lab2 phase1

PB21051012 刘祥辉

1.在 Light IR 简介里，你已经了解了 IR 代码的基本结构，请尝试编写一个有全局变量的 cminus 程序，并用 `clang` 编译生成中间代码，解释全局变量在其中的位置。

```
int globalVar = 10;  // 全局整数变量

int main() {
    int localVar = 5;  // 本地整数变量
    int result = globalVar + localVar;
    return result;
}
```

中间代码:

```
; ModuleID = 'cminus.c'
source_filename = "cminus.c"
target datalayout = "e-m:e-p270:32:32-p271:32:32-p272:64:64-i64:64-f8
target triple = "x86_64-pc-linux-gnu"

@globalVar = dso_local global i32 10, align 4

; Function Attrs: noinline nounwind optnone uwtable
define dso_local 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 5, i32* %2, align 4
  %4 = load i32, i32* @globalVar, align 4
  %5 = load i32, i32* %2, align 4
  %6 = add nsw i32 %4, %5
  store i32 %6, i32* %3, align 4
  %7 = load i32, i32* %3, align 4
  ret i32 %7
}

attributes #0 = { noinline nounwind optnone uwtable "frame-pointer"="a

!llvm.module.flags = !{!0, !1, !2, !3, !4}
!llvm.ident = !{!5}
```

```
!0 = !{i32 1, !"wchar_size", i32 4}
!1 = !{i32 7, !"PIC Level", i32 2}
!2 = !{i32 7, !"PIE Level", i32 2}
!3 = !{i32 7, !"uwtable", i32 1}
!4 = !{i32 7, !"frame-pointer", i32 2}
!5 = !{!"Ubuntu clang version 14.0.0-1ubuntu1.1"}
```

全局变量在其中的位置: `@globalVar = dso_local global i32 10, align 4`

2.Light IR 中基本类型 label 在 Light IR C++ 库中是如何用类表示的?

标签由 `BasicBlock` 表示

例如: `auto trueBB = BasicBlock::create(module,"trueBB",mainFun);` 创建了 `trueBB` 的标签

3.Light IR C++ 库中 `Module` 类中对基本类型与组合类型存储的方式是一样的吗? 请尝试解释组合类型使用其存储方式的原因。

不一样,

例如数组结构

```
ArrayType *Module::get_array_type(Type *contained, unsigned num_elemer
    if (array_map_.find({contained, num_elements}) == array_map_.end()
        array_map_[{contained, num_elements}] =
            std::make_unique<ArrayType>(contained, num_elements);
    }
    return array_map_[{contained, num_elements}].get();
}
```

这段代码确保每个不同的 `{contained, num_elements}` 组合都只创建一个相应的数组类型对象，以避免重复创建相同的数组类型对象，节省内存并提高性能。如果已经存在相同组合的数组类型对象，函数将返回现有的对象指针，否则将创建一个新对象。

分析 `calc` 程序在输入为 `4 * (8 + 4 - 1) / 2` 时的行为:

1. 请画出该表达式对应的抽象语法树（使用 `calc_ast.hpp` 定义的语法树节点来表示，并给出节点成员存储的值），并给节点使用数字编号。

2. 请给出示例代码在用访问者模式遍历该语法树时，访问者到达语法树节点的顺序。序列请按如下格式指明（序号为问题 1.a 中的编号）：3->2->5->1->1