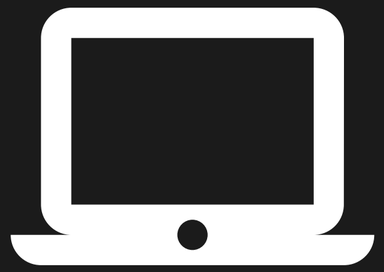
ARQUITECTURA DE COMPUTADORES

//01



Ayudantía 3

Tomás Contreras
Susana Figueroa
Andrés Gonzalez
Jorge Schenke
Sebastián Ramos
Rocío Hernández

06/09/2021

Opcodes

Delays

Pulsos

[Recordemos]

Opcodes

Codificaciones más cortas de las instrucciones en lenguaje de máquina. Evita que sobren tantas posibilidades de instrucciones.

[Recordemos]

Opcodes

Codificaciones más cortas de las instrucciones en lenguaje de máquina. Evita que sobren tantas posibilidades de instrucciones.

000000

[Recordemos]

Opcodes

Codificaciones más cortas de las instrucciones en lenguaje de máquina. Evita que sobren tantas posibilidades de instrucciones.

000000

1010 0000

[Recordemos]

Opcodes

Codificaciones más cortas de las instrucciones en lenguaje de máquina. Evita que sobren tantas posibilidades de instrucciones.

000000

↓

1010 0000

↓

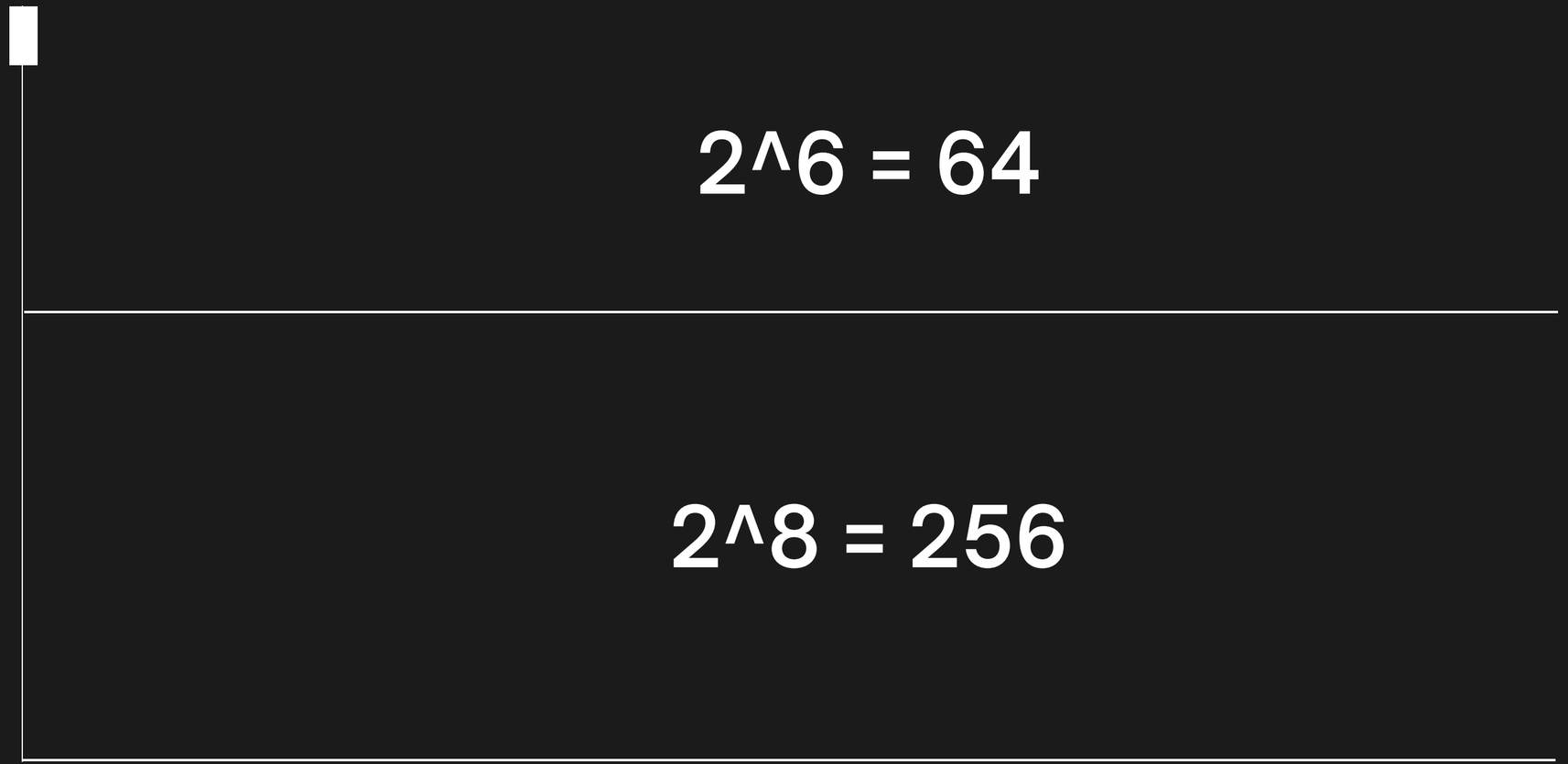
A = B

Opcode

2^6 = 64

Instrucción

2^8 = 256



Opcode	La	Lb	Sa0	Sb0	Sb1	Sop2	Sop1	Sop0	Operación
000000	1	0	1	0	0	0	0	0	A=B
000001	0	1	0	1	1	0	0	0	B=A
000010	1	0	0	0	1	0	0	0	A=Lit
000011	0	1	0	0	1	0	0	0	B=Lit
000100	1	0	0	0	0	0	0	0	A=A+B
000101	0	1	0	0	0	0	0	0	B=A+B
000110	1	0	0	0	1	0	0	0	A=A+Lit
000111	1	0	0	0	0	0	0	1	A=A-B
001000	0	1	0	0	0	0	0	1	B=A-B
001001	1	0	0	0	1	0	0	1	A=A-Lit
001010	1	0	0	0	0	0	1	0	A=A and B
001011	0	1	0	0	0	0	1	0	B=A and B
001100	1	0	0	0	1	0	1	0	A=A and Lit
001101	1	0	0	0	0	0	1	1	A=A or B
001110	0	1	0	0	0	0	1	1	B=A or B
001111	1	0	0	0	1	0	1	1	A=A or Lit
010000	1	0	0	0	0	1	0	0	A=notA
010001	0	1	0	0	0	1	0	0	B=notA
010010	1	0	0	0	1	1	0	0	A=notLit
010011	1	0	0	0	0	1	0	1	A=A xor B
010100	0	1	0	0	0	1	0	1	B=A xor B
010101	1	0	0	0	1	1	0	1	A=A xor Lit
010110	1	0	0	0	0	1	1	0	A=shift left A
010111	0	1	0	0	0	1	1	0	B=shift left A
011000	1	0	0	0	1	1	1	0	A=shift left Lit
011001	1	0	0	0	0	1	1	1	A=shift right A
011010	0	1	0	0	0	1	1	1	B=shift right A
011011	1	0	0	0	1	1	1	1	A=shift right Lit

[Recordemos]

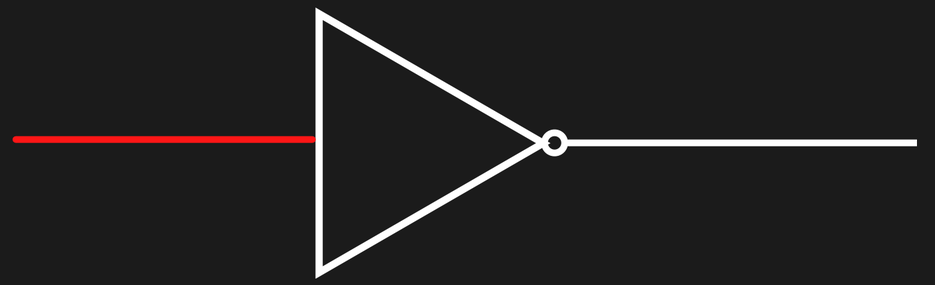
Delay

Efecto producido por el tiempo de ejecución de una compuerta

[Recordemos]

Delay

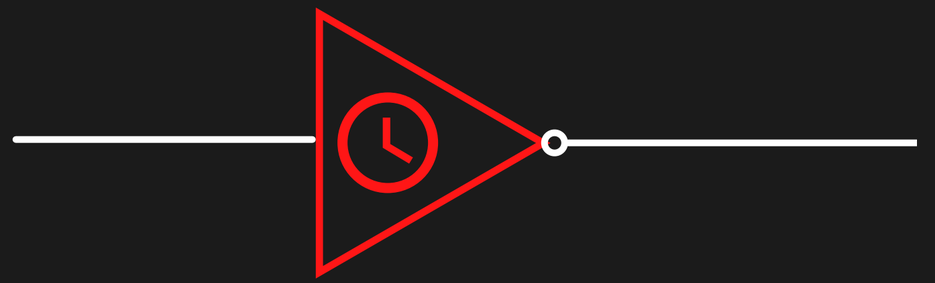
Efecto producido por el tiempo de ejecución de una compuerta



[Recordemos]

Delay

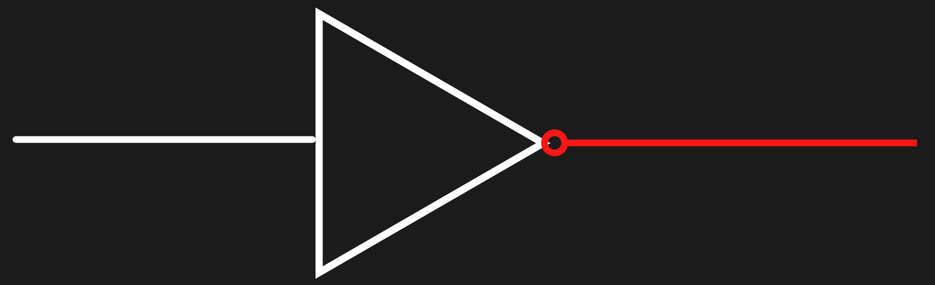
Efecto producido por el tiempo de ejecución de una compuerta



[Recordemos]

Delay

Efecto producido por el tiempo de ejecución de una compuerta



[Recordemos]

[Recordemos]

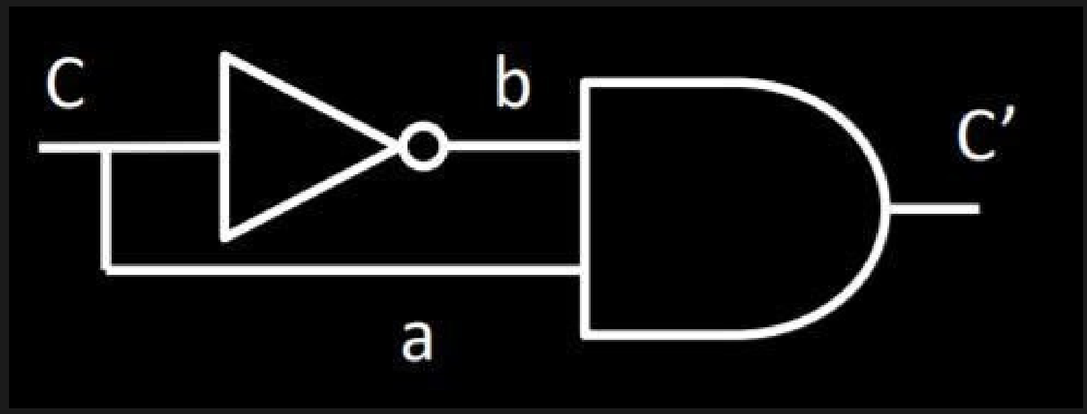
Generador de pulsos

Circuito que se aprovecha del delay para generar pulsos eléctricos extremaaaaaaaaaaaaaaaaadamente cortos.

[Recordemos]

Generador de pulsos

Circuito que se aprovecha del delay para generar pulsos eléctricos extremaaaaaaaaaaaaaaaaadamente cortos.

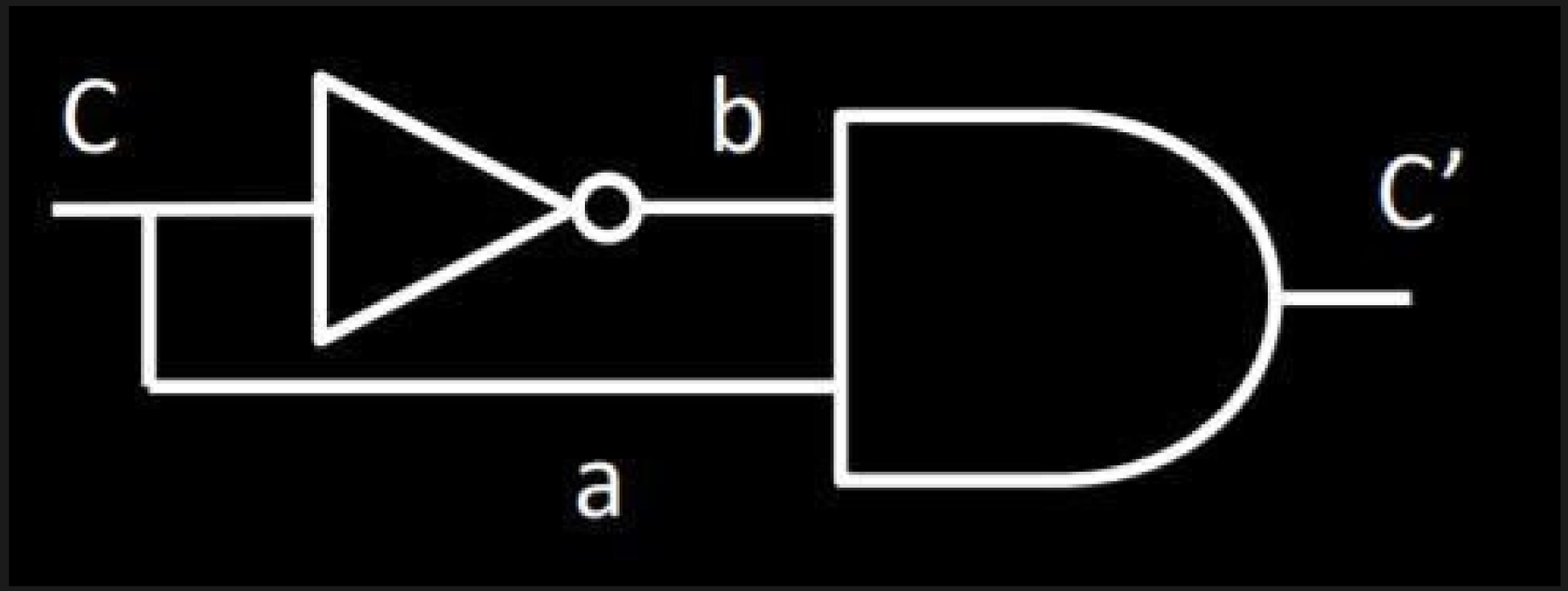


C = 0

b = 1

a = 0

C' = ?

