

FACULTAD DE INFORMATICA (UNLP)



ORGANIZACIÓN DE COMPUTADORAS

RESUMEN DE LA MATERIA - 2014

COD: SI-104

NICOLAS CERESOLE

UNIDAD 01 – COMPUTADORAS DIGITALES

• 1.1 ¿Qué es una computadora?

Una computadora es una maquina digital y sincrónica con cierta capacidad de cálculo y comunicación con el mundo exterior. Es **digital** porque se maneja con señales eléctricas y la información se representa por medio de dos valores (0 y 1). Es **sincrónica** porque todas las operaciones internas se realizan en instantes de tiempos definidos y controlados por un reloj central. Posee **capacidad de cálculo** numérico y lógico mediante la ALU (Unidad Aritmético Lógica), que es la encargada de realizar las operaciones simples como suma, resta, and, or. Tiene **comunicación con el mundo exterior** mediante el módulo de E/S que permite conectar dispositivos periféricos, con los que puede realizar operaciones de entrada y salida de datos. Todas estas funciones están controladas por la Unidad de Control.

Las funciones básicas de una computadora son: Procesamiento, almacenamiento y transferencia de datos y su estructura básica cuenta de 4 componentes principales:

- Unidad Central de Procesamiento (CPU).
- Memoria Principal (MP).
- Módulo de Entrada/Salida.
- Sistema de Interconexión.

• 1.2 Clasificación de las computadoras.

Se distinguen tres tipos de clasificaciones principales: **según su finalidad** (de propósito específico o de propósito general), **según su funcionamiento** (computadora digital, computadora analógica o computadora híbrida) y **según su clasificación comercial** (supercomputadoras, mainframes, minicomputadoras y microcomputadoras).

♦ Según su finalidad:

- **Propósito específico:** son construidas para realizar funciones concretas, como por ejemplo, el guiado de un cohete, el control de un robot, una caja registradora.
- **Propósito general:** son construidas pensando en un rango de aplicación mucho más amplio, como por ejemplo, la gestión de un almacén, el procesamiento de textos o bases de datos.

♦ Según su funcionamiento:

- **Computadora digital:** procesa datos cuya representación responde a valores como 0 y 1, operando con ellos en distintas y diversas etapas sucesivas.
- **Computadora analógica:** tienen semejanza con instrumentos de medida como amperímetros, voltímetros.
- **Computadora híbrida:** posee características de los dos tipos anteriores. Generalmente, los cálculos se realizan de forma analógica y la entrada y salida de datos se hace de forma digital.

♦ Según su clasificación comercial:

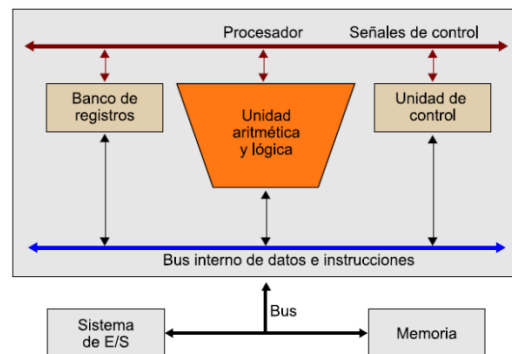
- **Supercomputadoras:** son las computadoras más potentes, complejas, grandes y costosas, utilizadas por científicos y técnicos para resolver cuestiones de alta complejidad.
- **Mainframes:** son computadoras con alta velocidad de proceso y capacidad de memoria, utilizadas a diario por grandes empresas para realizar diferentes operaciones.
- **Minicomputadoras:** son computadoras de tamaño medio, con costo y tamaño reducido, pero de igual forma cuentan con una capacidad de proceso elevada.
- **Microcomputadoras:** son las computadoras de menor tamaño, capacidad de proceso y memoria, utilizadas mayoritariamente a nivel doméstico por su relativo bajo costo.

• 1.3 Arquitectura von Neumann.

La mayoría de las computadoras actuales de propósito general, presentan una estructura interna basada en la arquitectura definida por **John von Neumann** en los años 40, mientras colaboraba con el proyecto ENIAC. Esta arquitectura define que una computadora está compuesta por cuatro elementos principales: Una *Unidad Central de Proceso (CPU)*, una *Memoria Principal (MP)*, un *Módulo de Entrada/Salida (E/S)* y un *Sistema de Interconexión* para estos componentes. La característica principal de esta arquitectura, es que la Unidad Central de Proceso está conectada a una única memoria donde se guardan conjuntamente, datos e instrucciones. La memoria es accesible por posición, independientemente si se trata de datos o instrucciones, y la ejecución de las instrucciones se realiza de forma secuencial.

En la arquitectura von Neumann, el tamaño de la unidad de datos o instrucciones está fijado por el ancho del bus que comunica la memoria con la CPU, por lo que si tenemos un procesador con un bus de 8 bits, este tendrá que manejar datos e instrucciones de una o más unidades de 8 bits de longitud. Si se tiene que acceder a una instrucción de mayor tamaño, tendrá que realizar más de un acceso a memoria, lo que implica las siguientes limitaciones:

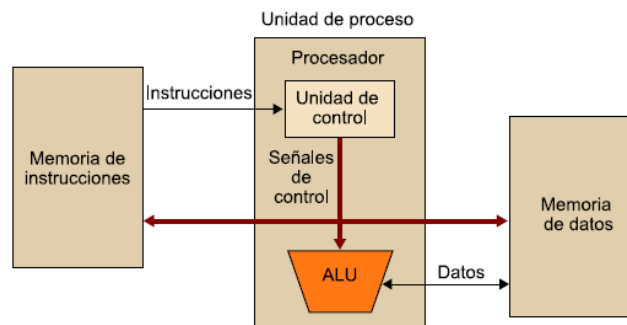
- Se requieren múltiples accesos para manejar instrucciones complejas.
- Al tener un único bus, no se puede buscar una nueva instrucción hasta que no finalice la que está en curso.
- Esto supone una limitación en la velocidad de operación del procesador.



• 1.4 Arquitectura Harvard.

La arquitectura **Harvard** fue diseñada para atacar las limitaciones de la arquitectura von Neumann. Se propuso una solución conceptualmente sencilla: construir un procesador unido a dos tipos diferentes de memoria (una memoria de instrucción y otra memoria de datos) mediante dos buses independientes entre sí (bus de instrucciones y bus de datos). Esto le permite a la computadora leer y ejecutar instrucciones de forma simultánea. Posee las siguientes ventajas respecto a la arquitectura von Neumann:

- El tamaño de las instrucciones no está relacionado con el tamaño de los datos, por lo tanto, puede ser optimizado para que cualquier instrucción ocupe una sola posición de memoria.
- El tiempo de acceso a las instrucciones se puede superponer con el tiempo de acceso a los datos, lo que nos proporciona una mayor velocidad de procesamiento.



● 1.5 Evolución histórica de las computadoras.

◇ GENERACIÓN CERO: COMPUTADORAS MECANICAS (1642 – 1944).

- **Calculadora de Pascal (1642):** construida por Blaise Pascal, podía sumar y restar. La construyo para ayudar a su padre a calcular impuestos.
- **Máquina de Leibniz (1694):** construida por von Leibniz basada en la máquina de Pascal pero que además podía multiplicar y dividir.
- **Z1 de Konrad Zuse (1938):** fue una calculadora mecánica basada en el sistema binario que operaba con electricidad y tenía capacidad de ser programada.
- **Harvard Mark I (1944):** recibía instrucciones y datos desde tarjetas perforadas y era capaz de realizar cinco operaciones (suma, resta, multiplicación, división y referencia a resultados anteriores).

◇ PRIMERA GENERACIÓN: TUBOS DE VACÍO (1946 – 1958).

- Computadoras construidas por **válvulas de vacío**.
- Disipaban gran cantidad de calor y ocupaban una superficie muy amplia.
- Las tareas se ejecutaban en forma secuencial y las operaciones de E/S estaban encadenadas en el tiempo.
- **ENIAC (1946):** construida en la Universidad de Pennsylvania, fue la primera computadora de propósito general, creada para dar respuesta a las necesidades militares de USA.
- **EDVAC (1949):** construida con la intención de resolver muchos de los problemas de la ENIAC. Recibió varias actualizaciones, incluyendo un dispositivo de E/S para tarjetas perforadas, memoria adicional y una ALU.
- **IAS (1952):** creada por von Neumann, era una computadora de programa almacenado. La estructura general de ésta computadora contaba con: Una MEMORIA PRINCIPAL, una ALU, una UC y un dispositivo de E/S.
- **UNIVAC I (1951):** fue la primer computadora fabricada para un propósito no militar, además de ser la primer computadora comercial de USA. Era usada tanto para administración como para negocios.

◇ SEGUNDA GENERACIÓN: TRANSISTORES (1955 – 1965).

- Caracterizada por el descubrimiento del **transistor** que permitió, al ser más pequeño, barato y de menor consumo, la creación de computadoras más accesibles en tamaño y precio.
- Aparecen los **primeros lenguajes de programación** de alto nivel (FORTRAN, ALGOL, COBOL). Los programas eran hechos a medida por un grupo de expertos.

◇ TERCERA GENERACIÓN: CIRCUITOS INTEGRADOS (1963 – 1971).

- Comienza con la invención del **Circuito Integrado de Silicio** (varios componentes en un solo bloque) que posibilitó la inserción de varios transistores en un solo chip.
- Circuitos del tipo **SSI (Small Scale Integration)** y **MSI (Medium Scale Integration)**, que permitieron un gran incremento en la velocidad interna de la computadora y la reducción del consumo energético.

◇ CUARTA GENERACIÓN: INTEGRACIÓN A GRAN ESCALA (1971 – 1990).

- Con la tecnología **LSI (Large Scale Integration)** se logró incluir una CPU completa en una sola pastilla.
- Nacen computadoras extremadamente baratas y pequeñas, logrando insertarlas en el mercado industrial.
- El software obtuvo un considerable avance, volviéndose más interactivo para el usuario. Aparecen los procesadores de texto y las hojas de cálculo.

◇ QUINTA GENERACIÓN: ACTUALIDAD (1990 – ACT).

- Surge la competencia por el dominio internacional del mercado de las computadoras.
- El objetivo principal de esta generación, es el desarrollo de un lenguaje que brinde la posibilidad de comunicarse con las computadoras en un lenguaje cotidiano y no a través de códigos.
- Se desarrollan las **microcomputadoras** y las **supercomputadoras**.
- En los microprocesadores aparecen técnicas para aumentar su velocidad, tales como la **segmentación de instrucciones** y la **paralelización**.

● 1.6 Ley de Moore.

Formulada por el cofundador de Intel **Gordon E. Moore** en 1965, se trata de una ley empírica cuyo cumplimiento se ha podido constatar hasta hoy. Esta ley establece que cada dieciocho meses, el número de transistores por unidad de superficie en circuitos integrados se duplica. La consecuencia directa de esta ley es que los precios bajan al mismo tiempo que las prestaciones aumentan.

Actualmente esta ley se aplica a ordenadores personales. Sin embargo, cuando se formuló no existían los microprocesadores, inventados en 1971, ni los ordenadores personales, popularizados en los años 1980. El propio Moore determinó una fecha de caducidad para su ley: cerca del 2020, no obstante, que una nueva tecnología aparezca.

• 1.7 Análisis de performance.

Desde la perspectiva de la organización y arquitectura de las computadoras, los bloques básicos de las potentes computadoras de hoy en día son prácticamente los mismos que los del computador IAS de hace casi 50 años, mientras que, por otra parte, las técnicas para sacar hasta la última gota del rendimiento de los elementos disponibles se han vuelto cada vez más sofisticadas.

♦ Velocidad del microprocesador.

Mientras que los fabricantes de chips han estado ocupados aprendiendo cómo se fabrican chips de densidad cada vez mayor, los diseñadores del procesador tienen que producir técnicas cada vez más elaboradas para alimentar al monstruo. Entre las técnicas incorporadas a los procesadores de hoy en día están:

- **Predicción de ramificación:** el procesador se anticipa al software y predice qué ramas, o grupos de instrucciones, se van a procesar después con mayor probabilidad. Si el procesador acierta la mayoría de las veces, puede pre captar las instrucciones correctas y almacenarlas para mantener al procesador ocupado.
- **Análisis del flujo de datos:** el procesador analiza qué instrucciones dependen de los resultados de otras instrucciones o datos, para crear una organización optimizada de instrucciones. De hecho, las instrucciones se regulan para ser ejecutadas cuando estén listas, independientemente del orden original del programa. Esto evita retrasos innecesarios.
- **Ejecución especulativa:** utilizando los dos anteriores, algunos procesadores ejecutan especulativamente instrucciones antes de que aparezcan en la ejecución del programa, manteniendo los resultados en posiciones temporales. Esto permite al procesador mantener sus máquinas de ejecución tan ocupadas como sea posible, ejecutando instrucciones que es probable que se necesiten.

♦ Equilibrio de prestaciones.

Mientras que la velocidad del procesador y la capacidad de la memoria han crecido rápidamente, la velocidad con la que los datos pueden ser transferidos entre la memoria principal y el procesador se ha quedado dramáticamente retrasada. La interfaz entre el procesador y la memoria principal es el camino más importante de todo el computador, ya que es el responsable de llevar el constante flujo de instrucciones y datos entre los chips de la memoria y el procesador. Si la memoria o la interfaz no logran mantener el ritmo de las insistentes demandas del procesador, éste se estanca en una posición de espera y se pierde así tiempo de procesamiento valioso.

Conforme los computadores se hacen más rápidos y potentes, se desarrollan aplicaciones más sofisticadas, que se apoyan en el uso de periféricos con demandas intensivas de E/S. La generación actual de procesadores puede manejar los datos producidos por estos dispositivos, pero aún queda el problema de mover esos datos entre el procesador y los periféricos. Las estrategias en relación con esto incluyen esquemas de caches y almacenamiento, más el uso de buses de interconexión de más alta velocidad y con estructuras más elaboradas.

♦ Rendimiento en las computadoras.

Se define rendimiento de un sistema como la capacidad que tiene dicho sistema para realizar un trabajo en un determinado tiempo. Es inversamente proporcional al tiempo, es decir, cuanto mayor sea el tiempo que necesite, menor será el rendimiento.

Los computadores ejecutan las instrucciones que componen los programas, por lo tanto el rendimiento de un computador está relacionado con el tiempo que tarda en ejecutar los programas. De esto se deduce que el tiempo es la medida del rendimiento de un computador. El rendimiento de un procesador queda en función de tres factores:

- **Frecuencia de la CPU,** la cual depende fundamentalmente de la tecnología de fabricación del procesador. Cuanto mayor sea la frecuencia de la CPU, mejor será el rendimiento.
- **Número de instrucciones del programa** el cual depende del programador, del lenguaje de programación y del compilador. Cuanto mayor sea el número de instrucciones del programa peor rendimiento tendrá.
- **CPI** que depende de diseño interno o arquitectura del computador y del software o instrucciones que se hayan elegido. Es importante optimizar el programa con instrucciones que tengan pocos ciclos. Cuanto mayor sea el CPI, peor será el rendimiento.

◇ Unidades de medida de rendimiento global.

En la búsqueda de una medida estándar del rendimiento de los computadores, se han desarrollado una serie de métricas populares como alternativa al uso del tiempo, el cual ha conducido en alguna ocasión a resultados distorsionados o interpretaciones incorrectas.

- **MIPS:** Es el acrónimo de 'Millones de Instrucciones Por Segundo'. Es una forma de medir la potencia de los procesadores. Sin embargo, esta medida sólo es útil para comparar procesadores con el mismo juego de instrucciones y usando benchmarks que fueron compilados por el mismo compilador y con el mismo nivel de optimización. Esto es debido a que la misma tarea puede necesitar un número de instrucciones diferentes si los juegos de instrucciones también lo son; y por motivos similares en las otras dos situaciones descritas.

$$MIPS = \frac{N^{\circ} \text{ Inst. Programa}}{T_{\text{Programa}} * 10^6} = \frac{N^{\circ} \text{ Inst. Programa}}{N^{\circ} \text{ Inst. Programa} * CPI * T_{CPU} * 10^6} = \frac{F_{CPU}}{CPI * 10^6}$$

La ventaja de esta unidad de medida es su fácil comprensión ya que un mayor número de MIPS indicará una mayor velocidad de la máquina. En el mundo de Linux se suelen referir a los MIPS como 'BogoMips'. El equivalente en la aritmética de punto flotante de los MIPS son los flops.

- **FLOPS:** Es el acrónimo de Floating point Operations Per Second (operaciones de punto flotante por segundo). Se usa como una medida del rendimiento de una computadora, especialmente en cálculos científicos que requieren un gran uso de operaciones de coma flotante.

Surgen ya que los MIPS no hacen distinción entre operaciones normales y operaciones en coma flotante. Las computadoras exhiben un amplio rango de rendimientos en punto flotante, por lo que a menudo se usan unidades mayores que el FLOPS.

Los prefijos estándar del Sistema Internacional de Medidas pueden ser usados para este propósito, dando como resultado:

- MegaFLOPS (MFLOPS, 10^6 FLOPS).
- GigaFLOPS (GFLOPS, 10^9 FLOPS).
- TeraFLOPS (TFLOPS, 10^{12} FLOPS).
- PetaFLOPS (PFLOPS, 10^{15} FLOPS).
- ExaFLOPS (EFLOPS, 10^{18} FLOPS).

◇ Benchmarks.

Las medidas de rendimiento vistas hasta ahora no son válidas hoy en día dado que algunas como los MIPS tienden a dar resultados erróneos y a que los computadores actuales tienen una elevada velocidad. La mejor y más fiable forma de calcular el rendimiento es medir el tiempo que los diversos computadores tardan en ejecutar los programas que realmente el usuario va a utilizar posteriormente. ~~Ese será el mejor rendimiento para ese usuario, pero no para todos los usuarios, ya que el rendimiento es un valor relativo de acuerdo con la aplicación que se va a hacer.~~

El rendimiento de una estación de trabajo se mide analizando una serie de componentes físicos que determinan el rendimiento completo del sistema. A la hora de determinar el rendimiento global de un sistema, también hay que evaluar el sistema operativo, los equipos lógicos de red, los compiladores y las librerías gráficas, etc. Para la evaluación del rendimiento de los sistemas se utilizan pruebas de rendimiento o benchmarks, que son programas modelo que efectúa la industria para comparar factores de rendimiento y relaciones rendimiento / precio de los diferentes modelos de computadores.

No obstante, estas evaluaciones no son siempre directamente comparables, y en ocasiones ofrecen poca información, porque las configuraciones con las que se realizan las evaluaciones no son expuestas con claridad. Hay multitud de programas de prueba o benchmarks.

Estos programas se dividen principalmente en 4 grupos, los tres primeros tipos han quedado en desuso:

- *Benchmarks Sintéticos.*
- *Benchmarks Reducidos.*
- *Benchmarks de Núcleo.*
- *Benchmarks Reales.*

UNIDAD 02 – ARITMÉTICA DE COMPUTADORAS

• 2.1 Sistemas de numeración.

Se define a un sistema de numeración como el conjunto de símbolos y reglas que se utilizan para la representación de cantidades o magnitudes. Los podemos dividir en 2 grupos: posicionales y no posicionales.

- **SISTEMA POSICIONAL:** están formados por un juego de n cantidad de símbolos cuya combinación representa valores diferentes. Cada dígito tiene un peso distinto según el lugar que ocupa; el peso es la base elevada a la posición que ocupa dentro del número. La suma de cada dígito multiplicado por su peso permitirá obtener el valor final del número. En todo sistema posicional, dada una base b , se tienen b dígitos disponibles.
- **SISTEMA NO POSICIONAL:** los símbolos no poseen valor propio y las magnitudes se representan por reglas.

♦ Representación generalizada de un número en base b :

$$N \equiv \dots n_2 * b^2 + n_1 * b^1 + n_0 * b^0 + n_{-1} * b^{-1} \dots$$

• 2.2 Teorema Fundamental de la Numeración.

El **Teorema Fundamental de la Numeración** relaciona una cantidad expresada en cualquier sistema de numeración posicional, con la misma cantidad expresa en el sistema decimal. La representación generalizada de un número en una base b , es la siguiente: $N^b \equiv \dots n_2 * b^2 + n_1 * b^1 + n_0 * b^0 + n_{-1} * b^{-1} \dots$

El Teorema Fundamental de la Numeración dice que el valor decimal de una cantidad expresada en otro sistema de numeración, está dado por la fórmula:

$$N^d = \sum_{i=-m}^n (digito)_i * (base)^i$$

- n : número de dígitos de la parte entera.
- m : número de dígitos de la parte fraccionaria.
- i : indica la posición del dígito respecto de la coma.
- $base$: es la base del sistema de numeración (2 para binario, 16 para hexadecimal, etc.).

• 2.3 Sistemas de numeración posicionales.

♦ **SISTEMA DECIMAL:** es un sistema de notación posicional formado por 10 dígitos ($b=10$) $\{0,1,2,3,4,5,6,7,8,9\}$. Por ejemplo, el número $842,97_{10} = 8 * 10^2 + 4 * 10^1 + 2 * 10^0 + 9 * 10^{-1} + 7 * 10^{-2} = 800 + 40 + 2 + 0,9 + 0,07$.

♦ **SISTEMA BINARIO:** es un sistema de notación posicional formado por 2 dígitos ($b=2$) $\{0,1\}$ a los que se denominan bits. El número $1011_2 = 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 * 2^0 = 8 + 0 + 2 + 1 = 11_{10}$.

♦ **SISTEMA HEXADECIMAL:** está formado por 16 dígitos ($b=16$) $\{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\}$ que representan los decimales del 0 al 15. El número $1A2_{16} = 1 * 16^2 + 10 * 16^1 + 2 * 16^0 = 256 + 160 + 2 = 418_{10}$.

♦ **SISTEMA OCTAL:** es un sistema de notación posicional formado por 8 dígitos ($b=8$) $\{0,1,2,3,4,5,6,7\}$. El número $275_8 = 2 * 8^2 + 7 * 8^1 + 5 * 8^0 = 128 + 56 + 5 = 189_{10}$.

Estos cuatro sistemas mencionados son los utilizados comúnmente en la electrónica digital. Para identificar el sistema en el que se trabaja se puede utilizar la notación matemática de la base o se puede añadir al final de número una letra para identificar el sistema (D=10, B=2, H=16, O=8).

• 2.4 Métodos de conversión entre los sistemas numéricos.

• PARA RECORDAR:

$2^{-1} = 0,5$	$2^1 = 2$	$8^{-1} = 0,125$	$8^1 = 8$
$2^{-2} = 0,25$	$2^2 = 4$	$8^{-2} = 0,015625$	$8^2 = 64$
$2^{-3} = 0,125$	$2^3 = 8$	$8^{-3} = 0,0019531$	$8^3 = 512$
$2^{-4} = 0,0625$	$2^4 = 16$	$8^{-4} = 0,000244140$	$8^4 = 4096$
$10^{-1} = 0,1$	$10^1 = 10$	$16^{-1} = 0,0625$	$16^1 = 16$
$10^{-2} = 0,01$	$10^2 = 100$	$16^{-2} = 0,003906$	$16^2 = 256$
$10^{-3} = 0,001$	$10^3 = 1000$	$16^{-3} = 0,00024410$	$16^3 = 4096$
$10^{-4} = 0,0001$	$10^4 = 10000$	$16^{-4} = 0,000015258$	$16^4 = 65536$

♦ **Otras bases a decimal:** se utiliza el *Teorema Fundamental de la Numeración*.

• BINARIO A DECIMAL:

- $1001,101_2 = 1 * 2^3 + 0 * 2^2 + 0 * 2^1 + 1 * 2^0 + 1 * 2^{-1} + 0 * 2^{-2} + 1 * 2^{-3} = 8 + 1 + 0,5 + 0,125 = 9,625_{10}$
- $10111_2 = 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 1 * 2^1 + 1 * 2^0 = 16 + 4 + 2 + 1 = 23_{10}$

• HEXADECIMAL A DECIMAL:

- $1A,0F_{16} = 1 * 16^1 + 10 * 16^0 + 0 * 16^{-1} + 15 * 16^{-2} = 16 + 10 + 0 + 0,058 = 26,058_{10}$
- $4D5_{16} = 4 * 16^2 + 13 * 16^1 + 5 * 16^0 = 1024 + 208 + 5 = 1237_{10}$

• OCTAL A DECIMAL:

- $17,3_8 = 1 * 8^1 + 7 * 8^0 + 3 * 8^{-1} = 8 + 7 + 0,375 = 15,375_{10}$
- $370_8 = 3 * 8^2 + 7 * 8^1 + 0 * 8^0 = 192 + 56 + 0 = 248_{10}$

♦ **Decimal a otras bases:**

- **DECIMAL A BINARIO:** método de *divisiones* (parte entera) y *multiplicaciones* (parte fraccionaria) sucesivas.
- **Método de divisiones sucesivas:** se efectúa sobre el número el método de divisiones sucesivas con respecto a la base (2, en este caso) hasta que el cociente sea 0. La unión de todos los restos, escritos en orden inverso, forman el número convertido.
- **Método de multiplicaciones sucesivas:** se obtiene multiplicando sucesivamente la parte fraccionaria por la base, del resultado se toma de nuevo la parte fraccionaria y se la vuelve a multiplicar hasta que esta sea nula, hasta que se repita una secuencia de dígitos o hasta que se hallen varios dígitos.

- $26,1875_{10}$:

$26/2 = 13$	----> Resto=0	$0,1875 * 2 = 0,3750 \downarrow$
$13/2 = 6$	----> Resto=1	$0,3750 * 2 = 0,7500$
$6/2 = 3$	----> Resto=0	$0,7500 * 2 = 1,5000$
$3/2 = 1$	----> Resto=1	$0,5000 * 2 = 1,0000$
$1/2 = 0$	----> Resto=1 \uparrow	entonces el número será: $11010,0011_2$

- **DECIMAL A HEXADECIMAL:** se utiliza el mismo método anterior, pero la parte entera se divide por 16 y la parte fraccionaria se multiplica por 16.

- $3511,65625_{10}$:

$3511/16 = 219$	----> Resto=7 (7)	$0,65625 * 16 = 10,5 (A) \downarrow$
$219/16 = 13$	----> Resto=11 (B)	$0,5 * 16 = 8,0 (8)$
$13/16 = 0$	----> Resto=13 (D) \uparrow	entonces el número será: $DB7, A8_{16}$

- **DECIMAL A OCTAL:** método anterior pero en base 8.

- $3511,65625_{10}$:

$3511/8 = 438$	----> Resto=7 (7)	$0,65625 * 8 = 5,25 (5) \downarrow$
$438/8 = 54$	----> Resto=6 (6)	$0,25 * 8 = 2,0 (2)$
$54/8 = 6$	----> Resto=6 (6)	entonces el número será: $6667,52_8$
$6/8 = 0$	----> Resto=6 (6) \uparrow	

◇ Conversión entre las demás bases:

- **BINARIO A OCTAL:** se divide el número en grupos de 3 bits, para la parte entera desde el bit menos significativo (LSB) hasta el bit más significativo (MSB) y al revés para la fraccionaria. Luego se sustituye cada grupo por el valor correspondiente en octal.
 - $1011001_2 = 001 | 011 | 001 = 131_8$
 - $1101,1_2 = 001 | 101 | 100 = 15,4_8$
- **OCTAL A BINARIO:** se realiza de forma inversa a la anterior, reemplazando cada dígito octal por su equivalente en binario en grupos de 3 bits.
 - $652_8 = 110 | 101 | 010 = 110101010_2$
 - $24,2_8 = 010 | 100 | 010 = 10100,010_2$
- **BINARIO A HEXADECIMAL:** se divide el número en grupos de 4 bits, empezando por el LSB para la parte entera y por el MSB para la parte fraccionaria.
 - $110101_2 = 0011 | 0101 = 35_{16}$
 - $1111,1_2 = 1111 | 1000 = F,8_{16}$
- **HEXADECIMAL A BINARIO:** de manera inversa a la anterior, reemplazando cada dígito hexadecimal por su equivalente en binario en grupos de 4 bits.
 - $35_{16} = 0011 | 0101 = 110101_2$
 - $F,8_{16} = 1111 | 1000 = 1111,1_2$

● 2.5 Representación de números enteros (Punto Fijo).

Las computadoras utilizan cuatro métodos para la representación de números enteros:

- *Modulo y Signo.*
- *Complemento a la base - 1.*
- *Complemento a la base.*
- *Exceso a 2^{n-1} .*
- **MODULO Y SIGNO:** también llamado Binario Con Signo (BCS), utiliza el bit más significativo (MSB) para representar el signo (0: positivo, 1: negativo) y el resto de los bits para el modulo del número.
 - $+10_{10} = 00001010_2$, MSB = 0.
 - $-10_{10} = 10001010_2$, MSB = 1.
 - *Rango de representación:* para 8 bits, $(-2^{n-1} + 1; 2^{n-1} - 1)$.
 - *Mayor número:* $+127_{10} = 01111111_2$
 - *Menor número:* $-127_{10} = 11111111_2$ ----> RANGO SIMÉTRICO.
 - $+0_{10} = 00000000_2$
 - $-0_{10} = 10000000_2$ ----> DOBLE REPRESENTACIÓN DEL CERO.
- **COMPLEMENTO A LA BASE-1 (C_1):** utiliza el MSB para el signo. Para los números positivos, los n-1 bits restantes representan el modulo y los números negativos se obtienen invirtiendo todos los dígitos (incluido el de signo) del número positivo.
 - $+5_{10} = 0000101_2$, MSB = 0.
 - $-5_{10} = 11111010_2$, MSB = 1.
 - *Rango de representación:* para 8 bits, $(-2^{n-1} + 1; 2^{n-1} - 1)$.
 - *Mayor número:* $+127_{10} = 01111111_2$
 - *Menor número:* $-127_{10} = 10000000_2$ ----> RANGO SIMÉTRICO.
 - $+0_{10} = 00000000_2$
 - $-0_{10} = 11111111_2$ ----> DOBLE REPRESENTACIÓN DEL CERO.

- **COMPLEMENTO A LA BASE (C_2):** igual que C_1 pero el negativo se obtiene en dos pasos: se realiza el C_1 al número positivo y al resultado se le suma 1.
 - $+10_{10} = 0\ 0\ 0\ 0\ 1\ 0\ 1\ 0_2$, MSB = 0.
 - $-10_{10} = 1\ 1\ 1\ 1\ 0\ 1\ 1\ 0_2$, MSB = 1.
 - *Rango de representación:* para 8 bits, $(-2^{n-1}; 2^{n-1} - 1)$.
 - *Mayor número:* $+127_{10} = 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1_2$
 - *Menor número:* $-128_{10} = 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0_2$ ----> RANGO ASIMÉTRICO.
 - $+0_{10} = 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0_2$
 - $-0_{10} = NO\ EXISTE$ (1000000₂ es -128 por convención).
- **EXCESO A 2^{n-1} :** no utiliza el MSB para representar el signo. El número a representar viene dado por su valor más el valor del exceso (en caso de 8 bits, $2^7 = 128$). De esta manera:
 - $+10_{10} = +10 + 128 = 138_{10} = 1\ 0\ 0\ 0\ 1\ 0\ 1\ 0_2$
 - $-10_{10} = -10 + 128 = 118_{10} = 0\ 1\ 1\ 1\ 0\ 1\ 1\ 0_2$
 - *Rango de representación:* para 8 bits, $(-2^{n-1}; 2^{n-1} - 1)$.
 - *Mayor número:* $+127_{10} = 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1_2$
 - *Menor número:* $-128_{10} = 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0_2$ ----> RANGO ASIMÉTRICO.
 - $0_{10} = 0 + 128 = 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0_2$

• 2.6 Capacidad de representación, Resolución y Rango.

♦ **CAPACIDAD DE REPRESENTACION:** es la cantidad de números que se pueden representar en un sistema. Tanto en punto fijo o no, es b^n .

♦ **RESOLUCIÓN:** es la mínima diferencia entre un número representable y el siguiente. Se podría decir que es la diferencia entre 0 y el siguiente.

♦ **RANGO:** el rango de un sistema está dado por el número mínimo representable y el número máximo representable. El número máximo para la parte entera es $b^n - 1$.

• SISTEMA BINARIO CON SIGNO (BCS) DE 5 BITS:

- *Número máximo:* $2^4 - 1 = 15_{10} = 0\ 1\ 1\ 1\ 1_2$
- *Número mínimo:* $1\ 1\ 1\ 1\ 1_2 = -15_{10}$
- *Rango:* $[-15; 15]$
- *Capacidad de Representación:* $2^5 = 32$ números (doble representación del cero)
- *Resolución:* $1 - 0 = 1$

• 2.7 Banderas de condición (Flags).

Son bits que existen en la CPU y que de acuerdo al resultado de una operación, tomaran el valor 0 o 1. Posteriormente, pueden ser consultados por el programador utilizando instrucciones especiales, para la toma de decisiones.

♦ BANDERAS ARITMETICAS (Existen otras).

- **Z (Cero):** toma el valor 1 si el resultado fue 0, de lo contrario, toma el valor 0.
- **N (Negativo):** toma el valor del MSB (1 negativo y 0 positivo).
- **V (Overflow):** toma el valor 1 indicando en una condición de desborde en C_2 .
- **C (Carry):** toma el valor 1 si existe acarreo en la resta. En operaciones sin signo, toma valor 1 si el resultado está fuera de rango.

$0^1 1^1 0^1 1$	N=1	$0\ 1\ 0\ 1$	N=1
$+ 0\ 1\ 1\ 1$	Z=0	$- 0\ 1\ 1\ 1$	Z=0
-----	C=0	-----	C=1
$1\ 1\ 0\ 0$	V=1	$1\ 1\ 1\ 0$	V=0

• 2.8 Decimal Codificado en Binario (BCD).

Binary-Coded Decimal (BCD) o **Decimal Codificado en Binario** es un estándar para representar números decimales en el sistema binario, en donde cada dígito decimal es codificado con una secuencia de 4 bits. Con esta codificación especial de los dígitos decimales en el sistema binario, se pueden realizar operaciones aritméticas como suma, resta, multiplicación y división de números en representación decimal, sin perder en los cálculos la precisión ni tener las inexactitudes en que normalmente se incurre con las conversiones de decimal a binario puro y de binario puro a decimal. La conversión de los números decimales a BCD y viceversa es muy sencilla, pero los cálculos en BCD se llevan más tiempo y son algo más complicados que con números binarios puros.

♦ TIPOS DE CODIGOS BCD MÁS COMUNES:

DECIMAL	NATURAL	AIKEN	5 4 2 1	EXCESO 3
0	0000	0000	0000	0011
1	0001	0001	0001	0100
2	0010	0010	0010	0101
3	0011	0011	0011	0110
4	0100	0100	0100	0111
5	0101	1011	1000	1000
6	0110	1100	1001	1001
7	0111	1101	1010	1010
8	1000	1110	1011	1011
9	1001	1111	1100	1100

En **BCD** cada cifra que representa un dígito decimal se representa con su equivalente binario en cuatro bits (nibble o cuarteto). Esto es así porque es el número de bits necesario para representar el nueve, el número más alto que se puede representar en BCD.

Una forma sencilla de calcular números en BCD es sumando normalmente bit a bit, y si el conjunto de 4 bits sobrepasa el número 9, entonces se le suma un 6 (0110) en binario, para poder volver a empezar, como si hiciéramos un módulo al elemento sumante.

Con el BCD sólo se utilizan 10 de las 16 posibles combinaciones que se pueden formar con números de 4 bits, por lo que el sistema pierde capacidad de representación, aunque se facilita la compresión de los números. Esto es porque el BCD sólo se usa para representar cifras, no números en su totalidad. Para números de más de una cifra hacen falta dos números BCD. Desde que los sistemas informáticos empezaron a almacenar los datos en conjuntos de ocho bits (octeto), hay dos maneras comunes de almacenar los datos BCD:

- **Desempaquetado:** Omisión de los cuatro bits más significativos.
- **Empaquetado:** Almacenamiento de dos datos BCD, en el que también se incluye en primer lugar el signo, por lo general con **1100** para el positivo y **1101** para el negativo.

De este modo, el número 127 sería representado como (11110001, 11110010, 11110111) en el EBCDIC o (00010010, 01111100) en el BCD empaquetado.

127 ---→ Desempaquetado (sin signo): 11110001, 11110010, 11110111 = F1F2F7 (F indica sin signo).
+127 ---→ Empaquetado (con signo): 00010010, 0111**1100** = 127**C**.

+834 ---→ Desempaquetado (con signo): 11111000, 11110011, **1100**0100 = F8F3**C4**.
-834 ---→ Empaquetado (con signo): 10000011, 0100**1101** = 834**D**.

+8435 ---→ Desempaquetado (con signo): 11111000, 11110100, 11110011, **1100**0101 = F8F4F3**C5**.
+8435 ---→ Empaquetado (con signo): 00001000, 01000011, 0101**1100** = 08435**C**.

El BCD sigue siendo ampliamente utilizado para almacenar datos, en aritmética binaria o en electrónica. Los números se pueden mostrar fácilmente en visualizadores de siete segmentos enviando cada cuarteto BCD a un visualizador. La BIOS de un ordenador personal almacena generalmente la fecha y la hora en formato BCD; probablemente por razones históricas se evitó la necesidad de su conversión en ASCII.

La ventaja del código BCD frente a la representación binaria clásica es que no hay límite para el tamaño de un número. Los números que se representan en formato binario están generalmente limitados por el número mayor que se pueda representar con 8, 16, 32 o 64 bits. Por el contrario, utilizando BCD, añadir un nuevo dígito sólo implica añadir una nueva secuencia de 4 bits.

• 2.9 Representación de números reales (Punto Flotante).

Punto Flotante surge de la necesidad de representar número reales y enteros con un rango de representación mayor que el que nos ofrece Punto Fijo y así posibilitar a la computadora el tratamiento de números muy grandes y muy pequeños. En su representación se utiliza la notación científica de la siguiente forma:

$$V(X) = \text{Mantisa} * \text{Base}^{\text{Exponente}}$$

De esta forma, 976.000.000.000.000 puede representarse como $9,76 * 10^{14}$ y 0,0000000000000976 puede expresarse como $9,76 * 10^{-14}$. Esta misma técnica puede aplicarse a números binarios. Un número binario en Punto Flotante puede almacenarse en una palabra binaria con tres campos. Si tenemos 32 bits, la representación será:

- 1 bit para el **SIGNO** (0 positivo y 1 negativo).
- 8 bits para el **EXPONENTE**.
- 23 bits para la **MANTISA** (parte significativa del número).

SIGNO	EXPONENTE	MANTISA
1 bit	8 bits	23 bits

Para simplificar los cálculos se requiere usualmente que los números estén "Normalizados", esto quiere decir que el bit más a la izquierda de la mantisa sea uno (0,1M o 1,1M), por lo tanto no hace falta almacenarlo, ya que al igual que la base, ambos se encuentran implícitos.

♦ ESTANDAR DEL IEEE754.

Este estándar se desarrolló para facilitar la portabilidad de los programas de un procesador a otro y para alentar el desarrollo de programas numéricos sofisticados. Ha sido ampliamente adoptado y se utiliza prácticamente en todos los procesadores actuales. El estándar del IEEE define el formato para precisión simple de 32 bits y para precisión doble de 64 bits.

• **PRECISIÓN SIMPLE (32 bits).**

SIGNO (1 bit)	EXPONENTE (8 bits)	MANTISA (23 bits)
----------------------	---------------------------	--------------------------

• **PRECISIÓN DOBLE (64 bits).**

SIGNO (1 bit)	EXPONENTE (11 bits)	MANTISA (52 bits)
----------------------	----------------------------	--------------------------

- **EXPONENTE** sin signo y en exceso $2^{n-1} - 1$ (127 para precisión simple y 1023 para precisión doble).

• **CASOS ESPECIALES:**

SIGNO	EXPONENTE	MANTISA	SIGNIFICADO
0	Todos 1	Todos 0	Más infinito ($+\infty$).
1	Todos 1	Todos 0	Menos infinito ($-\infty$).
0 ó 1	Todos 1	Distinta de 0	Not a Number (NaN).
0 ó 1	Todos 0	Todos 0	Ceros (Positivo y Negativo).
0 ó 1	Todos 0	Distinta de 0	Número muy pequeño (cerca al cero).

• **RANGOS DE REPRESENTACION Y ERRORES:**

MAXIMO Nº POSITIVO	$MANTISA\ MAXIMA * BASE^{MAXIMO\ EXPONENTE\ POSITIVO}$
MINIMO Nº POSITIVO	$MANTISA\ MINIMA * BASE^{MAXIMO\ EXPONENTE\ NEGATIVO}$
MAXIMO Nº NEGATIVO	$-(MANTISA\ MINIMA) * BASE^{MAXIMO\ EXPONENTE\ NEGATIVO}$
MAXIMO Nº POSITIVO	$-(MANTISA\ MAXIMA) * BASE^{MAXIMO\ EXPONENTE\ POSITIVO}$

- **ERROR ABSOLUTO:** $EA(x) = |N^{\circ} \text{ a Representar} - N^{\circ} \text{ Representado}|$
- **ERROR RELATIVO:** $ER(x) = |N^{\circ} \text{ a Representar} - N^{\circ} \text{ Representado}| / N^{\circ} \text{ a Representar}$

- Representar el número $-118,625_{10}$ en precisión simple:

$118_{10} = 1110110_2$ ---→ 1110110,101 ---→ NORMALIZADO: $1,110110101 * 2^6$
 $0,625_{10} = 101_2$ ---→ EXPONENTE ---→ $6+127 = 133 = 1000101_2$

SIGNO= 1	EXPONENTE= 1000101	MANTISA= 1101101010000000000000
-----------------	---------------------------	--

• 2.10 Representaciones Alfanuméricas.

Estas representaciones determinan la forma en la que se representara la información de tipo texto que genera la computadora, donde cada atributo que una letra, dígito o símbolo recibe el nombre de carácter, el cual es único dentro del sistema. Existen también grupos adicionales conocidos como "Caracteres de Control", los cuales se utilizan para controlar a los periféricos.

Existen diferentes formatos en la actualidad, tales como el *ASCII*, *EBCDIC* y *UNICODE*. Todos poseen el mismo grupo básico de caracteres:

- **Letras Minúsculas:** a, b, c ... x, y, z.
- **Letras Mayúsculas:** A, B, C ... X, Y, Z.
- **Dígitos:** 0, 1, 2 ... 7, 8, 9.
- **Caracteres Especiales:** @, #, €, &, etc.

♦ **ASCII (American Standard Code for Information Interchange).**

Es un código de caracteres basado en el alfabeto latino, tal como se usa en inglés moderno y en otras lenguas occidentales. Fue creado en 1963 por el Comité Estadounidense de Estándares (ASA, conocido desde 1969 como el Instituto Estadounidense de Estándares Nacionales, o ANSI) como una refundición o evolución de los conjuntos de códigos utilizados entonces en telegrafía. Más tarde, en 1967, se incluyeron las minúsculas, y se redefinieron algunos códigos de control para formar el código conocido como US-ASCII.

El código **ASCII** utiliza 7 bits para representar los caracteres, aunque inicialmente empleaba un bit adicional (bit de paridad) que se usaba para detectar errores en la transmisión. A menudo se llama incorrectamente ASCII a otros códigos de caracteres de 8 bits, como el estándar ISO-8859-1, que es una extensión que utiliza 8 bits para proporcionar caracteres adicionales usados en idiomas distintos al inglés, como el español.

Fue publicado como estándar por primera vez en 1967 y fue actualizado por última vez en 1986. En la actualidad define códigos para 32 caracteres no imprimibles, de los cuales la mayoría son caracteres de control obsoletos que tienen efecto sobre cómo se procesa el texto, más otros 95 caracteres imprimibles que les siguen en la numeración (empezando por el carácter espacio). Casi todos los sistemas informáticos actuales utilizan el código ASCII o una extensión compatible para representar textos y para el control de dispositivos que manejan texto como el teclado. No deben confundirse los códigos ALT+número de teclado con los códigos ASCII.

ASCII es, en sentido estricto, un código de siete bits, lo que significa que usa cadenas de bits representables con siete dígitos binarios (que van de 0 a 127 en base decimal) para representar información de caracteres. En el momento en el que se introdujo el código ASCII muchos ordenadores trabajaban con grupos de ocho bits (bytes u octetos), como la unidad mínima de información; donde el octavo bit se usaba habitualmente como bit de paridad con funciones de control de errores en líneas de comunicación u otras funciones específicas del dispositivo.

Las máquinas que no usaban la comprobación de paridad asignaban al octavo bit el valor cero en la mayoría de los casos, aunque otros sistemas como las computadoras Prime, que ejecutaban PRIMOS ponían el octavo bit del código ASCII a uno. El código ASCII define una relación entre caracteres específicos y secuencias de bits; además de reservar unos cuantos códigos de control para el procesador de textos, y no define ningún mecanismo para describir la estructura o la apariencia del texto en un documento; estos asuntos están especificados por otros lenguajes como los lenguajes de etiquetas.

♦ **Unicode.**

Unicode es un estándar de codificación de caracteres diseñado para facilitar el tratamiento informático, transmisión y visualización de textos de múltiples lenguajes y disciplinas técnicas, además de textos clásicos de lenguas muertas. El término Unicode proviene de los tres objetivos perseguidos: universalidad, uniformidad y unicidad. Especifica un nombre e identificador numérico único para cada carácter o símbolo, el code point o punto de código, además de otras informaciones necesarias para su uso correcto: direccionalidad, mayúsculas y otros atributos. Unicode trata los caracteres alfabéticos, ideográficos y símbolos de forma equivalente, lo que significa que se pueden mezclar en un mismo texto sin la introducción de marcas o caracteres de control.

El establecimiento de Unicode ha sido un ambicioso proyecto para reemplazar los esquemas de codificación de caracteres existentes, muchos de los cuales están muy limitados en tamaño y son incompatibles con entornos plurilingües. Unicode se ha vuelto el más extenso y completo esquema de codificación de caracteres, siendo el dominante en la internacionalización y adaptación local del software informático. El estándar ha sido implementado en un número considerable de tecnologías recientes, que incluyen XML, Java y sistemas operativos modernos.

UNIDAD 03 – LÓGICA DIGITAL

Los circuitos electrónicos que conforman una computadora suelen estar capacitados para reconocer señales eléctricas de tipo digital; por lo tanto se hace necesario que los métodos de codificación internos tengan su origen en el sistema binario y con ellos se pueda representar todo tipo de información.

La electrónica digital está fundamentada en la base matemática formada por el álgebra de Boole. Este método considera que todos los elementos poseen únicamente dos estados (verdadero o falso), sin estados intermedios.

• 3.1 Álgebra de Boole.

En 1815 George Boole propuso una herramienta matemática llamada **Álgebra de Boole**. Luego en 1938 Claude Shannon propuso que con esta álgebra es posible modelar los llamados Sistemas Digitales. El Álgebra de Boole es un sistema matemático que utiliza variables y operadores lógicos. Las variables pueden valer 0 ó 1. Y las operaciones básicas son NOT (Negación), OR (Suma) y AND (Multiplicación).

- **COMPLEMENTACIÓN LÓGICA (NOT):** sea una variable A con valor 1, diremos que su complemento o estado inverso será $\sim A = 0$.
- **SUMA LÓGICA (OR):** es la suma de dos o más conjuntos. Esta operación también se denomina reunión de conjuntos y puede representarse como $A+B$ o $A \cup B$.
- **PRODUCTO LÓGICO (AND):** es la multiplicación de dos o más conjuntos. También se denomina intersección de conjuntos y se representa como $A*B$ o $A \cap B$.
- **NEGACIÓN CONJUNTA (NOR).**
- **NEGACIÓN ALTERNATIVA (NAND).**
- **DISYUNCIÓN EXCLUSIVA (XOR).**

A	B	NOT A	A OR B	A AND B
0	0	1	0	0
0	1	1	1	0
1	0	0	1	0
1	1	0	1	1

A	B	A NOR A	A NAND B	A XOR B
0	0	1	1	0
0	1	0	1	1
1	0	0	0	1
1	1	0	0	0

♦ POSTULADOS DEL ÁLGEBRA DE BOOLE.

Conmutativa	$A+B = B+A$	$A*B = B*A$
Distributiva	$A+(B*C) = (A+B) * (A+C)$	$A*(B+C) = (A*B) + (A*C)$
Elemento Neutro	$0+A = A$	$1*A = A$
Elemento Complemento	$A+\sim A = 1$	$A*\sim A = 0$
Asociativa	$A+(B+C) = (A+B)+C$	$A*(B*C) = (A*B)*C$
DeMorgan	$\sim(A+B) = \sim A*\sim B$	$\sim(A*B) = \sim A+\sim B$



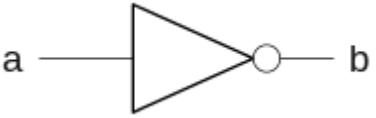



♦ FUNCIONES BOOLEANAS.

Es todo conjunto de variables relacionadas entre sí por una expresión que representa:

- La combinación de un conjunto finito de símbolos, representando constantes o variables.
- Unidos por las operaciones AND (producto lógico), OR (suma lógica) o NOT (complementación).

- **Término producto:** es una expresión lógica que consiste en un conjunto de variables (o sus complementadas) unidas por la operación AND. $f(x, y, z) = \sim x * y$
- **Término suma:** es una expresión lógica que consiste en un conjunto de variables (o sus complementadas) unidas por la operación OR. $f(x, y, z) = x + \sim y$
- Un término producto o **MINTERM:** expresión lógica que consiste en un conjunto de TODAS las variables (o sus complementadas) unidas por la operación AND. $f(x, y, z) = \sim x * y * z$
- Un término suma o **MAXTERM:** expresión lógica que consiste en un conjunto de TODAS las variables (o sus complementadas) unidas por la operación OR. $f(x, y, z) = x + \sim y + \sim z$

• 3.2 Puertas lógicas.

NOMBRE	SÍMBOLO	FUNCIÓN	TABLA															
AND		$f(a,b) = a * b$	<table><tr><th>A</th><th>B</th><th>f(a,b)</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>	A	B	f(a,b)	0	0	0	0	1	0	1	0	0	1	1	1
			A	B	f(a,b)													
			0	0	0													
			0	1	0													
			1	0	0													
1	1	1																
OR		$f(a,b) = a + b$	<table><tr><th>A</th><th>B</th><th>f(a,b)</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>	A	B	f(a,b)	0	0	0	0	1	1	1	0	1	1	1	1
			A	B	f(a,b)													
			0	0	0													
			0	1	1													
			1	0	1													
1	1	1																
NOT		$f(a) = \sim a$	<table><tr><th>A</th><th>f(a)</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	f(a)	0	1	1	0									
			A	f(a)														
			0	1														
1	0																	
NAND		$f(a,b) = \sim(a * b)$	<table><tr><th>A</th><th>B</th><th>f(a,b)</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>	A	B	f(a,b)	0	0	1	0	1	1	1	0	1	1	1	0
			A	B	f(a,b)													
			0	0	1													
			0	1	1													
			1	0	1													
1	1	0																
NOR		$f(a,b) = \sim(a + b)$	<table><tr><th>A</th><th>B</th><th>f(a,b)</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>	A	B	f(a,b)	0	0	1	0	1	0	1	0	0	1	1	1
			A	B	f(a,b)													
			0	0	1													
			0	1	0													
			1	0	0													
1	1	1																
XOR		$f(a,b) = a \oplus b$ $= a * \sim b + \sim a * b$	<table><tr><th>A</th><th>B</th><th>f(a,b)</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>	A	B	f(a,b)	0	0	0	0	1	1	1	0	1	1	1	0
			A	B	f(a,b)													
			0	0	0													
			0	1	1													
			1	0	1													
1	1	0																

• 3.3 Circuitos combinacionales.

Un **Circuito Combinacional** es un conjunto de puertas lógicas interconectadas entre sí, cuya salida en un momento dado, es función solamente de la entrada en ese instante. La aparición de la entrada viene seguida casi inmediatamente por la aparición de la salida, con solo retardos de las puertas. En general, un circuito combinacional consiste en n entradas y m salidas. Ejemplos son: Multiplexor, Demultiplexor, Codificador, Decodificador y Sumador.

♦ IMPLEMENTACION DE FUNCIONES BOOLEANAS.

A	B	C	f(a,b,c)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Se puede expresar esta función booleana detallando las combinaciones de los valores de A, B y C que hacen que **f** valga 1. Este tipo de expresiones reciben el nombre de *suma de productos (SOP)*. También se pueden expresar en forma de *productos de suma (POS)*, tomando los valores que hacen que **f** valga 0 y complementándolos.

- **SUMA DE PRODUCTOS:** $f(a,b,c) = \sim a * b * \sim c + \sim a * b * c + a * b * \sim c$
- **PRODUCTOS DE SUMA:** $f(a,b,c) = (a + b + c) * (a + b + \sim c) * (\sim a + b + c) * (\sim a + b + \sim c) * (\sim a + \sim b + \sim c)$

• 3.4 Minterminos y Maxiterminos.

- **MINITERMINOS (mi):** es una expresión lógica que consiste en un conjunto de todas las variables unidas por la operación AND. Si disponemos de la tabla de verdad podemos obtener la función como suma de Minterminos, tomando aquellas combinaciones de valores para las cuales la función vale 1, asignando un 1 para la variable sin complementar y 0 para las complementadas.
- **MAXITERMINOS (Mi):** es una expresión lógica que consiste en un conjunto de todas las variables unidas por la operación OR. Con la tabla de verdad podemos obtener la función como productos de Maxiterminos, tomando aquellas combinaciones de valores para las cuales la función vale 0, asignando un 0 para la variable sin complementar y 1 para las complementadas.
- Teniendo en cuenta la función: $f(x, y, z) = x * (y + z)$

MINITERMINOS	X	Y	Z	$f(x, y, z)$	MAXITERMINOS
$m_0 = \sim x * \sim y * \sim z$	0	0	0	0	$M_0 = x + y + z$
$m_1 = \sim x * \sim y * z$	0	0	1	0	$M_1 = x + y + \sim z$
$m_2 = \sim x * y * \sim z$	0	1	0	0	$M_2 = x + \sim y + z$
$m_3 = \sim x * y * z$	0	1	1	0	$M_3 = x + \sim y + \sim z$
$m_4 = x * \sim y * \sim z$	1	0	0	0	$M_4 = \sim x + y + z$
$m_5 = x * \sim y * z$	1	0	1	1	$M_5 = \sim x + y + \sim z$
$m_6 = x * y * \sim z$	1	1	0	1	$M_6 = \sim x + \sim y + z$
$m_7 = x * y * z$	1	1	1	1	$M_7 = \sim x + \sim y + \sim z$

$$f(x, y, z) = x * \sim y * z + x * y * \sim z + x * y * z = (x + y + z) * (x + y + \sim z) * (x + \sim y + z) * (x + \sim y + \sim z) * (\sim x + y + z)$$

• 3.5 Mapas de Karnaugh.

Son una forma conveniente de simplificar una función booleana con pocas variables. El mapa es un conjunto de 2^n cuadrículas, donde cada cuadrícula tiene asociada una combinación de la tabla de verdad en la que se escribe el valor de la salida para ese Mintermino/Maxitermino.

- **REGLAS DE APLICACIÓN:**
 - Agrupar todas las celdas con el mismo valor en uno o más grupos adyacentes.
 - La cantidad de celdas en un grupo debe ser potencia de 2 (2, 4, 8).
 - Maximizar la cantidad de celdas en cada grupo.
 - Minimizar la cantidad de grupos.
 - Superponer grupos siempre que sea posible.
- $f(a, b, c) = a * \sim b * \sim c + a * \sim b * c + a * b * \sim c + \sim a * b * \sim c$

• TABLA DE VERDAD:

A	B	C	$f(a, b, c)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0


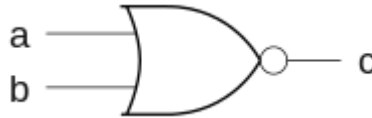
• MAPA DE KARNAUGH:

	$\sim b \sim c$	$\sim b c$	$b c$	$b \sim c$
$\sim a$				1
a	1	1		1

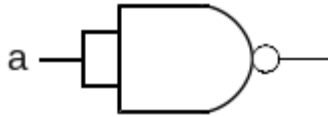
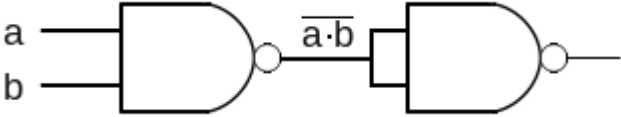
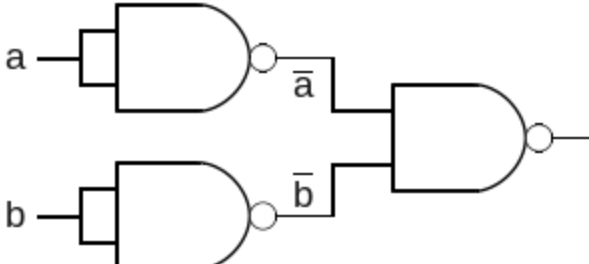
- Solución: $f(a, b, c) = a \sim b + b \sim c$

• 3.6 Implementación de circuitos con puertas NAND y NOR.

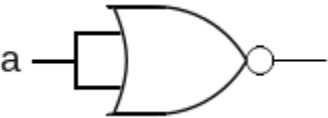
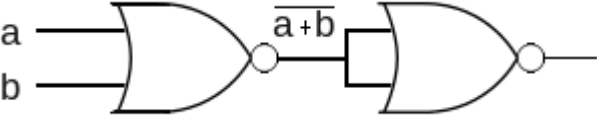
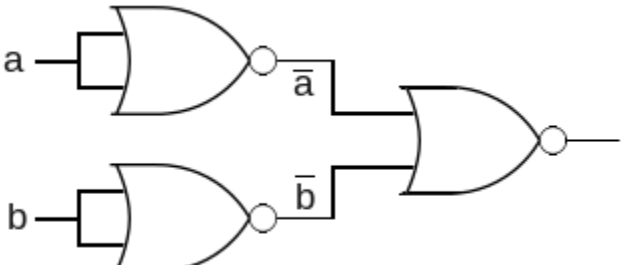
Los circuitos combinacionales se construyen más a menudo con compuertas NAND o NOR, esto es así porque cualquier función lógica puede expresarse usando solo estas puertas, además son las más sencillas de construir.

NAND	NOR
	
$f(a, b) = \sim(a * b) = \sim a + \sim b$	$f(a, b) = \sim(a + b) = \sim a * \sim b$

♦ NOT, AND Y OR CON COMPUERTAS NAND:

OPERACIÓN	COMPUERTA	FUNCIÓN DE SALIDA
NOT		$\sim(a * a) = \sim a$
AND		$a * b$
OR		$\sim(\sim a * \sim b) = a + b$

♦ NOT, AND Y OR CON COMPUERTAS NOR:

OPERACIÓN	COMPUERTA	FUNCIÓN DE SALIDA
NOT		$\sim(a + a) = \sim a$
AND		$a * b$
OR		$\sim(\sim a + \sim b) = a * b$

• PASOS PARA LA CONSTRUCCIÓN DE LOS CIRCUITOS:

1. Implementación normal del circuito con puertas AND/OR/NOT.
2. Se sustituyen con las funciones equivalentes NAND/NOR.
3. Se eliminan los pares inversores.

• 3.7 Circuitos secuenciales.

Un **Circuito Secuencial** es un conjunto de puertas lógicas interconectadas entre sí, en el cual, la salida en cualquier instante de tiempo t depende de las entradas en ese instante y del valor de las entradas en $t-1$. Se las puede clasificar como asíncronos (con retardos asociados a las puertas lógicas) o síncronos (solamente se permiten cambios de estados en instantes marcados por el reloj). Ejemplos son: Biestables RS, JK, D, Registros y Contadores.

• 3.8 Biestables (Flip-Flop ó Latch).

Un **Biestable**, también llamado Flip-Flop, es la forma más sencilla de un Circuito Secuencial, con la capacidad de permanecer en uno de dos estados posibles (0 y 1) durante un tiempo indefinido en ausencia de entrada, utilizando el principio de retroalimentación. Posee dos salidas que siempre son complementarias entre sí (Q y $\sim Q$) y puede funcionar como una memoria de 1 bit. Dependiendo del tipo de entradas, se pueden clasificar en:

- **Sincronismo activado por nivel:** el sistema lee las entradas cuando el reloj está en estado ALTO (1 o H) o BAJO (0 o L). También llamado activación por pulso.



- **Sincronismo activado por flanco:** el sistema lee las entradas cuando se produce la transición de sus pulsos, puede ser: ascendente (cuando cambia de 0 a 1) o descendente (cuando cambia de 1 a 0).



- Un **Biestable RS** consiste en un circuito digital de dos entradas, una denominada R (Reset), que cuando tiene valor 1 pone en 0 la salida, y otra denominada S (Set), que cuando tiene valor 1 pone en 1 la salida. Cuando las dos entradas tienen valor 0, no se producen cambios en la salida, pero cuando ambos valores valen 1, el Biestable RS no sabe bien cómo actuar y la salida puede cambiar o quedarse inalterada de forma aleatoria. Existen dos tipos de Biestables RS, uno es asíncrono y el otro es síncrono. En el **Biestable RS Síncrono**, además de las dos entradas R y S, posee una entrada más, denominada CLK, que consiste en un reloj. Funciona exactamente igual que el RS asíncrono, con la diferencia de que sus estados cambian durante los pulsos del reloj.
- El **Biestable JK** fue diseñado para evitar el problema de indeterminación que poseen los Biestables RS cuando ambas entradas tenían el valor 1. En el Biestable JK, la entrada J tiene la función de SET (poner en 1 la salida cuando J vale 1) y la entrada K, funciona como RESET (poniendo en 0 la salida cuando K vale 1). Cuando ambas entradas valen 0, no se producen cambios y cuando ambas entradas valen 1, se realiza la función denominada conmutación: la salida se invierte.
- Un **Biestable D** es un circuito digital, también denominado 'Biestable de datos' porque sirve en efecto para almacenar 1 bit de datos. Posee una sola entrada D (Data) y la salida Q, obtiene el valor de la entrada D cuando la señal del reloj se encuentra activada.

BIESTABLE RS	BIESTABLE JK	BIESTABLE D
<p>BIESTABLE RS SINCRONO</p> <p>BIESTABLE RS ASINCRONO</p>		

UNIDAD 04 – UNIDAD CENTRAL DE PROCESAMIENTO (CPU)

● 4.1 Organización de la Unidad Central de Procesamiento (CPU).

La **Unidad Central de Procesamiento (CPU)** es el cerebro de la computadora. Es el encargado de realizar el control y procesamiento de los datos. Sus funciones principales son:

- Ejecutar las instrucciones de los programas almacenados en la memoria del sistema.
- Controlar la transferencia de datos entre la CPU y los circuitos de memoria y de E/S
- Responder a las peticiones de servicio procedentes de los dispositivos de E/S.

Para que un programa pueda ser ejecutado por la CPU, debe estar guardado en un determinado lugar de la memoria del sistema y escrito en un lenguaje que la CPU pueda entender. Un programa, básicamente, es una lista de instrucciones que la CPU lee ordenadamente, las interpreta y posteriormente controla su ejecución una tras otra. La ejecución completa de cada instrucción lleva varios pasos (llamados ciclo de instrucción) y para realizarlos la CPU se vale de dos unidades: La Unidad de Control (UC) y la Unidad Aritmético-Lógica (ALU).

♦ **LA UNIDAD DE CONTROL (UC):** Es muy importante que un computador tenga unidades funcionales muy eficientes y rápidas, pero si no se coordinan y no se controlan correctamente, es imposible aprovechar todas las potencialidades que se habían previsto en el diseño. La UC controla todas las funciones que realiza una computadora. Tiene 3 funciones principales:

- Controlar el funcionamiento de los componentes internos de la CPU.
- Leer e Interpretar las instrucciones de los programas.
- Control el flujo de datos hacia y desde la CPU.

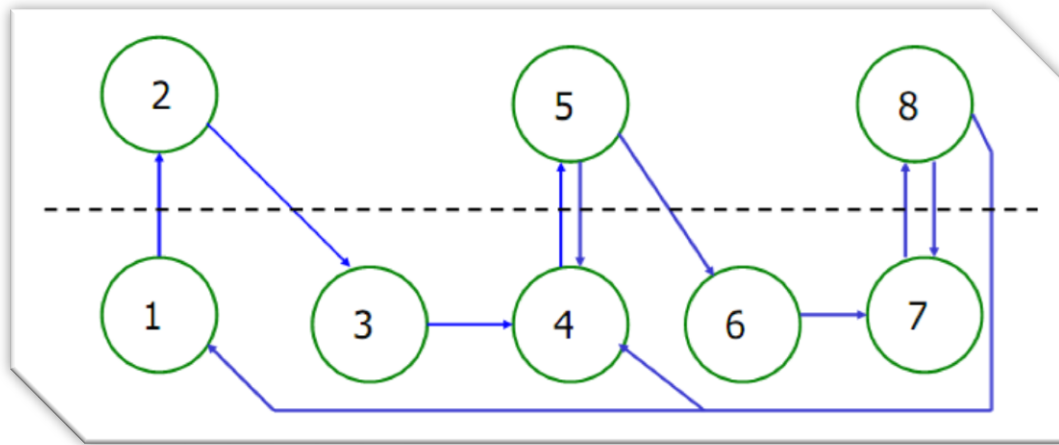
♦ **LA UNIDAD ARITMETICO-LOGICA (ALU):** La unidad aritmética y lógica o ALU, es un circuito combinacional capaz de realizar operaciones aritméticas y lógicas con números enteros y también con números reales. Las operaciones que puede efectuar vienen definidas por el conjunto de instrucciones aritméticas y lógicas de las que dispone el juego de instrucciones del computador. Las operaciones aritméticas habituales que puede hacer una ALU incluyen suma, resta, multiplicación y división. Además, se pueden incluir operaciones específicas de incremento positivo (+1) o negativo (-1). Dentro de las operaciones lógicas se incluyen operaciones AND, OR, NOT, XOR, operaciones de desplazamiento de bits a la izquierda y a la derecha y operaciones de rotación de bits.

♦ **REGISTROS INTERNOS DE LA CPU:** La CPU también consta de un conjunto de posiciones de almacenamiento, llamadas registros. Por lo general, todos los registros tienen el mismo tamaño. Pueden leerse y escribirse a alta velocidad porque están dentro de la CPU. Existen registros de propósito general y otros con fines específicos.

♦ **BUS INTERNO DE LA CPU:** Todas las partes de la CPU están unidas mediante diversas líneas eléctricas. El conjunto de estas líneas se denominan bus interno de la CPU y se pueden diferenciar tres. Por este bus interno circulan los datos (bus de datos), las señales de control (bus de control) o las direcciones de memoria (bus de direcciones). Cuando se habla de un microprocesador de 32 bits, se está diciendo que el número de líneas del bus interno es de 32.

● 4.2 Ciclo de instrucción (Sin Interrupciones).

- 1) *Cálculo de dirección de la instrucción (**Instruction Address Calculation**).*
- 2) *Búsqueda instrucción (**Instruction Fetch**).*
- 3) *Decodificación de la instrucción (**Instruction Operation Decoding**).*
- 4) *Cálculo dirección operando (**Operand Address Calculation**).*
- 5) *Búsqueda del operando (**Operand Fetch**).*
- 6) *Operación sobre los datos (**Data Operation**).*
- 7) *Cálculo dirección resultado (**Operand Address Calculation**).*
- 8) *Almacenamiento resultado (**Operand Store**).*



Ciclo de Instrucción (Sin Interrupciones)

• 4.3 Interrupciones.

Prácticamente todos los computadores disponen de un mecanismo mediante el que otros módulos (E/S, Memoria) pueden interrumpir el procesamiento de la CPU. Las **interrupciones** proporcionan una forma de mejorar la eficiencia del procesador, mediante ellas el procesador puede dedicarse a ejecutar otras instrucciones mientras una operación de E/S está en marcha.

Cuando un dispositivo externo está preparado para aceptar datos del procesador, el módulo de E/S de este dispositivo envía una señal de petición de interrupción al procesador, que responde suspendiendo la operación del programa que estaba ejecutando y salta a un programa que da servicio a ese dispositivo concreto, conocido como gestor de interrupción, y prosigue con la ejecución del programa original después de haber dado servicio al dispositivo. El programa de usuario no tiene que incluir ningún código para posibilitar las interrupciones, el procesador y el Sistema Operativo son los responsables de detener el programa y permitir después que prosiga en el mismo punto.

♦ CLASES DE INTERRUPCIONES:

- **Programa:** Generadas por alguna condición que se produce como resultado de la ejecución de una instrucción, tal como el desbordamiento aritmético (Overflow), división por cero, intento de ejecutar una instrucción máquina inexistente, e intento de acceder fuera del espacio de memoria permitido para el usuario.
- **Temporización:** Generadas por un temporizador interno al procesador. Esto permite al Sistema Operativo realizar ciertas funciones de manera regular.
- **Entrada/Salida:** Generadas por un controlador de E/S, para indicar la finalización sin problemas de una operación o para avisar ciertas condiciones de error.
- **Fallo de hardware:** Generadas por un fallo como la falta de potencia de alimentación o un error de paridad en memoria.

• 4.4 Ciclo de instrucción (Con Interrupciones).

El funcionamiento de un computador cuando ejecuta programas consiste en una secuencia de ciclos de instrucción, con una instrucción máquina por ciclo. Cada **ciclo de instrucción** puede considerarse compuesto por varias pequeñas unidades como ciclo de captación, indirecto, ejecución e interrupción, si bien los únicos que aparecen siempre son los de captación y de ejecución.

- 1) *Ciclo de captación.*
- 2) *Lectura de operandos fuente.*
- 3) *Ciclo de ejecución.*
- 4) *Ciclo de Interrupciones.*

1) CICLO DE CAPTACIÓN: Las operaciones realizadas en esta fase son las siguientes:

A) Leer la instrucción: Cuando leemos la instrucción, el registro contador del programa (PC) nos indica la dirección de memoria donde está la instrucción que hemos de leer. Si el tamaño de la instrucción es superior a la palabra de la memoria, hay que hacer tantos accesos a la memoria como sean necesarios para leerla completamente y cargar toda esta información en el registro de instrucción (IR).

B) Decodificar la instrucción: Se identifican las diferentes partes de la instrucción para determinar qué operaciones hay que hacer en cada fase del ciclo de ejecución. Esta tarea la realiza la unidad de control del procesador leyendo la información que hemos cargado en el registro de instrucción (IR).

C) Actualizar el Contador de Programa: El contador del programa se actualiza según el tamaño de la instrucción, es decir, según el número de accesos a la memoria que hemos hecho para leer la instrucción.

2) LECTURA DE OPERANDOS FUENTE: Se debe repetir para todos los operandos que tenga la instrucción.

Las operaciones que hay que realizar en esta fase dependen del modo de direccionamiento que tengan los operandos: para los más simples, como el inmediato o el directo a registro, no hay que hacer ninguna operación; para los indirectos o los relativos, hay que hacer cálculos y accesos a memoria. Si el operando fuente es implícito, vamos a buscar el dato en el lugar predeterminado por aquella instrucción.

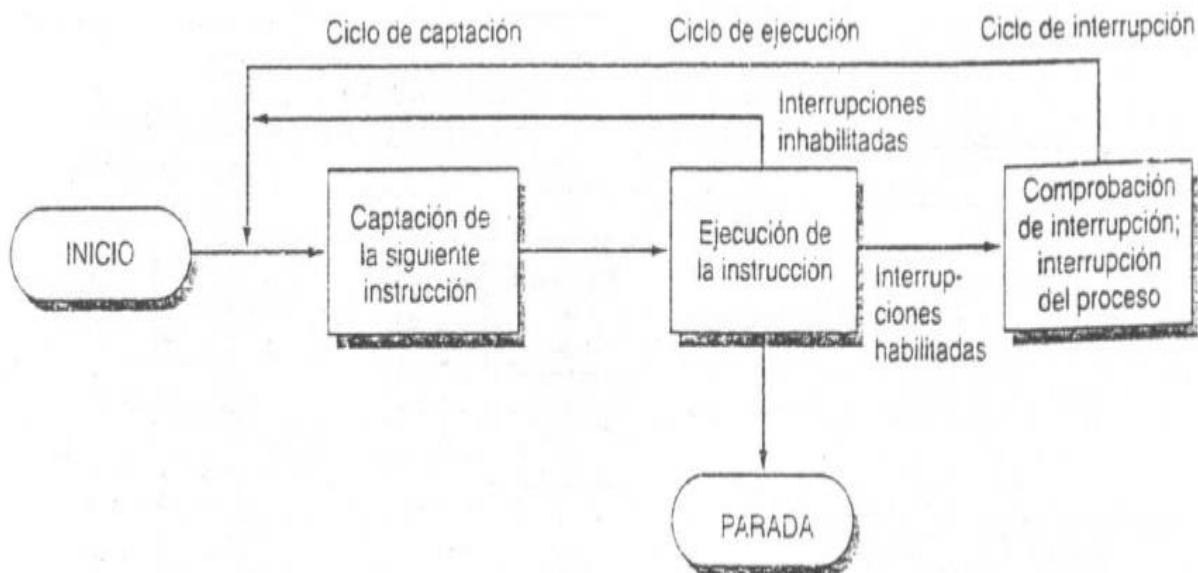
3) CICLO DE EJECUCIÓN:

A) Ejecución de la instrucción: Las operaciones que se llevan a cabo son diferentes para cada código de operación. Durante la ejecución, además de obtener el resultado de la ejecución de la instrucción, se pueden modificar los bits de resultado de la palabra de estado del procesador.

B) Almacenamiento del operando de destino (resultado): La función básica es recoger el resultado obtenido durante la ejecución y guardarlo en el lugar indicado por el operando, a menos que el operando sea implícito, en cuyo caso se guarda en el lugar predeterminado por aquella instrucción. El operando de destino puede ser uno de los operandos fuente; de esta manera, no hay que repetir los cálculos para obtener la dirección del operando.

4) CICLO DE INTERRUPCIONES:

En esta fase se verifica si se ha activado alguna línea de petición de interrupción del procesador en el transcurso de la ejecución de la instrucción. Si no se ha activado ninguna, continuamos el proceso normalmente; es decir, se acaba la ejecución de la instrucción en curso y se empieza la ejecución de la instrucción siguiente. En caso contrario, hay que transferir el control del procesador a la rutina de servicio de interrupción que debe atender esta interrupción y hasta que no se acabe no podemos continuar la ejecución de la instrucción en curso.



Ciclo de Instrucción (Con Interrupciones)

● 4.5 Formato de instrucción.

Para poder ejecutar un programa escrito en un lenguaje de alto nivel en un procesador, este programa se debe traducir primero a un lenguaje que pueda entender el procesador, diferente para cada familia de procesadores. El conjunto de instrucciones que forman este lenguaje se denomina juego de instrucciones o repertorio de instrucciones. Para poder definir un juego de instrucciones, habrá que conocer bien la arquitectura del computador para sacarle el máximo rendimiento.

◇ ELEMENTOS DE UNA INSTRUCCIÓN.

Los elementos que componen una instrucción, independientemente del tipo de arquitectura, son los siguientes:

- **Código de operación:** especifica la operación que hace la instrucción.
- **Operando Fuente:** para hacer la operación pueden ser necesarios uno o más operandos fuente; uno o más de estos operandos pueden ser implícitos.
- **Operando Destino:** almacena el resultado de la operación realizada. Puede estar explícito o implícito. Uno de los operandos fuente se puede utilizar también como operando destino.
- **Dirección de la instrucción siguiente:** especifica dónde está la instrucción siguiente que se debe ejecutar; suele ser una información implícita, ya que el procesador va a buscar automáticamente la instrucción que se encuentra a continuación de la última instrucción ejecutada. Solo las instrucciones de ruptura de secuencia especifican una dirección alternativa.

◇ TAMAÑO DE LAS INSTRUCCIONES.

Uno de los aspectos más importantes a la hora de diseñar el formato de las instrucciones es determinar su tamaño. En este sentido, encontramos dos alternativas:

- **Instrucciones de tamaño fijo:** todas las instrucciones ocuparán el mismo número de bits. Esta alternativa simplifica el diseño del procesador y la ejecución de las instrucciones puede ser más rápida.
- **Instrucciones de tamaño variable:** el tamaño de las instrucciones dependerá del número de bits necesario para cada una. Esta alternativa permite diseñar un conjunto amplio de códigos de operación, el direccionamiento puede ser más flexible y permite poner referencias a registros y memoria. Como contrapartida, aumenta la complejidad del procesador.

Es deseable que el tamaño de las instrucciones sea múltiplo del tamaño de la palabra de memoria.

Para determinar el tamaño de los campos de las instrucciones utilizaremos:

- 1) Código de operación:** La técnica más habitual es asignar un número fijo de bits de la instrucción para el código de operación y reservar el resto de los bits para codificar los operandos y los modos de direccionamiento.
- 2) Operandos y modos de direccionamiento:** Una vez fijados los bits del código de operación, podemos asignar los bits correspondientes a los operandos y a los modos de direccionamiento.

◇ NÚMERO DE OPERANDOS.

Las instrucciones pueden utilizar un número diferente de operandos según el tipo de instrucción del que se trate. Hay que tener presente que una misma instrucción en máquinas diferentes puede utilizar un número diferente de operandos según el tipo de arquitectura del juego de instrucciones que utilice la máquina. La instrucción aritmética de suma ($C = A + B$) utiliza dos operandos fuente (A y B) y produce un resultado que se almacena en un operando destino (C).

◇ LOCALIZACIÓN DE LOS OPERANDOS.

Los operandos representan los datos que hemos de utilizar para ejecutar una instrucción. Estos datos se pueden encontrar en lugares diferentes dentro del computador. Según la localización podemos clasificar los operandos de la siguiente manera:

- **Inmediato:** El dato está representado en la instrucción misma. Podemos considerar que se encuentra en un registro, el registro IR (registro de instrucción), y está directamente disponible para el procesador.
- **Registro:** El dato estará directamente disponible en un registro dentro del procesador.
- **Memoria:** El procesador deberá iniciar un ciclo de lectura/escritura a memoria. En el caso de operaciones de E/S, habrá que solicitar el dato al módulo de E/S adecuado y para acceder a los registros del módulo de E/S según el mapa de E/S que tengamos definido.

Según el lugar donde esté el dato, el procesador deberá hacer tareas diferentes con el fin de obtenerlo: puede ser necesario hacer cálculos y accesos a memoria indicados por el modo de direccionamiento que utiliza cada operando.

• 4.6 Repertorio de instrucciones.

◇ DISEÑO DEL REPERTORIO DE INSTRUCCIONES.

El **repertorio de instrucciones** de una arquitectura es la herramienta que tiene el programador para controlar la computadora; por lo tanto, debería facilitar y simplificar la tarea del programador. Por otra parte, las características del juego de instrucciones condicionan el funcionamiento del procesador y, por ende, tienen un efecto significativo en el diseño del procesador. Así, nos encontramos con el problema de que muchas de las características que facilitan la tarea al programador dificultan el diseño del procesador; por este motivo, es necesario llegar a un compromiso entre facilitar el diseño del computador y satisfacer las necesidades del programador. Una manera de alcanzar este compromiso es diseñar un juego de instrucciones siguiendo un criterio de ortogonalidad.

La **ortogonalidad** consiste en que todas las instrucciones permitan utilizar como operando cualquiera de los tipos de datos existentes y cualquiera de los modos de direccionamiento, y en el hecho de que la información del código de operación se limite a la operación que debe hacer la instrucción.

Si el juego de instrucciones es ortogonal, será regular y no presentará casos especiales, lo que facilitará la tarea del programador y la construcción de compiladores.

Los aspectos más importantes que hay que tener en cuenta para diseñar un juego de instrucciones son los siguientes:

- **Tipos de datos:** identificación de los tipos de datos necesarios para llevar a cabo las operaciones.
- **Direccionamiento:** identificación de los modos de direccionamiento que se pueden utilizar en los operandos.
- **Registros:** identificación del número de registros del procesador.
- **Conjunto de operaciones:** identificación de las operaciones que hay que llevar a cabo y su complejidad.
- **Formato de las instrucciones:** longitud y número de operandos, y tamaño de los diferentes campos.

◇ TIPOS DE DATOS.

Los operandos de las instrucciones sirven para expresar el lugar donde están los datos que hemos de utilizar. Estos datos se almacenan como una secuencia de bits y, según la interpretación de los valores almacenados, podemos tener tipos de datos diferentes que, generalmente, también nos determinarán el tamaño de los operandos.

En muchos juegos de instrucciones, el tipo de dato que utiliza una instrucción viene determinado por el código de operación de la instrucción. Hemos de tener presente que un mismo dato puede ser tratado como un valor lógico, como un valor numérico o como un carácter, según la operación que se haga con él, lo que no sucede con los lenguajes de alto nivel. A continuación presentamos los tipos generales de datos más habituales:

1) Dirección. Tipo de dato que expresa una dirección de memoria. Para operar con este tipo de dato, puede ser necesario efectuar cálculos y accesos a memoria para obtener la dirección efectiva.

2) Número. Tipo de dato que expresa un valor numérico. Habitualmente distinguimos tres tipos de datos numéricos (y cada uno de estos tipos se puede considerar con signo o sin signo): Números enteros, Números en punto fijo, Números en punto flotante. Como disponemos de un espacio limitado para expresar valores numéricos, tanto la magnitud de los números como su precisión también lo serán, lo que limitará el rango de valores que podemos representar.

3) Carácter. Tipo de dato que expresa un carácter. Habitualmente se utiliza para formar cadenas de caracteres que representarán un texto. Aunque el programador representa los caracteres mediante las letras del alfabeto, para poder representarlos en un computador que utiliza datos binarios será necesaria una codificación. La codificación más habitual es la ASCII, que representa cada carácter con un código de 7 bits, lo que permite obtener 128 caracteres diferentes.

4) Dato lógico. Tipo de dato que expresa un conjunto de valores binarios o booleanos; generalmente cada valor se utiliza como una unidad, pero puede ser interesante tratarlos como cadenas de N bits en las que cada elemento de la cadena es un valor booleano, un bit que puede valer 0 o 1. Este tratamiento es útil cuando se utilizan instrucciones lógicas como AND, OR o XOR.

◇ MODOS DE DIRECCIONAMIENTO.

Los operandos de una instrucción pueden expresar directamente un dato, la dirección, o la referencia a la dirección donde tenemos el dato. Esta dirección puede ser la de un registro o la de una posición de memoria, y en este último caso la denominaremos dirección efectiva. Entendemos por modo de direccionamiento las diferentes maneras de expresar un operando en una instrucción y el procedimiento asociado que permite obtener la dirección donde está almacenado el dato y, como consecuencia, el dato. Los modos de direccionamiento más comunes son:

- **DIRECCIONAMIENTO INMEDIATO:** El dato está dentro de la instrucción y su valor es fijo, por lo que se suele utilizar en operaciones aritméticas o para definir constantes y variables. Como ventaja, no se requiere acceso adicional a memoria para obtener el dato, pero el tamaño del operando está limitado por el tamaño del campo de direccionamiento. Las desventajas principales son que el valor del dato es constante y el rango de valores que se pueden representar está limitado por el tamaño de este operando.
- **DIRECCIONAMIENTO DIRECTO:** El operando indica dónde se encuentra el dato que se quiere utilizar. Si hace referencia a un registro de la máquina, el dato estará almacenado en este registro y hablaremos de direccionamiento directo a registro; si hace referencia a una posición de memoria, el dato estará almacenado en esta dirección de memoria (dirección efectiva) y hablaremos de direccionamiento directo a memoria. Estos modos de direccionamiento tienen una forma muy simple y no hay que hacer cálculos para obtener la dirección efectiva donde está el dato. El tamaño del operando, en el caso del direccionamiento directo a registro, dependerá del número de registros que tenga la máquina; en el direccionamiento directo a memoria, dependerá del tamaño de la memoria.
- **DIRECCIONAMIENTO INDIRECTO:** El operando indica dónde está almacenada la dirección de memoria (dirección efectiva) que contiene el dato que queremos utilizar. Si hace referencia a un registro de la máquina, la dirección de memoria (dirección efectiva) que contiene el dato estará en este registro y hablaremos de direccionamiento indirecto a registro; si hace referencia a una posición de memoria, la dirección de memoria (dirección efectiva) que contiene el dato estará almacenada en esta posición de memoria y hablaremos de direccionamiento indirecto a memoria. La desventaja principal de este modo de direccionamiento es que necesita un acceso más a memoria que el directo. Es decir, un acceso a memoria para el direccionamiento indirecto a registro y dos accesos a memoria para el direccionamiento indirecto a memoria; por este motivo este segundo modo de direccionamiento no se implementa en la mayoría de las máquinas.
- **DIRECCIONAMIENTO RELATIVO:** El operando expresará dos valores, una dirección de memoria y un desplazamiento respecto a esta dirección (salvo los casos en los que uno de los dos sea implícito). La dirección de memoria (dirección efectiva) donde tendremos el dato la obtendremos sumando el desplazamiento a la dirección de memoria. Este modo de direccionamiento es muy potente, ya que no requiere accesos extras a la memoria, como sucede con el indirecto, pero hay que hacer una suma, que no retrasa casi la ejecución de la instrucción, especialmente en las máquinas que tienen una unidad específica para el cálculo de estas direcciones. Desde el punto de vista del programador, es muy útil para acceder a estructuras de datos como vectores, matrices o listas, ya que se aprovecha la localidad de los datos, dado que la mayoría de las referencias en la memoria son muy próximas entre sí. Existen tres tipos de direccionamiento relativo:
 - Direccionamiento relativo a registro base: La dirección de memoria se almacena en el *registro base* (RB) y el desplazamiento se encuentra explícitamente en la instrucción. En algunas instrucciones el RB se utiliza de manera implícita y solo habrá que especificar explícitamente el desplazamiento.
 - Direccionamiento relativo a registro índice: La dirección de memoria se encuentra explícitamente en la instrucción y el desplazamiento se almacena en el registro que denominaremos *registro índice* (RI).
 - Direccionamiento relativo a PC: Es equivalente al relativo a registro base, con la diferencia de que utiliza el registro contador de programa (PC) de manera implícita en la instrucción y solo hay que expresar, mediante una etiqueta, el desplazamiento para calcular la dirección de memoria (dirección efectiva) a la que se quiere acceder.
- **DIRECCIONAMIENTO IMPLICITO:** La instrucción no contiene información sobre la localización del operando porque este se encuentra en un lugar predeterminado, y queda especificado de manera implícita en el CODOP.
 - Direccionamiento a pila: Se trabaja implícitamente con la cima de la pila por medio de registros de la máquina; habitualmente uno de estos registros se conoce como stack pointer (SP) y se utiliza para apuntar a la cima de la pila. Como es un modo de direccionamiento implícito, solo se utiliza en instrucciones determinadas, las más habituales de las cuales son PUSH (poner un elemento en la pila) y POP (sacar un elemento de la pila).

• 4.7 Organización de los Registros Internos de la CPU.

Los registros son, básicamente, elementos de memoria de acceso rápido que se encuentran dentro del procesador. Constituyen un espacio de trabajo para el procesador y se utilizan como un espacio de almacenamiento temporal. Se implementan utilizando elementos de memoria RAM estática (static RAM). Son imprescindibles para ejecutar las instrucciones, entre otros motivos, porque la ALU solo trabaja con los registros internos del procesador.

El conjunto de registros y la organización que tienen cambia de un procesador a otro; los procesadores difieren en el número de registros, en el tipo de registros y en el tamaño de cada registro. Una parte de los registros pueden ser visibles para el programador de aplicaciones, otra parte solo para instrucciones privilegiadas y otra solo se utiliza en el funcionamiento interno del procesador. Una posible clasificación de los registros del procesador es la siguiente: Registros de propósito general, Registros de instrucción, Registros de acceso a memoria, Registros de estado y de control.

◇ REGISTROS DE PROPOSITO GENERAL.

Los **registros de propósito general** son los registros que suelen utilizarse como operandos en las instrucciones del ensamblador. Estos registros se pueden asignar a funciones concretas: datos o direccionamiento. En algunos procesadores todos los registros se pueden utilizar para todas las funciones.

Los **registros de datos** se pueden diferenciar por el formato y el tamaño de los datos que almacenan; por ejemplo, puede haber registros para números enteros y para números en punto flotante.

Los **registros de direccionamiento** se utilizan para acceder a memoria y pueden almacenar direcciones o índices. Algunos de estos registros se utilizan de manera implícita para diferentes funciones, como por ejemplo acceder a la pila, dirigir segmentos de memoria o hacer de soporte en la memoria virtual.

◇ REGISTROS DE INSTRUCCIÓN.

Los dos registros principales relacionados con el acceso a las instrucciones son:

- **Program Counter (PC):** contiene la dirección de la siguiente instrucción a leer de la memoria.
- **Instruction Register (IR):** registro de instrucción, contiene la instrucción que hay que ejecutar.

◇ REGISTROS DE ACCESO A MEMORIA.

Hay dos registros necesarios para cualquier operación de lectura o escritura de memoria:

- **Memory Address Register (MAR):** contiene la dirección de memoria a la que queremos acceder.
- **Memory Buffer Register (MBR):** contiene la palabra leída más recientemente.

◇ REGISTROS DE ESTADO Y DE CONTROL.

La información sobre el estado del procesador puede estar almacenada en un registro o en más de uno, aunque habitualmente suele ser un único registro denominado registro de estado. Los bits del registro de estado son modificados por el procesador como resultado de la ejecución de algunos tipos de instrucciones o como consecuencia de algún acontecimiento, como las peticiones de interrupción.

Estos bits son parcialmente visibles para el programador, en algunos casos mediante la ejecución de instrucciones específicas. Cada bit o conjunto de bits del registro de estado indica una información concreta. Los más habituales son:

- **Bit de signo:** contiene el bit de signo del resultado de la última operación aritmética.
- **Bit de cero:** puesto a 1 cuando el resultado es cero.
- **Bit de acarreo:** puesto a 1 si una operación da lugar a un acarreo.
- **Bit de igualdad:** puesto a 1 si el resultado de una comparación lógica es la igualdad.
- **Bit de desbordamiento:** usado para indicar un desbordamiento aritmético.
- **Interrupciones habilitadas/inhabilitadas:** usado para permitir o inhabilitar interrupciones.
- **Supervisor:** indica si la CPU funciona en modo supervisor o usuario.

• 4.8 Little-Endian & Big-Endian.

En la mayoría de los computadores la memoria se dirige en bytes, es decir, el tamaño de la palabra de memoria es de un byte. Cuando trabajamos con un dato formado por varios bytes habrá que decidir cómo se almacena el dato dentro de la memoria, es decir, qué byte del dato se almacena en cada posición de la memoria. Se pueden utilizar dos sistemas diferentes:

- **Little-Endian:** almacenar el byte de menos peso del dato en la dirección de memoria más baja.
- **Big-Endian:** almacenar el byte de más peso del dato en la dirección de memoria más baja.

Una vez elegido uno de estos sistemas, habrá que tenerlo presente y utilizarlo en todos los accesos a memoria (lecturas y escrituras) para asegurar la coherencia de los datos.

Ejemplo: Supongamos que queremos almacenar el valor hexadecimal siguiente: 12345678h. Se trata de un valor de 32 bits, formado por los 4 bytes 12h 34h 56h y 78h. Supongamos también que se quiere almacenar en la memoria a partir de la dirección 200h. Como cada posición de la memoria permite almacenar un solo byte, necesitaremos 4 posiciones de memoria, correspondientes a las direcciones 200h, 201h, 202h y 203h.

Little-Endian		Big-Endian	
Dirección	Contenido	Dirección	Contenido
200h	78h	200h	12h
201h	56h	201h	34h
202h	34h	202h	56h
203h	12h	203h	78h

• 4.9 Estructuras de interconexión.

Al conjunto de líneas que conectan los módulos de un computador (procesador, memoria y E/S) se lo denomina **Estructura de Interconexión**. El diseño de dicha estructura dependerá de los intercambios que deban producirse entre los módulos. Los tipos de intercambio son:

- **Memoria:** un módulo de memoria está constituido por N palabras de la misma longitud. A cada palabra se le asigna una única dirección numérica (de 0 a N-1). Una palabra de datos puede leerse (Read) o escribirse (Write) en la memoria. La posición de memoria para la operación se especifica mediante una dirección.
- **Módulo de E/S:** la E/S es funcionalmente similar a la memoria. Hay dos tipos de operaciones: leer y escribir. Además, un módulo puede controlar más de un dispositivo externo (puerto), al cual se le asigna una dirección a cada uno. Existen líneas de datos externas para la entrada y la salida de datos por un dispositivo externo. Por último, un módulo de E/S puede enviar señales de interrupción al procesador.
- **Procesador:** lee instrucciones y datos, escribe datos una vez que los ha procesado, y usa ciertas señales para controlar el funcionamiento del sistema. También puede recibir señales de interrupción.

La estructura de interconexión debe dar cobertura a los siguientes tipos de transferencia:

- 1) **Memoria a Procesador:** el procesador lee una instrucción/dato desde la memoria.
- 2) **Procesador a Memoria:** el procesador escribe un dato en la memoria.
- 3) **E/S a Procesador:** el procesador envía datos al dispositivo de E/S.
- 4) **Memoria a E/S y Viceversa:** el módulo de E/S intercambia datos con la memoria, utilizando DMA.

♦ ESTRUCTURA DEL BUS.

El bus que conecta los componentes principales de la computadora (procesador, memoria y E/S) se denomina bus del sistema. El bus del sistema está constituido por entre 50 y 100 líneas. A cada línea se le asigna una función particular. Se pueden clasificar en tres grupos principales:

1) **Líneas de Datos:** proporcionan un camino para transmitir datos entre los módulos del sistema. El conjunto constituido por estas líneas, se denomina bus de datos. Consta generalmente de 8, 16 o 32 líneas distintas (anchura del bus). Puesto que cada línea solo puede transportar un bit cada vez, el ancho del bus determina cuantos bits se pueden transferir al mismo tiempo.

2) **Líneas de Dirección:** se usan para designar la fuente o el destino del dato situado en el bus de datos. La anchura del bus de direcciones determina la máxima capacidad de memoria posible. Además, las líneas de direcciones se utilizan también para direccionar a los puertos de E/S.

3) Líneas de Control: se utilizan para controlar el acceso y el uso de las líneas de datos y de direcciones, ya que son compartidas. Las señales de control transmiten tanto ordenes como información de temporización entre los módulos del sistema. Algunas líneas de control típicas son:

- Escritura en memoria: hace que el dato del bus se escriba en la posición direccionada.
- Lectura de memoria: hace que el dato de la posición direccionada se sitúe en el bus.
- Escritura de E/S: hace que el dato del bus se transfiera a través del puerto de E/S direccionado.
- Lectura de E/S: hace que el dato del puerto de E/S direccionado se sitúe en el bus.
- Transferencia reconocida: indica que el dato se ha aceptado o situado en el bus.
- Petición de bus: indica que un módulo necesita disponer del control del bus.
- Cesión de bus: indica que se cede el control del bus a un módulo que lo había solicitado.
- Petición de interrupción: indica si hay una interrupción pendiente.
- Interrupción reconocida: señala que la interrupción pendiente se ha aceptado.
- Reloj: se utiliza para sincronizar operaciones.
- Inicio: pone los módulos conectados en su estado inicial.

◇ JERARQUIA DE BUSES.

Si se conecta un gran número de dispositivos al bus, las prestaciones pueden disminuir por dos causas principales:

A) A más dispositivos conectados al bus, mayor es el retardo de propagación, el cual determina el tiempo que necesitan los dispositivos para coordinarse en el uso del bus.

B) El bus puede convertirse en un cuello de botella a medida que las peticiones de transferencia acumuladas se aproximan a la capacidad del bus. Este problema se puede resolver aumentando la velocidad en la que el bus puede transferir los datos y utilizando buses más anchos (por ejemplo, incrementar bus de datos de 32 a 64 bits).

Por consiguiente, se usan varios buses organizados **jerárquicamente**. En una estructura típica, hay un bus local que conecta el procesador a una memoria caché, y al que pueden conectarse también uno o más dispositivos locales. El controlador de caché conecta la caché no sólo al bus local, sino también al de sistema, donde se conectan todos los módulos de memoria principal. Los controladores de E/S se conectan a un bus de expansión. La interfaz del bus de expansión regula las transferencias de datos entre el bus del sistema y los controladores conectados al bus de expansión, lo que aísla el tráfico de información con el de la memoria y el procesador.

◇ ELEMENTOS DE DISEÑO DE UN BUS.

- **Tipos de buses:** Las líneas del bus se pueden dividir en dos tipos genéricos: dedicadas y multiplexadas.
 - A) Líneas de bus dedicada: está permanentemente asignada a una función o a un subconjunto físico de componentes del computador.
 - B) Líneas de bus multiplexada: las mismas líneas se pueden usar para diferentes operaciones. La ventaja es que se ahorra espacio y costo, pero necesita una circuitería más compleja.
- **Métodos de arbitraje:** Puesto que en un instante dado, solo una unidad puede transmitir a través del bus, se requiere algún método de arbitraje que controle las operaciones. Se clasifican en: centralizados o distribuidos.
 - A) Centralizados: un único dispositivo de hardware, denominado controlador del bus o árbitro, es responsable de asignar tiempos en el bus.
 - B) Distribuidos: cada módulo dispone de lógica para controlar el acceso y los módulos actúan conjuntamente para compartir el bus.
- **Temporización:** Hace referencia a la forma en la que se coordinan los eventos en el bus.
 - A) Síncrona: la presencia de un evento en el bus está determinada por un reloj. El bus incluye una línea de reloj a través de la que se transmite una secuencia de intervalos regulares de 1 a 0, conocida como ciclo de reloj. Todos los dispositivos pueden leer esta línea y los eventos comienzan al principio de un ciclo de reloj.
 - B) Asíncrona: la presencia de un evento en el bus es consecuencia de que se produzca un evento previo.
- **Anchura del bus:** La anchura del bus de datos afecta a las prestaciones del sistema. Cuanto más ancho es el bus de datos, mayor es el número de bits que se transmiten a la vez. Cuanto más ancho es el bus de direcciones, mayor es el rango de posiciones a las que se puede hacer referencia.
- **Tipos de transferencia de datos:** Todos los buses permiten tanto transferencias de lecturas como de escrituras. En ciertos buses también son posibles algunas operaciones combinadas, como lectura-modificación-escritura o lectura-después de escritura.

◇ BUS PCI.

El **bus PCI** es un bus de ancho de banda elevado, independiente del procesador, que se puede usar como bus de periféricos o bus para una arquitectura de entreplanta. Está diseñado para permitir una cierta variedad de configuraciones basadas en microprocesadores, incluyendo sistemas tanto de uno como de varios procesadores. Por consiguiente, proporciona un conjunto de funciones de uso general. Utiliza temporización síncrona y un esquema de arbitraje centralizado.

• 4.10 Descripción de procesadores actuales.

Para mediados de 1980, una nueva idea llamada RISC comenzó a dominar en la industria del diseño de computadoras, reemplazando arquitecturas complejas (CISC) por otras más sencillas y rápidas. En la década de 1990 comenzaron a aparecer CPU superescalares que pueden ejecutar varias instrucciones en paralelo, a menudo en un orden distinto del que tenían en el programa. Entre las técnicas incorporadas a los procesadores de hoy en día están:

- **Predicción de ramificación:** el procesador se anticipa al software y predice qué grupo de instrucciones se van a procesar después con mayor probabilidad. Si el procesador acierta la mayoría de las veces, puede precaptar las instrucciones correctas y almacenarlas para mantener al procesador ocupado.
- **Análisis de flujo de datos:** el procesador analiza qué instrucciones dependen de los resultados de otras instrucciones o datos, para crear una organización optimizada de instrucciones. Las instrucciones se regulan para ser ejecutadas cuando estén listas, independientemente del orden original del programa. Esto evita retrasos innecesarios.
- **Ejecución especulativa:** utilizando la predicción de ramificación y el análisis de flujo de datos, algunos procesadores ejecutan especulativamente instrucciones antes de que aparezcan en la ejecución del programa, manteniendo los resultados en posiciones temporales. Esto permite al procesador mantener sus máquinas de ejecución lo más ocupadas posible, ejecutando instrucciones que es probable que se necesiten.

♦ PRINCIPIOS DE DISEÑO PARA COMPUTADORAS MODERNAS.

1) Todas las instrucciones se ejecutan en hardware, éstas no se interpretan con microinstrucciones. La eliminación de un nivel de interpretación hace que la mayor parte de las instrucciones sean más rápidas.

2) Maximizar el ritmo con que se emiten instrucciones. Este principio sugiere que el paralelismo puede desempeñar un papel importante en el mejoramiento del desempeño.

3) Las instrucciones deben ser fáciles de decodificar. Todo lo que pueda agilizar el proceso de decodificación es útil, como por ejemplo hacer que las instrucciones tengan una longitud fija, con un número pequeño de campos. Cuando menor sea el número de formatos de instrucciones distintos, mejor.

4) Sólo las operaciones de carga y almacenamiento deben hacer referencias a memoria. Ya que el acceso a memoria puede tardar mucho, la mejor manera de evitar retrasos es que solo las instrucciones LOAD y STORE hagan referencia a memoria.

5) Incluir abundantes registros. Puesto que el acceso a memoria es bastante lento, es necesario contar con muchos registros (32 al menos) para que, una vez que se ha obtenido una palabra, se pueda mantener en un registro hasta que ya no se necesite. Quedarse sin registros y tener que escribirlos en la memoria solo para volver a cargarlos después es poco recomendable y debe evitarse preferentemente. La mejor manera de lograr esto es tener suficientes registros.

♦ EJEMPLOS DE PROCESADORES MODERNOS.

- **Pentium:** Con este procesador, en 1993, Intel introduce el uso de técnicas superescalares. El Pentium es un microprocesador con arquitectura x86.
- **Pentium II:** Fue introducido en el mercado en 1997. El Intel Pentium II es un descendiente directo de la CPU 8088 que se usó en la IBM PC original, y es totalmente compatible con el 8088 y con todos los procesadores intermedios. Posee 7,5 millones de transistores. Los cambios fundamentales respecto a éste último fueron mejorar el rendimiento en la ejecución de código de 16 bits, añadir el conjunto de instrucciones MMX (extensiones multimedia especiales) y eliminar la memoria caché de segundo nivel del núcleo del procesador, colocándola en una tarjeta de circuito impreso junto a éste.
- **Intel Celeron:** Era una línea de productos de bajo costo y desempeño de los Pentium II. Fue diseñada con la misma arquitectura que los Pentium II, con la idea de insertarla en los extremos inferiores del mercado. Esto sucedió en 1998.
- **Intel Xeon:** También en 1998, Intel introdujo una versión especial del Pentium II para los extremos superiores del mercado, el procesador Xeon, con una caché mayor, un bus más rápido y mejor apoyo multiprocesador.
- **Pentium D:** Estos procesadores fueron introducidos por Intel en el 2005. Un chip Pentium D consiste básicamente en 2 procesadores Pentium 4 metidos en un solo encapsulado.
- **Intel Core 2 Duo:** marcó la desaparición de los microprocesadores de la familia Intel Pentium; para dar paso a una nueva generación tecnológica en microprocesadores. Basada en la revolucionaria microarquitectura Intel® Core™, esta familia se diseñó para ofrecer un potente rendimiento con ahorro energético para que pueda hacer más al mismo tiempo sin ralentizar su marcha.

UNIDAD 05 – MEMORIA

Todo computador necesita un sistema de **memoria** para almacenar los programas que se ejecutan y los datos necesarios para ejecutar estos programas. La cantidad de memoria que puede tener un computador responde básicamente a un factor de coste: cuanto más memoria instalada, más elevado es el coste. De manera parecida, la velocidad de la memoria también depende del coste. Las memorias más rápidas tienen un coste más elevado, pero no se puede conseguir toda la velocidad necesaria simplemente incrementando el coste; hay además un factor tecnológico que limita la velocidad de la memoria: no podemos adquirir memoria más rápida que la que está disponible en el mercado en un momento dado.

Existen diferentes tipos de memorias, con capacidades y tiempos de acceso diferentes. En general, cuanto más capacidad de almacenamiento tiene una memoria, mayor es el tiempo de acceso. Es decir, las memorias con gran capacidad son memorias lentas, mientras que las memorias rápidas (tiempo de acceso pequeño) suelen tener poca capacidad de almacenamiento. Las memorias rápidas son más caras que las memorias lentas. Por ello, los diseñadores de computadores deben llegar a un compromiso a la hora de decidir cuánta memoria ponen en sus diseños y de qué velocidad o tiempo de acceso.

● 5.1 Características de los sistemas de memoria.

◇ UBICACIÓN.

Indica si la memoria es interna o externa al computador. La **memoria interna** suele clasificarse como memoria principal, sin embargo, existen otras formas de memoria interna. El procesador y la Unidad de Control necesitan su propia memoria local en forma de registros y uno o varios niveles de memoria caché. La **memoria externa** corresponde a los dispositivos de almacenamiento secundario, como discos duros, unidades ópticas (CD-ROM, DVD, o BluRay), unidades de cinta, etc., que son accesibles por la CPU a través de los controladores de E/S.

◇ CAPACIDAD.

La **capacidad** (o tamaño de la memoria) hace referencia a la cantidad de información que se puede almacenar. La unidad utilizada para especificar la capacidad de almacenamiento de información es el byte (1 byte = 8 bits) o la palabra (8, 16 y 32 bits), y a la hora de indicar la capacidad, se utilizan diferentes prefijos que representan múltiplos del byte. En informática, la capacidad de almacenamiento habitualmente se indica en múltiplos que sean potencias de dos; en este caso se utilizan los prefijos definidos por la International Electrotechnical Commission (IEC).

- 2^{10} bytes = 1.024 bytes = 1 KiB (kibibyte)
- 2^{20} bytes = 1.024 KiB = 1 MiB (mebibyte)
- 2^{30} bytes = 1.024 MiB = 1 GiB (gibibyte)
- 2^{40} bytes = 1.024 GiB = 1 TiB (tebibyte)

◇ METODOS DE ACCESO.

Cada tipo de memoria utiliza un método a la hora de acceder a las posiciones de memoria. Hay métodos de acceso diferentes característicos de cada tipo de memoria:

- 1) **Secuencial:** Se accede desde la última posición a la que se ha accedido, leyendo en orden todas las posiciones de memoria hasta llegar a la posición deseada. El tiempo de acceso depende de la posición a la que se quiere acceder y de la posición a la que se ha accedido anteriormente.
- 2) **Directo.** La memoria se organiza en bloques y cada bloque de memoria tiene una dirección única, se accede directamente al principio de un bloque y dentro de este se hace un acceso secuencial hasta llegar a la posición de memoria deseada. El tiempo de acceso depende de la posición a la que se quiere acceder y de la última posición a la que se ha accedido.
- 3) **Aleatorio.** Cada posición direccionable de memoria tiene un único mecanismo de acceso, cableado físicamente. El tiempo para acceder a una posición dada es constante e independiente de la secuencia de accesos previos. Por tanto, cualquier posición puede seleccionarse aleatoriamente y puede ser direccionada y accedida directamente.
- 4) **Asociativo.** Se trata de un tipo de memoria de acceso aleatorio donde el acceso se hace basándose en el contenido y no en la dirección. Se especifica el valor que se quiere localizar y se compara este valor con una parte del contenido de cada posición de memoria; la comparación se lleva a cabo simultáneamente con todas las posiciones de la memoria.

◇ ORGANIZACIÓN DE LOS DATOS (UNIDAD DE TRANSFERENCIA).

1) Palabra de memoria: Es la unidad de organización de la memoria desde el punto de vista del procesador; el tamaño de la palabra de memoria se especifica en bytes o bits y suele coincidir con el número de bits utilizados para representar números y con la longitud de las instrucciones. La palabra de memoria es el número de bytes máximo que se pueden leer o escribir en un solo ciclo de acceso a la memoria.

2) Unidad de direccionamiento: La memoria interna se puede ver como un vector de elementos, una colección de datos contiguos, en la que cada dato es accesible indicando su posición o dirección dentro del vector. La unidad de direccionamiento especifica cuál es el tamaño de cada elemento; habitualmente a la memoria se accede como un vector de bytes –cada byte tendrá su dirección–, aunque puede haber sistemas que accedan a la memoria como un vector de palabras, en los que cada dirección corresponda a una palabra. El número de bits utilizados para especificar una dirección de memoria fija el límite máximo de elementos dirigibles, el tamaño del mapa de memoria; si tenemos n bits para las direcciones de memoria, el número máximo de elementos dirigibles será de 2^n .

3) Unidad de transferencia: En un acceso a memoria se puede acceder a un byte o a varios, con un máximo que vendrá determinado por el número de bytes de una palabra de memoria; es decir, en un solo acceso se leen o escriben uno o varios bytes. Cuando se especifica la dirección de memoria a la que se quiere acceder, se accede a partir de esta dirección a tantos bytes como indique la operación de lectura o escritura. En memoria externa, se accede habitualmente a un bloque de datos de tamaño muy superior a una palabra. En discos es habitual transferir bloques del orden de los Kbytes.

◇ TIEMPO DE ACCESO Y VELOCIDAD (PRESTACIONES).

En memorias de acceso aleatorio, memoria RAM, el **tiempo de acceso** (o **latencia**) es el tiempo que transcurre desde que una dirección de memoria es visible para los circuitos de la memoria hasta que el dato está almacenado (escritura) o está disponible para ser utilizado (lectura). En memorias de acceso no aleatorio (discos) es el tiempo necesario para que el mecanismo de lectura o escritura se sitúe en la posición necesaria para empezar la lectura o escritura. En memorias de acceso aleatorio, el tiempo de un **ciclo de memoria** se considera el tiempo de acceso más el tiempo necesario antes de que se pueda empezar un segundo acceso a la memoria.

La **velocidad de transferencia** es la velocidad a la que se puede leer o escribir un dato de memoria o la velocidad en la que se puede transferir un dato desde y hacia memoria. En las memorias de acceso aleatorio será el inverso del tiempo de ciclo.

◇ COSTE.

Consideramos el **coste** por unidad de almacenamiento (coste por bit). Podemos observar que existe una relación directamente proporcional entre la velocidad y el coste/bit: a medida que aumenta la velocidad aumenta también el coste/bit. Eso implica que, con un presupuesto fijado, podremos adquirir memorias muy rápidas pero relativamente pequeñas, o memorias más lentas, pero de mucha más capacidad.

◇ CARACTERÍSTICAS FÍSICAS.

La memoria se puede clasificar según **características físicas** diferentes; básicamente podemos distinguir dos clasificaciones. La primera distingue entre:

- **Memoria volátil:** memoria que necesita una corriente eléctrica para mantener su estado; estas memorias incluyen registros, memoria caché y memoria principal.
- **Memoria no volátil:** mantiene el estado sin necesidad de corriente eléctrica, incluye memorias de solo lectura, memorias programables, memoria flash, dispositivos de almacenamiento magnético y óptico.

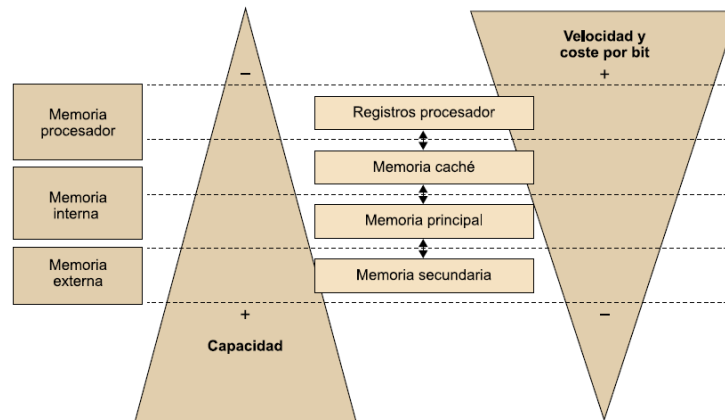
La segunda clasificación distingue entre:

- **Memoria de semiconductores:** es una memoria que utiliza elementos semiconductores, transistores, en su construcción; incluye: registros, memoria caché, memoria principal, memorias de solo lectura, memoria flash.
- **Memoria magnética:** utiliza superficies imantadas para guardar la información; dentro de esta categoría se incluyen básicamente discos y cintas magnéticas.
- **Memoria óptica:** utiliza elementos de almacenamiento que pueden ser leídos y escritos mediante luz láser; se incluyen dispositivos de CD, DVD, BluRay.

• 5.2 Jerarquía de memoria.

El objetivo en el diseño del sistema de memoria, es obtener gran capacidad y un tiempo de acceso reducido con el precio más bajo posible. Como no existe ninguna tecnología que cumple simultáneamente estos requisitos, la memoria de una computadora se estructura en varios niveles con el objetivo de conseguir mejores prestaciones, y forma lo que se denomina Jerarquía de Memoria.

En una **Jerarquía de Memoria** se utilizan varios tipos de memorias, con distintas características de capacidad, velocidad y coste, divididas en diferentes niveles (Memoria del procesador, memoria interna y memoria externa). Cada nivel de la Jerarquía se caracteriza también por la distancia a la que se encuentra del procesador. Los primeros niveles son los más próximos y esto es así porque también son los niveles con velocidad más elevada.



Existe un compromiso entre las tres características claves (costo, capacidad y tiempo de acceso):

- A menor tiempo de acceso, mayor coste por bit.
- A mayor capacidad, menor coste por bit.
- A mayor capacidad, mayor tiempo de acceso.

Cuando se desciende en la jerarquía, ocurre que:

- Disminuye el coste por bit.
- Aumenta la capacidad.
- Aumenta el tiempo de acceso.
- Disminuye la frecuencia de acceso a la memoria por parte del procesador.

El objetivo final de la jerarquía de memorias es conseguir que, cuando el procesador acceda a un dato, este se encuentre en el nivel más rápido de la jerarquía. Obtenemos así una memoria a un coste moderado, con una velocidad próxima a la del nivel más rápido y la capacidad del nivel más alto.

Los dos factores básicos que provocan que el esquema de jerarquía de memorias funcione satisfactoriamente en un computador son los siguientes: El flujo de datos entre los niveles de la jerarquía de memorias se puede hacer en paralelo con el funcionamiento normal del procesador y el principio de localidad de referencias.

El código de los programas se organiza en subrutinas, tiene estructuras iterativas y trabaja con conjuntos de datos agrupados. Esto, unido al hecho de que la ejecución del código es secuencial, lleva a que durante un intervalo de tiempo determinado se utilice solo una pequeña parte de toda la información almacenada: este fenómeno se denomina principio de localidad de referencias. A causa de esta característica, se ha probado empíricamente que aproximadamente el 90% de todas las instrucciones ejecutadas corresponden al 10% del código de un programa.

- **Localidad Temporal:** Si en un momento una posición de memoria particular es referenciada, entonces es muy probable que la misma ubicación vuelva a ser referenciada en un futuro cercano. En este caso es común almacenar una copia de los datos referenciados en caché para lograr un acceso más rápido a ellos.

- **Localidad Espacial:** Si una localización de memoria es referenciada en un momento concreto, es probable que las localizaciones cercanas a ella sean también referenciadas pronto. En este caso es común estimar las posiciones cercanas para que estas tengan un acceso más rápido.

- **Localidad Secuencial:** Las direcciones de memoria que se están utilizando suelen ser contiguas. Esto ocurre porque las instrucciones se ejecutan secuencialmente. Para obtener beneficios de la gran frecuencia con la que ocurren casos de localidad espacial o temporal, muchos sistemas de memoria utilizan una jerarquía de niveles de memoria.

● 5.3 Organización de la memoria en una computadora.

La memoria en una computadora moderna está formada típicamente por cuatro niveles fundamentales:

◇ REGISTROS.

El **registro** es el espacio de memoria que se encuentra dentro del procesador, integrado dentro del mismo chip de este. Se utilizan celdas de memoria de tipo estático, SRAM, para su implementación. Es el espacio de memoria en el cual el procesador puede acceder más rápidamente a los datos. Este espacio de memoria es accesible al programador de lenguaje de ensamblador y, si se gestiona bien, permite minimizar el número de accesos a la memoria interna, que son bastante más lentos.

◇ MEMORIA PRINCIPAL (MEMORIA INTERNA).

En la **memoria principal** se almacenan los programas que se deben ejecutar y sus datos, es la memoria visible para el programador mediante su espacio de direcciones. La memoria principal se implementa utilizando diferentes chips conectados a la placa principal del computador y tiene una capacidad mucho más elevada que la memoria caché (del orden de Gbytes o de Tbytes en supercomputadores). Utiliza tecnología DRAM (Dynamic RAM), que es más lenta que la SRAM, pero con una capacidad de integración mucho más elevada, hecho que permite obtener más capacidad en menos espacio.

◇ MEMORIA SECUNDARIA (MEMORIA EXTERNA).

La **memoria secundaria** corresponde a dispositivos de almacenamiento secundario: discos magnéticos, cintas magnéticas, discos ópticos, dispositivos de memoria flash, etc., y también se pueden considerar sistemas de almacenamiento en red. Estos dispositivos son gestionados por el sistema de ficheros del sistema operativo mediante el sistema de entrada/salida. Los dispositivos que forman la memoria externa se conectan al computador con algún tipo de bus (serie o paralelo). Estos dispositivos se pueden encontrar físicamente dentro del computador conectados por buses internos del computador (IDE, SATA, SCSI, etc.) o pueden estar fuera del computador conectados por buses externos (USB, Firewire, eSATA, Infiniband, etc.).

◇ MEMORIA VIRTUAL.

Decimos que un computador utiliza **memoria virtual** cuando las direcciones de memoria de los programas se refieren a un espacio de memoria superior al espacio de memoria físico, espacio de memoria principal. La memoria virtual libera al programador de las restricciones de la memoria principal. En estos computadores diferenciamos entre el mapa de direcciones lógicas o virtuales (las direcciones que utilizan los programas) y el mapa de direcciones físicas o reales (las direcciones de la memoria principal). El espacio de memoria virtual utiliza como soporte un dispositivo de almacenamiento externo (habitualmente un disco magnético), mientras que el espacio de memoria físico se corresponde con la memoria principal del computador.

● 5.4 Memoria Caché.

La **Memoria Caché** se sitúa entre la memoria principal y el procesador, puede estar formada por uno o varios niveles. Tiene un tiempo de acceso inferior al de la memoria principal con el objetivo de reducir el tiempo de acceso medio a los datos, pero también tiene un tamaño mucho más reducido que la memoria principal. Si un dato está en la memoria caché, es posible proporcionarlo al procesador sin acceder a la memoria principal, si no, primero se lleva el dato de la memoria principal a la memoria caché y después se proporciona el dato al procesador. Debido al fenómeno de localidad de las referencias, cuando un bloque de datos es captado por la cache para satisfacer una referencia a memoria simple, es probable que se hagan referencias futuras a otras palabras del mismo bloque.

Para trabajar con memoria caché, la memoria principal se organiza en bloques de palabras, de manera que cuando hay que trasladar datos de la memoria principal a la memoria caché se lleva un bloque entero de palabras de memoria. La memoria caché también se organiza en bloques que se denominan líneas. Cada línea está formada por un conjunto de palabras (el mismo número de palabras que tenga un bloque de memoria principal), más una etiqueta compuesta por unos cuantos bits. El contenido de la etiqueta permitirá saber qué bloque de la memoria principal se encuentra en cada línea de la memoria caché en un momento dado.

Cada vez que el procesador quiere acceder a una palabra de memoria, primero se accede a la memoria caché; si la palabra de memoria se encuentra almacenada en la memoria caché, se proporciona al procesador y diremos que se ha producido un **acierto**. En caso contrario, se lleva el bloque de datos de la memoria principal que contiene la palabra de memoria hacia la memoria caché y, cuando la palabra ya está en la memoria caché, se proporciona al procesador; en este caso diremos que se ha producido un **fallo**.

◇ ELEMENTOS DE DISEÑO DE LA MEMORIA CACHE.

- **Tamaño:** nos gustaría que el tamaño fuera lo suficientemente pequeño para que el coste total medio por bit se acerque al de la memoria principal, que fuera lo suficientemente grande como para que el tiempo de acceso medio total se acerque al de la caché. Cuánto más grande es, mayor es el número de puertas que direccionan la caché. Problemas: las cachés grandes tienden a ser lentas. Los tamaños óptimos sugeridos de caché se encuentran entre 1K y 512K palabras.
- **Función de correspondencia:** al haber menos líneas de caché que bloques de memoria principal, se necesita de un algoritmo que haga corresponderlos y de algún medio que determine qué bloque ocupa actualmente una línea de caché. Existen 3 técnicas de correspondencia:
 - Directa: simple y poco costosa de implementar; consiste en hacer corresponder cada bloque de memoria principal a sólo una línea posible de caché, lo que sería una desventaja ya que, si un programa referencia repetidas veces a palabras de dos bloques diferentes asignados en la misma línea, dichos bloques se estarían intercambiando continuamente en la caché, y la tasa de aciertos sería baja.
 - Asociativa: supera la desventaja de la anterior: cada bloque de memoria principal puede cargarse en cualquier línea de la caché. La lógica de control de la caché interpreta una dirección de memoria como una etiqueta y un campo (identifica unívocamente un bloque de memoria) de palabra; para saber si un bloque está en la caché, la lógica de control examina simultáneamente todas las etiquetas de líneas para buscar una coincidencia. Esta correspondencia genera flexibilidad para que cualquier bloque sea reemplazado cuando se va a escribir uno nuevo en la caché (ver 'Algoritmos de sustitución'), lo cual hace a la maximización de la tasa de aciertos. Desventaja: compleja circuitería necesaria para examinar en paralelo las etiquetas de las líneas de caché.
 - Asociativa por conjuntos: solución de compromiso que recoge lo positivo de las correspondencias directa y asociativa, sin sus desventajas.
- **Algoritmos de sustitución:** cuando se introduce un nuevo bloque en la caché, debe sustituirse uno de los bloques existentes (para la correspondencia asociativa, ya que en la directa no es posible). Estos algoritmos deben implementarse en hardware para conseguir una alta velocidad. Cuatro de los algoritmos probados son:
 - LRU (Last-Recent Used, Utilizado menos recientemente): se sustituye el bloque que se ha mantenido en la caché por más tiempo sin haber sido referenciado.
 - FIFO (First-In-First-Out, Primero en entrar-primero en salir): se sustituye aquel bloque del conjunto que ha estado más tiempo en la caché; puede implementarse mediante una técnica cíclica o buffer circular.
 - LFU (Least-Frequently Used, Utilizado menos frecuentemente): se sustituye aquel bloque del conjunto que ha experimentado menos referencias; podría implementarse asociando un contador a cada línea.
 - Técnica no basada en el grado de utilización: toma una línea al azar entre las posibles candidatas; proporciona prestaciones sólo ligeramente inferiores a un algoritmo basado en la utilización (LRU, FIFO, LFU).
- **Política de escritura:** antes de que pueda ser reemplazado un bloque que está en una línea de caché, es necesario saber si ha sido alterado en caché pero no en memoria principal. Si no ha sido alterado puede escribirse sobre la línea de caché; sino, la memoria principal debe actualizarse. Problemas: más de un dispositivo puede tener acceso a la memoria principal (se pueden generar palabras no válidas); y, cuando varios procesadores se conectan al mismo bus y cada uno de ellos tiene su propia caché local, al modificarse una palabra de alguna de las cachés, podría invalidar una palabra de las otras. Técnicas de escritura:
 - Escritura inmediata: la más sencilla; todas las operaciones de escritura se hacen, tanto en caché como en memoria principal, asegurando que el contenido de la memoria principal siempre es válido. Desventaja: genera tráfico sustancial a memoria.
 - Post-escritura: técnica alternativa que minimiza las escrituras en memoria, con la cual las actualizaciones se hacen sólo en caché, ya que cuando el bloque es sustituido, es escrito en memoria principal sí, y sólo sí, las actualizaciones se han realizado. Problema: a veces hay porciones de memoria no válidas, entonces los accesos por parte de los módulos E/S sólo podrían hacerse a través de la caché, lo que complica la circuitería y genera tráfico.
- **Tamaño de línea:** cuando se recupera y ubica en caché un bloque de datos, se recuperan la palabra y algunas palabras adyacentes. A medida que aumenta el tamaño de bloque, la tasa de aciertos primero aumenta debido al principio de localidad y más datos útiles son llevados a la caché. Sin embargo, la tasa de aciertos comenzará a decrecer cuando el tamaño de bloque aumente más y la probabilidad de utilizar la nueva información captada se haga menor que la de reutilizar la información que tiene que reemplazarse. Los bloques más grandes reducen el número de bloques que caben en la caché; a medida que un bloque se hace más grande, cada palabra adicional está más lejos de la requerida.

• 5.5 Memoria Principal (Memoria Interna).

Todos los **tipos de memoria principal** se implementan utilizando tecnología de semiconductores y tienen el transistor como elemento básico de su construcción. El elemento básico en toda memoria es la celda. Una celda permite almacenar un bit, un valor 0 o 1 definido por una diferencia de potencial eléctrico. La manera de construir una celda de memoria varía según la tecnología utilizada. La memoria interna es una memoria de acceso aleatorio; se puede acceder a cualquier palabra de memoria especificando una dirección de memoria. Una manera de clasificar la memoria interna según la perdurabilidad es la siguiente:

◇ MEMORIA VOLATIL.

La **memoria volátil** es la memoria que necesita una corriente eléctrica para mantener su estado, de manera genérica denominada RAM. Las memorias volátiles pueden ser de dos tipos:

1) SRAM. La **memoria estática de acceso aleatorio** (SRAM) implementa cada celda de utilizando un flip-flop básico para almacenar un bit de información, y mantiene la información mientras el circuito de memoria recibe alimentación eléctrica. Para implementar cada celda de memoria son necesarios varios transistores, por lo que la memoria tiene una capacidad de integración limitada y su coste es elevado en relación con otros tipos de memoria RAM, como la DRAM; sin embargo, es el tipo de memoria RAM más rápido.

2) DRAM. La **memoria dinámica** implementa cada celda de memoria utilizando la carga de un condensador. A diferencia de los flip-flops, los condensadores con el tiempo pierden la carga almacenada y necesitan un circuito de refresco para mantener la carga y mantener, por lo tanto, el valor de cada bit almacenado. Eso provoca que tenga un tiempo de acceso mayor que la SRAM. Cada celda de memoria está formada por solo un transistor y un condensador; por lo tanto, las celdas de memoria son mucho más pequeñas que las celdas de memoria SRAM, lo que garantiza una gran escala de integración y al mismo tiempo permite hacer memorias más grandes en menos espacio.

◇ MEMORIA NO VOLATIL.

La **memoria no volátil** mantiene el estado sin necesidad de corriente eléctrica. Las memorias no volátiles pueden ser de diferentes tipos:

1) Memoria de solo lectura o ROM (read only memory). Se trata de memorias de solo lectura que no permiten operaciones de escritura y, por lo tanto, la información que contienen no se puede borrar ni modificar. Este tipo de memorias se pueden utilizar para almacenar los microprogramas en una unidad de control microprogramada. La grabación de la información en este tipo de memorias forma parte del proceso de fabricación del chip de memoria.

2) Memoria programable de solo lectura o PROM (programmable read only memory). Cuando hay que fabricar un número reducido de memorias ROM con la misma información grabada, se recurre a otro tipo de memorias ROM: las memorias ROM programables (PROM). A diferencia de las anteriores, la grabación no forma parte del proceso de fabricación de los chips de memoria, sino que se efectúa posteriormente con un proceso eléctrico utilizando un hardware especializado para la grabación de memorias de este tipo. Como el proceso de programación no forma parte del proceso de fabricación, el usuario final de este tipo de memorias puede grabar el contenido según sus necesidades. El proceso de grabación o programación solo se puede realizar una vez.

3) Memoria reprogramable mayoritariamente de lectura. Esta puede ser de tres tipos:

a) EPROM (Erasable Programmable Read Only Memory). Se trata de memorias en las que habitualmente se hacen operaciones de lectura, pero cuyo contenido puede ser borrado y grabado de nuevo. Hay que destacar que el proceso de borrar es un proceso que borra completamente todo el contenido de la memoria; no se puede borrar solo una parte. Para borrar, se aplica luz ultravioleta sobre el chip de memoria EPROM; para permitir este proceso, el chip dispone de una pequeña ventana sobre la cual se aplica la luz ultravioleta. La grabación de la memoria se hace mediante un proceso eléctrico utilizando un hardware específico.

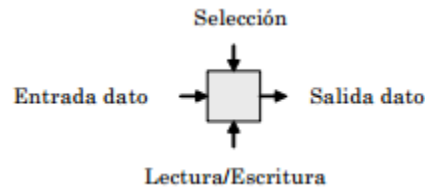
b) EEPROM (electrically erasable programmable read only memory). Permiten borrar el contenido y grabar información nueva; sin embargo, a diferencia de las memorias EPROM, todas las operaciones son realizadas eléctricamente. Para grabar datos no hay que borrarlos previamente; se permite modificar directamente solo uno o varios bytes sin modificar el resto de la información. Son memorias mayoritariamente de lectura, ya que el proceso de escritura es considerablemente más lento que el proceso de lectura.

c) Memoria flash. El borrado es eléctrico de la información, con la ventaja de que el proceso de borrar y grabar es muy rápido. La velocidad de lectura es superior a la velocidad de escritura, pero las dos son del mismo orden de magnitud. Este tipo de memoria no permite borrar la información byte a byte, sino que se deben borrar bloques de datos enteros.

• 5.6 Organización interna de la memoria.

Una memoria principal se compone de un conjunto de celdas básicas dotadas de una determinada organización. Cada celda soporta un bit de información. Los bits se agrupan en unidades direccionables denominadas palabras. La longitud de palabra la determina el número de bits que la componen y constituye la resolución de la memoria (mínima cantidad de información direccionable). La longitud de palabra suele oscilar desde 8 bits (byte) hasta 64 bits. Cada celda básica es un dispositivo físico con dos estados estables (o semi-estables) con capacidad para cambiar el estado (escritura) y determinar su valor (lectura). Aunque en los primeros computadores se utilizaron los materiales magnéticos como soporte de las celdas de memoria principal (memorias de ferritas, de película delgada, etc.) en la actualidad sólo se utilizan los materiales semiconductores.

Desde un punto de vista conceptual y con independencia de la tecnología, consideraremos la celda básica de memoria como un bloque con tres líneas de entrada (entrada dato, selección y lectura/escritura) y una de salida (salida dato). La celda sólo opera (lectura o escritura) cuando la selección está activa.



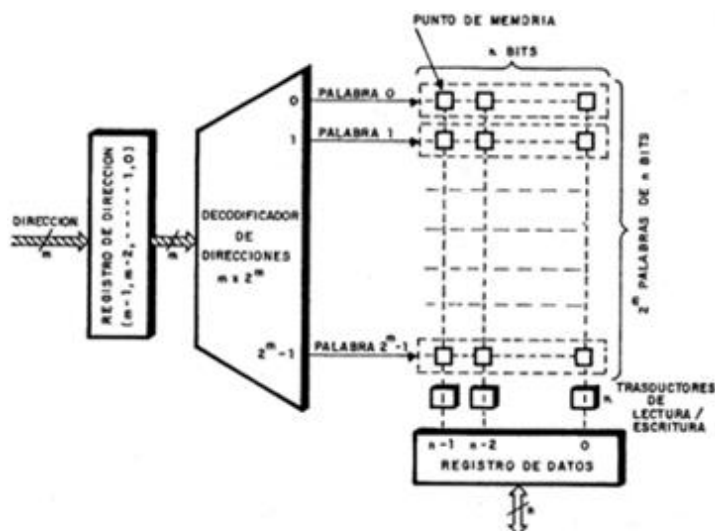
El mecanismo de direccionamiento de una memoria estática ofrece dos formas típicas llamadas 2D y 3D. Para analizarlas se considera que la memoria tiene 2^m palabras de n bits cada una. Se desea acceder a los n bits de una palabra, cuya dirección precisa de m bits para que quede definida.

♦ ORGANIZACIÓN 2D (MEMORIAS ESTATICAS).

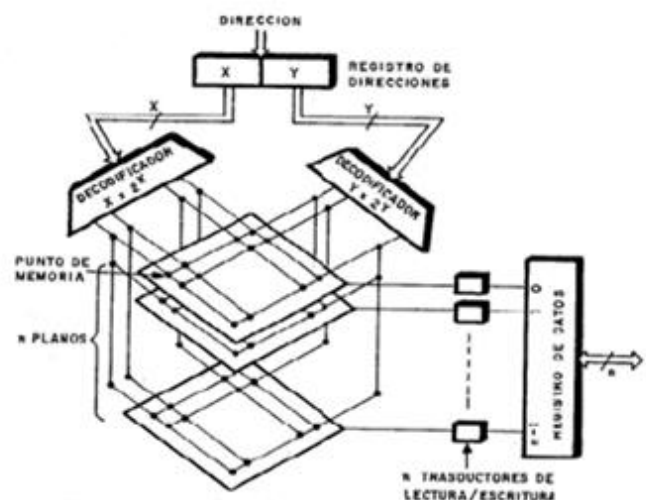
Las celdas se organizan en una matriz de dos dimensiones, en la que las filas vienen dadas por el número de palabras (n) y las columnas por la longitud (cantidad de bits) de cada palabra. Para seleccionar la palabra deseada se decodifican los m bits de dirección en un decodificador $m \times 2^m$, que tiene una señal de salida individualizada para cada palabra de memoria. Se usa la misma conexión para la lectura que para la escritura, bastando activar al transductor correspondiente para definir la operación.

♦ ORGANIZACIÓN 3D (MEMORIAS DINAMICAS).

Se establecen n planos de memoria (uno para cada bit de la palabra). Dentro de cada plano se selecciona el punto de memoria haciendo coincidir las líneas de selección X e Y. En lugar de una única selección (decodificador) de 2^n salidas en esta organización se utilizan dos decodificadores de $2^{n/2}$ operando en coincidencia. Las líneas de dirección se reparten entre los dos decodificadores. Para una configuración dada de las líneas de dirección se selecciona un único bit de la matriz.



Organización de una memoria con direccionamiento 2D.



Organización de una memoria con direccionamiento 3D

• 5.7 Memoria Secundaria (Memoria Externa).

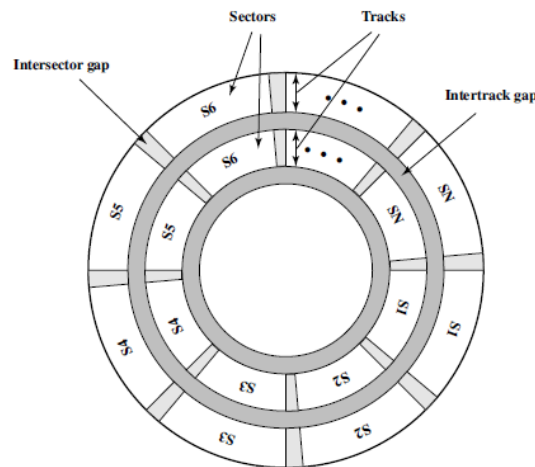
La **memoria externa** está formada por dispositivos de almacenamiento secundario (discos magnéticos, CD, DVD, BluRay, etc.). Estos dispositivos se pueden encontrar físicamente dentro o fuera del computador. La memoria externa es de tipo no volátil; por lo tanto, los datos que se quieran mantener durante un tiempo indefinido o de manera permanente se pueden almacenar en dispositivos de memoria externa. El método de acceso varía según el dispositivo: generalmente los dispositivos basados en disco utilizan un método de acceso directo, mientras que otros dispositivos, como las cintas magnéticas, pueden utilizar acceso secuencial.

Los datos almacenados en la memoria externa son visibles para el programador en forma de bloques de datos, no como datos individuales (bytes), normalmente en forma de registros o ficheros. El acceso a estos dispositivos se lleva a cabo mediante el sistema de E/S del computador y es gestionado por el sistema operativo.

♦ DISCOS MAGNETICOS.

Los **Discos Magnéticos** son dispositivos formados por un conjunto de platos con superficies magnéticas y un conjunto de cabezales de lectura y escritura que permiten grabar y recuperar datos en él. La información se graba en estas superficies. Durante una operación de escritura/lectura, la cabeza permanece quieta mientras el plato rota bajo ella. El mecanismo de escritura se basa en el campo magnético producido por el flujo eléctrico que atraviesa la bobina. El mecanismo de lectura se basa en la corriente eléctrica que atraviesa la bobina, producida por un campo magnético que se mueve respecto a la bobina. Los platos y los cabezales son accionados por motores eléctricos. Los platos hacen un movimiento de rotación continuo y los cabezales se pueden mover de la parte más externa del disco a la parte más interna, lo que permite un acceso directo a cualquier posición del disco.

♦ **Organización y formato de los datos:** Los datos se organizan en un conjunto de anillos concéntricos en el plato ('pistas', entre 500 y 2000 por superficie) del mismo ancho que la cabeza. Las pistas adyacentes están separadas por bandas vacías, lo cual minimiza los errores debidos a desalineamientos de la cabeza o interferencias del campo magnético. Se suele almacenar el mismo número de bits en cada pista, esto implica que la densidad aumenta según se avanza de la pista más externa a la más interna. Los datos se transfieren en bloques, los cuales se almacenan en regiones ('sectores', entre 10 y 100 por pista) de longitud fija o variable. Para evitar imprecisiones, los sectores adyacentes se separan con intrapistas vacías (intersectores).



♦ Características físicas de los discos magnéticos:

- **Tipos de cabezales:** pueden ser fijos o móviles con respecto a la dirección radial del plato. En un disco de cabeza fija, hay una cabeza de lectura/escritura por pista. Todas las cabezas se montan en un brazo rígido que se extiende a través de todas las pistas. En un disco de cabeza móvil, hay un solo cabezal de lectura/escritura. La cabeza se monta en un brazo y ésta debe poder posicionarse encima de cualquier pista.
- **Transportabilidad de disco:** un disco no extraíble está permanentemente montado en la unidad de disco (brazo, eje que hace rotar al disco y la electrónica necesaria para la E/S de datos binarios). Un disco extraíble puede ser quitado y sustituido por otro disco.
- **Superficies:** en la mayoría de los discos, la cubierta magnetizable se aplica a ambas caras del plato, denominándose estos discos de doble superficie. Algunos discos, menos caros, son de una sola superficie.
- **Platos:** algunas unidades de discos poseen varios platos, apilados verticalmente y separados por una distancia de alrededor de una pulgada. Los platos constituyen una unidad llamada paquete de disco.
- **Mecanismo de la cabeza:**
 - o **Separación fija:** se posiciona a una distancia fija sobre el plato dejando entre ambos una capa de aire.
 - o **Contacto:** la cabeza efectúa contacto físico con el medio durante la operación de lectura o escritura.

♦ **Parámetros para medir las prestaciones de un disco:** Cuando una unidad de disco funciona, el disco rota a una velocidad constante. Para leer/escribir, la cabeza debe posicionarse en la pista deseada y al principio del sector deseado. En un sistema de cabeza móvil, el tiempo que tarda en posicionarse en la pista es el '*tiempo de búsqueda*'. Una vez seleccionada la pista, se debe esperar hasta que el sector apropiado rote hasta alinearse con la cabeza; este tiempo que pasa se conoce como '*tiempo de latencia*'. El tiempo que se tarda en llegar a la posición de lectura/escritura, '*tiempo de acceso*', se conoce como la suma del '*tiempo de búsqueda*' más '*tiempo de latencia*'. Una vez posicionada la cabeza, se lleva a cabo la operación de lectura/escritura, lo que conlleva un tiempo de transferencia de datos.

- Tiempo de búsqueda = constante del disco * pistas atravesadas + tiempo de comienzo.
- Tiempo de latencia = medio: 8,3 ms. en discos, entre 100 y 200 ms. en disqueteras.
- Tiempo de transferencia = bytes a transferir / (bytes de una pista * velocidad de rotación).
- Tiempo de acceso medio total = tiempo e/ pistas * cantidad de pistas + tiempo de medio giro.
- Capacidad del disco = bytes/sector*sectores/pista*pistas/superficie*nº de superficies.

♦ REDUNDANT ARRAY OF INDEPENDENT DISKS (RAID).

Un sistema **RAID (Redundant Array of Independent Disks)** consiste en utilizar una colección de discos que trabajan en paralelo con el objetivo de mejorar el rendimiento y la fiabilidad del sistema de almacenamiento. Un conjunto de operaciones de E/S puede ser tratado en paralelo si los datos a los que se ha de acceder en cada operación se encuentran en diferentes discos; también una sola operación de E/S puede ser tratada en paralelo si el bloque de datos al cual hay que acceder se encuentra distribuido entre varios discos. Un RAID está formado por un conjunto de discos que el sistema operativo ve como un solo disco lógico.

Los datos se pueden distribuir entre los discos físicos según configuraciones diferentes. Se utiliza información redundante para proporcionar capacidad de recuperación en el caso de fallo en algún disco. La clasificación del tipo de RAID original incluye 7 niveles, del RAID 0 al RAID 6, en los que cada uno necesita un número diferente de discos y utiliza diferentes sistemas de control de la paridad y de detección y corrección de errores. El control de un sistema RAID se puede llevar a cabo mediante software o hardware, con un controlador específico.

♦ MEMORIA OPTICA.

♦ **CD y CD-ROM:** Tanto el **CD** de audio como el **CD-ROM** comparten una tecnología similar. La principal diferencia es que los lectores de CD-ROM son más robustos y tienen dispositivos de corrección de errores. El disco se forma a través de una resina, como el policarbonato, y se cubre con una superficie altamente reflectante (normalmente aluminio). La información digital se graba como una serie de hoyos microscópicos en la superficie reflectante. La superficie con los hoyos se protege con una capa final de laca transparente. La información se recupera con un láser de baja potencia, el cual pasa a través de la capa protectora transparente mientras un motor hace girar el disco sobre el láser. La intensidad de la luz reflejada cambia si se encuentra en un hoyo. Un foto sensor detecta este cambio que se convierte en una señal digital. La información se empaqueta con densidad uniforme a lo largo del disco en segmentos del mismo tamaño y se explora a la misma velocidad, rotando el disco a una velocidad variable. Se dice que el láser lee los hoyos a una velocidad lineal constante (CLV). Los datos de un CD-ROM se organizan en una secuencia de bloques, que constan de:

- Sincronización: Identifica el principio del bloque.
- Cabecera: Contiene la dirección del bloque y el byte de modo:
 - *Modo 0:* Campo de datos en blanco.
 - *Modo 1:* Uso de código de corrección de errores y 2.048 bytes de datos.
 - *Modo 2:* 2.336 bytes de datos de usuario sin código de corrección de errores.
- Datos: datos del usuario.
- Auxiliar:
 - *Modo 2:* Datos del usuario adicionales.
 - *Modo 1:* Código de corrección de errores de 288 bytes.

Los CD-ROM son apropiados para la distribución de grandes cantidades de datos a un gran número de usuarios.

Ventajas: (respecto con los discos magnéticos).

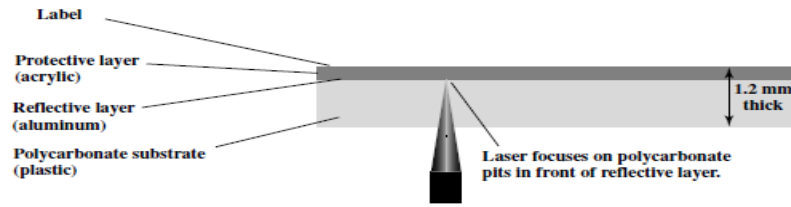
- La capacidad de almacenamiento es mucho mayor.
- Se puede replicar en grandes cantidades de forma barata.
- Es extraíble, puede ser usado como memoria de archivo.

Desventajas: (respecto con los discos magnéticos).

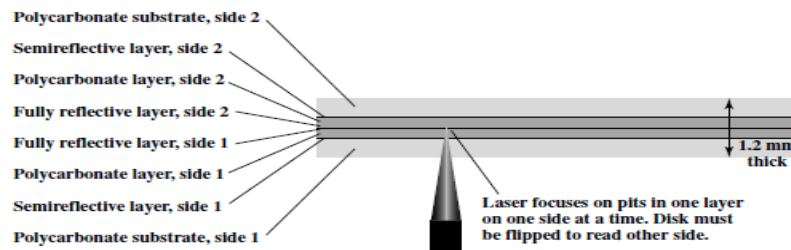
- Es de sólo lectura y no se puede actualizar.
- El tiempo de acceso es mayor que el de una unidad de disco magnético, tanto como medio segundo.

♦ **DVD:** Sustituye a las cintas VHS de vídeo analógicas con su gran capacidad de almacenamiento, y sustituye al CD-ROM en los PC y servidores: el DVD lleva al video a la edad digital. Proporciona películas con una calidad de imagen impresionante; se pueden acceder aleatoriamente como en los CDs. Se puede grabar un gran volumen de datos (7 veces más que un CD-ROM). Claves que los diferencia de los CD-ROM:

- Un DVD estándar almacena 4,7 GB por capa; los de doble capa y una cara almacenan 8,5 GB.
- Utiliza un formato de compresión (MPEG) par imágenes de pantalla completa de alta calidad.
- Un DVD de una capa puede almacenar una película de dos horas y media.



(a) CD-ROM—Capacity 682 MB



(b) DVD-ROM, double-sided, dual-layer—Capacity 17 GB

♦ **WORM:** Es un CD de una-escritura-varias-lecturas, para las aplicaciones en las que sólo se necesitan unas pocas copias de un conjunto de datos. Para proporcionar un acceso más rápido, el WORM usa *velocidad angular constante**, sacrificando parte de su capacidad.

Técnica típica para fabricar el disco: usar un láser de alta potencia para producir una serie de ampollas en el disco; así un láser de baja potencia puede producir calor suficiente para reventar las ampollas pregrabadas. Este disco óptico es atractivo para almacenar archivos de documentos y ficheros; proporciona una grabación permanente de grandes cantidades de datos del usuario.

* *Velocidad angular constante:* incrementa el espacio lineal entre bits de información grabados en los segmentos más externos del disco; explorando la información a la misma velocidad, girando el disco a una velocidad fija.

♦ **CINTA MAGNÉTICA:** Los sistemas de cinta usan la misma técnica de lectura y grabación que los discos. El medio es una cinta de plástico flexible, cubierta por un óxido magnético, análoga a una cinta de grabación doméstica. Se estructura en un pequeño número de pistas paralelas. Las primeras usaban 9 pistas (almacén de datos de un byte, con un bit de paridad); las nuevas usan 18 o 36 pistas (una palabra o doble palabra digital); los datos se leen y escriben en bloques contiguos (registros físicos) que están separados por bandas vacías (bandas inter-registros).

Es un dispositivo de acceso secuencial (tiene que leer los registros uno a uno para llegar al requerido; tiene que rebobinar un cierto trecho y empezar a leer si el registro deseado está en alguna posición anterior) que está en movimiento sólo durante las operaciones de lectura/escritura. Los extremos contienen unas marcas metálicas pegadas denominadas BOT y EOT para la detección automática del inicio y fin de la cinta, respectivamente. La capacidad depende de su longitud, densidad de grabación, longitud de bloque y formato de grabación. Las cintas magnéticas fueron el primer tipo de memorias secundarias.

Tipos de unidades de cinta magnética:

- **Cintas tradicionales de columnas de vacío:** las columnas de vacío tienen como objetivo mantener constante la tensión de la cinta bajo la estación de lectura/grabación.
- **Cintas tradicionales de brazos tensores:** Son más sencillas ya que no necesitan columnas de vacío pero con ellas obtiene menor velocidad.
- **Unidades de cassette de audio:** se utilizan en microcomputadoras domésticas y pequeños sistemas informáticos; el movimiento de la cinta se realiza con motores que actúan directamente sobre los carretes, no siendo tan rápidos ni precisos como las dos anteriores.
- **Unidades de cassette digitales:** disponen de cabestrantes para control de la velocidad de lectura/grabación.

UNIDAD 06 – PERIFÉRICOS

Todo computador necesita llevar a cabo intercambio de información con personas u otros computadores mediante unos dispositivos que denominamos de manera genérica **dispositivos periféricos**. Para hacer una operación de E/S entre el computador y un periférico, es necesario conectar estos dispositivos al computador y gestionar de manera efectiva la transferencia de datos. Para hacerlo, el computador dispone del **sistema de entrada/salida (E/S)**. Dada la gran variedad de periféricos, es necesario dedicar un hardware y un software específicos para cada uno. Por este motivo se ha intentado normalizar la interconexión de los periféricos y el computador mediante lo que se denomina **módulos de E/S o controladores de E/S**.

Hay que tener presente que la gestión global del sistema de E/S de un computador la hace el sistema operativo (SO). Las técnicas para controlar este sistema de E/S las utiliza el SO y el programador cuando quieren acceder al periférico, pero en las máquinas actuales, a causa de la complejidad de controlar y gestionar los periféricos, el acceso se lleva a cabo generalmente mediante llamadas al SO, que es quien gestiona la transferencia. El conjunto de rutinas que permiten controlar un determinado periférico es lo que denominamos habitualmente **programas controladores o drivers** y cuando el SO quiere hacer una operación de E/S con un periférico llama a una de estas rutinas.

Los periféricos no se conectan directamente al bus del sistema por las siguientes razones:

- Hay una amplia variedad de periféricos con formas de funcionamiento diferentes, por lo que sería imposible incorporar la lógica necesaria dentro del procesador para controlar a todos ellos.
- A menudo, la velocidad de transferencia de datos de los periféricos es muy lenta, por lo que no es práctico usar un bus de sistema de alta velocidad.
- Con frecuencia, los periféricos utilizan datos con formatos y tamaños de palabra diferentes de los de la computadora.

♦ ASPECTOS BASICOS DE E/S.

Cuando hablamos de E/S de información entre un computador y un periférico lo hacemos siempre desde el punto de vista del computador. Así, decimos que es una **transferencia de entrada** cuando el periférico es el emisor de la información y tiene como receptor el computador (procesador o memoria) y decimos que es una **transferencia de salida** cuando el computador es el emisor de la información y tiene como receptor el periférico.

De manera más concreta, toda operación de E/S que se lleva a cabo entre el computador y un periférico es solicitada y gobernada desde el procesador, es decir, es el procesador quien determina en qué momento se debe hacer y con qué periférico, si la operación es de lectura o escritura, qué datos se han de transferir, y también quién da la operación por acabada. Para llevar a cabo la operación de E/S, hemos de conectar el periférico al computador. Para hacerlo, es necesario que el computador disponga de unos dispositivos intermedios por donde ha de pasar toda la información que intercambia el computador con el periférico y que nos permite hacer una gestión y un control correctos de la transferencia. Estos dispositivos los llamamos de manera genérica **módulo de E/S**.

♦ DISPOSITIVOS EXTERNOS

Las operaciones de E/S se realizan a través de una amplia gama de dispositivos. Un dispositivo externo se conecta a la computadora mediante un enlace a un módulo de E/S. El enlace se utiliza para intercambiar señales de control, estado y datos entre los módulos de E/S y el dispositivo externo (periférico). Se pueden clasificar en:

A) Interacción con humanos (hombre-máquina): Permite la comunicación con el usuario de la computadora. Esta comunicación se realiza a través de periféricos (monitor, etc.)

B) Interacción con máquinas (máquina-máquina): Permite la comunicación con elementos de equipo (discos magnéticos, cintas, etc.). La comunicación es digital. Si los computadores están cerca se conectan directamente en Banda Base, mediante un bus paralelo que transmite bytes o palabras. Si existe una distancia de más de 100 mts se utilizan Módems.

C) De comunicación: Permite la comunicación con dispositivos remotos.

La conexión con el módulo de E/S se realiza a través de señales de control, estado y datos. Los datos se intercambian en forma de un conjunto de bits que son enviados/recibidos a/desde el módulo de E/S. Las señales de control determinan la función que debe realizar el dispositivo. Las de estado indican el estado del chip.

• 6.1 Tipos de periféricos.

Los **periféricos** son dispositivos que se conectan al computador mediante los módulos de E/S y que sirven para almacenar información o para llevar a cabo un tipo determinado de comunicación con el exterior con humanos, con máquinas o con otros computadores.

La clasificación más habitual es la siguiente:

- Para la interacción con humanos:
 - Entrada.
 - Salida.
- Para la interacción con otros computadores o sistemas físicos:
 - Almacenamiento.
 - Comunicación.

En un periférico distinguimos habitualmente dos partes: una parte mecánica y una parte electrónica. La parte mecánica hace funcionar los elementos principales que forman el periférico, como el motor para hacer girar un disco o mover el cabezal de una impresora, el botón de un ratón o el láser de un dispositivo óptico. La parte electrónica nos permite, por una parte, generar las señales eléctricas para gestionar los elementos mecánicos y, por otra parte, hacer la conversión de los datos provenientes del computador a señales eléctricas o al revés.

La conexión física entre un periférico y el computador se lleva a cabo mediante lo que denominamos sistema de interconexión de E/S. Este sistema de interconexión de E/S nos permite hacer la gestión de las señales de control, de estado y de datos necesarias para llevar a cabo una transferencia de información que, como veremos más adelante, es gestionada desde el módulo de E/S del computador.

♦ **TECLADO:** Cuando utilizamos un teclado, al pulsar una tecla se cierra un conmutador que hay en el interior del teclado, esto hace que unos circuitos codificadores generen el código de E/S correspondiente al carácter seleccionado, apareciendo éste en la pantalla si no es un carácter de control. Los teclados contienen los siguientes tipos de teclas:

- Teclado principal: Contiene los caracteres alfabéticos, numéricos y especiales.
- Teclas de desplazamiento del cursor: Permiten desplazar el cursor a izquierda, derecha, arriba y abajo.
- Teclado numérico: Es habitual en los teclados de computadora que las teclas correspondientes a los caracteres numéricos (cifras decimales), signos de operaciones básicas y punto decimal estén repetidas para facilitar al usuario la introducción de datos numéricos.
- Teclas de funciones: Son teclas cuyas funciones son definibles por el usuario o están.
- Teclas de funciones locales: Controlan funciones propias del terminal, como impresión del contenido de imagen cuando el computador está conectada a una impresora.

♦ **MOUSE:** es una pequeña caja de plástico que descansa sobre la mesa junto al teclado. Cuando el ratón se mueve sobre la mesa, también se mueve un puntero en la pantalla, que apunta a los elementos. El ratón tiene uno, dos o tres botones en la parte superior, que permiten seleccionar opciones de menú. Existen tres tipos de ratones: mecánicos, ópticos y optomecánicos.

1) Mecánicos: tenían dos ruedas de caucho que sobresalían por debajo y tenían sus ejes perpendiculares entre sí. Cuando el ratón se movía paralelo a su eje principal, una rueda giraba. Cuando se movía perpendicularmente, giraba la otra rueda. Cada rueda impulsaba un resistor variable. Si se medían los cambios en la resistencia, era posible ver qué tanto había girado cada rueda y así calcular qué tanto se había movido el ratón en cada dirección. Este tipo se reemplazó por uno con una esfera que sale ligeramente de su base.

2) Ópticos: no tiene rueda ni esfera, tiene un LED y un fotodetector en la base. El ratón se usa sobre una base que contiene una cuadrícula de líneas. A medida que el ratón se mueve sobre ella, el fotodetector percibe el paso de las líneas por los cambios en la cantidad de luz que se refleja. Unos circuitos internos cuentan el número de líneas que se cruzan en cada dirección.

3) Optomecánicos: tiene una esfera que hace girar dos ejes perpendiculares entre sí. Los ejes se conectan a codificadores provistos de ranuras a través de las cuales puede pasar la luz. A medida que el ratón se mueve, los ejes giran y pulsos de luz inciden sobre los detectores cada vez que una ranura pasa entre un LED y su detector. El número de pulsos detectados es proporcional a la cantidad de movimiento.

La configuración más común de un mouse es hacer que envíe una secuencia de 3 bytes a la computadora cada vez que se mueva cierta distancia mínima, llamada mickey. Los bytes llegan por una línea serial, bit por bit. El primer byte contiene un entero con signo que indica cuántas unidades se movió el ratón en la dirección x en los últimos 100ms. El segundo byte da la misma información para la dirección y. El tercer byte contiene la situación actual de los botones. Software de bajo nivel acepta esta configuración y convierte los movimientos relativos en una posición absoluta. Luego exhibe una flecha en la pantalla en la posición correspondiente.

♦ **SCANNER:** es un dispositivo que recuerda a una fotocopidora y que se emplea para introducir imágenes en un computador. Las imágenes que se desee capturar deben estar correctamente iluminadas para evitar brillo y tonos no deseados. Son dispositivos de entrada de datos de propósito especial que se emplean conjuntamente con paquetes software para gráficos y pantallas de alta resolución. La mayor parte de los scanners capturan imágenes en color generando una determinada cantidad de bits por cada punto. La cantidad de espacio de almacenamiento que se necesita para una imagen depende de las dimensiones máximas de la imagen y de la resolución de captura del equipo; la resolución se describe en 'cantidad de *puntos por pulgada*' ('*dot per inch*' en inglés o con el acrónimo '*dpi*') en el sentido horizontal (eje x) y en el vertical (eje y).

♦ **MONITOR CRT:** un monitor es una caja que contiene un tubo de rayos catódicos (CRT) y sus fuentes de potencia. El CRT contiene un cañón que puede disparar un haz de electrones contra una pantalla fosforescente cerca del frente del tubo (los monitores a color tienen 3 cañones: rojo, verde y azul). Una imagen de pantalla completa normalmente se redibuja entre 30 y 60 veces cada segundo. Para producir un patrón de puntos en la pantalla, hay una rejilla dentro del CRT que, cuando se aplica en ella un voltaje positivo el haz choca la pantalla y brilla y, cuando el voltaje es negativo la pantalla no brilla. Así, el voltaje aplicado a la rejilla hace que el patrón de bits aparezca en pantalla. Este mecanismo permite convertir una señal eléctrica binaria en una imagen formada por puntos brillantes y oscuros.

♦ **MONITOR LCD (PANTALLA DE CRISTAL LIQUIDO):** una pantalla LCD consiste en dos placas de vidrio paralelas entre las que hay un volumen sellado que contiene cristal líquido. Cada placa tiene conectados electrodos transparentes que crean campos eléctricos en el cristal. Una luz ilumina la pantalla desde atrás. Diferentes partes de la pantalla reciben diferentes voltajes, y con esto se controla la imagen que se exhibe. En los LCD color se utilizan filtros ópticos para separar la luz blanca en componentes rojo, verde y azul.

➤ Clasificación de pantallas:

- Según la capacidad o no de mostrar colores se clasifican en:
 - *Monitor monocromo:* Los colores usuales en un monitor monocromático son el blanco y negro.
 - *Monitor color:* El color de cada punto se obtiene con mezcla de los colores rojo, verde y azul.
- Según su capacidad de representación se pueden clasificar en:
 - *Pantallas de caracteres:* Sólo admiten caracteres.
 - *Pantallas gráficas:* Permiten trazados de líneas y curvas continuas.

➤ Principales parámetros que caracterizan a una pantalla:

- Tamaño: Se describen en función del tamaño de la diagonal principal, y se da en pulgadas.
- Número de celdas o caracteres: lo usual es una representación de 24 filas * 80 col. de caracteres.
- Resolución: número de puntos de imagen en pantalla. Éste número no depende del tamaño de la pantalla.

♦ **IMPRESORAS:** son periféricos que escriben la información de salida sobre papel. Su comportamiento inicialmente era muy similar al de las máquinas de escribir, pero hoy día son mucho más sofisticadas, pareciéndose algunas en su funcionamiento a máquinas fotocopadoras conectadas en línea con el computador.

➤ Clasificación y tipos de impresoras:

A) Calidad de impresión: Tiene en cuenta la calidad de presentación y de contraste de los caracteres impresos.

- *Impresoras normales:* Como las impresoras de línea, de rueda y térmicas.
- *Impresoras de semicalidad:* Como algunas impresoras matriciales.
- *Impresoras de calidad:* Como las impresoras margarita e impresoras láser.

B) Sistema de impresión: Según la forma en que realizan la impresión.

- *Por impacto de martillos:* El fundamento de las impresoras por impacto es similar al de las máquinas de escribir. Las impresoras de impacto son muy ruidosas y han sido las más utilizadas.
- *Sin impacto:* forman los caracteres sin necesidad de golpes mecánicos y utilizan otros principios físicos para transferir las imágenes al papel.

C) Forma de imprimir los caracteres:

- *Impresoras de caracteres:* Realizan la impresión por medio de un cabezal que va escribiendo la línea carácter a carácter.
- *Impresoras de líneas:* En estas impresoras se imprimen simultáneamente todos o varios de los caracteres correspondientes a una línea de impresión.
- *Impresoras de páginas:* Aquí se incluyen un grupo de impresoras que actúan de forma muy similar a las máquinas fotocopadoras.

➤ Tipos de impresoras:

1) Impresora de Matriz: El tipo de impresora más económico es la impresora de matriz, en la que una cabeza de impresión que contiene entre 7 y 24 agujas activables electromagnéticamente se mueve a lo largo de cada línea de impresión. Las impresoras de matriz son económicas y muy confiables, pero son lentas, ruidosas y malas para imprimir gráficos. Se usan en tickets, o para imprimir en formatos grandes, entre otros.

2) Impresora de Inyección de Tinta: La cabeza de impresión móvil, que lleva un cartucho de tinta, se mueve horizontalmente a lo ancho del papel mientras rocía tinta con sus boquillas. Dentro de cada boquilla, una pequeña gota de tinta se calienta eléctricamente más allá de su punto de ebullición, hasta que hace explosión, sale y choca con el papel. Luego la boquilla se enfría y el vacío que se produce succiona otra gota de tinta. Son económicas, silenciosas y con buena calidad, pero también son lentas, sus cartuchos son caros y producen impresiones saturadas de tinta.

3) Impresora Láser: La impresora láser combina una alta calidad, excelente flexibilidad, buena velocidad y costo moderado en un mismo periférico. Permite usar todos los colores y puede imprimir páginas completas de texto o gráfico a gran velocidad. Su desventaja consiste en que son costosas.

4) Impresoras térmicas: Son similares a las impresoras de agujas. Se utiliza un papel especial termosensible que se ennegrece al aplicar calor. El calor se transfiere desde el cabezal por una matriz de pequeñas resistencias en las que al pasar una corriente eléctrica por ellas se calientan, formándose los puntos en papel.

➤ Parámetros que caracterizan a una impresora:

A) Velocidad de escritura: normalmente se expresa en las siguientes unidades:

- Impresoras de caracteres: Caracteres por segundo (cps).
- Impresoras de líneas: Líneas por minuto (lpm).
- Impresoras de páginas: Páginas por minuto (ppm).

B) Caracteres por línea: es el número máximo de caracteres que se pueden escribir en una línea.

C) Ancho del papel: se suele expresar en pulgadas.

D) Densidad de líneas: se expresa normalmente en líneas por pulgada e indica el espaciado entre líneas.

E) Color: es la posibilidad de imprimir en colores o no.

F) Resolución: la resolución se suele expresar como número de puntos por unidad de superficie.

♦ **MODEM:** es un dispositivo que permite conectar dos computadores remotos utilizando la línea telefónica de forma que puedan intercambiar información entre sí. La información que maneja el computador es digital, sin embargo, por las limitaciones físicas de las líneas de transmisión, no es posible enviar información digital a través de un circuito telefónico, solo pueden transmitirse señales analógicas. Entonces, para poder utilizar las líneas telefónicas (y en general cualquier línea de transmisión) para el envío de información entre computadoras digitales, es necesario un proceso de transformación de la información. Durante este proceso la información se adecua para ser transportada por el canal de comunicación. Este proceso se conoce como modulación-demodulación.

Un **Módem** es un dispositivo que posee conversores A/D y D/A especialmente adecuados para conectar líneas telefónicas al computador. De este modo las señales provenientes de una línea telefónica (por ejemplo por una llamada) son interpretadas y "atendidas" por el módem, permitiendo que otra computadora transmita información directamente a la nuestra. Recíprocamente, nuestra voz, una imagen o datos guardados en nuestra computadora pueden ser manejados por el módem para ponerlos (previa llamada) sobre la línea telefónica que del otro lado tendrá una computadora receptora (o eventualmente un ser humano que descuelga un teléfono).

• **Baudios:** los módems envían datos como una serie de tonos a través de la línea telefónica. Los tonos se "encienden" (ON) o apagan (OFF) para indicar un 1 o un 0 digital. El baudio es el número de veces que esos tonos se ponen a ON o a OFF. Los módems modernos pueden enviar 4 o más bits por baudio.

• **Bits por segundo (BPS):** es el número efectivo de bits/seg que se transmiten en una línea por segundo. Un módem de 600 baudios puede transmitir a 1200, 2400 o, incluso a 9600 BPS.

♦ **TERMINALES INTERACTIVAS:** La combinación de un monitor de vídeo con su correspondiente teclado se llama frecuentemente terminal y es normal acoplar varios terminales a un computador que se encarga de procesar las distintas tareas que cada usuario (desde su terminal) le ordena. Podemos distinguir dos tipos de terminales:

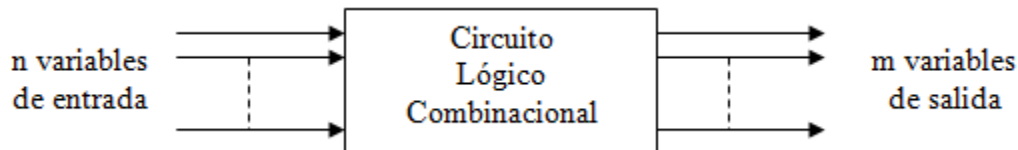
1) Terminales no inteligentes: Sólo son capaces de ejecutar operaciones de E/S simples.

2) Terminales inteligentes: Capaces de ejecutar ciertos procesos tales como manipulación de texto, posibilidades gráficas o programas simples dirigidos por menús para ayudar a la entrada de datos. Esto es posible al incluir microprocesadores en los terminales.

APENDICE A – SISTEMAS COMBINACIONALES

• A.1 Sistemas Combinacionales.

Un **circuito combinacional** consta de compuertas lógicas cuyas salidas en cualquier momento se determinan directamente a partir de los valores de las entradas presentes. Un circuito combinatorio realiza una operación de procesamiento de información determinada que se puede especificar lógicamente por medio de un conjunto de expresiones booleanas. Los circuitos combinacionales se emplean para generar decisiones de control binarias y para proporcionar los componentes digitales requeridos para el procesamiento de datos.

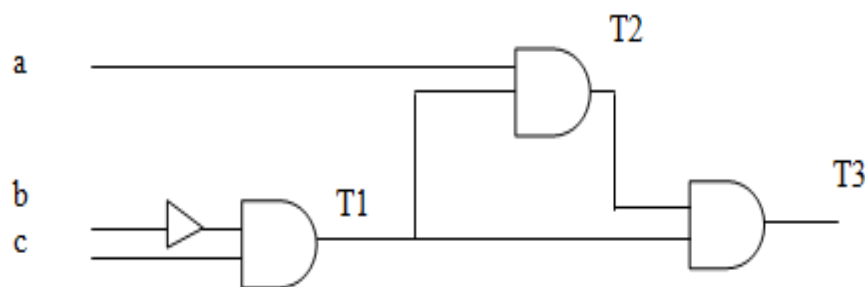


Para n variables de entrada existen 2^n combinaciones binarias posibles. Para cada combinación binaria de las variables de entrada existe un valor binario de salida posible. Por lo tanto un circuito combinacional se puede especificar a través de una tabla de verdad que presente los valores de salida de cada combinación de las variables de entradas.

♦ ANALISIS DE LOS SISTEMAS COMBINACIONALES.

El **análisis de un circuito combinacional** consiste en determinar la función que ejecuta el circuito. Se inicia con un diagrama de circuito lógico y culmina con un conjunto de funciones booleanas, una tabla de verdad y una posible explicación de la operación del circuito. El primer paso en el análisis consiste en asegurarse que el circuito dado sea combinacional y no secuencial. El diagrama de un circuito combinatorio tiene compuertas lógicas sin elementos de retroalimentación o almacenamiento. Cuando se verifica que un diagrama lógico es un circuito combinacional, se puede proceder a obtener la función booleana de salida o la tabla de verdad.

- Obtención de funciones booleanas a partir de un diagrama:



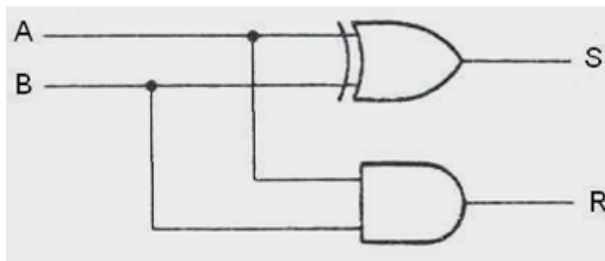
- 1) Identificar y nombrar las salidas de aquellas compuertas cuyas entradas están en función de la variable de entrada, para luego escribir la función booleana que la expresa $\rightarrow T1 = b * c$.
- 2) Identificar y nombrar aquellas compuertas que tengan como entrada las variable de entrada y/ o las salidas de las compuertas identificadas en el paso anterior. Escribir su función $\rightarrow T2 = a * T1$.
- 3) Repetir el paso anterior hasta las variables de salida $\rightarrow T3 = T1 * T2 = (b * c) * (a * (b * c)) = a * b * c$.

- Obtención de la tabla de verdad a partir de la función booleana:

Una vez que se conocen las funciones booleanas de salida, se procede a construir una tabla de verdad en la cual se escribirán todas las combinaciones posibles de las variables que intervienen en la función y luego tomando estas combinaciones se reemplaza en la función booleana para así obtener el valor de la salida.

• A.2 Semisumador (Sumador Medio).

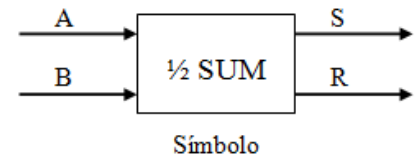
El **semisumador** es un circuito que realiza la suma de dos dígitos binarios (bits). El resultado de la suma consta normalmente de dos dígitos binarios, uno de menor peso S y otro de mayor peso o arrastre R. El bit de arrastre es 0 a no ser que ambas entradas sean 1. La salida S representa el bit menos significativo de la suma, en cambio el arrastre (acarreo) es el bit más significativo de la suma.



$$\text{Suma: } S = A \oplus B$$

$$\text{Arrastre: } R = A \cdot B$$

Entradas		Suma	
A	B	R	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



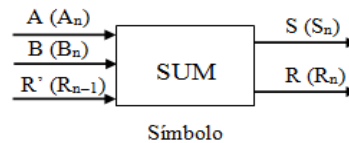
• A.3 Sumador Completo.

El semisumador correspondía a la adición de dos dígitos aisladamente considerados. Para realizar la suma de dos números binarios es preciso tener en cuenta, en cada posición n, los dos dígitos de dicha posición y el arrastre de la posición n-1. Al operador que realiza la adición de 2 dígitos binarios y del arrastre eventual (es decir, una suma de 3 bits) se le llama etapa de sumador. El **sumador completo** es un circuito combinacional que consiste en 3 entradas y 2 salidas. A las 3 entradas las designamos A, B, R' y a las 2 salidas S y R. La salida S vale 0, si todos los bits de entrada son ceros, en cambio, S es igual a 1 solamente si una entrada es igual a 1 o cuando las 3 entradas son 1. Por otra parte la salida R tiene un bit de arrastre de 1 si dos de las tres entradas son iguales a 1.

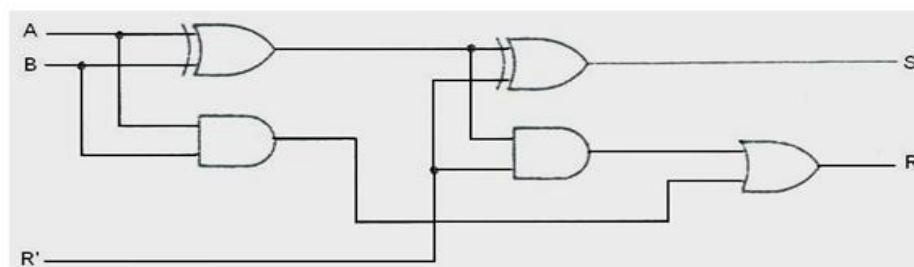
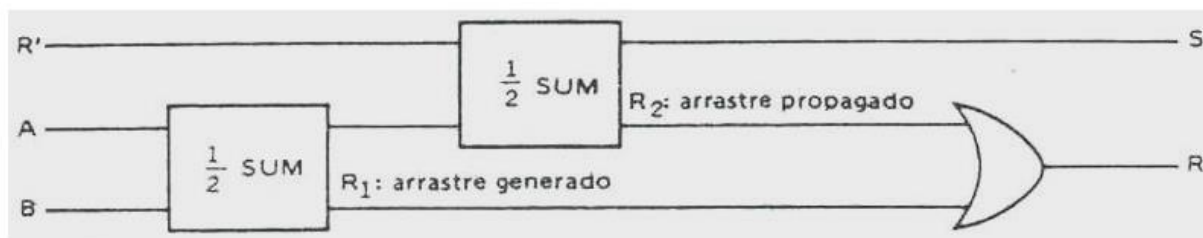
Entradas			Salidas	
A	B	R'	R	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$\text{Suma: } S = (A \oplus B) \oplus R'$$

$$\text{Arrastre: } R = (A \oplus B) R' + AB$$

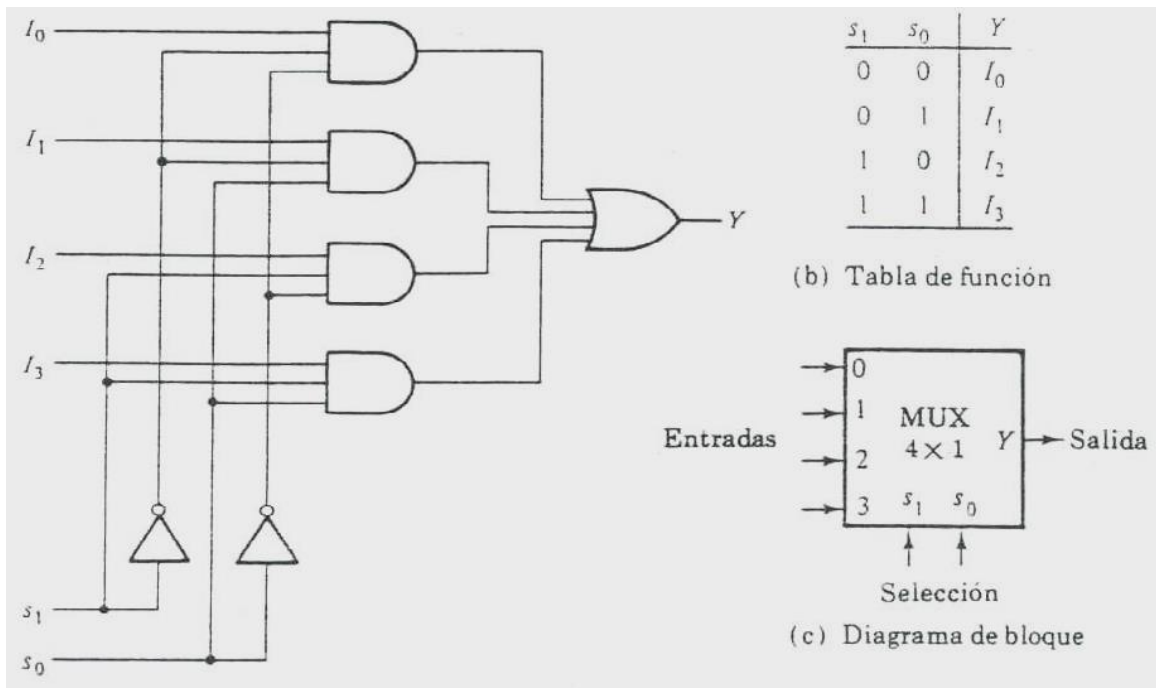


Construyamos una etapa de sumador a partir de 2 semisumadores: sumemos A y B en un primer semisumador; luego sumemos al resultado S1 el arrastre R' proveniente de la etapa anterior; obtenemos el resultado S de la suma y dos arrastres: R1 generado en la etapa, R2 propagado por la misma; los reunimos mediante un OR lógico y obtenemos el arrastre de la suma.



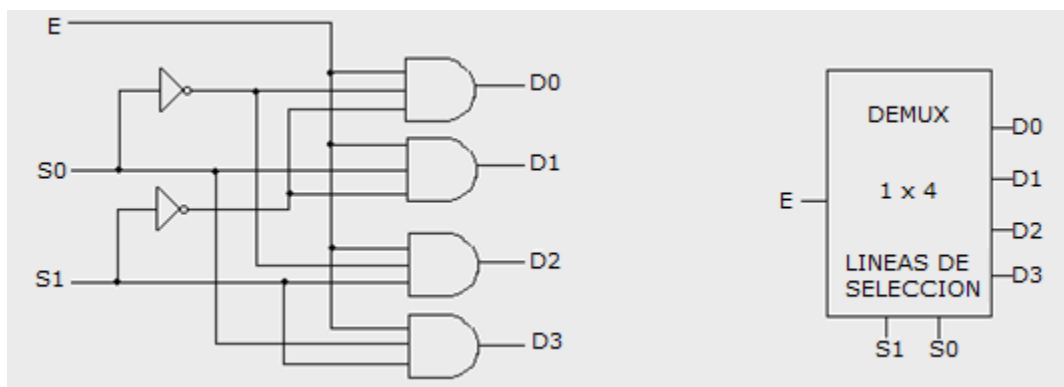
• A.4 Multiplexor.

Un **multiplexor** digital es un circuito combinacional que selecciona información binaria de una de muchas líneas de entrada para dirigirla a una sola línea de salida. La selección de una línea de entrada en particular es controlada por un conjunto de líneas de selección. Normalmente hay 2^n líneas de entrada y n líneas de selección cuyas combinaciones de bits determinan cual entrada se selecciona. Un multiplexor se llama también **selector de datos** ya que selecciona una de muchas entradas y guía la información binaria a la línea de salida.



• A.5 Demultiplexor.

Un **demultiplexor** es un circuito lógico combinacional que recibe información por una sola línea de entrada y transmite esta información en una de las 2^n líneas posibles de salida. La selección de una línea de salida específica se controla por los valores de los bits de las n líneas de selección.

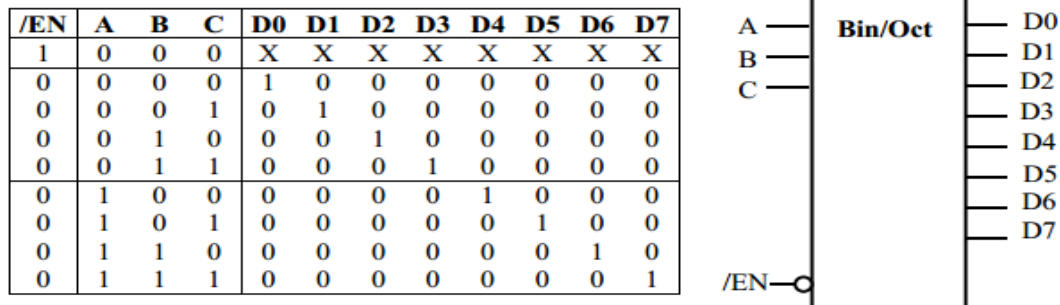


La tabla de verdad del Demultiplexor: (con K se representa un valor binario que podrá ser '0' o '1')

S1	S0	E	D0	D1	D2	D3
0	0	K	K	0	0	0
0	1	K	0	K	0	0
1	0	K	0	0	K	0
1	1	K	0	0	0	K

• A.6 Decodificador.

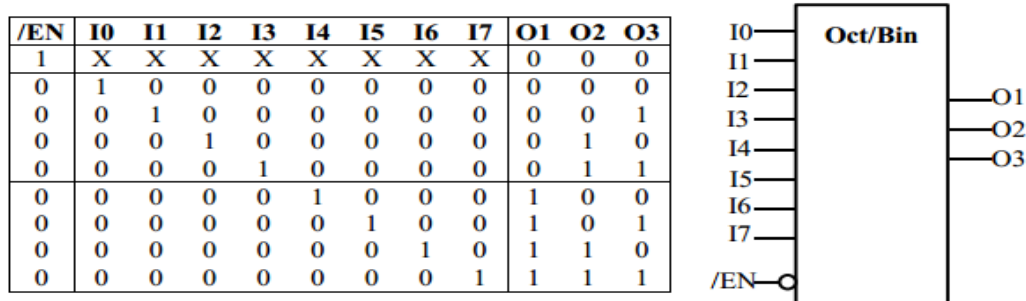
Un **decodificador** es un circuito lógico con n entradas y 2^n salidas, tal que para cada combinación de entradas se activa al menos una salida. Puede servir para convertir una información codificada, como puede ser un número binario, en una información no codificada como puede ser un número decimal. Dispone de una entrada de habilitación (EN: Enable) que conecta o desconecta el dispositivo y puede ser activa a nivel bajo, ya que el dispositivo se activa cuando dicha entrada recibe un '0' lógico o activa a nivel alto, que se activa al recibir un '1' lógico.



DECODIFICADOR BINARIO/OCTAL 3 A 8 ACTIVO A NIVEL BAJO

• A.7 Codificador.

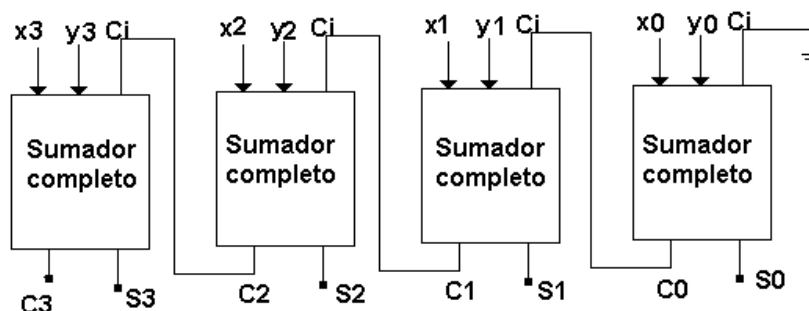
Un **codificador** es un dispositivo MSI (Medium Scale Integration) que realiza la operación inversa a la realizada por los decodificadores. Generalmente, poseen 2^n entradas y n salidas. Las salidas codificadas, generalmente se usan para controlar un conjunto de 2^n dispositivos, suponiendo claro está que sólo uno de ellos está activo en cualquier momento.



CODIFICADOR OCTAL/BINARIO 8 A 3 ACTIVO A NIVEL BAJO

• A.8 Sumador Paralelo Binario.

Un **sumador paralelo binario** es un circuito digital que produce la suma aritmética de dos números binarios en paralelo. Consta de circuitos sumadores completos conectados en cascada, con el acarreo de salida de un sumador completo conectado al acarreo de entrada del sumador completo siguiente. Los acarreos se conectan en cadena a través de los sumadores completos. El acarreo de entrada al sumador en paralelo es C0 y el de salida es C4. Un sumador paralelo de n bits requiere n sumadores completos con cada acarreo de salida conectado al acarreo de entrada del sumador completo de siguiente orden superior.

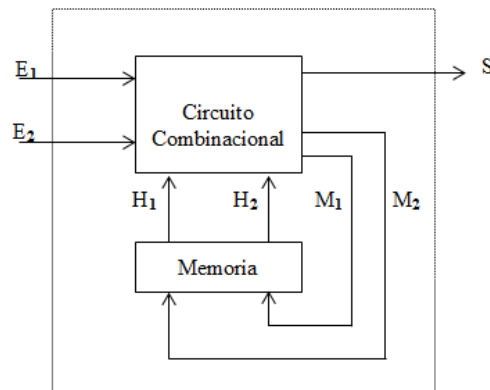


SUMADOR PARALELO BINARIO DE 4 BITS

APENDICE B – SISTEMAS SECUENCIALES

• B.1 Sistemas Secuenciales.

Los **circuitos secuenciales** usan elementos de memoria (celdas binarias) además de las compuertas lógicas. Las salidas de un circuito secuencial son una función no solamente de las entradas externas sino del presente estado de los elementos de la memoria. El estado de los elementos de memoria, a su vez, es una función de las entradas previas. Como consecuencia las salidas de un circuito secuencial dependen no solo de las entradas presentes, sino también, de las entradas del pasado y el comportamiento del circuito debe especificarse por una secuencia de tiempos de las entradas y estados internos.



Hay dos tipos de circuitos secuenciales. Su clasificación depende del temporizado de sus señales. Un circuito secuencial sincrónico es un sistema cuyo comportamiento puede definirse a partir del conocimiento de sus señales en instantes discretos de tiempos. El comportamiento de un circuito secuencial asincrónico depende del orden en que cambien las señales de entrada y puedan ser afectadas en cualquier instante de tiempo dado. Los elementos de memoria comúnmente usados en los circuitos secuenciales asincrónicos son mecanismos retardadores de tiempo.

• B.2 Biestables (Flip-Flops).

Los **biestables** son los elementos de memoria que se usan en los circuitos secuenciales temporizados. Estos circuitos son celdas binarias capaces de almacenar un bit de información. Un circuito flip-flop tiene dos salidas, una para el valor normal y uno para el valor complemento del bit almacenado en él. La información binaria puede entrar a un flip-flop en una gran variedad de formas, hecho que da lugar a diferentes tipos de flip-flops.

Según en qué instante pueden actuar sus entradas y cambiar de estado (y en correspondencia cambiar el valor de sus salidas), los flip-flops pueden ser:

- Asincrónico: sus entradas pueden actuar en cualquier instante para cambiar el estado del circuito, y en conformidad cambiar el valor de sus salidas.
- Sincrónico: sus entradas pueden actuar sobre el estado del circuito sólo en determinados instantes, definidos en relación a pulsos que llegan por otra entrada de sincronismo conocida como reloj o "clock". La entrada CLK recibe pulsos de sincronismo que son generados por lo general a una frecuencia regular por algún tipo de oscilador.
- Flip-Flops más utilizados:

Flip – Flop RS { Asincrónico
Sincrónico

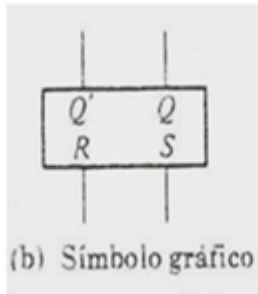
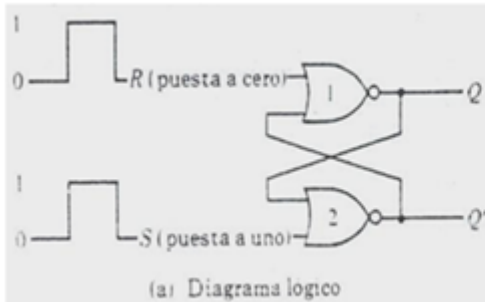
Flip – Flop JK { Asincrónico
Sincrónico

Flip – Flop D { Sincrónico

Flip – Flop T { Asincrónico
Sincrónico

• B.3 Biestable RS Asíncrono.

El **biestable RS asíncrono** dispone de 2 entradas S (set), que cuando tiene valor 1 pone en 1 la salida y R (reset), que cuando tiene valor 1 pone en 0 la salida y de dos salidas Q y Q', donde una es el complemento de la otra, y se les trata como salidas normales y de complemento respectivamente. Cuando las dos entradas tienen valor 0, no se producen cambios en la salida, pero cuando ambos valores valen 1, el Biestable RS no sabe bien cómo actuar y la salida puede cambiar o quedarse inalterada de forma aleatoria.

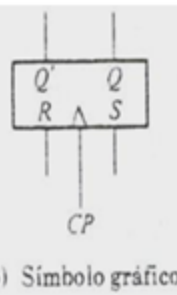
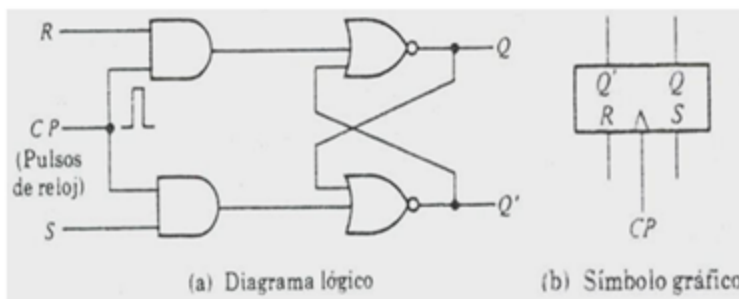


S (Set)	R (Reset)	Q	$\sim Q$
0	0	0	0 (Memoria)
0	1	0	1 (Reset)
1	0	1	0 (Set)
1	1	?	?

TABLA DE ESTADOS

• B.4 Biestable RS Síncrono.

El **biestable RS síncrono** dispone de 3 entradas Set, Reset y CLK; la entrada CLK no se escribe sino que se simboliza con un triángulo, que denota el hecho de que el flip-flop responde a una transición del reloj. Funciona exactamente igual que el RS asíncrono, con la diferencia de que sus estados cambian durante los pulsos del reloj.

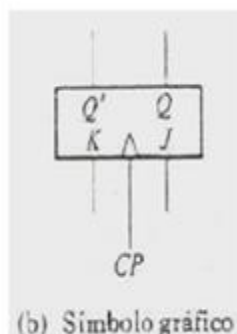
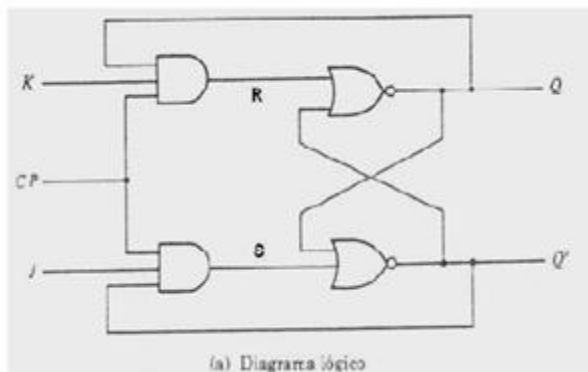


CLK (Clock)	S (Set)	R (Reset)	Q_{t+1}
0	X	X	Q_t
1	0	0	Q_t
1	0	1	0
1	1	0	1
1	1	1	?

TABLA DE ESTADOS

• B.5 Biestable JK.

El **biestable JK** fue diseñado para evitar el problema de indeterminación que poseen los Biestables RS cuando ambas entradas tenían el valor 1. La entrada J tiene la función de SET (poner en 1 la salida cuando J vale 1) y la entrada K, funciona como RESET (poniendo en 0 la salida cuando K vale 1). Cuando ambas entradas valen 0, no se producen cambios y cuando ambas entradas valen 1, se realiza la función denominada conmutación: la salida se invierte.

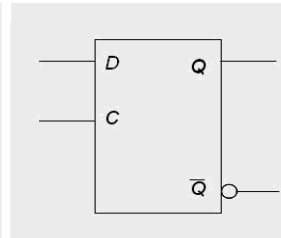
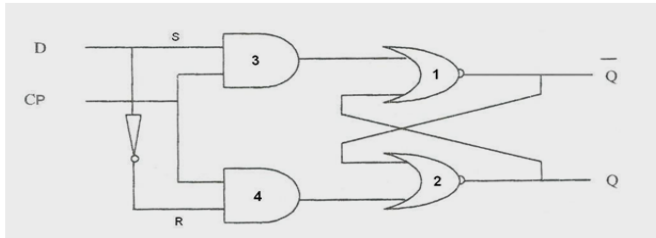


CLK (Clock)	J (Set)	K (Reset)	Q_{t+1}
0	X	X	Q_t
1	0	0	Q_t
1	0	1	0
1	1	0	1
1	1	1	$\sim Q_t$

TABLA DE ESTADOS

• B.6 Biestable D.

Un **biestable D** es un circuito digital, también denominado 'biestable de datos' porque sirve en efecto para almacenar 1 bit de datos. Posee una sola entrada D (Data) y la salida Q, obtiene el valor de la entrada D cuando la señal del reloj se encuentra activada.

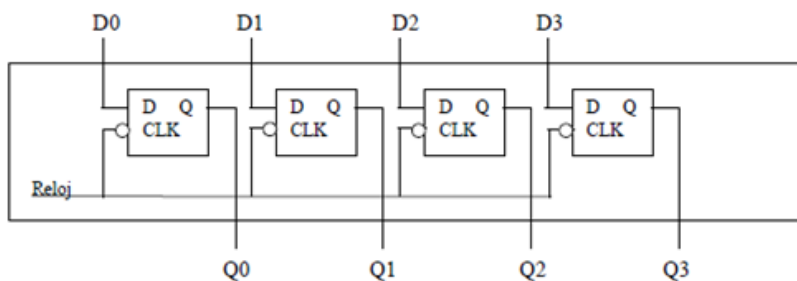


CLK (Clock)	D	Q_{t+1}
0	X	Q_t
1	0	0
1	1	1

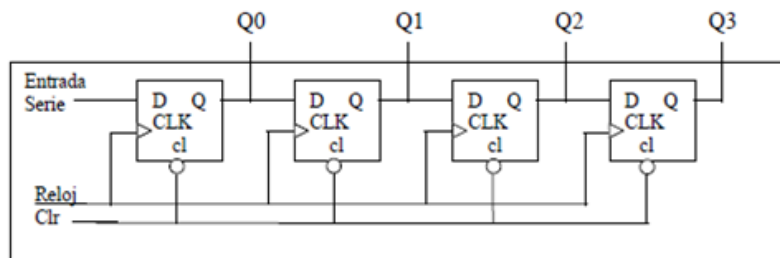
TABLA DE ESTADOS

• B.7 Registros.

Una colección de dos o más biestables D con una entrada común se conoce como un **registro**. Los registros se usan para almacenar una serie de bits relacionados, como un byte (8 bits) de una computadora. Como una celda almacena un bit de información, se desprende que un registro de n celdas puede almacenar cualquier cantidad discreta de información que contenga n bits. Un registro de n bits tiene un grupo de n flip-flops y tiene la capacidad de acumular cualquier información binaria que contiene n bits.



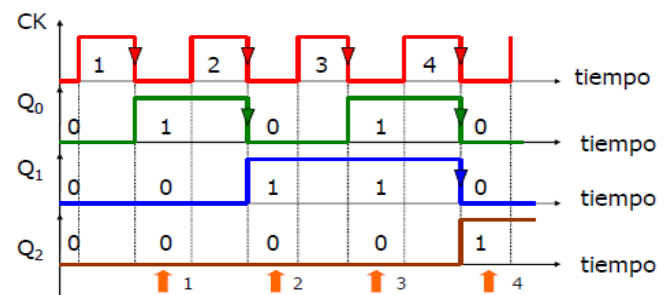
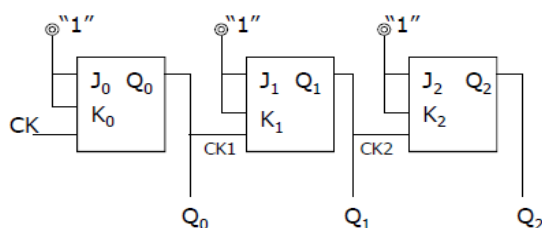
REGISTRO DE ALMACENAMIENTO DE 4 BITS



REGISTRO DE DESPLAZAMIENTO DE 4 BITS

• B.8 Contadores.

Un **contador** es un circuito secuencial que cambia de estado ante el cambio de una señal de entrada, evolucionando cíclicamente entre un número concreto de estados. La cantidad de estados por lo que pase define su módulo. El contador está formado por 3 Flip-Flops JK con sus entradas a "1" ($J=K=1$). Cada Flip-Flop tiene un bit de salida, por lo tanto con 3 Flip-Flops $\rightarrow 2^3 = 8$, podemos contar de 0 a 7 (8 números distintos).



APENDICE C – LENGUAJE ENSAMBLADOR

● C.1 Lenguaje Máquina.

El **lenguaje máquina** está compuesto por una serie de instrucciones, que son las únicas que pueden ser reconocidas y ejecutadas por el microprocesador. Cada instrucción tiene su propio y único código llamado Código de Operación (OpCode). Este lenguaje es un conjunto de números que representan las operaciones que realiza el Microprocesador a través de su circuitería interna. Estas instrucciones, por decirlo así, están grabadas en el hardware y no pueden ser cambiadas. El nivel más bajo al que podemos aspirar a llegar en el control de un microprocesador es precisamente el del lenguaje de máquina.

Mientras que con el lenguaje de máquina obtenemos un control total del microprocesador, la programación en este lenguaje resulta muy difícil y es fácil cometer errores. No tanto por el hecho de que las instrucciones son sólo números, sino porque se debe calcular y trabajar con las direcciones de memoria de los datos, los saltos y las direcciones de llamadas a subrutinas. Para facilitar la elaboración de programas a este nivel, se desarrollaron los Ensambladores y el Lenguaje Ensamblador.

● C.2 Lenguaje Ensamblador.

El programa ensamblador es el programa que realiza la traducción de un programa escrito en **lenguaje ensamblador** a lenguaje máquina. Esta traducción es directa e inmediata, ya que las instrucciones en ensamblador no son más que nemotécnicos de las instrucciones máquina que ejecuta directamente la CPU. Cada uno de los valores numéricos del lenguaje de máquina tiene una representación simbólica de 3 a 5 letras como instrucción del lenguaje ensamblador. Adicionalmente, este lenguaje proporciona un conjunto de pseudo-operaciones (también conocidas como directivas del ensamblador) que sirven para definir datos, rutinas y todo tipo de información para que el programa ejecutable sea creado de determinada forma y en determinado lugar.

◇ PASOS PARA LA CREACION DE UN PROGRAMA.

- 1) **Diseño del algoritmo:** se plantea el problema a resolver y se propone la mejor solución, creando diagramas esquemáticos utilizados para el mejor planteamiento de la solución.
- 2) **Codificación del programa:** consiste en escribir el programa en algún lenguaje de programación; en este caso específico en ensamblador, tomando como base la solución propuesta en el paso anterior.
- 3) **Traducción al lenguaje máquina:** es la creación del programa objeto, esto es, el programa escrito como una secuencia de ceros y unos que pueda ser interpretado por el procesador.
- 4) **Prueba del programa:** consiste en verificar que el programa funcione sin errores.
- 5) **Eliminación de las fallas:** consiste en solucionar las fallas detectadas en el programa durante la fase de prueba.

