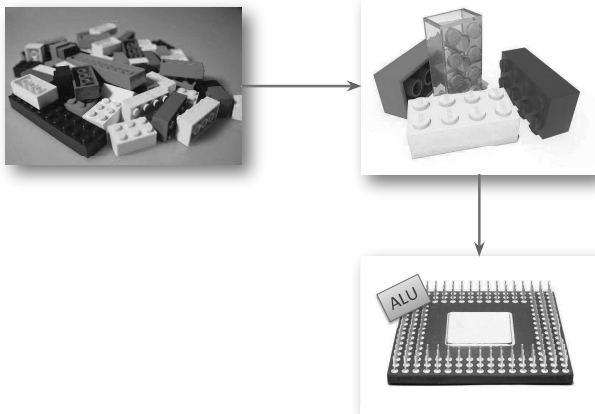




VJ1214 - Consolas y dispositivos

Bloque 1: Arquitectura de computadores

Aritmética booleana



Índice de contenidos

1. Suma de números binarios
2. Números binarios negativos
3. Sumadores
4. ALU

Libros de referencia

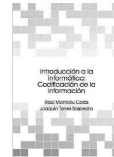
- Capítulo 2

<http://nand2tetris.org/chapters/chapter%2002.pdf>



- Capítulos 2 y 5

<https://montoliu.github.io/docs/LibroMOOCCodInfo.pdf>



- Capítulos 2 y 6



1. Suma de números binarios

1. Suma de números binarios

Primer sumando	Segundo sumando	Resultado	Acarreo
0	0	0	No
0	1	1	No
1	0	1	No
1	1	0	Si

1. Suma de números binarios

$$\begin{array}{r}
 1 \quad 0 \quad 0 \quad 1 \\
 + \quad 0 \quad 0 \quad 1 \quad 1 \\
 \hline
 \boxed{?} \boxed{?} \boxed{?} \boxed{?}
 \end{array}$$

1. Suma de números binarios

$$\begin{array}{r}
 1 \quad 0 \quad 0 \quad 1 \\
 \quad \quad +1 \quad +1 \\
 + \quad 0 \quad 0 \quad 1 \quad 1 \\
 \hline
 \boxed{?} \boxed{1} \boxed{0} \boxed{0}
 \end{array}$$

1. Suma de números binarios

$$\begin{array}{r}
 1 \quad 0 \quad 0 \quad 1 \\
 \quad \quad +1 \\
 + \quad 0 \quad 0 \quad 1 \quad 1 \\
 \hline
 \boxed{?} \boxed{?} \boxed{?} \boxed{0}
 \end{array}$$

1. Suma de números binarios

$$\begin{array}{r}
 1 \quad 0 \quad 0 \quad 1 \\
 \quad \quad +1 \quad +1 \\
 + \quad 0 \quad 0 \quad 1 \quad 1 \\
 \hline
 \boxed{1} \boxed{1} \boxed{0} \boxed{0}
 \end{array}$$

1. Suma de números binarios

$$\begin{array}{r}
 1 \quad 0 \quad 0 \quad 1 \\
 \quad \quad +1 \quad +1 \\
 + \quad 0 \quad 0 \quad 1 \quad 1 \\
 \hline
 \boxed{?} \boxed{?} \boxed{0} \boxed{0}
 \end{array}$$

1. Suma de números binarios

- ¿Qué ocurre si la suma de los dos dígitos más representativos produce acarreo?

$$\begin{array}{r}
 \quad +1 \quad +1 \quad +1 \\
 1 \quad 0 \quad 0 \quad 1 \\
 1 \quad 0 \quad 1 \quad 1 \\
 \hline
 \boxed{?} \boxed{0} \boxed{1} \boxed{0} \boxed{0}
 \end{array}$$

1. Suma de números binarios

- ¿Qué ocurre si la suma de los dos dígitos representativos produce acarreo?

