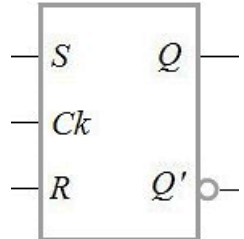
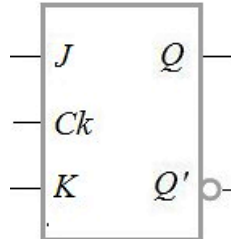
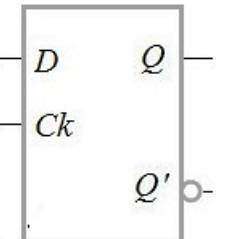
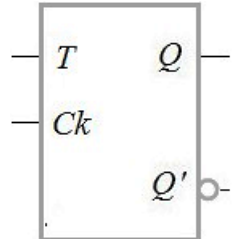


1. Registros
2. Memorias RAM
3. Nivel de transferencia de registros (RTL)

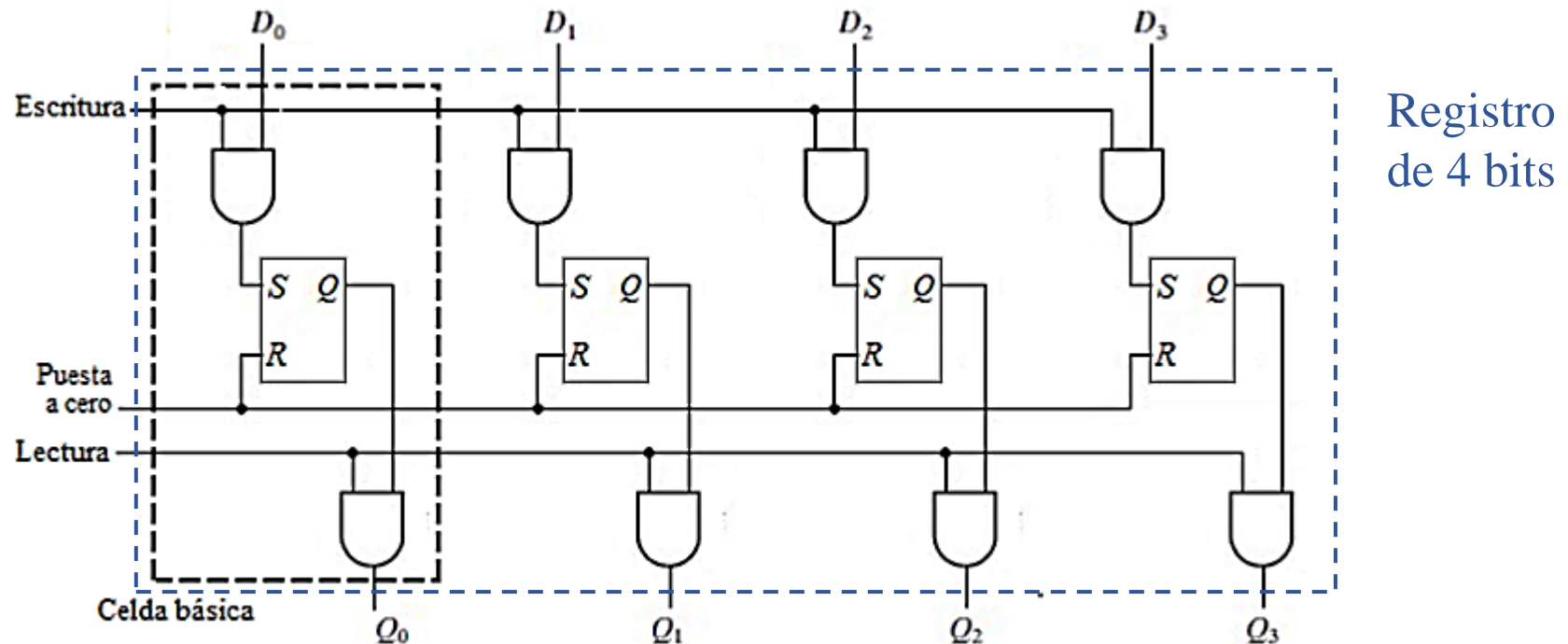


# Resumen de biestables

	$R-S$	$J-K$	$D$	$T$
Tabla característica	$  \begin{array}{cc c}  S & R & Q_{(t+1)} \\  \hline  0 & 0 & Q_{(t)} \\  0 & 1 & 0 \\  1 & 0 & 1 \\  1 & 1 & X  \end{array}  $	$  \begin{array}{cc c}  J & K & Q_{(t+1)} \\  \hline  0 & 0 & Q_{(t)} \\  0 & 1 & 0 \\  1 & 0 & 1 \\  1 & 1 & Q_{(t)}'  \end{array}  $	$  \begin{array}{c c}  D & Q_{(t+1)} \\  \hline  0 & 0 \\  1 & 1  \end{array}  $	$  \begin{array}{c c}  T & Q_{(t+1)} \\  \hline  0 & Q_{(t)} \\  1 & Q_{(t)}'  \end{array}  $
Tabla de excitación	$  \begin{array}{cc cc}  Q_{(t)} & Q_{(t+1)} & S & R \\  \hline  0 & 0 & 0 & X \\  0 & 1 & 1 & 0 \\  1 & 0 & 0 & 1 \\  1 & 1 & X & 0  \end{array}  $	$  \begin{array}{cc cc}  Q_{(t)} & Q_{(t+1)} & J & K \\  \hline  0 & 0 & 0 & X \\  0 & 1 & 1 & X \\  1 & 0 & X & 1 \\  1 & 1 & X & 0  \end{array}  $	$  \begin{array}{cc c}  Q_{(t)} & Q_{(t+1)} & D \\  \hline  0 & 0 & 0 \\  0 & 1 & 1 \\  1 & 0 & 0 \\  1 & 1 & 1  \end{array}  $	$  \begin{array}{cc c}  Q_{(t)} & Q_{(t+1)} & T \\  \hline  0 & 0 & 0 \\  0 & 1 & 1 \\  1 & 0 & 1 \\  1 & 1 & 0  \end{array}  $
$Q_{(t+1)} =$	$S + R'Q_{(t)}$	$JQ_{(t)}' + K'Q_{(t)}$	$D$	$TQ_{(t)}' + T'Q_{(t)}$
Símbolo				

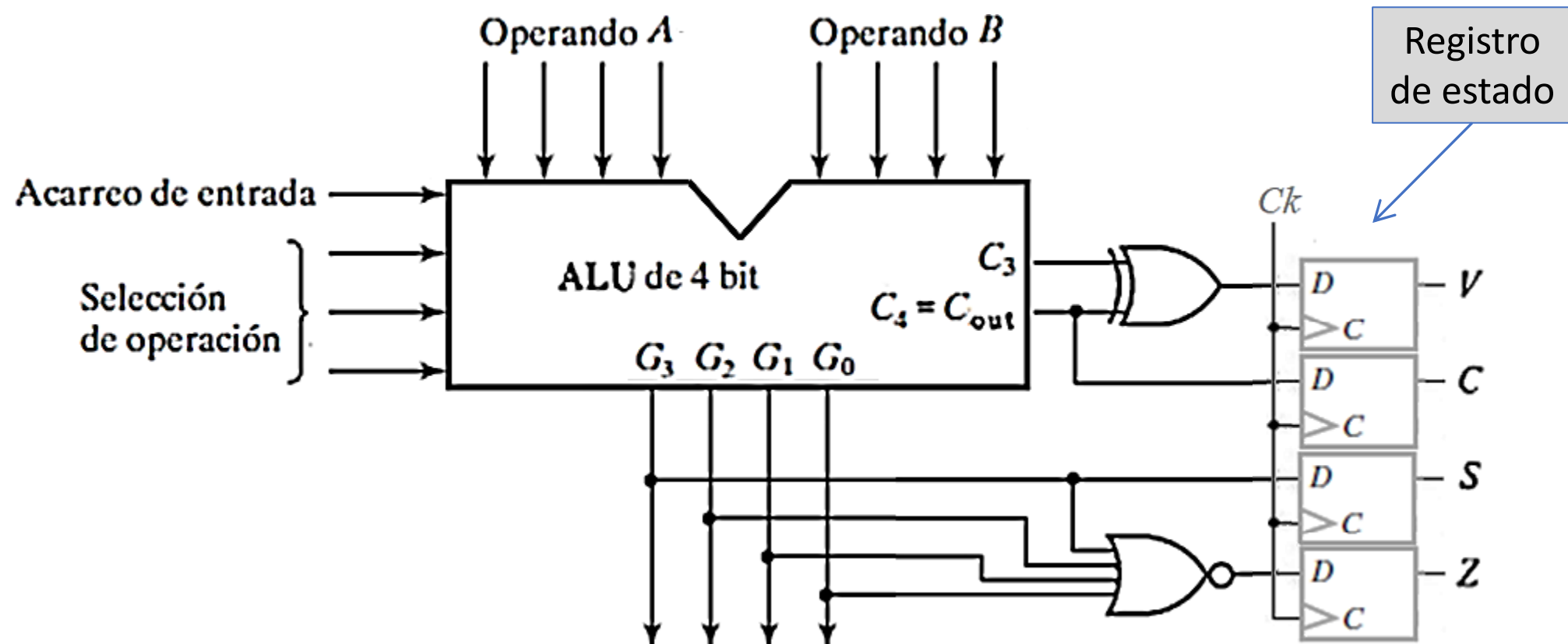
# Registros

- Un biestable tiene la capacidad de almacenar 1 bit.
- Un *registro* (o *registro paralelo*) es un conjunto de  $n$  biestables que tienen en general una o más líneas comunes para la selección de una o más operaciones en todos ellos.



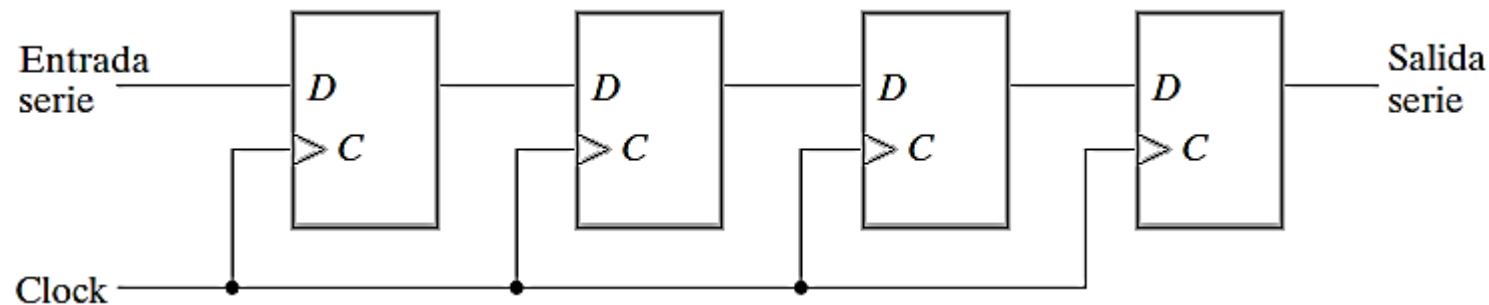
# Registro de estado

La palabra de estado ( $V$ ,  $C$ ,  $S$  y  $Z$ ) que resulta de una operación de una ALU debe memorizarse para que los *flags* sean utilizados en operaciones siguientes :



