
Programación estructurada

Informática I

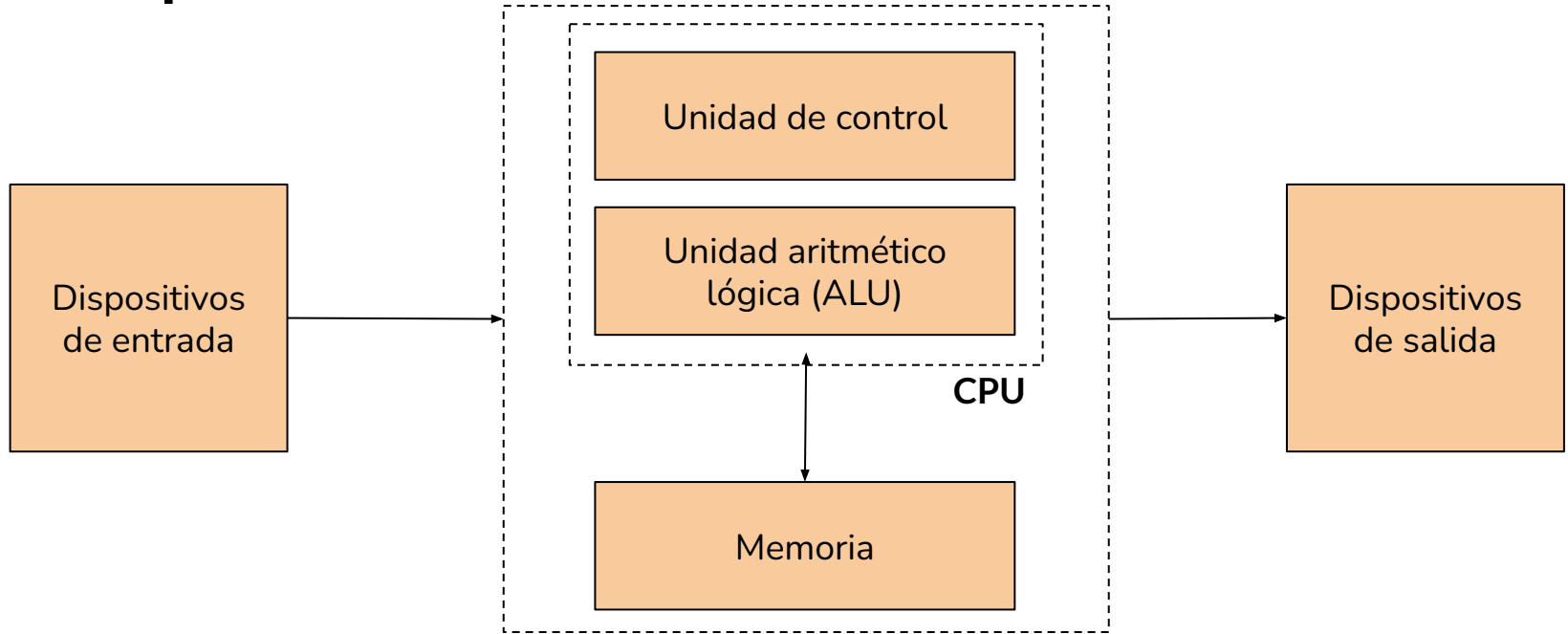
Instituto Universitario Aeronáutico
Universidad de la Defensa Nacional
(UNDEF)

Mg. Ing. Facundo S. Larosa

Arquitectura de una computadora

Arquitectura de un sistema de computadora

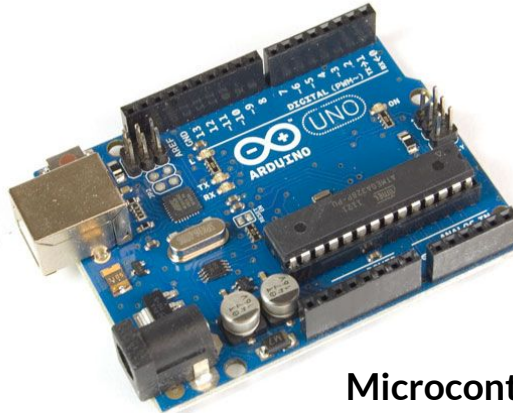
Arquitectura de Von Neumann



Arquitectura de un sistema de computadora



PC



Microcontrolador



PLC



Smartphone

Arquitectura de un sistema de computadora

Set de instrucciones

El set de instrucciones de un procesador es el conjunto completo de instrucciones que ese procesador puede ejecutar.

Cada procesador tiene su propio set de instrucciones con sus características propias:

- Nivel de complejidad
- Manejo de memoria
- Tamaño de las operaciones
- Extensiones

Arquitectura de un sistema de computadora

Set de instrucciones

Extracto de set de instrucciones de un microprocesador PIC

Operation		Mnemonic	Flags			Description
			Z	DC	C	
Add	Literal to W	addlw k	✓	✓	✓	Binary addition [W] <- [W] + #kk
	W to File	addwf f,d	✓	✓	✓	[d] <- [W] + [f]
Clear						Zeroes destination byte or bit
	File	clrf f	✓	•	•	[f] <- #00
	W	clrw	✓	•	•	[W] <- #00
	Bit	bcf f,n	•	•	•	[f _n] <- #0
Decrement						Subtract one, produce no borrow
	File	decf f,d	✓	•	•	[f] <- [f] - #01
Increment						Add one, produce no carry
	File	incf f,d	✓	•	•	[f] <- [f] + #01
Set						Sets any bit in a file to one
	Bit	bsf f,n	•	•	•	[f _n] <- #1
Subtract						Binary subtraction
	W from literal	sublw k	✓	✓	✓	[W] <- #kk - [W]
	W from File	subwf f,d	✓	✓	✓	[d] <- [f] - [W]

