

## **Unidad 2: La Unidad Aritmético Lógica (ALU)**

Escuela Politécnica Superior - UAM

## Índice

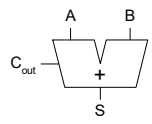
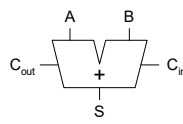
- **Circuitos aritméticos: Sumador, Restador y Comparador**
- ALU
- Circuitos aritméticos: Desplazador y Multiplicador
- Sistemas de numeración para números reales

## Introducción

- Para diseñar un microprocesador no partimos de cero, sino que se hace un diseño jerárquico reutilizando bloques.
- Bloques que usaremos:
  - ✓ Multiplexores, decodificadores, registros, circuitos aritméticos, contadores, memorias.
- Tres conceptos:
  - ✓ **Jerarquía:** divide y vencerás.
  - ✓ **Modularidad:** hay que definir bien las interfaces y la funcionalidad de cada bloque.
  - ✓ **Regularidad:** una estructura regular se puede aplicar a distintos tamaños.

3

## Sumadores de 1 bit

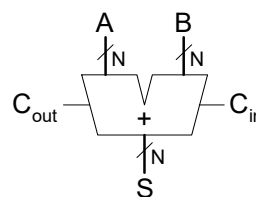
Half Adder				Full Adder				
								
A	B	C <sub>out</sub>	S	C <sub>in</sub>	A	B	C <sub>out</sub>	S
0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	1	0	1
1	0	0	1	0	1	0	0	1
1	1	1	0	0	1	1	1	0
$S = A \oplus B$ $C_{out} = AB$				$S = A \oplus B \oplus C_{in}$ $C_{out} = AB + BC_{in} + AC_{in}$				

4

## Sumadores multibit

- Sumadores multibit por propagación del acarreo (*carry propagate adders, CPA*) de tres tipos:
  - Sumadores ripple-carry, RCA (lento)
  - Sumadores carry-lookahead, CLA (rápido)
  - Sumadores prefijo-paralelo, PPA (+ rápido)
- Los sumadores CLA y PPA son más rápidos, pero requieren más hardware.

### Symbol

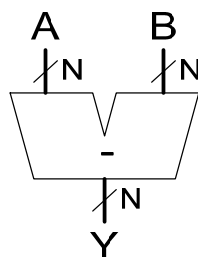


5

## Otros operadores

### Restador: $Y = A - B$

### Symbol



### Recuerda

(Circuitos Electrónicos Digitales)

$$Y = A - B = A + (-B)$$

¿Cómo calcular el inverso aditivo de un número en complemento a 2?

Haciendo la operación **NOT**, y después **sumarle 1**.

$$\text{Ej: } Y = 8_{10} - 5_{10} = 3_{10} = 01000_2 - 00101_2 = ?$$

$$\text{Inverso}(00101_2) = \text{NOT}(00101) + 1 = 11010 + 1 = 11011$$

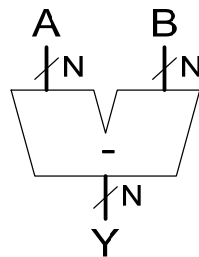
$$Y = A + (-B) = 01000_2 + 11011_2 = 00011_2$$

