En muchos circuitos digitales, el orden en el cual las cosas pasan es crítico:

- a veces, una acción tiene que preceder a otra
- a veces, dos acciones deben ocurrir simultáneamente

Los circuitos digitales usan *relojes* para proporcionar sincronización

También, para permitir que el hardware ejecute una acción sin que sea necesario un cambio en los inputs

… y para actualizar componentes que contienen *estado*

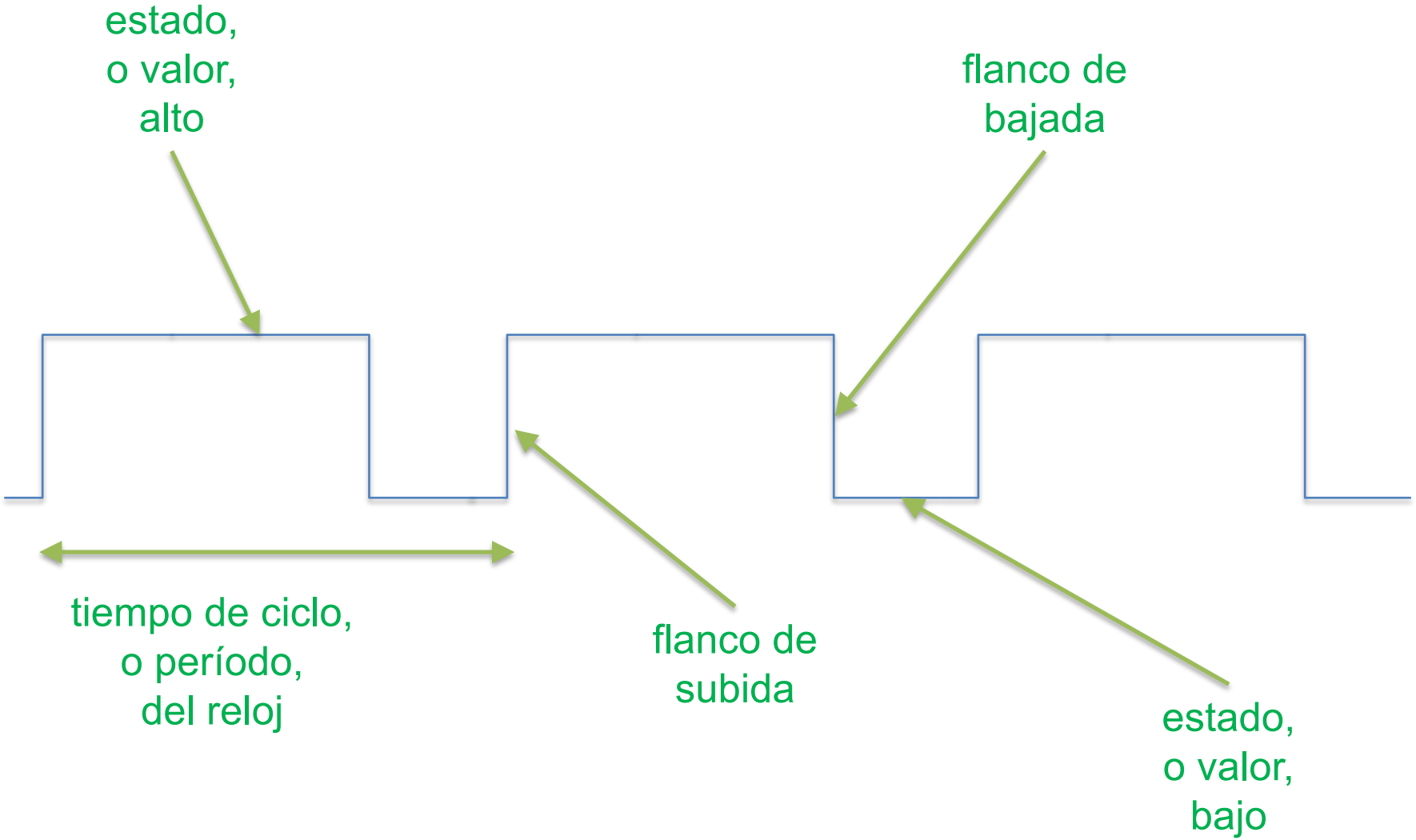
Un **reloj** —en un circuito digital— es un circuito electrónico que oscila a una tasa regular, emitiendo una serie de pulsos con dos propiedades:

- el ancho de pulso es preciso
- el intervalo entre pulsos consecutivos es preciso

El **tiempo de ciclo** (o *período*) del reloj es el intervalo de tiempo entre los flancos correspondientes —los *flancos de subida* o los *flancos de bajada*— de dos pulsos consecutivos

La **frecuencia** del reloj —p.ej., desde 100 MHz y hasta 4 GHz— es el inverso del tiempo de ciclo

[La mayoría de los relojes en circuitos digitales usan un cristal de cuarzo, que oscila naturalmente a una frecuencia precisa; el circuito del reloj amplifica la señal y la cambia de una onda sinusoidal a una onda cuadrada **]**



Los relojes son simétricos: el tiempo en el estado alto es igual al tiempo en el estado bajo. Para generar un tren de pulsos asimétricos, desplazamos el reloj básico usando un circuito de retardo y hacemos un AND entre esta señal y la original

Una componente esencial de todo computador es su *memoria*:

- sin memoria, no habría computadores tal como los conocemos hoy día
- la memoria se usa para almacenar tanto instrucciones a ser ejecutadas

  … como datos que se usan en la ejecución de las instrucciones

Para tener una memoria (de un bit), necesitamos un circuito que de alguna manera recuerde valores de input previos —es decir, que almacene su *estado*:

- el output del circuito depende tanto de los datos de input

  … como del valor almacenado dentro del circuito

El funcionamiento de un ***latch* S-R** se basa en que cada compuerta tiene un pequeño *retardo de propagación*:

- hay un retardo entre el instante en que el input cambia y el instante en que el output cambia consecuentemente (ver próx. diap.)
- lo construimos a partir de dos compuertas NAND (o dos compuertas NOR)

Efecto del retardo de propagación en un circuito muy simple:

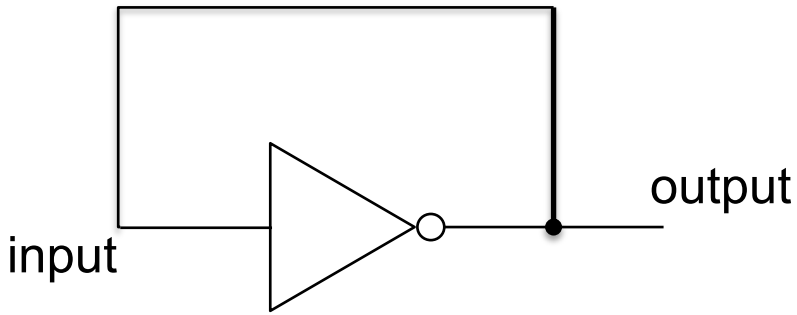
- si el output del circuito es 0, entonces el input del circuito es 0

  … esto parece una contradicción, ya que la (única) compuerta del circuito es un inversor (NOT)

  … excepto que sólo una vez que ha transcurrido el *retardo de propagación* (menos de un nanosegundo), el output cambia a 1

