ESTRUCTURA DE COMPUTADORES
UNIDAD 2.- LA UNIDAD ARITMÉTICO LÓGICA (ALU)

2.1. Demostrar por medio de una tabla de verdad, que la expresión lógica $Cn \oplus Cn\text{-}1$ genera una señal denominada bandera de desbordamiento (overflow, V), para identificar errores que se generan al sumar números enteros con signo, en codificación complemento a 2.

Solución:

El desbordamiento se produce al sumar dos números X, Y de n bits del mismo signo , positivo o negativo y el resultado tiene un signo distinto, negativo o positivo respectivamente. Por tanto la tabla de verdad para la función desbordamiento (*overflow*, V) será:

X_{n-1}	Y_{n-1}	C_{n-1}	S_{n-1}	C_n	$C_n \oplus C_{n-1}$	*Overflow*	**V**
0(+)	0(+)	0	0(+)	0	0	No	0
0(+)	0(+)	1	1(-)	0	**1**	**Si**	**1**
0(+)	1(-)	0	1(-)	0	0	No	0
0(+)	1(-)	1	0(+)	1	0	No	0
1(-)	0(+)	0	1(-)	0	0	No	0
1(-)	0(+)	1	0(+)	1	0	No	0
1(-)	1(-)	0	0(+)	1	**1**	**Si**	**1**
1(-)	1(-)	1	1(-)	1	0	No	0

2.2. Además del resultado, la ALU genera un conjunto de bits que pueden ser utilizados por el sistema o por los usuarios para el control de las operaciones aritmético-lógicas desarrolladas. Entre estos los más conocidos son el bit o bandera de signo (N, N = '1' si el resultado es negativo), el bit o bandera de cero (Z, Z = '1' si el resultado es cero), el bit o bandera de acarreo (C, C = '1' si hay acarreo en la operación de suma entre los bits más significativos de ambos operandos), y el bit o bandera de desbordamiento u overflow (V, V = '1' si se supera la capacidad de representación del sistema). Utilizando números binarios de 8 bits con signo y representados en complemento a 2, realice las operaciones señaladas con dos operandos en decimal y compruebe en cada caso, el valor de estos cuatro bits, N, Z, C y V, señale en cada caso su significado.

a) 46 + 67 **b)** 112 - 89 **c)** 75 + 95 **d)** -34 – 97

Solución:

a) 46 + 67 = 113

$0010\ 1110_2$
$0100\ 0011_2$

$0111\ 0001_2$
$+\ 113_{10}$
71_{16}

N = 0, positivo
Z = 0, resultado ≠ 0
C = 0, no hay acarreo.
V = 0, correcto

b) 112 - 89 = 23

$0111\ 0000_2$
$1010\ 0111_2$

$0001\ 0111_2$
$+\ 23_{10}$
17_{16}

N = 0, positivo
Z = 0, resultado ≠ 0
C = 1, hay acarreo.
V = 0, correcto.

c) 75 + 95 = 170

$0100\ 1011_2$
$0101\ 1111_2$

$1010\ 1010_2$
$-\ 86_{10}$
AA_{16}

N = 1, negativo
Z = 0, resultado ≠ 0
C = 0, no hay acarreo.
V = 1, incorrecto

d) -34 – 97 = - 131

$1101\ 1110_2$
$1001\ 1111_2$

$0111\ 1101_2$
$+125_{10}$
$7D_{16}$

N = 0, positivo.
Z = 0, resultado ≠ 0
C = 1, hay acarreo.
V = 1, incorrecto.

ESTRUCTURA DE COMPUTADORES
UNIDAD 2.- LA UNIDAD ARITMÉTICO LÓGICA (ALU)

2.3. Utilizando números binarios de 8 bits con signo y representados en complemento a 2, realice en el orden señalado por los paréntesis las operaciones indicadas. Para cada resultado parcial, compruebe si se produce un desbordamiento aritmético y calcule también la validez del resultado final. Analice los resultados obtenidos en relación con el valor de la bandera V del problema 2.1 y observe que, resultados parciales incorrectos, no suponen necesariamente que el resultado final lo sea.

a) (((((32 +100) + 70) + 24) – 62) – 50). **b) ((((**43 - 12) + 34) + 75) – 47) **c) ((((**15 – 77) – 43) – 38) + 32)

Solución:

a) 32 + 100 = 132; 132 + 70 = 202; 202 + 24 = 226; 226 – 62 = 164; 164 – 50 = 114.

