Contenido

UNIDAD 1. COMPUTADORAS DIGITALES. 2
¿A qué llamamos computadora? 2
¿Cómo podemos describirla? 2
¿A que nos referimos con arquitectura y organización? 4
Estructura y Funcionamiento de una computadora 5
Estructura de la CPU 5
Funciones básicas de una computadora 5
Modelo Von Neumann 5
UNIDAD 2. Información en una computadora 6
¿Cómo se representa la información en las computadoras? 6
¿Cómo se representa la información? 7
Sistemas de codificación binarios 7
¿Cómo identifica la computadora el tipo de dato? 7
Sistemas de Numeración 7
¿Cómo se representan estos sistemas? 8
Sistema decimal. 8
Sistema binario. 8
Sistema hexadecimal 9
Teorema fundamental de la numeración 9
Conversión entre sistemas 10
Representación binaria (Coma fija) 11
Modulo y signo 12
Complemento a 1 (C-1) 12
Complemento a 2 (C-2) 12
Exceso a 2 elevado a la n-1 13
Operaciones aritméticas 13
Suma binaria 13
Overflow y Carry 13
Punto Flotante 14
Rango y precisión 15
Suma y resta en punto flotante 15
Procesamiento de la coma flotante 16
Observaciones de los números reales 17
Unidades de medida de la información 17
Representación de texto 18
TEMA 3. Circuitos combinacionales 19
Símbolos y notación 19

Postulados y teoremas 20
Funciones lógicas 20
Teorema de DeMorgan 21
Circuitos combinacionales y secuenciales 21
Componentes combinacionales MSI 22
Sumadores 22
Decodificadores 22
Multiplexores 23
Demultiplexor 24
Resumiendo: 24
Funciones combinacionales con MSI 25
Matrices lógicas programables (PLAs) 26
Memorias de solo lectura (ROMs) 26

UNIDAD 1. COMPUTADORAS DIGITALES.

¿A qué llamamos computadora?

Es un dispositivo electrónico programable, diseñado para aceptar datos de entrada y realizar operaciones sobre ellos (organizadas en una secuencia lógica y predeterminada por un algoritmo), para elaborar resultados que se puedan obtener como salidas.

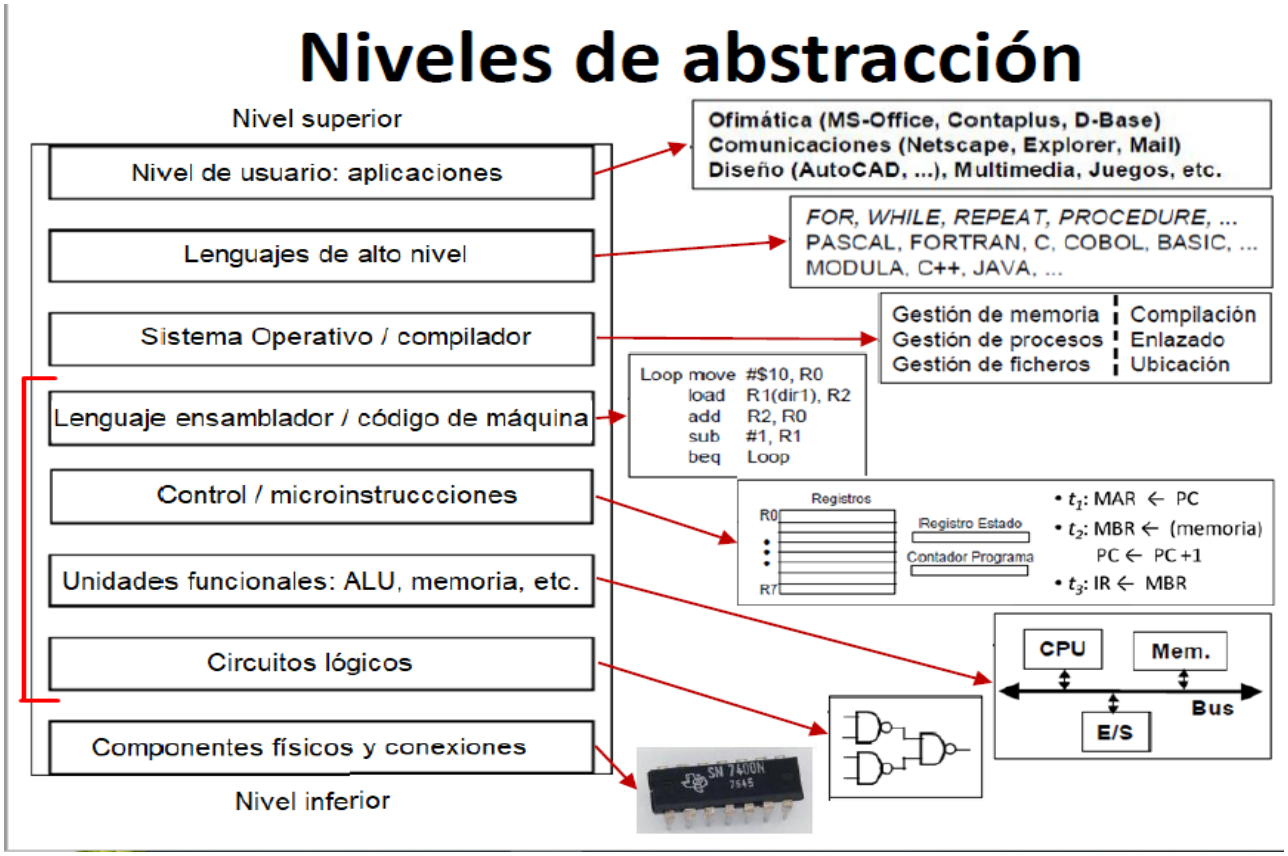
Según P. Quiroga (Arquitectura de Computadoras)

¿Cómo podemos describirla?

Si queremos describir una computadora debemos ser específicos, ya que existen una gran variedad de sistemas:

- Computadoras Personales (PC)
- Estaciones de trabajo (workstations)
- Supercomputadoras / mainframes
- Sistemas embebidos (de distintos niveles de complejidad)

De igual forma, existen Niveles de abstracción en describir una computadora, ya que una persona que no esté muy interesada en el tema tendrá una **idea básica,** sin embargo, los niveles de abstracción van desde el más básico (Nivel del usuario), hasta el más técnico (componentes físicos y conexiones), mientras más se descienda en estos niveles la abstracción va desapareciendo y comienza a formarse la estructura física de la computadora.



(En la materia veremos los niveles marcados en rojo)

Nivel del lenguaje de ensamblador / código de maquina:
Cada procesador posee un conjunto de instrucciones o lenguaje de máquina que se denomina juego o repertorio de instrucciones.
Deben tratar con circuitos tales como la estructura de los registros y la transferencia de datos entre ellos.
Nivel de control / microinstrucciones:
Las instrucciones decodificadas por la unidad de control generan una serie de señales secuenciales, se interconectan a cada componente del procesador.
Este nivel se expresa a través de "microinstrucciones", en un lenguaje RTL (de las siglas en inglés; Lógica de Transferencia de Registro).

Nivel de unidades funcionales: ALU[1], memoria, etc:
Las transferencias de registros y las demás operaciones implementadas por la unidad de control mueven información desde y hacia unidades funcionales: ALU, Registros de uso general y particular, etc.

Nivel de Circuitos lógicos, transistores y cables:

[1] ALU: (Arithmetic Logic Unit) es un componente de la CPU que realiza operaciones aritméticas y lógicas en datos binarios. Es responsable de realizar cálculos como sumas, restas, multiplicaciones, divisiones y operaciones lógicas como AND, OR, NOT y XOR.
Es como la calculadora de una computadora. Realiza operaciones matemáticas como sumas y restas, y también operaciones lógicas como comparar números para ver si son iguales o si uno es mayor que otro. Es un componente importante que permite que la computadora realice cálculos y tome decisiones basadas en esos cálculos.

Los circuitos lógicos se utilizan para construir las unidades funcionales y los transistores se usan para construir los circuitos lógicos.
Es el nivel en el que el hardware se hace más visible.

¿A que nos referimos con arquitectura y organización?

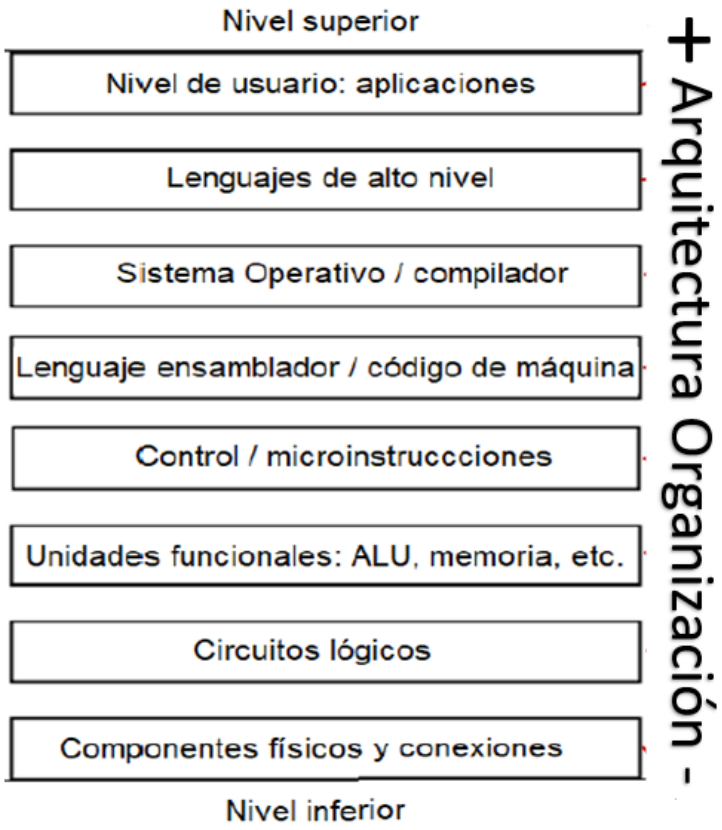
Cuando hablamos de arquitectura, nos referimos a los atributos del sistema que son visibles para un programador, por ejemplo: Conjunto de instrucciones. Numero de bits usados para representar tipos de datos (numéricos o caracteres), técnicas de direccionamiento de memoria, mecanismo de E/S[2], etc.

