

# Clase 04 - Almacenamiento de datos

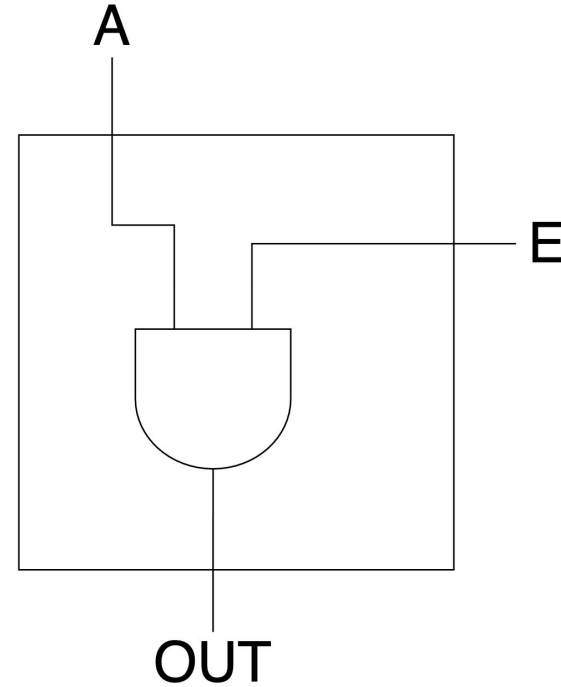


Profesor: **IIC2343 - Arquitectura de Computadores**  
- Felipe Valenzuela González  
Correo:  
frvalenzuela@alumni.uc.cl

# **Resumen de la clase pasada**

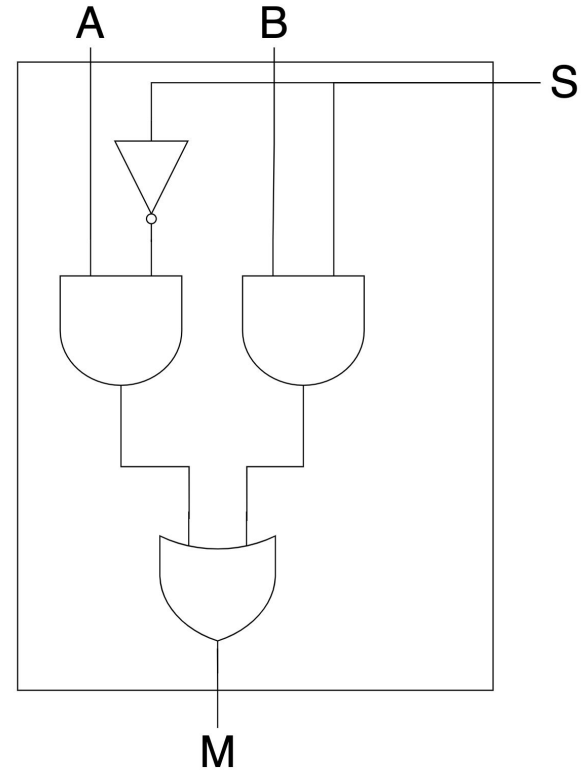
# Circuitos de control - Enabler

OUT	E
0	0
A	1



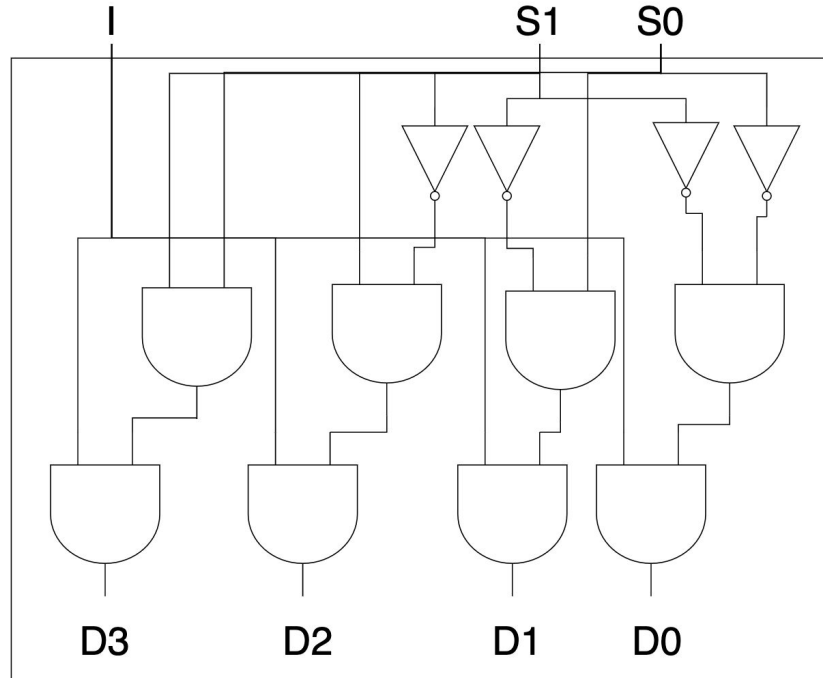
# Circuitos de control - Multiplexor

S	M
0	A
1	B



# Circuitos de control - Demux

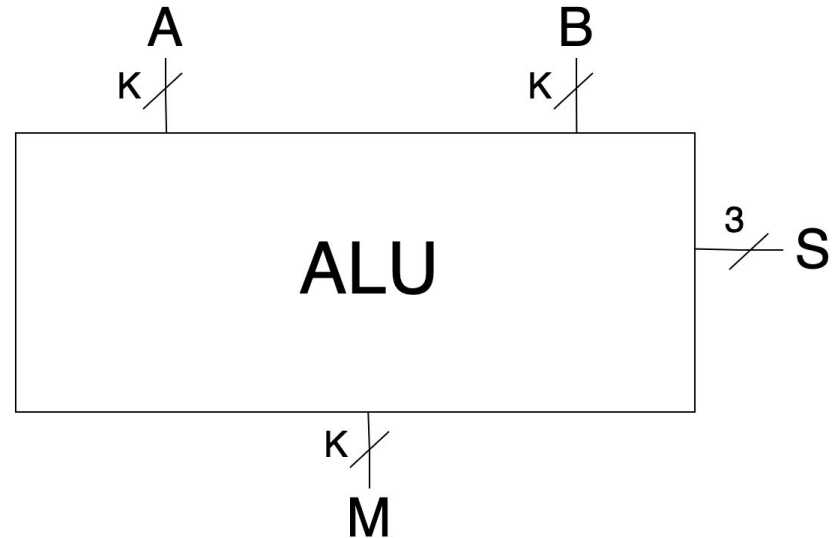
S1	S0	Output
0	0	D0 = I
0	1	D1 = I
1	0	D2 = I
1	1	D3 = I



# Unidad Aritmética Lógica: ALU

- Tabla de valores:

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



**¿Dudas?**

# Introducción del curso:

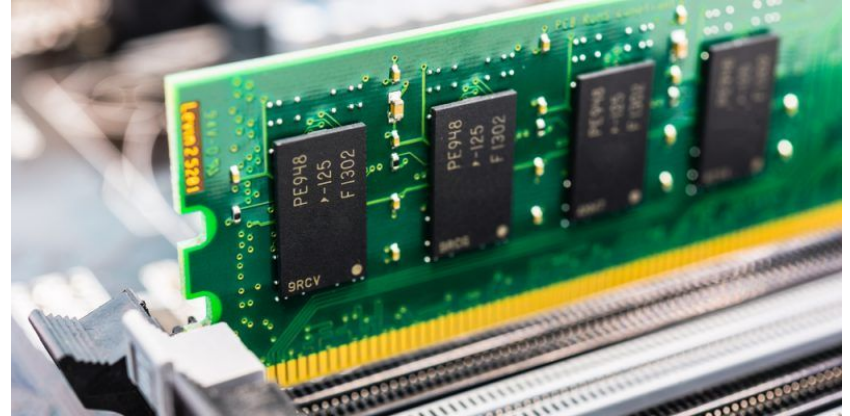
- Un computador lo definimos como una **máquina programable que ejecuta programas.**
- Para programar necesitamos:
  - Datos: números (enteros, reales) , texto, imágenes, etc
  - Operaciones: suma, resta, multiplicación, división, etc
  - Variables: simples, arreglos
  - Control de flujo: comparaciones, manejo de ciclos
- Hoy veremos **lo básico para tener variables**





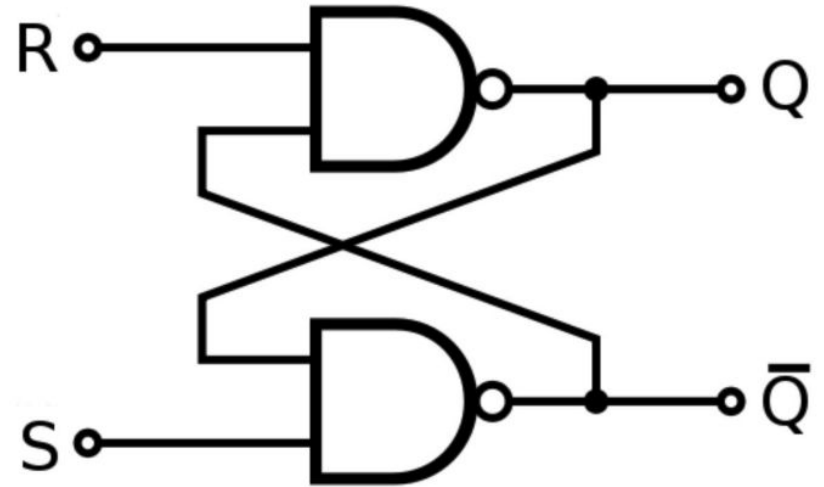
# Almacenamiento de datos

- Es componente esencial de todo computador
- Es *hardware* especializado para **almacenar** un estado
- Permite realizar **un cambio** en un **instante determinado de tiempo**



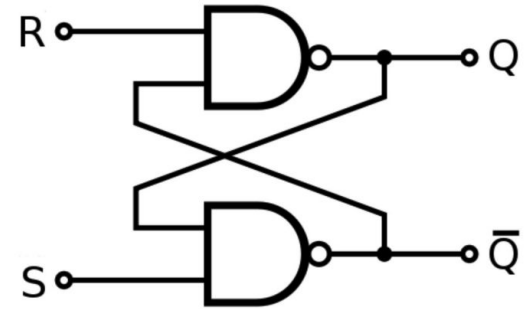
# Almacenamiento de datos: Latch RS

S	R	Q(t+1)
0	0	-
0	1	0
1	0	1
1	1	Q(t)



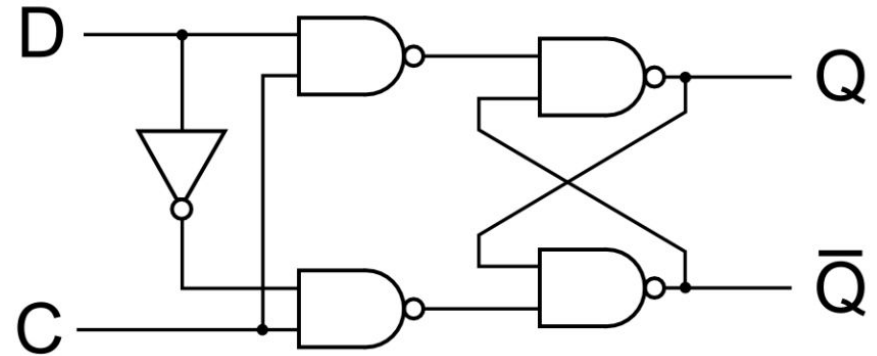
# Almacenamiento de datos: Latch RS

S	R	Q(t+1)
0	0	-
0	1	0
1	0	1
1	1	Q(t)



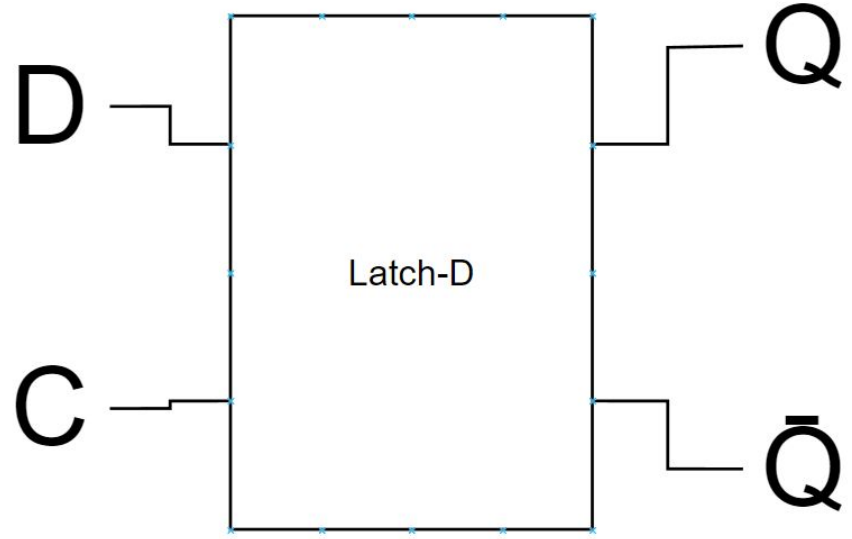
# Almacenamiento de datos: Latch D

C	D	$Q(t+1)$
0	0	$Q(t)$
0	1	$Q(t)$
1	0	0
1	1	1



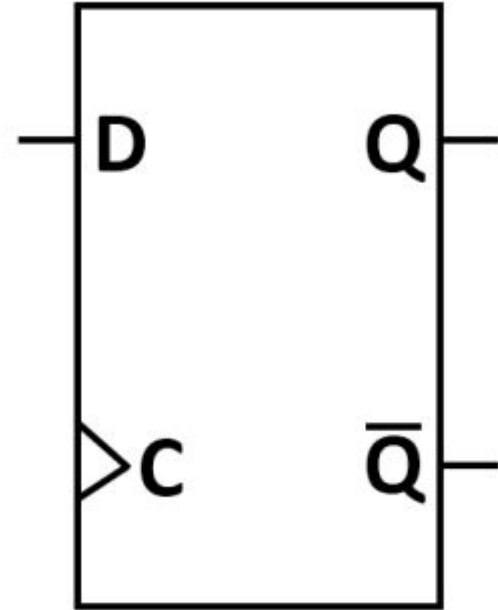
# Almacenamiento de datos: Latch D

- D: Dato
- C: control
- Q: estado
- Esta es una abstracción usada para denotar el latch D



