

TEMA 1. SISTEMAS INFORMÁTICOS. ESTRUCTURA FUNCIONAL

Fundamentos de Hardware
1º ASIR

ÍNDICE

1. Definiciones
2. Historia de la computación
3. Arquitectura de ordenadores
4. Puertas lógicas
5. Circuitos integrados
6. Conceptos básicos
7. Práctica

1. DEFINICIONES

INFORMÁTICA: Ciencia que estudia el tratamiento automático de la información
Información + Automática

SISTEMA INFORMÁTICO: Herramienta que permite el tratamiento automático de la información facilitando su organización, proceso, transmisión y almacenamiento

Su objetivo es dar soporte al procesado, almacenamiento y entrada y salida de datos

1. DEFINICIONES

ELEMENTOS DE UN SISTEMA INFORMÁTICO

HARDWARE: Elementos físico eléctrico-electrónicos que componen el ordenador

SOFTWARE: Programas que ponen en funcionamiento el hardware del ordenador.

RECURSOS HUMANOS

S.I.= HARDWARE + SOFTWARE + RECURSOS HUMANOS

1. DEFINICIONES

ESTRUCTURA FUNCIONAL: Funciones de los componentes, cómo interactúan unos con otros para que el sistema funcione.

Distintas arquitecturas, p.e arquitectura de Von Neumann.

TEMA 1

ESTRUCTURA FÍSICA: Cómo es físicamente, qué componentes lo constituyen.

Como son físicamente, para que sirven, y que características tienen los diferentes componentes actuales de un PC.

TEMA 2

ENSAMBLAJE: Cómo colocar correctamente todos los componentes para que el sistema informático funcione.

TEMA 3

1. DEFINICIONES

PROGRAMA: Conjunto de instrucciones u órdenes agrupadas de forma adecuada que son ejecutadas de forma consecutiva.

SISTEMA OPERATIVO: Software de un sistema informático capaz de hacer que los programas procesen información sobre los componentes electrónicos de un ordenador o sistema informático. Gestión de procesos, memoria, E/S, ...

FIRMWARE: Parte intangible de componentes del hardware (software incrustado en el hardware).

2. HISTORIA DE LA COMPUTACIÓN

1ª Generación (1940-1960):

ENIAC – Válvulas de vacío
UNIVAC
Lenguaje máquina



2ª Generación (1960-1965):

Transistor
Lenguajes de alto nivel



3ª Generación (1965-1975):

Circuitos integrados
Primeros sistemas operativos
Lenguajes de alto nivel



4ª Generación (1975-1990):

Microprocesadores
Memoria de semiconductores



5ª Generación (1990-Hoy):

Aumento de la integración
Nuevas arquitecturas

3. ARQUITECTURA DE ORDENADORES

MÁQUINA DE TURING

Modelo de máquina diseñado por Alan Turing entre 1935 y 1945.

Permitía resolver, en teoría, cualquier problema matemático siempre y cuando se pudiera reducir a un algoritmo.

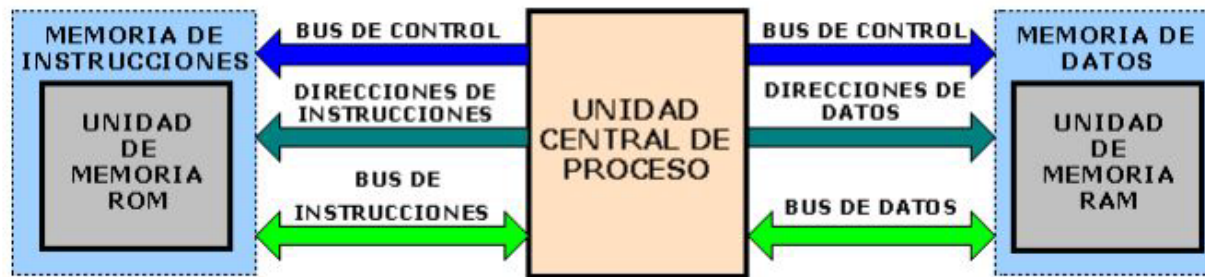
Contaba con una memoria, un cabezal de lectura-escritura y un procesador

Funciona de la siguiente forma: para realizar el cálculo de $f(x)$, el dato de entrada se guarda en memoria, se realizan los pasos para resolver la función, también guardados en memoria, y se muestra el resultado

3. ARQUITECTURA DE ORDENADORES

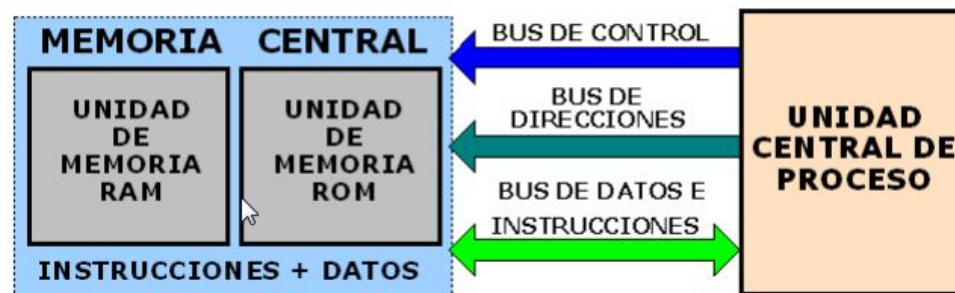
ARQUITECTURA HARVARD

La memoria de datos está separada de la memoria del programa. Permite acceder simultáneamente a las dos memorias.



ARQUITECTURA VON NEUMANN

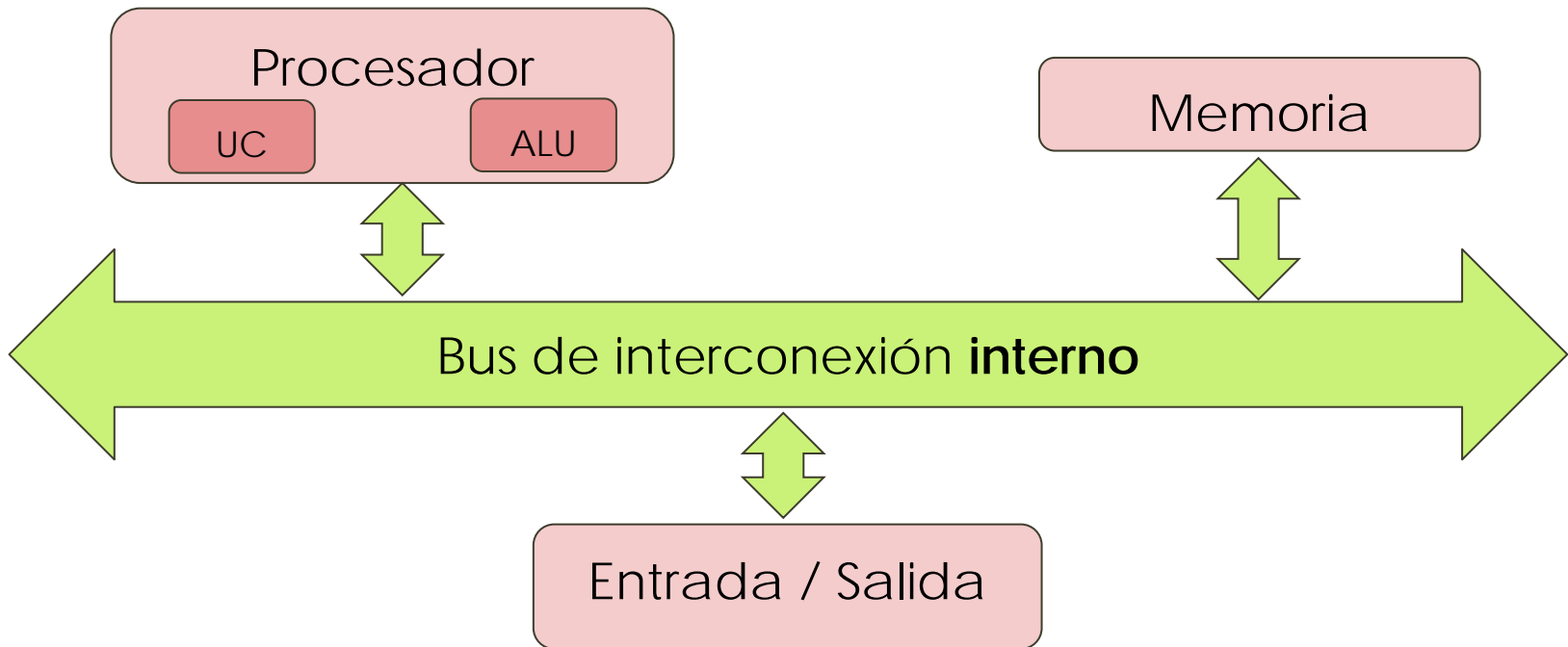
Mismo dispositivo de almacenamiento para datos e instrucciones



3. ARQUITECTURA DE ORDENADORES

ARQUITECTURA VON NEUMANN

La arquitectura de John von Neumann es la que se utiliza en la mayoría de los ordenadores actuales.



3. ARQUITECTURA DE ORDENADORES

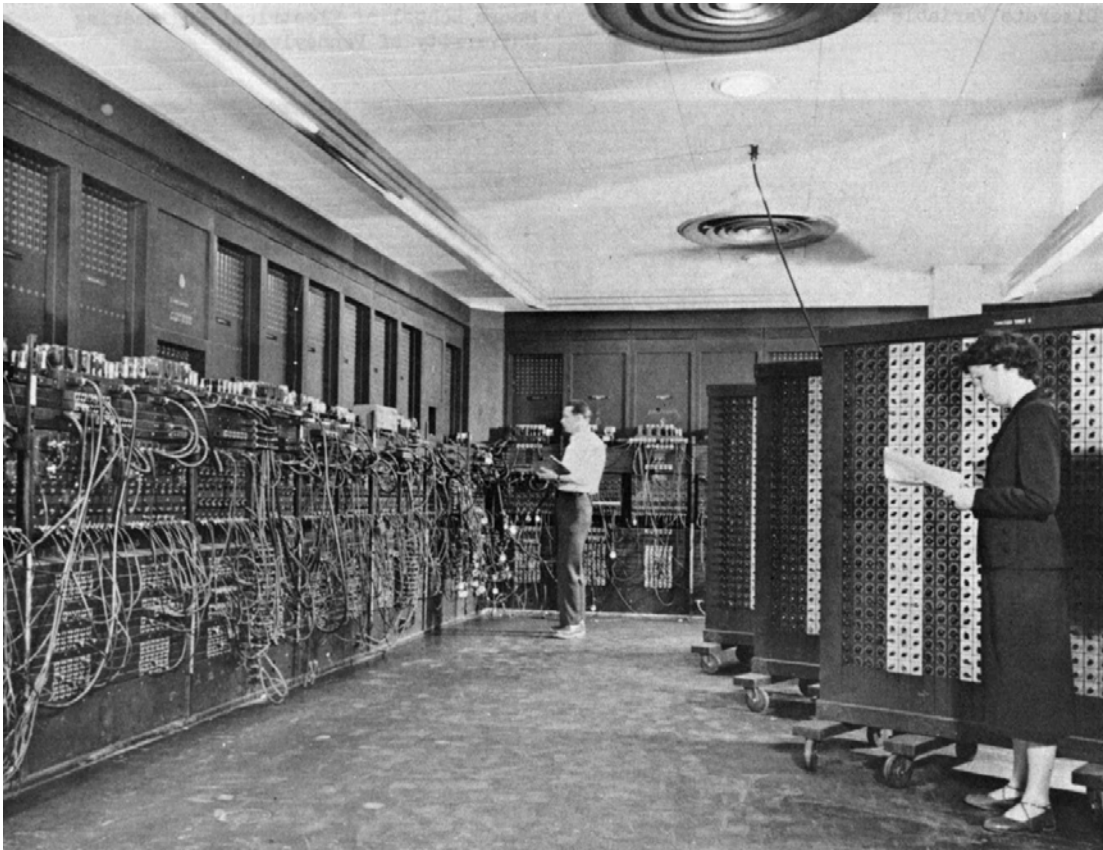
ARQUITECTURA VON NEUMANN

Es responsable del concepto de "programa almacenado en memoria".

Esta arquitectura surge a raíz de la colaboración de Von Neumann en el proyecto ENIAC (primer computador basado en válvulas de vacío, 1946)

El primer computador comercial, el UNIVAC I (1951), se construyó también teniendo en cuenta la arquitectura de von Neumann.

