```
grammar ExprAGCalc;

@header {
package ag;
import java.util.*;
}

@parser::members {
  Map<String, Integer> memory = new HashMap<>();

  int eval(int left, int right, int op) {
    switch (op) {
      case ADD : return left + right;
      case SUB : return left - right;
      case MUL : return left * right;
      case DIV : return left / right;
      default : return 0;
    }
  }
}

prog : stat* EOF ;

stat : expr { System.out.println($expr.val); }
     | ID '=' expr { memory.put($ID.text, $expr.val); }
     ;

expr returns [int val]
    : left = expr op = ('*' | '/') right = expr { $val = eval($left.val, $right.val, $op.type); }
    | left = expr op = ('+' | '-') right = expr { $val = eval($left.val, $right.val, $op.type); }
    | '(' expr ')'                              { $val = $expr.val; }
    | ID                                        { $val = memory.getOrDefault($ID.text, 0); }
    | INT                                       { $val = $INT.int; }
    ;

ADD : '+' ;
SUB : '-' ;
MUL : '*' ;
DIV : '/' ;

ID : [a-z] ;
INT : [0-9] ;
WS : [ \t\r\n] -> skip;
```