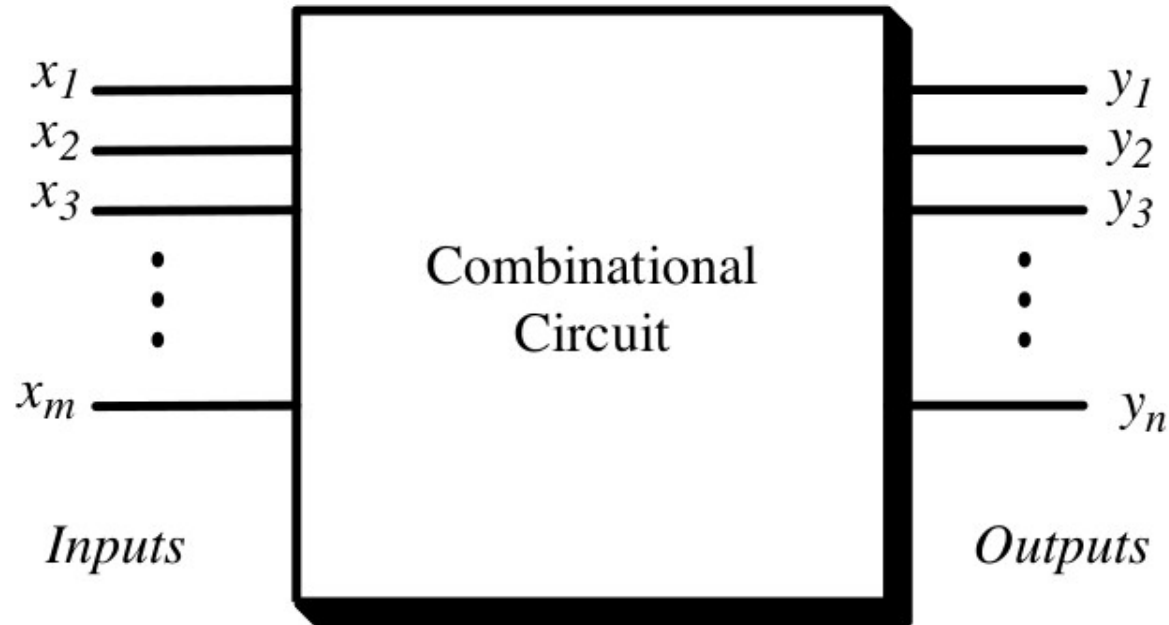




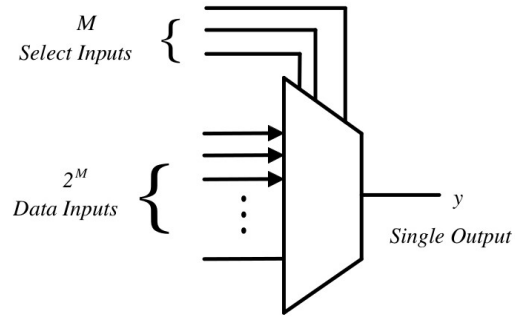
# Sistemas secuenciales

# Sistemas combinacionales

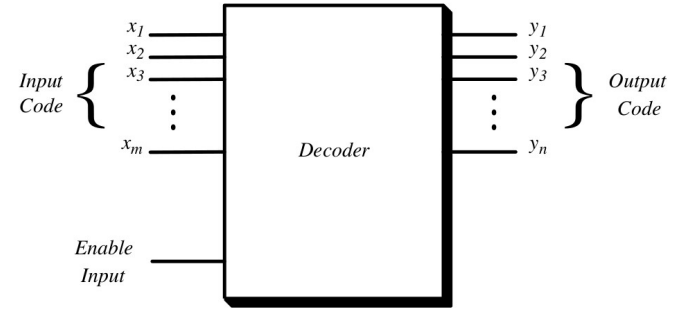


En este tipo de sistemas, las salidas en cada instante son función exclusiva de las entradas en el mismo instante.

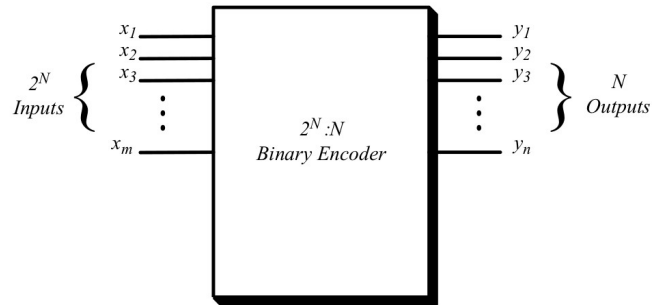
# Sistemas combinacionales: ejemplos



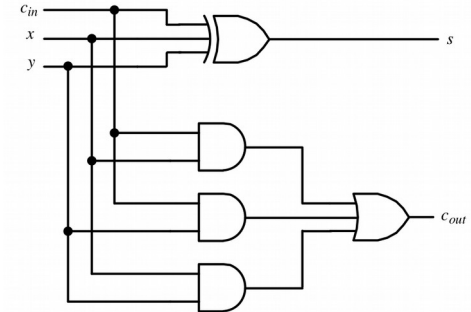
Multiplexer



Decoder



Encoder

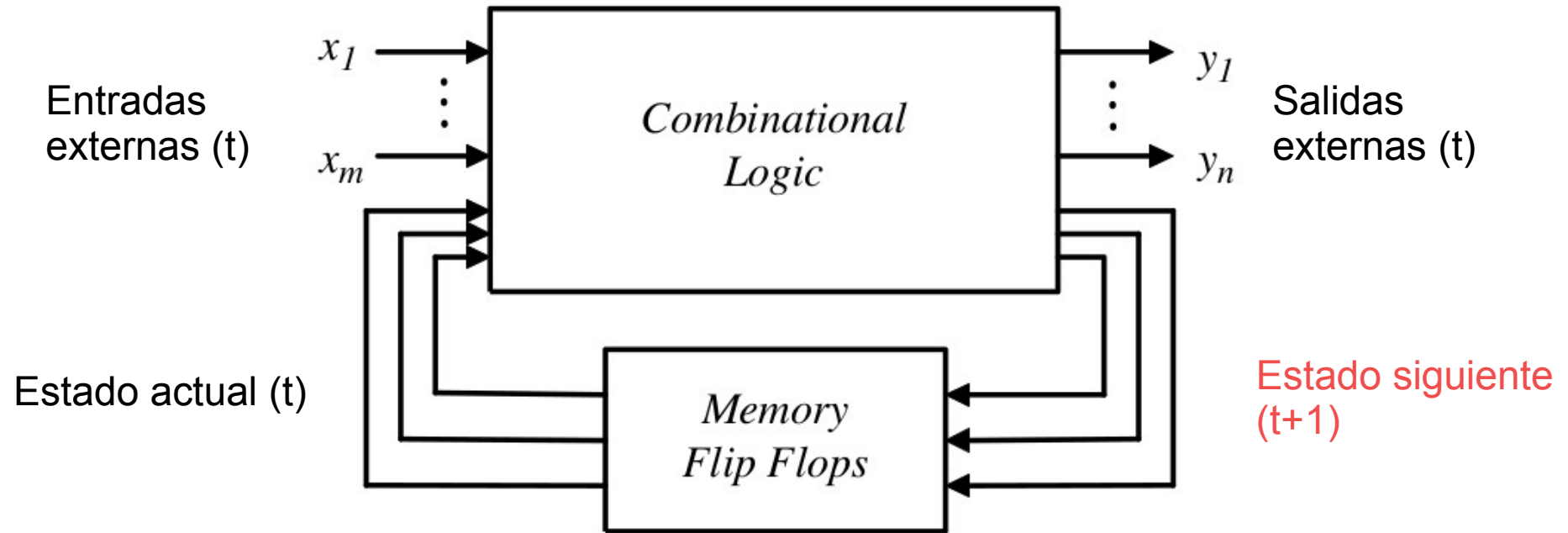


1 Bit full adder

# Sistemas secuenciales

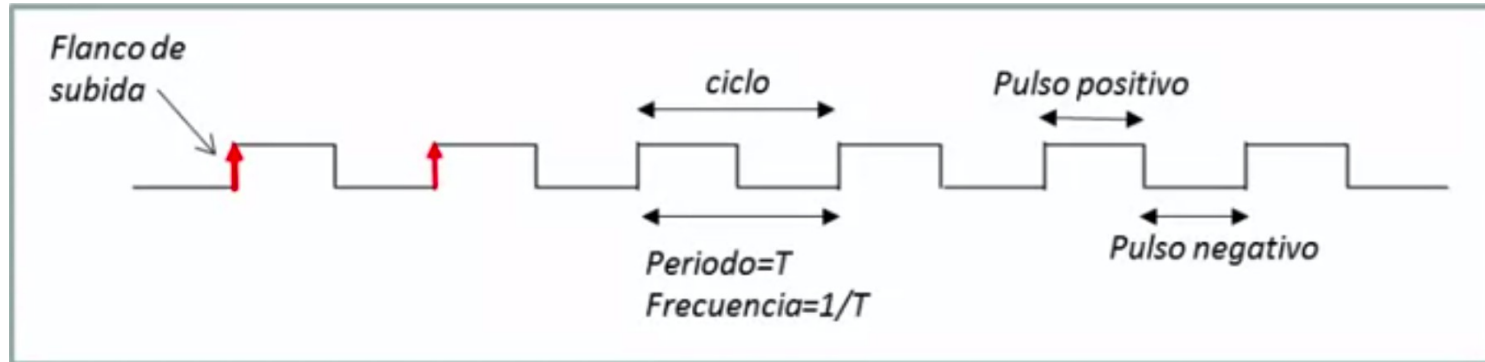
- Existen un conjunto de problemas que no pueden ser resueltos mediante sistemas combinacionales.
- Ejemplo: la realización de la lógica de una máquina de venta de bebidas, que recibe el monto de la venta a través de la inserción de monedas.
- Es obvio que la salida de la máquina (en este caso, habilitar el expendio de café, por ejemplo) está formado por una secuencia de inserción de monedas que lleguen (o sobrepasen), en las diferentes combinaciones, al monto exigido para la bebida.
- Este tipo de sistemas se llaman *secuenciales*