$0010\ 0000_2$	$1000\ 0100_2$	$1100\ 1010_2$	$1110\ 0010_2$	$1010\ 0100_2$
$0110\ 0100_2$	$0100\ 0110_2$	$0001\ 1000_2$	$1100\ 0010_2$	$1100\ 1110_2$
$1000\ 0100_2$	$1100\ 1010_2$	$1110\ 0010_2$	$1010\ 0100_2$	$0111\ 0010_2$
-124_{10}	$-\ 54_{10}$	$-\ 30_{10}$	$-\ 92_{10}$	$+114_{10}$
84_{16}	CA_{16}	$E2_{16}$	$A4_{16}$	72_{16}
V^1=1. Incorrecto	V^2=0. Correcto	V^3=0. Correcto	V^4=0. Correcto	V^5=1. Incorrecto

La solución final es correcta ya que los dos overflows habidos (V = 1), corresponden a signos contrarios (N = 1, N = 0).

b) 43 - 12 = 31; 31 + 34 =65; 65 + 75 = 140; 140 – 47 = 93.

$0010\ 1011_2$	$0001\ 1111_2$	$0100\ 0001_2$	$1000\ 1100_2$
$1111\ 0100_2$	$0010\ 0010_2$	$0100\ 1011_2$	$1101\ 0001_2$
$0001\ 1111_2$	$0100\ 0001_2$	$1000\ 1100_2$	$0101\ 1101_2$
$+\ 31_{10}$	$+\ 65_{10}$	$-\ 116_{10}$	$+\ 93_{10}$
$1F_{16}$	41_{16}	$8C_{16}$	$5D_{16}$
V^1=0. Correcto	V^2=0. Correcto	V^3=1. Incorrecto	V^4=1. Incorrecto

La solución final es correcta ya que los dos overflows habidos (V = 1), corresponden a signos contrarios (N = 1, N = 0).

c) 15 - 77 = - 62; -62 - 43 = - 105; -105 -38 = -143; -143 + 32 = - 111.

$0000\ 1111_2$	$1100\ 0010_2$	$1001\ 0111_2$	$0111\ 0001_2$
$1011\ 0011_2$	$1101\ 0101_2$	$1101\ 1010_2$	$0010\ 0000_2$
$1100\ 0010_2$	$1001\ 0111_2$	$0111\ 0001_2$	$1001\ 0001_2$
$-\ 62_{10}$	$-\ 105_{10}$	$+\ 113_{10}$	$-\ 111_{10}$
$C2_{16}$	97_{16}	71_{16}	91_{16}
V^1=0. Correcto	V^2=0. Correcto	V^3=1. Incorrecto	V^4=1. Incorrecto

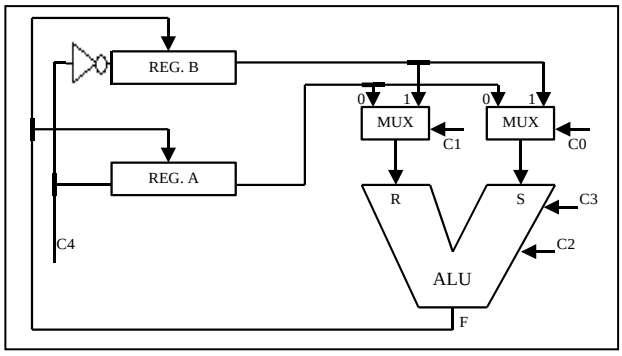
La solución final es correcta ya que los dos overflows habidos (V = 1), corresponden a signos contrarios (N = 0, N = 1).