6

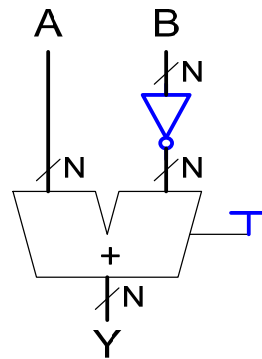
## Otros operadores

**Restador:  $Y = A - B$**

### Symbol



### Implementation

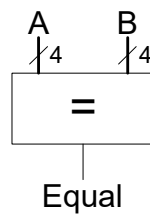


7

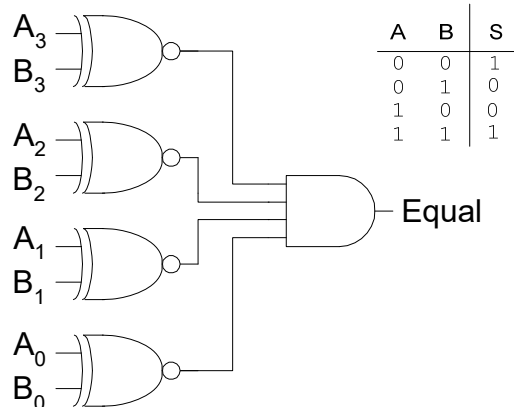
## Otros operadores

**Comparador igualdad (*Equal*): ¿  $A = B$  ?**

### Symbol



### Implementation

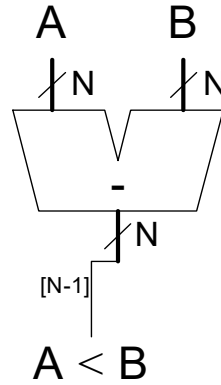


Dos números son iguales cuando todos sus bits son iguales

8

## Otros operadores

**Comparador menor que (*Less Than*): ¿  $A < B$  ?**



Para comprobar si un número es mayor/menor que otro, sólo hace falta restarlos y comprobar el signo del resultado (MSB)

9

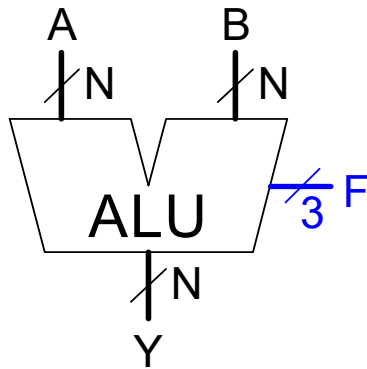
## Índice

- Circuitos aritméticos: Sumador, Restador y Comparador
- **ALU**
- Circuitos aritméticos: Desplazador y Multiplicador
- Sistemas de numeración para números reales

10

## Arithmetic Logic Unit (ALU)

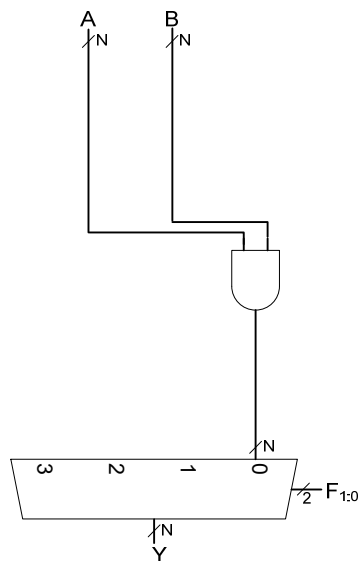
$F_{2:0}$  = Selector de función



$F_{2:0}$	Función
000	A and B
001	A or B
010	A + B
011	Sin usar
100	A and /B
101	A or /B
110	A - B
111	SLT

11

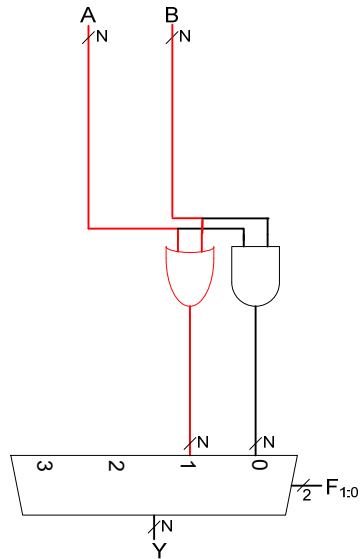
## Diseño de la ALU



$F_{2:0}$	Función
000	A and B
001	A or B
010	A + B
011	Sin usar
100	A and /B
101	A or /B
110	A - B
111	SLT

12

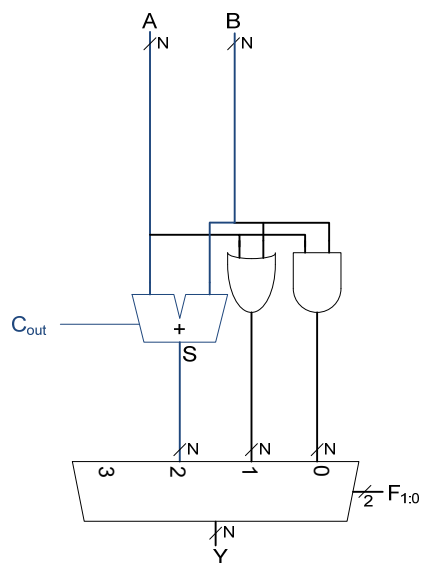
## Diseño de la ALU



$F_{2:0}$	Función
000	A and B
001	A or B
010	A + B
011	Sin usar
100	A and /B
101	A or /B
110	A - B
111	SLT

