Arquitectura de computadores 1

Resumen práctico

Diego González
2007
diesgomo@gmail.com

1. Representaciones de datos

1.1 Sistemas de codificación

- Definición de bit:
  Es un símbolo 0 ó 1
- Definición de byte:
  Es una colección ordenada de 8 bits
- Definición de código binario:
  Es un conjunto ordenado de bits
- Definición de distancia entre códigos binarios:
  La distancia entre el código binario A y el código binario B es el número de bits a cambiar en A para llegar a B
- Definición de distancia d de un sistema de codificación binario:
  d es la menor de las distancias entre sus códigos

1.2 Detección y corrección de errores

- Condiciones de detección y corrección de errores[1]:

$t < d$	Es posible detectar errores
$t < \frac{d}{2}$	Es posible corregir errores

- Método de la paridad:
  Se agregan más bits controladores de la paridad del código. Esto permite detectar errores; para corregir es posible implementar la paridad vertical, donde se controla la paridad por fila y por columna.
- Bits de redundancia:
  Para generar códigos de $d = 3$ para objetos representables en k bits, se necesitan utilizar p bits adicionales tal que $2^p \geq p + k + 1$
- Códigos de Hamming[2]:
  Para un código $a_4 a_3 a_2 a_1$ se agregan los bits $p_3 p_2 p_1$ de redundancia como $a_3 a_2 p_3 a_1 p_2 p_1$ calculando p_i como

  $p_1 = a_4 \oplus a_2 \oplus a_1$

  $p_2 = a_4 \oplus a_3 \oplus a_1$

  $p_3 = a_4 \oplus a_3 \oplus a_2$

  Luego se calcula s_i como

	a_4	a_3	a_2	p_3	a_1	p_2	p_1
s_0	X		X		X		X
s_1	X	X			X	X	
s_2	X	X	X	X			
Posición	7	6	5	4	3	2	1

  Se define y calcula $s = s_2 s_1 s_0$ si $\begin{cases} s = 0 \text{ no hay errores} \\ s = x,\ x \text{ indica la posición del bit errado} \end{cases}$

---

[1] Se asumen las siguientes hipótesis para el cálculo siguiente y para los 3 puntos dados a continuación:

i. La probabilidad de que falle un bit es baja
ii. Las fallas de bits son sucesos independientes entre sí

t es la cantidad máxima de bits admitida como modificados

[2] $\oplus$ es la operación XOR

- Códigos de redundancia cíclica:
  Si m es la cantidad de bits, se define el polinomio $M(x)$ de grado $m-1$ cuyos coeficientes coinciden con los bits del mensaje, y $G(x)$ polinomio de grado r y coeficientes binarios.
  Se calcula $x^r M(x) = Q(x)G(x) + R(x)$ y $T(x) = x^r M(x) - R(x)$ y se envía $T(x)$.
  Si se verifica que $T(x) \mid G(x)$, entonces, el código es correcto, sino, hay errores

1.3 Representación interna de datos

- Tipo carácter:
  Estos se representan asociando un código binario distinto para cada carácter distinto según determinados estándares, los actualmente más usados son el ISO 10646 y el UNICODE.
- Tipo string:
  Se representan como una sucesión de caracteres implementada como:
  i. de largo fijo
  ii. de largo indeterminado indicando el final con un carácter especial
  iii. un registro de dos campos: el primero contiene el largo y el segundo la cadena de caracteres del largo especificado en el primer campo
- Tipo entero sin signo[3]:
  Se representan expresando el número en base 2. La cantidad de números representables son $0 \le N \le 2^n - 1$ y sus características son:
  i. El 0 es representable
  ii. El orden en la representación se conserva
  iii. Las operaciones de las representaciones coinciden con lo que se representa
- Tipo entero con signo:

Nombre de la representación	Descripción	Cantidad de números representables
Valor absoluto y signo	Se utilizan todos los bits disponibles – 1 para almacenar el número y el bit restante para almacenar el signo	$-\left(2^{n-1}-1\right) \le N \le 2^{n-1}-1$
Desplazamiento	Aplica un desplazamiento d (típicamente $d = 2^{n-1}$ ó $d = 2^{n-1}-1$) al número en binario: $N \to (N+d)_{(2)}$	$-d \le N \le 2^n - 1 - d$
Complemento a uno	Los números positivos se representan en binario, y los negativos en el complemento de cada bit del positivo correspondiente	$-\left(2^{n-1}-1\right) \le N \le 2^{n-1}-1$
Complemento a dos	Representa los números por complemento a uno y luego le suma 1	$-2^{n-1} \le N \le 2^{n-1}-1$

[3] En esta sección se considerará n la cantidad de bits disponibles para la codificación y N el número a representar.
Esta representación tiene el problema de overflow dado por el bit de acarreo (carry bit), que es el bit más significativo "sobrante" de la operación.

Nombre de la representación	¿El 0 tiene una única representación?	¿El orden de la representación se conserva?	¿Las operaciones de las representaciones coinciden con lo que representa?
Valor absoluto y signo	No, tiene 2	No	No
Desplazamiento	Sí	Sí	No
Complemento a uno	No, tiene 2	No	No
Complemento a dos	Sí	No	Sí

- Representación decimal:

Nombre de la representación	Descripción
Decimal empaquetado	Se representa el número como un string de caracteres: un número es representado como una sucesión de caracteres que son símbolos numéricos. El final del carácter implementado se indica con un caracteres especial
Punto flotante	N se representa como $N=(-1)^s b^e M$, donde s es el signo, e el exponente, M la mantisa, y b la base de la representación. Para representarlos se convenciona la mantisa como máxima (se toma su bit más significativo distinto de 0). Generalmente se expresa en base 2 como: $N=(-1)^S 1.F\times 2^E$ y se almacena la terna (S,F,E). Las especificaciones de cómo hacerlo están dadas en el estándar IEEE 754. Se considera normalizado si es $1.F$ y denormalizado si es $0.F$

2. Álgebra de Boole

2.1 Axiomas y propiedades

- Axiomas[4]:
  1. Existe un conjunto G de objetos, sujetos a una relación de equivalencia, denotada por "=" que satisface el principio de sustitución.
  2. Se define una regla de combinación $\begin{cases}"+"\\"\cdot"\end{cases}$ tal que $\begin{cases}a+b\\a\cdot b\end{cases}\in G$ siempre que $\begin{cases}a \text{ o } b\\a \text{ y } b\end{cases}\in G$
  3. Neutros:
     Existe un elemento $\begin{cases}0\\1\end{cases}\in G$ tal que para cada $a\in G$, $\begin{cases}a+0=a\\a\cdot 1=a\end{cases}$
  4. Conmutativos:
     Para todo $a,b\in G$ ocurre que $a+b=b+a$ y $a\cdot b=b\cdot a$
  5. Distributivos:
     Para todo $a,b,c\in G$ ocurre que $a+(b\cdot c)=(a+b)\cdot(a+c)$ y $a\cdot(b+c)=a\cdot b+a\cdot c$
  6. Complemento:
     Para todo $a\in G$, existe $\bar{a}$ tal que $\bar{a}\cdot a=0$ y $a+\bar{a}=1$
  7. Existen por lo menos dos elementos $a,b\in G$ tales que estos sean distintos
- Propiedades[5]:
  - Dualidad:
    Todo enunciado válido posee un dual también válido intercambiando "0" por "1" y "+" por "·" simultáneamente.
  - Asociatividad:
    $a+(b+c)=(a+b)+c$
  - Idemptoencia:
    $a + a = a$
  - Neutros cruzados:
    $a + 1 = 1$
  - Complemento de complemento:
    $a=\bar{\bar{a}}$
    $a+ab=a$
    $a+\bar{a}b=a+b$
  - Ley de De Morgan:
    $\overline{(a+b)}=\bar{a}\bar{b}$

2.2 Modelos

- Modelo lógico:
  $G=(\{V,F\},\vee,\wedge)$. Este es consecuencia del siguiente isomorfismo:

V	**1**
F	0
$\wedge$	·
$\vee$	+

[4] A + se le dice suma, y a ·, producto.

[5] Por comodidad $a\cdot b$ se notará ab. Las propiedades se enuncian para todo $a,b,c\in G$

- Modelo circuital[6]:
  $G=(\{A,C\},\ circuito\ en\ paralelo,\ circuito\ en\ serie)$.
  Este es consecuencia del isomorfismo:

A	**0**
C	1
circuito en paralelo	+
circuito en serie	·

- Modelo binario[7]:
  $G=(\{0,1\},\text{AND},\text{OR})$. Este es consecuencia del isomorfismo:

0	**0**
1	1
OR	+
AND	·

- Conectivas binarias[8]:

a	*b*	*a* OR *b*	*a* AND *b*	*a* XOR *b*	*a* NOR *b*	*a* NAND *b*	*a* IGUAL *b*	NOT *a*
0	0	0	0	0	1	1	1	1
0	1	1	0	1	0	1	0	1
1	0	1	0	1	0	1	0	0
1	1	1	1	0	0	0	1	0

2.3 Funciones booleanas

- Definición por extensión:
  $F:G^n\to G$ se define por extensión mediante una tabla llamada "tabla de verdad"
- Definición por comprensión:
  $F:G^n\to G$ se define por comprensión indicando los puntos donde se anula
  $(F=\sum(a_1,...,a_n))$ o los puntos donde no se anula $(F=\prod(b_1,...,b_n))$
- Teorema de la expresión algebraica:
  Toda función booleana *f* de *n* variables puede expresarse como:
  $f(x_1,...x_n)=x_1x_2x_3...x_nf(1,1,1,...,1)+\overline{x_1}x_2x_3...x_nf(0,1,1,...,1)+$
  $x_1\overline{x_2}x_3...x_nf(1,0,1,...,1)+...+\overline{x_1}\,\overline{x_2}\,\overline{x_3}...\overline{x_n}f(0,0,0,...,0)$
- Método de expresión algebraica:
  Dada la tabla de verdad de una función, su expresión algebraica vendrá dada por la suma de los productos de las variables (conjugando para que con estas entradas el producto no se anule) de la suma de todas aquellas filas donde la función no se anule.
- Simplificación de funciones: Método algebraico:
  Este consiste en combinar las siguientes propiedades en pos de simplificar la función:
  1. $f\overline{f}=0$
  2. $f+\overline{f}=1$
  3. $gf+\overline{g}f=f$
  4. $gf+f=f$
  5. $f+\overline{f}g=f+g$

---

[6] *A* hace referencia a una llave de un circuito abierta, y *C* a la llave de un circuito cerrada.
[7] Este será el que se utilizará el resto del curso.
[8] NAND es la negación del AND
NOR es la negación del OR
XOR es equivalente a "distinto" ("no igual")

- Simplificación de funciones: Mapa de Karnaugh:
    1. Dibujar una cuadrícula con columnas encabezadas por 00, 01, 11 y 10 (correspondiente a las variables a y b de la función respectivamente) y las filas por c o d (si la función es de 3 o 4 variables respectivamente). Si fuera de 5 variables, basta con hacer dos mapas de Karnaugh superpuestos. En uno se supone la quinta variable de valor 0 y en la otra 1.
    2. Se marca con 1 los lugares para los cuales la combinación de valores de las variables hace que la función valga 1.
    3. Se agrupan (considerando que la cuadrícula no tiene borde) la mayor cantidad posible de unos cuya cantidad encerrada sea potencia de dos.
    4. La mínima expresión de la función se obtiene sumando los términos asociados a cada rectángulo, los cuales son el producto de las variables (simples o complementadas, dependiendo si su valor es 1 o 0 respectivamente) cuyo valor no cambia en él.