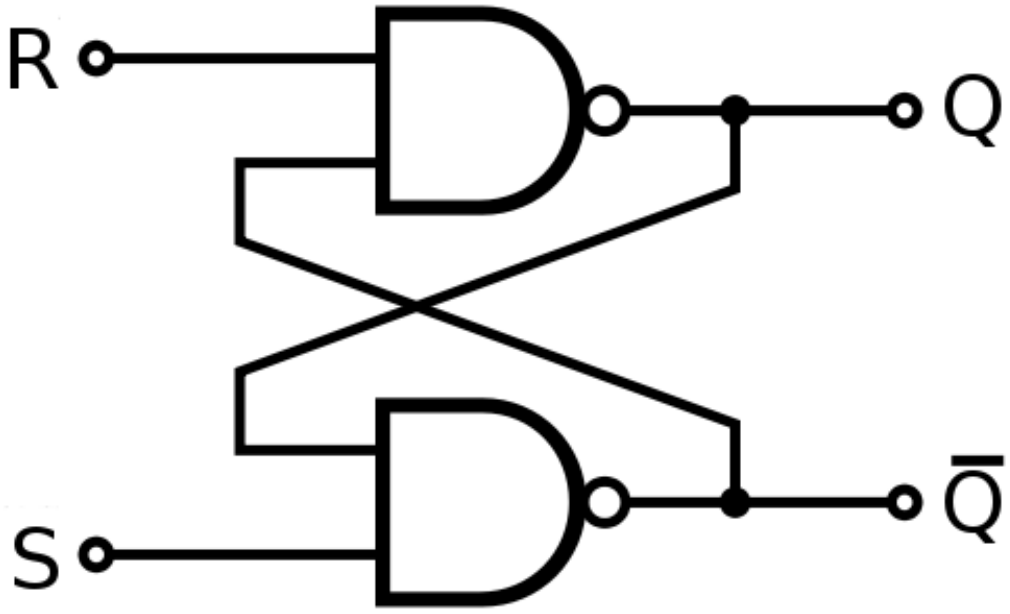
  … y sólo una vez que ha transcurrido otro retardo de propagación, el output vuelve a 0 nuevamente

- en principio, este ciclo continúa para siempre, por lo que el circuito oscila: el output cambia una y otra vez entre 0 y 1



Latch S-R:

- construido a partir de dos compuertas NAND (o dos compuertas NOR)
- dos inputs, *S* (*set*) y *R* (*reset*)
- dos outputs, *Q* y ¬*Q*, complementarios
- los outputs no están determinados únicamente por los inputs vigentes —no es un circuito combinacional (o, más propiamente, *combinatorio*)
- cuando *R* = *S* = 0 (*Q* = ¬*Q* = 1), el circuito se vuelve no determinista si *R* y *S* cambian a 1 simultáneamente