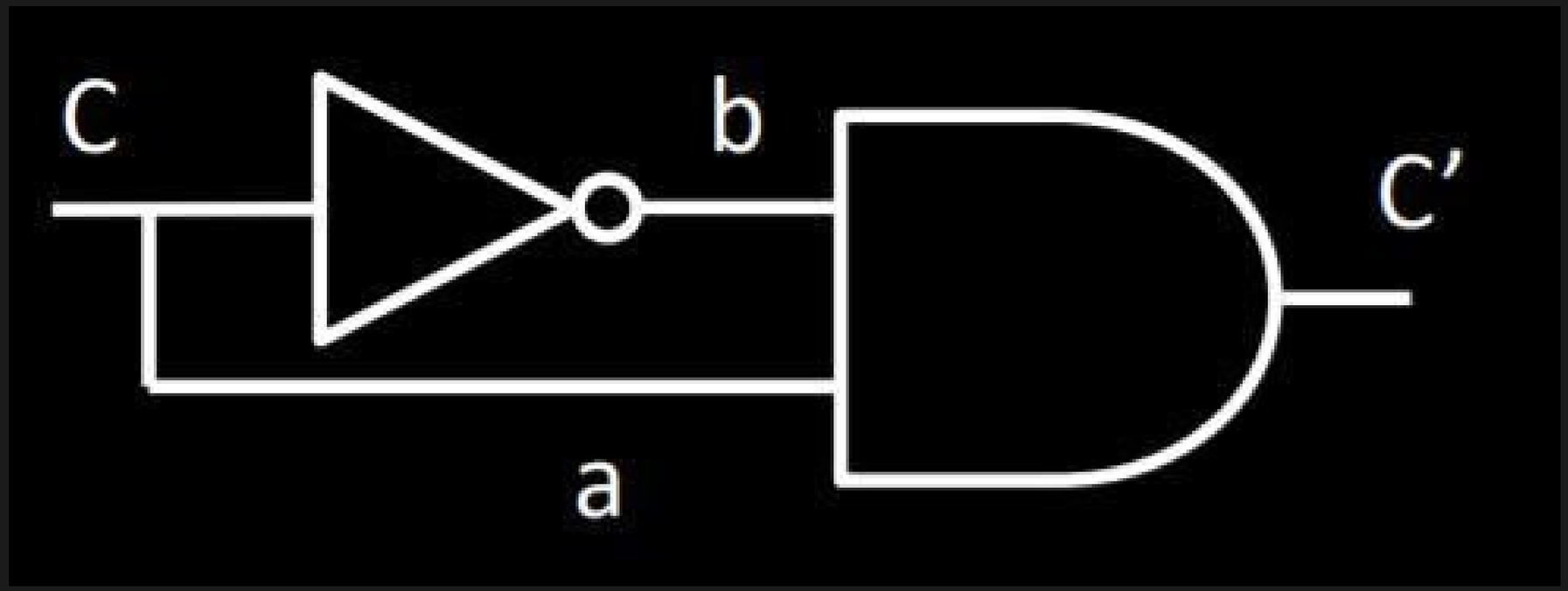


C = 0

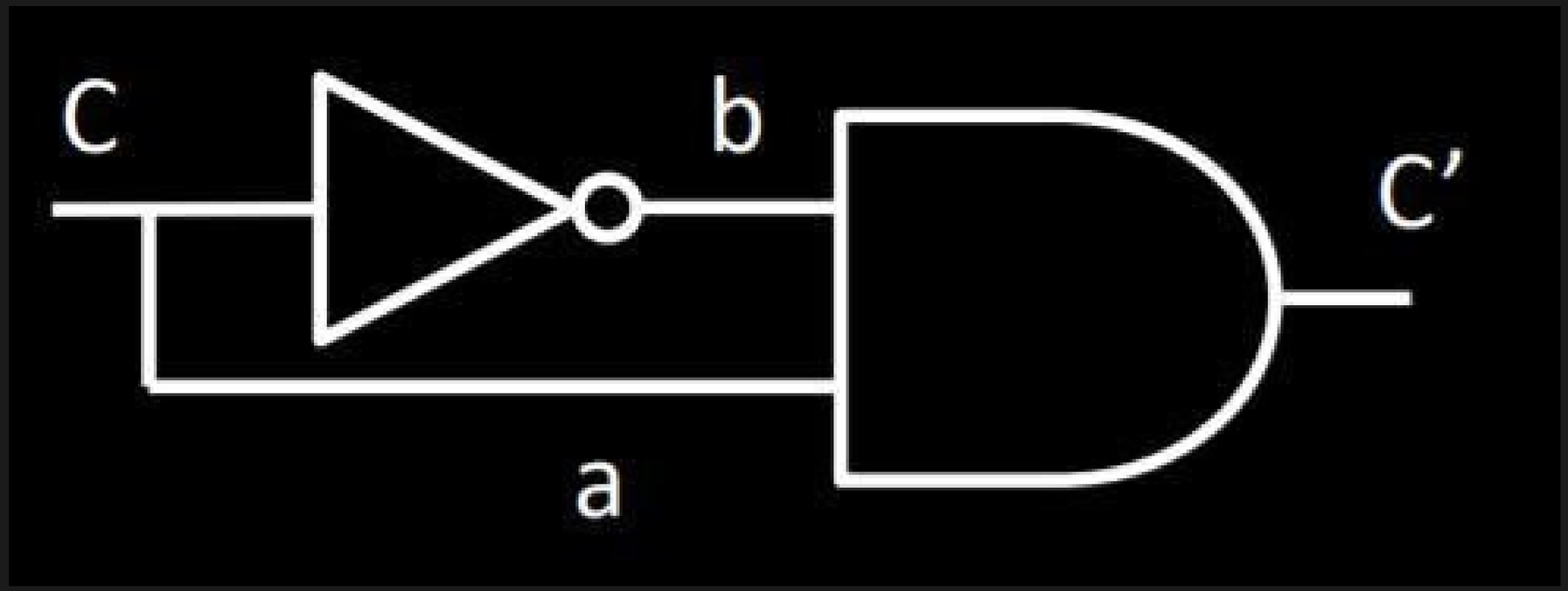
b = 1

a = 0

C' = 0



C = 0
b = 1
a = 0
C' = 0

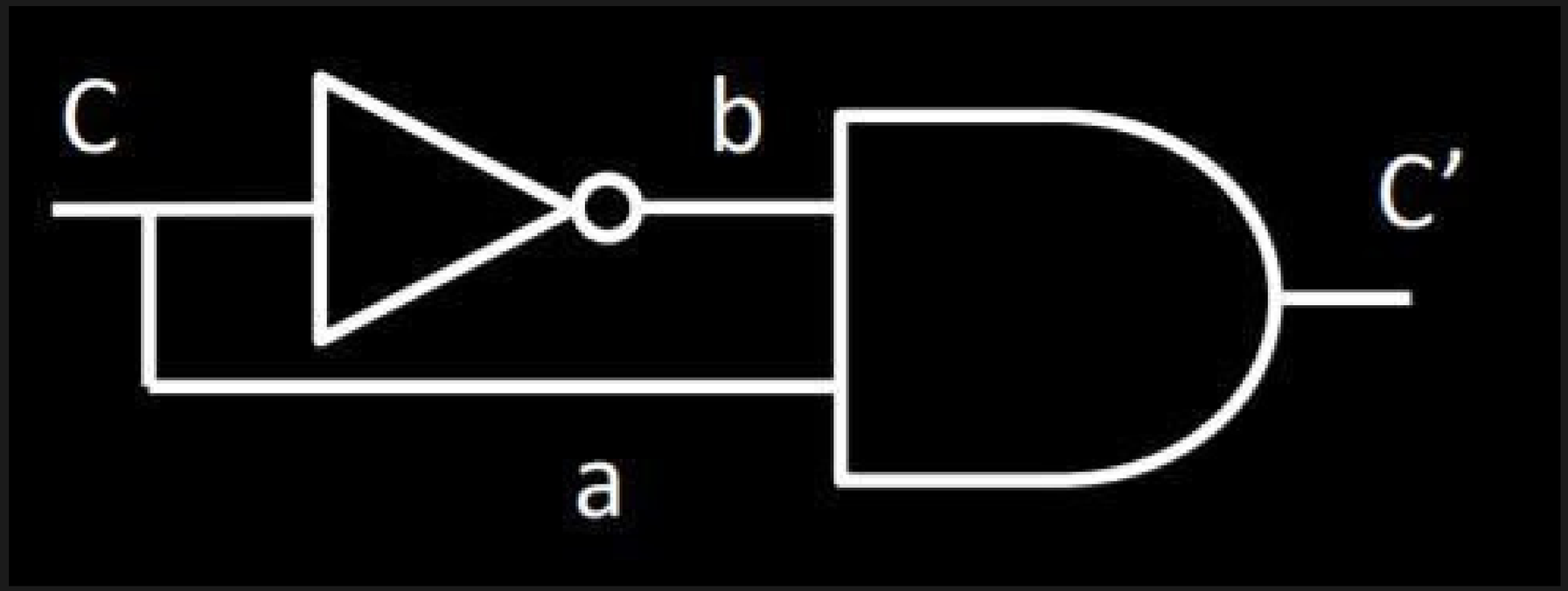


C = 1
b = 0
a = 1
C' = 0

C = 0
b = 1
a = 0
C' = 0

?

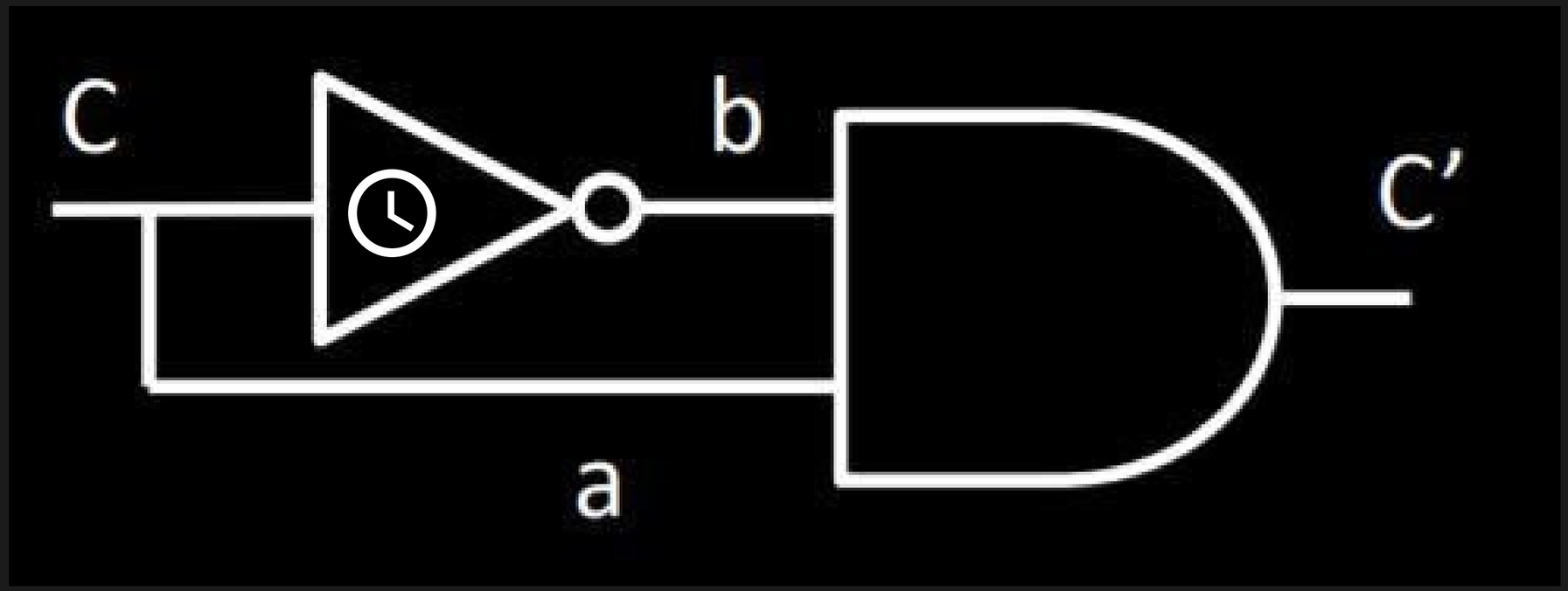
C = 1
b = 0
a = 1
C' = 0



C = 0
b = 1
a = 0
C' = 0

?