● C.3 Registros Internos de la CPU.

La Unidad Central de Proceso (CPU o UCP) tiene 14 **registros internos**, cada uno de 16 bits. Los primeros cuatro, **AX**, **BX**, **CX** y **DX**, son de uso general y se pueden usar también como registros de 8 bits. Es decir, AX se puede dividir en **AH** (High) y **AL** (Low). Lo mismo es aplicable a los otros tres (**BH/BL**, **CH/CL** y **DH/DL**).

◇ REGISTROS DE PROPOSITO GENERAL.

- **Registro AX (Acumulador):** empleado para operaciones aritméticas de cualquier tipo. También es usado en las instrucciones de entrada y salida y en la manipulación de cadenas de caracteres.
- **Registro BX (Base):** es el único registro de propósito general que se usa como un índice en el direccionamiento. Se usa para indicar un desplazamiento aunque también en operaciones aritméticas.
- **Registro CX (Contador):** es conocido como registro contador ya que puede contener un valor para controlar el número de veces que se repite una cierta operación o un ciclo de iteración.
- **Registro DX (Datos):** algunas operaciones de E/S requieren su uso, y las operaciones de multiplicación y división con cifras grandes suponen el funcionamiento de DX y AX trabajando juntos.

◇ REGISTROS INDICE.

Además del registro BX, existen cinco registros específicamente diseñados para el direccionamiento indexado de la memoria. Estos son los registros IP, SI, DI, SP y BP.

Los registros **SP (Stack Pointer)** y **BP (Base Pointer)** son empleados para direccionamiento en el área de la pila, la cual es una estructura especial de memoria en la cual los datos son *apilados* operando según la convención LIFO (Last In, First Out).

- **SP:** Apuntador de Pila. Proporciona un valor de desplazamiento que se refiere a la palabra actual que está siendo procesada en la pila.
- **BP:** Apuntador Base. Facilita la referencia a los parámetros de las rutinas, los cuales son datos y direcciones transmitidos vía la pila.

Los registros **SI** y **DI**, en adición al direccionamiento indexado, son empleados en las instrucciones para manipulación de cadenas donde sirven de apuntadores para los rangos de memoria manipulados.

- **SI:** Registro Índice-Fuente. Es requerido por algunas operaciones con cadenas de caracteres.
- **DI:** Registro Índice-Destino. También es requerido por operaciones con cadenas de caracteres.

El registro **IP (Instruction Pointer)** es un registro de 16 bits que contiene el desplazamiento de la dirección de la siguiente instrucción que se ejecutará. Está asociado con el registro CS en el sentido de que IP indica el desplazamiento de la siguiente instrucción a ejecutar dentro del segmento de código determinado por CS:

Dirección del segmento de código en CS:	25A40h
Desplazamiento dentro del segmento de código en IP:	+ 0412h
<u>Dirección de la siguiente instrucción a ejecutar:</u>	→ 25E52h

◇ REGISTROS DE ESTADO (BANDERAS O FLAGS).

El registro de estado, contiene el estado del microprocesador tras la ejecución de ciertas instrucciones. Este es almacenado en forma de banderas las cuales corresponden cada una a un bit. De los 16 bits disponibles sólo se emplean 9 los cuales se dividen en banderas de control y banderas de estado.

- Banderas de estado:

- **CF: Carry.** Contiene el acarreo del bit de mayor orden.
- **AF: Carry Auxiliar.** Contiene un acarreo del bit 3 en un dato de 8 bits.
- **OF: Overflow.** Indica desbordamiento en operaciones de números con signo. (1=overflow).
- **SF: Signo.** Contiene el signo resultante de una operación. (1=negativo).
- **ZF: Cero.** Indica el resultado de una operación aritmética. (1=cero).
- **PF: Paridad.** Indica paridad par o impar en una operación. (1=par).

- Banderas de control:

- **IF: Interrupción.** Indica si una interrupción externa puede ser procesada. (1=habilitada).
- **TF: Trampa.** Controla la operación en modo "paso a paso" del microprocesador (Debug).
- **DF: Dirección.** Controla la dirección de los registros índice en las instrucciones de manejo de cadenas.

◇ REGISTROS DE SEGMENTO.

El microprocesador 8086 cuenta externamente con 20 líneas de direcciones, con lo cual puede direccionar hasta 1 MB (00000h--FFFFFh) de localidades de memoria. En los días en los que este microprocesador fue diseñado, alcanzar 1MB de direcciones de memoria era algo extraordinario, sólo que existía un problema: internamente todos los registros del microprocesador tienen una longitud de 16 bits, con lo cual sólo se pueden direccionar 64 KB de localidades de memoria. Resulta obvio que con este diseño se desperdicia una gran cantidad de espacio de almacenamiento; la solución a este problema fue la **segmentación**.

La segmentación consiste en dividir la memoria de la computadora en segmentos. Un segmento es un grupo de localidades con una longitud mínima de 16 bytes y máxima de 64KB. La mayoría de los programas diseñados en lenguaje ensamblador y en cualquier otro lenguaje definen cuatro segmentos. El segmento de código, el segmento de datos, el segmento extra y el segmento de pila.

- 1) **CS (Segmento de Código):** Establece el área de memoria dónde está el programa durante su ejecución.
- 2) **DS (Segmento de Datos):** Especifica la zona donde los programas leen y escriben sus datos.
- 3) **SS (Segmento de Pila):** Permite la colocación en memoria de una pila, para almacenamiento temporal.
- 4) **ES (Segmento Extra):** Para acceder a un segmento distinto de los anteriores sin necesidad de modificar los otros registros de segmento.

• C.4 Modos de Direcccionamiento.

En la mayoría de las instrucciones en lenguaje ensamblador, se hace referencia a datos que se encuentran almacenados en diferentes medios: registros, localidades de memoria, variables, etc.

Para que el microprocesador ejecute correctamente las instrucciones y entregue los resultados esperados, es necesario que se indique la fuente o el origen de los datos con los que va a trabajar, a esto se le conoce como direccionamiento de datos.

1) Direccionamiento Directo: se conoce como directo, debido a que en el segundo operando se indica la dirección de desplazamiento donde se encuentran los datos de origen.

MOV AL, DATO → AL = contenido de la variable DATO.
MOV AX, [1000h] → AX = contenido de la dirección DS:1000h.
ADD AX, [35AFh] → AX = contenido de AX + contenido de la dirección DS:35AFh.

2) Direccionamiento Inmediato: los datos son proporcionados directamente como parte de la instrucción.

MOV AL, 3Eh → Copia en AL el valor hexadecimal 3E.
MOV CX, 10 → Copia en CX el número 10 en decimal.
ADD AX, 35AFh → AX = contenido de AX + valor hexadecimal 35AF.

3) Direccionamiento por Registro: el segundo operando es un registro, el cual contiene los datos con los que el microprocesador ejecutará la instrucción.

MOV AX, BX → Copia en AX el contenido del registro BX.
MOV AL, CH → Copia en AL el contenido del registro CH.
ADD AX, BX → AX = contenido de AX + contenido de BX.

4) Direccionamiento Indirecto por Registro: el segundo operando es un registro, el cual contiene la dirección desplazamiento correspondiente a los datos para la instrucción.

MOV [BX], AL → Dato en la dirección contenida por BX = contenido de AL.
MOV AX, [CX] → Copia en AX el dato almacenado en la dirección contenida por CX.
ADD AX, [BX] → AX = AX + dato almacenado en la dirección contenida por BX.

• C.5 Repertorio de Instrucciones.

◇ INSTRUCCIONES DE TRANSFERENCIA DE DATOS.

INSTRUCCIÓN	OPERACIÓN	COMENTARIO
MOV destino, fuente	(destino) ← (fuente)	Copia fuente en destino.
PUSH fuente	SP ← (fuente); Actualiza SP.	Carga fuente en el tope de la pila.
POP destino	(destino) ← SP; Actualiza SP.	Desapila el tope de la pila y lo almacena en destino.
PUSHF	SP ← (flags); Actualiza SP.	Apila los flags.
POPF	(flags) ← SP; Actualiza SP.	Desapila los flags.
IN destino, fuente	(destino) ← (fuente)	Carga el valor en el puerto fuente a destino.
OUT destino, fuente	(destino) ← (fuente)	Carga en el puerto destino el valor en fuente.

◇ INSTRUCCIONES ARITMETICAS.

INSTRUCCIÓN	OPERACIÓN	COMENTARIO
ADD destino, fuente	$(\text{destino}) \leftarrow (\text{destino}) + (\text{fuente})$	Suma fuente y destino y lo guarda en destino.
ADC destino, fuente	$(\text{destino}) \leftarrow (\text{destino}) + (\text{fuente}) + C$	Igual que el anterior pero suma el flag C (Carry).
SUB destino, fuente	$(\text{destino}) \leftarrow (\text{destino}) - (\text{fuente})$	Resta fuente a destino.
SBB destino, fuente	$(\text{destino}) \leftarrow (\text{destino}) - (\text{fuente}) - C$	Resta fuente y flag C (Carry) a destino.
CMP destino, fuente	$(\text{destino}) - (\text{fuente})$	Compara fuente con destino.
INC destino	$(\text{destino}) \leftarrow (\text{destino}) + 1$	Incrementa destino en 1.
DEC destino	$(\text{destino}) \leftarrow (\text{destino}) - 1$	Decrementa destino en 1.
NEG destino	$(\text{destino}) \leftarrow \text{CA2}(\text{destino})$	Realiza la negación de destino.

◇ INSTRUCCIONES DE TRANSFERENCIA DE CONTROL.

INSTRUCCIÓN	OPERACIÓN	COMENTARIO
CALL etiqueta	-	Llama a subrutina cuyo inicio es etiqueta.
RET	-	Retorna de la subrutina.
JZ etiqueta	Si $Z=1$, $IP \leftarrow (\text{memoria})$	Salta si el último valor calculado es cero.
JNZ etiqueta	Si $Z=0$, $IP \leftarrow (\text{memoria})$	Salta si el último valor calculado no es cero.
JS etiqueta	Si $S=1$, $IP \leftarrow (\text{memoria})$	Salta si el último valor calculado es negativo.
JNS etiqueta	Si $S=0$, $IP \leftarrow (\text{memoria})$	Salta si el último valor calculado no es negativo.
JC etiqueta	Si $C=1$, $IP \leftarrow (\text{memoria})$	Salta si el último valor calculado produjo carry.
JNC etiqueta	Si $C=0$, $IP \leftarrow (\text{memoria})$	Salta si el último valor calculado no produjo carry.
JO etiqueta	Si $O=1$, $IP \leftarrow (\text{memoria})$	Salta si el último valor calculado produjo overflow.
JNO etiqueta	Si $O=0$, $IP \leftarrow (\text{memoria})$	Salta si el último valor calculado no produjo overflow.
JMP etiqueta	$IP \leftarrow (\text{memoria})$	Salto incondicional a etiqueta.

◇ INSTRUCCIONES LOGICAS.

INSTRUCCIÓN	OPERACIÓN	COMENTARIO
AND destino, fuente	$(\text{destino}) \leftarrow (\text{destino}) \text{ AND } (\text{fuente})$	Operación: fuente AND destino, bit a bit.
OR destino, fuente	$(\text{destino}) \leftarrow (\text{destino}) \text{ OR } (\text{fuente})$	Operación: fuente OR destino, bit a bit.
XOR destino, fuente	$(\text{destino}) \leftarrow (\text{destino}) \text{ XOR } (\text{fuente})$	Operación: fuente XOR destino, bit a bit.
NOT destino	$(\text{destino}) \leftarrow \text{CA1}(\text{destino})$	Complemento a 1 de destino.

◇ INSTRUCCIONES DE CONTROL Y MANEJO DE INTERRUPCIONES.

INSTRUCCIÓN	OPERACIÓN	COMENTARIO
NOP	-	No hace nada.
HLT	-	Detiene la ejecución del microprocesador.
INT N	-	Salva los flags y ejecuta la interrupción por software N.
IRET	-	Retorna de la interrupción y restablece los flags.
CLI	-	Inhabilita interrupciones enmascarables.
STI	-	Habilita interrupciones enmascarables.

• C.6 Direccionamiento de Localidades de Memoria.

El 8086 posee un bus de datos de 16 bits y por tanto manipula cantidades de esta longitud (llamadas palabras). Cada palabra a la que se accede consta de dos bytes, un byte de orden alto o más significativo y un byte de orden bajo o menos significativo. El sistema almacena en memoria estos bytes de la siguiente manera: el byte menos significativo en la dirección baja de memoria y el byte más significativo en la dirección alta de memoria. El mecanismo que utiliza la CPU SX88 para almacenar bytes en memoria direccionable se conoce como **'little-endian'**. El número es almacenado en sentido invertido.

```

ORG 1000H
NUM1 DW 1000
NUM2 DW 1004

```

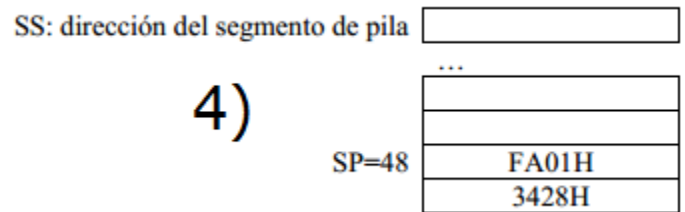
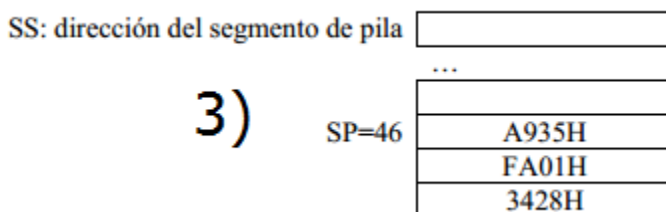
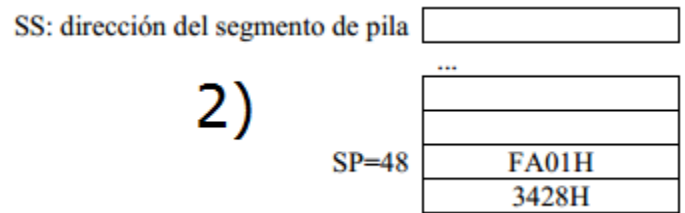
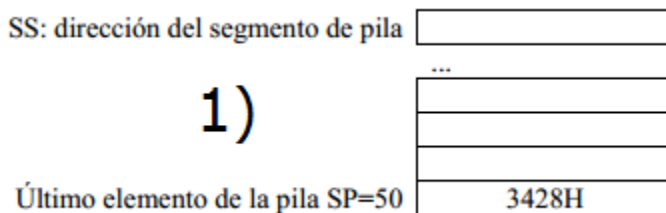
ORG 2000H			L	H
MOV	AX, NUM1	Localidad 2000	8B	00
MOV	DX, NUM1+2	Localidad 2004	8B	02
MOV	CX, NUM2	Localidad 2008	8B	04
MOV	BX, NUM2+2	Localidad 200C	8B	06

• C.7 Pila (Stack).

Una **pila** es una estructura de datos de tipo LIFO (Last In First Out, el último en entrar es el primero en salir) que permite almacenar y recuperar datos usando dos operaciones: PUSH (apilar) y POP (desapilar). Esta estructura de datos funciona como si fuera una pila de platos: cuando apilo un plato sólo puedo hacerlo en el tope de la pila y para quitar un plato sólo puedo hacerlo desde el tope de la pila. Una pila bien definida no permite acceder a otro elemento que no sea el que se encuentre en el tope; sólo una vez que haya quitado el elemento del tope, puedo sacar el que estaba inmediatamente debajo (y ningún otro más hasta que no quite éste, y así sucesivamente).

El registro SS indica la dirección de inicio de la pila, mientras que el registro SP apunta al byte siguiente al último que pertenece a la pila. La pila se caracteriza porque empieza a almacenar datos en la localidad más alta y avanza hacia abajo en la memoria, esto es, cada vez que se introduce un dato en la pila se decrementa SP, y cuando se saca, se incrementa. Los datos que se introduzcan en la pila siempre son de dos bytes. Ejemplo:

- 1) Los registros AX y BX contienen FA01H y 35A9H. Suponemos que el registro SP contiene 50 bytes.
- 2) **PUSH AX:** Se introduce en la pila AX → se decrementa SP y se introduce el dato.
- 3) **PUSH BX:** Se introduce en la pila BX → se decrementa SP y se introduce el dato.
- 4) **POP BX:** Se extrae de la pila un dato y se introduce en BX ⇒ se extrae el dato y se incrementa SP.



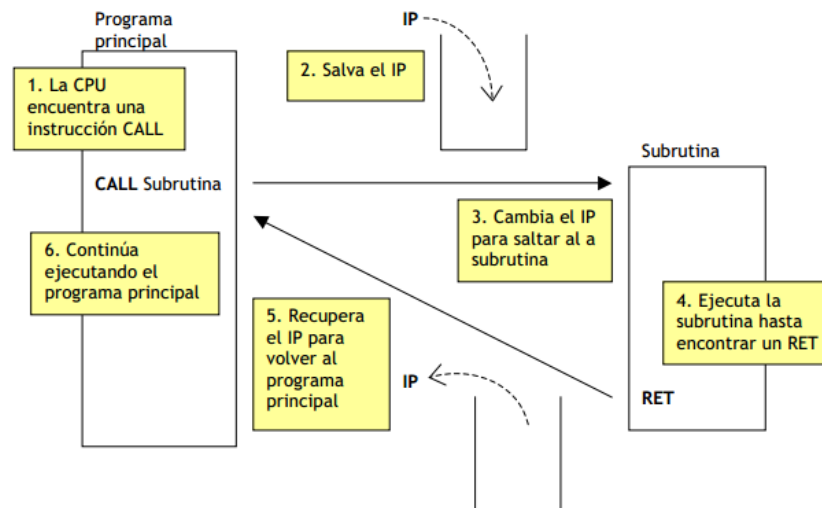
El simulador del MSX88 inicializa el SP (Puntero de Pila) apuntando a la pila en la dirección 8000H.

• C.8 Subrutinas.

Al desarrollar sistemas relativamente grandes, una de las técnicas más utilizadas es la de dividir el mismo en distintos módulos lo más independientes posibles entre sí, que trabajando en conjunto logran proveer la funcionalidad total requerida por el sistema. La idea es que, subdividiendo el sistema en partes más pequeñas, cada parte es más sencilla de desarrollar que el sistema en su totalidad. La unidad de descomposición modular básica se conoce como **subrutina**. Una subrutina (también conocida como función, método, procedimiento o subprograma, dependiendo del lenguaje de programación) es una porción de código dentro de un programa más grande, que realiza una tarea específica y es relativamente independiente del resto del código.

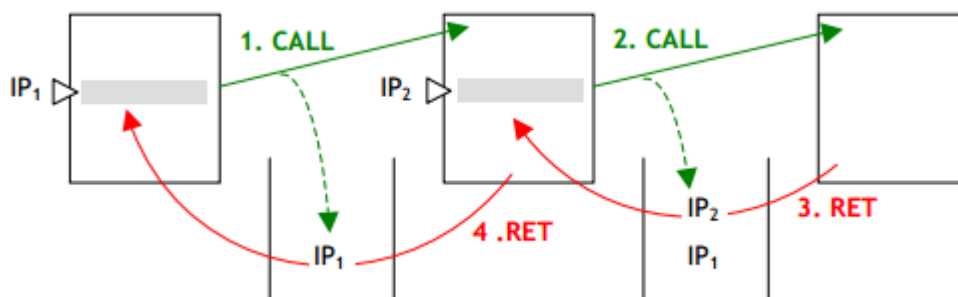
Podríamos pensar que una subrutina se implementa simplemente como un salto dentro del código del programa... pero hay un problema: el salto sabe ir pero no volver. Es decir, con el salto voy a donde comienza la subrutina, pero después, desde la subrutina, ¿cómo sé a qué punto del programa principal volver? (una subrutina puede ser llamada muchas veces en distintos puntos del programa, incluso desde otras subrutinas). Para resolver este problema, el procesador cuenta con dos instrucciones especiales **CALL** (llamada a subrutina) y **RET** (retorno de subrutina). Ambas instrucciones hacen uso de la pila.

Cuando se realiza una llamada a una subrutina con **CALL**, esta instrucción automáticamente guarda el contenido del registro IP en la pila y salta al comienzo de la subrutina. En algún punto de la subrutina el procesador se encontrará con la instrucción **RET** que, al ejecutarla, desapilará el valor del IP guardado en la pila de modo que cuando el procesador avance a la siguiente instrucción (incremente IP), estará de vuelta en la instrucción siguiente a donde se hizo la llamada.



Hay que asegurarse de que al ejecutar la instrucción **RET**, el dato que se encuentra en el tope de la pila sea la dirección IP de retorno, de lo contrario se desapilará un valor incorrecto y el procesador irá a ejecutar instrucciones a otra área de memoria (lo que terminará, probablemente, con que el programa se cuelgue). Entonces, siempre que una subrutina apile datos (**PUSH**) debe recordar desapilarlos (**POP**).

Se pueden 'anidar' las llamadas a subrutinas, es decir tener un programa que llama a una subrutina, que llama a otra subrutina, que llama a otra, etc... Lo que sucederá en ese caso es que se apilarán sucesivamente varios valores de IP. Por el mecanismo de la pila, el último IP en entrar en la pila será el primero en desapilarse, de modo que esto nos permitirá volver de una subrutina al código que nos llamó sin problemas:



• C.9 Pasaje de Parámetros.

◇ PASAJE DE PARAMETROS POR REGISTROS.

- POR VALOR: Son solo parámetros de entrada. Se pasa el valor de una variable al procedimiento y este no puede ser modificado. Por ejemplo: MOV AX, NUM1.
- POR REFERENCIA: Se pasa la dirección de la variable y no su valor. Se utiliza la palabra OFFSET para referencia la dirección de la variable. Por ejemplo: MOV AX, OFFSET NUM1.

El programa principal escribe los datos a pasar a la subrutina en uno o varios registros. Luego, la subrutina lee esos mismos registros para obtener los datos. A su vez, la subrutina puede devolver datos al programa principal utilizando el mismo mecanismo. Lo más importante en este caso es que el programa principal y la subrutina respeten una misma interfaz. Veamos un ejemplo: la interfaz de una subrutina que devuelva el mayor de dos números podría ser: recibe los números en AX y BX y devuelve el mayor entre ambos en AX. Entonces, el invocador deberá asegurarse de copiar en AX y BX los valores antes de llamar a la subrutina. Cuando la subrutina arranca supone que AX y BX tienen cargados los números.

```
ORG 1000H
NUM1      DW    100Ah      ; Se definen los dos números a comparar.
NUM2      DW    F53Ch      ;
ELMAYOR   DW    ?          ; Variable para almacenar el mayor.

ORG 2000H
MOV    AX, NUM1      ; Se copian los números en AX y BX
MOV    BX, NUM2      ; como pide la subrutina.
CALL   MAYOR
MOV    ELMAYOR, AX   ; Guarda el mayor en la variable 'ELMAYOR'.
END

ORG 3000H
MAYOR: CMP    AX, BX      ; Comparar AX con BX.
        JG     FINSUB      ; Si AX es el mayor, terminar.
        MOV    AX, BX      ; BX es el mayor, lo guardamos en AX.
FINSUB: RET              ; Retornar al programa principal.
```

◇ PASAJE DE PARAMETROS POR PILA.

Se apilan los datos (PUSH) antes de invocar a la subrutina. La subrutina leerá los datos desde la pila y operará sobre ellos. Se puede utilizar la pila tanto para enviar datos a una subrutina como para recibir datos desde ésta o bien sólo recibir por la pila y devolver por registros. Si reescribimos la subrutina anterior para que ahora utilice la pila, la nueva interfaz podría ser: la subrutina Mayor recibe dos números en la pila y devuelve en AX el mayor de ambos.

```
ORG 1000H
NUM1      DW    100Ah      ; Se definen los dos números a comparar.
NUM2      DW    F53Ch      ;
ELMAYOR   DW    ?          ; Variable para almacenar el mayor.

ORG 2000H
PUSH    NUM1          ; Se copian los números en AX y BX
PUSH    NUM2          ; como pide la subrutina.
CALL    MAYOR
MOV     ELMAYOR, AX   ; Guarda el mayor en la variable 'ELMAYOR'.
END

ORG 3000H
MAYOR: PUSH    BP      ; Salvar BP.
        MOV     BP, SP   ; BP = SP.
        MOV     AX, SS:[BP+4] ; Obtener uno de los parámetros.
        CMP     AX, SS:[BP+6] ; Comparar con el otro valor.
        JG      FINSUB   ; Si AX es el mayor, terminar.
        MOV     AX, SS:[BP+6] ; BX es el mayor, lo guardamos en AX.
FINSUB: POP     BP      ; Restaurar BP.
        RET              ; Retornar al programa principal.
```

• C.10 Directivas para la definición de datos.

Un programa en lenguaje ensamblador no está comprendido únicamente por instrucciones del lenguaje de máquina sino que existen otro tipo de instrucciones que le indican al ensamblador como realizar algunas tareas. Un hecho notable cuando se trata del uso de variables es que en las instrucciones del lenguaje de máquina no se hace referencia a ellas por un nombre sino por su dirección en la memoria. Para evitarnos la tarea de recordar que tal variable está almacenada en una dirección particular (que, de hecho, podría cambiar a lo largo del desarrollo de un programa, forzándonos a revisar el programa entero para realizar los ajustes necesarios), en lenguaje ensamblador existen instrucciones para definir variables. La sintaxis para definir una **variable** es la siguiente:

nombre_variable especificador_tipo valor_inicial

El nombre de la variable debe comenzar con una letra o un underscore (_) seguido por números, letras o underscores. El tipo indica el tamaño que ocupa en memoria dicha. El valor inicial puede no especificarse, usando el carácter '?'. Los valores numéricos se interpretan en decimal, a menos que terminen con una letra 'h', que en cuyo caso se interpretarán como valores en hexadecimal. Además, como los números deben comenzar con un dígito decimal, en el caso del A000h, se antepone un cero para evitar que se la confunda con una variable llamada A000h.

Especificador	Tipo	Tamaño
DB	Byte	1 byte
DW	Word	2 bytes
DD	Double Word	4 bytes

A veces resulta bueno poder definir valores **constantes**. Esto se hace del siguiente modo: nombre **EQU** valor. Debe notarse que en este caso el ensamblador reemplazará cualquier ocurrencia de 'nombre' por el valor indicado, pero que dicho valor no va a ocupar ninguna dirección de memoria, como lo hacen las variables.

Es posible extender la definición de variables para incluir la idea de **tablas**, de la siguiente manera:

nombre_variable especificador_tipo valores

En la definición anterior, *valores* es una lista de datos del mismo tipo de la variable separados por coma:

tabla **DB** 1, 2, 4, 8, 16, 32, 64, 128

Esto genera una tabla con los ocho valores especificados, uno a continuación del otro. Esto se puede ver como un arreglo de ocho bytes pero en el que se inicializaron sus celdas con dichos valores. Por otro lado, si quisiéramos definir algo equivalente a un string, podemos aplicar la misma idea de la tabla anterior, en donde en cada celda se almacenaría cada carácter del string. Sin embargo, escribir los códigos ASCII de cada carácter no simplifica mucho las cosas, así que existe una sintaxis alternativa:

string **DB** "Esto es un String."

Si quisiéramos definir una tabla en la que los datos que contienen son iguales o cumplen algún patrón repetitivo, es posible utilizar el modificador DUP en la lista de valores, de la siguiente manera:

cantidad **DUP** (valores)

En este caso, cantidad indica la cantidad de veces que se repiten el o los valores indicados entre paréntesis. Por ejemplo, para definir un arreglo de 20 palabras, inicialmente conteniendo 1234h y 4321h alternadamente, se le indica al ensamblador lo siguiente:

cantidad **EQU** 10
arreglo **DW** cantidad DUP (1234h, 4321h)

Otra cuestión que se obvió hasta el momento es que dirección se le asigna a cada variable. El ensamblador lleva cuenta de las variables e instrucciones que va procesador y de la dirección que le corresponde a cada una, dependiendo de una dirección inicial y del tamaño correspondiente. Existe una directiva del ensamblador, llamada ORG, que permite cambiar sobre la marcha la dirección a partir de la cual se colocarán las cosas que estén a continuación de la misma. La sintaxis de la misma es: **ORG** dirección.