Ayudantía 3

Arquitectura de Computadores – IIC2343

Rodrigo Alonso
rialonso@uc.cl

Contenidos

Lógica digital

- **ALU**: Operaciones Shift Right / Shift Left.
- **Almacenamiento**: Latch SR, Latch D, Flip Flop, Registro.

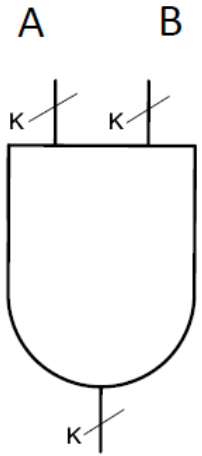
Procesador

- Componentes computador básico.
- Assembly.

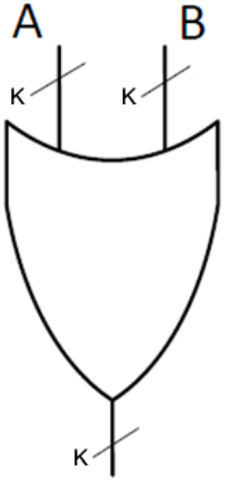
Ejercicios

Lógica Digital: ALU

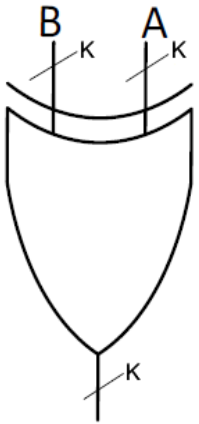
ALU: Compuertas Lógicas



A and B



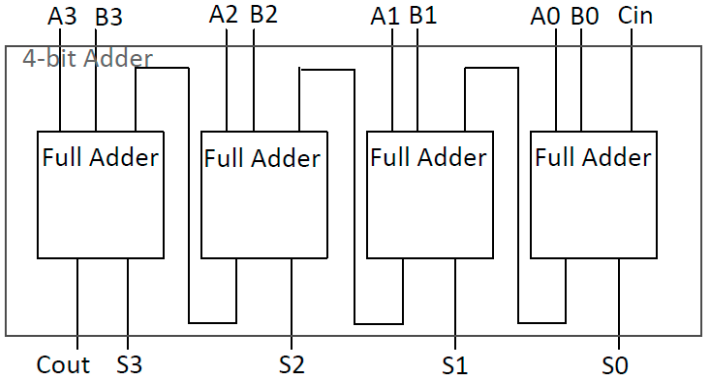
A or B

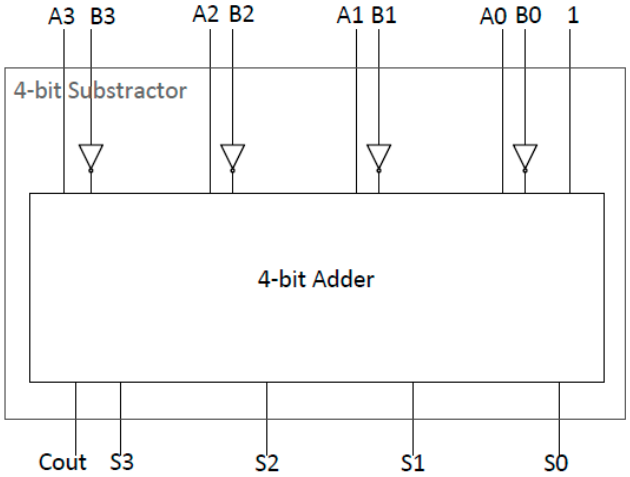


A xor B

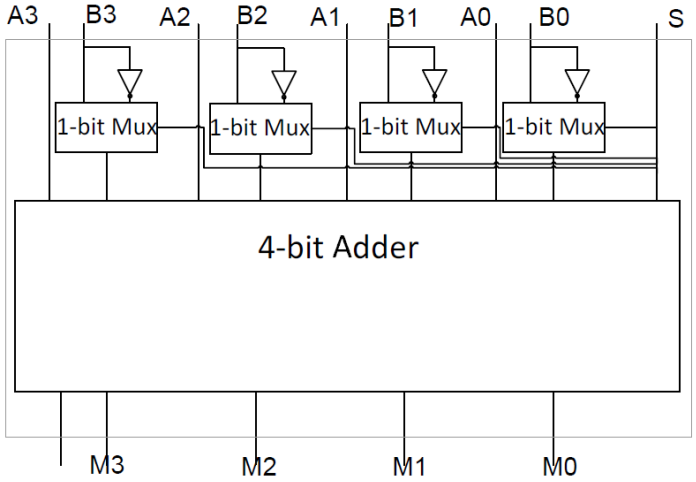
not(A)

ALU: Sumador/Restador



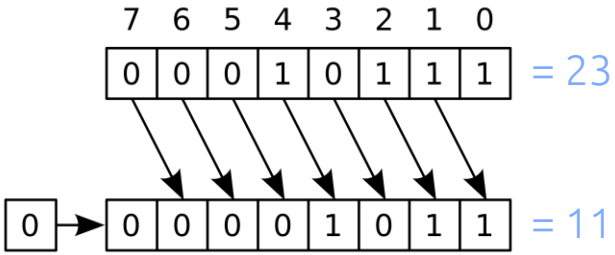


ALU: Sumador/Restador



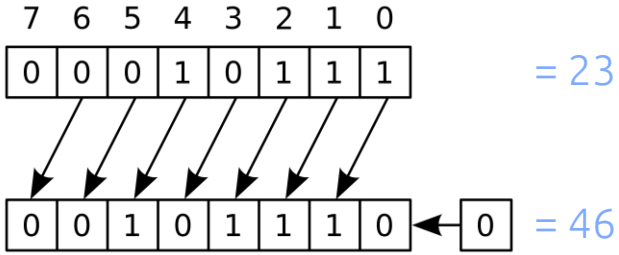
ALU: Shift Right / Shift Left

- **Shift Right**: Desplaza todos los dígitos (bits) a la derecha.
- **Shift Left**: Desplaza todos los dígitos (bits) a la izquierda.