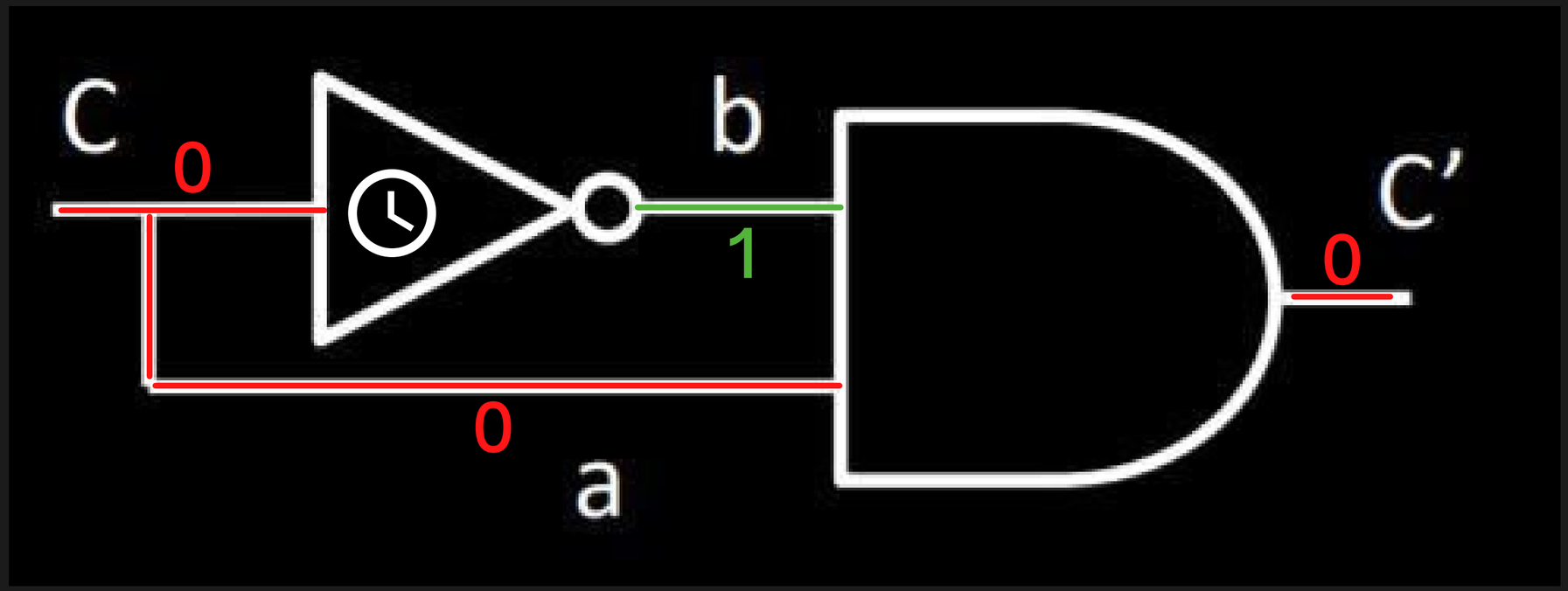
C = 1
b = 0
a = 1
C' = 0



C = 0
b = 1
a = 0
C' = 0

?

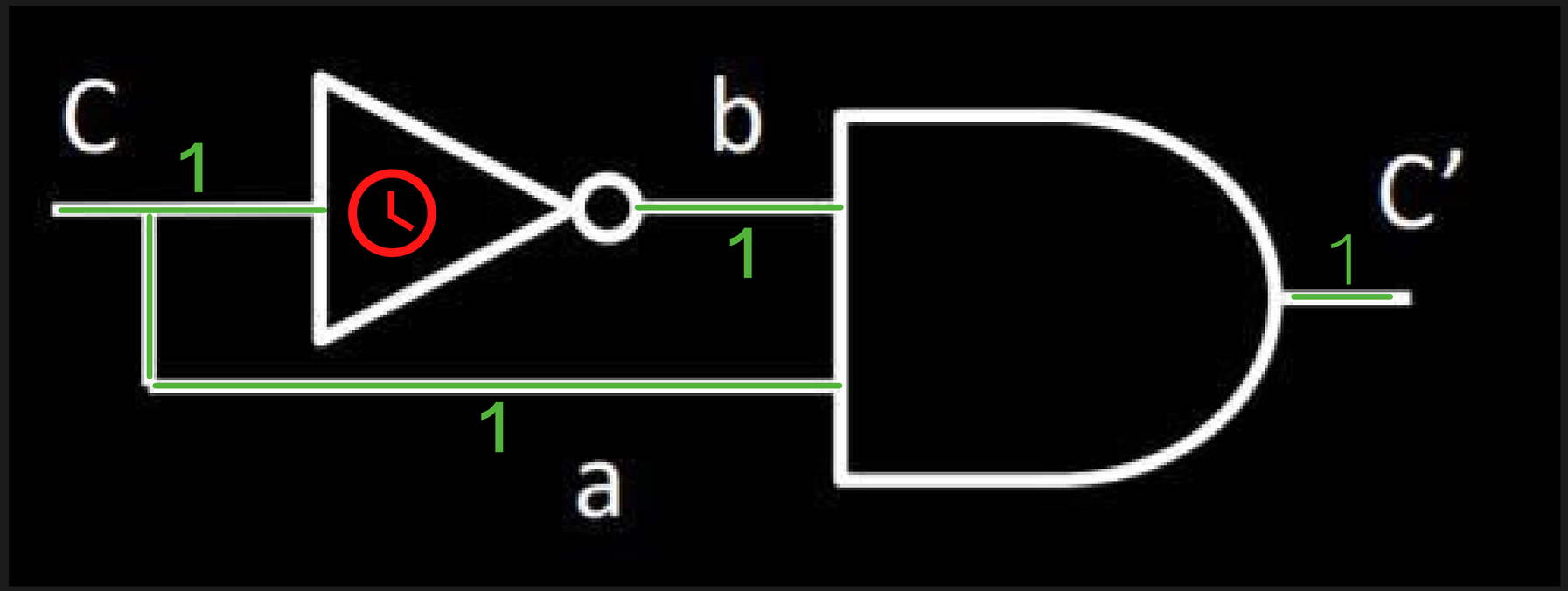
C = 1
b = 0
a = 1
C' = 0



C = 0
b = 1
a = 0
C' = 0

?

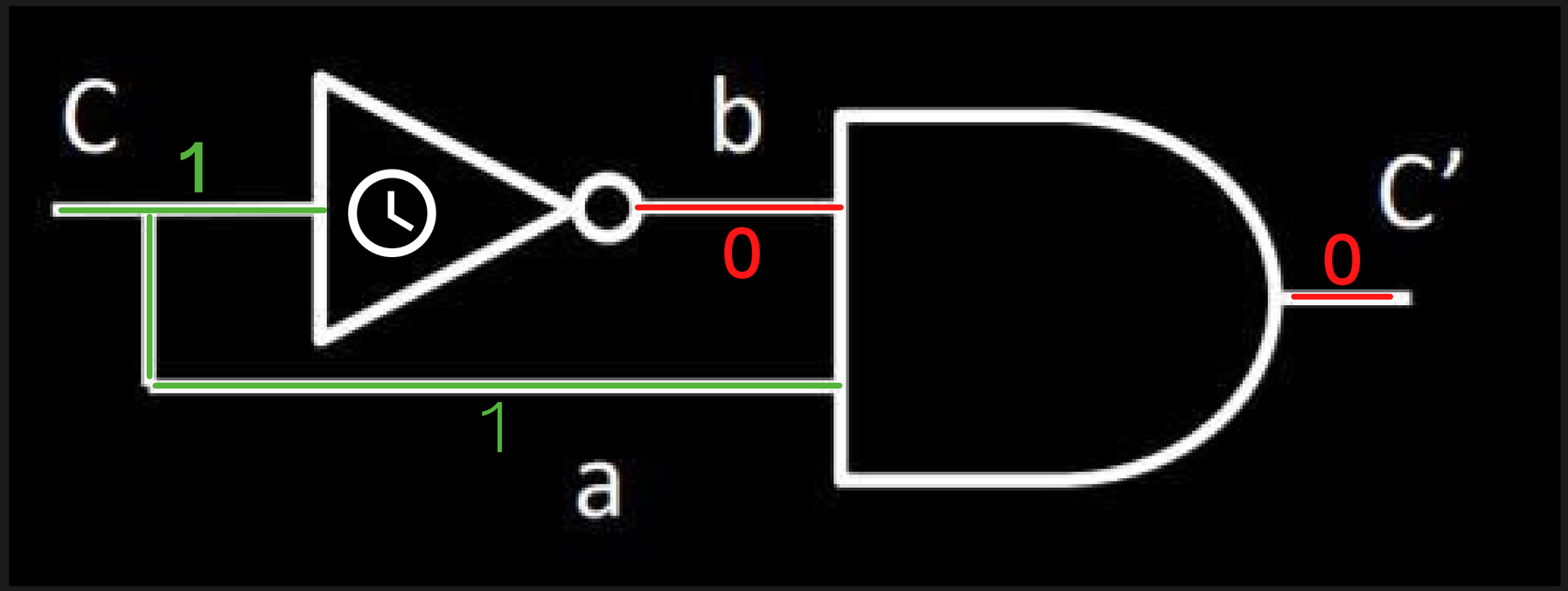
C = 1
b = 0
a = 1
C' = 0



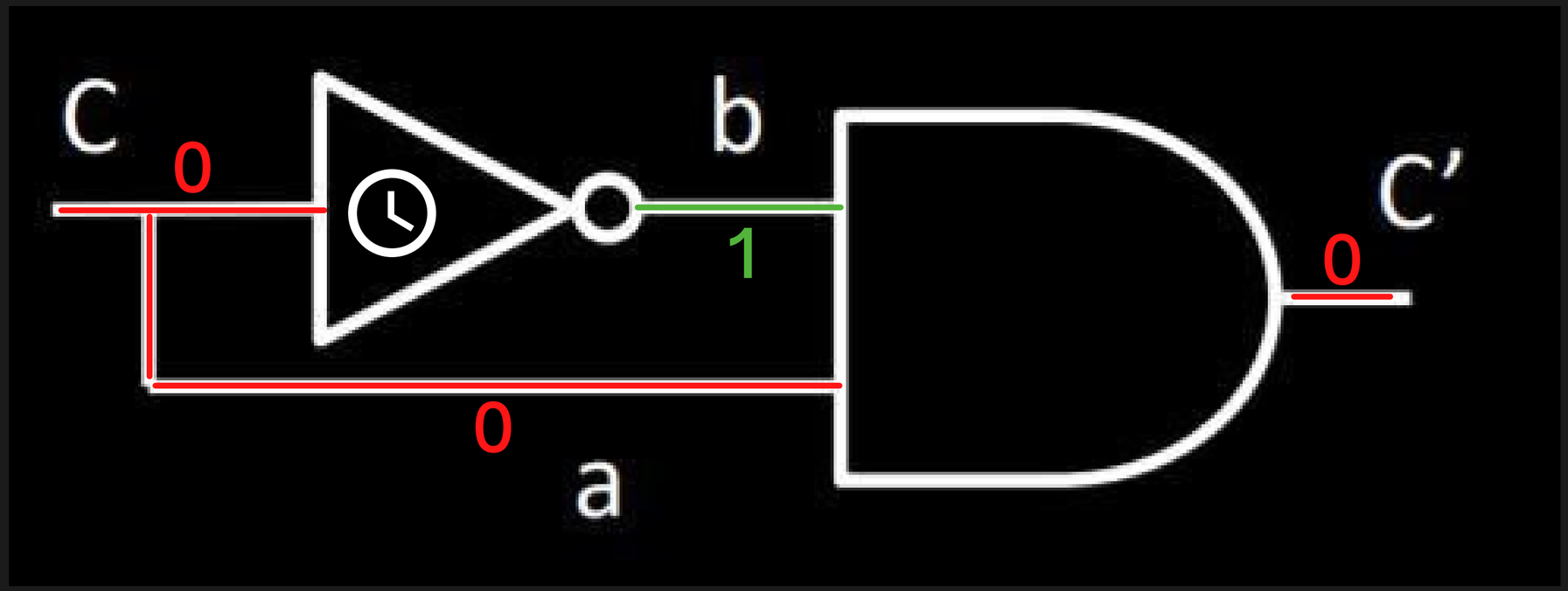
C = 0
b = 1
a = 0
C' = 0

?