ESTRUCTURA DE COMPUTADORES
UNIDAD 2.- LA UNIDAD ARITMÉTICO LÓGICA (ALU)

2.4. Considerar la ALU y los registros que se muestran en la figura. Responder a las siguientes cuestiones escribiendo la(s) palabra(s) de control adecuada(s). Cada palabra de control debe especificarse de acuerdo con el formato $C_4C_3C_2C_1C_0$. Por ejemplo, la operación "A+B→A" se escribiría 10001.

c) Sugerir dos métodos para llevar el registro A al valor cero.

d) Sugerir una secuencia de control que intercambie los contenidos de los registros A y B. La interpretación de los distintos puntos de control se resume en la tabla adjunta.

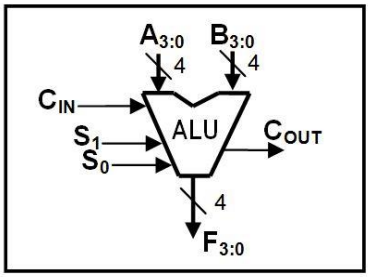


C_1C_0	→R	→S	C_3C_2	F	C_4	ACCION
00	A	A	00	R + S	0	F → B
01	A	B	01	R – S	1	F → A
10	B	A	10	R **AND** S		
11	B	B	11	R **XOR** S		

Solución:

FUNCION	OPERACIONES	C4	C3	C2	C1	C0
a1) $0 \Rightarrow R_A$	$R_A \oplus R_A \Rightarrow R_A$	1	1	1	0	0
a2) $0 \Rightarrow R_A$	$R_A - R_A \Rightarrow R_A$	1	0	1	0	0
b1) $R_A \Leftrightarrow R_B$	$R_A + R_B \Rightarrow R_B$	0	0	0	0	1
	$R_B - R_A \Rightarrow R_A$	1	0	1	1	0
	$R_B - R_A \Rightarrow R_B$	0	0	1	1	0
b2) $R_A \Leftrightarrow R_B$	$R_A \oplus R_B \Rightarrow R_A$	1	1	1	0	1
	$R_A \oplus R_B \Rightarrow R_B$	0	1	1	0	1
	$R_A \oplus R_B \Rightarrow R_A$	1	1	1	0	1

2.5. Utilizando la ALU de la figura, indique las operaciones a realizar en la ALU para que las salidas representen el módulo del resultado de la diferencia (X - Y), en donde X e Y son números positivos de 4 bits en complemento a 2.



S_1 S_0	Operación
0 0	F = AND (A,B)
0 1	F = /A + C_{IN}
1 0	F = A + B + C_{IN}
1 1	F= A + /B + C_{IN}

Solución:

|X - Y| = X - Y, si X ≥ Y. |X - Y| = Y - X = - (X - Y), si X < Y.

Asociando las entradas X e Y a las entradas A y B y tomando C_{IN} = '1' =>
El programa sería: El control sería:
- Restar A – B - Activar S_1 S_0 = 1 1 // se ejecuta F = A + C2(B) => (X - Y)
- Si C_{OUT} = '1' (X ≥ Y) - F = |X - Y|
- Si C_{OUT} = '0' (X < Y) - y por tanto se debe invertir el valor de F para obtener (Y - X).

2.- Con F = A y C_{IN} = '1'. Activando S_1 S_0 = 0 1, se ejecuta F = /A + 1 => C2(A) => (Y - X)

2.6. Diseñar en VHDL un circuito *Barrel Shift*, para números de 32 bits y un máximo de 31 desplazamientos, tanto a la izquierda como a la derecha.