# Sistemas secuenciales: esquema

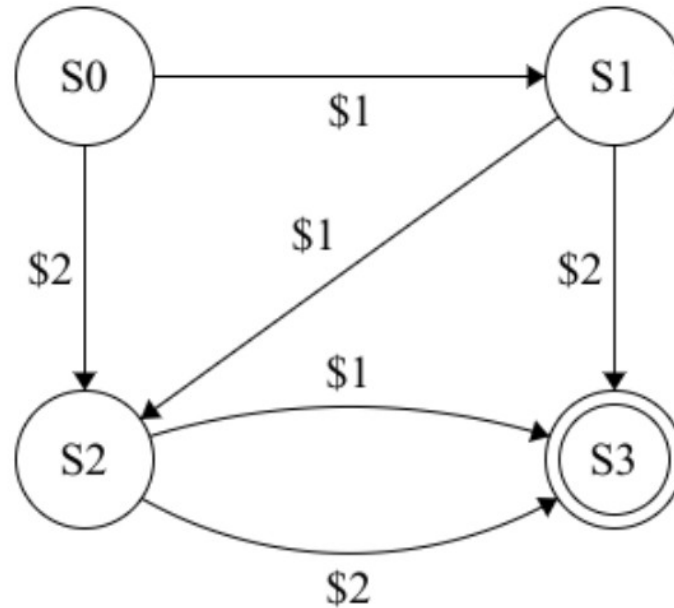


# Sistemas secuenciales: ¿qué se entiende por $t$ ó por $t+1$ ?

- Por  $t$  ó por  $t + 1$  se entiende instantes de tiempo predeterminados en los cuales se toman las muestras del estado del sistema y se producen las acciones.
- Es común (pero no necesariamente obligatorio) que exista un *reloj* o *clock* continuo y regular que es quien dé el ritmo adecuado en forma centralizada.

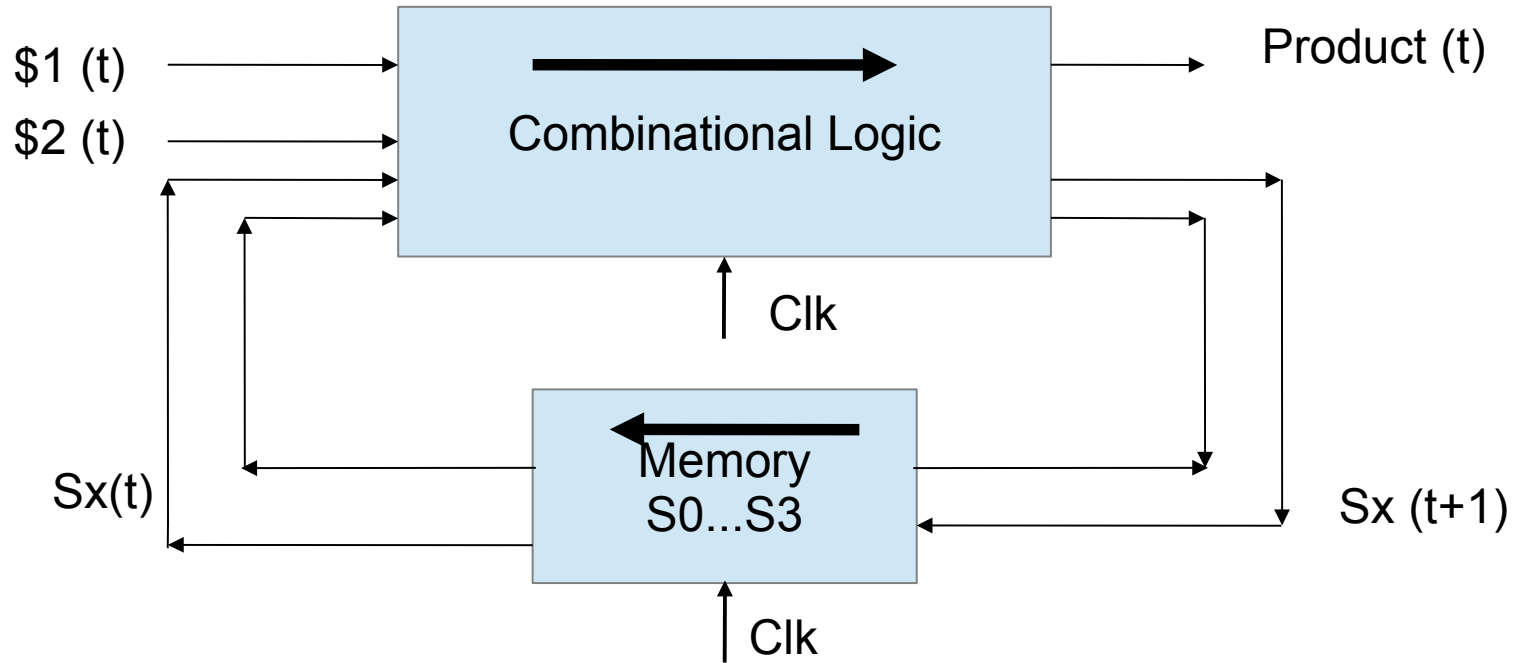


# Sistemas secuenciales: ejemplo



Maquina de venta: precio \$3 sin cambio:  $S0 \rightarrow$  comienzo,  $S3 \rightarrow$  expendio

# Sistemas secuenciales: esquema maquina venta



Memory: pequeño banco de memoria que permite cuatro estados distintos

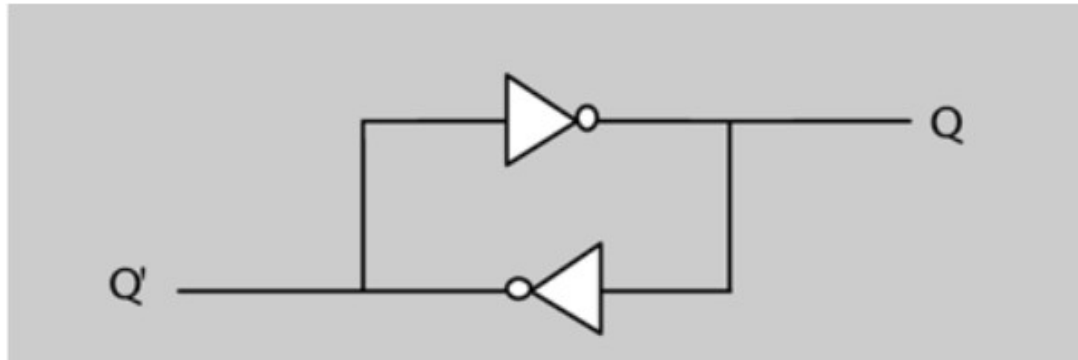


# Sistemas secuenciales

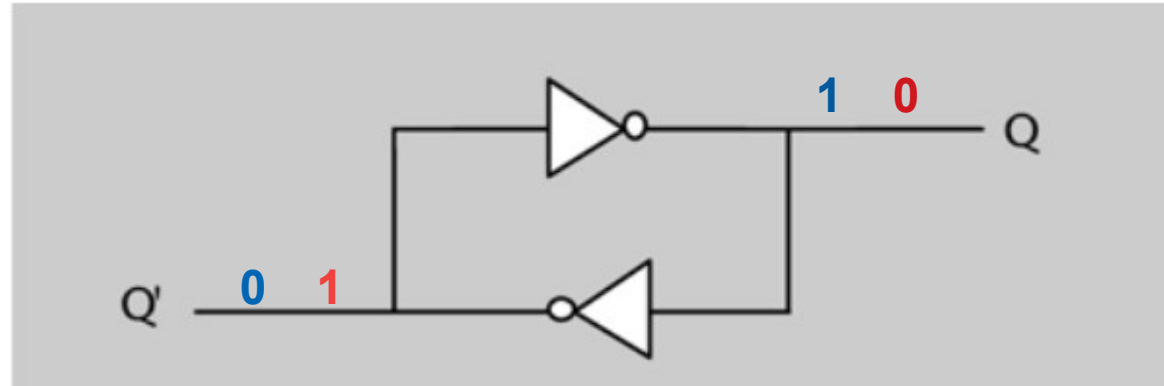
*Latches y Flip-Flops*

# Sistemas secuenciales: circuitos de memoria

- Para realizar un sistema secuencial nos hace falta disponer de circuitos que memoricen estados.
- Para ello, comenzaremos por un circuito sencillo y básico denominado *elemento biestable*:



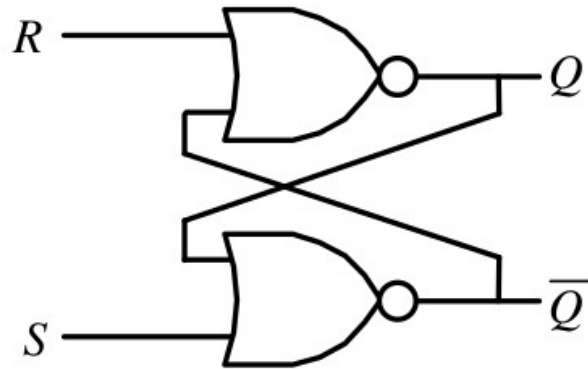
# Sistemas secuenciales: análisis elemento biestable



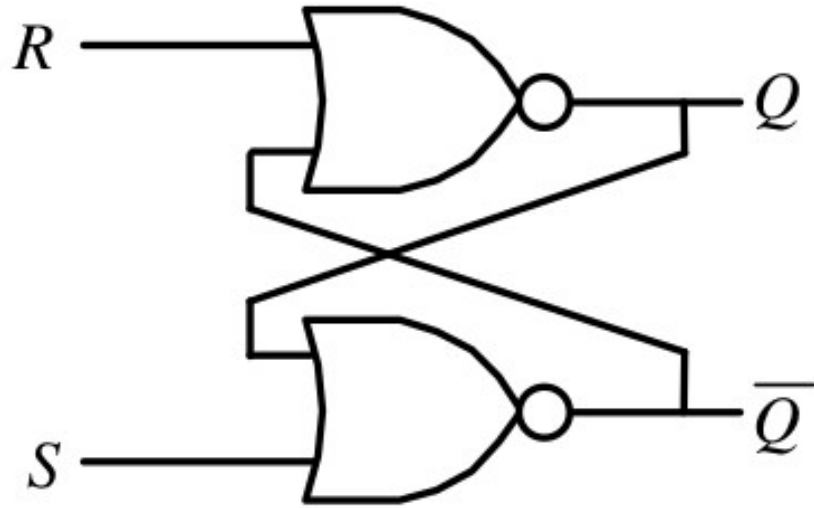
# Sistemas secuenciales: del elemento biestable al *SR latch*

Sin embargo, si bien el *elemento biestable* cumple con la función de memorizar, desde el punto de vista práctico, es muy difícil de comandar

Por ello, se lo transforma al circuito con compuertas NOR denominado *SR Latch*



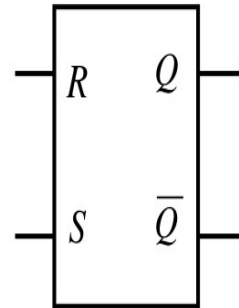
# Sistemas secuenciales: SR-Latch



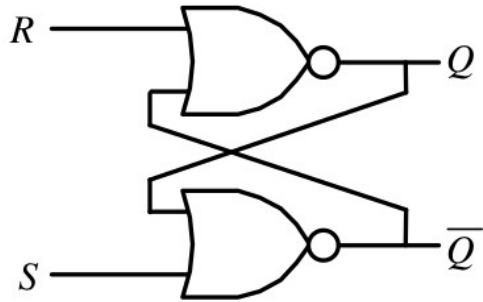
*Truth Table*

$S$	$R$	$Q$	$\overline{Q}$	<i>Comments</i>
0	0	1/0	0/1	<i>No change</i>
0	1	0	1	<i>Reset</i>
1	0	1	0	<i>Set</i>
1	1	0	0	<i>Not used</i>

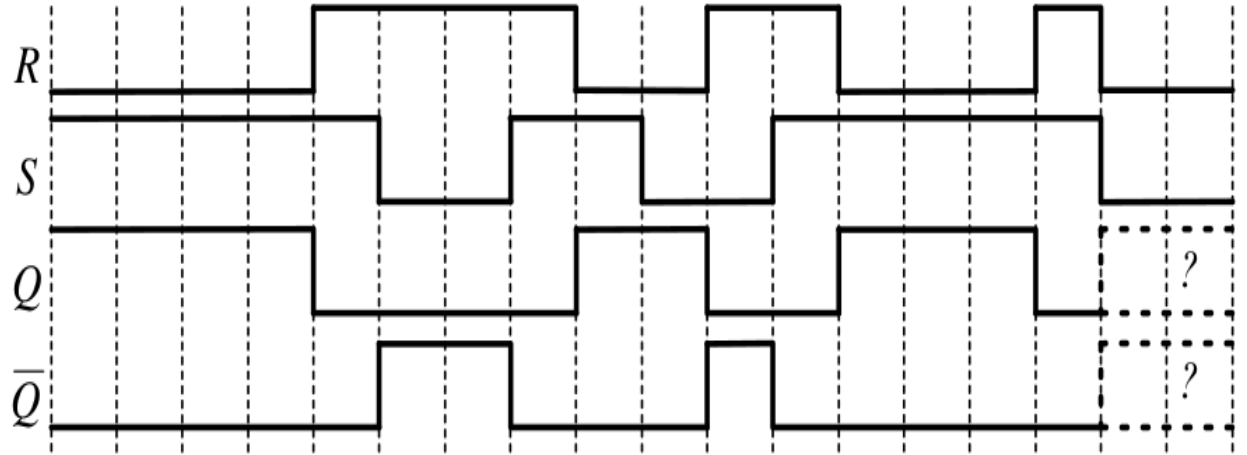
*Graphic Symbol*



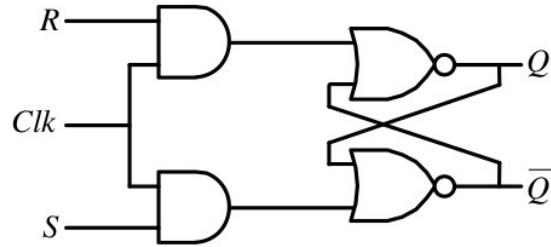
# Sistemas secuenciales: diagrama de tiempo SR Latch



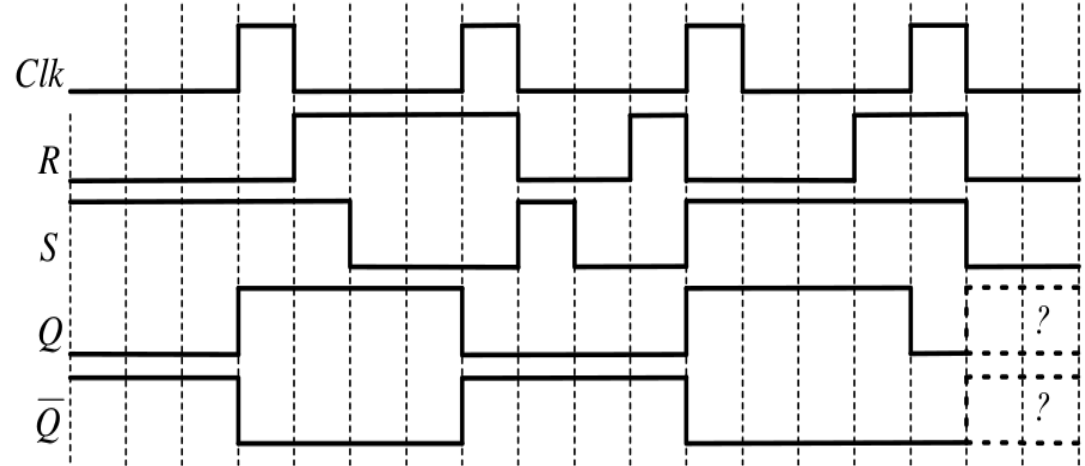
$R$	$S$	$Q_{n+1}$
0	0	$Q_n$
0	1	1
1	0	0
1	1	-



# Sistemas secuenciales: diagrama de tiempo Gated SR Latch



<i>Clk</i>	<i>S</i>	<i>R</i>	<i>Q</i>	$\bar{Q}$	<i>Comments</i>
1	0	0	1/0	0/1	No change
1	0	1	1	0	Set
1	1	0	0	1	Reset
1	1	1	1	1	Not used
0	x	x	1/0	0/1	Disabled



Error en tabla de verdad: 1 1 1  $\rightarrow$  0 0

# Sistemas secuenciales: *latches* vs *flip-flops*

- Se denomina *latch* a los elementos de memoria en los cuales la salida cambia en cuanto las entradas cambian; dos ejemplos de ello son el *RS-Latch* y el *Gated RS-Latch* cuando la entrada de *Clk* es por nivel
- Al contrario, se denomina *flip-flop* cuando la salida cambia sólo en alguno de los flancos del *Clk*.
- De acuerdo a lo establecido en la figura 5, *esquema de un sistema secuencial*, deben establecerse claramente cuales son los tiempos en los cuales el sistema reacciona; ello no podría hacerse si las señales de entrada están habilitadas por tiempos muy grandes

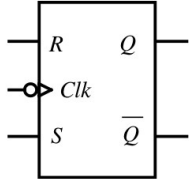


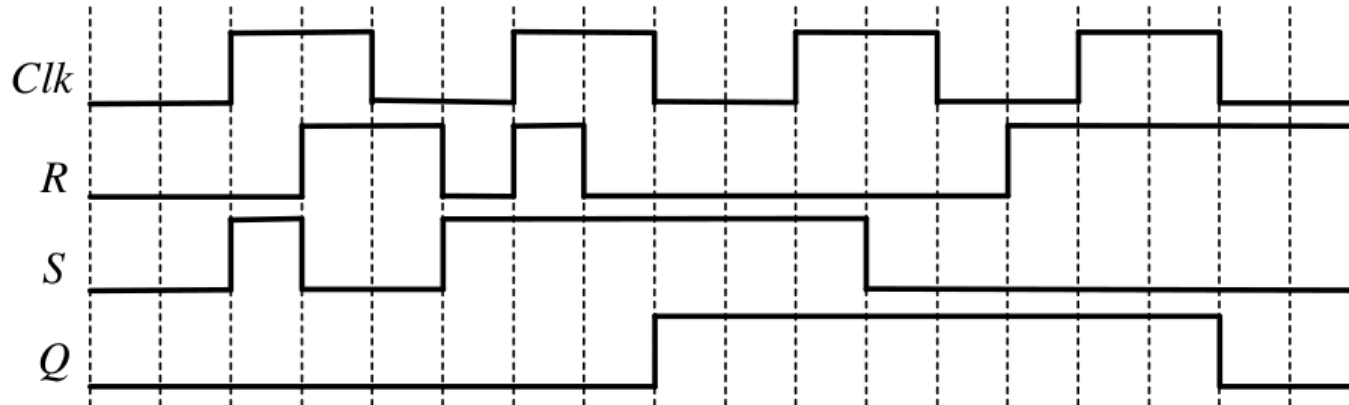
# Sistemas secuenciales: asincrónicos y sincrónicos

- También, los sistemas en los cuales las señales están validadas en los flancos de *Clk* se denominan *sincrónicos*, mientras que los otros se denominan *asincrónicos*.
- Los sistemas *asincrónicos* son más veloces que los *sincrónicos* pero, sin embargo, estos últimos son preferidos en casi todos los sistemas secuenciales por el grado de seguridad de funcionamiento que muestran
- En lo que sigue, sólo se tratará con *flip-flops* o sea con sistemas sincrónicos

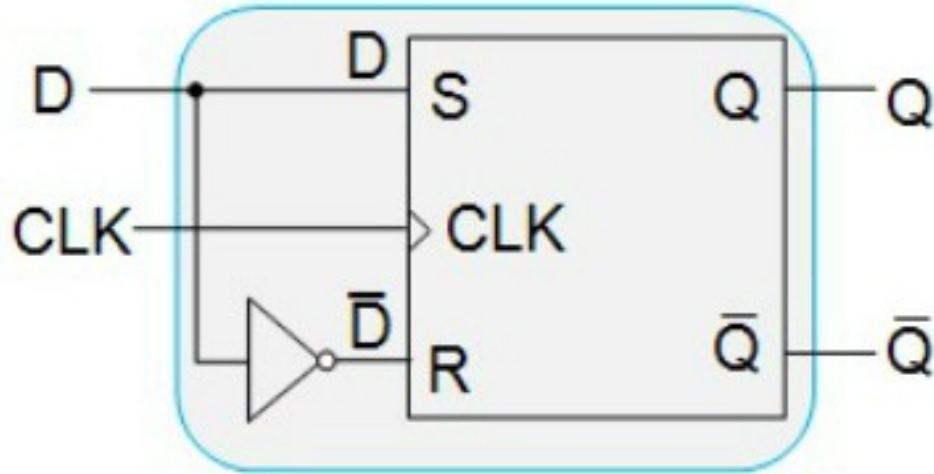
# Sistemas secuenciales: *flip-flop*

## *SR*

Truth Table				Graphic Symbol	
<i>Clk</i>	<i>S</i>	<i>R</i>	<i>Q(t+1)</i>		
↓	0	0	<i>Q(t)</i>		
↓	0	1	0 ( <i>Reset</i> )		
↓	1	0	1 ( <i>Set</i> )		
↓	1	1	<i>Undefined</i>		

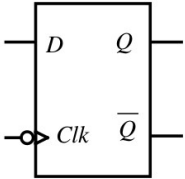


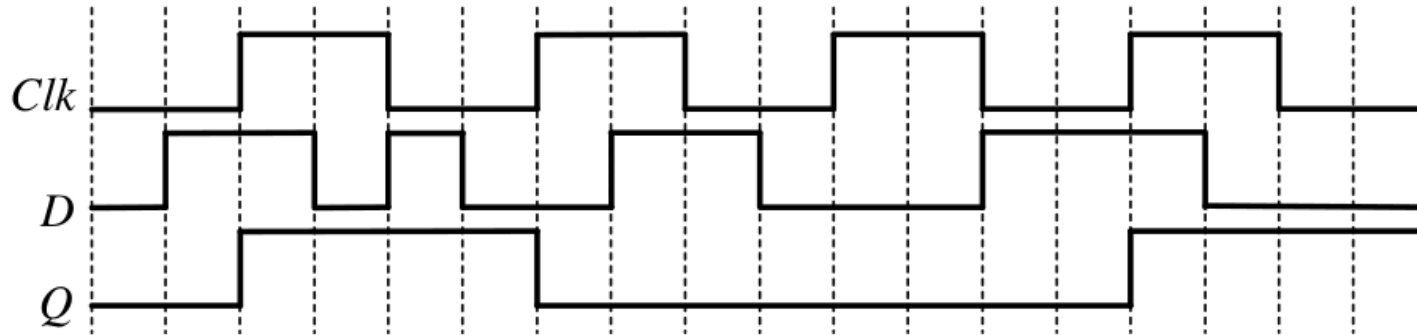
# Sistemas secuenciales: del *flip-flop RS* al *flip-flop D*



Se denomina *flip-flop D* por *Data flip-flop*: la salida Q sigue a la entrada D  
Además, evita la posibilidad de indeterminación dada por  $S = R = 1$

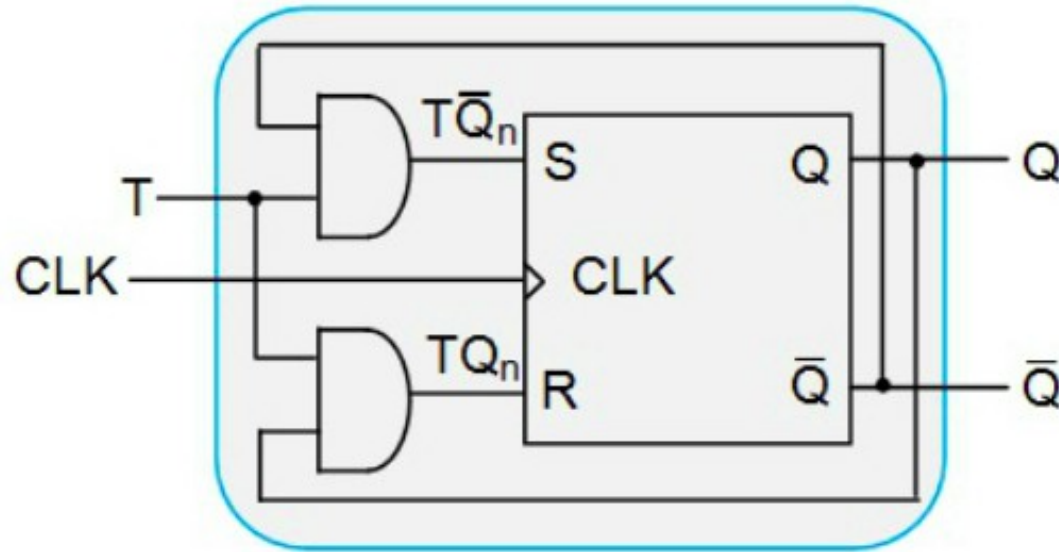
# Sistemas secuenciales: *D flip-flop*

Truth Table				Graphic Symbol	
<i>Clk</i>	<i>D</i>	<i>Q(t)</i>	<i>Q(t+1)</i>		
↓	0	0	0		
↓	0	1	0		
↓	1	0	1		
↓	1	1	1		



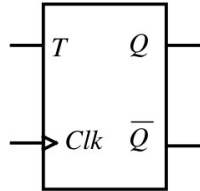
Error: el diagrama corresponde a un flip-flop con clk de flanco positivo

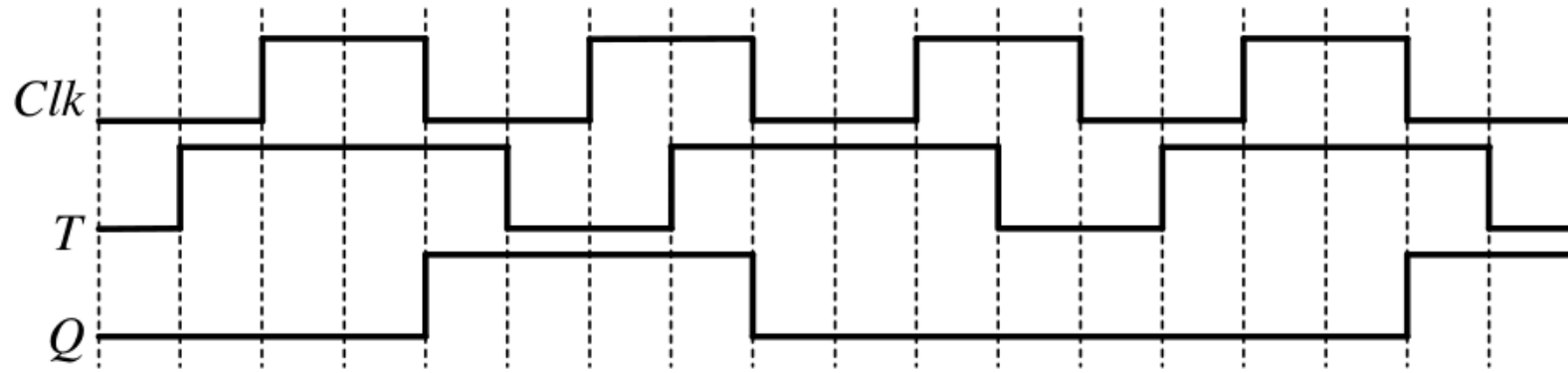
# Sistemas secuenciales: del *flip-flop RS* al *flip-flop T*



Se denomina *flip-flop T* por *Toggle*, ya que si  $T = 1$ , la salida  $Q$  *cambia* con cada Clk. También en este caso se evita la indeterminación dada por  $S = R = 1$ .

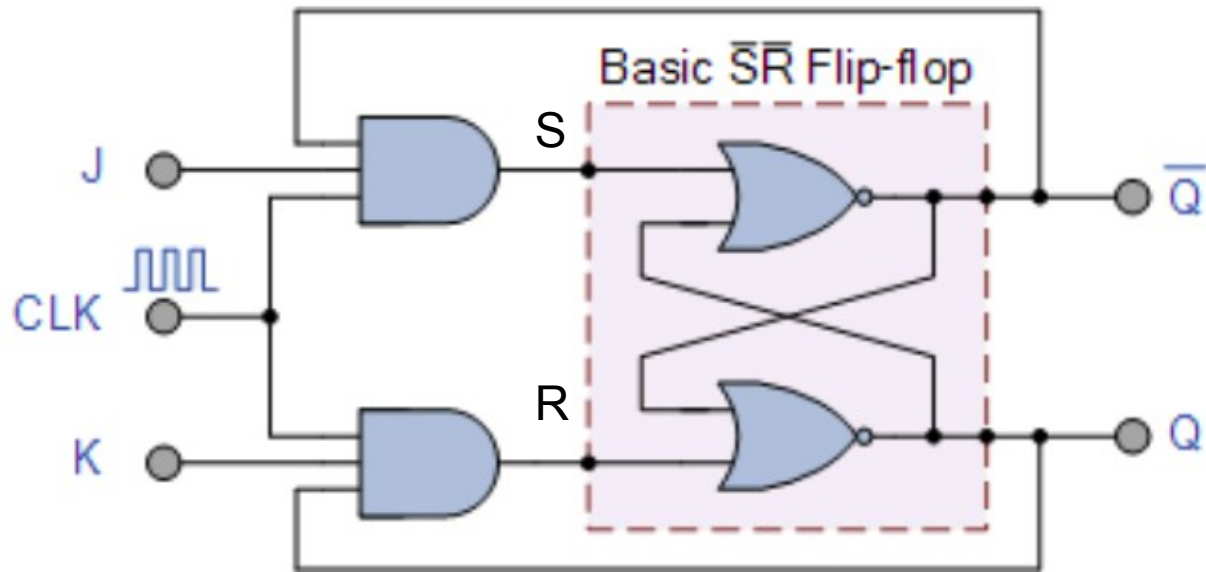
# Sistemas secuenciales: *flip-flop T*

Truth Table			Graphic Symbol	
<i>Clk</i>	<i>T</i>	<i>Q(t+1)</i>		
↑	0	<i>Q(t) (Hold)</i>		
↑	1	<i>Q̄(t) (Toggle)</i>		

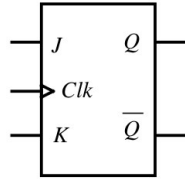


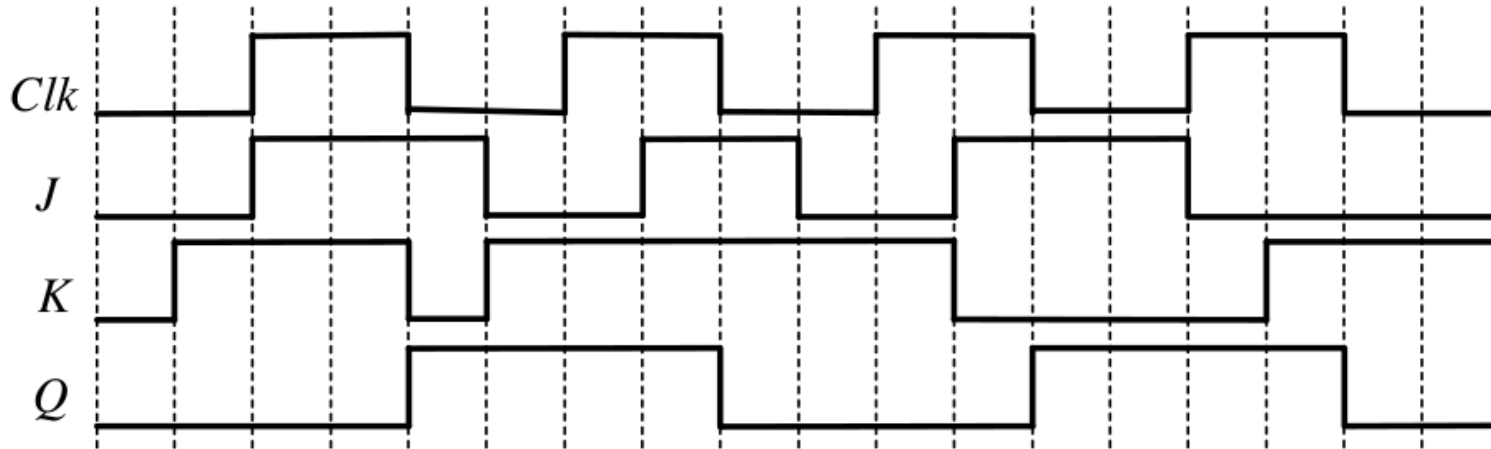
Error: el diagrama de tiempos corresponde a un flip-flop T con Clk flanco negativo

# Sistemas secuenciales: del *flip-flop RS* al *flip-flop JK*



# Sistemas secuenciales: *flip-flop JK*

Truth Table				Graphic Symbol	
<i>Clk</i>	<i>J</i>	<i>K</i>	<i>Q(t+1)</i>		
↑	0	0	<i>Q(t) (Hold)</i>		
↑	0	1	0 ( <i>Reset</i> )		
↑	1	0	1 ( <i>Set</i> )		
↑	1	1	$\overline{Q(t)}$ ( <i>Toggle</i> )		



Error: el diagrama corresponde al de un flip-flop JK con Clk flanco negativo



# Sistemas secuenciales: características del flip-flop JK

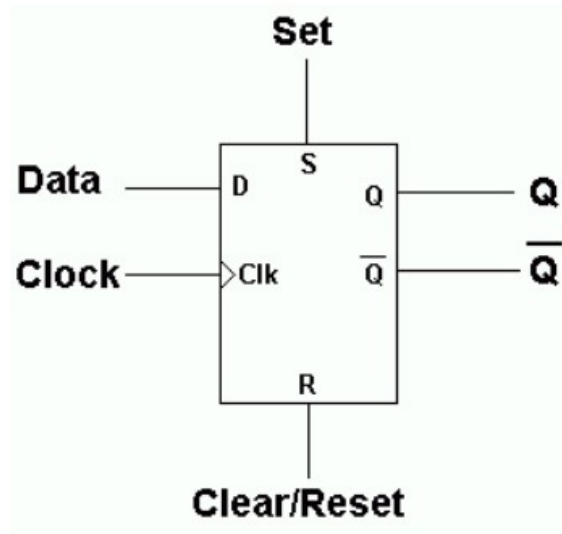
- El *flip-flop JK* es el más versátil y el más comunmente usado cuando se utilizan dispositivos discretos para implementar máquinas de estado finitos.
- Un *flip-flop JK* puede configurarse para trabajar como un flip-flop RS, T o D:
  - Las entradas J y K son equivalentes, respectivamente, a S y R salvo que ambas estén en 1
  - Si las entradas J y K se unen, entonces se transforma en la entrada T de un flip-flop T
  - J será la entrada de un flip-flop D si se mantiene que  $K = \text{not } J$

# Sistemas secuenciales: entradas asíncronas

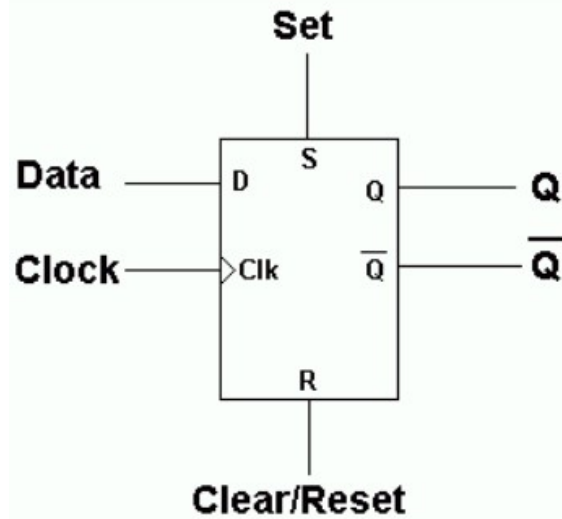
- Como se mostró en la filmina 5, un sistema secuencial mantiene la historia mediante el estado del sistema.
- Este estado se mantiene con un conjunto de *flip-flops* que, en el caso del ejemplo de la máquina de venta, está formado por dos *flip-flops* que da  $2^N$  estados donde  $N = 2$
- Sin embargo, se debe fijar el estado inicial (**S0** en filmina 7) de alguna forma antes de poner a funcionar el sistema.
- Por lo tanto, los *flip-flops* que constituyen la *memoria* de estados deben forzarse por otros medios.

# Sistemas secuenciales: entradas asíncronas (2)

- No mostraremos exactamente el circuito, pero se logra mediante un *latch RS* adicional y por lo tanto, sus entradas se denominan *entradas asincrónicas*: aquí se muestra como sería el símbolo de un *flip-flop D* con entradas asincrónicas adicionales



# Sistemas secuenciales: entradas asíncronas (3)



Inputs				Outputs		Action
D	CLK	SET	RST	Q	QN	
0	$f$	0	0	0	1	Transfer 0 from D to Q
1	$f$	0	0	1	0	Transfer 1 from D to Q
0 or 1	0 or 1	1	0	1	0	Asynchronous Set
0 or 1	0 or 1	0	1	0	1	Asynchronous Reset
0 or 1	0 or 1	1	1	Last Q	Last QN	Illegal concurrent SET and RST

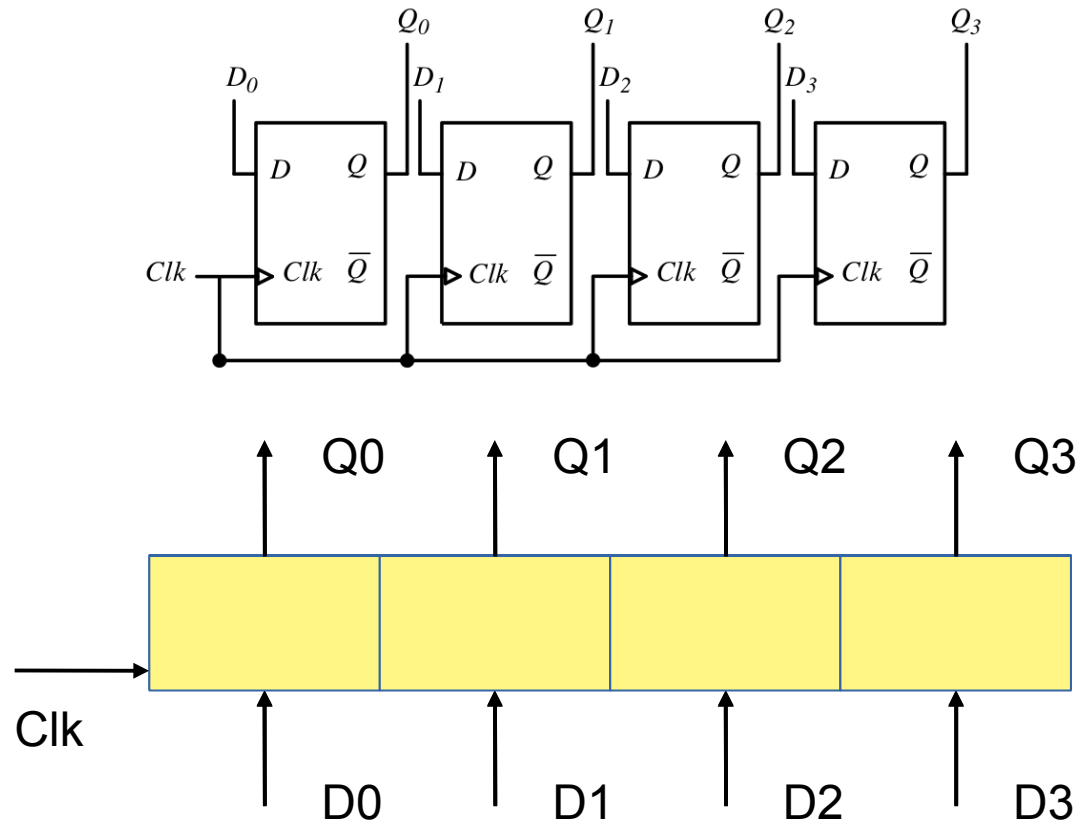
# Sistemas sincrónicos

## Registros

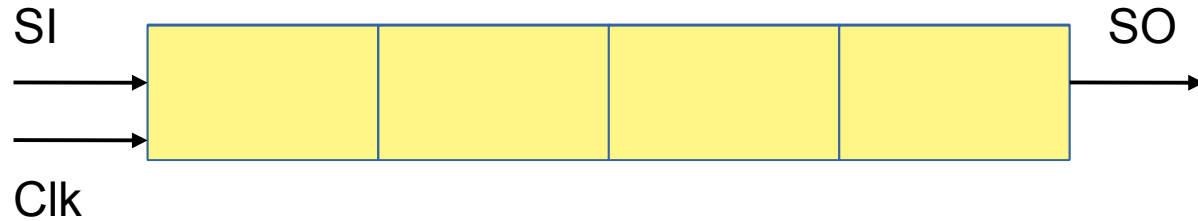
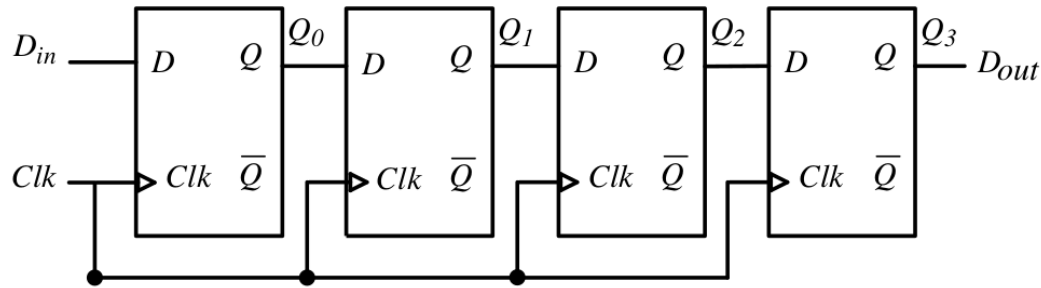
# Sistemas sincrónicos: tipos de registros

- Los registros se pueden clasificar de acuerdo a:
  - Registro de entrada paralelo y salida paralelo (**PIPO**)
  - Registros de desplazamiento:
    - Entrada serie y salida serie (**SISO**)
    - Entrada serie y salida paralelo (**SIPO**)
    - Entrada paralelo y salida serie (**PISO**)