Ejemplo de Shift Right

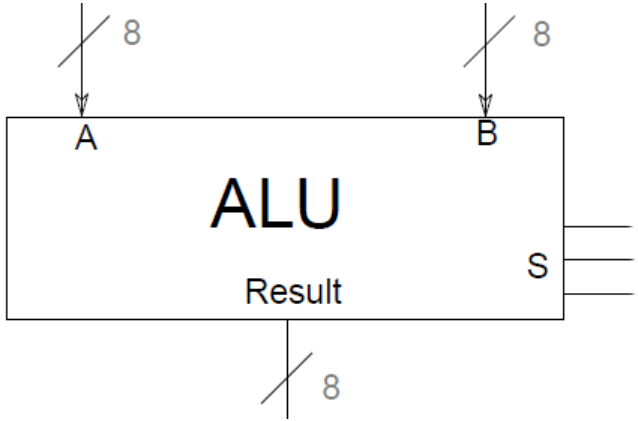
¡División Entera por 2!



Ejemplo de Shift Left

¡Multiplicación por 2!

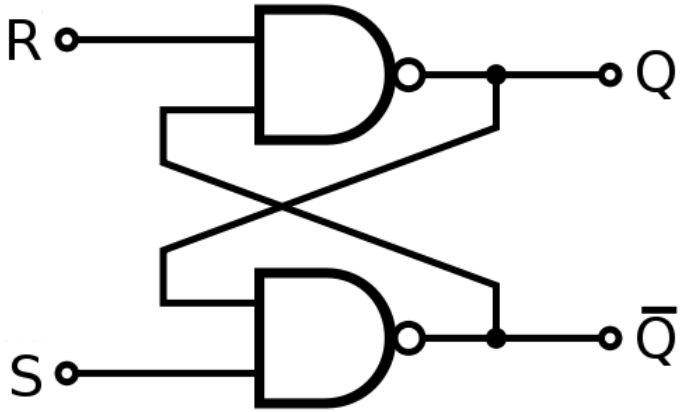
ALU (Unidad Aritmética Lógica)



S2	S1	S0	M
0	0	0	Suma
0	0	1	Resta
0	1	0	And
0	1	1	Or
1	0	0	Not
1	0	1	Xor
1	1	0	Shift left
1	1	1	Shift right

Lógica Digital: Almacenamiento

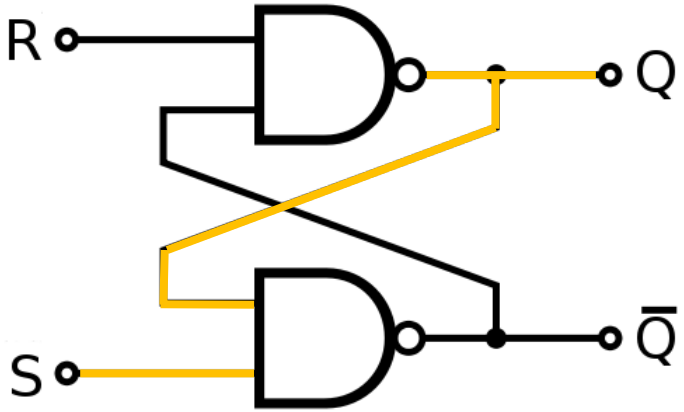
Latch SR



R	S	Q^{t+1}
0	0	-
0	1	1
1	0	0
1	1	Q^t

- Arovecha el retardo de propagación de un circuito para almacenar temporalmente un valor.
- El cambio en el valor de salida (output) depende tanto del estado presente como del estado previo.

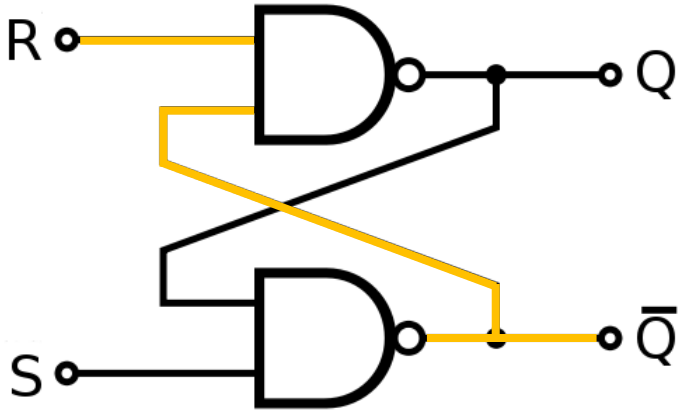
Latch SR



R	S	Q^{t+1}
0	0	-
0	1	1
1	0	0
1	1	Q^t

- Arovecha el retardo de propagación de un circuito para almacenar temporalmente un valor.
- El cambio en el valor de salida (output) depende tanto del estado presente como del estado previo.