Solución:

```
----------------------------------------------------------------------------------
-- Desplazador de barril parametizable
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Desplazador is
            generic (n: integer   := 4);  -- n: 16,8,4,2,1
   Port ( sright : in  STD_LOGIC;
          data_in : in  STD_LOGIC_VECTOR (31 downto 0);
          data_out : out STD_LOGIC_VECTOR (31 downto 0);
          enable : in  STD_LOGIC);
end Desplazador;

architecture generica of Desplazador is
signal ceros: std_logic_vector (n-1 downto 0) ;
begin
            ceros <= (others => '0');
            process (enable, data_in, sright)
            begin
                        if enable = '1' then
                                    if sright = '1' then data_out <= ceros & data_in(31 downto n) ;
                                    else data_out <= data_in(31-n downto 0) & ceros ;
                                    end if;
                        else data_out <= data_in;
                        end if;
            end process;
end generica;
```

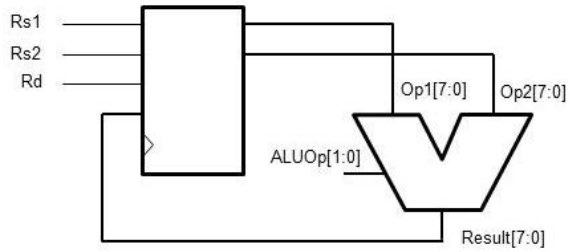
```
----------------------------------------------------------------------------------
-- Barrel Shift
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Barrel is
    Port ( despl : in  STD_LOGIC_VECTOR (4 downto 0);
           data_in : in  STD_LOGIC_VECTOR (31 downto 0);
           data_out : out  STD_LOGIC_VECTOR (31 downto 0);
           sright : in  STD_LOGIC;
           enable : in std_logic);
end Barrel;

architecture Behavioral of Barrel is
COMPONENT Desplazador
    generic ( n: integer );
                PORT(
         sright : IN  std_logic;
         data_in : IN  std_logic_vector(31 downto 0);
         data_out : OUT  std_logic_vector(31 downto 0);
         enable : IN  std_logic
        );
END COMPONENT;
signal habilitar: std_logic_vector (4 downto 0);
signal dieciseis: std_logic_vector (31 downto 0);
signal ocho: std_logic_vector (31 downto 0);
signal cuatro: std_logic_vector (31 downto 0);
signal dos: std_logic_vector (31 downto 0);

begin
habilitar <= ( 4 => enable and despl(4), 3 => enable and despl(3), 2 => enable and despl(2),
                                                1 => enable and despl(1), 0 => enable and despl(0)) ;
u16: desplazador
                        generic map (n => 16)
                        PORT MAP (sright => sright, data_in => data_in, data_out => dieciseis, enable => habilitar(4));
u8: desplazador
                        generic map (n => 8)
```

```
                        PORT MAP (sright => sright, data_in => dieciseis, data_out => ocho, enable => habilitar(3));
u4: desplazador
                        generic map (n => 4)
                        PORT MAP (sright => sright, data_in => ocho, data_out => cuatro, enable => habilitar(2));
u2: desplazador
                        generic map (n => 2)
                        PORT MAP (sright => sright, data_in => cuatro, data_out => dos, enable => habilitar(1));
u1: desplazador
                        generic map (n => 1)
                        PORT MAP (sright => sright, data_in => dos, data_out => data_out, enable => habilitar(0));
end Behavioral;
```

```
--------------------------------------------------------------------------------
-- Simulador para el Barrel Shift
--------------------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.all;
USE ieee.numeric_std.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;

ENTITY Barrel_TB IS
END Barrel_TB;

ARCHITECTURE behavior OF Barrel_TB IS
    COMPONENT Barrel
   PORT(
        despl : IN  std_logic_vector(4 downto 0);
        data_in : IN  std_logic_vector(31 downto 0);
        data_out : OUT  std_logic_vector(31 downto 0);
        sright : IN  std_logic;
             enable : IN  std_logic
       );
   END COMPONENT;

  --Inputs
  signal despl : std_logic_vector(4 downto 0) := (others => '0');
  signal data_in : std_logic_vector(31 downto 0) := (others => '0');
  signal sright : std_logic := '0';
  signal enable : std_logic := '0';

             --Outputs
  signal data_out : std_logic_vector(31 downto 0);

BEGIN

             -- Instantiate the Unit Under Test (UUT)
  uut: Barrel PORT MAP (
        despl => despl,
        data_in => data_in,
        data_out => data_out,
        sright => sright,
        enable => enable
                        );

 -- Stimulus process
  stim_proc: process
  begin
             enable <= '1' ;
                        sright <= '1' ;
 -- Desplazamiento a la derecha.
                        data_in <= x"80000000" ;
                        for i in 0 to 31 loop
                                    despl <= conv_std_logic_vector (i,5);
                                    wait for 10 ns;
     end loop;
-- Desplazamiento a la izquierda.
                        sright <= '0';
                        data_in <= x"00000001" ;
                        for i in 0 to 31 loop
                                    despl <= conv_std_logic_vector (i,5);
                                    wait for 10 ns;
     end loop;
  end process;
END;
```