Desbordamiento (overflow)

$$\begin{array}{rcccc} & +1 & & +1 & +1 \\ & 1 & 0 & 0 & 1 \\ + & 1 & 0 & 1 & 1 \\ \hline \boxed{?} & \boxed{0} & \boxed{1} & \boxed{0} & \boxed{0} \end{array}$$

1. Suma de números binarios

$$\begin{array}{rcccc} & & +1 & +1 & \\ & 1 & 0 & 0 & 1 \\ + & 0 & 0 & 1 & 1 \\ \hline \boxed{1} & \boxed{1} & \boxed{0} & \boxed{0} \end{array}$$

La suma en binario es en realidad una suma de tres elementos:

1. el primer sumando
2. el segundo sumando
3. el acarreo de la suma anterior

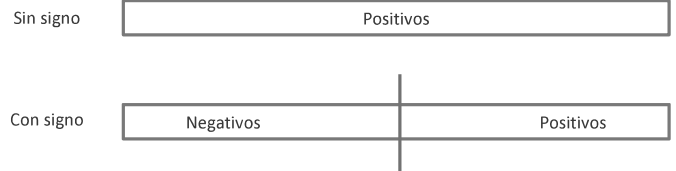
2. Números binarios con signo

2. Números binarios con signo

- Los números binarios con signo se representan usando un sistema de codificación llamado Complemento a 2.

2. Números binarios con signo

- Con el mismo número de bits el número positivo máximo se reduce a la mitad.



Complemento a 2

- Dado un número x :

- Si es **positivo**:

- Binario

- Si es **negativo**, tres pasos:

- Convertir $|x|$ a binario
- Cambiar 0s por 1s y viceversa
- Sumar 1

Complemento a 2

- Expresa el número 3_{10} en Complemento a 2 con $N=4$:

N es el número de bits del número

Complemento a 2

- Expresa el número 3_{10} en Complemento a 2 con $N=4$:

Es positivo

$$3_{10} \equiv 0011_2$$

Complemento a 2

- Expresa el número 3_{10} en Complemento a 2 con $N=4$:

Es positivo

$$3_{10} \equiv 0011_2$$

- El resultado es: $3_{10} \equiv 0011_2$

Complemento a 2

- Expresa el número -3_{10} en Complemento a 2 con $N=4$:

Complemento a 2

- Expresa el número -3_{10} en Complemento a 2 con $N=4$:

- Es negativo

- Paso 1: $|-3_{10}| = 3_{10} \equiv 0011_2$

- Paso 2: $0011_2 \rightarrow 1100_2$

- Paso 3: $1100_2 + 1_2 = 1101_2$

- El resultado es $-3_{10} \equiv 1101_2$

Complemento a 2

- Rango:

$$[-2^{N-1}, +2^{N-1} - 1]$$

Complemento a 2

- Con $N=4$, **sin signo**

$$[0, +2^N-1] \quad [0, 15]$$

- Con $N=4$, **con signo**, en Ca2

$$[-2^{N-1}, +2^{N-1} - 1] \quad [-8, +7]$$

Complemento a 2

- Restar $A - B$, es equivalente a sumar A y el Ca2 de B .

- Veremos que esta propiedad es fundamental en el diseño de la ALU.

Complemento a 2

- Con $N=8$, **sin signo**

$$[0, +2^N-1] \quad [0, 255]$$

- Con $N=8$, **con signo**, en Ca2

$$[-2^{N-1}, +2^{N-1} - 1] \quad [-128, +127]$$

Complemento a 2

- $A=5_{10}$, $B=3_{10}$ ¿ $A-B$? con $N=4$

Complemento a 2

- Pregunta: ¿Para qué se usa?

- **Positivo:**

- Los positivos empiezan por "0" y los negativos por "1".

- Única representación del cero.

- **Facilita** el diseño de los procesadores.

Complemento a 2

- $A=5_{10}$, $B=3_{10}$ ¿ $A-B$? con $N=4$

$$\begin{aligned} 5_{10} &\equiv 0101_2 \\ -3_{10} &\equiv 1101_2 \quad (\text{en Ca2}) \end{aligned}$$

Complemento a 2

• $A=5_{10}$, $B=3_{10}$ ¿A-B? con N=4

$$\begin{array}{r} 5_{10} \equiv 0101_2 \\ -3_{10} \equiv 1101_2 \text{ (en Ca2)} \\ \hline \end{array} \quad \begin{array}{r} 0 \ 1 \ 0 \ 1 \\ + 1 \ 1 \ 0 \ 1 \\ \hline \end{array}$$

Complemento a 2

• $A=5_{10}$, $B=3_{10}$ ¿A-B? con N=4

Hay que descartar el último acarreo

$$\begin{array}{r} 5_{10} \equiv 0101_2 \\ -3_{10} \equiv 1101_2 \text{ (en Ca2)} \\ \hline \end{array} \quad \begin{array}{r} \overset{+1}{0} \ \overset{+1}{1} \ \overset{+1}{0} \ 1 \\ + 1 \ 1 \ 0 \ 1 \\ \hline \end{array}$$

