工作流：

Code Generator

Intermediate representation

Intermediate Code Generator

token stream

Lexical Analyzer

Syntax Analyzer

DNA machine code

Syntax tree

Character stream

Syntax：

stmts 🡪 stmts stmt | （先不用） (statements)

stmt 🡪 id=expr; (statement)

|if expr: stmt else： stmt

| while expr: stmt

| for (expr; expr; expr) stmt

|expr logop expr

expr 🡪 expr+term (expression)

| expr – term

| term relop term

| term

term 🡪 term\*factor (term)

| term / factor

| factor

factor 🡪 (expr) (factor)

| num

| id

relop 🡪 > | < | >= | <= | == (relation operator)

logop -> or |and |xor|not (logic operator)

d\_lexer:

将source program 转成 token stream

parser ：

递归的方法构建Syntax tree

调试记录：

1. d\_par中需要区分中间节点和端点，在内部的变量命名上做了一些调整，后面如果出现与node有关的错误，需到当地检查。
2. 建立树的思路：

先确定整个是什么表达式（默认每次编译的是一个stmt，还没有考虑stmts），建立对应的节点，然后进入到此节点，判断每一个child属于哪种类型，建立对应的节点，并建立母子关系，依次递归向下建立，完成后append到一个headers list中。

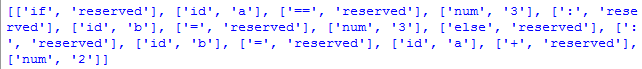
1. 需要注意的事项：默认只编译一个stmt，没有定义block，所以while的body部分只执行一句话，可以是复杂的运算，但不能是独立的两句。以后应该把这部分补上
2. for语句还没有定义。
3. 每一个class的children只定义了某些具体的种类，并没有涵盖所有的可能性。如果需要，可以再补充。

运行记录：

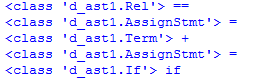
mypro=' if a == 3 : b = 3 else : b = a + 2 '

tokens=d\_lexer1.d\_lex(mypro)

得到：

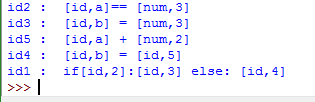


headers=d\_parser.d\_par(tokens)



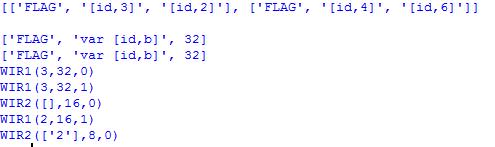
生成syntax tree，建立了5个对应的节点。

然后调用d\_inte\_gene:



生成对应的中间表达式

产生机器码：



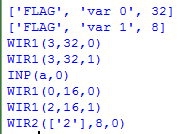
错误：需要逐个解决

1. flag部分两个b赋值不同，还不能区分。

通过指向变量地址而解决。

1. 对于没有预定义的变量，是一个空值。
2. 赋值语句还没有很合适的指令。
3. 识别出所有的input并编号。
4. 区分常数和imput的变量

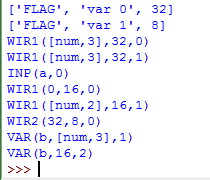
再次修改后：



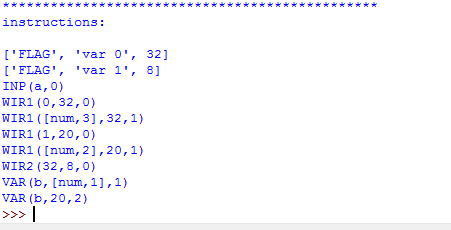
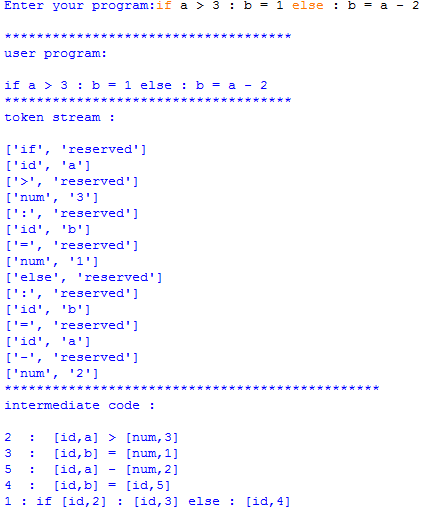
人工输入的变量，给一个与名字对应的序号

修改后，增加一个var指令，上述问题基本解决

所以computer中必须要有memory



1.12 测试结果：



**待解决问题：**

1. 只考虑了if语句，其他条件语句需要加进去
2. 关于逻辑运算的还没有写进去。
3. 关系运算的输出还不明确：一个比较器有== < > <= >= 五个输出端口，需要根据运算符号激活对应的端口
4. 需要考虑其他用到的逻辑门的开关情况。