UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

PAQUETE GRAFICO DIDACTICO:

PROGRAMA SIMULADOR DE CIRCUITOS PROSIC

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A N :

JAIME AGUILAR BECERRA
JUAN GUTIERREZ QUIROZ
JOSE LUIS ORTIZ VALLEJO

Director: Ing. Raymundo Hugo Rangel Gutierrez

México, D. F. 1988

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

I N D I C E

CAPITULO 1

I N T R O D U C C I O N

I.- INTRODUCCION.

Con el objeto de que, tanto estudiantes como diseñadores de hardware, en principio, puedan contar con herramientas, que les permitan realizar su trabajo de una manera fácil y económica, se pensó en la creación de un conjunto de programas de computadora que permitieran la simulación de circuitos digitales, éste conjunto de programas integra un paquete, cuyo nombre es PROSIC.

La simulación de circuitos digitales por medio de computadora, debe proporcionar al usuario facilidades tanto económicas como prácticas, esto es, no necesariamente se tiene que contar con un laboratorio de electrónica, y tener a la mano los componentes electrónicos que formarán parte de su diseño para obtener los resultados buscados.

Ahora el usuario podrá realizar su diseño, primeramente en papel y posteriormente utilizar PROSIC para simularlo, y una vez con los resultados obtenidos podrá tomar la decisión de realizar, hacer algunas modificaciones, o no montar el diseño.

PROSIC está encaminado a personas que tengan nociones de programación, que estén familiarizadas con los elementos básicos del lenguaje de programación PASCAL, y que estén familiarizados con el Diseño Lógico.

PROSIC permite al usuario simular desde circuitos muy sencillos como: conmutadores, contadores, comparadores etc., hasta circuitos más complicados como pueden ser: sumadores, multiplicadores, relojes digitales, etc. Además nos da la facilidad de trabajar por módulos, esto es, podemos programar las partes del circuito por separado y después juntarlas, de ésta forma se podrán simular los circuitos digitales que el usuario desee.

El comportamiento de cada uno de los componentes que constituye al circuito es dado por PROSIC. Además se puede hacer variar cualesquier entrada del circuito en cuestión, y observar la salida o salidas que presente dicho circuito.

Los componentes que, generalmente, se utilizan en los diseños son contenidos en èste paquete y son los siguientes:

COMPUERTAS:

- AND
- OR
- NAND
- NOR
- NOT

FLIP-FLOPS:

- DELAY
- TRIGGER
- RS
- JK
- RST

A èste tipo de componentes se les denomina componentes básicos. Cabe señalar que para simular el circuito deseado es necesario programarlo correctamente y esto dependerà de la habilidad que tenga el usuario para programar.

CAPITULO 2

P R O P U E S T A

D E L

S I S T E M A

PROPUESTA DEL PAQUETE PROSIC.

OBJETIVOS.-

- Crear un paquete de computadora que permita al usuario simular circuitos lògicos digitales, de una forma pràctica y sencilla. El paquete propuesto debe permitir al usuario programar las entradas al circuito simulado, las salidas de èste, y las caracteristicas de los componentes que integran al circuito.

  De la simulaciòn el usuario debe obtener resultados tales como :

  . Tabla de conexiones de los componentes.

  . Lista de errores detectados.

  . Tabla de Estados Lògicos que permita visualizar la simulaciòn del circuito.

  . Dibujo del diagrama del circuito.

- Que la diferencia entre el uso del paquete propuesto, y las herramientas de un laboratorio de electrònica, sea minima para el usuario.

- Los resultados obtenidos de la simulaciòn deben ayudar al usuario de tal forma, que en base a estos pueda montar su diseño o en caso de ser necesario, hacer las modificaciones pertinentes.

Para poder llevar a cabo los objetivos plantoados el paquete deberà constar de las partes siguientes :

- Programa simulador.

- Programa que tome los datos obtenidos por el programa simulador y los procese para que puedan ser tomados por el programa que hace el dibujo.

- Programa que dibuje el diagrama del circuito.

Para que el usuario pueda hacer uso del paquete deberà programar tres subrutinas, estas son:

ENTRADA : Aquí se especifica la secuencia de las entradas externas del circuito.

CIRCUITO : En èsta subrutina se programan los componentes que formarán parte del circuito.

SALIDA : Aquí el usuario deberá especificar como quiere las salidas del circuito.

Una vez que el usuario ha programado las tres subrutinas anteriores, estas serán tomadas por el programa simulador, y entonces, se llevarà a cabo la simulaciòn del circuito.

Después de haberse ejecutado el Programa Simulador el usuario deberá obtener los siguientes resultados :

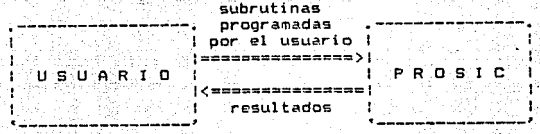
. Tabla de conexiones del circuito.

. En caso de haber ocurrido algún error, el programa debe proporcionar una lista de èl, o los, errores encontrados.

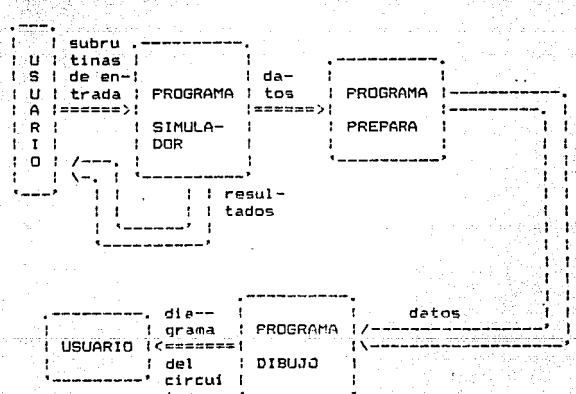
. Tabla que indique el estado lògico en el que se encuentran las entradas y salidas de cada componente ("1" ò "0").

En el siguiente esquema podemos visualizar el sistema propuesto.

I.-



II.-



CAPITULO 3

DESCRIPCION DEL PROGRAMA SIMULA

1.- INTRODUCCION.

El programa SIMULA se encarga de simular el comportamiento electrònico de los elementos del circuito, para esto el programa generarà dos tablas que contendrán información de las conexiones entre los elementos del circuito y el comportamiento lògico de estos.

La información obtenida por èste programa va a ser almacenada en los archivos DATOS y TAB, para que posteriormente sea utilizada por el programa PREPARA.

2.- DESCRIPCION DEL FUNCIONAMIENTO.

Este programa toma las subrutinas dadas por el usuario, que son: ENTRADA, SALIDA, y CIRCUITO; además pedirá los siguientes datos: número de compuertas, número de flip-flops, número de entradas, y número de conexiones, estos datos serán almacenados en el archivo VALOR, de ésta forma, si el usuario quiere repetir la simulación del circuito, el programa leerá los datos almacenados en éste archivo. Cuando un circuito se está simulando por primera vez los datos tienen que ser dados directamente por el usuario.

El programa lee los procedimientos proporcionados por el usuario y verifica que los datos sean correctos, en caso de existir algún error se activa la subrutina que detecta errores y manda un mensaje del error encontrado. En caso de no detectar error se procede a la simulación del circuito.

La simulación la hace de la siguiente forma: escoge los elementos que va a utilizar, los relaciona entre sí en base a las conexiones que existen entre ellos, ejecuta elemento por elemento, según la secuencia electrónica del circuito, y finalmente obtiene las salidas especificadas por el usuario.

3.- DESCRIPCION DE ENTRADAS Y SALIDAS.

ENTRADAS.-

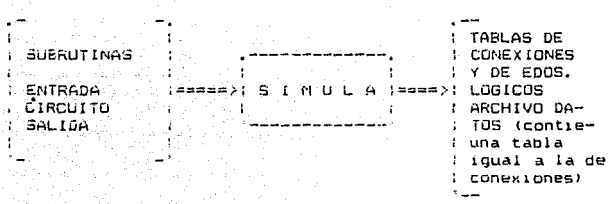
El circuito a simular es definido por el usuario mediante una subrutina llamada CIRCUITO. Además proporciona las variables correspondientes a número de compuertas, número de flip-flops, número de entradas, y número de conexiones, el programa le preguntará al usuario si quiere asignar nuevos valores a estas variables, de no ser así tomará los valores que tiene almacenados en el archivo VALOR. Además el usuario debe proporcionar las subrutinas ENTRADA y SALIDA, en donde se especifican las características de entrada y salida del circuito a simular.

SALIDAS.-

Genera una tabla que contendrá la siguiente información:

ELEMENTO	:	Elemento.
TIPO	:	Tipo.
LIGA	:	Liga (siguiente elemento a ejecutarse en la secuencia lógica del circuito).
X1	:	Entrada número 1.
X2	:	Entrada número 2.
X3	:	Entrada número 3.
X4	:	Entrada número 4 o salida número 1.
X5	:	SALIDA o salida número 2.
BANDERA	:	Indicador de inicio del circuito.

También genera una tabla que indica los estados lógicos de las entradas y salidas de cada elemento del circuito (en unos y ceros), para cada secuencia en el tiempo de ejecución del circuito. Cada columna de ésta tabla corresponderá a una conexión del circuito, los renglones nos indicarán el ciclo correspondiente.

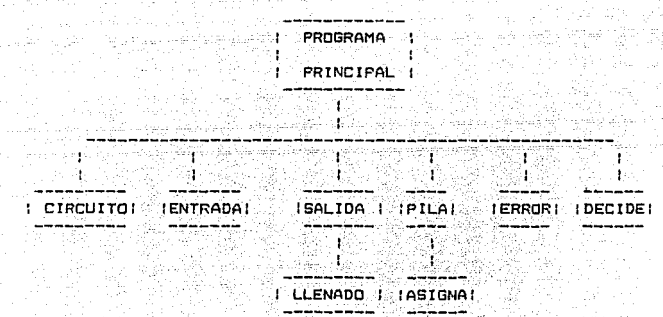


ESQUEMA QUE REPRESENTA LAS ENTRADAS Y SALIDAS DEL PROGRAMA DIBUJO

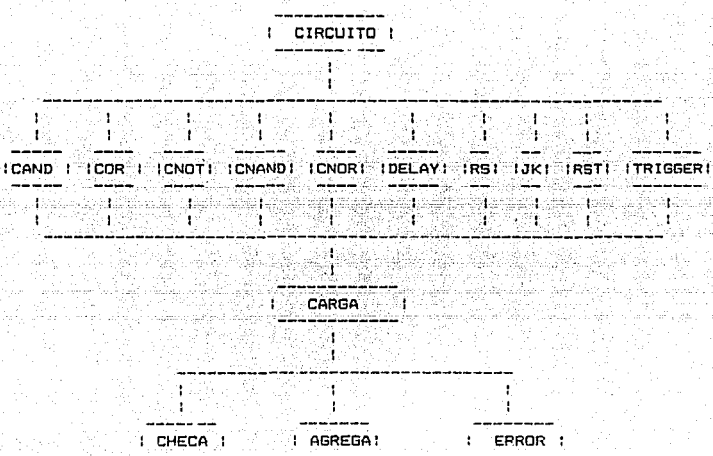
4.- DIAGRAMA DE BLOQUES DEL SISTEMA.

I.-

En el siguiente diagrama de bloques, se visualiza la forma en que se relacionan las subrutinas que forman parte del programa que realiza la simulación de los componentes del circuito.



I.1.-


CIRCUITO
CAND
COR
CNOT
CNAND
CNOR
DELAY
RS
JK
RST
TRIGGER
CARGA
CHECA
AGREGA
ERROR

5.- DESCRIPCION DE LAS SUBRUTINAS QUE COMPONEN AL PROGRAMA SIMULA.

DESCRIPCION DEL SUBPROGRAMA CARGA.

La función que desempeña el subprograma CARGA es la siguiente:

- Verifica las interconexiones de los componentes.
- Verifica si el número de entradas es el correcto y además que el número de interconexiones sea mayor que el número de entradas, lo anterior lo realiza por medio del subprograma CHECA.
- Determina si la salida de cada componente es la correcta.
- Verifica que el número de elementos sea el adecuado.
- Determina el número de componentes, el tipo de componente y el último componente reconocido. Esto lo realiza por medio del subprograma AGREGA.

DESCRIPCION DEL SUBPROGRAMA ERROR.

El subprograma error desempeña la siguiente función:

- Despliega mensajes del tipo de error que haya encontrado. (Ver apèndice de mensajes de error)

DESCRIPCION DEL SUBPROGRAMA LLENADO.

El subprograma LLENADO desempeña la siguiente función:

- Imprime el ciclo de reloj, los estados de las conexiones, y las entradas y salidas donde un "uno" equivale a que la conexión o entrada está habilitada, y un "cero" a que está deshabilitada.

DESCRIPCION DEL SUBPROGRAMA ENTRADA.

El subprograma ENTRADA desempeña las siguientes funciones:

- En él se especifican las entradas deseadas al circuito.
- El efecto del llamado de ENTRADA es cargar en el arreglo 'ALAMBRE[1:cent]', los valores externos apropiados durante cada periodo de tiempo t.
- Es responsabilidad de èste procedimiento determinar cuando la corrida está completa, transfiriendo el control a "EXIT".

DESCRIPCION DEL SUBPROGRAMA PILA.

El subprograma PILA realiza la siguiente función:

- Determina la salida de los elementos (que son los componentes básicos y compuestos), almacenàndola en el arreglo 'ALAMBRE'.

  Para realizar ésta función PILA se ayuda del subprograma ASIGNA que determina el valor del arreglo ALAMBRE[n].

DESCRIPCION DEL SUBPROGRAMA SALIDA.

El subprograma SALIDA realiza la siguiente funciòn:

- Llama a la subrutina LLENADO, cuya funciòn es llenar la tabla que nos indica los estados lògicos de los componentes del circuito, y manda imprimir el arreglo ALAMBRE.
- El usuario puede determinar las salidas que desee.

DESCRIPCION DEL SUBPROGRAMA CIRCUITO.

- En èste subprograma el usuario da las caracterìsticas del circuito a simular.
- Este subprograma puede llamar a las siguientes subrutinas:

. CAND	: Simula la compuerta AND.
. COR	: Simula la compuerta OR.
. CNOT	: Simula el inversor.
. CNAND	: Simula la compuerta NAND.
. CNOR	: Simula la compuerta NOR.
. DELAY	: Simula al flip-flop tipo D.
. RS	: Simula al flip-flop tipo RS.
. JK	: Simula al flip-flop tipo JK.
. RST	: Simula al flip-flop tipo RST.
. TRIGGER	: Simula al flip-flop tipo T.

Ver caracterìsticas en el manual del usuario.

DESCRIPCION DEL PROGRAMA PRINCIPAL.

. Llama al subprograma CIRCUITO.

. Verifica que las conexiones, componentes, y entradas no rebasen los límites definidos.

. Determina el estado de cada conexión, y cuando estan definidas las conexiones determina el tipo de componentes, el número de componente, y el último componente reconocido, esto lo hace mediante el subprograma DECIDE.

. Hace el despliegue de componentes utilizados, así como las conexiones entre ellos.

. La información del buffer, donde estan almacenados los estados de las interconexiones, es pasada al arreglo 'ALAMBRE'.

A continuación y como parte final de este capítulo presentamos un listado del programa SIMULA.

