**UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO**

**FACULTAD DE INGENIERÍA**

**LABORATORIO DE ORGANIZACIÓN Y ARQUITECTURA DE COMPUTADORAS**

**CLAVE 6867**

Profesor: M. I. José Luis Cruz Mora

Integrantes del equipo:

García Fernández Jesús Alejandro

Hernández Arrieta Carlos Alberto

Moreno Guerra Marco Antonio

**Reporte Práctica 3**

Fecha de entrega: 11 de septiembre del 2019

Grupo: 1

Semestre: 2020-1

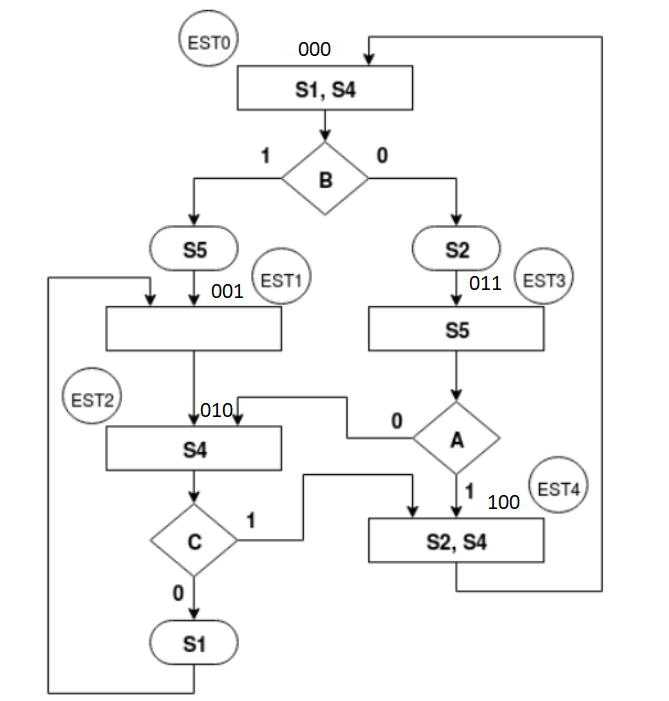
DIRECCIONAMIENTO POR TRAYECTORIA

OBJETIVO

Familiarizar al alumno en el conocimiento de construcción de máquinas de estado usando el direccionamiento de memorias con el método de direccionamiento por trayectoria.

DESARROLLO

A partir de la siguiente carta ASM se procedió a obtener la tabla del contenido de la memoria.

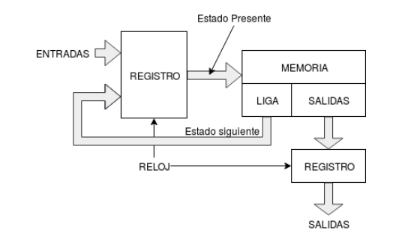


Carta ASM a resolver



Tabla del contenido de la memoria

Una vez obtenido el contenido de la memoria y tratándose del direccionamiento por trayectoria tenemos la siguiente arquitectura:



A partir de la arquitectura anterior y con ayuda de Quartus desarrollamos en VHDL cada uno de los componentes que conforman esta arquitectura de la siguiente manera:

La memoria se conforma del contenido obtenido en la tabla anterior:

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity memory is

    Port( dir : in std\_logic\_vector(5 downto 0);

            data : out std\_logic\_vector(6 downto 0)

            );

end memory;

architecture Behavioral of memory is

    type mem is array(0 to 63) of std\_logic\_vector(6 downto 0);

    signal internal\_mem : mem;

    begin

        -- ESTADO 0

        internal\_mem(0) <= "011" & "1110";

        internal\_mem(1) <= "011" & "1110";

        internal\_mem(2) <= "001" & "1011";

        internal\_mem(3) <= "001" & "1011";

        internal\_mem(4) <= "011" & "1110";

        internal\_mem(5) <= "011" & "1110";

        internal\_mem(6) <= "001" & "1011";

        internal\_mem(7) <= "001" & "1011";

        -- ESTADO 1

        internal\_mem(8) <= "010" & "0000";

        internal\_mem(9) <= "010" & "0000";

        internal\_mem(10) <= "010" & "0000";

        internal\_mem(11) <= "010" & "0000";

        internal\_mem(12) <= "010" & "0000";

        internal\_mem(13) <= "010" & "0000";

        internal\_mem(14) <= "010" & "0000";

        internal\_mem(15) <= "010" & "0000";

        -- ESTADO 2

        internal\_mem(16) <= "001" & "1010";

        internal\_mem(17) <= "100" & "0010";

        internal\_mem(18) <= "001" & "1010";

        internal\_mem(19) <= "100" & "0010";

        internal\_mem(20) <= "001" & "1010";

        internal\_mem(21) <= "100" & "0010";

        internal\_mem(22) <= "001" & "1010";

        internal\_mem(23) <= "100" & "0010";

        -- ESTADO 3

        internal\_mem(24) <= "010" & "0001";

        internal\_mem(25) <= "010" & "0001";

        internal\_mem(26) <= "010" & "0001";

        internal\_mem(27) <= "010" & "0001";

        internal\_mem(28) <= "100" & "0001";

        internal\_mem(29) <= "100" & "0001";

        internal\_mem(30) <= "100" & "0001";

        internal\_mem(31) <= "100" & "0001";

            -- ESTADO 4

        internal\_mem(32) <= "000" & "0110";

        internal\_mem(33) <= "000" & "0110";

        internal\_mem(34) <= "000" & "0110";

        internal\_mem(35) <= "000" & "0110";

        internal\_mem(36) <= "000" & "0110";

        internal\_mem(37) <= "000" & "0110";

        internal\_mem(38) <= "000" & "0110";

        internal\_mem(39) <= "000" & "0110";

        process(dir)

            begin

                data <= internal\_mem(conv\_integer(unsigned(dir)));

        end process;

end Behavioral;