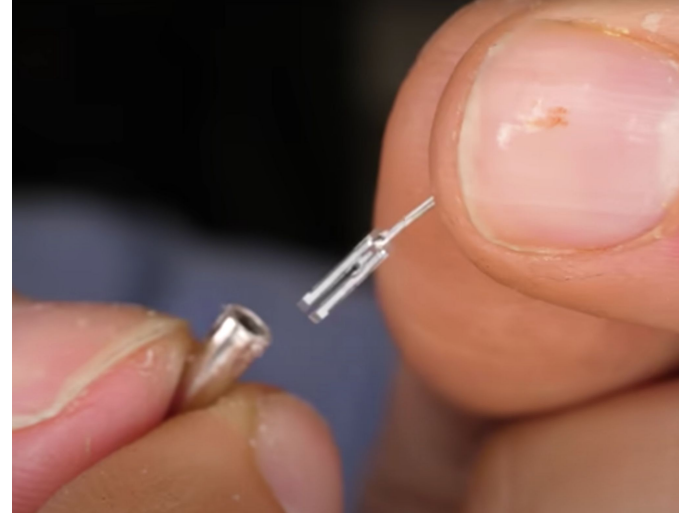
# Almacenamiento de datos: Latch D

- Los **latches** se activan en un estado (1 ó 0), lo que no soluciona del todo nuestros problemas.
- Necesitamos una pieza similar, pero que se active sólo en **un instante dado**.

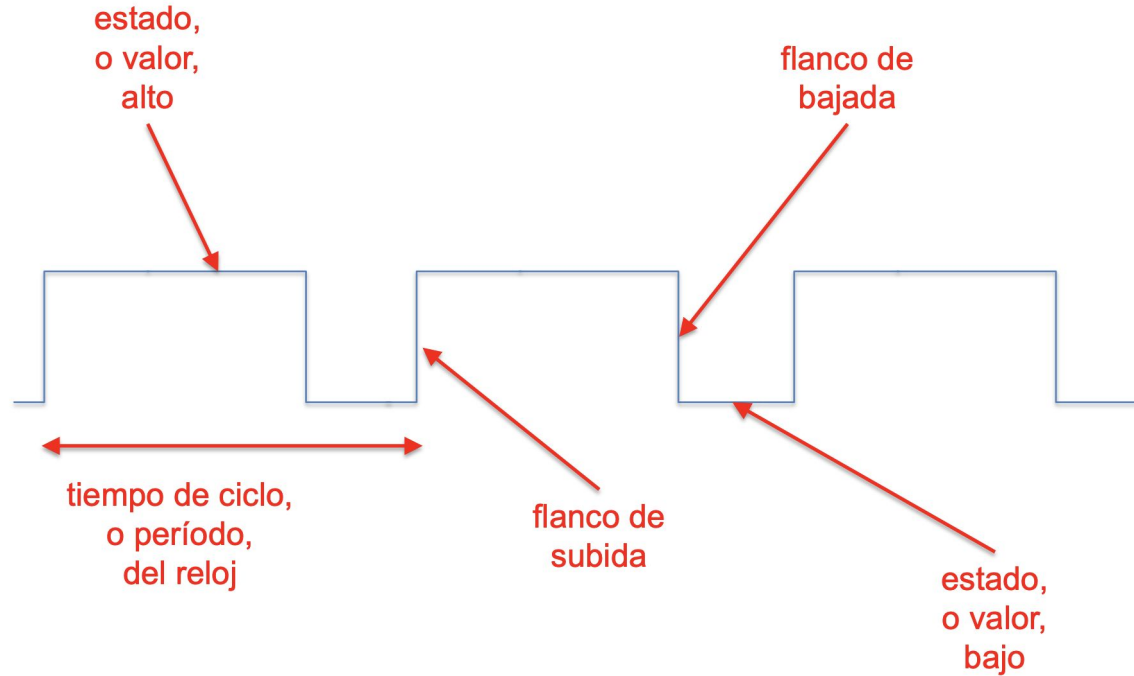


# Almacenamiento de datos: Relojes

- Un reloj —en un circuito digital— es simplemente un circuito que emite una serie de pulsos, con dos propiedades:
- El ancho de pulso es preciso
- El intervalo entre pulsos consecutivos es preciso



# Almacenamiento de datos: Relojes

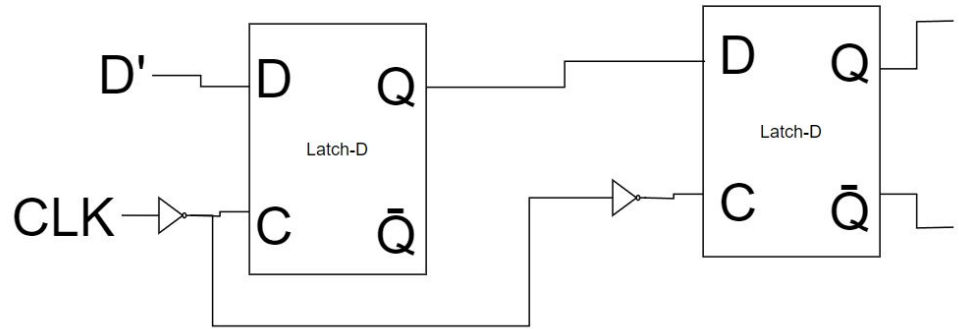




**¿Dudas?**

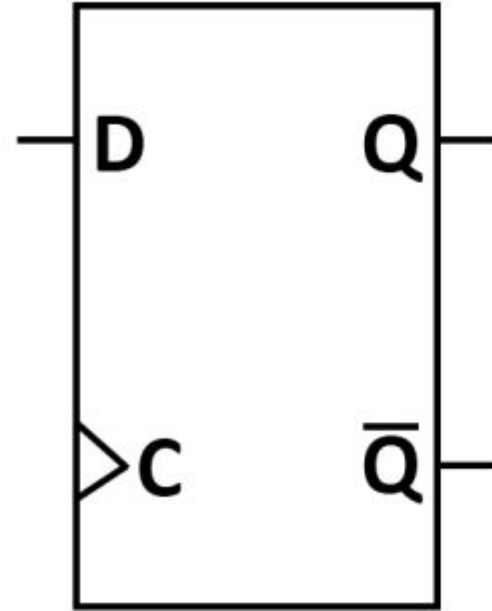
# Almacenamiento de datos: Flip-Flop D

C	D	Q(t+1)
↑	0	0
↑	1	1
*	*	Q(t)