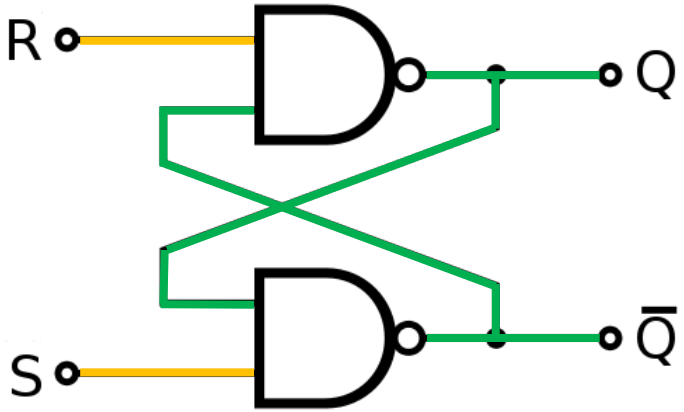
Latch SR



R	S	Q^{t+1}
0	0	-
0	1	1
1	0	0
1	1	Q^t

- Arovecha el retardo de propagación de un circuito para almacenar temporalmente un valor.
- El cambio en el valor de salida (output) depende tanto del estado presente como del estado previo.

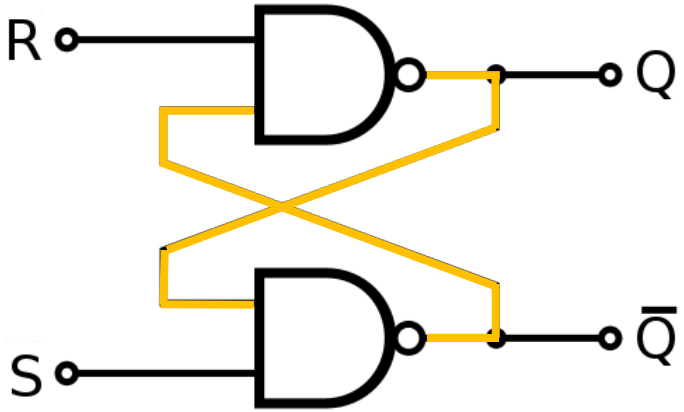
Latch SR



R	S	Q^{t+1}
0	0	-
0	1	1
1	0	0
1	1	Q^t

- Arovecha el retardo de propagación de un circuito para almacenar temporalmente un valor.
- El cambio en el valor de salida (output) depende tanto del estado presente como del estado previo.

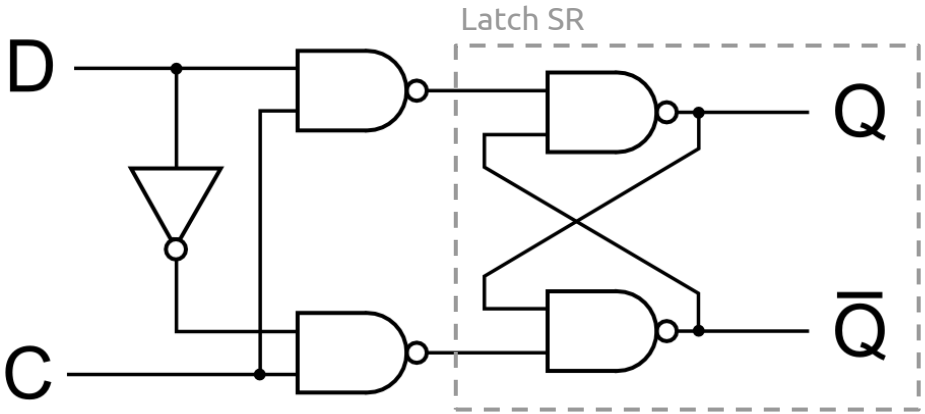
Latch SR



R	S	Q^{t+1}	
0	0	-	!!
0	1	1	
1	0	0	
1	1	Q^t	

- Arovecha el retardo de propagación de un circuito para almacenar temporalmente un valor.
- El cambio en el valor de salida (output) depende tanto del estado presente como del estado previo.

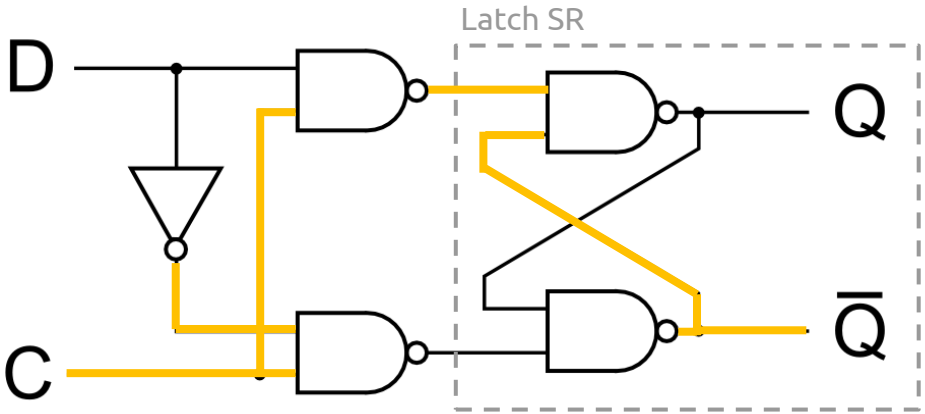
Latch D



C	D	Q^{t+1}
0	0/1	Q^t
1	0	0
1	1	1

- Mismo objetivo que el Latch SR: almacenar temporalmente un valor.
- Evita el caso no determinado del Latch SR.