P R O G R A M A S I M U L A

```
PROGRAM PROSIC(INPUT,OUTPUT,DATOS,VALOR);
LABEL   66,77,888,9999;
VAR     S,R,PW,KW,PC,PF,LISCO,LISFF,ULCOM,ULFF,T,U,V,K,B:INTEGER;
        AUX,AUX2,IND,SV :INTEGER;
        E1,E2,E3,E4,E5,E6,E7,E8:INTEGER;
        dec,CCOMP,CFF,CENT,CALAMB:INTEGER;
        FIN,AB1,AB2,AB3,BAND:BOOLEAN;
        BANDER:INTEGER;
        DATOS:TEXT;
        valor:file of integer;
        EPAR:ARRAY[1..10000] OF INTEGER;
        DPAR:ARRAY[1..10000] OF INTEGER;
        CPAR:ARRAY[1..10000] OF INTEGER;
        BPAR:ARRAY[1..10000] OF INTEGER;
        APAR:ARRAY[1..10000] OF INTEGER;
        TIPO:ARRAY[1..10000] OF INTEGER;
        LIGA:ARRAY[1..10000] OF INTEGER;
        ALAMBRE:ARRAY[1..10000] OF BOOLEAN;
        BUFFER:ARRAY[0..1000] OF BOOLEAN;
    BAND1,SS1,T1,QQ1,RR,SS,QQ,JJ,KK,QQ2,RR1,SSS1,TT,QQ3:BOOLEAN;

PROCEDURE CAND(W,X,Y,Z,F:INTEGER);FORWARD;
PROCEDURE COR(W,X,Y,Z,F:INTEGER);FORWARD;
PROCEDURE CNOT(X,F:INTEGER);FORWARD;
PROCEDURE CNAND(W,X,Y,Z,F:INTEGER);FORWARD;
PROCEDURE CNOR(W,X,Y,Z,F:INTEGER);FORWARD;
PROCEDURE LLENADO;FORWARD;
PROCEDURE DELAY(D,Q1,QQ:INTEGER);FORWARD;
PROCEDURE TRIGGER(T,Q1,QQ:INTEGER);FORWARD;
PROCEDURE RS(R,S,Q1,QQ:INTEGER);FORWARD;
PROCEDURE JK(J,K,Q1,QQ:INTEGER);FORWARD;
PROCEDURE RST(R,S,T,Q1,QQ:INTEGER);FORWARD;
{$I CIRCUITO.PRO}
{$I ENTRADA.PRO}
{$I SALIDA.PRO}

PROCEDURE CARGA(KEY,A,B,C,D,E:INTEGER);FORWARD;
PROCEDURE ERROR(N,CLUE:INTEGER);FORWARD;
PROCEDURE CAND;
          BEGIN
              CARGA(1,W,X,Y,Z,F);
          END;
PROCEDURE COR;
          BEGIN
              CARGA(2,W,X,Y,Z,F);
           END;
PROCEDURE CNOT;
          BEGIN
```

```
            CARGA(3,X,0,0,0,F);
        END;
PROCEDURE CNAND;
        BEGIN
            CARGA(4,W,X,Y,Z,F);
        END;
PROCEDURE CNOR;
        BEGIN
            CARGA(5,W,X,Y,Z,F);
        END;
PROCEDURE DELAY;
        BEGIN
            CARGA(6,D,0,0,Q1,Q0);
        END;
PROCEDURE TRIGGER;
        BEGIN
            CARGA(7,T,0,0,Q1,Q0);
        END;
PROCEDURE RS;
        BEGIN
            CARGA(8,R,S,0,Q1,Q0);
        END;
PROCEDURE JK;
        BEGIN
            CARGA(9,J,K,0,Q1,Q0);
        END;
PROCEDURE RST;
        BEGIN
        CARGA(10,R,S,T,Q1,Q0);
        END;
PROCEDURE AGREGA(ELEMENT:INTEGER;VAR LIST,COLA:INTEGER);
        BEGIN
            IF LIST=0 THEN
              LIST:=ELEMENT
            ELSE
                LIGA[COLA]:=ELEMENT;
             LIGA[ELEMENT]:=0;
             COLA:=ELEMENT;
        END(* AGREGA *);

PROCEDURE PILA(XB,VB:BOOLEAN);
        LABEL 55;
        VAR
           I:INTEGER;
           F:BOOLEAN;
           PROCEDURE ASIGNA;
                  BEGIN
                      IF I<>0 THEN
                         BEGIN
                             IF ALAMBRE[I] = XB THEN
                                BEGIN
                                   F:=VB;
                                   BAND:=TRUE;
                                END;
                          END;
```

```
                    END(* asigna *);
               BEGIN
                    F:=NOT(VB);
                    I:=APAR[U];ASIGNA;IF BAND THEN GOTO 55;
                    I:=BPAR[U];ASIGNA;IF BAND THEN GOTO 55;
                    I:=CPAR[U];ASIGNA;IF BAND THEN GOTO 55;
                    I:=DPAR[U];ASIGNA;IF BAND THEN GOTO 55;
                    55: ALAMBRE[EPAR[U]]:=F;BAND:=FALSE;
                END(* pila *);

PROCEDURE CARGA;
          PROCEDURE CHECA(X:INTEGER;OUT:BOOLEAN);
                    BEGIN
                         IF X>PW THEN
                           BEGIN
                             PW:=X;
                             IF PW>CALAMB THEN ERROR(1,PC);
                           END;
                         IF (X>0) AND (OUT) THEN
                           BEGIN
                                IF X<=CENT THEN ERROR(2,X);
                                IF ALAMBRE[X] THEN ERROR(3,X)
                                   ELSE
                                        ALAMBRE[X]:=TRUE;
                           END;
                    END(* checa *);
               BEGIN
                    PC:=PC+1;
                    IF PC>CCOMP THEN ERROR(4,KEY);
                    CHECA(A,FALSE);
                    CHECA(B,FALSE);CHECA(C,FALSE);CHECA(D,KEY>5);
                    CHECA(E,TRUE);
                    TIPO[PC]:=KEY;
                    IF KEY<=5 THEN AGREGA(PC,LISCO,ULCOM)
                       ELSE
                            BEGIN
                                 AGREGA(PC,LISFF,ULFF);
                                 PF:=PF+1;
                                 IF PF>CFF THEN ERROR(5,KEY);
                                 BUFFER[PF]:=FALSE;
                            END;
                         APAR[PC]:=A;
                         BPAR[PC]:=B;
                         CPAR[PC]:=C;
                         DPAR[PC]:=D;
                         EPAR[PC]:=E;
               END(* carga *);

PROCEDURE ERROR;
BEGIN
CASE N OF
     1:BEGIN
       WRITELN('**** ERROR : 1');
       WRITELN('El componente numero : ',CLUE,' tiene ',
       'parametros mayores que el num. de conexiones',
```

```
    ' CALAMB = ',CALAMB);
   END;
 2:BEGIN
   WRITELN('**** ERROR : 2');
   WRITELN('La entrada externa : ', CLUE,' aparece tam',
           'bien como   salida');
   END;
 3:BEGIN
   WRITELN('**** ERROR : 3');
   WRITELN('El alambre : ',CLUE,' aparece mas de una vez',
           ' como salida de un componente basico');
   END;
 4:BEGIN
   WRITELN('**** ERROR : 4');
   WRITELN('Se estan utilizando mas COMPONENTES BASICOS',
           '  de los indicados al',
           ' inicio del programa : ',CCOMP);
   WRITELN(' El  extra detectado es de tipo : ',CLUE);
   END;
 5:BEGIN
   WRITELN('**** ERROR : 5');
   WRITELN('Se estan utilizando mas FLIP-FLOPS de los indicados al',
           ' inicio del programa : ',CFF);
   WRITELN('El  extra detectado es de tipo : ',CLUE);
   END;
 6:BEGIN
   WRITELN('**** ERROR : 6');
   WRITELN('El num. de conexiones utilizadas : ',CLUE,
           ' es menor que las especificadas : CALAMB = ',CALAMB);
   END;
 7:BEGIN
   WRITELN('**** ERROR : 7');
   WRITELN('El numero de COMPONENTES BASICOS  utilizados  : ',CLUE);
   WRITELN('es menor que el especificado : CCOMP = ',CCOMP);
   END;
 8:BEGIN
   WRITELN('**** ERROR : 8');
   WRITELN('El numero de FLIP-FLOPS utilizados : ',CLUE);
   WRITELN('es menor que el especificado : CFF = ',CFF);
   END;
 9:BEGIN
   WRITELN('**** ERROR : 9');
   WRITELN('El alambre : ',CLUE,' no  es salida ',
           ' de un componente basico');
   END;
10:BEGIN
   WRITELN('**** ERROR : 10');
   WRITELN('No es valido el sucesor en la secuencia  ',
    ' computacional, para el componente basico en la posicion ',
           CLUE);
   END;
11:BEGIN
   WRITELN('**** ERROR : 11');
   WRITELN('El elemento RS en la linea ',CLUE,' tiene ',
           ' dos UNOS en la entrada');
```

```
      END;
    12:BEGIN
       WRITELN('**** ERROR : 12');
       WRITELN('En el elemento RST en la linea ',CLUE,' tie',
            'ne dos UNOS en la entrada');
       END;
  END(* case *);
            FIN:=TRUE;
             END(* error *);

PROCEDURE LLENADO;
         VAR R:INTEGER;
         BEGIN
              IF T=0 THEN
                 WRITELN(' ','TIEMPO              ALAMBRADO');
                 WRITE(' ':2,T:3);
                 S:=0;
                 FOR R:=1 TO CALAMB DO
                     IF ALAMBRE[R] THEN
                        WRITE(' ':3,'1')
                      ELSE
                        WRITE(' ':3,'0');
                      S:=S+1;
                      IF S=10 THEN
                         BEGIN
                              WRITE(' ':29,'  ');
                              S:=0;
                          END;
                   WRITELN;
          END(* llenado *);

PROCEDURE DECIDE(K:INTEGER);
         BEGIN
              BAND1:=FALSE;
              IF K<>0 THEN
                 BEGIN
                      AB1:=ALAMBRE[K];
                      IF NOT(AB1) THEN
                         BAND1:=TRUE;
                 END;
         END(* decide *);




(****** P R O G R A M A       P R I N C I P A L ******)

BEGIN
assign(valor,'valor.pro');
CLRSCR;
WRITELN;WRITELN;WRITELN;WRITELN;WRITELN;WRITELN;WRITELN;
writeln ('Deseas cambiar la constantes almacenadas en la',
        ' simulacion anterior ??');
WRITELN;
```

```
write( '          SI: teclea [1],   NO : teclea [0]  ==>  ');
READ(DEC);
CLRSCR;
if dec=1 then
begin
rewrite(valor);
writeln;writeln;writeln;
WRITELN(' ':5,'DAME LOS SIGUIENTES VALORES :');
writeln;WRITELN;
write(' ':5,'CCOMP (Numero de componentes)  ==> ');
READ(CCOMP);
WRITELN;WRITELN;
WRITE(' ':5,'CFF   (Numero de Flip-Flops )  ==> ');
READ(CFF);
WRITELN;WRITELN;
WRITE(' ':5,'CENT  (Numero de entradas   )  ==> ');
READ(CENT);
WRITELN;WRITELN;
WRITE(' ':5,'CALAMB (Numero de conexiones)  ==> ');
READ(CALAMB);
WRITELN;WRITELN;
write(valor,ccomp,cff,cent,calamb);
CLRSCR;
end;
reset(valor);
  read(valor,ccomp,cff,cent,calamb);
     ULFF:=0;IND:=5;ULCOM:=0;
     FOR KW:=1 TO CALAMB DO
         ALAMBRE[KW]:=FALSE;
     PW:=0;PC:=0;PF:=0;LISCO:=0;LISFF:=0;FIN:=FALSE;BAND:=FALSE;
     CIRCUITO;IF (FIN) THEN GOTO 9999;
     IF PW<CALAMB THEN ERROR(6,PW); IF (FIN) THEN GOTO 9999;
     IF PC<CCOMP THEN ERROR(7,PC); IF (FIN) THEN GOTO 9999;
     IF PF<CFF THEN ERROR(8,PF); IF (FIN) THEN GOTO 9999;
     FOR KW:=CENT+1 TO CALAMB DO
    IF NOT(ALAMBRE[KW]) THEN ERROR(9,KW); IF (FIN) THEN GOTO 9999;

     BEGIN (* DEL ANIDADO *)
          FOR R:=1 TO CENT DO
              ALAMBRE[R]:=TRUE;
          FOR R:=CENT+1 TO CALAMB DO
              ALAMBRE[R]:=FALSE;
          R:=LISFF;
          WHILE R<>0 DO
                BEGIN
                     S:=DPAR[R];
                     IF S<>0 THEN
                        ALAMBRE[S]:=TRUE;
                     S:=EPAR[R];
                     IF S<>0 THEN
                        ALAMBRE[S]:=TRUE;
                     R:=LIGA[R];
                END;
           U:=CCOMP-CFF;
           V:=0;
```

```
        R:=LISCO;
        S:=ULCOM;

        LISCO:=0;
        WHILE R<>0 DO
          BEGIN
            PC:=LIGA[R];
            K:=APAR[R];DECIDE(K);IF BAND1 THEN GOTO 888;
            K:=BPAR[R];DECIDE(K);IF BAND1 THEN GOTO 888;
            K:=CPAR[R];DECIDE(K);IF BAND1 THEN GOTO 888;
            K:=DPAR[R];DECIDE(K);IF BAND1 THEN GOTO 888;
            AGREGA(R,LISCO,ULCOM);
            ALAMBRE[EPAR[R]]:=TRUE;
            U:=U-1;
            V:=0;
            GOTO 77;
      888:  AGREGA(R,R,S);
            V:=V+1;
            IF U=V THEN ERROR(10,ULCOM);IF (FIN) THEN GOTO 9999;
       77:  R:=PC;
      END(* del While*);
       LIGA[ULCOM]:=LISFF;
    END(* del Anidado *);


BEGIN (* Del despliegue de Componentes *)
 WRITELN;
 ASSIGN(DATOS,'DATOS.YOO');
 REWRITE(DATOS);
 WRITELN(' ':12,'TIPO      LIGA                  CONEXIONES');
  FOR R:=1 TO CCOMP DO
    BEGIN
      AUX:=TIPO[R];
        CASE AUX OF
              1: BEGIN
                  WRITE(R:5,' ':5,'AND');
                  WRITE(DATOS,R:4,AUX:4);
                 END;
              2: BEGIN
                  WRITE(R:5,' ':5,' OR');
                  WRITE(DATOS,R:4,AUX:4);
                 END;
              3: BEGIN
                  WRITE(R:5,' ':5,'NOT');
                  WRITE(DATOS,R:4,AUX:4);
                 END;
              4: BEGIN
                  WRITE(R:5,' ':5,'NAND');
                  WRITE(DATOS,R:4,AUX:4);
                 END;
              5: BEGIN
                  WRITE(R:5,' ':5,'NOR');
                  WRITE(DATOS,R:4,AUX:4);
                 END;
              6: BEGIN
```

```
                WRITE(R:5,' ':5,'  D');
                WRITE(DATOS,R:4,AUX:4);
              END;
            7: BEGIN
                WRITE(R:5,' ':5,'  T');
                WRITE(DATOS,R:4,AUX:4);
              END;
            8: BEGIN
                WRITE(R:5,' ':5,' RS');
                WRITE(DATOS,R:4,AUX:4);
              END;
            9: BEGIN
                WRITE(R:5,' ':5,' JK');
                WRITE(DATOS,R:4,AUX:4);
              END;
           10: BEGIN
                WRITE(R:5,' ':5,'RST');
                WRITE(DATOS,R:4,AUX:4);
              END;
         END(* del Case *);

       S:=LIGA[R];WRITE(DATOS,S:4);WRITE('  ':IND,S:3);
       S:=APAR[R];WRITE(DATOS,S:4);WRITE('  ':IND,S:3);
       S:=BPAR[R];WRITE(DATOS,S:4);WRITE('  ':IND,S:3);
       S:=CPAR[R];WRITE(DATOS,S:4);WRITE('  ':IND,S:3);
       S:=DPAR[R];WRITE(DATOS,S:4);WRITE('  ':IND,S:3);
       S:=EPAR[R];WRITE(DATOS,S:4);WRITE('  ':IND,S:3);
       IF R=LISCO THEN
        BEGIN
          BANDER:=1;
          WRITE(DATOS,BANDER:4);WRITE('  ':5,' INICIO');
        END
        ELSE
        BEGIN
          BANDER:=0;
          WRITE(DATOS,BANDER:4);
       END;
       WRITELN;WRITELN(DATOS);
       END(* for *);
       CLOSE(DATOS);
       WRITELN;WRITELN;WRITELN;WRITELN;
END(* Despliegue de componentes *);

BEGIN (* De la ejecucion *)
      T:=0;
   66: U:=LISFF;
      B:=0;
      WHILE U<>0 DO
            BEGIN
                B:=B+1;
                K:=DPAR[U];
                IF K<>0 THEN
                ALAMBRE[K]:=BUFFER[B];
                K:=EPAR[U];
                IF K<>0 THEN
```

```
                ALAMBRE[K]:=NOT(BUFFER[B]);
              U:=LIGA[U];
        END;
ENTRADA(E1,E2,E3,E4,E5,E6,E7,E8);
 IF (FIN)=TRUE THEN GOTO 9999;
U:=LISCO;
B:=0;
WHILE U<>0 DO
  BEGIN
   AUX2:=TIPO[U];
  CASE AUX2 OF
       1: PILA(FALSE,FALSE);
       2: PILA(TRUE,TRUE);
       3: ALAMBRE[EPAR[U]]:=NOT(ALAMBRE[APAR[U]]);
       4: PILA(FALSE,TRUE);
       5: PILA(TRUE,FALSE);
       6: BEGIN
            B:=B+1;
            BUFFER[B]:=ALAMBRE[APAR[U]];
          END;
       7: BEGIN
            B:=B+1;
            T1:=ALAMBRE[APAR[U]];
            QQ1:=BUFFER[B];
            BUFFER[B]:=NOT(T1=QQ1);
          END;
       8: BEGIN
            B:=B+1;
            RR:=ALAMBRE[APAR[U]];
            SS:=ALAMBRE[BPAR[U]];
            QQ:=BUFFER[B];
            IF RR AND SS THEN ERROR(11,U);
             BUFFER[B]:=(QQ) AND NOT(RR) OR (SS);
          END;
       9: BEGIN
            B:=B+1;
            JJ:=ALAMBRE[APAR[U]];
            KK:=ALAMBRE[BPAR[U]];
            QQ2:=BUFFER[B];
            BUFFER[B]:=(QQ2) AND NOT(KK) OR
            NOT(QQ2) AND (JJ);
          END;
      10: BEGIN
            B:=B+1;
            RR1:=ALAMBRE[APAR[U]];
            SS1:=ALAMBRE[BPAR[U]];
            TT:=ALAMBRE[CPAR[U]];
            QQ3:=BUFFER[B];
            IF (RR1) AND (SS1) OR (SS1) AND (TT)
               OR (RR1) AND (TT) THEN ERROR(12,U);
               BUFFER[B]:=(SS1) OR (TT) AND (NOT(QQ3))
               OR (QQ3) AND (NOT(RR1)) AND (NOT(TT));
          END;
END(* CASE *);
 U:=LIGA[U];
```

```
        END(* while *);
         SALIDA(SV);
         T:=T+1;
       GOTO 66;
    END(* De la ejecucion *);
  9999: WRITELN;WRITELN;
  WRITELN(' ':25,'FIN DE SIMULA ********');
END.
```

CAPITULO 4

D E S C R I P C I O N D E L P R O G R A M A

P R E P A R A

1.- DESCRIPCIÓN DEL PROGRAMA PREPARA.

INTRODUCCION.

El programa PREPARA toma los datos generados por el programa SIMULA y los procesa y ordena para que sirvan como entrada del programa DIBUJO, esto es, el programa que estamos describiendo, hace el papel de interface entre el programa SIMULA (que simula el circuito) y el programa DIBUJO (que dibuja el diagrama del circuito simulado).

2.- DESCRIPCION DEL FUNCIONAMIENTO.

El funcionamiento del programa PREPARA, en forma general es el siguiente: lee el archivo DATOS.YOO que es generado por el programa SIMULA. Este archivo contiene los siguientes datos: elemento, tipo, liga, las variables de las entradas y salidas del circuito y una variable que sirve como bandera, la función de cada una de èstas variables es explicada en el capítulo en donde se describe al programa SIMULA, una vez que se ha leído el archivo DATOS.YOO el programa se encarga de ordenar la información con respecto a la variable liga y entonces genera el archivo TAB.DOS que contiene la información que fuè ordenada.

Después de haber realizado el proceso anterior, se leen en el archivo TAB.DOS las variables liga y tipo, para cada registro, se entiende que en cada registro se almacena la información correspondiente a cada uno de los componentes electrónicos utilizados en la simulación del circuito, y se realiza el siguiente proceso: la información del primer registro es comparada con la información leída en el registro que le sigue en el archivo. El propósito de èsta comparación es conocer cuáles son las entradas y/o salidas que tienen en común los componentes utilizados, de ésta relación es de donde se obtiene la estructura básica del dibujo del circuito.

Una vez que se ha detectado la relación entre entradas y salidas de los componentes, se procesará èstà información para determinar si el componente a dibujar es: el primero que se dibuja, se dibujará enfrente de otro que ya ha sido dibujado o abajo de otro que ya fuè dibujado, la información que determinará èsta estructura será almacenada en el archivo DIB.DOS, para que de ahí sea tomada por el programa DIBUJO.

3 .- DESCRIPCION DE ENTRADAS Y SALIDAS.

ENTRADAS.-

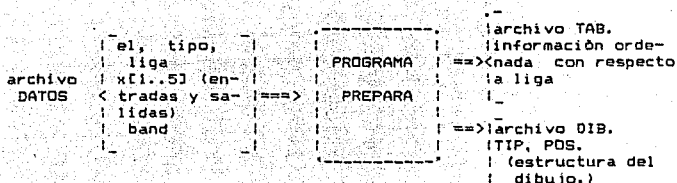
Las variables de entrada serán: elemento, tipo, liga, cuatro entradas y una salida, en el caso de las compuertas, y variables para representar hasta tres entradas y dos salidas en el caso de los flip-flops. Los datos mencionados anteriormente serán leídos del archivo DATOS.

SALIDAS.-

La información que se obtendrá después de haberse ejecutado el programa PREPARA es la misma información tomada del archivo DATOS pero ordenada con respecto a la variable liga, ésta información será almacenada en el archivo TAB.