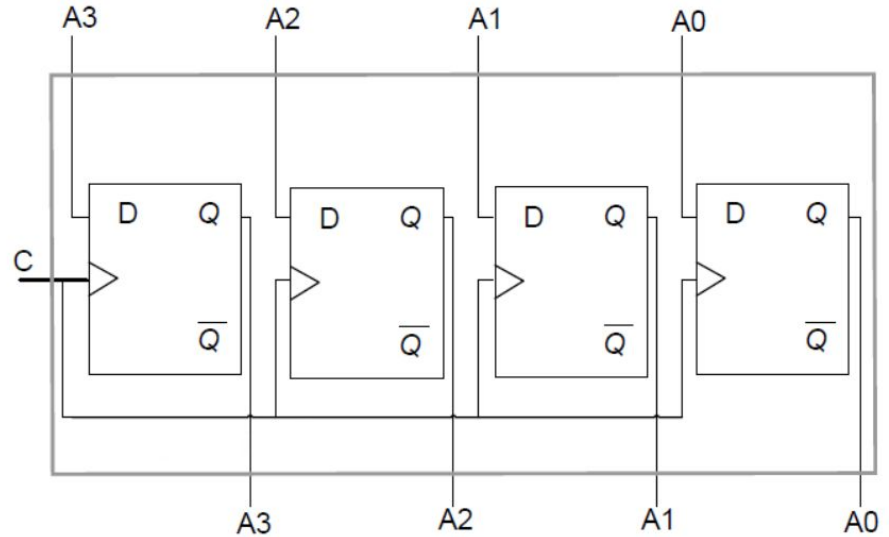
# Almacenamiento de datos: Registro

- Los **flip-flop** se activan solo en una transición o flanco (subida o bajada).
- Cuando una componente tenga un input con símbolo un triángulo, significa que ese input es un clock.



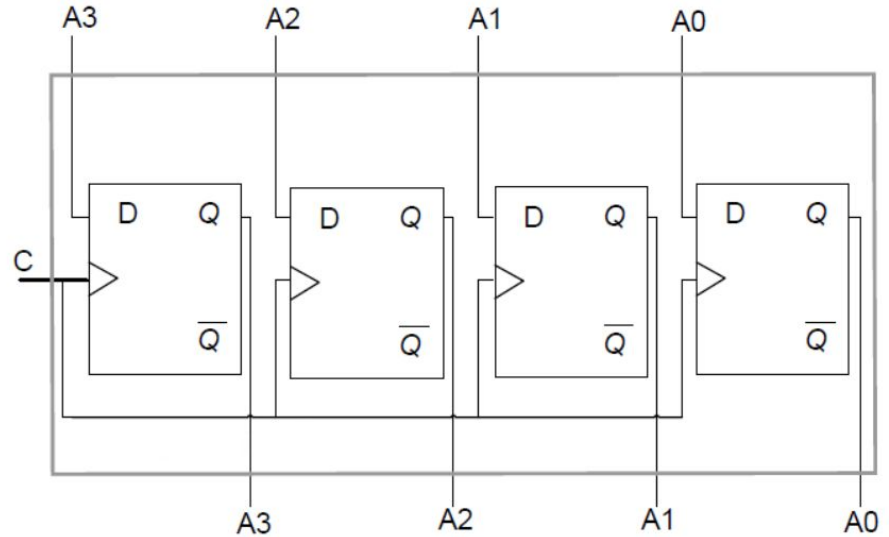
# Almacenamiento de datos: Registros

- Llamaremos registros al conjunto de flip-flops
- Si queremos almacenar  $N$  bits necesitaremos  $N$  flip-flops



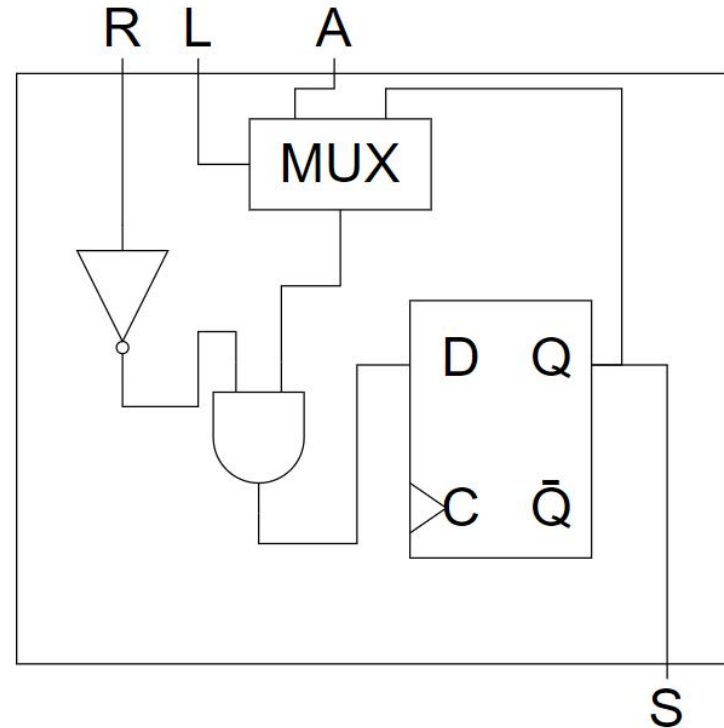
# Almacenamiento de datos: Registros

- Llamaremos registros al conjunto de flip-flops
- Si queremos almacenar  $N$  bits necesitaremos  $N$  flip-flops
- Nos **falta control de almacenamiento**



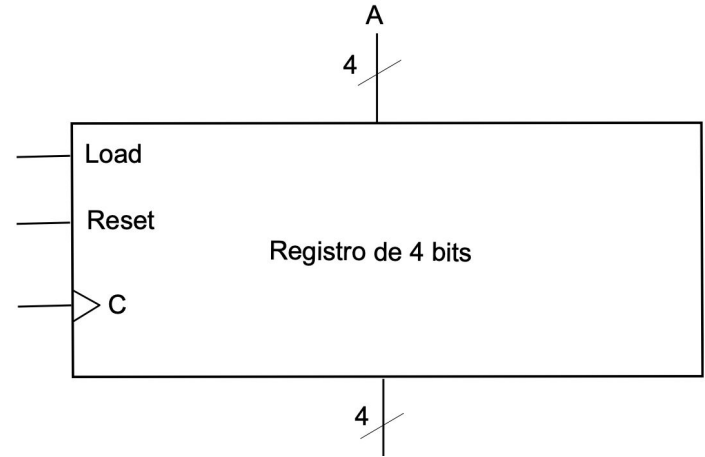
# Almacenamiento de datos: Registros

C	L	R	S
↑	1	0	A
↑	*	1	0
*	0	0	Q(t)