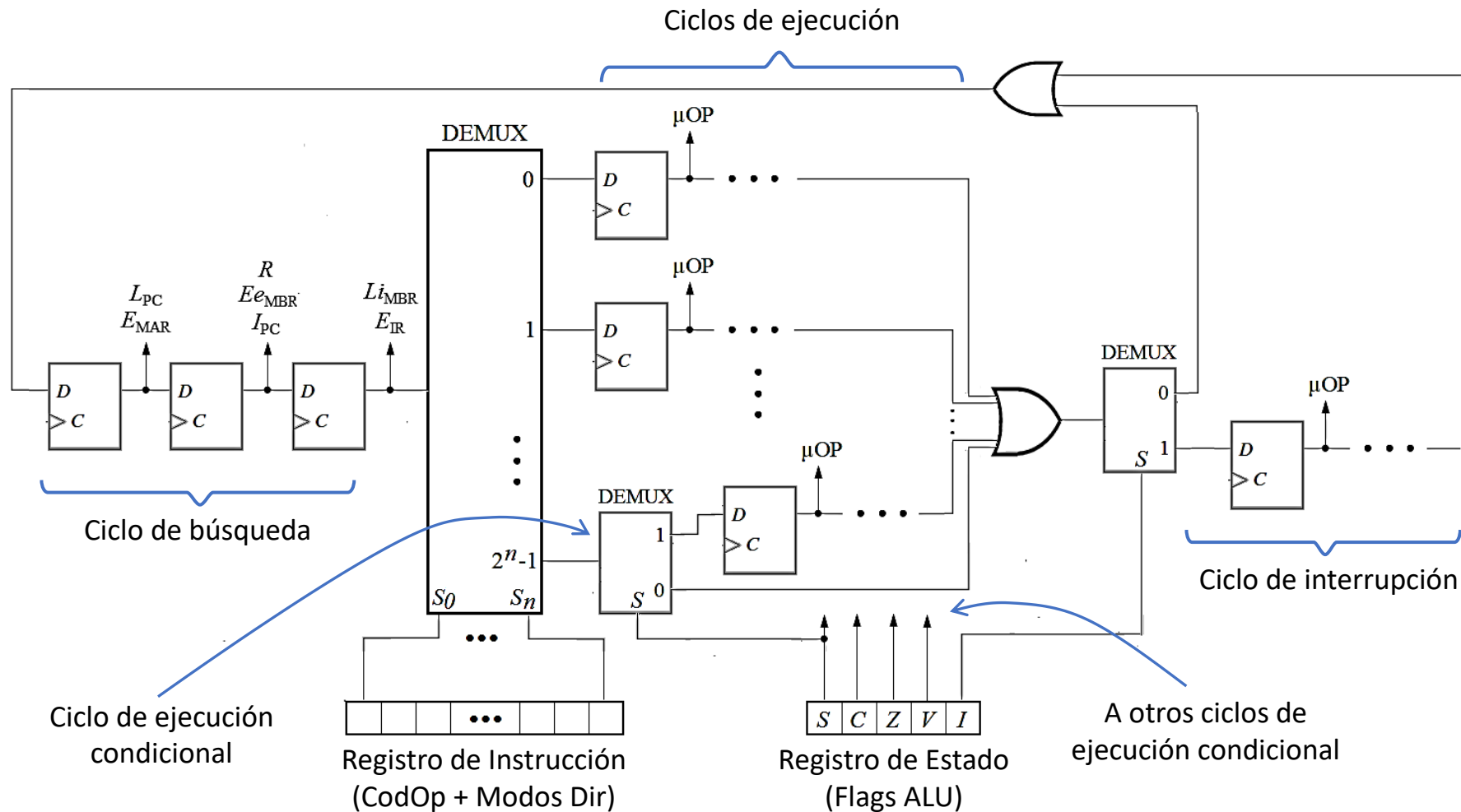
# Registros de desplazamiento

- Un *registro de desplazamiento* traslada en forma secuencial un pulso a lo largo de una cadena de biestables.
- En cada pulso de reloj, el valor presente en la entrada en serie se copia en el primer biestable, y los contenidos de cada uno se copian en el siguiente de la cadena. El contenido del último se pierde.
- La señal de reloj es común a todos los flip-flops, por lo que deben ser de tipo M/E para que la operación descrita sea posible.



# Desplazamiento con bifurcaciones condicionales

Unidad de Control simple:

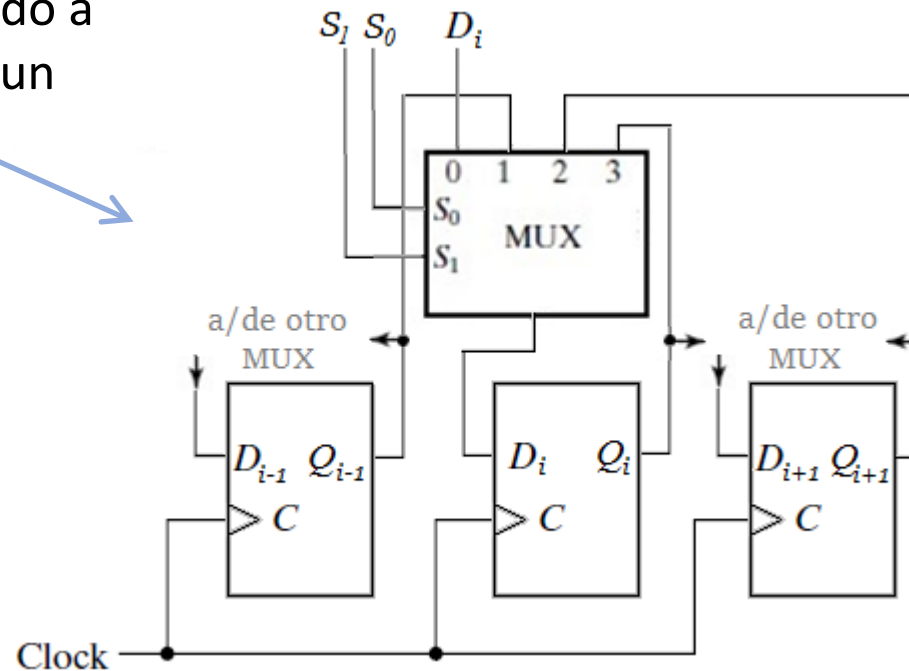


# Registro de desplazamiento a izquierda/derecha y carga en paralelo

Con la adición de lógica combinatorial, es posible controlar el sentido del desplazamiento en el registro a derecha o izquierda. Asimismo, es factible cargar un valor binario en forma directa cada biestable, o hacer que estos queden sin cambios en cada transición del reloj.

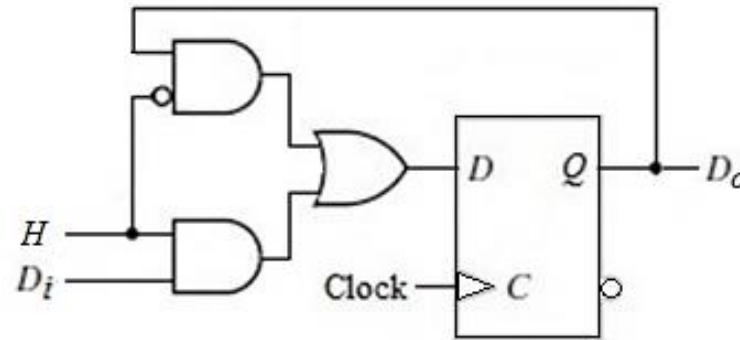
Se representa el combinacional asociado a cada FF; para cada uno de ellos existe un MUX conectado en forma análoga.

Modo de control		Operaciones del registro
$S_1$	$S_0$	
0	0	Carga paralela
0	1	Desplaza hacia derecha
1	0	Desplaza hacia izquierda
1	1	No cambia

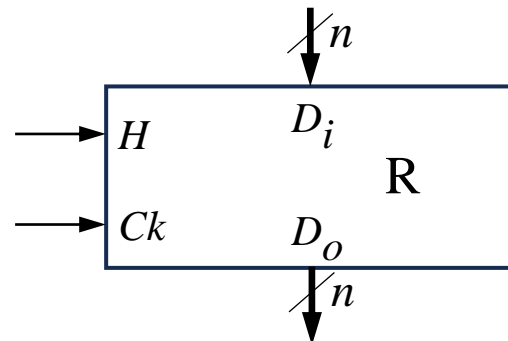


# Registros con habilitación

En sistemas con varios registros, es necesario controlar en cual de ellos se va a realizar una operación de escritura, lo que se implementa mediante una línea de habilitación  $H$ :



Para un registro  
de  $n$  bits:

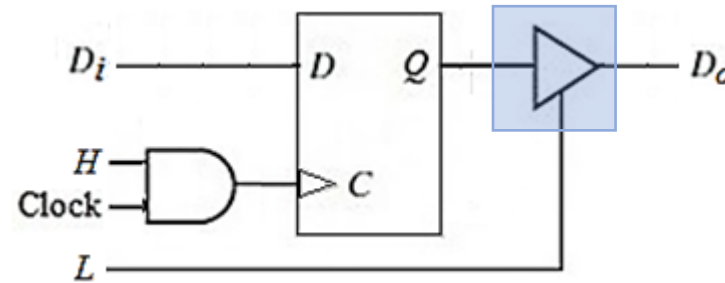


$H$	$Do$	Operación
0	$Do$	Mantiene el valor (lectura)
1	$Di$	Carga el valor de entrada (escritura)

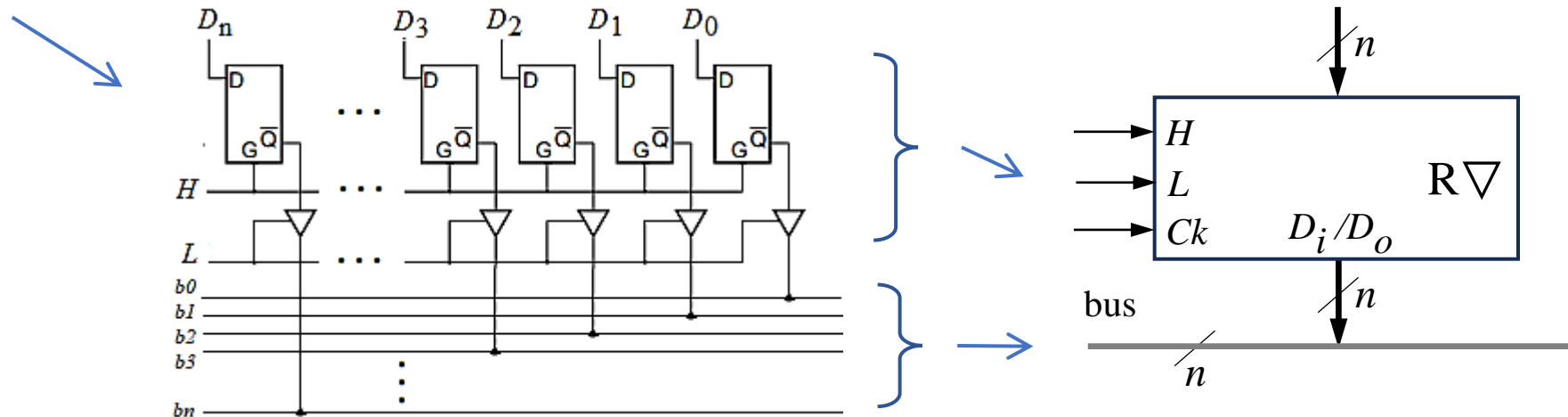


# Registros con conexión a bus

Los registros utilizados en sistemas con buses triestado, deben utilizar un separador de alta impedancia (“tri-state”) en la habilitación de salida.

