:       48 89 c7                mov    %rax,%rdi
    120f:       e8 9c fe ff ff          call   10b0 <_ZNSolsEi@plt>
    1214:       48 8b 15 bd 2d 00 00    mov    0x2dbd(%rip),%rdx        # 3
        fd8 <
        _ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_@GLIBCXX_3
        .4>
    121b:       48 89 d6                mov    %rdx,%rsi
    121e:       48 89 c7                mov    %rax,%rdi
    1221:       e8 6a fe ff ff          call   1090 <_ZNSolsEPFRSoS_E@plt>
    1226:       b8 00 00 00 00          mov    $0x0,%eax
    122b:       48 8b 55 f8             mov    -0x8(%rbp),%rdx
    122f:       64 48 2b 14 25 28 00    sub    %fs:0x28,%rdx
    1236:       00 00
    1238:       74 05                   je     123f <main+0x96>
    123a:       e8 61 fe ff ff          call   10a0 <__stack_chk_fail@plt>
    123f:       c9                      leave
    1240:       c3                      ret

Disassembly of section .fini:

0000000000001244 <_fini>:
    1244:       f3 0f 1e fa             endbr64
    1248:       48 83 ec 08             sub    $0x8,%rsp
    124c:       48 83 c4 08             add    $0x8,%rsp
    1250:       c3                      ret
```