19335030_陈至雪

编译器构造

实验要求

> 一个简单文法的编译器的设计与实现
>
> 一个简单文法的编译器前段的设计与实现
>
> 定义一个简单程序设计语言文法（包括变量说明语句、算术运算表达式、赋值语句；扩展包括逻辑运算表达式、If语句、While语句等）；
>
> 扫描器设计实现；
>
> 符号表系统的设计实现；
>
> 语法分析器设计实现；
>
> 中间代码设计；
>
> 中间代码生成器设计实现。
>
> 一个简单文法的编译器后段的设计与实现
>
> 中间代码的优化设计与实现（鼓励）；
>
> 目标代码的生成（使用汇编语言描述，指令集自选）；
>
> 目标代码的成功运行。

实验步骤

说明

此处实现的是简易的c语言编译器，实现了词法分析器，语法分析器，语义分析器，和快成功的目标代码生成器。

词法分析器

（1）首先定义标识符、保留字、常数、运算符、界符和需略除的符 号的范围。

标识符	**标识符可以是字母、数字、下划线(A ~ Z, a ~ z, 0~9, _)组成的字符串，并且第一个字符必须是字母或下划线。**
保留字	指在高级语言中已经定义过的字，使用者不能再将这些字作为变量名或过程名使用。
常数	数字0~9
运算符	包括算术运算符、关系运算符、逻辑运算符、赋值运算符、位运算符、其他运算符
界符	限制界限所用，有单字节界符和双字节界符
需略除的符号	空格，制表符，换行符

标识符：

1. 标识符是严格区分大小写的。
2. 标识符不能是C语言的关键字和保留标识符。
3. 标识符长度限制跟C语言标准和编译器环境有关。

保留字：

字	**int,long,short,float,double,char,unsigned,signed,const,void,volatile,enum,struct,union**
语句定义保留字	if,else,goto,switch,case,do,while,for,continue,break,return,default,typedef
存储类说明保留字	auto,register,extern,static
长度运算符保留字	sizeof

运算符：

算术运算符	**+，-，*，/,%,++,--**
关系运算符	==，!=, >, <, >=, <=
逻辑运算符	&&, ||, !
赋值运算符	=, +=, -=, *=, /=, %=, <<=, >>=, &=, ^=, |=
位运算符	&, |, ^, ~, <<, >>
其他运算符	Condition ? X:Y, . , ->, &, *, ,

界符：

单字节界符	**；**
双字节界符	{},(),[], “和”，‘和’

需略除的符号：

注释符	**\ \，/* 和 */,**
其他	空格，制表符，换行符

（2）进行种别编码：

单词符号	种别编码	助记符	内码值
常数	0	num	num在常数表中的位置
标识符	1	id	id在符号表中的位置
字符串	2	s	s在字符串表中的位置
int	3	int	
long	4	long	
short	5	short	
float	6	float	
double	7	double	
char	8	char	
unsigned	9	unsigned	
signed	10	signed	
const	11	const	
void	12	void	
volatile	13	volatile	

