<program> -> <write\_stat>

<write\_stat> -> write( E )

write (1 + 2 \* 3)

we should be able to parse the above code.

void compile\_call(){

gettoken();

ip = 0;

parse\_write();

gen1(op\_stop);

}

void gen1(char b){

code[ip++]=b;

}

void parse\_write(){

match(TK\_WRITE);

match(TK\_LP);

E();

match(TK\_RP);

gen1(op\_printint);

}

**Expression Parsing**

E -> T E'

E' -> + T E' | -T E' | e

T -> F T'

T' -> \* F T' | - F T' | e

F -> + F | - F | lit | id | (E)

void E(){

T(); E'();

}

void E'(){

if (curtoken == TK\_PLUS){

match(TK\_PLUS); T(); E'();}

else if (curtoken == TK\_MINUS){

}

else{

}

}

Implement tail recursion as while loop:

void E(){

T();

curop = curtoken;

while( curop == TK\_PLUS | curop == TK\_MINUS){

match(curop);

T();

if ( curop == TK\_PLUS)

gen1(op\_add);

else

gen1(op\_sub);

}

}

void T(){

F();

curop=curtoken;

while(curop == TK\_TIMES || curop == TK\_DIV){

match(curop);

F();

if ( curop == TK\_TIMES)

gen1( op\_mul );

else

gen1( op\_div );

}

}

F(){

if (curtoken == TK\_INTLIT ){

//push it

gen1(op\_pushi);

gen4(token\_value);

gettoken();

}

if (curtoken == TK\_REALLIT){

//push it

gettoken();

}

else

else error();

}

All the parsing function will return the type. F, T, E will return a value TYPE. This has four values:

I - Integer

R - Real

B - Boolean

C - Char

F will return the type of the literal.

in E(),

Type1 = T();

Type2=T();

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| I\2 | I | R | B | C |
| I | Add/I |  | NO | NO |
| R |  | Add/f | NO | NO |
| B | NO | NO | NO | NO |
| C | NO | NO | NO | NO` |

R + I ---> OK!

I + R:

xchg --> swap two top elements of the stack.

Then convert to real

xchg again

addF

T1 = genOP(t1,t2,cur\_op)

Division:

two operators in pascal

/ - Real division, but will also work on integers.

DIV - Integer Division

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| AND | I | R | B | C |
| I |  | X | X | X |
| R | X | X | X | X |
| B | X | X | AND/B | X |
| C | X | X | X | X |