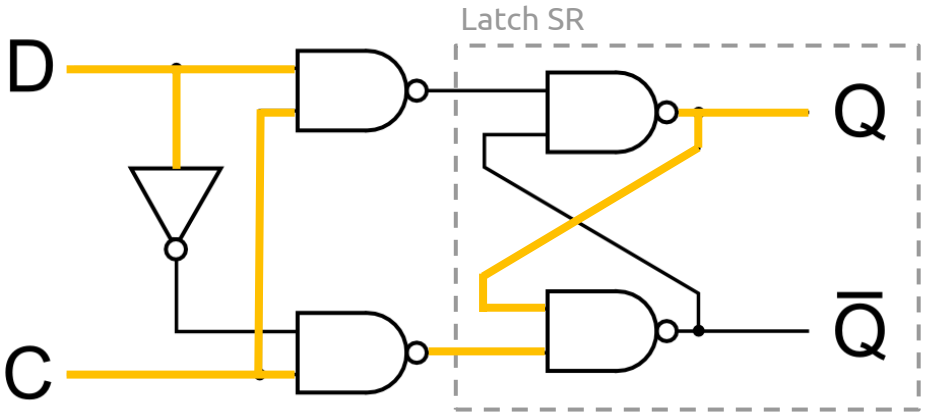
Latch D



C	D	Q^{t+1}
0	0/1	Q^t
1	0	0
1	1	1

- Mismo objetivo que el Latch SR: almacenar temporalmente un valor.
- Evita el caso no determinado del Latch SR.

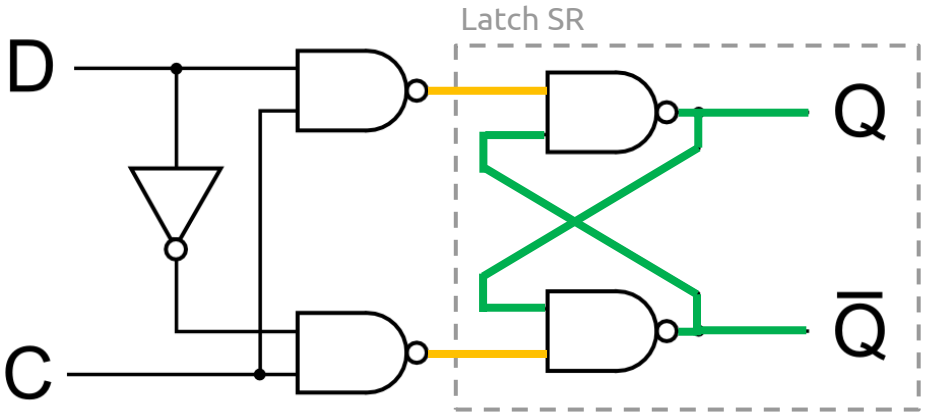
Latch D



C	D	Q^{t+1}
0	0/1	Q^t
1	0	0
1	1	1

- Mismo objetivo que el Latch SR: almacenar temporalmente un valor.
- Evita el caso no determinado del Latch SR.

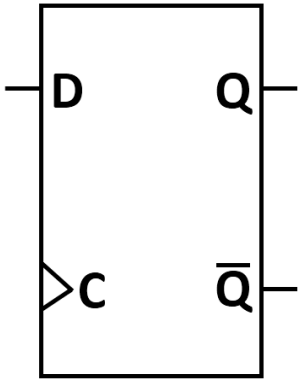
Latch D



C	D	Q^{t+1}
0	0/1	Q^t
1	0	0
1	1	1

- Mismo objetivo que el Latch SR: almacenar temporalmente un valor.
- Evita el caso no determinado del Latch SR.

Latch D

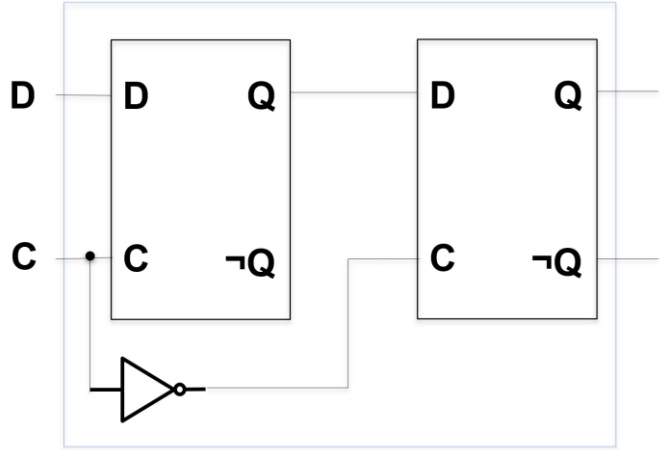


D: dato (input)
C: control (clock)
***Q*: estado** (output)

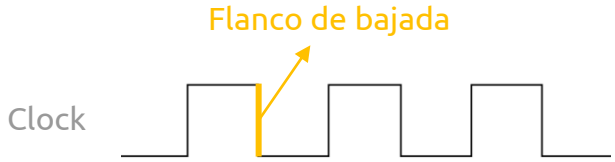
C	D	Q^{t+1}
0	0/1	Q^t
1	0	0
1	1	1

- Mismo objetivo que el Latch SR: almacenar temporalmente un valor.
- Evita el caso no determinado del Latch SR.