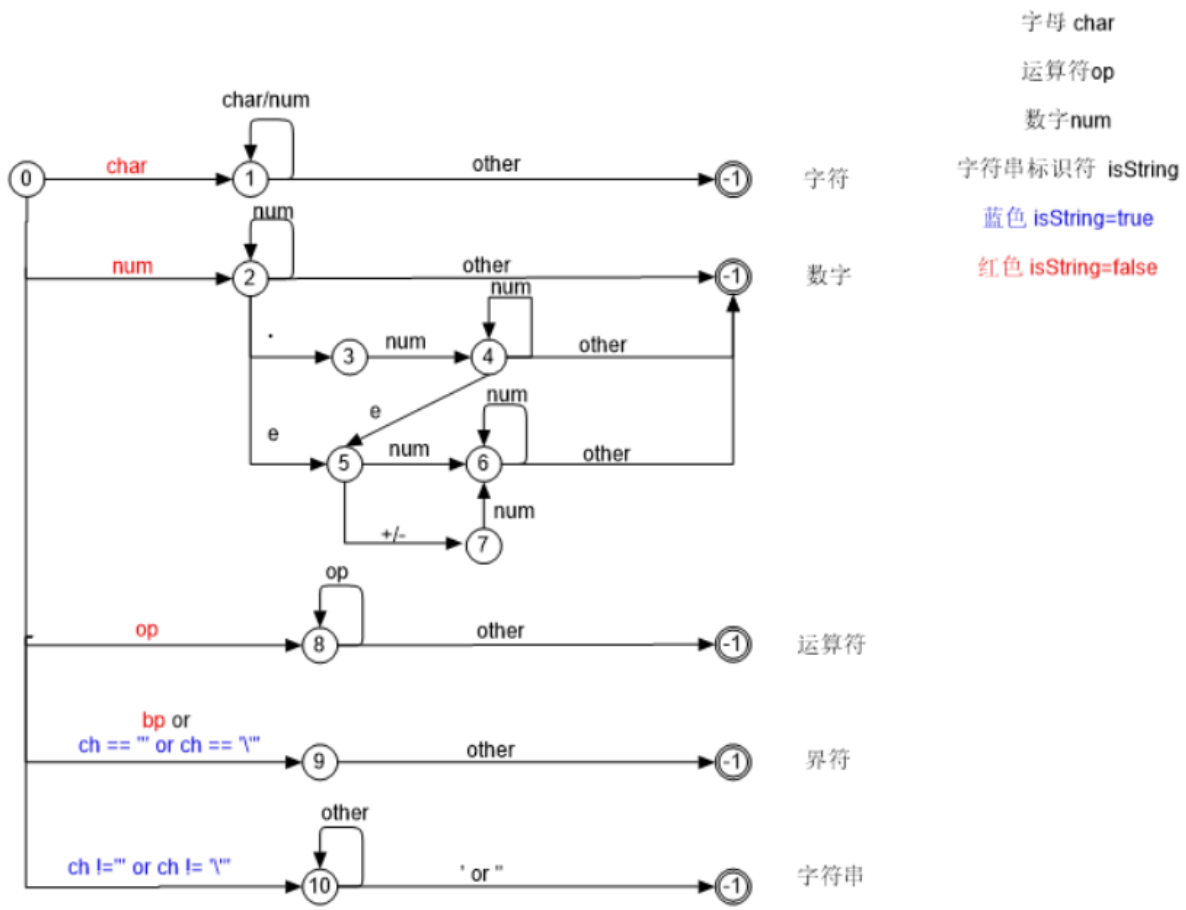
enum	14	enum	
struct	15	struct	
union	16	union	
if	17	if	-
else	18	else	-
goto	19	goto	-
switch	20	switch	-
case	21	case	-
do	22	do	-
while	23	while	-
for	24	for	-
continue	25	continue	-
break	26	break	-
return	27	return	-
default	28	default	-
typedef	29	typedef	-

单词符号	种别编码	助记符	内码值
auto	30	auto	-
register	31	register	-
extern	32	extern	-
static	33	static	-
sizeof	34	sizeof	-
+	35	+	-
-	36	-	-
*	37	*	-
/	38	/	-
%	39	%	-
++	40	++	-
--	41	--	-
==	42	==	-
!=	43	!=	-
>	44	>	-

<	45	<	-
>=	46	>=	-
<=	47	<=	-
&&	48	&&	-
||	49	||	-
!	50	!	-
=	51	=	-
+=	52	+=	-
-=	53	-=	-
*=	54	*=	-
/=	55	/=	-
%=	56	%=	-
<<=	57	<<=	-
>>=	58	>>=	-
&=	59	&=	-

