GRAMÁTICA 4.0

Literal: (-?[0-9]*) U (0x[abcdef0-9]*) U ([a-z]) U (“[a-z]*[^\n$”]”)

ID: ((_ | .)(_ | . | a-z0-9)*(a-z0-9)* | ((a-z)(a-z0-9 | _ | .)*([(a-z0-9)*])?)

(#): ação no esquema de tradução

Programa	S -> DBF
Declarações	D -> {(V | C) D} | B
Declaração de variáveis	V -> 'var' TL; {TL;}*
Tipos	T -> ‘char’ | ‘integer’ (3)
Lista de IDs ou valores	L -> ID1 (1,2) { '='literal (4) | '['literal(6)']' | ,(5)ID2 }* ';'(5 ID)
Declaração Constantes	C -> ‘const’ ID (1) ‘=’ Literal (2) ’;’
Bloco de comandos	B -> {A | R | T | ';' | E | W}*
Comando unico ou bloco	B1 -> '{' B '}' | A | R | T | ';' | E | W
Atribuição	A -> ID (7)(8)(9) ‘=’ X (4);
Repetição	R -> 'for' ID (7)(8) '=' X (10) 'to' X(10) ['step' X(10)] 'do' B1
Teste	T -> 'if' X (11) 'then' B1 ['else' B1]
Leitura	E -> ‘readln(‘ID(7)’);’ (12)
Escrita	W -> {'write' | 'writeln'} '('X2');'(13)
op relacionais	O -> = | <> | < | > | >= | <=
Expressão	X -> Xs1 (9) [O Xs2 (16)]
Expressao simples	Xs -> [-] (17) T1 (9) {(+ | - | ‘or’) T2 (18)} (19)
Termo	T -> F1 (9) {(* | / | % | ‘and’) F2 (18)}
Fator	F -> ‘(‘ X (9) ‘)’ | const (14) | id (7) (15)
lista de exp	X2 -> X {,X}*
Fim de arquivo	F -> EOF

Esquema de tradução

Lex: registro léxico contendo tipo, classe, lexema e tamanho. Global.
CD: contador de posição de memória. Global.
TP: contador de temporários. Global.

(1) -> { se id.classe == 0 então id.classe = lex.classe senão ERRO }
(2) -> {id.tipo = lex.tipo}
(3) -> {se lexema='integer' lex.tipo = inteiro senão lex.tipo = caractere}
(4) -> {se id1.tipo <> lex.tipo então ERRO senão gera código id1}
(5) -> {se id.tipo == inteiro então escreva "sword id.val", CD=CD+2
    (a) senão escreva "byte id.val", CD=CD+1}
(6) -> {se lex.tipo <> inteiro então ERRO
    (a) senão se lex.lexema*tamanho(lex.tipo) > 4k então ERRO
    (b) senão id.tamanho = lex.lexema, gera código id}
(7) -> {se id.classe == 0 então ERRO}
(8) -> {se id.classe == const então ERRO}
(9) -> {pai.tipo = filho.tipo, pai.end = filho.end, pai.tamanho = filho.tamanho}
(10) {se lex.tipo<>inteiro então ERRO senão gera código}
(11) {se X.tipo <> inteiro então ERRO}
(12) {gera codigo leitura}
(13) {gera codigo escrita}
(14) {F.end=novoTemp, F.tamanho=1, se lex.tipo = inteiro escreva
    (a) "mov ax, lex.lexema mov DS:[F.end], ax"
    (b) senão escreva "dseg SEGMENT PUBLIC byte "lex.lexema$" dseg ENDS"}
(15) {F.end=id.end, F.tamanho = id.tamanho, F.tipo = id.tipo}
(16) {se Xs1.tipo <> inteiro ou Xs2.tipo <> inteiro então ERRO
    (a) senão X.end = novoTemp, X.tam = 1, gera código de acordo com OP e coloca em X.end}
(17) {Xs.negativo = verdadeiro}
(18) {se 1.tipo <> 2.tipo então ERRO
    (a) senão gera código de acordo com OP}
(19) {se Xs.negativo == verdadeiro então escreva
    (a) "mov ax, DS:[Xs.end] neg ax mov DS:[Xs.end], ax"}