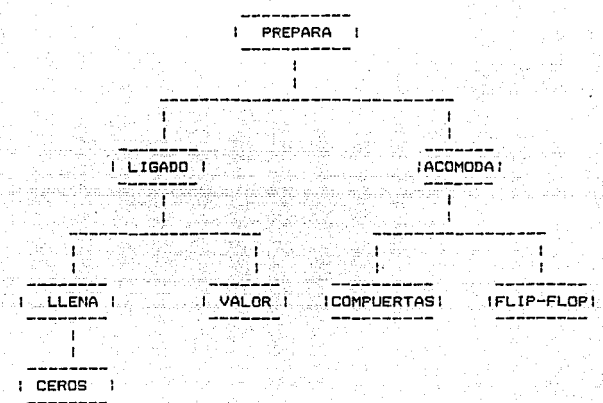
También será generado un archivo llamado DIB que contendrá la siguiente información: TIPO y POS donde TIPO es el tipo de componente utilizado y POS es la clave que determinará en que parte de la pantalla se dibujará el símbolo del elemento.

En el siguiente esquema se representa gráficamente lo que se describió en los párrafos anteriores:



ESQUEMA QUE REPRESENTA LAS ENTRADAS Y SALIDAS DEL PROGRAMA PREPARA.

4.- DIAGRAMA DE BLOQUES DEL PROGRAMA PREPARA.



DESCRIPCION DE LA SUBRUTINAS QUE COMPONEN AL

PROGRAMA PREPARA

DESCRIPCION DE LA SUBRUTINA LIGADO.

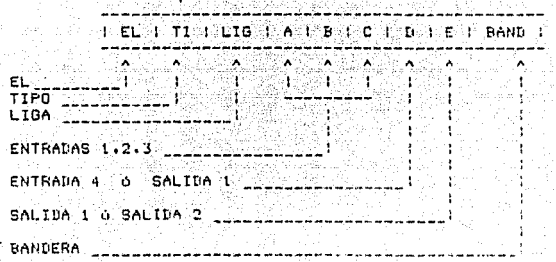
OBJETIVO.

Tomar la información obtenida después de haber ejecutado el programa SIMULA, y ordenarla en base a la variable LIGA.

- FUNCION.

La función a desarrollar por ésta subrutina dentro del proceso de simulación del circuito es la siguiente:

Tomar la tabla generada por el programa SIMULA, que contiene la información representada en el siguiente esquema:



En el esquema de arriba tenemos la representación de un renglón de la tabla generada por SIMULA y que está almacenada en el archivo DATOS.

En base a la información antes mencionada, la subrutina que estamos describiendo, debe generar otra tabla que debe contener la información anterior, pero ordenada con respecto a liga. Esto es, vamos a tener de entrada una información como la siguiente:

EL	TI	LIGA	CONEXIONES					BAND
1	7	2	1	0	0	2	0	0
2	7	3	2	0	0	3	0	0
3	7	4	7	0	0	4	0	0
4	7	5	8	9	0	5	0	0
5	7	0	9	0	0	6	0	0
6	1	7	2	3	0	0	7	1
7	1	8	7	4	0	0	8	0
8	1	1	5	9	0	0	9	0

La tabla anterior contiene la información que se obtuvo después de haber ejecutado el programa SIMULA, para simular un circuito contador, y que está almacenada en el archivo DATOS.

La salida que obtendremos después de ejecutar el programa PREPARA es la siguiente :

EL	TI	LIGA		CONEXIONES				BAND
6	1	7	2	3	0	0	7	1
7	1	8	7	4	0	0	8	0
8	1	1	5	8	0	0	9	0
1	7	2	1	0	0	2	0	0
2	7	3	2	0	0	3	0	0
3	7	4	7	0	0	4	0	0
4	7	5	8	0	0	5	0	0
5	7	0	9	0	0	6	0	0

La tabla anterior contiene la misma información que está almacenada en el archivo DATOS, pero está ordenada con respecto a liga, y se encuentra almacenada en el archivo TAB. Como se puede ver el primer registro del archivo TAB tiene LIGA=7 y el segundo elemento es el número 7, el segundo elemento a su vez tiene LIGA=8 y el tercer elemento es el número 8, ésto es, el elemento siguiente siempre debe ser igual a la liga del elemento actual, si estamos en el último renglón del archivo la liga debe ser igual a cero. Lo anterior es explicado con más detalle en el capítulo donde se describe al programa SIMULA.

DESARROLLO.

El procedimiento a seguir para poder llevar a cabo lo anteriormente expuesto es el siguiente:

Primero se leerá la información almacenada en el archivo DATOS hasta encontrar aquél registro en el que la variable BAND sea igual a 1, esto nos indicará que la información leida en éste registro es la correspondiente al componente con el que se inició la simulación, y por lo tanto será con éste componente con el que se inicie la ordenación con respecto a liga.

El proceso de ordenación se hará de la siguiente manera:

1.- Se almacena el valor de liga del primer componente encontrado (aquel que tiene BAND=1), en la variable AUX .

2.- Se almacena en el archivo TAB.

3.- Leemos la información de los demás componentes hasta encontrar aquél que tenga EL=AUX, y entonces tendremos el segundo elemento ordenado, y se almacena la liga de éste elemento en la variable AUX.

4.- Repetimos el proceso a partir del paso 2 hasta terminar de leer el archivo.

Lo anterior lo podemos expresar, también en el siguiente algoritmo:

```
inicio
    abre archivo para escritura (TAB)
    abre archivo para lectura (DATOS)
    mientras no termines de leer el archivo DATOS has:
      inicio(1)
        lee renglón de el archivo datos
        si band=1 haz lo siguiente:
        inicio(2)
          almacena los valores del reg. leído en
          variables temporales.
        termina(2);
        escribe en el archivo tab  el registro para el cual
        band = 1;
```

```
        mientras no termines de leer el archivo DATOS
        realiza lo siguiente:
         inicio(3)

           lee registro en el archivo DATOS;

           si EL=AUX  entonces

            almacena reg. en el archivo TAB.

            si esto no ocurriò

            ponte en el primer registro del archivo DATOS

      termina(3);

  termina(2);

fin de LIGADO.
```

ALGORITMO DE LA SUBRUTINA LIGADO

La subrutina liqado incluye tres subrutinas pequeñas, la funciòn de èstas subrutinas, es simplemente preparar la informaciòn tomada del archivo datos para que pueda ser procesada por el subprograma LIGADO.

Estas subrutinas son:

CEROS : Convierte espacios blancos en ceros.

LLENA : Utiliza la suorutina CEROS y llena los campos requeridos con ceros.

VALOR : Exiten algunas variables que son de tipo caracter y èsta subrutina las convierte a nùmeros.

DESCRIPCION DE LA SUBRUTINA ACOMODA.

OBJETIVO.

Generar la estructura del dibujo, esto es, determinar el lugar en el que serà dibujado cada elemento.

FUNCION.-

La funciòn a desarrollar por èsta subrutina es la siguiente:

Toma la tabla generada por la subrutina LIGADO y en base a èsta genera el archivo DIB que a su vez serà tomado por el programa DIBUJO. El archivo DIB tendrà la siguiente informaciòn :

TIPO	POS

donde : TIPO = Tipo del elemento.

Pos = Clave del lugar donde se dibujarà el componente.

En el programa SIMULA se ha explicado lo referente a la variable tipo, ahora vamos a hablar de la variable POS, èsta variable podrà tener tres valores clave que son :

POS = 1 : Serà el primer componente que se dibuje.

POS = 2 : Se dibujarà abajo del que se dibujò anteriormente.

POS = 3 : Se dibujarà enfrente del òltimo componente dibujado.

DESARROLLO.

Para determinar el valor que deberá tener la variable POS se realiza lo siguiente :

1 .- Se leè la información contenida en el archivo TAB.

2 .- Se toma el primer registro y se comparan las variables de entrada del componente ahí almacenado, con las correspondientes de los demás, de tal forma que si se encuentra algún componente que tenga alguna entrada común al componente actual, el componente encontrado se dibujará abajo del que tomamos como referencia de comparación.

3 .- Si no se encuentran entradas comunes, entonces se comparan las entradas del componente actual con las salidas de los demás componentes, si se encuentra alguno cuya salida sea igual a alguna de las entradas del componente actual, entonces el componente encontrado se dibujará enfrente del componente actual.

4 .- Si no ha ocurrido alguno de los dos casos anteriores entonces se compara la salida del componente actual con las entradas de los demás componentes, si se encuentra alguno que tenga alguna entrada igual a la salida del componente actual, entonces el componente encontrado será dibujado enfrente de éste.

Una vez que se comparò el registro actual con los demàs, ya no es tomado en cuenta para las siguientes operaciones, esto es, en la primera operación se compara el primer registro contra todos , pero para la segunda ya no es tomado en cuenta y se comparará el segundo registro contra los restantes, de tal forma que en la última operación solo se tomarán en cuenta al penúltimo y último registros. De ésta forma todos los registro se habràn comparado entre si.

La subrutina acomoda se compone, a su vez de dos subrutinas más que son COMPUERTAS y FLIPFLOP, éstos subprogramas realizan una función similar, esto es, realizan los cuatro pasos del proceso descrito arriba, pero COMPUERTAS lo hace para el conjunto de componentes que son compuertas y FLIPFLOP lo hace cuando el componente es un flipflop.

La función realizada por la subrutina acomoda se puede resumir en el siguiente algoritmo:

```
inicia
    abre el archivo TAB para leer datos
    abre archivo DIB para escribir datos
    mientras no termines de leer TAB haz lo siguiente:
     inicia(1)
          lee el primer reg. de TAB;
          pos=1;
          escribe en DIB las variables 'tip' y 'pos'
          repite
               asigna los valores leídos a variables
               temporales;
               lee el siguiente registro de TAB;
               si tipo<6 entonces
                llama a la subrutina COMPUERTAS
               de no ser así:
                llama a la subrutina FLIPFLOP;
          hasta que se termine TAB;
    termina(1)
fin de ACOMODA.
```

ALGORITMO DE LA SUBRUTINA ACOMODA.

A continuación, y como parte final de éste capítulo presentamos un listado de el programa prepara.

```
PROGRAM PREPARA(INPUT,OUTPUT,DATOS,TAB,DIB);
TYPE TABLA=RECORD
                  ELEM:INTEGER;
                  TIPO:INTEGER;
                  LIGA:INTEGER;
                  W    :ARRAY[1..5] OF INTEGER;
                  BAN :INTEGER;
             END;

type car=string[4];

VAR
   TAB: FILE OF TABLA;
   VECTOR:TABLA;
   DATOS:TEXT;

PROCEDURE LIGADO;

 VAR
   EL,LIG,A,B,C,D,E,BAND,ti:string[4];
   AUX,GH,RE,BA:INTEGER;
   ELE,TIP,LI,AA,BB,CC,DD,EE:integer;

procedure ceros(var x:car);
var i:integer;
 begin
      for i:=1 to 4 do
         begin
              if x[i]=' ' then
                x[i]:='0';
         end;
end;(*ceros*)

procedure llena;
begin
     ceros(el);ceros(ti);ceros(lig);
     ceros(a);ceros(b);ceros(c);ceros(d);
     ceros(e)  eros(band);
end;

PROCEDURE VALOR;
BEGIN
     VAL(EL,ELE,RE);
     val(ti,tip,re);
     VAL(LIG,LI,RE);
     VAL(A,AA,RE);
     VAL(B,BB,RE);
     VAL(C,CC,RE);
     VAL(D,DD,RE);
```

```
     VAL(E,EE,RE);
     VAL(BAND,BA,RE);
END;

BEGIN
     ASSIGN(TAB,'TAB.DOS');
     ASSIGN(DATOS,'DATOS.YOO');
     REWRITE(TAB);
     RESET(DATOS);
     BA:=0;
     WHILE NOT EOF(DATOS) DO
     BEGIN
          READLN(DATOS,EL,TI,LIG,A,B,C,D,E,BAND);
          llena;
          valor;
          writeln(ele,tip,li,aa,bb,cc,dd,ee,ba);
          IF BA=1 THEN
            BEGIN
                 AUX:=ELE;
                 WITH VECTOR DO
                 BEGIN
                      ELEM:=ELE;
                      TIPO:=TIP;
                      LIGA:=LI;
                      W[1]:=AA;
                      W[2]:=BB;
                      W[3]:=CC;
                      W[4]:=DD;
                      W[5]:=EE;
                      BAN:=BA;
                      WRITELN;
                   END;
                   BA:=0;
              END;
        END;(*DEL WHILE*)
        RESET(DATOS);
        with vector do
        begin
             AUX:=LIGA;
             write(tab,vector);
        end;
       WHILE NOT EOF(DATOS) AND (AUX<>0) DO
        BEGIN
             READLN(DATOS,EL,TI,LIG,A,B,C,D,E,BAND);
             llena;
             VALOR;
             while ele<>aux do
             begin
                  readln(datos,el,ti,lig,a,b,c,d,e,band);
                  llena;
                 valor
             end;
             IF ELE=AUX THEN
                BEGIN
```

```
                        WITH VECTOR DO
                            BEGIN
                                ELEM:=ELE;
                                TIPO:=TIP;
                                LIGA:=LI;
                                W[1]:=AA;
                                W[2]:=BB;
                                W[3]:=CC;
                                W[4]:=DD;
                                W[5]:=EE;
                              END;
                             WRITE(TAB,VECTOR);
                         reset(datos);
                      AUX:=LI;
                    end
                     else
                        RESET(DATOS);
          END;(*DEL WHILE*)
     close(datos);close(tab);
END;(*LIGADO*)


PROCEDURE ACOMODA;
VAR
   F,I,J,K,L,L1,FLAG: INTEGER;
    TI,LI,BAN,EL,POS: INTEGER;
              X:ARRAY[1..5] OF INTEGER;
       DIB,TAB:FILE OF INTEGER;

PROCEDURE COMPUERTAS;
LABEL SAL;
BEGIN
      WITH VECTOR DO
          BEGIN
               FLAG:=0;L:=0;L1:=0;
               FOR K:=1 TO 4 DO
                  BEGIN
                       FOR J:=1 TO 4 DO
                          BEGIN
                               IF W[K]=X[J] THEN
                                 BEGIN
                                   IF (W[K]<>0) AND (X[J]<>0) AND
(FLAG=0) THEN
                                          BEGIN
                                               POS:=3; (abajo)
                                               WRITE(DIB,TI,POS);
                                               FLAG:=1;
                                        END;
                                        if flag=1 then
                                         goto sal;
                                  END
                                ELSE
                                begin
                                    IF   (X[J]=W[5])    AND(FLAG=0)
THEN
```

```
                                        BEGIN
                                             POS:=2;  {enfrente}
                                             WRITE(DIB,TI,POS);
                                              FLAG:=1;
                                        END;
                                   if FLAG = 1 then goto sal;
                                   end;
                         END;
                         end;
                         for J:=1 to 4 do
                         FOR J:=1 TO 4 DO
                            BEGIN
                                 IF (W[J]=X[5]) AND (W[J]<>0) AND (L=0)
THEN
                                    BEGIN
                                         POS:=2;  {enfrente}
                                         WRITE(DIB,TI,POS);
                                          L:=1;
                                          goto sal;
                                    END
                                 ELSE
                                     BEGIN
                                          IF (L=0) AND (L1=0) THEN
                                            BEGIN
                                                 POS:=2; {enfrente}
                                                 WRITE(DIB,TI,POS);
                                                 L1:=1;
                                                 goto sal;
                                            END;
                                     END;
                            END;
                    END;
   sal: writeln(' ');

END;{*COMPUERTAS*}


PROCEDURE FLIPFLOP;
LABEL SAL;
BEGIN
      WITH VECTOR DO
          BEGIN
               FLAG:=0;L:=0;L1:=0;
               FOR K:=1 TO 3 DO
                  BEGIN
                       FOR J:=1 TO 4 DO
                          BEGIN
                               IF W[K]=X[J] THEN
                                 BEGIN
                                 if ((j=4) and (ti<5)) or (j<4) then
                                   IF   (W[K]<>0) AND (X[J]<>0) AND
(FLAG=0) THEN
                                          BEGIN
                                               POS:=3; {abajo}
                                               WRITE(DIB,TI,POS);
```

```
                                        FLAG:=1;
                                    END;
                                    if flag=1 then
                                     goto sal;
                                END
                            ELSE
                              IF   (X[J]=W[4]) AND(FLAG=0)   AND
(W[4]<>0) THEN
                                  BEGIN
                                      POS:=2; {enfrente}
                                      WRITE(DIB,TI,POS);
                                       FLAG:=1;
                                       goto sal;
                                  END;
                              IF (X[J]=W[5]) AND (W[5]<>0) AND
(FLAG=0) THEN
                                  BEGIN
                                     POS:=2; {enfrente}
                                     WRITE(DIB,TI,POS);
                                     FLAG:=1;
                                     GOTO SAL;
                                  END
                        END;
                                          IF    (L=0)    AND
(L1=0) THEN
                                           BEGIN
                                            POS:=3;{abajo}
                                            WRITE(DIB,TI,POS);
                                            L1:=1;
                                            goto sal;
                                           END;
                    END;
          END;
    sal: writeln(' ');
END;{*FLIPFLOP*}


BEGIN
     ASSIGN(TAB,'TAB.DOS');
     RESET(TAB);
     ASSIGN(DIB,'DIB.DOS');
     REWRITE(DIB);
     WHILE NOT EOF(TAB) DO
     BEGIN
          READ(TAB,EL,TI,LI,X[1],X[2],X[3],X[4],X[5],BAN);
          POS:=1;  {inicio}
          WRITE(DIB,TI,POS);
          repeat
                    WITH VECTOR DO
                    BEGIN
                         ELEM:=EL;
                         TIPO:=TI;
                         LIGA:=LI;
                         FOR I:=1 TO 5 DO
                         W[I]:=X[I];
```

```
                    END;
                    READ(TAB,EL,TI,LI,X[1],X[2],X[3],X[4],X[5],BAN);
                    IF TI<6 THEN
                       COMPUERTAS
                    ELSE
                       FLIPFLOP;
            UNTIL EOF(TAB);
    END;
  CLOSE(TAB);
END;(*ACOMODA*)


(**** PROGRAMA PRINCIPAL***)

BEGIN
LIGADO;
ACOMODA;
END.
```

CAPITULO 5

DESCRIPCION DEL PROGRAMA DIBUJO

DESCRIPCION DEL PROGRAMA DIBUJO.

1.- INTRODUCCION.

El programa DIBUJO es la parte final del proceso de simulaciòn del circuito, es el encargado de dibujar fisicamente los elementos lògicos del circuito simulado (compuertas, flip-flops,etc).

El programa DIBUJO toma datos del archivo DIB, los datos almacenados en èste archivo son los referentes a la posiciòn de cada uno de los elementos en el circuito dibujado, tambièn lee datos del archivo TAB, el cual contiene los elementos ordenados con respecto a liga, y ademàs toma del archivo VALOR el dato referente al nùmero de circuitos.

