

Universidad Autónoma del Estado de México

Centro Universitario UAEM Valle de Chalco

Programa de Ingeniería en Computación

Unidad de Aprendizaje: Ensambladores

Créditos institucionales: 7

Material:

Conceptos relacionados con sistemas numéricos, sistemas de cómputo, arquitectura de procesadores y programación de sistemas

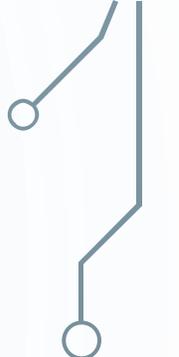
Elaborado por:

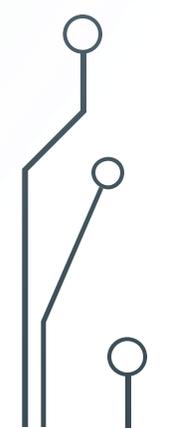
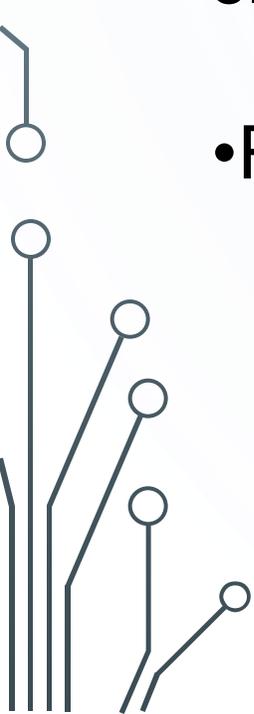
Mtro. Marco Alberto Mendoza Pérez

Julio - Septiembre 2019

CONTENIDO

- Presentación
- Estructura de la Unidad de Aprendizaje:
Unidad I. Analizar los conceptos relacionados con sistemas numéricos, sistemas de cómputo, arquitectura de procesadores y programación de sistemas
 1. Elementos de los sistemas numéricos decimal, binario y hexadecimal y conversión de números.
 2. Elementos de un sistema de cómputo.
 3. Arquitectura de procesadores enfatizando en el que se basará el ensamblador objeto de estudio.

- 
- 
4. Método de gestión de memoria del procesador seleccionado.
 5. Conceptos de la programación de sistemas (sistema, sistema de cómputo, programación, programación de sistemas).
 6. Evolución de los lenguajes de programación.
 7. Conceptos y diferencias entre ensamblador, compilador e intérprete.
 8. Conceptos de ligador y cargador.



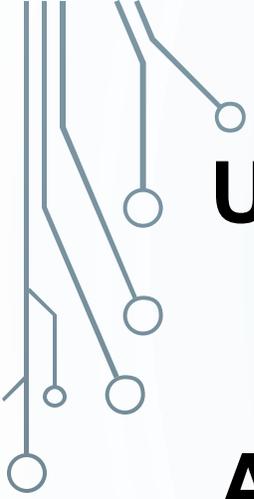
- Referencias

PRESENTACIÓN

Las presentes diapositivas fueron desarrolladas en apego a la primera unidad que conforma el programa de la Unidad de Aprendizaje de Ensambladores, con la finalidad de servir de apoyo en el proceso de enseñanza y aprendizaje entre profesores y alumnos, siendo este material creativo, innovador y amigable para ambos usuarios. En la siguiente diapositiva, se menciona la estructura de la Unidad de Aprendizaje de Ensambladores.

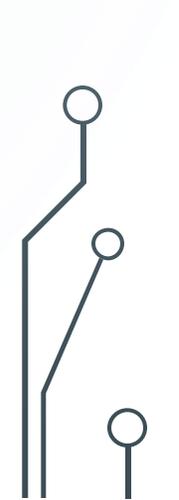
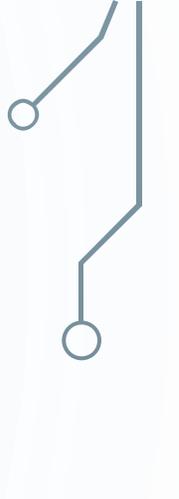
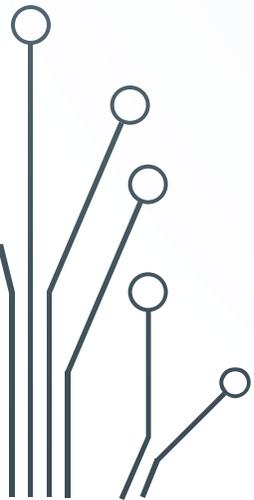
ESTRUCTURA DE LA UNIDAD DE APRENDIZAJE

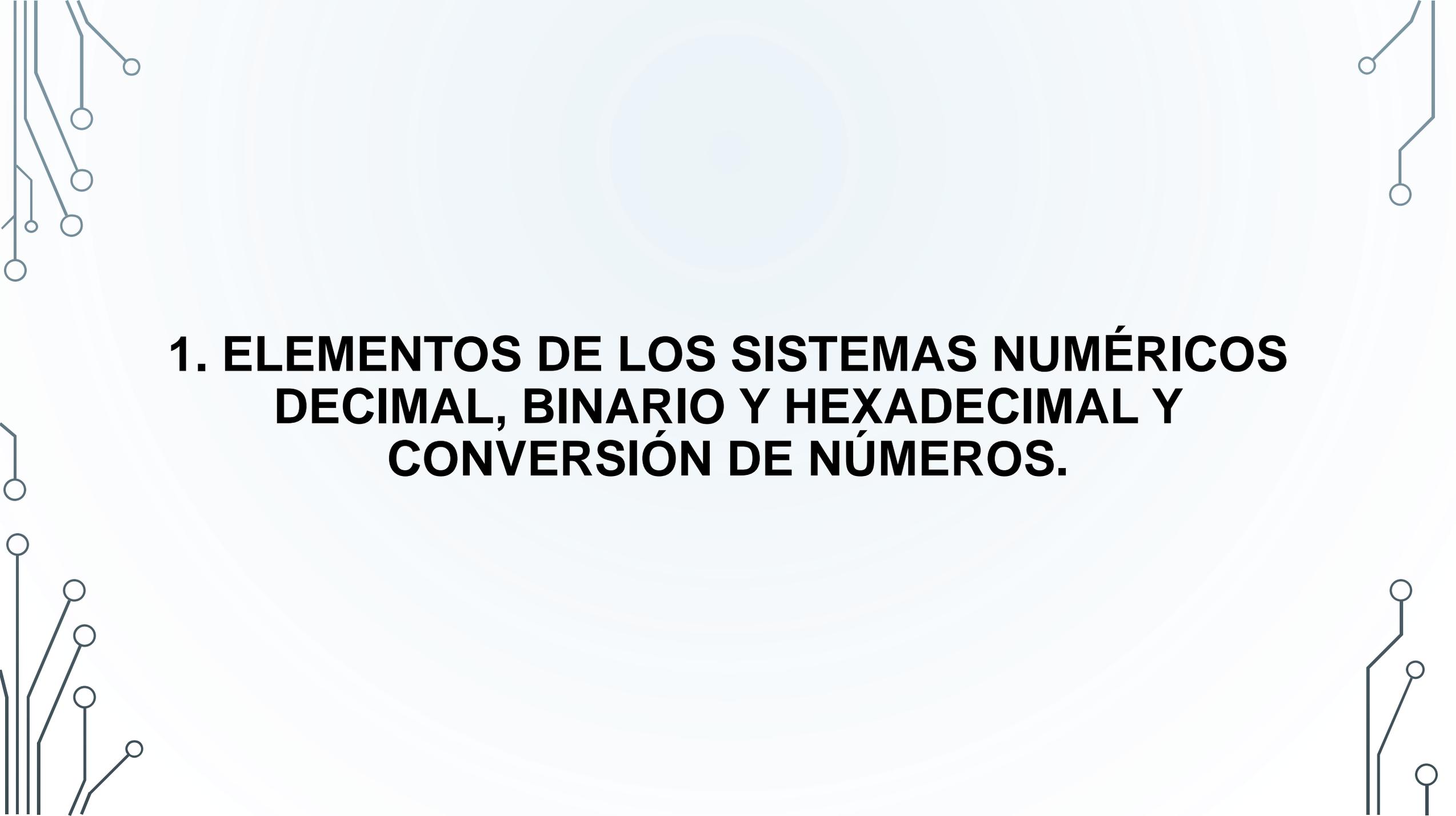
1. Analizar los conceptos relacionados con sistemas numéricos, sistemas de cómputo, arquitectura de procesadores y programación de sistemas.
2. Conocer, analizar y comprender los elementos y etapas de un ensamblador y aplicarlo en el desarrollar un ensamblador.
3. Conocer, analizar y comprender los elementos y fases de un cargador.



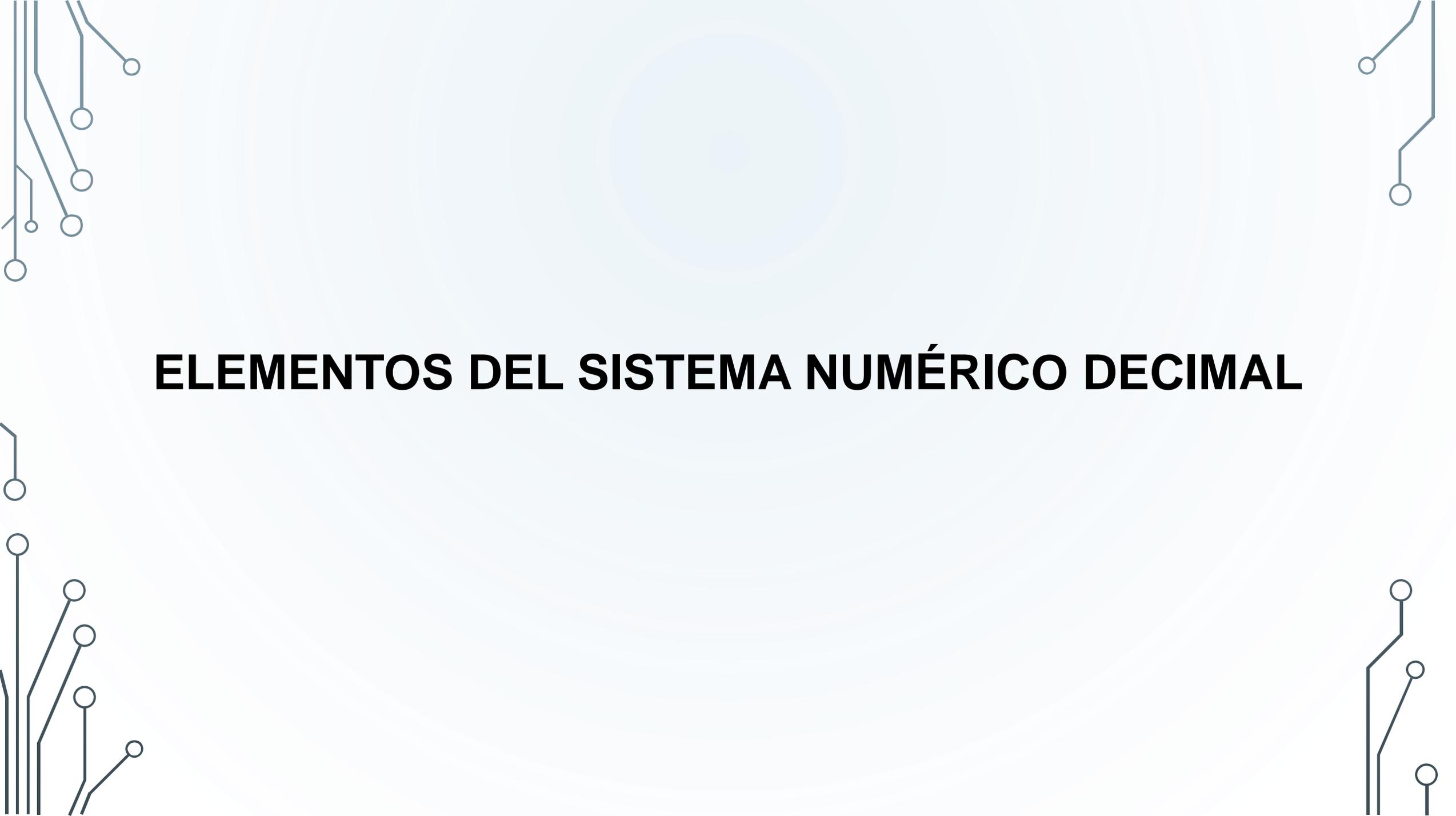
UNIDAD DE COMPETENCIA I:

**ANALIZAR LOS CONCEPTOS RELACIONADOS
CON SISTEMAS NUMÉRICOS, SISTEMAS DE
CÓMPUTO, ARQUITECTURA DE PROCESADORES
Y PROGRAMACIÓN DE SISTEMAS.**



The image features a light blue background with a faint, large-scale grid pattern. In the four corners, there are decorative elements consisting of thin, grey lines that resemble circuit traces or data paths, ending in small circles. The main content is a bold, black, sans-serif title centered on the page.

1. ELEMENTOS DE LOS SISTEMAS NUMÉRICOS DECIMAL, BINARIO Y HEXADECIMAL Y CONVERSIÓN DE NÚMEROS.

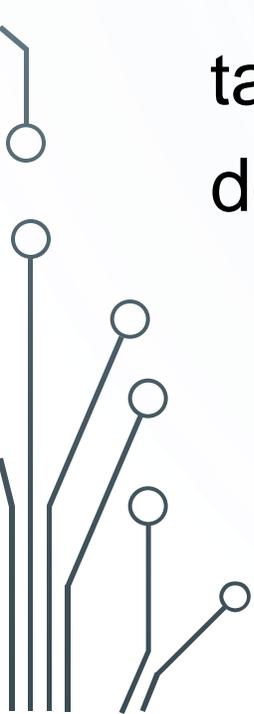
The image features a light blue background with a large, faint circular graphic in the center. The corners are decorated with stylized circuit board traces and nodes. The main text is centered horizontally and reads "ELEMENTOS DEL SISTEMA NUMÉRICO DECIMAL" in a bold, black, sans-serif font.

ELEMENTOS DEL SISTEMA NUMÉRICO DECIMAL



LA BASE DEL SISTEMA

El elemento básico de los sistemas de numeración es el agrupamiento de unidades. La mayoría de los sistemas, incluyendo el nuestro, han usado agrupaciones de 10. El tamaño de la agrupación recibe el nombre de base del sistema de numeración.



LOS SÍMBOLOS DEL SISTEMA

Los símbolos elementales que se utilizan en los sistemas de numeración dependen de su base. Esta nos indica cuantos símbolos se utilizan para representar números. Por ejemplo.- El decimal o base 10, utiliza los siguientes símbolos: {0, 1, 2, 3, 4, 5, 6, 7, 8 y 9}. Su notación es $(N)_{10}$. Ejemplo de un número decimal: $(5529)_{10}$.

ELEMENTOS DEL SISTEMA NUMÉRICO BINARIO

Es un sistema de base 2, en el que solo se tienen 2 símbolos (0 y 1) que significan (apagado y encendido). Es de gran importancia debido a que es el lenguaje que manejan las computadoras (lenguaje máquina). Su notación es $(N)_2$. Ejemplo de un número binario: $(10101101)_2$.

ELEMENTOS DEL SISTEMA NUMÉRICO HEXADECIMAL

Es un sistema de base 16. Está formado por elementos o símbolos {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E y F}. Su notación es $(N)_{16}$ y las letras tienen los siguientes valores: A=10, B=11, C=12, D=13, E=14 y F=15. Ejemplo de un número hexadecimal: $(8AC1F9D5)_{16}$.

The image features a light blue background with a large, faint circular graphic in the center. The corners are decorated with stylized circuit board traces and nodes. The central text is in a bold, black, sans-serif font.

CONVERSIÓN DE NÚMEROS

CONVERSIÓN DE DECIMAL A BINARIO

Para hacer la conversión de decimal a binario, tenemos que ir dividiendo el número decimal entre dos. Se pondrá un 0 si el resultado de la división es par y un 1 si es impar.

El divisor siempre será 2, el dividendo la primer ocasión será el número a convertir, una vez hecha la primer división el cociente obtenido será el nuevo dividendo y el residuo será el resultado del número binario leído de abajo hacia arriba.

EJEMPLO:

Convertir el numero $(93)_{10}$ a $(N)_2$.

Resultado de la conversión:

$$(93)_{10} = (1011101)_2$$

46
$2 \overline{)93}$
13
1
23
$2 \overline{)46}$
0
11
$2 \overline{)23}$
1
5
$2 \overline{)11}$
1
2
$2 \overline{)5}$
1
1
$2 \overline{)2}$
0
0
$2 \overline{)1}$
1

CONVERSIÓN DE DECIMAL A HEXADECIMAL

Para hacer la conversión de decimal a hexadecimal, tenemos que ir dividiendo el número decimal entre 16.

El divisor siempre será 16, el dividendo la primer ocasión será el número a convertir, una vez hecha la primer división el cociente obtenido será el nuevo dividendo y el residuo será el resultado del número hexadecimal leído de abajo hacia arriba.

EJEMPLO:

Convertir el numero $(350)_{10}$ a $(N)_{16}$.

RESULTADO:
 $(350)_{10} = (15E)_{16}$

Resultado de la conversión:

$$(350)_{10} = (15E)_{16}$$

$$\begin{array}{r} 21 \\ 16 \overline{) 350} \\ \underline{14} \\ 1 \\ 16 \overline{) 21} \\ \underline{5} \\ 0 \\ 16 \overline{) 1} \\ \underline{1} \end{array}$$

CONVERSIÓN DE BINARIO A DECIMAL

Si tenemos el número binario 1000011011 y queremos saber cuál es su equivalente en la notación decimal, debemos escribir las potencias de dos. De derecha a izquierda, comenzamos por 2^0 , luego 2^1 , 2^2 , 2^3 y así sucesivamente. Es importante recordar que empezamos por la derecha, o sea, en el orden inverso de la lectura tradicional. Para que nos sea más fácil el cálculo, es recomendable escribir también el valor de cada potencia, es decir, $2^0=1$, luego $2^1=2$, $2^2=4$, $2^3=8$, etc.

Sumamos los valores correspondientes de estas potencias: $512+16+8+2+1$ y el resultado de esta suma es el número decimal correspondiente.

En este caso, el número binario 1000011011 es igual al número decimal 539.

2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
512	256	128	64	32	16	8	4	2	1
1	0	0	0	0	1	1	0	1	1

$$\begin{array}{r} 512 \\ 16 \\ 8 \\ 2 \\ + 1 \\ \hline \end{array}$$

$$= 539$$

CONVERSIÓN DE HEXADECIMAL A DECIMAL

Escribe el número $482,137_{10}$. Recuerda que el subíndice $_{10}$ nos indica que el número está escrito en una base diez. Comienza con el dígito en el extremo derecho, $7 = 7 \cdot 10^0$ o $7 \cdot 1$

Luego ve hacia la izquierda, $3 = 3 \cdot 10^1$ o $3 \cdot 10$

Al repetir todos los dígitos, resolvemos que $482,137 = 4 \cdot 100,000 + 8 \cdot 10,000 + 2 \cdot 1,000 + 1 \cdot 100 + 3 \cdot 10 + 7 \cdot 1$.

Ejemplo: $(CA21)_{16} = (1 \cdot 16^0) + (2 \cdot 16^1) + (A \cdot 16^2) + (C \cdot 16^3) = 1 \cdot 1 + 2 \cdot 16 + 10 \cdot 256 + C \cdot 4096 = 1 + 32 + 2560 + 49152 = (51,745)_{10}$

Ejemplo: Conversión de hexadecimal a binario y viceversa. A continuación se muestra un programa en lenguaje ensamblador con la utilidad DEBUG de MS-DOS para una operación lógica and (Y).

Paso 1: Convertir de hexadecimal a binario los Valores de los registros de AX y BX.

$$(A28E)_{16} = (1010)_2 \mid (0010)_2 \mid (1000)_2 \mid (1110)_2$$

A 2 8 E

$$(C4F5)_{16} = (1100)_2 \mid (1000)_2 \mid (1FFF)_2 \mid (0101)_2$$

C 4 F 5

```
D:\>debug
-a0100
06B0:0100 mov ax,A28E
06B0:0103 mov bx,C4F5
06B0:0106 and ax,bx
06B0:0108 int 20
06B0:010A
-rCX
CX 0000 :0A
-n and.com
-w
Writing 000A bytes
-d ds:0100_
```

Paso 2: Realizar la operación AND AX,BX.

$$\begin{array}{l} AX = (1010)_2 \mid (0010)_2 \mid (1000)_2 \mid (1110)_2 \\ \underline{BX = (1100)_2 \mid (1000)_2 \mid (1111)_2 \mid (0101)_2} \\ AX = (1000)_2 \mid (0000)_2 \mid (1000)_2 \mid (0100)_2 \end{array}$$

Paso 3: Convertir el resultado en binario del registro AX a hexadecimal.

$$\begin{array}{l} AX = \underline{(1000)_2 \mid (1000)_2 \mid (1000)_2 \mid (0100)_2} \\ AX = \quad \quad 8 \quad \quad 0 \quad \quad 8 \quad \quad 4 \\ AX = (8084)_{16} \end{array}$$

Resultado de la ejecución del programa:

```
AX=A28E BX=C4F5 CX=0000 DX=0000
DS=06B0 ES=06B0 SS=06B0 CS=06B0
06B0:0106 21D8 AND
-t
AX=8084 BX=C4F5 CX=0000 DX=0000
DS=06B0 ES=06B0 SS=06B0 CS=06B0
06B0:0108 CD20 INT
```

The image features a light blue background with a large, faint circular graphic in the center. The corners are decorated with stylized circuit board traces and nodes. The main text is centered horizontally and reads:

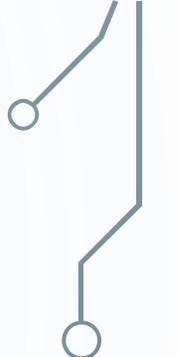
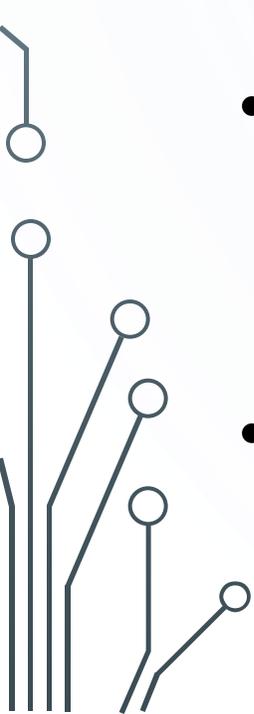
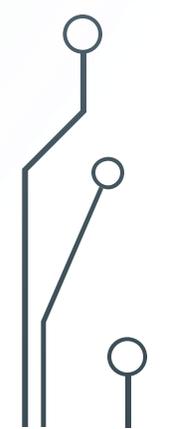
2. ELEMENTOS DE UN SISTEMA DE CÓMPUTO.