单词符号	种别编码	助记符	内码值
^=	60	^=	-
|=	61	|=	-
&	62	&	-
|	63	|	-
^	64	^	-
~	65	-	-
<<	66	<<	-
>>	67	>>	-
;	68	;	-
{	69	{	-
}	70	}	-
(	71	(	-
)	72	)	-
[	73	[	-
]	74	]	-
"	75	"	-
'	76	'	-
,	77	,	-
#	78	#	-
.	79	.	-
bool	**80**	**bool**	**-**

(3) 建立自动机



语法语义分析器

LL(1)文法:

```
文法开始:
S->int main(){A return 0;}  (0)

算术运算:
L->TL'(1)
L'->+TL'(2)|-TL'(3)|$(4)
T->FT'(5)
T'->*T(6)|/T(7)|$(8)
F->(L)(9)
F->id(10)|num(11)

声明:
U'->=L(12)|$(13)
U->idU'(14)
Z'->,Z(15)|$(16)
Z->UZ'(17)
Y->int(18)|char(19)|bool(20)
X->YZ;(21)

O->++(22)|--(23)|$(24)
Q->idO(25)|$(26)

布尔运算
E->HE'(27)
E'->&&E(28)|$(29)
H->GH'(30)
```

```
H’->||H（31）|$（32）
G->FDF（33）
D->><（34）|>（35）|==（36）|!=（37）
G->!E（39）

控制语句
B->if (E){A}else{A}（40）
B->while(E){A}（41）
B->for(YZ;G;Q){A}（42）

复合语句
A->CA（43）
C->X（44）|B（45）|R（46）
A->$（47）

赋值
R->id=L;（48）
```

求LL(1)分析表：

	return	+ (35)	- (36)	* (37)	/ (38)	++ (40)	-- (41)	&& (48)	|| (49)	int (3)	char (8)	bool (80)	((71)	) (72)	id (1)	num (0)	= (51)	, (77)	< (45)	> (44)	== (42)	!= (43)	! (50)	if (17)	{ (69)	} (70)	else (18)	while (23)	for (24)	; (68)
S										0																				
X										21	21	21																		
Y										18	19	20																		
Z															17															
Z'										16	16	16			16			15						16		16		16	16	16
U															14															
U'																	12	13												13
R															48															
L													1		1	1														
L'		2	3							4	4	4		4	4			4						4		4		4	4	4
T													5		5	5														
T'		8	8	6	7					8	8	8		8	8			8						8		8		8	8	8
F													9		10	11														
O						22	23							24																
Q														26	25															
E														27	27	27														
E'								28						29																29
H													30		30	30							30							
H'								32	31					32																32
G													33		33	33							39							
D																			34	35	36	37								
B																								40				41	42	
A	47									43	43	43			43									43				43	43	
C										44	44	44			46									45				45	45	