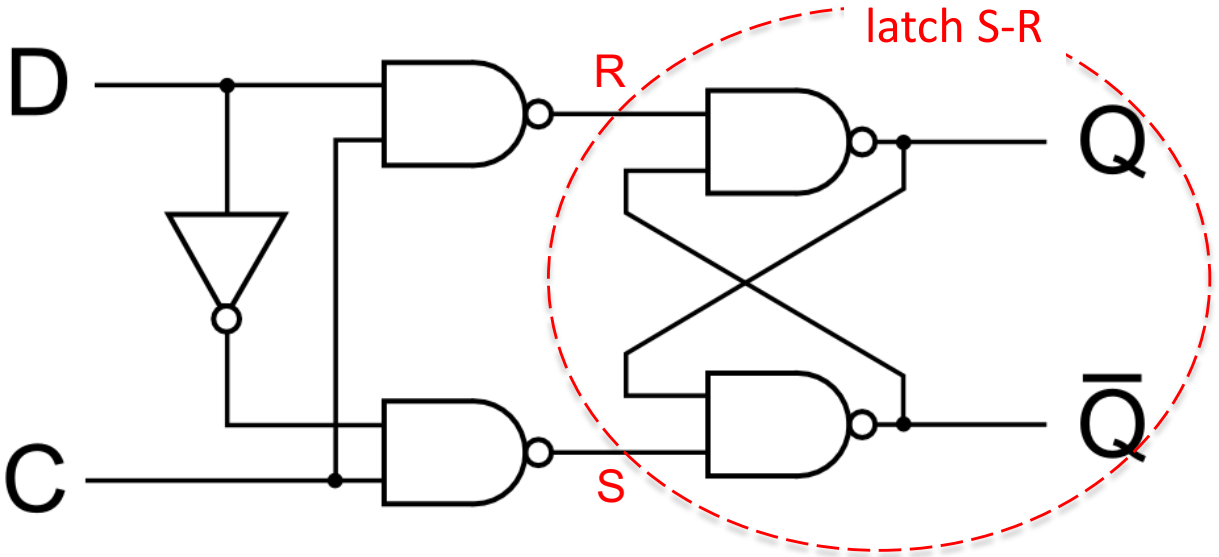


R	S	Q^{t+1}
0	0	-
0	1	1
1	0	0
1	1	Q^t

***Latch* D** (controlado por reloj) —a menudo conviene permitir que el latch cambie de estado sólo en ciertos instantes específicos:

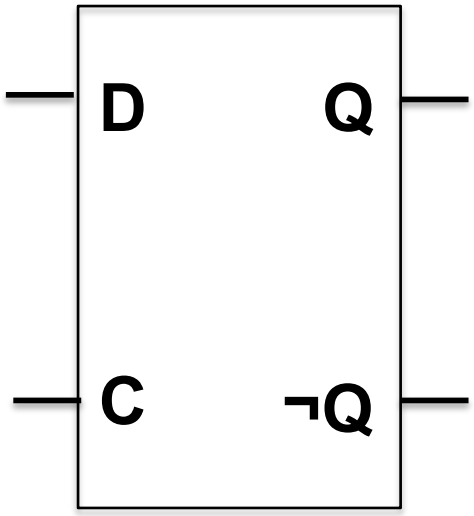
- evita el caso no determinista del *latch* S-R previniendo su ocurrencia
- el input *D* y su complemento, ¬*D*, entran a sendas compuertas NANDs justo antes del *latch* S-R

  … → los inputs *R* y *S* del *latch* S-R no pueden ser ambos 0 simultáneamente
- cuando el reloj, *C*, está en 1, el *latch* "se carga con" el valor de *D*
- el valor almacenado en el *latch* está siempre disponible en el output *Q*



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

***Latch* D** (controlado por reloj) —más esquemáticamente:



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

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

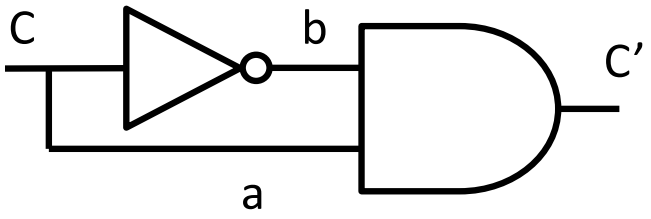
En el diseño y construcción de computadores digitales, necesitamos circuitos que puedan leer el valor de una señal *en un instante particular en el tiempo* y almacenarlo

Esto elimina los problemas que podrían ocurrir si varias señales fueran leídas en momentos en el tiempo levemente diferentes