- Una cuestión de diseño arquitectónico es decidir si la computadora tendrá instrucciones de multiplicar, por ejemplo: La familia x86 de Intel comparte la misma arquitectura básica. Esto asegura la compatibilidad de código (al menos entre programas antiguos).

La organización de una computadora refiere a cómo son implementados esos atributos, como, por ejemplo: Detalles de los hardware transparentes al programador, señales de control, interfaces entre el computador y periféricos, tecnologías de memoria, frecuencia del reloj, etc.

- Una cuestión de organización es decir si la instrucción se implementará por una unidad especializada o mediante el uso iterativo de la unidad sumadora, por ejemplo: La organización cambia entre diferentes versiones de una misma familia (80286,80386,80486…).

Las cuestiones de arquitectura se harán más visibles en los niveles de abstracción superiores, mientras que las cuestiones de organización se verán con más claridad en los niveles inferiores.



[2] E/S: Se refiere a Dispositivos de Entrada/Salida, estos dispositivos son los que permiten que el sistema gestionado recopile, almacene y transmita datos.

Estructura y Funcionamiento de una computadora

Una computadora es una entidad que interactúa con su entorno externo, también se dice que es un sistema complejo constituido por componentes interrelacionados, organizados jerárquicamente, y en cada nivel jerárquico, se pueden distinguir:

- **Estructura**: Es el cómo se interconectan los componentes, por ejemplo: Memoria Ram
- **Función:** Es la operación de cada componente individual como parte de la estructura, siguiendo el ejemplo de la memoria ram; su función es almacenar datos temporales mientras están en uso, los cuales estos son eliminados cuando ya no están en uso, todo para no sobrecargar la memoria principal.

Existen cuatro componentes estructurales principales:

- **Unidad Central de Procesamiento**: (Por sus siglas en inglés: CPU, es el cerebro de la maquina)
- **Memoria principal**: Es la encargada de almacenar datos, y su tecnología ha mejorado considerablemente con los años, (Diskettes, CDs, HDD, SSD…)
- **Entrada/Salida**: Transfieren datos entre la computadora y el entorno externo
- **Sistema de interconexión**: Es el mecanismo encargado de la conexión entre todo lo anterior.

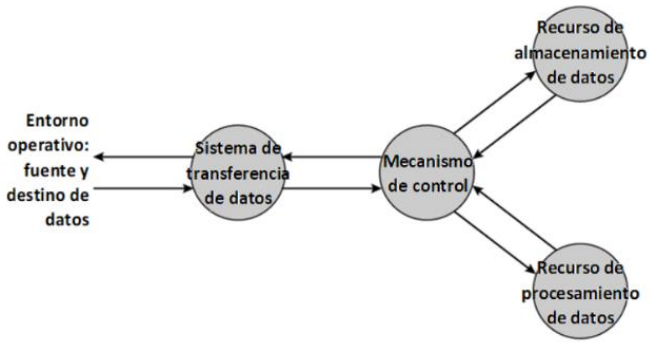
Estructura de la CPU

La CPU a su vez también tiene una estructura interna de los siguientes componentes:

- Unidad de control: Su función principal es obtener y ejecutar instrucciones de la memoria.
- Unidad aritmético-lógica (Por sus siglas en inglés, ALU)
- Registros: Es donde se almacenan los valores de datos, comandos, instrucciones o estados binarios que ordenan qué y cómo debe ordenarse un dato. Existen muchos tipos de registros
- Interconexiones CPU: Es el "sistema de interconexión" de la CPU

Funciones básicas de una computadora

1. Procesamiento de datos: Una computadora tiene que ser capaz de procesar datos, en general los datos tienen distintas formas (números, letras, etc)
2. Almacenamiento de datos: Se puede dividir en Corto plazo (datos almacenados de manera temporal) y Largo plazo (Se almacenan los datos en archivos para ser utilizados cuando se necesite)
3. Transferencia de datos: Debe ser capaz de transferir datos de y hacia el mundo exterior, y se le llama comunicación de datos a la transferencia entre largas distancias.
4. Unidad de control: Es ejercido por instrucciones a la computadora, también la unidad de control es la que gestiona los recursos de la computadora.



Modelo Von Neumann

En este modelo de arquitectura de la computadora, su funcionamiento se basa en el concepto de que siempre habrá un programa almacenado en memoria. En la memoria principal se almacenan:

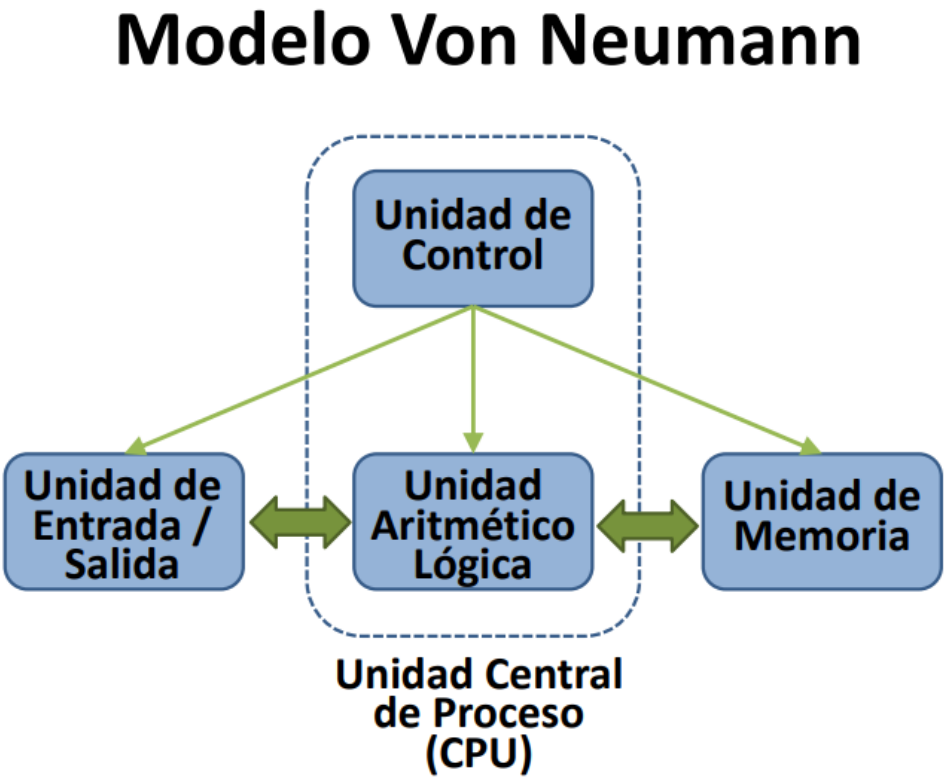
Instrucciones: Son cadena de bits que controla el funcionamiento del computador

Datos: Son procesados y generan cada instrucción.

La ejecución de las instrucciones es secuencial, alguna de sus características es:

Es implícito, y viene determinado por el orden en que han sido almacenadas en la memoria. Solo puede ser alterado por intrusiones específicas de salto.

El contador de programa o PC (Por sus siglas en ingles Program Counter o también llamado Instruction Pointer), se encarga de indicar en cada instante cual será la siguiente instrucción a ejecutar.



UNIDAD 2. Información en una computadora

El objetivo principal de una maquina es procesar información ejecutando programas. Para esto, requiere dos tipos de información:

1. Las instrucciones que forman el programa.
2. Los datos con los que debe operar ese programa.

Un aspecto muy importante es la informática es como representar y como materializar o registrar físicamente la información.

Es importante comprender que la computadora transforma la información externa a la computadora con procesos internos en patrones de bits fácilmente almacenables y procesables por los elementos internos de la misma.

¿Cómo se representa la información en las computadoras?

Textos

- BCD de 6 bits (Binary Coded Decimal)
- EBCDIC (Extended Binary Coded Decimal Interchange Code)
- ASCII (American Estándar Code for Information Interchange)
- UNICODE

Datos Numéricos

Enteros

1. Dígitos decimales codificados en Binario (BCD puede ser Empaquetado o Desempaquetado)

2. Representación binaria Coma fija (Signo y Magnitud, Complemento a 1, Complemento a 2, Exceso a 2 elevado a N1.
   Reales
3. Coma Flotante.
4. Notación exponencial.
5. Normalización IEE754.

Sonidos

WAV, MIDI, MP3

Imágenes

Mapa de bits: JPEG, GIF, PNG

Mapa de vectores: DXF, IGES, EPS, Truetype

¿Cómo se representa la información?