13

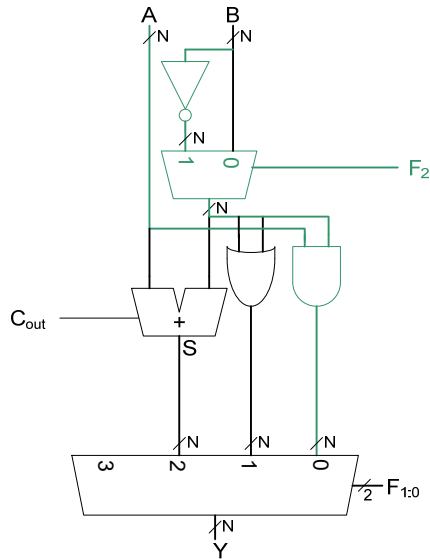
## Diseño de la ALU



$F_{2:0}$	Función
000	A and B
001	A or B
010	A + B
011	Sin usar
100	A and /B
101	A or /B
110	A - B
111	SLT

14

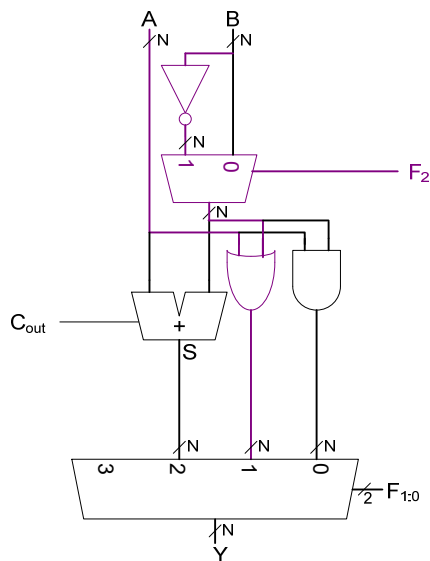
## Diseño de la ALU



$F_{2:0}$	Función
000	A and B
001	A or B
010	A + B
011	Sin usar
100	A and /B
101	A or /B
110	A - B
111	SLT

15

## Diseño de la ALU

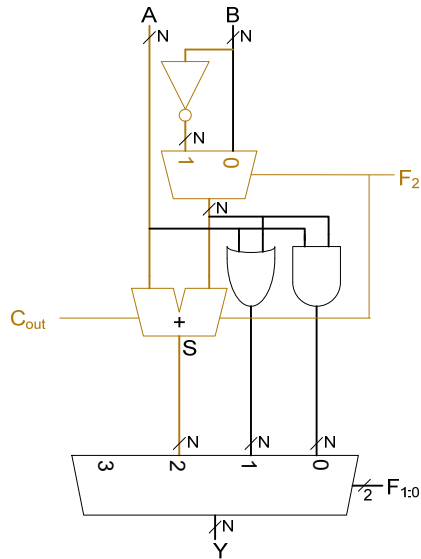


$F_{2:0}$	Función
000	A and B
001	A or B
010	A + B
011	Sin usar
100	A and /B
101	A or /B
110	A - B
111	SLT

16



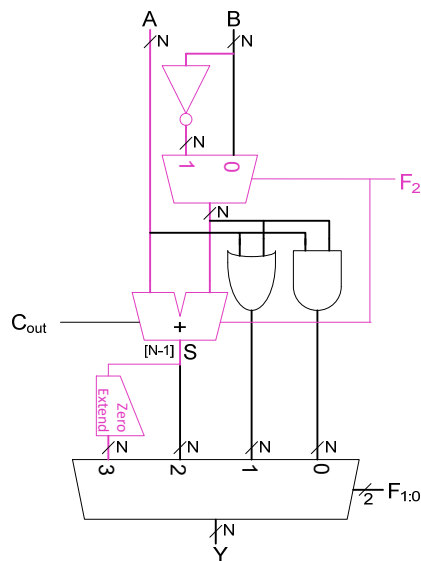
## Diseño de la ALU



$F_{2:0}$	Función
000	A and B
001	A or B
010	A + B
011	Sin usar
100	A and /B
101	A or /B
110	A - B
111	SLT