Este programa genera los archivos NUEVO y FINAL, en los que se almacenan las coordenadas del circuito a dibujar.

2 .- DESCRIPCION DEL FUNCIONAMIENTO.

El funcionamiento del programa DIBUJO, en forma general, es el siguiente:

En primer lugar dibuja los componentes (compuertas y flip-flops), para esto el programa lee los datos necesarios, del archivo DIB, en el que se encuentran el TIPO y POSICION del elemento en el circuito. Al mismo tiempo que son dibujados los componentes del circuito, sus coordenadas son almacenadas en un archivo llamado NUEVO.

Después del proceso anterior, y en base al archivo NUEVO, se realiza el proceso de generación de las líneas que conectan a los elementos del circuito entre sí. Las coordenadas de estas líneas son guardadas o almacenadas en el archivo llamado FINAL, inmediatamente después de esto se realiza un proceso de distribución de líneas, para proceder a dibujarlas enseguida y de ésta manera obtener el circuito completamente dibujado.

De ésta forma finaliza la simulación del circuito.

3.- DESCRIPCION DE ENTRADAS Y SALIDAS.

ENTRADAS.-

Datos que han sido almacenados previamente en archivos.

Las variables de TIPO y POSICION del elemento son tomadas del archivo DIB, las variables de ELEMENTO, TIPO, LIGA, ENTRADAS Y SALIDAS son tomadas del archivo TAB, y la variable del número de circuitos es tomada del archivo VALOR.

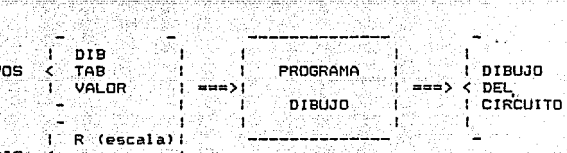
Datos proporcionados por el usuario.

El factor de escala R y las coordenadas del centro del dibujo.

SALIDAS.-

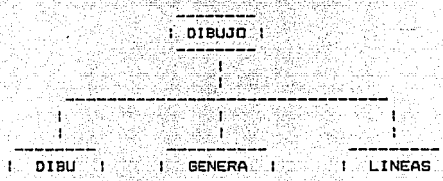
EL dibujo del circuito simulado.

El esquema siguiente representa lo escrito anteriormente.

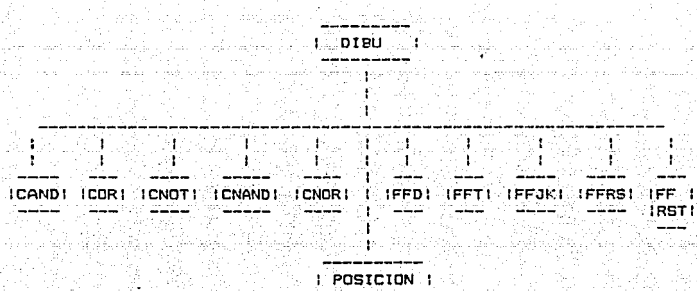


4.- DIAGRAMAS DE BLOQUES DEL PROGRAMA DIBUJO.

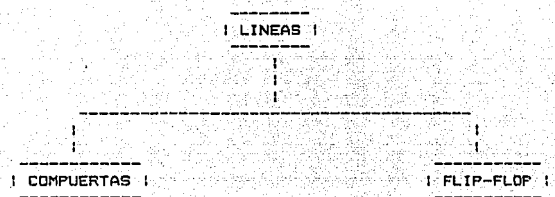
I.-



II.-



III.-



5.- DESCRIPCION DE LAS SUBRUTINAS QUE COMPONEN AL PROGRAMA DIBUJO.

DESCRIPCION DE LA SUBRUTINA DIBU.

OBJETIVO.-

El objetivo de la subrutina DIBU, es dibujar los elementos que componen al circuito de tal forma que tengan una distribución adecuada en la pantalla, y generar un archivo en donde queden almacenadas las coordenadas de cada componente dibujado, para que con ésta información sea posible dibujar las líneas que conectan a los elementos del circuito entre sí.

FUNCION .-

La función desarrollada por ésta subrutina es la siguiente: lee el número de componentes a dibujar, éste dato es tomado del archivo VALOR; del archivo DIB, toma la información referente al tipo y posición del elemento, y en base a esto procede a dibujar los elementos que integran al circuito.

DESARROLLO.-

Para poder realizar la función encomendada se lleva a cabo el procedimiento siguiente:

Primeramente se leerá el valor del número de circuitos a dibujar.

Después se procede a realizar el dibujo, para lo cual se hace lo siguiente:

1.- Leer la posición y tipo del elemento correspondiente.

2.- Según la clave de posición el elemento se dibujará: primero que los demás, enfrente o abajo del que se dibujó anteriormente.

3.- Según la clave del tipo, será el elemento que se dibuje (AND, OR, etc.), y se almacenarán las coordenadas del elemento dibujado.

La subrutina DIBUJO se compone a su vez, de las siguientes subrutinas:

CAND : Dibuja la compuerta AND.

COR : Dibuja la compuerta OR.

CNOT : Dibuja la compuerta NOT.

CNAND: Dibuja la compuerta NAND.

CNOR : Dibuja la compuerta NOR.

FFD : Dibuja al flip-flop tipo D.

FFT : Dibuja al flip-flop tipo T.

FFRS : Dibuja al flip-flop tipo RS.

FFJK : Dibuja al flip-flop tipo JK.

FFRST: Dibuja al flip-flop typo RST.

POSICION : Determina el lugar que va a ocupar el elemento en el dibujo.

La escala a la que será dibujado el circuito dependerá del factor de escala R, que es proporcionado por el usuario.

DESCRIPCION DE LA SUBRUTINA GENERA.

OBJETIVO.

En base a los archivos TAB y NUEVO generar un nuevo archivo llamado FINAL, que debe tener la siguiente información:

EL	: Elemento.
TI	: Tipo.
LI	: Liga.
X1	: Entrada número 1.
AX,AY	: Coordenadas de la entrada 1.
X2	: Entrada número 2.
BX,BY	: Coordenadas de la entrada 2.
X3	: Entrada número 3.
CX,CY	: Coordenadas de X3.
X4	: Entrada número 4 o salida número 1.
DX,DY	: Coordenadas de X4.
X5	: SALIDA o salida número 2.
EX,EY	: Coordenadas de la salida.

FUNCION.

Generar un archivo con los datos necesarios para el subprograma que dibuja las líneas.

DESARROLLO.

Hace una concatenación de los archivos NUEVO y TAB.

DESCRIPCION DE LA SUBRUTINA LINEAS.

OBJETIVO.

Dibujar las líneas que conectan a los elementos del circuito entre sí.

FUNCION.

Lee las coordenadas de las entradas y salidas de cada elemento y dibuja la línea hacia el punto de conexión correspondiente, el trazo de las líneas es en base a una cuadrícula, en la que previamente se ha dividido la pantalla, ésta cuadrícula es transparente para el usuario.

La información necesaria para realizar lo anterior va a ser tomada de los archivos VALOR (número de conexiones y elementos) y FINAL (toda la información referente al elemento así como las coordenadas de las entradas y/o salidas).

DESARROLLO.

La función encomendada a ésta rutina se lleva a cabo de la siguiente manera:

Lee en el archivo VALOR los datos correspondientes a: número de circuitos, número Flip-Flops, total de entradas, total de conexiones. También se lee en el archivo FINAL toda la información correspondiente al elemento, ésto es, número del elemento, tipo, liga, entradas y salidas y sus coordenadas correspondientes.

Después de haber leído la información necesaria se trazan las líneas de conexión de la siguiente formal

1.0) Toma el primer registro del archivo FINAL y se comparan las variables de entrada tomando como base de comparación el elemento ahí almacenado, con las entradas correspondientes de los restantes elementos del circuito, y si se encuentra algún elemento con alguna entrada en común al elemento actual se dibuja una línea cuyo punto inicial

estará dado por las coordenadas de la entrada que se ésta comparando, y las coordenadas del punto final serán las coordenadas de la entrada encontrada.

1.b) La trayectoría de la línea trazada será la siguiente:

a) Trazar una línea hacia la izquierda hasta llegar al límite del cuadro imaginario.

b) Trazar una línea hacia abajo, la ordenada "Y" del segundo punto de ésta línea debe ser igual a la ordenada "Y" de la entrada encontrada más un incremento, de ésta manera la línea trazada debe llegar hasta el límite inferior del cuadro que ocupa el elemento encontrado.

c) Trazar una línea hacia la derecha, la abcisa "X" del segundo punto de ésta línea debe ser igual a la abcisa de la entrada encontrada.

d) Trazar una línea que una los dos puntos restantes.

2.- Si no se encontraron entradas comunes, entonces, se comparan las entradas del componente actual con las salidas de los componentes restantes, si encontramos algún elemento con salida común a la entrada comparada, entonces trazamos una línea al frente, la trayectoria de ésta línea será trazada mediante un procedimiento similar al explicado en los incisos a) al d) del punto 1.

3.- Se lee el registro siguiente y se repite el proceso, tomando en cuenta que los elementos que ya han sido tomados como base de comparación, ya no se toman en cuenta para comparaciones subsecuentes.

La subrutina LINEAS incluye otras dos subrutinas que son: COMPUERTAS y FLIP-FLOP.

COMPUERTAS : Realiza el procedimiento anteriormente explicado, sólo cuando los elementos corresponden al grupo de las compuertas (AND,OR, NOT, etc.).

FLIPFLOP : Realiza el procedimiento anterior cuando los elementos corresponden al grupo de los Flip-Flops (T, D, JK, etc.).