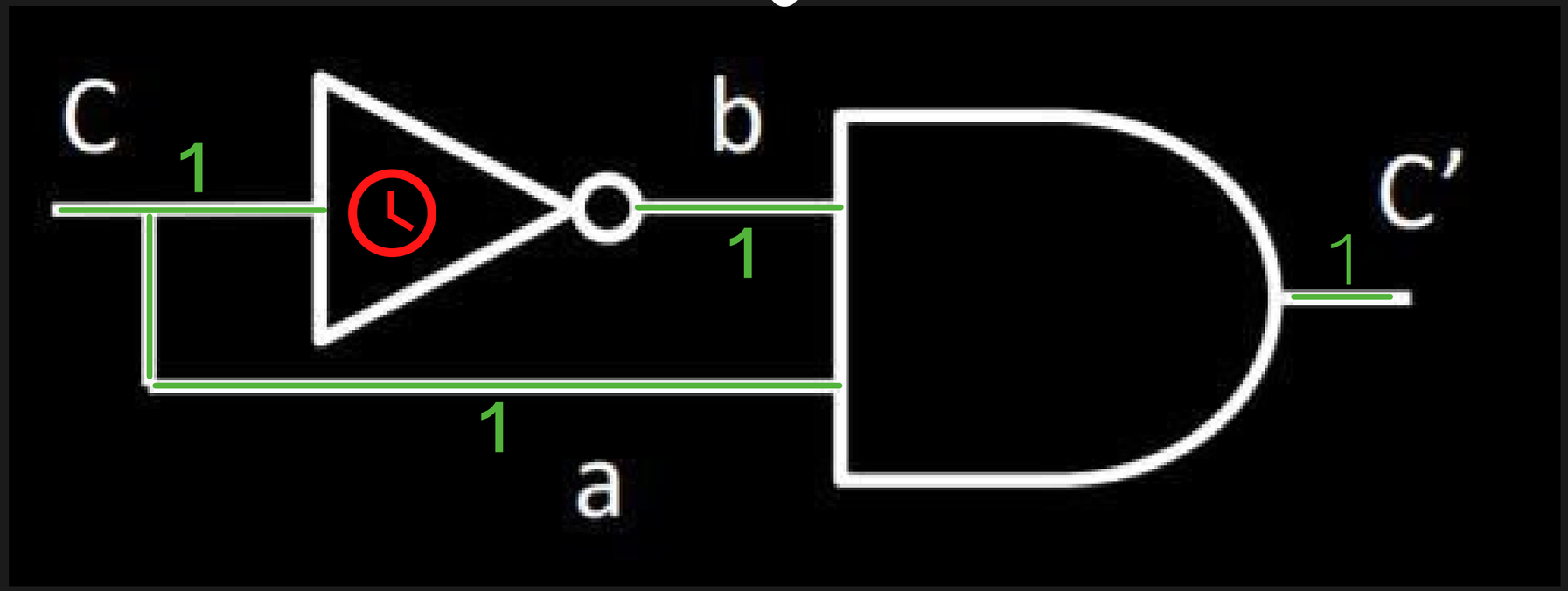
C = 1
b = 0
a = 1
C' = 0



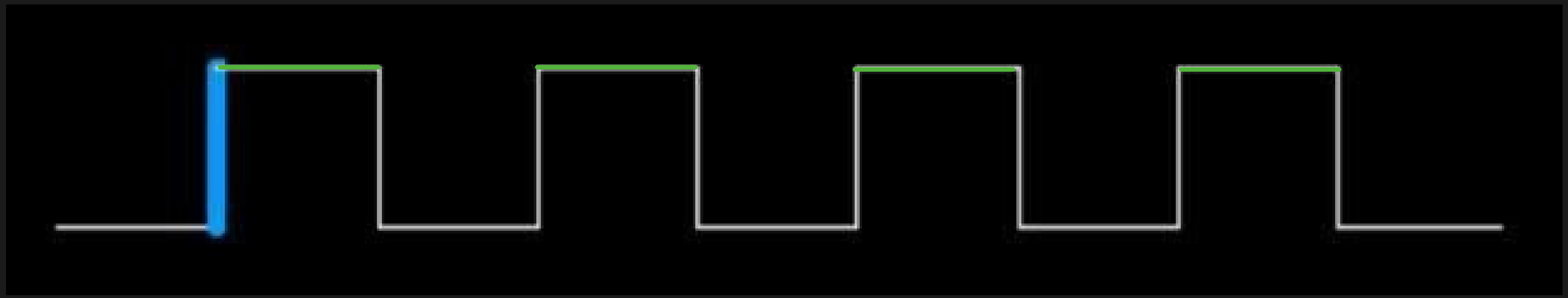
//05

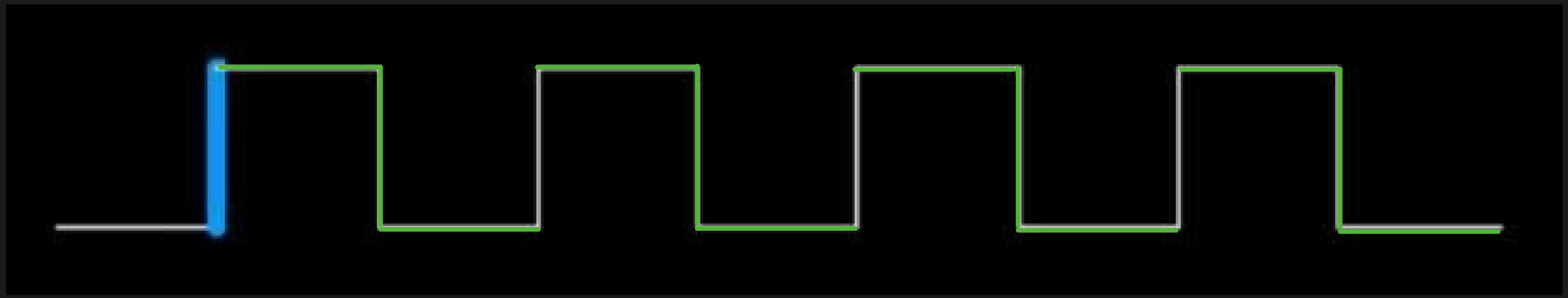


//05



//05





Base: 3.6 GHz

Turbo: 4.3 GHz

Ejercicios

1) Demuestra que en la representación *complemento de 2*, pasar de números de n bits a números de m bits, en que $m > n$, se puede hacer simplemente por la vía de agregar los $m - n$ nuevos bits a la izquierda de los n bits originales, y ponerlos todos en cero, en el caso de números positivos, o todos en uno, en el caso de números negativos: esta conversión mantiene el valor del número original.

Caso 1:

Tomemos como ejemplo el número positivo de n bits: 0000 ... abcd

1) Demuestra que en la representación *complemento de 2*, pasar de números de n bits a números de m bits, en que $m > n$, se puede hacer simplemente por la vía de agregar los $m - n$ nuevos bits a la izquierda de los n bits originales, y ponerlos todos en cero, en el caso de números positivos, o todos en uno, en el caso de números negativos: esta conversión mantiene el valor del número original.

Caso 1:

Tomemos como ejemplo el número positivo de n bits: 0000 ... abcd

Calculemos su valor numérico en base 10:

1) Demuestra que en la representación *complemento de 2*, pasar de números de n bits a números de m bits, en que $m > n$, se puede hacer simplemente por la vía de agregar los $m - n$ nuevos bits a la izquierda de los n bits originales, y ponerlos todos en cero, en el caso de números positivos, o todos en uno, en el caso de números negativos: esta conversión mantiene el valor del número original.

Caso 1:

Tomemos como ejemplo el número positivo de n bits: 0000 ... abcd

Calculemos su valor numérico en base 10:

0 = positivo

1) Demuestra que en la representación *complemento de 2*, pasar de números de n bits a números de m bits, en que $m > n$, se puede hacer simplemente por la vía de agregar los $m - n$ nuevos bits a la izquierda de los n bits originales, y ponerlos todos en cero, en el caso de números positivos, o todos en uno, en el caso de números negativos: esta conversión mantiene el valor del número original.

Caso 1:

Tomemos como ejemplo el número positivo de n bits: 0000 ... abcd

Calculemos su valor numérico en base 10:

0 = positivo

0*2^(n-1)+0*2^(n-2)+0*2^(n-3)+...+a*2^(3)+b*2^(2)+c*2^(1)+d*2^(0)

=0+8a+4b+2c+d

1) Demuestra que en la representación *complemento de 2*, pasar de números de n bits a números de m bits, en que $m > n$, se puede hacer simplemente por la vía de agregar los $m - n$ nuevos bits a la izquierda de los n bits originales, y ponerlos todos en cero, en el caso de números positivos, o todos en uno, en el caso de números negativos: esta conversión mantiene el valor del número original.

Caso 1:

Tomemos como ejemplo el número positivo de n bits: 0000 ... abcd

Calculemos su valor numérico en base 10:

0 = positivo

0*2^(n-1)+0*2^(n-2)+0*2^(n-3)+...+a*2^(3)+b*2^(2)+c*2^(1)+d*2^(0)

=0+8a+4b+2c+d

No importa cuantos 0s pongamos a la izquierda, se mantiene el valor.

1) Demuestra que en la representación *complemento de 2*, pasar de números de n bits a números de m bits, en que $m > n$, se puede hacer simplemente por la vía de agregar los $m - n$ nuevos bits a la izquierda de los n bits originales, y ponerlos todos en cero, en el caso de números positivos, o todos en uno, en el caso de números negativos: esta conversión mantiene el valor del número original.

Caso 2:

Tomemos como ejemplo el número negativo de n bits: 1111 ... abcd

1) Demuestra que en la representación *complemento de 2*, pasar de números de n bits a números de m bits, en que $m > n$, se puede hacer simplemente por la vía de agregar los $m - n$ nuevos bits a la izquierda de los n bits originales, y ponerlos todos en cero, en el caso de números positivos, o todos en uno, en el caso de números negativos: esta conversión mantiene el valor del número original.

Caso 2:

Tomemos como ejemplo el número negativo de n bits: 1111 ... abcd

Calculemos su valor numérico en base 10:

1 = negativo

1) invertimos bit a bit (a' = not(a)):

0000 ... a'b'c'd'

1) Demuestra que en la representación *complemento de 2*, pasar de números de n bits a números de m bits, en que $m > n$, se puede hacer simplemente por la vía de agregar los $m - n$ nuevos bits a la izquierda de los n bits originales, y ponerlos todos en cero, en el caso de números positivos, o todos en uno, en el caso de números negativos: esta conversión mantiene el valor del número original.

Caso 2:

Tomemos como ejemplo el número negativo de n bits: 1111 ... abcd

Calculemos su valor numérico en base 10:

1 = negativo

1) invertimos bit a bit (a' = not(a)):

0000 ... a'b'c'd'

2) sumamos 1:

0000 ... a'b'c'(d'+1)

1) Demuestra que en la representación *complemento de 2*, pasar de números de n bits a números de m bits, en que $m > n$, se puede hacer simplemente por la vía de agregar los $m - n$ nuevos bits a la izquierda de los n bits originales, y ponerlos todos en cero, en el caso de números positivos, o todos en uno, en el caso de números negativos: esta conversión mantiene el valor del número original.

Caso 2:

Tomemos como ejemplo el número negativo de n bits: 1111 ... abcd

Calculemos su valor numérico en base 10:

3) en base 10:

0000 ... a'b'c'(d'+1) = 0*2^(n-1)+0*2^(n-2)+0*2^(n-3)+...

+a'*2^(3)+b'*2^(2)+c'*2^(1)+d'*2^(0) + 1*2^0 = 8a'+4b'+2c'+d'+1

1) Demuestra que en la representación *complemento de 2*, pasar de números de n bits a números de m bits, en que $m > n$, se puede hacer simplemente por la vía de agregar los $m - n$ nuevos bits a la izquierda de los n bits originales, y ponerlos todos en cero, en el caso de números positivos, o todos en uno, en el caso de números negativos: esta conversión mantiene el valor del número original.

Caso 2:

Tomemos como ejemplo el número negativo de n bits: 1111 ... abcd

Calculemos su valor numérico en base 10:

3) en base 10:

0000 ... a'b'c'(d'+1) = 0*2^(n-1)+0*2^(n-2)+0*2^(n-3)+...

+a'*2^(3)+b'*2^(2)+c'*2^(1)+d'*2^(0) + 1*2^0 = 8a'+4b'+2c'+d'+1

4) agregamos el negativo: = -(8a'+4b'+2c'+d'+1)

1) Demuestra que en la representación *complemento de 2*, pasar de números de n bits a números de m bits, en que $m > n$, se puede hacer simplemente por la vía de agregar los $m - n$ nuevos bits a la izquierda de los n bits originales, y ponerlos todos en cero, en el caso de números positivos, o todos en uno, en el caso de números negativos: esta conversión mantiene el valor del número original.

Caso 2:

Tomemos como ejemplo el número negativo de n bits: 1111 ... abcd

Calculemos su valor numérico en base 10:

3) en base 10:

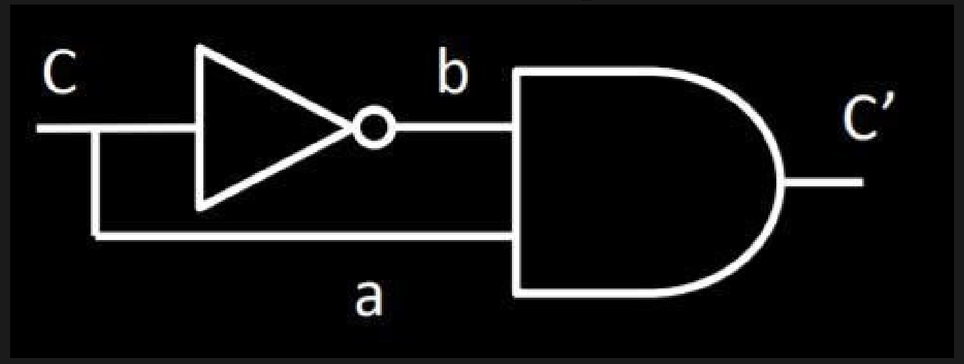
0000 ... a'b'c'(d'+1) = 0*2^(n-1)+0*2^(n-2)+0*2^(n-3)+...

+a'*2^(3)+b'*2^(2)+c'*2^(1)+d'*2^(0) + 1*2^0 = 8a'+4b'+2c'+d'+1

4) agregamos el negativo: = -(8a'+4b'+2c'+d'+1)

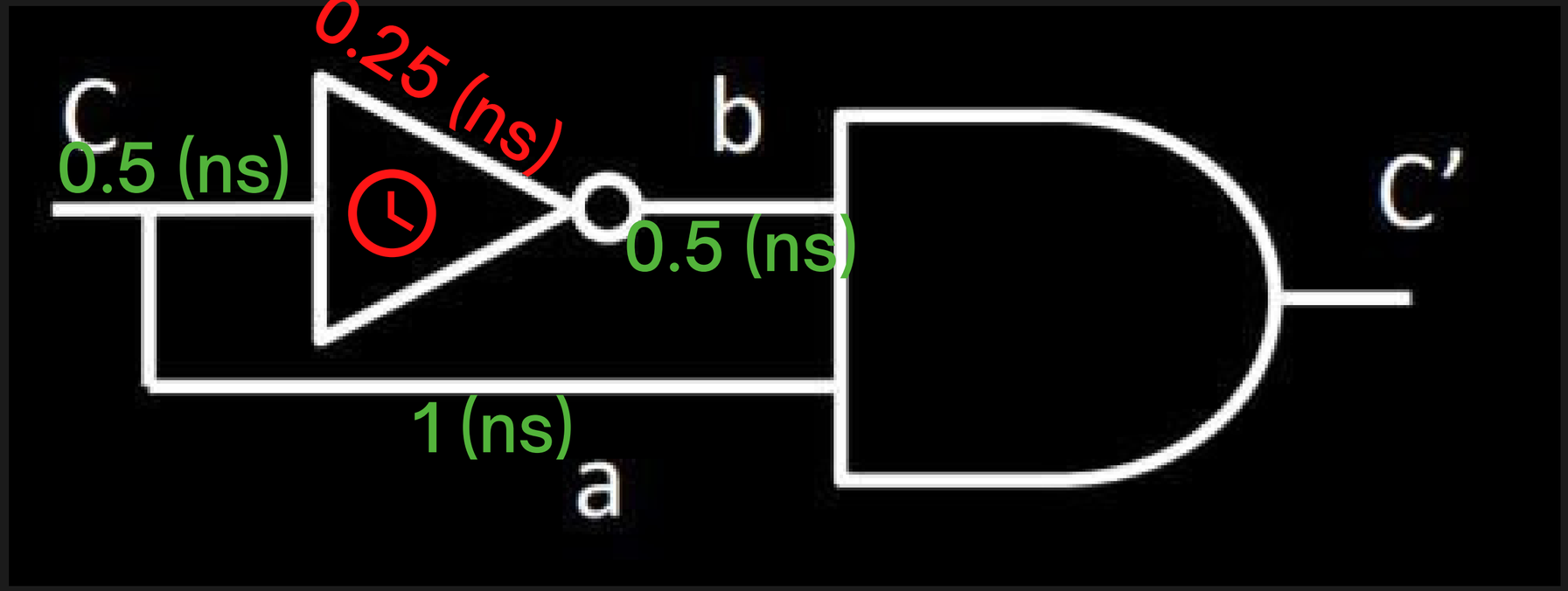
No importa cuantos 0s pongamos a la izquierda, se mantiene el valor.