3. ARQUITECTURA DE ORDENADORES



ENIAC – 1946
Ocupaba 167 m²
Pesaba 27 toneladas
Temperatura de 50°
Consumo de 160 KW

3. ARQUITECTURA DE ORDENADORES

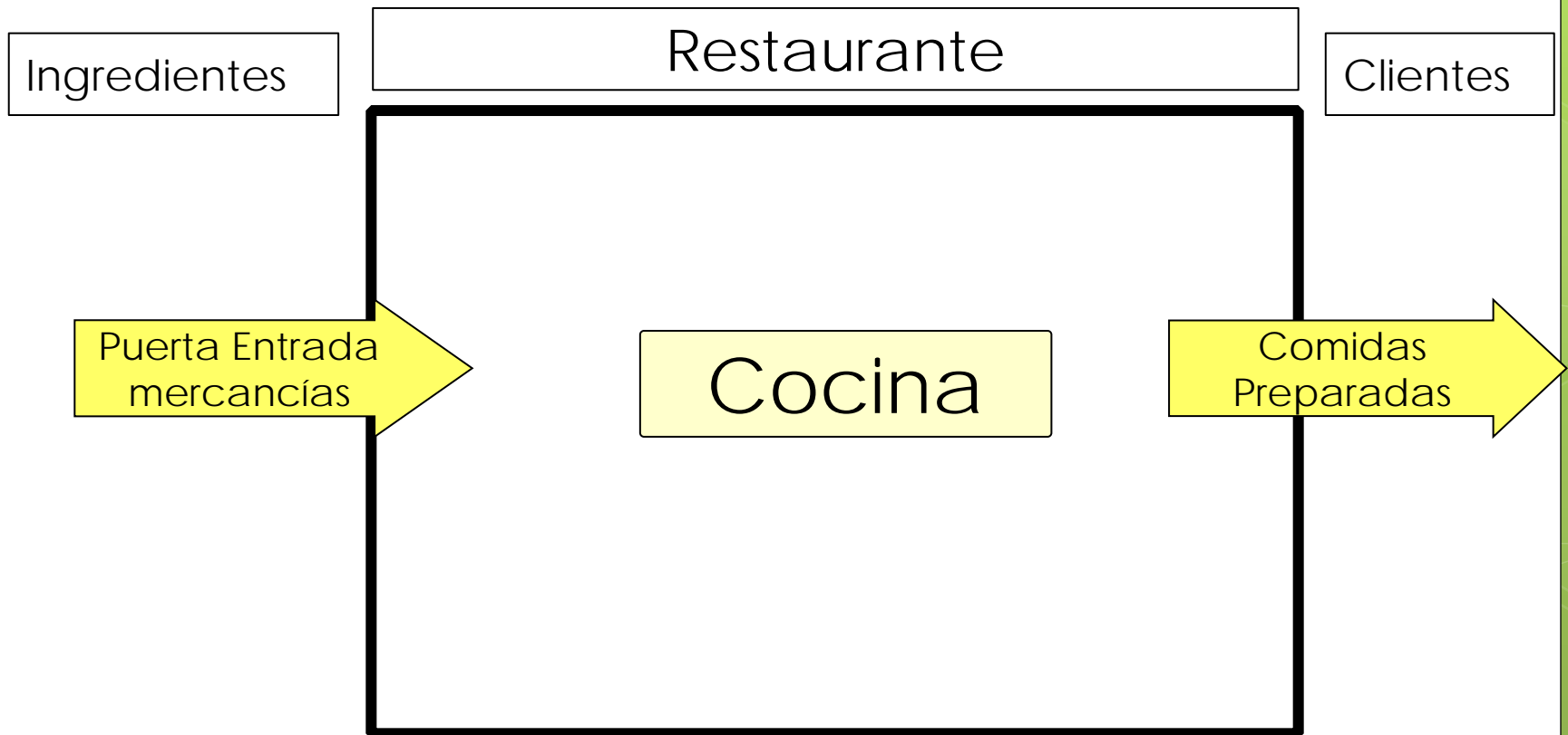


UNIVAC- 1951
Pesaba 7 toneladas
Coste de 1 millón de
dólares



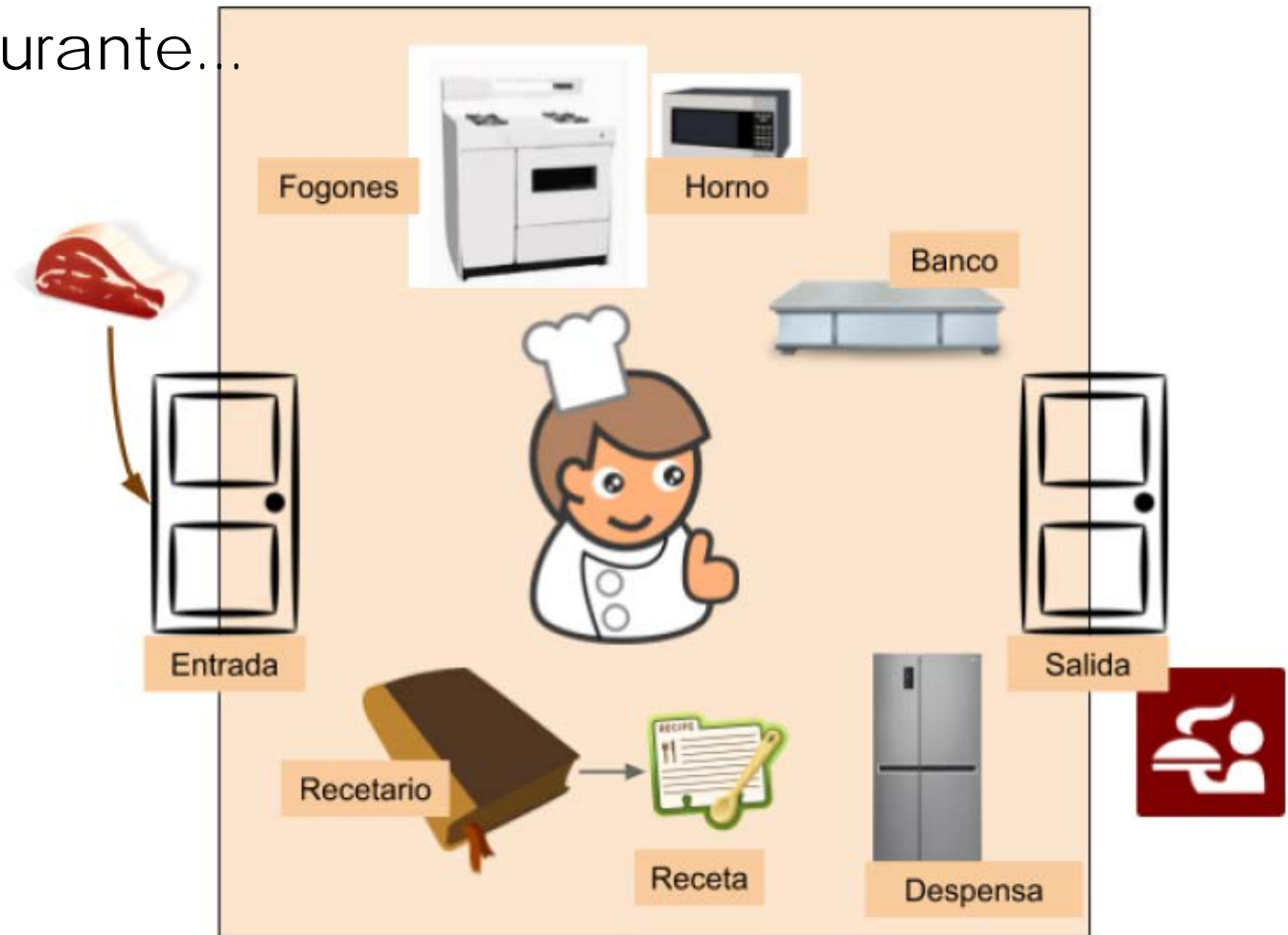
3. ARQUITECTURA DE ORDENADORES

En un restaurante...



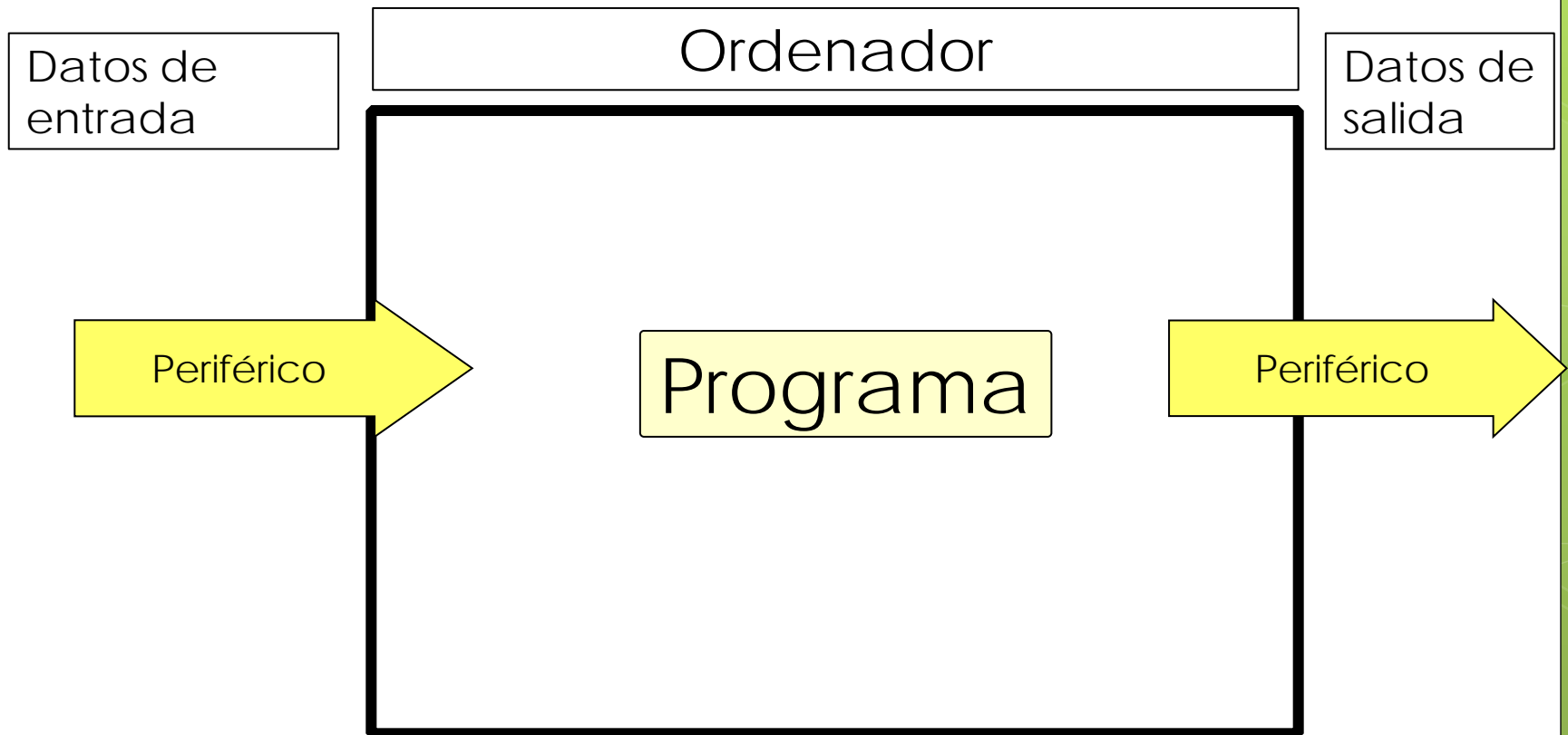
3. ARQUITECTURA DE ORDENADORES

En un restaurante...



