CODIGO 1:

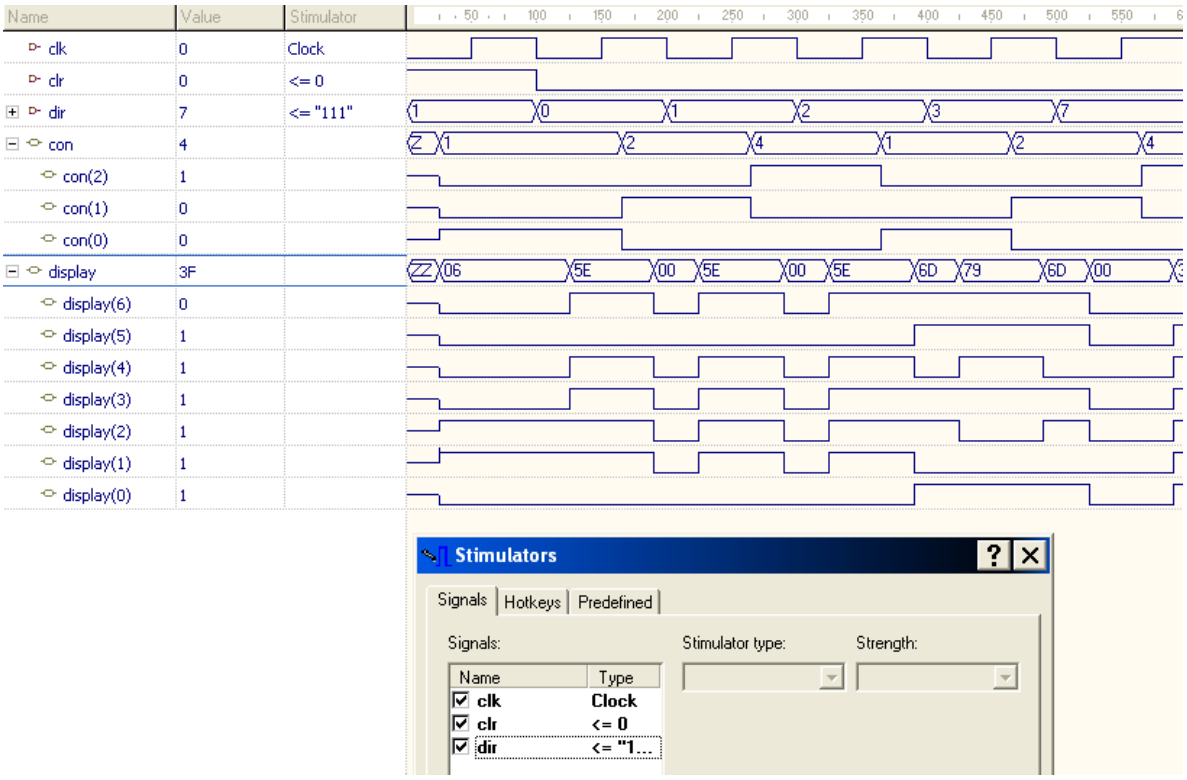
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity pract10A is port (
    clk, clr: in std_logic;
    dir: in std_logic_vector(2 downto 0);
    con: inout std_logic_vector(2 downto 0);
    display: out std_logic_vector(6 downto 0)
);
end entity;

architecture apract10A of pract10A is
type matrix is array(0 to 7) of std_logic_vector(6 downto 0);
signal anodos: std_logic_vector(2 downto 0);
constant rom0: matrix:= (
    "1011110", --d
    "0000110", --I
    "1101101", --S
    "1111001", --E
    "1010101", --n
    "0111111", --O
    "0000000",
    "0000000"
);
constant rom1: matrix:= (
    "0000000",
    "1011110", --d
    "0000110", --I
    "1101101", --S
    "1111001", --E
    "1010101", --n
    "0111111", --O
    "0000000"
);
constant rom2: matrix:= (
    "0000000",
    "0000000",
    "1011110", --d
    "0000110", --I
    "1101101", --S
    "1111001", --E
    "1010101", --n
```