ESTRUCTURA DE COMPUTADORES
UNIDAD 2.- LA UNIDAD ARITMÉTICO LÓGICA (ALU)

2.7. Se tiene la arquitectura de la figura con un banco de dos registros (R0 y R1) de 8 bits y una ALU de 8 bits que realiza cuatro operaciones (ver tabla). La señal de control ALUOp, de 2 bits, sirve para elegir la operación a realizar. Las señales de control Rs1, Rs2 y Rd sirven para indicar cuáles son los 2 registros utilizados en la ALU (Rs1 para Op1 y Rs2 para Op2) y cuál es el registro escrito (Rd para Result). En todas ellas, '0' indica el registro R0 y '1' indica el registro R1.La operación SLT pone la salida a 1 (número entero 1) si Op1 < Op2 y a 0 (número entero 0) en caso contrario, considerando que los operandos tienen signo y están en complemento a 2.



ALUOp	Operación
00	Result=Op1+Op2
01	Result=NAND(Op1,Op2)
10	Result=XOR(Op1,Op2)
11	Result=SLT(Op1,Op2)

La palabra de control es de 5 bits, (ALUOp,Rs1,Rs2,Rd). Por ejemplo, si la palabra de control es ¨00011¨ se realizará la operación R1 <= R0 + R1;

a. Indique la palabra de control para conseguir que R0 quede a 0.

ALUOp,Rs1,Rs2,Rd => 10000 => R0 = XOR (R0, R0).

también => 10110 => R0 = XOR (R1, R1).

también => 11000 => R0 = SLT (R0, R0).

también => 11110 => R0 = SLT (R1, R1).

b. Indique la palabra de control para conseguir que R1 reciba NOT(R0)

ALUOp,Rs1,Rs2,Rd => 01001 => R1= NAND (R0, R0).

c. Si inicialmente R0=0x08 y R1=0xFF, indique a qué queda el registro R1 tras realizar la siguiente secuencia de instrucciones: 1. “00010” y 2. “11101”

00010 => R0 = R0 + R1 = 0x07
11101 => Si R1 < R0, then R1 = 0x01, else R1 = 0x00.

R1 = 0x01

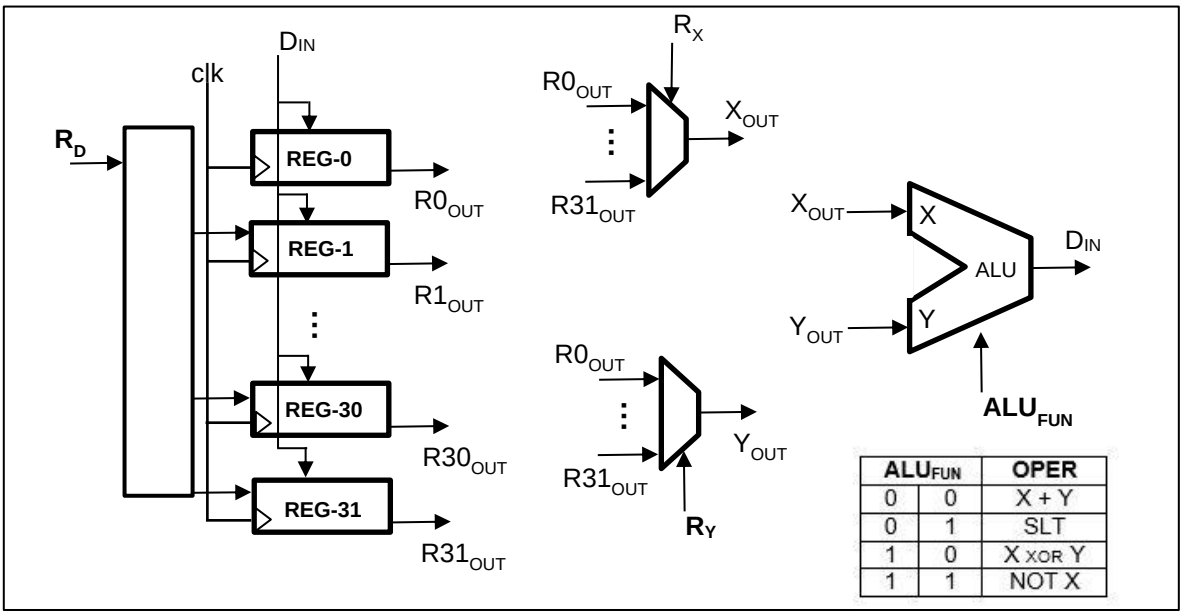
d. Si inicialmente R0=0xFE y R1=0x05, indique a qué queda el registro R1 tras realizar la siguiente secuencia de instrucciones: 1. “11011” y 2. “10011”

