Compilers Project3 - Code Generation

Pre-requisite

Since I'm using c++ to compile flex and bison files, **g++** supporting **c++17** is required.

Visitor

As is said in the project 2 report, I'm using Visitor Pattern to perform traversing of the AST.

Ref to Wiki:

> The visitor design pattern is a way of separating an algorithm from an object structure on which it operates. A practical result of this separation is the ability to add new operations to existing object structures without modifying the structures. It is one way to follow the open/closed principle.
>
> In essence, the visitor allows adding new virtual functions to a family of classes, without modifying the classes. Instead, a visitor class is created that implements all of the appropriate specializations of the virtual function. The visitor takes the instance reference as input, and implements the goal through double dispatch.

Self defined class:

<visitor.hpp>:

```
template<typename T, typename S, typename E>
    class AST_Visitor
```

Interface for visitor pattern, which defines the `visit` functions as virtual function.

```
class Visitor : public AST_Visitor<void, void, Exp_Info *>
```

Abstract base class for visiting Abstract Syntax Tree, which more concretely define how to visit children of AST Nodes in `visit_children(AST_Node *node)`

<tac.hpp>:

1. `Label` Defines the label representation ,which stores the name of label.
2. `Quadruple` The base class for all quadruples. Define the pure virtual function `to_string`, which is used in the IR generation.
3. `Label_Quadru : public Quadruple`, `Func_Quadru : public Quadruple`... All the derivative class that represents the three address code structure.
4. `TAC` Container for the quadruple. Constructed by `std::vector`.

<ir.hpp>

1. `IR_Generator : public Visitor` Defines the visitor using Visitor Pattern to traverse the Abstract Syntax Tree, and generate `Quadruple`s.

<optimizer.hpp>

1. `Basic_Block` Defines the basic block consisting a portion of instructions.
2. `TAC_Optimizer` Optimizer that would perform local optimization such as Neutral elimination, Constant folding, Strenght reduction and Constant propagation. However, I fail to complete them.

Features

Required

1. Generate given linear IR using spl file input.

Bonus

1. The structure can appear in the program, and they can be declared as function parameters.
2. Single-dimensional array can be declared as function parameters, and multi-dimensional arrays can be defined as local variables.