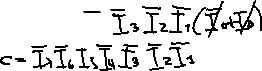
1. Complete la tabla e implemente la *Decode* (unidad de decodificación). Esta unidad permite la activación de las señales de control a partir del valor del código de instrucción almacenado en *IR*. Se recomienda realizar la minimización de las funciones.



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Instrucción** | ***Ins\_Binario*** | ***bus\_sel*** | ***alu\_op*** | ***reg\_a\_we*** | ***out\_we*** | ***reg\_we*** |
| *IN (0x01)* | 0000 0001 | 10 | 000 | 0 | 0 | 1 |
| *OUT (0x02)* | 0000 0010 | 00 | 000 | 0 | 1 | 0 |
| *MOV (0x03)* | 0000 0011 | 00 | 000 | 0 | 0 | 1 |
| *LDA (0x04)* | 0000 0100 | 00 | 000 | 1 | 0 | 0 |
| *LDI (0x05)* | 0000 0101 | 01 | 000 | 1 | 0 | 0 |
| *ADD (0x10)* | 0001 0000 | 00 | 010 | 0 | 0 | 1 |
| *SUB (0x11)* | 0001 0001 | 00 | 011 | 0 | 0 | 1 |
| *AND (0x12)* | 0001 0010 | 00 | 100 | 0 | 0 | 1 |
| *OR (0x13)* | 0001 0011 | 00 | 101 | 0 | 0 | 1 |
| *XOR (0x14)* | 0001 0100 | 00 | 110 | 0 | 0 | 1 |
| *SHL (0x20)* | 0010 0000 | 00 | 001 | 0 | 0 | 1 |
| *SHR (0x21)* | 0010 0001 | 00 | 111 | 0 | 0 | 1 |

La instrucción en Hexadecimal será traducida a Binario y se hará referencia a la misma con “”

Funciones:



|  |  |
| --- | --- |
| ***Ins\_Binario*** | ***reg\_we*** |
| 0000 0001 | 1 |
| 0000 0011 | 1 |
| 0001 0000 | 1 |
| 0001 0001 | 1 |
| 0001 0010 | 1 |
| 0001 0011 | 1 |
| 0001 0100 | 1 |
| 0010 0000 | 1 |
| 0010 0001 | 1 |



|  |  |
| --- | --- |
| ***Ins\_Binario*** | ***alu\_op*** |
| 0001 0000 | 010 |
| 0001 0001 | 011 |
| 0001 0010 | 100 |
| 0001 0011 | 101 |
| 0001 0100 | 110 |
| 0010 0000 | 001 |
| 0010 0001 | 111 |



1. Contenido de decode.vhd

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity decode is

Port (

input : in STD\_LOGIC\_VECTOR (7 downto 0);

out\_we : out STD\_LOGIC;

regs\_we : out STD\_LOGIC;

alu\_op : out STD\_LOGIC\_VECTOR (2 downto 0);

reg\_a\_we : out STD\_LOGIC;

bus\_sel : out STD\_LOGIC\_VECTOR (1 downto 0));

end decode;

architecture Behavioral of decode is

begin

out\_we <= '1' when input = x"02" else

'0';

regs\_we <= '1' when input = x"01" or

input = x"03" or

input = x"10" or

input = x"11" or

input = x"20" or

input = x"21" or

input = x"12" or

input = x"13" or

input = x"14" else

'0';

alu\_op <= "010" when input = x"10" else

"011" when input = x"11" else

"100" when input = x"20" else

"101" when input = x"21" else

"110" when input = x"12" else

"001" when input = x"13" else

"111" when input = x"14" else

"000";

reg\_a\_we <= '1' when input = x"04" or

input = x"05" else

'0';

bus\_sel <= "10" when input = x"01" else

"01" when input = x"05" else

"00";

End Behavioral;

El circuito Decode implementado es combinacional, esto debido que su salida depende únicamente de los datos de entrada y no almacena estados entre cada ejecución

1. VHDL
2. Cálculo de Reloj:
3. Dado el siguiente programa, y considerando que todos los registros del procesador se encuentran inicializados en cero:

ldi 3 add r1, r0 ldi 5 add r2, r1 sub r3, r2 and r4, r1 xor r5, r1

or r7, r1 mov r14, r3 out r2 out r3 out r4 out r5 out r7

out r14

* + 1. Determine para cada ciclo de reloj, cuales son los registros que se actualizan, y el valor. Solo se debe especificar los registros correspondientes a los datos.
    2. Describir el contenido ROM para la ejecución de este programa, indicando celda y contenido.

Por cada ciclo se ejecuta una operación

ldi 3 0x0503 cambia el contenido del registro A(se actualiza el registro A) Nuevo valor rA: 3

add r1, r0 0x1010 suma el contenido del registro A y del registro 0 y lo almacena en el registro 1 (se actualiza r1) Nuevo valor r1: 3

ldi 5 0x0505 cambia el contenido del registro A(se actualiza el registro A) Nuevo valor rA: 5

add r2, r1 0x1021 suma el contenido del registro A y del registro 1 y lo almacena en el registro 2 (se actualiza r2) Nuevo valor r2: 8

sub r3, r2 0x1132 le resta al registro 2 el contenido del registro A y lo almacena en el registro 3 (se actualiza r3) Nuevo valor r3: 3

and r4, r1 0x1241 se hace un and entre el registro A y el registro 1 y el resultado se almacena en el registro 4(se actualiza r4) Nuevo valor r4: 1

101 and 011 = 001

xor r5, r1 0x1451 se hace un xor entre el registro A y el registro 1 y el resultado se almacena en el registro 5(se actualiza r5) Nuevo valor r5: 6

101 xor 011 = 110

or r7, r1 0x1371 se hace un or entre el registro A y el registro 1 y el resultado se almacena en el registro 7(se actualiza r7) Nuevo valor r7: 7

101 or 011 = 111

mov r14, r3 0x03E3 se almacena el valor del registro 3 en el registro 14(se actualiza r14) Nuevo valor r14: 3

out r2 0x0202 Se guarda en el registro out el valor del registro r2(se actualiza el registro out) Nuevo valor reg\_out: 8

out r3 0x0203 Se guarda en el registro out el valor del registro r2(se actualiza el registro out) Nuevo valor reg\_out: 3

out r4 0x0204 Se guarda en el registro out el valor del registro r2(se actualiza el registro out) Nuevo valor reg\_out: 1

out r5 0x0205 Se guarda en el registro out el valor del registro r2(se actualiza el registro out) Nuevo valor reg\_out: 6

out r7 0x0207 Se guarda en el registro out el valor del registro r2(se actualiza el registro out) Nuevo valor reg\_out: 7

out r14 0x020E Se guarda en el registro out el valor del registro r2(se actualiza el registro out) Nuevo valor reg\_out: 3