将上述文法扩展为LL(1)属性翻译文法：

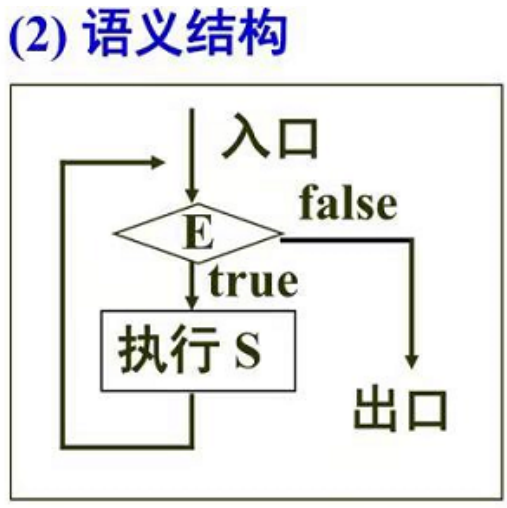
用<>表示动作。

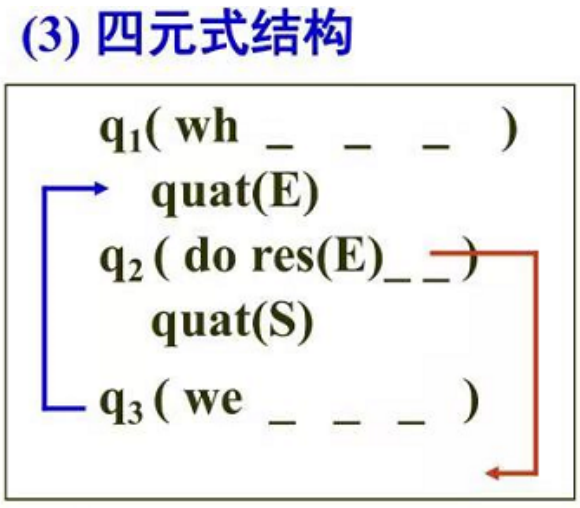
```
文法开始：
S->int main()<main,I1,-,->{A return 0;}<end,I1,-,-> （0）

算术运算：
L->TL’（1）
L’->+T <GEQ(+)> L'（2）|-T <GEQ(-)> L'（3）|$（4）
T->FT’（5）
T’->*F <GEQ(*)> T'（6）|/F <GEQ(/)> T'（7）|$（8）
F->(L)（9）
F->id <PUSH(id)>（10）|num <PUSH(num)>（11）

声明：
U’->=L <GEQ(=)>（12）|$（13）
U->id <PUSH(id)> U’（14）
Z’->,Z（15）|$（16）
Z->UZ’（17）
Y->int（18）|char（19）|bool（20）
```

```
X->YZ;（21）

O->++<GEQ(++)>（22）|--<GEQ(--)>（23）|$（24）
Q->id<PUSH(id)>O（25）|$（26）

布尔运算
E->HE’（27）
E’->&&E <GEQ(&&)>（28）|$（29）
H->GH’（30）
H’->||H <GEQ(||)>（31）|$（32）
G->FDF <GEQ(D)>（33）
D->< <PUSH(<)>（34）|> <PUSH(>)>（35）|== <PUSH(==)>（36）|!= <PUSH(!=)>（37）
G->!E <GEQ(!)>（39）

控制语句
B->if (E) <IF(if)> {A}else <EL(el)> {A} <IE(ie)>（40）
B->while <WH(wh)> (E) <DO(do)> {A} <WE(we)>（41）
B->for(YZ; <FOR(for)> G; <DO(do)> Q){A} <FE(fe)>（42）

复合语句
A->CA（43）
C->X（44）|B（45）|R（46）
A->$（47）

赋值
R->id <PUSH(id)> =L <GEQ(=)>;（48）
```

四元式设计：

while循环语句的四元式：





S -> while {WH(wh)} (R) {DO(do)} S {WE(we)};

※ WH(wh) --- 循环头函数；

(1) SEND(wh, _ , _ , _);

※ DO(do) --- DO 函数；

(1) SEND(do, SEM[m] ,_ , _);

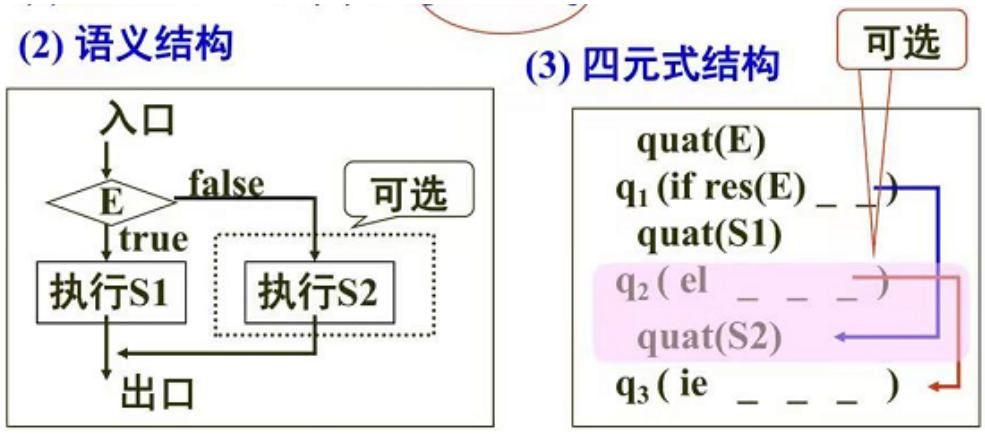
(2) POP ;