A continuación y como parte final de éste capítulo presentamos un listado del programa DIBUJO.

```
PROGRAM DIBUJO(INPUT,OUTPUT,DIB,TAB,nuevo,FINAL,VALOR);
VAR r,c0,c1,nc,i,xx,yy,cdib,ti,pos:integer;
   ax,ay,bx,by,cx,cy,dx,dy,ex,ey:integer;
   alam,p,tc:array[1..10] of integer;
   cc,cf,ce,ca:integer;
   TAB,FINAL,nuevo,DIB,VALOR:FILE OF INTEGER;
{$i GRAPH.P}
PROCEDURE GENERA;FORWARD;
PROCEDURE DIBU;

procedure cand;
var hx,hy,x0,y0,n,r1,r2:integer;
    ang:real;
begin
n:=1;
ang:=n*3.1459/180;
(* c0:=160;c1:=100*)
x0:=c0;y0:=c1;
draw(x0,y0,x0,y0-r,2);
draw(x0,y0,x0,y0+r,2);
r1:=trunc(r/2);
r2:=trunc(r1/2);
(***** rayas de entrada ******)
draw(x0-r1,y0-r1-r2,x0,y0-r1-r2,2);
ax:=x0-r1;ay:=y0-r1-r2;
draw(x0-r1,y0+r1+r2,x0,y0+r1+r2,2);
bx:=ax;by:=y0+r1+r2;
draw(x0-r1,y0-r1+r2,x0,y0-r1+r2,2);
cx:=ax;cy:=y0-r1+r2;
draw(x0-r1,y0+r1-r2,x0,y0+r1-r2,2);
dx:=ax;dy:=y0+r1-r2;
while n<=180 do
begin
ang:=n*3.14159/180;
hx:=trunc(round(r*sin(ang)));
hy:=trunc(round(r*cos(ang)));
plot(hx+x0,hy+y0,2);
n:=n+1;
end;
(***** raya de salida ****)
draw(x0+r,y0,x0+r+2*r2,y0,2);
ex:=x0+r+2*r2;ey:=y0;
write(nuevo,ax,ay,bx,by,cx,cy,dx,dy,ex,ey);
end;

procedure cor;
var hx,hy,x0,y0,n,r1,r2,hx1,hy1,n1:integer;
    ang:real;
```

```
begin
n:=1;
n1:=50;
ang:=n*3.1459/180;
(* c0:=160;c1:=100*)
x0:=c0;y0:=c1;
r1:=trunc(r/2);
r2:=trunc(r1/2);
(***** rayas de entrada *****)
draw(x0-r1,y0-r1-r2,x0+r2,y0-r1-r2,2);
ax:=x0-r1;ay:=y0-r1-r2;
draw(x0-r1,y0+r1+r2,x0+r2,y0+r1+r2,2);
bx:=ax;by:=y0+r1+r2;
draw(x0-r1,y0-r1+r2,x0+r1,y0-r1+r2,2);
cx:=ax;cy:=y0-r1+r2;
draw(x0-r1,y0+r1-r2,x0+r1,y0+r1-r2,2);
dx:=ax;dy:=y0+r1-r2;
while n<=180 do
begin
ang:=n*3.14159/180;
hx:=trunc(round(r*sin(ang)));
hy:=trunc(round(r*cos(ang)));
plot(hx+x0,hy+y0,2);
n:=n+1;
end;
while n1<=130 do
begin
ang:=n1*3.14159/180;
hx1:=trunc(round((r1+r)*sin(ang)));
hy1:=trunc(round((r1+r)*cos(ang)));
plot(x0+hx1-r,y0+hy1,2);
n1:=n1+1;
end;
(***** raya de salida *****)
draw(x0+r,y0,x0+r+2*r2,y0,2);
ex:=y0+r+2*r2;ey:=y0;
write(nuevo,ax,ay,bx,by,cx,cy,dx,dy,ex,ey);
end;
procedure cnot;
var
x0,y0,r1,r2,r3,r4:integer;
begin
(* c0:=160;c1:=100*)
x0:=c0;y0:=c1;
r1:=trunc(r/4);
draw(x0,y0,x0,y0+r1,2);
draw(x0,y0,x0,y0-r1,2);
r3:=trunc((sqrt(15)/4)*r);
draw(x0,y0-r1,x0+r3,y0,2);
draw(x0,y0+r1,x0+r3,y0,2);
(* raya de entrada *)
draw(x0,y0,x0-2*r1,y0,2);
ax:=x0-2*r1;ay:=y0;
r2:=trunc(r*0.5/3);
r4:=trunc(r2/5);
```

```
circle(x0+r3+r2,y0-r4,r2,2);
(***** raya de salida *****)
draw(x0+r3+2*r2,y0,x0+r3+3*r2,y0,2);
ex:=x0+r3+3*r2;ey:=y0;
bx:=0;by:=0;cx:=0;cy:=0;dx:=0;dy:=0;
write(nuevo,ax,ay,bx,by,cx,cy,dx,dy,ex,ey);
end;

procedure cnand;
var hx,hy,x0,y0,n,r1,r3,r2:integer;
    ang:real;
begin
n:=1;
ang:=n*3.1459/180;
(* c0:=160;c1:=100*)
x0:=c0;y0:=c1;
draw(x0,y0,x0,y0-r,2);
draw(x0,y0,x0,y0+r,2);
r1:=trunc(r/2);
r2:=trunc(r1/2);
(******* rayas de entrada *******)
draw(x0-r1,y0-r1-r2,x0,y0-r1-r2,2);
ax:=x0-r1;ay:=y0-r1-r2;
draw(x0-r1,y0+r1+r2,x0,y0+r1+r2,2);
bx:=ax;by:=y0+r1+r2;
draw(x0-r1,y0-r1+r2,x0,y0-r1+r2,2);
cx:=ax;cy:=y0-r1+r2;
draw(x0-r1,y0+r1-r2,x0,y0+r1-r2,2);
dx:=ax;dy:=y0+r1-r2;
while n<=180 do
begin
ang:=n*3.14159/180;
hx:=trunc(round(r*sin(ang)));
hy:=trunc(round(r*cos(ang)));
plot(hx+x0,hy+y0,2);
n:=n+1;
end;
r3:=trunc(r1/3);
circle(x0+r+r3,y0,r3,2);
(***** raya de salida *****)
draw(x0+r+2*r3,y0,x0+r+2*r3+r2,y0,2);
ex:=x0+r+2*r3+r2;ey:=y0;
write(nuevo,ax,ay,bx,by,cx,cy,dx,dy,ex,ey);
end;

procedure cnor;
var hx,hy,x0,y0,n,r1,r2,r3,hx1,hy1,n1:integer;
    ang:real;
begin
n:=1;
n1:=50;
ang:=n*3.1459/180;
(* c0:=160;c1:=100*)
x0:=c0;y0:=c1;
(draw(x0,y0,x0,y0,2);)
```

```
(draw(x0,y0,x0,y0+r,2);)
r1:=trunc(r/2);
r2:=trunc(r1/2);
(******* rayas de entrada ******)
draw(x0-r1,y0-r1-r2,x0+r2,y0-r1-r2,2);
ax:=x0-r1;ay:=y0-r1-r2;
draw(x0-r1,y0+r1+r2,x0+r2,y0+r1+r2,2);
bx:=ax;by:=y0+r1+r2;
draw(x0-r1,y0-r1+r2,x0+r1,y0-r1+r2,2);
cx:=ax;cy:=y0-r1+r2;
draw(x0-r1,y0+r1-r2,x0+r1,y0+r1-r2,2);
dx:=ax;dy:=y0+r1-r2;
while n<=180 do
begin
ang:=n*3.14159/180;
hx:=trunc(round(r*sin(ang)));
hy:=trunc(round(r*cos(ang)));
plot(hx+x0,hy+y0,2);
n:=n+1;
end;
while n1<=130 do
begin
ang:=n1*3.14159/180;
hx1:=trunc(round((r1+r)*sin(ang)));
hy1:=trunc(round((r1+r)*cos(ang)));
plot(x0+hx1-r,y0+hy1,2);
n1:=n1+1;
end;
r3:=trunc(r1/3);
circle(x0+r+r3,y0,r3,2);
(******* raya de salida *******)
draw(x0+r+2*r3,y0,x0+r+2*r3+r2,y0,2);
ex:=x0+r+2*r3+r2;ey:=y0;
write(nuevo,ax,ay,bx,by,cx,cy,dx,dy,ex,ey);
end;

procedure ffd;
var x0,y0,r1,r2,r3,r4,r22:integer;
begin
x0:=c0;y0:=c1;
r1:=trunc(r/2);
r2:=trunc(r/4);
draw(x0-r1,y0,x0-r1,y0-r,2);
draw(x0-r1,y0-r,x0-r1+r,y0-r,2);
draw(x0-r1+r,y0-r,x0-r1+r,y0-r+2*r,2);
draw(x0-r1+r,y0+r,x0-r1,y0+r,2);
draw(x0-r1,y0+r,x0-r1,y0,2);
(***** raya de entrada *******)
draw(x0-r1,y0,x0-r1-r2,y0,2);
ax:=x0-r1-r2;ay:=y0;
(*  rayas  de salida  *)
draw(x0+r1,y0-r1,x0+2*r1,y0-r1,2);
dx:=x0+2*r1;dy:=y0-r1;
draw(x0+r1,y0+r1,x0+2*r1,y0+r1,2);
ex:=dx;ey:=y0+r1;
```

```
bx:=0;by:=0;cx:=0;cy:=0;
write(nuevo,ax,ay,bx,by,cx,cy,dx,dy,ex,ey);
(* trazo de la letra D  *)
r2:=trunc(r/4);
r22:=trunc(r2/2);
r3:=trunc(r2/8);
draw(x0,y0,x0,y0-r2,2);
draw(x0,y0-r2,x0-r2+r1-r3,y0-r2,2);
draw(x0-r2+r1-r3,y0-r2,x0-r2+r1-r3,y0-r2+2*r2,2);
draw(x0-r2+r1-r3,y0+r2,x0,y0+r2,2);
draw(x0,y0+r2,x0,y0,2);
(* signos de mas y de menos  *)
draw(x0+r2+5*r3,y0-r1,x0+r1+r3,y0-r1,2);
r4:=trunc(r2/2);
draw(x0+r1+r3-r4,y0-r1-r4,x0+r1+r3-r4,y0-r1+r22,2);
draw(x0+r2+4*r3,y0+r1,x0+r1+r3,y0+r1,2);
end;

procedure fft;
var x0,y0,r1,r2,r3,r4,r22:integer;
begin
x0:=c0;y0:=c1;
r1:=trunc(r/2);
r2:=trunc(r/4);
draw(x0-r1,y0,x0-r1,y0-r,2);
draw(x0-r1,y0-r,x0-r1+r,y0-r,2);
draw(x0-r1+r,y0-r,x0-r1+r,y0-r+2*r,2);
draw(x0-r1+r,y0+r,x0-r1,y0+r,2);
draw(x0-r1,y0+r,x0-r1,y0,2);
(* raya de entrada *)
draw(x0-r1,y0,x0-r1-r2,y0,2);
ax:=x0-r1-r2;ay:=y0;
(*  rayas  de salida  *)
draw(x0+r1,y0-r1,x0+2*r1,y0-r1,2);
dx:=x0+2*r1;dy:=y0-r1;
draw(x0+r1,y0+r1,x0+2*r1,y0+r1,2);
ex:=dx;ey:=y0+r1;
bx:=0;by:=0;cx:=0;cy:=0;
write(nuevo,ax,ay,bx,by,cx,cy,dx,dy,ex,ey);
(* trazo de la letra T  *)
r2:=trunc(r/4);
r22:=trunc(r2/2);
r3:=trunc(r2/8);
draw(x0-r22+r3,y0-r2,x0-r2+r1-r3,y0-r2,2);
draw(x0+r22,y0-r2,x0+r22,y0+r2,2);
(* signos de mas y de menos  *)
draw(x0+r2+4*r3,y0-r1,x0+r1+r3,y0-r1,2);
r4:=trunc(r2/2);
draw(x0+r1-r4,y0-r1-r4,x0+r1-r4,y0-r1+r22,2);
draw(x0+r2+4*r3,y0+r1,x0+r1+r3,y0+r1,2);
end;

procedure ffrs;
var x0,y0,r1,r2,r3,r4,r22:integer;
begin
```

```
x0:=c0;y0:=c1;
r1:=trunc(r/2);
draw(x0-r1,y0,x0-r1,y0-r,2);
draw(x0-r1,y0-r,x0-r1+r,y0-r,2);
draw(x0-r1+r,y0-r,x0-r1+r,y0-r+2*r,2);
draw(x0-r1+r,y0+r,x0-r1,y0+r,2);
draw(x0-r1,y0+r,x0-r1,y0,2);
r2:=trunc(r/4);
(* rayas de entrada *)
draw(x0-r1,y0-r1,x0-r1-r2,y0-r1,2);
ax:=x0-r1-r2;ay:=y0-r1;
draw(x0-r1,y0+r1,x0-r1-r2,y0+r1,2);
bx:=ax;by:=y0+r1;
(*  rayas  de salida  *)
draw(x0+r1,y0-r1,x0+2*r1,y0-r1,2);
dx:=x0+2*r1;dy:=y0-r1;
draw(x0+r1,y0+r1,x0+2*r1,y0+r1,2);
ex:=dx;ey:=y0+r1;
cx:=0;cy:=0;
write(nuevo,ax,ay,bx,by,cx,cy,dx,dy,ex,ey);
r22:=trunc(r2/2);
r3:=trunc(r2/8);
(* trazo de la letra R *)
draw(x0-r1+r4,y0-r1-r22+r3,x0-r1+r2+r4+r3,y0-r1-r22,2);
r4:=trunc(r2/2);
draw(x0-r1+r4,y0-r1-r4,x0-r1+r4,y0-r1+r2,2);
(*trazo de la letra  S*)
draw(x0-r1+r4,y0+r22,x0-r1+r2+r4,y0+r22,2);
draw(x0-r1+r4,y0+r22,x0-r1+r4,y0+r1,2);
draw(x0-r1+r4,y0+r1,x0-r1+r2+r4,y0+r1,2);
draw(x0-r1+r2+r4,y0+r1,x0-r1+r2+r4,y0+r1+r2+r22,2);
draw(x0-r1+r4,y0+r1+r2+r22,x0-r1+r2+r4,y0+r1+r2+r22,2);
(* signos de mas y de menos *)
draw(x0+r2+4*r3,y0-r1,x0+r1+r3,y0-r1,2);
draw(x0+r1-r4,y0-r1-r4,x0+r1-r4,y0-r1+r22,2);
draw(x0+r2+4*r3,y0+r1,x0+r1+r4,y0+r1,2);
end;

procedure ffjk;
var x0,y0,r1,r2,r3,r4,r22:integer;
begin
x0:=c0;y0:=c1;
r1:=trunc(r/2);
r2:=trunc(r1/2);
draw(x0-r1,y0,x0-r1,y0-r,2);
draw(x0-r1,y0-r,x0-r1+r,y0-r,2);
draw(x0-r1+r,y0-r,x0-r1+r,y0-r+2*r,2);
draw(x0-r1+r,y0+r,x0-r1,y0+r,2);
draw(x0-r1,y0+r,x0-r1,y0,2);
r2:=trunc(r/4);
(* rayas de entrada *)
draw(x0-r1,y0-r1,x0-r1-r2,y0-r1,2);
ax:=x0-r1-r2;ay:=y0-r1;
draw(x0-r1,y0+r1,x0-r1-r2,y0+r1,2);
bx:=ax;by:=y0+r1;
```

```
(*  rayas  de salida  *)
draw(x0+r1,y0-r1,x0+2*r1,y0-r1,2);
dx:=x0+2*r1;dy:=y0-r1;
draw(x0+r1,y0+r1,x0+2*r1,y0+r1,2);
ex:=dx;ey:=y0+r1;
cx:=0;cy:=0;
write(nuevo,ax,ay,bx,by,cx,cy,dx,dy,ex,ey);
r22:=trunc(r2/2);
r3:=trunc(r2/8);
(* trazo de la letra J *)
draw(x0-r1+r22,y0-r1-r22,x0-3*r3,y0-r1-r22,2);
r4:=trunc(r2/2);
draw(x0-r1+r2,y0-r1-r4,x0-r1+r2,y0-r1+r2,2);
draw(x0-r1+r22,y0-r1+r2,x0-r1+r22,y0-r1+r2-r22,2);
draw(x0-r1+r2,y0-r1+r2,x0-r1+r22,y0-r1+r2,2);
(*trazo de la letra  k*)
draw(x0-r1+r22,y0+2*r2,x0-r1+r22,y0+r1+r2+r22,2);
draw(x0-r1+r22,y0+r1+r22,x0-r1+3*r4,y0+r1,2);
draw(x0-r1+r22,y0+r1+r22,x0-r1+3*r4,y0+r1+r2+r22,2);
(* signos de mas y menos *)
draw(x0+r2+4*r3,y0-r1,x0+r1+r3,y0-r1,2);
draw(x0+r1-r4,y0-r1-r4,x0+r1-r4,y0-r1+r22,2);
draw(x0+r2+4*r3,y0+r1,x0+r1+r3,y0+r1,2);
end;

procedure ffrst;
var x0,y0,r1,r2,r3,r4,r22:integer;
begin
x0:=c0;y0:=c1;
r1:=trunc(r/2);
draw(x0-r1,y0,x0-r1,y0-r,2);
draw(x0-r1,y0-r,x0-r1+r,y0-r,2);
draw(x0-r1+r,y0-r,x0-r1+r,y0-r+2*r,2);
draw(x0-r1+r,y0+r,x0-r1,y0+r,2);
draw(x0-r1,y0+r,x0-r1,y0,2);
r2:=trunc(r/4);
(* rayas de entrada *)
draw(x0-r1,y0-r1,x0-r1-r2,y0-r1,2);
ax:=x0-r1-r2;ay:=y0-r1;
draw(x0-r1,y0,x0-r1-r2,y0,2);
bx:=ax;by:=y0;
draw(x0-r1,y0+r1,x0-2*r1,y0+r1,2);
cx:=ax;cy:=y0+r1;
(*  rayas  de salida  *)
draw(x0+r1,y0-r1,x0+2*r1,y0-r1,2);
dx:=x0+2*r1;dy:=y0-r1;
draw(x0+r1,y0+r1,x0+2*r1,y0+r1,2);
ex:=dx;ey:=y0+r1;
write(nuevo,ax,ay,bx,by,cx,cy,dx,dy,ex,ey);
r22:=trunc(r2/2);
r3:=trunc(r2/8);
(* trazo de la letra R *)
draw(x0-r1+r4,y0-2*r1+r3+r22,x0-r1+r2+r4+r3,y0-2*r1+r22,2);
r4:=trunc(r2/2);
draw(x0-r1+r4,y0-2*r1-r4+2*r22,x0-r1+r4,y0-2*r1+r2+2*r22,2);
```

```
(* trazo de la letra T *)
draw(x0-r1+r4,y0-3*r3,x0-r1+r2+r4+r3,y0-3*r3,2);
draw(x0-r1+r2,y0-3*r3,x0-r1+r2,y0+r2,2);
(*trazo de la letra  S*)
draw(x0-r1+r4,y0+r2+2*r22,x0-r1+r2+r4,y0+r2+2*r22,2);
draw(x0-r1+r4,y0+r2+2*r22,x0-r1+r4,y0+r2+r1,2);
draw(x0-r1+r4,y0+r2+r1,x0-r1+r2+r4,y0+r2+r1,2);
draw(x0-r1+r2+r4,y0+r2+r1,x0-r1+r2+r4,y0+r2+r1+r2,2);
draw(x0-r1+r4,y0+r2+r1+r2,x0-r1+r2+r4,y0+r2+r1+r2,2);
(* signos de mas y de menos *)
draw(x0+r2+4*r3,y0-r1,x0+r1+r3,y0-r1,2);
draw(x0+r1-r4,y0-r1-r4,x0+r1-r4,y0-r1+r22,2);
draw(x0+r2+4*r3,y0+r1,x0+r1+r4,y0+r1,2);
end;
procedure posicion;
begin
     if p[i]=1 then
     begin
     c0:=xx;c1:=yy;
     end
     else
     if p[i]=2 then
     begin
     xx:=xx+3*r;
     c0:=xx;
     end
     else
     if p[i]=3 then
     begin
     yy:=yy+3*r;
     c1:=yy;
     end
     else
     begin
     yy:=yy-3*r;
     c1:=yy;
     end;
end;
(**PROGRAMA PRINCIPAL DE LA SUBRUTINA QUE DIBUJA COMPONENTES**)
begin
ASSIGN(DIB,'DIB.DOS');
RESET(DIB);
assign(nuevo,'nuevo.dos');
rewrite(nuevo);
{graphics;}
cdib:=0;
while not eof(dib) do
begin
  read(dib,ti,pos);
  cdib:=cdib+1;
end;
reset(dib);
writeln('dame el centro ');
read(xx,yy);
c0:=xx;c1:=yy;
```

```
writeln;
writeln('dame r ');
read(r);
writeln;
(****** ASIGNACION DEL NUM. DE CIRCUITOS ***)
nc:=cdib;
(**********)
writeln;
for i:= 1 to nc do
begin
read(DIB,tc[i]);
(*write('inicio <1>, enfr<2>,aba<3>, arri<4>');*)
read(DIB,p[i]);
end;
for i:=1 to nc do
clearscreen;
for i:= 1 to nc do
begin
case tc[i] of
      1:begin
        posicion;cand;
        end;
      2:begin
        posicion;cor;
        end;
      3:begin
        posicion;cnot;
        end;
      4:begin
          posicion;cnand;
          end;
      5:begin
         posicion;cnor;
         end;
      6:begin
         posicion;ffd;
        end;
      7:begin
        posicion;fft;
        end;
      8:begin
        posicion;ffrs;
        end;
      9:begin
        posicion;ffjk;
        end;
      10:begin
         posicion;ffrst;
         end;
end;
end;
end;(*DIBU*)
PROCEDURE LINEAS;
TYPE TAB=RECORD
                    ELEM:INTEGER;
```

```
                TIPO:INTEGER;
                LIGA:INTEGER;
                W    :ARRAY[1..5] OF INTEGER;
                A,B :ARRAY[1..5] OF INTEGER;
                BAN :INTEGER;
            END;
VAR
   VECTOR:TAB;
   F,I,J,K,L,L1,FLAG,DX,DY,AUX,CUX,INC,INC2,INC3,INC4:INTEGER;
   TI,LI,BAN,EL,BA,II,JJ,J11,J1,J10,ELE,TIP,LIG,INC5,temp,t:INTEGER;
             X:ARRAY[1..5] OF INTEGER;
           C,D :ARRAY[1..5] OF INTEGER;
PROCEDURE COMPUERTAS;
LABEL SAL;
BEGIN
 WITH VECTOR DO
  BEGIN
    L1:=0;
    FOR K:=1 TO 4 DO
     BEGIN
      FOR J:=1 TO 4 DO
       if w[k]<>temp then
        BEGIN
         IF W[K]=X[J] THEN
          BEGIN
           if ti<6 then
            begin
             IF (W[K]<>0) AND (X[J]<>0) THEN
              BEGIN
               temp:=x[j];
               {WRITELN(LIGA,TI,'ABAJO');}
               INC:=TRUNC((R/4)*J);
               IF TI=3 THEN
                INC2:=TRUNC(6*R/5)
               ELSE
                BEGIN
                INC2:=TRUNC(R/8+(R/4)*(J))+
                  TRUNC(ROUND(ABS(R*SIN(SQRT(EL)*EL*3.1416/180))));
                END;
                INC3:=TRUNC((3*R/4)*(J-1));
                IF (J=1) OR (J=3) THEN
                BEGIN
                 DRAW(A[K],B[K],A[K]-INC,B[K],2);
                 DRAW(A[K]-INC,B[K],A[K]-INC,D[J]-INC2,2);
                 DRAW(A[K]-INC,D[J]-INC2,C[J]-INC3,D[J]-INC2,2);
                 DRAW(C[J]-INC3,D[J]-INC2,C[J]-INC3,D[J],2);
                 DRAW(C[J]-INC3,D[J],C[J],D[J],2);
                END
                ELSE
                BEGIN
                 DRAW(A[K],B[K],A[K]-INC,B[K],2);
                 DRAW(A[K]-INC,B[K],A[K]-INC,D[J]+INC2,2);
                 DRAW(A[K]-INC,D[J]+INC2,C[J]-INC,D[J]+INC2,2);
                 DRAW(C[J]-INC,D[J]+INC2,C[J],D[J]+INC2,2);
                 DRAW(C[J],D[J]+INC2,C[J],D[J],2);
```

```
                END;
               END;
              end
              else
              begin
               if (W[K]<>0) and (X[J]<>0) then
                begin
                 temp:=x[j];
                 INC:=TRUNC(R*2/(3*K)+R/6);
                 INC5:=TRUNC((R/6)*(J-1));
                 IF J>=4 THEN
                  BEGIN
                   INC5:=TRUNC(-R/8);
                   INC2:=TRUNC(R/2)+TRUNC((R/8)*J)
                  END
                 ELSE INC2:=R+TRUNC((R/4)*(J-1));
                 DRAW(A[K],B[K],A[K]-INC,B[K],2);
                 DRAW(A[K]-INC,B[K],A[K]-INC,D[J]-INC2,2);
                 DRAW(A[K]-INC,D[J]-INC2,C[J]-INC5,D[J]-INC2,2);
                 DRAW(C[J]-INC5,D[J]-INC2,C[J]-INC5,D[J],2);
                 DRAW(C[J]-INC5,D[J],C[J],D[J],2);
                end;
               end;
              END
             ELSE
             IF (X[J]=W[5]) AND (X[J]<>0) AND (W[5]<>0)  AND(FLAG=0) THEN
             BEGIN
              INC3:=TRUNC((R/4)*J);
              INC4:=R+TRUNC((R/5)*(J-1));
              {WRITELN(LIGA,TI,'ENFRENTE');}
              draw(A[5],B[5],A[5],B[5]+INC4,2);
              DRAW(A[5],B[5]+INC4,C[J]-INC3,B[5]+INC4,2);
              DRAW(C[J]-INC3,B[5]+INC4,C[J]-INC3,D[J],2);
              DRAW(C[J]-INC3,D[J],C[J],D[J],2);
              FLAG:=1;
              goto sal;
             END;
             IF (W[K]=X[5]) AND (X[5]<>0) THEN
              BEGIN
               INC2:=TRUNC((3*R/2)+(R/8)*5);
               INC5:=TRUNC(-3*R/8);
               INC:=TRUNC(R*2/(3*K)+R/6);
               DRAW(A[K],B[K],A[K]-INC,B[K],2);
               DRAW(A[K]-INC,B[K],A[K]-INC,D[5]-INC2,2);
               DRAW(A[K]-INC,D[5]-INC2,C[5]-INC5,D[5]-INC2,2);
               DRAW(C[5]-INC5,D[5]-INC2,C[5]-INC5,D[5],2);
               DRAW(C[5]-INC5,D[5],C[5],D[5],2);
              END;
           END;(*for j*)
       END;(* for k *)
    END;
  sal: {writeln;}
END;(*COMPUERTAS*)

PROCEDURE FLIPFLOP;
```

```
LABEL SAL;
BEGIN
 WITH VECTOR DO
  BEGIN
   FLAG:=0;L:=0;L1:=0;
   FOR K:=1 TO 3 DO
    BEGIN
     FOR J:=1 TO 4 DO
      BEGIN
       if w[k]<>temp then
        IF W[K]=X[J] THEN
         BEGIN
          IF TI>5 THEN
           BEGIN
            IF (W[K]<>0) AND (X[J]<>0) THEN
             BEGIN
              temp:=x[j];
              INC:=TRUNC(R*2/(3*K)+R/6);
              INC5:=TRUNC((R/6)*(J-1));
              IF J>=4 THEN
               BEGIN
                INC2:=TRUNC(3*R/2)+TRUNC((R/8)*J);
                INC5:=TRUNC(-R/8);
               END
              ELSE
              INC2:=R+TRUNC((R/4)*(J-1));
              DRAW(A[K],B[K],A[K]-INC,B[K],2);
              DRAW(A[K]-INC,B[K],A[K]-INC,D[J]-INC2,2);
              DRAW(A[K]-INC,D[J]-INC2,C[J]-INC5,D[J]-INC2,2);
              DRAW(C[J]-INC5,D[J]-INC2,C[J],D[J]-INC2,2);
              DRAW(C[J],D[J]-INC2,C[J],D[J],2);
             END
            ELSE
            IF (W[K]<>0) AND (X[J]<>0) THEN
             BEGIN
              temp:=x[j];
              {WRITELN(LIGA,TI,'ABAJO');}
              INC:=TRUNC((R/4)*J);
              IF TI=3 THEN
               INC2:=TRUNC(6*R/5)
              ELSE
               BEGIN
               INC2:=TRUNC(R/8+(R/4)*(J))+
                  TRUNC(ROUND(ABS(R*SIN(SQRT(EL)*EL*3.1416/180))));
              END;
              INC3:=TRUNC((3*R/4)*J-1);
              IF (J=1) OR (J=3) THEN
              BEGIN
               DRAW(A[K],B[K],A[K]-INC,B[K],2);
               DRAW(A[K]-INC,B[K],A[K]-INC,D[J]-INC2,2);
               DRAW(A[K]-INC,D[J]-INC2,C[J]-INC3,D[J]-INC2,2);
               DRAW(C[J]-INC3,D[J]-INC2,C[J]-INC3,D[J],2);
               DRAW(C[J]-INC3,D[J],C[J],D[J],2);
              END
             ELSE
```

```
            BEGIN
             DRAW(A[K],B[K],A[K]-INC,B[K],2);
             DRAW(A[K]-INC,B[K],A[K]-INC,D[J]+INC2,2);
             DRAW(A[K]-INC,D[J]+INC2,C[J]-INC,D[J]+INC2,2);
             DRAW(C[J]-INC,D[J]+INC2,C[J],D[J]+INC2,2);
             DRAW(C[J],D[J]+INC2,C[J],D[J],2);
            END;
          END;
        END;
       END
       ELSE
        IF (X[J]=W[4]) {AND(FLAG=0)} AND (X[J]<>0) AND (W[4]<>0) THEN
         BEGIN
          INC3:=TRUNC((3*R/5)*(J-1));
          {WRITELN(LIGA,TI,'ENFRENTE');}
          DRAW(A[4],B[4],A[4]+TRUNC(R/4),B[4],2);
    DRAW(A[4]+TRUNC(R/4),B[4],A[4]+TRUNC(R/4),B[4]+TRUNC(5*R/3),2);
    DRAW(A[4]+TRUNC(R/4),B[4]+TRUNC(5*R/3),
          C[J]-INC3,B[4]+TRUNC(5*R/3),2);
          DRAW(C[J]-INC3,B[4]+TRUNC(5*R/3),C[J]-INC3,D[J],2);
          DRAW(C[J]-INC3,D[J],C[J],D[J],2);
          goto sal;
         END;
        IF (X[J]=W[5]) AND (X[J]<>0) AND(W[5]<>0) AND (FLAG=0) THEN
         BEGIN
          INC3:=TRUNC((3*R/5)*(J-1));
          {WRITELN(LIGA,TI,'ENFRENTE');}
          DRAW(A[5],B[5],A[5],B[5]+R,2);
          DRAW(A[5],B[5]+R,C[J]-INC3,B[5]+R,2);
          DRAW(C[J]-INC3,B[5]+R,C[J]-INC3,D[J],2);
          DRAW(C[J]-INC3,D[J],C[J],D[J],2);
          GOTO SAL;
         END
       END;(*for j *)
        IF (W[K]=X[5]) AND (X[5]<>0) THEN
         BEGIN
          INC2:=TRUNC((3*R/2)+(R/8)*5);
          INC5:=TRUNC(-3*R/8);
          INC:=TRUNC(R*2/(3*K)+R/6);
          DRAW(A[K],B[K],A[K]-INC,B[K],2);
          DRAW(A[K]-INC,B[K],A[K]-INC,D[5]-INC2,2);
          DRAW(A[K]-INC,D[5]-INC2,C[5]-INC5,D[5]-INC2,2);
          DRAW(C[5]-INC5,D[5]-INC2,C[5]-INC5,D[5],2);
          DRAW(C[5]-INC5,D[5],C[5],D[5],2);
         END;
      END;(* for k *)
     END;
    sal: {writeln;}
END;(*FLIPFLOP*)
{** PROGRAMA PRINCIPAL DE LA SUBRUTINA QUE DIBUJA LINEAS **}
BEGIN
reset(VALOR);
read(VALOR,cc,cf,ce,ca);
     RESET(FINAL);
     {graphics;}
```

```
FOR JJ:=1 TO NC-1 DO
 BEGIN
  RESET(FINAL);
  FOR J1:=1 TO JJ DO
   READ(FINAL,EL,TI,LI,X[1],C[1],D[1],X[2],C[2],D[2],X[3],C[3],D[3],
        X[4],C[4],D[4],X[5],C[5],D[5],BAN);
    WITH VECTOR DO
    BEGIN
     ELEM:=EL;
     TIPO:=TI;
     LIGA:=LI;
     FOR I:=1 TO 5 DO
      BEGIN
       W[I]:=X[I];
       A[I]:=C[I];
       B[I]:=D[I];
      END;
      END;
      flag:=0;L:=0;temp:=0;t:=0;
      ELE:=EL;TIP:=TI;LIG:=LI;
      REPEAT
       BEGIN
        IF NOT EOF(FINAL) THEN
         BEGIN
          READ(FINAL,EL,TI,LI,X[1],C[1],D[1],X[2],C[2],D[2],
          X[3],C[3],D[3],X[4],C[4],D[4],X[5],C[5],D[5],BAN);
         END;
         IF NOT((EL=ELE) AND (TI=TIP) AND (LI=LIG)) THEN
          BEGIN
           IF TIP<6 THEN
            COMPUERTAS
           ELSE
            FLIPFLOP;
           END;
       END;
       UNTIL EOF(FINAL);
     END;(*FOR*)
     reset (final);
     for jj:=1 to cc do
     begin
     reset (final);
     repeat
     begin
      if not eof(final) then
       begin
   read(final,el,ti,li,x[1],c[1],d[1],x[2],c[2],d[2],x[3],c[3],d[3],
             x[4],c[4],d[4],x[5],c[5],d[5],ban);
       end;
       for i:=1 to 4 do
       if (x[i]=jj)  and (x[i]<>temp) then
        begin
          temp:=jj;
          draw(c[i],d[i],c[i]-trunc(r),d[i],2);
          draw(c[i]-trunc(r),d[i]-trunc(r/6),c[i]-trunc(r),d[i],2);
        end;
```

```
      end;
      until eof(final);
    end; (* for *)
  CLOSE(FINAL);
  close (VALOR);
END;(*LINEAS*)
PROCEDURE GENERA;
VAR
  AX,AY,BX,BY,CX,CY,DX,DY,EX,EY:INTEGER;
  EL,TI,LI,BAN:INTEGER;
  X:ARRAY[1..5] OF INTEGER;
BEGIN
{ASSIGN(NUEVO,'NUEVO.DOS');}
ASSIGN(TAB,'TAB.DOS');
{ASSIGN(FINAL,'FINAL.DOS');}
RESET (NUEVO);
RESET (TAB);
REWRITE(FINAL);
WHILE NOT EOF(NUEVO) AND NOT EOF(TAB) DO
BEGIN
 read(TAB,EL,TI,LI,X[1],X[2],X[3],X[4],X[5],BAN);
 READ(NUEVO,AX,AY,BX,BY,CX,CY,DX,DY,EX,EY);
 write(final,el,ti,li,x[1],ax,ay,x[2],bx,by,x[3],cx,cy,x[4],
       dx,dy,x[5],ex,ey,ban);
END;
close(final);close(tab);close(nuevo);
END;(*GENERA*)

(* PROGRAMA PRINCIPAL *)
BEGIN
GRAPHICS;
ASSIGN(FINAL,'FINAL.DOS');
ASSIGN(NUEVO,'NUEVO.DOS');
ASSIGN(TAB,'TAB.DOS');
assign(VALOR,'VALOR.pro');
DIBU;
GENERA;
LINEAS;
END.
```

