

# ESCUELA POLITÉCNICA NACIONAL

**FACULTAD DE INGENIERÍA DE SISTEMAS CIENCIAS DE LA COMPUTACION**

**COMPILADORES Y LENGUAJES**

# Informe – Prueba 9

**GRUPO:** GR1COM

## FECHA DE ENTREGA: 12/03/2021

**INTEGRANTES:** Alejandro Moya, Leonardo Andrade, Jefferson Cando

Usamos LEX y YACC para implementar un programa de lógica matemática, que nos permitirá determinar por medio de tablas de verdad el resultado de una expresión de compuertas lógicas.

**CÓDIGO LEX**

|  |
| --- |
| %{  #include <stdlib.h>  void yyerror(char \*);  #include "bcalc.tab.h"  %}  %%  not return NOT;  and return AND;  or return OR;  xor return XOR;  xnor return XNOR;  nand return NAND;  nor return NOR;  [a-z] {  yylval = \*yytext - 'a';  return VARIABLE;  }  [0-1]+ {  yylval = atoi(yytext);  return BOOLEAN;  }  [()=\n] { return \*yytext; }  [ \t] ;  , {return \*yytext; }  . yyerror("invalid character");  %%  int yywrap(void) {  return 1;  } |

**CÓDIGO YACC**

|  |
| --- |
| %{  #include <stdio.h>  void yyerror(char \*);  int yylex(void);  int sym[26];  %}  %token BOOLEAN VARIABLE  %token AND OR NOT NAND NOR XOR XNOR  %left AND OR NOT NAND NOR XOR XNOR  %%  program:  program statement '\n'  |  ;  statement:  expr { printf("%d\n", $1); }  | VARIABLE '=' expr { sym[$1] = $3; }  | statement ',' statement  ;  expr:  BOOLEAN  | VARIABLE { $$ = sym[$1];}  | expr OR expr { if($1==1||$3 ==1){$$=1;}else{$$=0;} }  | expr AND expr { $$ = $1 \* $3;}  | expr NAND expr { if( $1 \* $3 ==1){$$=0;}else{$$=1;}}  | expr XOR expr { if( $1 ==0 && $3==1 || $1 ==1 && $3==0){ $$=1; }else{$$=0 ;} }  | expr NOR expr { if($1==0&&$3 ==0){$$=1;}else{$$=0;} }  | expr XNOR expr { if( $1 ==0 && $3==0 || $1 ==1 && $3==1){ $$=1; }else{$$=0;} }  | NOT expr { if($2==1){ $$=0; }else{ $$=1;} }  | '(' expr ')' { $$ = $2; }  ;  %%  #include "lex.yy.c"  void yyerror(char \*s) {  fprintf(stderr, "%s\n", s);  }  int main(void) {  yyparse();  return 0;  } |

|  |  |
| --- | --- |
| Entrada | Salida |
| not not not (1 or 1) and not 1 or 0 | 0 |