Flip-Flop D



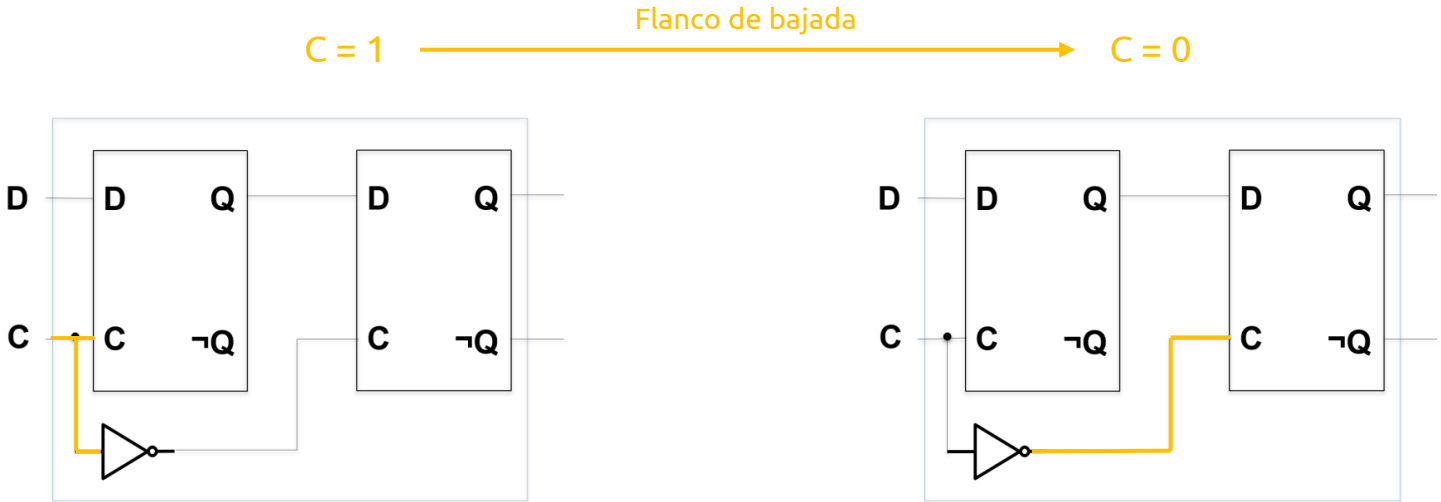
C	D	Q^{t+1}
0/1/↑	0/1	Q^t
↓	0	0
↓	1	1



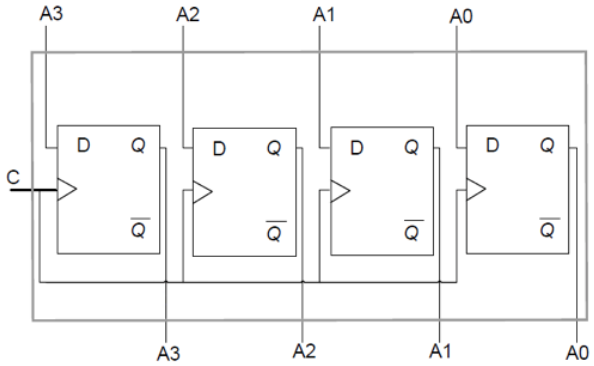
- Construido a partir de Latches D.
- Su estado (output) se actualiza solo en un instante de tiempo determinado (flanco de bajada).

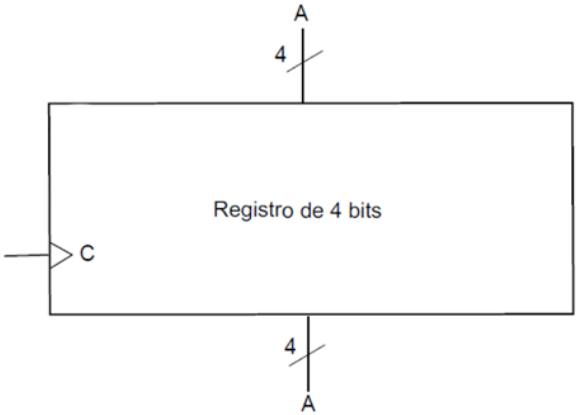
Flip-Flop D

¿Por qué este flip flop se carga en flanco de bajada?



Registro





- Arreglo de Flip-Flops D.
- Útil para almacenar datos de varios bits.

Procesador

Procesador: Componentes

- ALU
- Registros: A, B, PC, Status
- Clock
- Instruction Memory
- Control Unit
- Data Memory

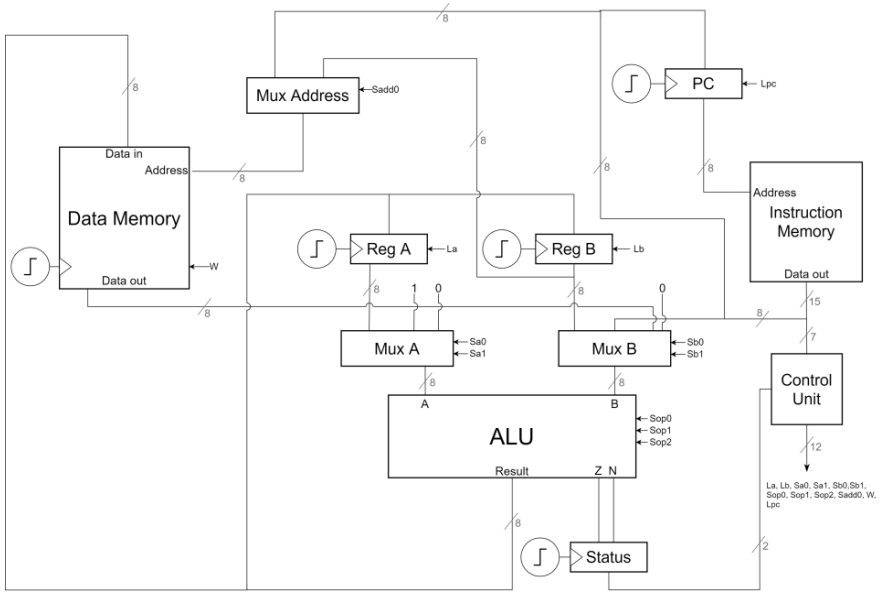


Diagrama Computador Básico

¿Cómo se programa en el computador básico?