$5_{10} - 3_{10} = 2_{10}$
 $2_{10} \equiv 0010_2$

Complemento a 2

• $A=5_{10}$, $B=3_{10}$ ¿A-B? con N=4

$$\begin{array}{r} 5_{10} \equiv 0101_2 \\ -3_{10} \equiv 1101_2 \text{ (en Ca2)} \\ \hline \end{array} \quad \begin{array}{r} \overset{+1}{0} \ \overset{+1}{1} \ \overset{+1}{0} \ 1 \\ + 1 \ 1 \ 0 \ 1 \\ \hline \end{array}$$

\swarrow 0 0 1 0

Complemento a 2

• $A=2_{10}$, $B=3_{10}$ ¿A-B? con N=4

Complemento a 2

• $A=5_{10}$, $B=3_{10}$ ¿A-B? con N=4

$$\begin{array}{r} 5_{10} \equiv 0101_2 \\ -3_{10} \equiv 1101_2 \text{ (en Ca2)} \\ \hline \end{array} \quad \begin{array}{r} \overset{+1}{0} \ \overset{+1}{1} \ \overset{+1}{0} \ 1 \\ + 1 \ 1 \ 0 \ 1 \\ \hline \end{array}$$

\swarrow 0 0 1 0

$5_{10} - 3_{10} = 2_{10}$
 $2_{10} \equiv 0010_2$

Complemento a 2

• $A=2_{10}$, $B=3_{10}$ ¿A-B? con N=4

$$\begin{array}{r} 2_{10} \equiv 0010_2 \\ -3_{10} \equiv 1101_2 \text{ (en Ca2)} \end{array}$$

Complemento a 2

• $A=2_{10}$, $B=3_{10}$ ¿A-B? con N=4

$$\begin{array}{r} 2_{10} \equiv 0010_2 \\ -3_{10} \equiv 1101_2 \text{ (en Ca2)} \\ \hline \end{array} \quad \begin{array}{r} 0010 \\ + 1101 \\ \hline \end{array}$$

• $A=2_{10}$, $B=3_{10}$ ¿A-B? con N=4

No es necesario descartar el último acarreo

$$\begin{array}{r} 2_{10} \equiv 0010_2 \\ -3_{10} \equiv 1101_2 \text{ (en Ca2)} \\ \hline \end{array} \quad \begin{array}{r} 0010 \\ + 1101 \\ \hline \end{array}$$

$$2_{10} - 3_{10} = -1_{10}$$

$$-1_{10} \equiv 1111_2 \text{ (en Ca2)}$$

Complemento a 2

• $A=2_{10}$, $B=3_{10}$ ¿A-B? con N=4

$$\begin{array}{r} 2_{10} \equiv 0010_2 \\ -3_{10} \equiv 1101_2 \text{ (en Ca2)} \\ \hline \end{array} \quad \begin{array}{r} 0010 \\ + 1101 \\ \hline \end{array}$$

Complemento a 2

Decimal	Binario
-8	1000
-7	1001
-6	1010
-5	1011
-4	1100
-3	1101
-2	1110
-1	1111
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

N=4 **N=8**

$$3_{10} \equiv 0011_2 \quad 3_{10} \equiv 0000\ 0011_2$$

Complemento a 2

• $A=2_{10}$, $B=3_{10}$ ¿A-B? con N=4

$$\begin{array}{r} 2_{10} \equiv 0010_2 \\ -3_{10} \equiv 1101_2 \text{ (en Ca2)} \\ \hline \end{array} \quad \begin{array}{r} 0010 \\ + 1101 \\ \hline \end{array}$$

$$2_{10} - 3_{10} = -1_{10}$$

$$-1_{10} \equiv 1111_2 \text{ (en Ca2)}$$

Complemento a 2

Decimal	Binario
-8	1000
-7	1001
-6	1010
-5	1011
-4	1100
-3	1101
-2	1110
-1	1111
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