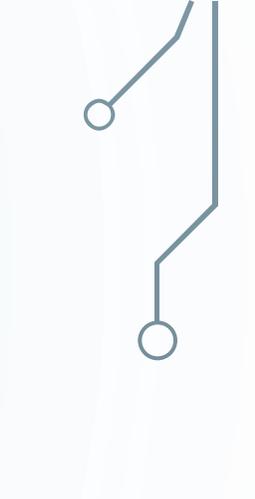
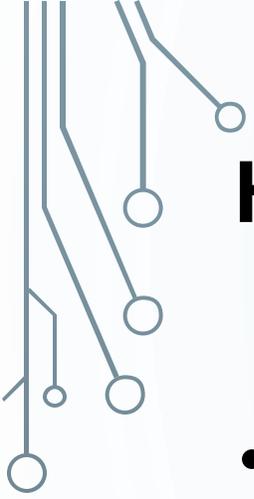


HARDWARE

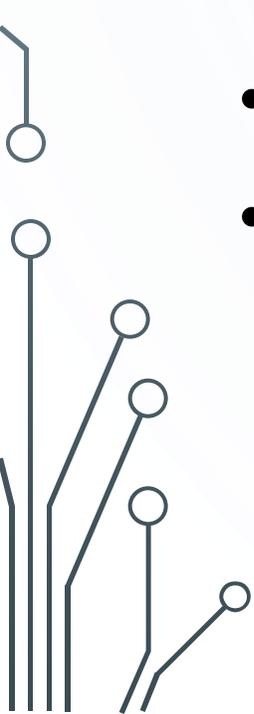
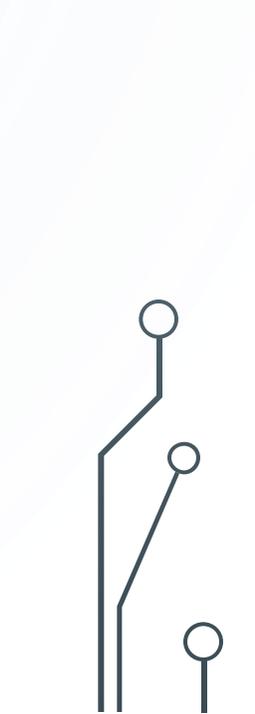
HARDWARE INTERNO

- La tarjeta madre es el componente más importante, funciona como una plataforma para integrar y conectar todos los demás elementos.
- El microprocesador: es el cerebro de la computadora, es un circuito conformado por millones de componentes electrónicos miniaturizados. Constituye el CPU y es el encargado de ejecutar los programas.
- La memoria RAM: es la memoria de acceso aleatorio, es donde se guardan los datos y programas que se están utilizando en el momento, y su almacenamiento es temporal.

- 
- 
- 
- 
- La memoria ROM: es la memoria de solo lectura. Contiene el BIOS, es un programa que al encender la máquina realiza el inventario del hardware conectado a ella y efectúa una prueba llamada POST y sirve para comprobar que el equipo funciona correctamente.
 - La pila: es una pequeña batería de litio, con capacidad de 3 voltios que acumula energía mientras la máquina está encendida.
 - El bus de datos: son los canales de comunicación por donde se transportan los datos entre los componentes de la computadora.



HARDWARE EXTERNO

- Gabinete.
 - Monitor.
 - Teclado.
 - Mouse.
 - Bocinas.
- 
- 

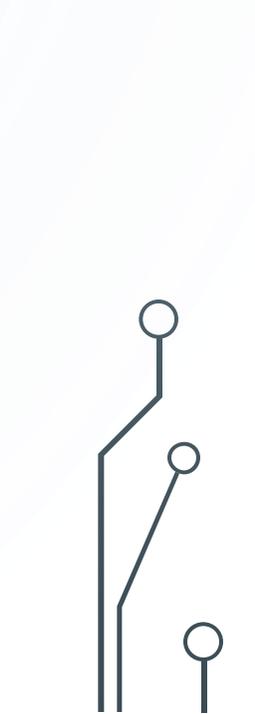
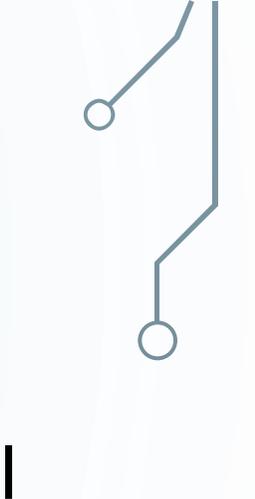
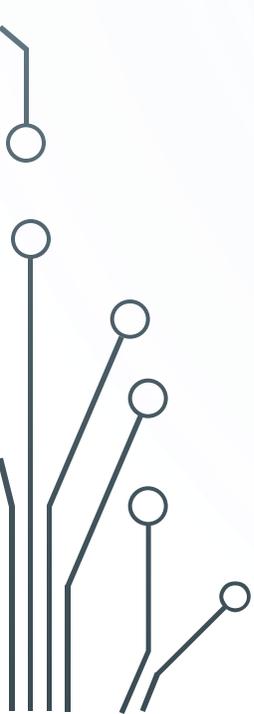


SOFTWARE



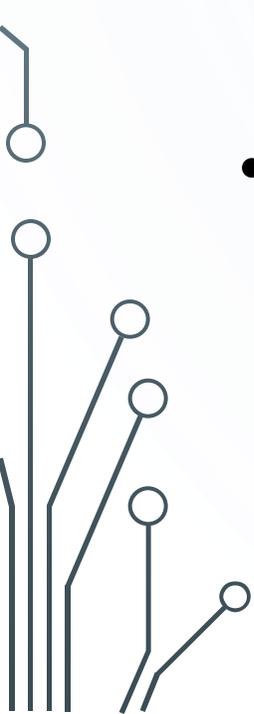
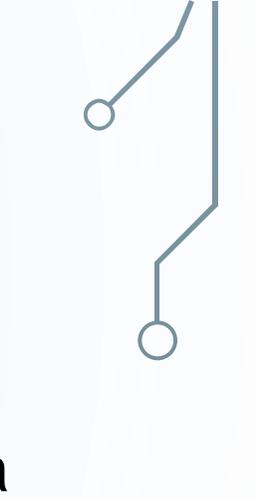
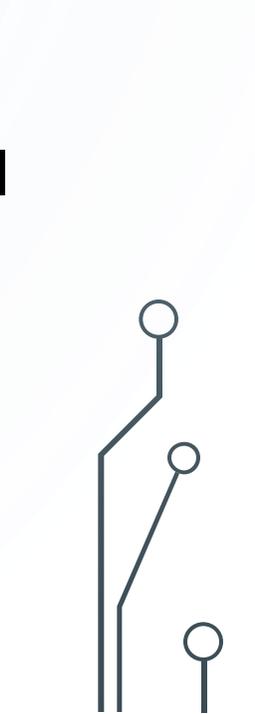
SOFTWARE DE SISTEMA

Es un software que sirve para controlar e interactuar con el sistema operativo, proporcionando control sobre el hardware y dando soporte a otros programas.





EJEMPLOS:

- El firmware de la computadora proporciona la funcionalidad básica para operar y controlar el hardware.
 - El sistema operativo es el programa que toma el control de la máquina y administra sus procesos y recursos.
 - Las utilerías ayudan a analizar, configurar, optimizar y mantener el equipo funcionando correctamente.
- 
- 
- 

SOFTWARE DE APLICACIÓN

Son los programas que le sirven al usuario para realizar tareas específicas.

Ejemplos:

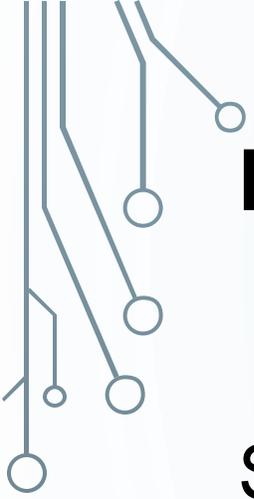
- Microsoft Word.
- Photoshop.
- Skype.
- Google.
- Sketchup.

SOFTWARE DE PROGRAMACIÓN

Son los programas que sirven para crear otros programas.

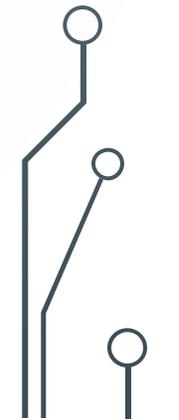
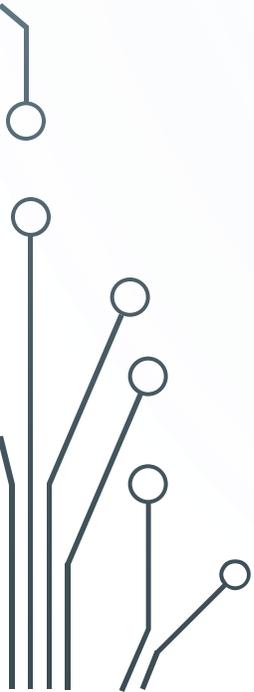
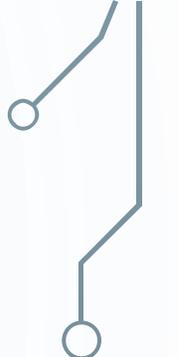
Ejemplos:

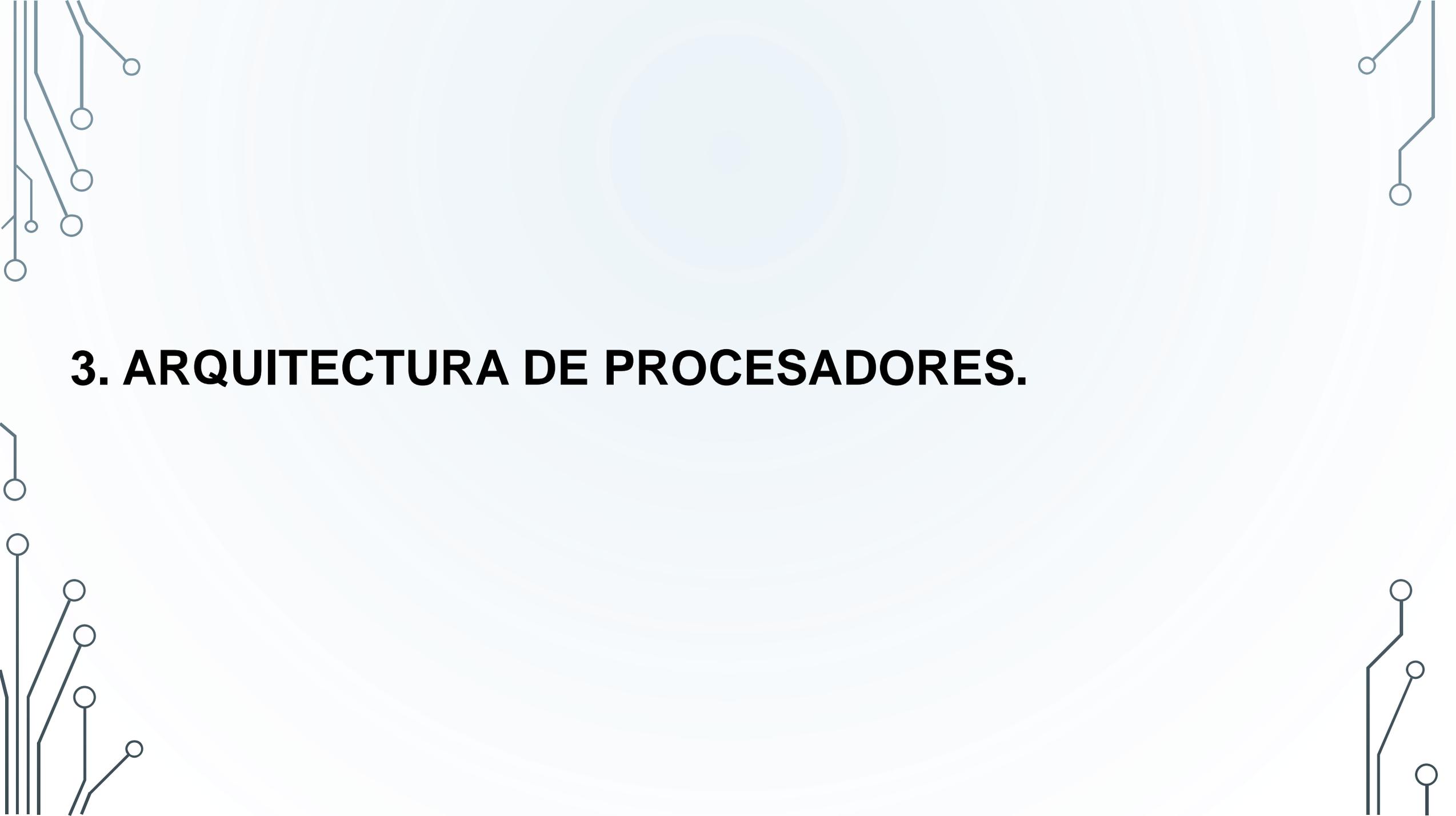
- Visual Basic.
- Java.
- Python.
- Cobol.
- Ensamblador.
- C/C++.



HUMANWARE

Son las personas que diseñan y utilizan las computadoras.

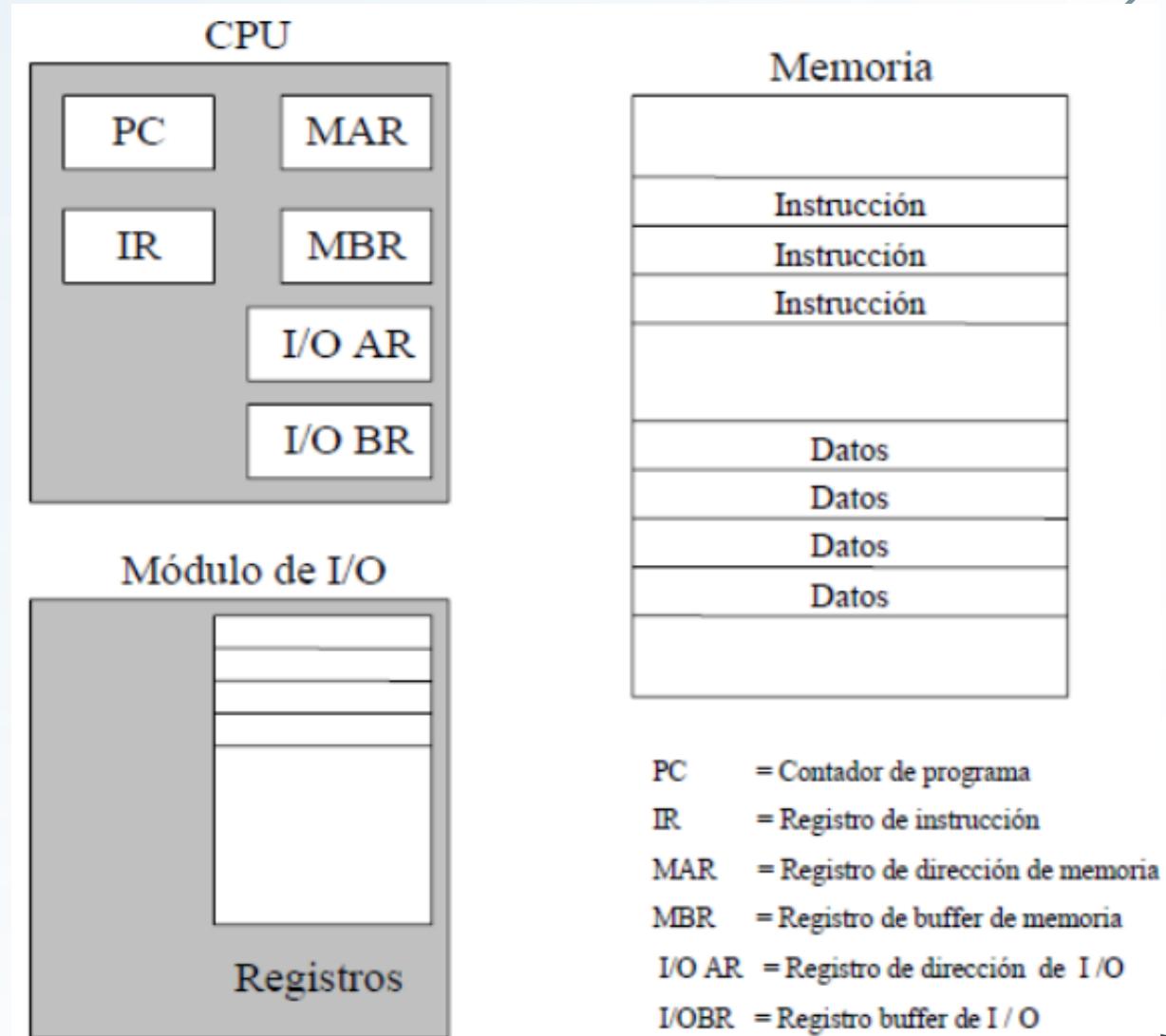


The image features a light blue background with a large, faint circular graphic in the center. The corners are decorated with stylized circuit board traces and nodes. The main text is centered horizontally and reads:

3. ARQUITECTURA DE PROCESADORES.

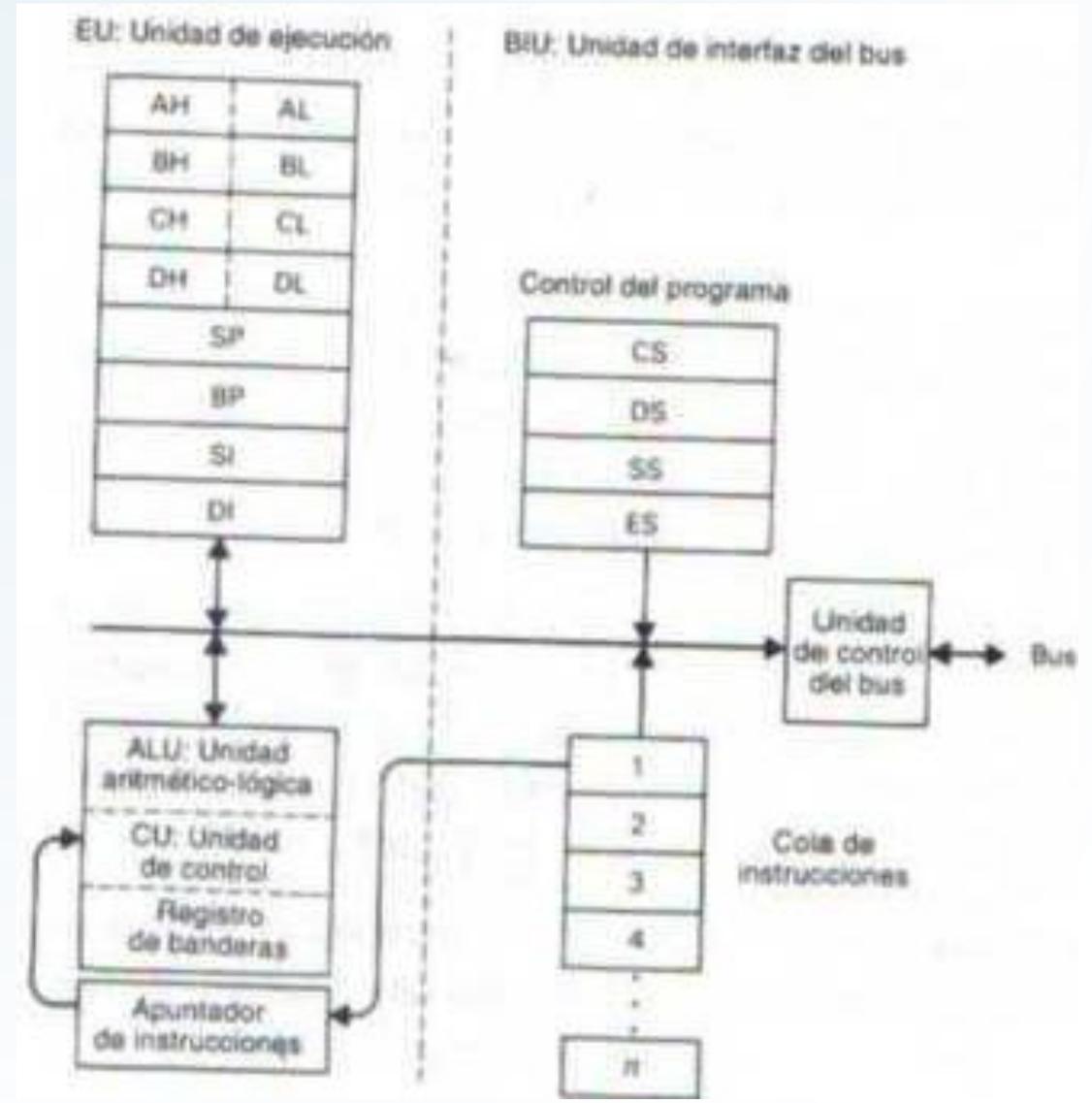
ARQUITECTURA X86

La familia x86 está diseñada siguiendo el modelo de arquitectura de Von Newman, la cual consta de unidad de CPU, memorias, dispositivos de I/O como se muestran a continuación:

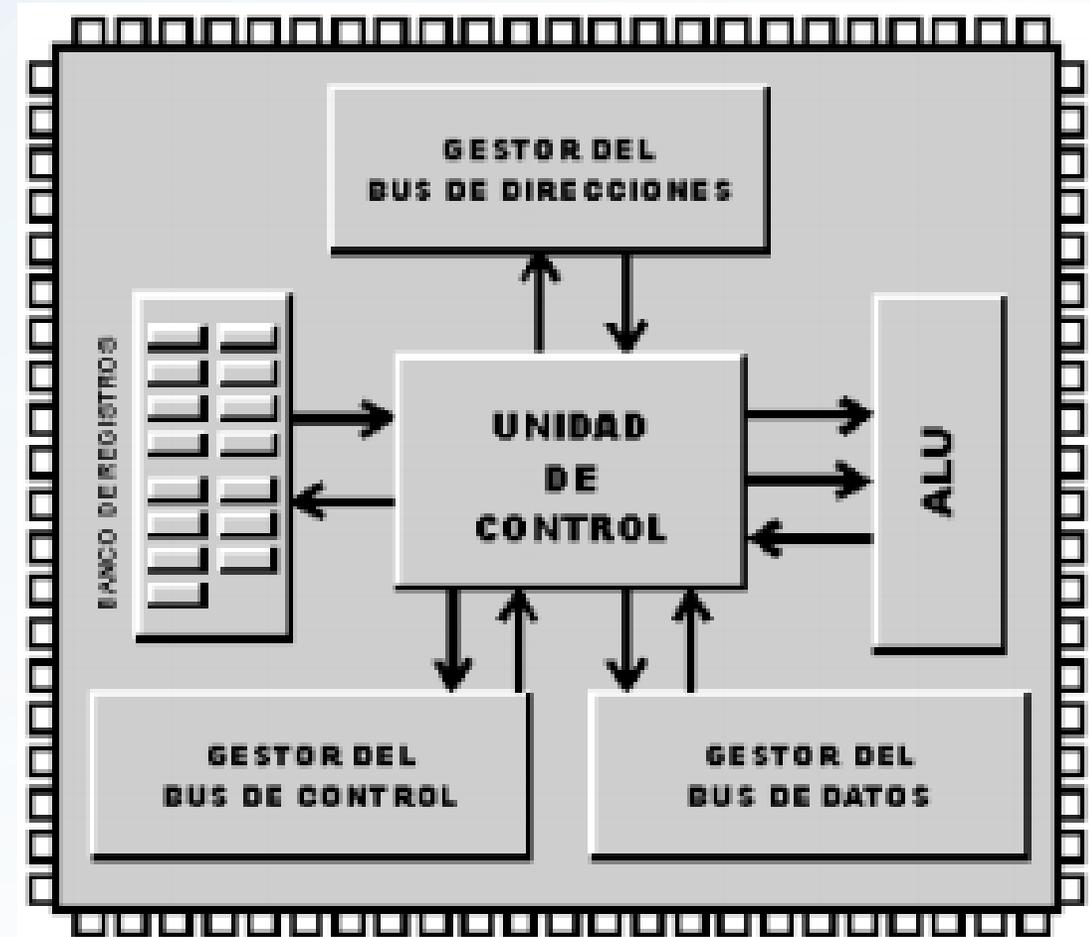


UNIDAD DE EJECUCIÓN (EU) Y UNIDAD DE INTERFAZ DEL BUS (BIU)

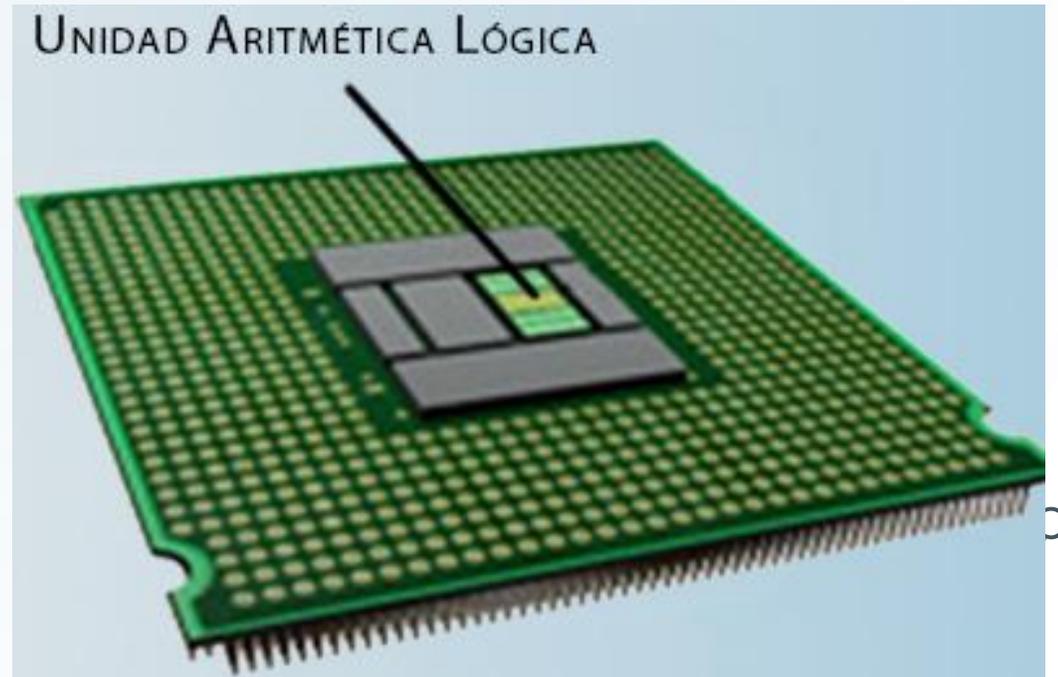
El procesador se divide en 2 unidades lógicas: La EU ejecuta instrucciones y operaciones aritméticas y lógicas. Mientras que la BIU envía instrucciones y datos a la EU.



El microprocesador es el que se encarga de realizar cálculos y transferencia de datos como respuesta a las peticiones de los programas almacenados en memoria. Este esta conformado por la Unidad de Control, la Unidad de Proceso o Unidad Aritmética Lógica (ALU) y los Registros.

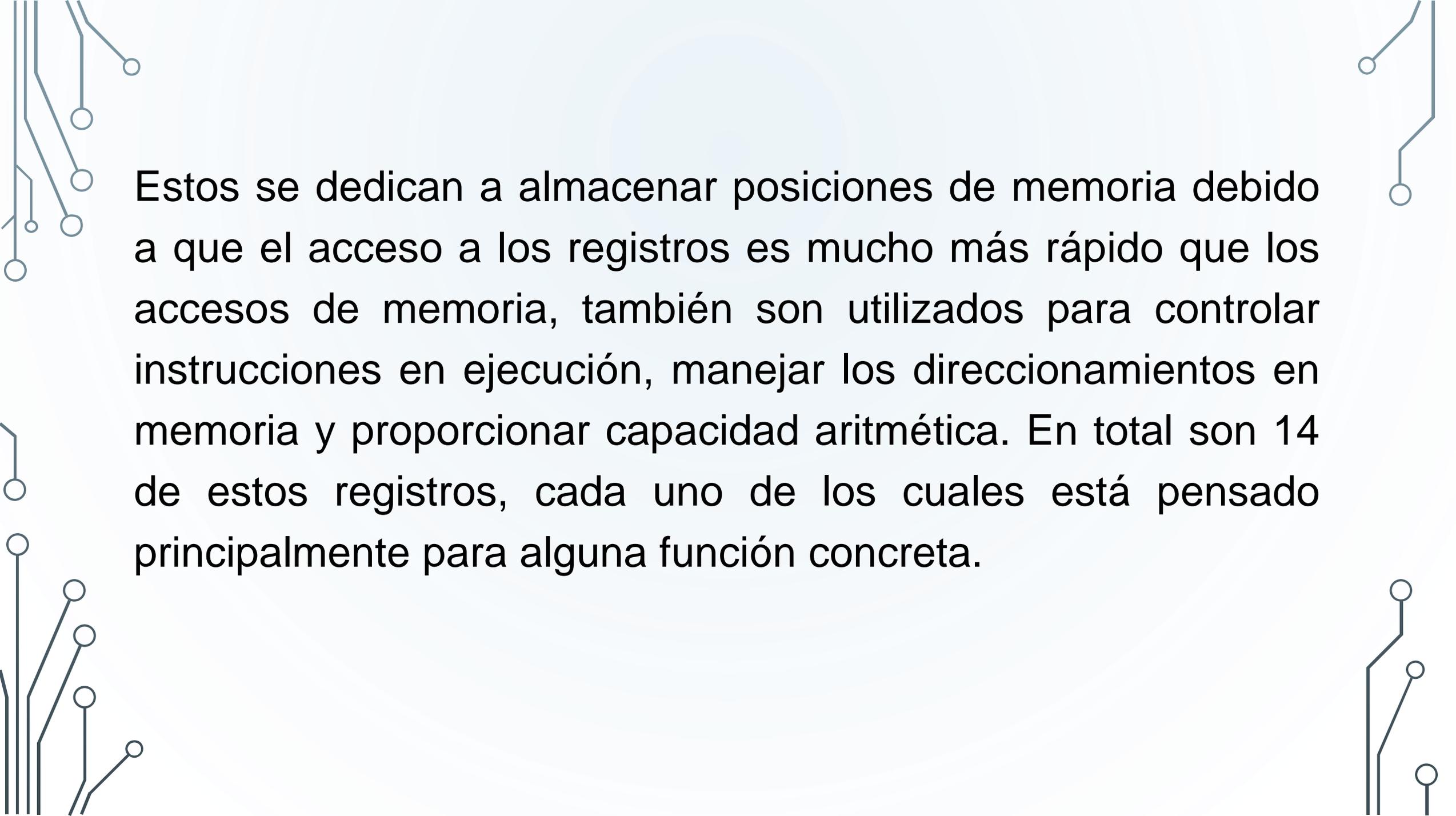


La ALU es la que se encarga de ejecutar las operaciones aritméticas y lógicas. Los registros son dispositivos de un ordenador para el almacenamiento temporal de datos.



The image features a light blue background with a large, faint circular graphic in the center. The corners are decorated with stylized circuit board traces and nodes. The main text is centered in a bold, black, sans-serif font.

LOS REGISTROS DE LA CPU

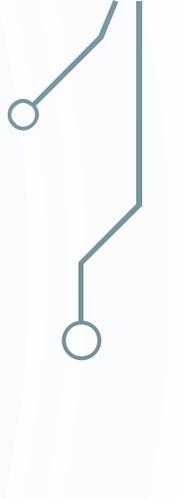
The image features a light blue background with decorative circuit-like lines in the corners. These lines consist of thin, greyish-blue paths that branch out and terminate in small circles, resembling a stylized PCB or network diagram. The lines are positioned in the top-left, top-right, bottom-left, and bottom-right corners, framing the central text.

Estos se dedican a almacenar posiciones de memoria debido a que el acceso a los registros es mucho más rápido que los accesos de memoria, también son utilizados para controlar instrucciones en ejecución, manejar los direccionamientos en memoria y proporcionar capacidad aritmética. En total son 14 de estos registros, cada uno de los cuales está pensado principalmente para alguna función concreta.



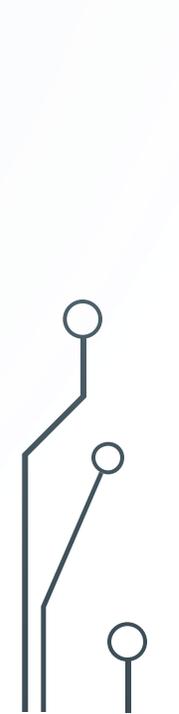
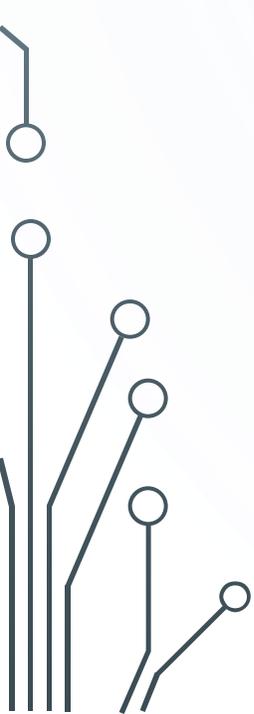
```

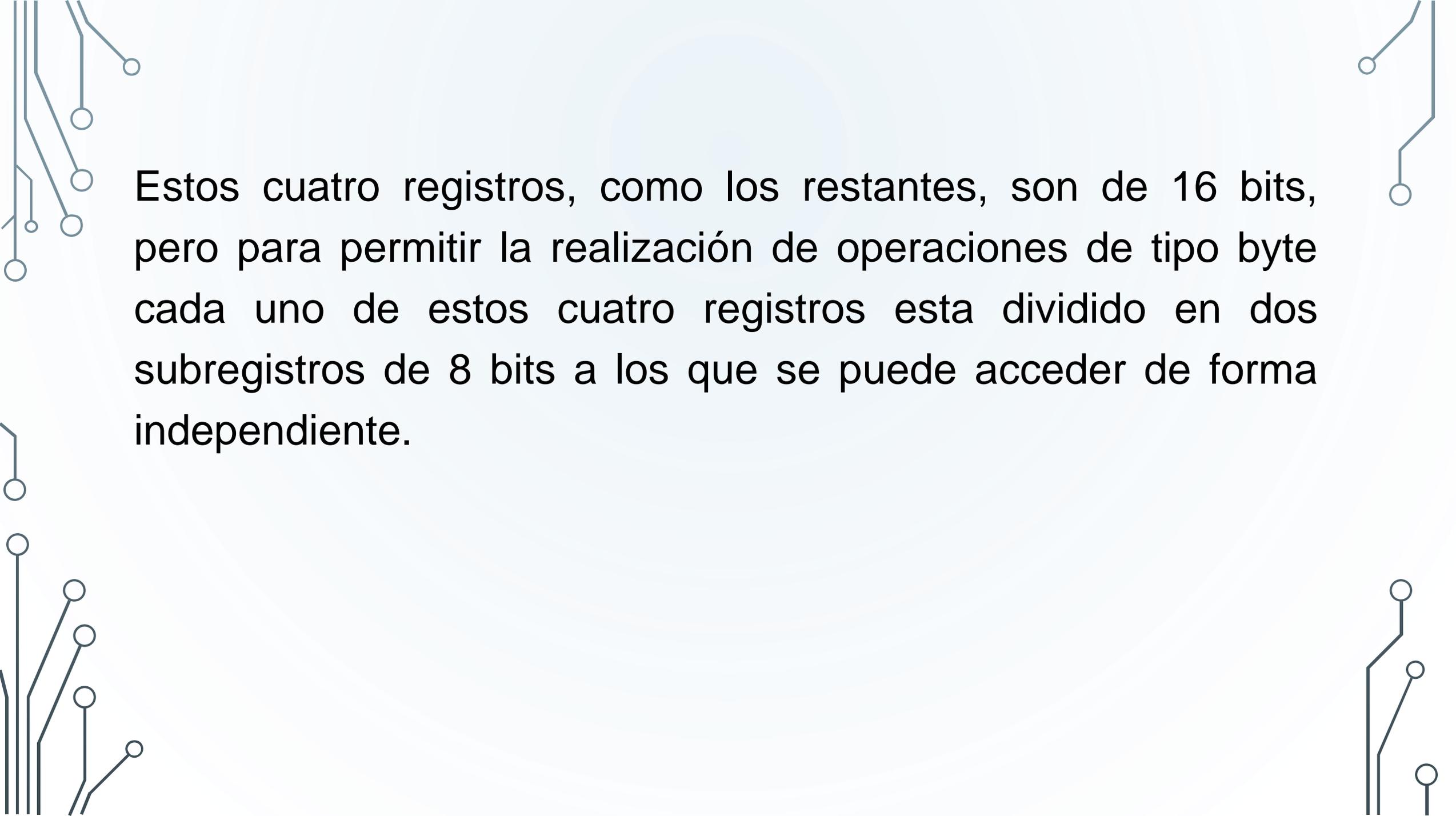
AX=0050 BX=0005 CX=0010 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0CF6 ES=0CF6 SS=0CF6 CS=0CF6 IP=0108  NU UP EI PL NZ NA PO NC
  
```



REGISTROS DE PROPÓSITO GENERAL O DE ALMACENAMIENTO TEMPORAL

Estos son únicos en el sentido de que se les puede direccionar como una palabra o como un byte.



The image features a light blue background with decorative circuit board patterns in the corners. These patterns consist of thin, dark blue lines representing traces, with small white circles at various points representing vias or components. The patterns are located in the top-left, top-right, bottom-left, and bottom-right corners, framing the central text.

Estos cuatro registros, como los restantes, son de 16 bits, pero para permitir la realización de operaciones de tipo byte cada uno de estos cuatro registros está dividido en dos subregistros de 8 bits a los que se puede acceder de forma independiente.

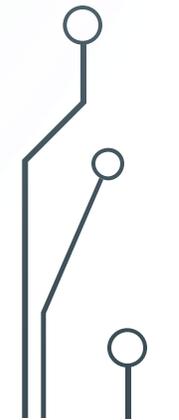
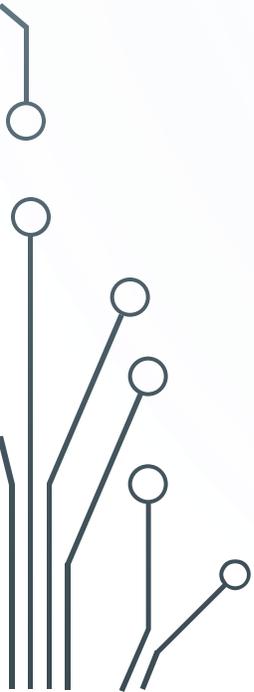
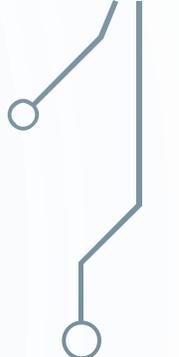
AX (REGISTRO ACUMULADOR)

Este registro es usado en operaciones aritméticas como primer operando y también como registro de propósito general a disposición del programador. AX está dividido en AL que son los 8 bits inferiores y AH que son los 8 bits superiores. En consecuencia los demás registros de propósito general se dividen de la misma manera.



BX (REGISTRO BASE)

Se usa principalmente para indicar posiciones de memoria.



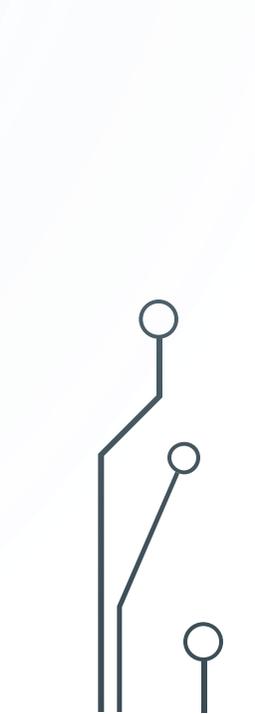
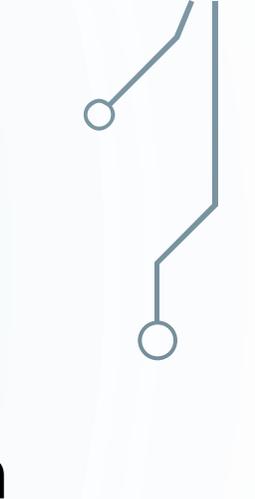
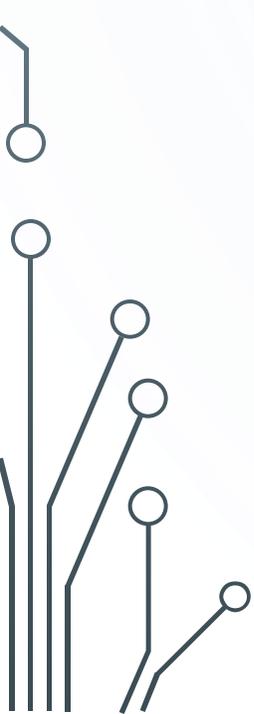
CX (REGISTRO CONTADOR)

Puede contener un valor para controlar el número de veces que un ciclo se repite o un valor para corrimiento de bits.



DX (REGISTRO DE DATOS)

Se usa como registro auxiliar en operaciones aritméticas con cifras grandes y como contenedor de datos a la hora de usar instrucciones de comunicación de puertos.



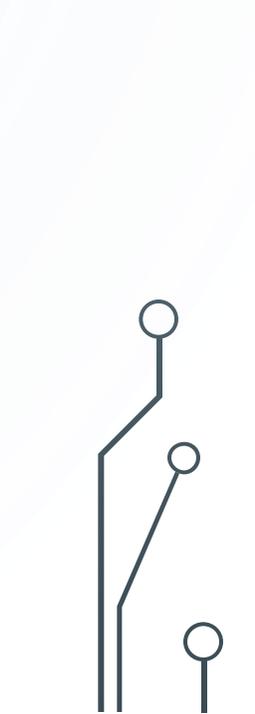
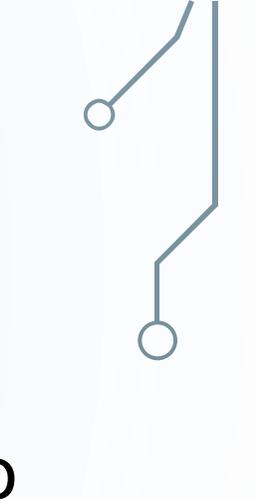
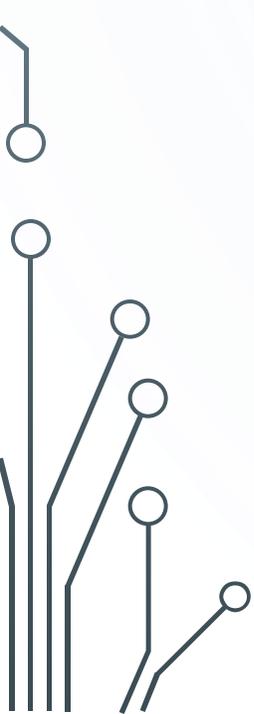
REGISTROS DE SEGMENTO

Son cuatro registros de 16 bits los cuales definen áreas de 64 Kbits dentro del espacio de direcciones de memoria. Estas áreas pueden solaparse total o parcialmente. No es posible acceder a una posición de memoria no definida por algún segmento: si es preciso, habrá de moverse alguno.



CS (SEGMENTO DE CÓDIGO)

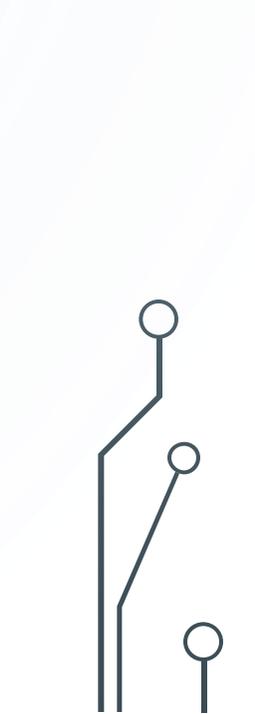
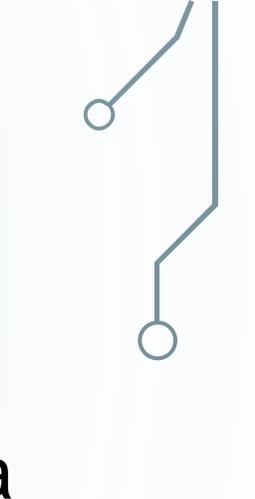
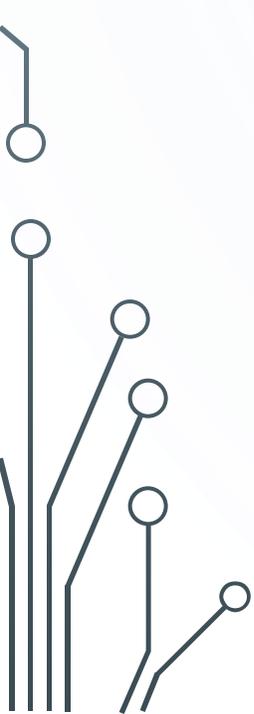
Este registro es usado por el procesador, junto con el registro IP, para conocer la ubicación de la instrucción que esta siendo ejecutada.





DS (SEGMENTO DE DATOS)

Se usa para indicar dónde están todos los datos del programa que se está ejecutando.



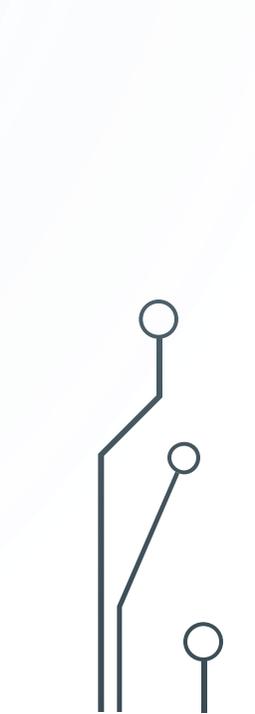
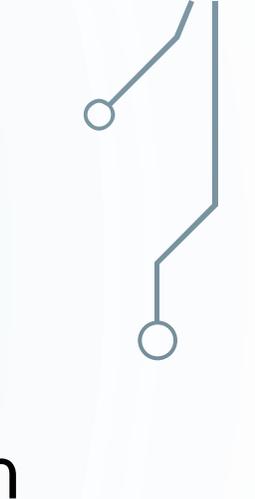
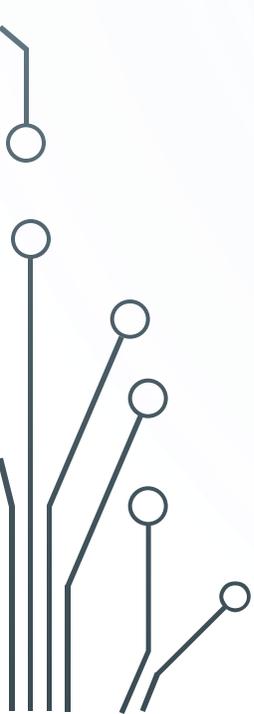
SS (SEGMENTO DE PILA)

En este registro se indica al procesador dónde esta la zona de memoria que se usa como segmento de pila, la cual se utilizara para el almacenamiento temporal de direcciones y datos.



ES (SEGMENTO EXTRA)

Es utilizado como apuntador de memoria auxiliar en operaciones complejas donde se necesitan dos punteros de datos simultáneos.



REGISTROS ÍNDICE

Son utilizados como desplazamientos complementarios para DS y ES a la hora de indicar la posición donde se encuentran los datos a los que se desea acceder.

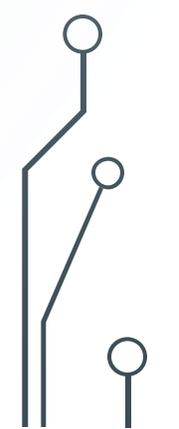
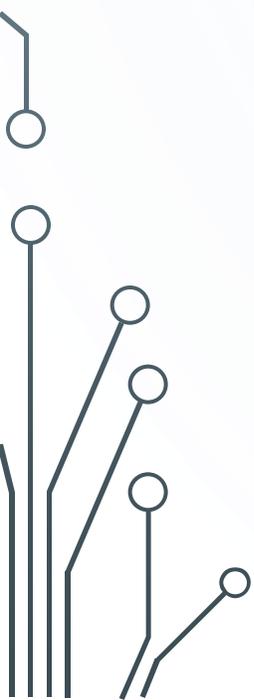
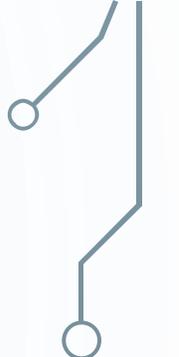
SI (ÍNDICE FUENTE)

Se usa como puntero origen en operaciones de desplazamiento de datos entre dos zonas de memoria.



DI (ÍNDICE DESTINO)

Utilizado como puntero de destino en operaciones de desplazamiento de datos entre zonas de memoria.



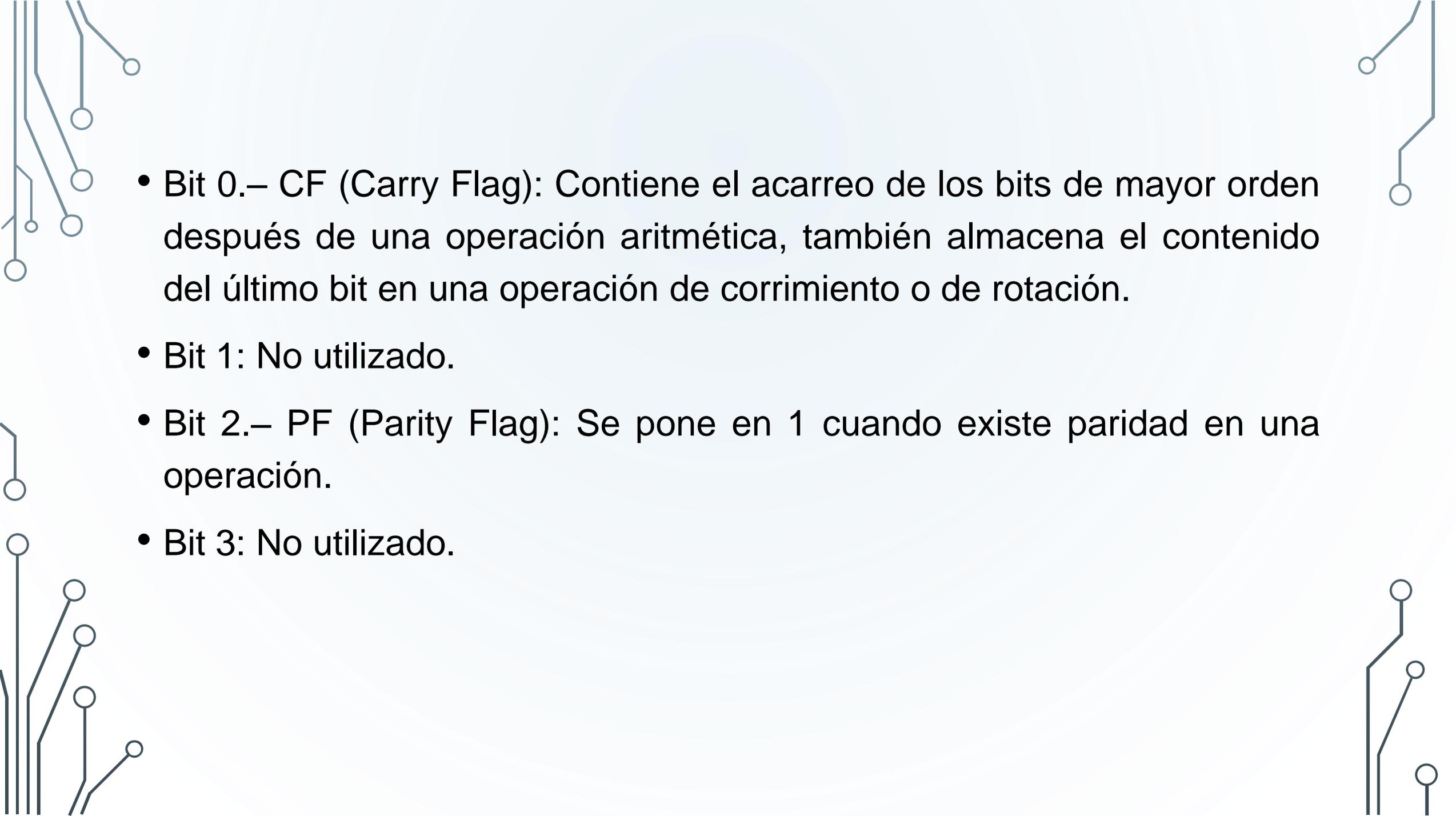
REGISTRO DE INSTRUCCIÓN

Solo hay uno, el registro IP (Puntero Instrucción), el cual es utilizado por la CPU para conocer la posición relativa a la base CS donde se encuentra la instrucción que se esta ejecutando en ese momento. Este puntero cambia automáticamente cada vez que se ejecuta una instrucción o se realiza un salto a otro punto del programa a causa de una instrucción de salto.

REGISTRO BANDERA

Este es un registro de 16 bits, de los cuales 9 sirven para indicar el estado actual de la máquina y el resultado del procesamiento. Muchas instrucciones aritméticas y de comparación cambian el estado de las banderas y apoyándose de ellas determinan la acción subsecuente:



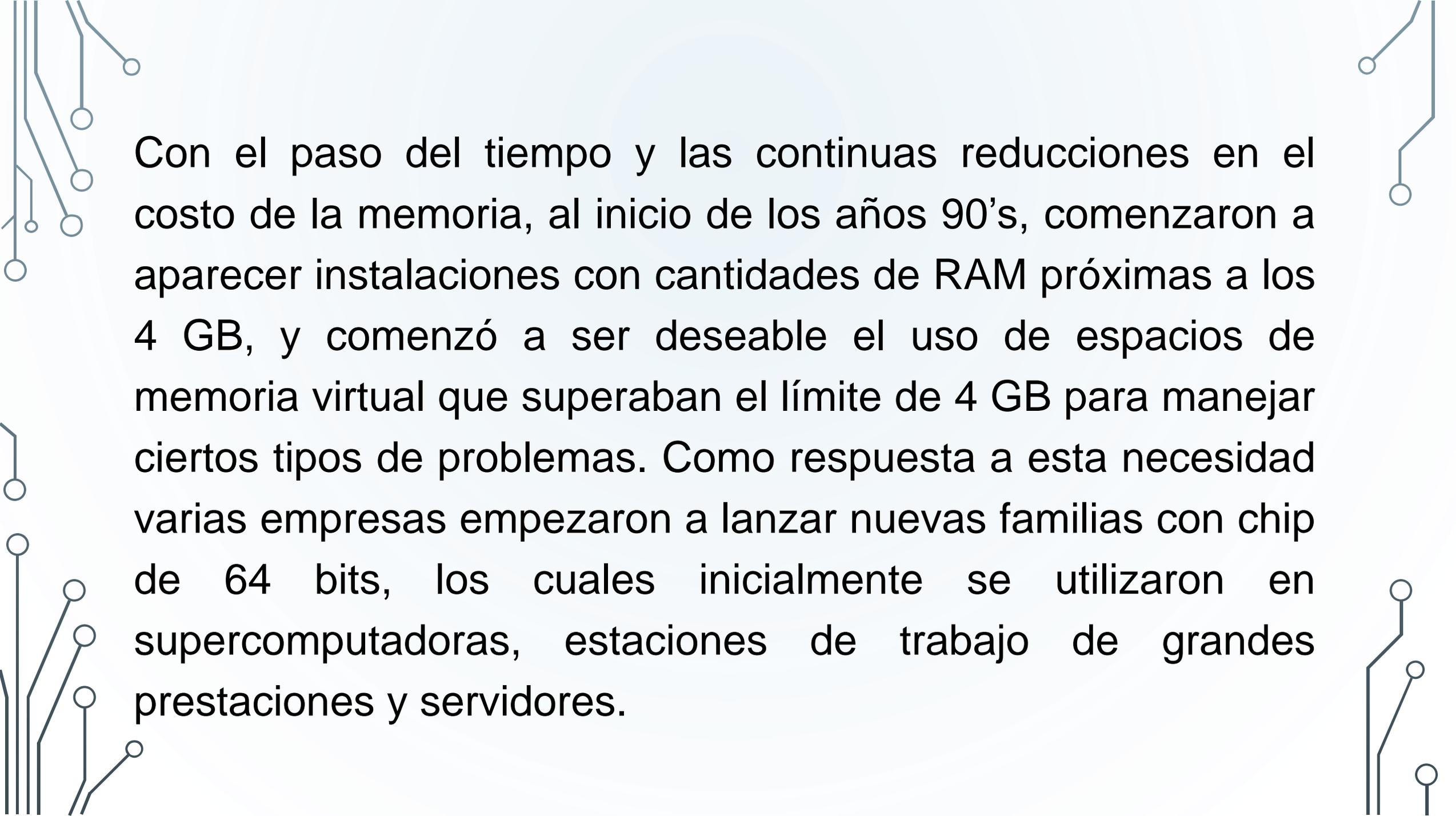
- 
- Bit 0.– CF (Carry Flag): Contiene el acarreo de los bits de mayor orden después de una operación aritmética, también almacena el contenido del último bit en una operación de corrimiento o de rotación.
 - Bit 1: No utilizado.
 - Bit 2.– PF (Parity Flag): Se pone en 1 cuando existe paridad en una operación.
 - Bit 3: No utilizado.

- Bit 4.– AF (Auxiliar Flag): Contiene un acarreo auxiliar del bit 3 en un dato de 8 bits, para aritmética especializada. Se pone en 1 cuando existe la necesidad de realizar ajustes tras una operación de tipo BCD.
- Bit 5: No utilizado.
- Bit 6.– ZF (Zero Flag): Indica el resultado de una operación lógica o de comparación.
- Bit 7.– SF (Sign Flag): Contiene el signo resultante de una operación aritmética (0=positivo, 1=negativo).
- Bit 8.– TF (Trap Flag): Permite la operación del procesador en modo de depuración (paso a paso).

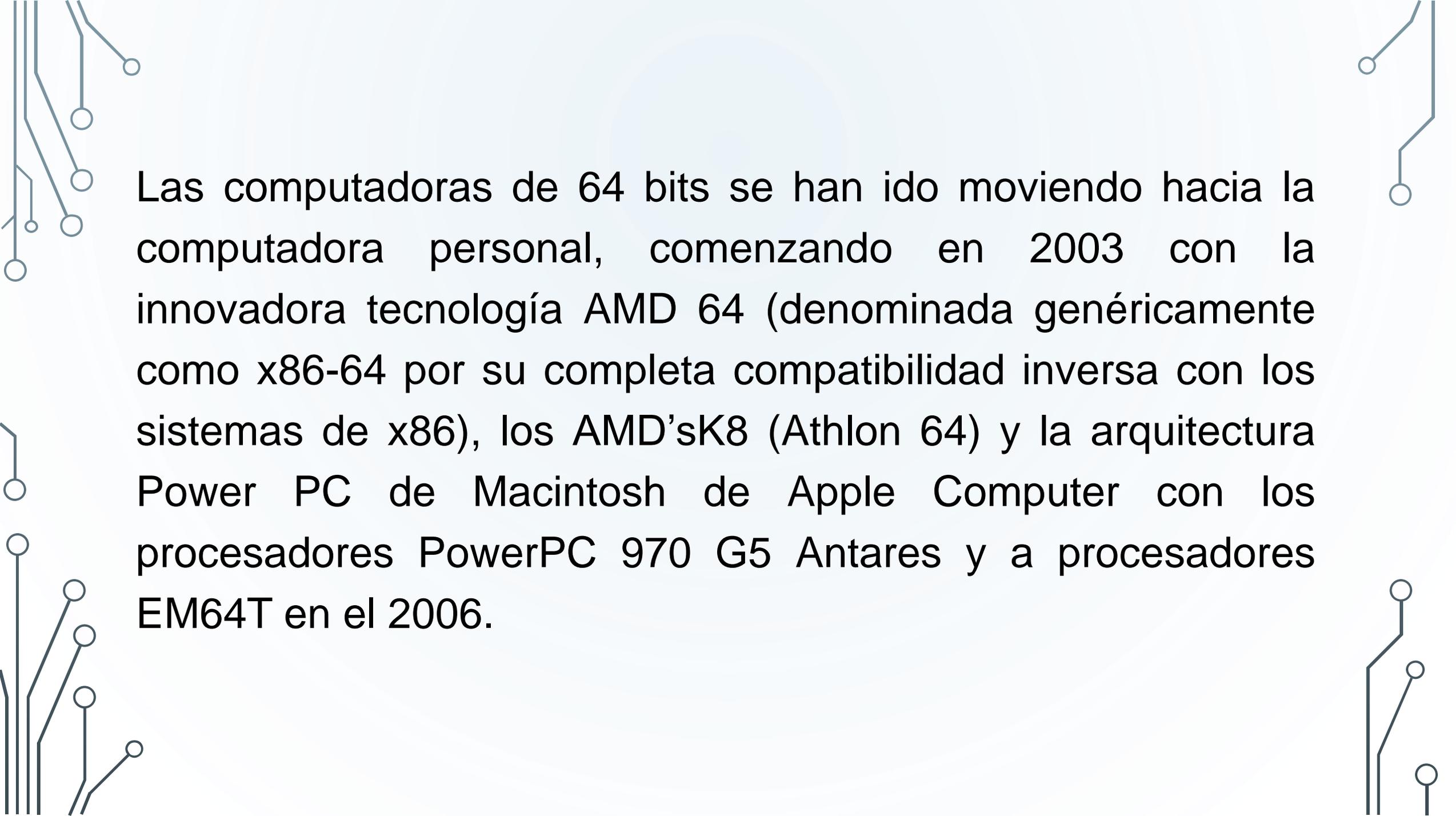
- Bit 9.– IF (Interrupt Flag): Si está a 1, indica que esta permitida la generación de interrupciones externas.
- Bit 10.– DF (Direction Flag): Indica a la CPU hacia donde se desplazan los punteros relativos en operaciones repetitivas de cadenas de datos (1=Decremento automático, 0=aumento).
- Bit 11.– OF (Overflow Flag): Indica desbordamiento del bit de mayor orden después de una operación aritmética de números signados (1=existe overflow, 0=no existe overflow).
- Bit 12 – 15: No utilizados.

The image features a light blue background with a large, faint circular graphic in the center. The corners are decorated with stylized circuit board traces and nodes. The central text is in a bold, black, sans-serif font.

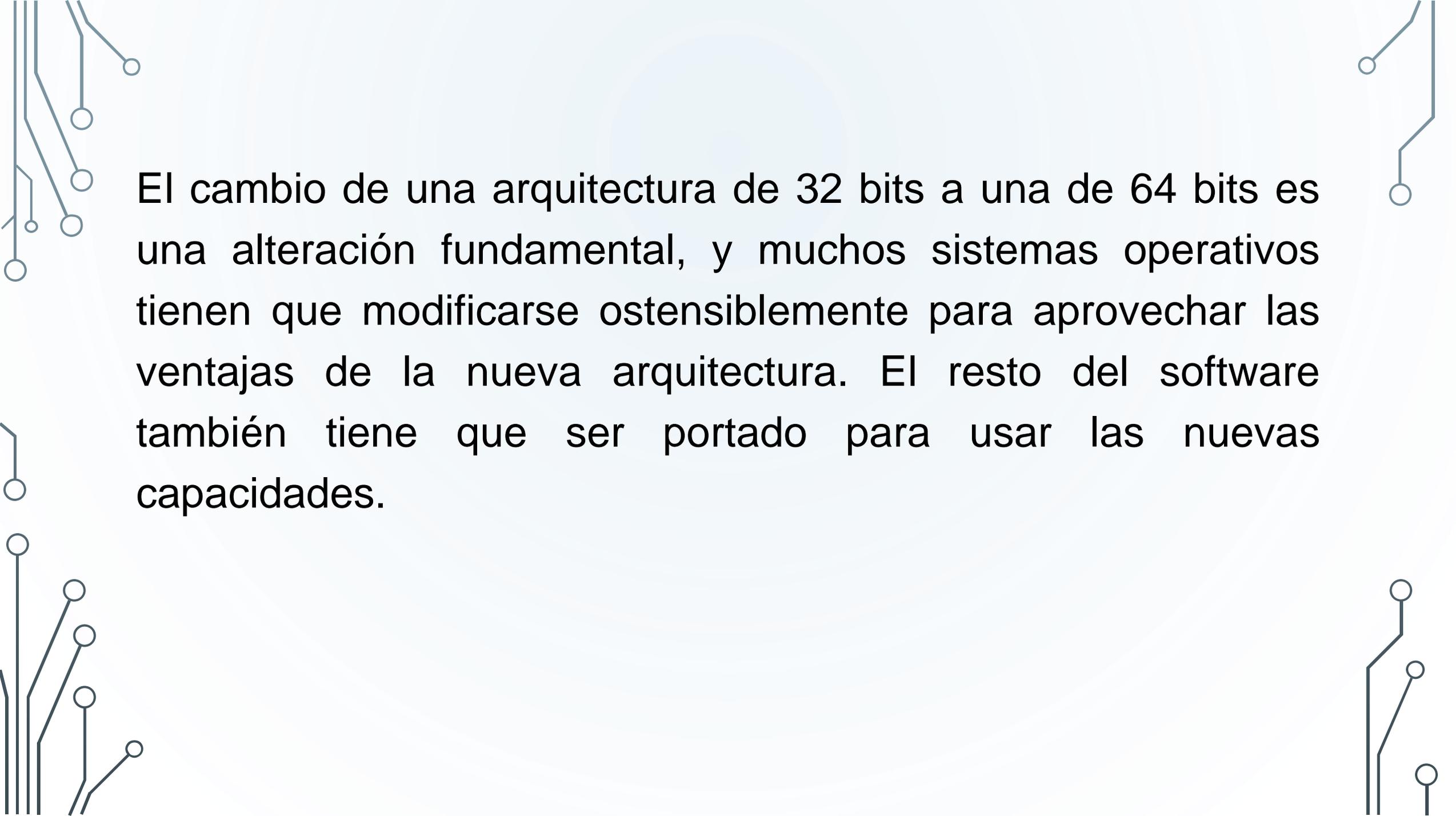
ARQUITECTURA X64

The image features a white background with decorative circuit-like lines in the corners. These lines are composed of thin grey lines that branch out and terminate in small grey circles, resembling a stylized PCB or network diagram. The lines are located in the top-left, top-right, bottom-left, and bottom-right corners, framing the central text.

Con el paso del tiempo y las continuas reducciones en el costo de la memoria, al inicio de los años 90's, comenzaron a aparecer instalaciones con cantidades de RAM próximas a los 4 GB, y comenzó a ser deseable el uso de espacios de memoria virtual que superaban el límite de 4 GB para manejar ciertos tipos de problemas. Como respuesta a esta necesidad varias empresas empezaron a lanzar nuevas familias con chip de 64 bits, los cuales inicialmente se utilizaron en supercomputadoras, estaciones de trabajo de grandes prestaciones y servidores.

The image features a light blue background with decorative circuit-like lines in the corners. These lines consist of thin, greyish-blue paths that branch out and terminate in small circles, resembling a stylized PCB or network diagram. The lines are positioned in the top-left, top-right, bottom-left, and bottom-right corners, framing the central text.

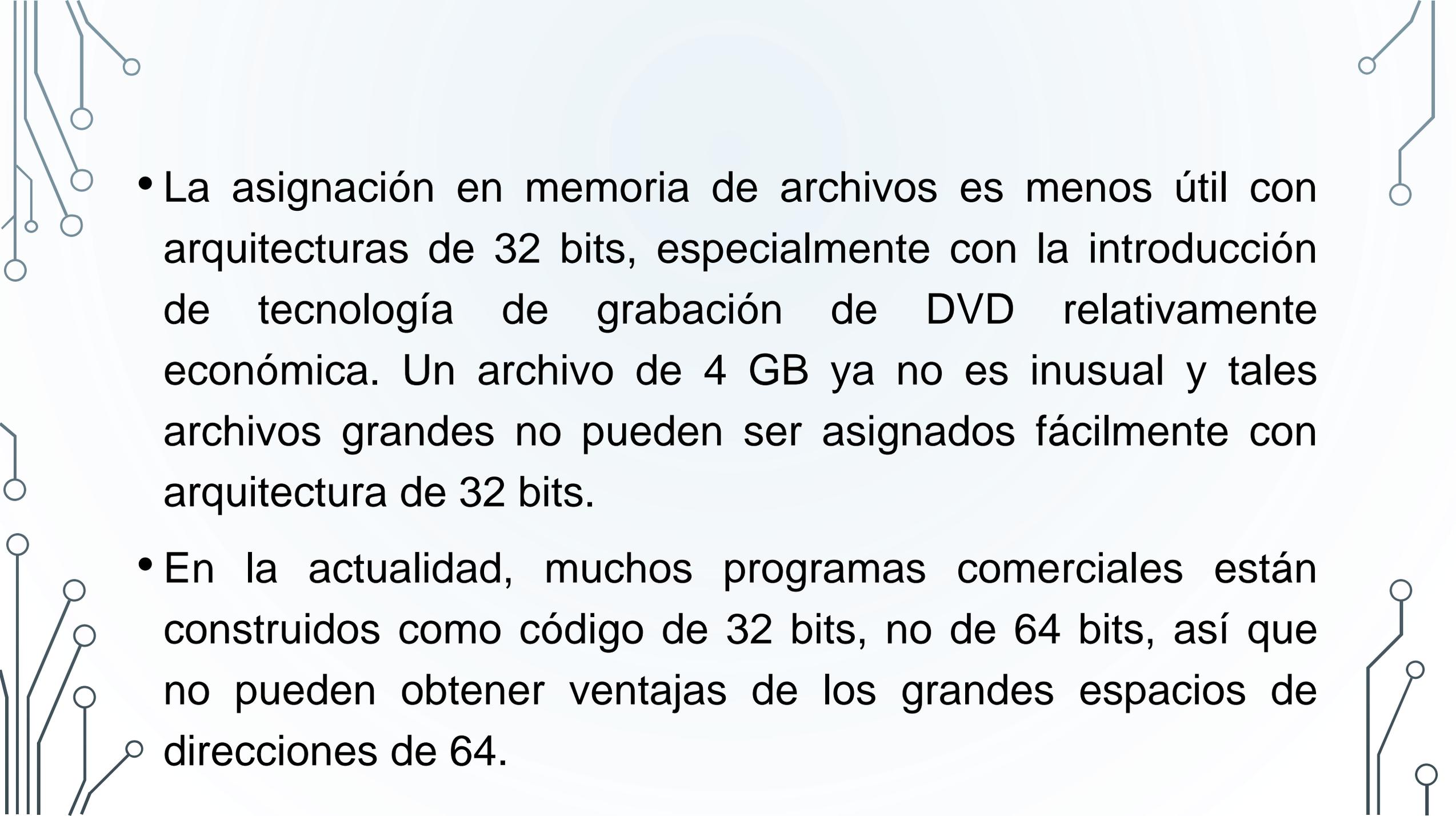
Las computadoras de 64 bits se han ido moviendo hacia la computadora personal, comenzando en 2003 con la innovadora tecnología AMD 64 (denominada genéricamente como x86-64 por su completa compatibilidad inversa con los sistemas de x86), los AMD'sK8 (Athlon 64) y la arquitectura Power PC de Macintosh de Apple Computer con los procesadores PowerPC 970 G5 Antares y a procesadores EM64T en el 2006.

The image features a light blue background with decorative circuit-like lines in the corners. These lines consist of thin, greyish-blue paths that branch out and terminate in small, hollow circles, resembling a stylized PCB or network diagram. The lines are positioned in the top-left, top-right, bottom-left, and bottom-right corners, framing the central text.

El cambio de una arquitectura de 32 bits a una de 64 bits es una alteración fundamental, y muchos sistemas operativos tienen que modificarse ostensiblemente para aprovechar las ventajas de la nueva arquitectura. El resto del software también tiene que ser portado para usar las nuevas capacidades.

PROS Y CONTRAS

- La principal desventaja de las arquitecturas de 64 bits es que, con respecto a las de 32 bits, los mismos datos ocupan ligeramente más espacio en memoria debido al crecimiento de los punteros.
- Algunos sistemas operativos reservan porciones de espacio de direcciones de procesos para uso del sistema operativo, reduciendo el espacio total de direcciones disponibles para asignar memoria para programas de usuario. Esta restricción solo está presente en las versiones de Windows de 32 bits.

- 
- The image features decorative circuit-like lines in the corners, consisting of thin grey lines with small circles at various points, resembling a stylized PCB or network diagram.
- La asignación en memoria de archivos es menos útil con arquitecturas de 32 bits, especialmente con la introducción de tecnología de grabación de DVD relativamente económica. Un archivo de 4 GB ya no es inusual y tales archivos grandes no pueden ser asignados fácilmente con arquitectura de 32 bits.
 - En la actualidad, muchos programas comerciales están contruidos como código de 32 bits, no de 64 bits, así que no pueden obtener ventajas de los grandes espacios de direcciones de 64.

The image features a light blue background with a large, faint circular graphic in the center. The corners are decorated with stylized circuit board traces and nodes. The main text is centered and reads:

4. MÉTODO DE GESTIÓN DE MEMORIA DEL PROCESADOR SELECCIONADO.

The image features a light blue background with a large, faint circular graphic in the center. The corners are decorated with stylized circuit board traces and nodes. The main text is centered in a bold, black, sans-serif font.

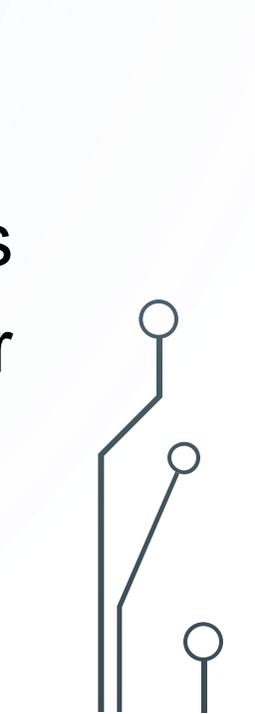
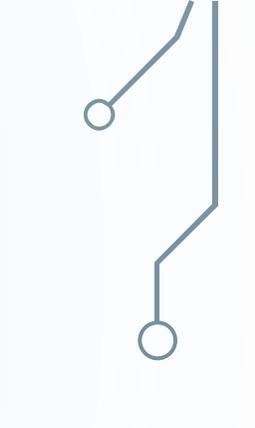
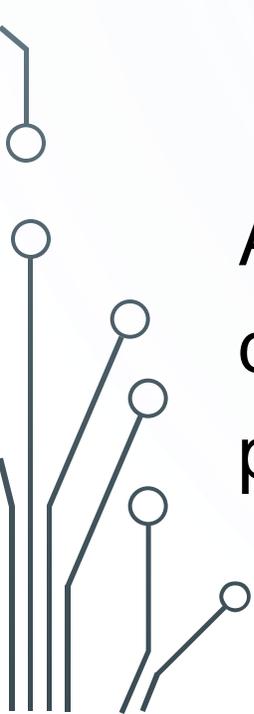
GESTIÓN DE MEMORIA



PROTECCIÓN

Si varios procesos comparten la memoria principal, se debe asegurar que ninguno de ellos pueda modificar posiciones de memoria de otro proceso.

Aunque la escritura de memoria tiene efectos más desastrosos, la lectura de memoria ajena tampoco debe estar permitida, pues cada proceso debe mantener su privacidad.



COMPARTIMIENTO

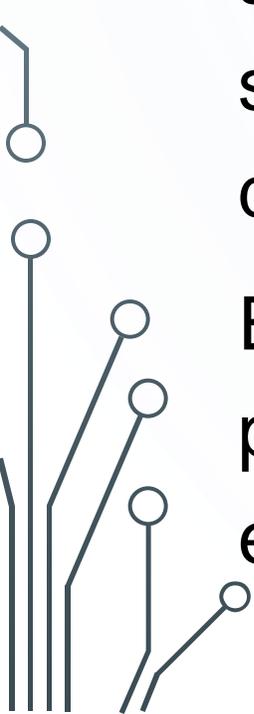
El compartimiento de la memoria parece estar en contradicción con la protección, pero es que a menudo también es necesario que varios procesos puedan compartir y actualizar estructuras de datos comunes. Por ejemplo, en un sistema de bases de datos. En otras ocasiones, lo que se requiere es compartir zonas de código. Por ejemplo, en rutinas de biblioteca, para no tener en memoria distintas copias de la misma rutina. En este caso, se hace necesaria alguna protección para que un proceso no modifique el código de las rutinas.



REUBICACIÓN

La multiprogramación requiere que varios procesos residan simultáneamente en memoria. Lo que no se puede saber antes de llevarlo a memoria es la dirección absoluta en la que se va a cargar el proceso, por lo que no es práctico utilizar direcciones absolutas en el programa.

En su lugar, es preferible realizar direccionamientos relativos para permitir que un programa pueda ser cargado y ejecutado en cualquier parte de la memoria.



ORGANIZACIÓN DE MEMORIA

La memoria se debe organizar tanto física como lógicamente. Debido al costo de la rápida memoria RAM, normalmente se necesita ampliarla con memoria secundaria más barata y más lenta, utilizando para ello dispositivos tales como discos o cintas magnéticas. Por el contrario, también puede resultar conveniente añadir memoria de acceso más rápido que la RAM principal, como es el caso de la memoria caché, en la que se mantienen los datos de acceso más frecuentes.

The image features a light blue background with a large, faint circular graphic in the center. The corners are decorated with stylized circuit board traces and nodes. The main text is centered horizontally and vertically.

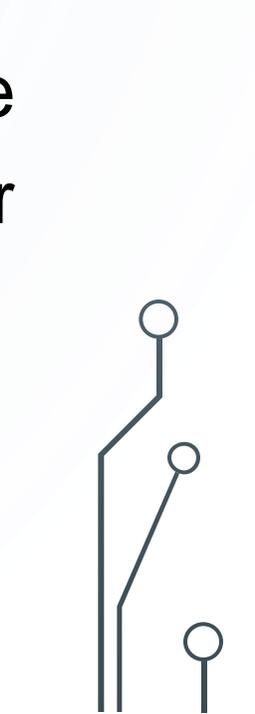
GESTIÓN DE MEMORIA INTERNA

MONO PROGRAMACIÓN

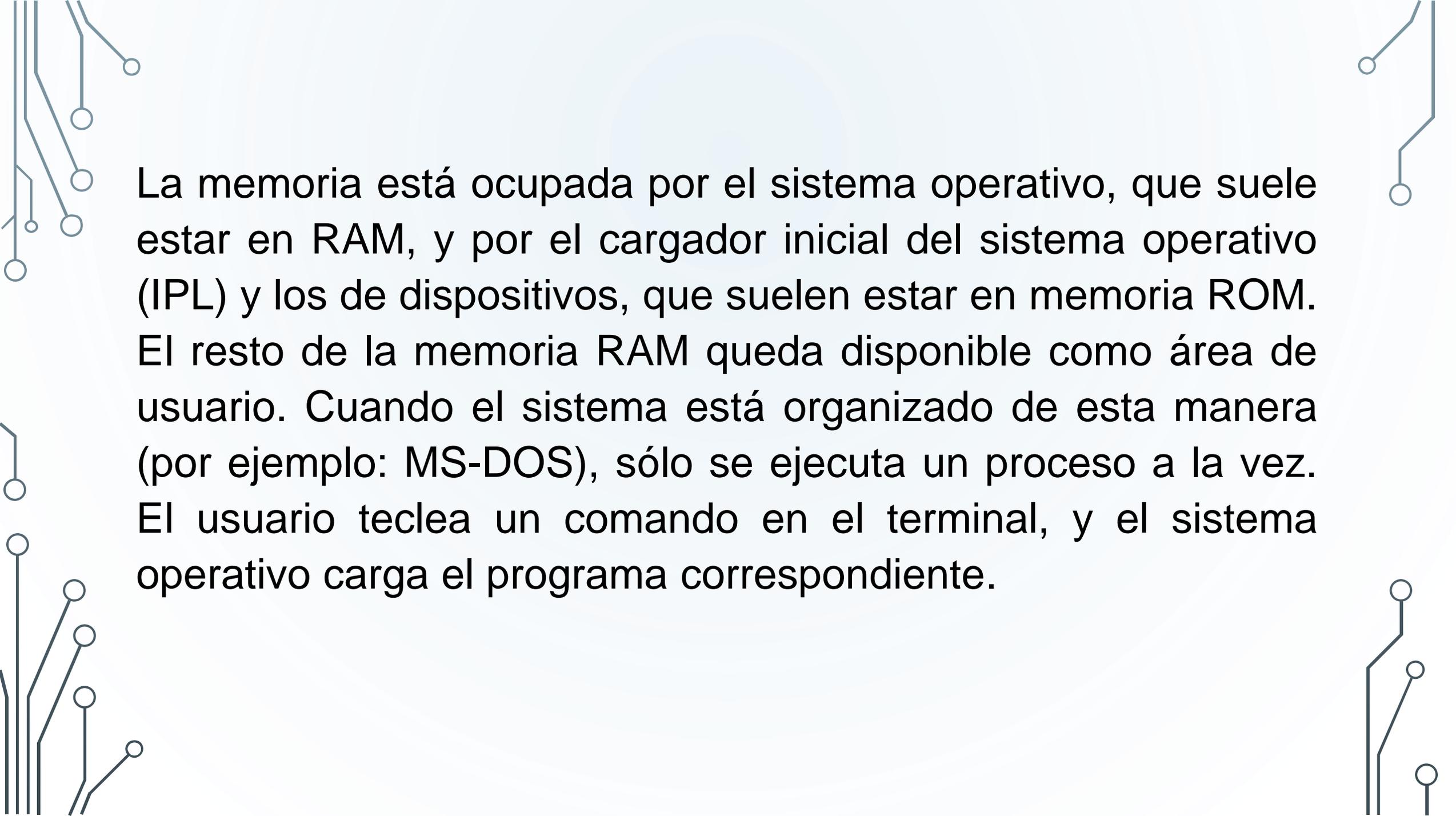
El esquema de memoria más simple consiste en mantener la memoria ocupada con un único proceso. Cuando se carga un programa que se hace cargo de toda la memoria y del control completo de la máquina, se dice que el programa se carga sobre una máquina desnuda es decir, una máquina en la que solamente se ofrece el hardware puro, sin ninguna ayuda software que lo recubra. Las máquinas desnudas se utilizaron de forma general hasta principios de los años 60.



Más tarde, a las máquinas desnudas se les añadió una especie de sistema operativo muy rudimentario al que se le denominó monitor.



Las funciones que ofrecía el monitor eran estrictamente necesarias para cargar un programa en memoria y controlar su ejecución, todo de forma manual.

The image features a light blue background with decorative circuit-like lines in the corners. These lines consist of thin, grey, angular paths that terminate in small, white circles with grey outlines, resembling nodes or components on a printed circuit board. The lines are positioned in the top-left, top-right, bottom-left, and bottom-right corners, framing the central text.

La memoria está ocupada por el sistema operativo, que suele estar en RAM, y por el cargador inicial del sistema operativo (IPL) y los de dispositivos, que suelen estar en memoria ROM. El resto de la memoria RAM queda disponible como área de usuario. Cuando el sistema está organizado de esta manera (por ejemplo: MS-DOS), sólo se ejecuta un proceso a la vez. El usuario teclea un comando en el terminal, y el sistema operativo carga el programa correspondiente.

MULTIPROGRAMACIÓN CON PARTICIONES FIJAS

En general, es deseable que haya varios procesos de usuario residiendo en memoria al mismo tiempo, se hace necesario considerar el problema de cómo asignar memoria disponible a varios de los procesos que están en la cola de espera para ser traídos a memoria principal. Lo más inmediato y simple es dividir la memoria en particiones (posiblemente de distinto tamaño), de tal forma que en cada partición se mete un proceso, donde permanece hasta que finaliza su ejecución. Una vez terminado el proceso, la partición queda libre para acoger a un nuevo trabajo. Con la llegada de la multiprogramación y este esquema de memoria aparecen algunos problemas y cuestiones que deben solventarse, como la planificación de procesos a largo plazo.

PLANIFICACIÓN

Un esquema posible para la planificación de procesos a largo plazo, o sea, para seleccionar los procesos que van cargarse en memoria para ser ejecutados, puede consistir en que cada una de las particiones tenga una cola asociada, de tal manera que en cada una de ellas se van encolando los trabajos o procesos dependiendo del espacio de memoria requerido. Cuando hay que cargar un trabajo, se le pone en la cola de entrada de la partición más pequeña en la que quepa. Ya que en este esquema las particiones son de tamaño fijo preestablecido, cualquier espacio de una partición no utilizado por el proceso cargado, se desaprovecha.

REUBICACIÓN DE PROGRAMAS

Cuando se monta o enlaza un programa compuesto por diferentes módulos, todos ellos se combinan en un único módulo cargable, en el que las referencias a sus objetos locales (rutinas o datos) son direcciones que van desde cero hasta la correspondiente al tamaño del módulo. Así, si el módulo se carga en la dirección cero de memoria principal, se ejecutará correctamente, pero no si se carga en cualquier otra dirección. Según esto, habría que decirle al montador en qué dirección se va a cargar ese programa.

INTERCAMBIO DE MEMORIA

- Consiste en trasladar el código y los datos de un proceso completo de memoria al sistema de almacenamiento secundario, para cargar otro previamente almacenado, no permiten a un proceso utilizar más memoria RAM de la que realmente existe en el sistema. Esta técnica puede ser ineficiente ya que se tiene que hacer el intercambio completo del proceso, aunque éste solo vaya a ejecutar una pequeña porción del código.
- Durante el intercambio un proceso puede ser sacado temporalmente de memoria y llevado a un lugar especial del disco y posteriormente vuelto a memoria y continuada su ejecución.

ASIGNACIÓN CONTIGUA

La memoria principal normalmente se divide en dos particiones: Sistema operativo residente, normalmente en la parte baja de memoria con los vectores de interrupción. Procesos de usuario en la parte alta.

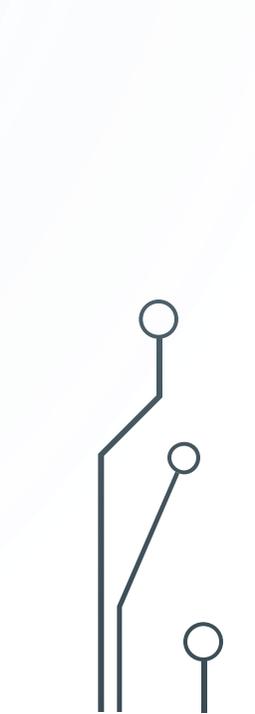
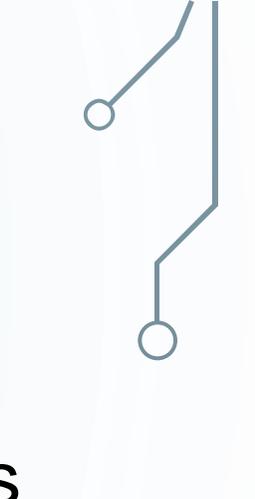
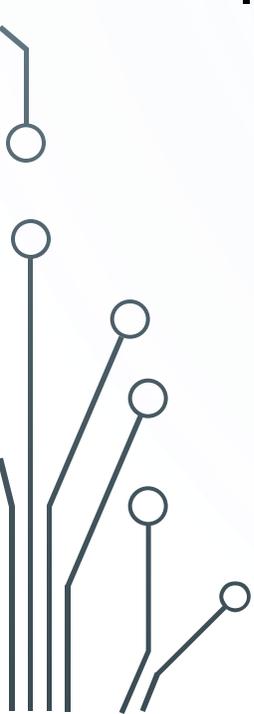
ASIGNACIÓN DE PARTICIÓN SIMPLE

- Puede utilizarse un esquema de registro de relocalización y límite para proteger un proceso de usuario de otro y de cambios del código y datos del sistema operativo.
- El registro de relocalización contiene la dirección física más pequeña; el registro limite contiene el rango de las direcciones lógicas. Cada dirección lógica debe ser menor al registro limite.



GESTIÓN DE BLOQUES DE MEMORIAS

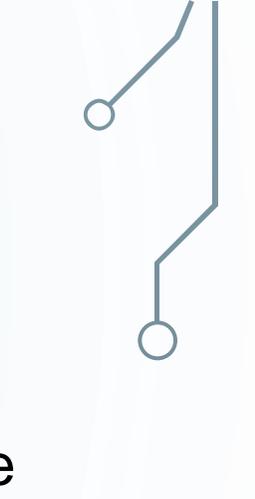
En general, hay tres métodos utilizados por los sistemas operativos para llevar la cuenta de la memoria utilizada y de los huecos libres.

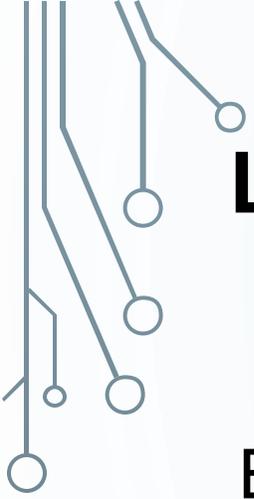




MAPA DE BITS

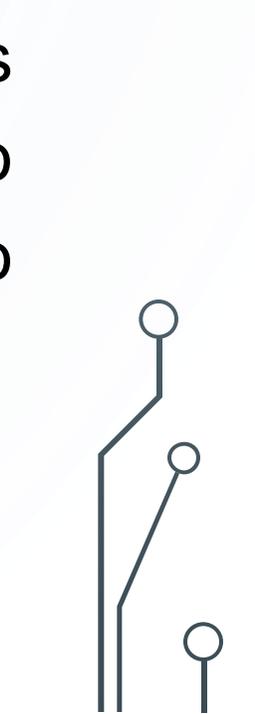
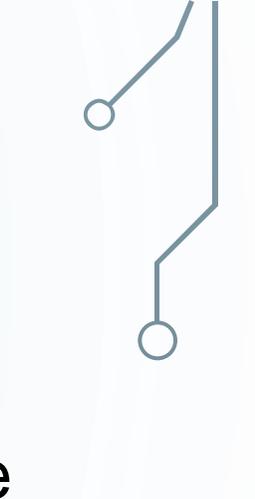
Con este método de gestión de memoria, ésta se divide en cuyo tamaño puede ir desde unas cuantas palabras a varios Kbytes, cada una de estas unidades de asignación se corresponde con un bit en el mapa de bits de memoria; la primera unidad con el primer bit, la segunda con el segundo, etc., de tal forma que si el bit está a 0 indica que la unidad de asignación correspondiente se encuentra libre, mientras que si está a 1 quiere decir que la unidad está ocupada, o sea, asignada a algún proceso. El tamaño de la unidad de asignación es una cuestión muy importante. Cuanta más pequeña sea la unidad, mayor será el mapa de bits.





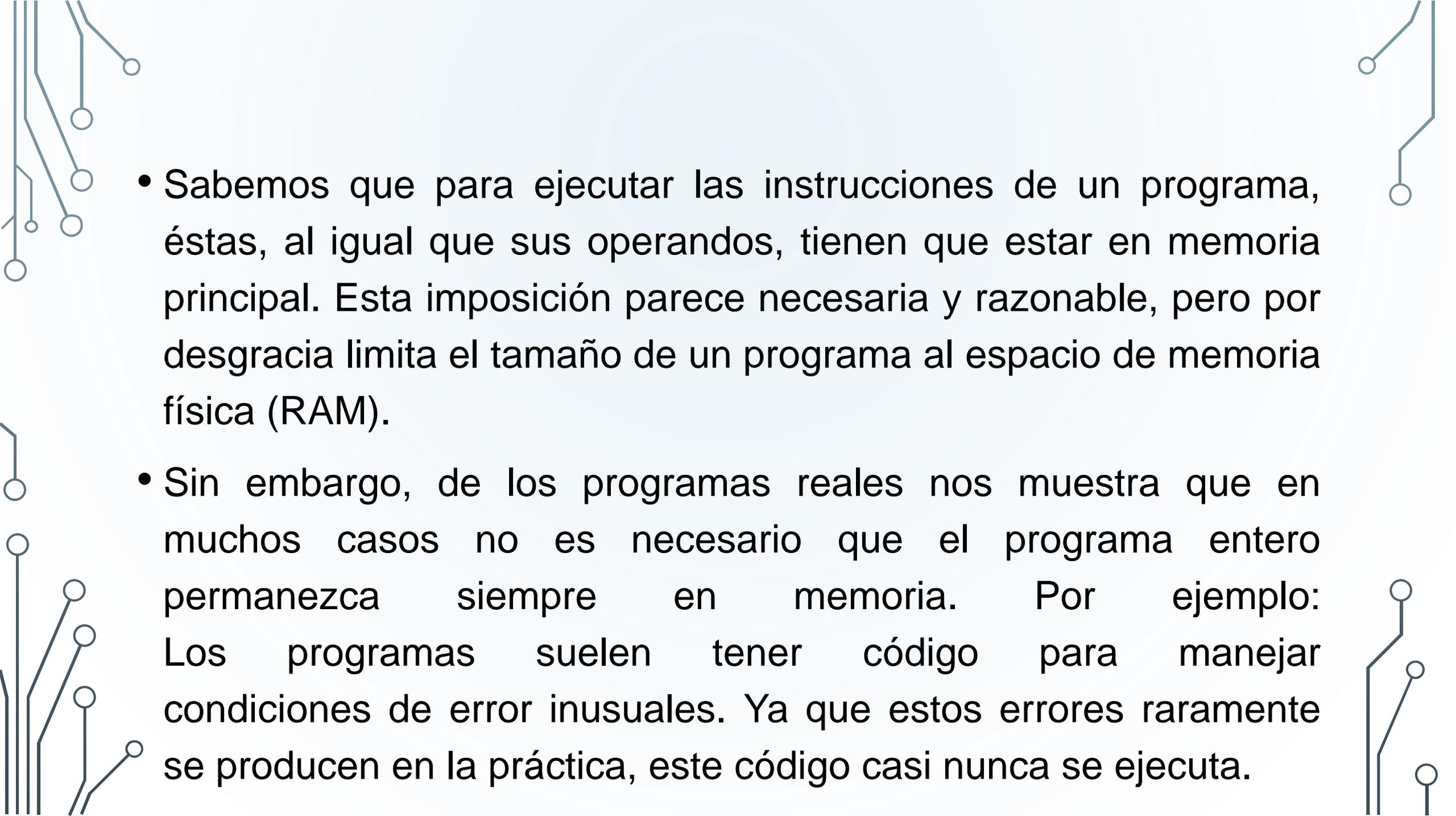
LISTA DE BLOQUES

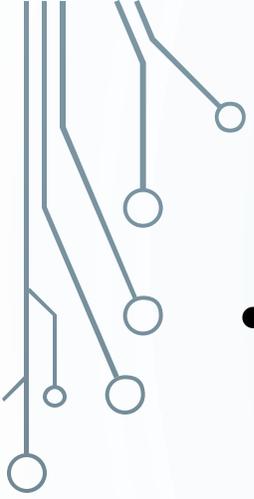
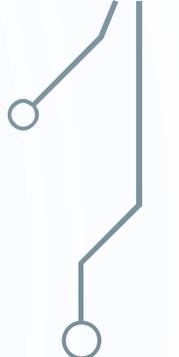
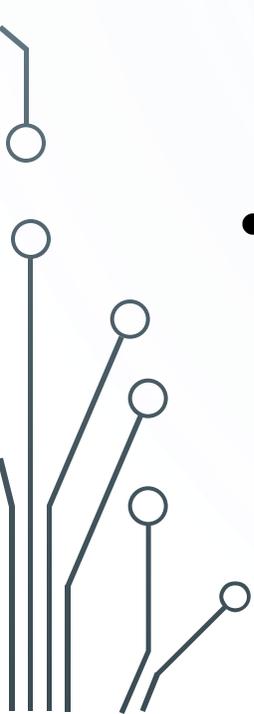
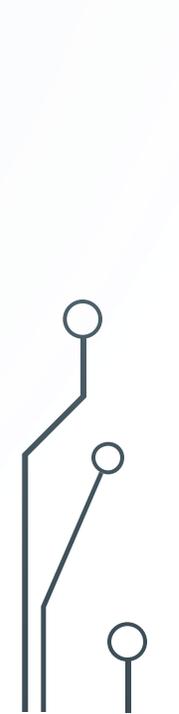
El algoritmo más simple es el primero que sirve. El gestor de memoria recorre la lista hasta encontrar un hueco que sea suficientemente grande para satisfacer la petición. Si el hueco elegido no es exactamente del tamaño solicitado, se divide en dos partes: una (del tamaño solicitado) se le asigna al proceso que lo solicitó, y la otra (con la memoria sobrante) se deja como un hueco de menor tamaño.

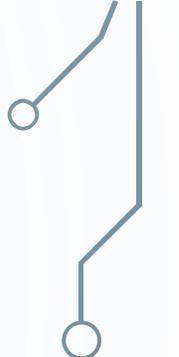


The image features a central graphic of three overlapping circles in shades of light blue and white. The text 'MEMORIA VIRTUAL' is centered within these circles. The corners of the image are decorated with stylized circuit board traces and nodes, rendered in a light gray color.

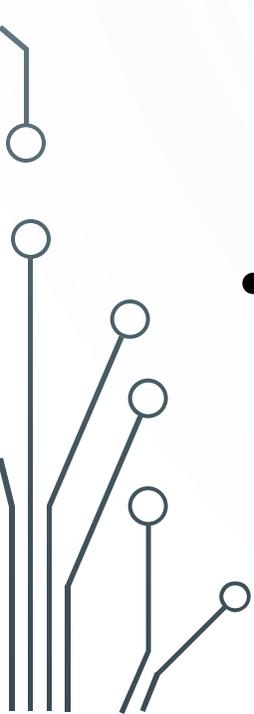
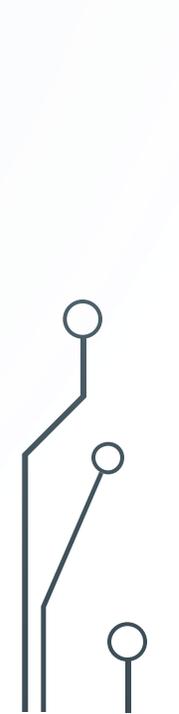
MEMORIA VIRTUAL

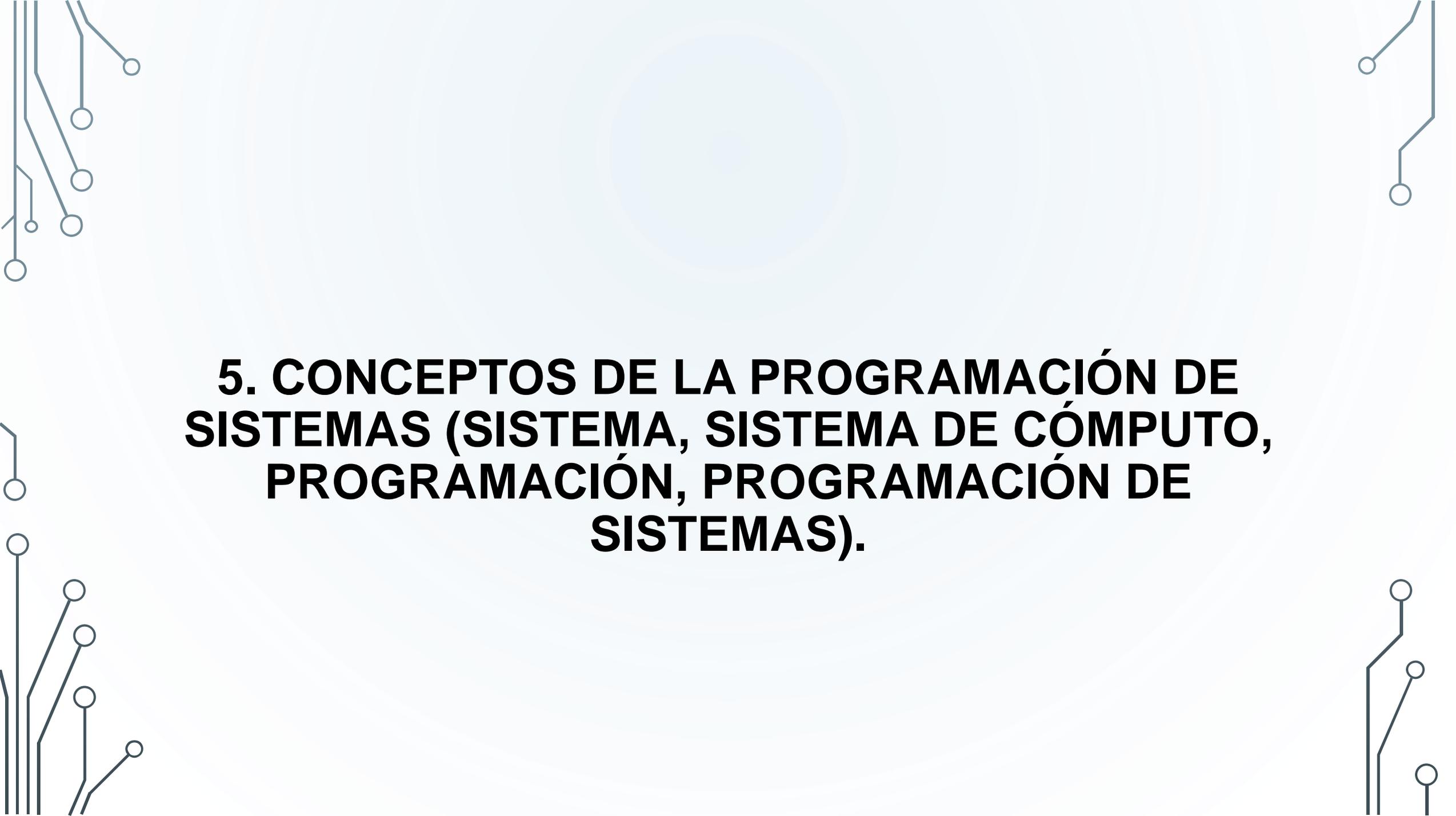
- 
- Sabemos que para ejecutar las instrucciones de un programa, éstas, al igual que sus operandos, tienen que estar en memoria principal. Esta imposición parece necesaria y razonable, pero por desgracia limita el tamaño de un programa al espacio de memoria física (RAM).
 - Sin embargo, de los programas reales nos muestra que en muchos casos no es necesario que el programa entero permanezca siempre en memoria. Por ejemplo: Los programas suelen tener código para manejar condiciones de error inusuales. Ya que estos errores raramente se producen en la práctica, este código casi nunca se ejecuta.

- 
- 
- 
- 
- A veces, las tablas, listas o matrices se declaran con más tamaño del que luego realmente necesitan. Una matriz se puede declarar de 100 por 100 aunque raramente ocupe más de 10 por 10; o un ensamblador puede tener una tabla para albergar a 3000 símbolos, aunque la media de los programas no tengan más de 200.
 - La memoria virtual es una técnica que permite la ejecución de procesos que pueden no estar completamente en memoria principal.



La habilidad de poder ejecutar un programa que sólo está parcialmente en memoria principal acarrea los siguientes beneficios:

- El tamaño de un programa no está limitado por la cantidad de memoria física disponible. Los usuarios escriben programas contando con un espacio de direcciones virtuales extremadamente grande.
 - Debido a que cada programa puede necesitar para ejecutarse menos memoria que la que ocupa su tamaño total, se pueden cargar más programas en memoria para ejecutarlos al mismo tiempo, con la consiguiente mejora en el aprovechamiento de la CPU.
- 
- 

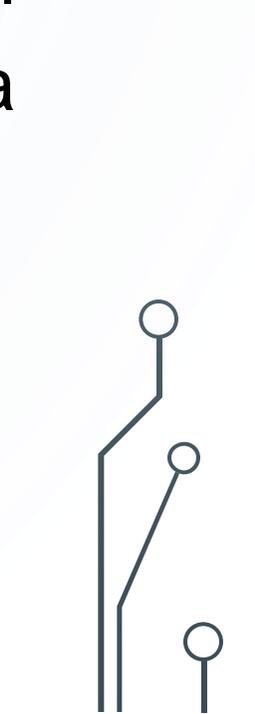
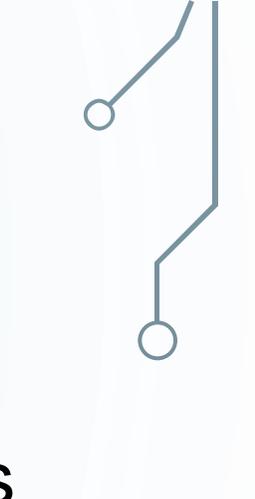
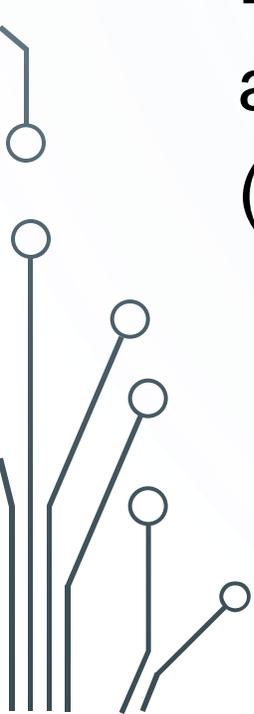
The image features a light blue background with a large, faint watermark of a computer monitor in the center. The corners are decorated with stylized circuit board traces and nodes. The main text is centered and reads:

5. CONCEPTOS DE LA PROGRAMACIÓN DE SISTEMAS (SISTEMA, SISTEMA DE CÓMPUTO, PROGRAMACIÓN, PROGRAMACIÓN DE SISTEMAS).



SISTEMA

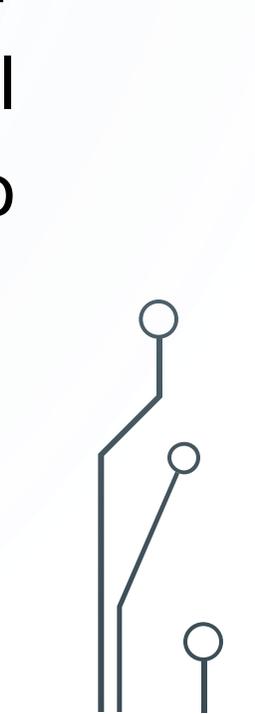
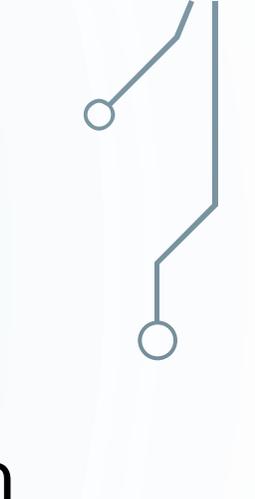
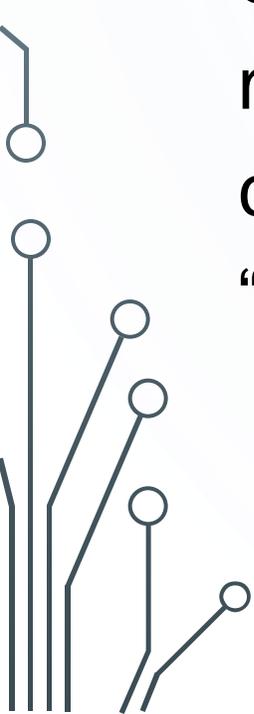
Un sistema es un conjunto de partes o elementos organizadas y relacionadas que interactúan entre sí para lograr un objetivo. Los sistemas reciben (entrada) datos, energía o materia del ambiente y proveen (salida) información, energía o materia (Alegsa, 2018).





SISTEMA DE CÓMPUTO

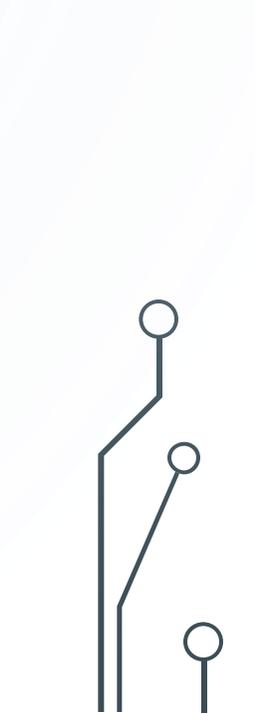
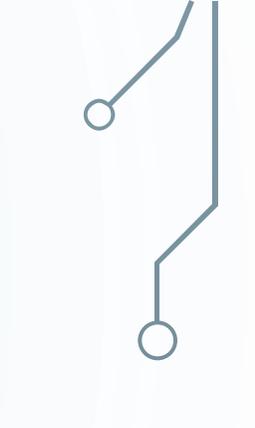
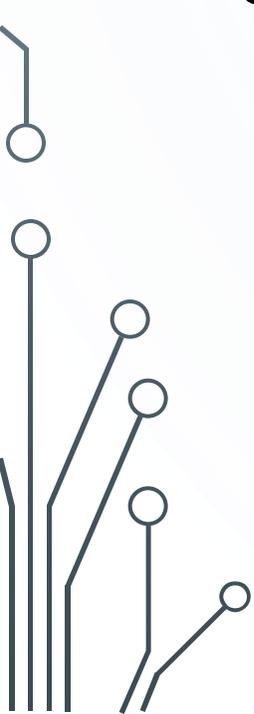
En general podemos definir un sistema de cómputo como un sistema complejo que está constituido por millones de componentes electrónicos elementales que funcionan por medio de un programa específico para realizar una tarea. El conjunto de componentes físicos y electrónicos es llamado “Hardware” y a programas se les da el nombre de “Software”.





PROGRAMACIÓN

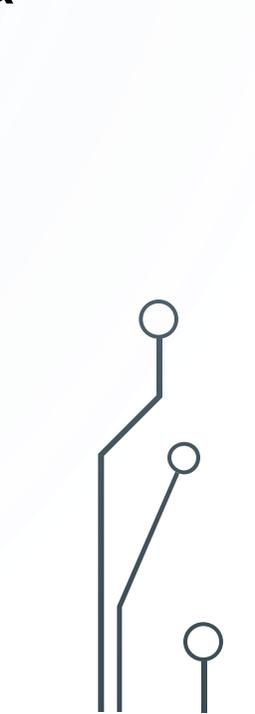
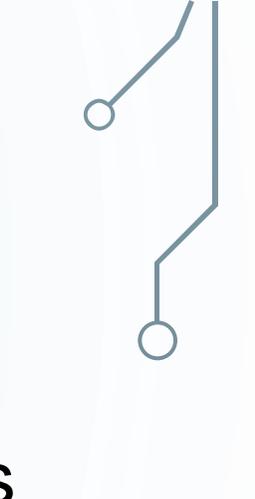
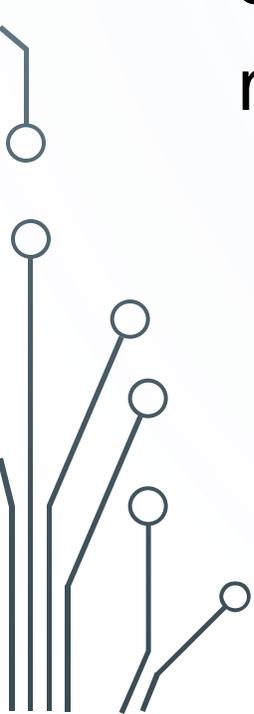
La programación, es el proceso de diseñar, codificar, depurar y mantener el código fuente de programas de computadora. El código fuente es escrito en un lenguaje de programación.





PROGRAMACIÓN DE SISTEMAS

Conjunto de reglas para crear soluciones a problemas computables. Conjunto de herramientas que nos permite crear software de base que es de utilidad para interactuar con la máquina.



The image features a light blue background with a large, faint circular graphic in the center. The corners are decorated with stylized circuit board traces and nodes. The main text is centered and reads:

6. EVOLUCIÓN DE LOS LENGUAJES DE PROGRAMACIÓN.

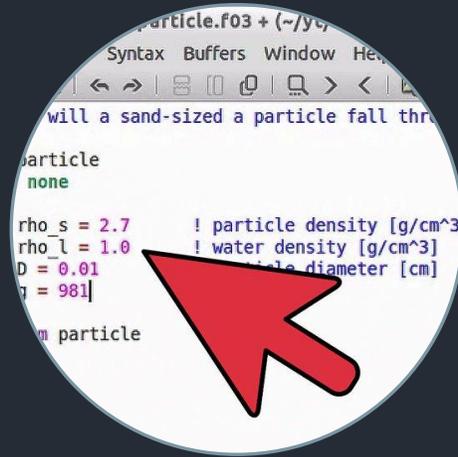


1950

Lenguaje Ensamblador

Mauricio V. Vilkes inventa el lenguaje ensamblador.

Anteriormente la programación se efectuaba directamente en binario. Cada modelo de ordenador tenía su propio código lo que dificultaba su manejo.



1957

Lenguaje Fortran

Jhon Backus de IBM inventa Fortran, el primer lenguaje de programación universal considerado de alto nivel, de propósito general e imperativo.



1958

Lenguaje Lisp

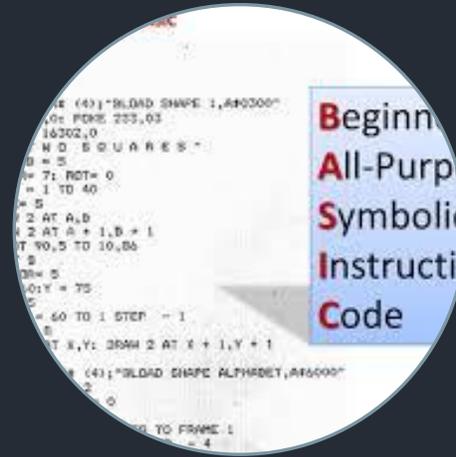
Fue el segundo lenguaje de programación de alto nivel. Creado por Jhon McCarthy en el MIT. El elemento fundamental de Lisp es la lista, pues tanto un dato o una función se expresa como una lista. Fue desarrollado inicialmente sobre un IBM 7090.





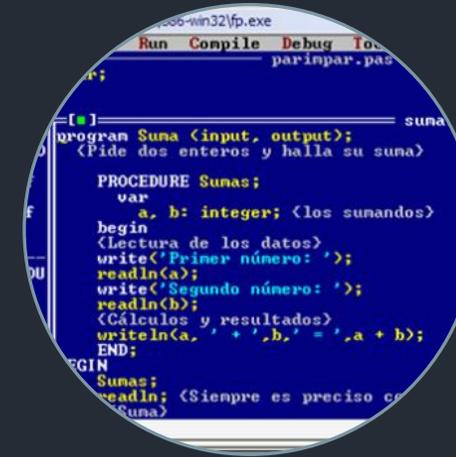
1959 Lenguaje Cobol

Lenguaje de programación que se inventó con el objetivo de utilizarse en cualquier computadora. Se caracterizó por tener una excelente capacidad de autodocumentación, buena gestión de archivos y de datos de la época.



1964 Lenguaje Basic

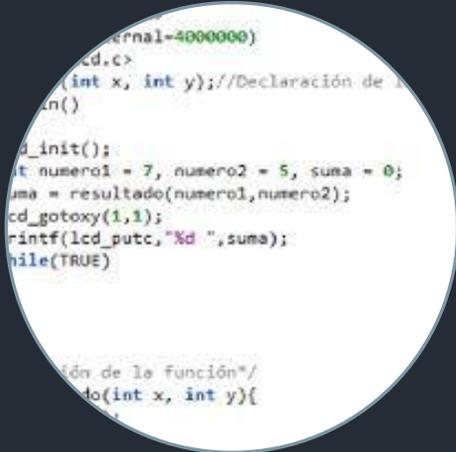
Lenguaje de programación desarrollado por Jhon Kemeny y Thomas Kurts en Estados Unidos. Inicialmente se desarrolló para facilitar a los estudiantes la programación de computadoras. Su uso era para propósito general, aunque un tanto lento y simple.



1970 Lenguaje Pascal

Pascal es un lenguaje de programación de alto nivel, creado por Nicklaus Wirth. Se convirtió en uno de los lenguajes más utilizados en cursos de introducción a la programación. Se desarrolló para hacer posible la programación estructurada y también soporta la programación orientada a objetos.

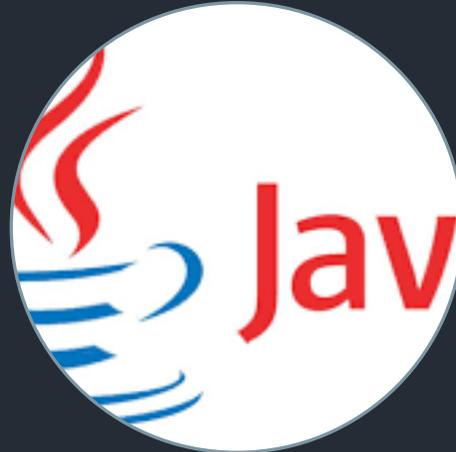




1972

Lenguaje de Programación C

Dennis Ritchie de los Laboratorios Bell, reanuda el lenguaje B escrito por Ken Thompson y lo convierte en un verdadero compilador que genera el código máquina (B era un intérprete). C es utilizado por la eficiencia de su código para generar aplicaciones y software de sistemas.



1990

Lenguaje de Programación Java

Lenguaje de programación desarrollado por Sun Microsystems. Se desarrolló de tal forma que los programas desarrollados con Java puedan ejecutarse de la misma manera en diferentes arquitecturas de computadora. Java permite escribir programas de gráficos o textuales.



1991

Lenguaje de Programación Python

Desarrollado por Guido van Rossum, es un lenguaje de programación de alto nivel, posee una sintaxis amplia así como favorece el código legible. Soporta programación imperativa, orientada a objetos, multiplataforma, manejo de excepciones y es un lenguaje interpretado.





1991

Lenguaje HTML

Gracias a Tim Berners-Lee (que trabajaba en el CERN), que ante la necesidad de compartir información entre científicos creó la primera definición del lenguaje.



1994

Lenguaje PHP

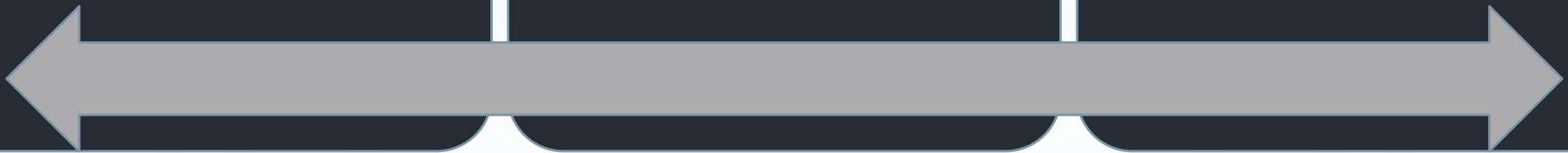
Acrónimo de "HyperText Processor", es un lenguaje de programación de script desarrollado por Rasmus Lerdof. Se utiliza para la programación de páginas dinámicas en servidores y el desarrollo de aplicaciones en diferentes sistemas operativos.

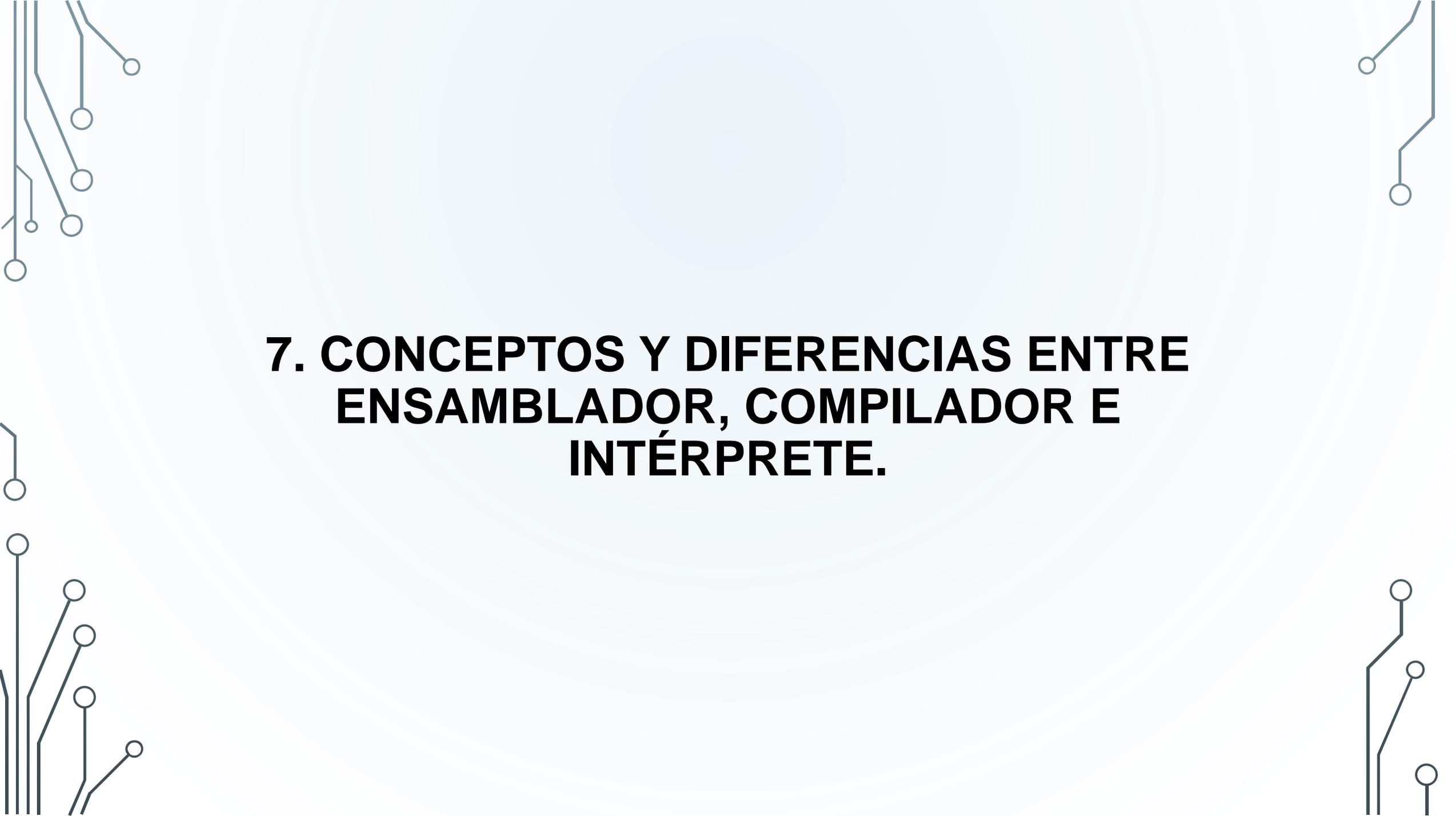


1999

Lenguaje C#

Anders Hejlsberg formó un equipo con la misión de desarrollar un nuevo lenguaje de programación llamado Cool (Lenguaje C orientado a objetos). Este nombre tuvo que ser cambiado debido a problemas de marca, pasando a llamarse C#.



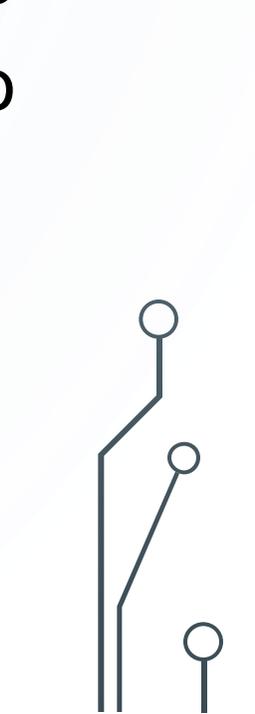
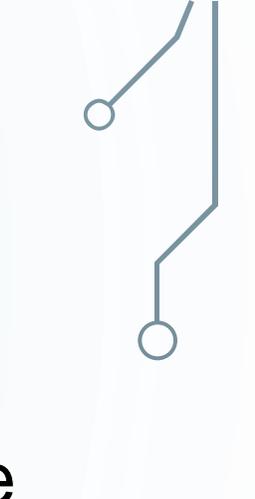
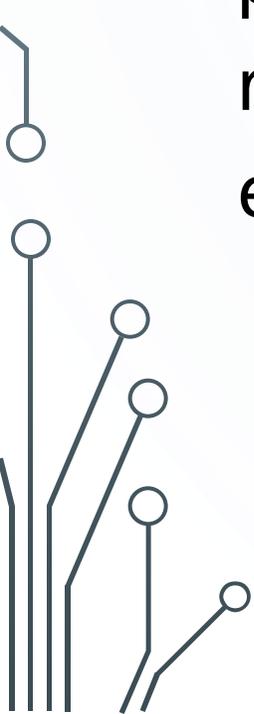
The image features a light blue background with a large, faint circular graphic in the center. The corners are decorated with stylized circuit board traces and nodes. The main text is centered and reads:

7. CONCEPTOS Y DIFERENCIAS ENTRE ENSAMBLADOR, COMPILADOR E INTÉRPRETE.

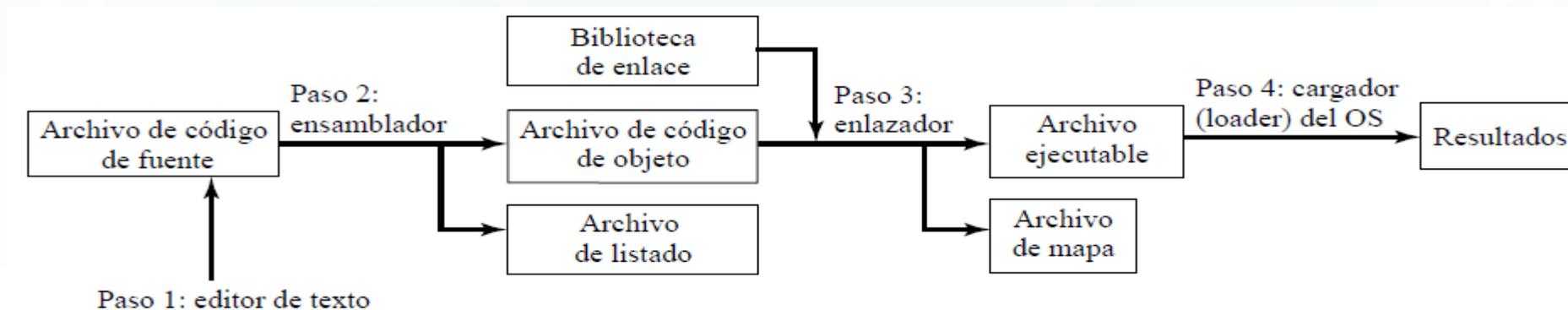


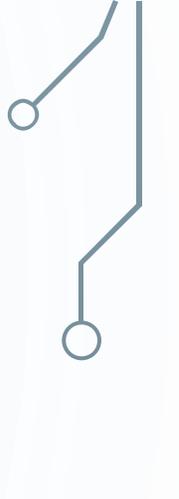
ENSAMBLADORES

Cuando se empezaron a utilizar símbolos nemotécnicos, se escribieron programas para traducir automáticamente los programas escritos en lenguaje ensamblador a lenguaje máquina. A estos programas traductores se les llamo ensambladores.

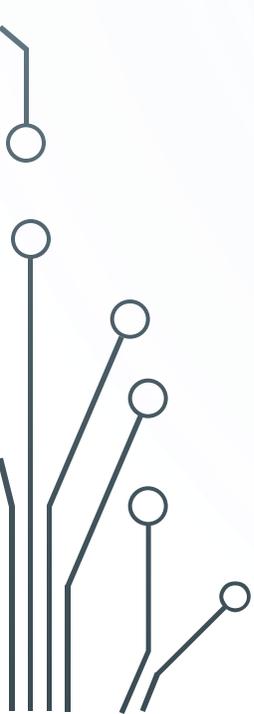
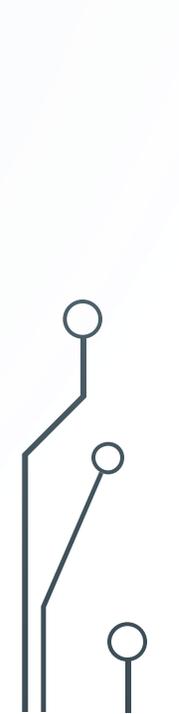


La entrada para un ensamblador es un programa fuente escrito en lenguaje ensamblador. La salida es un programa objeto, escrito en lenguaje de máquina. El programa objeto incluye también la información necesaria para que el cargador pueda preparar el programa objeto para su ejecución. A continuación se muestra el ciclo de ensamblado-enlazado-ejecución de un programa en lenguaje ensamblador.





MOTIVOS PARA UTILIZARLO

- Rapidez.
 - Mayor control de la computadora.
 - Independencia del lenguaje.
 - La mayoría de las computadoras pueden ensamblarlo.
- 
- 

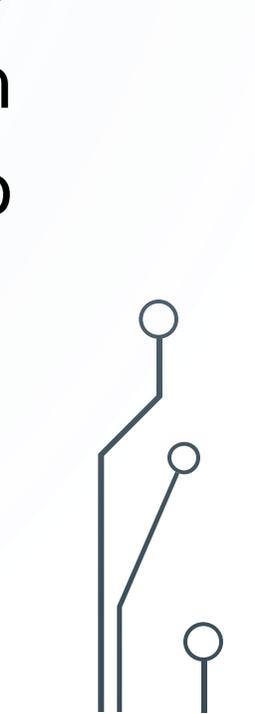
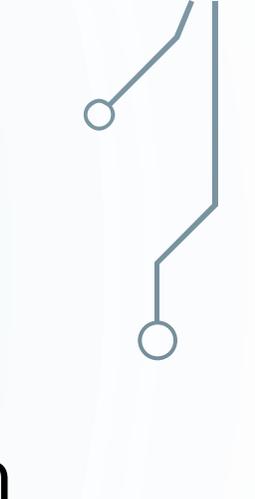
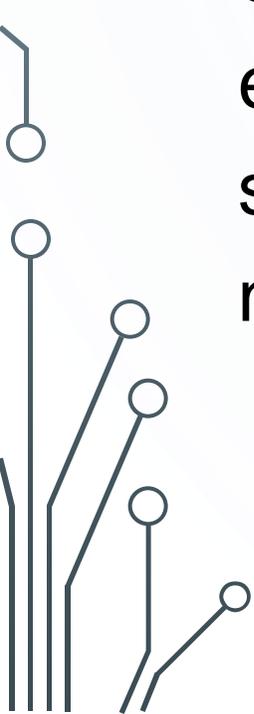
MOTIVOS PARA NO UTILIZARLO

- Dependencia de hardware.
- Mayor tiempo de codificación.
- Comprensión más profunda de la computadora.
- Errores más frecuentes en el programa.