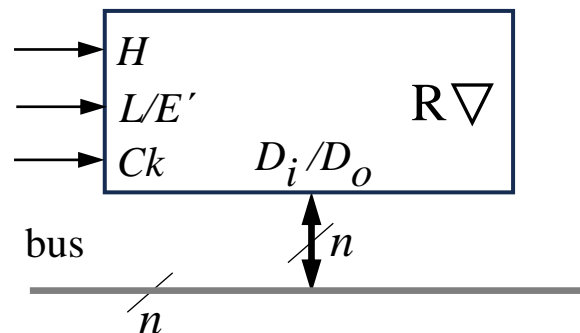
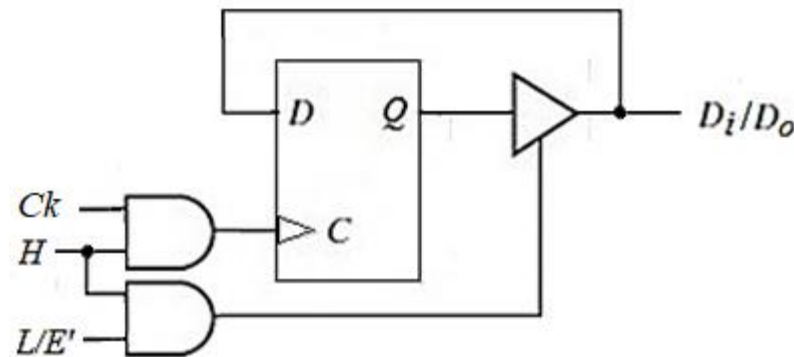


Para un registro de  $n$  bits:



# Registros con conexión a bus

En los casos en los que los buses son bidireccionales (es decir actúan como entradas y salidas):

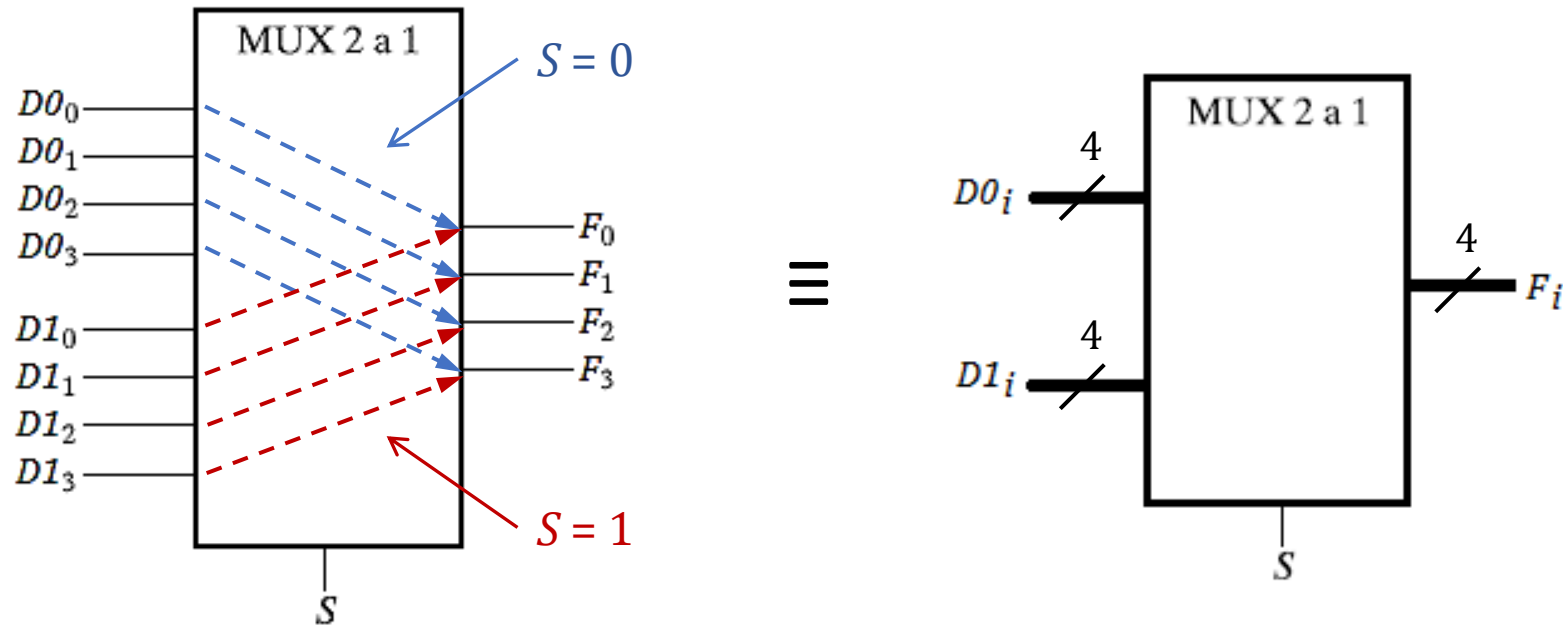


$H$	$L/E'$	Operación
1	0	Escritura desde el bus
1	1	Lectura en el bus
0	X	Sin cambios

# Multiplexores de varias vías

Las  $n$  entradas del multiplexor también pueden estar constituidas por  $m$  líneas, que se seleccionan para ser presentadas en su salida, también compuestas por  $m$  líneas.

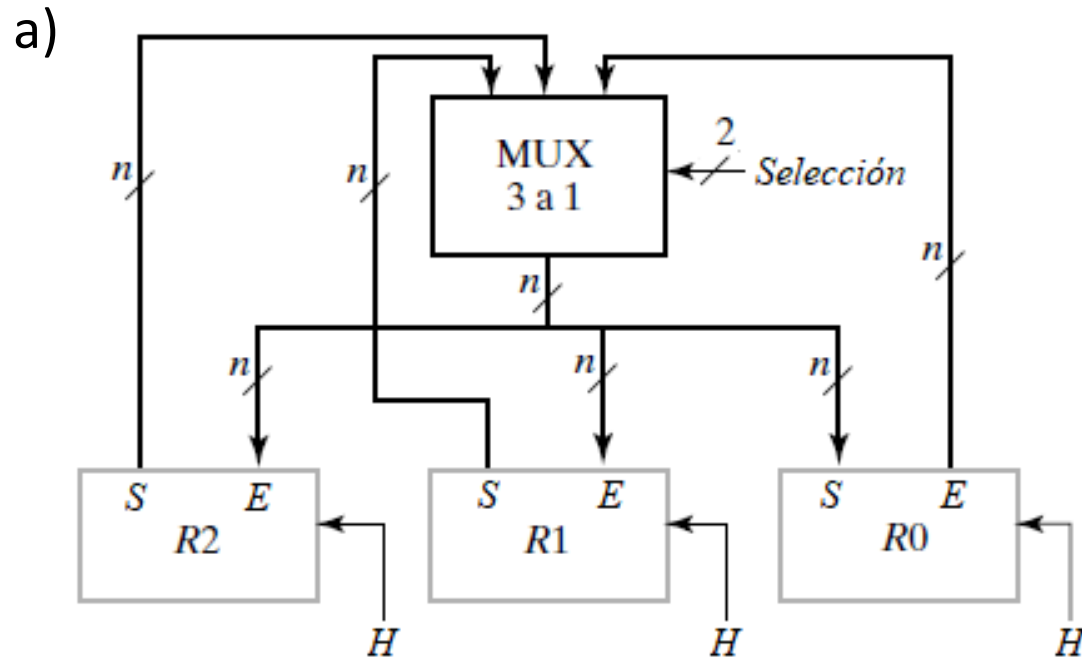
Por ejemplo, para un multiplexor cuádruple de 2x1:



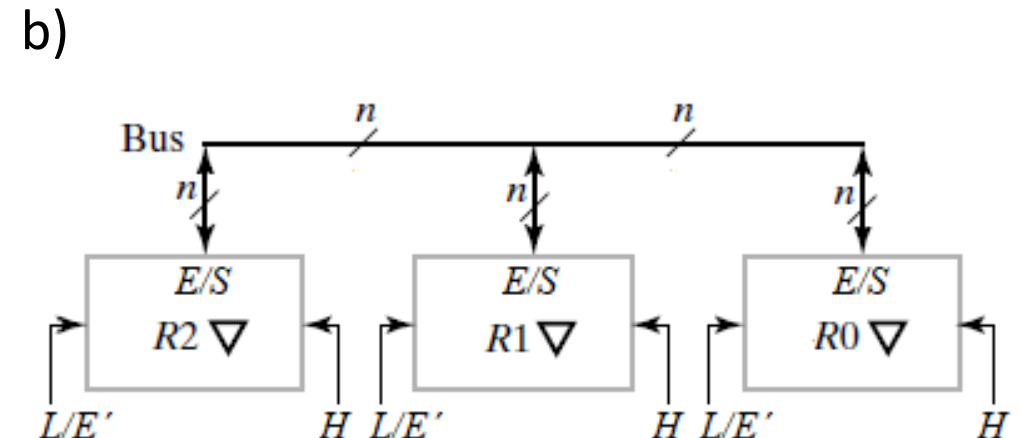
# Buses multiplexados y buses *tri-state*

La transferencia de información entre varios registros se realiza mediante dos técnicas:

- a) Bus multiplexado
- b) Bus triestado

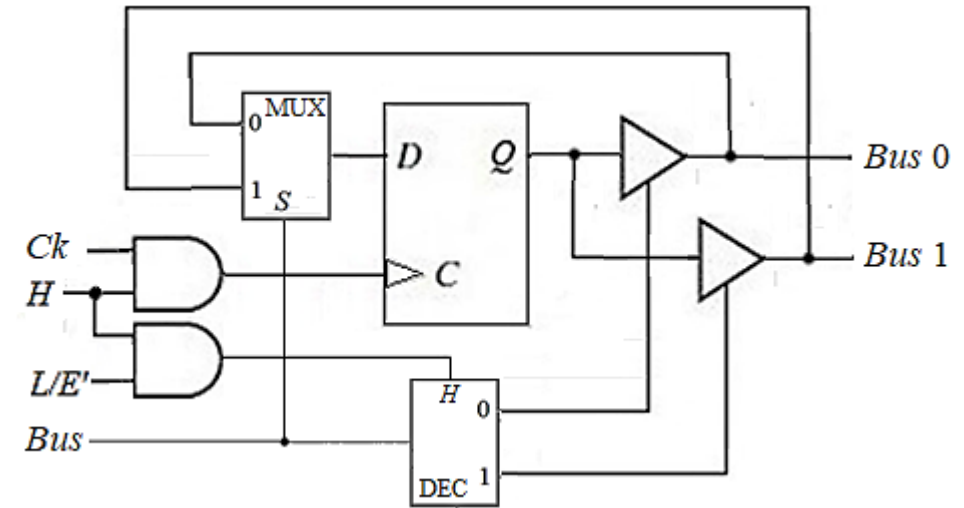
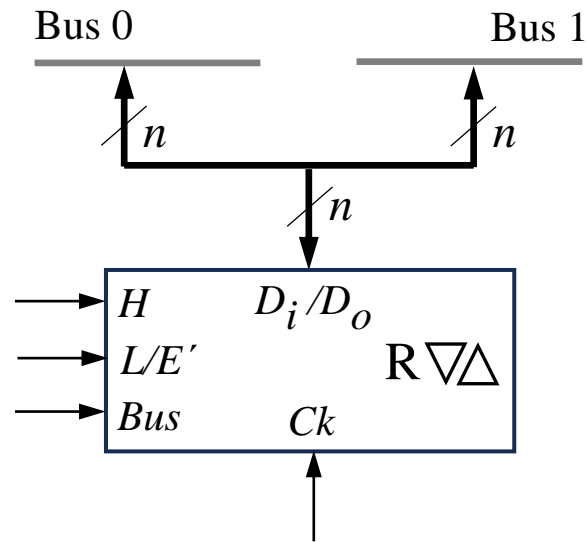


*E*: Entradas  
*S*: Salidas  
*H*: Habilitación  
*L/E'*: Lectura/(Escritura)'



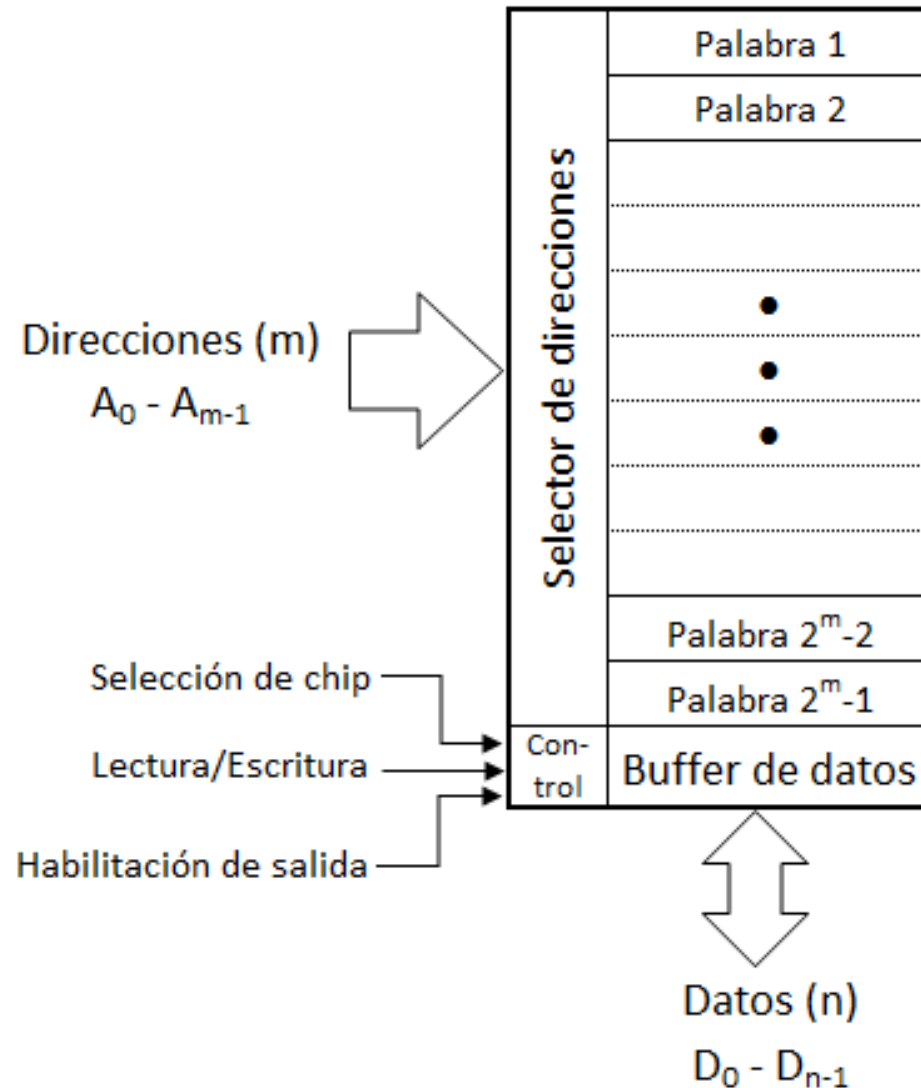
# Registro con comunicación con dos buses

Un registro como el **Separador de Memoria** (MBR, *Memory buffer register*) es un registro que puede leer en una línea (bus) y escribir en otra, y viceversa:



$H$	$Bus$	$L/E'$	Operación
1	0	0	Escritura desde el bus 0
1	0	1	Lectura en el bus 0
1	1	0	Escritura desde el bus 1
1	1	1	Lectura en el bus 1
0	X	X	Sin cambios

# Memorias RAM



Desde el punto de vista de su esquema lógico una memoria RAM puede considerarse como un conjunto de  $m$  registros (*palabras*) de  $n$  bits, que pueden accederse (leerse o escribirse) *de uno a la vez*.

El acceso a cada palabra se realiza mediante un selector, que determina la seleccionada mediante su *dirección* binaria.

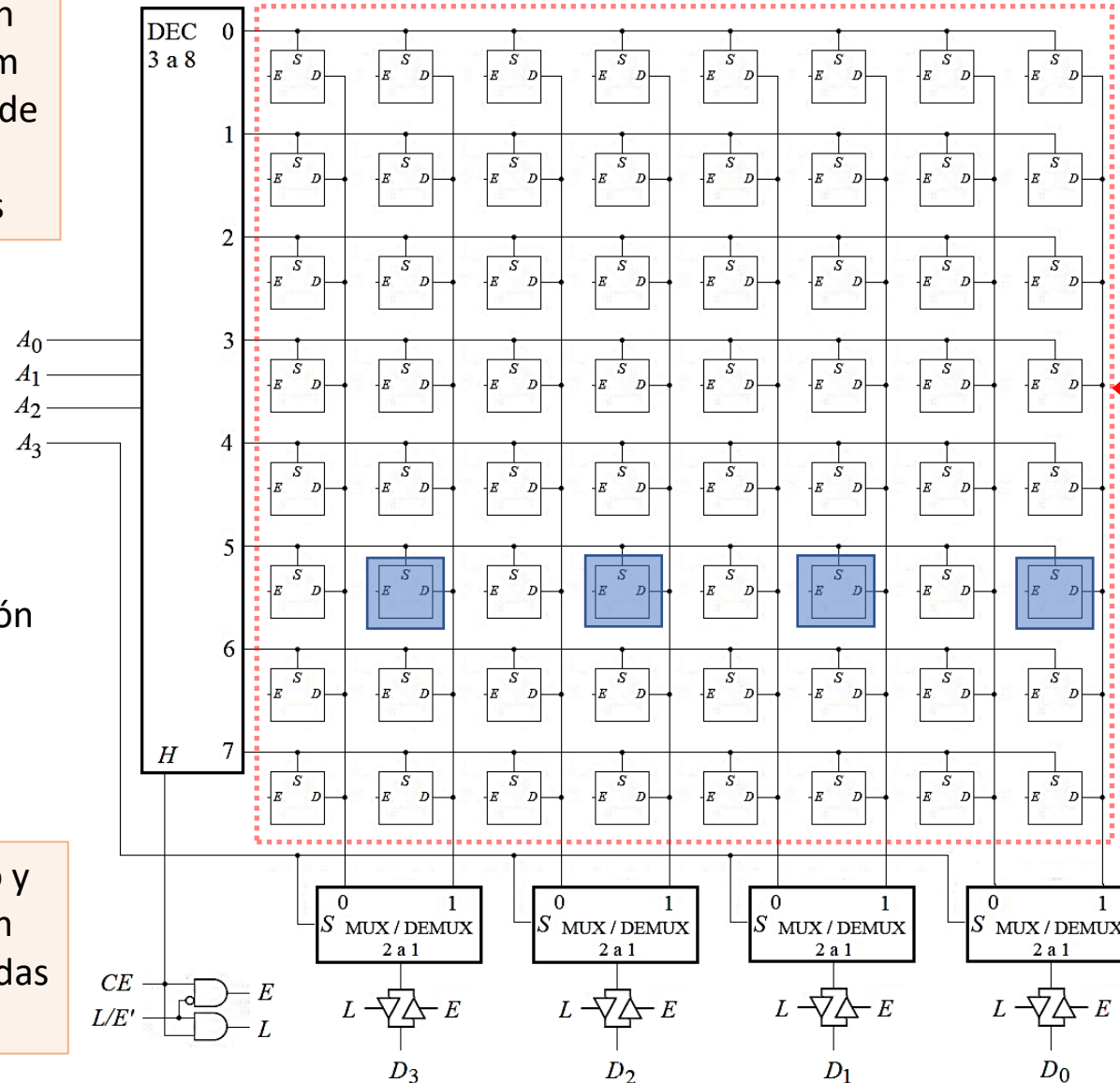
La operación a realizar se determina mediante un conjunto de líneas de control.

# Memorias RAM

Las palabras se seleccionan mediante un conjunto de  $m$  líneas de dirección ( $A_0$ - $A_m$ ), de modo que el tiempo de acceso es igual para todas

Ubicación de la palabra de la dirección **1101**

La selección del dispositivo y las operaciones se realizan desde un conjunto de entradas de control



Memoria de 16 palabras de 4 bits ( $n$ )

$$16 = 2^m \rightarrow m=4$$

Capacidad:  $2^m \times n = 64$  bits

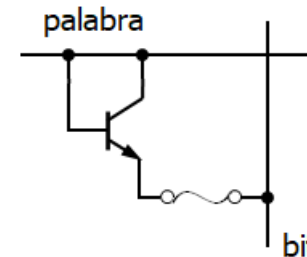
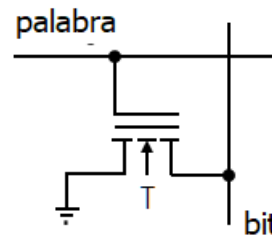
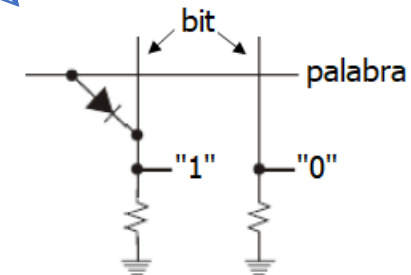
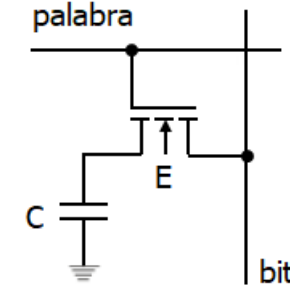
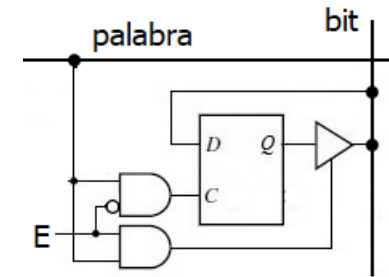
Generalmente se organizan en matrices cuadradas de  $n^{1/2} \times n^{1/2} = 8 \times 8$  bits

Las líneas de dirección se reparten:

- una parte en un decodificador para las filas de la matriz, y
- las líneas restantes seleccionan los bits en las columnas mediante multiplexores

# Memorias RAM

	Retención de la información	Tecnología		Elemento de memoria
Memorias de Acceso Aleatorio (RAM)	Volátiles	RAM estáticas (SRAM)		Latch lógico
		RAM dinámicas (DRAM)		Carga en capacitor
	No volátiles	ROM		Diodo o canal MOS
		PROM		Fusible
		RPROM	EPROM	Carga en compuerta flotante MOS
			EEPROM	
			FLASH	





# Operaciones entre registros

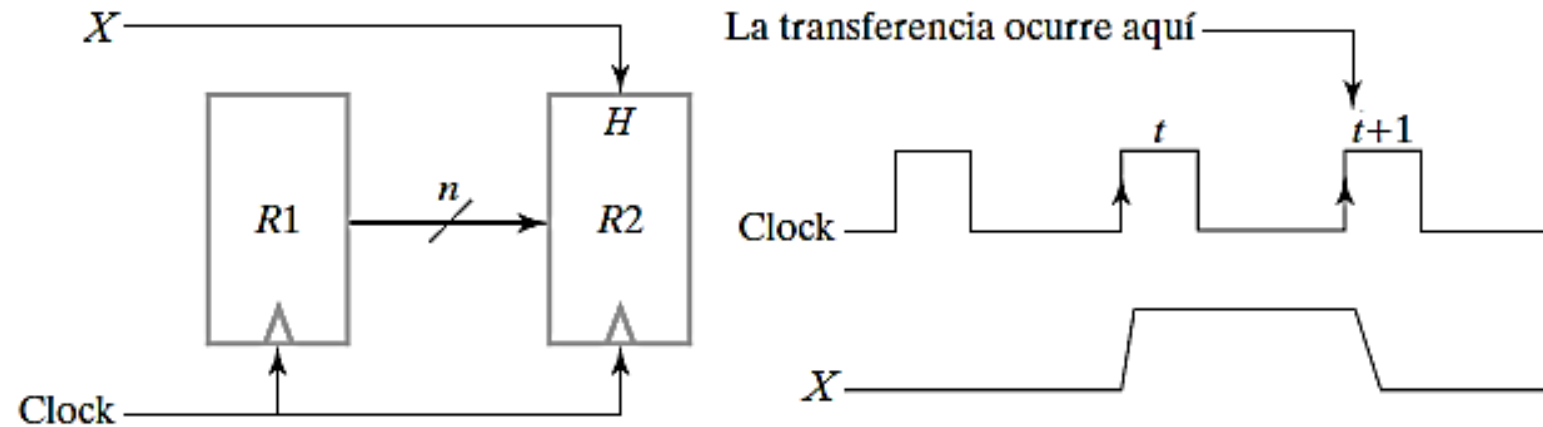
Las operaciones básicas realizada sobre los datos almacenados en un registro o entre dos de estos, y que tiene la particularidad de efectuarse en un solo ciclo de reloj, se denominan *microoperaciones*.

Se pueden clasificar en 4 tipos:

1. De transferencia, cuando se transfieren datos de un registro a otro.
2. Aritméticas, cuando se realizan operaciones aritméticas en los datos de los registros.
3. Lógicas, cuando se realizan operaciones lógicas sobre los bits de los datos de los registros.
4. De desplazamiento, cuando se desplazan los datos de los registros.

# Operaciones entre registros

Para el caso de una operación de transferencia entre dos registros:



Las microoperaciones se describen mediante la notación de Nivel de Transferencia de Registro (RTL, *Register Transfer Level*), que utiliza un conjunto de expresiones y sentencias particular. Así, por ejemplo, la transferencia de datos de un registro a otro será:

$$X: R2 \leftarrow R1$$

y significa que el contenido del registro R1 (fuente) se copia en el R2 (destino). El contenido del registro fuente no cambia.

# Notación RTL

La notación RTL utiliza la siguiente simbología :

Para indicar los elementos que componen o están relacionados con los registros:

Y para las operaciones que pueden realizarse entre ellos:



Símbolo	Descripción	Ejemplos
Letras (y números)	Indica un registro	$AR, R2, DR, IR$
Paréntesis	Indica parte de un registro	$R2(1), R2(7:0), AR(L)$
Flecha	Indica transferencia del dato	$R1 \leftarrow R2$
Coma	Separa transferencias simultáneas	$R1 \leftarrow R2, R2 \leftarrow R1$
Corchetes	Especifica una dirección de memoria	$DR \leftarrow M[AR]$

Operación	RTL
Asignación combinacional	$=$
Transferencia de registro	$\leftarrow$
Suma $+$	$+$
Resta $-$	$-$
Concatenación	$\parallel$
AND (Bit a Bit)	$\wedge$
OR (Bit a Bit)	$\vee$
XOR (Bit a Bit)	$\oplus$
NOT (Bit a Bit)	$-$
Desplazamiento lógico a la izquierda	$sl$
Desplazamiento a la derecha	$sr$
Vectores/registros	$A(3:0)$

# Notación RTL

Los distintos tipos de microoperaciones se denotan como:

## Microoperaciones aritméticas

Designación simbólica	Descripción
$R0 \leftarrow R1 + R2$	El contenido de $R1$ más el contenido de $R2$ se transfiere a $R0$
$R2 \leftarrow \overline{R2}$	Complemento del contenido de $R2$ (complemento a 1)
$R2 \leftarrow \overline{R2} + 1$	Complemento a 2 del contenido de $R2$
$R0 \leftarrow R1 + \overline{R2} + 1$	$R1$ más el complemento a 2 de $R2$ se transfiere a $R0$ (resta)
$R1 \leftarrow R1 + 1$	Incrementa el contenido de $R1$ (cuenta ascendente)
$R1 \leftarrow R1 - 1$	Decrementa el contenido de $R1$ (cuenta descendente)

## Microoperaciones lógicas

Designación simbólica	Descripción
$R0 \leftarrow \overline{R1}$	Bit a Bit lógico NOT (complemento a 1)
$R0 \leftarrow R1 \wedge R2$	Bit a Bit lógico AND (pone a 0 los bits)
$R0 \leftarrow R1 \vee R2$	Bit a Bit lógico OR (pone a 1 los bits)
$R0 \leftarrow R1 \oplus R2$	Bit a Bit lógico XOR (complementa bits)

## Ejemplos de desplazamientos

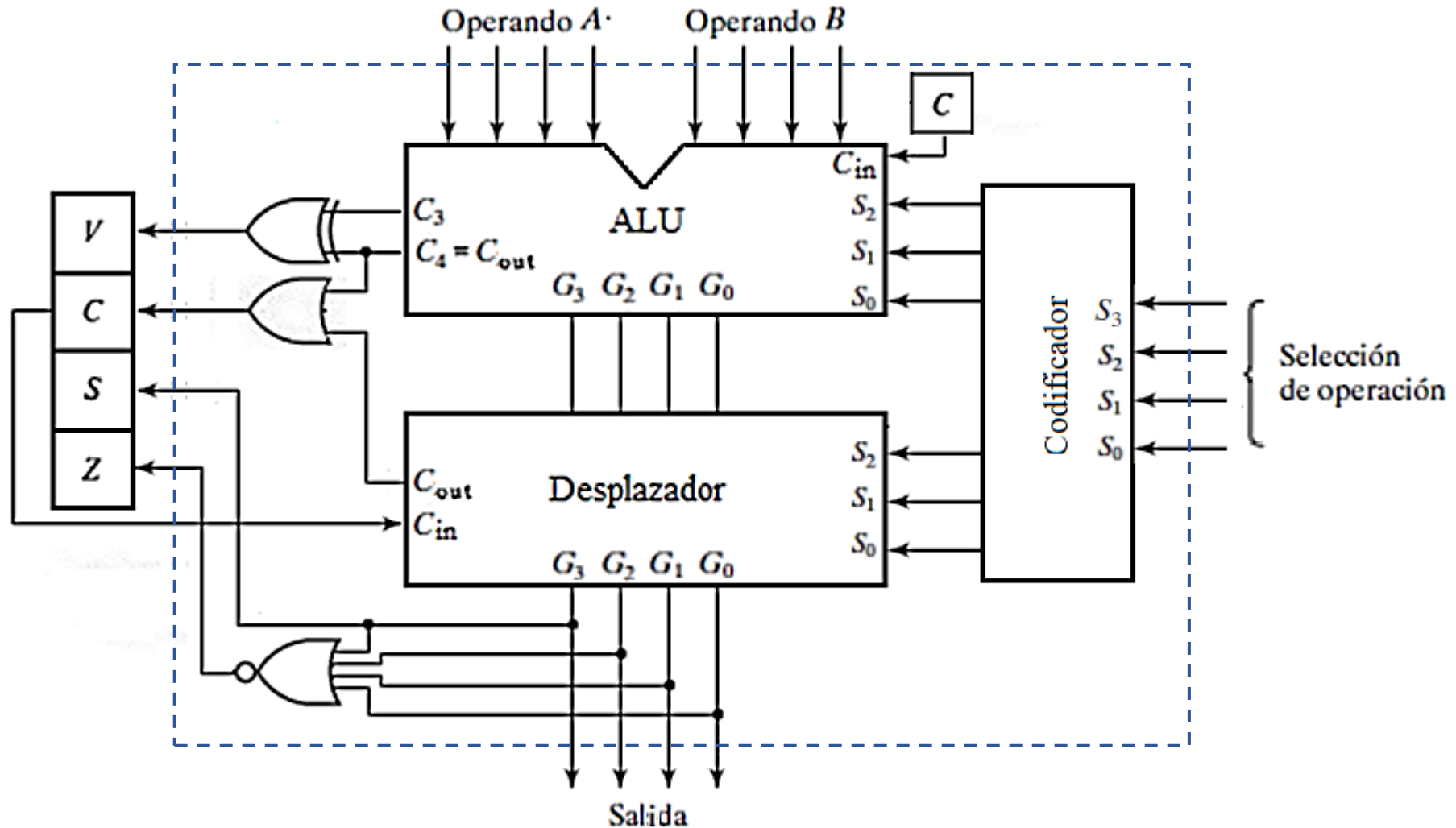
Tipos	Designación simbólica	Ejemplos de 8 bits	
		Fuente $R2$	Después de desplazar: Destino $R1$
Desplazamiento a la izquierda	$R1 \leftarrow sl\ R2$	10011110	00111100
Desplazamiento a la derecha	$R1 \leftarrow sr\ R2$	11100101	01110010

En sistemas con varios registros, deben observarse reglas sencillas, como:

- Un registro no puede leerse y escribirse al mismo tiempo;
- En sistemas basados en buses, debe asegurarse que solo un registro escriba en el bus al mismo tiempo.

# ALU con desplazador

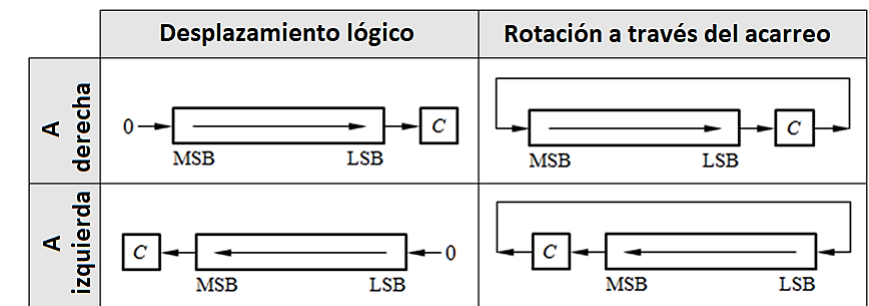
Combinando un desplazador combinacional con la ALU diseñada, resulta:



# Unidad Aritmética Lógica + Desplazador

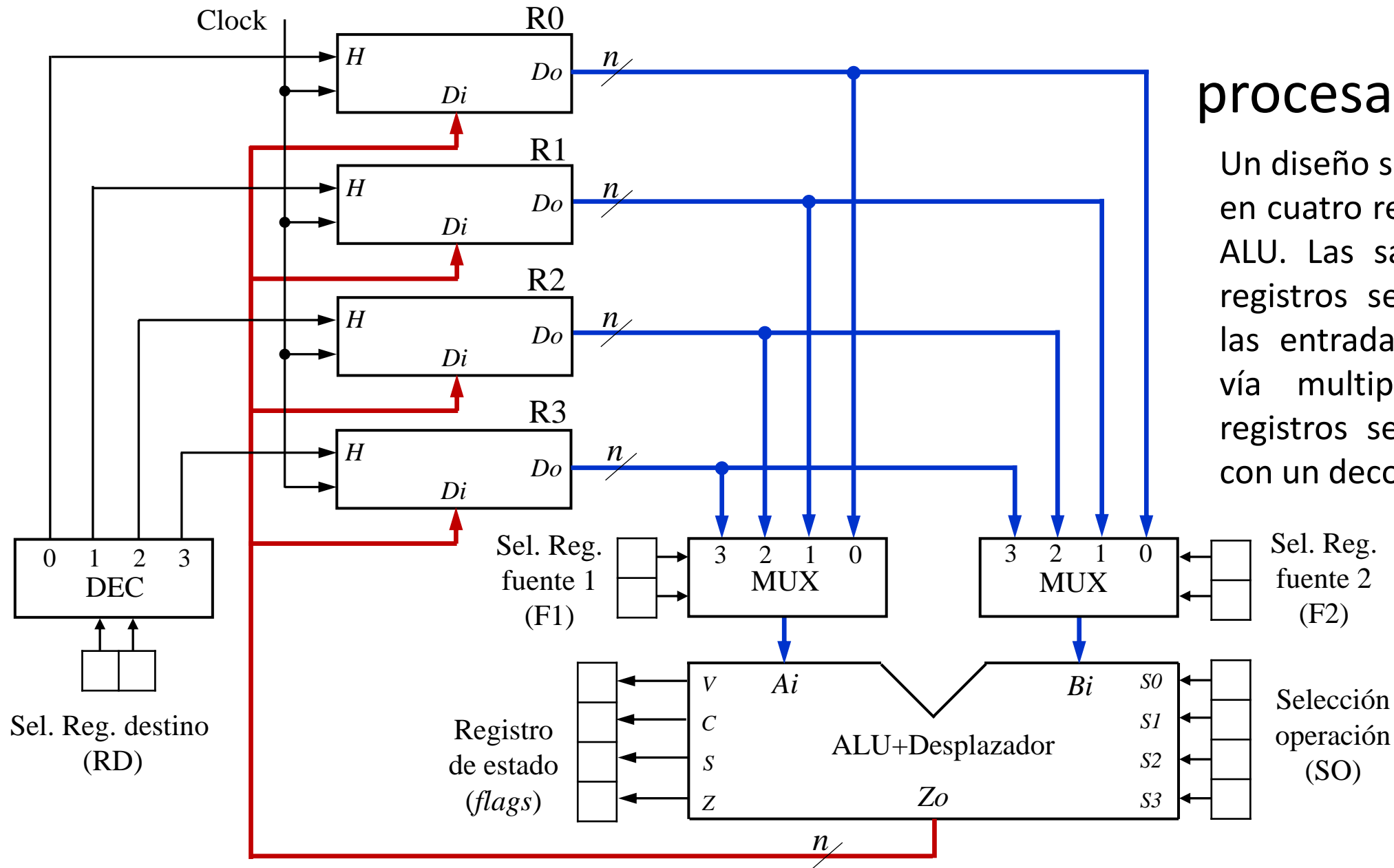
Combinando las entradas de selección de ambos módulos y recodificando, se puede obtener una unidad funcional que realice todas las operaciones de acuerdo a la siguiente tabla:

$S_3$	$S_2$	$S_1$	$S_0$	Micro operación	Descripción
0	0	0	0	$A=0$	Puesta a cero de A
0	0	0	1	$A+1$	Incremento de A
0	0	1	0	$A-1$	Decremento de A
0	0	1	1	$A+B$	Suma aritmética de A y B
0	1	0	0	$A+B+C_i$	Suma aritmética de A y B y el acarreo de entrada
0	1	0	1	$A+B'$	Suma aritmética de A y el complemento de B (A menos B+1)
0	1	1	0	$A+B'+1$	Suma aritmética de A y el complemento de B más 1 (A menos B)
0	1	1	1	A	Transfiere A (de entrada a salida sin cambio)
1	0	0	0	$A'$	Complemento de A
1	0	0	1	A and B	Producto lógico de A y B
1	0	1	0	A or B	Suma lógica de A y B
1	0	1	1	A oexc B	Or exclusiva de A y B
1	1	0	0	sl A	Desplazamiento aritmético a la izquierda de A ( $A \times 2$ )
1	1	0	1	sr A	Desplazamiento aritmético a la derecha de A ( $A / 2$ )
1	1	1	0	rl A	Rotación a la izquierda a través del acarreo de A
1	1	1	1	rr A	Rotación a la derecha a través del acarreo de A

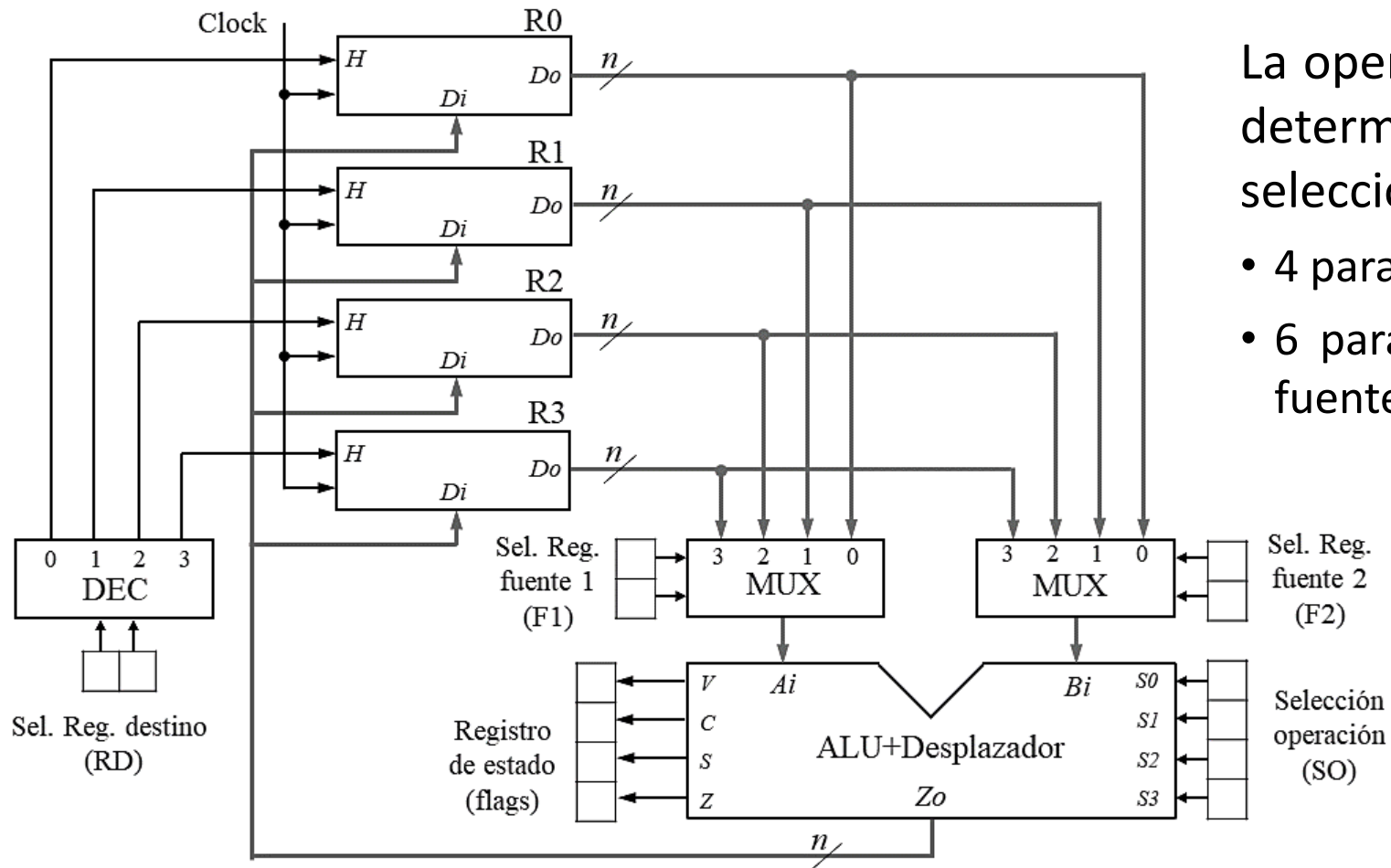


# Unidad procesadora (1)

Un diseño simple basado en cuatro registros y una ALU. Las salidas de los registros se conectan a las entradas de la ALU vía multiplexores. Los registros se seleccionan con un decodificador.



# Unidad procesadora (1)

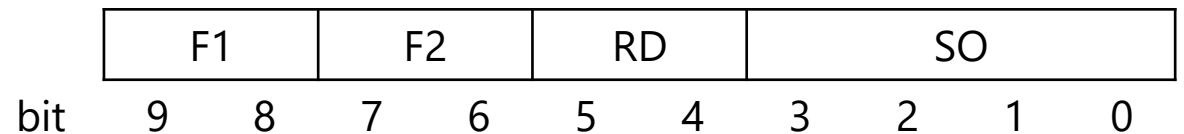


La operación que realiza el conjunto se determina a partir de las entradas de selección de los elementos:

- 4 para seleccionar la operación de la ALU.
- 6 para la selección de los dos registros fuente y el registro destino:

- 2 del registro fuente 1 (MUX1)
- 2 del registro fuente 2 (MUX2)
- 2 del registro destino (DEC)

El conjunto de entradas de selección forma la Palabra de Control:





# Unidad procesadora (1)

Ejemplos de microoperaciones:

1.  $R3 \leftarrow R1 + R2$

Se seleccionan fuentes, destinos, y operación de la ALU:

0	1	1	0	1	1	0	0	1	1
F1		F2		RD		SO			

2.  $R1 \leftarrow R0$

El registro fuente 2 no interviene  $\rightarrow$  es irrelevante (X):