N=4 **N=8**

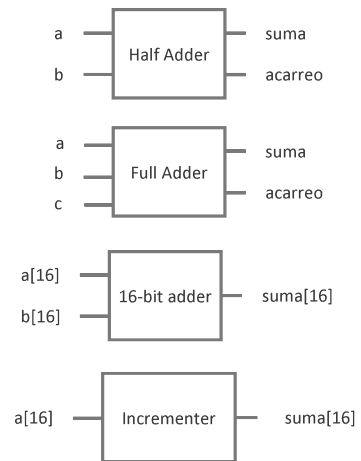
$$3_{10} \equiv 0011_2 \quad 3_{10} \equiv 0000\ 0011_2$$

$$5_{10} \equiv 0101_2 \quad 5_{10} \equiv 0000\ 0101_2$$

Complemento a 2

Decimal	Binario
-8	1000
-7	1001
-6	1010
-5	1011
-4	1100
-3	1101
-2	1110
-1	1111
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

$$\begin{array}{ll}
 \text{N=4} & \text{N=8} \\
 3_{10} \equiv 0011_2 & 3_{10} \equiv 0000\ 0011_2 \\
 5_{10} \equiv 0101_2 & 5_{10} \equiv 0000\ 0101_2 \\
 -2_{10} \equiv 1110_2 & -2_{10} \equiv 1111\ 1110_2
 \end{array}$$



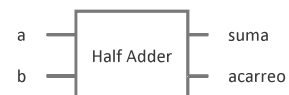
Complemento a 2

Decimal	Binario
-8	1000
-7	1001
-6	1010
-5	1011
-4	1100
-3	1101
-2	1110
-1	1111
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

$$\begin{array}{ll}
 \text{N=4} & \text{N=8} \\
 3_{10} \equiv 0011_2 & 3_{10} \equiv 0000\ 0011_2 \\
 5_{10} \equiv 0101_2 & 5_{10} \equiv 0000\ 0101_2 \\
 -2_{10} \equiv 1110_2 & -2_{10} \equiv 1111\ 1110_2 \\
 -6_{10} \equiv 1010_2 & -6_{10} \equiv 1111\ 1010_2
 \end{array}$$

$$\begin{array}{rcccc}
 & & +1 & +1 & \\
 & 1 & 0 & 0 & 1 \\
 + & 0 & 0 & 1 & 1 \\
 \hline
 & 1 & 1 & 0 & 0
 \end{array}$$

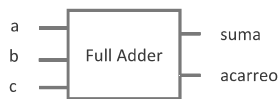
3. Sumadores



a	b	suma	acarreo
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

3. Sumadores

3. Sumadores



a	b	c	suma	acarreo
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

3. Sumadores



a	b	suma	acarreo
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

3. Sumadores



- Realiza la suma entre **a** y **b**
- **a** y **b** son números binarios de 16 bits
- El resultado es un número de 16 bits
- No se tiene en cuenta el acarreo final

3. Sumadores



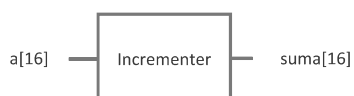
a	b	suma	acarreo
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



Ahora es tu turno.

Inténtalo varias veces antes de continuar con la siguiente diapositiva

3. Sumadores



- Realiza la suma entre **a** y 1
- **a** es un número binario de 16 bits
- El resultado es un número binario de 16 bits
- No se tiene en cuenta el acarreo final





3. Sumadores

a

b

Half Adder

suma

acarreo

$\text{suma} = \bar{a}b + a\bar{b} = \text{xor}(a,b)$

$\text{acarreo} = ab$

a	b	suma	acarreo
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

a

b

XOR

AND

suma

acarreo

3. Sumadores

a

b

Half Adder

suma

acarreo

$\text{suma} = \bar{a}b + a\bar{b}$

$\text{acarreo} = ab$

a	b	suma	acarreo
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

3. Sumadores

a

b

c

Full Adder

suma

acarreo

a	b	c	suma	acarreo
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

3. Sumadores

a

b

Half Adder

suma

acarreo

$\text{suma} = \bar{a}b + a\bar{b} = \text{xor}(a,b)$

$\text{acarreo} = ab$

a	b	suma	acarreo
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

3. Sumadores

a

b


c

Full Adder

suma

acarreo

a	b	c	suma	acarreo
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

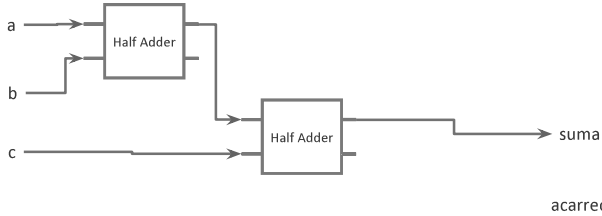
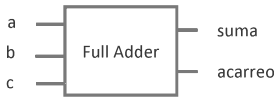


Ahora es tu turno.
Inténtalo varias veces antes de
continuar con la siguiente diapositiva



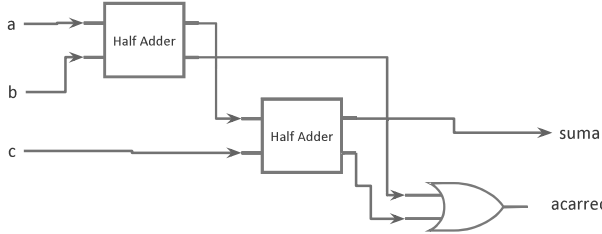
Si estás usando mapas de Karnaugh **NO** vas por el mejor camino

3. Sumadores



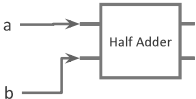
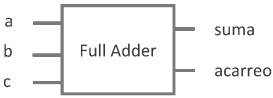
Si estás usando mapas de Karnaugh **NO** vas por el mejor camino

3. Sumadores



Usa lo que acabas de construir.

3. Sumadores



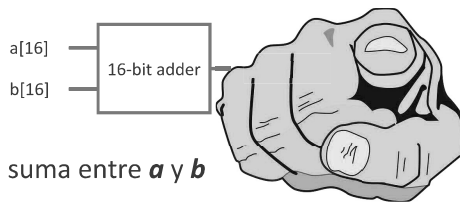
suma
acarreo

3. Sumadores



- Realiza la suma entre **a** y **b**
- a** y **b** son números binarios de 16 bits
- El resultado es un número de 16 bits
- No se tiene en cuenta el acarreo final

3. Sumadores

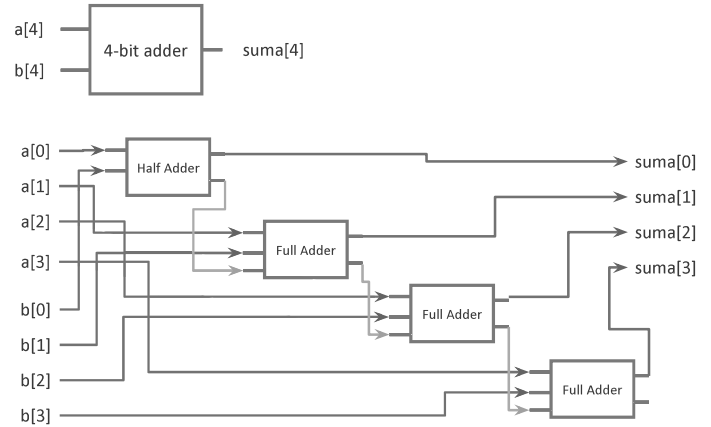


- Realiza la suma entre **a** y **b**
- **a** y **b** son números binarios de 16 bits

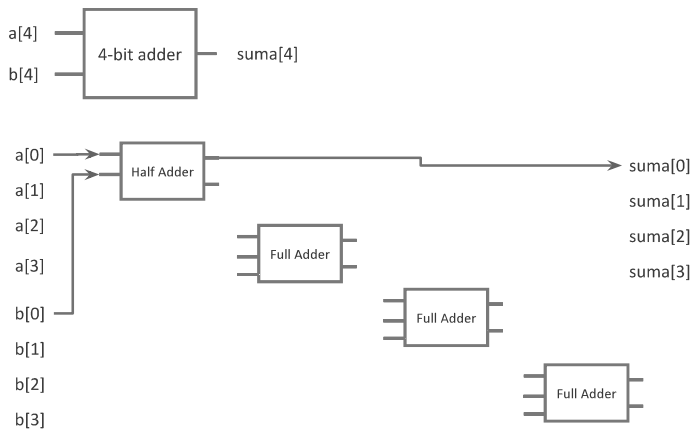
- El resultado es un número binario de 16 bits
- No se tiene en cuenta el acarreo final

Ahora es tu turno.
Inténtalo varias veces antes de continuar con la siguiente diapositiva

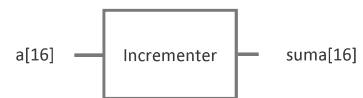
3. Sumadores



3. Sumadores

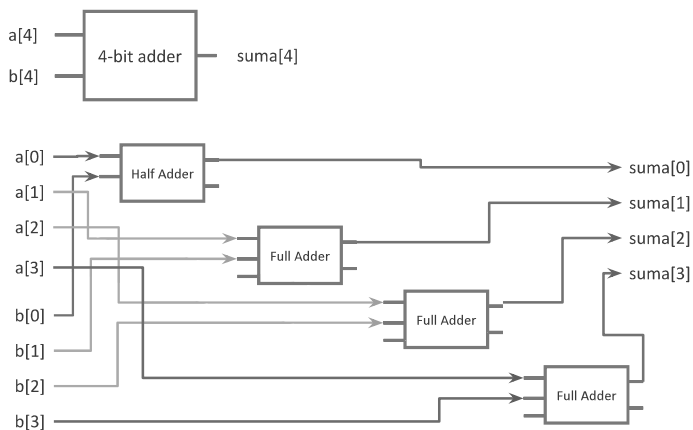


3. Sumadores

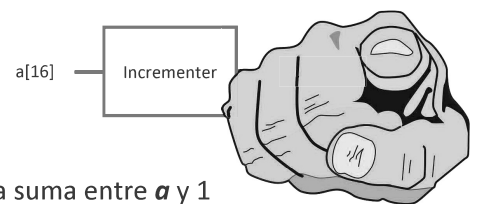


- Realiza la suma entre **a** y 1
- **a** es un número binario de 16 bits
- El resultado es un número binario de 16 bits
- No se tiene en cuenta el acarreo final

3. Sumadores



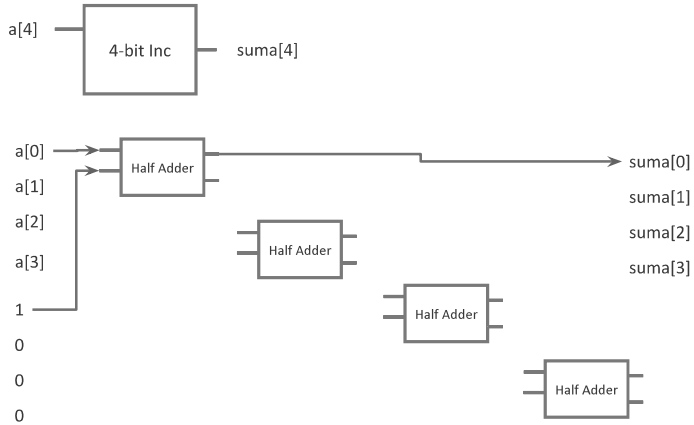
3. Sumadores



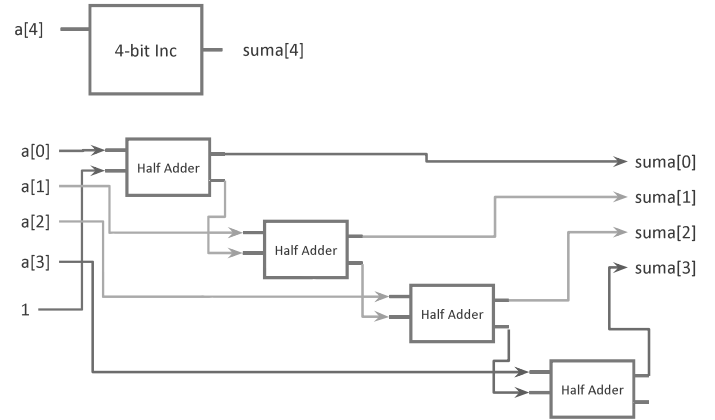
- Realiza la suma entre **a** y 1
- **a** es un número binario de 16 bits
- El resultado es un número binario de 16 bits
- No se tiene en cuenta el acarreo final

Ahora es tu turno.
Inténtalo varias veces antes de continuar con la siguiente diapositiva

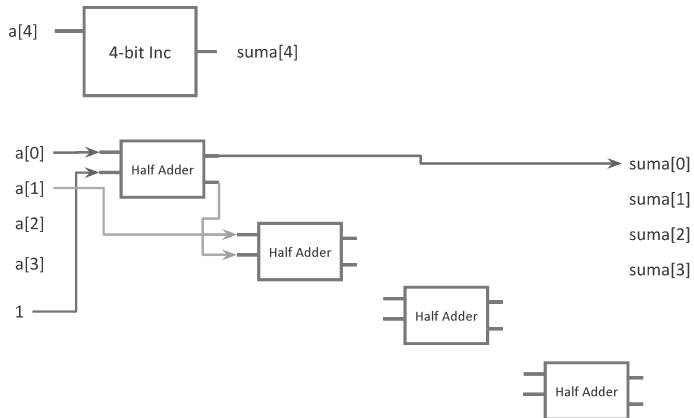
3. Sumadores



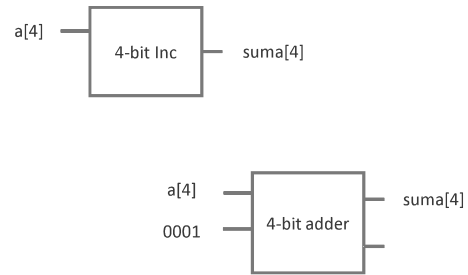
3. Sumadores



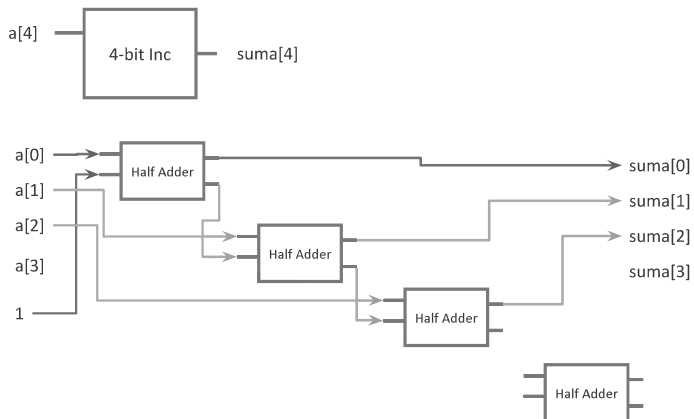
3. Sumadores



3. Sumadores

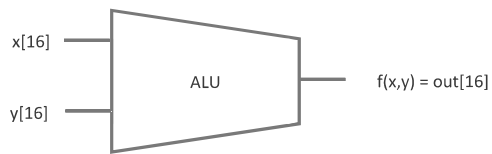


3. Sumadores



4. ALU

4. ALU

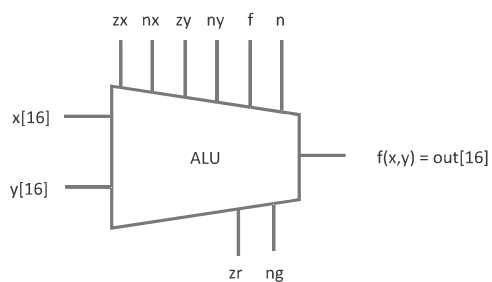


4. ALU

•ATENCIÓN:

- Este diseño de la ALU no detecta el desbordamiento.

4. ALU



4. ALU

zx	nx	zy	ny	f	no	out
1	0	1	0	1	0	0
1	1	1	1	1	1	1
1	1	1	0	1	0	-1
0	0	1	1	0	0	x
1	1	0	0	0	0	y
0	0	1	1	0	1	!x
1	1	0	0	0	1	!y
0	0	1	1	1	1	-x
1	1	0	0	1	1	-y

4. ALU

• Entradas

- x: primer operando
- y: segundo operando
- zx: poner a cero x
- nx: negar x (cambiar 0's por 1's)
- zy: poner a cero y
- ny: negar y (cambiar 0's por 1's)
- f: 1 para ADD, 0 para AND
- no: negar la salida (cambiar 0's por 1's)

• Salidas

- out: salida
- zr: 1 si out es igual a cero
- ng: 1 si out es negativo

4. ALU

zx	nx	zy	ny	f	no	out
0	1	1	1	1	1	x+1
1	1	0	1	1	1	y+1
0	0	1	1	1	0	x-1
1	1	0	0	1	0	y-1
0	0	0	0	1	0	x+y
0	1	0	0	1	1	x-y
0	0	0	1	1	1	y-x
0	0	0	0	0	0	x&y
0	1	0	1	0	1	x y

4. ALU



zx	nx	zy	ny	f	no	out
0	1	1	1	1	1	x+1
1	1	0	1	1	1	y+1
0	0	1	1	1	0	x-1
1	1	0	0	1	0	y-1
0	0	0	0	1	0	x+y
0	1	0	0	1	1	x-y
0	0	0	1	1	1	y-x
0	0	0	0	0	0	x&y
0	1	0	1	0	1	x y

4. ALU



zx	nx	zy	ny	f	no	out
...
0	1	0	0	1	1	x-y
...

- **x** se niega (**nx**)
- **y** permanece igual
- **out** se niega (**no**), después de realizar la suma
- Ejemplo x=7, y=1
 - x -> 0000 0000 0000 0111
 - y -> 0000 0000 0000 0001

4. ALU



zx	nx	zy	ny	f	no	out
0	1	1	1	1	1	x+1
1	1	0	1	1	1	y+1
0	0	1	1	1	0	x-1
1	1	0	0	1	0	y-1
0	0	0	0	1	0	x+y
0	1	0	0	1	1	x-y
0	0	0	1	1	1	y-x
0	0	0	0	0	0	x&y
0	1	0	1	0	1	x y

4. ALU



zx	nx	zy	ny	f	no	out
...
0	1	0	0	1	1	x-y
...

- **x** se niega (**nx**)
- **y** permanece igual
- **out** se niega (**no**), después de realizar la suma
- Ejemplo x=7, y=1
 - x -> 0000 0000 0000 0111 -> 1111 1111 1111 1000
 - y -> 0000 0000 0000 0001 -> 0000 0000 0000 0001

4. ALU



zx	nx	zy	ny	f	no	out
...
0	0	1	1	1	0	x-1
...

- **x** permanece sin cambios
- **y** primero se hace cero (**zy**) y luego se niega (**ny**).
 - y -> 0000 0000 0000 0000
 - y -> 1111 1111 1111 1111 (es el número -1)
- **f** es 1, la función es suma:
 - out = x + (-1)

4. ALU



zx	nx	zy	ny	f	no	out
...
0	1	0	0	1	1	x-y
...

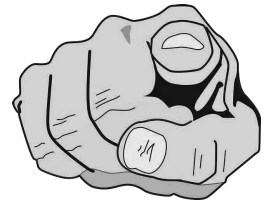
- **x** se niega (**nx**)
- **y** permanece igual
- **out** se niega (**no**), después de realizar la suma
- Ejemplo x=7, y=1
 - x -> 0000 0000 0000 0111 -> 1111 1111 1111 1000
 - y -> 0000 0000 0000 0001 -> 0000 0000 0000 0001
 - $$\begin{array}{r} 1111\ 1111\ 1111\ 1000 \\ +\ 0000\ 0000\ 0000\ 0001 \\ \hline 1111\ 1111\ 1111\ 1001 \end{array}$$

4. ALU



zx	nx	zy	ny	f	no	out
...
0	1	0	0	1	1	x-y
...

- **x** se niega (**nx**)
- **y** permanece igual
- **out** se niega (**no**), después de realizar la suma
- Ejemplo x=7, y=1
 - x -> 0000 0000 0000 0111 -> 1111 1111 1111 1000 +
 - y -> 0000 0000 0000 0001 -> 0000 0000 0000 0001
 - 1111 1111 1111 1001
 - Negar out -> 0000 0000 0000 0110



Ahora es tu turno.

Inténtalo varias veces antes de mirar la solución en Internet

4. ALU



zx	nx	zy	ny	f	no	out
...
0	1	1	1	1	1	x+1
...

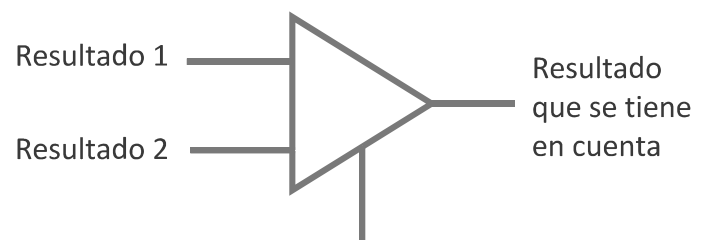
- **x** se niega (**nx**)
- **y** se pone a cero y se niega (será todo unos)
- **out** se niega (**no**), después de realizar la suma
- Ejemplo x=5
 - x -> 0000 0000 0000 0101 -> 1111 1111 1111 1010 +
 - y -> 1111 1111 1111 1111
 - 1111 1111 1111 1001
 - Negar out -> 0000 0000 0000 0110



4. ALU

¿Cómo implementar la ALU?:

- Proyecto 02
- Ayuda:
 - Harán falta las siguientes puertas ya implementadas en el proyecto 01:
 - Mux16 (6)
 - Not16 (3)
 - And16 (1)
 - Add16 (1)
 - Or8Way (2) [ORi8o1b1]
 - Or (1)
 - Not (1)





Lo siento, pero no se proporciona la solución :-)