#0 Single zero bit

#00 Zero byte

#kk 8-bit constant

f_n Bit n of file

#1 Single one bit

#01 Byte 01h

n 3-bit bit specifier 0 - 7

Arquitectura de un sistema de computadora

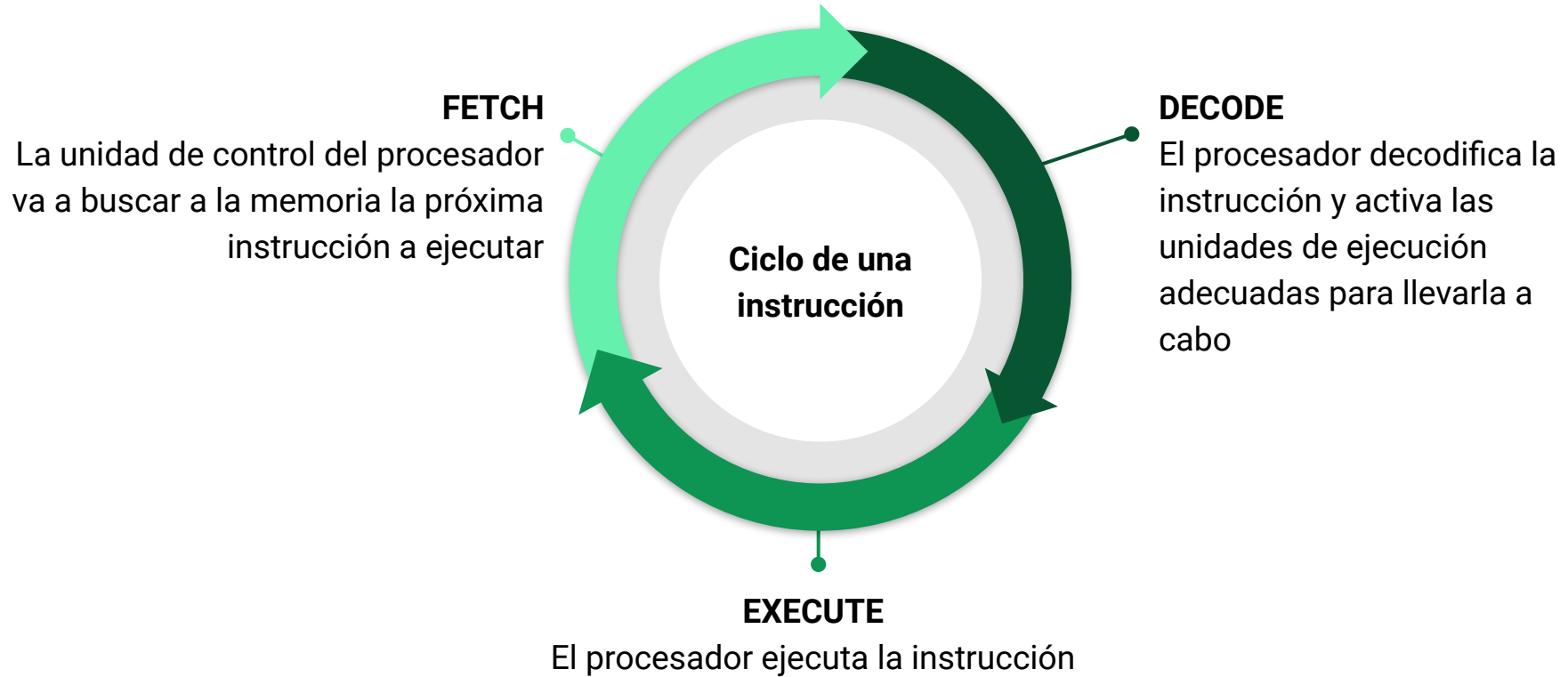
Set de instrucciones

Extracto de set de instrucciones de un microprocesador Cortex M

Mnemonic	Brief description	See
ADR	Generate PC-relative address	<i>ADR</i>
CLREX	Clear Exclusive	<i>CLREX</i>
LDM{mode}	Load Multiple registers	<i>LDM and STM</i>
LDR{type}	Load Register using immediate offset	<i>LDR and STR, immediate offset</i>
LDR{type}	Load Register using register offset	<i>LDR and STR, register offset</i>
LDR{type}T	Load Register with unprivileged access	<i>LDR and STR, unprivileged</i>
LDR	Load Register using PC-relative address	<i>LDR, PC-relative</i>
LDREX{type}	Load Register Exclusive	<i>LDREX and STREX</i>
POP	Pop registers from stack	<i>PUSH and POP</i>
PUSH	Push registers onto stack	<i>PUSH and POP</i>
STM{mode}	Store Multiple registers	<i>LDM and STM</i>
STR{type}	Store Register using immediate offset	<i>LDR and STR, immediate offset</i>
STR{type}	Store Register using register offset	<i>LDR and STR, register offset</i>
STR{type}T	Store Register with unprivileged access	<i>LDR and STR, unprivileged</i>
STREX{type}	Store Register Exclusive	<i>LDREX and STREX</i>

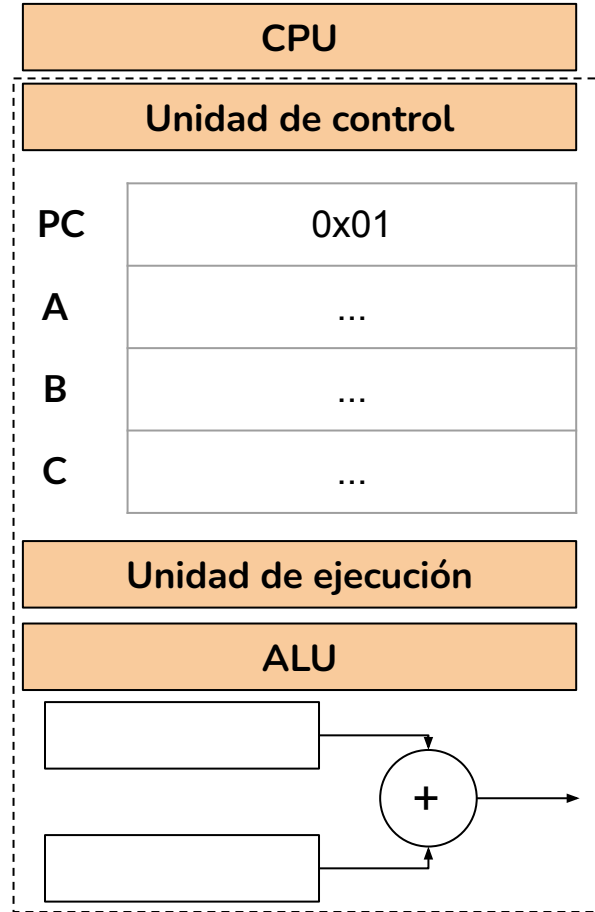
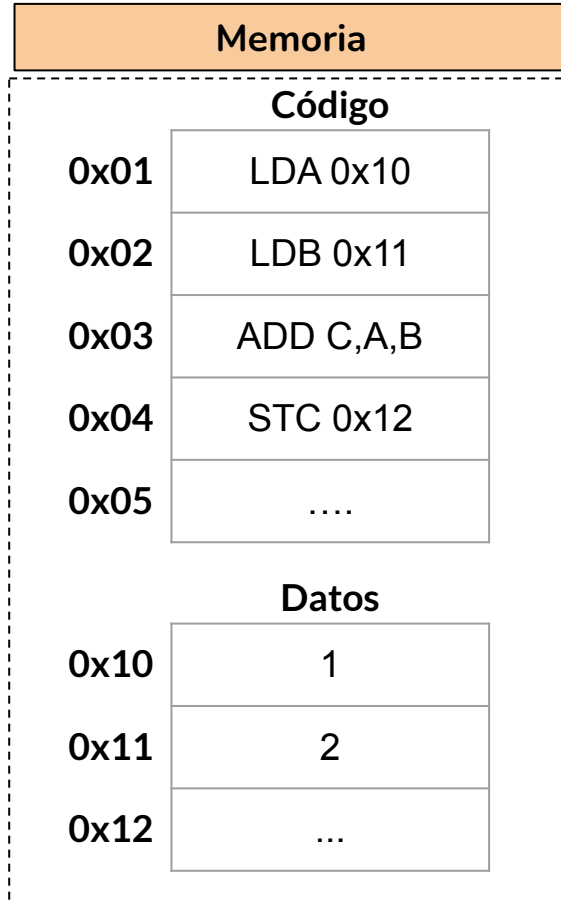
Arquitectura de un sistema de computadora

