YatCC-AI
编译原理课程实验

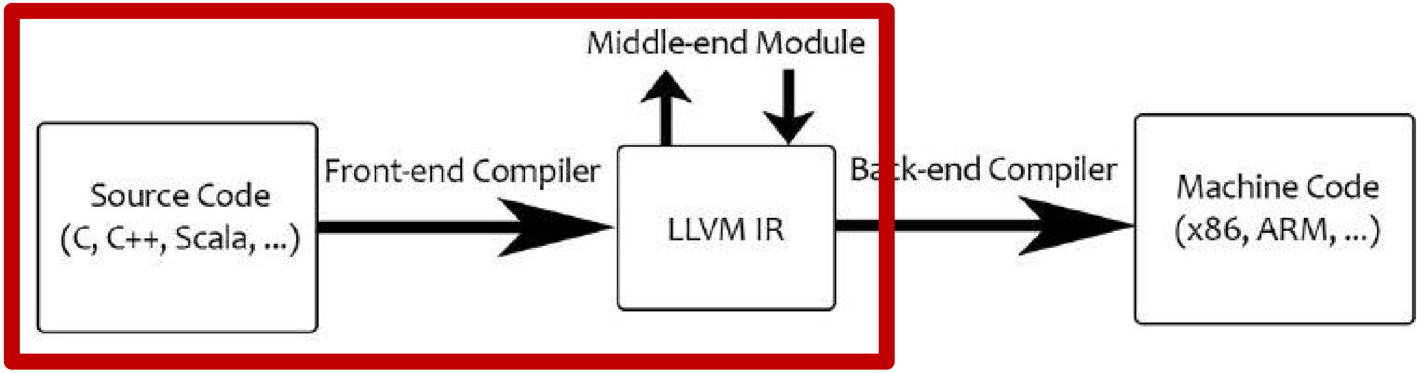
中山大学
助教：顾宇浩
2025/2/28
YatCC 团队敬上

课程简介

❖YatCC实验是基于*LLVM*的编译器构造实验

在本课程中，你需要：

- 使用LLVM工业级编译器基础设施开发一个C语言子集的编译器
- 完成从C语言源代码到LLVM中间表示（甚至是汇编代码）的转换过程
- 预估编码总量：3000LOC左右



课程特色：专业的授课团队

吴露

何静仪

顾宇浩

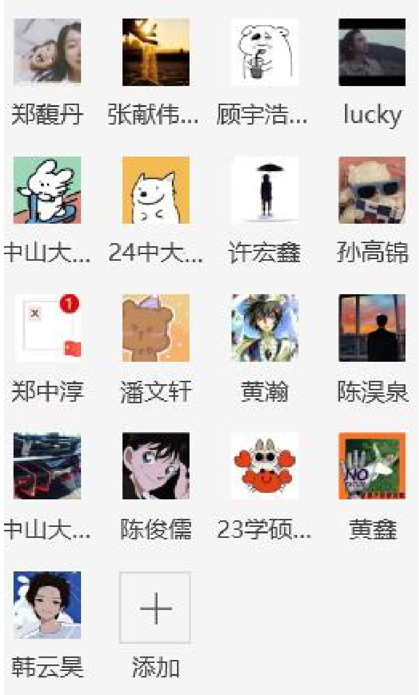
许宏鑫

黄瀚

潘文轩

郑腾扬

孙高锦


郑馥丹 张献伟... 顾宇浩... lucky
中山大... 24中大... 许宏鑫 孙高锦
郑中淳 潘文轩 黄瀚 陈淏泉
中山大... 陈俊儒 23学硕... 黄鑫
韩云昊 添加

课程特色：一站式集成实验平台

- https://yatcc-ai.com/ol.html
  - VS Code在线开发平台，基于我们自己的超算
  - “配环境”的终极解决方案就是：不用配环境