COMPILADORES

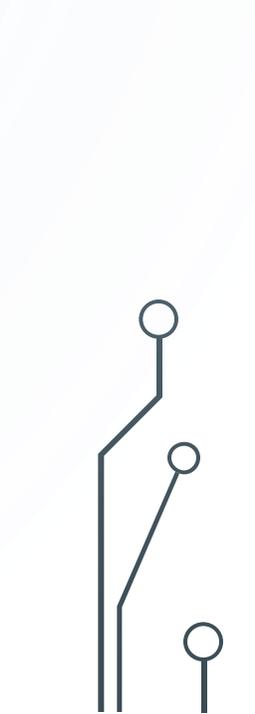
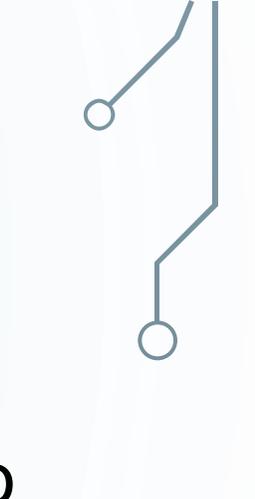
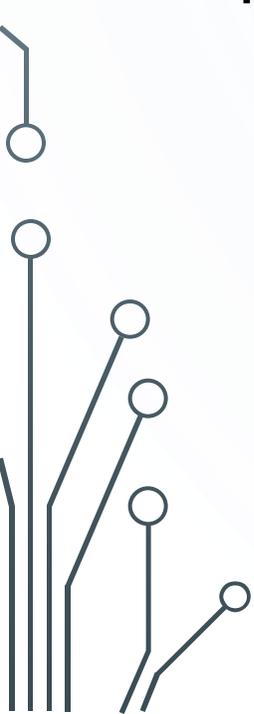
Un compilador es un programa que lee las líneas escritas en un lenguaje de programación y las traduce a otro que pueda ejecutar la computadora. Los programas compilados se ejecutan más rápido que los interpretados, debido a que han sido completamente traducidos a lenguaje de máquina y no necesitan compartir memoria con el intérprete.





ESTRUCTURA DE UN COMPILADOR

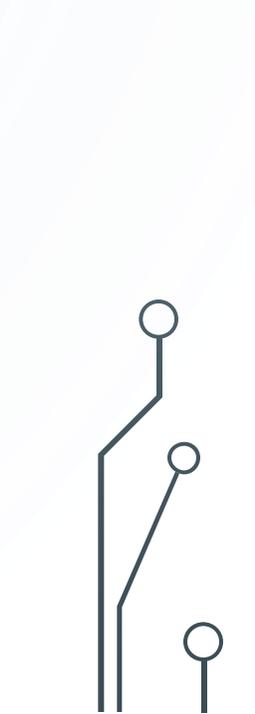
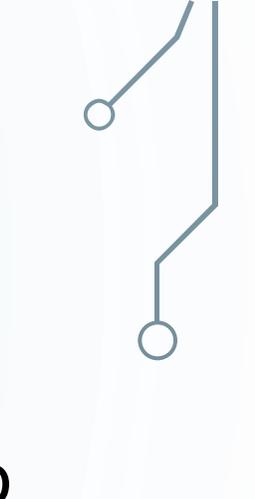
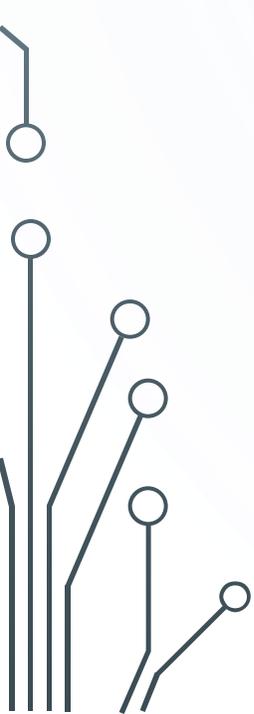
Esta dividida en cuatro grandes módulos, cada uno independiente del otro, se podría decir que un compilador esta formado por cuatros módulos más a su vez.





PRIMER MÓDULO

El preprocesador, es el encargado de transformar el código fuente de entrada original en el código fuente puro.



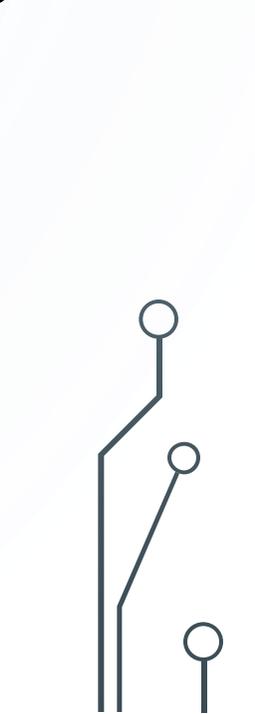
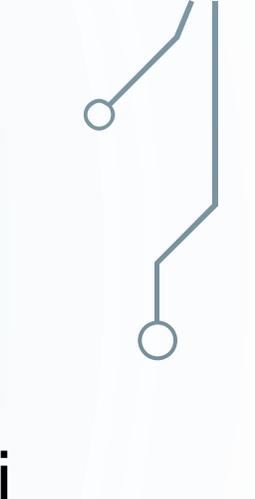
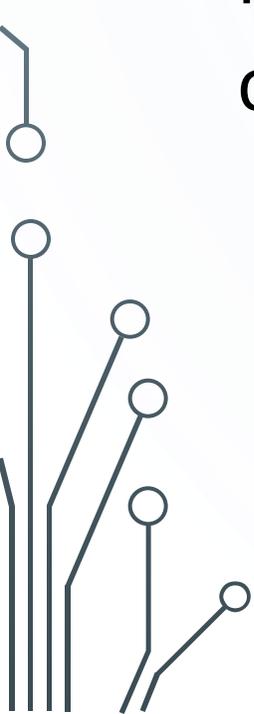
SEGUNDO MÓDULO

El de compilación que recibe el código fuente puro, este es el módulo principal de un compilador, pues si ocurriera algún error en esta etapa el compilador no podría avanzar. En esta etapa se somete el código fuente puro de entrada a un análisis léxico, sintáctico y semántico, que construyen la tabla de símbolos, se genera un código intermedio al cual se optimiza para así poder producir un código de salida generalmente en algún lenguaje ensamblador.



TERCER MÓDULO

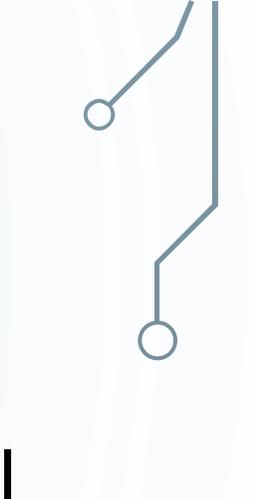
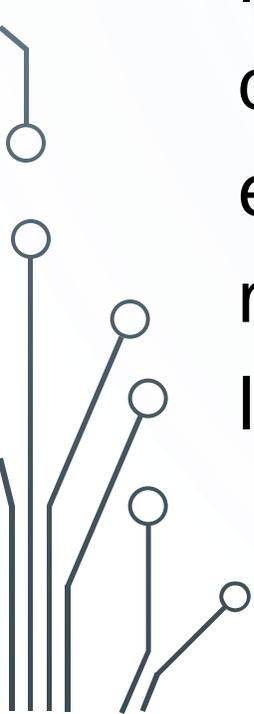
Es el llamado módulo de ensamblado, este módulo no es ni más ni menos que otro compilador que recibe un código fuente de entrada escrito en ensamblador, y produce otro código de salida, llamado código binario no enlazado.

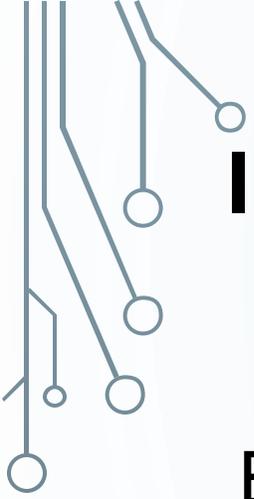




CUARTO MÓDULO

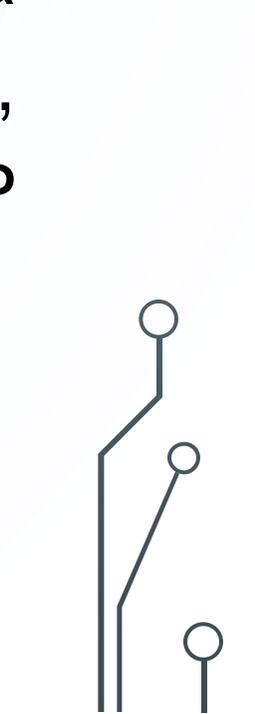
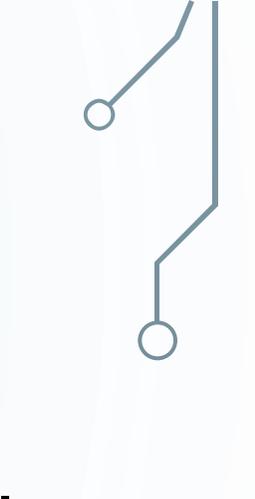
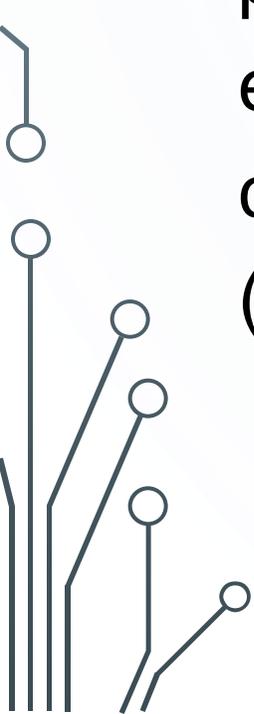
El cuarto y último módulo es el encargado de realizar el enlazado del código fuente de entrada (código máquina re-localizable) con las librerías que necesita, así como también de proveer al código de las rutinas necesarias para poder ejecutarse y cargarse a la hora de llamarlo para su ejecución, modifica las direcciones re-localizables y ubica los datos en las posiciones apropiadas de la memoria.

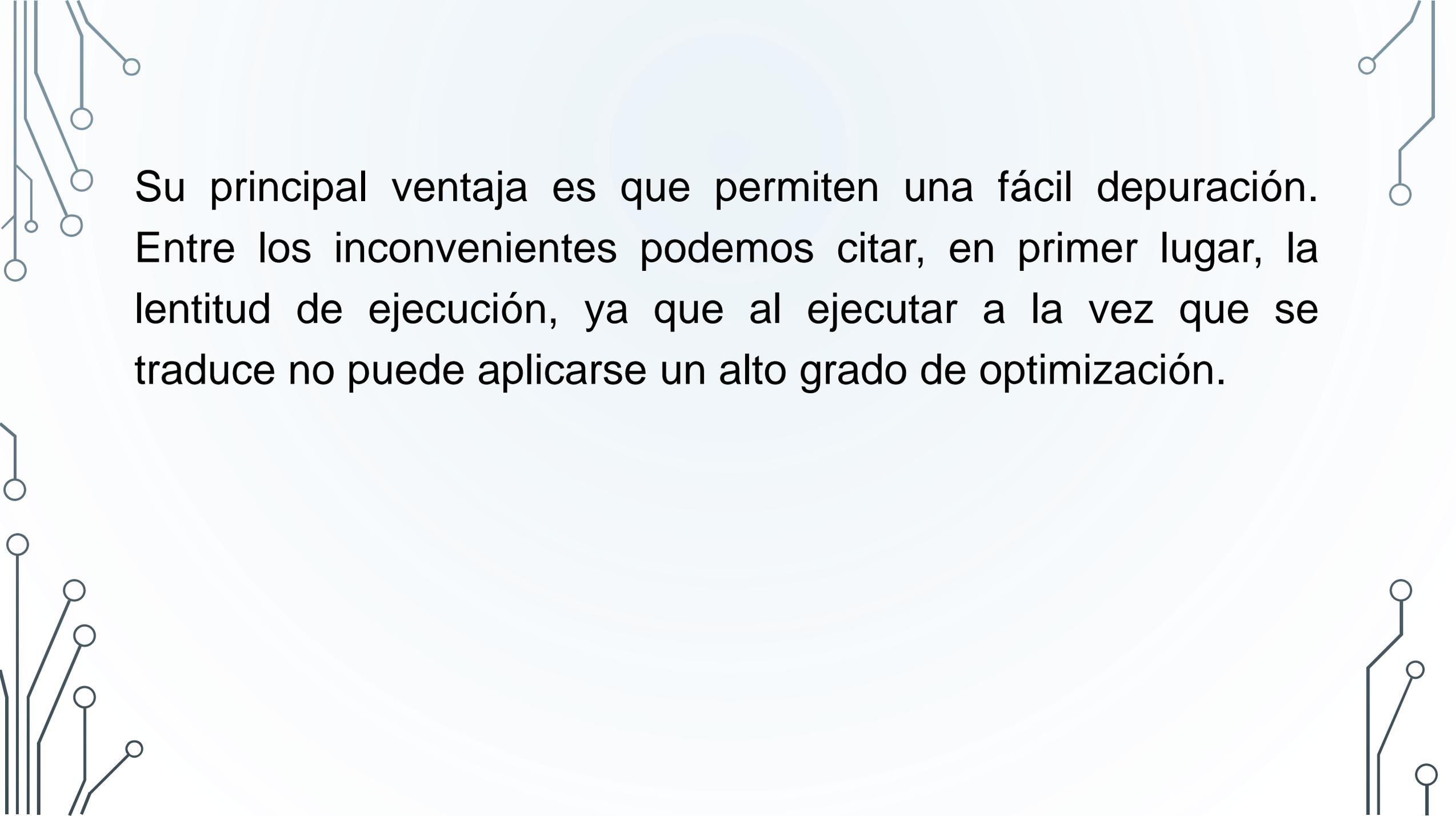




INTERPRETES

Es como un compilador, solo que la salida es una ejecución. El programa de entrada se reconoce y ejecuta a la vez. No se produce un resultado físico (código máquina), sino lógico (una ejecución). Hay lenguajes que sólo pueden ser interpretados, como: SNOBOL (StriNg Oriented SimBOlyc Language), LISP (LISt Processing), etc.

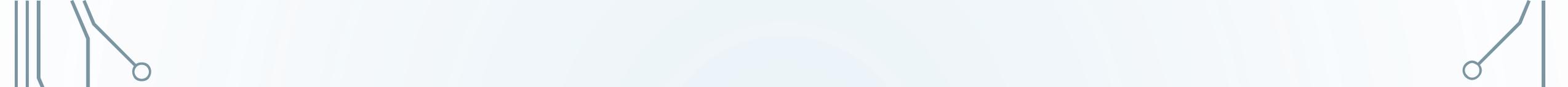


The image features a light blue background with decorative circuit-like lines in the corners. These lines are composed of straight segments and small circles, resembling a stylized PCB or network diagram. They are located in the top-left, top-right, bottom-left, and bottom-right corners.

Su principal ventaja es que permiten una fácil depuración. Entre los inconvenientes podemos citar, en primer lugar, la lentitud de ejecución, ya que al ejecutar a la vez que se traduce no puede aplicarse un alto grado de optimización.

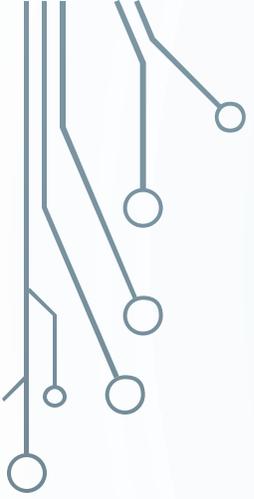
DIFERENCIAS

- Un intérprete traduce instrucciones de alto nivel en una forma intermedia para ser ejecutado. En contraste, un compilador, traduce instrucciones de alto nivel directamente en lenguaje de máquina.
- El intérprete traduce un programa línea a línea mientras que el compilador traduce el programa entero y luego lo ejecuta.

- 
- El intérprete detecta si el programa tiene errores y permite su depuración durante el proceso de ejecución, mientras que el compilador espera hasta terminar la compilación de todo el programa para generar un informe de errores.
 - Un programa compilado es más seguro que uno interpretado, porque no contiene el código fuente, que puede ser modificado incorrectamente por el usuario.
 - Ambos, intérpretes y compiladores están disponibles en la mayoría de los lenguajes de alto nivel.
 - El ensamblador solo es el traductor de alto nivel a lenguaje máquina.

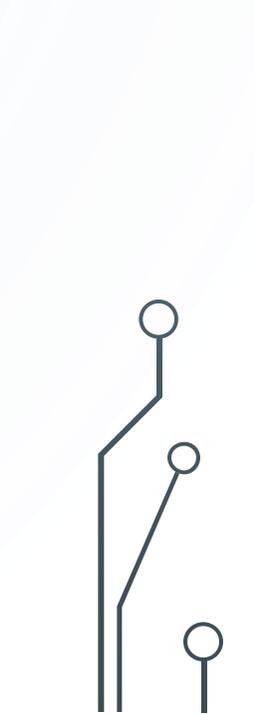
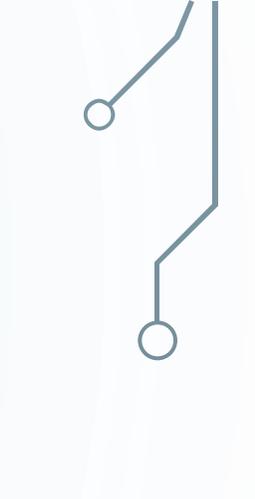
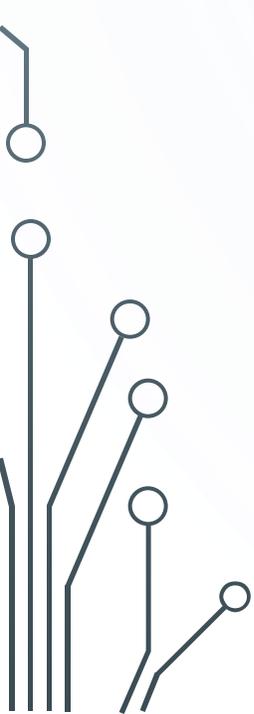
The image features a light blue background with a large, faint circular graphic in the center. The corners are decorated with stylized circuit board traces and nodes. The main text is centered and reads:

8. CONCEPTOS DE LIGADOR Y CARGADOR.



LIGADOR

Es un programa que enlaza distintos módulos o programas que poseen subprogramas. Además incorporan las denominadas rutinas de librerías en caso de solicitarlas el propio programa.



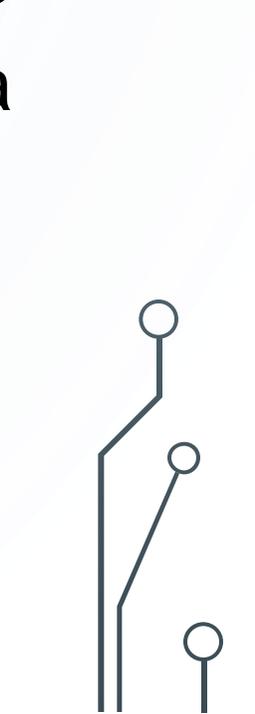
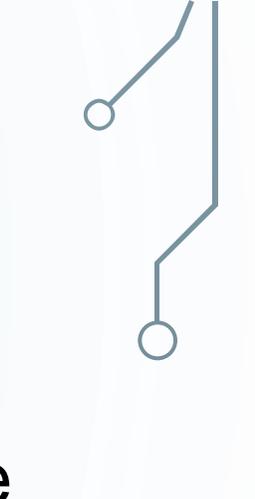
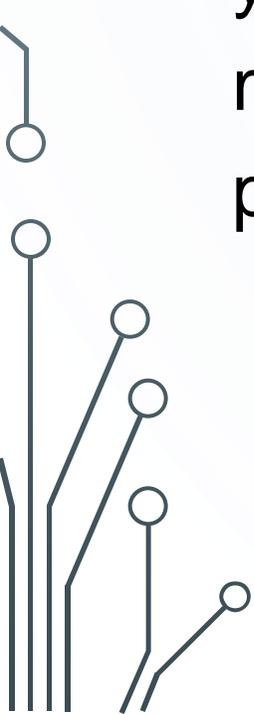
FUNCIONES

- Enlazar código intermedio compilado independientemente en un solo módulo de carga resolviendo las diferencias entre Tokens.
- Incorpora las denominadas rutinas de librerías en caso de solicitarlas el propio programa.
- Su función es reducir procedimientos traducidos por separado y enlazarlos para que se ejecuten como una unidad llamada programa binario ejecutable.



CARGADOR

Es un programa especial, parte del sistema operativo que tiene como propósito colocar en la memoria las instrucciones y datos de un programa o información codificada en lenguaje máquina, para que entonces la computadora pueda procesarla.



FUNCIONES

- Colocar un programa objeto en la memoria e iniciar su ejecución.
- El cargador realiza la última etapa del proceso de traducción: cargar el programa en memoria donde puede ser ejecutado.
- Las funciones de un cargador son relativamente sencillas y consisten en extraer información de algún medio exterior de la memoria (por ejemplo: CD o DVD) y colocarlo en celdas sucesivas de la memoria a partir de una celda pre-especificada.

BIBLIOGRAFÍA

Abel, P. (2006). *Lenguaje Ensamblador y Programación para PC IBM y Compatibles*. México: Prentice Hall.

Alegsa, L. (2018). *Definición de sistema*. Recuperado el 27 julio de 2019 de <http://www.alegsa.com.ar/Dic/sistema.php>

Brey, B. (2006). *Microprocesadores Intel. Arquitectura, Programación e Interfaz*. México: Prentice Hall.

Irvine, K. (2008). *Lenguaje ensamblador para computadoras basadas en Intel*. México: Prentice Hall.

Mostacero, R. (2012). *Microprocesadores*. Recuperado el 19 de agosto de 2019 de <http://tododemicroprocesadores.blogspot.mx/2012/08/gestion-de-memoria-segmentacion-y.html>

Pérez, A. (2010). *Arquitectura x86 y x64*. Recuperado el 10 de agosto de 2019 de <https://electrouni.files.wordpress.com/2010/12/arquitectura-x86-y-x64.pdf>

Timetoast timelines. (2019). *Línea del tiempo: Evolución de los lenguajes de programación*. Recuperado el 10 de agosto de 2019 de <https://www.timetoast.com/timelines/linea-del-tiempo-evolucion-de-los-lenguajes-de-programacion>

Xikotenkaltb. (2011). *Cargadores y Ligadores*. Recuperado el 5 de agosto de 2019 de <http://xikotenkaltb.blogspot.mx/2011/04/41-cargadores-y-ligadores.html>