```
    "0111111" --0
);
begin
    process(clk, clr) begin
        if(clr = '1') then anodos <= "001";
        elsif(rising_edge(clk)) then
            case anodos is
                when "001" => anodos <= "010";
                when "010" => anodos <= "100";
                when "100" => anodos <= "001";
                when others => anodos <= "001";
            end case;
        end if;
    end process;
    con <= anodos;
    display <= rom0(conv_integer(dir)) when (con = "001")
    else rom1(conv_integer(dir)) when (con = "010")
    else rom2(conv_integer(dir));
end architecture;
```

SIMULACION



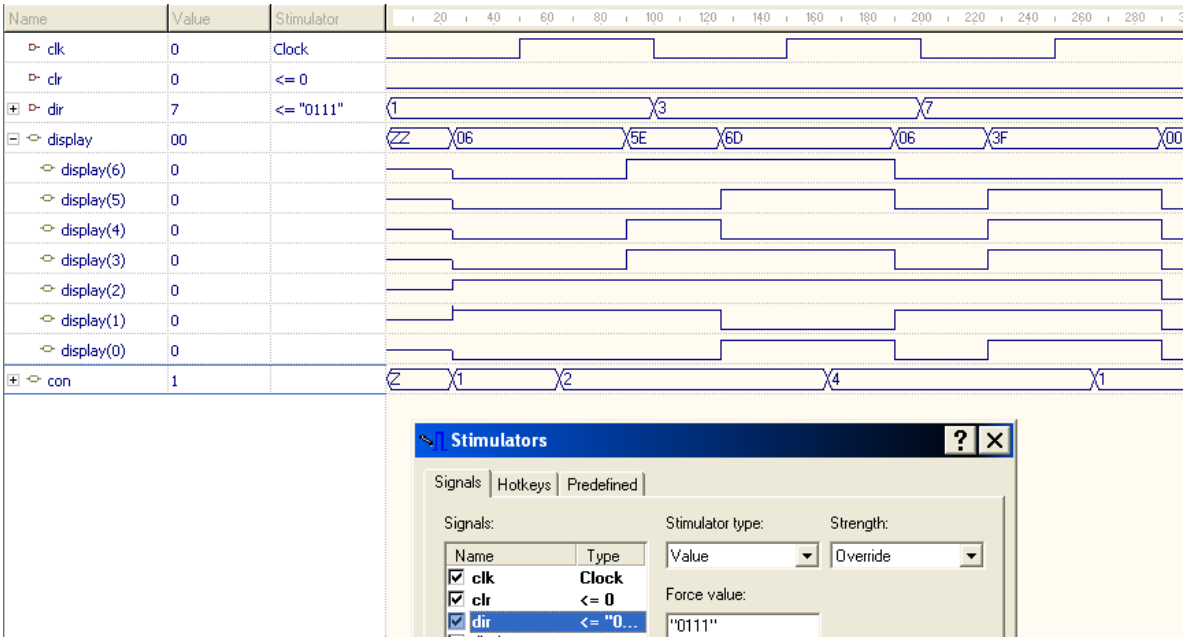
CODIGO 2:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity pract10A is port (
    clk, clr: in std_logic;
    dir: in std_logic_vector(3 downto 0);
    con: inout std_logic_vector(2 downto 0);
    display: out std_logic_vector(6 downto 0)
);
end entity;

architecture apract10A of pract10A is
type matrix is array(0 to 15) of std_logic_vector(6 downto 0);
signal anodos: std_logic_vector(2 downto 0);
constant rom: matrix:= (
    "0000000", --_
    "0000000", --_
    "1011110", --d
    "0000110", --I
    "1101101", --S
    "1111001", --E
    "1010101", --n
    "0111111", --O
    "0000000", --_
    "1011110", --d
    "0000110", --I
    "1101111", --g
    "0000110", --I
    "0110001", --T
    "1110111", --A
    "0111000" --L
);
begin
    --contador de anillo
    process(clk, clr) begin
        if(clr = '1') then anodos <= "001";
        elsif(rising_edge(clk)) then
            case anodos is
                when "001" => anodos <= "010";
                when "010" => anodos <= "100";
                when "100" => anodos <= "001";
                when others => anodos <= "001";
```

```
            end case;
        end if;
    end process;
    con <= anodos;
    display <= rom(conv_integer(dir) + 2) when (con = "001")
    else rom(conv_integer(dir) + 1) when (con = "010")
    else rom(conv_integer(dir));
end architecture;
```



CUESTIONARIO:

1. ¿Cuántos dispositivos PLD 22V10 son necesarios para el desarrollo de esta práctica? R= 2

2. ¿Cuántos dispositivos de la serie 74xx (TTL) ó 40xx (CMOS) hubieras necesitado para el desarrollo de esta práctica? R= 25-35

3. ¿Cuántos pines de entrada/salida del PLD 22V10 se usan en el diseño? R= 15/22 en total

4. ¿Cuántos términos producto ocupan las ecuaciones para cada señal de salida y que porcentaje se usa en total del PLD 22V10? R= 75/121 = 61%

5. ¿Qué puedes concluir de esta práctica?

R= En esta practica se pudo observar el funcionamiento interno que tienen las memorias que usamos día con día. Aunque de manera muy reducida, es suficiente para darse una idea de la forma en la que estas trabajan internamente. Ahora se puede ver de forma mas clara que una memoria esta compuesta por arreglos de bytes que a su vez son arreglos de bits, donde se guardan las direcciones de memoria de los archivos en una computadora.