Ciclo de trabajo: fetch / decode / execute

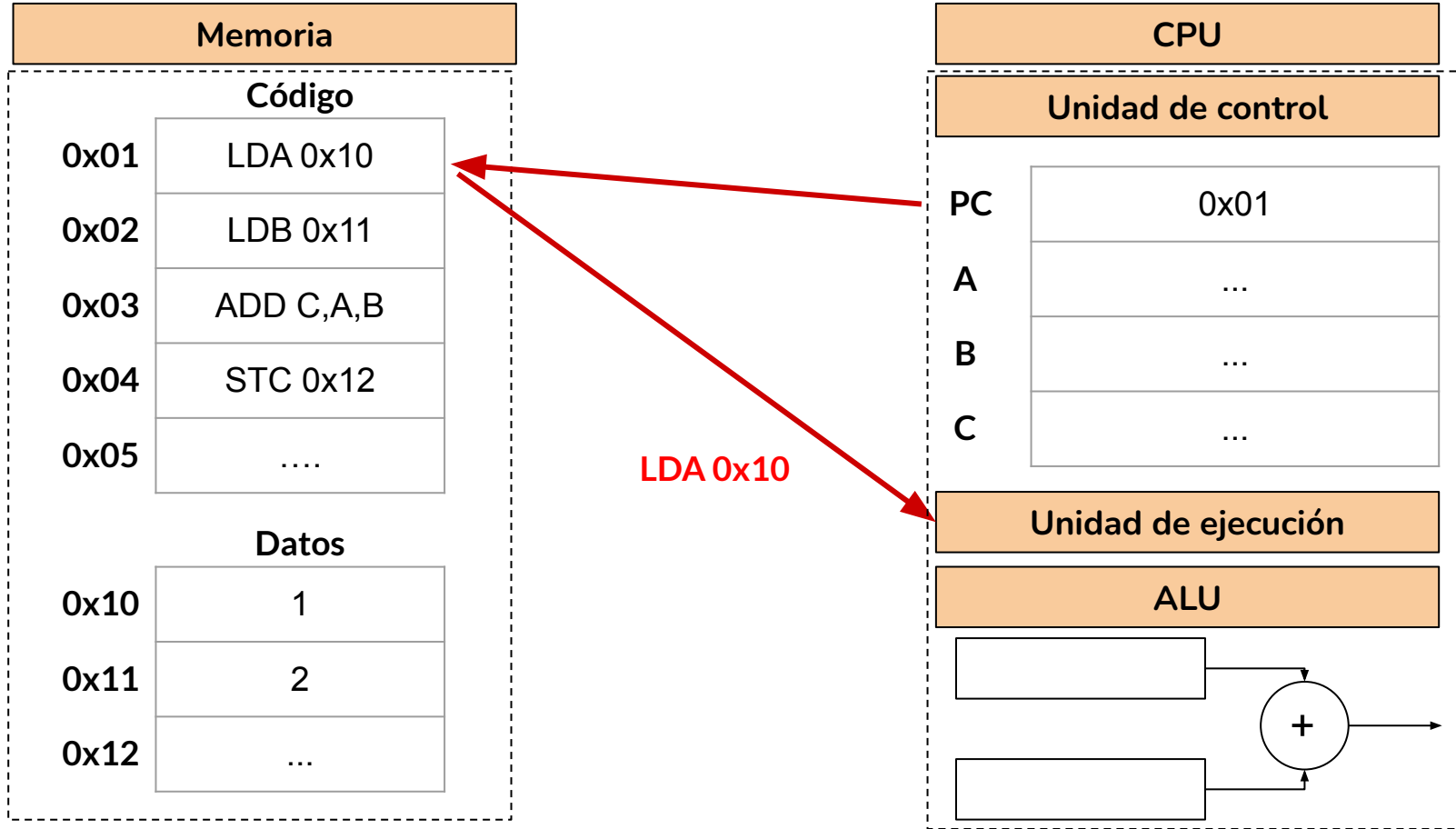


Arquitectura de un sistema de computadora

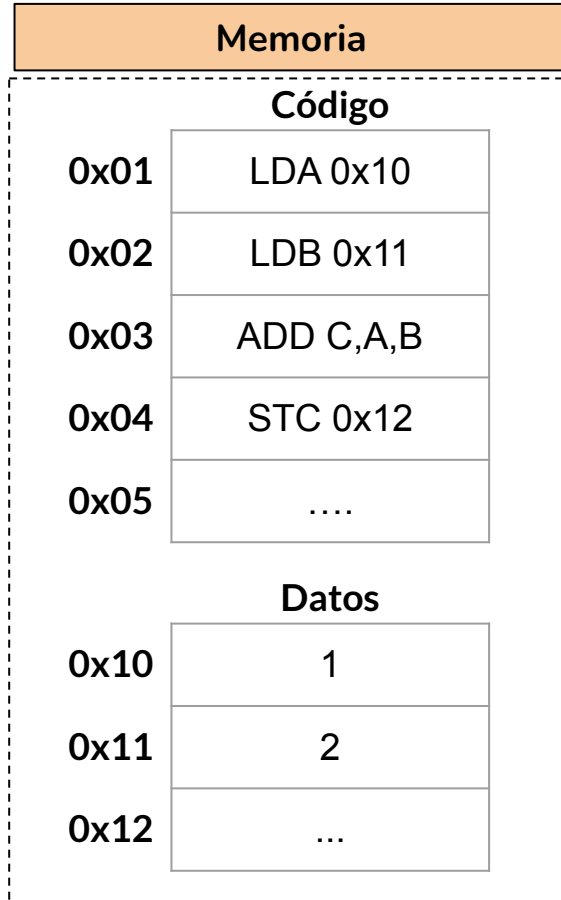
Ciclo de trabajo: fetch / decode / execute



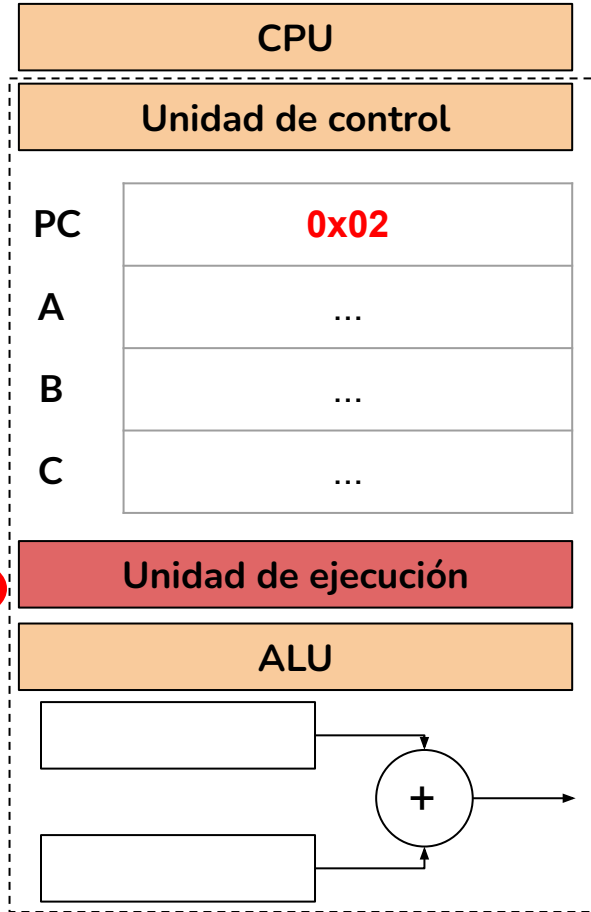
1. Fetch: LDA 0x10



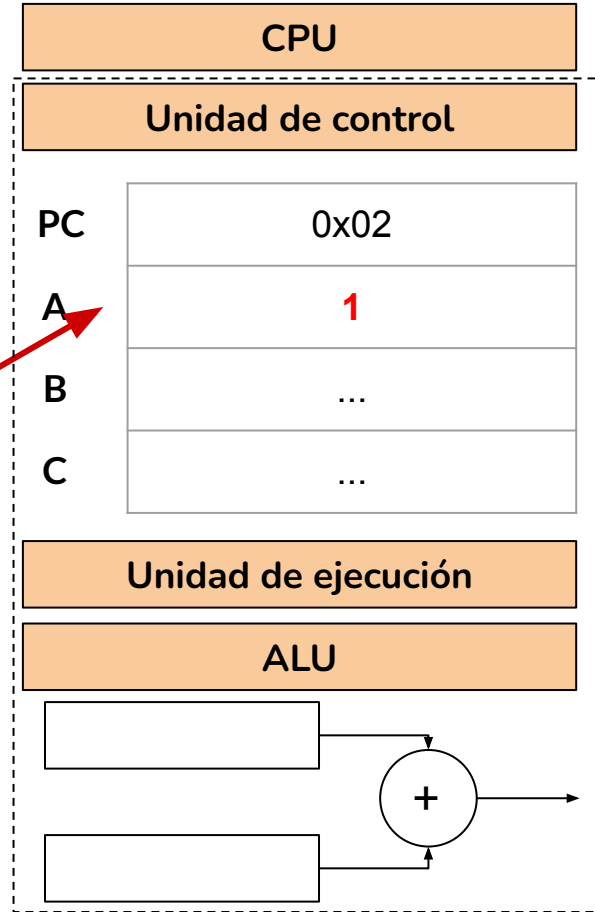
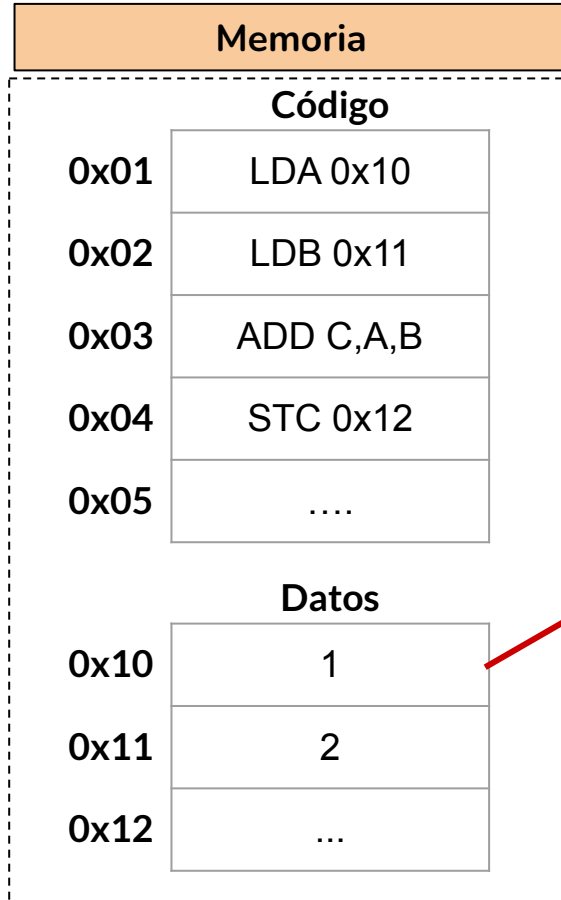
1. Decode: LDA 0x10