Procesador: Assembly

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

Instrucción	Operandos	Operación	Condiciones	Ejemplo de uso
CMP	A,B	A-B		
	A,Lit	A-Lit		CMP A,0
JEQ	Dir	PC = Dir	Z=1	JEQ label
JNE	Dir	PC = Dir	Z=0	JNE label
JGT	Dir	PC = Dir	N=0 y Z=0	JGT label
JLT	Dir	PC = Dir	N=1	JLT label
JGE	Dir	PC = Dir	N=0	JGE label
JLE	Dir	PC = Dir	Z=1 o N=1	JLE label

Ejercicios

1. Un registro de 8 bits tiene almacenado el número entero 114. ¿Qué número se obtiene al realizar 2 operaciones shift right seguido de 3 operaciones shift left?

Tenemos que $(114)_{10} = (01110010)_2$. Entonces, los pasos a realizar son los siguientes:

Paso 1: Shift Right: 00111001
Paso 2: Shift Right: 00011100
Paso 3: Shift Left: 00111000
Paso 4: Shift Left: 01110000
Paso 5: Shift Left: 11100000

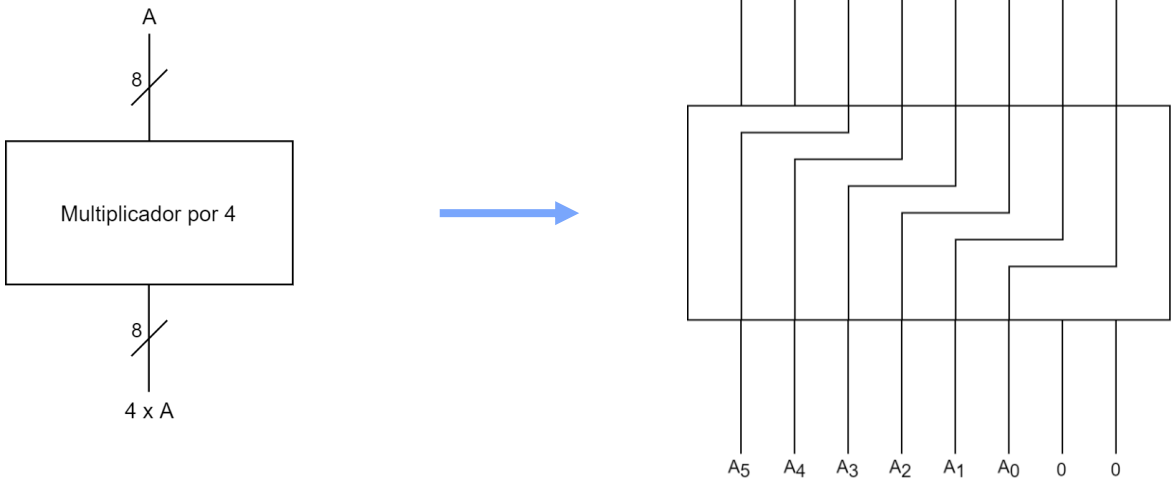
Resultado: $(11100000)_2 = (-32)_{10}$

Observaciones importantes:

- Al hacer operaciones shift es posible perder información (bit más significativo o menos significativo dependiendo del caso).
- Al hacer shift left, es posible sobrescribir el “bit de signo”, obteniendo un número completamente diferente.

2. Diseñe un circuito que multiplique por 4 un número natural de 8 bits. Asuma que no ocurrirá overflow. [I1-2016-1]

Podemos aprovechar el concepto de shift left para la multiplicación de números binarios.



3. ¿Es posible utilizar una ALU de 8 bits para realizar en paralelo dos sumas de números de 4 bits? Explique claramente su respuesta [I1-2014-2]

ALU recibe dos inputs, A y B de 8 bits cada uno.

Podemos dividir cada número en dos, y utilizar los 4 bits más significativos para almacenar un número y los 4 bits menos significativos para almacenar otro.

$A = a_7a_6a_5a_4a_3a_2a_1a_0$ → $A_1 = a_7a_6a_5a_4$, $A_2 = a_3a_2a_1a_0$

$B = b_7b_6b_5b_4b_3b_2b_1b_0$ → $B_1 = b_7b_6b_5b_4$, $B_2 = b_3b_2b_1b_0$

ADD A, B

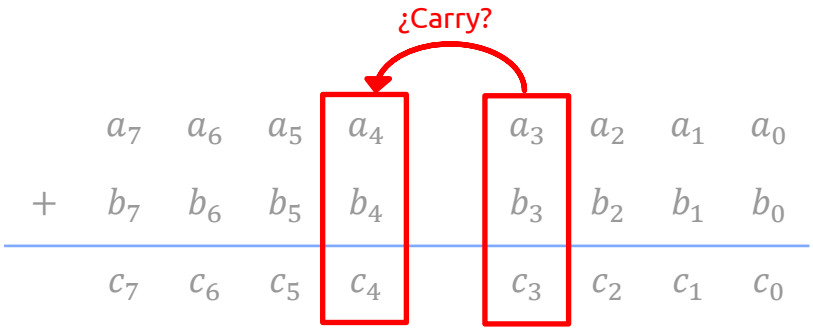
Se realizarán las sumas:

- $A_1 + B_1$
- $A_2 + B_2$

De forma simultánea

3. ¿Es posible utilizar una ALU de 8 bits para realizar en paralelo dos sumas de números de 4 bits? Explique claramente su respuesta [I1-2014-2]