CAPITULO 6

CONCLUSIONES

CONCLUSIONES.

En la actualidad se han desarrollado un gran número de paquetes de software cuya finalidad es impulsar el desarrollo del hardware, sin embargo, en la mayoría de los casos, el software existente no cubre algunas necesidades de los usuarios (diseñadores de hardware), esto es, hemos visto que algunos programas solo dibujan los diagramas de los circuitos pero no los simulan y, por el contrario existen otros que simulan el funcionamiento de los circuitos pero no dibujan los diagramas, es por esto que se pensó en la creación de un conjunto de programas que pudiera realizar estas dos funciones, es decir, un paquete de software que le permitiera al usuario visualizar de una forma sencilla el funcionamiento del circuito diseñado, y que además le dibuje el diagrama correspondiente. Al conjunto de programas obtenidos le hemos llamado PROSIC.

PROSIC permite a las personas que diseñan circuitos digitales, comprobar el funcionamiento de su diseño sin tener que estar en un laboratorio de electrónica y, obviamente, sin contar con los elementos físicos que conforman al circuito.

Por lo anteriormente expuesto y por los resultados obtenidos después de haber utilizado PROSIC podemos concluir lo siguiente:

- Que PROSIC es una herramienta muy útil para estudiantes y personas que trabajen en el área del Diseño Lógico, o simplemente para usuarios que quieran diseñar un circuito digital.

- Que PROSIC permite realizar diseños a bajo costo pues los resultados obtenidos después de haber corrido el paquete, permiten que el usuario utilice sólo el material necesario.

- Que es bastante accesible ya que su manejo es sencillo y los resultados son de fácil interpretación.

- Que PROSIC puede ser la base para diseñar elementos que en conjunto pueden formar un dispositivo bastante complicado.

- Que cuando un usario èsta utilizando PROSIC èsta visualizando de una forma mas objetiva el circuito que diseñò.

En sintesis un paquete de software como PROSIC impulsa el desarrollo del hardware, esto es, ayuda a desarrollar de forma mas ràpida dispositivos que a su vez impulsan el desarrollo de otras àreas tales como las ciencias, la la mèdicina, etc.

EXTENSIONES.

Las partes o mòdulos que pueden ser agregados a PROSIC para tener un mejor funcionamiento son los siguientes:

- Mòdulo que permita almacenar los circuitos simulados, para que despuès puedan ser, a su vez, utilizados en la simulaciòn de circuitos mas complicados, de esta forma se podrian almacenar circuitos como contadores, sumadores, conmutadores, etc., que despuès podrian ser utilizados en la simulaciòn de circuitos màs complicados como relojes digitales,calculadoras, etc., de èsta forma el usuario podria tener almacenados los elementos suficientes para realizar simulaciones de los diseños que desee.

- Mòdulo que permita representar los elementos almacenados como "cajas negras", de èsta forma, la simulacion se veria màs clara y no tendria que ver todo el alambrado en el mismo diagrama, esto es, el diagrama que PROSIC generaria seria un diagrama de bloques.

- Mòdulo que permita que el usuario pueda utilizar PROSIC sin necesidad de invocar antes al TURBOPASCAL, este mòdulo haria una funciòn similar a la de un compilador, pero solamente para las necesidades de PROSIC.

APENDICE A

M A N U A L

D E L

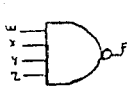
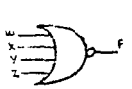
U S U A R I O

MANUAL DEL USUARIO.

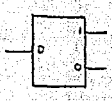
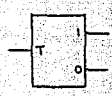
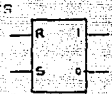
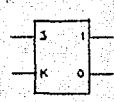
INDICE

1.- CARACTERISTICAS DE LOS COMPONENTES QUE PUEDEN SER UTILIZADOS.

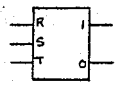
COMPUERTAS

TIPO	SIMBOLO	FUNCION	NOMBRE DEL SUBPROGRAMA	NUM. MAXIMO DE ENTRADAS
AND		XY F 00 0 01 0 10 0 11 1	CAND(W,X,Y,Z,F)	4
OR		XY F 00 0 01 1 10 1 11 1	COR(W,X,Y,Z,F)	4
NOT		X F 0 1 1 0	CNOT(X,F)	1
NAND		XY F 00 1 01 1 10 1 11 0	CNAND(W,X,Y,Z,F)	4
NOR		XY F 00 1 01 0 10 0 11 0	CNOR(W,X,Y,Z,F)	4

FLIP FLOP

TIPO	SIMBOLO	FUNCION	NOMBRE DEL SUBPROGRAMA	COMENTARIO
D		QD Q+ 00 0 01 1 10 0 11 1	DELAY(D,Q1,Q0)	Los datos se transfieren a la salida durante la transición de BAJO a ALTO del pulso de reloj.
T		QT Q+ 00 0 01 1 10 1 11 0	TRIGGER(T,Q1,Q0)	Cambia sólo para la transición de BAJO a ALTO del pulso de reloj.
RS		QRS Q+ 000 0 001 1 010 0 011 * 100 1 101 1 110 0 111 *	RS(R,S,Q1,Q0)	Cambia sólo a las bajadas del pulso de reloj. Cuando S=R=1,Q=0 con 55% de probabilidad.
JK		QJK Q+ 000 0 001 0 010 1 011 1 100 1 101 0 110 1 111 0	JK(J,K,Q1,Q0)	Cambia sólo a las bajadas del pulso de reloj.

RST



QRST	Q*	RST(R,S,T,Q1,Q0)
0000	0	
0001	1	
0010	1	
0011	*	
0100	0	
0101	*	
0110	*	
0111	*	
1000	1	
1001	0	
1010	1	
1011	*	
1100	0	
1101	*	
1110	*	
1111	*	

Raliza la funciòn de un F-F RS con un T=0 cuando el reloj=1, y de un F-F T con un RS con R=S=0 cuando el reloj=0.

Nota : Q = salida en t; Q* = salida en t+1

2.- EQUIPO REQUERIDO

Cualquier Microcomputadora IBM/PC o compatible con un mínimo de 256 kb de RAM y monitor de alta resolución.

Impresora que pueda ser conectada a la microcomputadora y que pueda imprimir gráficas.

La versión del Sistema Operativo MS-DOS puede ser cualquiera.

3.- COMO UTILIZAR EL PAQUETE PROSIC.

Para poder utilizar PROSIC, el usuario debe tener diseñado el circuito a simular.

Ahora bien, para exlicar la forma de utilizar PROSIC vamos a tomar como ejemplo un circuito que hace la función de un conmutador (SWITCH), siguiendo los siguientes pasos:

A.- Dibujar el circuito, como una caja negra, y numerar primeramente las entradas, y después las salidas, como se muestra en la figura 3.1. Las entradas y salidas del circuito las denominaremos Conexiones Externas.

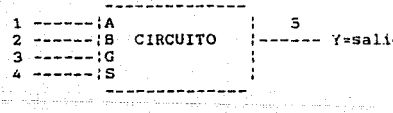


FIGURA 3.1

B.- Dibujar el circuito y en base al número asignado a la última conexión externa (EXT), numerar las conexiones internas, de ésta forma la primera conexión interna tendrá el número "N", donde N es el número de la última conexión externa más uno (EXT+1); la segunda "N+1", etc., como se puede observar en la figura 3.2.

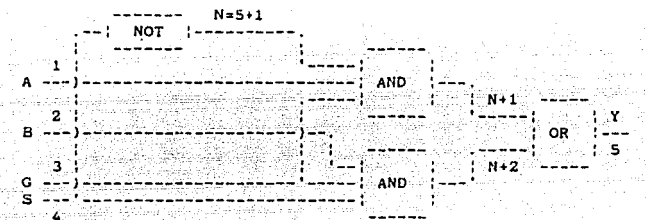


FIGURA 3.2

C.- Programación del circuito en la subrutina CIRCUITO. Esta subrutina es codificada por el usuario con instrucciones del lenguaje de programación PASCAL o TURBO PASCAL.

En ésta subrutina se hacen las llamadas a las subrutinas que simulan los componentes que se van a utilizar, cada llamada de subrutina llevará los parametros de entrada(s) y salida(s) correspondientes para cada componente. Para el circuito ilustrado en la figura 3.2 la subrutina CIRCUITO es:

```
PROCEDURE CIRCUITO;
VAR A,B,G,S,Y,N: INTEGER;
PROCEDURE SWITCH(A,B,G,S,Y,N: INTEGER);
BEGIN
      CNOT(S,N);
      CAND(A,N,G,0,N+1);
      CAND(G,S,B,0,N+2);
      COR(N+1,N+2,0,0,Y);
END;
SWITCH(1,2,3,4,5,6);
END (* CIRCUITO *);
```

D.- Programación de las salidas del circuito mediante la subrutina SALIDA.

El usuario podrá especificar en ésta subrutina, en que forma quiere la salida obtenida por el circuito; si se quiere la salida tal como la da el circuito, la subrutina SALIDA consistirá simplemente en la llamada a la subrutina LLENADO. Siguiendo el ejemplo de la figura 3.2 la subrutina SALIDA será la siguiente:

```
PROCEDURE SALIDA(VAR SV: INTEGER);
BEGIN
     LLENADO;
END (* SALIDA *);
```

E.- Programación de las entradas mediante la subrutina ENTRADA.

En ésta subrutina el usuario podrá especificar en que ALAMBRES estan las entradas, también podrá especificar durante que secuencia de tiempo quiere la simulación. Para el circuito de la figura 3.2 especificaremos algunas posibles entradas y una secuencia de nueve ciclos de reloj. La subrutina ENTRADA será la siguiente:

```
PROCEDURE ENTRADA(VAR E1,E2,E3,E4,E5,E6,E7,E8:INTEGER);
BEGIN
     IF T=0 THEN
     BEGIN
          ALAMBRE[1]:= TRUE;
          ALAMBRE[2]:= FALSE;
          ALAMBRE[3]:= FALSE;
          ALAMBRE[4]:= FALSE;
     END;
     IF T<=8 THEN
     BEGIN
          ALAMBRE[1]:= NOT ALAMBRE[2];
          ALAMBRE[2]:= NOT ALAMBRE[3];
          ALAMBRE[3]:= NOT ALAMBRE[1];
          ALAMBRE[4]:= NOT ALAMBRE[2];
     END
     ELSE
          FIN:= TRUE;
END(* ENTRADA *);
```

F.- Realizados los pasos anteriores corremos el paquete PROSIC de la siguiente forma:

a) Cuando definamos ,o ubiquemos, el subdirectorio donde se encuentre el paquete se procedera al llamado del compilador TURBO-PASCAL escribiendo, por ejemplo:

C>TURBO <RETURN>

e inmediatamente después se deberá responder la pregunta que hace dicho compilador, de que si carga a memoria principal su manejador de errores, la cual deberemos responder afirmativamente.

b) Posteriormente, el compilador nos despliega otra pantalla, mostrándonos una serie de comandos opcionales de los cuales deberemos seleccionar "W" (Workfile: el archivo de trabajo que emplearemos), y procederemos a invocar al programa SIMULA.

c) Tecleamos "R" (para ordenar su ejecución).

d) Si no hubo error, procedemos de forma similar al inciso (b), invocando ahora al programa PREPARA y repetimos el inciso (c).

e. Repetir el inciso (d) para correr el programa DIBUJO.