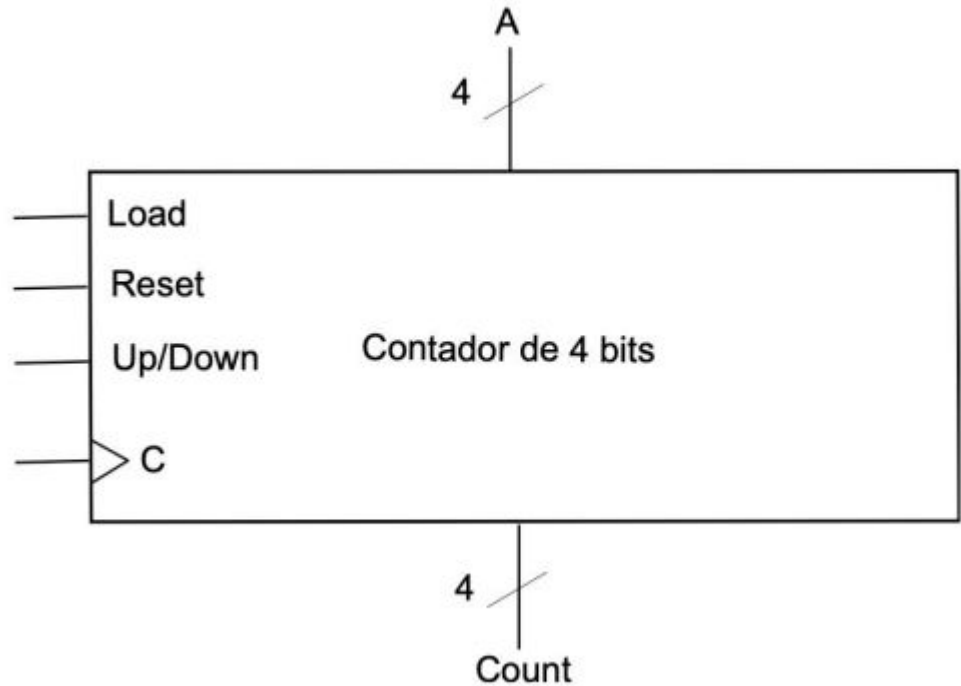
# Almacenamiento de datos: Registros

- El registro tradicional tiene la función de carga y reset
- Se extiende con el manejo de varios flip-flops
- Nos permite almacenar unidades de información o **palabras**



# Almacenamiento de datos: Registro Contador

- Con lo anterior se puede generalizar para tener un registro contador
- Una señal Up indicando que el valor almacenado se suma uno
- Una señal Down indicando que el valor almacenado se resta uno





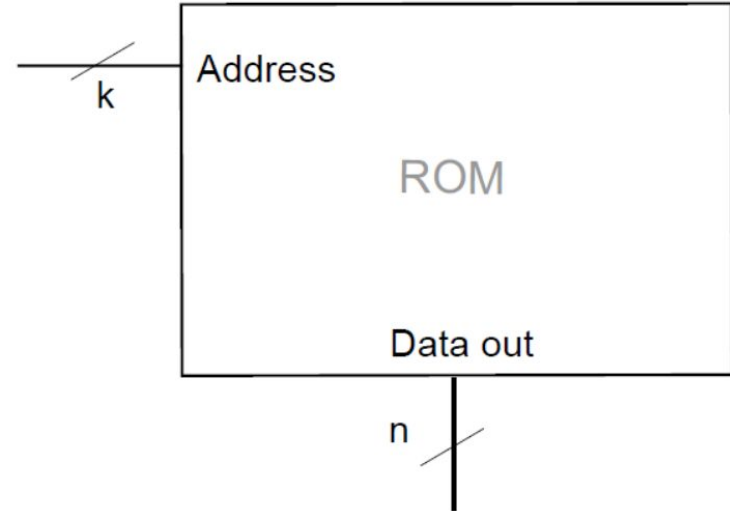
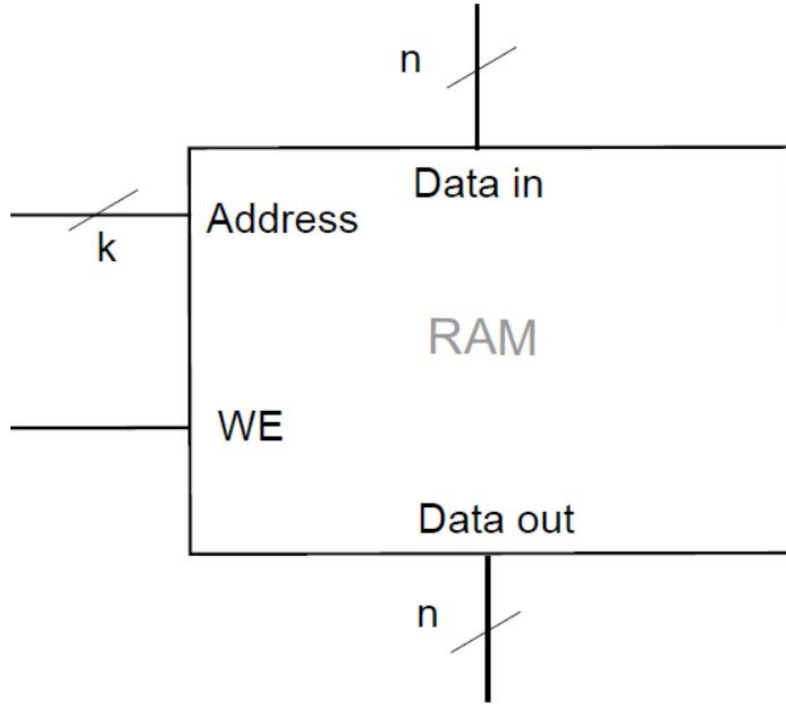
**¿Dudas?**

# Almacenamiento de datos: Memorias

- Componentes que permite acceder, y en algunos casos, modificar distintos datos según una **dirección de memoria**
- El proceso de acceso se conoce como **direccionamiento**

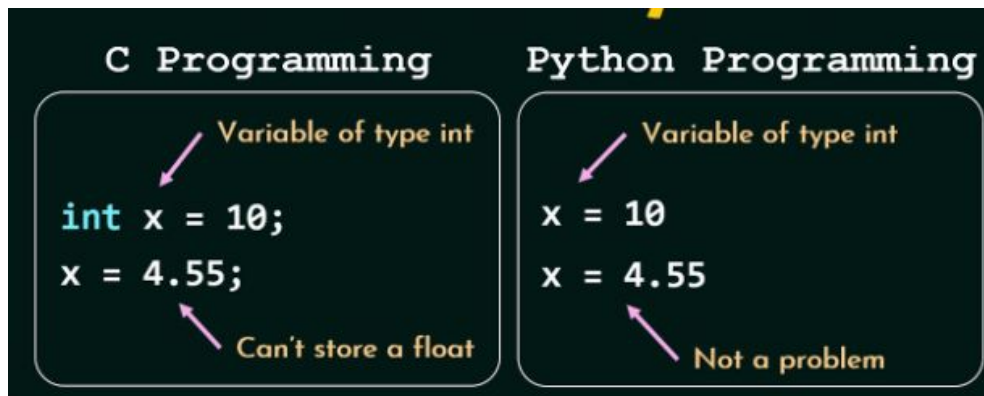
Dirección en decimal	Dirección en binario	Palabra
0	000	<i>Palabra<sub>0</sub></i>
1	001	<i>Palabra<sub>1</sub></i>
2	010	<i>Palabra<sub>2</sub></i>
3	011	<i>Palabra<sub>3</sub></i>
4	100	<i>Palabra<sub>4</sub></i>
5	101	<i>Palabra<sub>5</sub></i>
6	110	<i>Palabra<sub>6</sub></i>
7	111	<i>Palabra<sub>7</sub></i>

# Almacenamiento de datos: Memorias



# Almacenamiento de datos: Variable

- Valores temporales que se actualizan según flujo del código
- Se almacenan en distintos componentes, como puede ser registros o RAM
- Todo variable tiene un único **tipo de dato**



# Almacenamiento de datos: Endianness

Address	Value
00	0x0F
01	0xA7
10	0x1D
11	0x23

- int 32 bits big endian:

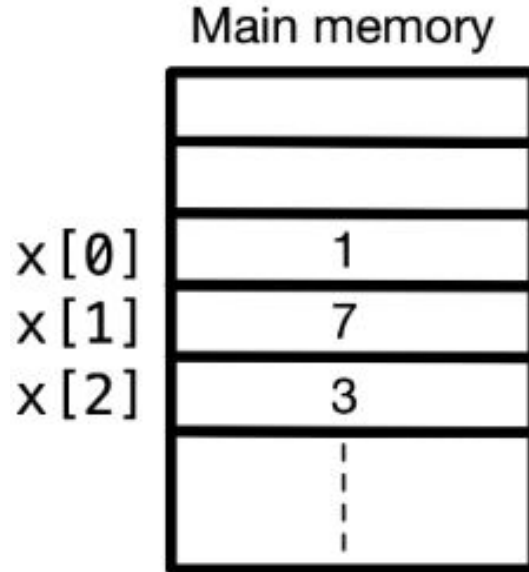
**0x 0F A7 1D 23**

- int 32 bits little endian:

**0x 23 1D A7 0F**

# Almacenamiento de datos: Arreglos

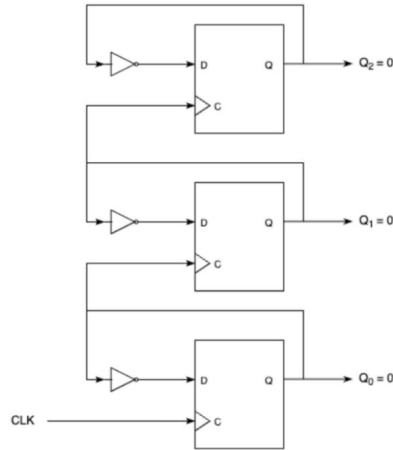
- Es un caso especial de variable
- Se compone de cuatro atributos: **tipo de dato**, **endianness**, **largo** y **dirección de inicio**
- Se puede **indexar** cada elemento para acceder a ellos en nuestra memoria



**¿Dudas?**

# Ejercicio prueba I1 - 2024-1

- (c) (3 ptos.) Suponga que posee un componente con una salida de 3 bits igual a  $Q_2Q_1Q_0$  conectado a una señal de control CLK. Su circuito interno se compone de flip-flops de tipo D conectados de la siguiente forma:



Asumiendo que se tiene un valor inicial  $Q_2Q_1Q_0 = 000$ , indique, justificadamente, cómo se va modificando el estado del componente con cada flanco de subida de la señal CLK.

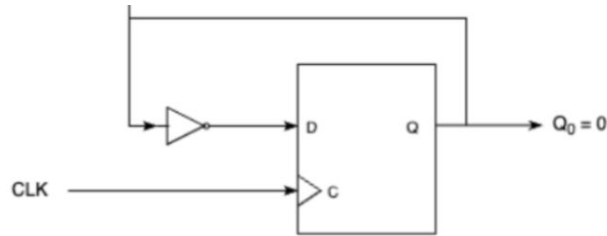


# Ejercicio prueba I1 - 2024-1

C	D	$Q(t+1)$
$\uparrow$	0	0
$\uparrow$	1	1
*	*	$Q(t)$

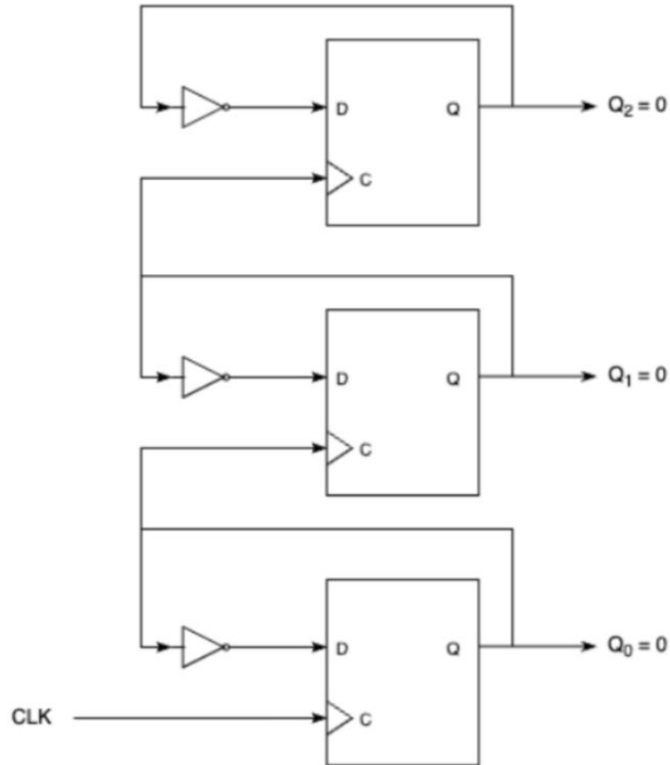
- Es D es el dato que quiero guardar.
- Q es el estado (lo que tengo guardado).
- Solo cuando ocurre un flanco de subida del reloj ( $\uparrow$ ), el flip-flop copia D en Q.
- En cualquier otro momento, Q se queda igual

# Ejercicio prueba I1 - 2024-1



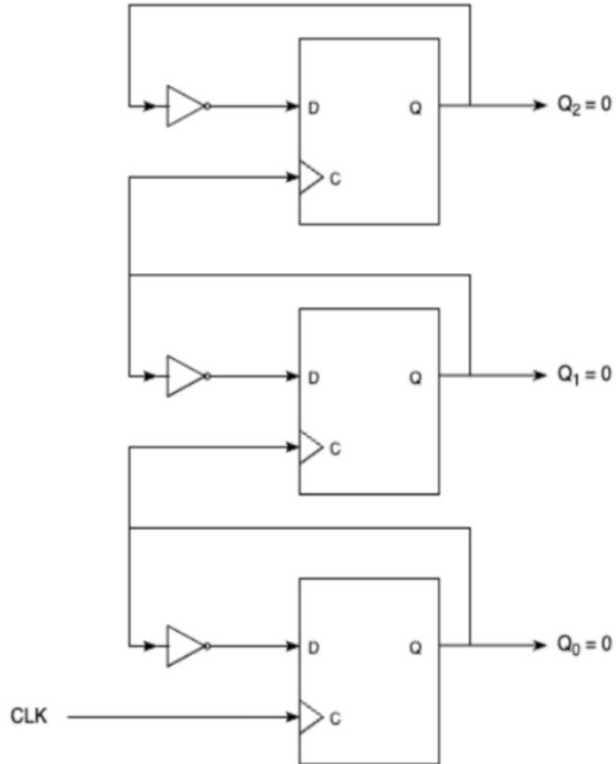
- Si al flip-flop D le conecto como entrada el valor contrario de su salida ( $D = \text{NOT } Q$ ):
- En cada flanco de subida del reloj, el flip-flop guarda lo opuesto de lo que tenía.
- Ejemplo: si  $Q=0$ , en el próximo flanco sube a 1; en el siguiente, baja a 0; después sube a 1, etc.
- Eso significa que la salida alterna:  $0 \rightarrow 1 \rightarrow 0 \rightarrow 1 \dots$  cada vez que llega un pulso de reloj.
- Este es el “**truco**” para que cada flip-flop cambie de estado de manera automática.

# Ejercicio prueba I1 - 2024-1



- Hay tres flip-flops D que generan las salidas  $Q_2$ ,  $Q_1$  y  $Q_0$ .
- El primero ( $Q_0$ ) recibe el reloj global.
- El segundo ( $Q_1$ ) no recibe el reloj global, sino que usa  $Q_0$  como su reloj.
- El tercero ( $Q_2$ ) usa  $Q_1$  como su reloj.
- Eso significa que:
  - $Q_0$  cambia en cada flanco del reloj global.
  - $Q_1$  cambia cuando  $Q_0$  pasa de 0 a 1.
  - $Q_2$  cambia cuando  $Q_1$  pasa de 0 a 1

# Ejercicio prueba I1 - 2024-1

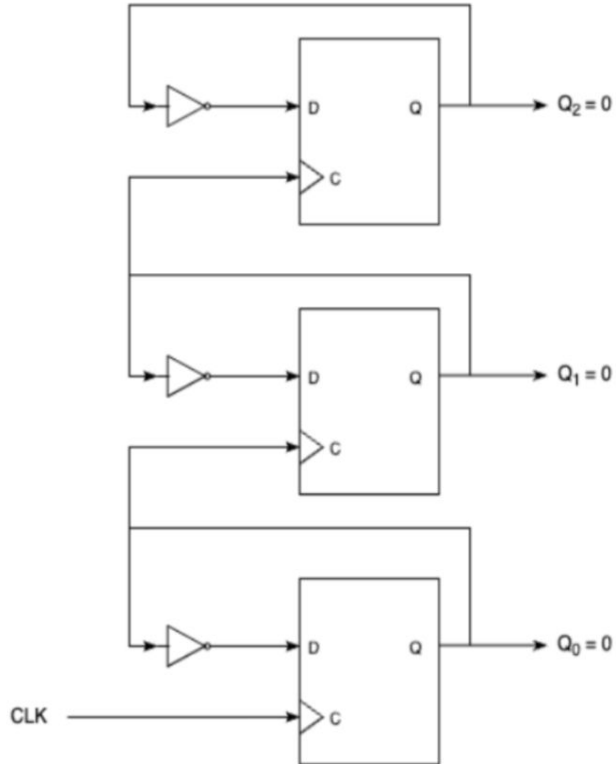


## Evolución del estado (paso a paso)

Partimos en **Q<sub>2</sub>Q<sub>1</sub>Q<sub>0</sub> = 000**. Veamos qué pasa en cada flanco del reloj global:

- 000 → 111**
  - Reloj sube → Q<sub>0</sub>: 0→1.
  - Ese cambio de Q<sub>0</sub> hace que Q<sub>1</sub> también cambie: 0→1.
  - Ese cambio de Q<sub>1</sub> hace que Q<sub>2</sub> también cambie: 0→1.
- 111 → 110**
  - Q<sub>0</sub>: 1→0.
  - Como fue bajada, Q<sub>1</sub> no cambia → sigue en 1.
  - Q<sub>2</sub> sigue en 1.
- 110 → 101**
  - Q<sub>0</sub>: 0→1.
  - Ahora Q<sub>1</sub> cambia: 1→0.
  - Como Q<sub>1</sub> bajó, Q<sub>2</sub> no cambia → queda en 1.

# Ejercicio prueba I1 - 2024-1

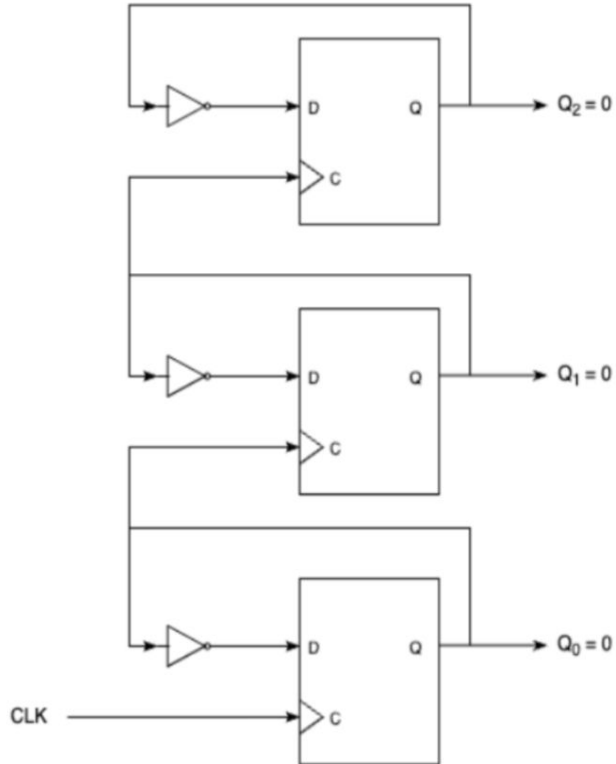


## Evolución del estado (paso a paso)

Partimos en  $Q_2Q_1Q_0 = 000$ . Veamos qué pasa en cada flanco del reloj global:

4. **101 → 100**
  - $Q_0: 1 \rightarrow 0$ .
  - $Q_1$  no cambia.
5. **100 → 011**
  - $Q_0: 0 \rightarrow 1$ .
  - $Q_1: 0 \rightarrow 1$ .
  - $Q_2: 1 \rightarrow 0$ .
6. **011 → 010**
  - $Q_0: 1 \rightarrow 0$ .
  - $Q_1$  no cambia.

# Ejercicio prueba I1 - 2024-1



## Evolución del estado (paso a paso)

7. **010 → 001**

- Q0: 0 → 1.
- Q1: 1 → 0.
- Q2 no cambia.

8. **001 → 000**

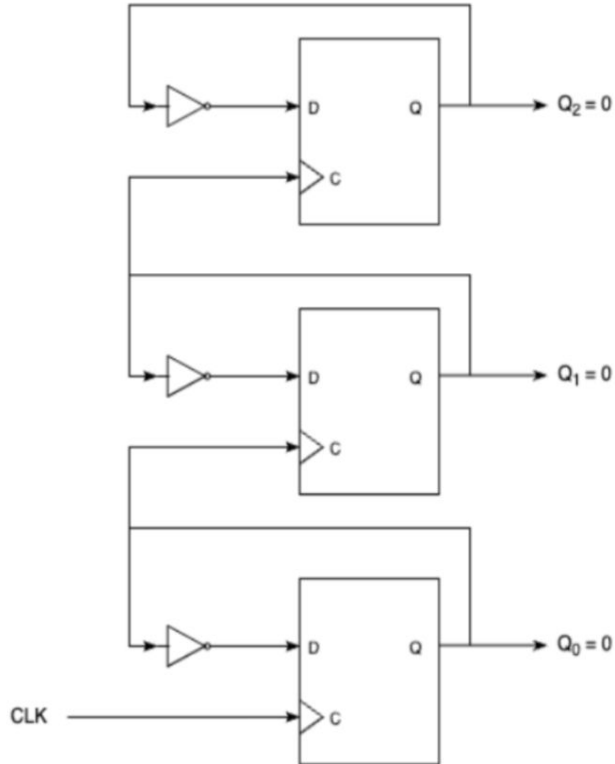
- Q0: 1 → 0.
- Q1 y Q2 no cambian.

Y volvemos al inicio. La secuencia completa es:

**000 → 111 → 110 → 101 → 100 → 011 → 010 → 001 → 000.**

○

# Ejercicio prueba I1 - 2024-1



Si miramos los estados como números binarios:

- 000 = 0
- 111 = 7
- 110 = 6
- 101 = 5
- 100 = 4
- 011 = 3
- 010 = 2
- 001 = 1
- 000 = 0 (reinicio)

La salida va **bajando de a uno en binario**. Es decir: este circuito es un **contador de 3 bits que va decrementando en cada pulso de reloj**

**¿Dudas?**



# Bibliografía

- Apuntes históricos. Hans Löbel, Alejandro Echeverría

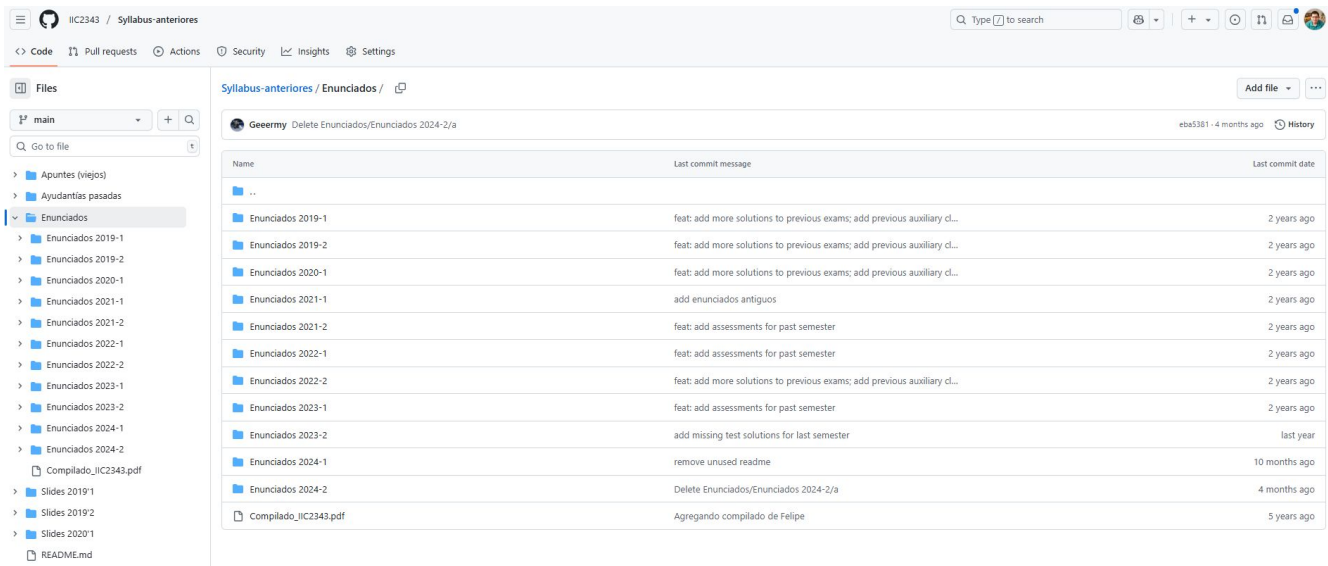
## 04 - Almacenamiento de datos

- D. Patterson, Computer Organization and Design RISC-V. Edition: The Hardware Software Interface. Morgan Kaufmann, 2020. Capítulos A.7-A.8-A.9. Página A-47, 792 en PDF

▼ Bibliografía			✓	▼	+	⋮
⋮	🔗	<a href="#">Sarah Harris, David Harris Digital Design and Computer Architecture.pdf</a>	✓	⋮		
⋮	🔗	<a href="#">Computer Organization and Design RISC-V edition.pdf</a>	✓	⋮		
⋮	🔗	<a href="#">Tanenbaum, Andrew Stuart Austin, Todd Chandavarkar, B R Structured.pdf</a>	✓	⋮		
⋮	Apuntes históricos		✓	⋮		
⋮	🔗	<a href="#">01 - Representaciones Numéricas Parte 1 - Números Enteros.pdf</a>	✓	⋮		
⋮	🔗	<a href="#">02 - Representaciones Numéricas Parte 2 - Números Racionales.pdf</a>	✓	⋮		
⋮	🔗	<a href="#">03 - Operaciones Aritméticas y Lógicas.pdf</a>	✓	⋮		
⋮	🔗	<a href="#">04 - Almacenamiento de datos.pdf</a>	✓	⋮		

# Bibliografía

- <https://github.com/IIC2343/Syllabus-antiores/tree/main/Enunciados>



The screenshot displays the GitHub interface for the repository `IIC2343 / Syllabus-antiores`. The left sidebar shows the file tree with the `Enunciados` directory selected. The main content area shows the commit history for the `Enunciados` directory, listing various commits with their messages and dates.

**Files**





- main
- Go to file
- Apuntes (viejos)
- Ayudantías pasadas
- Enunciados**
  - Enunciados 2019-1
  - Enunciados 2019-2
  - Enunciados 2020-1
  - Enunciados 2021-1
  - Enunciados 2021-2
  - Enunciados 2022-1
  - Enunciados 2022-2
  - Enunciados 2023-1
  - Enunciados 2023-2
  - Enunciados 2024-1
  - Enunciados 2024-2
    - Compilado\_IIC2343.pdf
  - Slides 2019\*1
  - Slides 2019\*2
  - Slides 2020\*1
  - README.md

**Syllabus-antiores / Enunciados**

Geermy Delete Enunciados/Enunciados 2024-2/a eba5381 · 4 months ago History

Name	Last commit message	Last commit date
...		
Enunciados 2019-1	feat: add more solutions to previous exams; add previous auxiliary cl...	2 years ago
Enunciados 2019-2	feat: add more solutions to previous exams; add previous auxiliary cl...	2 years ago
Enunciados 2020-1	feat: add more solutions to previous exams; add previous auxiliary cl...	2 years ago
Enunciados 2021-1	add enunciados antiguos	2 years ago
Enunciados 2021-2	feat: add assessments for past semester	2 years ago
Enunciados 2022-1	feat: add assessments for past semester	2 years ago
Enunciados 2022-2	feat: add more solutions to previous exams; add previous auxiliary cl...	2 years ago
Enunciados 2023-1	feat: add assessments for past semester	2 years ago
Enunciados 2023-2	add missing test solutions for last semester	last year
Enunciados 2024-1	remove unused readme	10 months ago
Enunciados 2024-2	Delete Enunciados/Enunciados 2024-2/a	4 months ago
Compilado_IIC2343.pdf	Agregando compilado de Felipe	5 years ago

# Introducción del curso:

- Un computador lo definimos como una **máquina programable que ejecuta programas.**
- Para programar necesitamos:
  - Datos: números (enteros, reales) , texto, imágenes, etc 
  - Operaciones: suma, resta, multiplicación, división, etc 
  - Variables: simples, arreglos 
  - Control de flujo: comparaciones, manejo de ciclos 
- Para poder tener esta última parte requerimos de comenzar el capítulo de **¡programabilidad!**

# Clase 04 - Almacenamiento de datos

---

Profesor: **IIC2343 - Arquitectura de Computadores**  
- Felipe Valenzuela González  
Correo:  
frvalenzuela@alumni.uc.cl