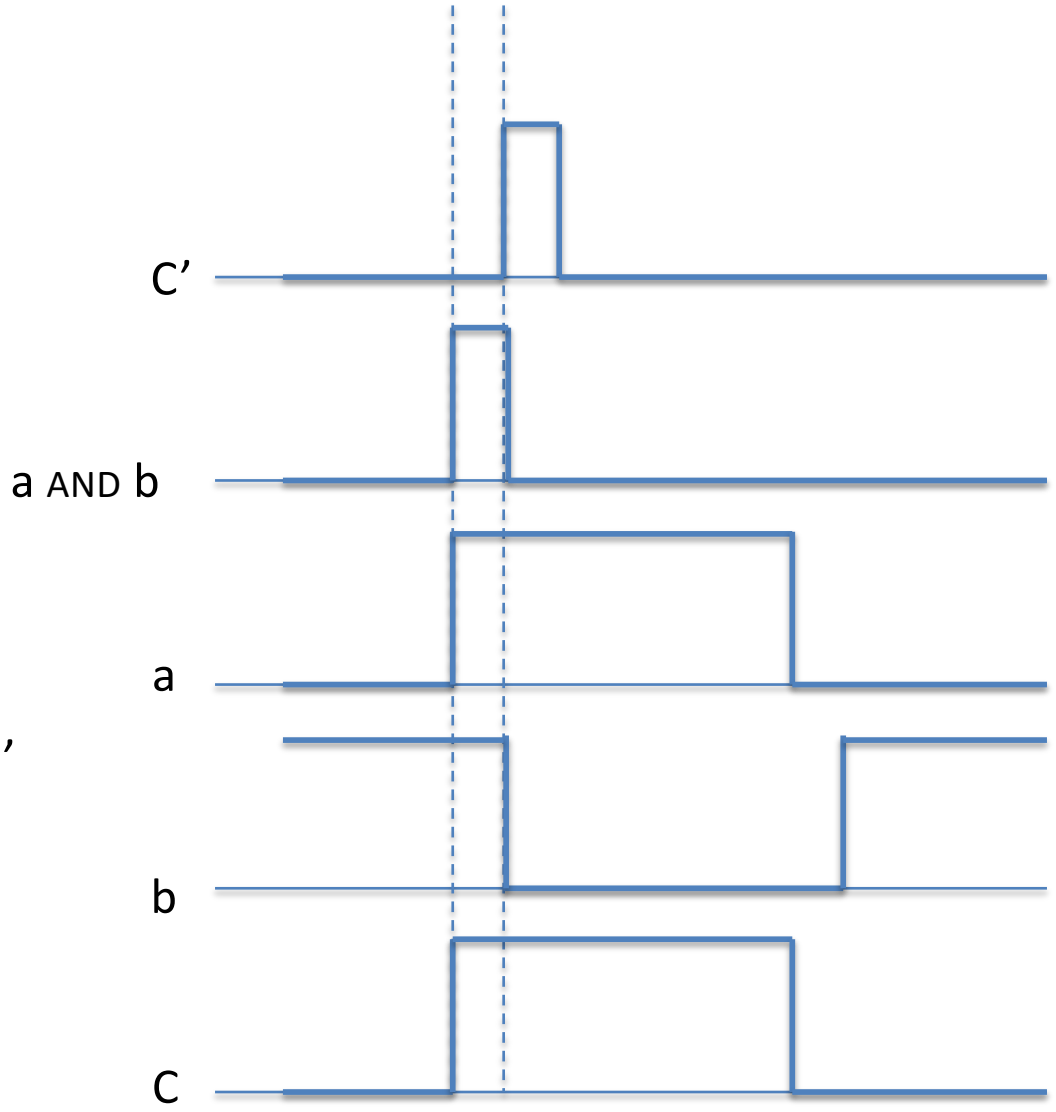
… que es lo que podría ocurrir en el caso de los *latches* como el anterior, en que la señal *D* puede ser leída en cualquier instante mientras el reloj está en su valor alto, o 1 (o, equivalentemente, en su valor bajo)

Una posibilidad es "estrechar" al máximo el intervalo de tiempo durante el cual el reloj está en su valor alto, p.ej., usando un *generador de pulsos C'* a partir del reloj *C* (próx. diap.):

- el ancho del pulso *C'* es el retardo de propagación de las compuertas



generador de pulsos



Estos circuitos se conocen como **flip-flops** y pueden ser construidos de diferentes maneras

Lo importante es que la transición de estado —es decir, el almacena-miento en el *latch* del valor que hay en el input *D*