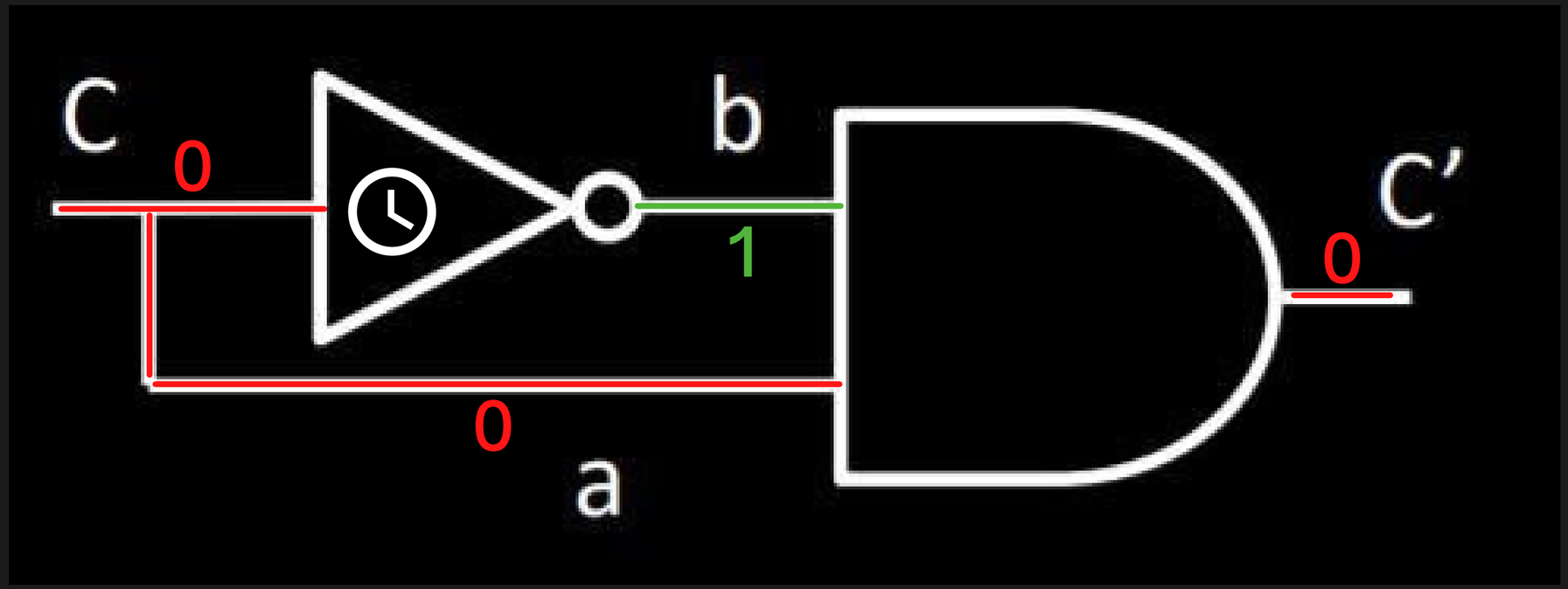
2) Don Ignacio se entera de tus conocimientos en arquitectura de computadores y te pone a cargo del generador de pulsos del nuevo procesador desarrollado por la universidad, el "PUChip".
El equipo de desarrollo diseñó el siguiente generador de pulsos y se describieron los siguientes datos:



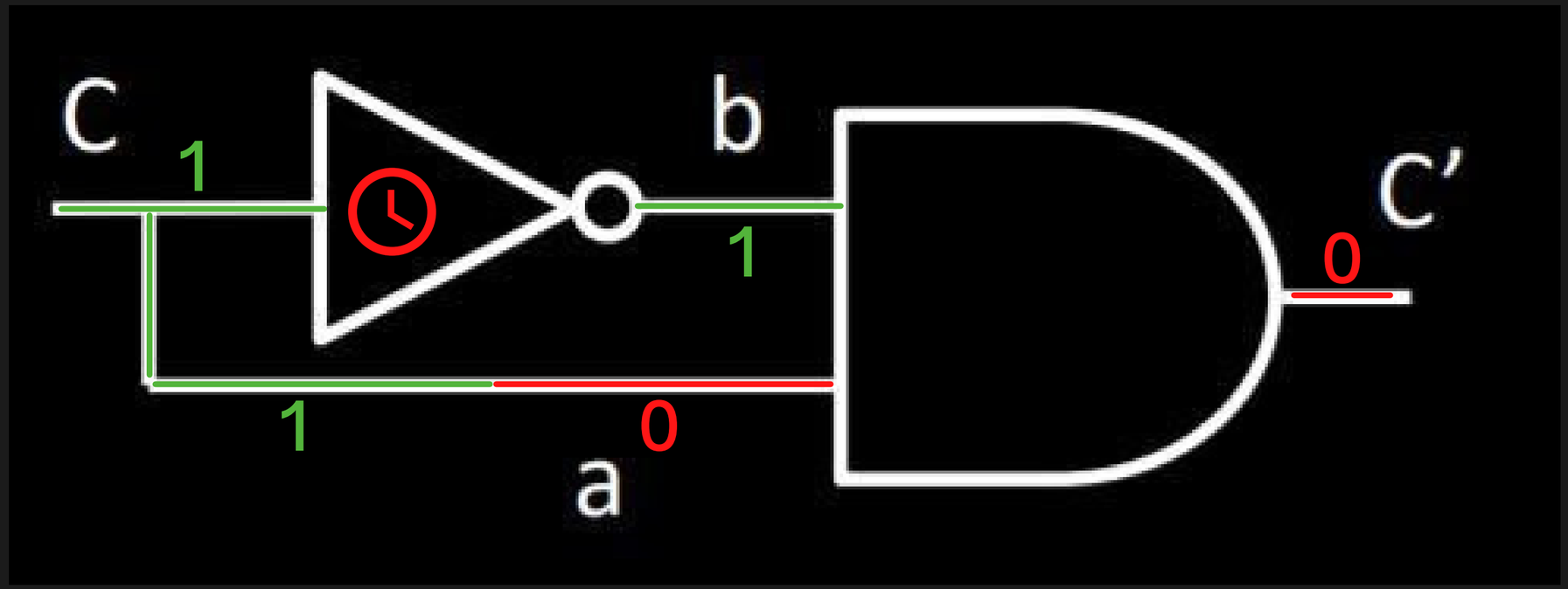
- Tiempo desde C hasta inicio del NOT: 0.5(ns)
- Tiempo de ejecucuión del NOT: 0.25 (ns)
- Tiempo en recorrer el cable b: 0.5 (ns)
- Tiempo desde C hasta el final de a: 1 (ns)

¿Cual es la frecuencia máxima teórica del "PUChip"?

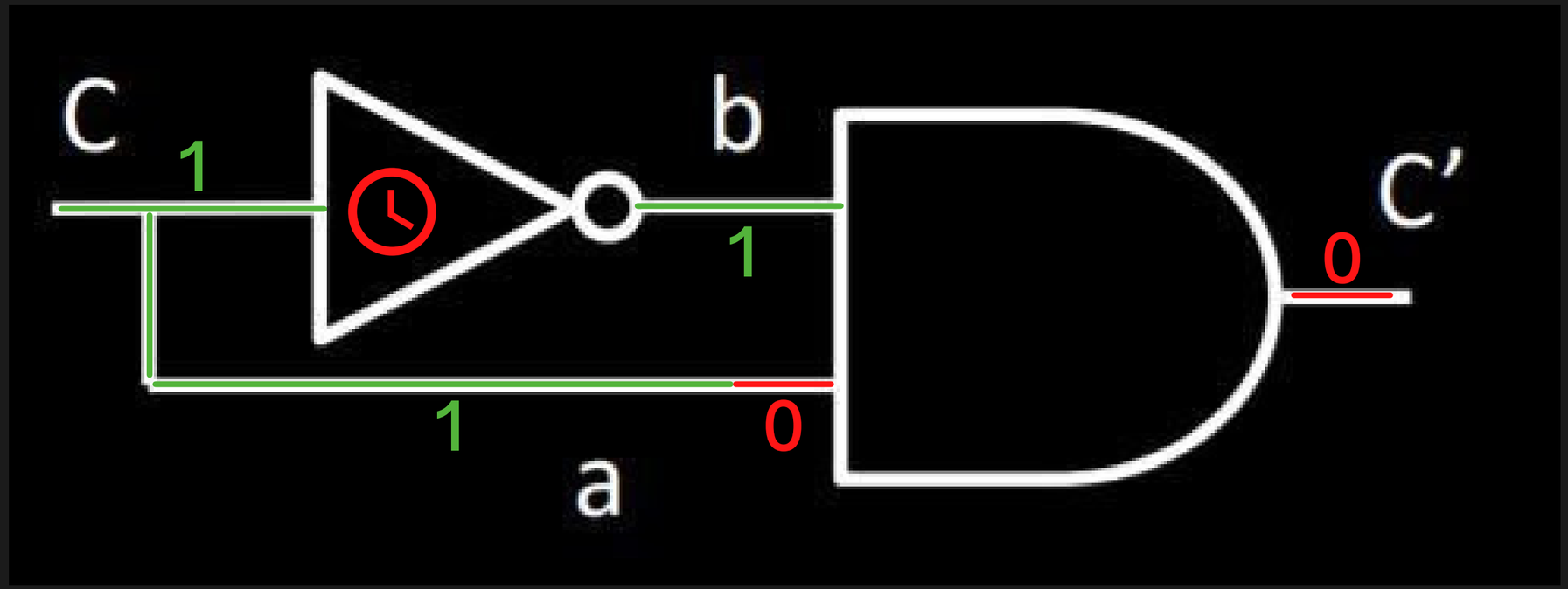

0.25 (ns)
c
0.5 (ns)
b
0.5 (ns)
c'
1 (ns)
a



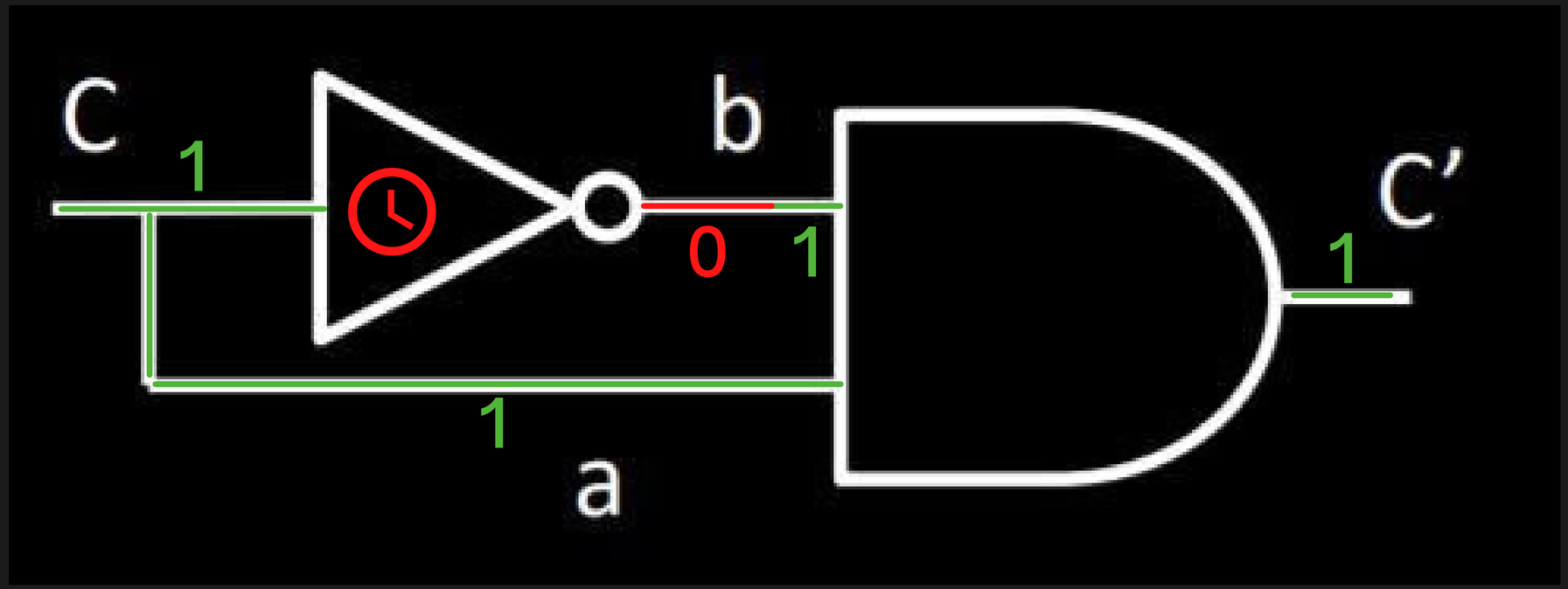
t = 0 (ns)