需要回填

※ WE(we) --- 循环尾(while end) 函数；

(1) SEND(we,_ ,_ , _);

if判断语句的四元式：



S -> if(R) {IF(if)} S; [else {EL(el)} S] {IE(ie)}

※ IF(if) --- if 函数；

(1) SEND(if, SEM[m], _ , _);

(2) POP;

※ EL(el) --- else 函数；

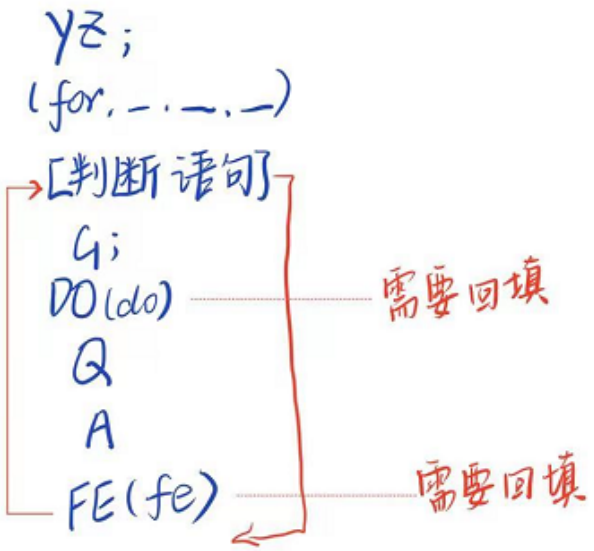
(1) SEND(el,_ ,_ ,_);

※ IE(ie) --- ifend 函数；

(1) SEND(ie, _ ,_ ,_);

需要回填

for循环的四元式：



目标代码生成：

目标代码采用的是MIPS汇编语言，其寄存器规定如下：

名称	编号	用途	被调保留?
$zero	$0	常量0	不适用
$at	$1	汇编临时用	否
$v0–$v1	$2–$3	函数返回值和表达式求值	否
$a0–$a3	$4–$7	函数参数	否
$t0–$t7	$8–$15	临时	否
$s0–$s7	$16–$23	临时保存值	是
$t8–$t9	$24–$25	临时	否
$k0–$k1	$26–$27	保留内核用	不适用
$gp	$28	全局指针	是（除PIC代码）
$sp	$29	栈指针	是
$fp	$30	帧指针	是
$ra	$31	返回地址	不适用

实验结果：

测试代码一：

```
int main(){
        int a = 10;
        int b = 5;
        int c = a + b;
        if( c > 0 && a > 5 ){
                c = 0;
        }
        else{
                c = 1;
        }
        return 0;
}
```