Entonces:



Finalmente:

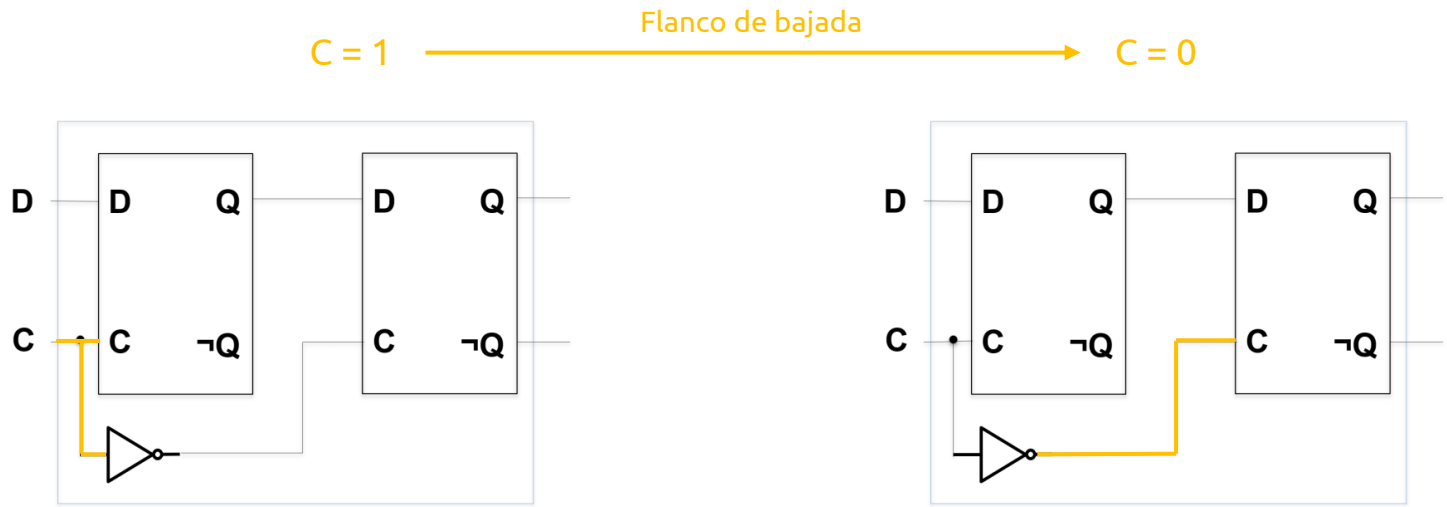
$$C_1 = A_1 + B_1$$
$$C_2 = A_2 + B_2$$

!! Es necesario hacerse cargo de un posible carry entre el bit 3 y 4. De lo contrario, las sumas paralelas no serán consistentes. Esto se puede llevar a cabo modificando el circuito de la ALU, añadiendo un Mux por ejemplo, y una señal de control correspondiente que permita desechar este carry al realizar este tipo suma.

3. ¿Es posible utilizar una ALU de 8 bits para realizar en paralelo dos sumas de números de 4 bits? Explique claramente su respuesta [I1-2014-2]

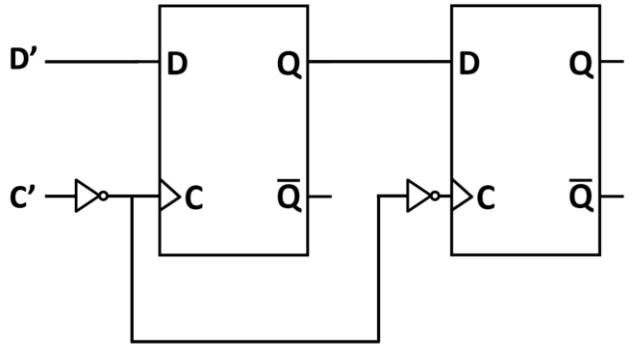
Conclusión: sí es posible realizar dos sumas paralelas de números de 4 bits en una ALU de 8 bits, siempre y cuando se hagan las modificaciones correspondientes al computador básico, con tal de evitar un posible carry entre el bit 3 y 4 de la ALU.

4. Implemente mediante compuertas lógicas, elementos de control y latches, un flip-flop D que funcione con flanco de subida.



4. Implemente mediante compuertas lógicas, elementos de control y latches, un flip-flop D que funcione con flanco de subida.

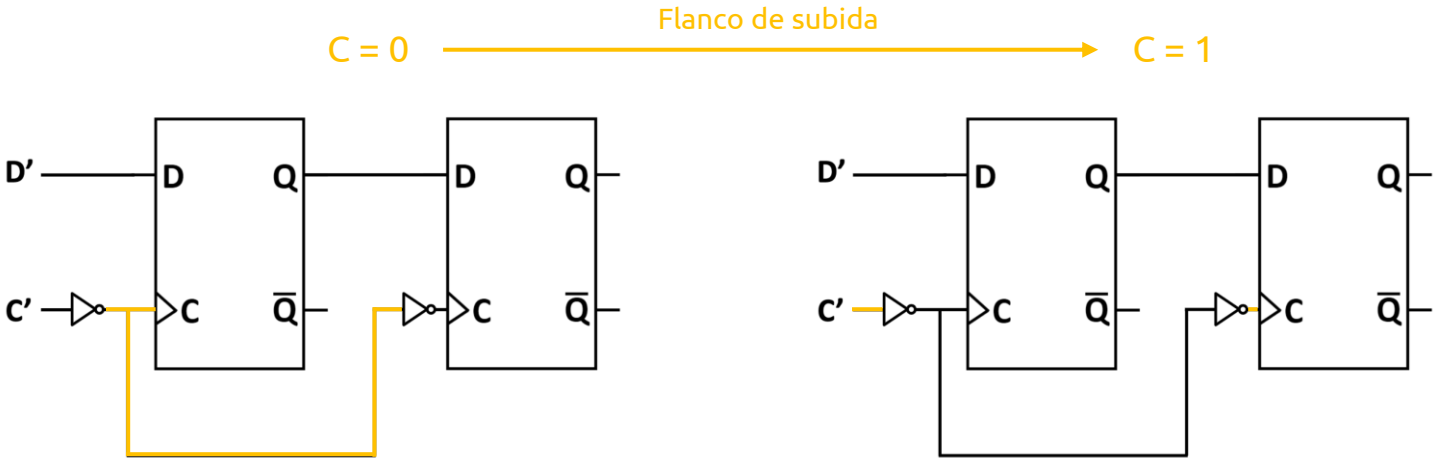
Añadimos una compuerta NOT a la señal de control (C).