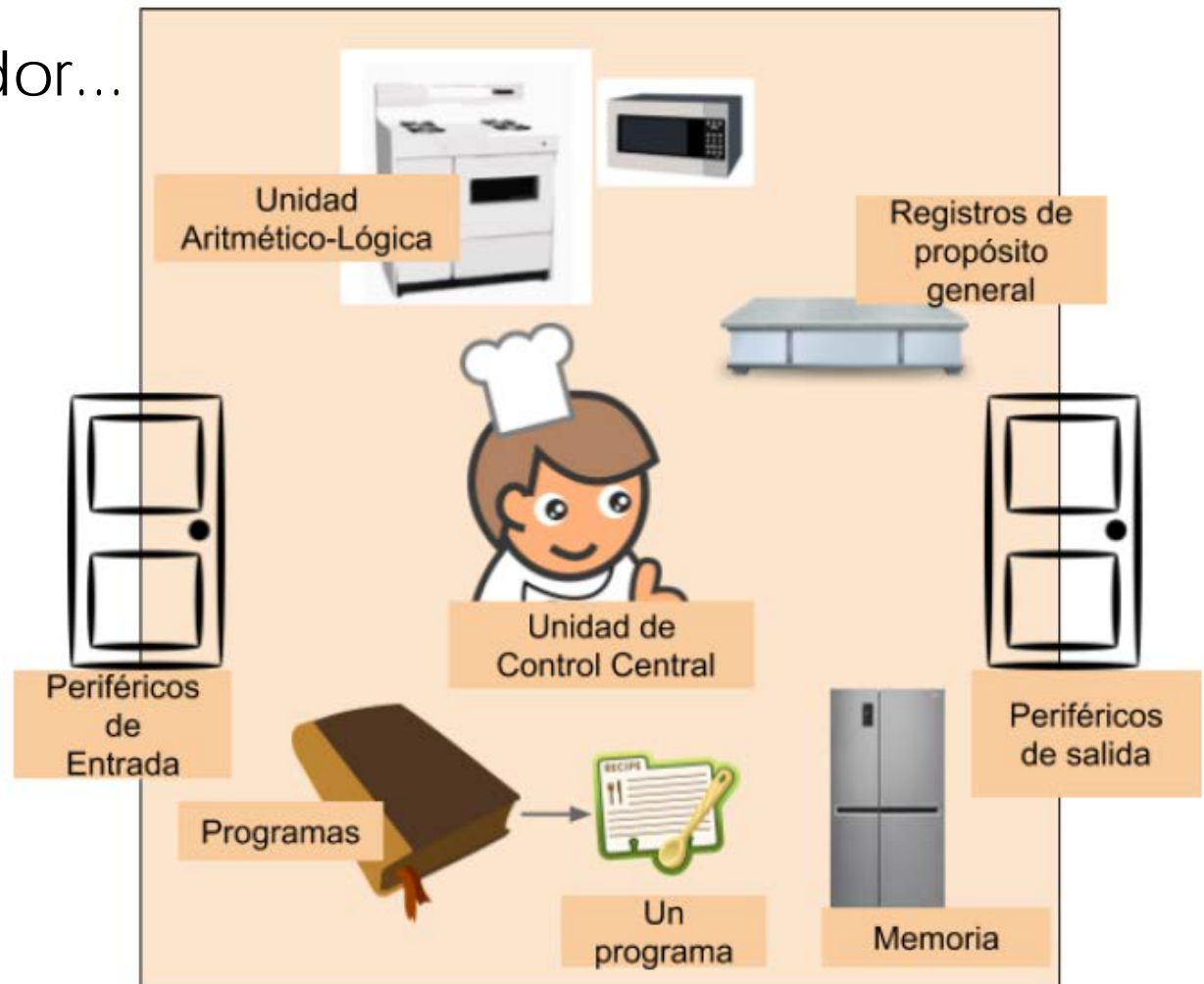
3. ARQUITECTURA DE ORDENADORES

En un ordenador...



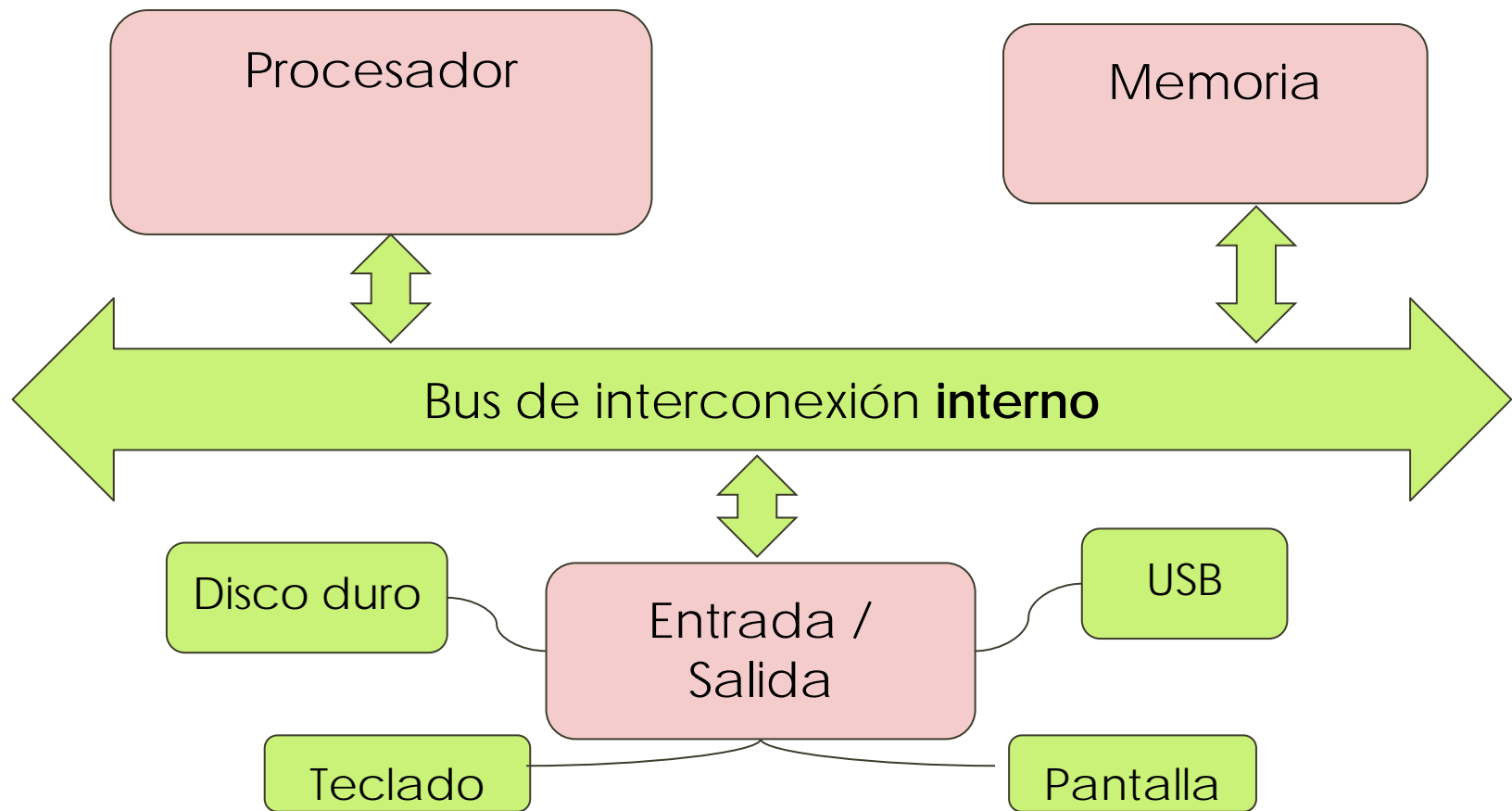
3. ARQUITECTURA DE ORDENADORES

En un ordenador...