# Sistemas sincrónicos: **PIPO**

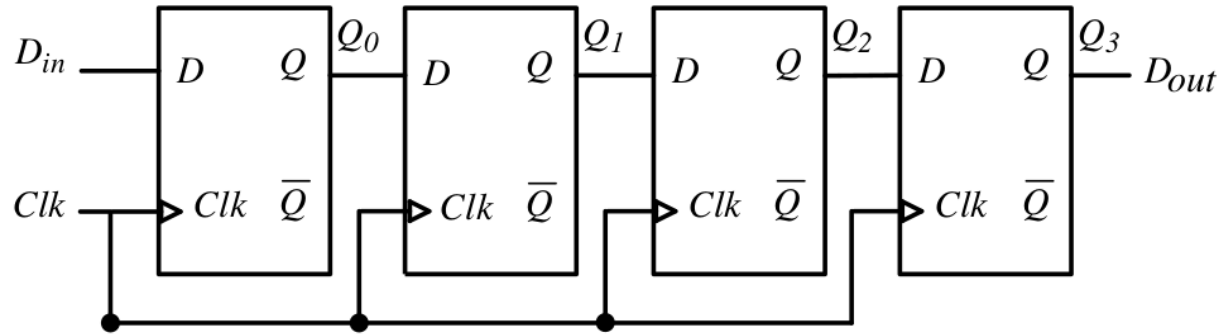


# Sistemas sincrónicos: SISO



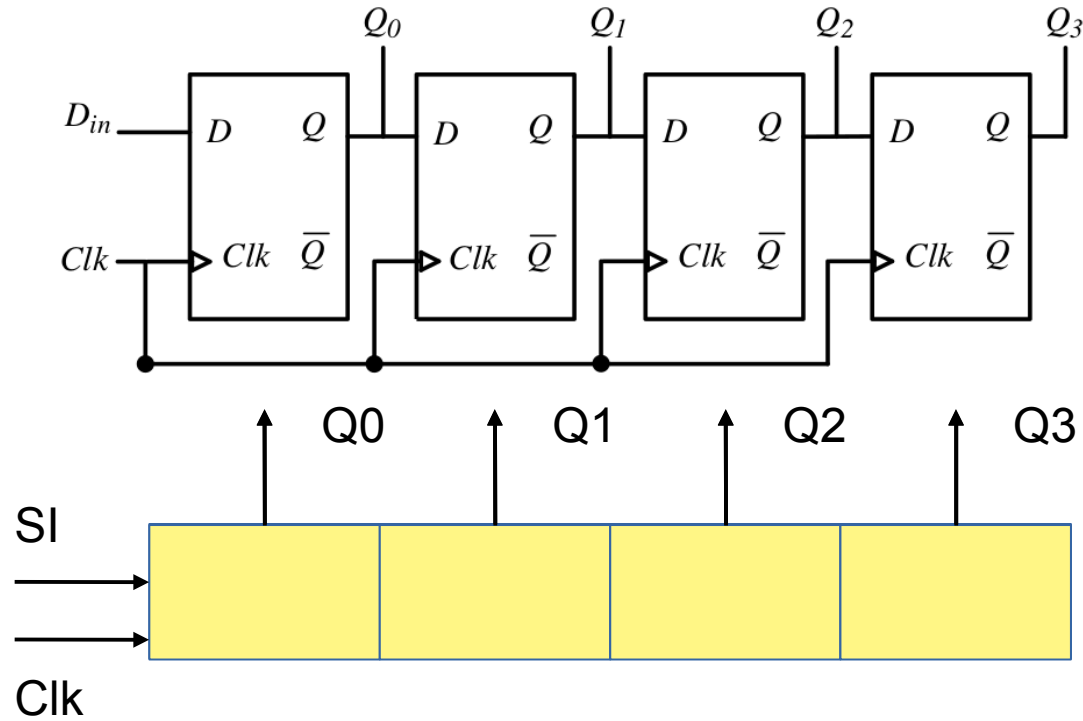


# Sistemas sincrónicos: SISO (2)

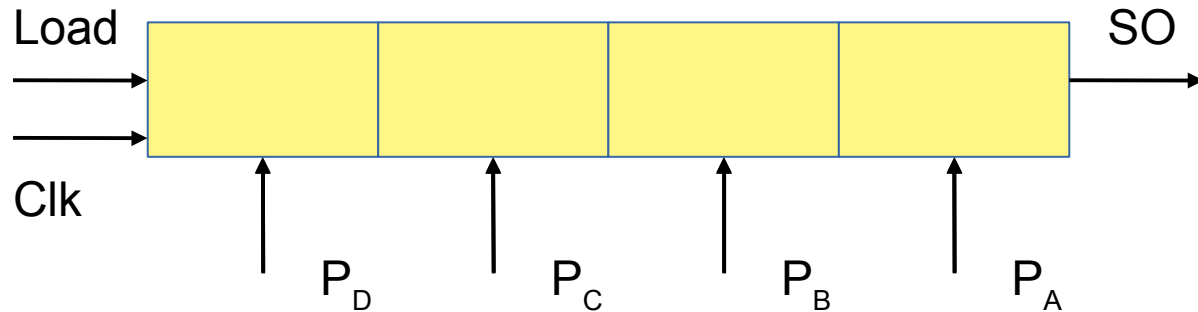
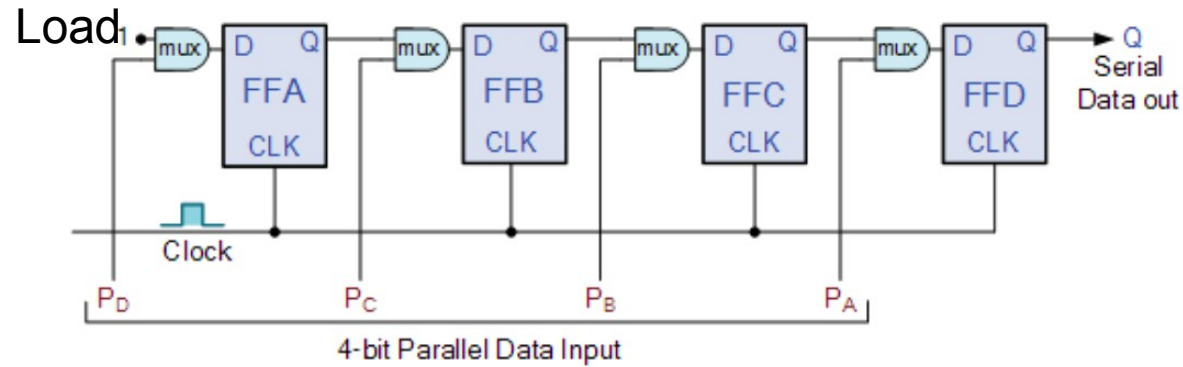


<i>Time</i>	$D_{in}$	$Q_0$	$Q_1$	$Q_2$	$Q_3$
$t_0$	1	0	0	0	0
$t_1$	0	1	0	0	0
$t_2$	0	0	1	0	0
$t_3$	1	0	0	1	0
$t_4$	0	1	0	0	1
$t_5$	1	0	1	0	0
$t_6$	1	1	0	1	0
$t_7$	0	1	1	0	1
$t_8$	1	0	1	1	0

# Sistemas sincrónicos: SIPO



# Sistemas sincrónicos: PISO



# Sistemas sincrónicos: resumen de registros

- El término *registro* se refiere a un grupo de  $N$  *flip-flops* operando como una única unidad para guardar o desplazar datos.
- Un *registro* es un conjunto de flip-flops con un *clock* común a todos los *flip-flops*
- Por ejemplo, un *registro de desplazamiento* es un *registro* de  $N$  *bits* que desplaza la posición de los datos almacenados en un bit (hacia la izquierda o derecha) en cada flanco de *clock*.
- Los *flip-flops* están conectados en una cadena de tal forma que la salida de un *flip-flop* es la entrada del siguiente.
- Un *clock* común excita todos los *flip-flops*.
- En las anteriores filmillas, se describieron los tipos básicos de registros

# Sistemas sincrónicos

## Contadores

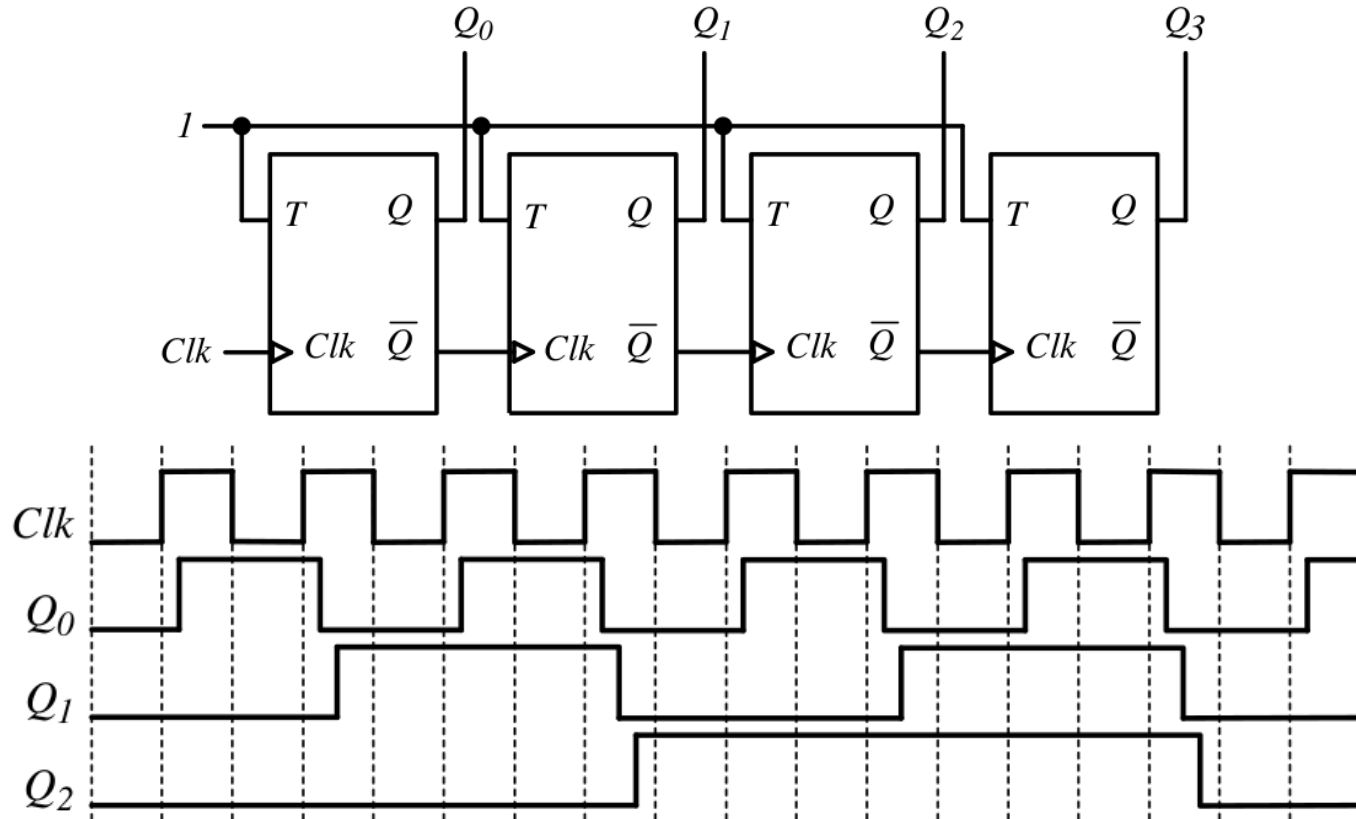
# Sistemas secuenciales: contadores

- Se denomina *contadores* a dispositivos simples que cuentan hacia arriba o hacia abajo (*up and down counters*).
- Se utilizan para contar números, operaciones, cantidades, períodos de tiempo, como divisores de frecuencia y para direccionamiento de información en memoria.
- Están realizados por una serie de *flip-flops* comunicados en un circuito.
- El número total de cuentas o estados estables se llama *módulo*.
- Por ejemplo, el módulo de un contador de cuatro etapas es de  $16_{10}$ , ya que es capaz de indicar desde  $(0000)_2$  hasta  $(1111)_2$ .