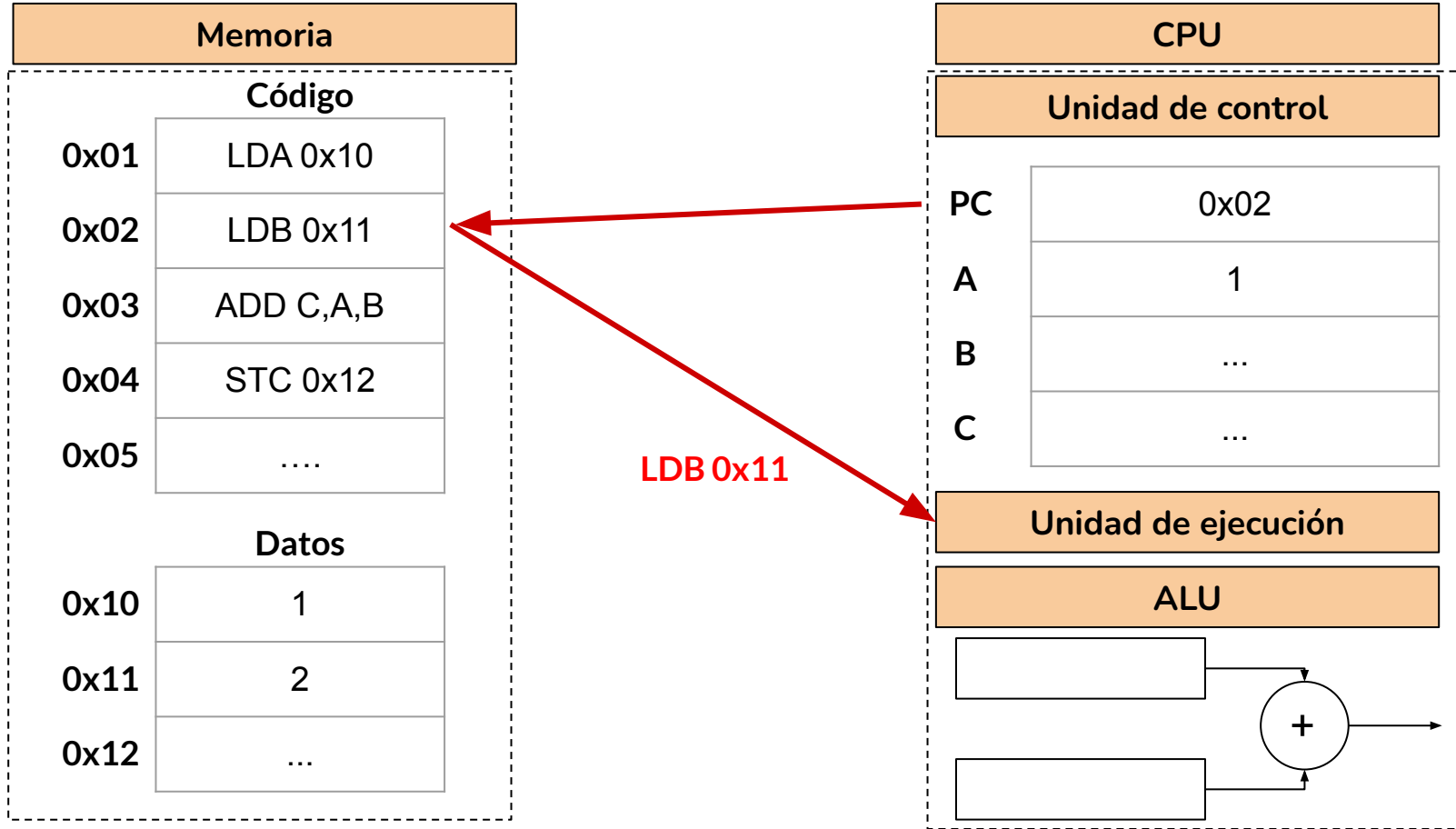
LDA 0x10



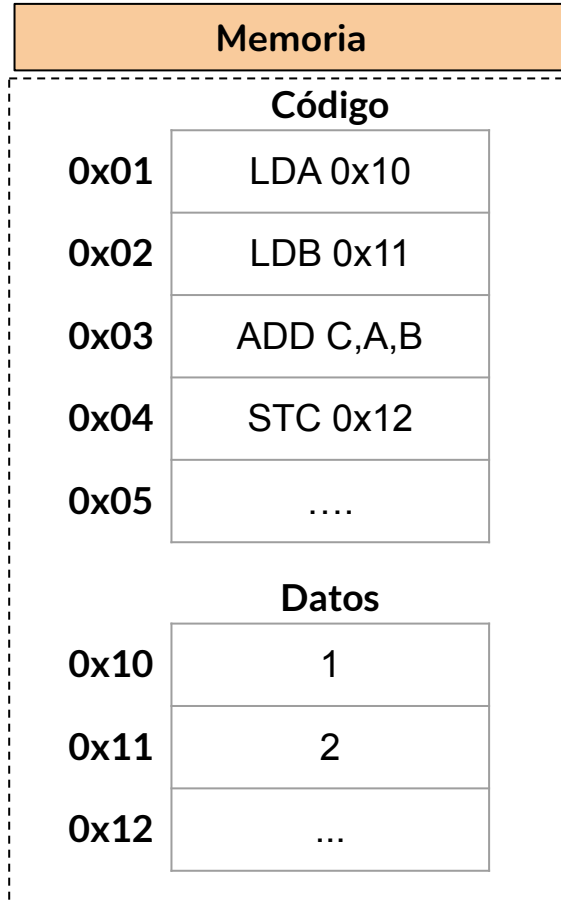
1. Execute: LDA 0x10



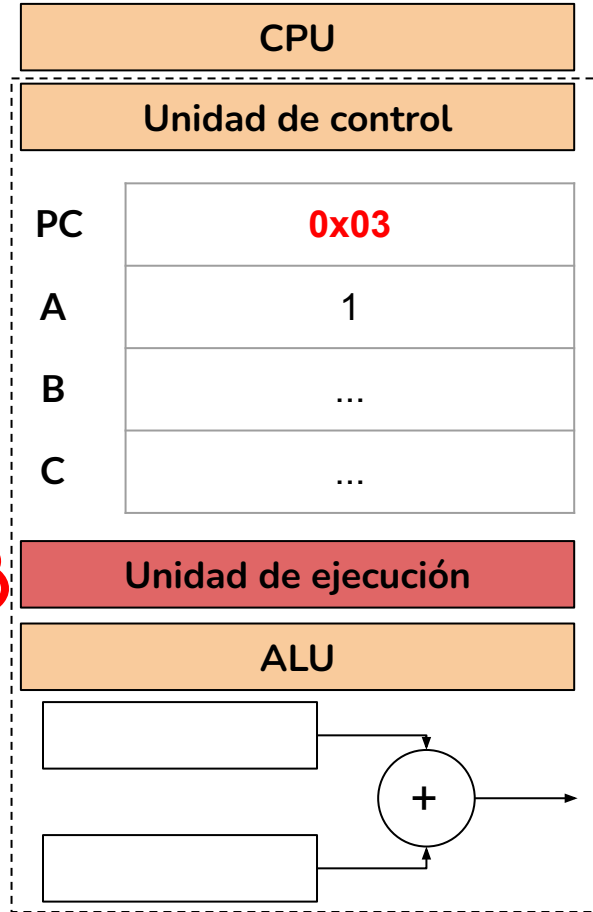
2. Fetch : LDB 0x11



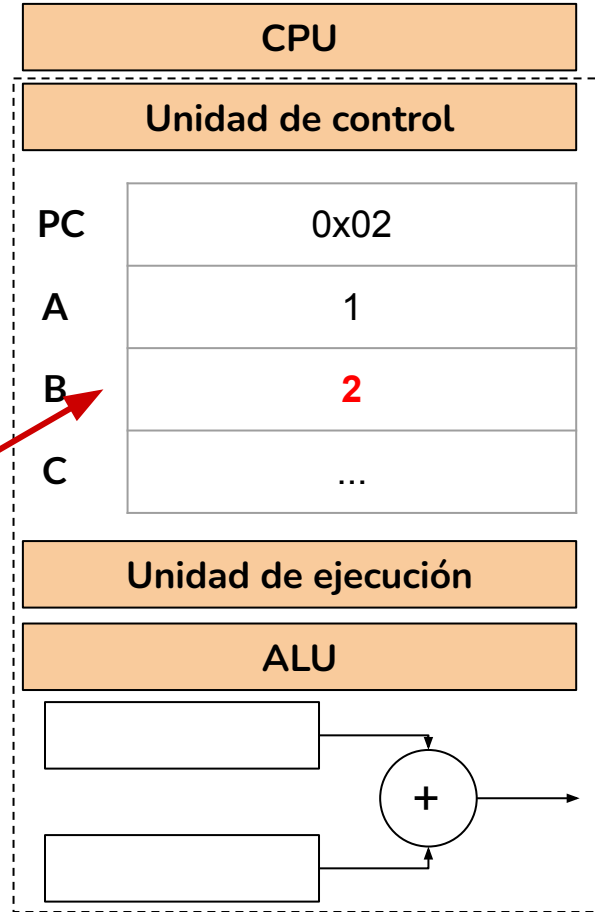
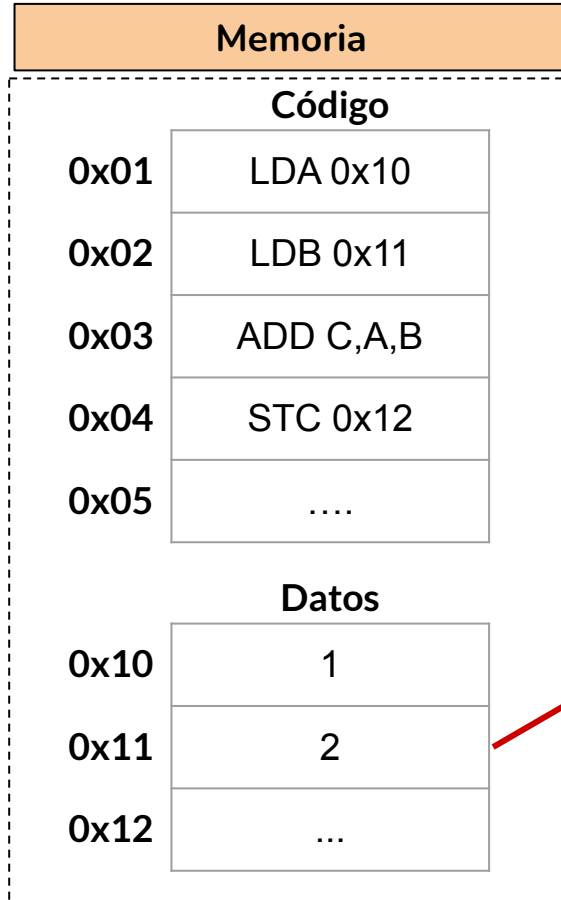
2. Decode: LDB 0x11



