CS323 Project Phase 3 Intermediate Code generation Report

组员：李昕 12012138 & 廖铭骞 12012919

如何运行

1. 先通过 `make clean` 清除上次编译的多余文件
2. 通过 `make` 编译运行
3. 生成的 `splc` 可执行文件将会在 `bin` 文件夹下面
4. 使用命令 `./bin/splc test/test_3_r01.spl` 将会生成对应的三地址码

数据结构

在本次项目当中，我们先设计了一个三地址码的抽象类。由于每一种三地址码都有独一无二的输出表示方式，所以针对每一种可能的三地址码结构，我们都专门设计了单独的类继承自这个抽象类，并且在对应的类当中，我们制定了`to_string()` 方法用于三地址码的输出。

举个例子，用赋值语句对应的三地址码类来进行阐释，我们设计了`AssignTAC` 这个类，它的构造器将使用两个 `int` 类型的地址进行构造，一个地址指明将要被赋值寄存器的地址，而另一个地址指明了将要将该寄存器赋值成的值所在的位置。该类对应的输出方式如下所示。

```
string to_string()
{
        char buffer[1024];
        sprintf(buffer, "t%d := %s", TAC::address,
addr2str(right_address).c_str());
        return buffer;
    }
```

其他的三地址码类也是类似于这样的定义，这些三地址码类都定义在 `inter_rep.hpp` 当中。

设计思路

自顶向下的解析

在此次 project 当中，我们按照第一次 project 定义的文法规则自顶向下地进行解析，并且针对每一种文法符号我们都实现了 `inter_X()` 的函数对其进行解析，以`ExtDef` 文法符号的解析作为例子，它有三种解析方式

```
/**
 * ExtDef: Specifier ExtDecList SEMI
 *       | Specifier SEMI
 *       | Specifier FunDec CompSt
 */
```

针对每一种解析方式我们都需要进行处理，如果是第一种解析方式，那么就需要对第一个子元素进行 `translate_Specifier()` 处理，对第二个子元素进行 `translate_ExtDecList()` 处理，通过这样一步步地自顶向下进行处理，最终会将最底部的式子解析成为三地址码，实例化成对应的三地址码类，最终统一调用对应的 `to_string()` 方法打印出来， 就像前面提到的一样。

Bonus 部分

对一维数组作为函数参数的支持 以及 对多维数组作为局部变量的支持

对于数组类型的处理对应于文法符号中的 `Exp: Exp LB Exp RB`

在我们的程序当中，如果检测到 `LB`， 即左中括号 ('[') 的时候，就会递归地进行计算，将其看做一个多维数组进行处理，由于在本次 project 当中基本类型只会涉及 `int`，所以，基本类型的大小可以统一设置为4，用以计算数组的长度以及偏移量。对于中括号内部的 `Exp` 文法符号，则需要再次调用 `translate_Exp()`进行解析，从而达到解析多维数组的目的。

解析范例：
对于以下这段程序，在程序当中涉及了多维数组的局部变量以及函数参数有多维数组。

```
int test(int m[1])
{
    int n[1][2];
    n[0][0] = 3;
    return 1;
}
```

生成的三地址码如下所示：

```
FUNCTION test :
PARAM t2
DEC t3 8
t4 := #3
t5 := &t3
t6 := #0
t7 := t6 * #8
t8 := t5 + t7
t9 := #0
t10 := t9 * #4
t11 := t8 + t10
*t11 := t4
t13 := #1
RETURN t13
```

对于此处的二维数组`n`而言，第二位的大小是 2，所以 结合基本类型的大小是 4 字节，所以 产生了`DEC t3 8`这样的三地址码，然后通过一系列偏移量的计算来实现对相应的数组位置进行赋值。

对 struct 类型的支持

解析范例：

对于以下包含有`struct`类型的程序

```
struct Apple
{
    int ID;
    int length;
    int size;
};

struct Farm
{
    int ID;
};

int main()
{
    struct Apple a1;
    struct Farm farm;
    a1.length = 1;
    a1.size = 2;
    write(a1.length);
    farm.ID = 20;
    return 0;
}
```

我们经过解析，可以产生以下一系列的三地址码

```
FUNCTION main :
DEC t2 12
DEC t3 4
t4 := #1
t5 := &t2
t6 := t5 + #4
*t6 := t4
t8 := #2
t9 := &t2
t10 := t9 + #8
*t10 := t8
t12 := &t2
t13 := t12 + #4
```

```
t14 := *t13
WRITE t14
t16 := #20
t17 := &t3
t18 := t17 + #0
*t18 := t16
t20 := #0
RETURN t20
```

和数组类似，我们也是通过计算偏移量的方式去得到`struct`对应的成员信息，从而进行赋值以及计算。

最终能够通过`extra`测试中的 1, 2, 4 测试，生成的三地址码也已经打包上传在同一个文件夹当中。

以上就是我们小组的 project phase 3 的报告，感谢助教以及老师的阅读！！！