La información es representada en base a cadenas de símbolos, un alfabeto en sí, no es más que un conjunto fijado por acuerdo cultural, de símbolos elementales en base a los cuales se forma la información.

Sabiendo que cualquier alfabeto se fija arbitrariamente, no es necesario que el alfabeto (o lenguaje) que usa una maquina en su interior sea el mismo que utiliza el hombre que la ha construido y maneja, basta con que la transformación de símbolos internos a externos o viceversa se efectúe de una manera eficiente, sencilla y de ser posible, automáticamente por la propia máquina.

Sistemas de codificación binarios

Cuando queremos transformar los símbolos de un lenguaje conocido a otro que solo tiene dos símbolos, diremos que tenemos un sistema de codificación binario, este alfabeto es utilizado para comunicaciones técnicas nada más, ya que es complicado lograr que los dispositivos físicos puedan diferenciar de forma confiable más de dos estados que están claramente separados. Por esa razón se recurren a dispositivos físicos biestables (los que tienen dos estados físicos diferenciados en forma clara y estable), como, por ejemplo:

- Corriente eléctrica: pasa o no pasa corriente, es económico y concede un amplio margen de tolerancia
- Intensidad de la luz: luz apagada o luz encendida
- Sentido de la magnetización: magnetización norte sur y su contraria.

Internamente la computadora representa la información en unos y ceros, transformando la información como nosotros la conocemos mediante códigos binarios, tanto en la entrada como en la salida de la misma.

¿Cómo identifica la computadora el tipo de dato?

No lo identifica, la memoria almacena los datos como Patrones de bits, es responsabilidad de los dispositivos de E/S o de los programas interpretar un patrón de bits como un número, texto o algún otro tipo de datos. Los datos se codifican cuando entran a la computadora y se decodifican cuando se presentan al usuario.

Sistemas de Numeración

Las computadoras y el procesamiento de datos requieren algún conocimiento de los sistemas numéricos, ya que estos constituyen la base de todas las operaciones de una computadora. Los diferentes sistemas

numéricos difieren entre sí en cuanto a la disposición y al tipo de símbolos que utilizan, los sistemas más utilizados son:

- Decimal
- Binario
- Hexadecimal

¿Cómo se representan estos sistemas?

Todo sistema de numeración en base ***b*** utiliza para representar los números un alfabeto compuesto por ***b*** símbolos.

Así todo número se expresa por un conjunto de cifras, contribuyendo cada una de ellas con un valor que depende de dos factores:

- La cifra en si (Su valor absoluto)
- La posición que ocupe dentro del número (Su valor posicional)

Como regla general podemos decir, que la posición del digito del extremo derecho es la de menor valor, y el digito que la ocupa se denomina “digito menos significativo”

Sistema decimal.

Es el más importante en el desarrollo de la ciencia y la matemática siendo el más conocido a nivel mundial, este sistema utiliza diez símbolos (0,1,2,3,4,5,6,7,8,9), denominado generalmente como “cifras decimales”, siendo la base de este sistema ***b***=10.

Por ejemplo: el número decimal 3.278,52 se puede obtener como la suma de:

$$\mathbf{3.278,52=3.000+200+70+8+0,5+0,02}$$

O también se puede escribir como:

$$\mathbf{3.278,52 = 3x10^3 + 2x10^2 + 7x10^1 + 8x10^0 + 5x10^{-1} + 2x10^{-2}}$$

Cada posición tiene un peso y un nombre especifico, en el ejemplo la posición 2 tiene un peso $b^2 = 100$ centenas.

Sistema binario.

Usa dos símbolos diferentes: 0 y 1 denominados bits, la base de este sistema es b=2, se puede utilizar una plantilla para poder saber qué posición estamos, empezando con 2^0 y aumentando en una potencia de dos hacia la izquierda o disminuyendo hacia la derecha, siendo la parte entera las potencias positivas y la parte fraccionaria las potencias negativas:

Por ejemplo: el número binario 101101,11 se puede obtener como:

$101101{,}11 = 1x2^5 + 0x2^4 + 1x2^3 + 1x2^2 + 0x2^1 + 1x2^0 + 1x2^{-1} + 1x2^{-2}$

Realizamos todas las operaciones quedando:

1x32 + 0 + 1x8 + 1x4 + 0 +1x1 + 1x0,5 + 1x0,25 = 45,75 (En el sistema decimal)

Sistema hexadecimal

Los números binarios de gran magnitud consisten en largas series de ceros y unos, que son difíciles de interpretar y manejar, como un medio conveniente para representar esos números binarios de gran magnitud se utiliza el sistema número hexadecimal, de base ***b***=16, utilizando 16 símbolos diferentes (0,1,2,3,4,5,6,7,8,9, A,B,C,D,E,F) donde cada digito hexadecimal representa cuatro dígitos binarios, como se puede ver en la siguiente tabla:

Decimal	Hexadecimal	Binario
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Por ejemplo, el número hexadecimal 2CA es igual a:

$2CA = 2x16^2 + 12x16^1 + 10x16^0$

$=2x256 + 12x16 + 10x1$

$= 512 + 192 + 10 = 714$ (en sistema decimal)

Teorema fundamental de la numeración

El teorema fundamental de la numeración relaciona una cantidad expresada en cualquier sistema de numeración, con la misma cantidad expresada en el sistema decimal, usando la formula:

$$N = \sum_{i=-d}^{n} (\text{dígito})_i * (\text{Base})^i$$

Una determinada cantidad, que denominaremos N, se puede expresar de la siguiente manera:

Donde:

Base (***b***) = (Base de cada sistema numérico, por ej., 10 en el decimal)

i=Posición respecto de la coma

d=Nro. De dígitos a la derecha de la coma

n=Nro. de dígitos a la izquierda de la coma, menos 1

Digito=Cada uno de los que componen el número.

Conversión entre sistemas

Binario a decimal: Se suman los productos de todos los valores posicionales por el número que ocupa la posición.

Ej.: El número binario 1101,01

$$\begin{aligned} & \mathbf{1\quad 1\quad 0\quad 1,\quad 0\quad 1} \\ &= 1x2^3 + 1x2^2 + 0x2^1 + 1x2^0 + 0x2^{-1} + 1x2^{-2} \\ &= 8 + 4 + 2 + 1 + 0 + 0{,}25 \\ &= \mathbf{13{,}25}\ \text{(decimal)} \end{aligned}$$

Hexadecimal a Decimal: se multiplica el número representado por el valor posicional que le corresponde, y se suman los resultados.

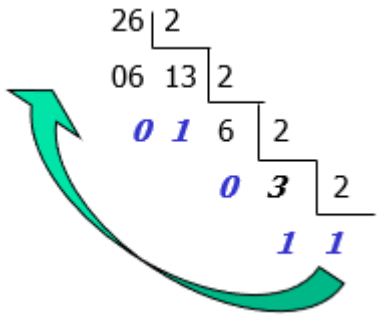
Ej.: El numero hexadecimal AE1B

$$\begin{aligned} & \mathbf{A\quad E\quad 1\quad B} \\ &= 10x16^3 + 14x16^2 + {}^1x16^1 + 11x16^0 \\ &= 40960 + 3584 + 16 + 11 \\ &= \mathbf{44571}\ \text{(decimal)} \end{aligned}$$

Decimal a binario:

Se divide el número que se quiere convertir por la base del sistema (b=2), siguiendo los siguientes pasos:

- Los resultados que se obtengan en el cociente deben seguir dividiéndose hasta que este resultado sea menor que la base
- Si la división nos da como resultado un número con decimal, se toma la parte entera únicamente
- Si el cociente da como resultado un numero impar se coloca un 1 y si es par se coloca un 0
- Los resultados de las divisiones de escribirán de derecha a izquierda
- Para convertir la parte fraccionaria de un numero decimal el proceso es diferente; se multiplica por 2 la parte fraccionaria y del resultado se tomará la parte entera para el numero binario y la parte fraccionaria para seguir dividiendo, seguiremos hasta que la parte fraccionaria nos dé como resultado 0.



Ej. Numero decimal: (26)10 = (11010)2

Decimal a hexadecimal:

Se divide el número que se quiere convertir por la base del sistema (b=16), siguiendo los siguientes pasos:

Los resultados que se obtengan en el cociente deben seguir dividiéndose hasta que el resultado sea menor que la base

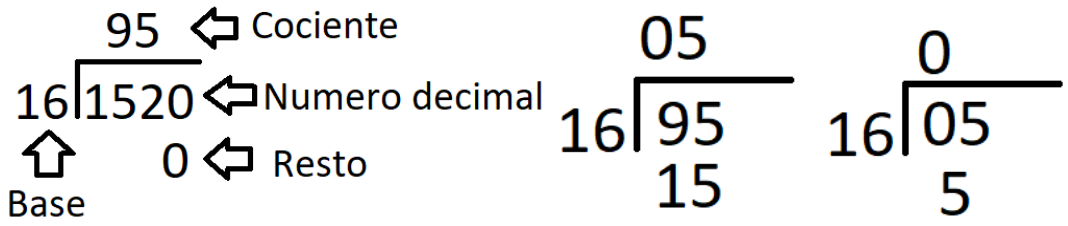
Si algún resultado nos da una fracción tomamos la parte entera

Los restos de cada división serán nuestros números decimales que luego buscaremos en nuestra tabla su equivalente en hexadecimal

El cociente se vuelve a dividir por la base, hasta que obtengamos 0 de cociente

El número hexadecimal se forma con los restos, de derecha a izquierda

Por ej.: el número decimal 1520



Nuestros restos dieron 5,15 y 0 (Derecha a izquierda) buscando los equivalentes en nuestra tabla, nos da como resultado que el número decimal 1520 en hexadecimal seria 5F0.

Binario a Hexadecimal:

Se divide el número binario en grupos de cuatro dígitos binarios, comenzando desde la derecha y se reemplaza cada grupo por el correspondiente símbolo hexadecimal, si el grupo de la extrema izquierda no tiene cuatro dígitos se deben agregar ceros hasta completar 4 dígitos

Ej. *Número Binario*: **111110011011010011**
= 0011 / 1110 / 0110 / 1101 / 0011
= 3 E 6 D 3
= **3E6D3** (hexadecimal)

Hexadecimal a binario: De una forma muy parecida a la anterior, se reemplaza cada símbolo por el correspondiente grupo de cuatro dígitos binarios y se descartan los ceros innecesarios.

Ej. *Número Hexadecimal*: **6 C 4 F 2 E**
= 0110 / 1100 / 0100 / 1111 / 0010 / 1110
= **110110001001111100101110** (binario)

Representación binaria (Coma fija)

Existen varios métodos de representación binaria para enteros con signo:

- Modulo y signo
- Complemento a 1
- Complemento a 2
- Exceso a 2 elevado a n-1

Modulo y signo

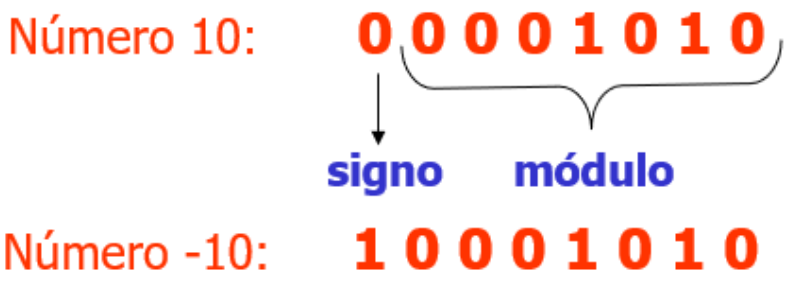
El bit que está situado más a la izquierda representa el signo, y su valor será 0 para positivo y 1 para negativo. El resto de los bits (n-1) representan el módulo del número.

Este sistema posee la ventaja de tener un rango simétrico y la desventaja de tener dos representaciones para el 0:

+0 = 0 0000000

-0 = 1 0000000

Ej.: $(10)_{10}$



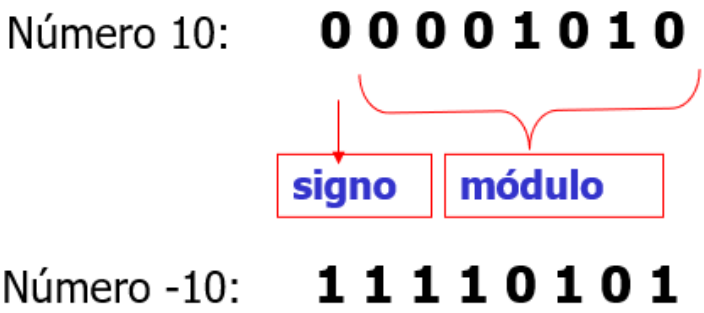
Complemento a 1 (C-1)

El bit que está situado más a la izquierda representa el signo y su valor será de 0 para positivo y 1 para negativo. Para los números positivos el resto de los bits (n-1) representan el módulo del número.

El negativo de un numero positivo se obtiene complementando todos sus dígitos, incluido el bit de signo (Como se haría en teoría de conjuntos).

Este sistema posee la misma ventaja y desventaja del anterior

Ej. $(10)_{10}$



Complemento a 2 (C-2)

Es muy parecido al anterior con única diferencia en la forma de obtener el numero negativo, que se obtiene en dos pasos:

Primer paso: Se complementa el numero positivo en todos sus bits, incluido el bit de signo (Es decir el complemento a1)

Segundo paso: Al resultado obtenido en el primer paso se le suma 1, ignorando el ultimo acarreo en caso de que haya uno, quedando:

Número -10: **1 1 1 1 0 1 0 1**

+ 1

1 1 1 1 0 1 1 0

Exceso a 2 elevado a la n-1

Este método no utiliza ningún bit para el signo, con lo cual todos los bits representan un módulo o valor que corresponde al número representado más el exceso.

Exceso para n bits: 2^{n-1}

Exceso para 8 bits: $2^{8-1} = 2^7 = 128$

Para saber si se produce un desbordamiento, debemos sumar el número entero al resultado de la potencia para n bits, luego representar ese número en código binario, si como resultado de las operaciones, se obtiene un numero fuera de los límites, se ha producido un desbordamiento:

Número 12 → (12 + 128) = 140 : **1 0 0 0 1 1 0 0**

Número 25 → (25 + 128) = 153 : **1 0 0 1 1 0 0 1**

Longitud de palabra	Límite superior N (max)	Límite inferior N (min)	
		Complemento a 1	Complemento a 2
8	127	-127	-128
16	32.767	-32.767	-32.768
32	2.147.483.649	-2.147.483.649	-2.147.483.650
64	$9{,}223372 * 10^{18}$	$-9{,}223372 * 10^{18}$	$-9{,}223372 * 10^{18}$

Operaciones aritméticas

Suma binaria

Para poder efectuar operaciones aritméticas con números de cualquier otra base se utilizan las reglas de la aritmética decimal, las reglas de la suma binaria son las siguientes:

- 0+0=0
- 0+1=1
- 1+0=1
- 1+1=10 (Con acarreo de un 1)
- 1+1+1=11 (Con acarreo de un 1)

Overflow y Carry[3]

Junto con el resultado de cada operación, la ULA genera varios indicadores acerca del mismo, que se conocen por sus iniciales inglesas:

[3] Carry: se refiere al bit que se lleva o se acarrea de una posición de un registro o una operación a otra posición contigua de mayor peso en un sistema de numeración binario, cuando se realiza una operación aritmética como la suma o la resta.

Indicador de signo S=1 indica que el resultado de la operación es negativo

Indicador de resultado cero Z=1 indica si el resultado es cero

Indicador de acarreo C=1 indica si existe un acarreo (también llamado “carry”)

Indicador de desborde u overflow V=1 si el resultado de una suma entre números con bit de signo excede el mayor valor positivo o negativo que se puede representar.

Al sumar dos números el overflow se puede dar solo si los dos tienen el mismo signo, el overflow se reconoce cuando los bits de signo de los dos números que se suman son iguales entre sí, pero distintos del bit de signo del resultado, ósea cuando los números son positivos y da resultado negativo o viceversa. Por ejemplo:

$$\begin{array}{r} 0\ 1\ 1\ (+3) \\ +\ 0\ 0\ 1\ (+1) \\ \hline 1\ 0\ 0 \end{array}$$

Punto Flotante

Representación de datos reales

Cuando se opera con números muy grandes o muy pequeños se suele utilizar la notación exponencial, también conocida como notación científica o notación en coma flotante, según esta notación cualquier número se puede representar como:

Numero = mantisa [4]* base ^exponente

Por ej., si quisiéramos representar el número 12.257,3285 se puede hacer de las siguientes formas:

Potencia positiva	Potencia negativa
(Se desplaza la coma hacia la derecha)	(Se desplaza la coma hacia la izquierda)
$1{,}32573285 * 10^{4}$	**$132.573.285 * 10^{-4}$**
$0{,}132573285 * 10^{5}$	**$13.257.328.500 * 10^{-6}$**

Sistema binario:

Se almacena en una palabra binaria con tres campos, se llama empaquetado:

- Signo: + o –
- Exponente (E)
- Parte significativa o mantisa (S = significant)
- La base B está implícita y no es necesario almacenarla

Por ej. Representar en coma flotante el numero decimal 34, en c a2, con 1 bit para el signo, 5 bits para el exponente y 8 bits para la mantisa, con bit implícito.

[4] Mantisa: Parte fraccionaria de un numero en notación científica. Por ejemplo, si tenemos el número 3.14159 × 10^2, la mantisa sería 0.14159

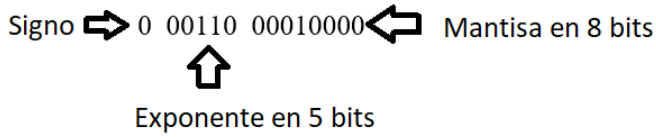
Convertir a binario: 00100010 (binario de 8 bits)

Notación exponencial (normalizar): $00100010 * 2^0 \rightarrow 0{,}100010 * 2^6$

Mantisa a almacenar (con bit implícito) 00010

Exponente: 110 = 6 pasado a binario

Empaquetar elementos según especificaciones (rellenar con 0)



Rango y precisión

Este formato permite representar amplio rango de números con poca cantidad de dígitos binarios, los dígitos utilizados para determinar la precisión se le llama mantisa y los dígitos utilizados para representar el rango se le llama exponente, por ej.

$+6{,}023 * 10^{23}$ Se escribe el signo, dos dígitos para el exponente y cuatro dígitos para la mantisa

El rango queda determinado por la cantidad de dígitos del exponente, que en el ejemplo se expresa a través de una potencia de la base 10

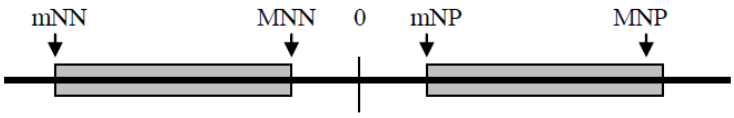
La precisión queda determinada por la cantidad de dígitos de la mantisa, que en nuestro ejemplo son cuatro.

Si se requiere un rango mayor, y si a cambio se está dispuesto a sacrificar precisión, se pueden usar 3 dígitos para la parte fraccionaria y dejar tres dígitos para el exponente, quedando:

±	2 3	6 0 2 3
Signo	Exponente Dos dígitos	Mantisa Cuatro dígitos

±	0 2 3	6 0 2	**+Rango** **-Precisión**

Hay que tomar en cuenta que la representación en punto flotante existe valores no representables, el rango de representación tiene la siguiente estructura:



mNN es el mínimo número negativo = - máxima mantisa * base $^{\text{máximo exponente positivo}}$
MNN es el máximo número negativo = - mínima mantisa * base $^{\text{máximo exponente negativo}}$
mNP es el mínimo número positivo = mínima mantisa * base $^{\text{máximo exponente negativo}}$
MNP es el máximo número positivo = máxima mantisa * base $^{\text{máximo exponente positivo}}$

Suma y resta en punto flotante

Hay cuatro etapas básicas en el algoritmo para sumar o restar:

- Comprobar valores cero

- Ajuste de exponente y mantisa
- Sumar o restar las mantisas
- Normalizar el resultado

Hay que tener en cuenta mantisa y exponente, también los exponentes de los operandos deben ser iguales y los procesos de ajuste implican perdida de precisión.

Ej.

$(0,101 * 2^3) + (0,111 * 2^4)$

Se iguala el menor de los exponentes al mayor, modificando en forma acorde la mantisa (movemos a la izquierda la coma)

En este proceso se pierde precisión en la cifra menos significativa del valor original, quedando:

$0,101 * 2^3 = 0,010 * 2^4$

Se suman las mantisas (Suma binaria): (0,010 + 0,111) = 1,001

Se normaliza 0,1001

Se vuelve a redondear a 3 dígitos, perdiendo precisión nuevamente y quedando:

$0,100 * 2^5$

Procesamiento de la coma flotante

La representación y manejo de los datos puede ser responsabilidad del hardware de la computadora o de los traductores de lenguajes. Los microprocesadores actuales disponen internamente un procesador de coma flotante (FPU por sus siglas en ingles Float Point Unit) que contiene circuitos aritméticos para operar con este tipo de datos.

Hasta la década de los 80, cada fabricante utilizaba un sistema propio para la representación de los números reales, pero surgió la necesidad de un sistema normalizado en el que trabajó la asociación IEEE[5].

El IEEE ha creado un estándar sobre la presentación de números en coma flotante. Este estándar especifica como deben representarse los números en coma flotante con simple precisión (32 bits) o doble precisión (64 bits), y también como deben realizarse las operaciones aritméticas con ellos.

Simple precisión (32 bits)

Exponente:

- 8 bits.
- Representación sesgada ($2^{n-1}-1$) siendo n el numero bits para el exponente

Mantisa:

- 23 bits.
- Bit implícito = 24 bits efectivos.
- Normalizada coma a la derecha del primer digito significativo (1,bb… =23 bits)

[5] Las siglas vienen de Institute of Electrical and Electronics Engineers, la cual es una asociación mundial de ingenieros dedicada a la normalización y el desarrollo en áreas técnicas

Doble precisión (64 bits)

Exponente**:**

- 11 bits
- $2^{11-1}-1 = 1023$

Mantisa**:**

- 52 bits.
- Bit implícito = 53 bits efectivos
- Normalizada coma a la derecha del primer digito significativo (igual que en 32 bits)

Observaciones de los números reales

Por la obtención, en resultados intermedios, de números excesivamente pequeños puede perderse precisión de los cálculos o producirse un desbordamiento a cero o agotamiento

Por el caso contrario, por la obtención de resultados numéricos excesivamente altos, es decir por desbordamiento

En la comparación de dos números reales, hay que tener en cuenta que cada dato real en la computadora representa a infinitos números reales, por lo que en general no puede representarse exactamente con n bits, por lo que genera un error de representación

Esto también da lugar a problemas al comparar si un número es igual a otro, ya que el computador considera que dos números son iguales únicamente si son iguales todos sus bits.

La comparación de números debe hacerse con números enteros o considerando que dos números son iguales si la diferencia entre ellos es menor que un valor dado.

Otra consecuencia de los números reales, es que la suma y la multiplicación no siempre cumplen con las propiedades asociativas y distributivas, se pueden obtener resultados distintos dependiendo del orden en que se realizan las operaciones.

Unidades de medida de la información

Las computadoras debido a su construcción basada en circuitos electrónicos digitales, almacena información numérica y alfabética con el sistema binario, por lo tanto, la información que es ingresada sea cual sea, debe ser volcada a dicho sistema. Este sistema de numeración que utiliza internamente el hardware, se basa en la representación de cantidades utilizando los dígitos 1 y 0. Cada digito de un numero representado en este sistema se denomina BIT[6]. En informática al igual que en la vida real, existen distintos tipos de medidas para almacenar la información, siendo estas:

[6] BIT: Contracción de binary digit

MEDIDA	EQUIVALENCIA
1 Byte	8 Bits
1 Kilobyte (**KB**)	1.024 bytes = 2^{10} bytes
1 Megabyte (**MB**)	1.024 KB = 2^{20} bytes = 1.048.576 bytes
1 Gigabyte (**GB**)	1.024 MB = 2^{30} bytes = 1.073.741.824 bytes
1 Terabyte (**TB**)	1.024 GB = 2^{40} bytes
1 Petabyte (**PB**)	1.024 TB = 2^{50} bytes

Normalmente los más comunes y usados hoy en día son GB y TB para almacenamiento y MB para archivos multimedia (música, imágenes, etc.), y KB para archivos de texto plano o archivos muy pequeños.

Representación de texto

Existen tres formas:

- **CODIGO EBCDIC**
- **CODIGO ASCII**
- **UNICODE**

La computadora recibe la información en alguno de los lenguajes escritos:

- **Caracteres alfabéticos**: Letras mayúsculas y minúsculas del abecedario inglés (A,B,C,D..a,b,c,d…)
- **Caracteres Numéricos**: Las diez cifras decimales (0,1,2,3,4,5,6,7,8,9)
- **Caracteres Especiales**: Símbolos que no están incluido en los grupos anteriores, como por ejemplo ({) (, * / ; : + Ñ ñ = ! ? . " & > # <] Ç [SP }) El espacio en blanco, también se incluye.
- **Caracteres Geométricos y gráficos**: Son símbolos o módulos con los que se pueden representar figuras o iconos (no muy usados), por ej. □ ◊ ♫ ¶ ♠ ♣ ♥ ☼ ☺ ♂ ♀ ♦ β £
- **Caracteres Control**: Representan ordenes de control, como el carácter indicador de fin de línea o el carácter de que se emita un pitido en la terminal, entre otros (Muchos de los caracteres de control son generados e insertado por la propia computadora).

Al tener que convertir toda la información suministrada a la computadora a ceros y unos es necesario establecer una codificación entre 2 conjuntos:

$$A=\{A,B,C,D,\ldots a,b,c\ldots 1,2,3,4,5\ldots +/(,)\ldots\} \text{ y } B=\{0,1\}^n$$

De forma tal que a cada elemento de A le corresponda un elemento distinto de B (n bits). Estos códigos se denominan códigos de E/S y pueden definirse de forma arbitraria, no obstante, existen códigos de E/S normalizados que son utilizados por diferentes constructores de computadoras (ASCII, EBCDIC, BCD de intercambio normalizado, UNICODE, etc.)

- **Codificación BCD de 6 bits**: Al evolucionar la computación, este esquema resultó inadecuado para representar todos los caracteres que se necesitan en los sistemas de computación modernos.
- **Codificación EBCDIC[7] de 8 bits**: Aunque su uso no es tan común, sigue siendo utilizado por maquinas que necesitan retrocompatibilidad.

[7] EBCDIC (Por sus siglas en inglés) Se refiere al Código de intercambio Decimal Codificado en Binario Extendido

- **Codificación ASCII de 7 bits:** Es el más utilizado hoy en día y el que muchas computadoras siguen utilizando). Otras versiones ampliadas utilizan 8 bits y respetan los códigos normalizados del ASCII básico, por ejemplo, la norma ISO 8859-1 se proyectó para América y Europa Occidental e incluye vocales con acentos, tildes y otras letras latinas no usadas en los países anglosajones)
- **Codificación UNICODE**: Propuesto por un consorcio de empresas y entidades con el objetivo de permitir la escritura de aplicaciones capaces de procesar texto de muy diversas culturas (demanda que surge con la proliferación de aplicaciones web), los códigos anteriores presentan desventajas a consecuencia de la masiva expansión de internet hacía todas partes del mundo, tales como:
    - Símbolos insuficientes para representar caracteres especiales
    - La mayoría está basada en los caracteres latinos, pero otras culturas utilizan símbolos muy distintos, por ejemplo, las culturas asiáticas como la china, japonesa o coreana utilizan símbolos que representan palabras, frases o ideas completas, siendo inútiles los códigos que solo codifican letras individuales

  Por estas razones, UNICODE está reconocido como estándar y trata de ofrecer las siguientes propiedades:

    - **Universalidad**: Trata de cubrir la mayoría de lenguajes escritos existentes en la actualidad, usando 16 bits siendo igual a 65.356 símbolos
    - **Unicidad**: A cada carácter se le asigna exactamente un código.
    - **Uniformidad**: Ya que todos los símbolos se representan con un número fijo de bits.

TEMA 3. Circuitos combinacionales

Símbolos y notación

En general, las variables booleanas se representan con letras: A,B,C…etc. 0 y 1 son constantes.

El operador de la suma se representa con el signo (+) y el producto lógico con el signo (.) o se omite (por ej., ABC+CAB…)

Los complementos de una variable o de un grupo de variables se puede representar con el apostrofe (‘) por ej. A’ o (A+B)’, también puede representarse con: $\overline{A} \equiv A' \;\; ; \;\; \overline{A+B} \equiv (A+B)'$

En ocasiones se utilizan los símbolos X o * para indicar que el valor de una variable puede ser 0 o 1, sin importar su valor

Postulados y teoremas

	Relación	Dual	Propiedad
Postulados	$A\,B = B\,A$	$A + B = B + A$	Conmutativa
	$A\,(B + C) = A\,B + A\,C$	$A + B\,C = (A + B)\,(A + C)$	Distributiva
	$1\,A = A$	$0 + A = A$	Elemento neutro
	$A\,\overline{A} = 0$	$A + \overline{A} = 1$	Complemento
Teoremas	$0\,A = 0$	$1 + A = 1$	Identidad
	$A\,A = A$	$A + A = A$	Idempotencia
	$A\,(B\,C) = (A\,B)\,C$	$A + (B + C) = (A + B) + C$	Asociatividad
	$\overline{\overline{A}} = A$		Involución
	$\overline{A\,B} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A}\,\overline{B}$	DeMorgan
	$AB + \overline{A}C + BC = AB + \overline{A}C$	$(A+B)(\overline{A}+C)(B+C) = (A+B)(\overline{A}+C)$	Consenso
	$A\,(A + B) = A$	$A + A\,B = A$	Absorción

Principio de dualidad: El dual de una función booleana se obtiene intercambiando:

- Sumas lógicas (O, v, compuerta OR)
- Productos lógicos (Y, ^, compuerta AND)
- Unos (1, verdadero, variable sin complemento, pasa corriente) y ceros (0, falso, variable complementada, no pasa corriente)

Funciones lógicas

Todas las posibles combinaciones de salida de dos variables definen 16 funciones:

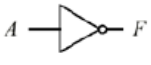
A	B	*Falso*	*AND*	$A\overline{B}$	A	$\overline{A}B$	B	*XOR*	*OR*	*NOR*	*XNOR*	$\overline{B}$	$A+\overline{B}$	$\overline{A}$	$\overline{A}+B$	*NAND*	*Verdad*
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Existen tres formas de expresar una función:

- Ecuación booleana
- Tabla de verdad
- Símbolo lógico

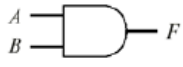
NOT $F = \overline{A}$

A	F
0	1
1	0



AND $F = A\,B$

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1



OR $F = A + B$

A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

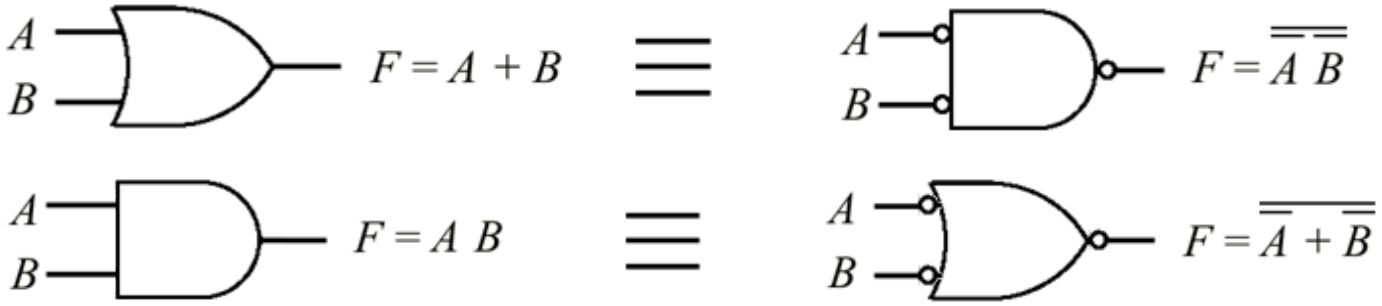
XOR $F = A \oplus B$

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

Teorema de DeMorgan

Permite transformar una compuerta en otra, por ej. Dadas dos variables A B:

A	B	$\overline{A\,B} = \overline{A} + \overline{B}$		$\overline{A+B} = \overline{A}\,\overline{B}$	
0	0	1	1	1	1
0	1	1	1	0	0
1	0	1	1	0	0
1	1	0	0	0	0



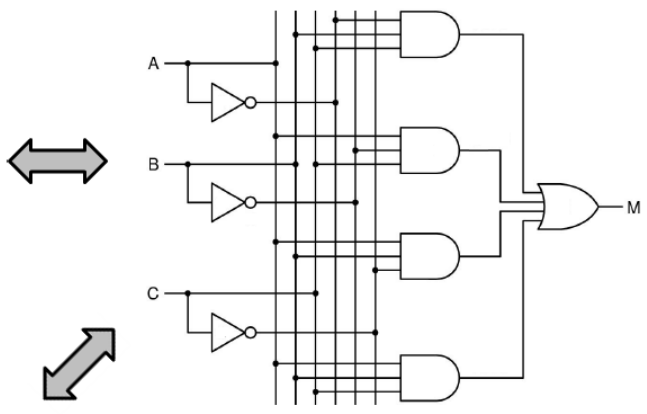
Circuitos combinacionales y secuenciales

Circuito combinacional: Es el circuito lógico en el cual el valor de las salidas depende exclusivamente de los valores lógicos presentes en las entradas, un cambio en una entrada siempre produce el mismo cambio en la salida

Circuito secuencial: Es el circuito lógico en el cual el valor de las salidas depende no solo de los valores lógicos presentes en las entradas, sino también de la secuencia temporal previa de esos valores en esas entradas, un cambio en una entrada no siempre produce el mismo cambio en la salida.

Las salidas de los circuitos combinacionales se pueden expresar como una expresión booleana, tabla de verdad y un esquema circuital. Es posible encontrar expresiones booleanas y diagramas circuitales diferentes, que representan la misma función lógica. Esto se verifica si las tablas de verdad son las mismas.

A	B	C	M
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



$$M(A,B,C) = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

Componentes combinacionales MSI

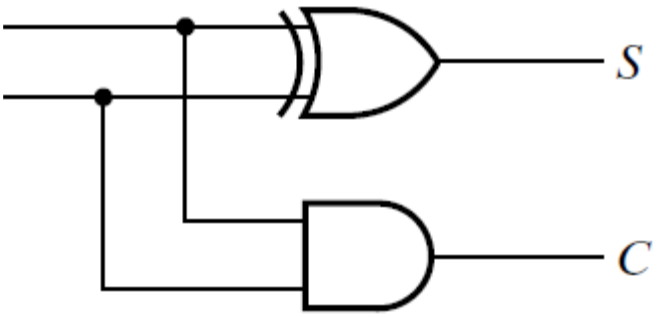
Su nombre viene de Middle Scale of Integration, ya que integran un alto numero de compuertas. Varios de ellos permiten una implementación más rápida de las funciones combinacionales. Se estudiarán los siguientes circuitos:

- Sumadores
- Decodificadores
- Multiplexores y demultiplexores

Sumadores

Los circuitos sumadores implementan la función de la suma aritmética. Para la suma de un digito, intervienen dos operandos de entrada y uno de salida. Además, es necesario establecer una salida adicional para contemplar el acarreo que se produce cuando la suma de dos dígitos es mayor que el número de dígitos de entrada (Ej. 5+6=11), otro ejemplo: en base 2, siendo S el resultado y C el acarreo, se obtiene un semi sumador

A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

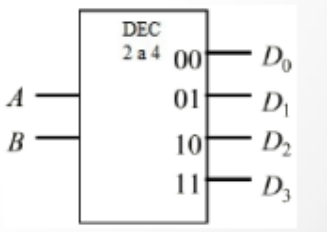


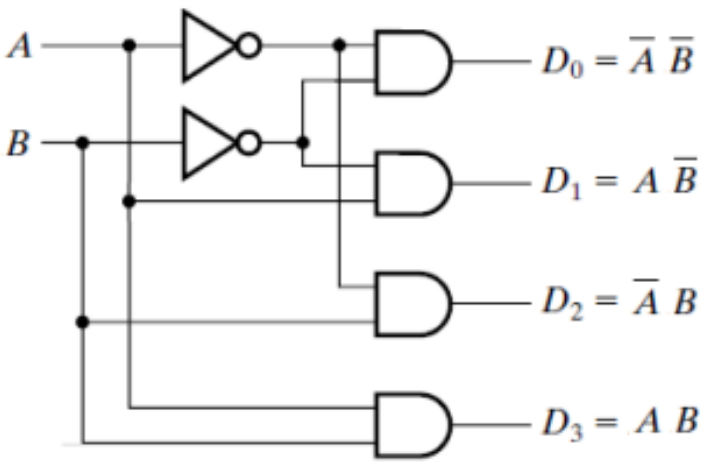
Decodificadores

Puede considerarse como un circuito que recibe como entradas n líneas conteniendo un código binario, y activa en sus m salidas solo las correspondientes al valor presente en la entrada. En general, es m (salida) = $2^{n.}$