11011 => Si R0 < R1, then R1 = 0x01, else R1 = 0x00.
10011 => R1 = XOR (R0, R1)

R1 = 0xFF

ESTRUCTURA DE COMPUTADORES
UNIDAD 2.- LA UNIDAD ARITMÉTICO LÓGICA (ALU)

2.8. En el circuito de la figura se muestra la arquitectura de un cierto sistema digital en el que se distinguen elementos combinacionales ya conocidos como multiplexores, decodificadores y una unidad aritmético lógica (ALU), y elementos secuenciales como un conjunto de 32 registros, en el que como es habitual, el registro R0 es de sólo lectura y su valor es siempre 0. Considere que todos los registros son de 32 bits y salvo R_0 con valores desconocidos. En la figura se identifican cuatro señales de control con distinto tamaño en bits, R_D, R_X, R_Y y ALU_{FUN}.



La operación SLT de la ALU, como se ha visto en clase, pone un '1' en el registro destino si el valor de la entrada X es menor que el de la entrada Y.

A la vista del esquema facilitado, se pide:

a) Señale, justificando necesariamente la respuesta, el tamaño en bits y la función de la señal de control "R_D".

"R_D" es la señal de entrada de un decodificador 5-32, por tanto es una señal de 5 bits, $2^5 = 32$. Su función en el circuito es seleccionar, habilitar para escritura el registro destino, un único registro entre los 32 posibles. Cuando R_D = "00000", el REG-0 no se habilita.

b) Señale, justificando necesariamente la respuesta, el tamaño en bits y la función de la señal de control "R_X".

"R_X" es la señal de control de un multiplexor 32-1. Por tanto es una señal 5 bits, $2^5 = 32$. Su función en el circuito es seleccionar, elegir entre los 32 posibles registros, el registro desde donde se lee el operando X de acceso a la ALU.

c) Describa un algoritmo como desee y asócielo a la palabra o palabras de control correspondientes para su ejecución para conseguir que R_5 <= 0xFFFFFFFF.

La operación sólo necesita una única instrucción. Palabra de control: R_D, R_X, R_Y, ALU_{FUN}
R_5 <= NOT R_0 **00101, 00000, XXXXX, 11**

d) Diseñe un algoritmo y escriba la palabra o palabras de control necesarias para su ejecución para calcular la operación R_4 <= Complemento a 2 de R_3.