0	0	X	X	0	1	0	1	1	1
F1		F2		RD		SO			

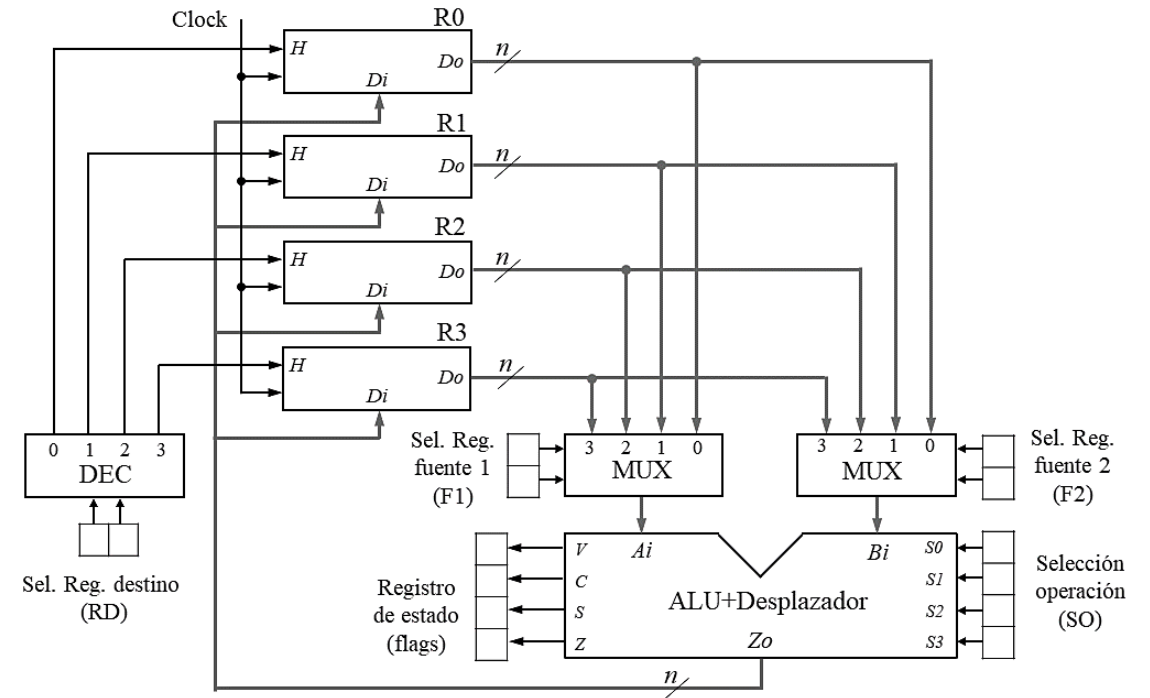
3.  $R0 \leftarrow sl R3$

Se desplaza R3 y se guarda en R0

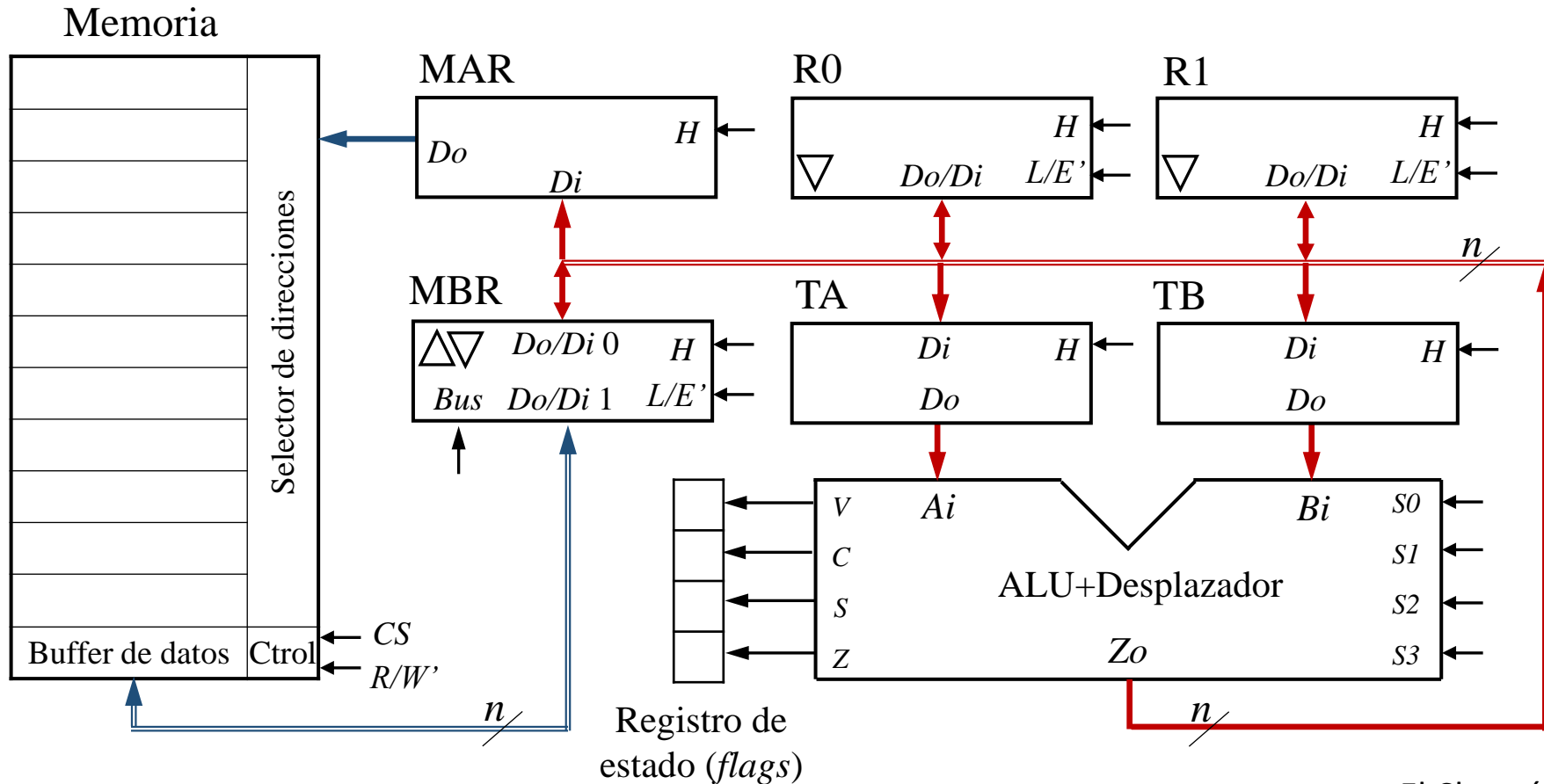
1	1	X	X	0	0	1	1	0	0
F1		F2		RD		SO			

R3 debe entrar como Fuente 1, ya que la operación de desplazamiento se realiza sobre le operando A de la ALU. La siguiente opción es incorrecta:

X	X	1	1	0	0	1	1	0	0
F1		F2		RD		SO			



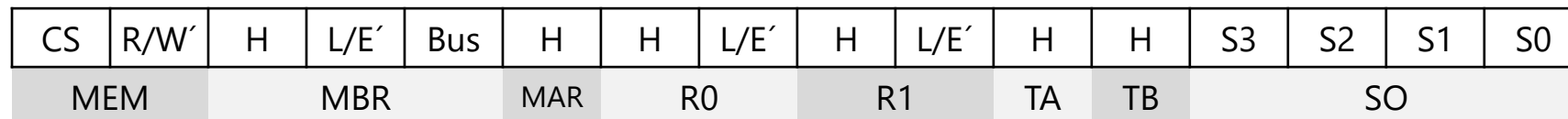
# Unidad procesadora (2)



Un diseño basado en buses, que utiliza:

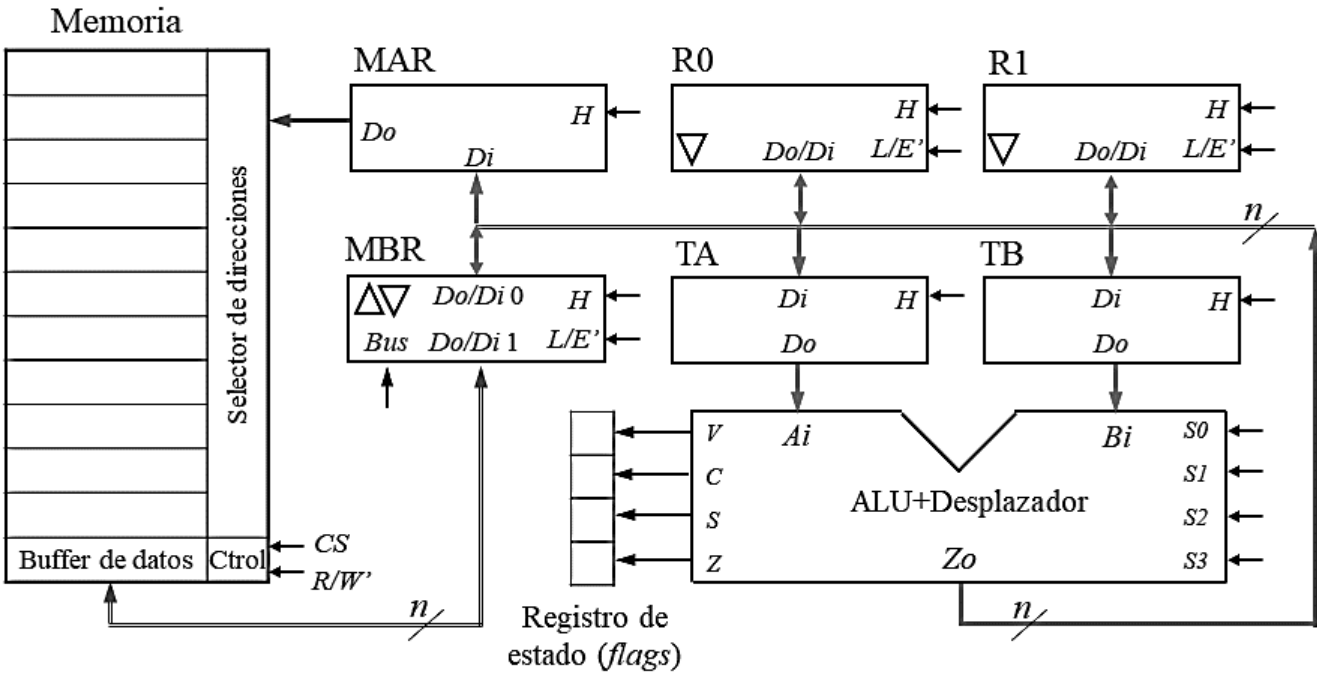
- Dos registros temporales (TA y TB) que almacenan los datos que va a operar la ALU;
- Dos registros de uso general, R0 y R1;
- Una unidad de memoria (= varios registros de uso general) comunicada vía los registros MAR (direcciones), y MBR (datos)

El Ck está conectado a todos los registros

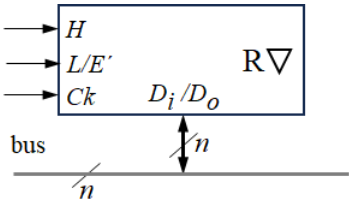


← Palabra de control

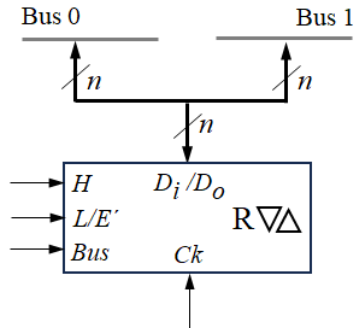
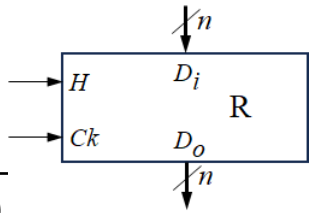
# Unidad procesadora (2)



$H$	$L/E'$	Operación
1	0	Escritura desde el bus
1	1	Lectura en el bus
0	X	Sin cambios



$H$	$Do$	Operación
0	$Do$	Mantiene el valor (lectura)
1	$Di$	Carga el valor de entrada (escritura)



$H$	$Bus$	$L/E'$	Operación
1	0	0	Escritura desde el bus 0
1	0	1	Lectura en el bus 0
1	1	0	Escritura desde el bus 1
1	1	1	Lectura en el bus 1
0	X	X	Sin cambios

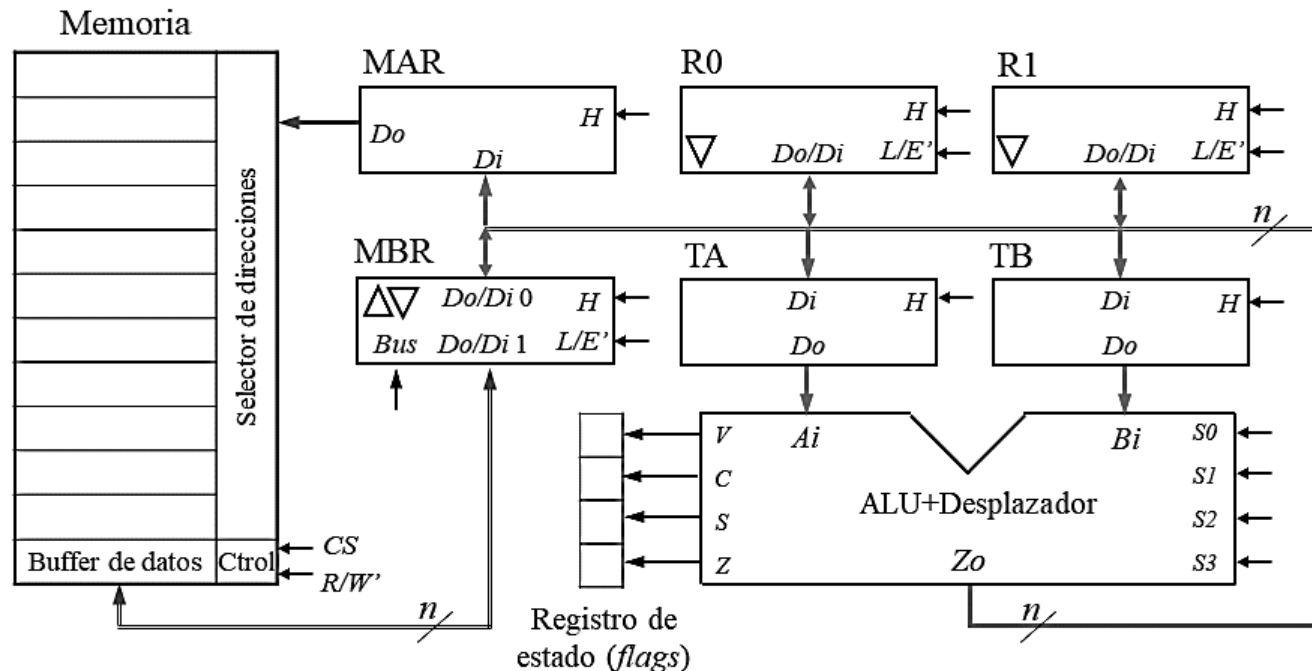
# Unidad procesadora (2)

## Ejemplos de operaciones internas de la UP

### 1. Cargar el contenido de R0 en R1

0	X	0	X	X	0	1	1	1	0	0	0	X	X	X	X
MEM		MBR			MAR	R0		R1	TA	TB					SO

La transferencia es directa a través del bus → el resto de los registros y la ALU no intervienen.  $R1 \leftarrow R0$



En este caso, la operación entre registros es directa y puede realizarse con una sola microoperación.