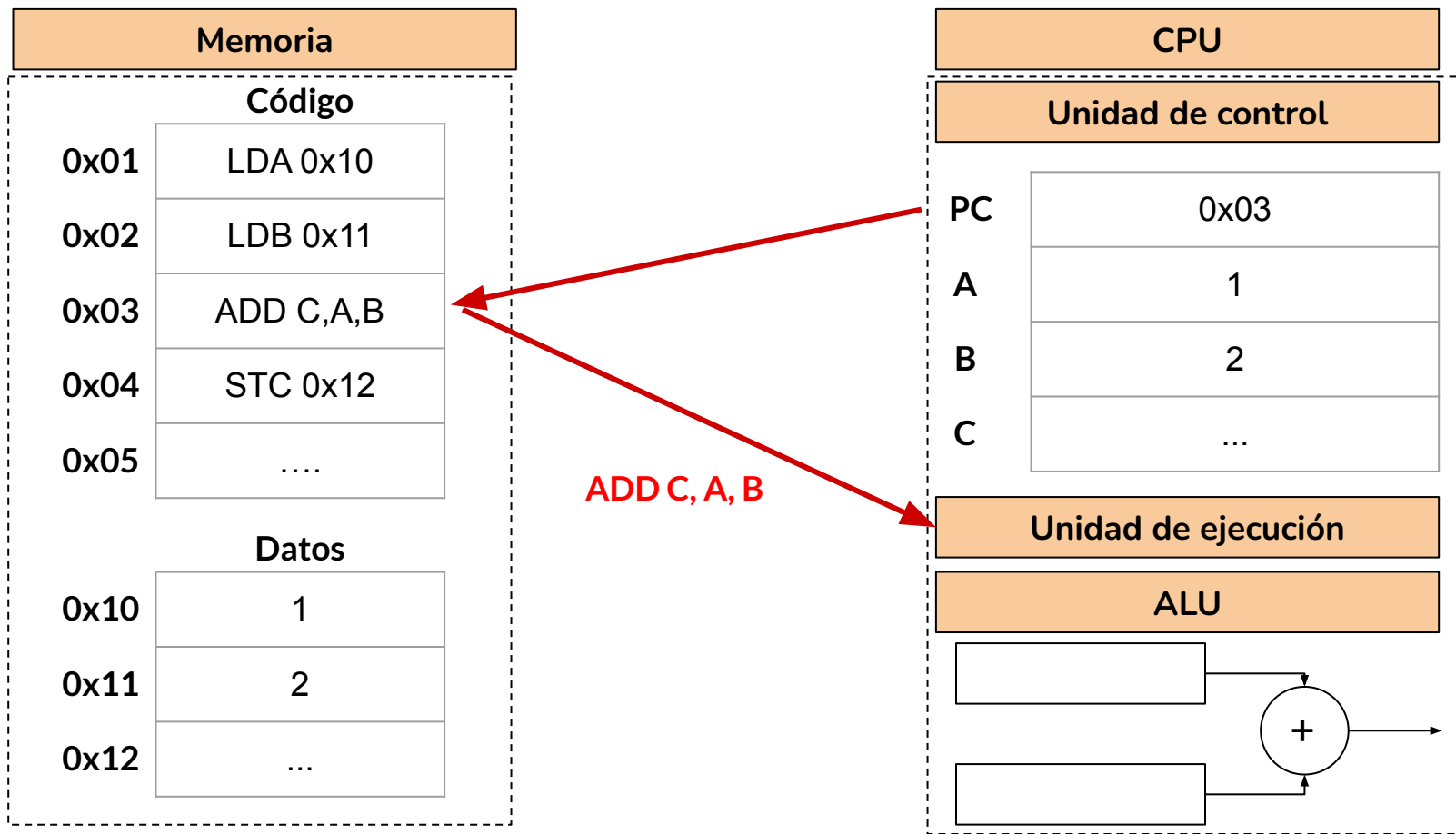
LDB 0x11



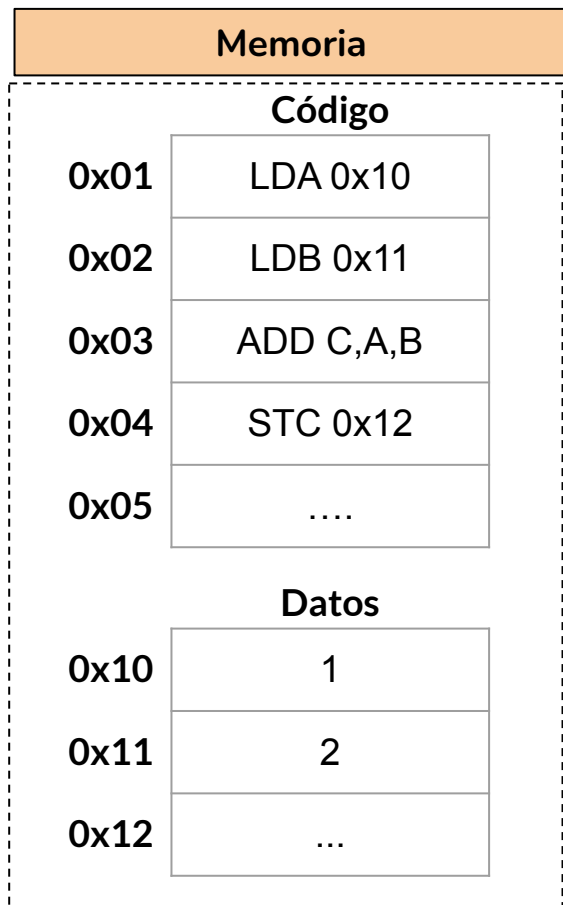
2. Execute: LDB 0x11



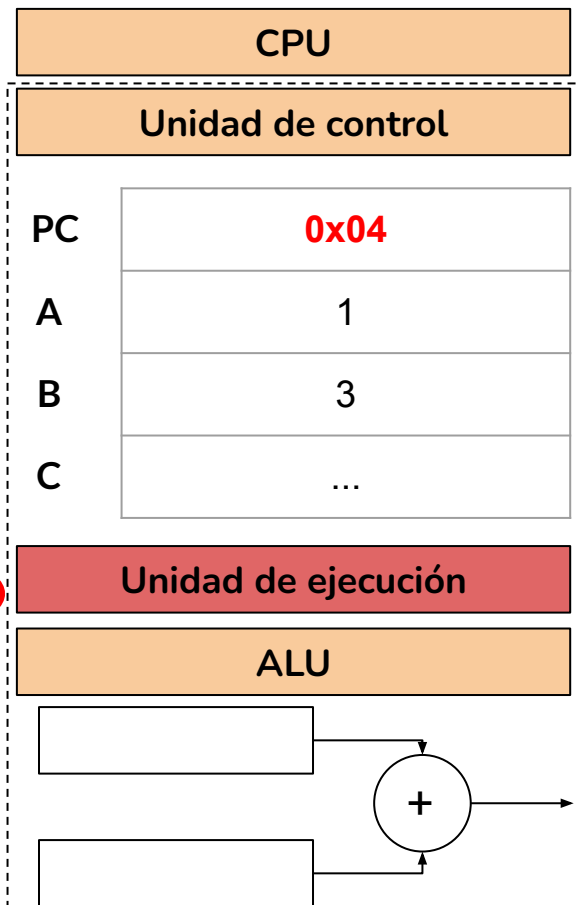
3. Fetch: add C,A,B



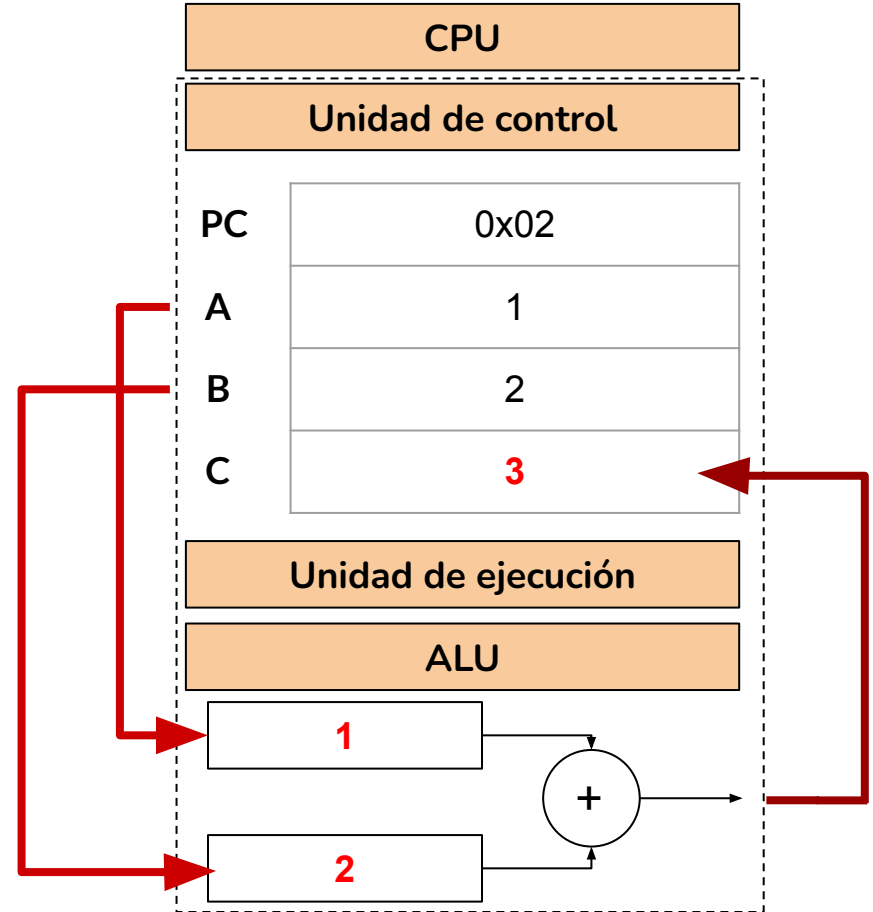
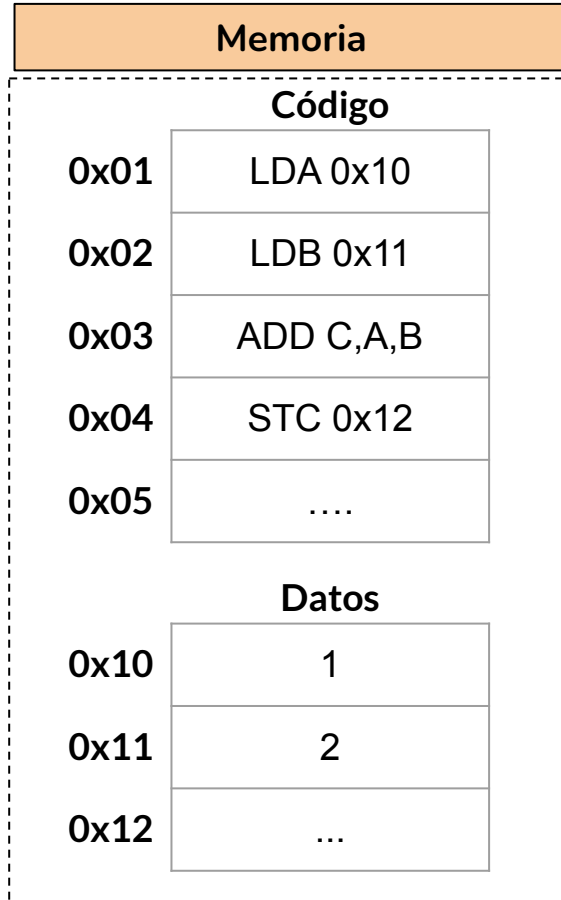
3. Decode: add C, A, B