C	D	Q^{t+1}
0/1/↓	0/1	Q^t
↑	0	0
↑	1	1

4. Implemente mediante compuertas lógicas, elementos de control y latches, un flip-flop D que funcione con flanco de subida.

¿Por qué este flip flop se carga en flanco de subida?



5. ¿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

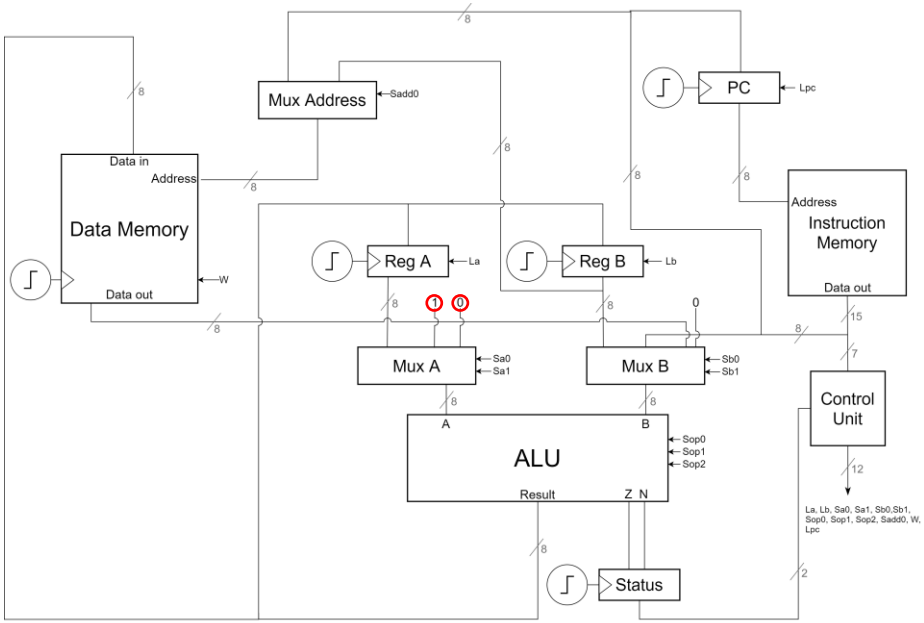
5. ¿En qué casos es posible soportar la instrucción ADD B, Lit en el computador básico, sin modificar su hardware ni sobrescribir datos? Para los casos negativos, indique modificaciones al hardware y/o assembly que se deberían hacer para soportarlos. [I1-2016-1]

ADD	A,B	000100
	B,A	000101
	A,Lit	000110

Según el diagrama del computador básico, las únicas instrucciones soportadas son:

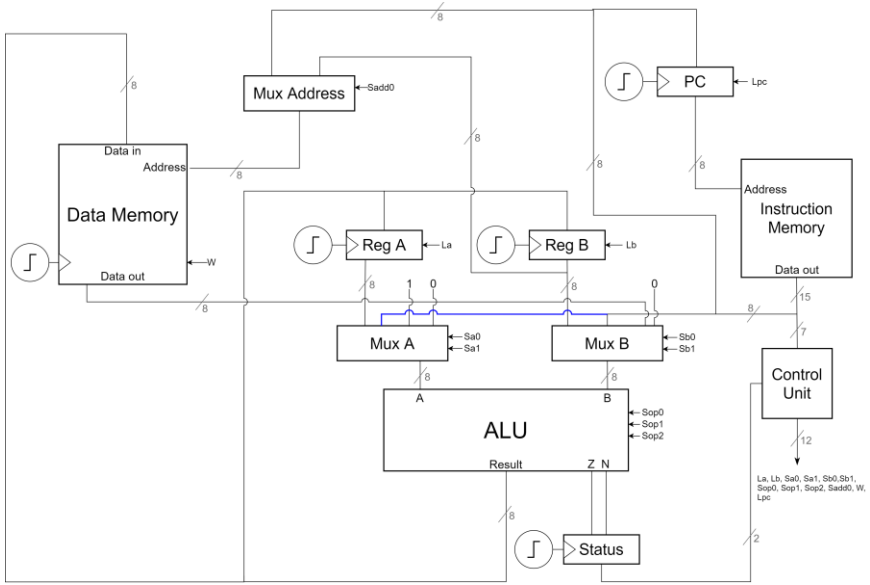
- ADD B, 0
- ADD B, 1

¿Qué habría que modificar para soportar cualquier valor para Lit?



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

Agregar bus de literales a Mux A



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

… y agregar instrucción al assembly.

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
…	…										

¡Muchas gracias!