17

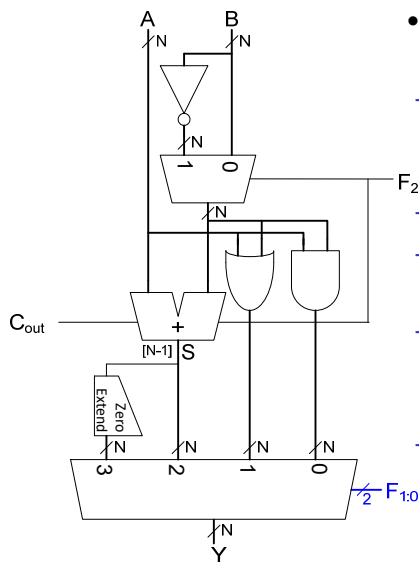
## Diseño de la ALU



$F_{2:0}$	Función
000	A and B
001	A or B
010	A + B
011	Sin usar
100	A and /B
101	A or /B
110	A - B
111	SLT

18

## Ejemplo de "Set Less Than" (SLT)



- El resultado es 1 si  $A < B$  y 0 en caso contrario. Suponemos  $A = 25$  y  $B = 32$ .
  - A es menor que B, así que esperamos que Y sea la representación en 32 bits de 1 (0x00000001).
  - Para SLT,  $F_{2:0} = 111$ .
  - $F_2 = 1$  hace que el sumador haga la resta. Así que  $25 - 32 = -7$ .
  - La representación en C2 de -7 tiene un 1 en el "most significant bit", así que  $S_{31} = 1$ .
  - Los bits  $F_{1:0} = 11$ , así que el mux elige  $Y = S_{31}$  (zero extended) = 0x00000001.

19

## Índice

- Circuitos aritméticos: Sumador, Restador y Comparador
- ALU
- **Circuitos aritméticos: Desplazador y Multiplicador**
- Sistemas de numeración para números reales

20

## Desplazadores

- **Desplazador lógico (logical shifter):** desplaza el valor a izquierda o derecha y rellena con 0's.
  - ✓ Ex:  $11001 \gg 2 = 00110$
  - ✓ Ex:  $11001 \ll 2 = 00100$
- **Desplazador aritmético (arithmetic shifter):** igual que el lógico, salvo que hacia la derecha rellena con el bit de signo (msb).
  - ✓ Ex:  $11001 \ggg 2 = 11110$
  - ✓ Ex:  $11001 \lll 2 = 00100$
- **Rotador (rotator):** rota a izquierda o derecha los bits en círculo, lo que sale por un lado entra por el otro.
  - ✓ Ex:  $11001 \text{ ROR } 2 = 01110$
  - ✓ Ex:  $11001 \text{ ROL } 2 = 00111$

21

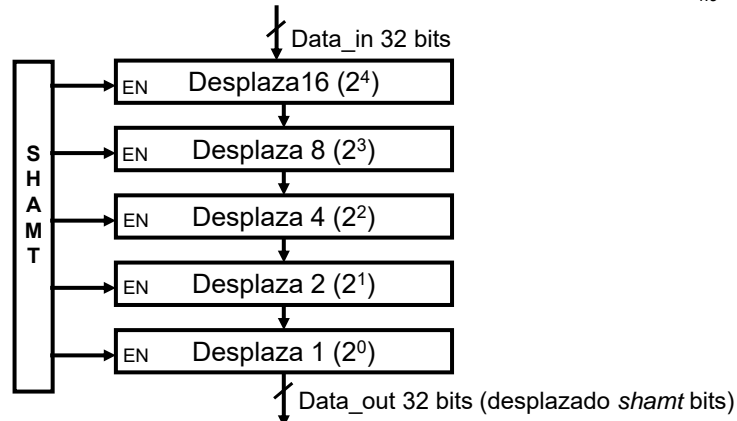
## Desplazar para multiplicar o dividir

- Un desplazamiento a izquierda de  $N$  bits equivale a multiplicar por  $2^N$ 
  - ✓ Ex:  $00001 \ll 2 = 00100$  ( $1 \times 2^2 = 4$ )
  - ✓ Ex:  $11101 \ll 2 = 10100$  ( $-3 \times 2^2 = -12$ )
- Un desplazamiento aritmético a derecha de  $N$  bits equivale a dividir entre  $2^N$ 
  - ✓ Ex:  $01000 \ggg 2 = 00010$  ( $8 \div 2^2 = 2$ )
  - ✓ Ex:  $10000 \ggg 2 = 11100$  ( $-16 \div 2^2 = -4$ )

