CS323 Compilers

Name: Yubin Hu

ID: 11712121

Basic Requirement

My code generator shall translate the input SPL program into the TAC instructions.

SPL program:

```
int main()
{
    int n;
    n = read();
    if (n > 0) write(1);
    else if (n < 0) write (-1);
    else write(0);
    return 0;
}
```

TAC:

```
FUNCTION main :
READ t1
v1 := t1
t2 := #0
IF v1 > t2 GOTO label1
GOTO label2
LABEL label1 :
t3 := #1
WRITE t3
GOTO label3
LABEL label2 :
t4 := #0
IF v1 < t4 GOTO label4
GOTO label5
LABEL label4 :
t5 := #1
t6 := #0 - t5
WRITE t6
GOTO label6
LABEL label5 :
t7 := #0
WRITE t7
LABEL label6 :
LABEL label3 :
t8 := #0
RETURN t8
```

Follow Appendix A(Three-Address Code Specification)'s `Table 5: Three-address-code specification`, we implement `ir.cpp` and `ir.hpp`. Special attention should be paid to the

fact that WRITE/READ is not defined in lex, syntax, semantic, so in addition to processing in ir, the definition must be implemented in the above places.

We implement this base class TAC in `ir.hpp`, other classes inherit the base class TAC.

- `void irProgram(AST *root);` Enter ir program.
- ```
  void irExtDecList(AST *node, Type * type);
  void irExtDef(AST *node);
  Type *irSpecifier(AST *node);
  Type *irType(AST *node);
  Type *irStructSpecifier(AST *node);
  void irFunc(AST *node, Type *type);
  void irCompSt(AST *node);
  void irDefList(AST *node);
  void irDef(AST *node);
  void irDecList(AST *node, Type *type);
  void irStmt(AST *node);
  void irStmtList(AST *node);
  void irDec(AST *node, Type *type);
  TAC* irVarDec(AST *node, Type* type);
  int irExp(AST *node, bool single=false);
  void irVarList(AST *node);
  void irParamDec(AST *node);
  vector<int> irArgs(AST *node);
  ```

  Follow the `Table 5: Three-address-code specification`, we implement them.

Test

We use `genIR.sh` to test the IR.

Bouns

NULL
  ```