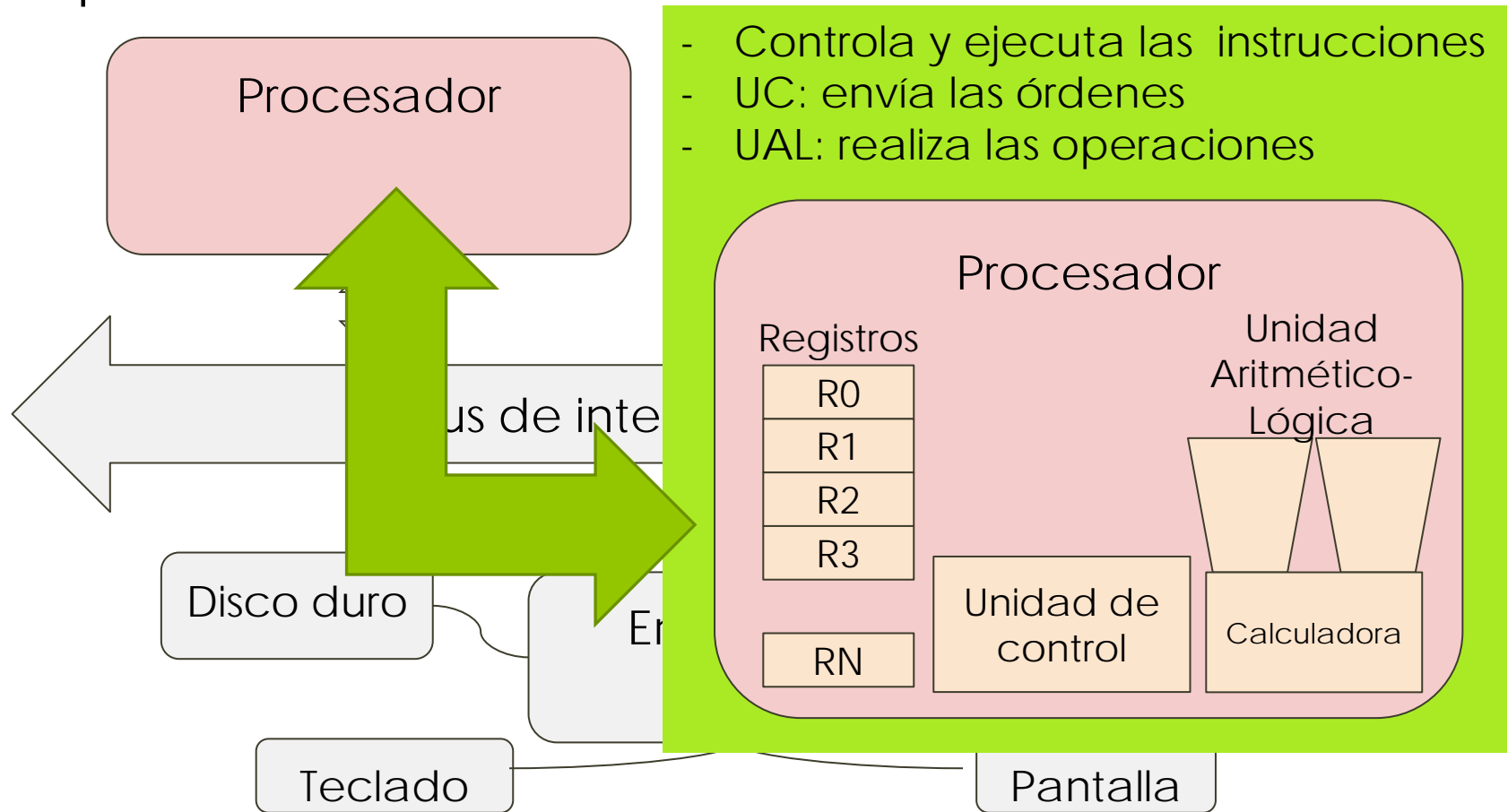
3. ARQUITECTURA DE ORDENADORES

Esquema



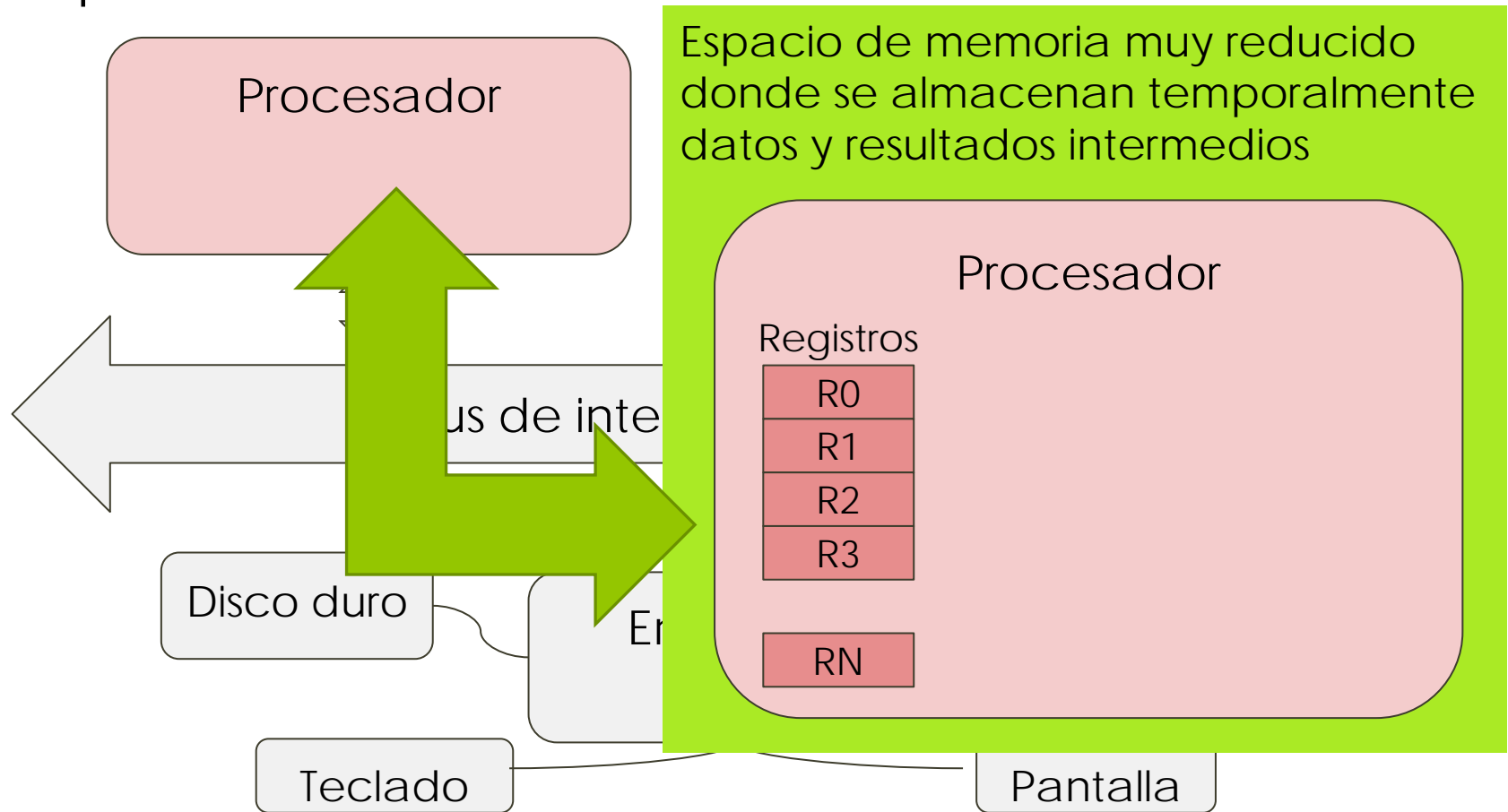
3. ARQUITECTURA DE ORDENADORES

Esquema



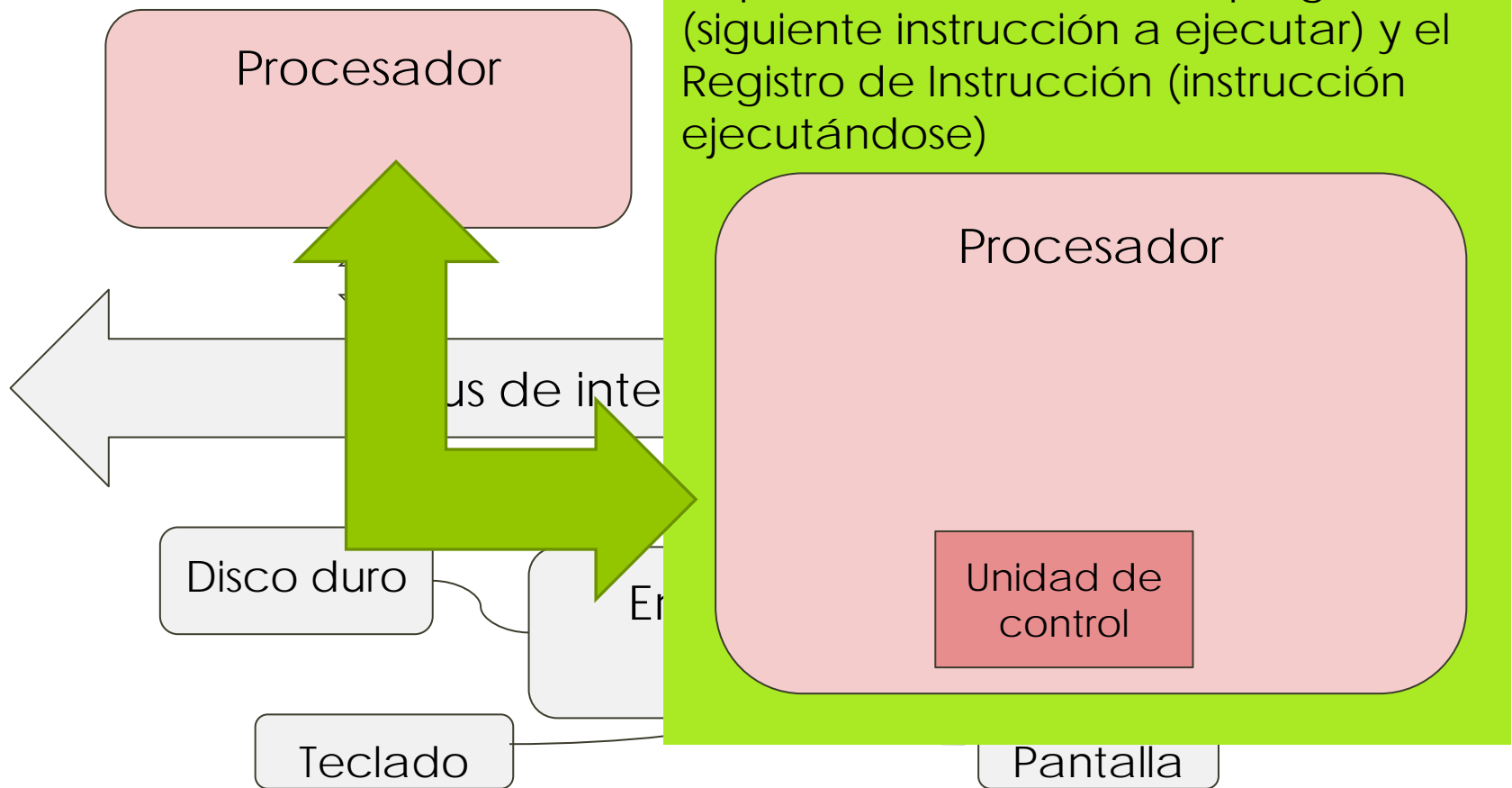
3. ARQUITECTURA DE ORDENADORES

Esquema



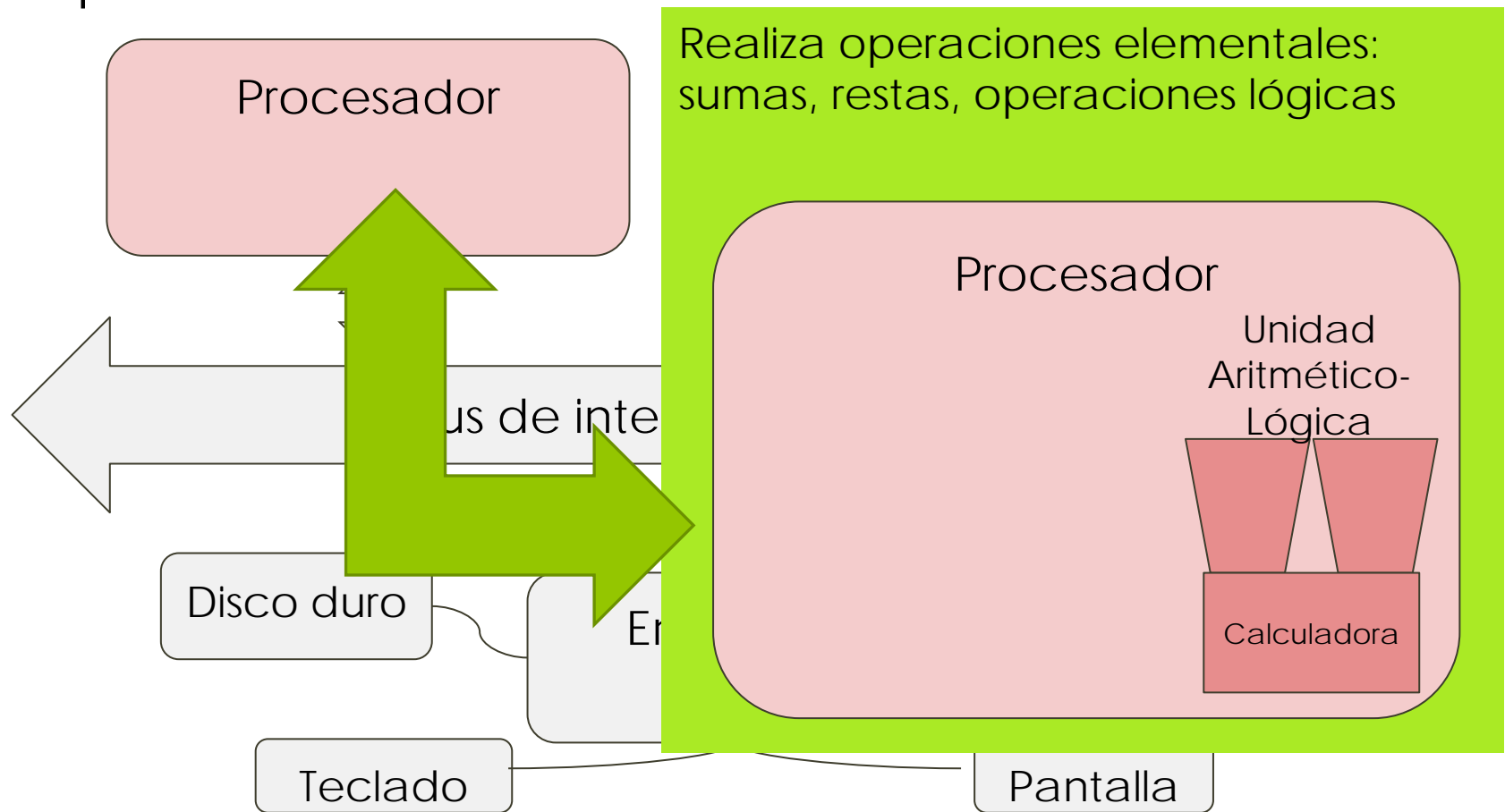
3. ARQUITECTURA DE ORDENADORES

Esquema



3. ARQUITECTURA DE ORDENADORES

Esquema

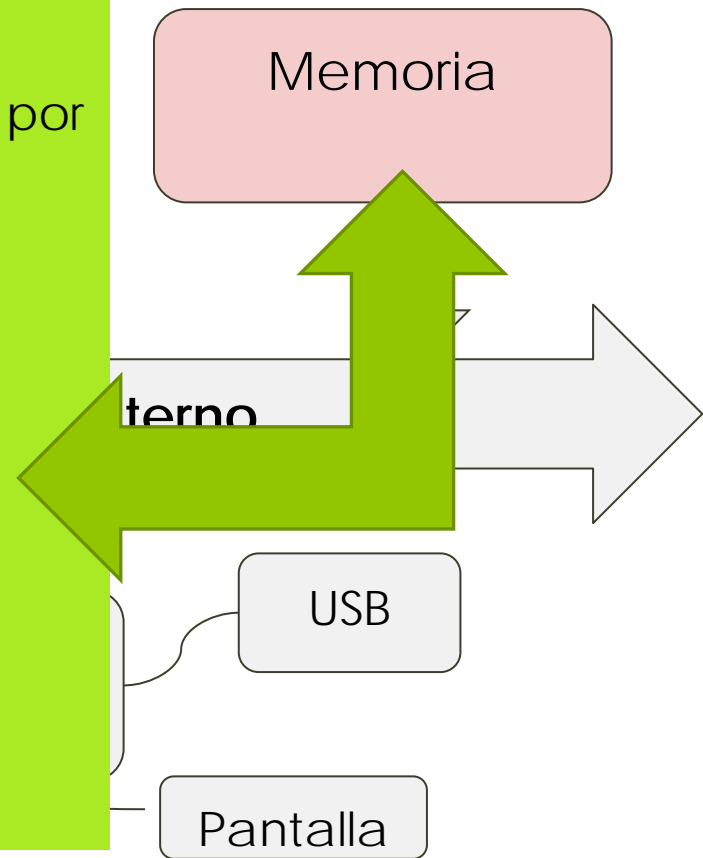


3. ARQUITECTURA DE ORDENADORES

Esquema

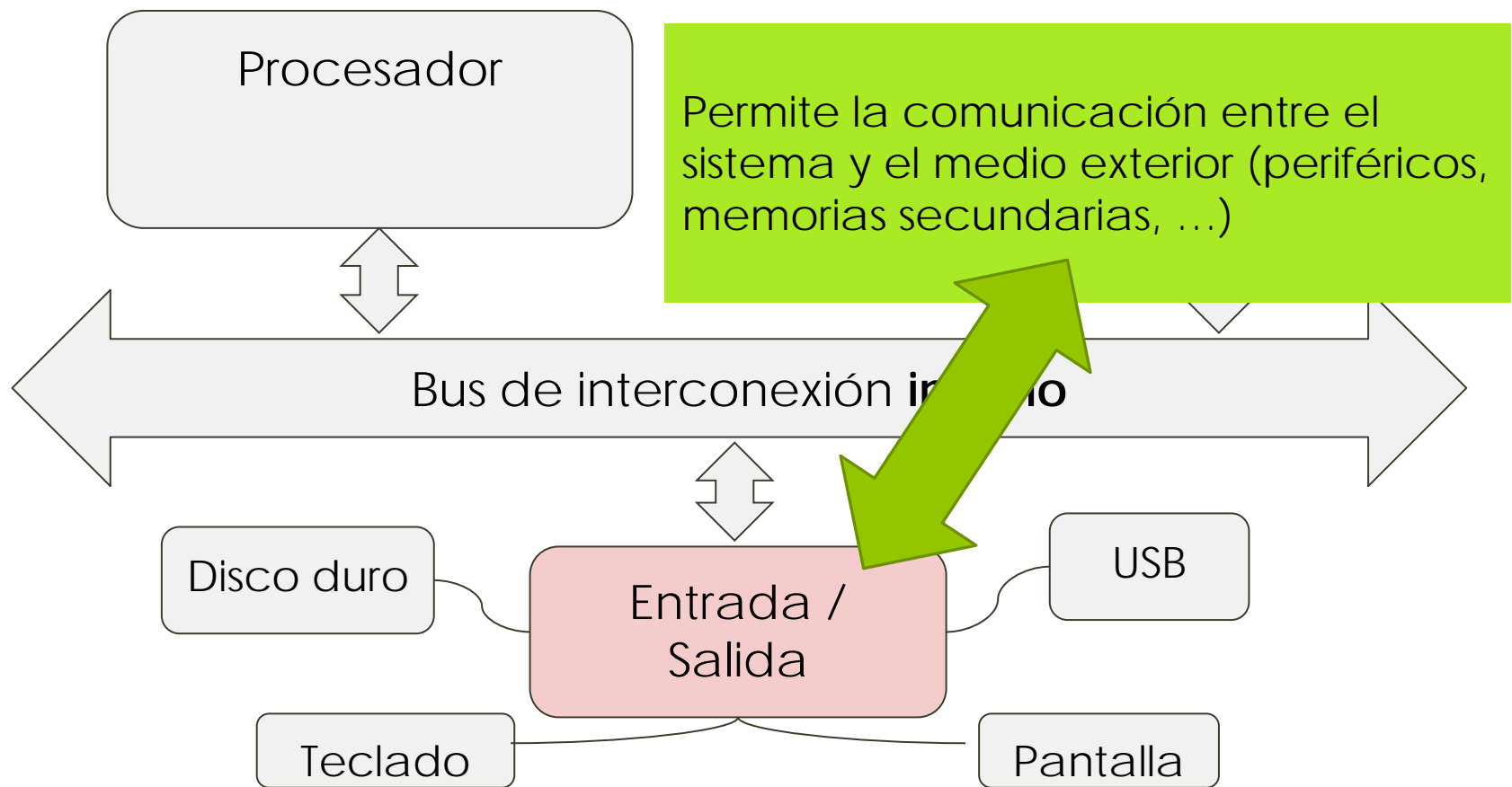
- Almacena datos en celdas
- Cada celda tiene una dirección