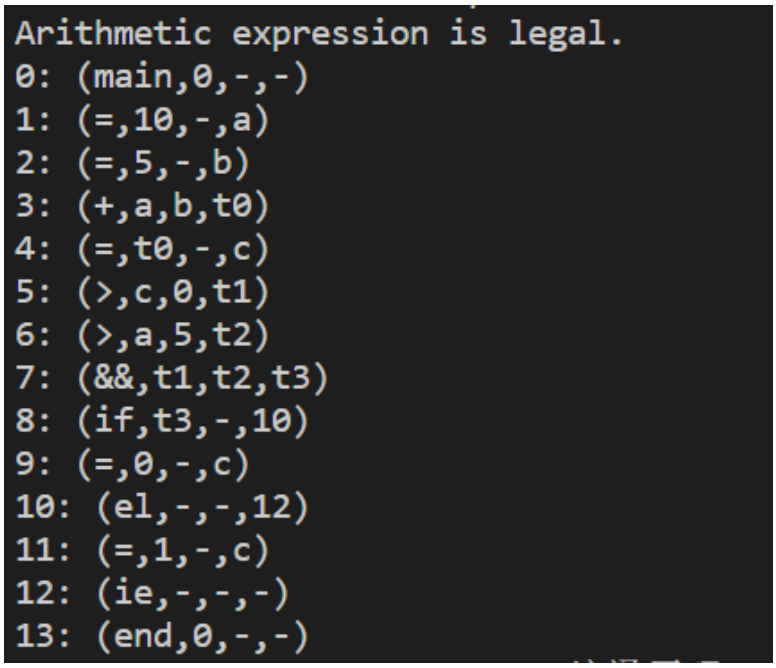
测试的功能包括：if-else逻辑，逻辑运算，加法运算以及赋值语句。

生成的token序列如下：

Characters	Code
int	3
main	1
(	71
)	72
{	69
int	3
a	1
=	51
10	0
;	68
int	3
b	1
=	51
5	0
;	68
int	3
c	1
=	51
a	1
+	35
b	1
;	68
if	17
(	71
c	1
>	44
0	0
&&	48
a	1
>	44
5	0
)	72

```
{                                                    69
c                                                    1
=                                                    51
0                                                    0
;                                                    68
}                                                    70
else                                                 18
{                                                    69
c                                                    1
=                                                    51
1                                                    0
;                                                    68
}                                                    70
return                                               27
0                                                    0
;                                                    68
}                                                    70
```

生成的四元式序列如下：



可以看到赋值语句正确，比较运算四元式生成正确，逻辑运算四元式生成正确，if 和 el 的跳转地址回填正确，因此 if-else逻辑四元式生成正确。

测试代码二：

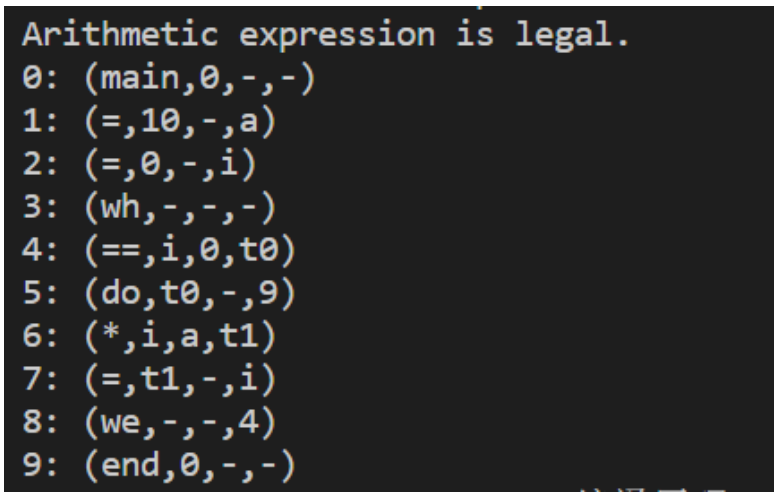
```
int main(){
        int a = 10;
        int i = 0;
        while(i==0){
                i = i * a;
        }
        return 0;
}
```

测试的功能包括：while循环逻辑，乘法运算。

生成的token序列如下：

```
Characters                                                Code
int                                                       3
main                                                      1
(                                                         71
)                                                         72
{                                                         69
int                                                       3
a                                                         1
=                                                         51
10                                                        0
;                                                         68
int                                                       3
i                                                         1
=                                                         51
0                                                         0
;                                                         68
while                                                     23
(                                                         71
i                                                         1
==                                                        42
0                                                         0
)                                                         72
{                                                         69
i                                                         1
=                                                         51
i                                                         1
*                                                         37
a                                                         1
;                                                         68
}                                                         70
return                                                    27
0                                                         0
;                                                         68
}                                                         70
```