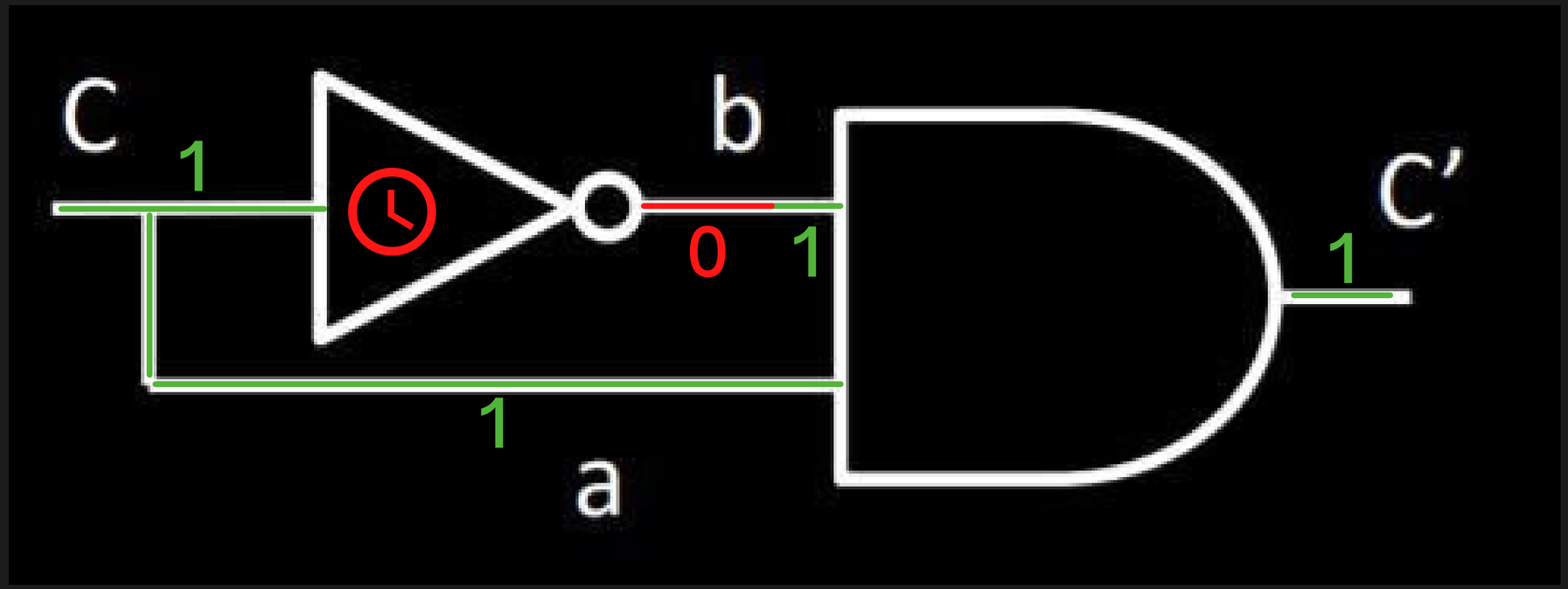
t = 0.5 (ns)



0.5 (ns) < t < 0.75 (ns)



1 (ns) < t < 1.25 (ns)



1 (ns) < t < 1.25 (ns)

El pulso dura 0.25 (ns)

P = 0.25 (ns)

ns → Hz

ns → Hz

1 (ns) → 1 * 10^-9 (s)

ns → Hz

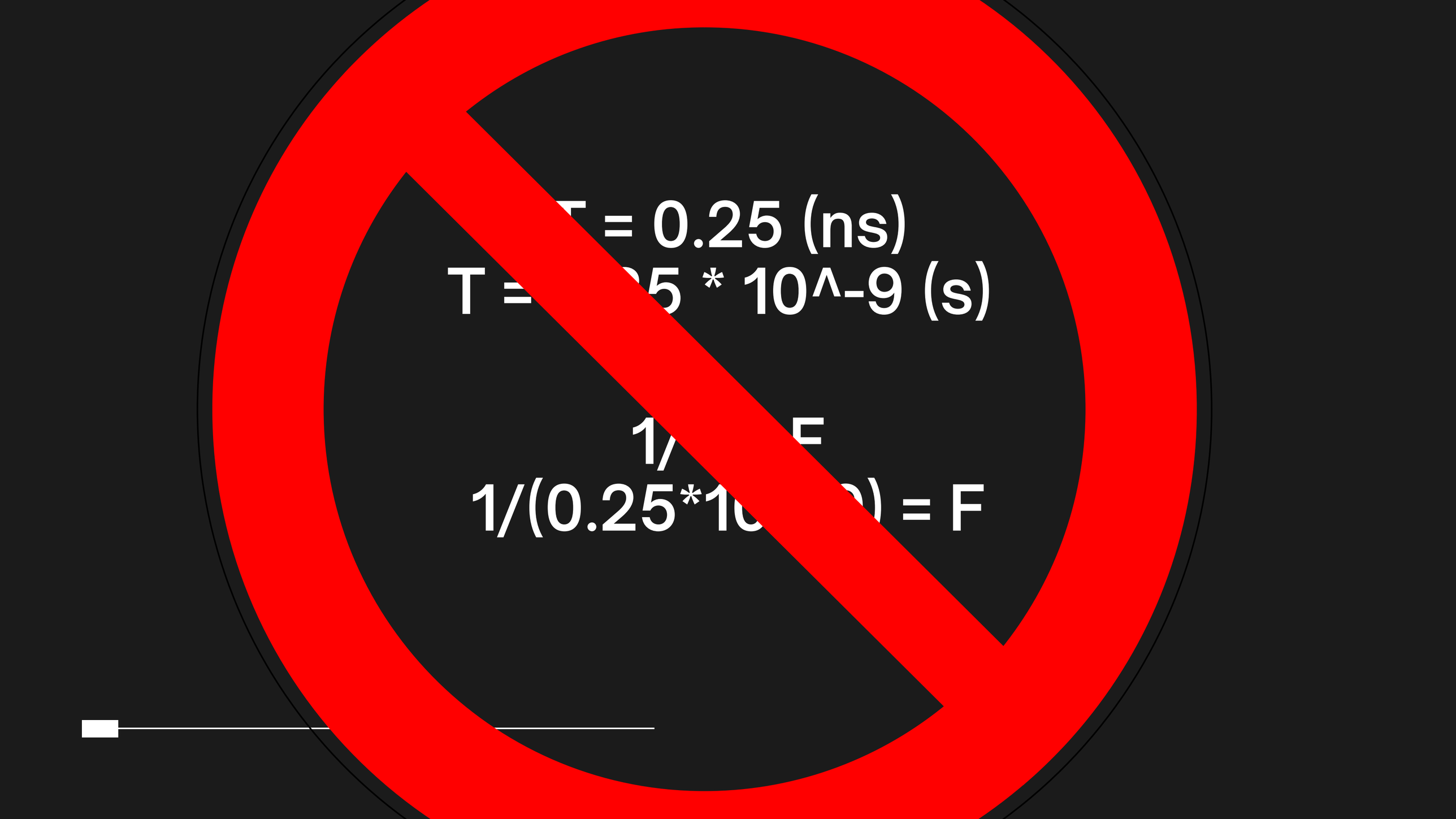
1 (ns) → 1 * 10^-9 (s)

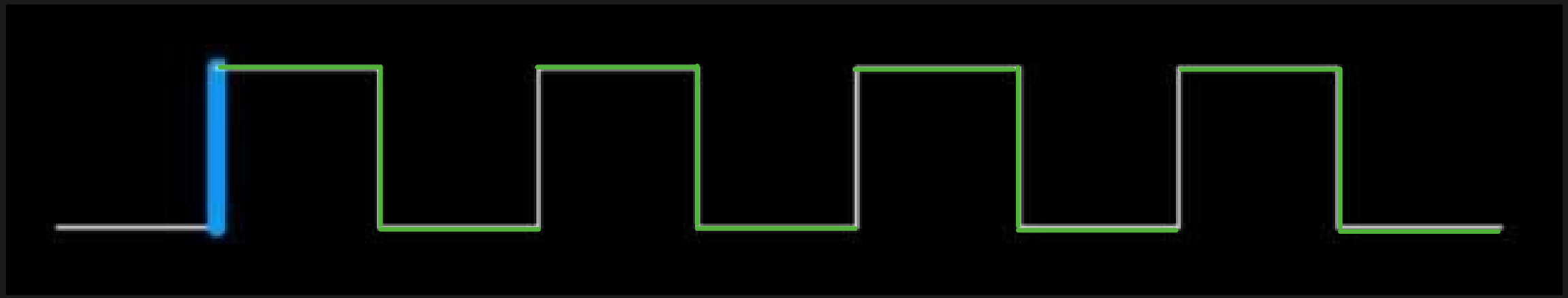
1/T → F

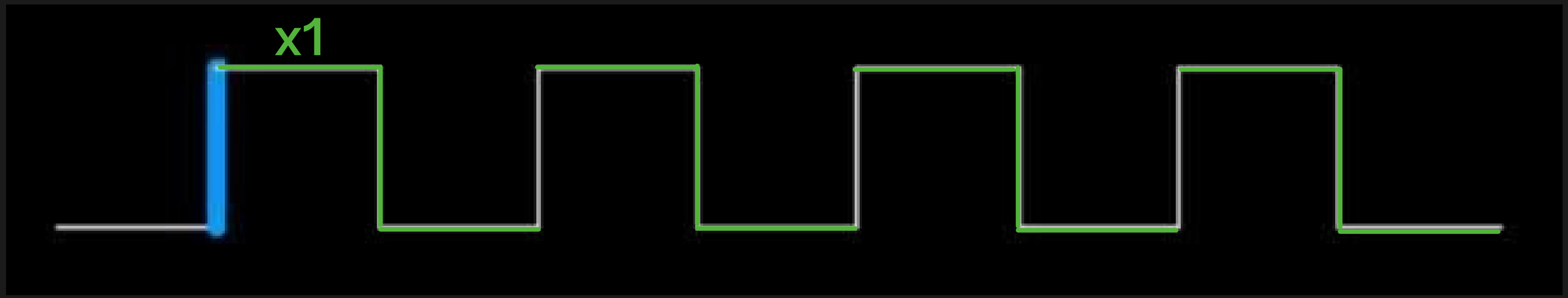
$$T = 0.25 \text{ (ns)}$$
$$T = 0.25 * 10^{-9} \text{ (s)}$$

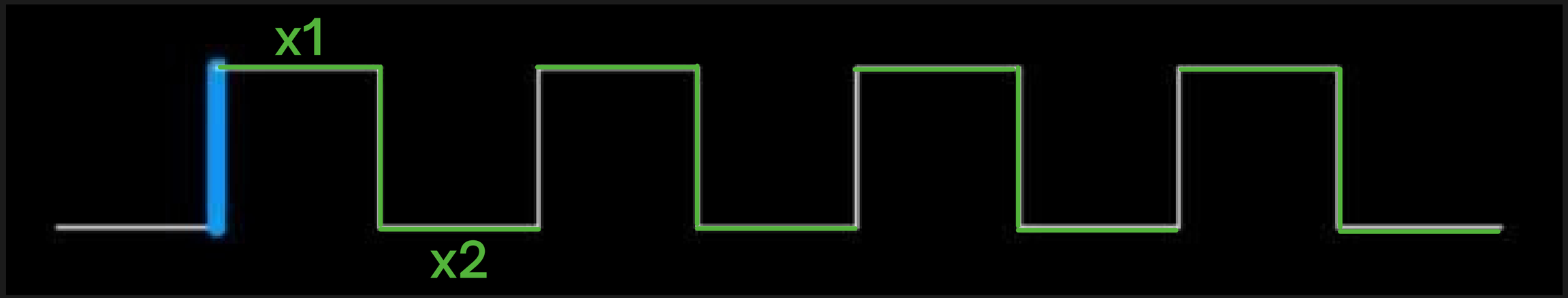
$$T = 0.25 \text{ (ns)}$$
$$T = 0.25 * 10^{-9} \text{ (s)}$$

$$1/T = F$$
$$1/(0.25*10^{-9}) = F$$


T = 0.25 (ns)
T = 25 * 10^-9 (s)
1/ F
1/(0.25*10) = F




x1



//05

P = 0.25 (ns)
T = 2*P = 0.5 (ns)
T = 0.5 * 10^-9 (s)

P = 0.25 (ns)
T = 2*P = 0.5 (ns)
T = 0.5 * 10^-9 (s)

1/T = F
1/(0.05*10^-9) = F

P = 0.25 (ns)
T = 2*P = 0.5 (ns)
T = 0.5 * 10^-9 (s)

1/T = F
1/(0.05*10^-9) = F

F = 2 (GHz)

3) ¿En qué casos es posible soportar la instrucción ADD B, Lit en el computador básico, sin modificar su hardware? Para los casos negativos, indique modificaciones al hardware y/o assembly que se deberían hacer para soportarlos. [I1-2016-1]

3) ¿En qué casos es posible soportar la instrucción ADD B, Lit en el computador básico, sin modificar su hardware? Para los casos negativos, indique modificaciones al hardware y/o assembly que se deberían hacer para soportarlos. [I1-2016-1]

Instrucción	Operandos	Opcode	La	Lb	Sa0	Sb0	Sb1	Sop2	Sop1	Sop0	Operación
MOV	A,B	000000	1	0	1	0	0	0	0	0	A=B
	B,A	000001	0	1	0	1	1	0	0	0	B=A
	A,Lit	000010	1	0	0	0	1	0	0	0	A=Lit
	B,Lit	000011	0	1	0	0	1	0	0	0	B=Lit
ADD	A,B	000100	1	0	0	0	0	0	0	0	A=A+B
	B,A	000101	0	1	0	0	0	0	0	0	B=A+B
	A,Lit	000110	1	0	0	0	1	0	0	0	A=A+Lit
SUB	A,B	000111	1	0	0	0	0	0	0	1	A=A-B
	B,A	001000	0	1	0	0	0	0	0	1	B=A-B
	A,Lit	001001	1	0	0	0	1	0	0	1	A=A-Lit
AND	A,B	001010	1	0	0	0	0	0	1	0	A=A and B
	B,A	001011	0	1	0	0	0	0	1	0	B=A and B
	A,Lit	001100	1	0	0	0	1	0	1	0	A=A and Lit
OR	A,B	001101	1	0	0	0	0	0	1	1	A=A or B
	B,A	001110	0	1	0	0	0	0	1	1	B=A or B
	A,Lit	001111	1	0	0	0	1	0	1	1	A=A or Lit
NOT	A,A	010000	1	0	0	0	0	1	0	0	A=notA
	B,A	010001	0	1	0	0	0	1	0	0	B=notA
	A,Lit	010010	1	0	0	0	1	1	0	0	A=notLit
XOR	A,A	010011	1	0	0	0	0	1	0	1	A=A xor B
	B,A	010100	0	1	0	0	0	1	0	1	B=A xor B
	A,Lit	010101	1	0	0	0	1	1	0	1	A=A xor Lit
SHL	A,A	010110	1	0	0	0	0	1	1	0	A=shift left A
	B,A	010111	0	1	0	0	0	1	1	0	B=shift left A
	A,Lit	011000	1	0	0	0	1	1	1	0	A=shift left Lit
SHR	A,A	011001	1	0	0	0	0	1	1	1	A=shift right A
	B,A	011010	0	1	0	0	0	1	1	1	B=shift right A
	A,Lit	011011	1	0	0	0	1	1	1	1	A=shift right Lit

Diagrama del computador basico

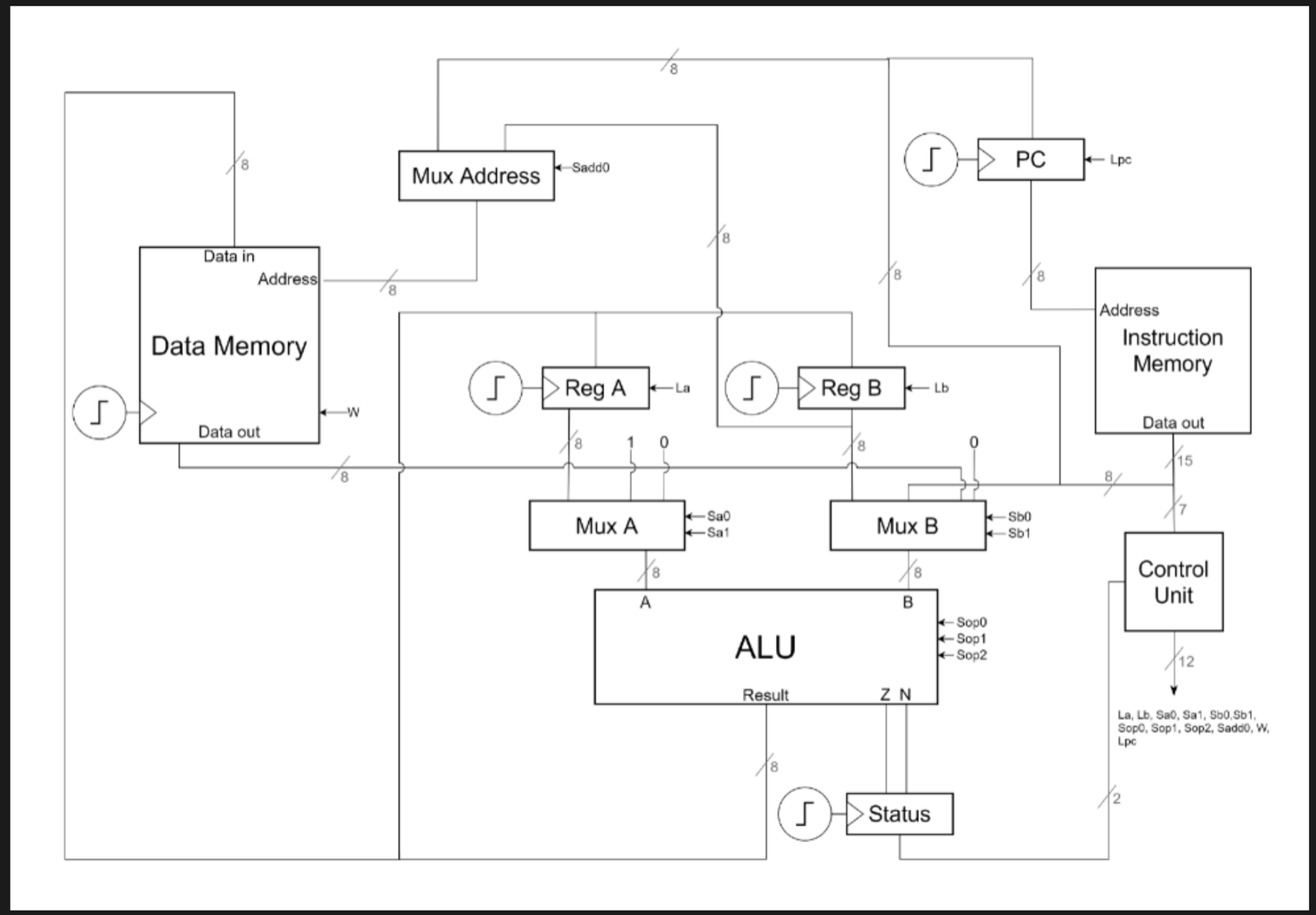
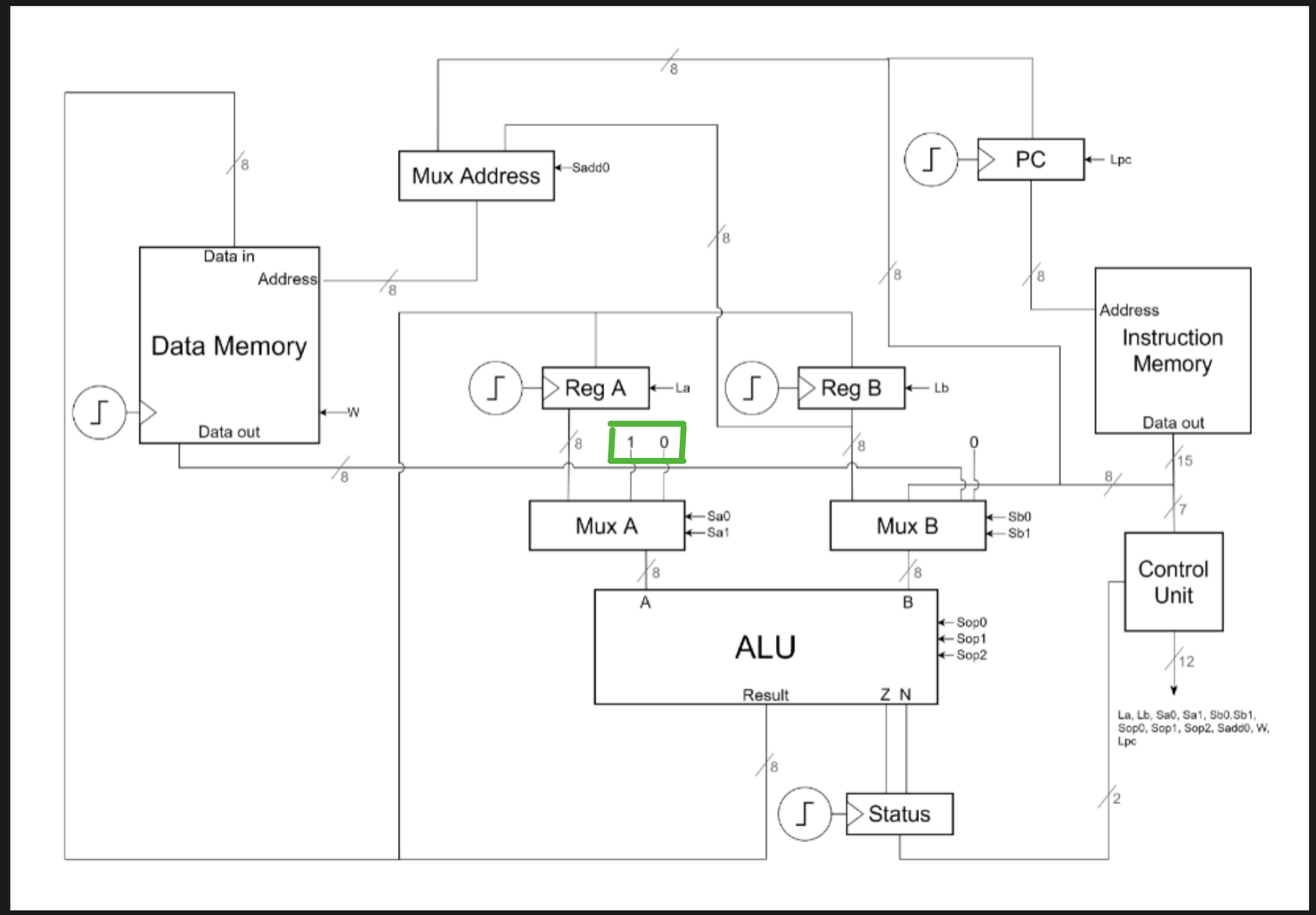
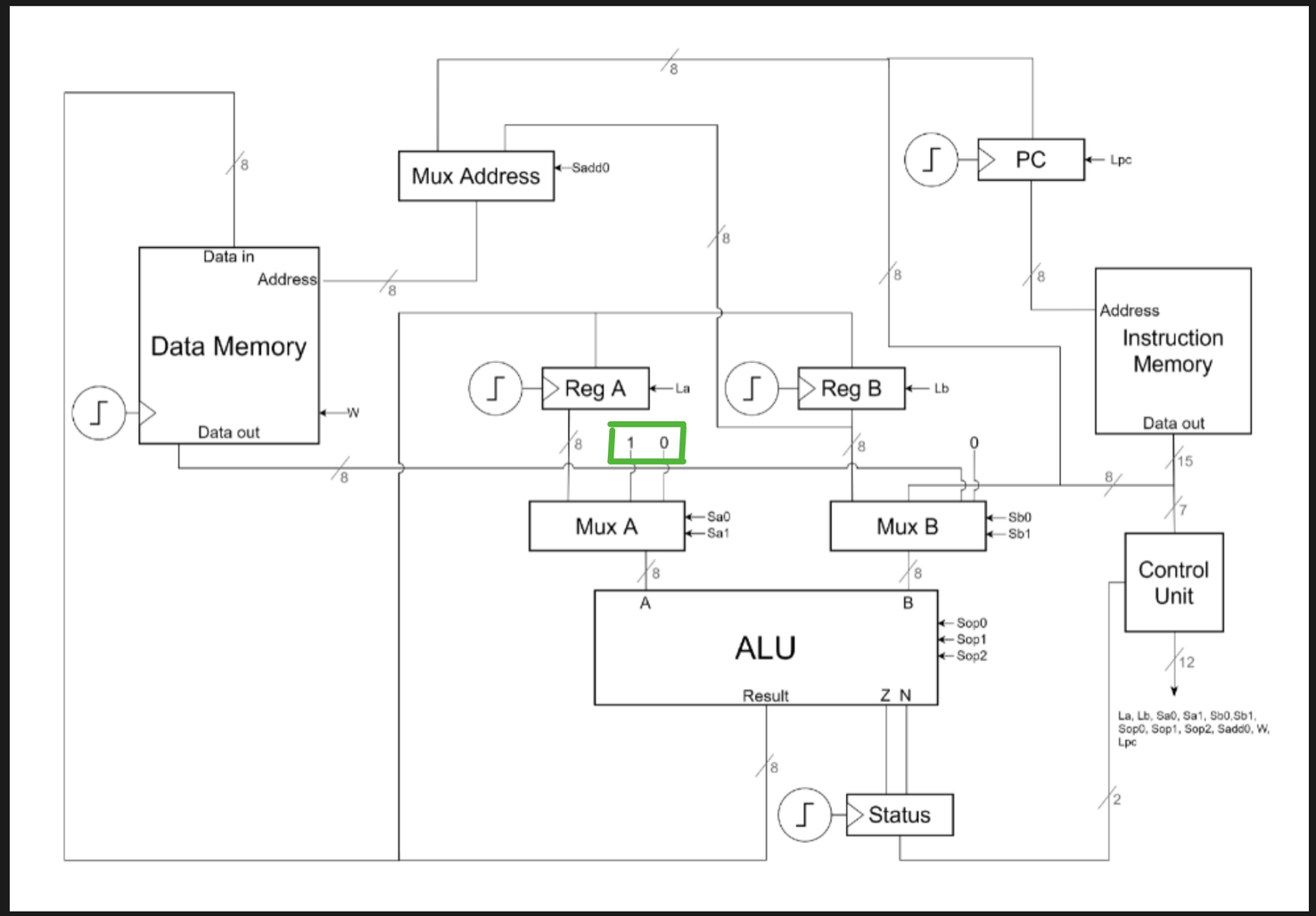


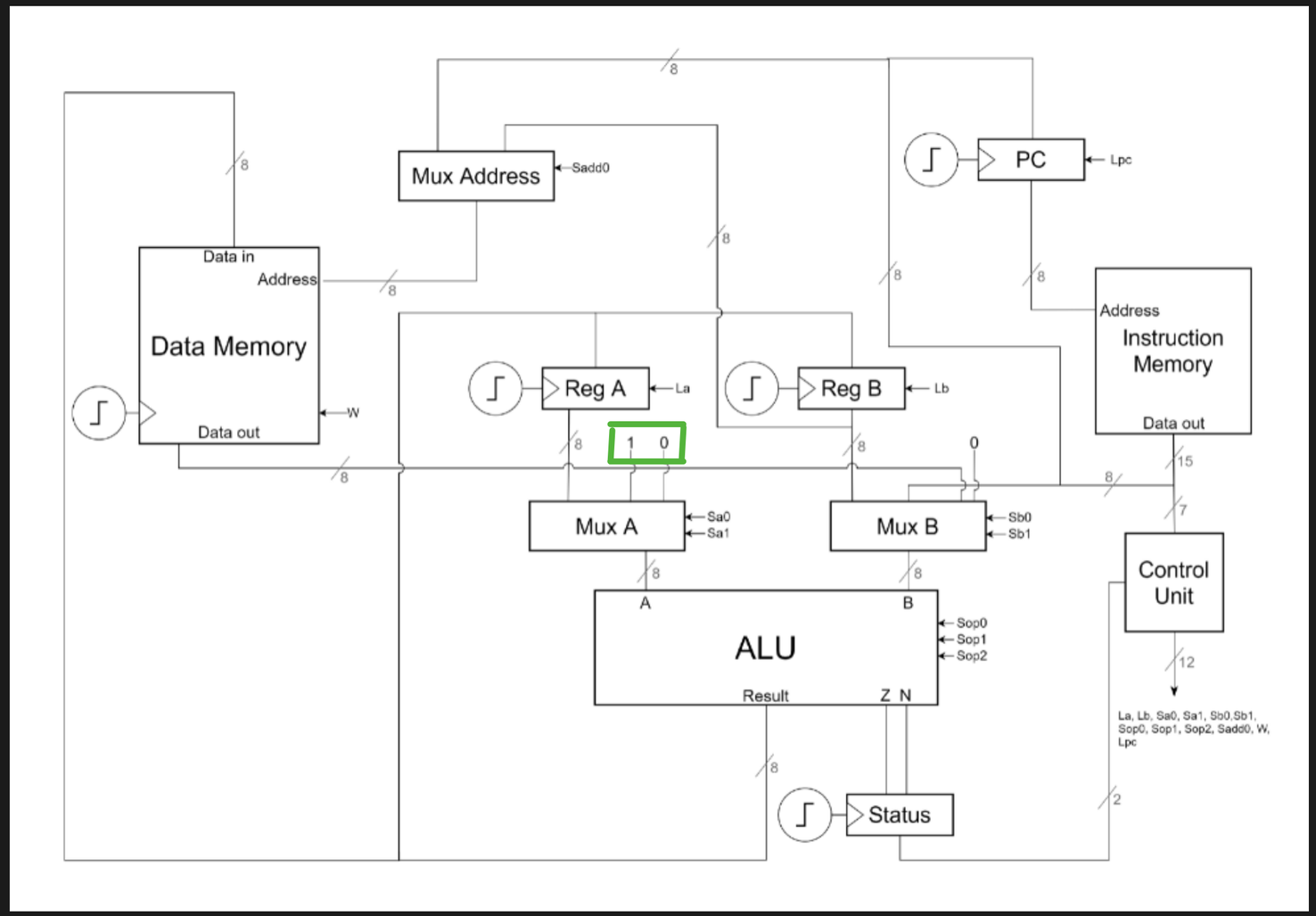
Diagrama del computador basico





Las instrucciones soportadas <u>sin</u> cambiar el hardware son:

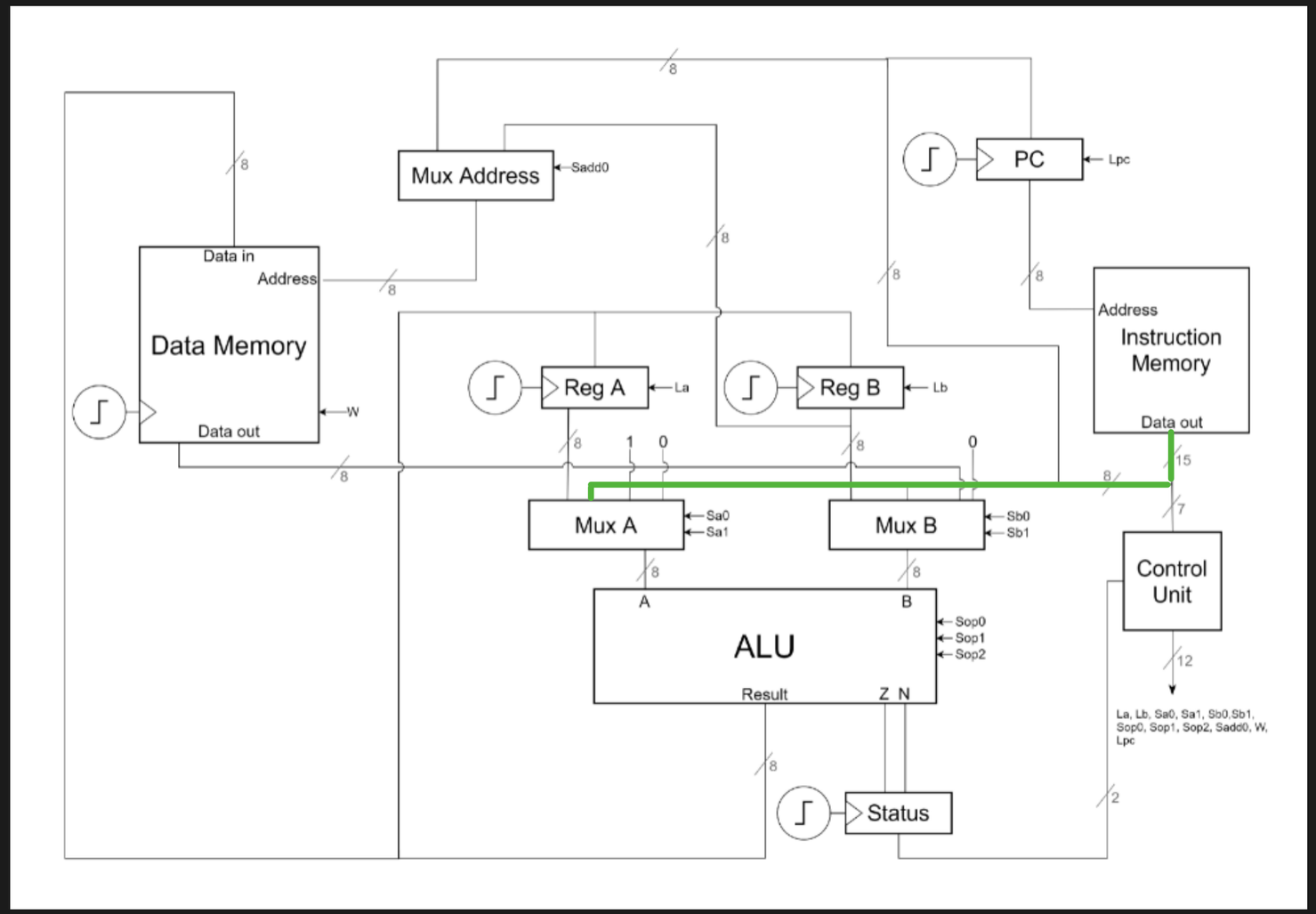
- ADD B, 0
- ADD B, 1

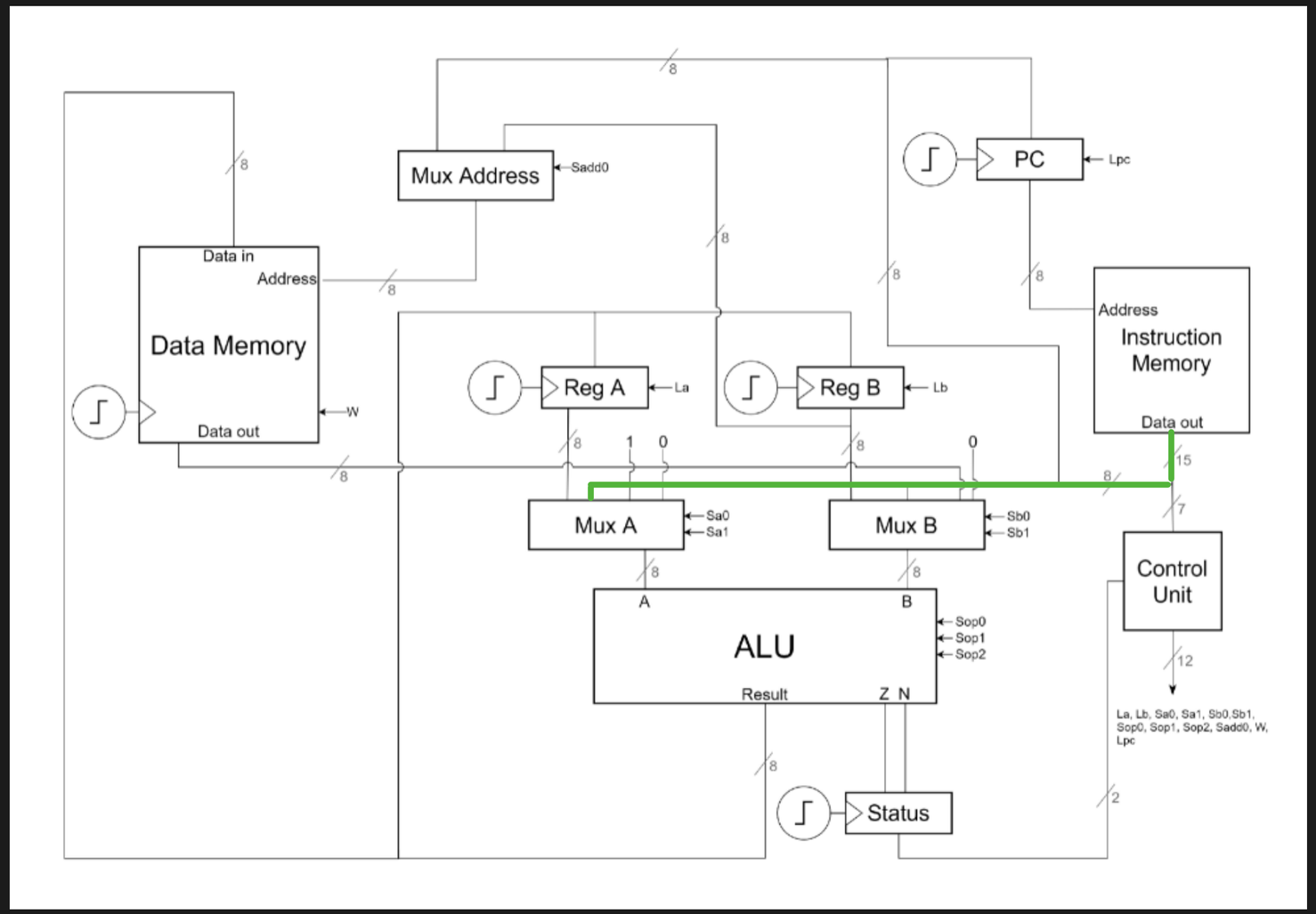


Las instrucciones soportadas <u>sin</u> cambiar el hardware son:

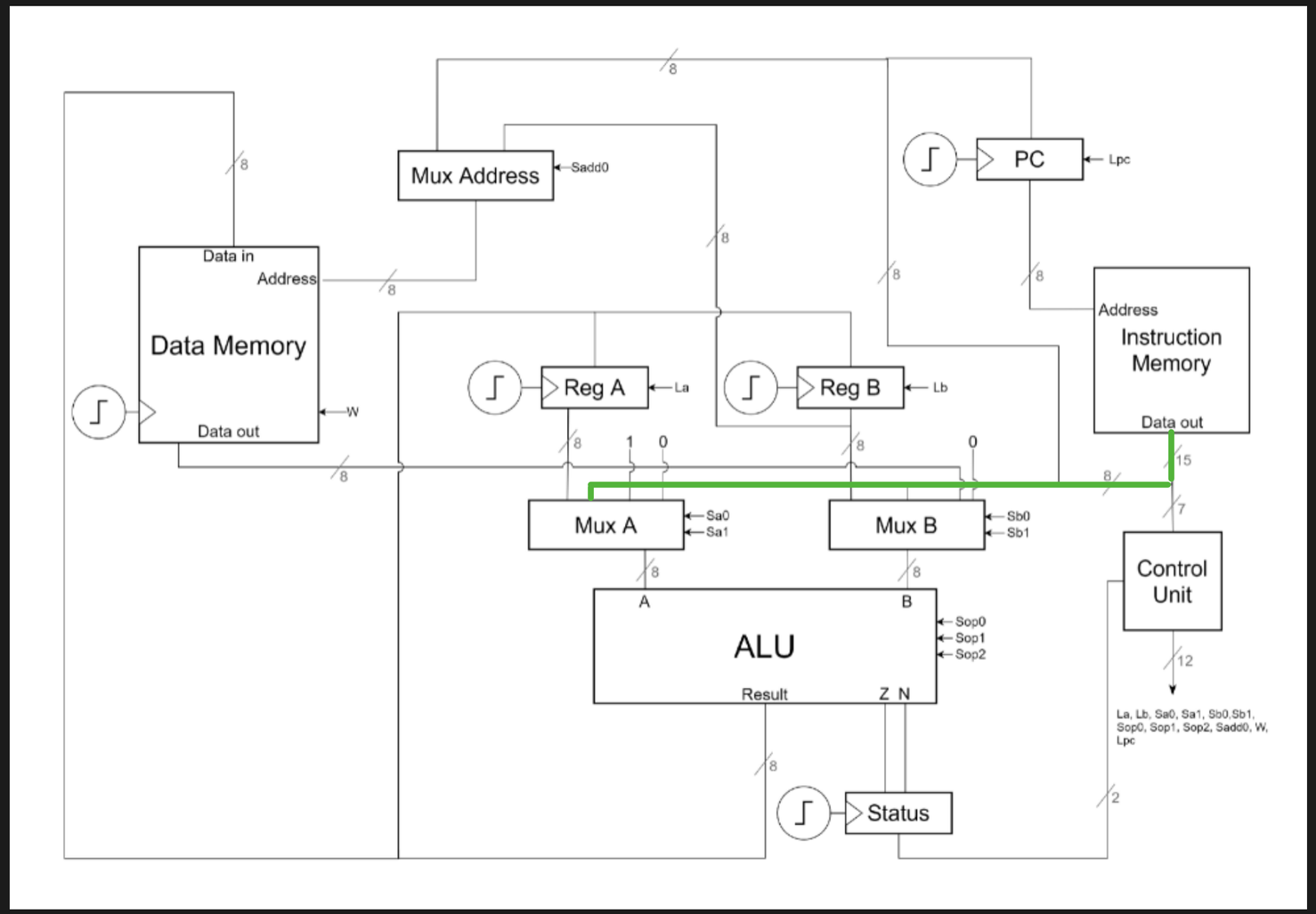
- ADD B, 0
- ADD B, 1

Como podemos modificar el hardware para que ahora se pueda realizar cualquier operacion tipo ADD B, Lit?


Mux Address
Sadd0
PC
Lpc
Data in
Address
Data Memory
Data out
W
Reg A
La
Reg B
Lb
Address
Instruction Memory
Data out
Mux A
Sa0
Sa1
Mux B
Sb0
Sb1
Control Unit
A
B
ALU
Sop0
Sop1
Sop2
Result
Z N
Status
La, Lb, Sa0, Sa1, Sb0,Sb1, Sop0, Sop1, Sop2, Sadd0, W, Lpc



Se debe agregar el bus de literales al Mux A



Se debe agregar el bus de literales al Mux A

Y que pasa con los opcodes?

Instrucción	Operandos	Opcode	La	Lb	Sa0	Sb0	Sb1	Sop2	Sop1	Sop0	Operación
MOV	A,B	000000	1	0	1	0	0	0	0	0	A=B
	B,A	000001	0	1	0	1	1	0	0	0	B=A
	A,Lit	000010	1	0	0	0	1	0	0	0	A=Lit
	B,Lit	000011	0	1	0	0	1	0	0	0	B=Lit
ADD	A,B	000100	1	0	0	0	0	0	0	0	A=A+B
	B,A	000101	0	1	0	0	0	0	0	0	B=A+B
	A,Lit	000110	1	0	0	0	1	0	0	0	A=A+Lit
	B, Lit	000111	0	1	0	0	0	0	0	0	B=B+Lit