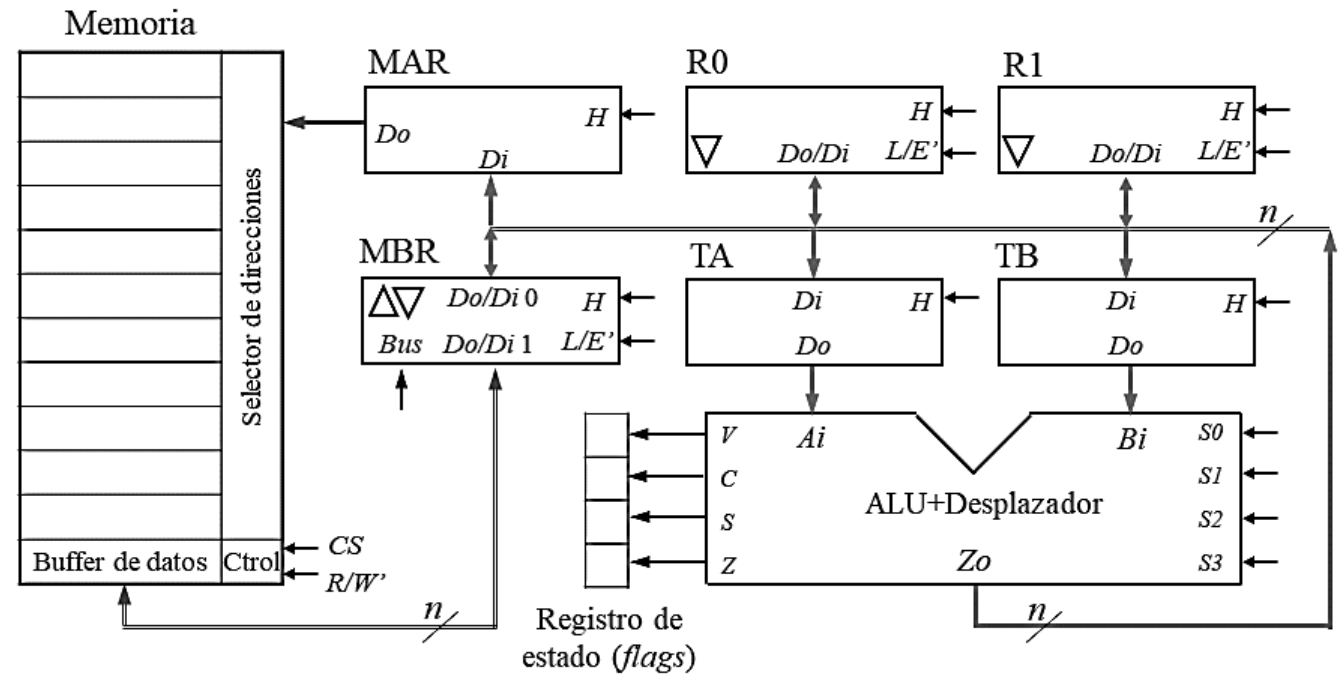
Sin embargo, existen operaciones que deben descomponerse en varias microoperaciones.

# Unidad procesadora (2)

2. Desplazar a la derecha R1 y guardarlo en R0

Es necesario

- Cargar R1 a TA (no TB; la operación sr se realiza sobre el operando A de la ALU).
- Realizar el desplazamiento y almacenar en R0



t1: TA ← R1

0	X	0	X	X	0	0	X	1	1	1	0	X	X	X	X
MEM		MBR			MAR		R0	R1	TA	TB				SO	

t2: R0 ← sr TA

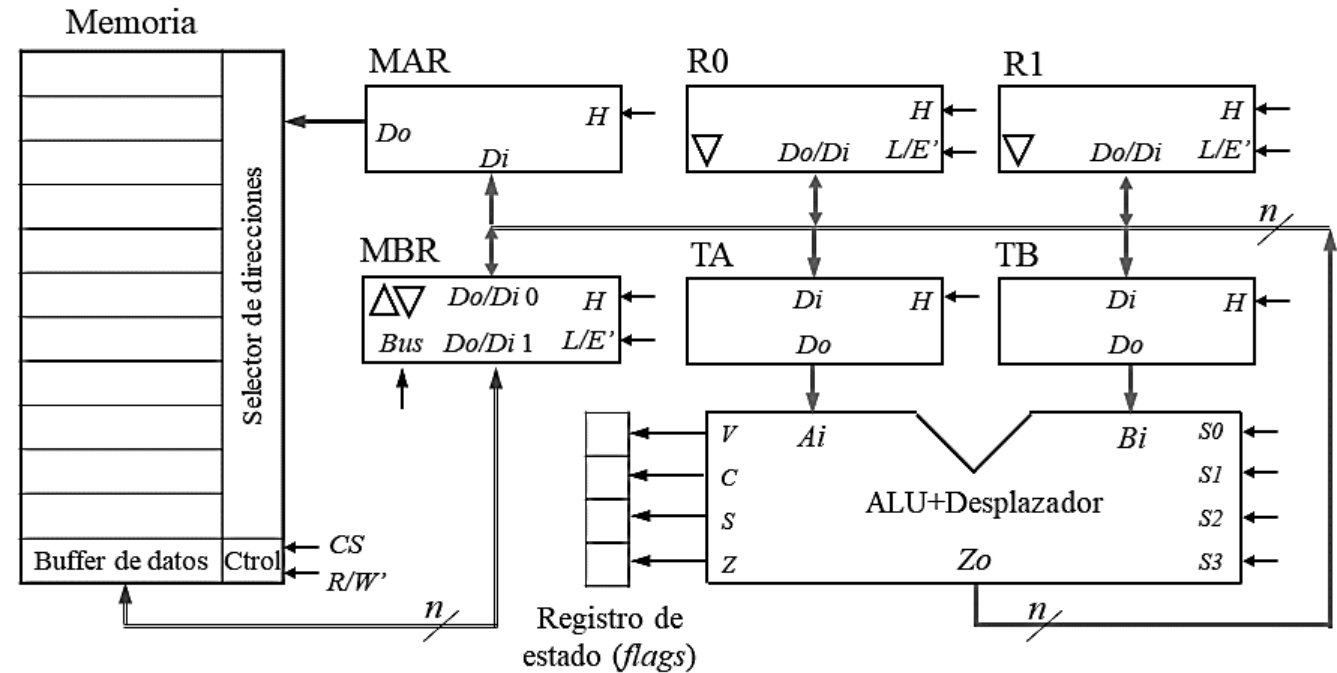
0	X	0	X	X	0	1	0	0	X	0	0	1	1	0	1
MEM		MBR			MAR		R0	R1	TA	TB				SO	

# Unidad procesadora (2)

3. Sumar R0 y R1 y guardar el resultado en R1

Es necesario

- Cargar R0 a TA;
- Cargar R1 a TB;
- Realizar la suma y almacenar en R1



t1: TA ← R0

0	X	0	X	X	0	1	1	0	X	1	0	X	X	X	X
MEM		MBR			MAR	R0		R1	TA	TB					SO

t2: TB ← R1

0	X	0	X	X	0	0	X	1	1	0	1	X	X	X	X
MEM		MBR			MAR	R0		R1	TA	TB					SO

t3: R1 ← TA + TB

0	X	0	X	X	0	0	X	1	0	0	0	0	0	1	1
MEM		MBR			MAR	R0		R1	TA	TB					SO

# Unidad procesadora (2)

4. Restar el contenido de la posición de memoria cuya dirección está en R0 con el contenido de R1 y guardar el resultado en R1:

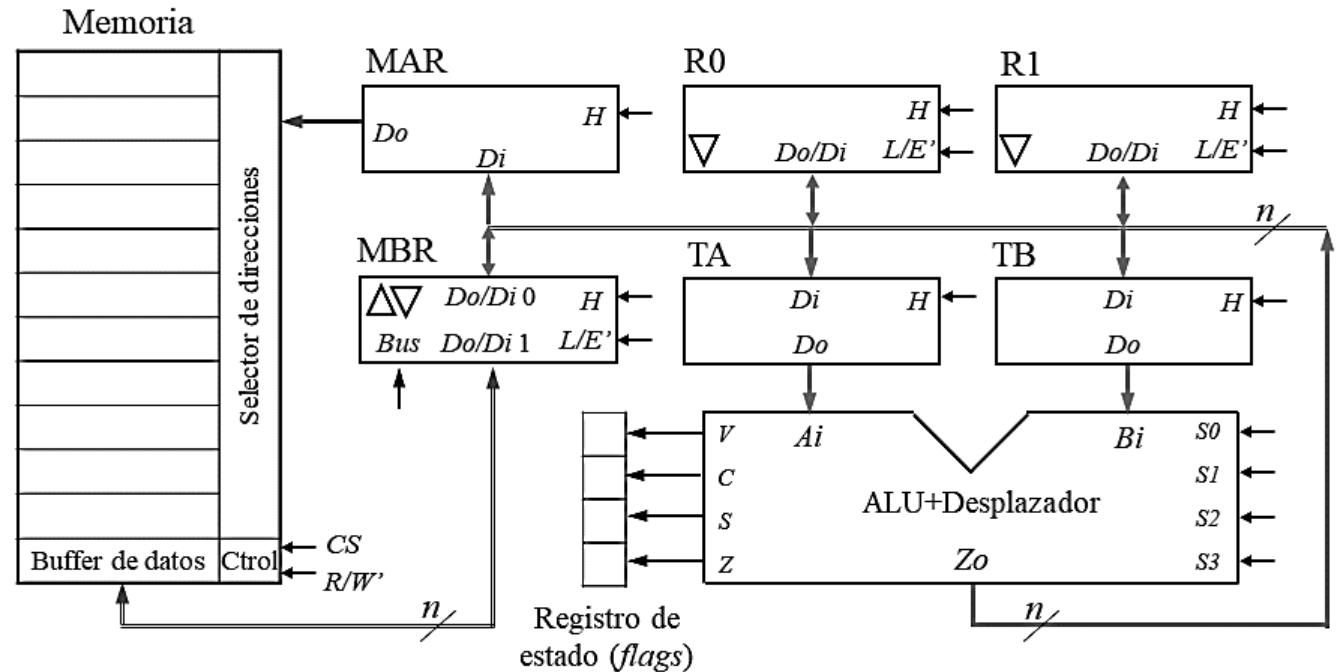
t1:  $MAR \leftarrow R0$

t2:  $MBR \leftarrow M[MAR], TB \leftarrow R1$

Como  $MBR \leftarrow M[MAR]$  y  $TB \leftarrow R1$  se realizan sobre buses distintos, pueden ocurrir simultáneamente

t3:  $TA \leftarrow MBR$

t4:  $R1 \leftarrow TA - TB$



0	X	0	X	X	1	1	1	0	X	0	0	X	X	X	X
MEM		MBR			MAR		R0		R1	TA	TB			SO	
1	1	1	0	1	0	0	X	1	1	0	1	X	X	X	X
MEM		MBR			MAR		R0		R1	TA	TB			SO	
0	X	1	1	0	0	0	X	0	X	1	0	X	X	X	X
MEM		MBR			MAR		R0		R1	TA	TB			SO	

# Repaso conceptual

1. ¿Qué es un registro paralelo? En los ejemplos vistos, ¿qué función cumplen las líneas de control *H* y *L/E'*?
2. Desde el punto de vista lógico, ¿qué puede considerarse una memoria RAM? ¿Qué es la *capacidad* de la memoria y cómo se calcula?
3. ¿Qué función cumplen las líneas de dirección y las líneas de datos en una RAM? Enumere sus principales líneas de control.
4. ¿Qué es una microinstrucción? ¿Qué tipos existen?
5. ¿Cómo se representan las microinstrucciones mediante el RTL? De ejemplos para cada tipo de microinstrucción.
6. ¿Cuál es el concepto de *palabra de control*? De ejemplos de ellas en las Unidades de Procesamiento vistas



# Lecturas recomendadas

- Mano M., Kime, C. - Fundamentos de diseño lógico y de computadoras - 3º Ed. - Prentice Hall. Año 2000.  
→ *Capítulos 5 – 7 – 9 – 10*