- Estos datos pueden ser consultados por el procesador

Dirección	Dato/instr.
000	96
001	5
002	a
...	...
999	34



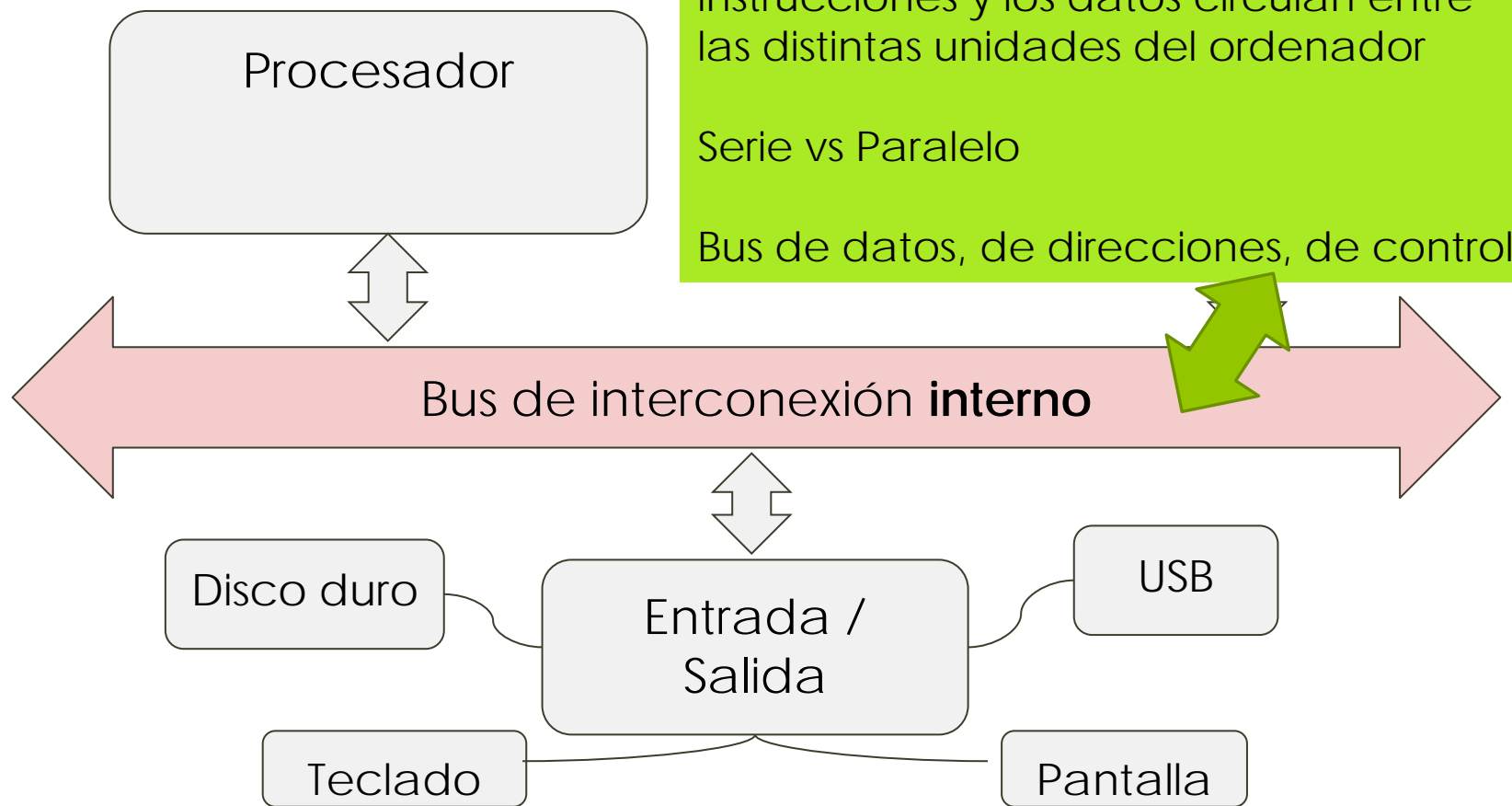
3. ARQUITECTURA DE ORDENADORES

Esquema



3. ARQUITECTURA DE ORDENADORES

Esquema



3. ARQUITECTURA DE ORDENADORES

ARQUITECTURA VON NEUMANN

¿Cómo funciona? → Objetivo: ejecutar programas

1. Un programa está compuesto por distintas instrucciones
2. Las instrucciones de un programa se ejecutan de forma secuencial, unas detrás de otras
3. Cada instrucción requiere de unas fases :