Por ej. Para dos bits de entrada su tabla de verdad quedaría:

A	B	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



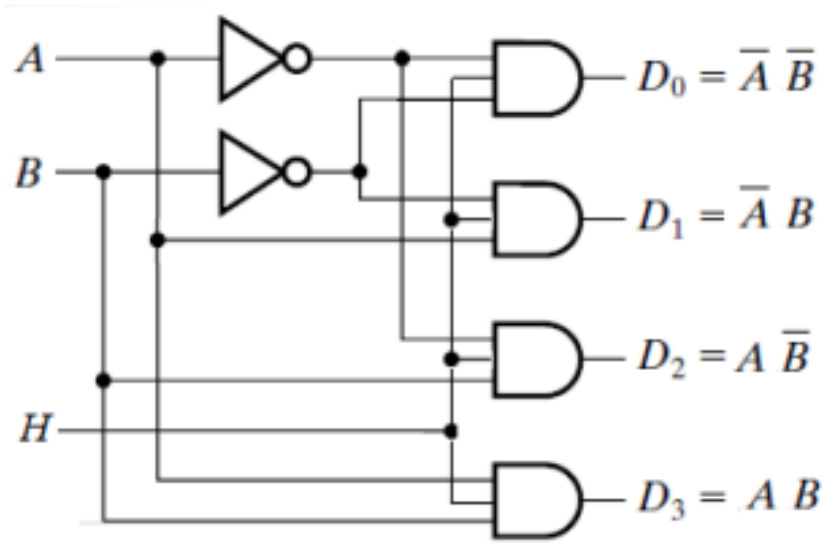


Y de su tabla de verdad podemos hacer circuito lógico

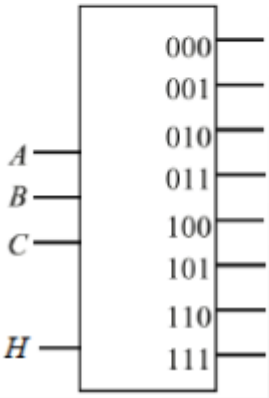
En la mayoría de casos, los decodificadores poseen una entrada adicional de habilitación H, que al estar inactiva provoca que ninguna salida esté activa, independientemente del código de entrada. Quedando su tabla de verdad:

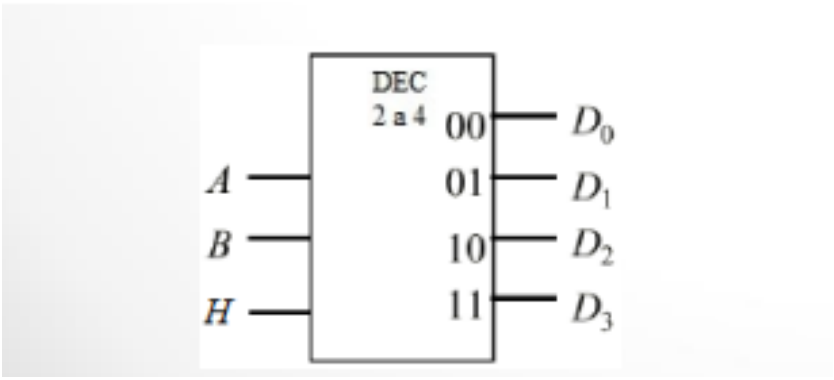
H	A	B	D_0	D_1	D_2	D_3
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

X = no importa



Esquemáticamente, un decodificador de 3 a 8 será:





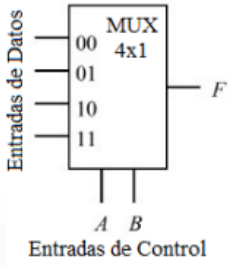
Aquí vemos un ejemplo del concepto mencionado antes de una entrada X, ya que si la entrada H está apagada (en 0) indiferentemente de los valores de entrada de A y B, la salida será la misma.

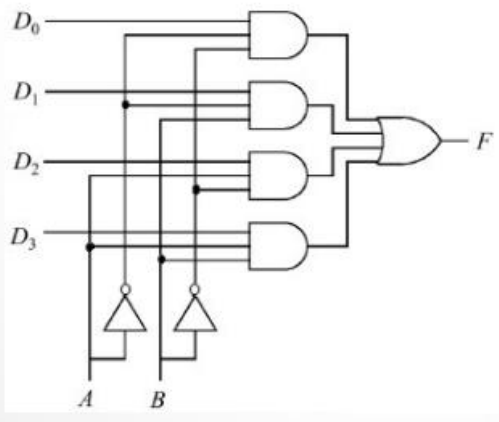
Multiplexores

Es un circuito que selecciona el valor lógico de una de sus n entradas y lo presenta su única salida, de acuerdo al valor presente en sus entradas de selección. se utiliza para seleccionar uno de varios flujos de datos y enviarlo a través de una única línea de transmisión. Básicamente, recibe múltiples entradas y selecciona una de ellas para transmitirla a la salida. Por ej.

A	B	F
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

$$F = \overline{A}\,\overline{B}\,D_0 + \overline{A}\,B\,D_1 + A\,\overline{B}\,D_2 + A\,B\,D_3$$



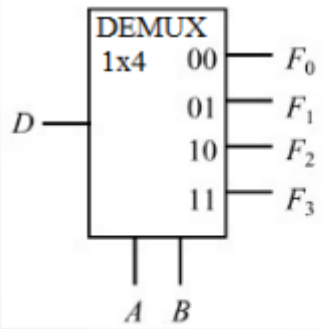


Demultiplexor

Realiza la función inversa del multiplexor: presenta el valor lógico presente en su única entrada en una de sus n salidas, de acuerdo al valor presente en sus entradas de selección. Tiene una sola entrada y múltiples salidas, su función principal es tomar una de las señales de entrada y enviarla a una de sus salidas correspondientes.

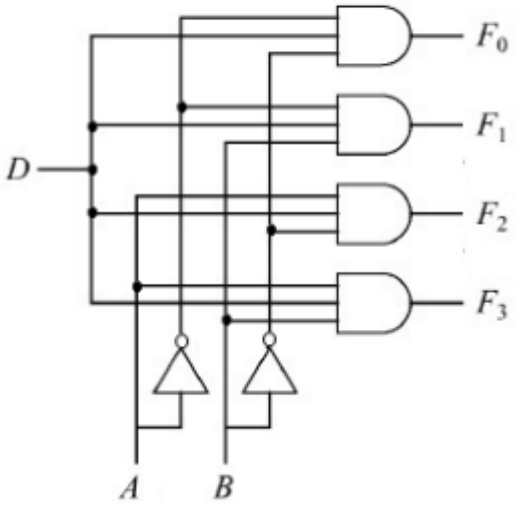
Podemos entenderlo mejor viéndolo como un paquete que se envía por correo a una dirección específica. El paquete es la señal de entrada, y el demultiplexor es como el cartero que lo dirige a la dirección correcta (salida). Por ej. Su tabla de verdad y circuito:

A	B	F_0	F_1	F_2	F_3
0	0	D	0	0	0
0	1	0	D	0	0
1	0	0	0	D	0
1	1	0	0	0	D



$$F_0 = D\overline{A}\,\overline{B} \qquad F_2 = DA\overline{B}$$

$$F_1 = D\overline{A}B \qquad F_3 = DAB$$



Resumiendo:

Sumadores: Implementan la función de la suma aritmética, tienen dos entradas y tendrán mínimo dos salidas; Resultado y Acarreo. Dos semi sumadores conectados hacen un sumador completo, conectando n sumadores completos podemos hacer uno de n bits.

Decodificadores: Siempre tendrá una entrada de habilitación (H) que afecta sus salidas. Sin contar la entrada H, recibe n entradas y tiene m salidas, siendo $m = 2^n$. Normalmente asumimos que la entrada H está presente siempre.

Multiplexores: Evalúa el valor lógico de n entradas y lo distribuye en una única salida, podríamos relacionarlo con un cartero que necesita entregar un mensaje (entrada) a una dirección en especifica(salida).

Demultiplexores: Hace la función inversa del multiplexor, evalúa el valor lógico de su única entrada y lo distribuye en sus n salidas (de acuerdo al valor presente en sus entradas), un ejemplo lo podemos ver en un sistema de telefonía, un demultiplexor separa las señales de voz de diferentes usuarios que se transmiten a través de un canal de comunicaciones común. Luego, las señales de voz se dirigen a sus respectivos destinos, es decir, a los teléfonos de los destinatarios correspondientes.

Funciones combinacionales con MSI

Además de su función propia como bloque, los decodificadores y multiplexores permiten implementar muy fácilmente circuitos combinacionales a partir de su tabla de verdad.

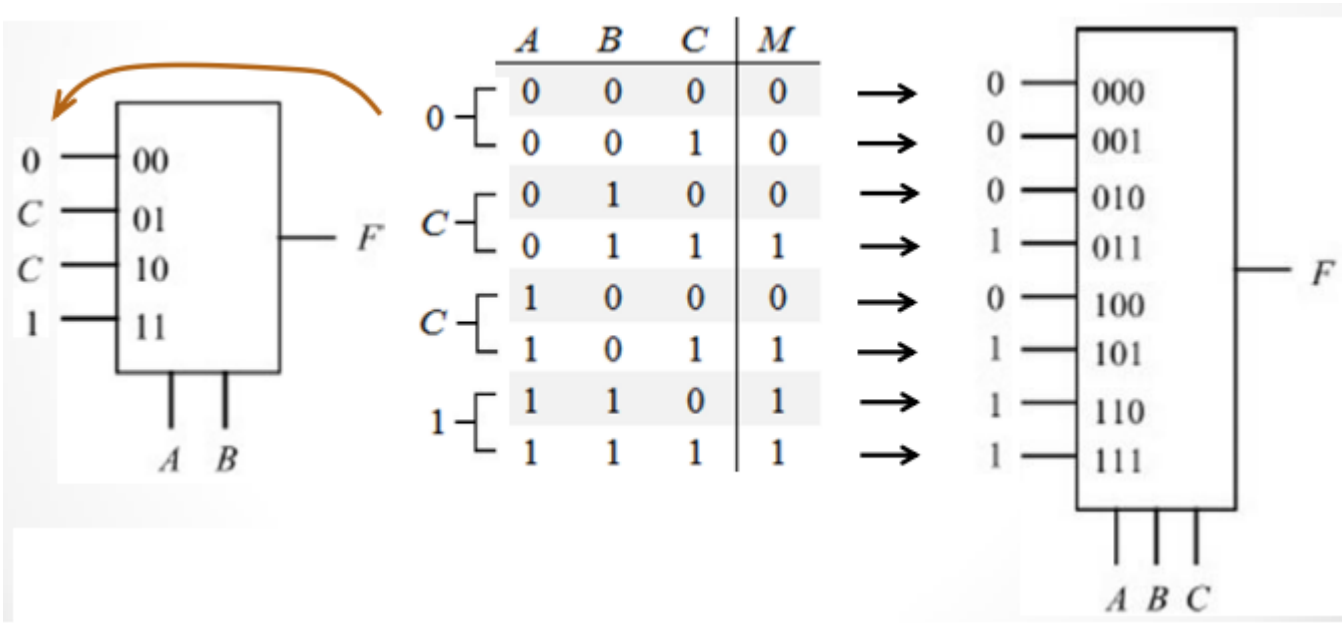
Mintérminos: Es cada expresión en la que intervienen todas las variables de la función, negadas o no, e interrelacionadas y se denota como m_i (m sub i) con i es el equivalente decimal del valor binario de las variables. Por ej. M = A'BC + AB'C + ABC' + ABC. Su tabla quedaría como:

	A	B	C	M
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

Entonces, nuestra función M puede expresarse como: $M=m_3+m_5+m_6+m_7= \sum m_i (3;5;6;7)$

Un decodificador puede verse como un "generador de mini términos" por lo que resulta muy simple la implementación de una función.

Recordando el funcionamiento de un multiplexor, se ve que, si se utiliza uno que tenga la misma cantidad de variables de selección que la de variables de la función, la implementación es directa. En otro caso si la cantidad de variables de selección es menor, es necesario determinar una función lógica para cada entrada, por ej.



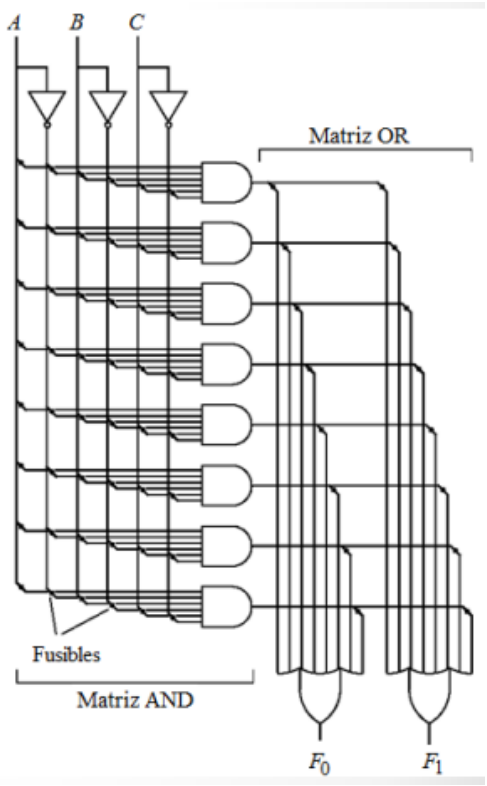
Matrices lógicas programables (PLAs)

Una PLA[8] es una matriz configurable de compuertas AND seguida de otra matriz configurable de compuertas OR.

Se personalizan anulando fusibles para una función especifica

La configuración y ejecución de las conexiones lo realiza un software especifico

Son muy útiles porque permiten a los diseñadores de circuitos integrados personalizar la funcionalidad de un circuito sin tener que diseñar un circuito específico para cada tarea. Por ej.



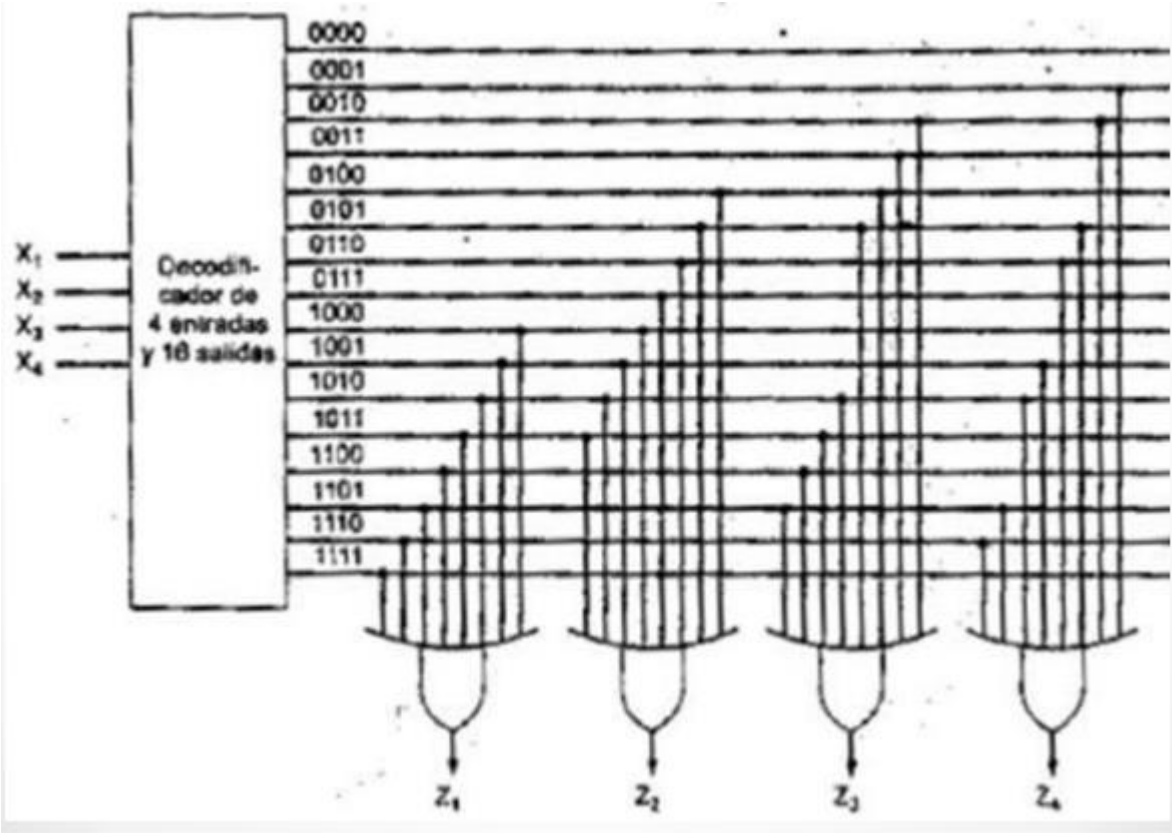
Memorias de solo lectura (ROMs)

Una ROM[9] puede considerarse un decodificador con varias compuertas OR conectadas. Cada conexión es programable. Solo puede ser usada para almacenar datos de forma permanente.

Una memoria ROM se puede reprogramar un numero limitado de veces antes de que deje de funcionar

[8] PLA del inglés (Programmable Logic Array), es un circuito integrado para implementar funciones lógicas programables. Es una matriz de puertas lógicas interconectadas que se pueden programar para realizar diferentes funciones lógicas.

[9] ROM del inglés (Read Only Memory) Es un tipo de memoria de computadora que se utiliza para almacenar datos que no se pueden cambiar o editar después de que se han programado en ella. (Por ej. Sistema operativo, Firmware, controladores, configuraciones, aplicaciones preinstaladas, archivos de arranque y recuperación, entre otros)


X1
X2
X3
X4
Decodificador de 4 entradas y 16 salidas
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111
Z1
Z2
Z3
Z4