22

## Desplazador en barril ( *Barrel Shifter* )

- ¿Cómo desplazar un número variable de posiciones, de 0 a 31?
  - ✓ En MIPS, dicha cantidad se codifica en  $shamt_{4:0}$
- Usando desplazadores fijos  $2^N$  en cadena
  - ✓ Cada desplazador se activa o no dependiendo de un bit en  $shamt_{4:0}$



23

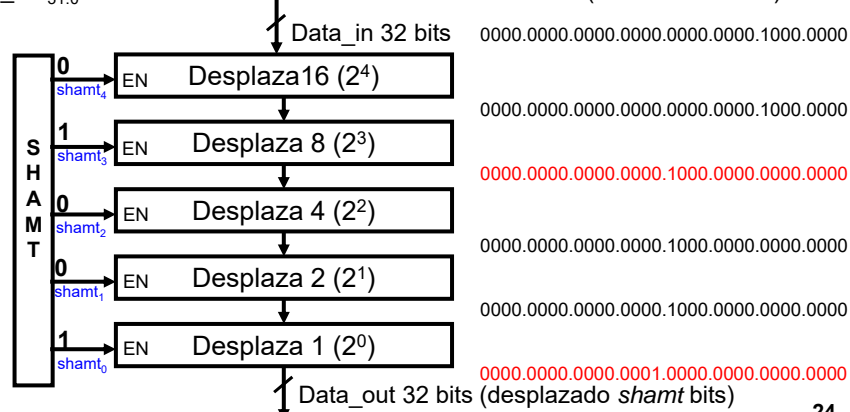
## Desplazador en barril ( *Barrel Shifter* )

Ejemplo:

$shamt_{4:0} = 01001$  (9)

$Data\_in_{31:0} = 0000.0000.0000.0000.0000.0000.1000.0000$  (128)

$Data\_out_{31:0} = 0000.0000.0000.0001.0000.0000.0000.0000$  ( $65536 = 128 * 2^9$ )



24

## Multiplicadores

- Pasos para multiplicar (en decimal o binario):
  - Los productos parciales se forman multiplicando un dígito del multiplicador por el multiplicando completo
  - Los productos parciales se desplazan y suman para tener el resultado final

### Decimal

$$\begin{array}{r}
 230 \\
 \times 42 \\
 \hline
 460 \\
 + 920 \\
 \hline
 9660
 \end{array}$$

multiplicand  
 multiplier  
 partial  
 products

result

$$230 \times 42 = 9660$$

### Binary

$$\begin{array}{r}
 0101 \\
 \times 0111 \\
 \hline
 0101 \\
 0101 \\
 0101 \\
 + 0000 \\
 \hline
 0100011
 \end{array}$$

$$5 \times 7 = 35$$

25

## Operación de multiplicar, sin signo

### Sin signo

$$\begin{array}{r}
 110001 \quad (49) \\
 \times 11010 \quad (26) \\
 \hline
 000000 \\
 110001 \\
 000000 \\
 110001 \\
 110001 \\
 \hline
 10011111010 \quad (1274)
 \end{array}$$

### Sin signo

$$\begin{array}{r}
 001001 \quad (9) \\
 \times 0110 \quad (6) \\
 \hline
 000000 \\
 001001 \\
 001001 \\
 000000 \\
 \hline
 000110110 \quad (54)
 \end{array}$$

26

## Operación de multiplicar, con signo

- **Aritmética sin signo:**

- Calcular el signo del resultado:
  - ✓ Pos\*Pos = Pos, Pos\*Neg = Neg, Neg\*Pos = Neg, Neg\*Neg = Pos.
- Si algún operando es negativo:
  - ✓ Hacer su complemento a 2 para hacerlo positivo.
- Multiplicar igual que sin signo.
- Si el signo del resultado debe ser negativo:
  - ✓ Hacer el complemento a 2 del resultado obtenido.