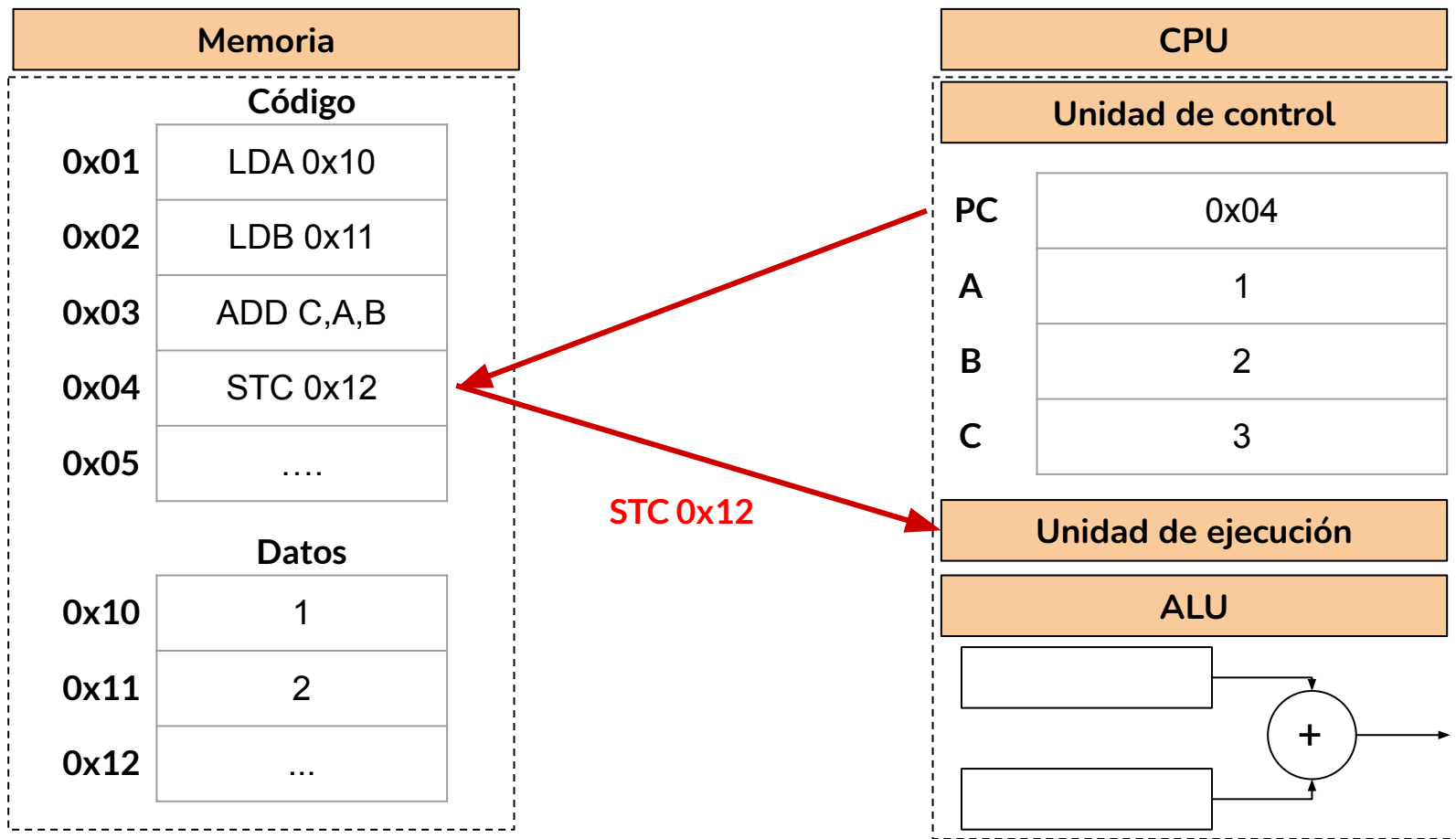
ADD C,A,B



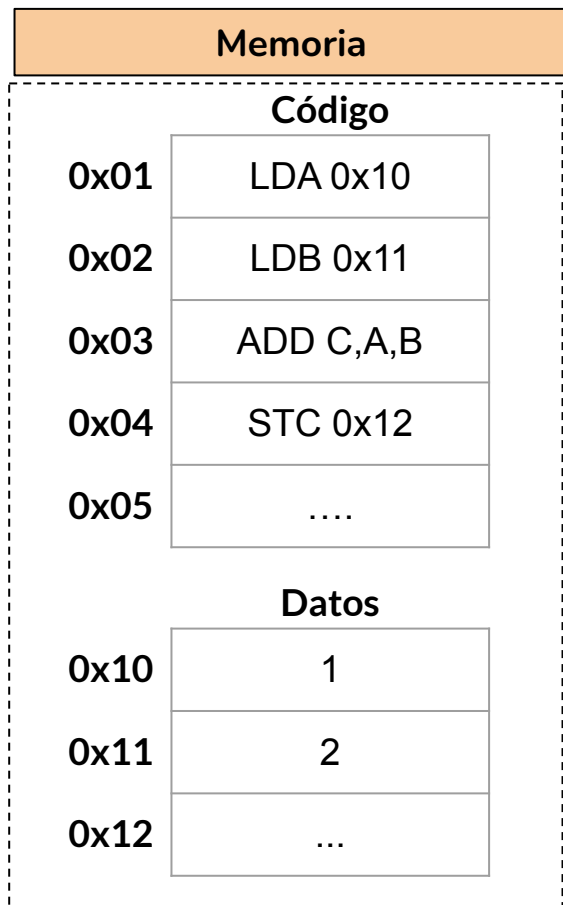
3. Execute: add C,A,B



4. Fetch: STC 0x12



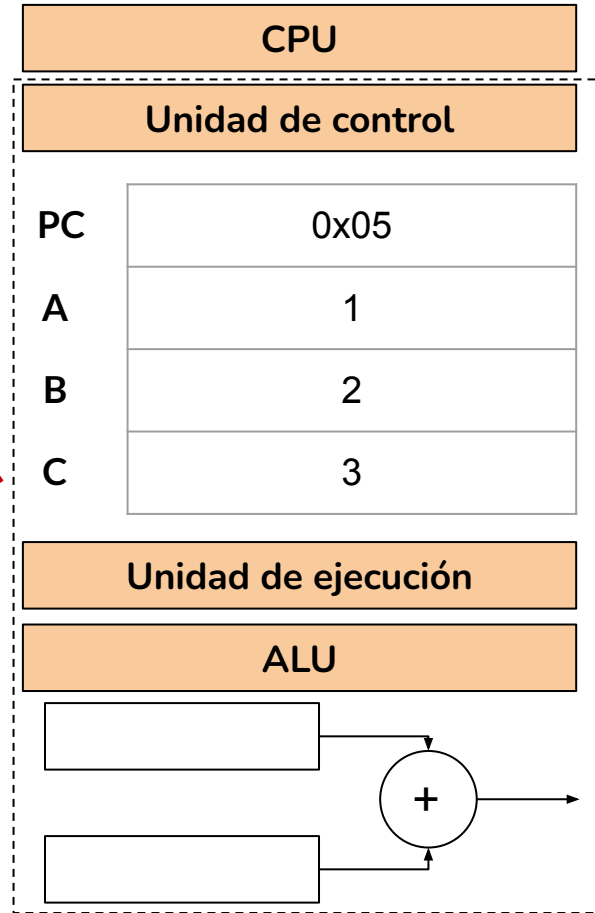
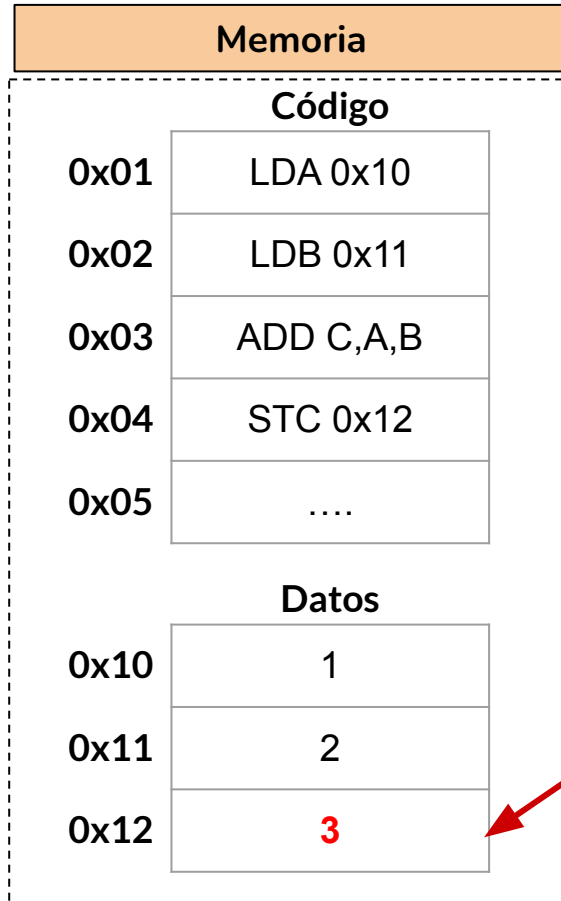
4. Decode: STC 0x12



STC 0x12



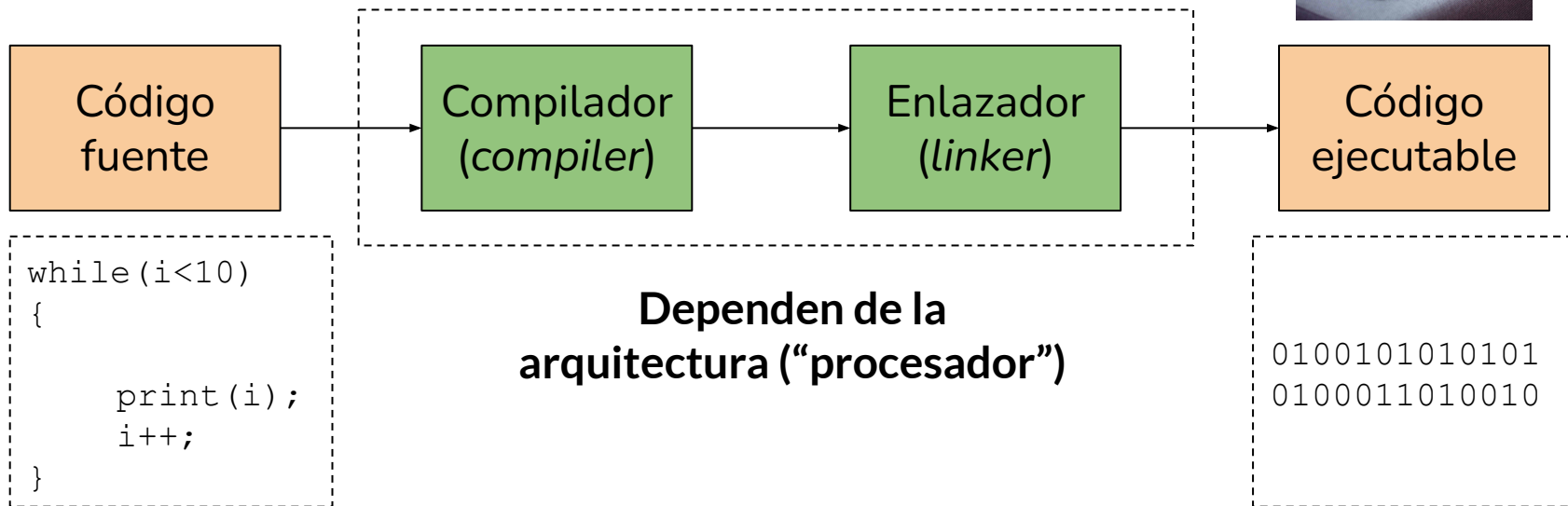
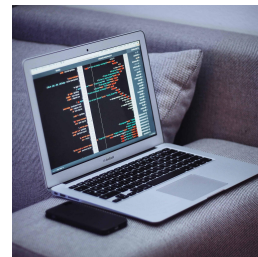
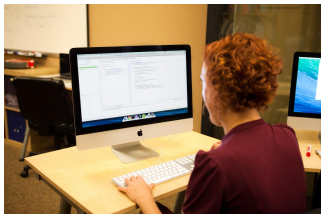
4. Execute: STC 0x12



Lenguajes compilados

- Desventajas de escribir un programa en código de máquina (*assembly*, lenguaje ensamblador):
 - Laborioso, tedioso, complicado
 - El código es **dependiente** del procesador

Por lo anterior, se desarrollaron lenguajes de programación que facilitan la tarea al programador y luego se traducen a código de máquina.



Diagramación estructurada

Algoritmo

Es un conjunto prescrito de instrucciones o reglas bien definidas, ordenadas y finitas que permiten llevar a cabo una actividad mediante pasos sucesivos.

Ejemplos:

- Receta de cocina
 - Instrucciones del Google Maps para llegar de un lugar a otro
 - Libro de yoga
-

¿Qué es la programación estructurada?

- Es un paradigma de programación destinado a mejorar la claridad, legibilidad, calidad y tiempo de desarrollo de programas de computadora
 - Se inicia a fines de los años 50 y se consolida hacia principios de los años 70
 - Es de interés para nosotros ya que la gran mayoría de los lenguajes modernos se basan en este modelo
-

Teorema de la programación estructurada

Todo algoritmo que posea un único punto de entrada y un único punto de salida puede describirse por medio de tres estructuras lógicas:

- **Secuencia:** ejecución de una instrucción tras otra
 - **Selección:** ejecución de una u otra instrucción dependiendo de un criterio lógico
 - **Iteración:** repetición de un conjunto de instrucciones mientras se cumpla un determinado criterio lógico
-

Teorema de la programación estructurada

Uso de las sentencias **goto**



For a number of years I have been familiar with the observation that the quality of programmers is a decreasing function of the density of **go to** statements in the programs they produce. More recently I discovered why the use of the **go to** statement has such disastrous effects, and I became convinced that the **go to** statement should be abolished from all “higher level” programming languages (i.e. everything except, perhaps, plain machine code).

E. Dijkstra “*Go to statement considered harmful*” (1968)

Técnicas de diagramación básicas

La diagramación de la arquitectura de un programa es una de las ramas de la ingeniería de software. Nosotros vamos a usar algunas técnicas básicas para ayudarnos:

- Diagramas de Chapin
- Diagramas de flujo o flujograma
- Pseudocódigo

En el apéndice se encuentra más información para los que quieran ampliar sobre el tema.