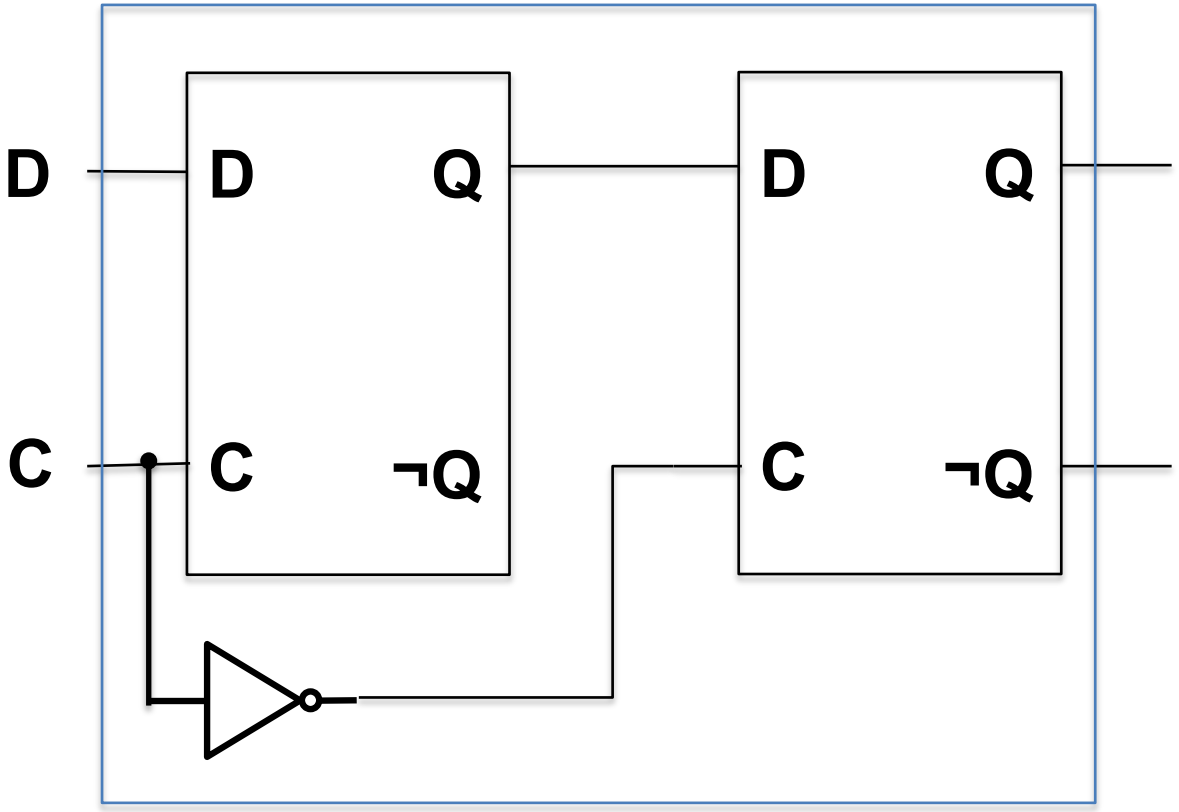
… ocurre, en la práctica, **durante la transición** del reloj de 0 a 1, o *flanco de subida* … y no cuando el reloj está en 1

(también podría ser que la transición de estado ocurra en el *flanco de bajada*)

La próxima diapositiva muestra un **flip-flop D** construido a partir de dos latches D y un inversor (compuerta NOT)

Flip-flop D:

- cuando la señal del reloj *C* cambia de 1 a 0 (o, en otros circuitos equivalentes, de 0 a 1), el output *Q* almacena el valor del input *D*
- usamos un arreglo de flip-flops D para construir un **registro** que puede almacenar un dato de varios bits, tal como un byte o una palanbra



C	D	Q^{t+1}
0/1/↑	0/1	Q^t
↓	0	0
↓	1	1

Usamos un arreglo de flip-flops D para construir un **registro** …

… que puede almacenar un dato de varios bits, tal como un *byte* (8 bits) o una *palabra* (32 bits en los computadores modernos):

- p.ej., un registro de 4 bits: