Tema 1:
Introducción a los Computadores

- Niveles de abstracción de un computador
- Conceptos básicos
- Evolución histórica de los computadores
- Arquitectura Von Neumann
- Fases de ejecución de una instrucción
- Lenguajes de programación

Bibliografía básica

- Estructura y diseño de Computadores (Capítulo 1 y capítulo 2)
  David A. Patterson, John L. Hennessy
  Ed. Reverté S.A.

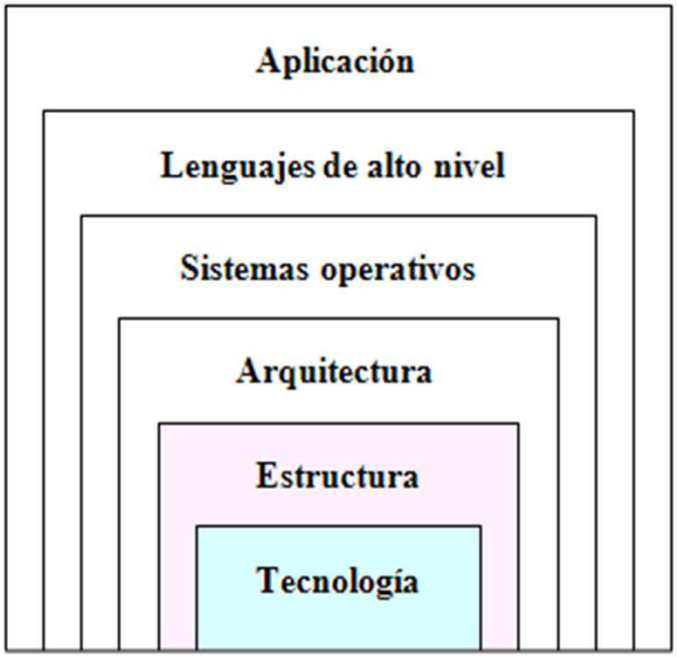
- Fundamentos de los Computadores (Capítulo 1)
  Pedro de Miguel Anasagasti
  Ed. Paraninfo

- Arquitectura de Computadores (Capítulo 1)
  J. Antonio de Frutos, Rafael Rico
  Ed. Universidad de Alcalá

- Estructura de Computadores (Capítulo 1)
  José Mª Angulo Usategui
  Ed. Paraninfo

Niveles de abstracción de un computador

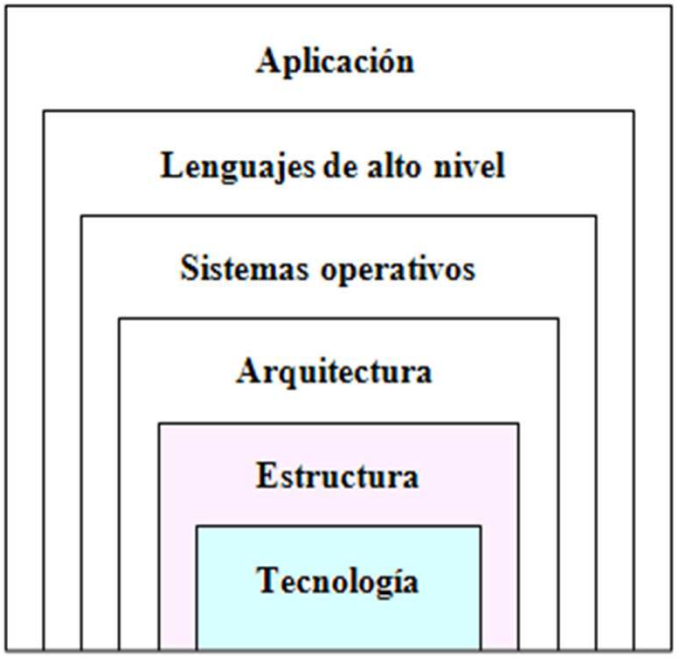
- Plan de estudios: Grado en Ingeniería Informática



Materia	Asignatura	Cuat.
Estructura y tecnología de computadores	Fundamentos de tecnología de computadores	1º
	Estructura y organización de computadores	3º
Sistemas Operativos	Sistemas Operativos	2º
	Sistemas Operativos Avanzados	3º
Programación	Fundamentos de programación	1º
	Programación	1º
	Programación Avanzada	4º
	Ampliación de Programación Avanzada	6º
	Procesadores del Lenguaje	5º
Bases de Datos	Bases de Datos	4º
	Bases de Datos Avanzadas	5º

Niveles de abstracción de un computador

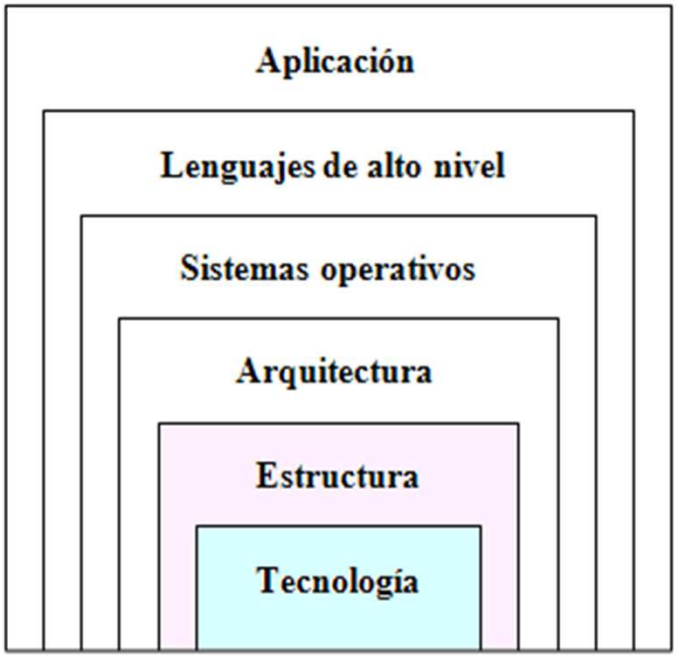
- Plan de estudios: Grado en Sistemas de la Información



Materia	Asignatura	Cuat.
Estructura y tecnología de computadores	Fundamentos de tecnología de computadores	1º
Sistemas Operativos	Sistemas Operativos	2º
Programación	Programación y estructuras de datos	1º,2º,3º
Bases de Datos	Bases de Datos	4º,5º

Niveles de abstracción de un computador

- Plan de estudios: Grado en Ingeniería de Computadores



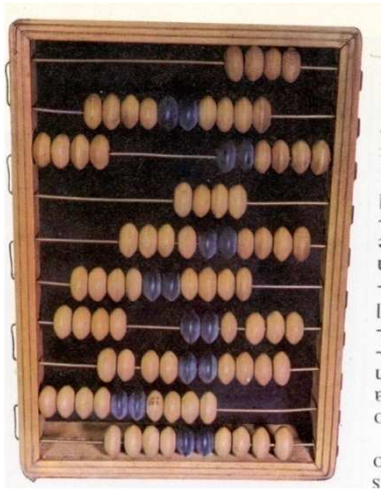
Materia	Asignatura	Cuat.
Estructura y tecnología de computadores	Fundamentos de Tecnología de Computadores	1º
	Estructura y Organización de Computadores	3º
	Electrónica	5º
Sistemas Operativos	Sistemas Operativos	2º
	Sistemas Operativos Avanzados	3º
Programación	Fundamentos de Programación	1º
	Programación Avanzada	4º
	Procesadores del Lenguaje	6º
Bases de Datos	Bases de Datos	4º
Arquitectura de Computadores	Arquitectura e Ingeniería de Computadores	5º

Conceptos básicos

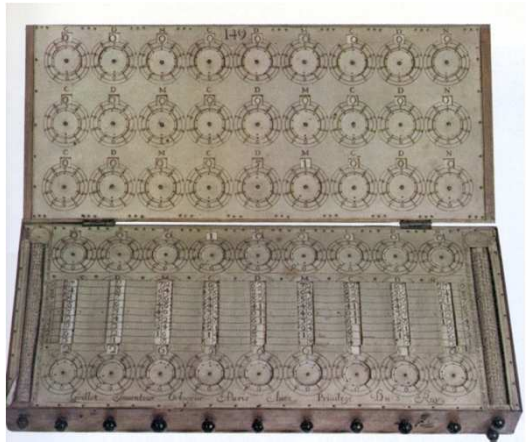
- **Computador**:
  Máquina destinada a procesar información, entendiéndose por proceso las sucesivas manipulaciones de la información para resolver un problema
- Información del computador:
  - **Bit** ⇨ Elemento básico de información (‘0’ ó ‘1’)
  - **Byte** u **octeto** ⇨ Grupo de 8 bits (‘01101111’)
  - **Palabra** ⇨ Grupo de bits con el que trabaja habitualmente el computador (8 bits, 16 bits, 32 bits ó 64 bits)
  - **Unidades:** $1\ K \Rightarrow 2^{10} = 1024$
    $1\ M \Rightarrow 2^{10} \cdot 2^{10} = 1024\ K$
    $1\ G \Rightarrow 2^{10} \cdot (2^{10} \cdot 2^{10}) = 1024\ M$
- **Instrucción**: Operación que realiza el computador
- **Dato**: Operando o resultado de una instrucción
- **Programa**: Conjunto ordenado de instrucciones

Evolución histórica de los computadores (I)
Antecedentes de los computadores (I)

- El ábaco como primer instrumento para calcular.
  - Es un dispositivo consistente en un conjunto de cuentas engarzadas en una varilla cuyo origen se remonta a los siglos III o IV a. De C.
  - No aportó nada al concepto de cálculo ni a su automatización
- Mecanismo de cálculo
  - Desarrollada por Blas Pascal (1642)
  - Constaba de un conjunto de ruedas dentadas, cada una de ellas numerada del 0 al 9. Al pasar una rueda del 9 al 0 arrastraba un décimo de vuelta la siguiente.
  - Además incluía un sistema de memoria que almacenaba los resultados



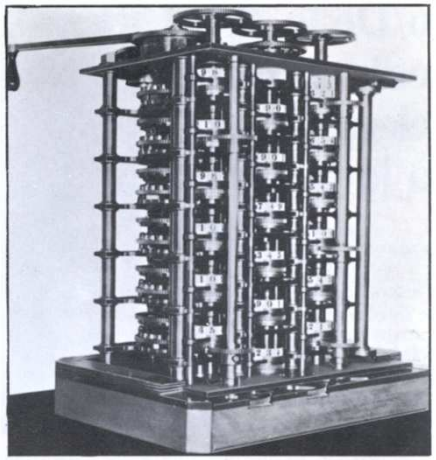
Ábaco



Máquina aritmética

Evolución histórica de los computadores (II)
Antecedentes de los computadores (II)

- La máquina de Leibnitz (1671)
  - Realizaba las cuatro operaciones aritméticas.
  - Perfecciona la de Pascal que solamente sumaba y restaba



Máquina de diferencias

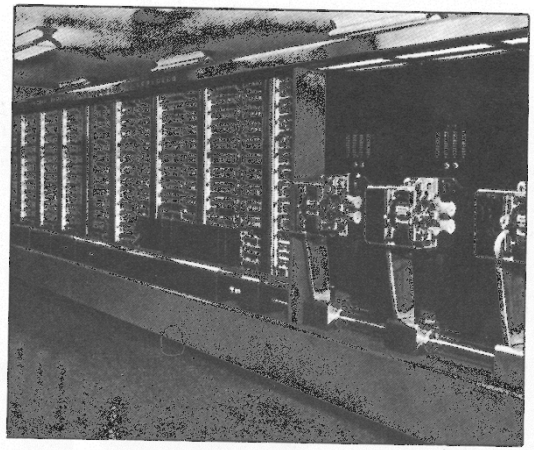
- Máquina de diferencias (abandonada) (1823) y la máquina analítica (1831) de Babbage
  - Permite ejecutar cualquier operación si intervención humana en el proceso de cálculo
  - Consta de una memoria, una unidad aritmética, sistema de engranajes para transferir datos entre memoria y la unidad aritmética y un dispositivo para introducir y sacar datos de la máquina
  - Empleaba tarjetas perforadas para programarse
  - Nunca llegó a construirse

Evolución histórica de los computadores (III)
1ª generación

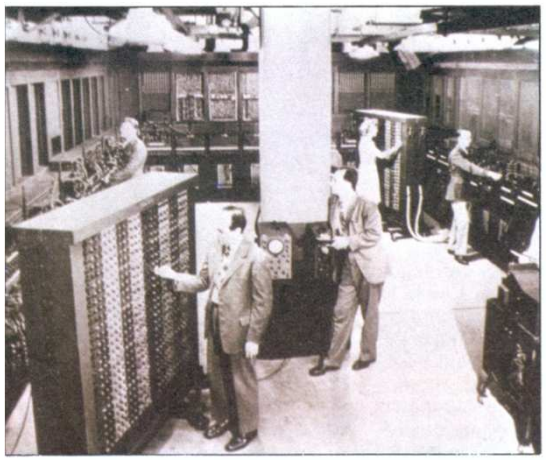
- Máquinas de carácter experimental construidas con tubos de vacío
- Calculadores de relés. H. Aiken construye la serie de calculadoras MARK
- 1941: ENIAC - *Electronic Numerical Integrator and Calculator.* Eckert y Mauchly

  Computador de propósito general con programa cableado
  (Cálculo de fuegos de artillería en la II Guerra Mundial)
- 1945: *First Draft of Report on the* EDVAC - *Electronic Discrete Variable Automatic Computer.* Von Neumann

  Computador de propósito general con programa almacenado (1952)



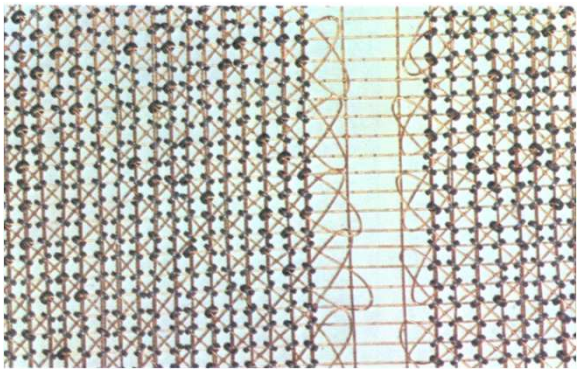
MARK I



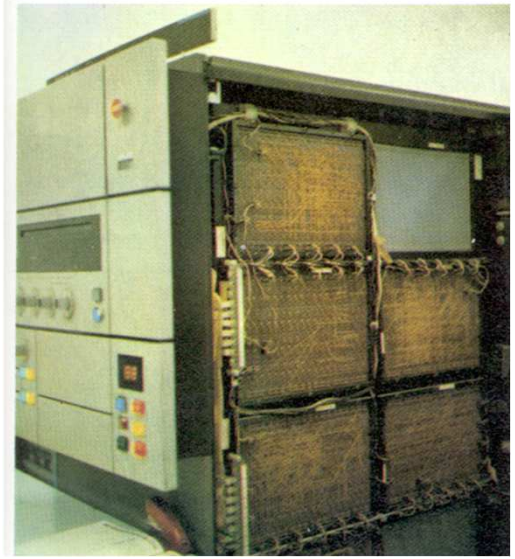
ENIAC

Evolución histórica de los computadores (IV)
2ª generación

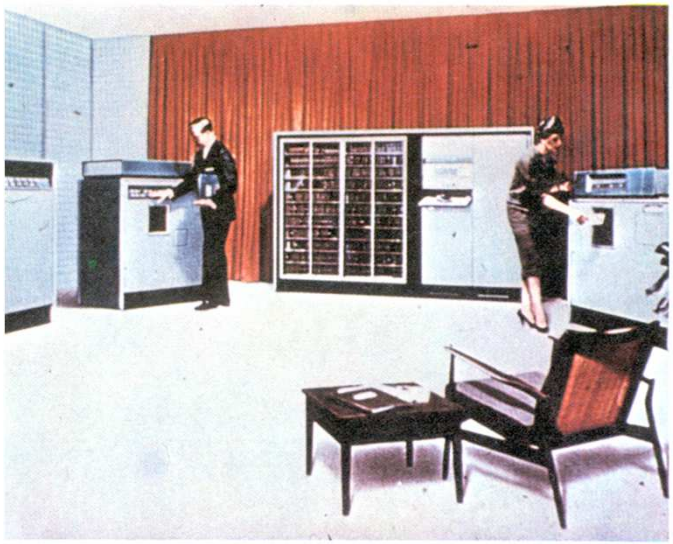
- Computadores comerciales
- Construidos con transistores ⇨Menor tamaño, menor disipación de calor, mayor fiabilidad
- Memorias de ferritas



Memoria de ferritas



Mueble para almacenar una memoria de ferritas

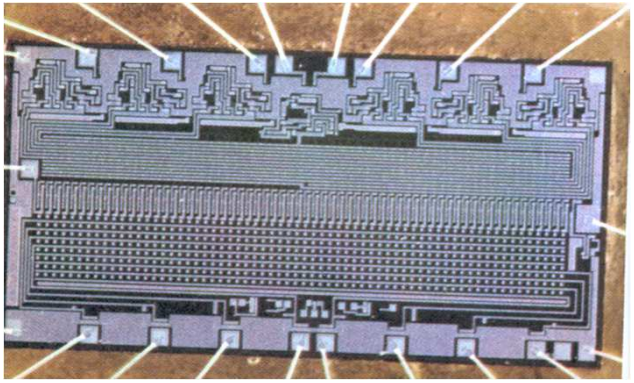


UNIVAC (2ª gen.)

Evolución histórica de los computadores (V)
3ª generación

3ª generación:

- Familias de computadores: Minicomputadores y supercomputadores
- Construidos con circuitos integrados ⇨ menor tamaño, más baratos, menor consumo de energía



Circuito integrado

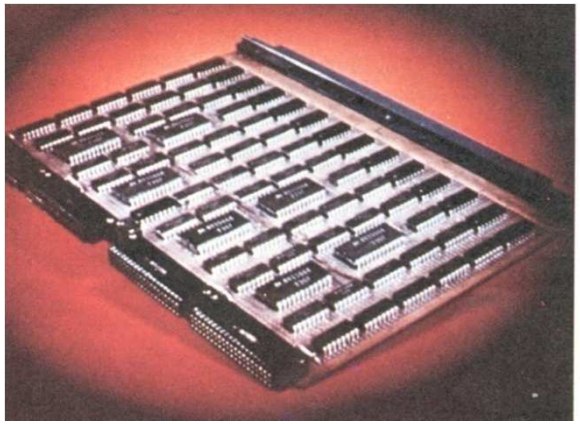
IBM serie 370 (3ª gen.)

Evolución histórica de los computadores (VI)
4ª generación

4ª generación:

- Computadores personales y estaciones de trabajo
- Otras aplicaciones: electrodomésticos, equipos de música y vídeo, etc.
- Construidas con microprocesadores y memorias de semiconductor
  1971: 1er microprocesador, INTEL 4004
- Década de los 80 ↑ procesamiento de información
- Década de los 90 ↑ comunicación de información (Redes)

PC (4ª gen.)



Memoria de semiconductores

Evolución histórica de los computadores (VI)
5ª generación

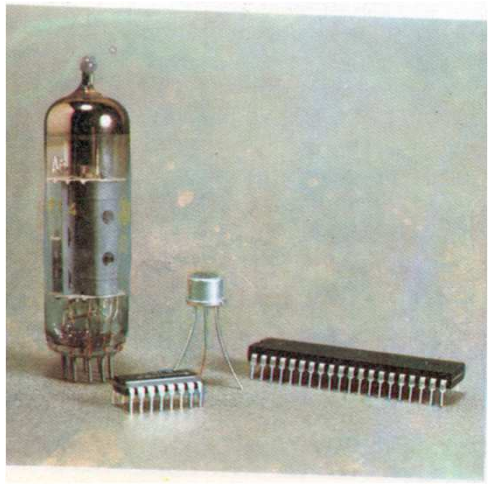
5ª generación:

- Proyecto ambicioso lanzado por Japón
- El microprocesador como elemento básico
- La computación masivamente paralela
- La comunicación y las conexiones entre computadores como algo generalizado.
- Internet. Correo electrónico. WWW
- ¿**6ª Generación**?
  - Miniaturización
  - Paralelismo
  - Clusters

Sistema de computadoras de quinta generación

Evolución histórica de los computadores (VII)

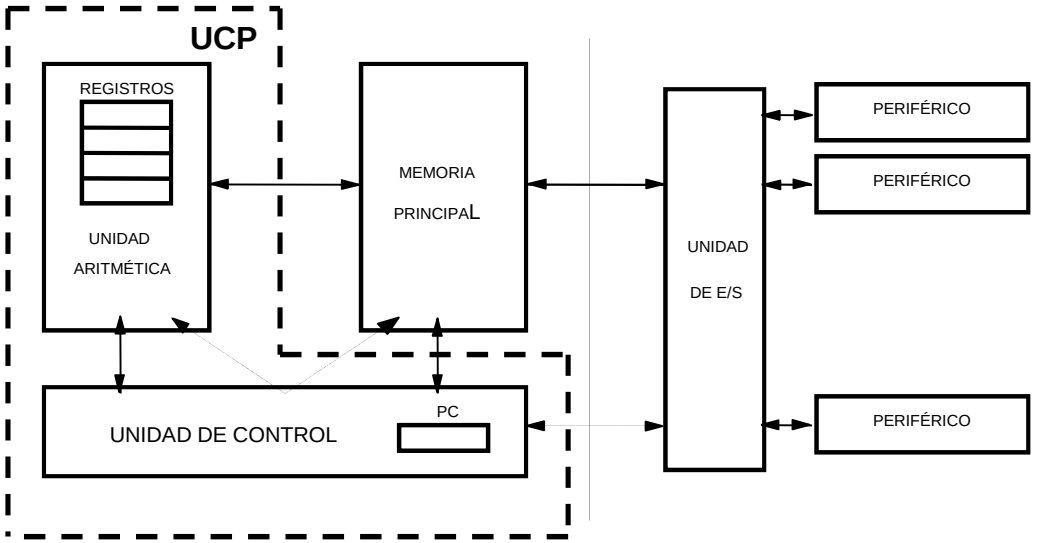
Generación / Características	1ª	2ª	3ª	4ª	5ª
Duración	1950 - 1960	1960 - 1970	1970 - 1980	1980 - 1990	1990 -200?
Tecnología	Válvulas electrónicas	Transistores	C.I. (SSI-MMI)	C.I (LSI)	C.I. (VLSI)
Máquinas	IBM 701	CDC 6600	PDP-8, PDP-11	Fujitsu M382 Cray X-MP	Alpha 21164 Pentium
Tipo de memoria	Tubos de Williams Tambores y cintas magnéticas	Núcleos de ferrita	Memorias en C.I. y memorias caché	Memorias virtuales	Memorias caché de varios niveles
Lenguajes	Máquina	FORTRAM, COBOL, ALGOL, PL1	BASIC, PASCAL	Alto nivel	Lenguaje natural, C
Producto	Computador	Computador comercial	Minicomputador	Microcomputador	Multiprocesador



Válvula, transistor, circuito integrado

Arquitectura Von Neumann

- Fue establecida en 1945 por John von Neumann
- Su característica principal es que ejecuta instrucciones de máquina de un programa almacenado en memoria
- Bloques:
  - Memoria principal
  - Unidad aritmética y banco de registros
  - Unidad de control (UC)
  - Unidad de entrada/salida



- Los **buses** son los elementos que interconectan los diferentes elementos de la arquitectura: bus de datos, bus de direcciones y bus de control

Fases de ejecución de una instrucción

1. **Fase de búsqueda de la instrucción:**
   La UC activa las señales de control necesarias para leer de memoria la instrucción a la que apunta el contador de programa (CP)
2. **Fase de decodificación:**
   La UC recibe la instrucción (RI) y la decodifica
3. **Búsqueda de operandos:**
   La UC, en caso necesario, lee los operandos de memoria o de los registros
4. **Ejecución y almacenamiento del resultado:**
   La UC genera las señales necesarias para realizar la operación, y en caso necesario, guarda el resultado en memoria principal o en un registro
5. La UC **actualiza el CP**, para pasar a ejecutar la siguiente instrucción
   - Funcionamiento secuencial
   - Modificación de secuencia ↓ modificar CP ↓ bifurcación o salto

Lenguajes de programación (I)
Tipos de lenguajes:

Lenguaje de alto nivel:
Posee instrucciones y sintaxis propia (Ej. PASCAL, C)

Lenguaje de alto nivel ⇓ **portabilidad** (se compila el mismo código en diferentes máquinas)

Lenguajes de bajo nivel

- **Lenguaje máquina**:
  Las instrucciones de un programa se escriben en binario
  - Incomodo y produce errores ⇓
    Solución: usar otros lenguajes de programación
- **Lenguaje ensamblador**:
  Las instrucciones se representan con nombres simbólicos o ***mnemónicos***
  - Cada instrucción en lenguaje ensamblador se corresponde con una instrucción máquina

Lenguajes de programación (II)

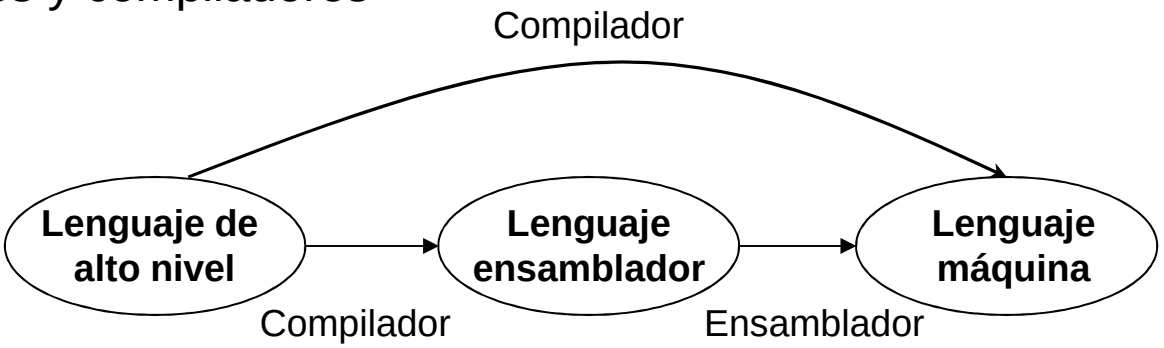
- Lenguaje de alto nivel (Ejemplo: PASCAL)

```
BEGIN
   Resta:= Minuendo - Sustraendo
END.
```

- Lenguaje máquina y lenguaje ensamblador (Ejemplo: Ensamblador i80x86)

```
A10000          MOV AX, Minuendo
2B060200        SUB AX, Sustrayendo
A30400          MOV Resta, AX
```

- La traducción de un programa a lenguaje máquina lo llevan a cabo los intérpretes y compiladores



Tema 2:
Sistemas de numeración

- Definiciones
- Bases de numeración
- Modos de representación
  - Representaciones numéricas
    - Coma fija (números enteros)
    - Suma-resta en base dos
  - Representaciones alfanuméricas

Bibliografía básica

- Fundamentos de los Computadores. (Capítulo 2)

  Pedro de Miguel Anasagasti

  Ed. Paraninfo

- Arquitectura de Computadores (Anexo A)

  J. Antonio de Frutos, Rafael Rico

  Ed. Universidad de Alcalá

- Arquitectura, programación y diseño de sistemas basados en microprocesadores (8086/80186/80286). (Capítulo 1)

  Yu-Cheng Lu, Glen A. Gibson

  Ed. Anaya Multimedia 86

Definiciones

- **Espacio material:** número de bits que se tienen para almacenar el dato (número o carácter)
  - **Byte** (8 bits)
  - **Palabra** (n bits)
- **Rango de representación:** valores máximo y mínimo que se pueden representar en un determinado sistema
- **Resolución de la representación:** diferencia entre un número y el siguiente inmediato
- **Longitud del código:** cuántos elementos diferentes se pueden obtener para una representación con *n* bits de espacio material. La longitud del código para *n* bits es 2^n

Bases de numeración (I)

- Bases 2, 8, 10 y 16

Binario (base 2)	**Octal** (base 8)	**Decimal** (base 10)	**Hexadecimal** (base 16)
0	0 (000)	0 (0000)	0 (0000) A (1010)
1	1 (001)	1 (0001)	1 (0001) B (1011)
	2 (010)	2 (0010)	2 (0010) C (1100)
	3 (011)	3 (0011)	3 (0011) D (1101)
	4 (100)	4 (0100)	4 (0100) E (1110)
	5 (101)	5 (0101)	5 (0101) F (1111)
	6 (110)	6 (0110)	6 (0110)
	7 (111)	7 (0111)	7 (0111)
		8 (1000)	8 (1000)
		9 (1001)	9 (1001)

- Cambio entre bases. Regla de *Horner*

Bases de numeración (II)

P_7	P_6	P_5	P_4	P_3	P_2	P_1	P_0

A cada posición le corresponde un peso

Unidades

Decenas

Centenas

Unidades de millar

Decenas de millar

$$Valor = \sum_{i=0}^{n-1} x_i \cdot base^i$$

- Ejemplos:
- Consideremos el número binario 10101. Este representa el valor decimal:

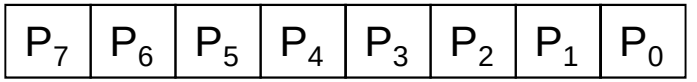
  $1.\ 2^4 + 0.2^3 + 1.2^2 + 0.2^1 + 1.2^0 = 21$

- El número 78A en base hexadecimal pasado a decimal:

  $7.16^2 + 8.16^1 + 10.16^0 = 1930$

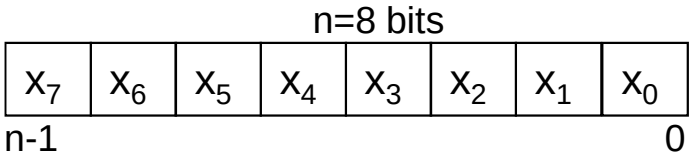
Representaciones numéricas en coma fija

- Coma fija:
  - Sin signo :
    - binario puro
  - Con signo:
    - Signo-magnitud
    - Complemento a la base, C2
    - C1
    - Exceso a M
  - BCD



A cada posición le corresponde un peso

Representaciones numéricas en coma fija
Binario Puro



- Sistema posicional de base 2 para números enteros
- Donde los pesos son: $P_i = 2^i$
- Con palabra de longitud n:
  - $\text{Valor} = \sum_{i=0}^{n-1} 2^i \cdot x_i$
  - Rango: $[0, 2^n - 1]$
- Resolución = 1
- Extensión de signo, añadiendo 0s por la izquierda del MSB (bit más significativo)
- El computador debe detectar cuándo ocurre desbordamiento *(overflow)*:
  - En suma y multiplicación
  - En la resta si el resultado es negativo

Representaciones numéricas en coma fija
Complemento a la base, Complemento a 1

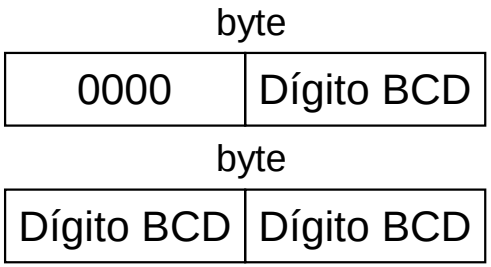
- Números positivos : comienzan por 0, representados en binario puro
- Números negativos : comienzan por 1, representados en C2
- El MSB indica el signo, pero se opera con los n bits como un conjunto indivisible
- -A = **Complemento a uno** de **A**, n=número de bits de la representación
  - 2^n -1- A
  - $\bar{A}$
- Con palabra de longitud n:
  - $$Valor = \begin{cases} +\sum_{i=0}^{n-1} 2^i \cdot x_i & \text{si } x_{n-1} = 0 \\ -Valor(C1(número)) & \text{si } x_{n-1} = 1 \end{cases}$$
  - Rango: $[-(2^{n-1}-1), -0, 0, (2^{n-1} - 1)]$
- Resolución = 1
- Extensión de signo, se realiza copiando el MSB en los bits de la izquierda

Representaciones numéricas en coma fija
Complemento a la base, Complemento a 2

- Números positivos : comienzan por 0, representados en binario puro
- Números negativos : comienzan por 1, representados en C2
- El MSB indica el signo, pero se opera con los n bits como un conjunto indivisible
- -A = **Complemento a dos** de **A**, n=número de bits de la representación
  - $2^n - A$
  - $\bar{A} + 1$
- Con palabra de longitud n:
  - $$\text{Valor} = \begin{cases} +\sum_{i=0}^{n-1} 2^i \cdot x_i & \text{si } x_{n-1} = 0 \\ -\text{Valor}(\text{C2}(\text{número})) & \text{si } x_{n-1} = 1 \end{cases}$$
  - Rango: $[-2^{n-1}, -1, 0, (2^{n-1} - 1)]$
- Resolución = 1
- Extensión de signo, se realiza copiando el MSB en los bits de la izquierda

Representaciones numéricas en coma fija
BCD

- Se convierten, uno a uno, los dígitos decimales a binario
- Dos clases:
  - BCD empaquetado
  - BCD desempaquetado
- Representación de BCD desempaquetado (alfanumérico)
- Representación de BCD empaquetado



Valor	BCD	Valor	BCD
0	0000	5	0101
1	0001	6	0110
2	0010	7	0111
3	0011	8	1000
4	0100	9	1001

Suma-resta en Complemento a 2

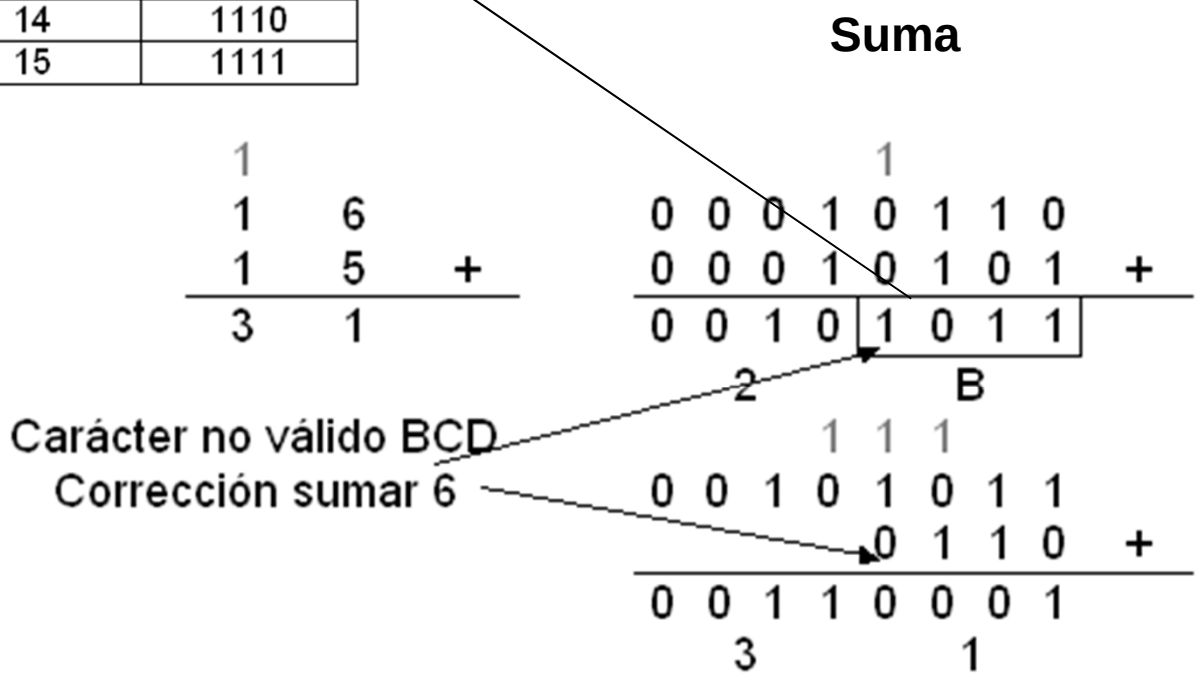
- Se simplifican las operaciones de suma y resta, se hacen sin tener en cuenta los signos de los operandos y el acarreo final se ignora
- La resta se reduce a sumar el número complementado A – B = A + Ca2(B)

- En la suma, el desbordamiento (overflow) se produce si:
  - A>=0 y B>=0 y A + B < 0
  - A < 0 y B < 0 y A + B>=0

- **Ejemplo: A= 0111 y B=0101** : -A= 1001 y -B= 1011
  - A + B = 0111 + 0101 = 1100 y C_f = 0 : Desbordamiento
  - A - B = A + (-B) = 0111 + 1011 = 0010 y C_f = 1
  - -A + B = 1001 + 0101 = 1110 y C_f = 0
  - -A - B = (-A) + (-B) = 1001 + 1011 = 0100 y C_f = 1 : Desbordamiento

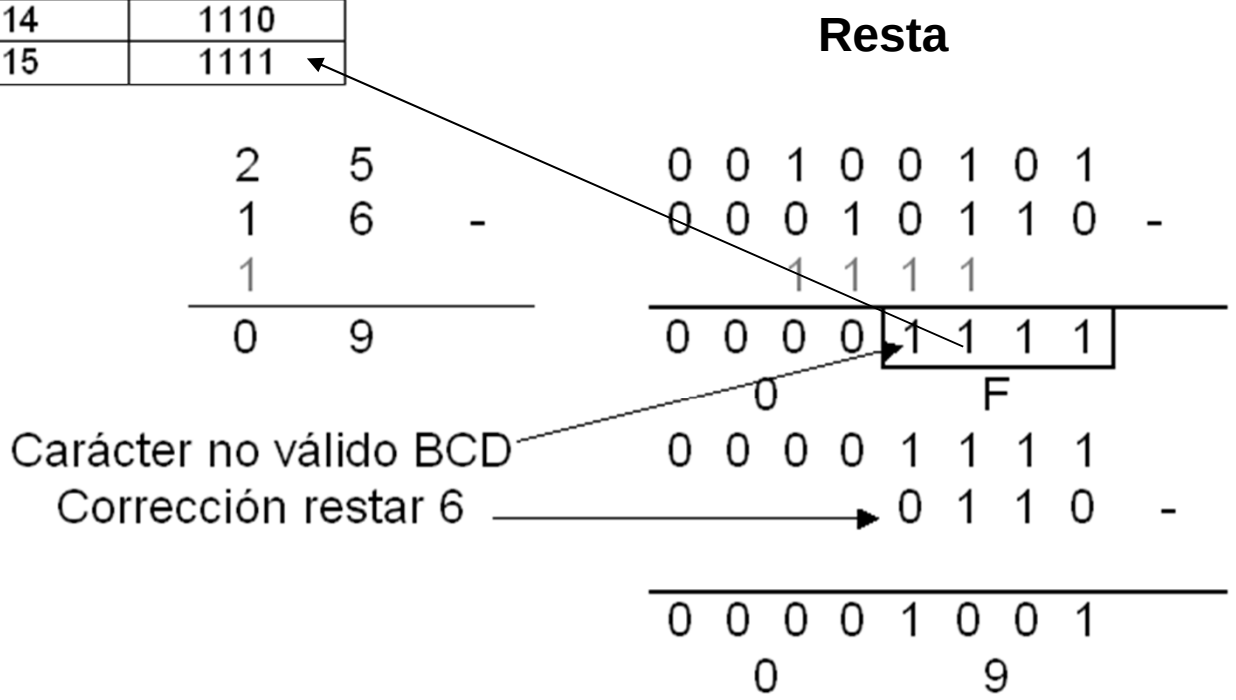
Suma-resta en BCD (I)

Valores válidos BCD		Valores NO válidos BCD	
0	0000	10	1010
1	0001	11	1011
2	0010	12	1100
3	0011	13	1101
4	0100	14	1110
5	0101	15	1111
6	0110		
7	0111		
8	1000		
9	1001		



Suma-resta en BCD (II)

Valores válidos BCD		Valores NO válidos BCD	
0	0000	10	1010
1	0001	11	1011
2	0010	12	1100
3	0011	13	1101
4	0100	14	1110
5	0101	15	1111
6	0110		
7	0111		
8	1000		
9	1001		



Suma en hexadecimal

+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	**0**
2	2	3	4	5	6	7	8	9	A	B	C	D	E	F	**0**	**1**
3	3	4	5	6	7	8	9	A	B	C	D	E	F	**0**	**1**	**2**
4	4	5	6	7	8	9	A	B	C	D	E	F	**0**	**1**	**2**	**3**
5	5	6	7	8	9	A	B	C	D	E	F	**0**	**1**	**2**	**3**	**4**
6	6	7	8	9	A	B	C	D	E	F	**0**	**1**	**2**	**3**	**4**	**5**
7	7	8	9	A	B	C	D	E	F	**0**	**1**	**2**	**3**	**4**	**5**	**6**
8	8	9	A	B	C	D	E	F	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
9	9	A	B	C	D	E	F	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
A	A	B	C	D	E	F	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
B	B	C	D	E	F	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**
C	C	D	E	F	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**
D	D	E	F	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**
E	E	F	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**
F	F	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**

Los valores implican que me llevo 1 de acarreo

```
      1   1
    C A F E
    1 2 F 4  +
  ---------
    D D F 2
```

```
  1 1 1 1
    F A B E
    C A F E  +
  ---------
  1 C 5 B C
```

Modos de representación alfanumérica (I)

- Representaciones alfanuméricas:
  - Codifican mediante un grupo de bits (6, 7, 8, 16) cada uno de los caracteres a representar.
  - Ejemplos de códigos alfanuméricos:
    - 6 bits (64 caracteres posibles) Fieldata y BCDIC
    - 7 bits (128 caracteres posibles) ASCII
    - 8 bits (256 caracteres posibles) ASCII extendido y EBCDIC
    - 16 bits (65536 caracteres posibles) UNICODE

Modos de representación alfanumérica (II)

- Las frases se forman agrupando caracteres. Existen varias alternativas:
  - Cadenas de longitud fija:
    Se define una longitud máxima para todas las cadenas.

P	E	P	E					A	N	T	O	N	I	O		R	O	S	A				

  - Cadenas de longitud variable:
    - Con carácter separador

*	P	E	P	E	*	A	N	T	O	N	I	O	*	R	O	S	A

    - Con longitud explícita

4	P	E	P	E	7	A	N	T	O	N	I	O	4	R	O	S	A

Modos de representación alfanumérica (III)
Código ASCII

Tema 3
Circuitos combinacionales

- 3.1 Introducción a los circuitos combinacionales. Algebra de Boole

- 3.2 Circuitos Combinacionales

3.1
Introducción a los circuitos combinacionales. Algebra de Boole

Índice

- Algebra de Boole. Definición.
- Operaciones lógicas: OR, AND, XOR y NOT
- Puertas lógicas
- Algebra de Boole
  - Postulados
  - Teoremas
  - Funciones lógicas: formas canónicas.
- Tablas de verdad
- Realización de funciones en puertas NAND y NOR

Bibliografía

- Fundamentos de sistemas digitales.
  Thomas Floyd.
  Prentice-Hall.
- Fundamentos de diseño lógico y computadoras.
  M. Morris Mano.
  Prentice-Hall

Algebra de Boole. Definición

- Algebra de Boole es todo conjunto de elementos capaz de adoptar dos valores (**0** y **1**).
- Cada uno de dichos elementos recibe el nombre de **Variable lógica**
- Están definidas dos operaciones: suma lógica y producto lógico (**+** y *****).

Operaciones lógicas

- **OR**

a	b	a OR b
0	0	0
0	1	1
1	0	1
1	1	1

- **NOT**

a	NOT a
0	1
1	0

- **AND**

a	b	a AND b
0	0	0
0	1	0
1	0	0
1	1	1

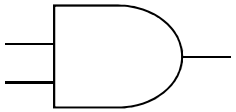
- **XOR**

a	b	a XOR b
0	0	0
0	1	1
1	0	1
1	1	0

Puertas lógicas (I)

a	b	a OR b
0	0	0
0	1	1
1	0	1
1	1	1

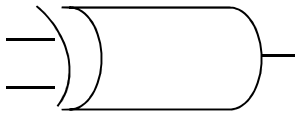
Se representa como: **a + b**



a	b	a AND b
0	0	0
0	1	0
1	0	0
1	1	1

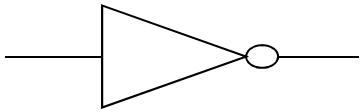
Se representa como: **a · b**

Puertas lógicas (II)



a	b	a XOR b
0	0	0
0	1	1
1	0	1
1	1	0

Se representa como: $a \oplus b$



a	NOT a
0	1
1	0

Se representa como: $\overline{a}$

Algebra de Boole. Postulados

- El Algebra de Boole cumple los siguientes postulados:
- 1. Propiedad conmutativa.
  - **a + b = b + a a * b = b * a**
- 2. Propiedad distributiva
  - **a*(b+c) = a*b + a*c**
  - **a+(b*c) = (a+b) * (a+c)**
- 3. Elemento neutro
  - **a + 0 = a a * 1 = a**
- 4. Elemento Simétrico (complementario):
  - **$a + \bar{a} = 1$ $a * \bar{a} = 0$**

Algebra de Boole. Teoremas

- Se demuestran a partir de los 4 postulados anteriores.
- 1. Ley de idempotencia:
  - **a+a = a a*a = a**
- 2. Ley de complemento:
  - **$a + \bar{a} = 1$ $a * \bar{a} = 0$**
- 3. Ley conmutativa:
  - **a + b = b + a a * b = b * a**
- 4. Ley asociativa:
  - **a+(b+c) = (a+)b+c a*(b*c) = (a*b)*c**
- 5. Ley distributiva:
  - **a*(b+c) = a*b + a*c a+(b*c) = (a+b) * (a+c)**

Algebra de Boole. Teoremas

- 6. Ley de cancelación:
  - **(a*b)+a = a** **(a+b)*a = a**
- 7. Ley de identidad:
  - **a+0 = a** **a*1 = a**
- 8. Ley de dominación:
  - **a+1 = 1** **a*0 = 0**
- 9. Ley de doble complemento
  - $\overline{\overline{a}} = a$
- 10. Leyes de Morgan:
  - $\overline{a+b} = \overline{a} * \overline{b}$ $\overline{a*b} = \overline{a} + \overline{b}$
  - Las leyes de morgan junto con la doble negación nos permiten pasar de expresiones en sumas lógicas a expresiones equivalentes en productos lógicos y viceversa

Algebra de Boole.
Funciones Lógicas. Formas Canónicas

- **Función lógica:** Expresión de variables booleanas o binarias unidas por las operaciones lógicas suma, producto y complementación. Ejemplo:

$$f_1(c,b,a) = a + c \cdot b + c \cdot b \cdot a$$

- **Término canónico:** Producto o suma en el que aparecen todas las variables (o sus complementos) de que depende una función.
- **Función canónica:** formada exclusivamente por términos canónicos
- **Minterm:** término canónico en forma de producto de variables (ej.: $c \cdot b \cdot a$).
  - **Conversión:** Multiplicar cada término no canónico por la suma de las variables que le falten, en su forma normal y complementada.
- **Maxterm:** término canónico en forma de suma de variables (ej.: c+b+a).
  - **Conversión:** Sumar a cada término no canónico productos formados por cada variable que falte y su complementada.

Tablas de verdad

- Es otra forma de representar un función lógica y sirve para obtener el desarrollo en forma canónica de la misma.
- Ejemplo:

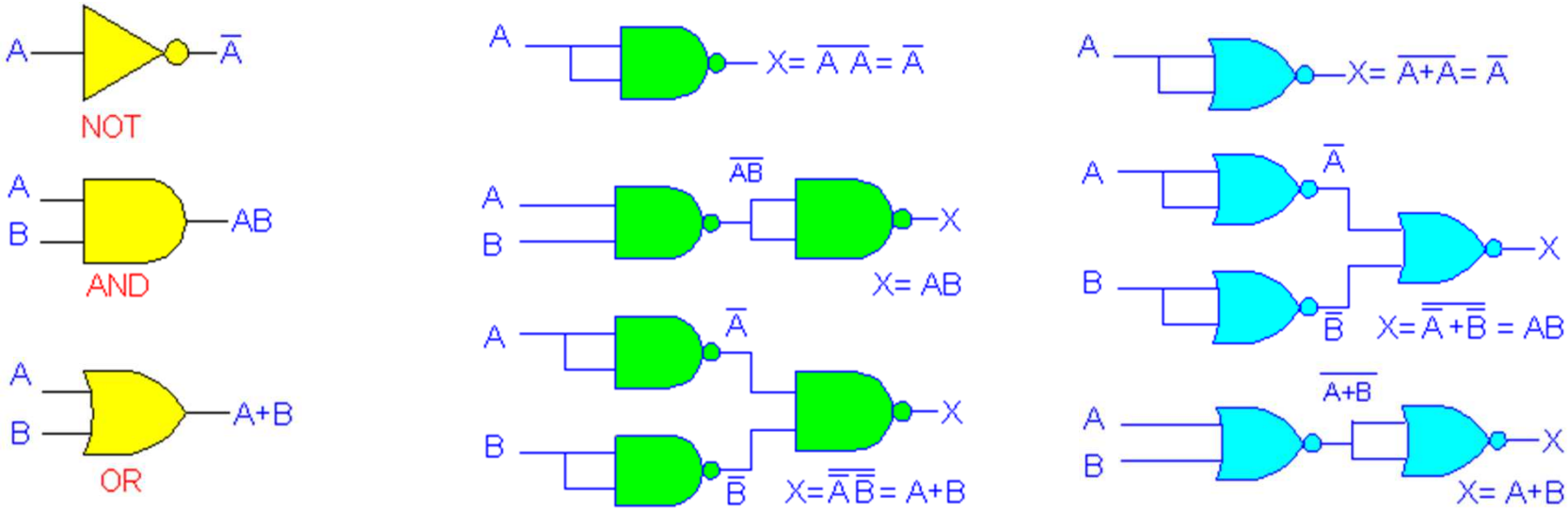
$f(c,b,a) = c \cdot b + \overline{c} \cdot a$

Tabla de verdad:

	c	b	a	f
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

Realización de funciones en puertas NAND y NOR (I)

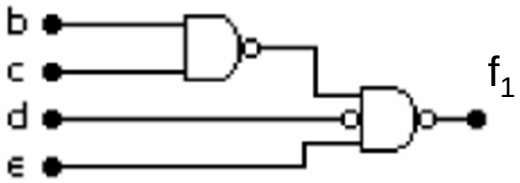
- Las puertas NAND y NOR son puertas Universales, es decir, cualquier función lógica se puede expresar utilizando solo puertas NAND o solo puertas NOR.
- Además son las mas fáciles de construir.



Realización de funciones en puertas NAND y NOR (II)

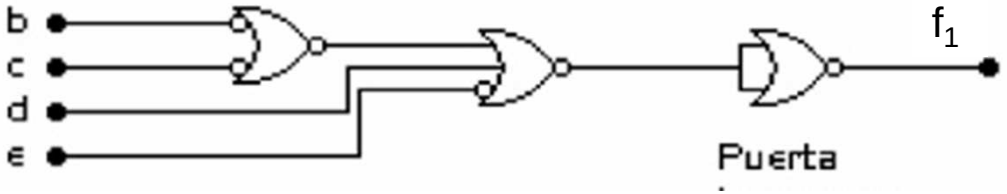
- Para pasar cualquier circuito a puertas NAND o NOR se emplean la ley de la doble negación y las leyes de Morgan.
- Ejemplo: Sea la función $f_1 = bc + d + \bar{e}$
- A NAND
- Pasamos sumas a productos

$$f_1 = \overline{\overline{\overline{bc + d + \bar{e}}}} = \overline{\overline{bc} \cdot \overline{d} \cdot \overline{\bar{e}}} = \overline{bc \cdot d \cdot e}$$



- A NOR
- Pasamos productos a sumas

$$f_1 = \overline{\overline{bc} + d + \bar{e}} = \overline{\overline{b} + \overline{c} + d + \bar{e}} = \overline{\overline{b + c + d + e}}$$



3. 2
Sistemas Combinacionales

Índice

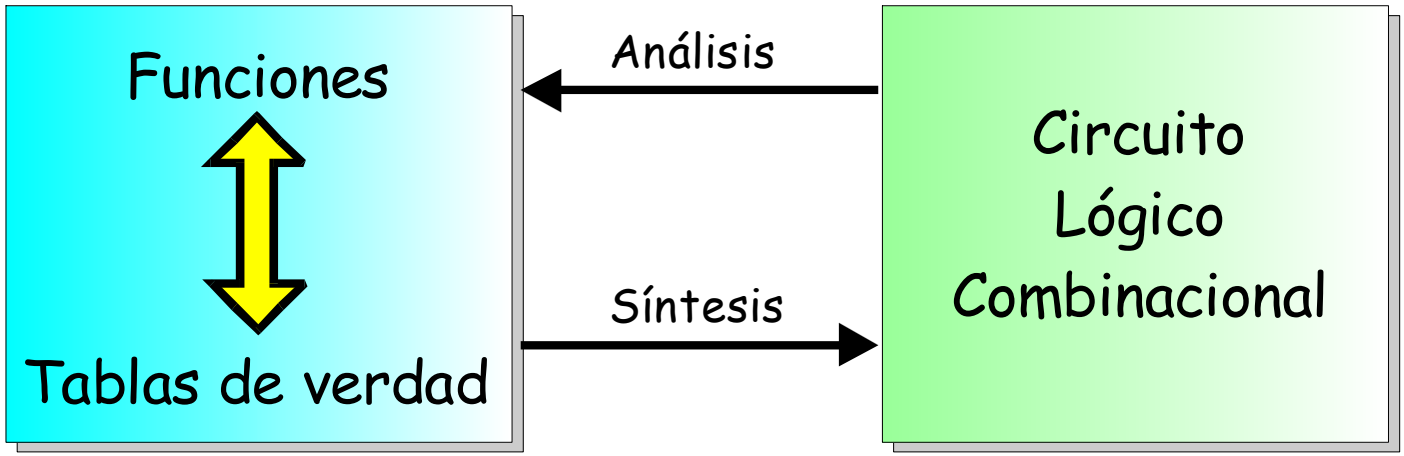
- Circuitos combinacionales: concepto, análisis y síntesis.
- Métodos de simplificación de funciones lógicas.
- Estructuras combinacionales básicas
  - Multiplexores
  - Demultiplexores
  - Decodificadores
  - Codificadores
  - Comparadores
  - Sumadores

Bibliografía

- Fundamentos de sistemas digitales.
  Thomas Floyd.
  Prentice-Hall.
- Fundamentos de diseño lógico y computadoras.
  M. Morris Mano.
  Prentice-Hall

Circuitos combinacionales (I)

- **Concepto:** son aquellos circuitos cuyas salidas, en un determinado instante, son función exclusivamente del valor de las entradas en ese instante.



- **Análisis:** Obtención de la función de salida de un circuito, para cada una de las posibles combinaciones de las “n” variables de entrada.
- **Síntesis**: Obtención del circuito que realiza la función especifícada.

Circuitos combinacionales (II)

Los circuitos combinacionales pueden ser:

- Sistemas *unifuncionales*: tienen una sola función de salida
- Sistemas *multifuncionales*: tienen varias funciones de salida

Una función puede ser:

- *Completa*. Su valor está determinado para todas las posibles combinaciones de las variables de entrada.
- *Incompleta*. Existen algunas combinaciones de entrada para las cuales el valor de la función es indeterminado.

  Causas:

  - Existencia de combinaciones de las variables que nunca se presentan.
  - Existencia de combinaciones de las variables para las que el valor que tome la función sea indiferente.

Síntesis de circuitos combinacionales

Proceso a seguir para obtener un circuito combinacional óptimo:

- Establecer la tabla de verdad, desde el enunciado del problema.
- Obtener la función canónica expresada en minterms o en maxterms, a partir de la tabla de verdad.
- Simplificar la función canónica, bien en forma algebraica (aplicando teoremas y postulados del Álgebra de Boole), bien mediante la aplicación de métodos gráficos sencillos (Karnaugh) o con el método tabular numérico de Quine-McCluskey.
- Realizar la función simplificada, mediante las oportunas puertas lógicas.

Métodos de simplificación de funciones lógicas (I)

Método algebraico

- Es el método básico de simplificación de funciones y consiste en aplicar directamente la propiedad distributiva a los términos de la función, eliminando variables. Por ejemplo:

$$f_1(d,c,b,a) = d\cdot c\cdot b\cdot a + d\cdot c\cdot b\cdot \overline{a} = d\cdot c\cdot b\cdot (a+\overline{a}) = d\cdot c\cdot b\cdot 1 = d\cdot c\cdot b$$

$$f_2(d,c,b,a) = (d+c+b+a)\cdot (d+c+\overline{b}+a) = (d+c+b\cdot \overline{b}+a) = (d+c+a)$$

- Sin embargo, pocas veces viene expresada la función de forma que sea fácilmente aplicable este método.

Métodos de simplificación de funciones lógicas (II)

Método de Karnaugh

- Método tabular gráfico que se basa en los llamados “mapas de Karnaugh”, consistentes en una tabla de cuadros, cada uno de los cuales representa un término canónico.

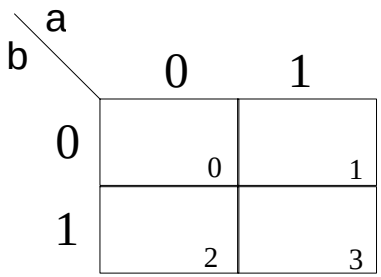
  Estos cuadros están distribuidos de tal modo que dos cualesquiera de ellos, contiguos físicamente, corresponden a términos canónicos adyacentes.

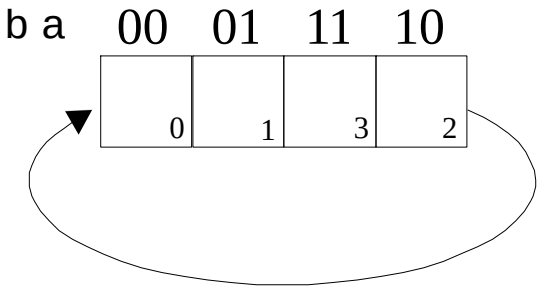
- ***Términos canónicos adyacentes***: son aquellos para los que sus respectivas configuraciones binarias difieren entre sí en un único bit.

  Se pueden definir también como aquellos términos a los que se les puede aplicar la propiedad distributiva para simplificar una variable.

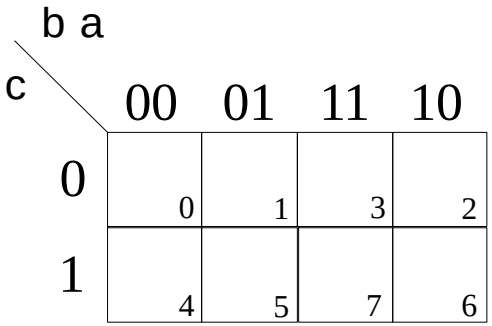
Métodos de simplificación de funciones lógicas (III)

- Mapa de Karnaugh para funciones de ***dos*** variables

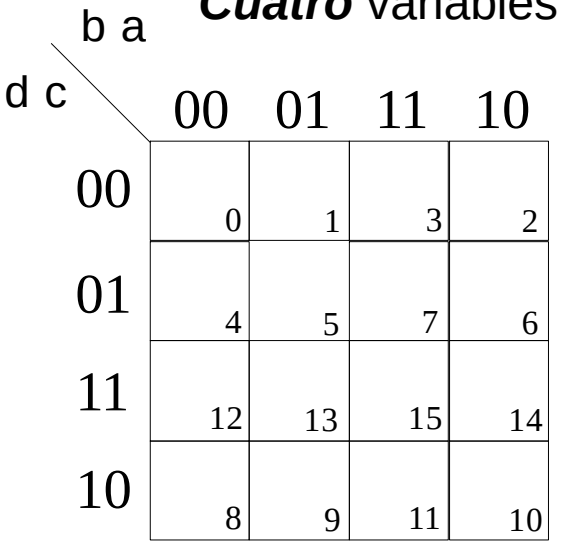




Tres variables



Cuatro variables



Métodos de simplificación de funciones lógicas (IV)

Procedimiento de simplificación mediante los mapas de Karnaugh

1.- Dibujar el mapa adecuado para la función a simplificar (2, 3 ó 4 variables).

2.- Marcar los cuadros correspondientes a los minterms o maxterms (“1”).

3.- Agrupar, mediante una curva cerrada, el máximo número posible de elementos adyacentes (potencia de 2 ⇨2, 4, 8, 16). A continuación lo mismo con los que queden y asi, sucesivamente, hasta que no reste ningún elemento marcado (con adyacente también marcado) sin agrupar.

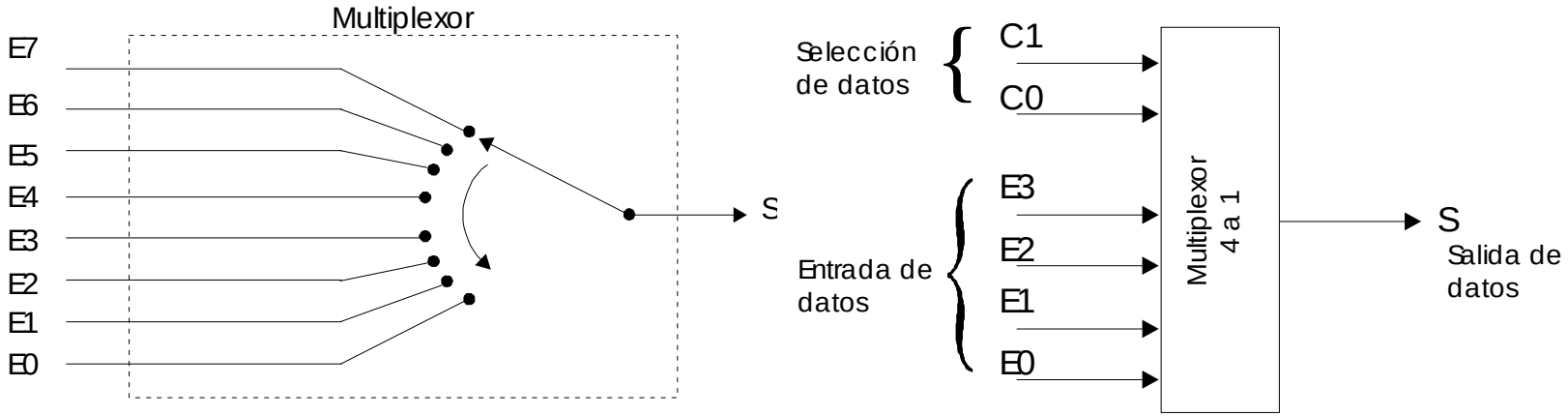
4.- Escribir la función mediante los términos simplificados obtenidos.

Criterio: en cada grupo desaparece la variable o variables cuyo valor es “0” en la mitad de los cuadros del grupo, y “1” en la otra mitad. Las variables que permanecen son tomadas como “no negadas” si su valor es 1 en todo el grupo de cuadros, y como “negadas” si su valor es 0.

Si algún bit no tiene ningúno adyacente seguirá como término canónico

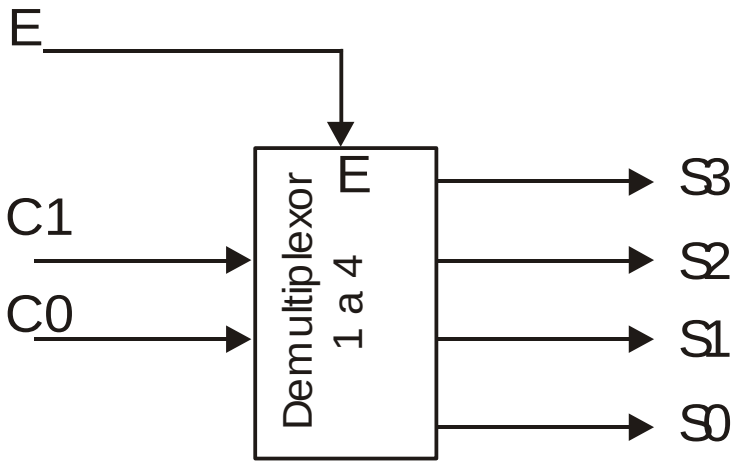
Estructuras Combinacionales básicas (I)
Multiplexores

- Un multiplexor es un circuito que tiene 2^n entradas de información (canales), una sola salida y un mecanismo de selección que determina cuál de las entradas es la que se transfiere su información a la única salida. Se comporta como un conmutador de entrada múltiple y salida única, pero cuyo control no es mecánico, sino electrónico.



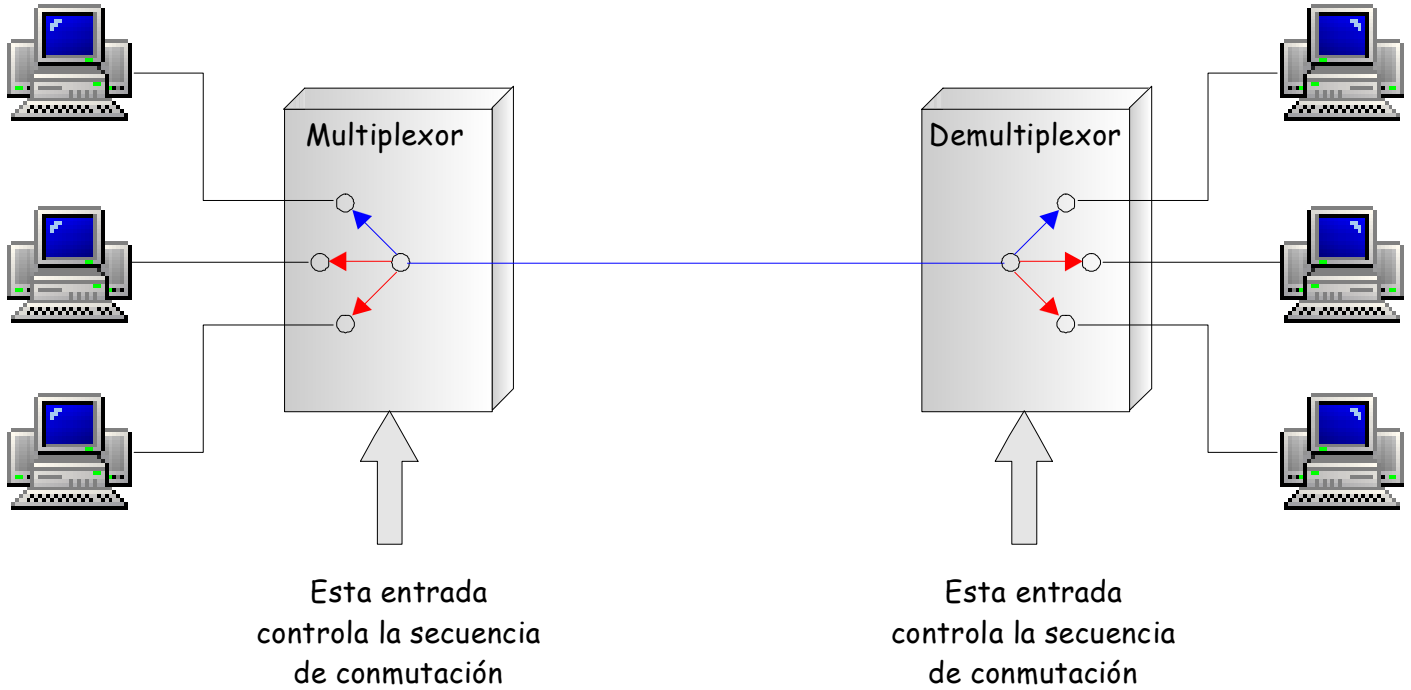
Estructuras Combinacionales básicas (II)
Demultiplexores

- Realizan la función inversa de un multiplexor, es decir, se comportan como conmutadores de entrada única y salida múltiple, existiendo un mecanismo de control que selecciona la salida hacia la que se envía la información de entrada.
- En general un demultiplexor tiene una única entrada de información, 2^n salidas y “n” entradas de control en las que se introduce el número binario correspondiente a la salida seleccionada.



Estructuras Combinacionales básicas (III)
Multiplexores /Demultiplexores

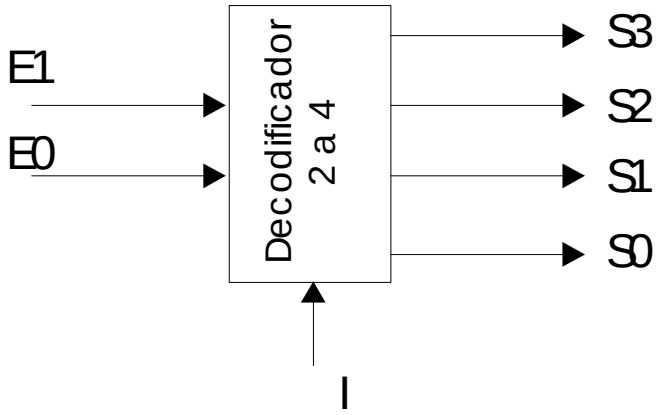
Aplicaciones básicas



Estructuras Combinacionales básicas (IV)
Decodificadores (1)

- Son circuitos digitales que tienen como entrada (n bits) la información codificada en binario, y tantas salidas como posibles configuraciones binarias distintas de entrada (2^n), activándose en cada momento una sola de ellas, la correspondiente a la combinación binaria aplicada a la entrada.

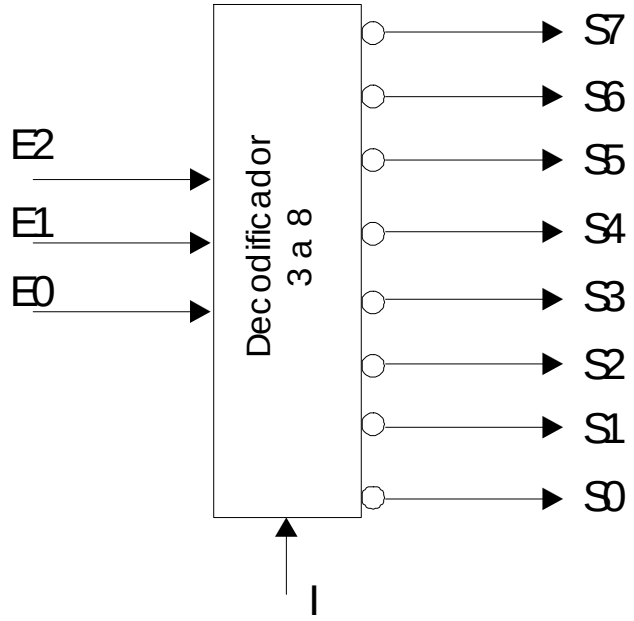
Decodificador 2 a 4



I	E1	E0	S3	S2	S1	S0
0	0	0	0	0	0	1
0	0	1	0	0	1	0
0	1	0	0	1	0	0
0	1	1	1	0	0	0
1	X	X	0	0	0	0

Estructuras Combinacionales básicas (V)
Decodificadores (2)

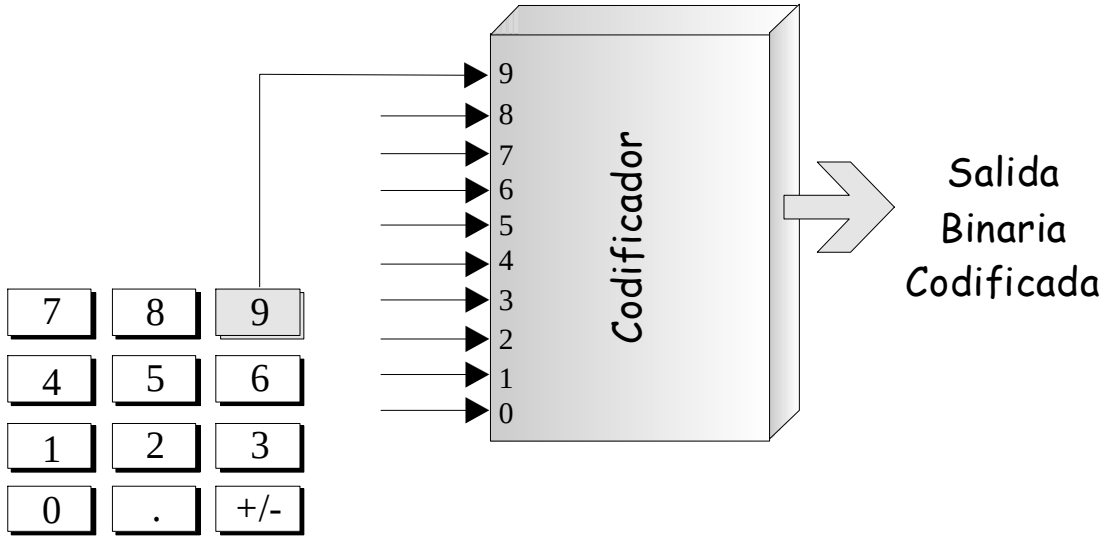
Decodificador 3 a 8



I	E2	E1	E0	S7	S6	S5	S4	S3	S2	S1	S0
0	0	0	0	1	1	1	1	1	1	1	0
0	0	0	1	1	1	1	1	1	1	0	1
0	0	1	0	1	1	1	1	1	0	1	1
0	0	1	1	1	1	1	1	0	1	1	1
0	1	0	0	1	1	1	0	1	1	1	1
0	1	0	1	1	1	0	1	1	1	1	1
0	1	1	0	1	0	1	1	1	1	1	1
0	1	1	1	0	1	1	1	1	1	1	1
1	X	X	X	1	1	1	1	1	1	1	1

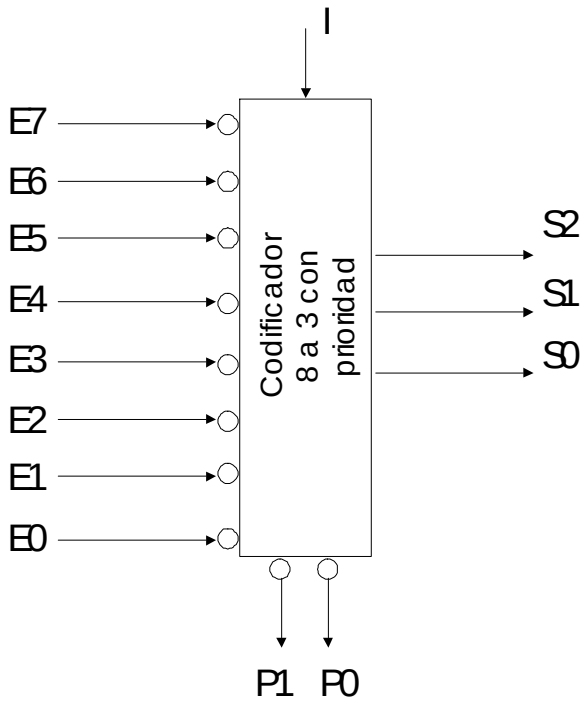
Estructuras Combinacionales básicas (VI)
Codificadores (1)

- Realizan la función inversa de los decodificadores, es decir, poseen en general N entradas y “n” salidas de código en las que aparece codificado en binario el valor de la entrada que ha sido activada ($N \leq 2^n$).
  - Codificadores sin prioridad.
  - Codificadores con prioridad.



Estructuras Combinacionales básicas (VII)
Codificadores (2)

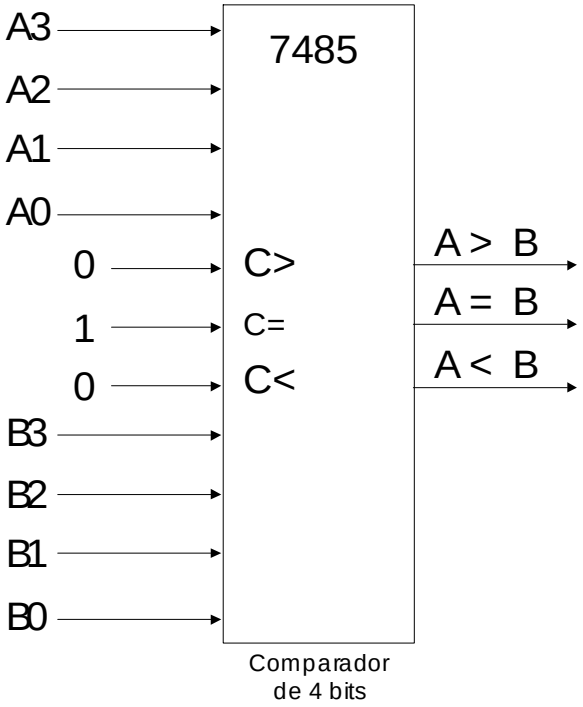
- Codificador 8 a 3 con prioridad



I	E7	E6	E5	E4	E3	E2	E1	E0	S2	S1	S0	P1	P0
1	X	X	X	X	X	X	X	X	0	0	0	1	1
0	1	1	1	1	1	1	1	1	0	0	0	1	0
0	X	X	X	X	X	X	X	0	0	0	0	0	1
0	X	X	X	X	X	X	0	1	0	0	1	0	1
0	X	X	X	X	X	0	1	1	0	1	0	0	1
0	X	X	X	X	0	1	1	1	0	1	1	0	1
0	X	X	X	0	1	1	1	1	1	0	0	0	1
0	X	X	0	1	1	1	1	1	1	0	1	0	1
0	X	0	1	1	1	1	1	1	1	1	0	0	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1

Estructuras Combinacionales básicas (VIII)
Comparadores

- Un comparador de “n” bits es un circuito combinacional que tiene por entradas dos números binarios de “n” bits cada uno (A,B), determinando a su salida si uno es mayor, menor o igual que el otro. Normalmente dispone de tres salidas: A>B, A=B y A<B



Ay B	C>	C=	C<	A>B	A=B	A<B
A >B	X	X	X	1	0	0
A <B	X	X	X	0	0	1
A = B	0	0	1	0	0	1
A = B	0	1	0	0	1	0
A = B	1	0	0	1	0	0

Tema 4.2
Diseño de Sistemas Secuenciales

Contenidos

- Introducción
- Máquinas de Moore
- Máquinas de Mealy
- Ejemplos

Bibliografía

- Diseño Digital.
  M. Morris Mano. Prentice-Hall
- Introduccion al Diseño Lógico Digital
  John P. Hayes. Addison-Wesley

Introducción

- Manera sistemática de diseñar circuitos digitales que pasan por diferentes estados.

  Ejemplos: Contadores, semáforos, máquinas expendedoras...

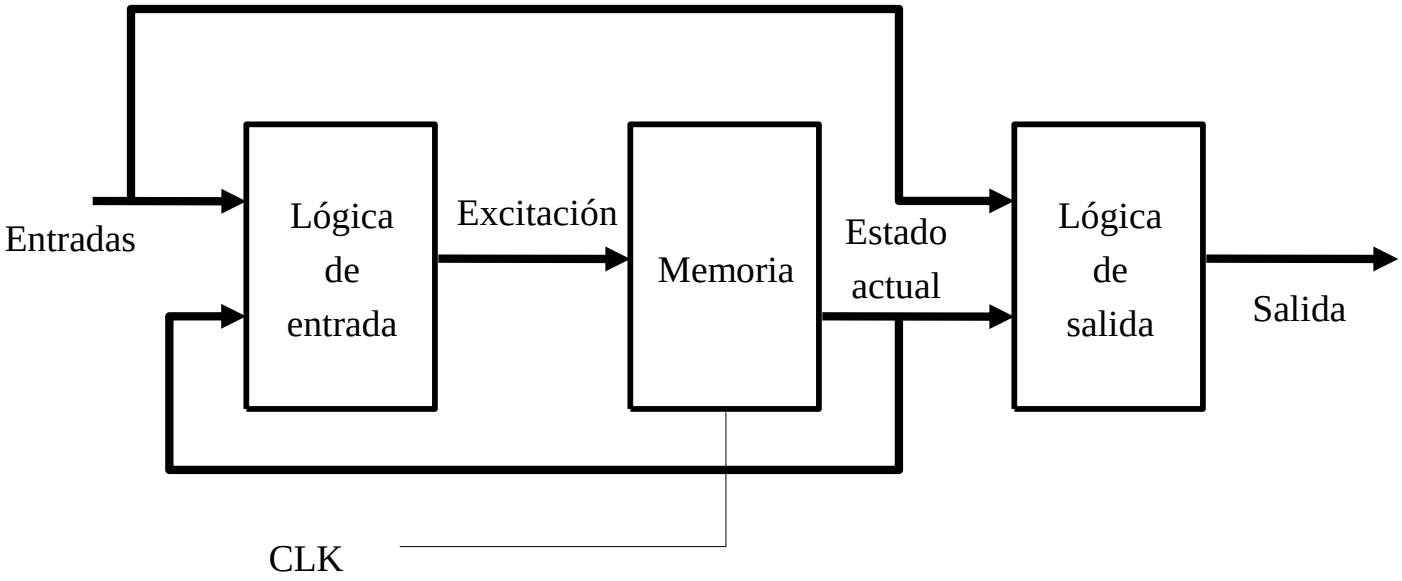
- Generalmente se llaman ***Máquinas o Autómatas finitos***

- Dos tipos:

  Máquinas de **Mealy**

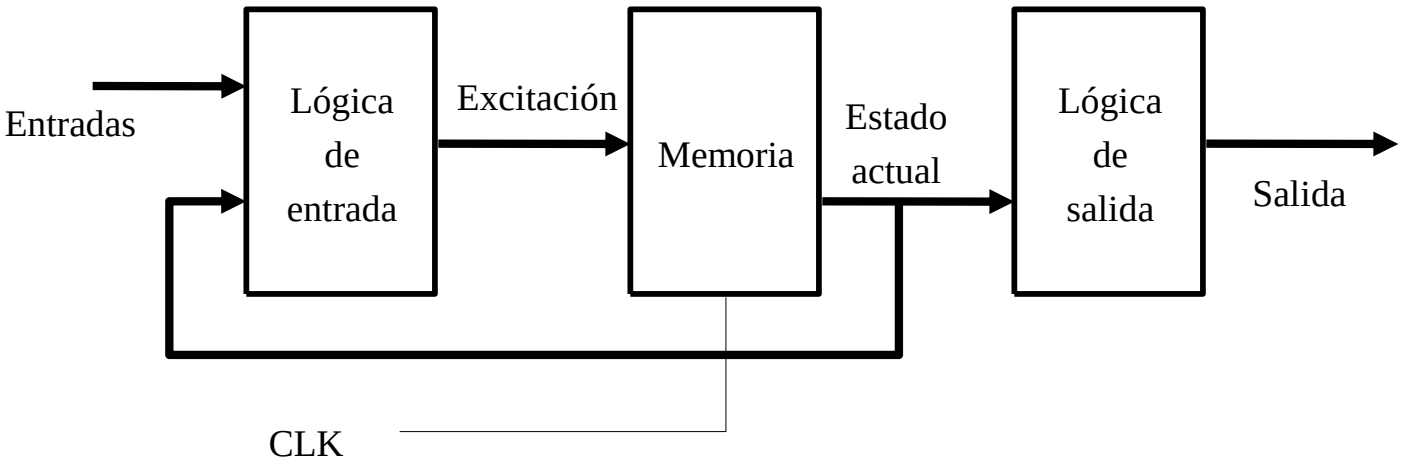
  Máquinas de **Moore**

Máquina de Mealy



La **salida** es una función tanto de la **entrada** como del **estado actual**

Máquina de Moore



La **salida** es una función solo del **estado actual**

Secuencia de diseño

1- Entender las especificaciones del problema

2- Elegir Mealy/Moore basándose en la simplicidad

3- Dibujar el diagrama de estado

4- Codificar los estados y elegir los biestables

5- Obtener la función de salida

6- Escribir la tabla de transición y excitación

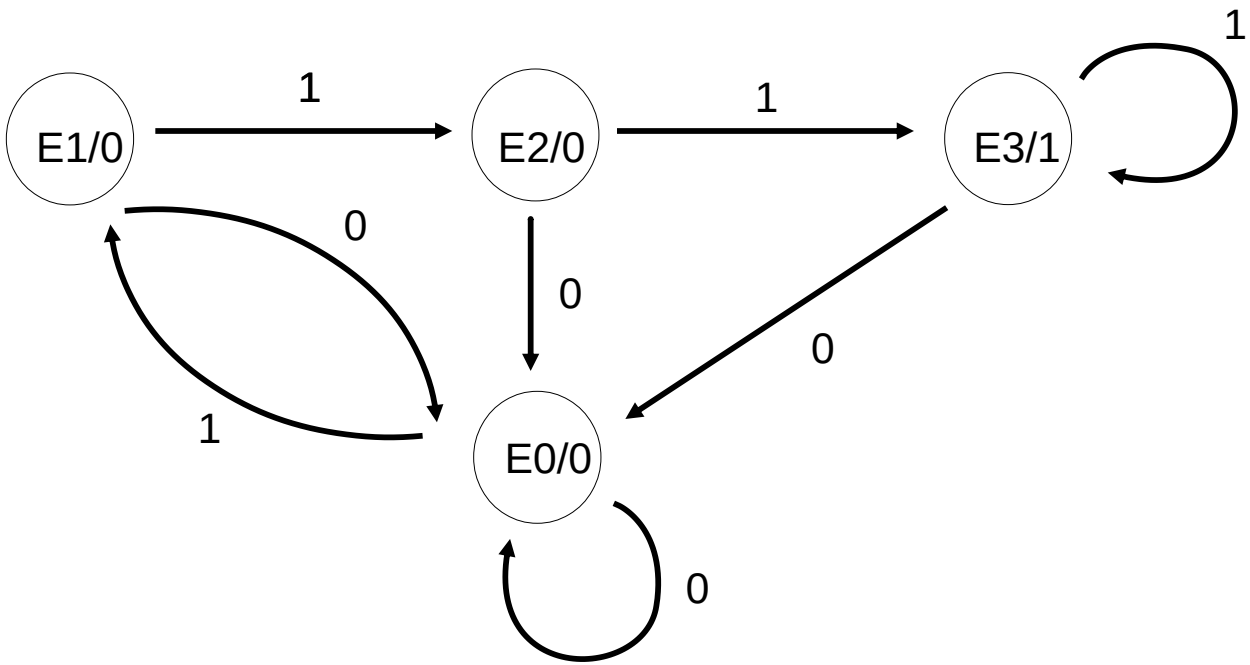
7- Obtener y simplificar las funciones de excitación

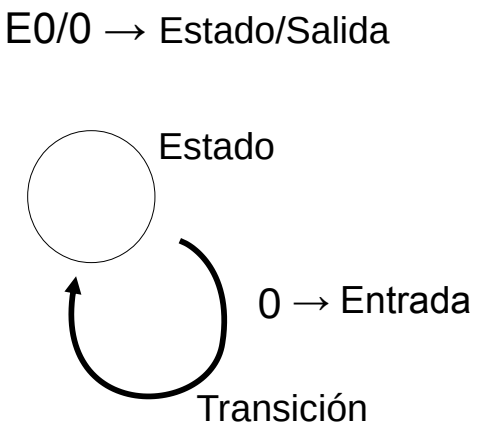
8- Diseñar el circuito

Ejemplo 1: Moore (I)

Diseñe un autómata de Moore que detecte una secuencia de tres o más “1” en la entrada: ...111...

Diagrama de estados:





Ejemplo 1: Moore (II)

Codificar estados y salida:

- Hay 4 estados luego necesitamos 2 bits para codificarlos
- Usamos dos biestables JK
- Codificación:

Estados	JKs	Salida
	Q1 Q0	Z
E0	0 0	0
E1	0 1	0
E2	1 0	0
E3	1 1	1

Obtener la función de salida:

$$Z = Q1\ Q0$$

Ejemplo 1: Moore (III)

Escribir la tabla de transición y excitación:

Estado actual	Entrada	Estado sgte.	Excitación JK	
$Q1^t$ $Q0^t$	Y	$Q1^{t+1}Q0^{t+1}$	J1 K1	J0 K0
E0: 0 0	0	0 0	0 X	0 X
E0: 0 0	1	0 1	0 X	1 X
E1: 0 1	0	0 0	0 X	X 1
E1: 0 1	1	1 0	1 X	X 1
E2: 1 0	0	0 0	X 1	0 X
E2: 1 0	1	1 1	X 0	1 X
E3: 1 1	0	0 0	X 1	X 1
E3: 1 1	1	1 1	X 0	X 0

Tabla excitación JK

Q^t Q^{t+1}	J K
0 0	0 X
0 1	1 X
1 0	X 1
1 1	X 0

Example 1: Moore (IV)

Obtener y simplificar las funciones de excitación:

-Obtener J1, K1, J0 y K0 en función de $Q1^{t+1}$, $Q0^{t+1}$ e Y con Karnaugh

-Ejemplo

J1 = Q0 Y

Q1 \ Q0Y	00	01	11	10
0			1	
1	X	X	X	X

Haciendo el resto de mapas de Karnaugh:

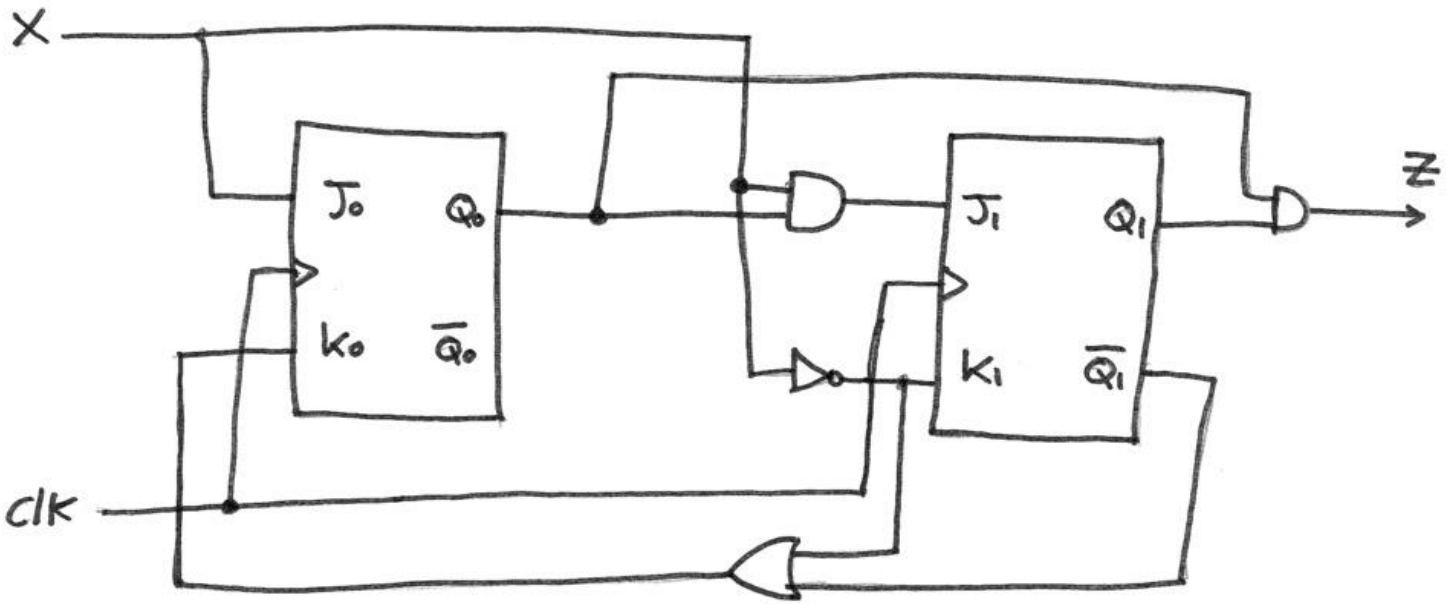
K1 = Y!

J0 = Y

K0 = Q1! + Y!

Example 1: Moore (V)

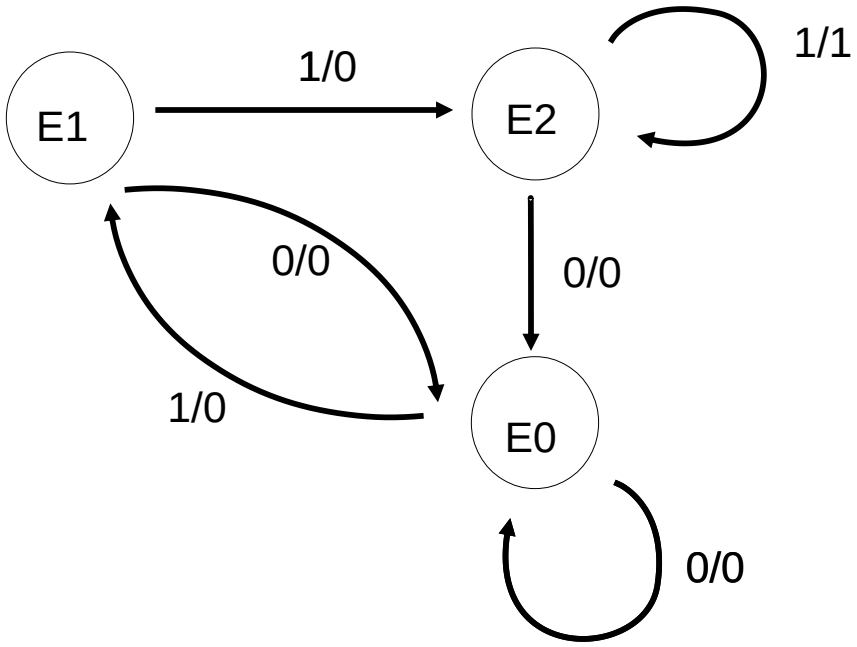
Implement the circuit:

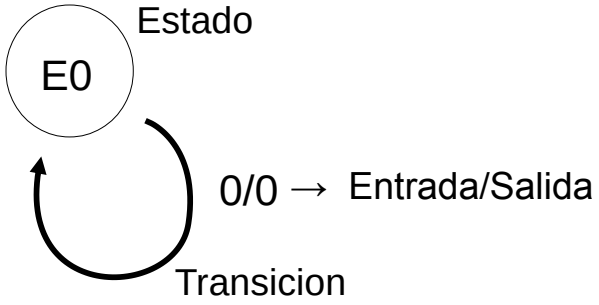


Ejemplo 2: Mealy (I)

Diseñe un autómata de Mealy que detecte una secuencia de tres o más “1” en la entrada: ...111...

Diagrama de estados:





Ejemplo 2: Mealy (II)

Codificar estados y salida:

- Hay 3 estados luego necesitamos 2 bits para codificarlos
- Usamos dos biestables JK
- Codificación:

Estados	JKs		Entrada	Salida
	Q1	Q0	Y	Z
E0	0	0	0	0
E0	0	0	1	0
E1	0	1	0	0
E1	0	1	1	0
E2	1	0	0	0
E2	1	0	1	1
E3	1	1	0	X
E3	1	1	1	X

Obtener la función de salida:

Q1 \ Q0X	00	01	11	10
0				
1		1	X	X

Z = Y Q1

Ejemplo 2: Mealy (III)

Escribir la tabla de transición y excitación:

Estado actual	Entrada	Estado sgte.	Excitación JK	
$Q1^t$ $Q0^t$	Y	$Q1^{t+1}Q0^{t+1}$	J1 K1	J0 K0
E0: 0 0	0	0 0	0 X	0 X
E0: 0 0	1	0 1	0 X	1 X
E1: 0 1	0	0 0	0 X	X 1
E1: 0 1	1	1 0	1 X	X 1
E2: 1 0	0	0 0	X 1	0 X
E2: 1 0	1	1 0	X 0	0 X
E3: 1 1	0	X X	X X	X X
E3: 1 1	1	X X	X X	X X

Tabla excitación JK

Q^t Q^{t+1}	J K
0 0	0 X
0 1	1 X
1 0	X 1
1 1	X 0

Ejemplo 2: Mealy (IV)

Obtener y simplificar las funciones de excitación:

-Obtener J1, K1, J0 y K0 en función de $Q1^{t+1}$, $Q0^{t+1}$ e Y con Karnaugh

-Ejemplo

J1 = Q0 Y

Q1 \ Q0Y	00	01	11	10
0			1	
1	X	X	X	X

Haciendo el resto de mapas de Karnaugh:

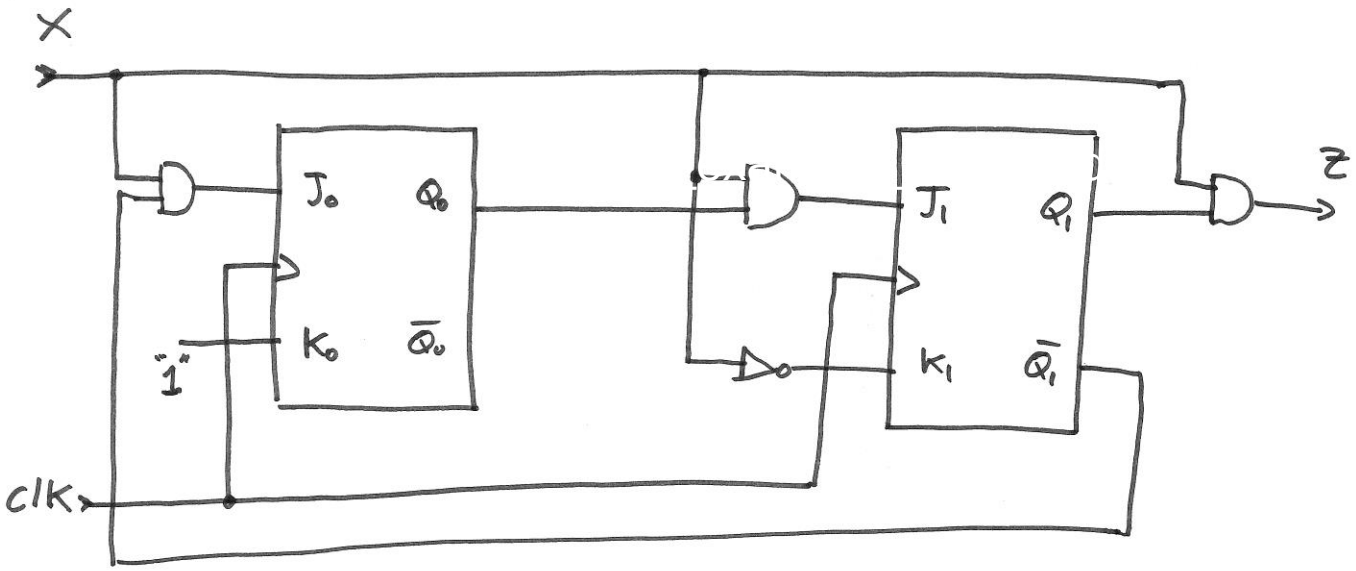
K1 = Y!

J0 = Y Q1!

K0 = 1

Ejemplo 2: Mealy (V)

Diseñar el circuito:



Tema 4.
Sistemas Secuenciales

Índice

- Conceptos básicos
- Biestables: concepto y tipos
- Registros
  - Almacenamiento
  - Desplazamiento
    - Serie-Serie / Serie-Paralelo
    - Paralelo-Serie / Paralelo-Paralelo
    - Universales
- Contadores
  - Asíncronos y síncronos
  - Ascendentes y descendentes
  - Módulo N

Bibliografía

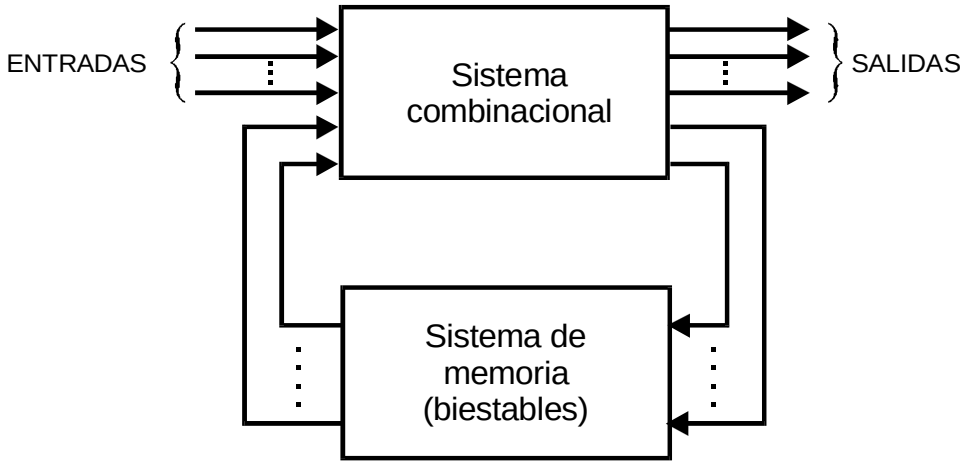
•Fundamentos de sistemas digitales
Thomas Floyd.
Prentice-Hall

• Fundamentos de diseño lógico y computadoras.
M. Morris Mano.
Prentice-Hall

Conceptos Básicos (I)

Circuitos secuenciales. Son aquellos en los que el valor actual de las salidas depende no sólo del valor actual de las entradas sino también de las situaciones por las que pasó el circuito anteriormente (valor anterior de las propias salidas).

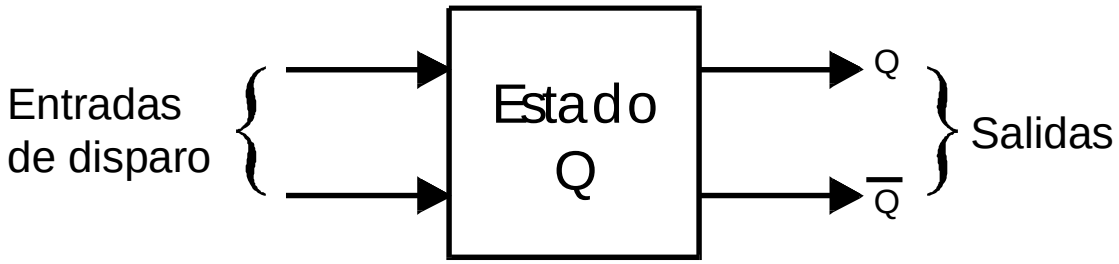
- Un sistema secuencial consta de dos bloques diferenciados: un sistema de memoria y un sistema combinacional asociado a él.



Generalmente, un sistema secuencial será un sistema realimentado.

Biestables (I)

- El circuito de memoria almacena la información en binario, por lo que ha de utilizar elementos capaces de retener dicho tipo de información. El elemento básico de almacenamiento es el **biestable**.
- Los biestables son circuitos lógicos elementales capaces de permanecer en uno de los dos posibles estados estables (Q=0 o Q=1), aún después de desaparecer la señal de entrada (entrada de disparo) que lo provocó. Almacenan la información binaria de un bit y permiten mantenerla como salida estable, en ausencia de las entradas.
- Los hay de muchos tipos, pero su esquema general es el siguiente:



Biestables (II)

Clasificación:

Según la **lógica** de disparo

R-S **J-K** **D** **T**

Según el **sincronismo** en el disparo

- **Síncronos.** La transición sólo está permitida coincidiendo con una señal de reloj o sincronismo. El impulso de reloj no contiene información en el sentido de que cambio va a ocurrir, simplemente sincroniza el cambio.
- **Asíncronos.** La transición puede producirse en cualquier instante, sólo depende de las entradas de disparo.

Según el **tipo de señal** de disparo

- **Por Nivel.** La activación del biestable se realiza en función del nivel Alto o Bajo de las variables de entrada.
- **Por Pulso.** La activación del biestable se realiza en función del cambio de nivel (flanco de subida o flanco de bajada) de las variables de entrada.

Biestables (III)

R-S NOR

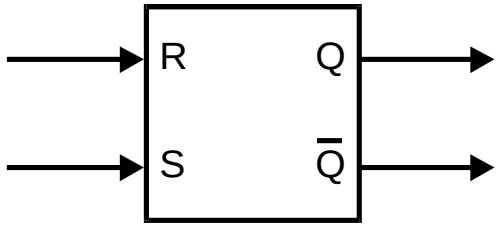
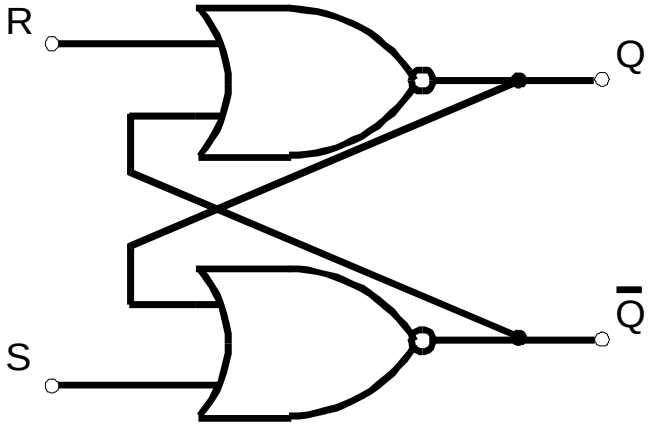


Tabla de excitación		
R^t	S^t	Q^{t+1}
0	0	Q^t
0	1	1
1	0	0
1	1	I



Estado transitorio indeterminado, debido al retardo de las puertas

Biestables (IV)

RS-NAND

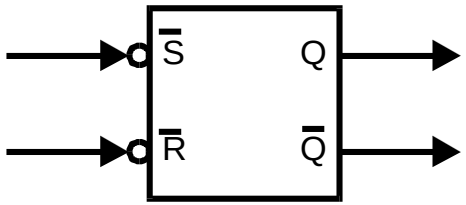
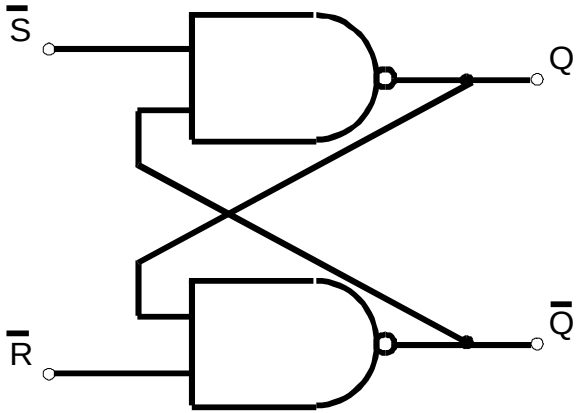


Tabla de excitación		
R^t	S^t	Q^{t+1}
0	0	I
0	1	0
1	0	1
1	1	Q^t

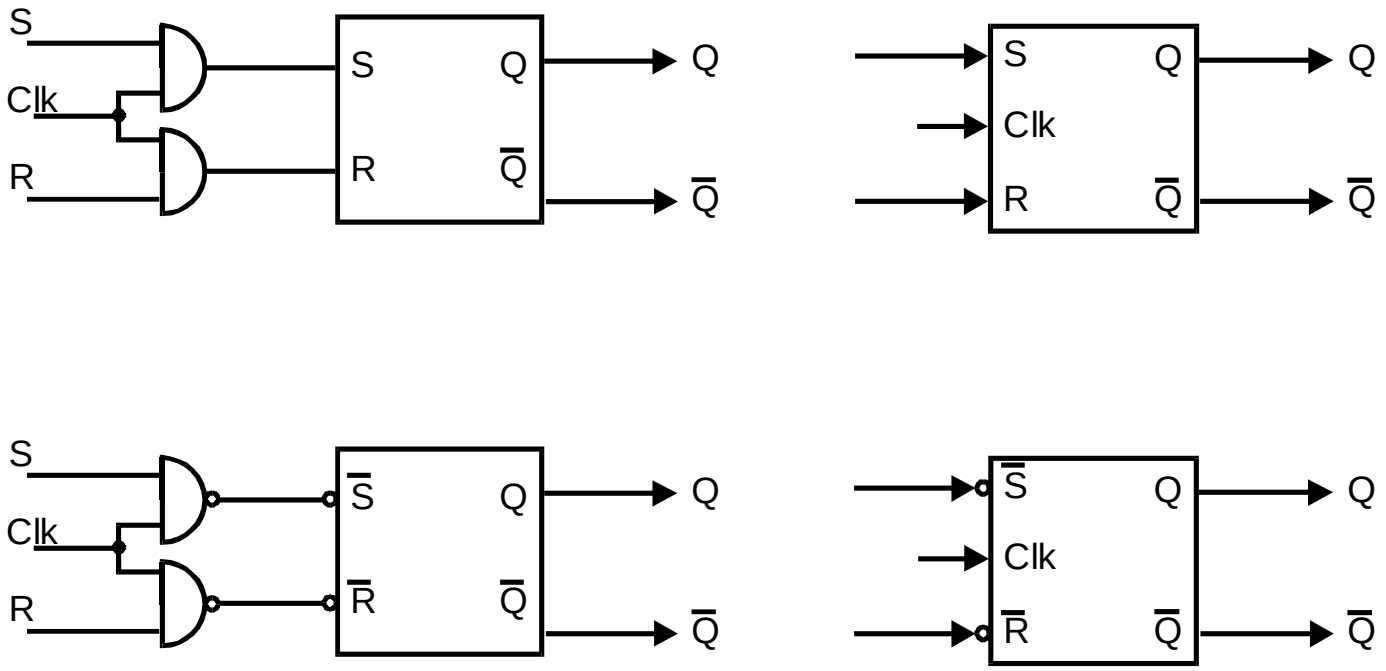


Estado transitorio indeterminado, debido al retardo de las puertas

Biestables (V)

R-S síncrono

- Activo por nivel



Biestables (VI)

Biestable R-S síncrono con entradas asíncronas

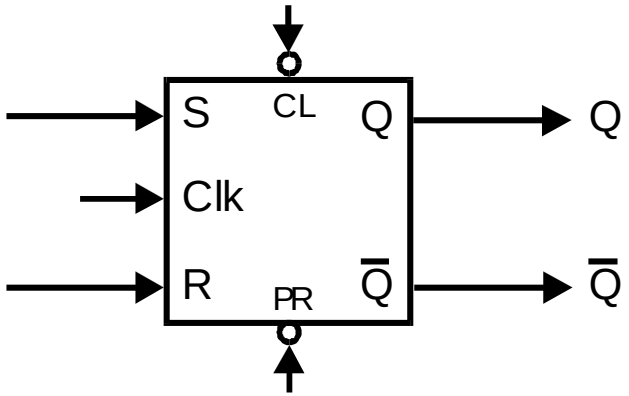


Tabla de excitación

Pr	Cl	S	R	Clk	Q^{t+1}
0	1	X	X	X	1
1	0	X	X	X	0
0	0	X	X	X	X
1	1	0	0	⎍	Q^t
1	1	1	0	⎍	1
1	1	0	1	⎍	0
1	1	1	1	⎍	X

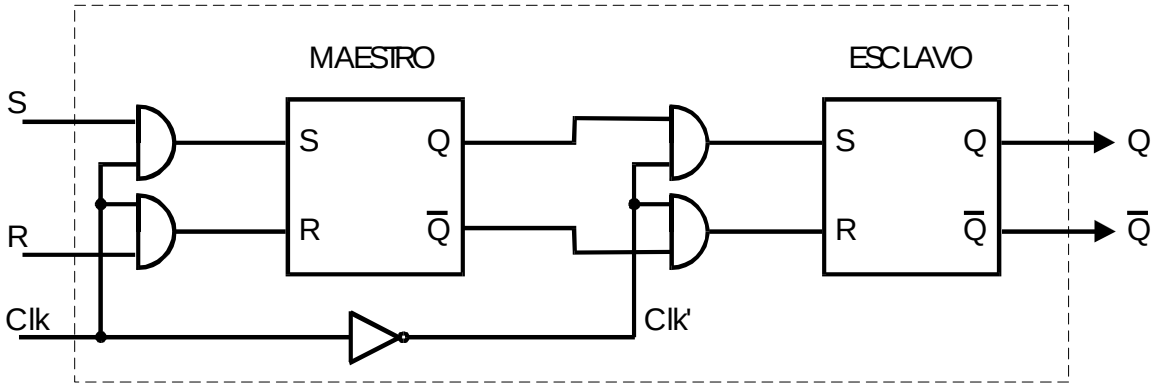
Sólo tiene en cuenta las entradas de excitación R-S, cuando las asíncronas Preset y Clear no están activas.

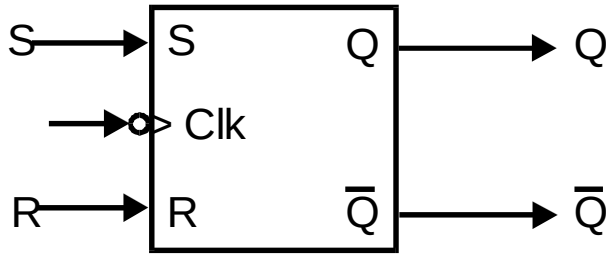
Biestables (VII)

R-S Master-Slave

La estructura “Master-Slave” se introduce para resolver los “problemas de tiempos”, que conducen a salidas incorrectas

- Activo por flanco (de bajada ↘)





Biestables (VIII)

Biestable J-K asíncrono

Es como el R-S, eliminando las situaciones de indeterminación.

J ~ S y K ~ R.

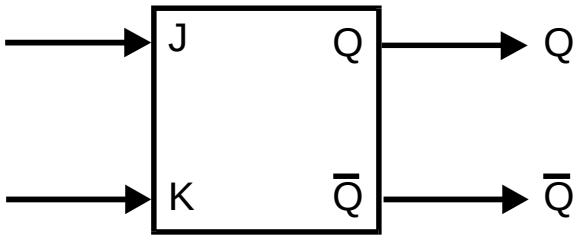
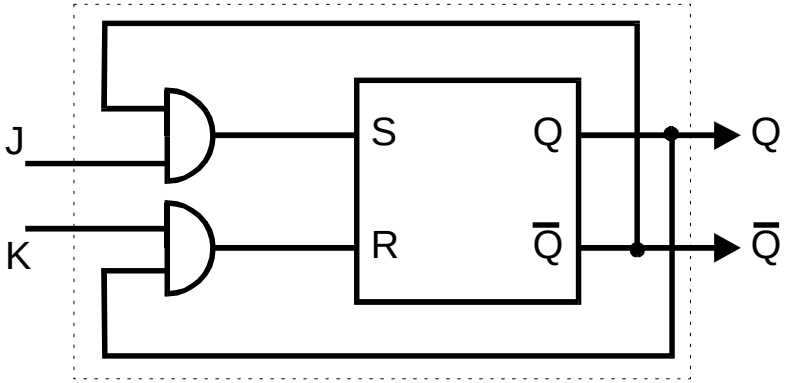
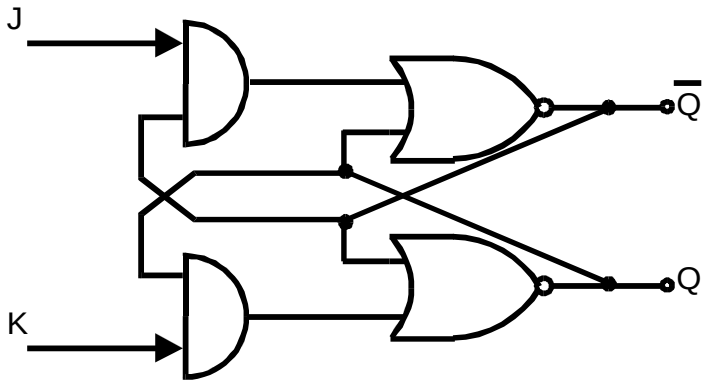


Tabla de excitación

J^t	K^t	Q^{t+1}
0	0	Q^t
0	1	0
1	0	1
1	1	$\overline{Q^t}$

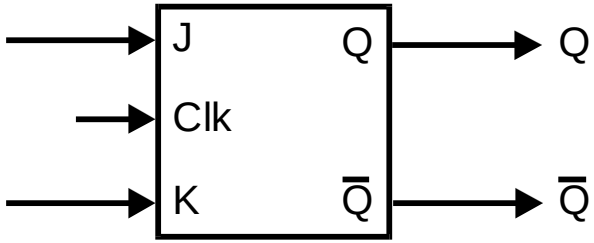


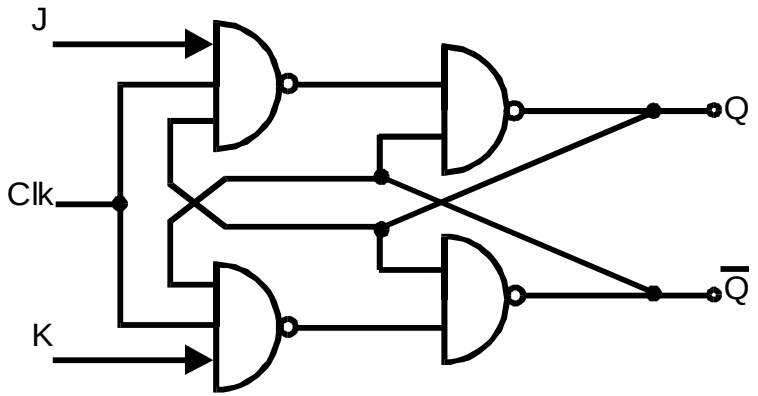


Biestables (IX)

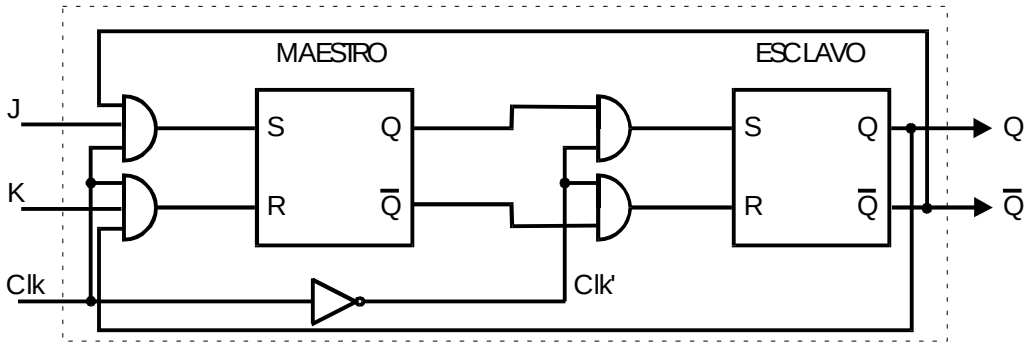
Biestable J-K síncrono

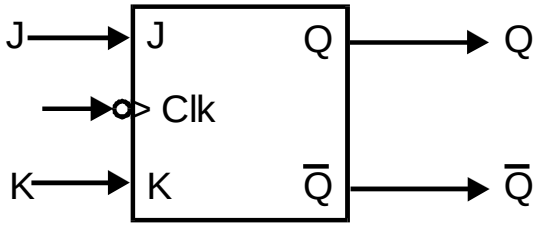
- Activo por nivel





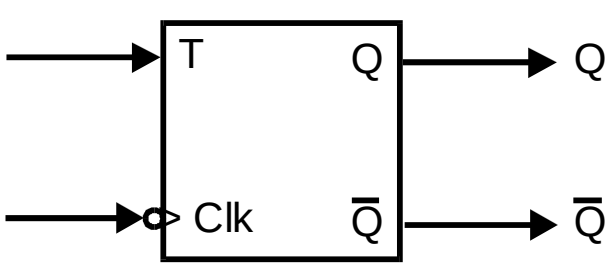
- Activo por flanco (M-S)





Biestables (X)

Biestable Tipo T



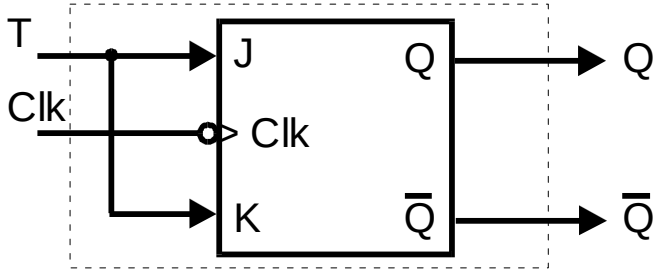


Tabla de excitación

T^t	Clk	Q^{t+1}
0	↓	Q^t
1	↓	$\overline{Q^t}$

Biestables (XI)

Biestable tipo D

- D latch (activo por nivel)

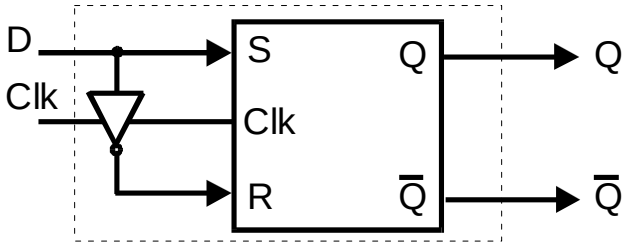
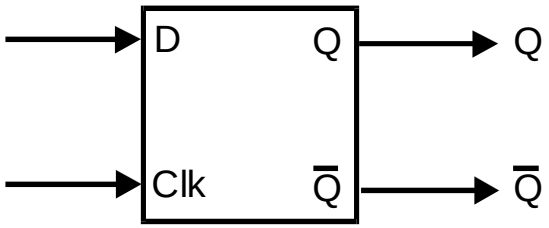


Tabla de excitación		
D^t	Clk	Q^{t+1}
X	0	Q^t
D	1	D



- D flip-flop (activo por flanco)

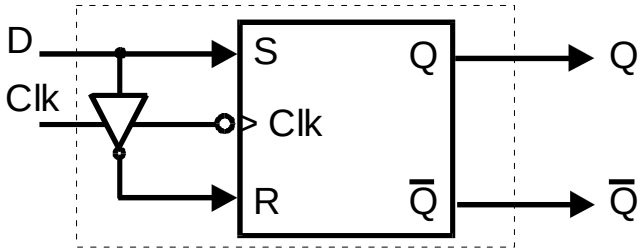
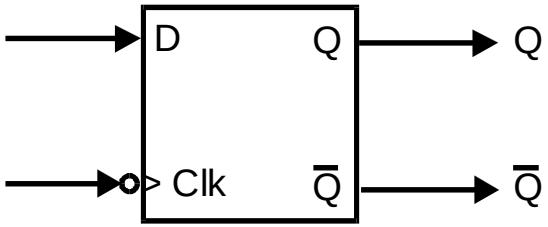


Tabla de excitación		
D^t	Clk	Q^{t+1}
D	↓	D



Registros (I)

Concepto de registro. Circuito capaz de almacenar una cantidad limitada de información binaria durante un determinado tiempo (mientras se mantenga su alimentación).

En el computador: dispositivo activo de memoria, para almacenar una palabra (n bits).

- Está compuesto, normalmente, por un conjunto de biestables.

Tipos básicos:

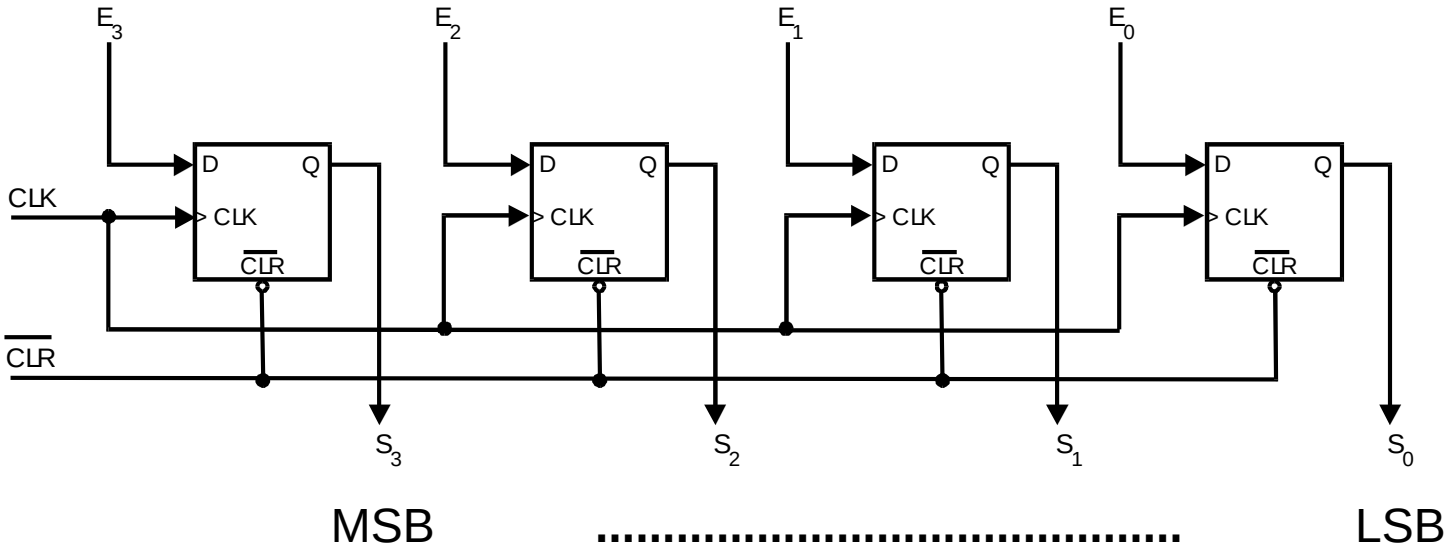
- De almacenamiento
- De desplazamiento
- Contadores

Registros (II)

De almacenamiento. Su única función es actuar de memoria, es decir almacenar información. Las operaciones permitidas son las de lectura y escritura (almacenamiento).

Los registros de almacenamiento pueden ser:

- De tipo latch
- De tipo Master Slave

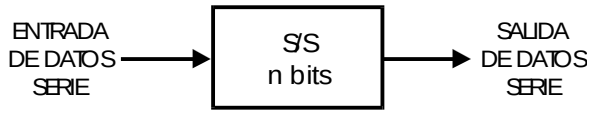


Registros (III)

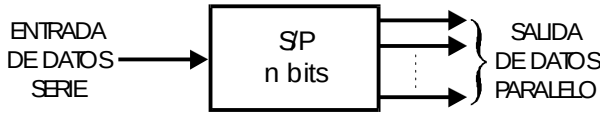
De desplazamiento. Además de almacenar la información, es capaz de desplazarla bit a bit.

Tipos (en función de cómo son las entradas y las salidas):

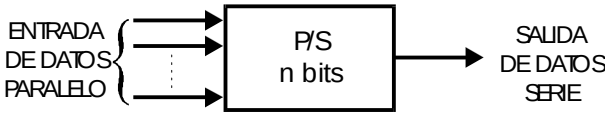
- Entrada Serie - Salida Serie



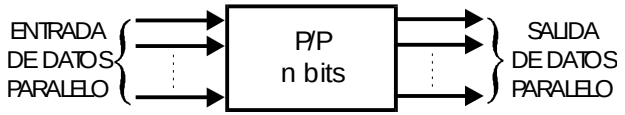
- Entrada Serie - Salida Paralelo



- Entrada Paralelo- Salida Serie



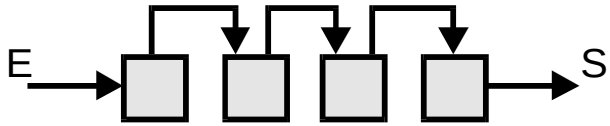
- Entrada Paralelo- Salida Paralelo



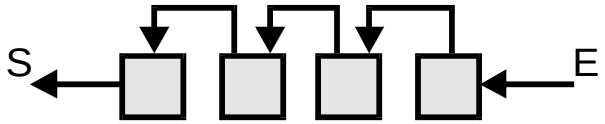
Registros (IV)

Tipos (en función del sentido del desplazamiento):

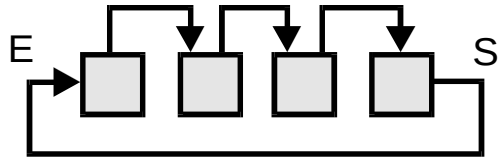
- Abiertos (desplazamientos)
  - Hacia la **derecha**



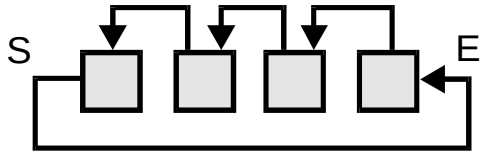
  - Hacia la **izquierda**



- Circulares o en anillo (rotación)
  - Hacia la **derecha**

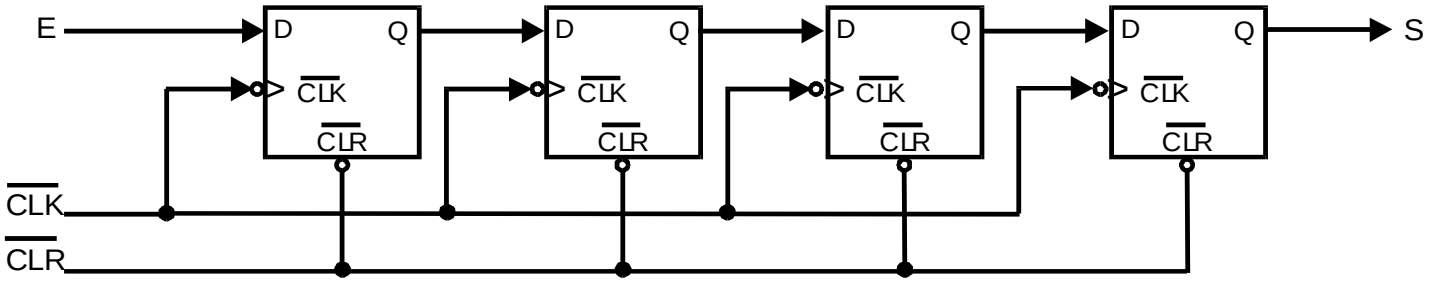


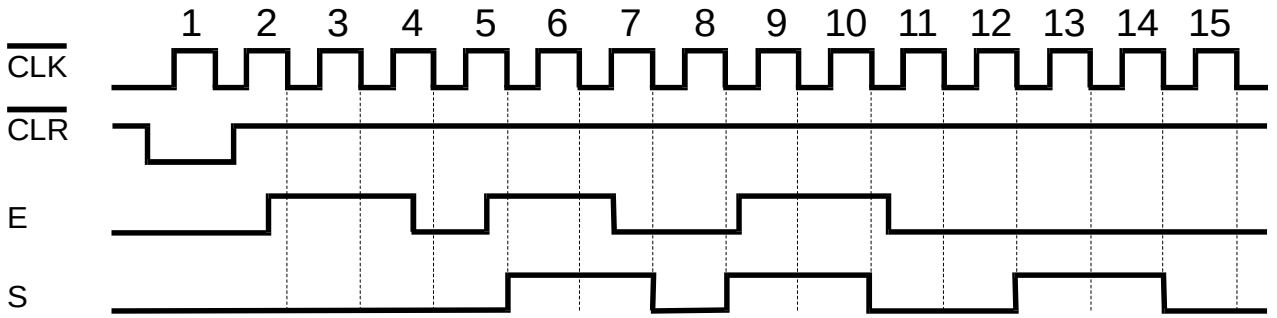
  - Hacia la **izquierda**



Registros (V)

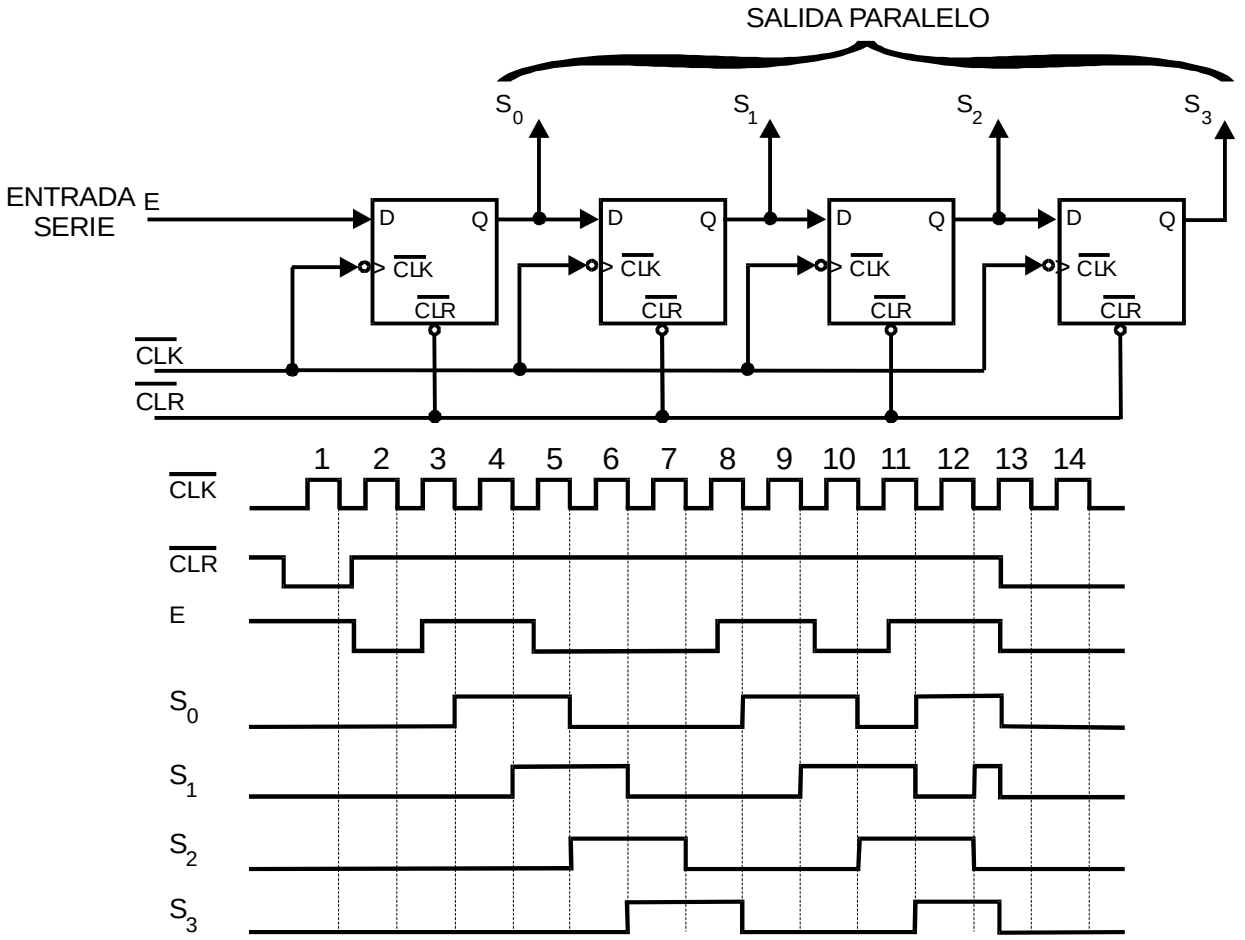
Registro de desplazamiento con Entrada Serie - Salida Serie





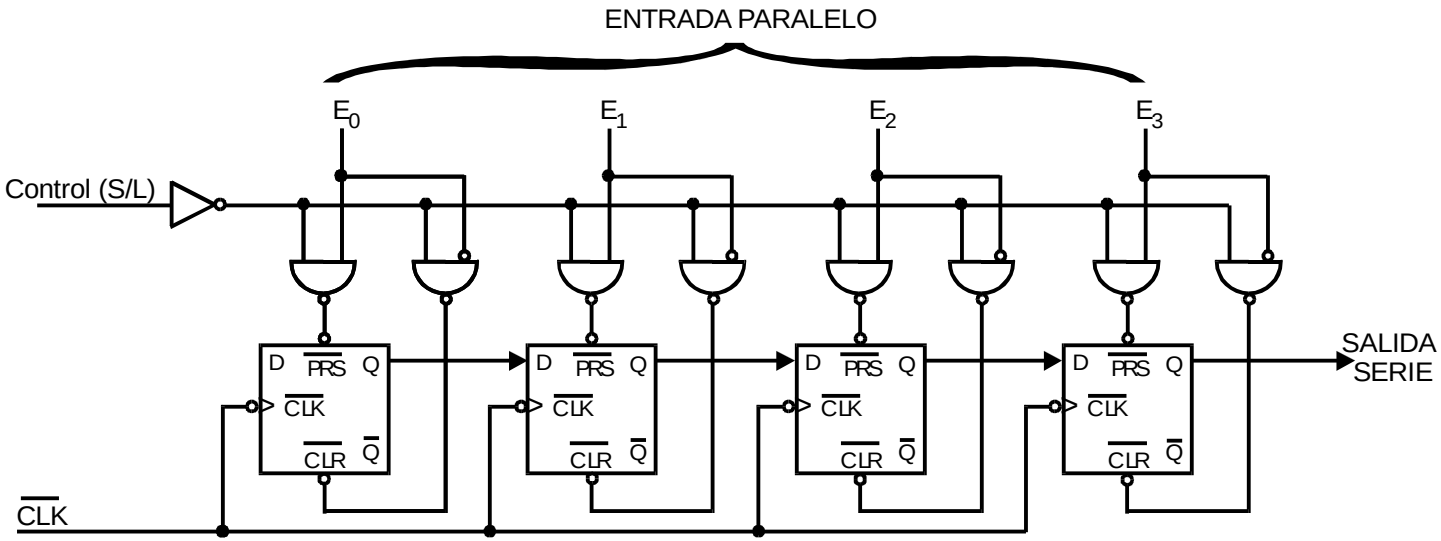
Registros (VI)

Registro de desplazamiento con Entrada Serie - Salida Paralelo



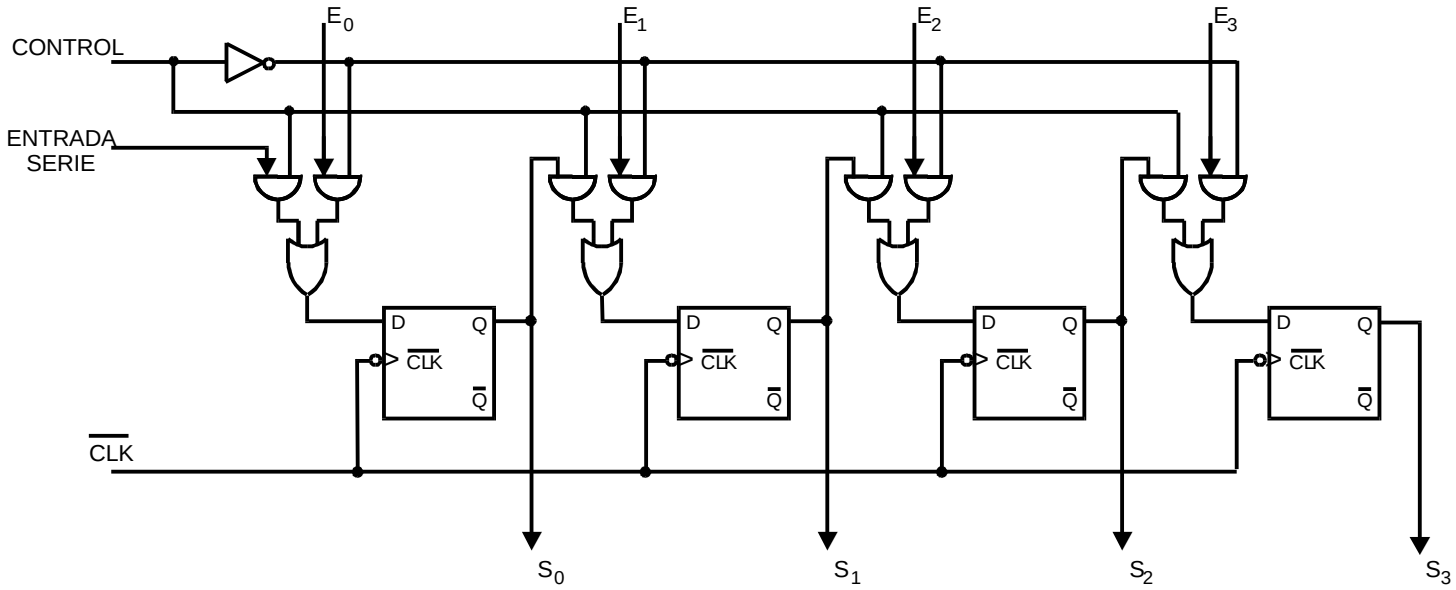
Registros (VII)

Registro de desplazamiento con Entrada Paralelo- Salida Serie



Registros (VIII)

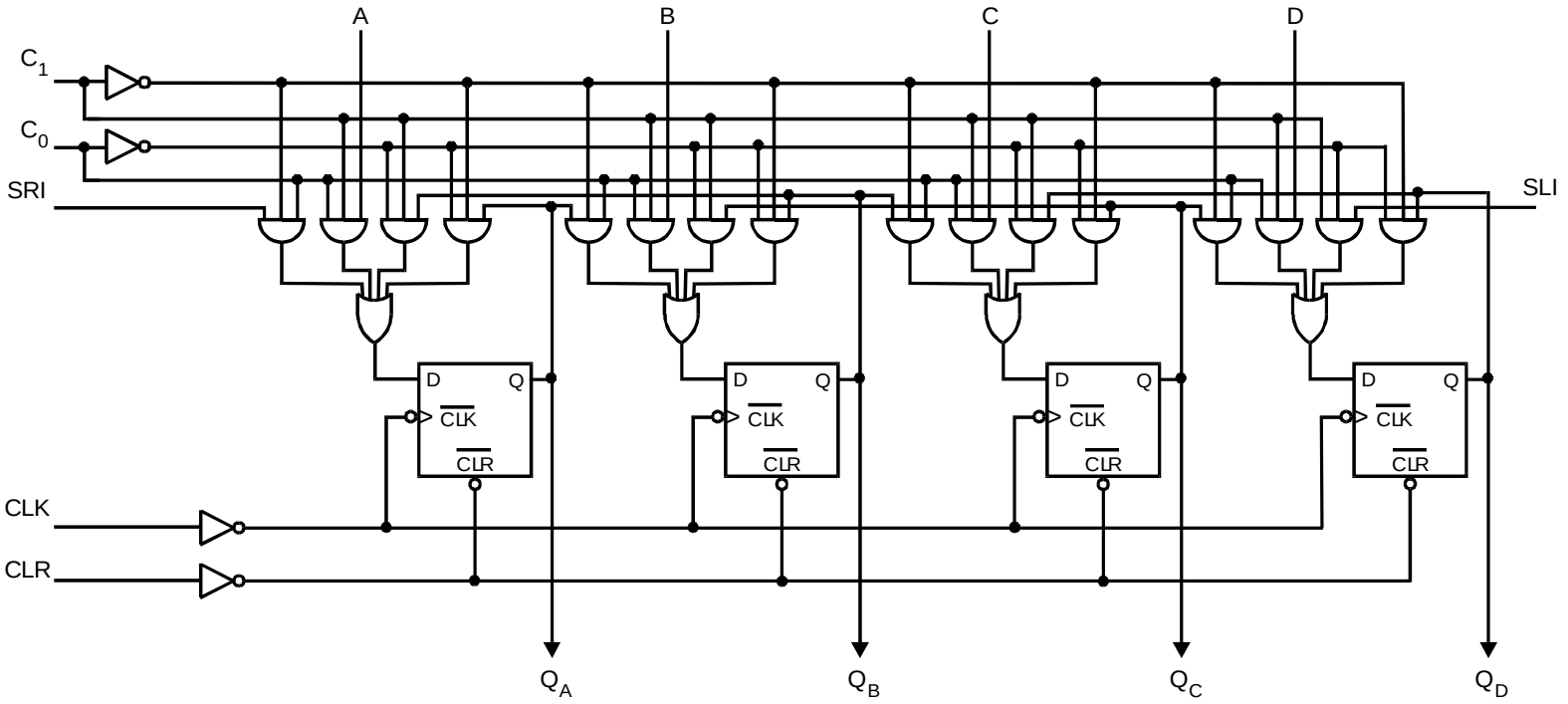
Registro de desplazamiento con Entrada Paralelo- Salida Paralelo



Registros (IX)

Registro de desplazamiento universal

C_1	C_0	Operación
0	0	Mantiene el estado
0	1	Desplazamiento derecha
1	0	Desplazamiento izquierda
1	1	Carga

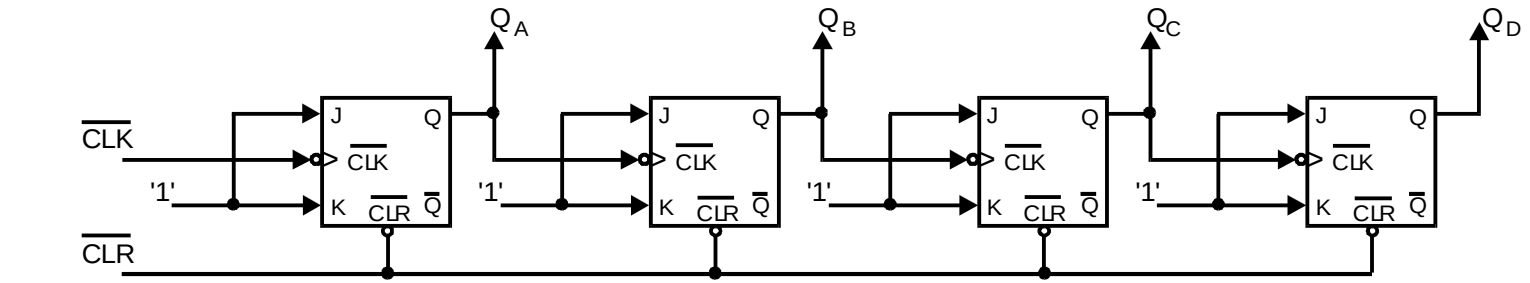


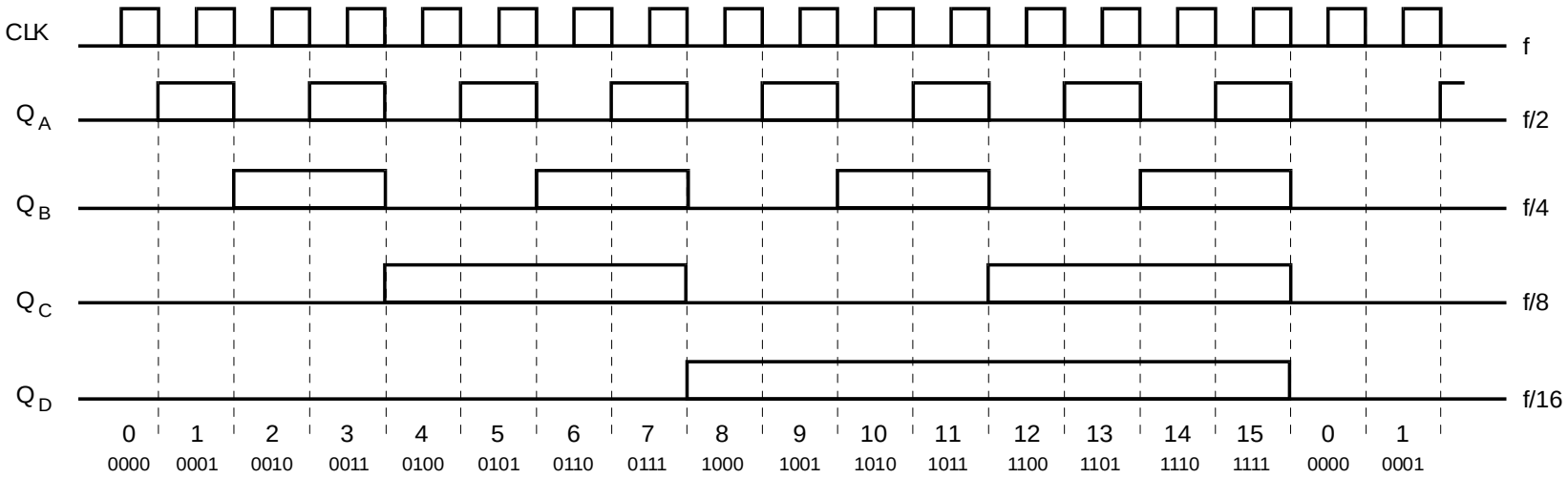
Contadores (I)

- Un contador es un circuito que “cuenta” y “recuerda” el número de impulsos que se le aplican a través de una entrada externa de “reloj”.
- Consta normalmente de una cadena de biestables Master-Slave en cuyas “n” salidas se lee un número binario puro que indica la cuenta realizada hasta el momento.
- Clasificaciones elementales:
  - Forma de activación:
    - Asíncronos y síncronos
  - Forma de contar:
    - Ascendentes y descendentes
  - Tipo de cuenta:
    - Binarios y módulo N (módulo 10)

Contadores (II)

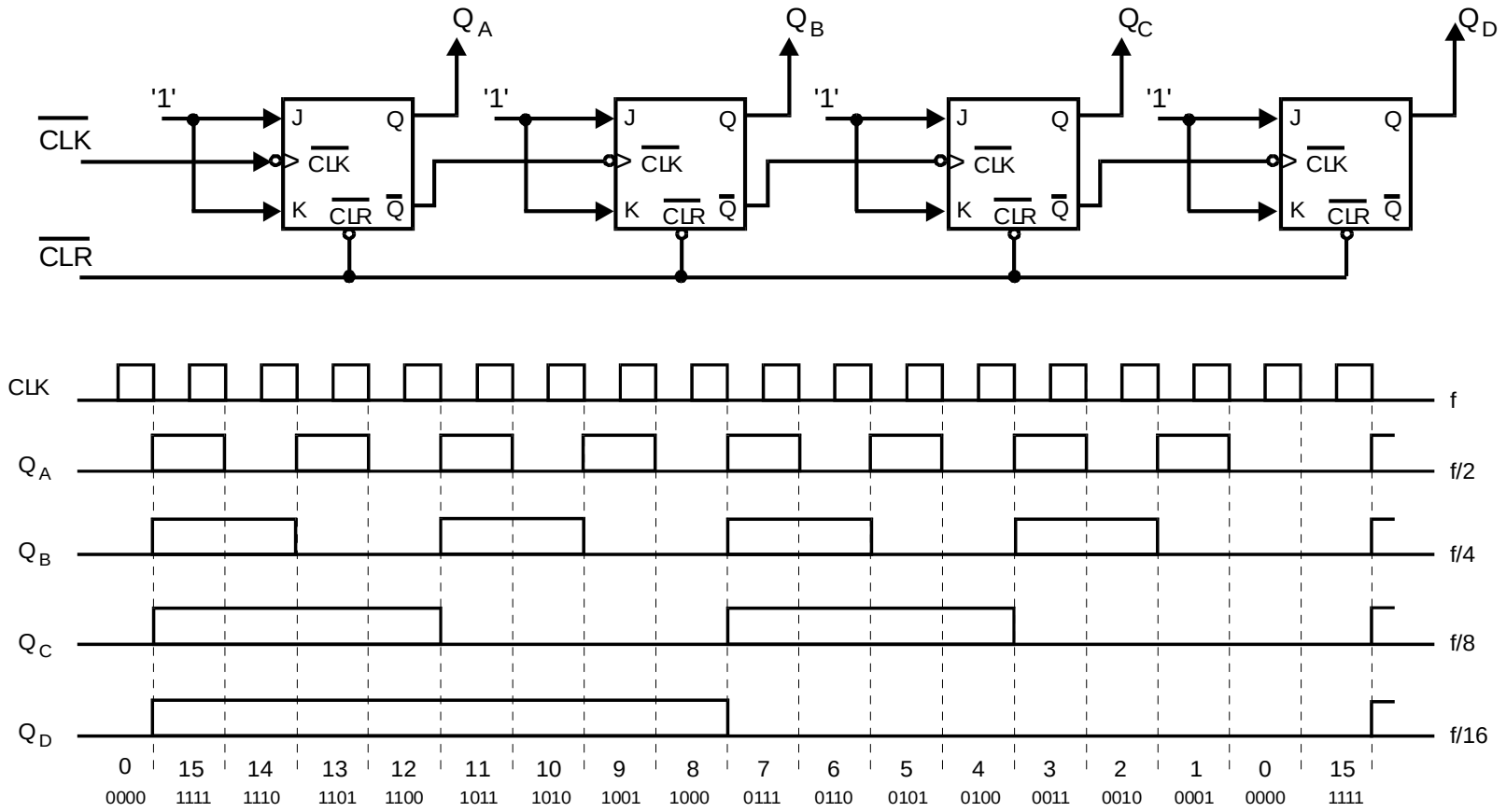
Contador asíncrono (ascendente y binario)





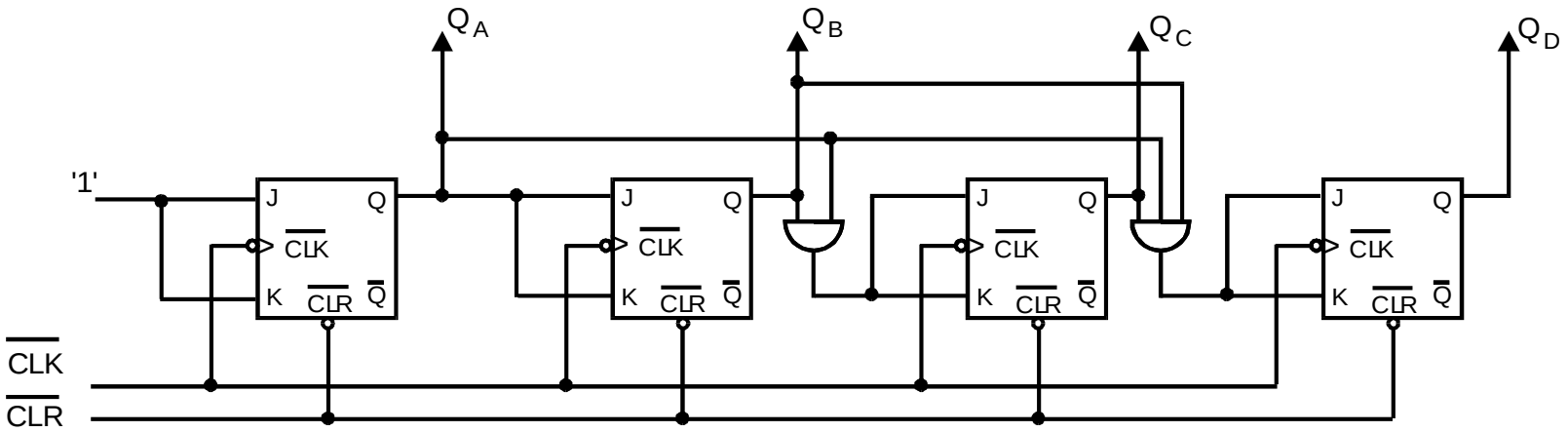
Contadores (III)

Contador asíncrono (descendente y binario)



Contadores (IV)

Contador síncrono (ascendente y binario)



Contadores (V)

Contador módulo N

Módulo: es el número de cuentas diferentes que realiza un contador.

- Para realizar un contador de módulo N se tiene que partir de un contador de n bits, tal que $2^{n-1} < N < 2^n$ y se eliminan las cuentas sobrantes, mediante la lógica combinacional necesaria.

  Ejemplo: contador BCD, de décadas o módulo 10.

  - Para contar los diez dígitos decimales se necesita un contador de 4 bits, ya que $2^{4-1} < 10 < 2^4$
  - La combinación que hay que detectar para eliminar las seis combinaciones binarias que sobran con 4 bits es la **1010.**

    (Como es un contador basta con detectar $Q_D Q_B = 11$)
  - Cuando se detecte 1010 hay que "resetear" el contador y volver a empezar la cuenta.

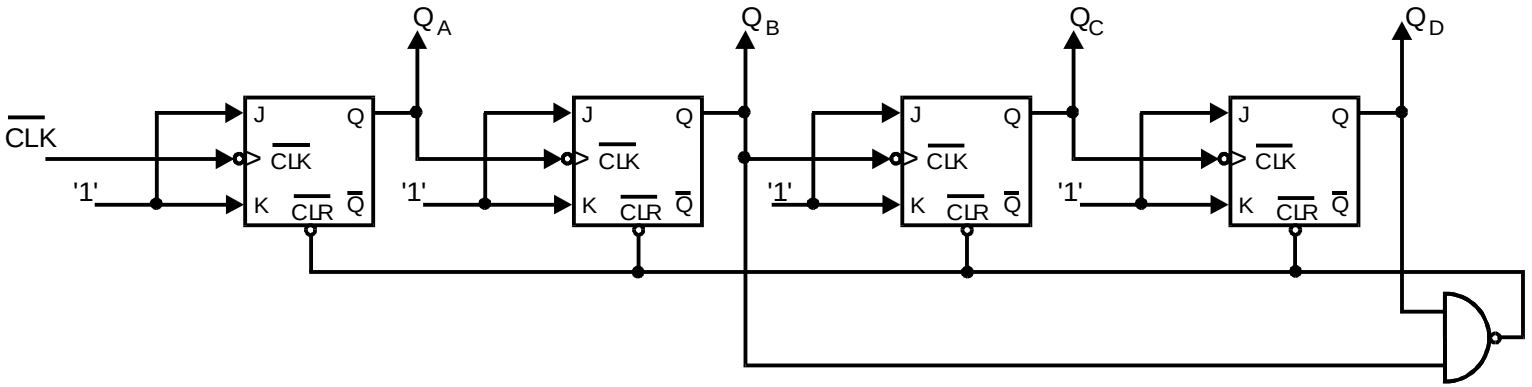
Contadores (VI)

Contador módulo N (de décadas, asíncrono y ascendente)

Q_D	Q_C	Q_B	Q_A
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	**0**	**1**	**0**
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

Puesta a cero del contador

Detección del 10



Tema 5: Memorias

Índice

Conceptos básicos

Parámetros característicos

Jerarquía de memoria

Memoria principal

- Tecnologías
- Estructura
- Mapa de memoria

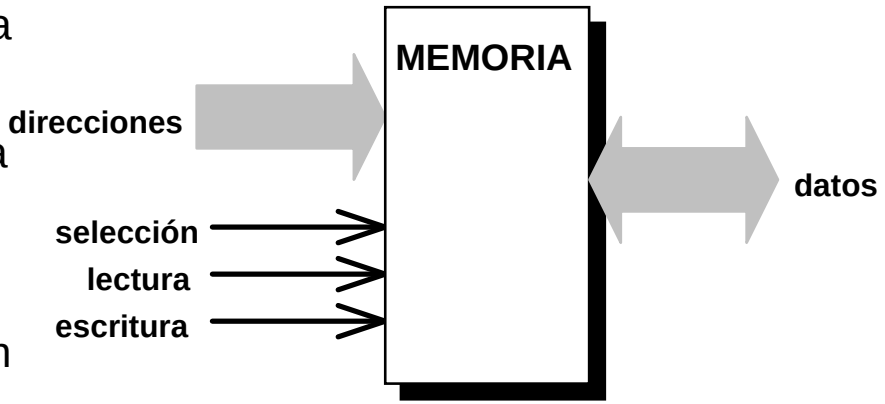
Bibliografía

- Fundamentos de sistemas digitales
  Thomas Floyd
  Prentice-Hall
- Estructura de Computadores
  José M. Angulo
  Ed. Paraninfo
- Fundamentos de los Computadores
  Pedro de Miguel Anasagasti
  Ed. Paraninfo

Conceptos básicos

- **Memoria**: parte del computador que almacena la información: instrucciones y datos. Conjunto de posiciones de memoria con:
  - **Dirección:** identificación de la posición de memoria
  - **Contenido:** información almacenada
- **Celda de memoria:** elemento que almacena un bit
- **Palabra:** número de bits implicados en cada operación con la memoria (8, 16, 32, 64, ... bits).

  Tamaño del bus de datos
- **Operaciones básicas:**
  - Lectura (R)
  - Escritura o almacenamiento (W)



Esquema básico de memoria

Parámetros característicos (I)

- **Capacidad**: cantidad de información que puede almacenar un dispositivo
  - Medidas más usuales:

    Kilobyte (Kb) = 2^{10} bytes

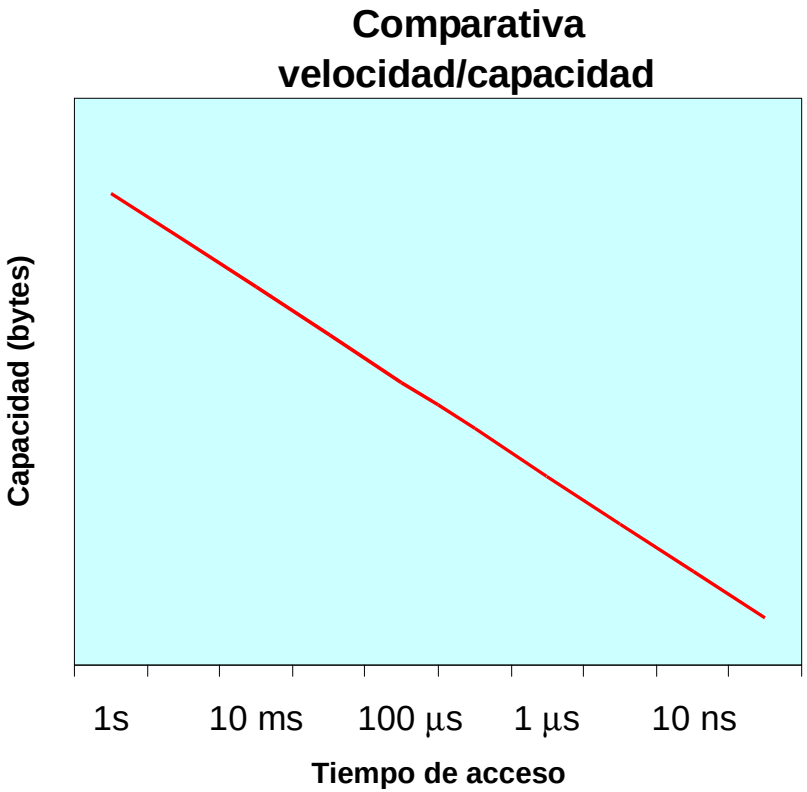
    Megabyte (Mb) = 2^{10} Kb = 2^{20} bytes

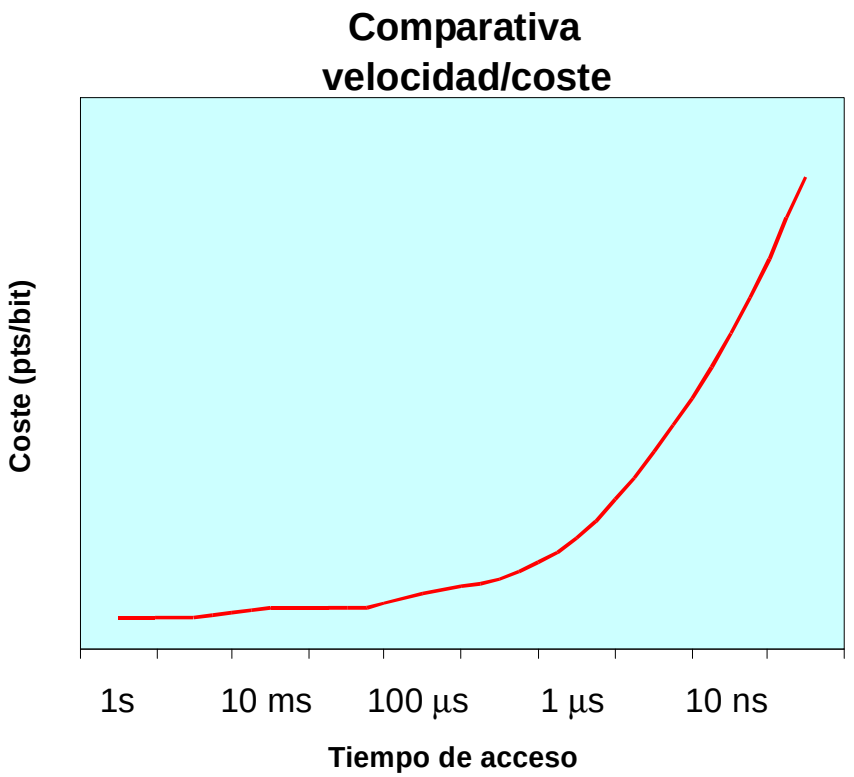
    Gigabyte (Gb) = 2^{10} Mb = 2^{30} bytes

    Terabyte (Tb) = 2^{10} Gb = 2^{40} bytes

- **Velocidad** o **tiempo de acceso:** tiempo que transcurre desde que se proporciona la dirección a la memoria y el dato está disponible
- **Ciclo de memoria:** tiempo que transcurre entre dos accesos consecutivos a memoria. Puede ser superior al tiempo de acceso
- **Coste por bit:** precio por cada bit de información

Parámetros característicos (II)

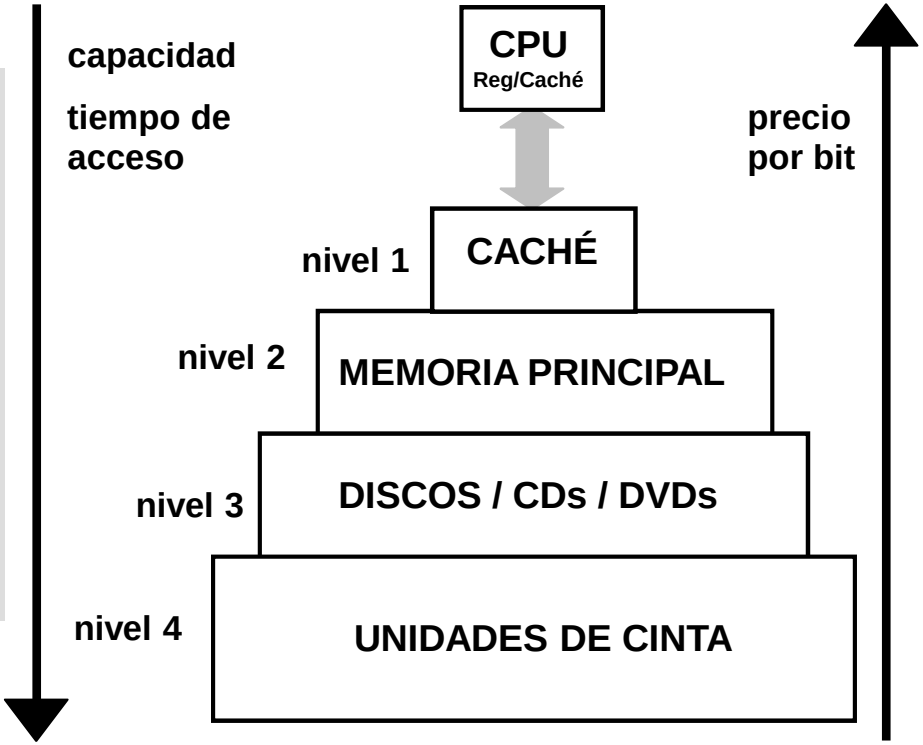




Jerarquía de memoria

Jerarquía:

- Registros CPU
- Caché interna
- Caché externa
- Memoria principal
- Dispositivos de almacenamiento auxiliar/secundario



Memoria principal: Tecnologías

RAM (volátil, lectura/escritura)

Random Access Memory - Memoria de acceso aleatorio

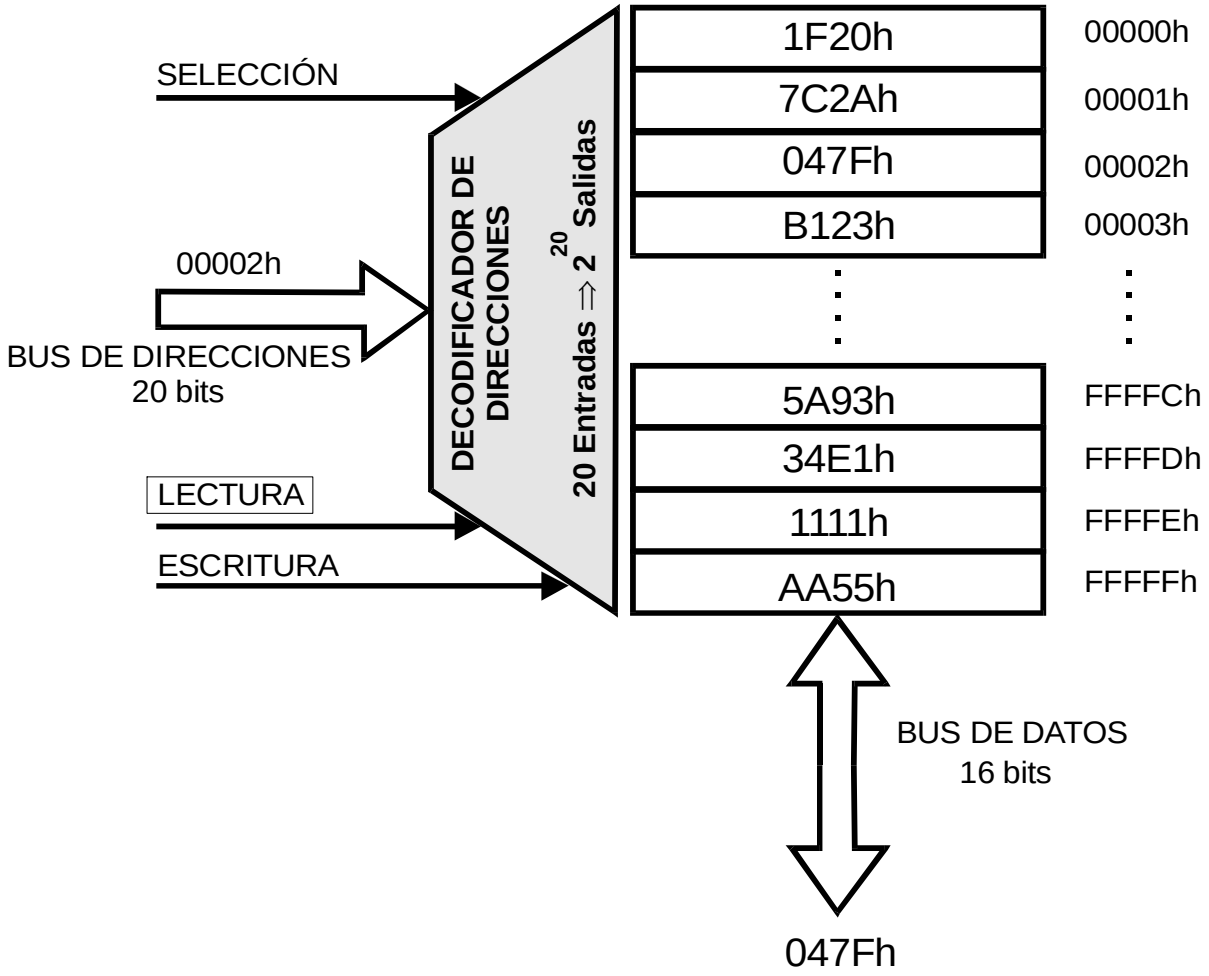
- SRAM - RAM estática
- DRAM - RAM dinámica
  - SDRAM - Synchronous Dynamic RAM - (RAM síncrona y dinámica)

ROM (no volátil, sólo lectura)

Read Only Memory - Memoria de sólo lectura

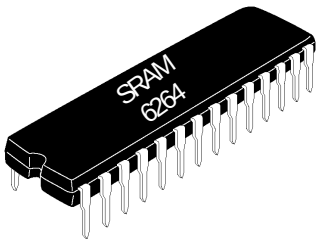
- PROM - *Programmable ROM* - ROM programable
- EPROM - *Erasable PROM* - PROM que se puede borrar (luz ultravioleta)
- EEPROM - *Electrically EPROM* - PROM que se puede borrar (señal eléctrica)
- Flash – Tiempos pequeños de borrado (ms) y escritura

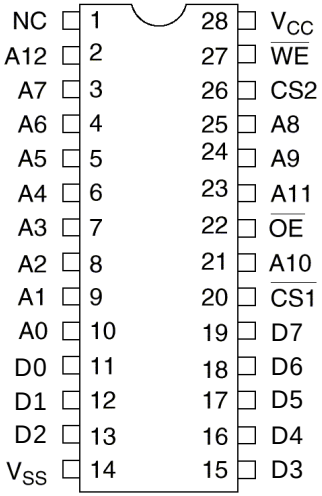
Memoria principal: estructura (I)

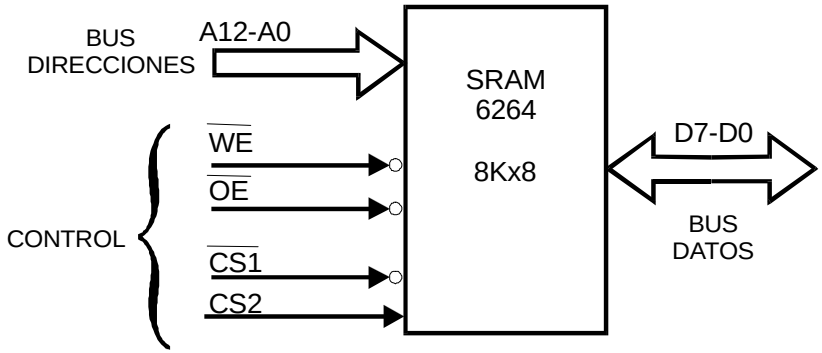


Memoria principal: estructura (II)

Ejemplo práctico: RAM estática de 8kx8



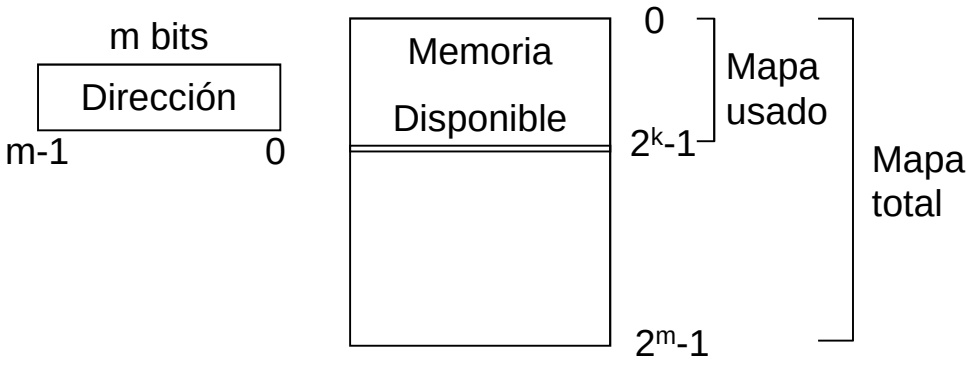




Memoria principal: mapa de memoria (I)

Mapa de memoria

- El mapa de memoria es el espacio direccionable de un computador
- El mapa de memoria viene determinado por el tamaño de las direcciones (y del bus de datos). Así, un tamaño de direcciones de "m" bits permite direccionar 2^m direcciones
- "m" es el ancho del bus de direcciones

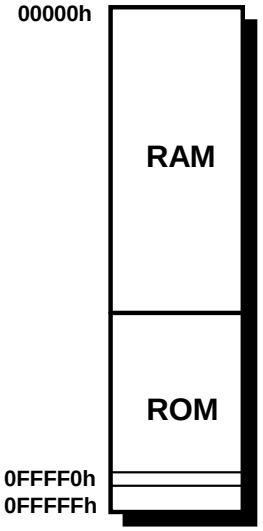


Ampliación del mapa de memoria

- Normalmente no se suele equipar el computador con toda la memoria que es capaz de direccionar

Ubicación de la RAM y la ROM

- Ejemplo práctico: mapa de memoria simplificado del µP 8086

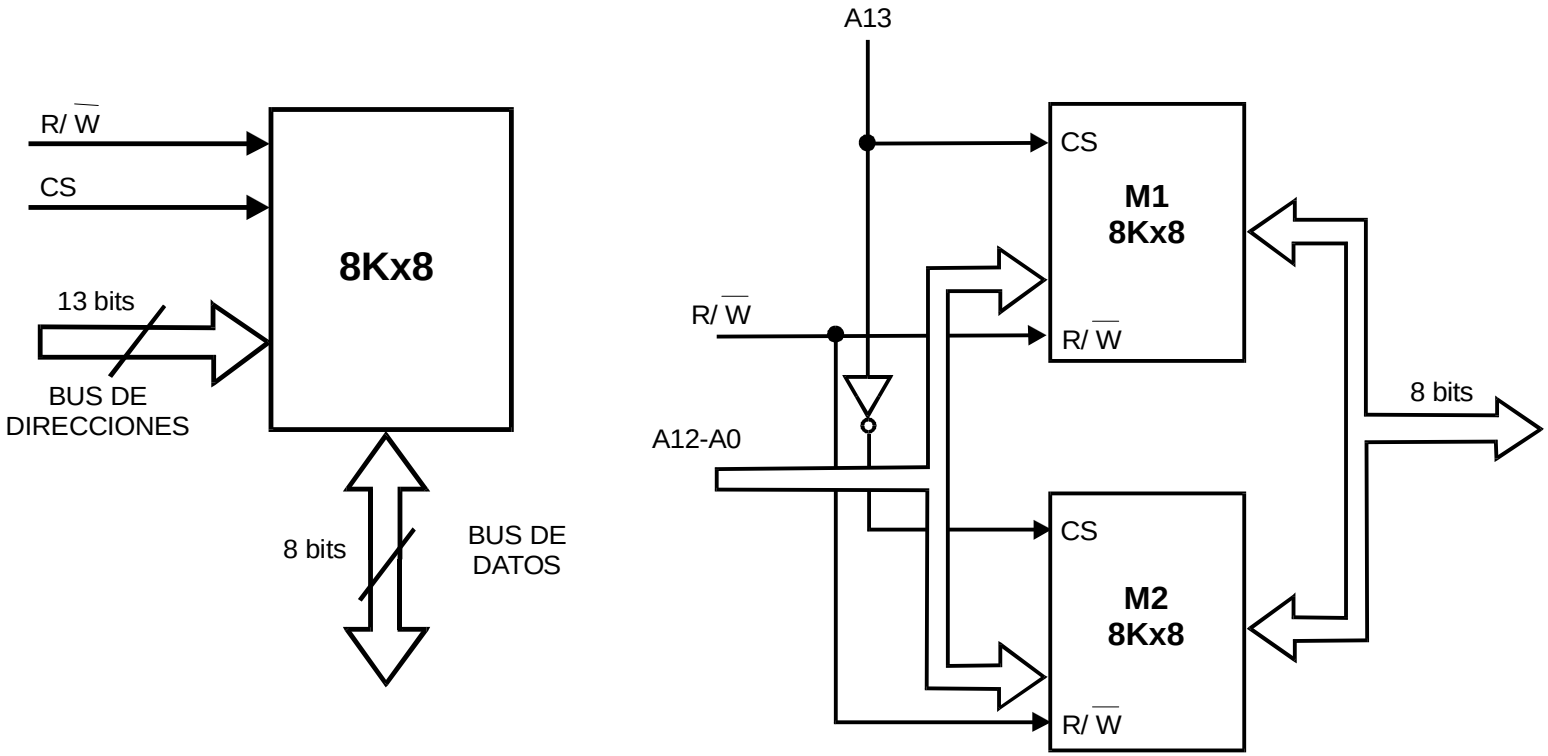


Memoria principal: mapa de memoria (II)

Ejemplo de expansión de la capacidad (nº de palabras):

Uso de más de un chip para incrementar el rango de direcciones

Se desea una memoria de 16 K-bytes con chips de 8Kx8



Memoria principal: mapa de memoria (III)

Ejemplo de expansión del tamaño de palabra:

Uso de más de un chip para incrementar el tamaño de los datos

Se desea una memoria de 8 K-palabras (16 bits) con chips de 8Kx8