4.- COMO HACER LAS SUBRUTINAS CIRCUITO, ENTRADA Y SALIDA.

Para codificar las subrutinas CIRCUITO, ENTRADA y SALIDA, el usuario puede utilizar el editor de TURBO PASCAL (recomendable).

La subrutinas CIRCUITO, ENTRADA y SALIDAD deben estar contenidas en archivos cuyos nombres sean CIRCUITO.PRO, ENTRADA.PRO y SALIDA.PRO respectivamente, de no ser así PROSIC no reconocerá dichas subrutinas.

4.1.- Diseño de la subrutina CIRCUITO.

El encabezado de CIRCUITO debe ser el siguiente:

```
PROCEDURE CIRCUITO;
```

Nótese que ésta declaración no debe tener parámetros.

A continuación y en caso de ser necesario se declararán las variables LOCALES que se vayan a utilizar. En el ejemplo que se está siguiendo las variables locales se declaran de la siguiente manera:

```
VAR A,B,G,S,Y,N:INTEGER;
```

Donde las variables son: A,B,G,S,Y,N

DEFINICION DE LAS PARTES DEL CIRCUITO.

Las siguientes líneas muestran la subrutina. que contendrá las llamadas a varias subrutinas, éstas simulan a los elementos que van a ser utilizados. Para el circuito de la figura 3.2.

```
PROCEDURE SWITCH(A,B,G,S,Y,N: INTEGER);
BEGIN
     CNOT(S,N);
     CAND(A,N,G,0,N+1);
     CAND(G,S,B,0,N+2);
     COR(N+1,N+2,0,0,Y);
END;
```

Nótese que en el encabezado del PROCEDURE SWITCH son declarados los parámetros que van a ser utilizados durante la simulación. Se debe recordar que estos parámetros son los nombres de las conexiones del circuito, como se muestra en la figura 3.2. También se debe notar que en la llamada de cada subrutina que simula un elemento, se especifican los parámetros de entrada/salida correspondientes; cuando una entrada o salida no es utilizada su valor será cero.

PROGRAMA PRINCIPAL DE LA SUBRUTINA CIRCUITO.

En ésta parte se llama(n) a la(s) subrutina(s) que conforman al circuito que se va a simular. Volviendo al ejemplo de la figura 3.2:

```
BEGIN
     SWITCH(1,2,3,4,5,6);
END (* CIRCUITO *);
```

4.2.- Diseño de la subrutina ENTRADA.

ENCABEZADO

El encabezado de la subrutina ENTRADA debe ser el siguiente:

```
PROCEDURE ENTRADA(VAR E1,E2,E3,E4,E5,E6,E7,E8:INTEGER);
```

Los parámetros especificados aquí siempre deberán escribirse aunque no sean utilizados, estos parámetros son utilizados cuando el usuario quiere especificar entradas especiales (ver en el apéndice de ejemplos el sumador serial).

CONTENIDO DE LA SUBRUTINA ENTRADA.

En ésta parte se definen las entradas. Los valores correspondientes a cada entrada del circuito serán almacenados en el arreglo ALAMBRE. El usuario puede definir las variables locales que vaya a utilizar, es decir aquellas variables que no son parámetros reservados de las subrutinas en cuestión pero que son necesarias para desarrollo interno o "local".

Volviendo al ejemplo de la figura 3.2:

```
BEGIN
     IF T=0 THEN
     BEGIN
          ALAMBRE[1]:=TRUE;
          ALAMBRE[2]:=FALSE;
          ALAMBRE[3]:=FALSE;
          ALAMBRE[4]:=FALSE;
     END;
```

En ésta sección de la subrutina estamos inicializando los valores de las entradas para el tiempo T=0.

```
IF T<=8 THEN
BEGIN
     ALAMBRE[1]:=NOT ALAMBRE[2];
     ALAMBRE[2]:=NOT ALAMBRE[3];
     ALAMBRE[3]:=NOT ALAMBRE[1];
     ALAMBRE[4]:=NOT ALAMBRE[2];
END
ELSE
     FIN:=TRUE;
END; (* ENTRADA *)
```

Después hacemos que los valores correspondientes a cada alambre vayan variando durante un periodo de tiempo definido y cerrado, en éste caso para T<=8, T se incrementa en SIMULA. Se debe asignar a la variable FIN el valor de TRUE cuando el periodo de tiempo se ha cumplido, esto le indica a SIMULA que la ejecución de ésta subrutina a llegado a su fin.

4.3.- Diseño de la subrutina SALIDA.

ENCABEZADO.

El encabezado de la subrutina SALIDA debe ser el siguiente:

```
PROCEDURE SALIDA(VAR SV:INTEGER);
```

El parámetro SV debe ser especificado aunque no se esté utilizando; éste parámetro es empleado cuando el usuario quiera una salida especial (ver apéndice de ejemplos).

CONTENIDO DE LA SUBRUTINA SALIDA.

El usuario puede definir las variables locales que necesite.

En ésta subrutina el usuario especifica en que forma quiere las salidas (ver apéndice de ejemplos). Si el usuario desea las salidas tal como las da el paquete (en unos y ceros), únicamente tiene que invocar a LLENADO.

Esto se puede ver claramente en la subrutina SALIDA del ejemplo de la figura 3.2.

```
PROCEDURE SALIDA (VAR SV:INTEGER);
BEGIN
    LLENADO;
END; (* SALIDA *)
```

5.- INTERPRETACION DE RESULTADOS.

Una vez que hemos invocado a SIMULA, se iniciará la ejecución de la simulación del circuito. Si existe algún error SIMULA mandará un mensaje en el que indicará que tipo de error fue encontrado y desplegará un mensaje con la explicación de la causa del error, y terminará ahí el proceso de simulación; en caso de no existir error SIMULA desplegará dos tablas de resultados.

La primer tabla es mostrada a continuación:

	TIPO	LIGA		CONEXIONES				
1	NOT	2	4	0	0	0	6	INICIO
2	AND	3	1	6	3	0	7	
3	AND	4	3	4	2	0	8	
4	OR	0	7	8	0	0	5	

Fig. 5.1 Tabla obtenida para el ejemplo de la fig. 3.2

La figura 5.1 muestra la tabla que se obtuvo para el circuito de la figura 3.2 . Los datos contenidos en ésta tabla son:

ELEMENTO : Número del componente (en el orden en que fue llamado en la subrutina CIRCUITO) .

TIPO : Tipo de elemento utilizado.

LIGA : Siguiente elemento a ejecutarse en la secuencia lógica-electrónica del circuito.

CONEXIONES: Conexiones de cada elemento, esto es, el número de alambre asignado a las entradas y/o salidas de cada componente.

INICIO : Bandera que indica en donde se inicia la ejecución lógica-electrónica del circuito.

La segunda tabla, desplegada por SIMULA, es mostrada en la figura 5.2 .

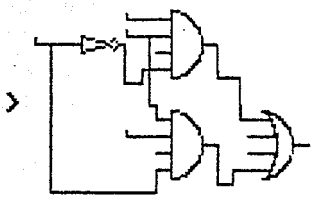
Esta tabla nos indica el comportamiento de los componentes durante la ejecución del circuito.

TIEMPO	ALAMBRADO							
0	1	1	0	0	0	1	0	0
1	0	1	1	0	0	1	0	0
2	0	0	1	1	0	0	0	0
3	1	0	0	1	0	0	0	0
4	1	1	0	0	0	1	0	0
5	0	1	1	0	0	1	0	0
6	0	0	1	1	0	0	0	0
7	1	0	0	1	0	0	0	0
8	1	1	0	0	0	1	0	0

Fig 5.2 Tabla obtenida para el circuito de la fig 3.2

La Tabla mostrada en la figura 5.2 nos indica los estados lógicos de las conexiones de cada elemento del circuito durante el ciclo de tiempo correspondiente. Las COLUMNAS de ésta tabla corresponden a las conexiones o alambres, y los RENGLONES indican el ciclo correspondiente, esto es, la primer columna nos indica el comportamiento del ALAMBRE no. 1, la segunda el comportamiento del ALAMBRE no. 2, etc. En la tabla de la fig. 5.2 se muestran los estados lógicos de las conexiones del circuito para 9 ciclos de tiempo, debemos recordar que la secuencia para la cual queremos que se simule el circuito, es dada en la subrutina ENTRADA.

Después de desplegar las dos tablas anteriores, PROSIC procesa la información obtenida y finalmente dibuja el diagrama del circuito, como se muestra a continuación:



6.- PROGRAMACION POR MODULOS.

En ésta sección se explica como programar por modulos, lo cual facilitará el diseño de circuitos lógicos, para su posterior simulación.

Para hacer la explicación más ilustrativa, se va a tomar como ejemplo un sumador serial. Primeramente se deberá numerar las entradas y salidas del circuito (en ese orden, como se explicó al inicio de éste manual).

6.1.- Un sumador serial está compuesto de un sumador completo y un flip-flop tipo D como se muestra en el siguiente diagrama:

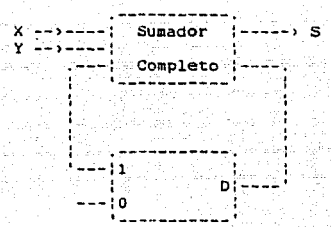


Figura 6.1

6.2.- El sumador completo se representa de la siguiente forma:

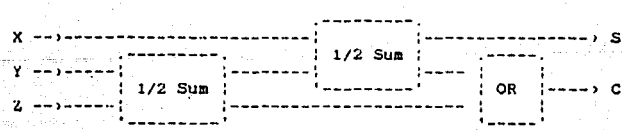


Figura 6.2

6.2.- El medio sumador se puede representar de la siguiente forma:

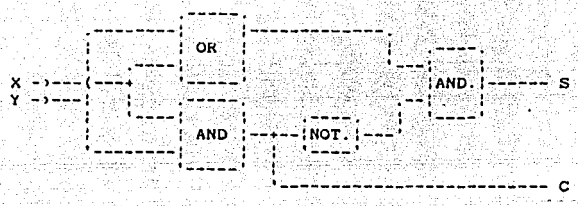


Figura 6.3

En base a los módulos diseñados en los puntos anteriores, ahora se hacen las subrutinas correspondientes:

Para el circuito de la figura 6.3 el programa es el siguiente:

```
PROCEDURE MEDIOSUMADOR(X,Y,S,C,N:INTEGER);
BEGIN
      COR(X,Y,0,0,N);
      CAND(N,N+1,0,0,S);
      CAND(X,Y,0,0,C);
      CNOT(C,N+1);
 END(* MEDIO SUMADOR *);
```

Para el circuito de la figura 6.2 el programa es el siguiente:

```
                    SV:=SV;
                END;
            IF T=12 THEN
               WRITE(' ':30,SV);
    END(' SALIDA ');
```

Ver los resultados de este ejemplo en el apendice de ejemplos, donde es mostrado.

APENDICE B

T A B L A

D E

E R R O R E S

TABLA DE ERRORES.

ERROR NUM.	EXPLICACION
1	El componente número "N" tiene parametros mayores que el número de conexiones.
2	La entrada externa "N" aparece también como salida .
3	El Alambre "N" aparece más de una vez como salida de un componente básico.
4	Se estan utilizando más Componentes Básicos de los indicados al inicio del programa.
5	Se estan utilizando más Flip-Flops de los indicados al inicio del programa.
6	El número de conexiones utilizadas "NC" es menor que las especificadas.
7	El número de Componentes Básicos utilizados es es menor que el especificado.
8	El número de Flip-Flops utilizados es menor que el especificado.
9	El Alambre "N" no es salida de un componente Básico.
10	No es válido el sucesor en la secuencia computacional, para el componente básico en la posición "N".
11	El elemento RS con el número "N", tiene dos UNOS en la entrada.
12	El elemento RST con el número "N",tiene dos UNOS en la entrada.

APENDICE C

E J E M P L O S

EJEMPLO NUMERO 1.- SUMADOR SERIAL.

Archivo : ENTRADA.PRO

```
PROCEDURE ENTRADA(VAR SUM1,SUM2,e3,e4,e5,e6,e7,e8:INTEGER);
    VAR X:INTEGER;
    BEGIN
        IF T=0 THEN
           BEGIN
                SUM1:=1066;
                SUM2:=1971;
           END;
        IF T<=12 THEN
           BEGIN
                X:=TRUNC(SUM1/2);
                ALAMBRE[1]:=SUM1<>2*X;
                SUM1:=X;
                X:=TRUNC(SUM2/2);
                ALAMBRE[2]:=SUM2<>2*X;
                SUM2:=X;
            END
        ELSE
            FIN:=TRUE;
      END{* entrada *};
```

Archivo : CIRCUITO.PRO

```
PROCEDURE CIRCUITO;
    PROCEDURE MEDIOSUMADOR(X,Y,S,C,N:INTEGER);
              BEGIN
                   COR(X,Y,0,0,N);
                   CAND(N,N+1,0,0,S);
                   CAND(X,Y,0,0,C);
                   CNOT(C,N+1);
               END{* MEDIOSUMADOR *};
     PROCEDURE SUMADORCOMPLETO(X,Y,Z,S,C,N:INTEGER);
               BEGIN
                    MEDIOSUMADOR(X,N,S,N+2,N+3);
                    MEDIOSUMADOR(Y,Z,N,N+1,N+5);
                    COR(N+1,N+2,0,0,C);
               END;
```

```
PROCEDURE SUMADORSERIE(X,Y,S,N:INTEGER);
          BEGIN
               SUMADORCOMPLETO(X,Y,N,S,N+1,N+2);
               DELAY(N+1,N,0);
          END(' SUMADORSERIE ');
 BEGIN
     SUMADORSERIE(1,2,3,4);
 END(' FIN DE CIRCUITO ');
```

Archivo : SALIDA.PRO

```
PROCEDURE SALIDA(VAR SV:INTEGER);
     VAR AUB:INTEGER;
       E,S:REAL;
      BEGIN
          LLENADO;
          IF T=0 THEN
            BEGIN
                 IF ALAMBRE[3] THEN
                   SV:=1
                 ELSE
                    SV:=0;
            END
            ELSE
                BEGIN
                    IF ALAMBRE[3] THEN
                      BEGIN
                           S:=0.6931472;
                           AUB:=TRUNC(EXP((S)*T));
                           SV:=SV+AUB;
                      END
                     ELSE
                        SV:=SV;
                END;
             IF T=12 THEN
                WRITE(' ':30,SV);
     END(' SALIDA ');
```

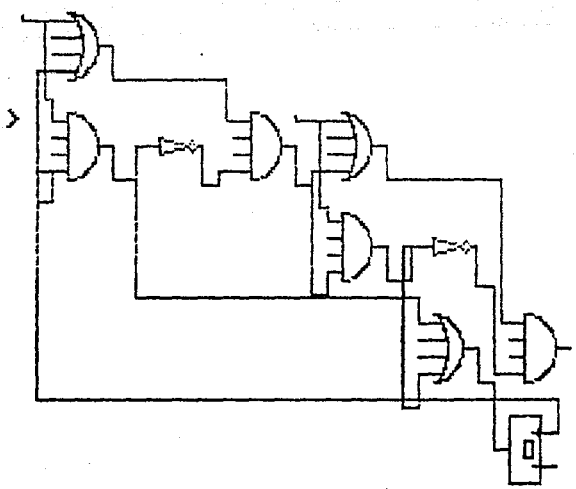
A continuación presentamos los mensajes desplegados en pantalla y los resultados obtenidos.

```
Deseas cambiar la constantes almacenadas en la simulacion anterior ??
            SI: teclea [1],   NO : teclea [0]  ==>   1


    DAME LOS SIGUIENTES VALORES :

    CCOMP (Numero de componentes)  ==> 10

    CFF   (Numero de Flip-Flops )  ==> 1

    CENT  (Numero de entradas    )  ==> 2

    CALAMB (Numero de conexiones)  ==> 12
```

	TIPO	LIGA		CONEXIONES				
1	OR	3	1	6	0	0	9	
2	AND	10	9	10	0	0	3	
3	AND	4	1	6	0	0	8	
4	NOT	9	8	0	0	0	10	
5	OR	7	2	4	0	0	11	INICIO
6	AND	1	11	12	0	0	6	
7	AND	8	2	4	0	0	7	
8	NOT	6	7	0	0	0	12	
9	OR	2	7	8	0	0	5	
10	D	0	5	0	0	4	0	

TIEMPO	ALAMBRADO											
0	0	1	1	0	0	1	0	0	1	1	1	1
1	1	1	0	0	1	1	0	1	1	0	1	1
2	0	0	1	1	0	1	0	0	1	1	1	1
3	1	0	1	0	0	0	0	0	1	1	0	1
4	0	1	1	0	0	1	0	0	1	1	1	1
5	1	1	0	0	1	1	0	1	1	0	1	1
6	0	0	1	1	0	1	0	0	1	1	1	1
7	0	1	1	0	0	1	0	0	1	1	1	1
8	0	1	1	0	0	1	0	0	1	1	1	1
9	0	1	1	0	0	1	0	0	1	1	1	1
10	1	1	0	0	1	1	0	1	1	0	1	1
11	0	0	1	1	0	1	0	0	1	1	1	1
12	0	0	0	0	0	0	0	0	0	1	0	1

FIN DE SIMULA ********



EJEMPLO NUMERO 2.- CONTADOR.

Archivo : ENTRADA.PRO

```
PROCEDURE ENTRADA(VAR E1,E2,E3,E4,E5,E6,E7,E8:INTEG
   BEGIN
       IF T<=9 THEN
         ALAMBRE[1]:=TRUE
       ELSE
         FIN:=TRUE;
    END;
```

Archivo : CIRCUITO.PRO

```
PROCEDURE CIRCUITO;
        VAR X,Q0,Q1,Q2,Q3,Q4:INTEGER;
              PROCEDURE CONTADOR(X:INTEGER);
              BEGIN
                  TRIGGER(X,X+1,0);
                  TRIGGER(X+1,X+2,0);
                  TRIGGER(X+6,X+3,0);
                  TRIGGER(X+5+2,X+4,0);
                  TRIGGER(X+5+3,X+5,0);
                  CAND(X+1,X+2,0,0,X+5+1);
                  CAND(X+5+1,X+3,0,0,X+5+2);
                  CAND(X+4,X+5+2,0,0,X+5+3);
              END;
              BEGIN
                  CONTADOR(1);
              END;
```

Archivo : SALIDA.PRO

```
PROCEDURE SALIDA(VAR SV:INTEGER);
            BEGIN
                LLENADO;
            END;
```

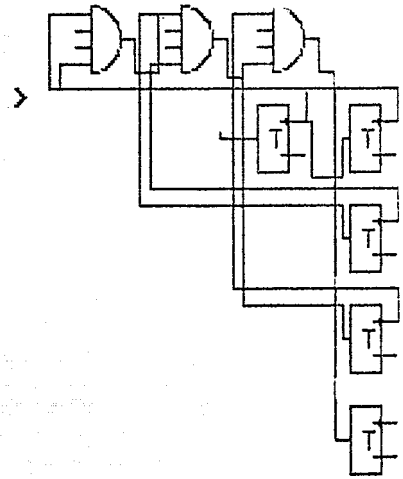
A continuación presentamos los mensajes desplegados en pantalla y los resultados obtenidos.

```
Deseas cambiar la constantes almacenadas en la simulacion anterior ??
          SI: teclea [1],   NO : teclea [0]  ==>   1




     DAME LOS SIGUIENTES VALORES :

     CCOMP  (Numero de componentes)  ==> 8

     CFF    (Numero de Flip-Flops )  ==> 5

     CENT   (Numero de entradas    )  ==> 1

     CALAMB (Numero de conexiones)   ==> 9
```

	TIPO	LIGA		CONEXIONES				
1	T	2	1	0	0	2	0	
2	T	3	2	0	0	3	0	
3	T	4	7	0	0	4	0	
4	T	5	8	0	0	5	0	
5	T	0	9	0	0	6	0	
6	AND	7	2	3	0	0	7	INICIO
7	AND	8	7	4	0	0	8	
8	AND	1	5	8	0	0	9	

TIEMPO	ALAMBRADO								
0	1	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0
2	1	0	1	0	0	0	0	0	0
3	1	1	1	0	0	0	1	0	0
4	1	0	0	1	0	0	0	0	0
5	1	1	0	1	0	0	0	0	0
6	1	0	1	1	0	0	0	0	0
7	1	1	1	1	0	0	1	1	0
8	1	0	0	0	1	0	0	0	0
9	1	1	0	0	1	0	0	0	0

FIN DE SIMULA ********


T
T
T
T
T

EJEMPLO NUMERO 3.- COMPARADOR.

Archivo : ENTRADA.PRO

```
PROCEDURE ENTRADA(VAR A,B,C,D,E,F,G,H:INTEGER);
BEGIN
 IF T<=9 THEN
 BEGIN
      IF T=0 THEN
      BEGIN
           ALAMBRE[1]:=FALSE;
           ALAMBRE[2]:=FALSE;
      END;
      IF T>0 THEN
      BEGIN
           ALAMBRE[1]:=NOT(ALAMBRE[1]);
           IF (T=2) OR (T=3) OR (T=6) OR (T=7) THEN
           ALAMBRE[2]:=TRUE
           ELSE
           ALAMBRE[2]:=FALSE;
       END;
      END
 ELSE
 FIN:=TRUE;
END;
```

Archivo : CIRCUITO.PRO

```
PROCEDURE CIRCUITO;
VAR A,B,C,D,E:INTEGER;
PROCEDURE COMPARADOR(A,B,C,D,E:INTEGER);
BEGIN
 CNOT(A,E+1);
 CNOT(B,E+2);
 CAND(E+1,B,0,0,E+3);
 CAND(A,E+2,0,0,E+4);
 CAND(A,E+2,0,0,C);
 CAND(E+1,B,0,0,D);
 CNOR(E+3,E+4,0,0,E);
END;
BEGIN
 COMPARADOR(1,2,3,4,5);
END;
```

Archivo : SALIDA.PRO

```
PROCEDURE SALIDA(VAR SV:INTEGER);
BEGIN
     LLENADO;
END;
```

A continuación presentamos los mensajes desplegados en pantalla y los resultados obtenidos.

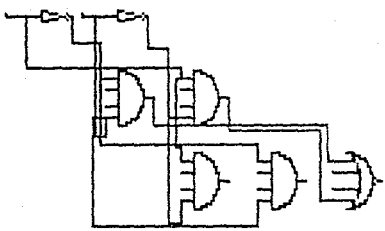
```
Deseas cambiar la constantes almacenadas en la simulacion anterior ??
            SI: teclea [1],   NO : teclea [0]  ==>   1



    DAME LOS SIGUIENTES VALORES :

    CCOMP  (Numero de componentes)  ==> 7
    CFF    (Numero de Flip-Flops )  ==> 0
    CENT   (Numero de entradas    )  ==> 2
    CALAMB (Numero de conexiones)   ==> 9
```

	TIPO	LIGA		CONEXIONES				
1	NOT	2	1	0	0	0	6	INICIO
2	NOT	3	2	0	0	0	7	
3	AND	4	6	2	0	0	8	
4	AND	5	1	7	0	0	9	
5	AND	6	1	7	0	0	3	
6	AND	7	5	2	0	0	4	
7	NOR	0	8	9	0	0	5	

TIEMPO	ALAMBRADO								
0	0	0	0	0	1	1	1	0	0
1	1	0	1	0	0	0	1	0	1
2	0	1	0	1	0	1	0	1	0
3	1	1	0	0	1	0	0	0	0
4	0	0	0	0	1	1	1	0	0
5	1	0	1	0	0	0	1	0	1
6	0	1	0	1	0	1	0	1	0
7	1	1	0	0	1	0	0	0	0
8	0	0	0	0	1	1	1	0	0
9	1	0	1	0	0	0	1	0	1

FIN DE SIMULA ********

>



EJEMPLO 4.-

Archivo : ENTRADA.PRO

```
PROCEDURE ENTRADA (VAR E1,E2,E3,E4,E5,E6,E7,E8:INTEGER);
BEGIN
 IF T<=12 THEN
 BEGIN
  ALAMBRE[1]:=TRUE;
  ALAMBRE[2]:=FALSE;
END
ELSE
 FIN:=TRUE;
END;
```

Archivo : CIRCUITO.PRO

```
PROCEDURE CIRCUITO;
VAR A,N:INTEGER;
PROCEDURE EJEMP4(N:INTEGER);
BEGIN
 CNAND(N,N+11,0,0,N+2);
 CNAND(N+1,N+9,0,0,N+7);
 CNAND(N+5,N+10,0,0,N+3);
 CNAND(N+9,N+5,0,0,N+6);
 CNAND(N+3,N+2,0,0,N+4);
 CNAND(N+7,N+6,0,0,N+8);
 DELAY(N+4,N+5,0);
 DELAY(N+8,N+9,0);
 CNOT(N+9,N+10);
 CNOT(N+9,N+11);
END;
BEGIN
     EJEMP4(1);
END;
```

Archivo : SALIDA.PRO

```
PROCEDURE SALIDA(VAR SV:INTEGER);
BEGIN
     LLENADO;
END;
```

Acontinuacion presentamos los mensajes desplegado en pantalla y los resultados obtenidos.

Deseas cambiar la constantes almacenadas en la simulacion anterior ??

SI: teclea [1], NO : teclea [0] ==> 1

DAME LOS SIGUIENTES VALORES :

CCOMP (Numero de componentes) ==> 10

CFF (Numero de Flip-Flops) ==> 2

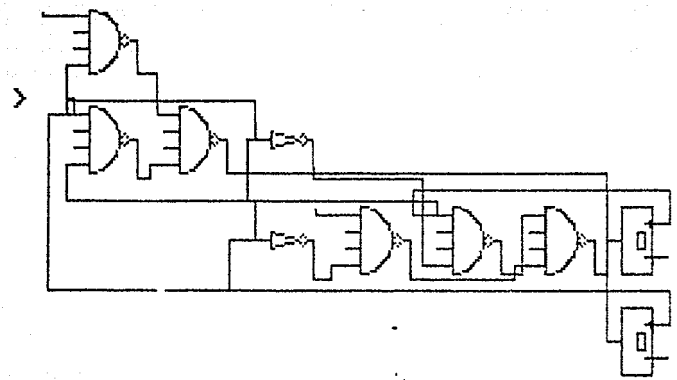
CENT (Numero de entradas) ==> 2

CALAMB (Numero de conexiones) ==> 12

	TIPO	LIGA		CONEXIONES				
1	NAND	3	1	12	0	0	3	
2	NAND	4	2	10	0	0	8	INICIO
3	NAND	5	6	11	0	0	4	
4	NAND	6	10	6	0	0	7	
5	NAND	7	4	3	0	0	5	
6	NAND	9	8	7	0	0	9	
7	D	8	5	0	0	6	0	
8	D	0	9	0	0	10	0	
9	NOT	10	10	0	0	0	11	
10	NOT	1	10	0	0	0	12	

TIEMPO	ALAMBRADO											
0	1	0	0	1	1	0	1	1	0	0	1	1
1	1	0	0	0	1	1	1	1	0	0	1	1
2	1	0	0	0	1	1	1	1	0	0	1	1
3	1	0	0	0	1	1	1	1	0	0	1	1
4	1	0	0	0	1	1	1	1	0	0	1	1
5	1	0	0	0	1	1	1	1	0	0	1	1
6	1	0	0	0	1	1	1	1	0	0	1	1
7	1	0	0	0	1	1	1	1	0	0	1	1
8	1	0	0	0	1	1	1	1	0	0	1	1
9	1	0	0	0	1	1	1	1	0	0	1	1
10	1	0	0	0	1	1	1	1	0	0	1	1
11	1	0	0	0	1	1	1	1	0	0	1	1
12	1	0	0	0	1	1	1	1	0	0	1	1

FIN DE SIMULA ********



EJEMPLO NUMERO 5 . SWITCH

Archivo ENTRADA.PRO

```
PROCEDURE ENTRADA(VAR E1,E2,E3,E4,E5,E6,E7,E8:INTEGER);
      BEGIN
           IF T=0 THEN
           BEGIN
                ALAMBRE[1]:= TRUE;
                ALAMBRE[2]:= FALSE;
                ALAMBRE[3]:= FALSE;
                ALAMBRE[4]:= FALSE;
           END;
           IF T<=8 THEN
           BEGIN
                 ALAMBRE[1]:= NOT ALAMBRE[2];
                 ALAMBRE[2]:= NOT ALAMBRE[3];
                 ALAMBRE[3]:= NOT ALAMBRE[1];
                 ALAMBRE[4]:= NOT ALAMBRE[2];
           END
           ELSE
               FIN:=TRUE;
       END(* ENTRADA *);
```

Archivo : CIRCUITO.PRO

```
PROCEDURE CIRCUITO;
VAR A,B,G,S,Y,N:INTEGER;
 PROCEDURE SWITCH(A,B,G,S,Y,N:INTEGER);
 BEGIN
   CNOT(S,1);
   CAND(A,N,G,0,N+1);
   CAND(G,S,B,0,N+2);
   COR(N+1,N+2,0,0,Y);
END;
 BEGIN
   SWITCH(1,2,3,4,5,6);
END (* CIRCUITO *);
```

Archivo :SALIDA.PRO

```
PROCEDURE SALIDA(VAR SV:INTEGER);
BEGIN
      LLENADO;
END (* SALIDA *);
```

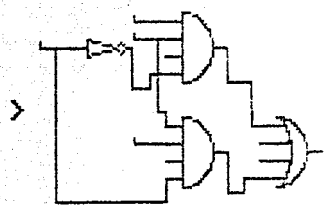
A continuación presentamos los mensajes desplegados en pantalla y los resultados.

```
Deseas cambiar la constantes almacenadas en la simulacion anterior ??
            SI: teclea [1].   NO : teclea [0]  ==>   1




    DAME LOS SIGUIENTES VALORES :

    CCOMP (Numero de componentes)  ==> 4

    CFF   (Numero de Flip-Flops )  ==> 0

    CENT  (Numero de entradas    )  ==> 4

    CALAMB (Numero de conexiones)  ==> 8
```

	TIPO	LIGA		CONEXIONES				
1	NOT	2	4	0	0	0	6	INICIO
2	AND	3	1	6	3	0	7	
3	AND	4	3	4	2	0	8	
4	OR	0	7	8	0	0	5	

TIEMPO	ALAMBRADO							
0	1	1	0	0	0	1	0	0
1	0	1	1	0	0	1	0	0
2	0	0	1	1	0	0	0	0
3	1	0	0	1	0	0	0	0
4	1	1	0	0	0	1	0	0
5	0	1	1	0	0	1	0	0
6	0	0	1	1	0	0	0	0
7	1	0	0	1	0	0	0	0
8	1	1	0	0	0	1	0	0

FIN DE SIMULA ********



EJEMPLO NUMERO 6

Archivo : ENTRADA.PRO

```
PROCEDURE ENTRADA(VAR A,B,C,D,E,F,G,H:INTEGER);
BEGIN
 IF T<=20 THEN
 BEGIN
      ALAMBRE[1]:=FALSE;
      ALAMBRE[2]:=TRUE;
      ALAMBRE[3]:=FALSE;
      ALAMBRE[4]:=TRUE;
  END
  ELSE
   FIN:=TRUE;
END;
```

Archivo : CIRCUITO.PRO

```
PROCEDURE CIRCUITO;
PROCEDURE PRU_FIN(A,B,C,D,N:INTEGER);
BEGIN
 CNOT(A,N+1);
 CNOT(B,N+2);
 CNOT(C,N+3);
 CNOT(D,N+4);
DELAY(N+1,N+5,N+6);
TRIGGER(N+2,N+7,N+8);
JK(N+2,N+3,N+9,N+10);
RS(N+3,N+4,N+11,N+12);
RST(N+2,N+3,N+4,N+13,0);
CAND(N+5,N+7,N+9,N+11,N+14);
COR(N+6,N+8,N+10,N+12,N+15);
CNAND(N+14,N+15,N+13,0,N);
END;
BEGIN
     PRU_FIN(1,2,3,4,5);
END;
```

Archivo : SALIDA.PRO

```
procedure salida(var sv:integer);
    .begin
        llenado;
     end;
```

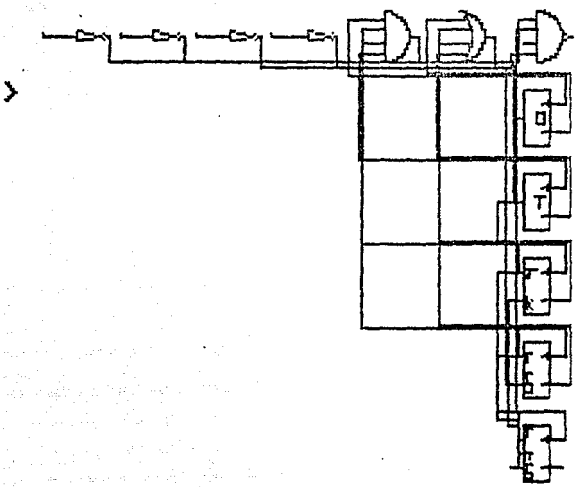
A continuación presentamos los mensajes deplegados en la pantalla y los resultados obtenidos.

```
Deseas cambiar la constantes almacenadas en la simulacion anterior ??
          SI: teclea [1],   NO : teclea [0]  ==>   1



     DAME LOS SIGUIENTES VALORES :

     CCOMP  (Numero de componentes)   ==> 12

     CFF    (Numero de Flip-Flops )   ==> 5

     CENT   (Numero de entradas    )  ==> 4

     CALAMB (Numero de conexiones)    ==> 20
```

	TIPO	LIGA		CONEXIONES				
1	NOT	2	1	0	0	0	6	INICIO
2	NOT	3	2	0	0	0	7	
3	NOT	4	3	0	0	0	8	
4	NOT	10	4	0	0	0	9	
5	D	6	6	0	0	10	11	
6	T	7	7	0	0	12	13	
7	JK	8	7	8	0	14	15	
8	RS	9	8	9	0	16	17	
9	RST	0	7	8	9	18	0	
10	AND	11	10	12	14	16	19	
11	OR	12	11	13	15	17	20	
12	NAND	5	19	20	18	0	5	

TIEMPO	ALAMBRADO																			
0	0	1	0	1	1	1	0	1	0	0	1	0	1	0	1	0	1	0	0	1
1	0	1	0	1	1	1	0	1	0	1	0	0	1	0	1	0	1	1	0	1
2	0	1	0	1	1	1	0	1	0	1	0	0	1	0	1	0	1	1	0	1
3	0	1	0	1	1	1	0	1	0	1	0	0	1	0	1	0	1	1	0	1
4	0	1	0	1	1	1	0	1	0	1	0	0	1	0	1	0	1	1	0	1
5	0	1	0	1	1	1	0	1	0	1	0	0	1	0	1	0	1	1	0	1
6	0	1	0	1	1	1	0	1	0	1	0	0	1	0	1	0	1	1	0	1
7	0	1	0	1	1	1	0	1	0	1	0	0	1	0	1	0	1	1	0	1
8	0	1	0	1	1	1	0	1	0	1	0	0	1	0	1	0	1	1	0	1
9	0	1	0	1	1	1	0	1	0	1	0	0	1	0	1	0	1	1	0	1
10	0	1	0	1	1	1	0	1	0	1	0	0	1	0	1	0	1	1	0	1
11	0	1	0	1	1	1	0	1	0	1	0	0	1	0	1	0	1	1	0	1
12	0	1	0	1	1	1	0	1	0	1	0	0	1	0	1	0	1	1	0	1
13	0	1	0	1	1	1	0	1	0	1	0	0	1	0	1	0	1	1	0	1
14	0	1	0	1	1	1	0	1	0	1	0	0	1	0	1	0	1	1	0	1
15	0	1	0	1	1	1	0	1	0	1	0	0	1	0	1	0	1	1	0	1
16	0	1	0	1	1	1	0	1	0	1	0	0	1	0	1	0	1	1	0	1
17	0	1	0	1	1	1	0	1	0	1	0	0	1	0	1	0	1	1	0	1
18	0	1	0	1	1	1	0	1	0	1	0	0	1	0	1	0	1	1	0	1
19	0	1	0	1	1	1	0	1	0	1	0	0	1	0	1	0	1	1	0	1
20	0	1	0	1	1	1	0	1	0	1	0	0	1	0	1	0	1	1	0	1

FIN DE SIMULA ********



APENDICE D

B I B L I O G R A F I A

BIBLIOGRAFIA.

- LOGICA DIGITAL Y DISE&O DE COMPUTADORES
  M. MORRIS MANO
  ED. PRENTICE HALL. MEX. 1984

- PRINCIPIOS DIGITALES
  ROGER L. TOKHEIM
  ED. MC. GRAW HILL. 1982

- TURBO PASCAL REFERENCE MANUAL (VERSION 3.0)
  BORLAND INTERNATIONAL
  ED. PRENTICE HALL COMPANY
  RESTON VIRGINIA

- THE ART OF DIGITAL DESIGN
  AND INTRODUCTION TO TOP-DOWN DESIGN
  DAVID WINKEL, FRANKLIN PROSSER
  ED. PRENTICE HALL, INC., ENGLEWOOD CLIFFS, NEW JERSEY, 1980

- GRAPHICS MADE EASY FOR THE IBM PC AND XT
  GABRIEL CUELLAR
  ED. PRENTICE HALL COMPANY
  RESTON VIRGINIA