1. Búsqueda de la instrucción

2. Decodificación de la instrucción

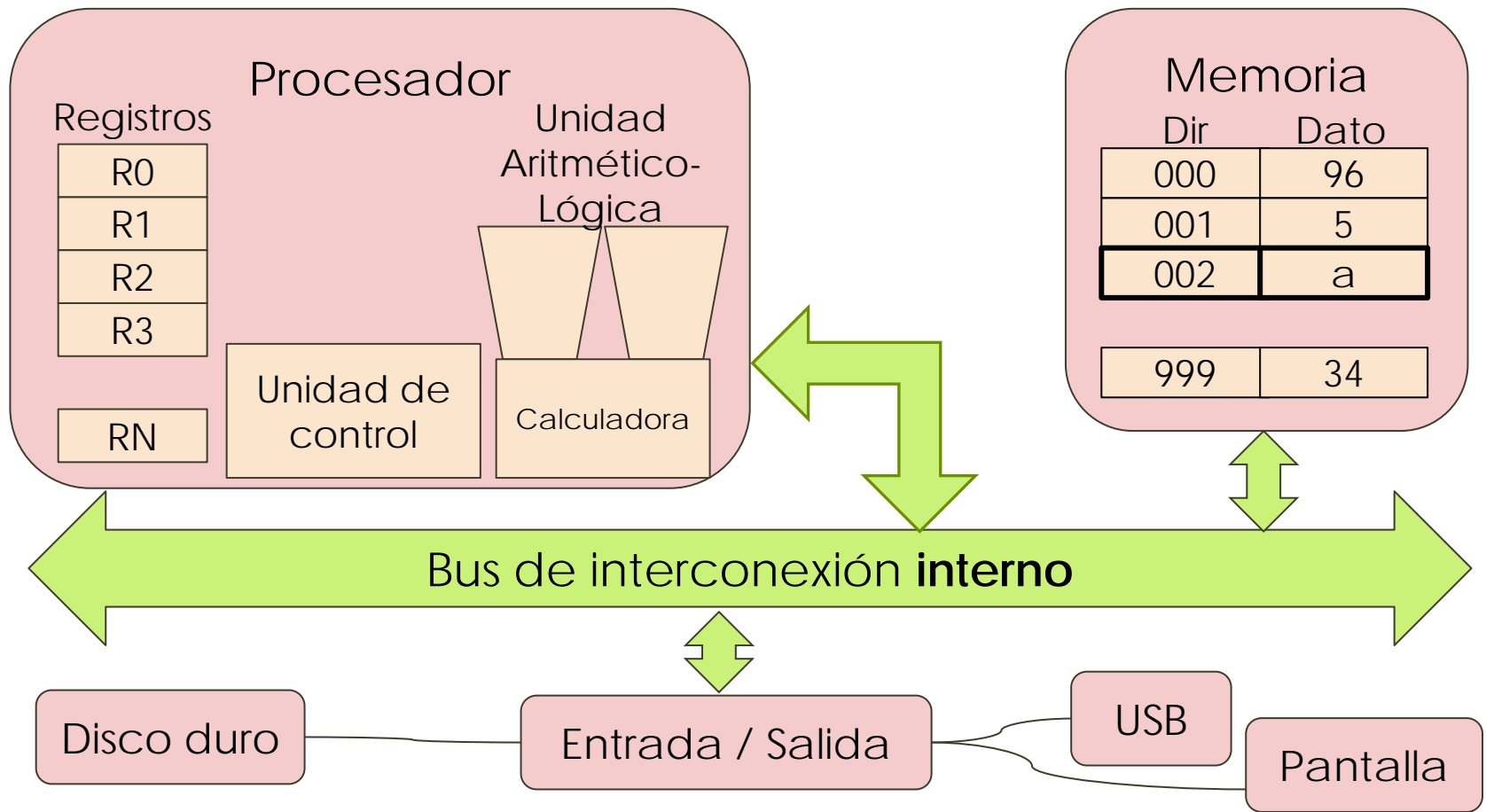
3. Búsqueda de los operandos

4. Ejecución u operación

5. Almacenamiento de resultados

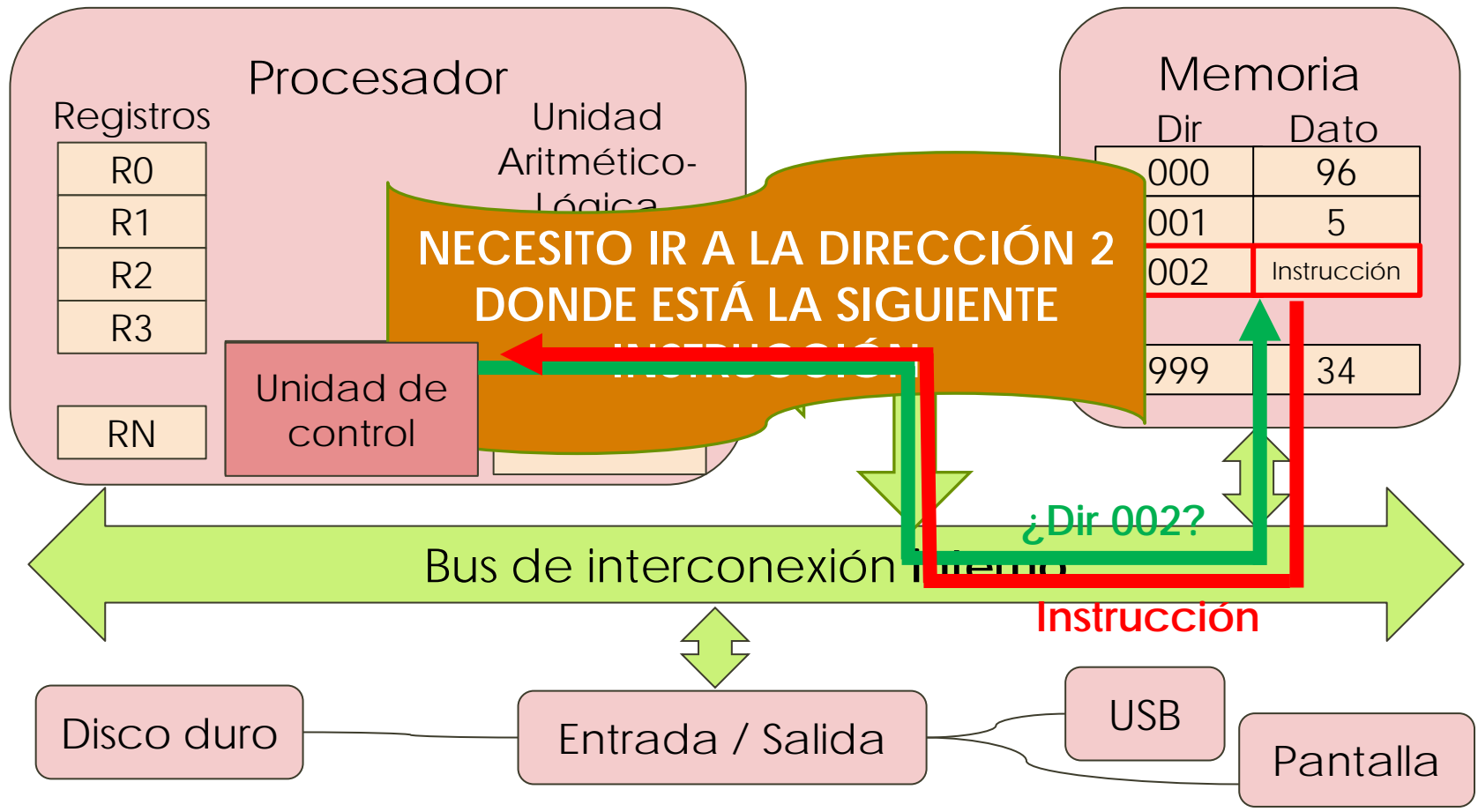
3. ARQUITECTURA DE ORDENADORES

ARQUITECTURA VON NEUMANN



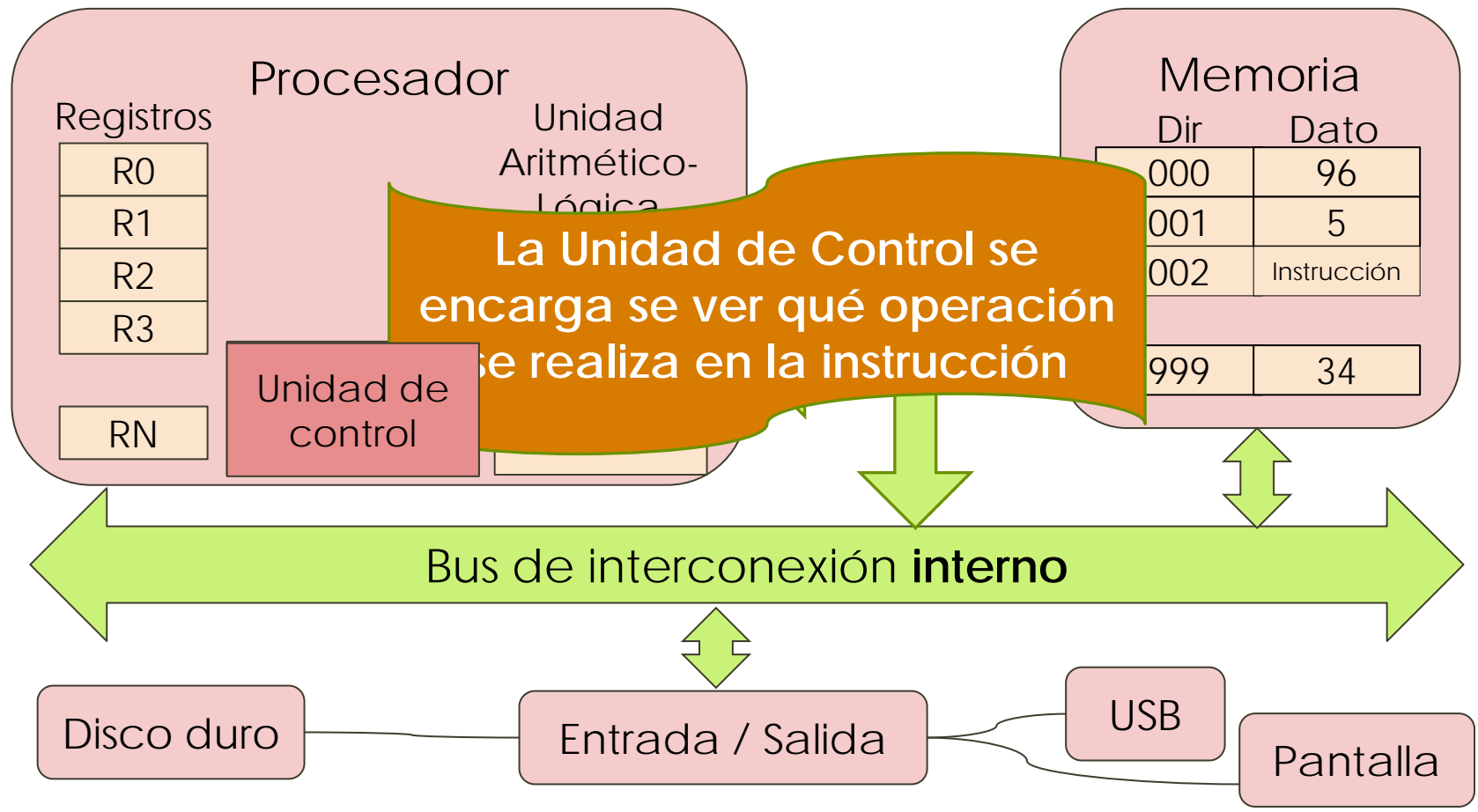
3. ARQUITECTURA DE ORDENADORES

ARQUITECTURA VON NEUMANN. FASE 1. Búsqueda de la instrucción



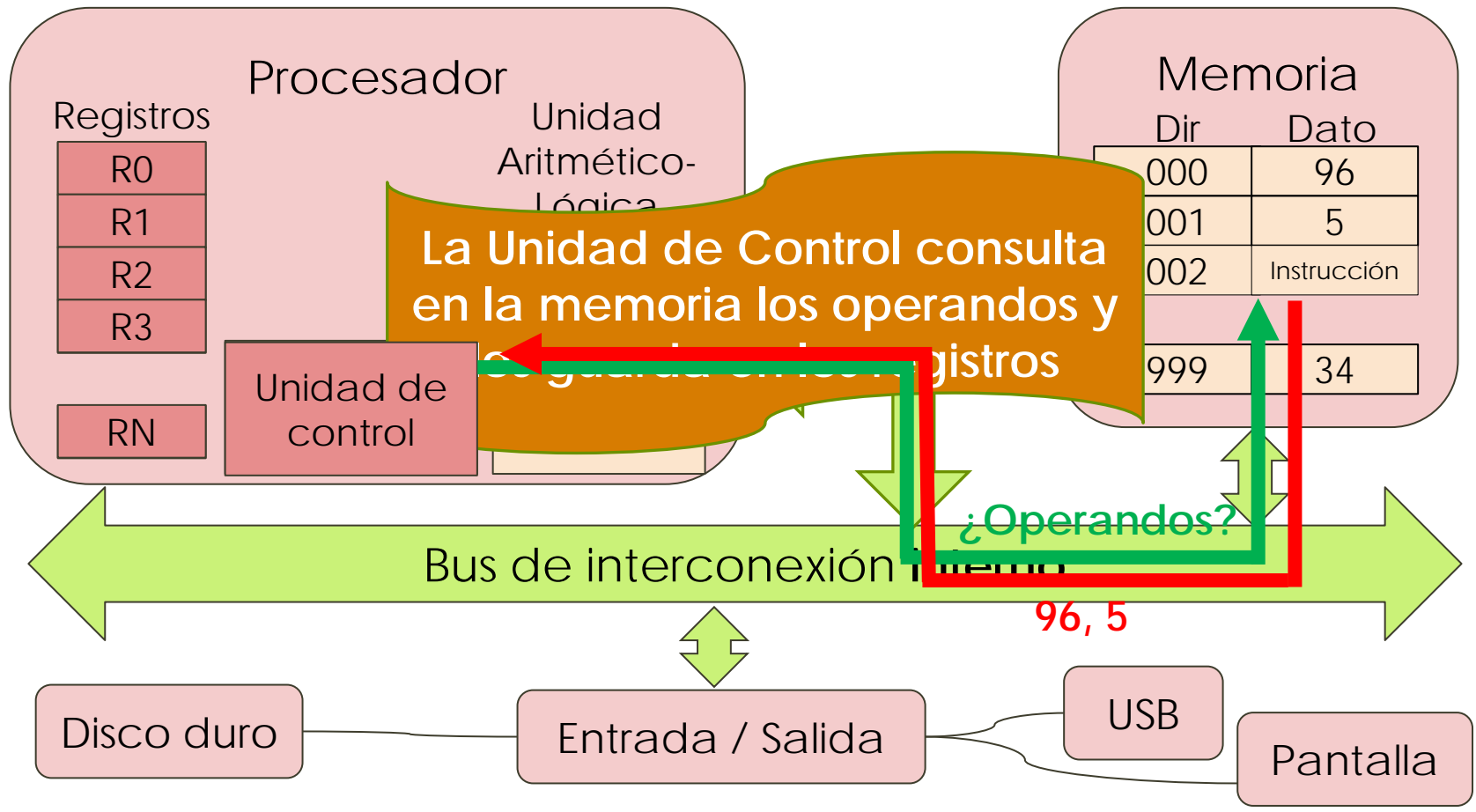
3. ARQUITECTURA DE ORDENADORES

ARQUITECTURA VON NEUMANN. FASE 2. Decodificar la instrucción



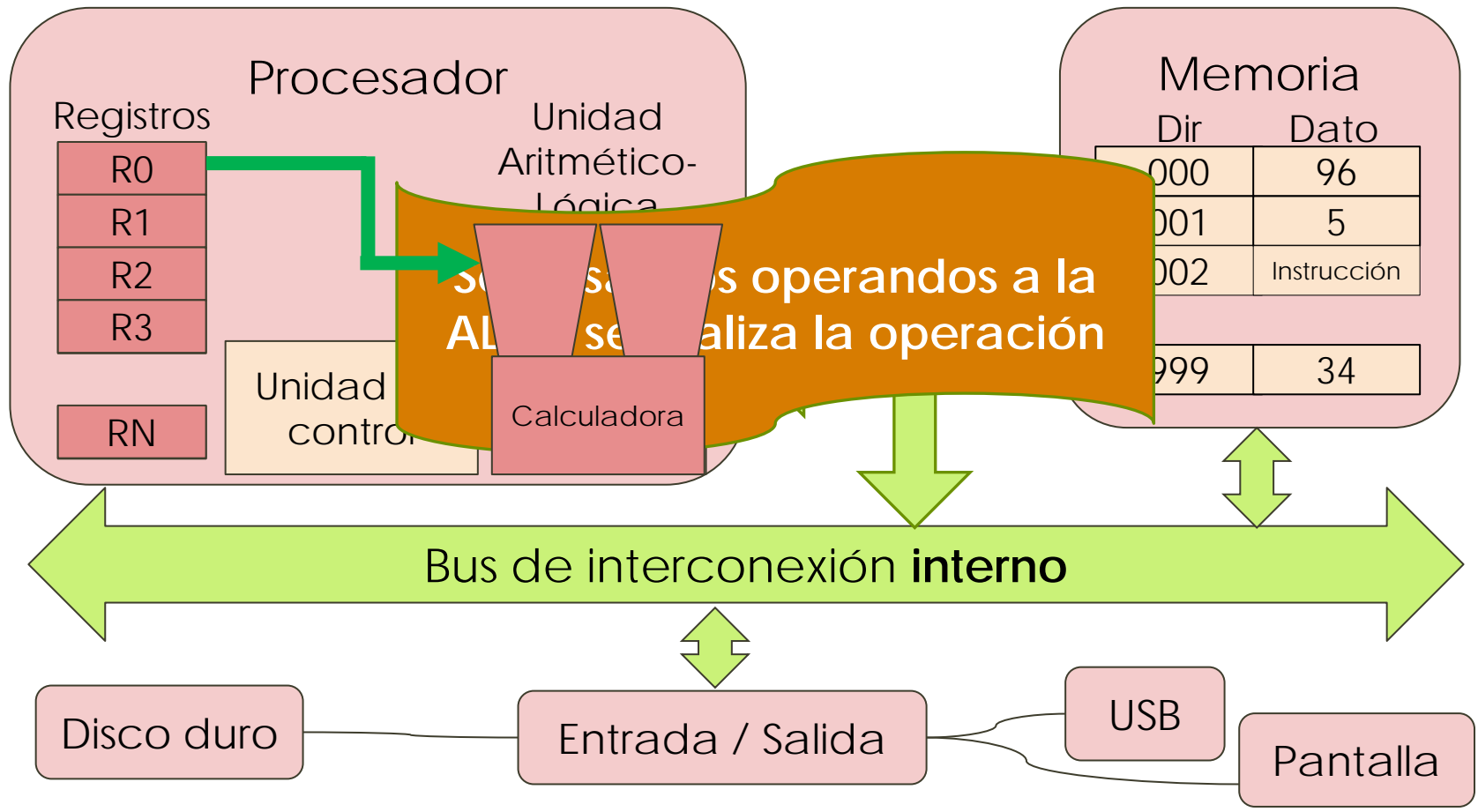
3. ARQUITECTURA DE ORDENADORES

ARQUITECTURA VON NEUMANN. FASE 3. Búsqueda de los operandos



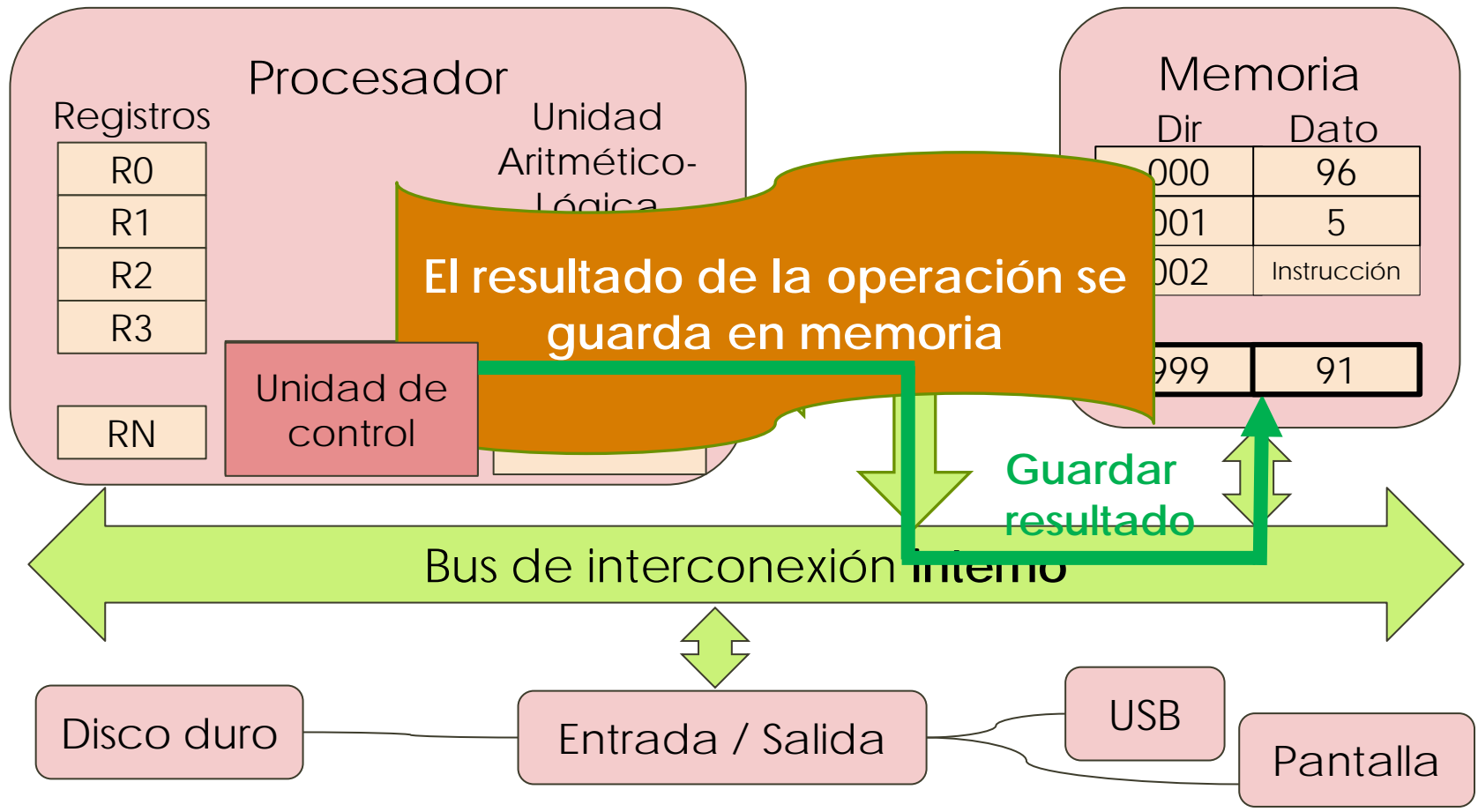
3. ARQUITECTURA DE ORDENADORES

ARQUITECTURA VON NEUMANN. FASE 4. Ejecución u operación



3. ARQUITECTURA DE ORDENADORES

ARQUITECTURA VON NEUMANN. FASE 5. Almacenar resultados



3. ARQUITECTURA DE ORDENADORES

¿CUÁNTAS INSTRUCCIONES DISTINTAS MANEJA UN PROCESADOR?

La misma operación se puede hacer mediante una instrucción compleja o mediante un conjunto de instrucciones simples

CISC (Complex Instruction Set Computing): instrucciones complejas y lentas de realizar con pocos accesos a memoria.
Procesadores x86 (Intel, AMD)

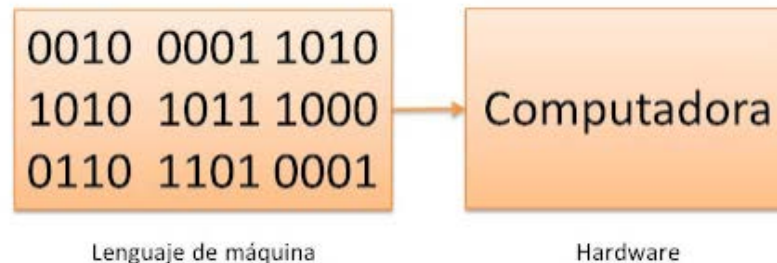
RISC (Reduced Instruction Set Computing): instrucciones simples y de ejecución más rápida. Hacen que los programas tengan más instrucciones.

Procesadores ARM (smartphones, sistemas empotrados, ...)