生成的四元式序列如下：



可以看到乘法运算的四元式生成正确，while循环的地址回填也无问题，因此while循环的四元式生成正确。

测试代码三：

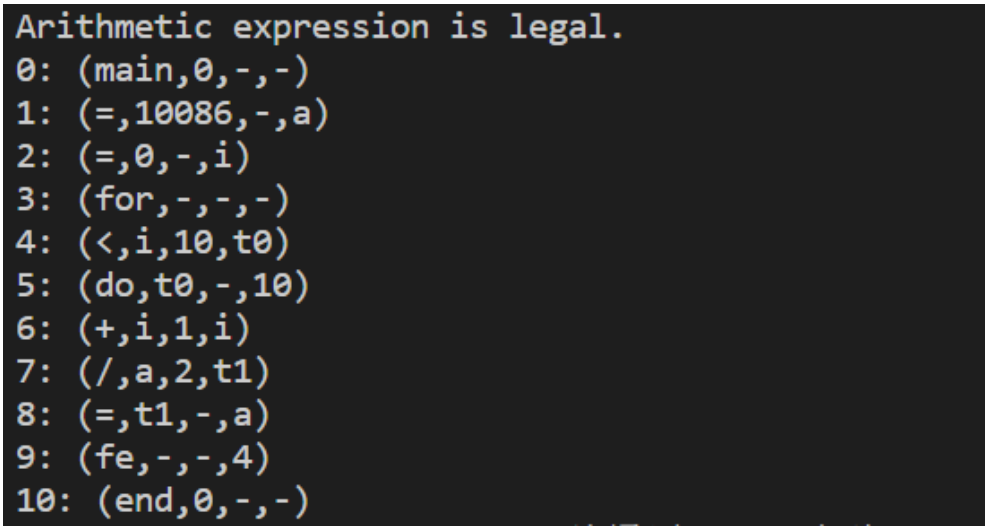
```
int main(){
    int a = 10086;
    for( int i = 0; i < 10; i ++ ){
        a = a / 2;
    }
    return 0;
}
```

测试的功能包括：for循环，除法运算。

生成的token序列如下：

Characters	Code
int	3
main	1
(	71
)	72
{	69
int	3
a	1
=	51
10086	0
;	68
for	24
(	71
int	3
i	1
=	51
0	0
;	68
i	1
<	45
10	0
;	68
i	1
++	40
)	72
{	69
a	1
=	51
a	1
/	38
2	0
;	68
}	70
return	27
0	0
;	68
}	70

生成的四元式序列如下：



可以看到除法的四元式生成正确，for循环的地址回填正确，因此for循环的四元式生成正确。

实验感想：

本次实验是一次综合实验，结合了本学期教的大部分内容，将理论与实践结合，使我通过实验对理论知识有了更深的了解和认识。

- 首先是找到一个C语言的文法，将其化简为我可以实现的简单C语言文法，然后对文法进行消除左递归和同一子项，使文法变为LL(1)文法；
- 然后，得到LL(1)文法后需要将其扩展为LL(1)属性翻译文法，此处我参考了理论课上的PPT和上次实验进行的算术表达式的语义分析器，成功将LL(1)文法扩展为了LL(1)属性翻译文法。
- 最后是代码的实现，代码的实现过程中开始的时候并不知道怎么设计发杂的符号表。虽然看了很久的理论资料，但到了要实现的时候仍然是无从下手，最后我决定先从简单的数据结构开始存储数据（即进行语义分析器的实现时需要用到什么数据才建立什么表进行存储），后面发现有些表是一个变量的不同属性，于是就可以将这些相关的表封装成一个结构体，如此优化我的代码结构。
- 目标代码的生成，我完成到一半，还有些语句尚未完成好。

注释：

- compiler.cpp为词法分析器
- 测试文件名需改为 test.cpp
- 词法分析器生成文件 LexicalAnalyzerOut.txt ，存储的是token序列。
- main.cpp 为语义分析器，运行后在终端输出四元式序列。
- Github链接：https://github.com/aZhiChen/hw.git