星光 STARLIGHT

HPC+AI多模式融合应用软件平台

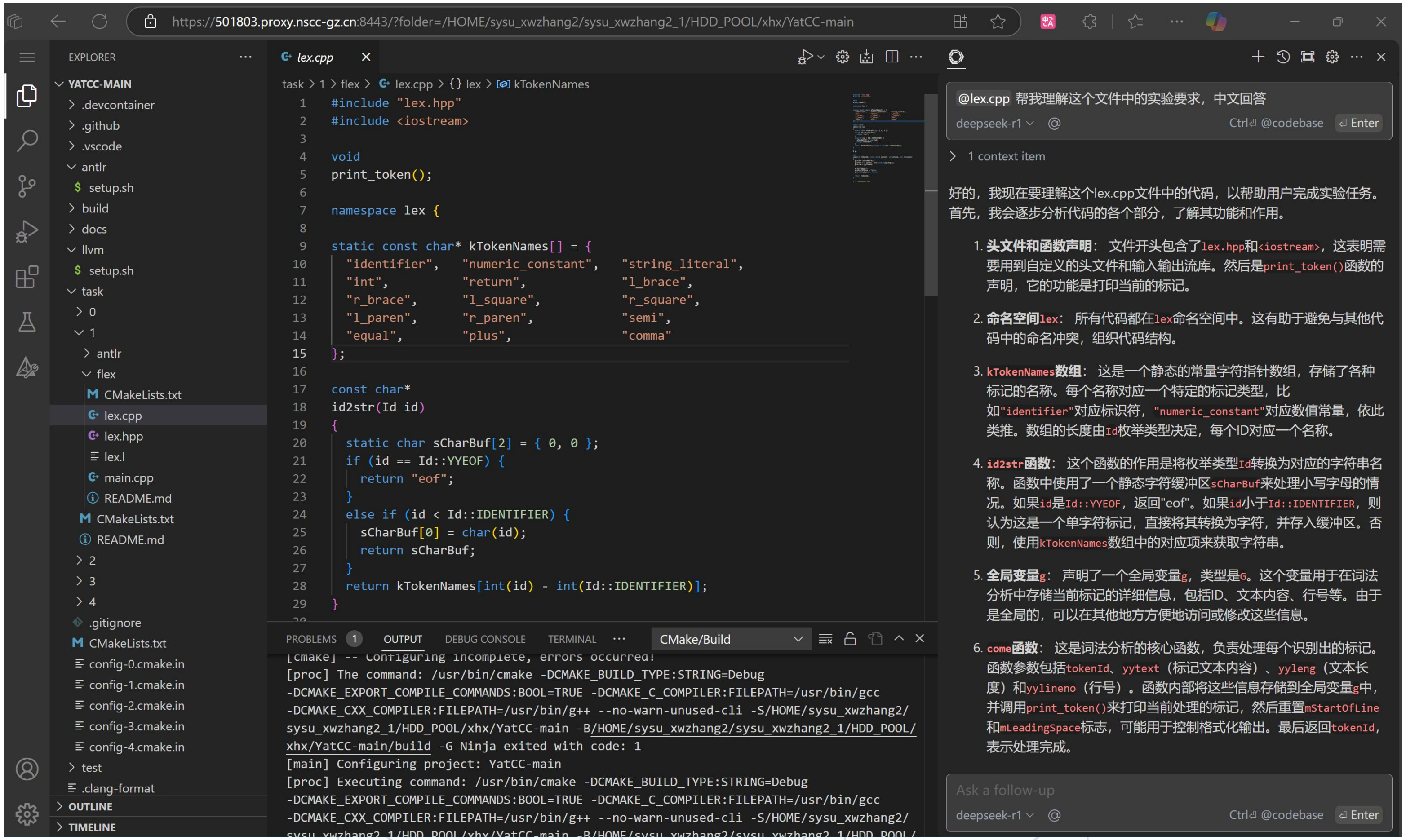
课程特色：多种打开方式

- 点击即用的一站式平台
  - https://yatcc-ai.com/ol.html

- 开箱即用的容器镜像
  - docker pull registry.cn-guangzhou.aliyuncs.com/yhgu2000/yatcc:latest

- 配置自动化的代码仓库
  - git clone https://github.com/arcsysu/YatCC.git

课程特色：DeepSeek AI大模型加持



```cpp
#include "lex.hpp"
#include <iostream>

void
print_token();

namespace lex {

static const char* kTokenNames[] = {
  "identifier",   "numeric_constant",   "string_literal",
  "int",          "return",             "l_brace",
  "r_brace",      "l_square",           "r_square",
  "l_paren",      "r_paren",            "semi",
  "equal",        "plus",               "comma"
};

const char*
id2str(Id id)
{
  static char sCharBuf[2] = { 0, 0 };
  if (id == Id::YYEOF) {
    return "eof";
  }
  else if (id < Id::IDENTIFIER) {
    sCharBuf[0] = char(id);
    return sCharBuf;
  }
  return kTokenNames[int(id) - int(Id::IDENTIFIER)];
}
```

```
[cmake] -- Configuring incomplete, errors occurred!
[proc] The command: /usr/bin/cmake -DCMAKE_BUILD_TYPE:STRING=Debug
-DCMAKE_EXPORT_COMPILE_COMMANDS:BOOL=TRUE -DCMAKE_C_COMPILER:FILEPATH=/usr/bin/gcc
-DCMAKE_CXX_COMPILER:FILEPATH=/usr/bin/g++ --no-warn-unused-cli -S/HOME/sysu_xwzhang2/
sysu_xwzhang2_1/HDD_POOL/xhx/YatCC-main -B/HOME/sysu_xwzhang2/sysu_xwzhang2_1/HDD_POOL/
xhx/YatCC-main/build -G Ninja exited with code: 1
[main] Configuring project: YatCC-main
[proc] Executing command: /usr/bin/cmake -DCMAKE_BUILD_TYPE:STRING=Debug
-DCMAKE_EXPORT_COMPILE_COMMANDS:BOOL=TRUE -DCMAKE_C_COMPILER:FILEPATH=/usr/bin/gcc
-DCMAKE_CXX_COMPILER:FILEPATH=/usr/bin/g++ --no-warn-unused-cli -S/HOME/sysu_xwzhang2/
```