3. ARQUITECTURA DE ORDENADORES

¿CÓMO ES UNA INSTRUCCIÓN?

Las instrucciones que ejecuta el procesador están escritas en **código máquina**, formado por ceros y unos.



Son instrucciones sencillas: operaciones con memoria, operaciones aritméticas y operaciones de control sobre la CPU

Ejemplos: mover un dato a un registro, guardar un dato a memoria, sumar, ir a una posición de memoria, ...

3. ARQUITECTURA DE ORDENADORES

LENGUAJE ENSAMBLADOR

Lenguaje utilizado para escribir programas informáticos de bajo nivel. Utiliza nemotécnicos para cada instrucción, para los registros, las posiciones de memoria, ...

Ejemplos de instrucciones en lenguaje ensamblador:

MOV destino, origen

LOAD dir_origen, destino

ADD operando1, operando2

PUSH

POP

GOTO

...

MOV AX, 6

MOV BX, AX

LOAD #0003, CX

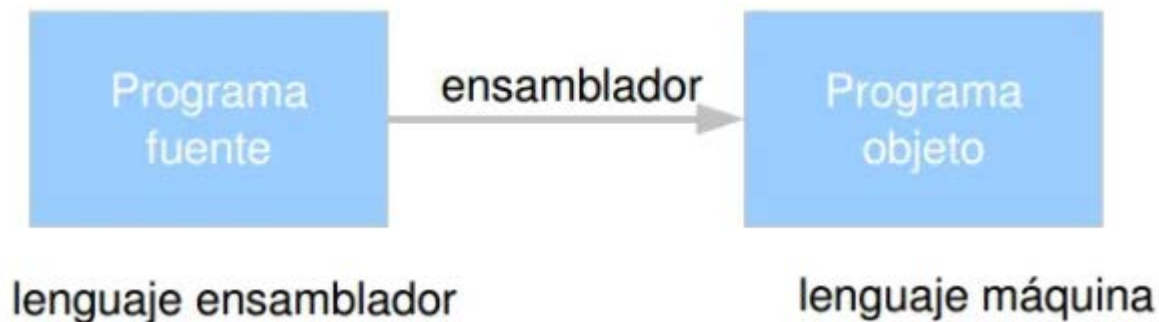
ADD BX, CX

3. ARQUITECTURA DE ORDENADORES

LENGUAJE ENSAMBLADOR

Un programa escrito en lenguaje ensamblador no puede ser ejecutado directamente, sino que hay que "traducirlo" a lenguaje máquina.

El programa encargado de este proceso se denomina ensamblador



3. ARQUITECTURA DE ORDENADORES

LENGUAJE ENSAMBLADOR

00010101
00010111
00010110

10000001
10000010
10000011

LOAD A
ADD B
STORE C

$C = A + B$

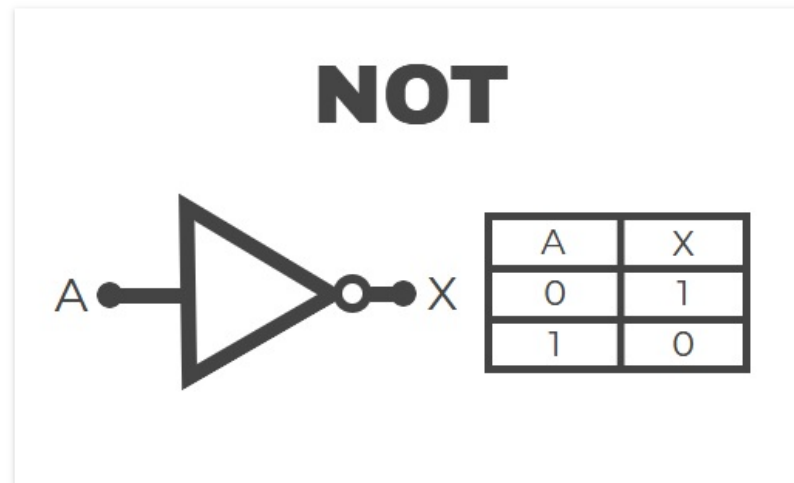
Código
máquina

Lenguaje
ensamblador

Lenguaje
alto nivel

4. PUERTAS LÓGICAS

Las puertas lógicas son circuitos electrónicos formados internamente por combinaciones de transistores.

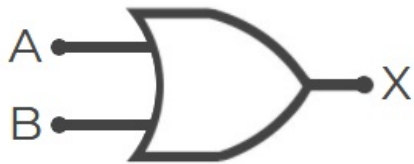


Hay diferentes puertas.