La operación necesita cuatro instrucciones. Palabra de control: R_D, R_X, R_Y, ALU_{FUN}
- Poner un valor < 0 en un registro (R_5 todos '1', valor -1)
  R_5 <= NOT R_0 **00101, 00000, XXXXX, 11**
- Utilizar SLT entre R_0 y R_5, (R_5 = "0..01")
  R_5 <= SLT X,Y **00101, 00101, 00000, 01**
- Complemento el registro R_4 y sumar '1' al resultado
  R_3 <= NOT R_3 **00011, 00011, XXXXX, 11**
  R_4 <= R_5 + R_3 **00100, 00101, 00011, 00**

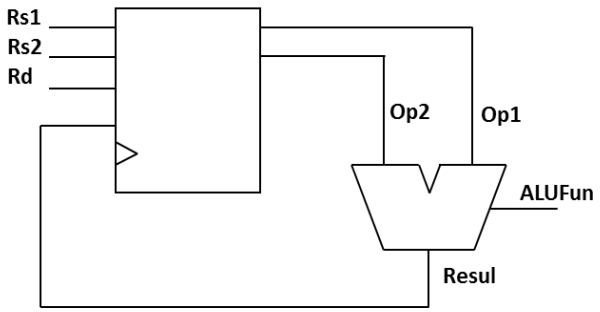
ESTRUCTURA DE COMPUTADORES
UNIDAD 2.- LA UNIDAD ARITMÉTICO LÓGICA (ALU)

NOTA: Para considerar válida una palabra de control, debe señalar la secuencia de bits que corresponda en el orden de las variables R_D, R_X, R_Y, ALU_{FUN}. Utilice una coma para separar cada señal.

2.9. En el circuito de la figura se muestra la arquitectura básica de un sistema microprocesador, con un banco de 8 registros de 16 bits y una ALU capaz de realizar 8 operaciones distintas. Los 8 registros se denominan R0 a R7. El registro R0 es constante e igual a 0, el registro R1 también es constante e igual a 1, y el registro R2 también es constante e igual a 2. El resto de registros, R3 a R7, son variables y de propósito general.

Los bits de control Rs1 y Rs2 indican, respectivamente, qué registros irán a Op1 y Op2 de la ALU, mientras que los bits de control Rd indican qué registro recibirá el valor de la salida de la ALU, Resul. Por su parte, la operación realizada en la ALU se decide mediante los bits de control ALUFun. En el caso de los desplazamientos, Op1 es el operando desplazado y Op2 indica en cuántas posiciones hay que desplazarlo.

ALUFun	Operación
000	Resul = Op1 + Op2
001	Resul = Op1 - Op2
010	Resul = Op1 NOR Op2
011	Resul = Op1 XOR Op2
100	Resul = Op1 AND Op2
101	Resul = Op1 SLL Op2
110	Resul = Op1 SRL Op2
111	Resul = Op1 SRA Op2



Se pide:

a. Dadas las instrucciones en código máquina de la tabla adjunta, se pide, traducir este código a la "Instrucción ensamblador" correspondiente y calcular el valor de los registros destinos del código dado. El estado inicial de los registros es desconocido, pero las instrucciones indicadas se ejecutan en el orden señalado.

ALUFun, Rd, Rs1, Rs2	Instrucción ensamblador
001, 011, 000, 001	R3 = R0 – R1
011, 100, 001, 010	R4 = R1 XOR R2
101, 101, 001, 010	R5 = R1 SLL R2
110, 110, 011, 010	R6 = R3 SRL R2
111, 111, 011, 010	R7 = R3 SRA R2

Registro	Valor final
R3	0xFFFF
R4	0x0003
R5	0x0004
R6	0x3FFF
R7	0xFFFF

b. Sin tener en cuenta los valores de los registros de las operaciones anteriores, ahora se quiere realizar la operación R4 = 7·R3. Al no existir dicha operación en este micro, se realizará a través de una serie de instrucciones. Se pide el código ensamblador así como el código máquina que realizan la operación solicitada. Se valorará, por este orden, utilizar el mínimo número de instrucciones posible y no modificar el contenido de los registros no implicados en la operación (o el mínimo número de registros).

Instrucción ensamblador	ALUFun, Rd, Rs1, Rs2	Comentarios (opcionales)
R4 = R1 + R2	000, 100, 001, 010	R4 = 3
R4 = R3 SLL R4	101, 100, 011, 100	R4 = 8·R3
R4 = R4 – R3	001, 100, 100, 011	R4 = 7·R3