@lex.cpp 帮我理解这个文件中的实验要求，中文回答

好的，我现在要理解这个lex.cpp文件中的代码，以帮助用户完成实验任务。首先，我会逐步分析代码的各个部分，了解其功能和作用。

1. **头文件和函数声明**： 文件开头包含了`lex.hpp`和`<iostream>`，这表明需要用到自定义的头文件和输入输出流库。然后是`print_token()`函数的声明，它的功能是打印当前的标记。
2. **命名空间`lex`**： 所有代码都在`lex`命名空间中。这有助于避免与其他代码中的命名冲突，组织代码结构。
3. **`kTokenNames`数组**： 这是一个静态的常量字符指针数组，存储了各种标记的名称。每个名称对应一个特定的标记类型，比如`"identifier"`对应标识符，`"numeric_constant"`对应数值常量，依此类推。数组的长度由`Id`枚举类型决定，每个ID对应一个名称。
4. **`id2str`函数**： 这个函数的作用是将枚举类型`Id`转换为对应的字符串名称。函数中使用了一个静态字符缓冲区`sCharBuf`来处理小写字母的情况。如果`id`是`Id::YYEOF`，返回"eof"。如果`id`小于`Id::IDENTIFIER`，则认为这是一个单字符标记，直接将其转换为字符，并存入缓冲区。否则，使用`kTokenNames`数组中的对应项来获取字符串。
5. **全局变量`g`**： 声明了一个全局变量`g`，类型是`G`。这个变量用于在词法分析中存储当前标记的详细信息，包括ID、文本内容、行号等。由于是全局的，可以在其他地方方便地访问或修改这些信息。
6. **`come`函数**： 这是词法分析的核心函数，负责处理每个识别出的标记。函数参数包括`tokenId`、`yytext`（标记文本内容）、`yyleng`（文本长度）和`yylineno`（行号）。函数内部将这些信息存储到全局变量`g`中，并调用`print_token()`来打印当前处理的标记，然后重置`mStartOfLine`和`mLeadingSpace`标志，可能用于控制格式化输出。最后返回`tokenId`，表示处理完成。

*由于大模型需求火爆，中心资源紧张，
AI服务会在课程过程中逐渐普及上线……

课程特色：丰富的学习资源

- Bilibili 教程
  - https://space.bilibili.com/3546650047941291
- Github 文档
  - https://arcsysu.github.io/YatCC
- LLVM 官方文档
  - https://llvm.org/docs/

课程特色：学术竞赛&产业实践

A3 Foresight Program
The 2024 Workshop of
Data Sharing Infrastructure across
Northeast-Asia Supercomputing
YAT-CC Team

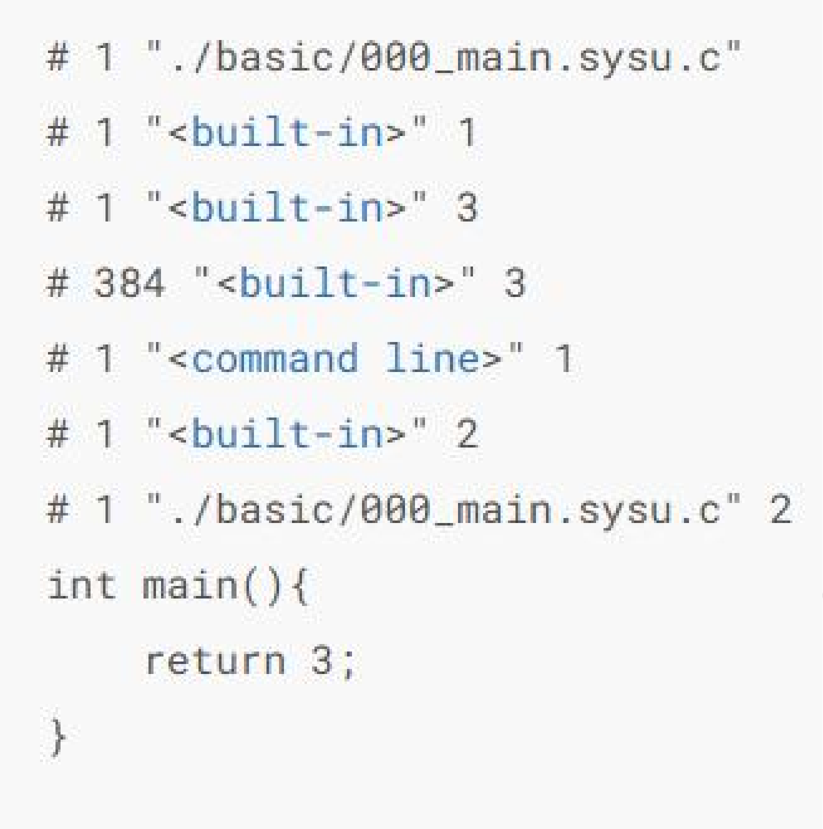
2023全国大学生计算机系统能力大赛总决赛颁奖典礼
编译系统设计赛（华为毕昇杯）
一等奖
ARM
RISC-V
CSCC全国大学生计算机系统能力大赛

编译优化奖
SYSU-LANG
YAT COMPILER
实验四排行榜上 90 分以上的同学：
李卓群(skspass)、冯一鸣、廖之语、梁爽
(apshuang)、林志扬(lzy-learning)、李博、吴
健强

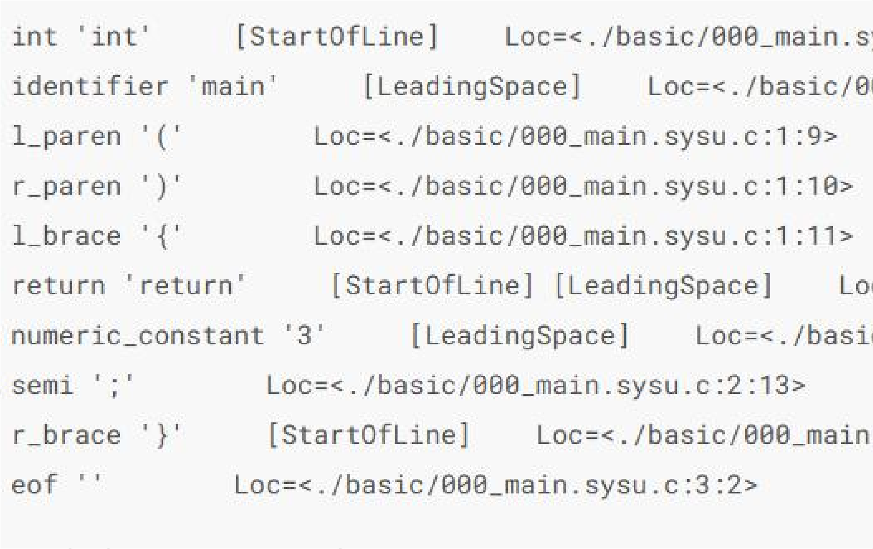
任务与考核

task1 （? 分）

- 使用flex/antlr读取源代码，解析成TOKEN流
- 评分标准：识别足够的TOKEN并识别位置

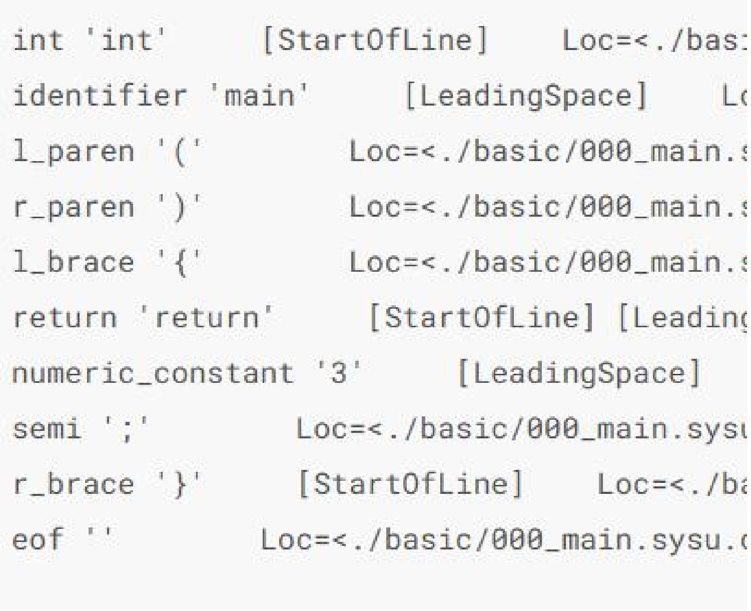


```
# 1 "./basic/000_main.sysu.c"
# 1 "<built-in>" 1
# 1 "<built-in>" 3
# 384 "<built-in>" 3
# 1 "<command line>" 1
# 1 "<built-in>" 2
# 1 "./basic/000_main.sysu.c" 2
int main(){
    return 3;
}
```

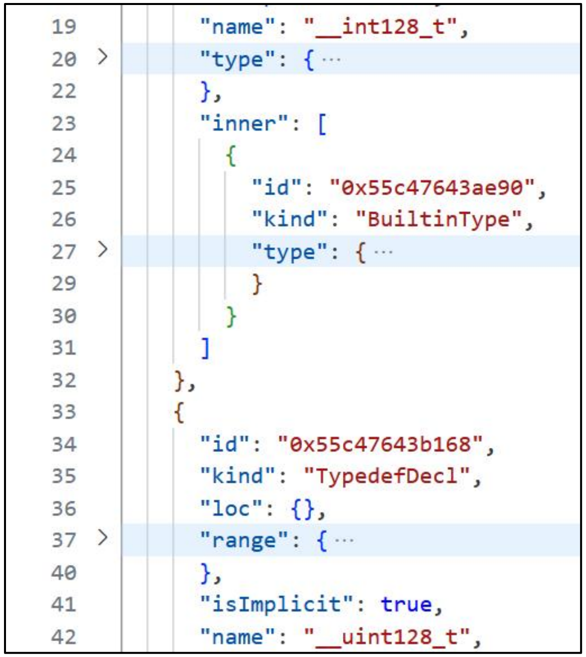


```
int 'int'        [StartOfLine]    Loc=<./basic/000_main.s
identifier 'main'       [LeadingSpace]     Loc=<./basic/0
l_paren '('         Loc=<./basic/000_main.sysu.c:1:9>
r_paren ')'         Loc=<./basic/000_main.sysu.c:1:10>
l_brace '{'         Loc=<./basic/000_main.sysu.c:1:11>
return 'return'      [StartOfLine] [LeadingSpace]     Lo
numeric_constant '3'      [LeadingSpace]     Loc=<./basi
semi ';'        Loc=<./basic/000_main.sysu.c:2:13>
r_brace '}'     [StartOfLine]    Loc=<./basic/000_main
eof ''          Loc=<./basic/000_main.sysu.c:3:2>
```

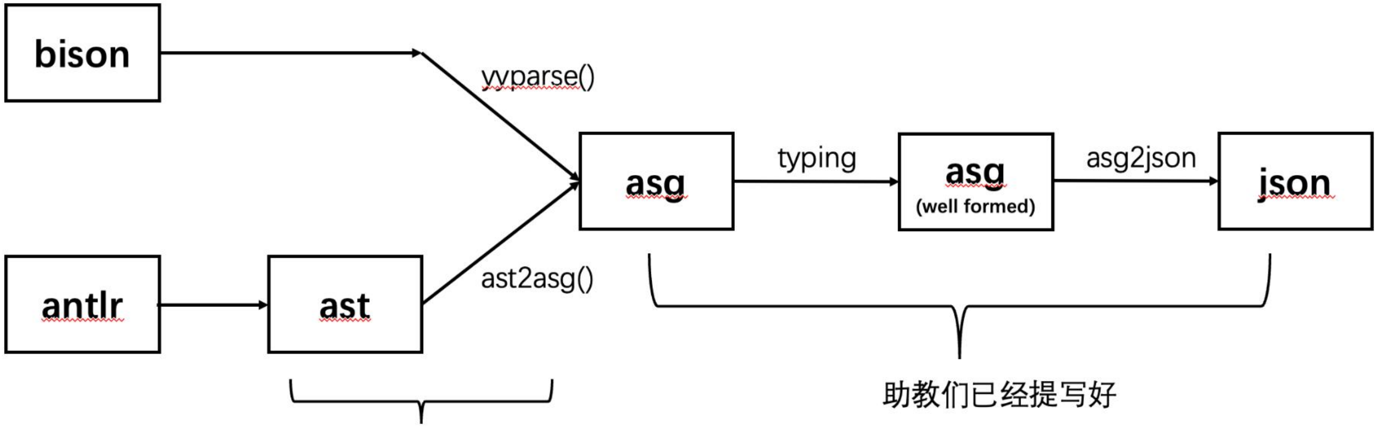
task2 （? 分）



```
int 'int'        [StartOfLine]     Loc=<./bas
identifier 'main'       [LeadingSpace]     Lo
l_paren '('         Loc=<./basic/000_main.s
r_paren ')'         Loc=<./basic/000_main.s
l_brace '{'         Loc=<./basic/000_main.s
return 'return'       [StartOfLine] [Leadin
numeric_constant '3'      [LeadingSpace]
semi ';'          Loc=<./basic/000_main.sysu
r_brace '}'       [StartOfLine]      Loc=<./ba
eof ''          Loc=<./basic/000_main.sysu.c
```



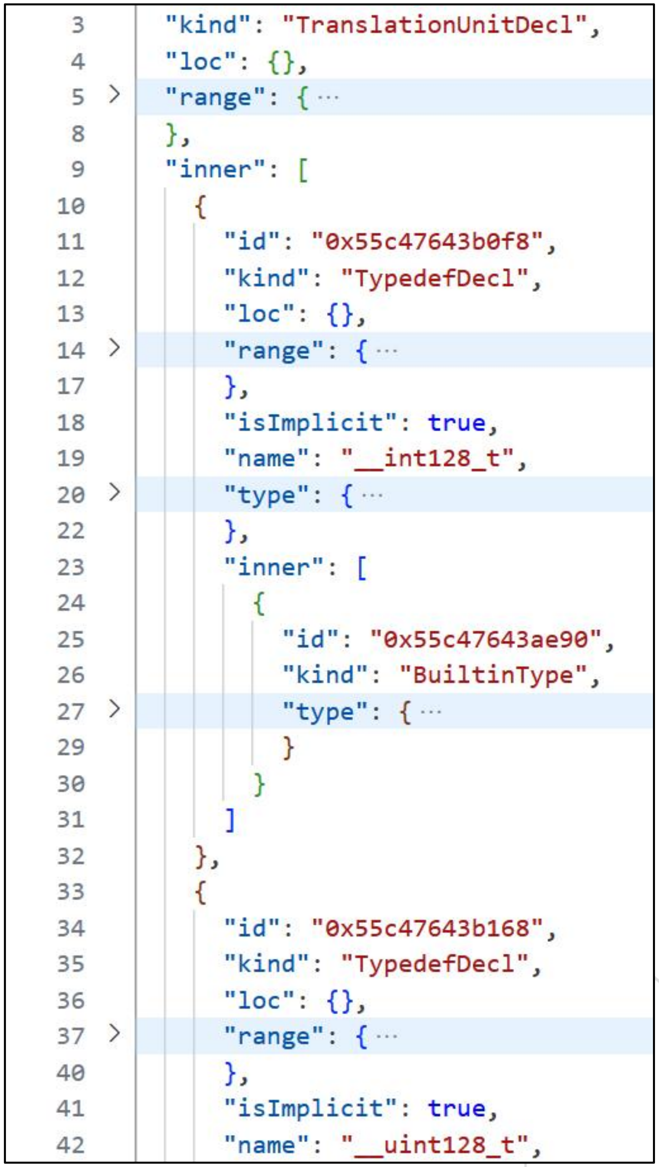
```
      "name": "__int128_t",
      "type": {…
      },
      "inner": [
        {
          "id": "0x55c47643ae90",
          "kind": "BuiltinType",
          "type": {…
          }
        }
      ]
    },
    {
      "id": "0x55c47643b168",
      "kind": "TypedefDecl",
      "loc": {},
      "range": {…
      },
      "isImplicit": true,
      "name": "__uint128_t",
```



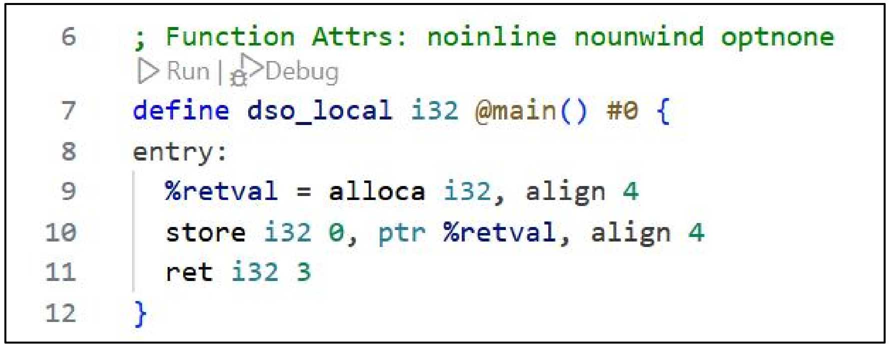
助教们已经提写好

需要同学们填空

task3 （？ 分）

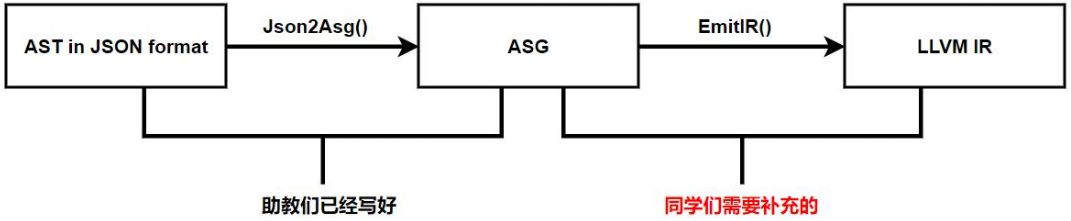


```
  "kind": "TranslationUnitDecl",
  "loc": {},
  "range": { ⋯
  },
  "inner": [
    {
      "id": "0x55c47643b0f8",
      "kind": "TypedefDecl",
      "loc": {},
      "range": { ⋯
      },
      "isImplicit": true,
      "name": "__int128_t",
      "type": { ⋯
      },
      "inner": [
        {
          "id": "0x55c47643ae90",
          "kind": "BuiltinType",
          "type": { ⋯
          }
        }
      ]
    },
    {
      "id": "0x55c47643b168",
      "kind": "TypedefDecl",
      "loc": {},
      "range": { ⋯
      },
      "isImplicit": true,
      "name": "__uint128_t",
```



```
; Function Attrs: noinline nounwind optnone
define dso_local i32 @main() #0 {
entry:
  %retval = alloca i32, align 4
  store i32 0, ptr %retval, align 4
  ret i32 3
}
```

- 使用LLVM库，将JSON语法树转换为LLVM IR
- 解析ASG的代码已由助教完成，节省工作量！
- 评分标准：生成的LLVM IR能正确编译运行



AST in JSON format

Json2Asg()

ASG

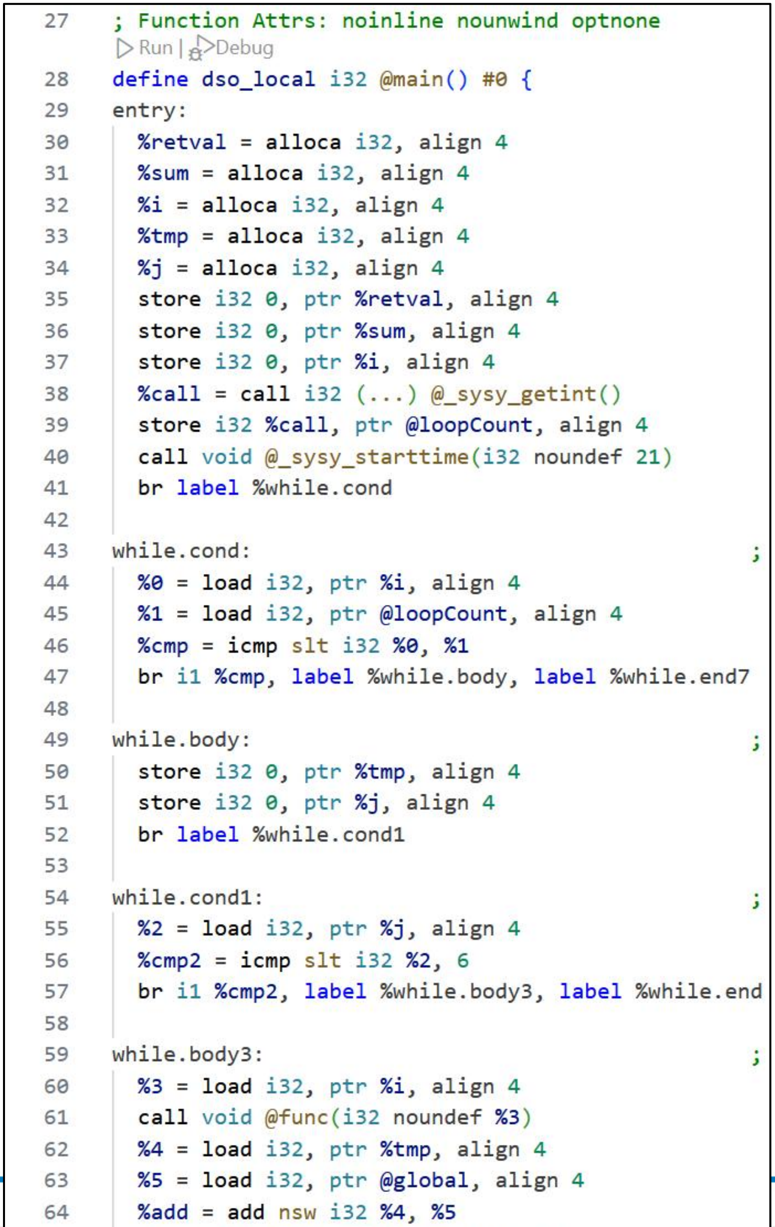
EmitIR()

LLVM IR

助教们已经写好

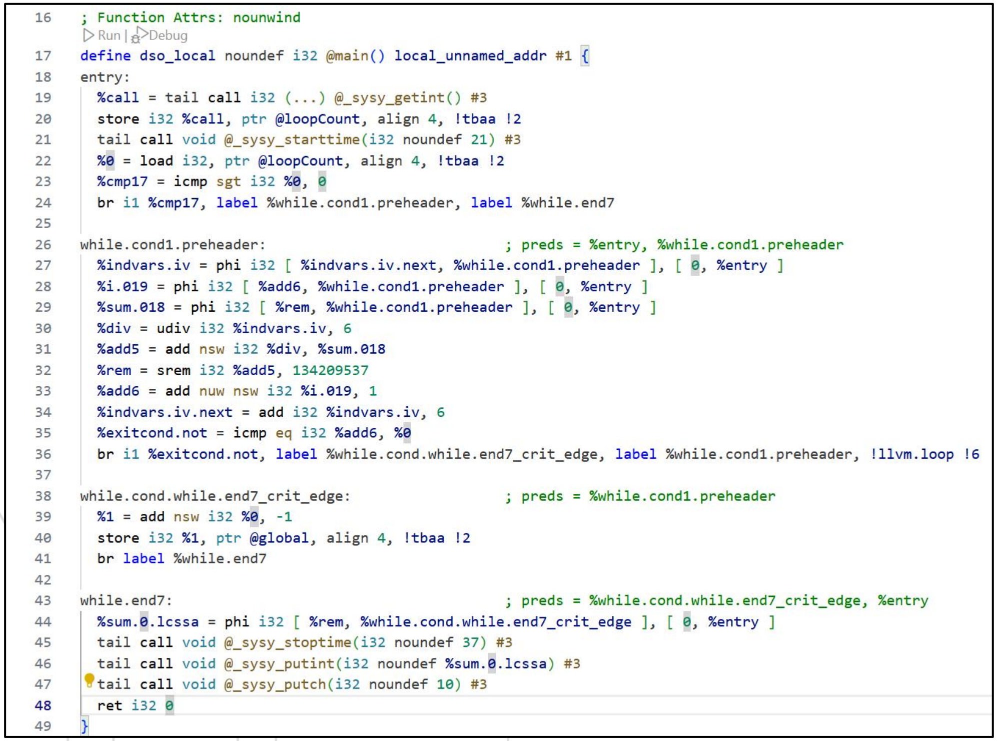
同学们需要补充的

task4 （？ 分）



```
; Function Attrs: noinline nounwind optnone
define dso_local i32 @main() #0 {
entry:
  %retval = alloca i32, align 4
  %sum = alloca i32, align 4
  %i = alloca i32, align 4
  %tmp = alloca i32, align 4
  %j = alloca i32, align 4
  store i32 0, ptr %retval, align 4
  store i32 0, ptr %sum, align 4
  store i32 0, ptr %i, align 4
  %call = call i32 (...) @_sysy_getint()
  store i32 %call, ptr @loopCount, align 4
  call void @_sysy_starttime(i32 noundef 21)
  br label %while.cond

while.cond:                                           ;
  %0 = load i32, ptr %i, align 4
  %1 = load i32, ptr @loopCount, align 4
  %cmp = icmp slt i32 %0, %1
  br i1 %cmp, label %while.body, label %while.end7

while.body:                                           ;
  store i32 0, ptr %tmp, align 4
  store i32 0, ptr %j, align 4
  br label %while.cond1

while.cond1:                                          ;
  %2 = load i32, ptr %j, align 4
  %cmp2 = icmp slt i32 %2, 6
  br i1 %cmp2, label %while.body3, label %while.end

while.body3:                                          ;
  %3 = load i32, ptr %i, align 4
  call void @func(i32 noundef %3)
  %4 = load i32, ptr %tmp, align 4
  %5 = load i32, ptr @global, align 4
  %add = add nsw i32 %4, %5
```

- 遵照LLVM框架要求，编写LLVM Pass，优化LLVM IR
- 评分标准：你的程序和clang -O1的相对性能



```
; Function Attrs: nounwind
define dso_local noundef i32 @main() local_unnamed_addr #1 {
entry:
  %call = tail call i32 (...) @_sysy_getint() #3
  store i32 %call, ptr @loopCount, align 4, !tbaa !2
  tail call void @_sysy_starttime(i32 noundef 21) #3
  %0 = load i32, ptr @loopCount, align 4, !tbaa !2
  %cmp17 = icmp sgt i32 %0, 0
  br i1 %cmp17, label %while.cond1.preheader, label %while.end7

while.cond1.preheader:                            ; preds = %entry, %while.cond1.preheader
  %indvars.iv = phi i32 [ %indvars.iv.next, %while.cond1.preheader ], [ 0, %entry ]
  %i.019 = phi i32 [ %add6, %while.cond1.preheader ], [ 0, %entry ]
  %sum.018 = phi i32 [ %rem, %while.cond1.preheader ], [ 0, %entry ]
  %div = udiv i32 %indvars.iv, 6
  %add5 = add nsw i32 %div, %sum.018
  %rem = srem i32 %add5, 134209537
  %add6 = add nuw nsw i32 %i.019, 1
  %indvars.iv.next = add i32 %indvars.iv, 6
  %exitcond.not = icmp eq i32 %add6, %0
  br i1 %exitcond.not, label %while.cond.while.end7_crit_edge, label %while.cond1.preheader, !llvm.loop !6

while.cond.while.end7_crit_edge:                  ; preds = %while.cond1.preheader
  %1 = add nsw i32 %0, -1
  store i32 %1, ptr @global, align 4, !tbaa !2
  br label %while.end7

while.end7:                                       ; preds = %while.cond.while.end7_crit_edge, %entry
  %sum.0.lcssa = phi i32 [ %rem, %while.cond.while.end7_crit_edge ], [ 0, %entry ]
  tail call void @_sysy_stoptime(i32 noundef 37) #3
  tail call void @_sysy_putint(i32 noundef %sum.0.lcssa) #3
  tail call void @_sysy_putch(i32 noundef 10) #3
  ret i32 0
}
```

task5 （? 分，可选加分）

task0 演示&详解

预祝大家实验顺利！

YatCC-AI
编译原理课程实验

助教：顾宇浩
2025/2/28