27

## Operación de multiplicar, con signo

$$0010_2 \times 1100_2 = 2_{10} \times -4_{10} \left\{ \begin{array}{l} \text{Signo resultado: Negativo (PxN)} \\ \text{C2 de } (1100_2) = 0100_2 \end{array} \right.$$

$\begin{array}{r} \uparrow \quad \quad \uparrow \\ P \quad \quad N \\ 0010 \times 1100 \\ \hline 0000 \\ 0000 \\ 0010 \\ 0000 \\ \hline 0001000 \\ \hline 1111000 \end{array}$

$P \rightarrow 0010 \quad (2)$   
 $P \rightarrow \times 0100 \quad (4)$   
 $\quad \quad \quad 0000 \quad (8)$   
 $\quad \quad \quad 1111000 \quad (-8)$

28

## Operación de multiplicar, con signo

- **Aritmética con signo:**

- Hacer las multiplicaciones parciales igual que sin signo, pero extender en signo los resultados.
- Si el MSB del multiplicador es -1, el multiplicador es negativo, por lo que el resultado parcial último no será el multiplicando, sino el complemento a 2 del mismo.
- El producto final es la suma de los productos parciales.

29

## Operación de multiplicar, con signo

$$0010_2 \times 1100_2 = 2_{10} \times -4_{10} \quad 1011_2 \times 0010_2 = -5_{10} \times 2_{10}$$

	0 0 1 0	(2)		1 0 1 1	(-5)
	x 1 1 0 0	(-4)		x 0 0 1 0	(2)
	<u>0 0 0 0 0 0 0 0</u>	$(2 \cdot (0 \cdot 2^0))$		<u>0 0 0 0 0 0 0 0</u>	$(-5 \cdot (0 \cdot 2^0))$
	0 0 0 0 0 0 0 0	$(2 \cdot (0 \cdot 2^1))$		1 1 1 1 0 1 1	$(-5 \cdot (1 \cdot 2^1))$
	0 0 0 0 1 0	$(2 \cdot (1 \cdot 2^2))$		0 0 0 0 0 0	$(-5 \cdot (0 \cdot 2^2))$
C2(0010) →	<u>1 1 1 1 0</u>	$(2 \cdot (-1 \cdot 2^3))$		<u>0 0 0 0 0</u>	$(-5 \cdot (0 \cdot 2^3))$
	1 1 1 1 1 0 0 0	(-8)		1 1 1 1 0 1 1 0	(-10)

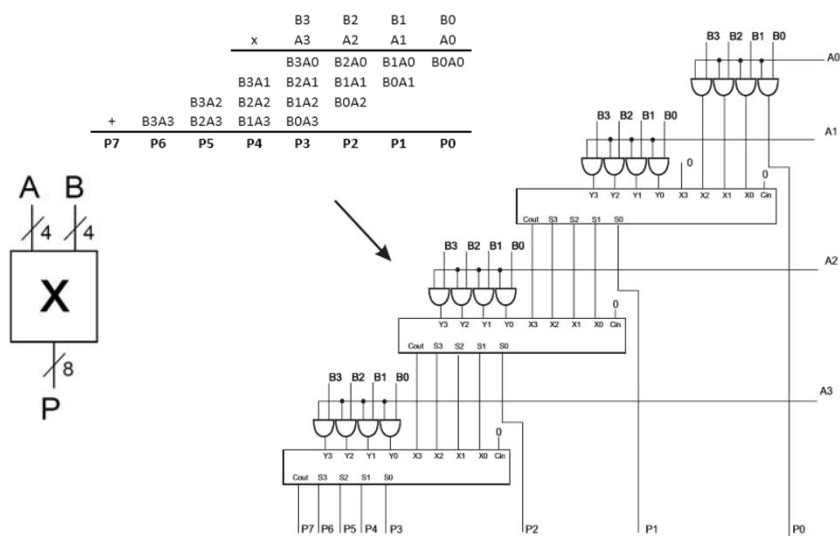
30

## Operación de multiplicar, con signo

<u>Sin signo</u>		<u>Con signo</u>
1 1 0 0 0 1 (49)		1 1 0 0 0 1 (- 15)
x 1 1 0 1 0 (26)		x 1 1 0 1 0 (- 6)
0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0
1 1 0 0 0 1		1 1 1 1 1 1 0 0 0 1
0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 0
1 1 0 0 0 1		1 1 1 1 0 0 0 1
1 1 0 0 0 1		0 0 0 1 1 1 1
1 0 0 1 1 1 1 0 1 0 (1274)		0 0 0 0 1 0 1 1 0 1 0 (+ 90)

31

## Multiplicador 4 x 4 (sin signo)



32



## **Unidad 2: La Unidad Aritmético Lógica (ALU)**

Escuela Politécnica Superior - UAM