# Sistemas secuenciales: tipos de contadores

- Aunque estén realizados por *flip-flops*, es decir, a pesar que cada uno de los elementos constituyentes tengan una validación de *clock*, se pueden asociar de manera de realizar dos tipos distintos de contadores:
  - *Asincrónicos o ripple counters*
  - *Sincrónicos*

# Sistemas secuenciales: *ripple* counter





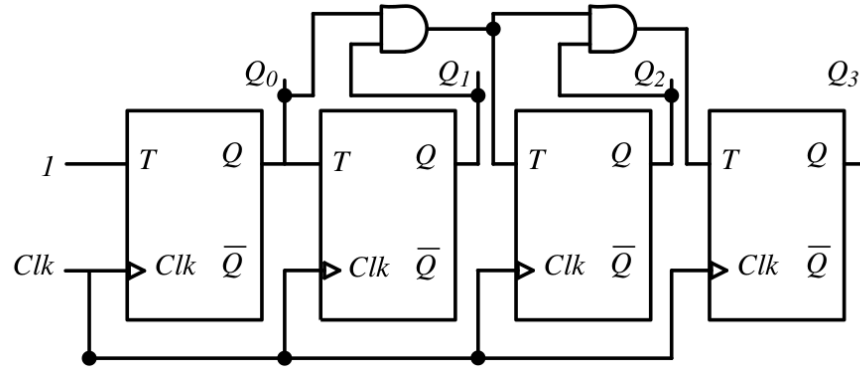
# Sistemas secuenciales: contadores asincrónicos o *ripple counters*

- En los contadores asincrónicos o *ripple counters* sólo en el primer *flip-flop* se conecta su entrada de *clk* al *clock* externo, mientras que en cada *flip-flop* sucesivo, su *clk* se conecta ya sea a la salida  $Q$  o  $Q_{negada}$  del *flip-flop* precedente.
- En la anterior filmiina, se mostró un *up counter* de módulo  $16_{10}$ , así como el diagrama de tiempos de un *up counter* de módulo  $8_{10}$  para que sea más claro el diagrama.
- Obsérvese que salvo la entrada de *clk* del primer *flip-flop* que se encuentra conectado al *clk* externo, los *clk* de las siguientes etapas se encuentran a la salida  $Q_{negada}$  de la etapa anterior
- La ventaja de este circuito es su sencillez pero su gran desventaja es que acumula retardos de propagación a lo largo de cada etapa.

# Sistemas secuenciales: contadores sincrónicos

- Como se indicó previamente, los contadores asincrónicos son simples pero su retardo de propagación acumulativo degrada severamente su utilización.
- Para el caso de contadores con una gran cantidad de etapas (piénsese en un contador de 64 bits en una parte interna de una moderna computadora), el retardo de propagación acumulativo bien puede ser tan grande como el período de *clock*.
- Para evitar este problema se usan *contadores sincrónicos*; en éstos, a diferencia de los asincrónicos, todos los *flip-flops* reciben una señal común de *clock* y todos los *flips-flops* cambian de estado *al mismo tiempo*.
- El inconveniente, entonces, es que siendo el *clk* común a todos ellos, la secuencia de conteo dependerá de las funciones lógicas que excitan las entradas de cada *flip-flop*.
- En la filmina siguiente, se muestra un *up counter* sincrónico de 4 etapas con el diagrama temporal de un *up counter* de 3 etapas para que el diagrama sea claro.

# Sistemas secuenciales: sincrónico *up counter*

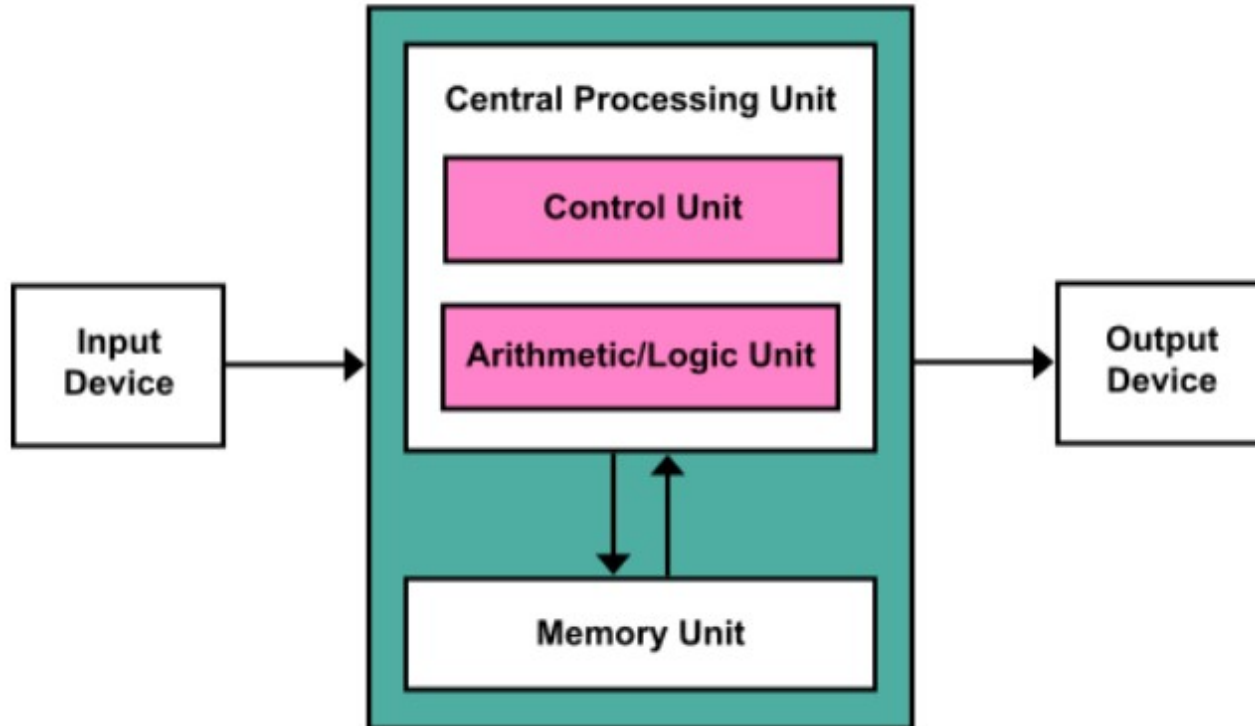


<i>Clock Cycle</i>	$Q_2$	$Q_1$	$Q_0$
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8	0	0	0

# Sistemas secuenciales

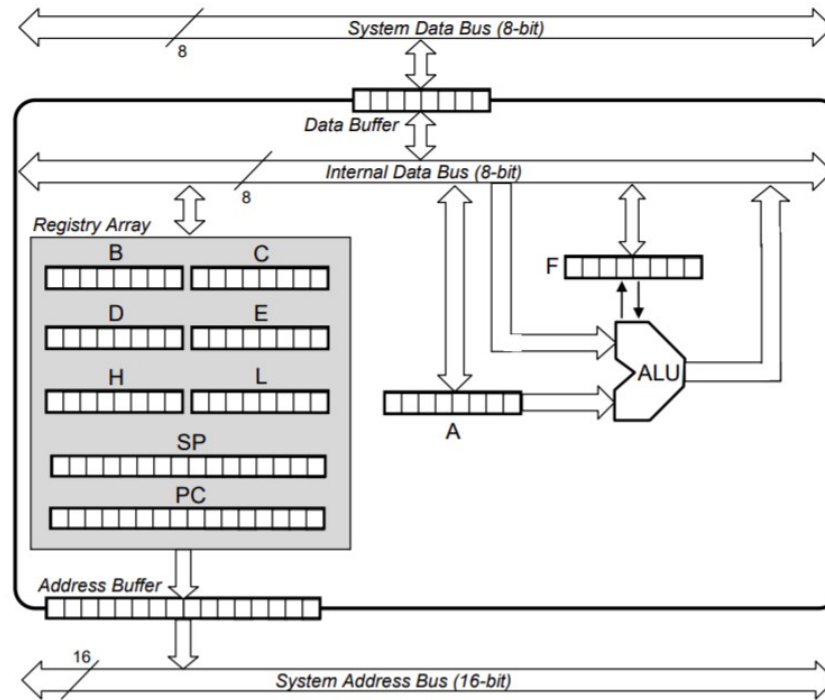
## Aplicaciones

# Sistemas secuenciales: Computadora y CPU



# Sistemas secuenciales: CPU 8080 simplificado

Intel 8080 CPU block diagram



# Sistemas secuenciales: CPU 8080