1. Secuencia

- Es la ejecución de una instrucción tras otra de forma secuencial

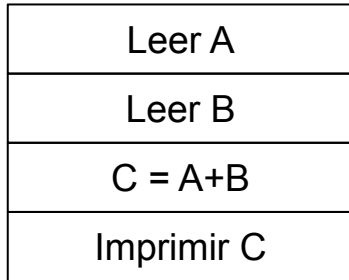
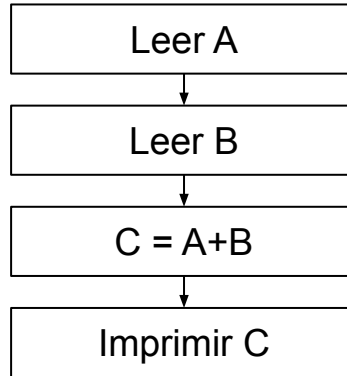
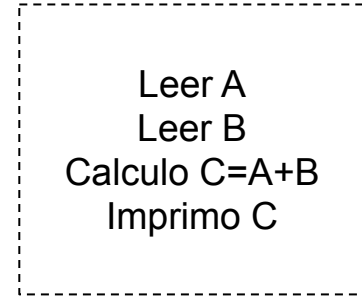


Diagrama Chapin



Flujograma



Pseudocódigo

2. Selección

- Es la ejecución de una u otra instrucción dependiendo de un criterio lógico

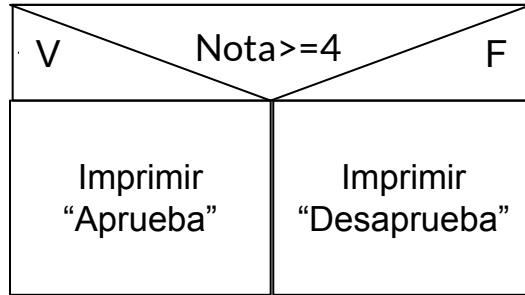
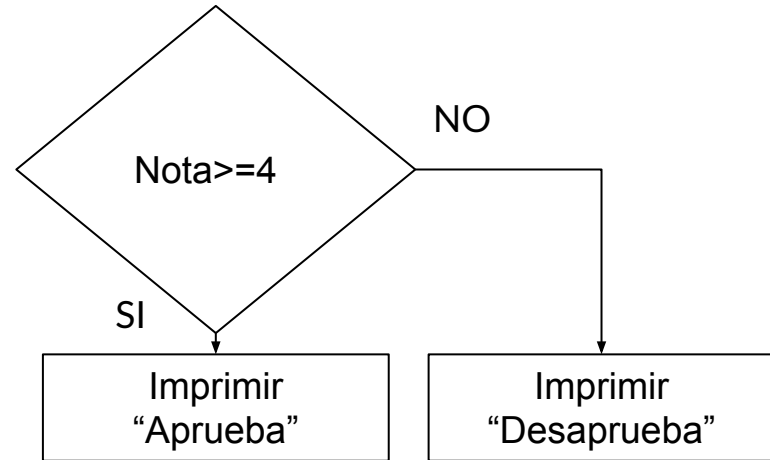


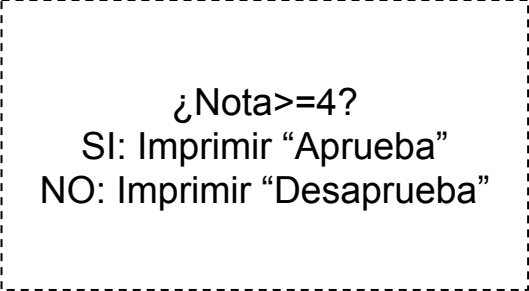
Diagrama Chapin



Flujograma

2. Selección

- Es la ejecución de una u otra instrucción dependiendo de un criterio lógico

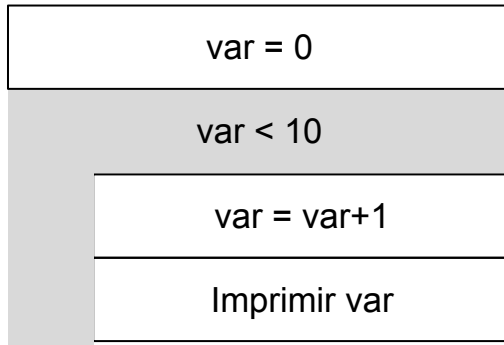


¿Nota \geq 4?
SI: Imprimir “Aprueba”
NO: Imprimir “Desaprueba”

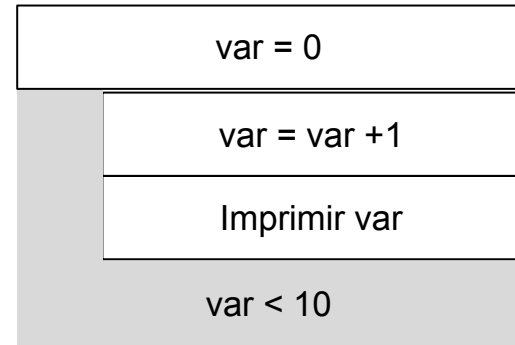
Pseudocódigo

3. Iteración

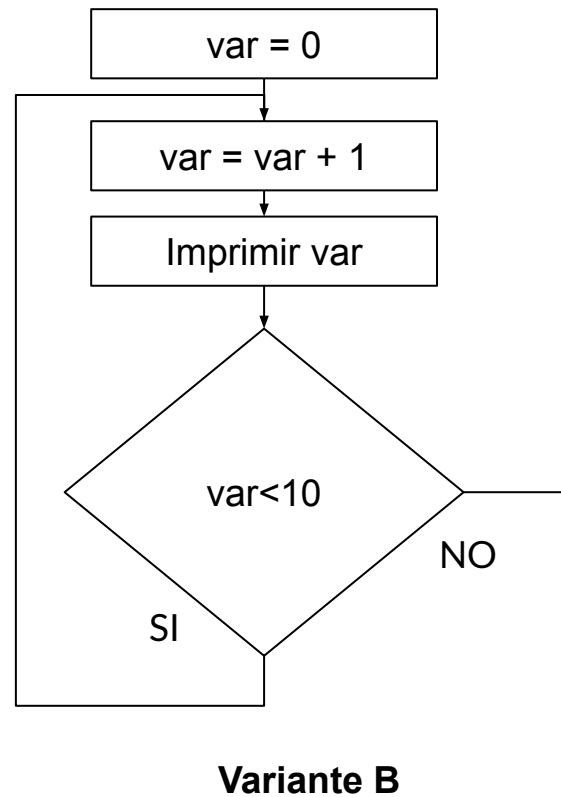
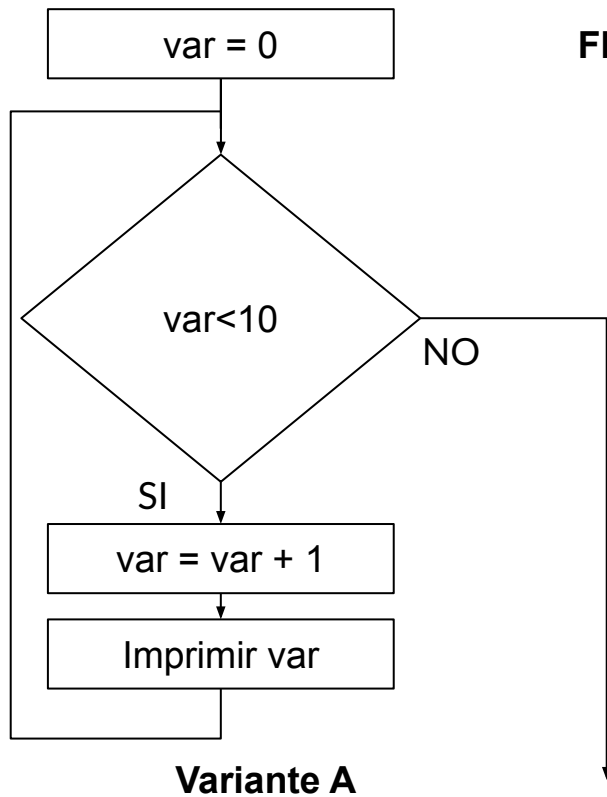
- Es la repetición de un conjunto de instrucciones mientras se cumpla un determinado criterio lógico



Variante A



Variante B



3. Iteración

Pseudocódigo

```
var = 0
MIENTRAS (var<10)
{
    var = var + 1
    Imprimir var
}
```

Variante A

```
var = 0
{
    var = var + 1
    Imprimir var
} MIENTRAS (var<10)
```

Variante B

Hagamos algunos ejercicios...

- Ingresar 100 números enteros, al finalizar, mostrar la suma y el promedio
 - Ingresar 100 valores enteros y mostrar si está el número 25. Decir en que posición se encuentra dentro del conjunto empezando con la posición 1. Avisar si no se encuentra.
 - Ingresar una cantidad desconocida de números enteros entre 1 y 100. Finalizar cuando se ingresa un 0. Mostrar cuántos números se ingresaron sin contar el 0
 - Ingresar 70 números naturales. Determinar e informar la cantidad de valores pares del conjunto
-

Bibliografía

- Para profundizar en los lenguajes de modelado de software
 - Fowler, Scott, “Distilled UML” (en inglés)
 - Rumbaugh, Jacobson, Boock, “The UML Reference Language” (en inglés)
 - Para leer sobre diagramación con diagramas de Chapin
 - Argibay, “C para Ingeniería Electrónica”
 - Sobre el uso del goto
 - <https://homepages.cwi.nl/~storm/teaching/reader/Dijkstra68.pdf>
-