Todas ellas tienen asociada una **tabla de verdad** que indica la salida obtenida en función de las entradas

4. PUERTAS LÓGICAS

OR



A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

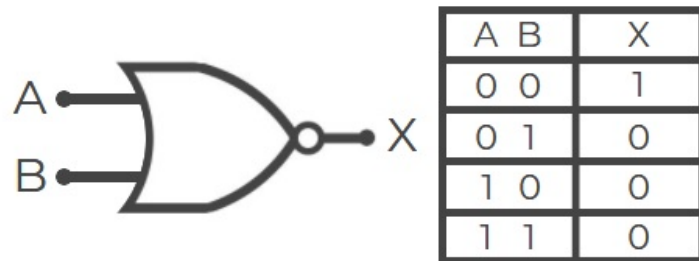
AND



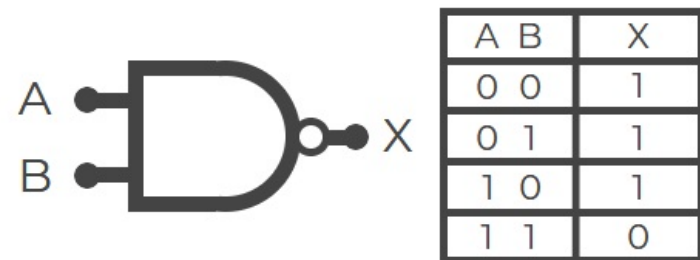
A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

4. PUERTAS LÓGICAS

NOR



NAND



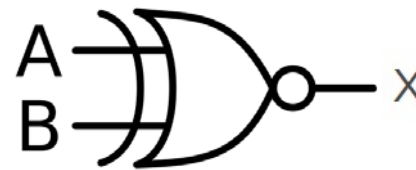
4. PUERTAS LÓGICAS

X-OR



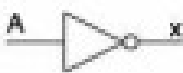
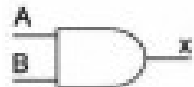


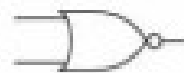
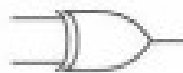

Entradas		Salida
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

X-NOR



A	B	X
0	0	1
0	1	0
1	0	0
1	1	1

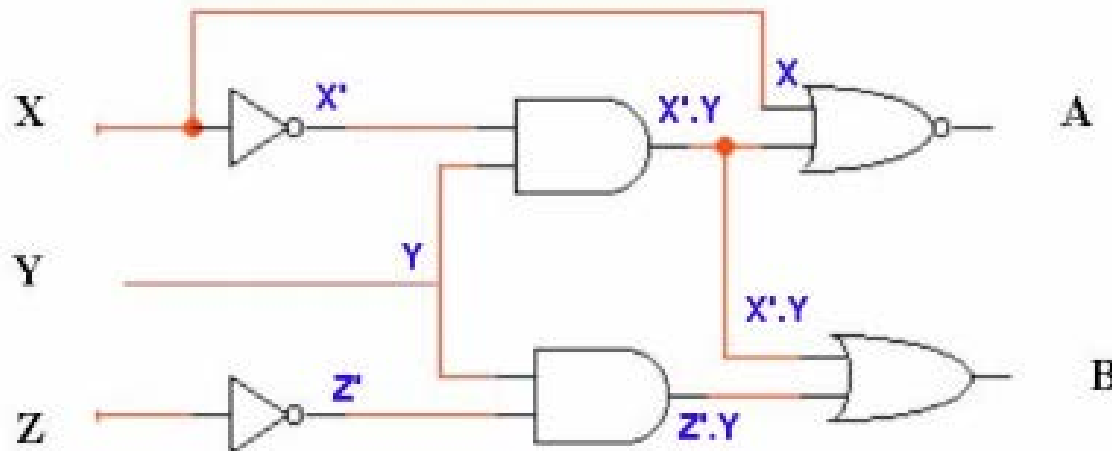
4. PUERTAS LÓGICAS

NOT	AND	NAND	OR	NOR	XOR	XNOR																																																																																																
\overline{A}	AB	\overline{AB}	$A + B$	$\overline{A + B}$	$A \oplus B$	$\overline{A \oplus B}$																																																																																																
																																																																																																						
<table><tr><th>A</th><th>X</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	X	0	1	1	0	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	B	A	X	0	0	0	0	1	0	1	0	0	1	1	1	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	B	A	X	0	0	1	0	1	1	1	0	1	1	1	0	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	B	A	X	0	0	0	0	1	1	1	0	1	1	1	1	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	B	A	X	0	0	1	0	1	0	1	0	0	1	1	0	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	B	A	X	0	0	0	0	1	1	1	0	1	1	1	0	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	B	A	X	0	0	1	0	1	0	1	0	0	1	1	1
A	X																																																																																																					
0	1																																																																																																					
1	0																																																																																																					
B	A	X																																																																																																				
0	0	0																																																																																																				
0	1	0																																																																																																				
1	0	0																																																																																																				
1	1	1																																																																																																				
B	A	X																																																																																																				
0	0	1																																																																																																				
0	1	1																																																																																																				
1	0	1																																																																																																				
1	1	0																																																																																																				
B	A	X																																																																																																				
0	0	0																																																																																																				
0	1	1																																																																																																				
1	0	1																																																																																																				
1	1	1																																																																																																				
B	A	X																																																																																																				
0	0	1																																																																																																				
0	1	0																																																																																																				
1	0	0																																																																																																				
1	1	0																																																																																																				
B	A	X																																																																																																				
0	0	0																																																																																																				
0	1	1																																																																																																				
1	0	1																																																																																																				
1	1	0																																																																																																				
B	A	X																																																																																																				
0	0	1																																																																																																				
0	1	0																																																																																																				
1	0	0																																																																																																				
1	1	1																																																																																																				

4. PUERTAS LÓGICAS

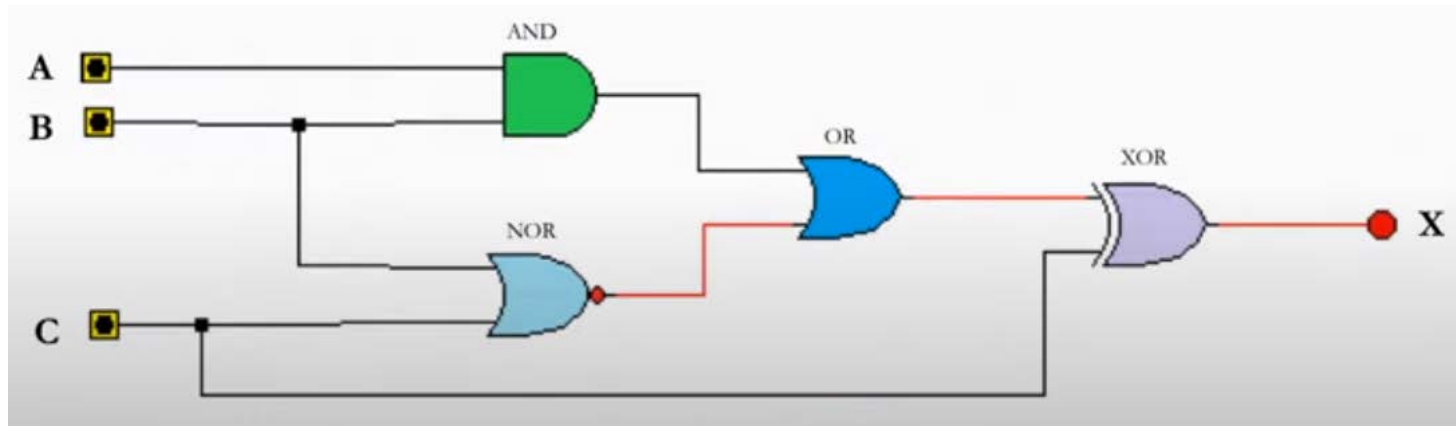
CIRCUITOS COMBINACIONALES

Los circuitos combinacionales son combinaciones complejas de puertas lógicas para llevar a cabo una función determinada. Se combinan las entradas y las salidas para obtener una salida concreta.



4. PUERTAS LÓGICAS

CIRCUITOS COMBINACIONALES - Ejemplo



Realizar la tabla de verdad del circuito anterior

4. PUERTAS LÓGICAS

SIMULADOR

<https://logic.ly/demo/>

The screenshot displays the Logic.ly online logic simulator. The main workspace shows a logic circuit with three input switches on the left, connected to a network of AND, OR, and NOT gates, which finally drive two light bulbs on the right. A sidebar on the right contains a library of components categorized into Input Controls (Toggle Switch, Push Button, Clock, High/Low Constant), Output Controls (Light Bulb, Digit), and Logic Gates (Buffer, NOT Gate, AND Gate, NAND Gate). The bottom of the image shows a Windows taskbar with several open applications, including a web browser displaying the Logic.ly website, a spreadsheet application, and a terminal window.

Spreadsheet content:

$(A + B) \times \bar{C}$ casos = $2^3 =$

A	B	C	$(A+B)$	\bar{C}	$(A+B) \times \bar{C}$
0	0	0	0	1	
0	0	1	0	0	
0	1	0	1	1	1
0	1	1	1	0	
1	0	0	1	1	1
1	0	1	1	0	
1	1	0	1	1	1
1	1	1	1	0	

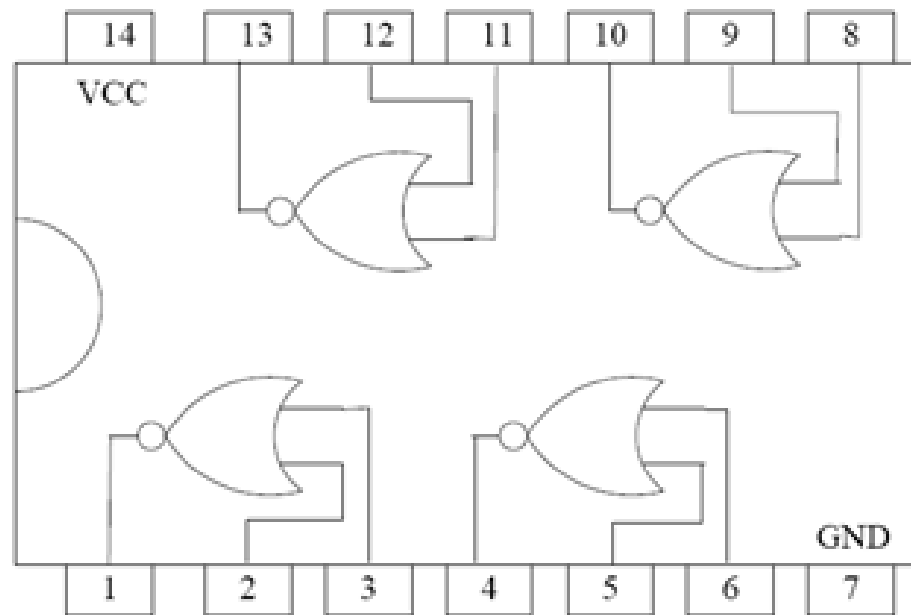
5. CIRCUITOS INTEGRADOS

Los **circuitos combinacionales** están integrados en **circuitos integrados**. Están formados principalmente por transistores, los cuales pueden estar acompañados de diodos, resistencias y condensadores, interconectados y ubicados en una pastilla de silicio.

Sus dimensiones son muy reducidas y sus elementos no se pueden separar.

Se conocen también como chips o “cucarachas”.

5. CIRCUITOS INTEGRADOS




El número de puertas lógicas de un circuito integrado va desde 3 o 4 (SSI – Small Scale Integration) hasta 1,000.000 de puertas (GSI – Giga Scale Integration)

5. CIRCUITOS INTEGRADOS

Simplificación de funciones

El objetivo es obtener las funciones necesarias para que dadas unas entradas concretas se produzca la salida deseada.

Esta función debe ser lo más simple posible → menor número de componentes electrónicos

Método de simplificación 

- Álgebra de Boole
- Método de Karnaugh

5. CIRCUITOS INTEGRADOS

Método de Karnaugh

- Calcular la tabla de verdad.
- Hacer una tabla con los resultados de la tabla teniendo en cuenta que las casillas deben ser adyacentes
- La adyacencia se define por un cambio de una única variable

BC					
A		00	01	11	10
	0	0	0	1	1
	1	1	1	1	1

6. CONCEPTOS BÁSICOS

SISTEMAS DE NUMERACIÓN: Conjunto de símbolos y reglas utilizadas para representar cantidades o datos numéricos

CODIFICACIÓN NUMÉRICA: Los 3 sistemas de codificación utilizados en un sistema informático son:

Binario: Utiliza dos símbolos diferentes (0 y 1). Sistema manejado internamente en el ordenador.

Octal: Utiliza ocho símbolos (0 al 7).

Hexadecimal: Sistema de numeración en base 16. Utiliza 16 símbolos, los números del 0 al 9 y las letras A, B, C, D, E y F.

6. CONCEPTOS BÁSICOS

¡¡RECORDAD!!

- ❑ Conversiones entre los distintos sistemas de numeración:
 - Decimal → Binario
 - Decimal → Hexadecimal
 - Binario → Decimal
 - Hexadecimal → Decimal
 - Binario → Hexadecimal
 - Hexadecimal → Binario
- ❑ Conjunto de elementos que se pueden representar con un número determinado de símbolos.

6. CONCEPTOS BÁSICOS

Unidades de almacenamiento

Unidades para la velocidad de transmisión

Conversión entre unidades

7. PRÁCTICAS

- Cambios entre sistemas de codificación
- Lógica combinacional