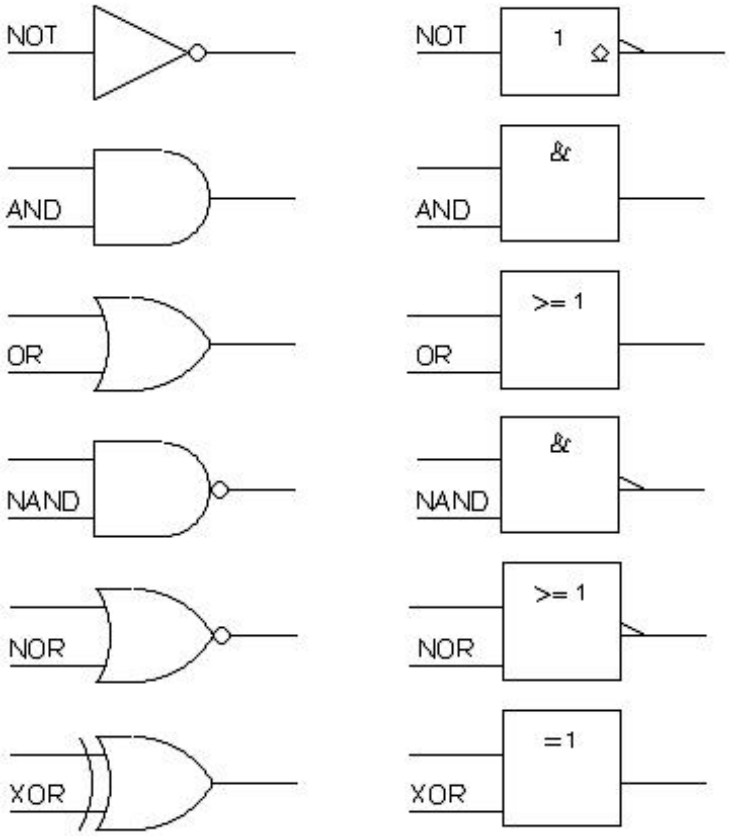
  Ejemplo: La imagen es un Mapa de Karnaugh que representa la función
  $f = a\bar{d} + \bar{a}\bar{b}\bar{c} + \bar{a}\bar{c}d$

cd\ab	00	01	11	10
00	1		1	1
01	1	1		
11				
10			1	1

3. Circuitos combinatorios[9]

3.1 Compuertas lógicas

- Simbología de los conectivos (compuertas lógicas):



- Compactación de circuitos lógicos (bloques lógicos):
  Los bloques lógicos, simbolizados como se muestra compactan circuitos lógicos.
  Cuando se dan estos bloques lógicos debe especificarse cada $s_i = f_i(a_1, ..., a_n)$

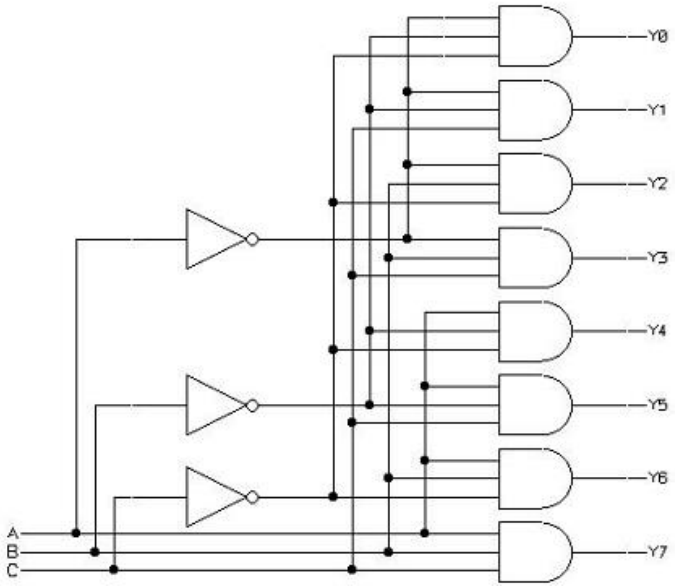
[9] Un circuito combinatorio se define como un circuito cuya salida está determinada en todo instante por la entrada.
Todas las tecnologías se basan en la interpretación circuital.

3.2 Bloques constructivos

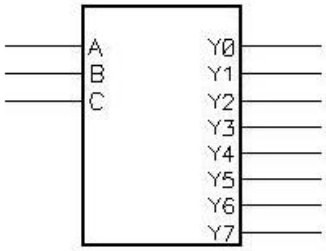
- Circuito decodificador:
  - Descripción:
    Dada una combinación de unos y ceros a la entrada, todas las salidas estarán en 0 salvo la que corresponda al número binario coincidente con la combinación de todas.
    Éste tiene N entradas y 2^N salidas.
  - Tabla de verdad (para $N = 3$):

A	B	C	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

  - Circuito lógico (para $N = 3$):



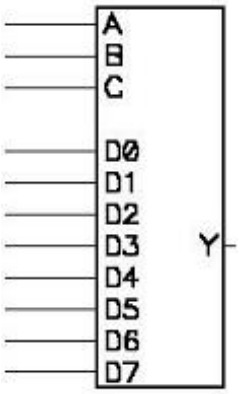
  - Bloque constructivo (para $N = 3$):



- Circuito multiplexor:
  - Descripción:
    La salida *Y* toma el valor de la entrada *D* cuyo índice coincida con el número binario representado por las entradas *A, B* y *C*.
    A, B y *C* se llaman entradas de control, y *D* entrada de datos.
    Éste tiene $N + 2^N$ entradas y una salida.
  - Tabla de verdad (para $N = 3$):

A	B	C	Y
0	0	0	D0
0	0	1	D1
0	1	0	D2
0	1	1	D3
1	0	0	D4
1	0	1	D5
1	1	0	D6
1	1	1	D7

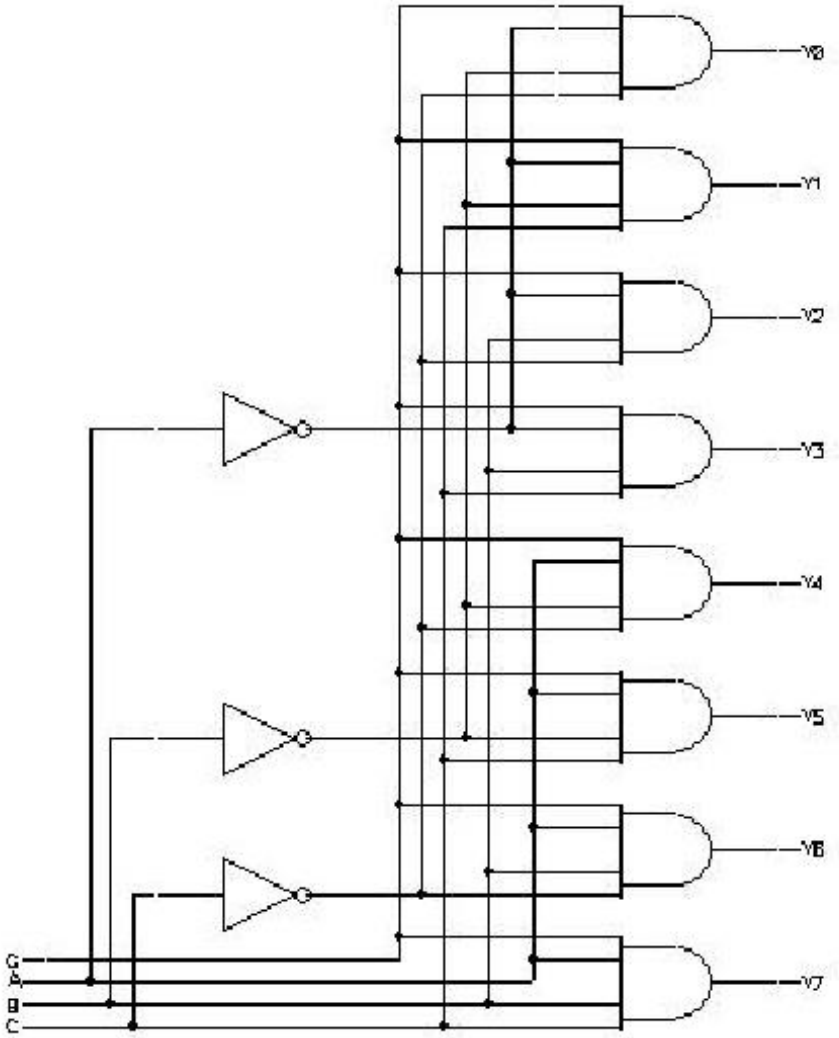
  - Bloque constructivo (para $N = 3$):



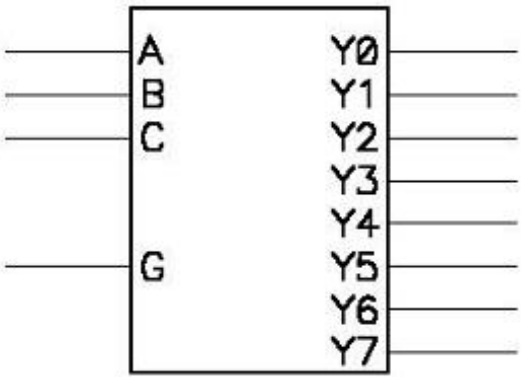
- Circuito demultiplexor:
  - Descripción:
    Dada la compuerta *G* (Gate), se devuelve *G* por la salida *Y* cuyo índice coincide con el número binario correspondiente al dado por las entradas de control.
  - Tabla de verdad (para $N = 3$):

A	B	C	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
0	0	0	0	0	0	0	0	0	0	G
0	0	1	0	0	0	0	0	0	G	0
0	1	0	0	0	0	0	0	G	0	0
0	1	1	0	0	0	0	G	0	0	0
1	0	0	0	0	0	G	0	0	0	0
1	0	1	0	0	G	0	0	0	0	0
1	1	0	0	G	0	0	0	0	0	0
1	1	1	G	0	0	0	0	0	0	0

- Circuito lógico (para $N = 3$):



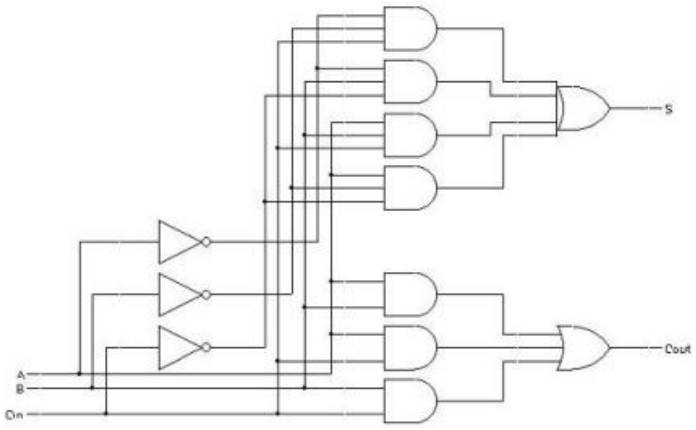
- Bloque constructivo (para $N = 3$):



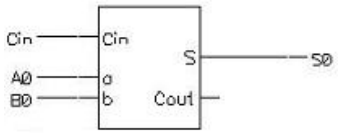
- Circuito sumador completo de 1 bit:
  - Descripción:
    Suma dos números de un bit cada uno con un acarreo de entrada (c_{in}) y devuelve la suma y un acarreo de salida (c_{out})
  - Tabla de verdad:

a	b	c_{in}	S	c_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

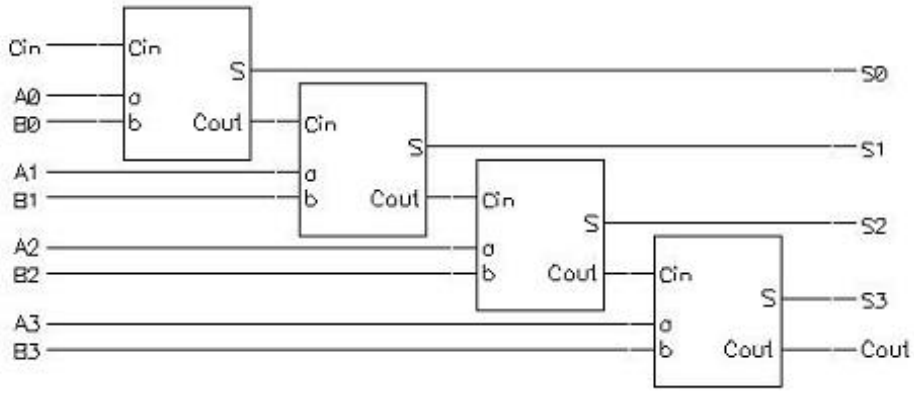
  - Circuito lógico:



  - Bloque constructivo:



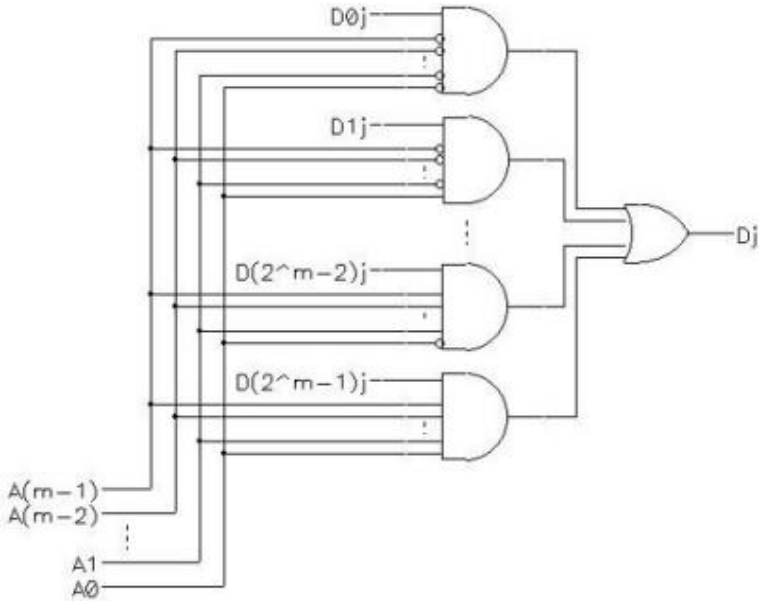
- Circuito sumador completo de *N* bits:
  - Descripción:
    Igual al anterior pero con *N* bits.
  - Bloque constructivo:
    Se conexiona en cascada los sumadores completos de 1 bit.
    Para *N* = 4 bits:



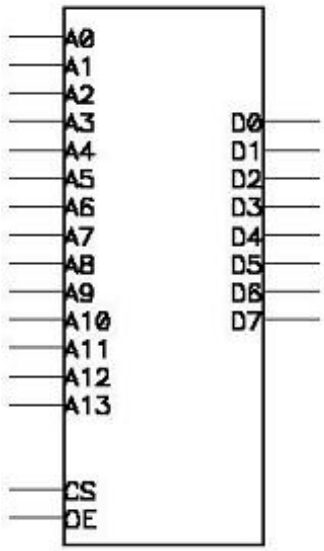
- ROMs:
  - Descripción:
    Este tiene *m* + 2 entradas y *n* salidas.
    Dada una entrada devuelve *n* bits (uno en cada salida) "almacenados" en esa entrada. *CS* selecciona o deselecciona la ROM (indica si sus salidas son 0 o son las que deberían ser)
  - Tabla de verdad:

Entradas								Salidas				
Am-1	Am-2	Am-3	....	....	A_2	A_1	A_0	Dn_{-1}	Dn_{-2}	...	D_1	D_0
0	0	0	....	....	0	0	0	0	0	....	0	0
0	0	0	....	....	0	0	1	1	0	....	1	0
0	0	0	....	....	0	1	0	0	0	....	0	0
....	....	....	....	....	....	....	....	....	....	....	....	....
....	....	....	....	....	....	....	....	....	....	....	....	....
1	1	1	....	....	1	0	1	1	1	....	1	1
1	1	1	....	....	1	1	0	0	1	....	0	1
1	1	1	....	....	1	1	1	1	1	....	1	1

  - Circuito lógico:

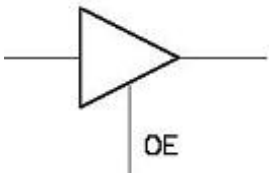


- Bloque constructivo (para $m = 14$ y $n = 8$):



3.3 Retardo de conmutación[10]

- Problema:
  Dados los tiempos de reacción por las corrientes transitorias es que existen resultados reales incorrectos.
- Solución:
  Se incorporan relojes tras los cambios que estos producen las salidas pueden tomarse como válidas. Estos dan tiempo a que las corrientes transitorias se disipen.
- Lógica del tercer estado[11]:
  - Descripción:
    Este agrega a los estados 0 y 1 el estado *Z*.
    Estas indican con $Z = 1$ que se habilite la salida, y con 0, que se inhabilite la salida. Estas se implementan con una entrada identidad llamada output enabled (*OE*).
  - Conectivo:



[10] Tiempo de propagación, retardo de propagación o tiempo de setup
[11] Estado de alta impedancia, estado diferente o tri-state.

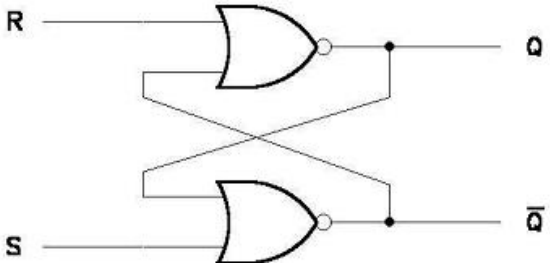
4. Circuitos secuenciales[12]

4.1 Flip-flops

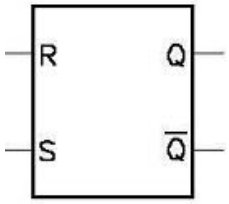
- Descripción:
  Este es un circuito capaz de "recordar" un valor de la entrada previa y por ello puede considerarse un "elemento de memoria"
- Flip-flops R-S asincrónico (Latch):
  - Tabla de verdad:

R	S	Q_{n+1}
0	0	Q_n
0	1	1
1	0	0
1	1	-

  - Circuito lógico:

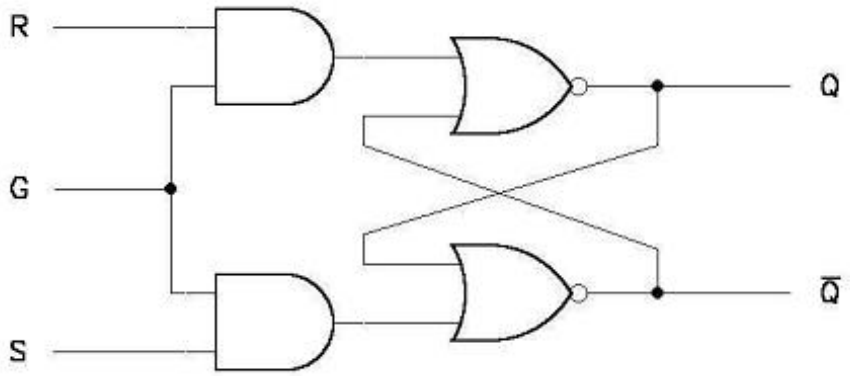


  - Bloque constructivo:

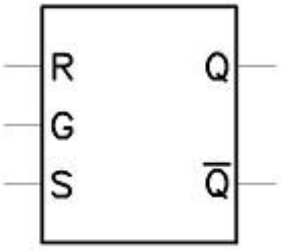


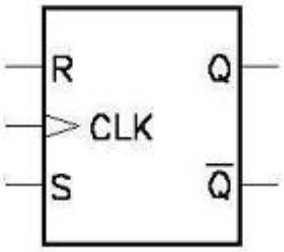
[12] Son secuenciales porque las salidas no son solo en función de las entradas, sino también en función de los valores anteriores.

- Flip-flop R-S sincrónico:
  - Descripción:
    Estos flip-flops habilitan el acceso al contenido mediante una señal *G* de control secuencialmente proveniente de un reloj (*CLK*). Hay dos estados:
    1. Si *G* vale 1, se habilita el acceso al contenido (control por reloj).
    2. Se habilita el acceso al contenido en cualquier cambio del valor en *G* (control por flanco de reloj)
  - Circuito lógico:



  - Bloque constructivo:
    El primero es con un control por reloj, el segundo por un flanco de reloj:





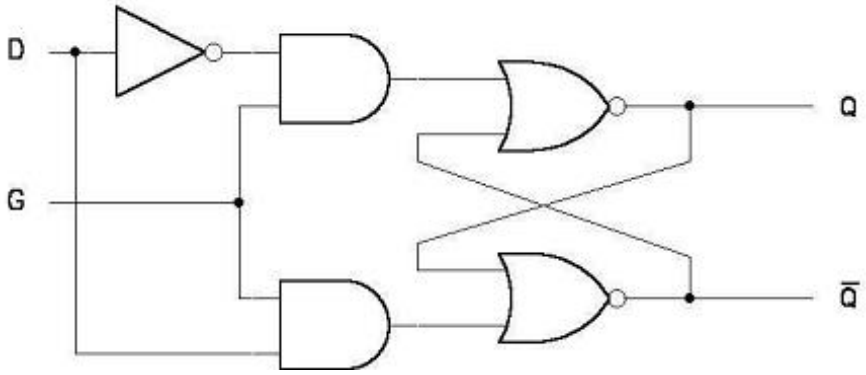
- Flip-flop D:
  - Descripción:
    Estos funcionan igual al flip-flop R-S sincrónico, con la diferencia de que el nuevo valor de la salida corresponde al valor de la entrada *D* mientras *G* está en 1. Mientras *G* está en 0 la salida mantiene el valor de la entrada *D* inmediatamente antes de la transición de 1 a 0 en la entrada *G*.
    Cuando se trabaja con flanco de reloj se hace ascendentemente (cuando el reloj va de 0 a 1)
  - Tabla de verdad:

D	Q_{n+1}
0	0
1	1

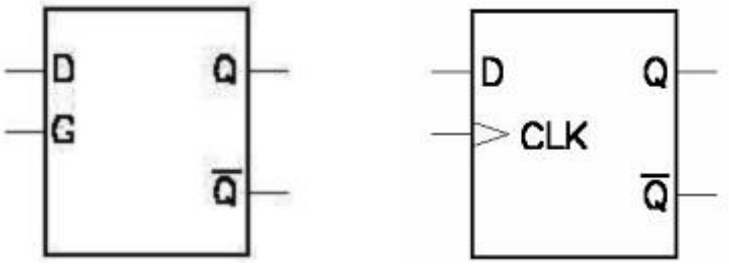
  - Ecuación característica:

$$Q_{n+1} = D$$

  - Circuito lógico:



  - Bloque constructivo:



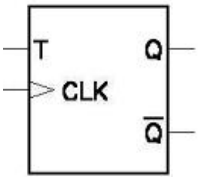
- Flip-flop T:
  - Descripción:
    O bien mantiene su salida incambiada ($T = 0$) o bien la invierte ($T = 1$)
  - Tabla de verdad:

T	Q_{n+1}
0	Qn
1	Q'n

  - Ecuación característica:

$$Q_{n+1} = Q_n \overline{T} + \overline{Q_n} T$$

  - Bloque constructivo:



- Flip-flop J-K:
  - Descripción:
    Estos funcionan bajo los flancos descendentes de reloj y permite emular otros flip-flops según:

Flip-flop	J	K
R-S	S	R
D	D	$\overline{D}$
T	T	T

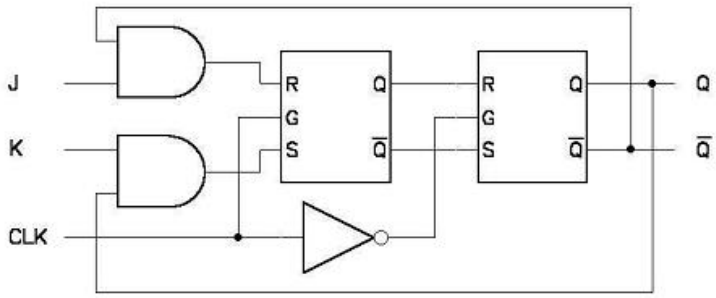
  - Tabla de verdad:

J	K	Q_{n+1}
0	0	Qn
0	1	0
1	0	1
1	1	Q'n

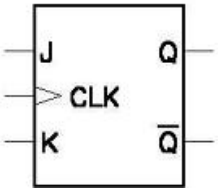
  - Ecuación característica:

$$Q_{n+1} = J\overline{Q_n} + \overline{K}Q_n$$

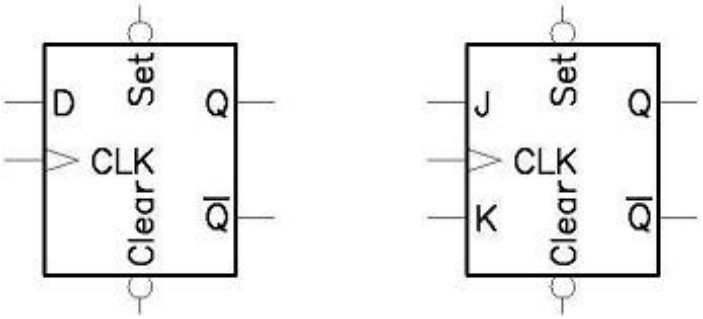
  - Circuito lógico:



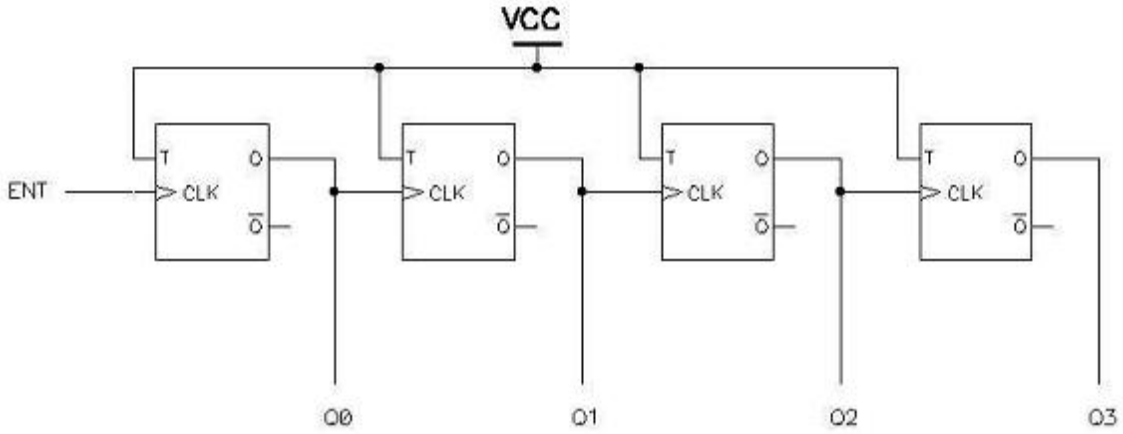
  - Bloque constructivo:



- Entradas *set* y *clear*:
  Estas con asincrónicas y sirven para poner 1 (*set* o *preset*) ó 0 (*clear*) en el circuito.



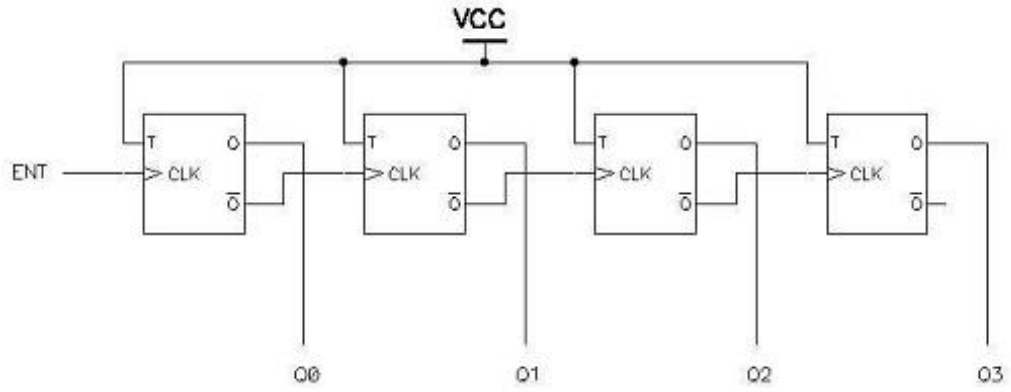
- Entrada *clock enabled* (*CE*):
  - Descripción:
    Sirve para indicarle al flip-flop cuándo debe modificar el contenido[13].
  - Circuito lógico:
    Regresivamente (para 4 bits):



	Q3	Q2	Q1	Q0
Inicial	0	0	0	0
Primer flanco	1	1	1	1
Segundo flanco	1	1	1	0
Tercer flanco	1	1	0	1
Cuarto flanco	1	1	0	0
Quinto flanco	1	0	1	1
Sexto flanco	1	0	1	0
Séptimo flanco	1	0	0	1
Octavo flanco	1	0	0	0
Noveno flanco	0	1	1	1
Décimo flanco	0	1	1	0
Onceavo flanco	0	1	0	1
Doceavo flanco	0	1	0	0
Treceavo flanco	0	0	1	1
Catorceavo flanco	0	0	1	0
Quinceavo flanco	0	0	0	1
Dieciseisavo flanco	0	0	0	0

[13] De esta forma se evita escribir en el flip-flop cuando no interesa (pese a que ocurra un flanco de reloj).
La información solo se guarda si *CE* = 1 y hay un franco de reloj.

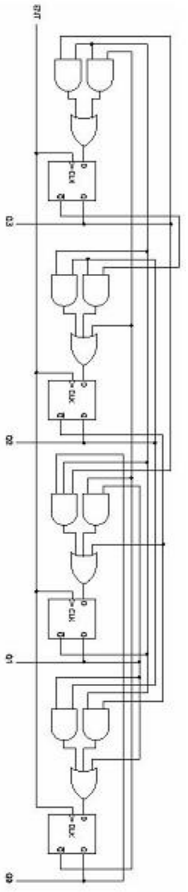
Progresivamente (para 4 bits):



- Contador a base de flip-flops D:
  - Descripción:
    Estos cuentan según cada flanco de reloj de forma progresiva.
  - Tabla de verdad:

Q3	Q2	Q1	Q0	D3	D2	D1	D0
0	0	0	1	0	0	1	1
0	0	1	1	0	0	1	0
0	0	1	0	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	0	1	0	1
0	1	0	1	0	1	0	0
0	1	0	0	1	1	0	0
1	1	0	0	1	1	0	1
1	1	0	1	1	1	1	1
1	1	1	1	0	0	0	1

  - Circuito lógico:

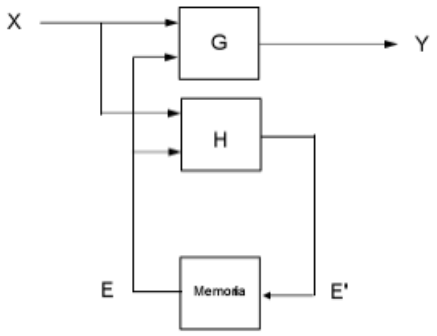


4.2 Modelación de circuitos secuenciales[14]

- Definición de circuito secuencial:
  Es un sistema en el cual la salida depende de la entrada y del valor de las entradas anteriores
- Modelación[15]:
  Matemáticamente puede modelarse como $\begin{cases} Y = G(X,E) \\ E' = H(X,E) \end{cases}$ donde:

Identificador	Descripción
Y	Salida
X	Entrada
E	Estado interno
E'	Valor del estado luego de la transición provocada por una nueva entrada
G	Función
H	Función

- Esquema modelo:



4.3 Diseño y ejemplo

- Pasos de diseño:
  1. Modelado del sistema a implementar mediante la especificación del Autómatas finitos deterministas (AFD)[16] correspondiente mediante un diagrama de Estados
  2. Deducir la Tabla de salida y la Tabla de transición a partir del Diagrama de estados
  3. Determinar el número de flip-flops necesarios para codificar todos los estados posibles y determinar la codificación a utilizar. Determinar la cantidad de bits y sistema de codificación para las entradas y las salidas
  4. Incorporar la codificación de los estados, entradas y salidas a las tablas de salida y transición
  5. Determinar las funciones lógicas que permitirán determinar el valor de las salidas en base a las entradas y las salidas de los flip-flops y el valor a presentar en las entradas de los flip-flops para almacenar el nuevo estado en ellos
  6. Minimizar las expresiones lógicas en dos niveles mediante Karnaugh
  7. Dibujar el circuito lógico en base a flip-flops, AND, OR y NOT

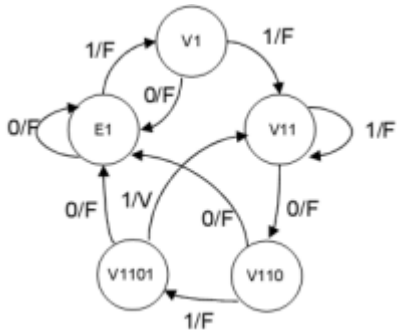
---

[14] Ver 5.1

[15] Este corresponde a una máquina de Mealy; una máquina de Moore se modelaría como $\begin{cases} Y = G(E) \\ E' = H(X,E) \end{cases}$

[16] Ver 5.1

- Ejemplo:
  Este ejemplo muestra un reconocedor de la secuencia 11011 en su único bit de entrada. Cuando esto ocurre devuelve verdadero (V), en caso contrario, devuelve falso (F).
  El diagrama de estados es:



La tabla de estados será:

E^n	Entrada	E^{n+1}	Salida
E1	0	E1	F
E1	1	V1	F
V1	0	E1	F
V1	1	V11	F
V11	0	V110	F
V11	1	V11	F
V110	0	E1	F
V110	1	V1101	V
V1101	0	E1	F
V1101	1	V11	V

Codificando F → 0 y V → 1 y simplificando el circuito final considerando los estados "indiferente" (X) se obtiene la tabla de codificación de los estados:

E^n $q_2\ q_1\ q_0$	Entrada	E^{n+1} $q_2\ q_1\ q_0$	Salida
000	0	000	0
000	1	001	0
001	0	000	0
001	1	010	0
010	0	011	0
010	1	010	0
011	0	000	0
011	1	100	0
100	0	000	0
100	1	010	1
101	0	XXX	X
101	1	XXX	X
110	0	XXX	X
110	1	XXX	X
111	0	XXX	X
111	1	XXX	X

Aplicando Karnaugh para simplificar las entradas y las salidas:

$q_0e \setminus q_2q_1$	00	01	11	10
00			X	
01			X	
11		1	X	X
10			X	X

$d_2 = q_1.q_0.e$

$q_0e \setminus q_2q_1$	00	01	11	10
00		1	X	
01	1		X	
11			X	X
10			X	X

$d_0 = q'_2.q'_1.q'_0.e + q_1.q'_0.e'$

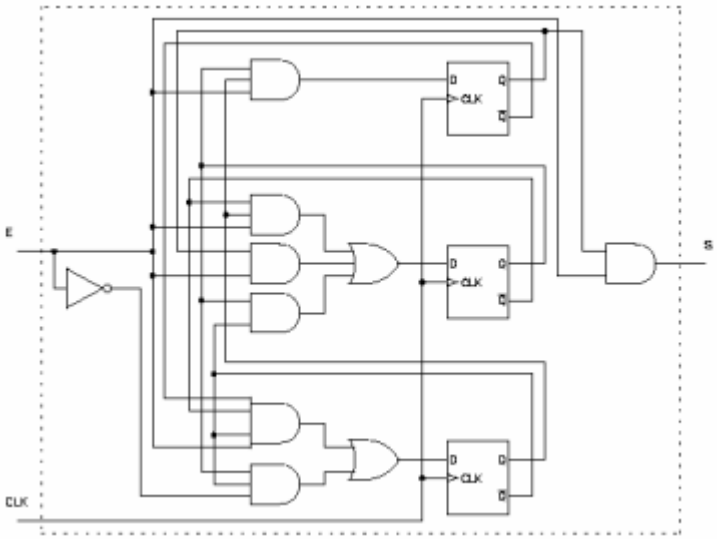
$q_0e \setminus q_2q_1$	00	01	11	10
00		1	X	
01		1	X	1
11	1		X	X
10			X	X

$d_1 = q'_1.q_0.e + q_2.e + q_1.q'_0$

$q_0e \setminus q_2q_1$	00	01	11	10
00			X	
01			X	1
11			X	X
10			X	X

$s = q_2.e$

El circuito que se obtiene es:



5. Máquinas

5.1 Máquinas de estado

- Definición de estado:
  Dadas dos secuencias de valores de entrada de un circuito secuencial, éstas serán equivalentes si a partir de cierto punto coinciden las entradas y las salidas.
  Cada clase de equivalencia que define la relación anterior es un estado.
- Definición de AFD:
  Es un modelo matemático de un sistema con entradas y salidas discretas que tiene el mismo comportamiento para la misma combinación entrada-salida.
- Definición de máquina de estado:
  Son cuaternas $M=(E,\Sigma,\delta,e_0)$ donde

Componente	Descripción
E	Conjunto finito de estados
e_0	Es el estado inicial ($e_0 \in E$)
Σ	Alfabeto de entrada
δ	$\delta : E\times\Sigma \to E$ función de transición

- Máquina de Mealy[17]:
  - Definición de Máquina de Mealy:
    Se define como $M=(E,\Sigma,\Delta,\delta,\lambda,e_0)$ donde

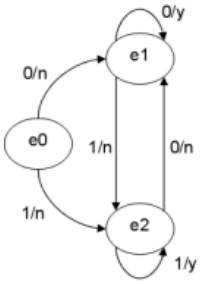
Componente	Descripción
Δ	Alfabeto de salida
λ	$\lambda : E\times E \to \Delta$ función de salida

  - Ejemplo:
    $M=(\{e_0,e_1,e_2\},\{0,1\},\{y,n\},\delta,\lambda,e_0)$ con

δ	0	1
e_0	e_1	e_2
e_1	e_1	e_2
e_2	e_1	e_2

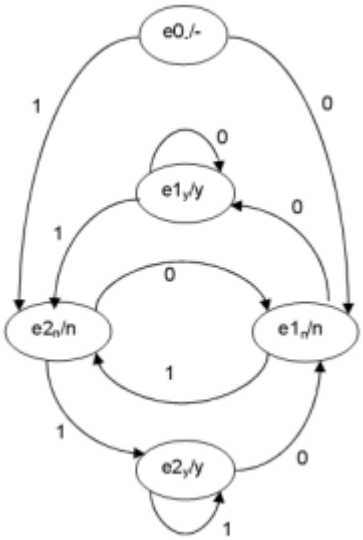
λ	0	1
e_0	n	n
e_1	y	n
e_2	n	y

Si bien esta definición es suficiente, también se puede definir como un grafo dirigido con nodos los estados y aristas las transiciones de estado *x…x/y…y* los valores de las entradas que provocan la transición y el valor de las salidas respectivamente.



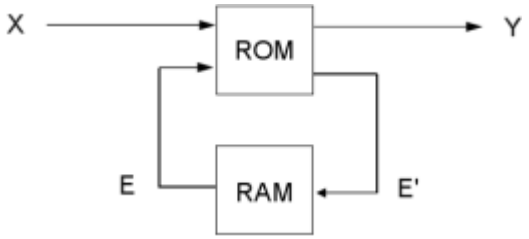
[17] En este caso se trata de una máquina de estado cuya salida está asociada con la transición

- Máquina de Moore[18]:
  - Definición:
    $M = (E, \Sigma, \Delta, \delta, \lambda, e_0)$
  - Ejemplo:

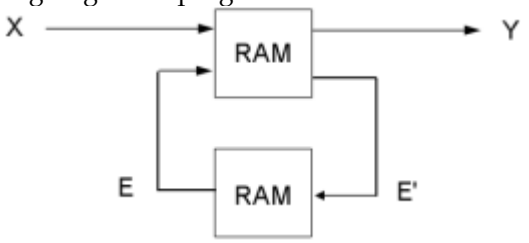


5.2 Máquina lógica general

- Características:
  - Una máquina combinatoria es un caso particular de una máquina secuencial, pudiéndose comportar tanto como un traductor de símbolos como una máquina reconocedora de secuencias.
  - Una máquina lógica general puede resolver cualquier problema computable[19].
- Diseño general:



- Diseño de la máquina lógica general programable:

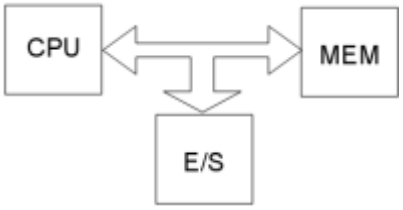


[18] En este caso se trata de una máquina de estado cuya salida está asociada con el estado
[19] Esto lo admitiremos sin demostrar

6. Computadora

6.1 Arquitectura de Von Neumann

- Definición de computadora:
  Es una máquina lógica general programable
- Arquitectura de Von Neumann:
  Utiliza el concepto de que normalmente una operación compleja puede dividirse como una secuencia ordenada de operaciones más simples.
  Para ello introdujo el concepto de "programa almacenado" como una "secuencia lógicamente ordenada de instrucciones", siendo éstas las operaciones básicas que se implementa en hardware.
- Bloques constructivos de la arquitectura de Von Neumann:



Componente	Función
CPU	Ejecuta los programas
Memoria (MEM)	Almacena el programa y los datos
Entrada/Salida (E/S)	Comunica el computador con los usuarios

- Variantes de la arquitectura de Von Neumann:

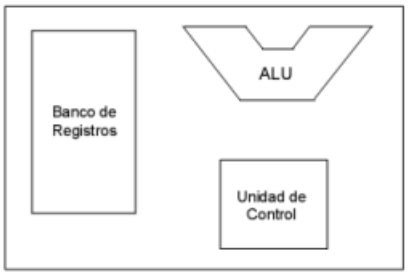
Variante	Descripción
Set de instrucciones	La cantidad de instrucciones disponibles y la calidad y complejidad de las operaciones implementadas en el hardware de la CPU
Formato de instrucción	La forma en que se codifican las instrucciones
Set de registros	La cantidad de registros disponibles al programador, así como la función que pueden cumplir
Modos de direccionamiento	Formas de generar las direcciones para hallar los operandos o almacenar los resultados de las operaciones
Manejo de la E/S	Forma de comunicación con los periféricos
Manejo de interrupciones	Manejo de una forma particular de invocar a ciertas subrutinas de los programas tratados en los siguientes puntos

6.2 CPU

- Componentes:

Componentes	Descripción
Unidad Aritmético-Lógica (ALU)	Circuito lógico que implementa operaciones de aritmética binaria, lógica y de desplazamiento o rotación de bits
Unidad de control (UC)	Circuito secuencial que implementa del denominado "ciclo de instrucción", permitiendo acceder a la siguiente instrucción de un programa, leer sus operandos, efectuar la operación indicada en la ALU y guardar el resultado de la misma
Banco (Set)	Son una serie de posiciones de memoria, ubicadas dentro de la propia CPU, que permiten un acceso a operando y lugares de almacenamiento de resultados mucho más veloz que si estuvieran en el sistema de memoria normal. Algunos de estos registros son de uso interno de la propia CPU y otros son utilizables por el programador

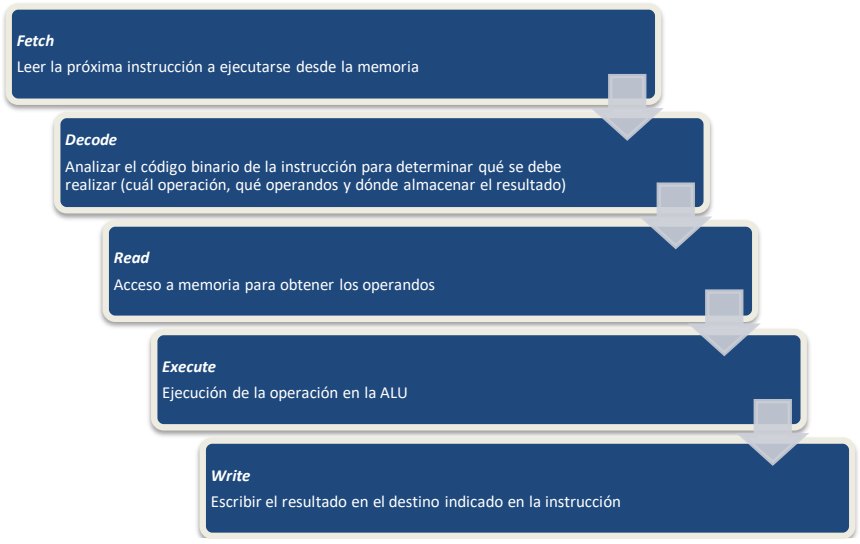
- Diagrama de la CPU:



- Componentes del Banco de registros:

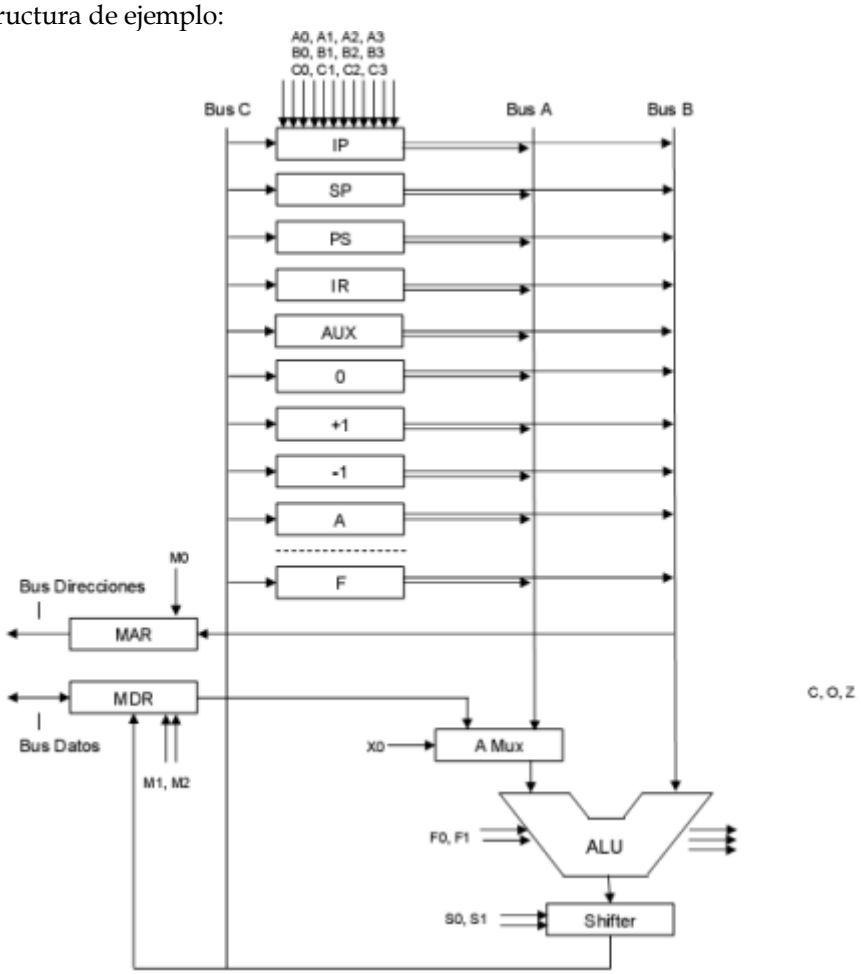
Componente	Descripción
Registro totalmente visible	Contiene operandos o direcciones para la utilización en instrucciones. Es accesible al programador
Registro parcialmente visible	Contiene funciones especiales. Es manipulado indirectamente por el programador para funciones específicas. Ejemplos de este registro: IP, PC, SP PS y FLAGS[20]
Registros internos	Los utiliza la CPU para poder ejecutar las instrucciones. Almacena constantes, el estado de la UC, la instrucción en ejecución y resultados intermedios de cálculos de direcciones entre otras.

- Unidad de control:
  Es una máquina de estados que realiza el "ciclo de instrucción"
- Ciclo de instrucción:
  Secuencia de micro-acciones que realiza la CPU para lograr ejecutar una instrucción del programa almacenado en memoria:



[20] Más adelante se detalla el significado y función de lo que estas siglas representan

- Estructura de ejemplo:



Recurso	Descripción (si es un registro, lo que almacena)
Registro IP	Puntero de instrucción
Registro SP	Puntero a la pila
Registro PS	Estado del procesador (en particular, FLAGS Carry, Overflow y Zero)
Registro IR	Instrucción leída desde memoria
Registro AUX	Datos internos de a la UC (por ejemplo, dirección de memoria)
Registro 0	Constante 0
Registro +1	Constante 1
Registro -1	Constante -1
Registros A a F	De uso general
Registro MAR (Memory Adress Register)	Dirección de memoria que se presenta en el bus de direcciones de la memoria durante una operación de lectura o escritura de la misma
Registro MDR (Memory Data Register)	Dato leído de la memoria (en una operación de lectura) o dato a escribir en la memoria (en una operación de escritura)
A MUX	Multiplexor que elige entre dos entradas posibles
Shifter	Unidad encargada de realizar la operación de desplazamiento de 0 bits o 1 bit a derecha o izquierda

Se omite la UC que tiene como entrada los bits del Registro IR y analiza qué debe hacer en función de lo especificado en la instrucción y como salidas toda las señales de control indicadas en el diagrama.

Señal de control	Función	Ejemplos
A0, A1, A2, A3	Seleccionar un registro y conectar su salida al bus A	• 0001 – IP • 0111 – -1
B0, B1, B2, B3	Seleccionar un registro y conectar su salida al bus B	• 0001 – IP • 0100 – IR
C0, C1, C2, C3	Seleccionar un registro y conectar su entrada al bus C	• 0001 – IP • 1110 – F
X0	Seleccionar si se conecta a su salida la entrada que viene desde el MDR o desde el bus A	• 0 – MDR • 1 – Bus A
F0, F1	Codificar l operación a realizar por la ALU	• 00 – Identidad • 11 – Nand bit a bit
S0, S1	Codificar la operación de desplazamiento	• 00 – No desplaza • 10 – Un bit a la izquierda
M0	Controlar la carga en el MAR	• 1 – Carga
M1, M2	Controlar la carga en el MDR, desde qué líneas se produce y qué líneas se activa	• 01 – Carga desde el bus de datos de memoria y activa salidas hacia el MUX

- Ejecución de ejemplo:

Ejecución de la suma de un operando de memoria con modo de direccionamiento indirecto (B) y un registro (A) guardando el resultado en otro registro (E)

Fetch
- Conectar la salida del registro IP al bus B (B0 = 0, B1 = 0, B2 = 0, B3 = 1)
- Seleccionar cargar MAR (M0 = 1)
- Seleccionar conectar el bus de datos de memoria con el mux (M1 = 1, M2 = 1)
- Seleccionar MDR en el A MUX (X0 = 0)
- Seleccionar la operación identidad A en la ALU (F0 = 0, F1 = 1)
- Seleccionar no desplazar en el Shifter (S0 = 0, S1 = 0)
- Conectar la entrada del registro IR al bus C (C0 = 0, C1 = 0, C2 = 0, C3 = 1)
- Activar la lectura en el bus de control de memoria (R/W = 0)

Decode
- Se analiza el código binario de la instrucción
- Se implementa en la UC una función que toma como entrada el contenido del IR y da como salida los valores apropiados de las señales de control, condicionando el próximo estado de la máquina secuencial

Read
- Conectar la salida del registro B al bus B (B0 = 1, B1 = 0, B2 = 0, B3 = 1)
- Seleccionar cargar MAR (M0 = 1)
- Seleccionar cargar el MDR desde el bus de datos de memoria (M1 = 0, M2 = 1)
- Activar la lectura en el bus de control de memoria (R/W = 0)

Execute
- Conectar la salida del registro A al bus B (B0 = 0, B1 = 0, B2 = 0, B3 = 1)
- Seleccionar MDR en el A MUX (X0 = 0)
- Conectar la entrada del registro E al bus C (C0 = 1, C1 = 1, C2 = 0, C3 = 1)
- Seleccionar la operación suma en la ALU (F0 = 0, F1 = 1)
- Seleccionar no desplazar en el Shifter (S0 = 0, S1 = 0)

Write
- No corresponde en este ejemplo

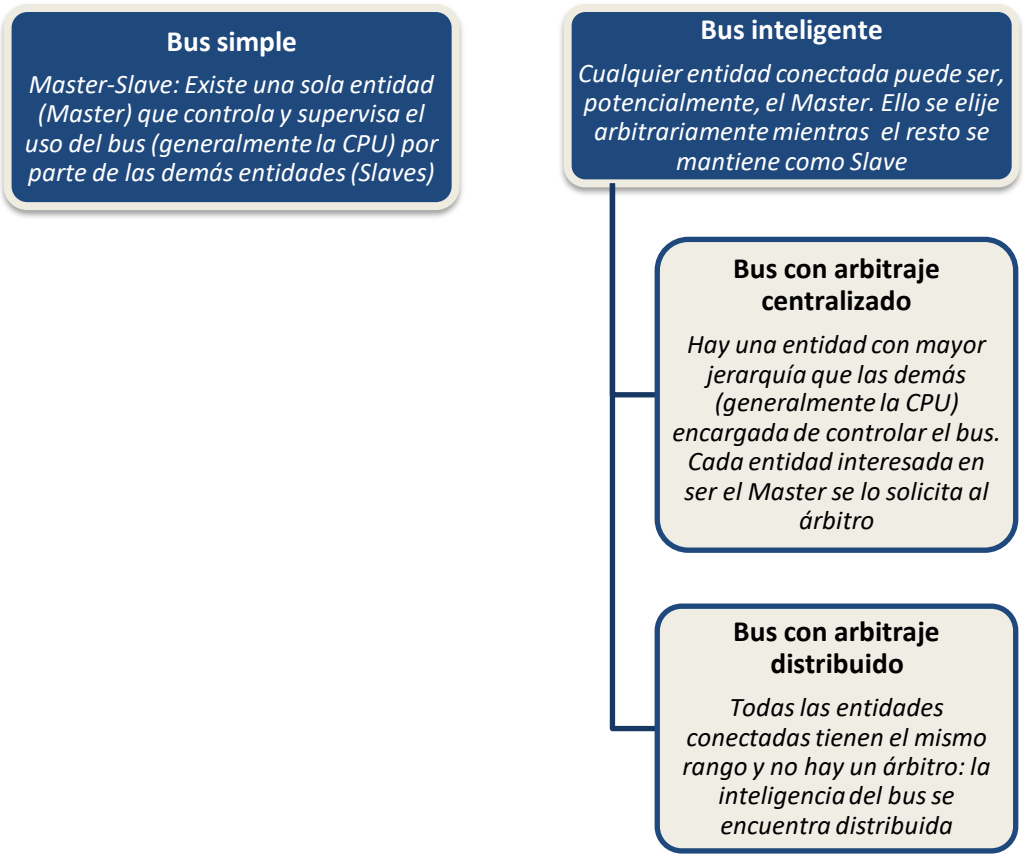
- Lógica cableada:
  Una CPU en base a la lógica cableada se construye en base a un diagrama de estados traduciéndola a una Máquina de Mealy.
- Lógica microprogramada:
  Para cada instrucción en la CPU existirá un microprograma consistente en una secuencia lógica de microinstrucciones que establecerán el orden de los eventos necesarios para lograr la ejecución de la instrucción en la CPU, incluyendo todas las "bifurcaciones" de la secuencia que se requieran en base a los distintos tipos de direccionamiento y cualquier otra variante que admita la instrucción específica.
- Microprogramación vertical:
  Cada microinstrucción contiene los bits codificados de las señales de control de los distintos caminos de la señales de datos.
- Microprogramación horizontal:
  Utiliza microinstrucciones con los bits de control sin codificar.

6.3 E/S

- Definición de bus:
  Es una agrupación de "líneas" que comunican tres tipos de información en el computador: Dirección, Datos y Control
- Tipos de buses:

<table>
<tr><th>Tipo</th><th>Composición y funcionalidad</th></tr>
<tr><td>Bus de direcciones</td><td>Líneas de conexión que transportan las direcciones de memoria o E/S a ser accedidas durante la transferencia</td></tr>
<tr><td>Bus de control</td><td>Líneas de conexión que transportan señales que controlan el uso del bus y la comunicación sobre él. Éstas son:
<table>
<tr><th>Señal</th><th>Función</th></tr>
<tr><td>Memory_Read</td><td>Indica una operación de lectura sobre la memoria</td></tr>
<tr><td>Memory_Write</td><td>Indica una operación de escritura sobre la memoria</td></tr>
<tr><td>I/O_Read</td><td>Indica una operación de lectura sobre la E/S</td></tr>
<tr><td>I/O_Write</td><td>Indica una operación de escritura sobre la E/S</td></tr>
<tr><td>Bus_Request</td><td>Indica que un sub-sistema desea tomar control del bus para iniciar una transferencia</td></tr>
<tr><td>Bus_Grant</td><td>Confira que el bus está disponible para quien lo solicitó</td></tr>
<tr><td>Transfer_ACK</td><td>Confirma la recepción de una transferencia de información</td></tr>
<tr><td>Interrupt_Request</td><td>Indica un pedido de interrpción hacia un sub-sistema</td></tr>
<tr><td>Interrupt_ACK</td><td>Confirma la aceptación del pedido de interrupción</td></tr>
<tr><td>Clock</td><td>Sincroniza las actividades del bus y sus señales</td></tr>
<tr><td>Reset</td><td>Fuerza el reset de todos los componentes conectados al bus</td></tr>
</table></td></tr>
<tr><td>Bus de datos</td><td>Líneas de conexión que transportan la información que es transferida sobre el bus</td></tr>
</table>

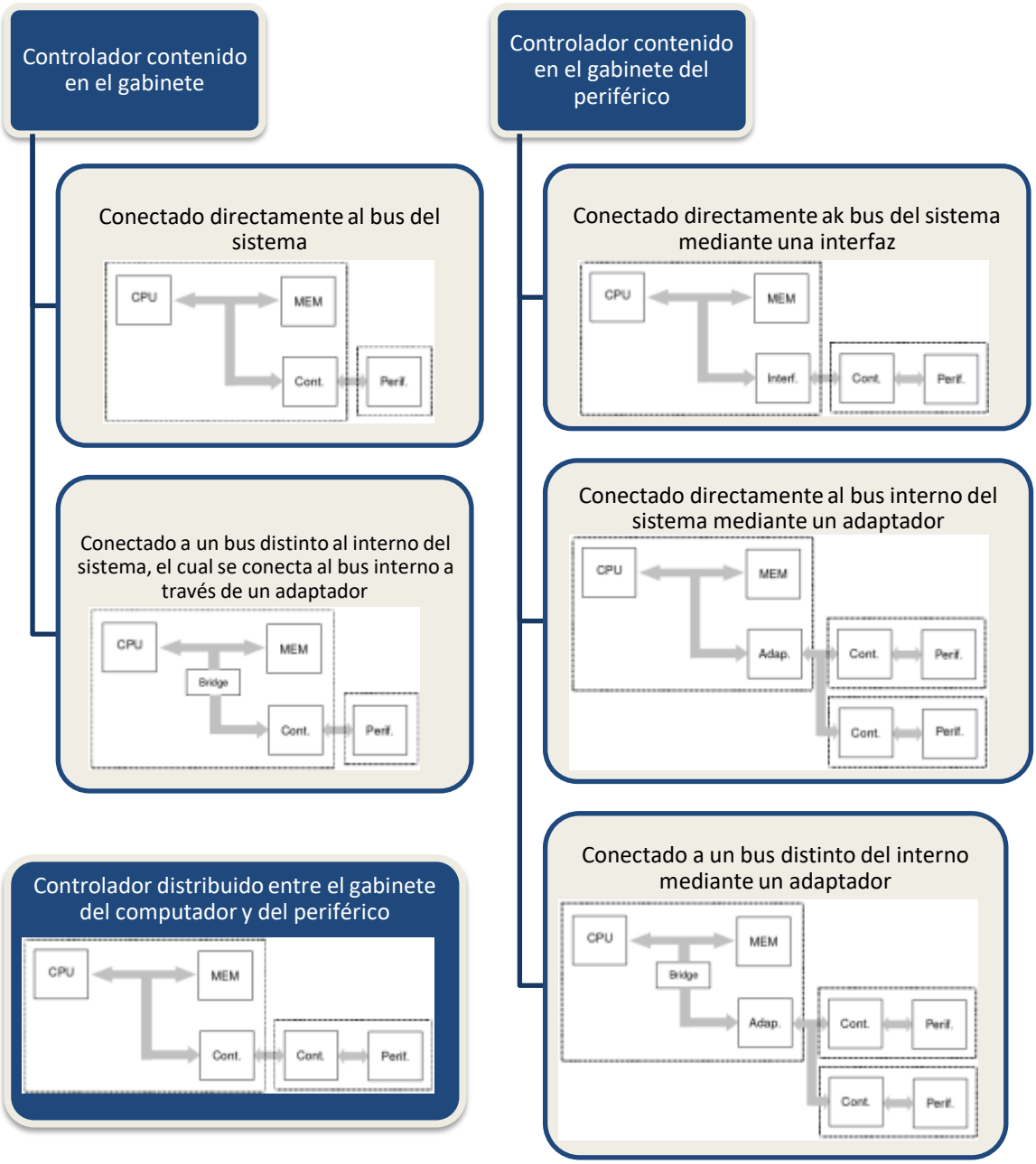
- Clasificación de buses según la complejidad:



- Clasificación de buses según la aplicación a la que se destinan:

- Periféricos:
  Son los que realizan el vínculo del computador con el mundo exterior. A través de ellos se realiza el ingreso de programas y datos a procesar y se obtienen los resultados del proceso en un formato que sea legible para el ser humano ó se provoca alguna alteración del mundo físico circundante.
- Definición de controlador:
  Es la parte del periférico que contiene su inteligencia.

- Conexión de los controladores:



- Acceso a los controladores:
  El acceso se obtiene mediante las posiciones de memoria especiales de estos dispositivos.
- Arquitectura[21]:
  Hay dos tipos:
  1. La CPU dispone de un espacio de direcciones reservado que es accedido mediante instrucciones especialmente destinadas
  2. La CPU accede a los controladores como si se tratara de posiciones normales de memoria

[21] Las posiciones de memoria referidas se les llama registros

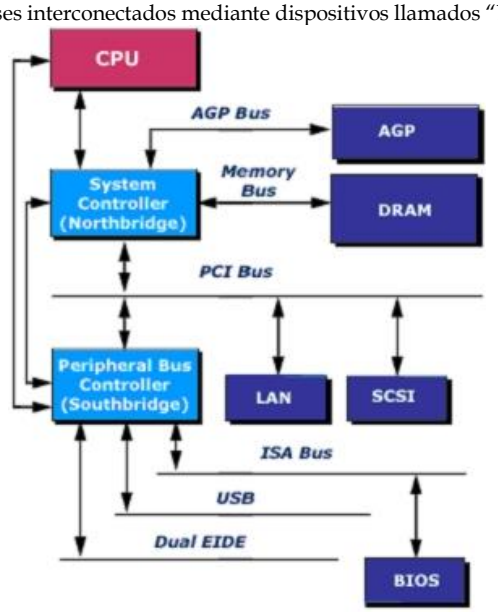
- Manejo de los registros:
  Estas memorias no siempre se comportan como las "comunes" sino como se detalla a continuación:

Nombre	Comportamiento
Sólo lectura	El registro puede solamente ser leído; la escritura no surte efecto
Sólo escritura	Er registro solo puede ser escrito; la lectura tiene un resultado impredecible
Lectura/Escritura independiente	Hay dos registros diferentes, uno para solo lectura y otro para solo escritura independientes según se detalla en los dos puntos anteriores
Lectura/Escritura normal	Se comportan como una posición de memoria normal

- Tipos de registros:

Tipo	Contenido
Entrada	Contiene un dato destinado a la CPU. Proviene del periférico, el controlador o la línea de comunicaciones
Salida	Contiene un dato destinado al periférico, el controlador o la línea de comunicaciones. Proviene de la CPU
Estado	Contiene bits que indican el estado del controlador o del periférico
Control	Contiene bits que le indican al controlador o al periférico realizar determinada acción

- Arquitectura de E/S y buses de un PC:
  Se basa en buses interconectados mediante dispositivos llamados "bridges" según:

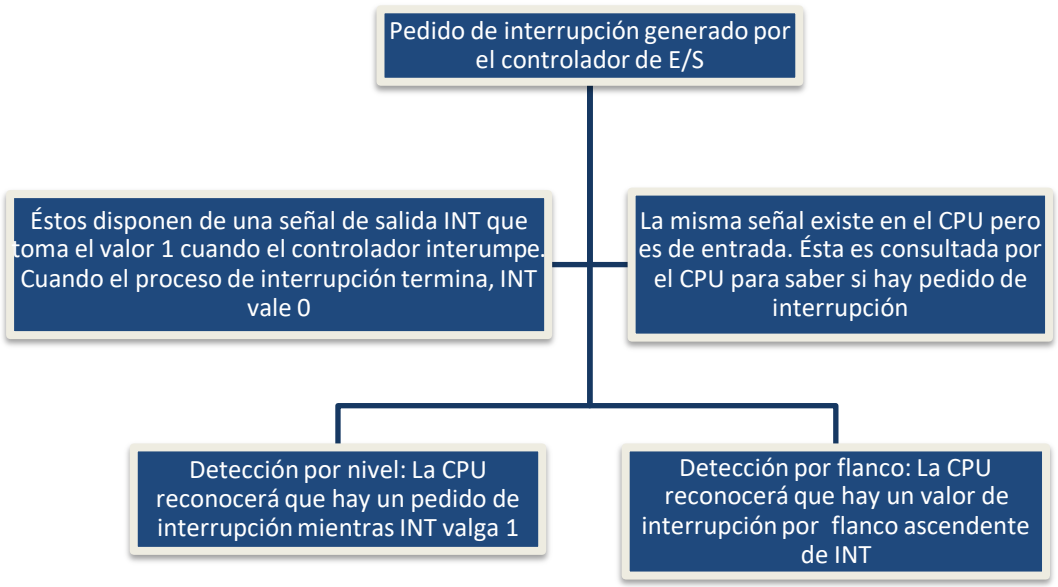


6.4 Interrupciones

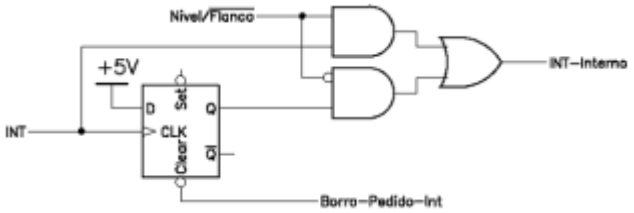
- Definición de interrupción[22]:
  Una interrupción consiste en un mecanismo que provoca la alteración del orden lógico de ejecución de instrucciones como respuesta a un evento externo, generado por el hardware de E/S en forma asincrónica al programa que está siendo ejecutado.

[22] Una definición alternativa es: Una interrupción consiste en un mecanismo que le permite al hardware la invocación de una rutina fuera del control del programa que está siendo ejecutado.

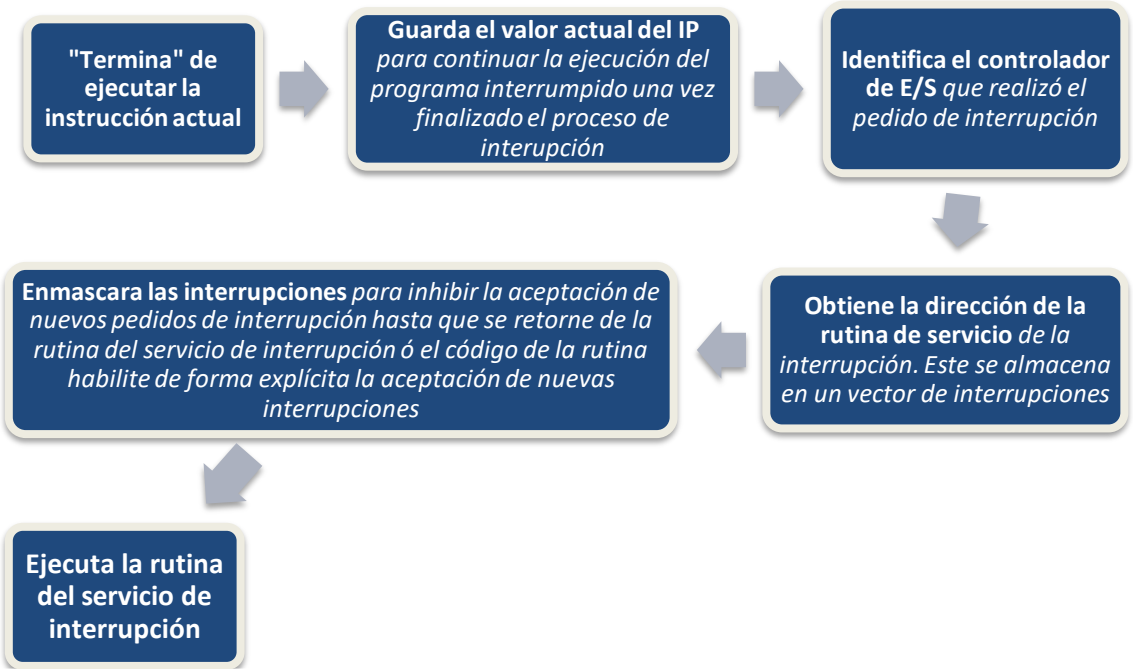
- Pedido de interrupción:



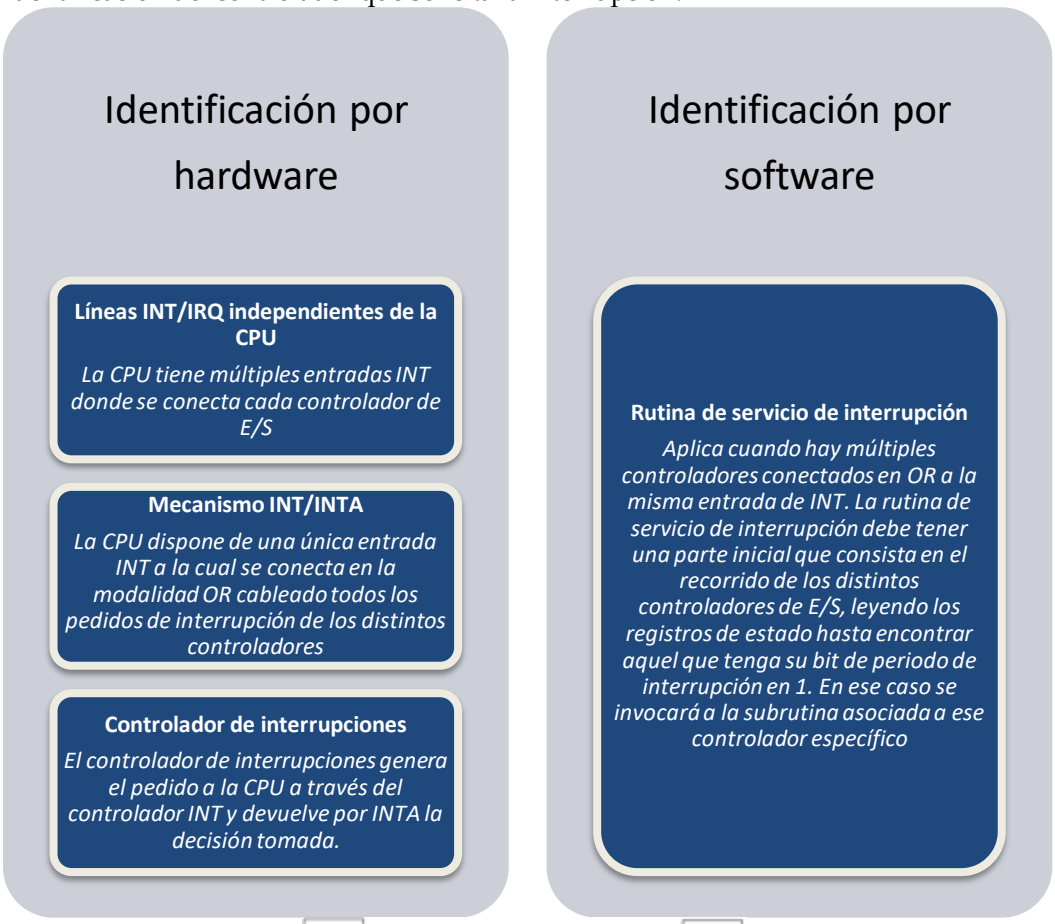
- Circuito de la entrada INT:

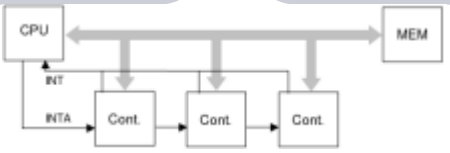


- Pedido de interrupción:

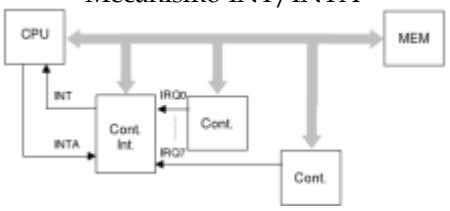


- Identificación del controlador que solicita la interrupción:





Mecanismo INT/INTA



Controlador de interrupciones

- Habilitación de interrupciones:
  La CPU puede aceptar o rechazar pedidos de interrupción, para ello se vale de lo siguiente:

Implementación	Función
Enmascaramiento	La CPU inhibe la aceptación de todas las solicitudes de interrupción (éste es su estado inicial)
Deshabilitación	La CPU actúa individualmente sobre cada controlador en forma de inhibir su eventual pedido de interrupción

- Tipos de prioridades:
  - **Prioridad fija:** Se atiende en orden jerárquico
  - **Prioridad configurable:** Cambia en función de las condiciones del sistema
  - **Sin prioridad:** Se implementa un sistema de selección Éstas funcionan según el comportamiento de interrupciones:

- Superposición de interrupciones:

Comportamiento	Descripción
Interrupciones simultáneas	Se adopta una de las estrategias descriptas en el punto anterior
Interrupción de interrupción	La atención o no de la solicitud dependerá del esquema de jerarquía/prioridades que tenga implementada la CPU o el controlador de interrupciones

- Definición de rutinas de interrupción:
  Son la pieza de código que se ejecuta como resultado del mecanismo que desencadena la CPU al procesar un pedido de interrupción. Éstas deben preservar el contexto.
- Estrategias de conservación del contexto:

Estrategia	Descripción
Máquina no dedicada	En la máquina donde se ejecuta la rutina de interrupción existen múltiples programas en ejecución los cuales han sido programados por distintos equipos sin coordinación
Máquina dedicada	La máquina está completamente dedicada a una función rogramas y rutinas que se ejecutan están desarrollados por el programadores o por equipos fuertemente coordinados

- Estrategias de programación:
  - Toda la lógica del sistema se implementa en el programa principal y las rutinas de interrupción solamente modifican banderas indicando que se ejecutaron
  - Toda la lógica se implementa en las rutinas de interrupción y el programa principal o bien está en loop infinito (máquina dedicada) o bien instalada e inicializa las rutinas de interrupción y termina (máquina no dedicada)
  - La lógica se implementa en parte en el programa principal y en parte en las rutinas de interrupción