INTRODUCCIÓN A LOS MICROPROCESADORES

INTRODUCCIÓN A LOS MICROPROCESADORES

Los microprocesadores (***mP***) y las microcomputadoras (***mC***) son máquinas digitales sincrónicas. En el presente trabajo se tratarán sus dos aspectos básicos. El primero, lo constituye la ***circuitería*** integrada conformada por la ***U***nidad ***C***entral de ***P***roceso (***CPU***, *por sus siglas en inglés*), la memoria, los puertos de ***ENTRADA/SALIDA*** (***E/S***), el reloj y la circuitería lógica de control conocida como ***mecamática*** (*hardware*) El segundo aspecto, es la ***programación*** del *hardware* para realizar tareas de control (***software***) Esta programación está orientada al lenguaje de máquina directamente y a través de un lenguaje de muy bajo nivel (***ensamblador***)

HARDWARE

El ***hardware*** está compuesto por circuitos integrados (***IC***, *por sus siglas en inglés*) de muy alta densidad y prácticamente uno por cada bloque básico, es decir: Un ***IC*** para la ***CPU***, otro para la ***M***emoria de ***A***cceso ***A***leatorio (***RAM***, *por sus siglas en inglés*), uno más para la ***M***emoria de ***S***ólo ***L***ectura ***P***rogramable (***PROM***, *por sus siglas en inglés*), otro más para los puertos de ***E/S*** paralelos y, finalmente, uno para los puertos de ***E/S*** seriales.

Teoría general de microprocesadores y microcomputadoras

Definiciones preliminares

Computador digital

El computador digital es una máquina de proceso de información al cual se le debe proporcionar un conjunto único de ***instrucciones*** (programa) para el trabajo que deba ejecutar. El programa se almacena en la memoria interna del computador antes de ser ejecutado.

El computador digital consta básicamente de los bloques mostrados en la ***Figura 1.1***:

Definición de ***microcomputador*** y ***microprocesador***.

Definición 1. Un ***microcomputador*** (***μC***)es un dispositivo que utiliza a un ***microprocesador*** como ***U***nidad de ***P***roceso ***C***entral.

Definición 2. Un ***microprocesador*** (***μP***) es una máquina que procesa números binarios (***datos***) siguiendo una secuencia organizada de pasos (***programa***) A cada paso de la secuencia se le llama ***instrucción***.

La ***Figura 1.2*** ilustra las definiciones anteriores:

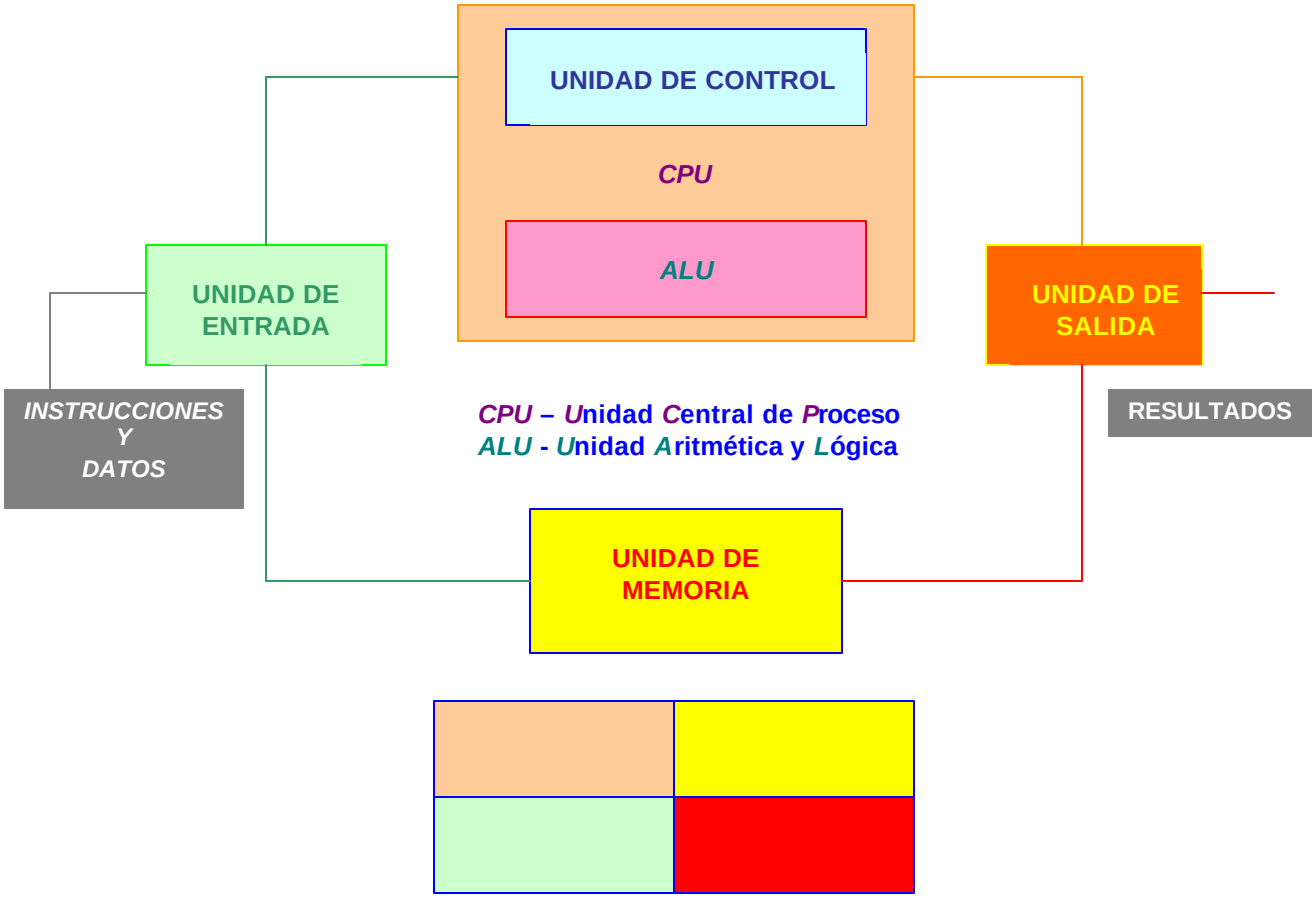


***FIGURA 1.1*. Diagrama a bloques básicos de un *computador digital*.**

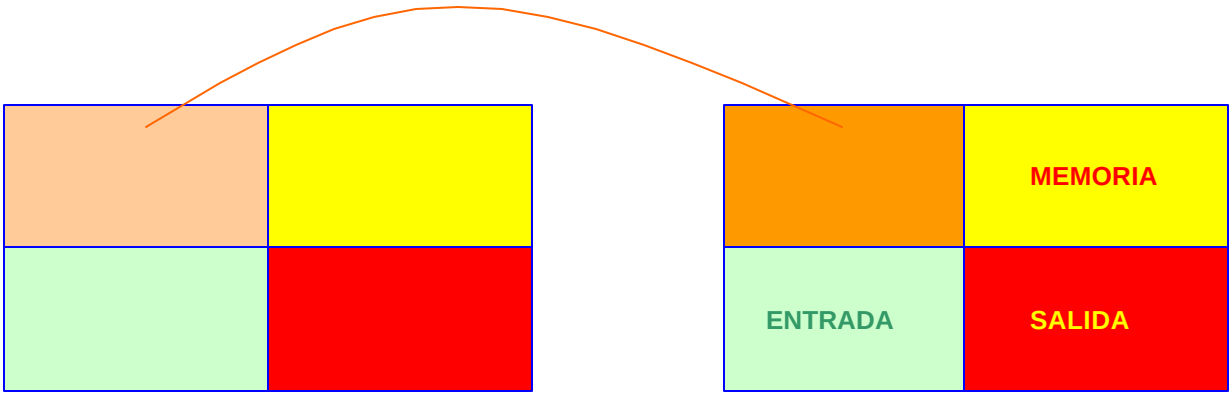


FIGURA 1.2*. Interrelación entre la *CPU* y el *µP

Las *microcomputadoras* son máquinas con las siguientes características:

1. ***Medio de entrada*** a través del cual se introducen las ***instrucciones*** y los ***datos***.
2. ***Memoria*** desde la cual, los ***datos*** e ***instrucciones*** pueden ser obtenidos por el *CPU* (***µP***) y donde se pueden almacenar resultados parciales y finales, esto es, ***M***emoria de ***A***cceso ***A***leatorio (***RAM***, *por sus siglas en inglés*) Otra sección de la ***memoria*** está compuesta por código fijo, llamada ***M***emoria de ***S***ólo ***L***ectura (***ROM***, *por sus siglas en inglés*)
3. ***Sección de cálculo*** la cual debe ser capaz de realizar operaciones ***aritméticas*** y ***lógicas*** sobre cualquier ***dato*** tomado de la ***memoria***.

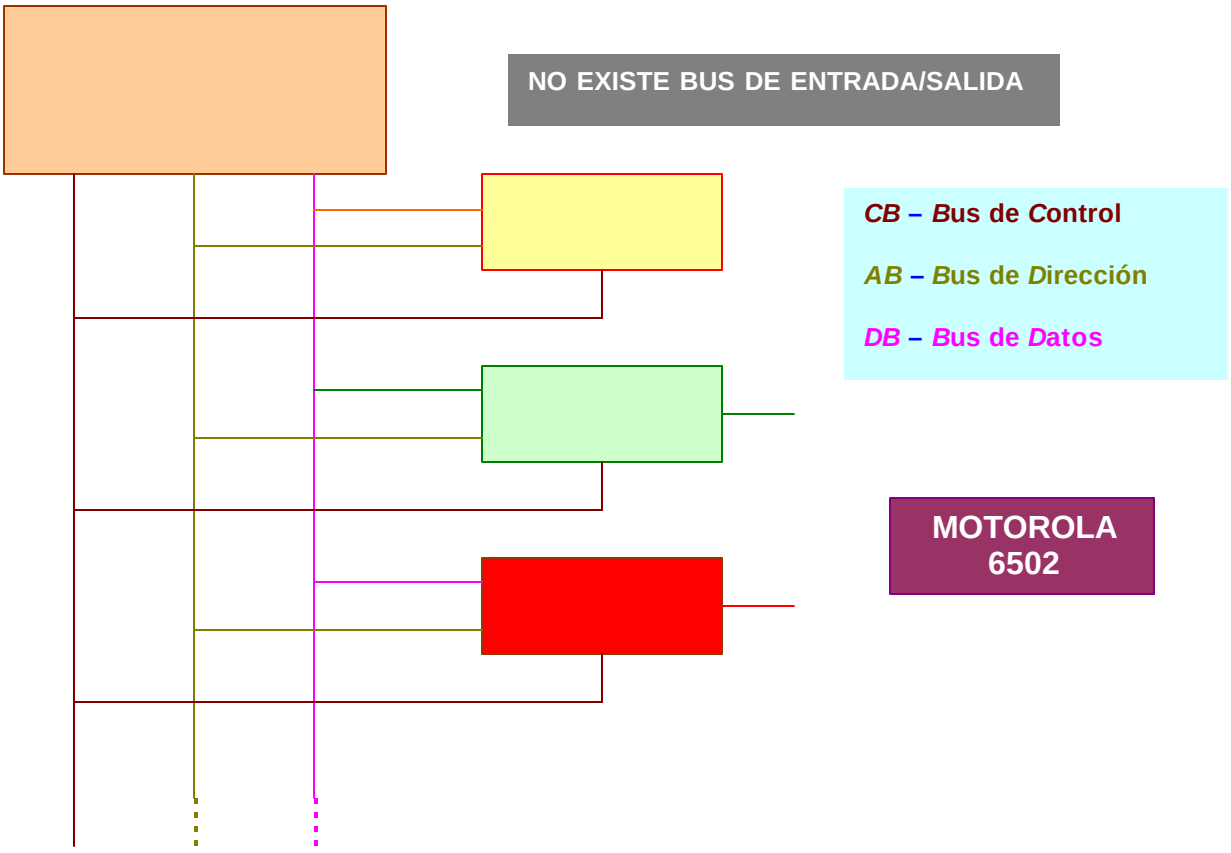
4. *Capacidad de* ***decisión*** por medio de la cual se pueden *seleccionar cursos* ***alternos*** *de acción* basándose en resultados calculados.
5. ***Medio*** *de salida* por medio del cual se entreguen al usuario los ***resultados***.

Las máquinas que satisfacen estas condiciones se les conoce como computadoras clase *HARDVARD*. Si además de estas condiciones, las ***instrucciones*** se almacenan en la misma forma que los ***datos*** (cada uno igualmente accesible a la sección de cálculo de la *µC*), entonces las **instrucciones** se pueden tratar como ***datos*** y la máquina puede modificar sus ***instrucciones***. A tal máquina se le conoce como computadora clase *VON NEWMAN* o *PRINCETON*.

El *diseño* de todos los *µC* se basa en **4** bloques:

a. Dispositivos de ***entrada*** (*punto **1***)
b. ***Memoria*** (*punto **2***)
c. ***Microprocesador*** (*puntos **3** y **4***)
d. Dispositivos de ***salida*** (*punto **5***)

Las ***arquitecturas*** de los *µC* más sobresalientes, en donde se muestran los grupos de líneas interconectados a los ***bloques básicos***, se presentan en la ***Figura 1.3***:



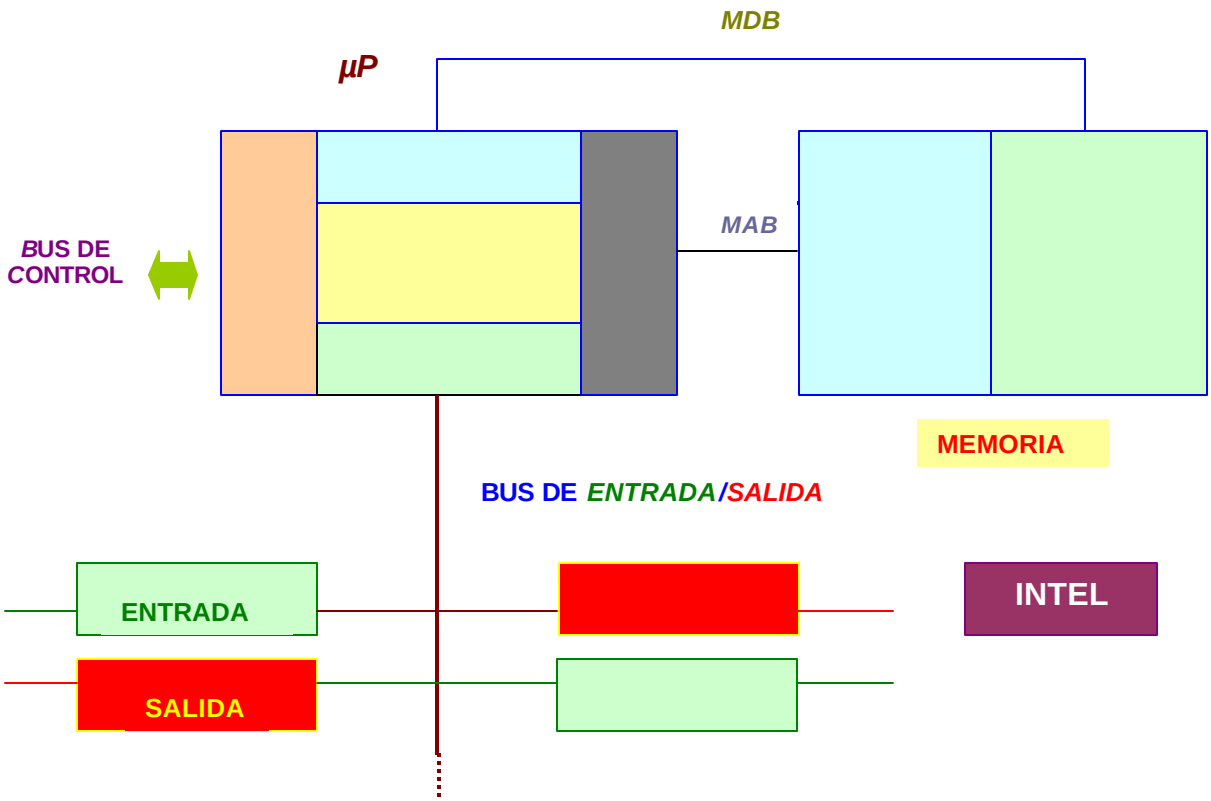


FIGURA 1.3. Arquitecturas más sobresalientes de *microcomputadoras*

Donde:

MDB = *B*us de *D*atos de *M*emoria	*(por sus siglas en inglés)*
MAB = *B*us de *D*irección de *M*emoria	*(por sus siglas en inglés)*
MAR = *R*egistro de *D*irección de *M*emoria	*(por sus siglas en inglés)*
ALU = *U*nidad *A*ritmética y *L*ógica	*(por sus siglas en inglés)*
MDR = *R*egistro de *D*atos de *M*emoria	*(por sus siglas en inglés)*

Las operaciones del *µC* son sincronizadas por un oscilador (***reloj***) Se requiere de un cierto número de pulsos de ***reloj*** para efectuar las pruebas que se especifican en una ***instrucción***.

Un ciclo de ***instrucción*** consiste de uno o más ciclos de ***máquina***. Durante un ciclo de ***máquina*** se realizan los siguientes ***subciclos***:

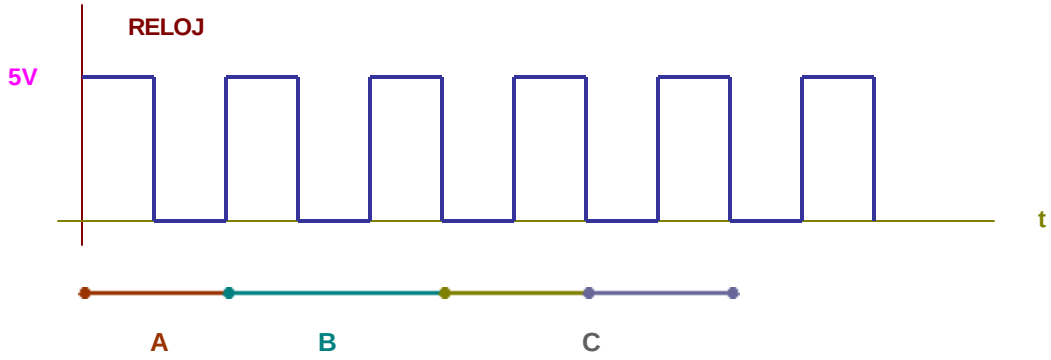
1. *Subciclo de* ***búsqueda*** **(***fetch subcycle***)**

   - El *µP* proporciona la *dirección* de una *instrucción* residente en ***memoria*** a través del *MAB*.
   - La unidad de ***memoria*** **decodifica** la *dirección* y el contenido de ésta se **transfiere** al *MDR* (el *µP* **lee** el contenido de la *dirección*)

2. *Subciclo de* ***ejecución*** **(***execution subcycle***)**

   - La instrucción se **decodifica** y posteriormente se **ejecuta**.

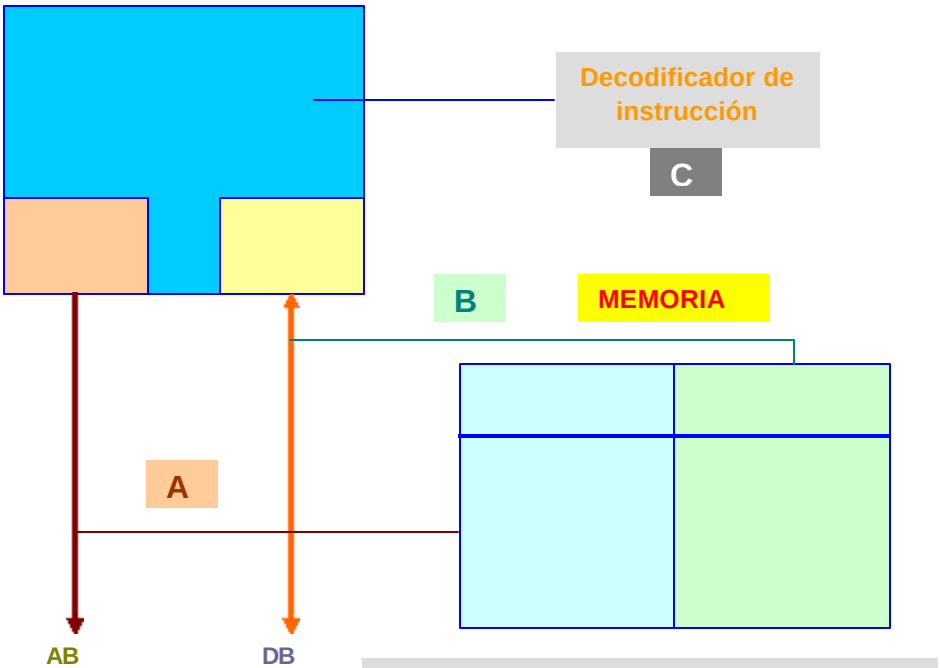
La ***Figura 1.4*** muestra una secuencia típica de estos ***subciclos***.



Subciclo de búsqueda — Subciclo de ejecución

Ciclo de máquina

A	Se envía dirección a la memoria
B	Lee instrucción desde la memoria
C	- Decodifica la instrucción - Ejecuta la instrucción



PC	*C*ontador de *P*rograma
MAR	*R*egistro de *D*irección de *M*emoria
MDR	*R*egistro de *D*irección de *D*atos
AB	*B*us de *D*irección
DB	*B*us de *D*atos

FIGURA 1.4*. Secuencia típica de los *subciclos* de *búsqueda* y *ejecución

Proceso de palabras de instrucción y palabras de datos

Durante un *ciclo de* ***instrucción*** se procesan dos tipos de ***palabras***:

- ***Palabras*** de ***instrucción***
- ***Palabras*** de ***datos***

Proceso de palabras de instrucción

La ***Figura 1.5***, muestra el proceso a bloques de las ***palabras*** de ***instrucción***, mostrando los pasos requeridos entre los distintos elementos del ***microprocesador*** y la ***memoria***:

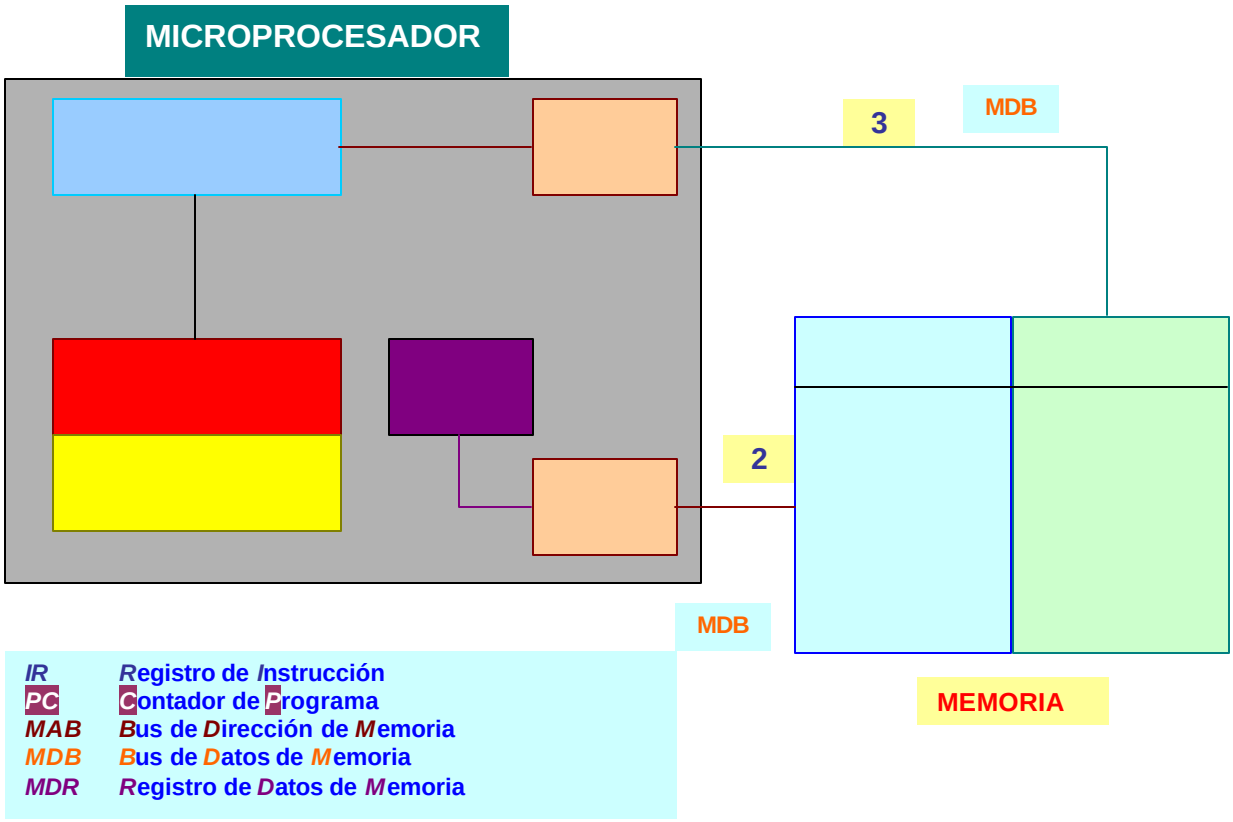


FIGURA 1.5*. Proceso de *palabras* de *instrucción

Durante un *ciclo* ***máquina*** se efectúan las siguientes ***operaciones***:

1. Al inicio del ciclo, el contenido del PC se ***coloca*** en el MAR.
2. El contenido del MAR se ***transfiere*** a través del ***MAB*** a la memoria. La memoria ***decodifica*** la dirección enviada.
3. Se ***lee*** la instrucción desde la memoria (vía ***MDB*** hacia el ***MDR***)
4. La instrucción se ***coloca*** en el registro de instrucción ***IR***.
5. La instrucción es ***decodificada*** por el decodificador de instrucción.
6. ***Ejecución*** de la instrucción.
7. El PC se **incrementa** o ***desactiva*** de acuerdo a la instrucción que se está ejecutando

Pila de datos (*stack*)

Cuando sucede una ***interrupción***, deseamos que después de atenderla, el programa continue su ejecución donde se quedó. Para lograr esto, es necesario que todos los ***registros internos*** del ***µP*** se almacenen en memoria y a esta área se le llama ***stack***.

Después de atender la ***interrupción***, sacamos del ***stack*** los valores de los ***registros*** del ***µP***.

Con esto, reanudamos el programa en la instrucción en que se suspendió.

Esta fue una descripción general de un ***µC***, en la práctica se debe tomar en cuenta que cada ***µC*** tiene su propia organización, la cual combina o expande las características descritas con anterioridad.

Microprocesador ideal

Definición:

Es un ***dispositivo digital*** que acepta ***datos*** desde cualquier número de líneas de ***entrada***, *procesa los* ***datos*** de acuerdo al dictado de un ***programa*** almacenado en ***memoria*** y produce cualquier número de señales de ***salida*** como consecuencia del ***proceso*** de datos, como lo muestra la ***Figura 1.6***.

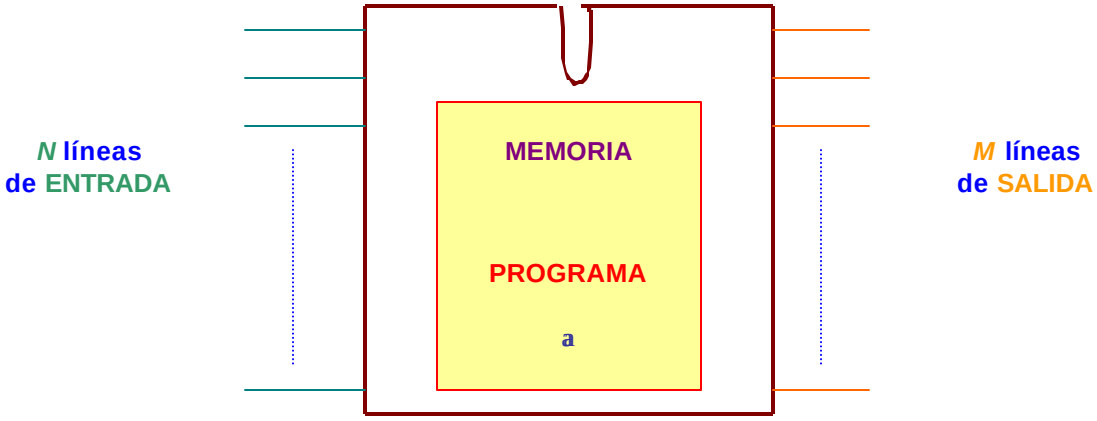


FIGURA 1.6. Líneas de ***entrada*** y ***salida*** de un ***microprocesador***

Las señales que se aplican a las líneas de ***entrada*** se les conoce como ***datos de*** entrada. Éstos pueden venir de ***interruptores*** (switches), ***sensores***, ***convertidores*** **A/D** (**A**nalógico / **D**igital), ***teclado*** o cualquier tipo de ***dispositivo de*** entrada. Dentro del ***µP*** ideal reside el programa, el cual es un conjunto de instrucciones secuenciales que determinan cómo será procesado el dato de entrada y qué información será enviada a las líneas de salida como consecuencia del proceso de las entradas.

Las líneas de ***salida*** se pueden conectar a ***actuadores***, ***indicadores***, ***convertidores*** **D/A**, ***impresoras***, ***alarmas***, etc.

En cualquier tiempo, los ***niveles lógicos*** en las líneas de salida del ***microprocesador*** se determinan por **2** factores:

- La **historia completa** de las señales de entrada al ***µP***
- El **programa almacenado** en el ***µP***

Microprocesador real

Debido al número limitado de patas (pines) disponibles en cualquier circuito integrado práctico, el ***µP*** no contiene ***N*** líneas de entrada y ***M*** líneas de salida tendientes a **a**

Para la mayoría de los microprocesadores ***N*** **=** ***M***

Este número se conoce como el ***ancho*** de la trayectoria de datos o ***longitud*** de la palabra del ***µP***. Al grupo de líneas utilizado para transferir datos ***hacia/desde*** el ***µP*** se le conoce como ***bus*** de datos.

Bus de Datos (DB)

Los datos en el ***bus*** de datos se pueden representar en las siguientes notaciones:

a. **BINARIO**. Usaremos $NÚMERO_2$ o $NÚMERO_B$, subíndice ***2*** o ***B*** para indicar que el dato es ***binario***.

b. **OCTAL**. Agrupamos la información de derecha a izquierda de ***3*** en ***3***. Usaremos la letra ***O*** para ***octal***.

EJEMPLO: 0100011_2 **=** 01000111_B
$01,000,111_2$ **=** 107_O

c. **HEXADECIMAL**. Agrupamos la información de **4** en **4** de derecha a izquierda. Usaremos la letra ***H***:

EJEMPLO: 01000111_2 **=** 47_H

Una de las limitaciones prácticas más severas de los ***µP*** es el número de patas disponibles en un circuito integrado (*IC*, *por sus siglas en inglés*) económico. Por esto el ***bus*** de datos (*DB*, *por sus siglas en inglés*) es ***bidireccional*** (multiplexado), como se muestra en la ***Figura 1.7***:

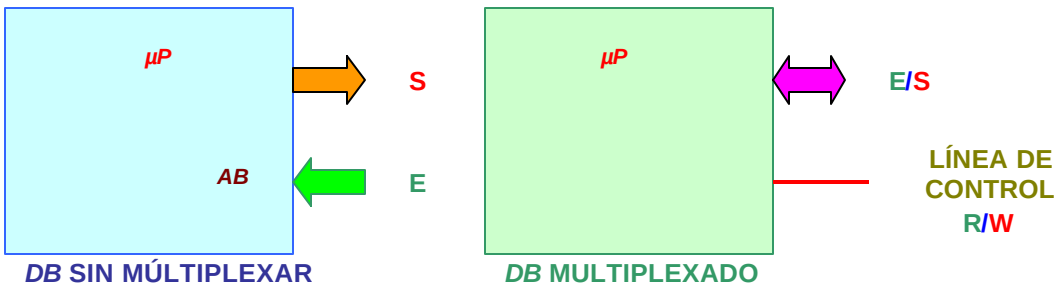


FIGURA 1.7. ***Bus*** de datos sin ***múltiplexar*** y ***multiplexado***

Bus de dirección

El ***microprocesador ideal*** contiene ***memoria*** interna ilimitada, los ***µP reales*** contienen ***memoria*** finita. Por lo tanto, el ***µP real*** contiene ***memoria*** externa, como se muestra en la ***Figura 1.8***:

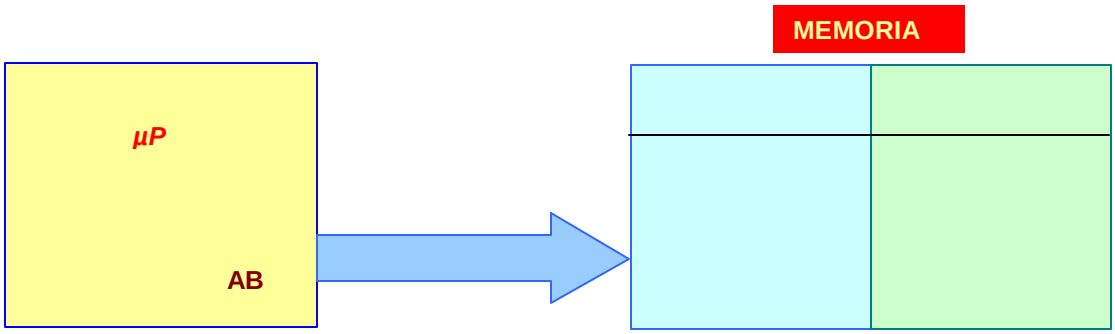


FIGURA 1.8. Microprocesador ***real*** con ***memoria*** externa

El ***µP*** debe ser capaz de almacenar o recuperar información de esta ***memoria***. Al proceso de **almacenar** información en ***memoria*** se conoce como ***escritura*** a memoria. Al proceso de **recuperar** información de la ***memoria*** se le conoce como ***lectura*** de memoria. El ***µP real*** contiene un grupo de líneas, ***b***us de ***d***irección, ***AB*** para acceder a las localidades de ***memoria***.

El conjunto de localidades de ***memoria*** que un ***µP*** puede acceder directamente se le conoce como ***espacio*** de memoria y se expresa en ***Kpalabras***. **1** ***Kpalabras*** **= 2^{10}** palabras **= 1024** palabras.

Recalcando que una de las limitantes de los encapsulados económicos es el número de patas, en algunos microprocesadores los buses de ***dirección*** y de ***datos*** están ***multiplexados***. Por ejemplo, los INTEL **8086** y **8088**.

Bus de control

El microprocesador ***real*** contiene un conjunto de líneas que sirven para controlar la circuitería externa del ***µP***. Al conjunto de estas líneas se les conoce como bus de ***control***.

Bus de alimentación

Sirve para proporcionar el ***voltaje*** de referencia de la ***lógica*** binaria del ***µP***. Los valores más comunes son:

- **GND = 0V**
- **V_{CC} = 5 V**

Registros internos

Los ***registros*** internos del microprocesador ***real*** se utilizan para almacenamiento temporal de ***datos*** e ***instrucciones***. Los más comunes son:

- ***MDR*** ***R***egistro de ***D***atos de ***M***emoria
- ***MAR*** ***R***egistro ***D***irección de ***M***emoria
- **Acumuladores**
- **Registros de índice**
- **Apuntadores de la pila de datos**
- **Apuntador de segmentos**
- ***IR*** ***R***egistro de ***I***nstrucción
- ***FR*** ***R***egistro de estado de las ***B***anderas de la ***ALU***

Memorias

Existen dos tipos de organización:

- **FÍSICA** — **Tal como aparece en el Bus de dirección** — **Existen dos métodos:**
  - **Lineal**
  - **Paginación**
- **LÓGICA** — **Como la "ve" el programador de *ensamblador*** — **Existen dos métodos:**
  - **Segmentación**
  - **Paginación lógica**

La ***distinción*** es importante en aplicaciones ***multiusuarios***

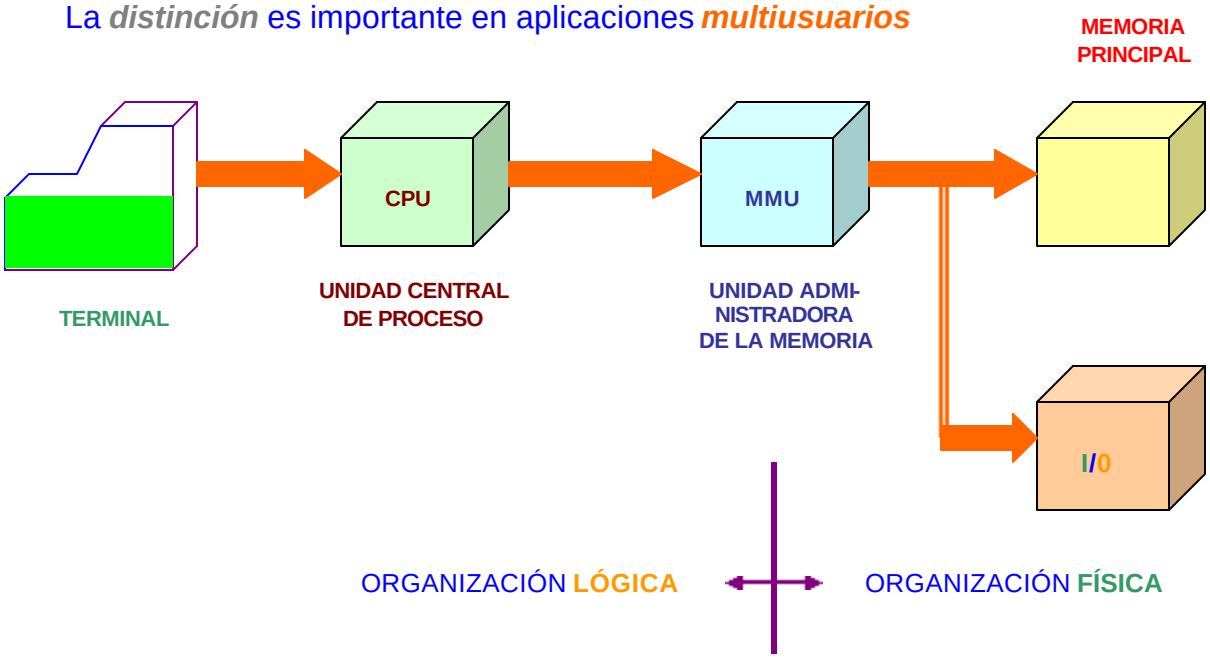
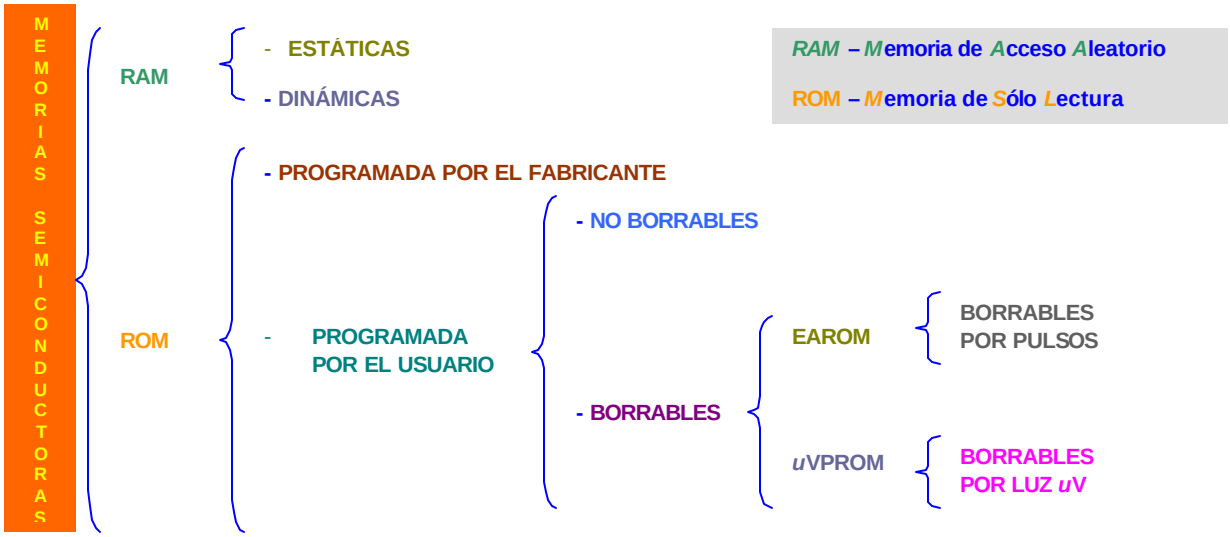


FIGURA 1.9. Organización ***Física*** y ***Lógica*** de la ***memoria***.

Los tipos de ***memoria*** semiconductora disponibles actualmente para los ***µC*** se resumen a continuación:



En *µC* se pueden utilizar 2 tipos de memoria:

- **Volátil**
- **No Volátil**

La memoria ***volátil*** tiene la característica de que pierde la información almacenada cuando se desactiva la fuente de poder (la celda de almacenamiento es un multivibrador ***biestable***)

La memoria ***no volátil*** retiene la información después de que se desactiva la fuente de poder (La celda de almacenamiento es un dispositivo de conmutación con un fusible como elemento programable)

Memorias de Acceso Aleatorio (***RAM****, por sus siglas en inglés*)

Las ***RAM*** se clasifican como memorias ***volátiles***. Hay **2** tipos de ***RAM***, ***estáticas*** y ***dinámicas***. En las ***RAM dinámicas*** la información se almacena en forma de carga eléctrica (capacitancia base-emisor de transistor **MOS**) Esta capacidad tiende a descargarse con el tiempo, por lo que es necesario tener circuitos para ***refrescar*** la información periódicamente.

La ***RAM estática*** no requiere circuitos de ***refresco*** ya que la información se almacena en un multivibrador tipo ***latch***.

Aunque la memoria ***dinámica*** requiere circuitería de ***refresco*** externa, es más barata que la ***RAM estática*** y consume menos energía.

Memorias de Sólo Lectura (***ROM****, sus siglas en inglés*)

Se clasifican en dos tipos:

- Las programadas por el ***fabricante***
- Las programadas por el ***usuario***

Las ***ROM*** programadas durante el proceso de manufactura son muy confiables y cuando se producen en volúmenes altos proporcionan el costo más bajo por bit que cualquier memoria semiconductora.

En las ***RAM***, el tipo de acceso es independiente de la localización del dato, en este tipo de memoria, un ***arreglo*** de biestables contiene la información binaria (las celdas se conectan en paralelo a la línea de salida del ***decodificador*** de dirección) Se conectan tantas celdas en paralelo como bits tiene la palabra.

Las ***RAM*** estáticas tienen los siguientes inconvenientes:

- **Densidad** limitada
- Alto **consumo** de energía
- Conforme se incrementa la **densidad** de la memoria, los circuitos **sensores** se vuelven más complejos y requieren mayor energía**.**

Las ***RAM*** dinámicas son de tecnología **MOS** y tienen la ventaja de que las celdas son de dimensiones menores a las bipolares y consumen poca energía, lo cual hace posible el que se puedan empaquetar muchas celdas en un circuito integrado.

La desventaja de las ***RAM*** dinámicas con respecto a las ***estáticas*** es la ***velocidad*** de acceso.

Las ***ROM*** programables por el usuario pueden ser de **2** tipos:

1. ***ROM Programable*** tipo **fusible** (***PROM****, por sus siglas en inglés*)

   Estas se pueden programar una sola vez, después de programadas, el patrón programado no puede ser alterado.

2. ***PROM borrables*** (***EPROM***, *por sus siglas en inglés*)

   Pueden ser ***programadas*** y ***reprogramadas*** por el usuario muchas veces siempre y cuando se respete la especificación de programación en cuanto a la amplitud y tiempo de duración del pulso de programación. Hay dos tipos de ***EPROM***:

   a. ***uVEPROM*** la cual es **borrada** cuando se hace incidir un haz de luz ultravioleta de suficiente intensidad.
   b. ***EAPROM*** (***electrically alterable PROM***) se borra aplicando un ***pulso*** en una patilla del IC. Este es mejor que las del inciso anterior, ya que se borra el contenido de una localidad. En las del inciso anterior se borra toda la memoria.

Organización interna de las memorias.

Existen dos tipos de arreglos:

- **Lineal**
- **Matricial**

Arreglo **lineal**. La ***Figura 1.10***, muestra los bloques básicos de un arreglo ***lineal***:

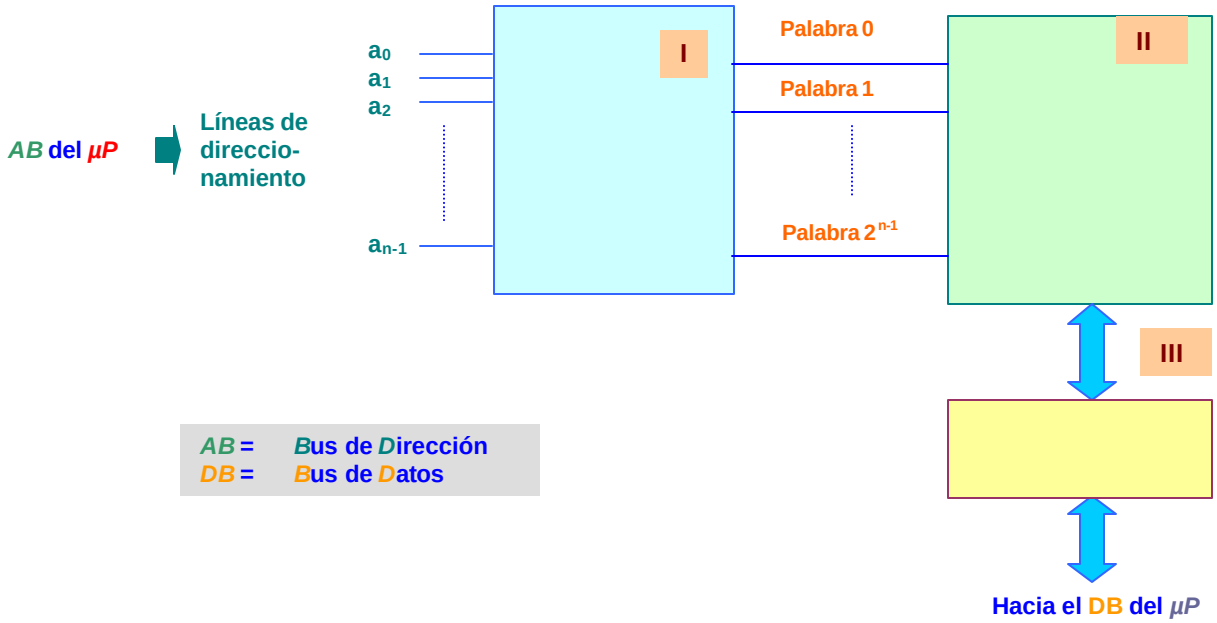


FIGURA 1.10. Diagrama a bloques de un arreglo ***lineal***

A continuación se describen los bloques básicos del arreglo ***lineal***:

I. Decodificador de dirección

Es un circuito que contiene 2^n líneas de **salida** y ***n*** líneas de ***entrada***. Para cada combinación entre las ***n*** líneas de ***entrada***, se selecciona una y sólo una línea de ***salida***.

EJEMPLO: Veamos el desarrollo de una memoria cuyo arreglo es ***lineal*** y contiene ***4*** palabras de ***4*** bits por palabra. Este ejemplo es didáctico.

Si tenemos ***2*** líneas de ***direccionamiento***, tendremos 2^2 líneas de ***salida***, como se muestra en la siguiente tabla:

DIRECCIÓN		CONTENIDO			
a_1	a_0	w_3	w_2	w_1	w_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

De la tabla se obtienen las siguientes ecuaciones lógicas:

$$w_0 = \bar{a}_1 \bar{a}_0$$
$$w_1 = \bar{a}_1 a_0$$
$$w_2 = a_1 \bar{a}_0$$
$$w_3 = a_1 a_0$$

En diagrama a bloques se tiene la siguiente representación:

El diagrama lógico que genera la tabla anterior se muestra en la ***Figura 1.11***:

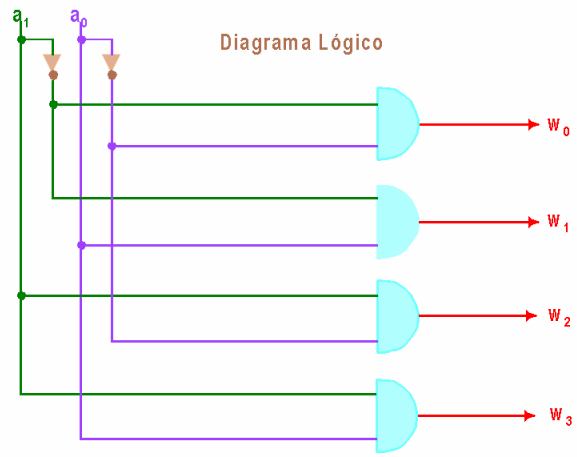


FIGURA 1.11. Diagrama lógico de un arreglo ***lineal*** de ***4*** palabras de ***4*** bits por palabra

En general, los ***decodificadores*** que contienen los circuitos integrado de ***memorias*** son:

n entradas	2^n salidas
10	1024
11	2048
12	4096
13	8K
14	16K
15	32K
16	64K

II. Celdas de almacenamiento

Están organizadas en ***palabras***, cada palabra corresponde a una línea de ***salida*** del ***decodificador*** de dirección.

El número de ***bits*** que contiene cada palabra es el número de celdas que se conecta en paralelo a cada línea de ***salida*** del ***decodificador***, como se muestra en ***Figura 1.11***.

III. Sensores de *LECTURA* / *ESCRITURA*

Debido a las características del ***decodificador*** "**una sola línea de salida w_m está activa con una combinación entre las *n* líneas de entrada**", la cual nos permite compartir los ***sensores*** de ***entrada*** / ***salida***.

Un ***sensor*** por cada ***bit*** de que conste la ***palabra***.

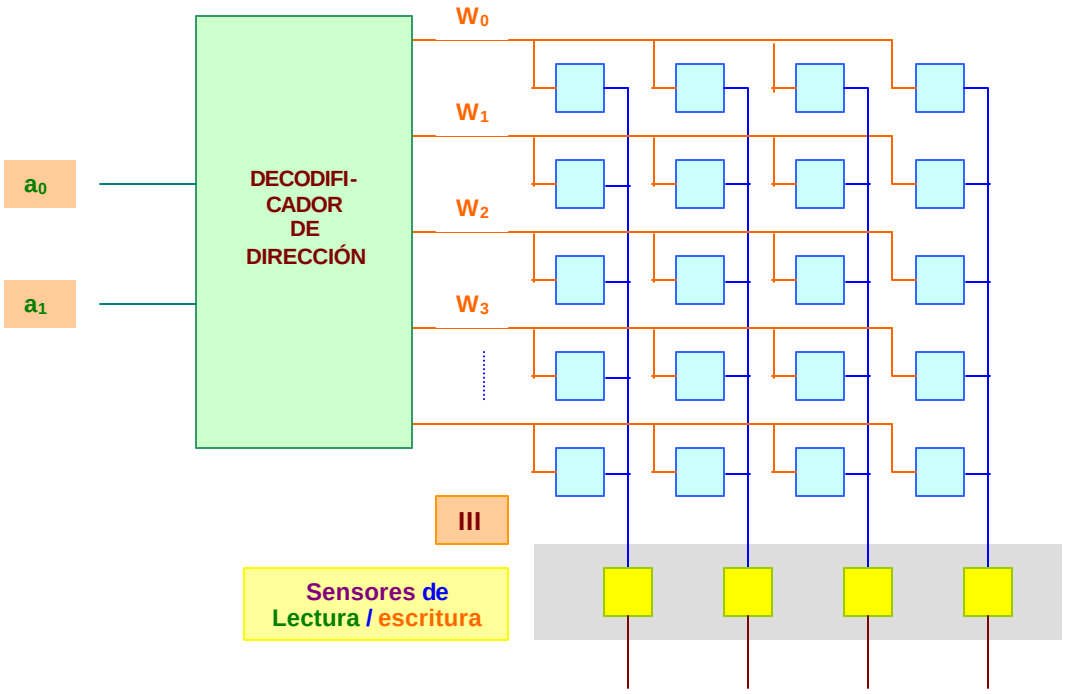


FIGURA 1.12. **Sensores de *lectura/escritura***

Arreglo matricial. La ***Figura 1.13***, muestra el ***arreglo matricial*** a bloques.

Organización:

a. Las líneas de **direccionamiento** se dividen en dos grupos. Unas líneas entran al **decodificador** en ***x*** y otras en ***y***.
b. Las líneas de **salida** de los **decodificadores** activan a una sola celda del bloque de celdas de **almacenamiento**.
c. Las celdas de **almacenamiento** deben contener dos líneas de entrada de selección, una en ***x*** y la otra en ***y***.
d. Ya que las combinaciones entre las líneas de ***direccionamiento*** son únicas, requerimos un solo **sensor** de ***lectura/escritura*** para todo el arreglo.
e. Un **parámetro** útil en todo tipo de memoria es:

Densidad de la memoria = *número de palabras* por *número de bits* de la palabra

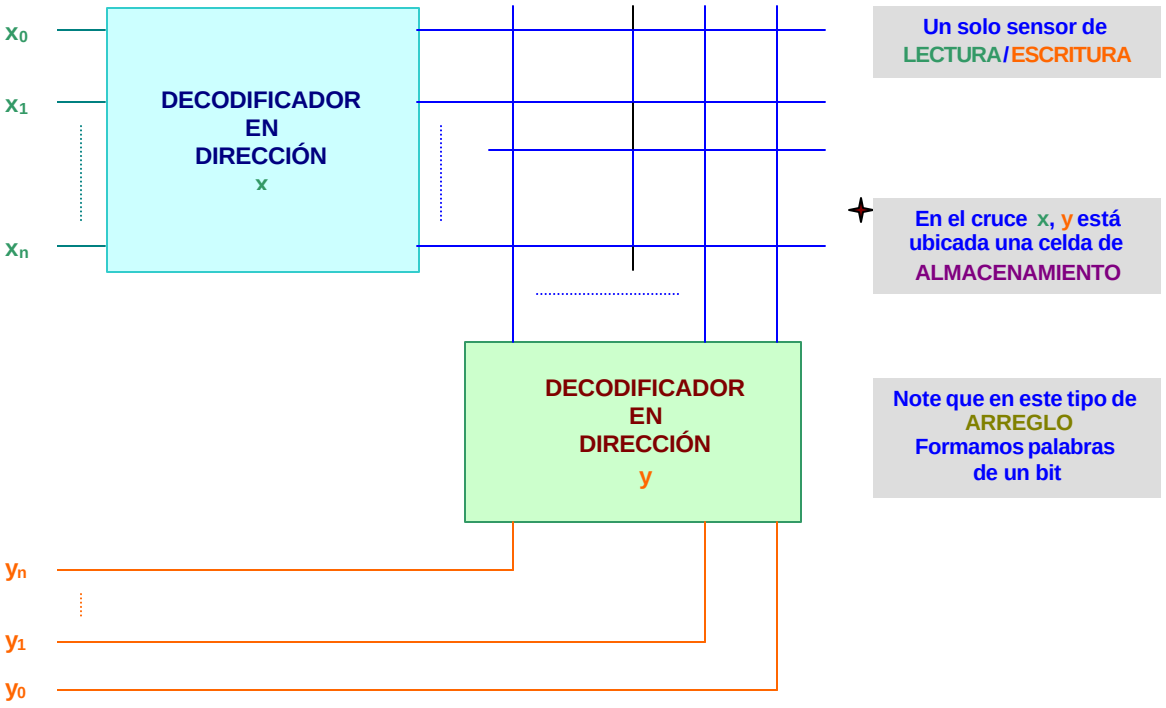


FIGURA 1.13. **Diagrama a bloques de un *arreglo matricial*.**

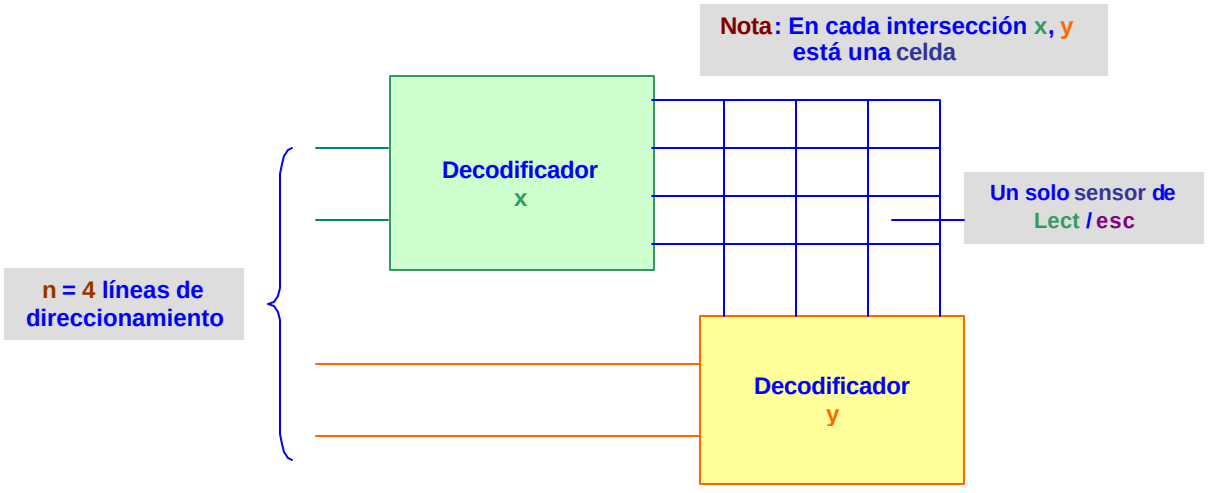
EJEMPLO: Obtener el ***diagrama lógico*** de una memoria de ***16 palabras*** de ***1 bit*** cada palabra.

Densidad de memoria = 16 x 1 = 16 líneas de direccionamiento

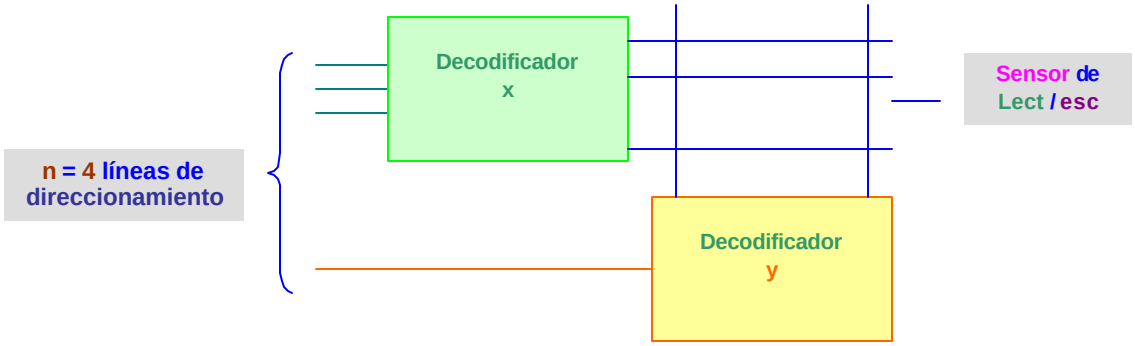
2^n = número de palabras = 16 Þ n = 4 líneas de direccionamiento

Podemos obtener los siguientes **arreglos**:

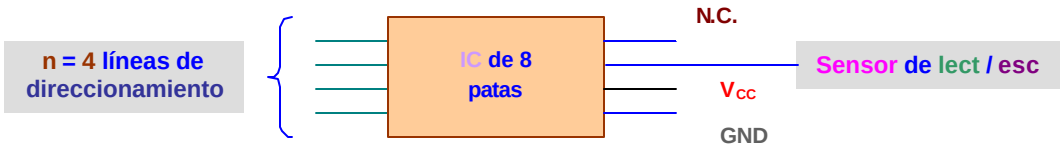
a.



b.



El circuito integrado que contiene a esta ***memoria***, comprende las siguientes patas:

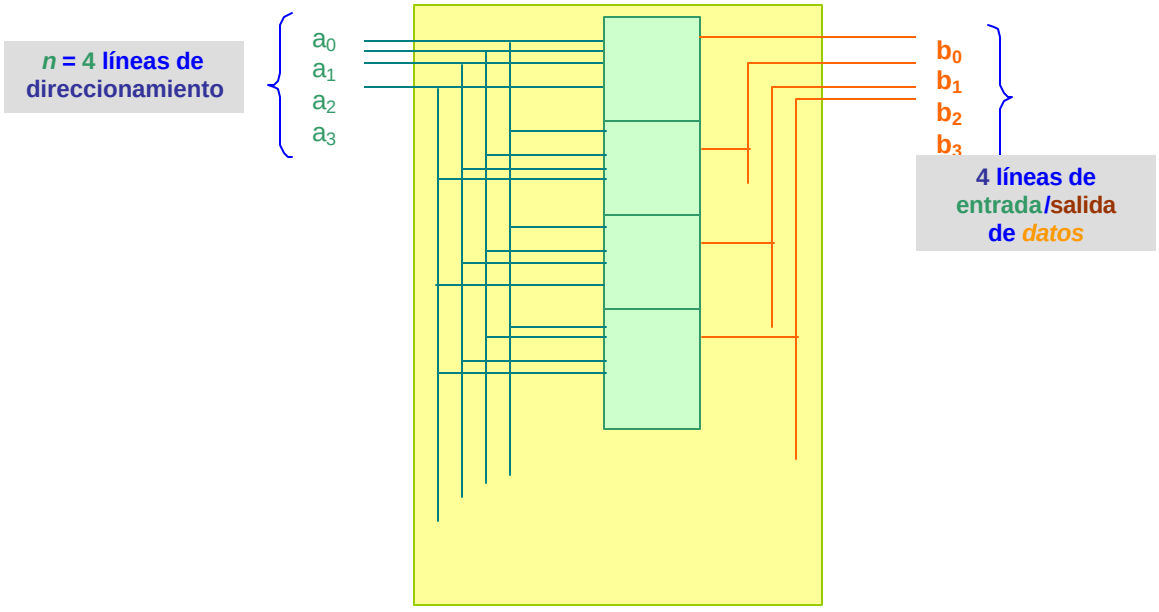


Algoritmos para expansión de memorias

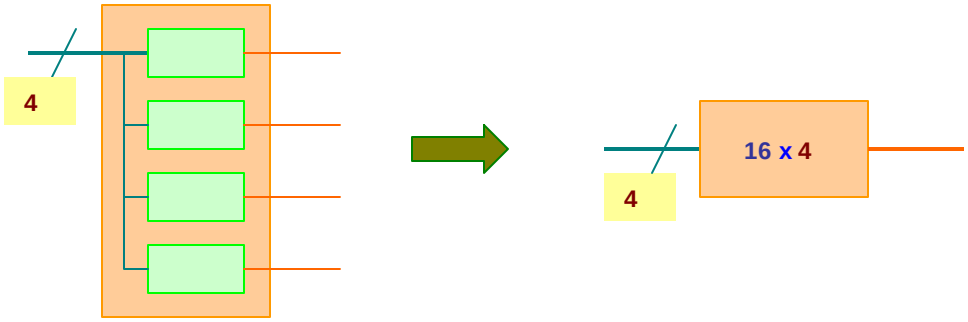
a. Expansión del número de bits de la palabra

Cuando se desea mantener fijo el ***número de direcciones*** y aumentar el ***número de bits de la palabra***, se determina el ***número de circuitos integrados*** que requiere el arreglo. Se **conectan** en paralelo las ***líneas de direccionamiento*** de los integrados y se dejan libres las líneas de ***lectura/escritura*** asignándoles un peso.

EJEMPLO: A partir del ejemplo anterior, obtener un **arreglo de memoria** de ***16 palabras*** de ***4 bits*** cada palabra.



Otras formas más ***simplificadas*** y ***equivalentes*** del diagrama anterior son:

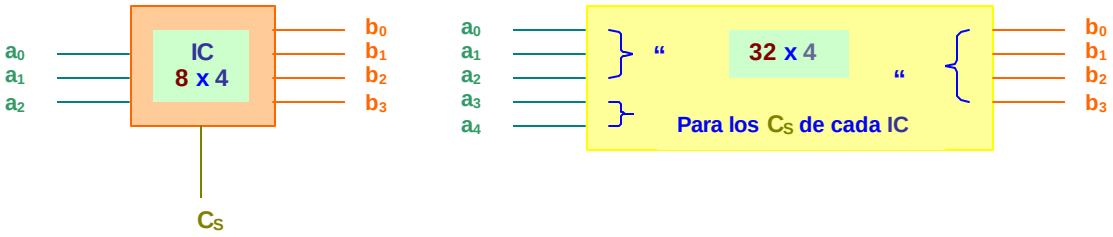


b. Expansión del número de localidades de memoria

Cuando se desea aumentar el **número de localidades de memoria** manteniendo constante el **número de bits de palabra**:

- **Determinamos** el ***número de integrados*** que requiere el arreglo.
- **Conectar** en paralelo las líneas de ***entrada/salida*** (***I/O***, *por sus siglas en inglés*) de ***datos*** (cada una con la del mismo peso)
- **Conectar** en paralelo todas las ***líneas de dirección*** (cada una con la del mismo peso)
- Las líneas que sobran (las más significativas) **conectarlas** a un **decodificador** de ***IC***.
- Las salidas del **decodificador** **conectarlas** a las líneas ***CS*** de cada circuito integrado.

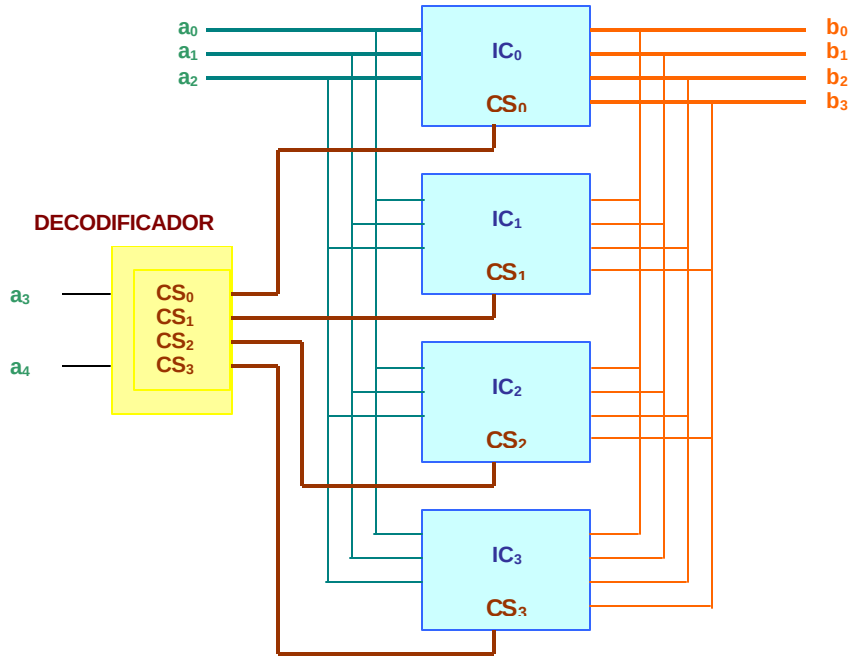
EJEMPLO: A partir de circuitos integrados de ***8 palabras*** de ***4 bits***, obtener un arreglo de ***32 palabras*** de ***4 bits***.



decimal	DIRECCIÓN					CONTENIDO			
	a_4	a_3	a_2	a_1	a_0	b_3	b_2	b_1	b_0
0	0	0	0	0	0				
1	0	0	0	0	1				
2	0	0	0	1	0				
3	0	0	0	1	1				
4	0	0	1	0	0				
5	0	0	1	0	1				
6	0	0	1	1	0				
7	0	0	1	1	1				
8	0	1	0	0	0				
9	0	1	0	0	1				
10	0	1	0	1	0				
11	0	1	0	1	1				
12	0	1	1	0	0				
13	0	1	1	0	1				
14	0	1	1	1	0				
15	0	1	1	1	1				

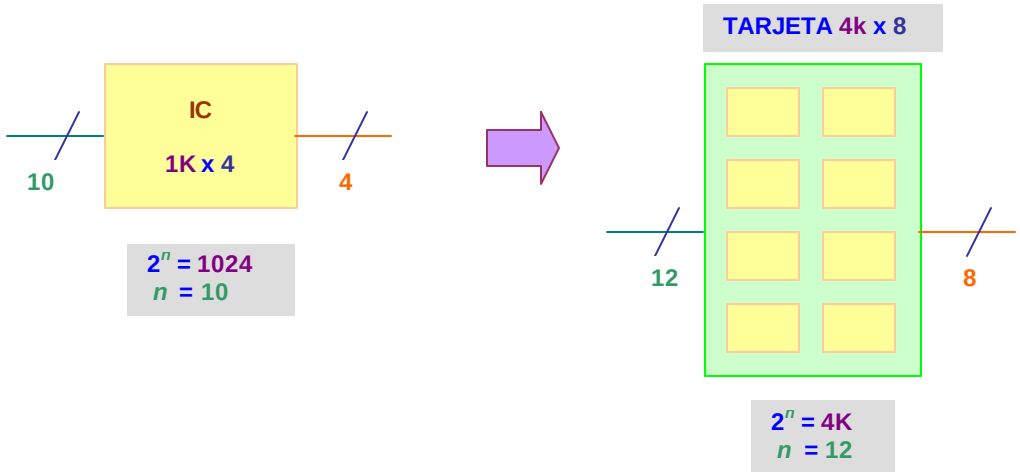
IC_0 (decimal 0–7) IC_1 (decimal 8–15)

decimal	DIRECCIÓN					CONTENIDO			
	a_4	a_3	a_2	a_1	a_0	b_3	b_2	b_1	b_0
16	1	0	0	0	0				
17	1	0	0	0	1				
18	1	0	0	1	0				
19	1	0	0	1	1				
20	1	0	1	0	0				
21	1	0	1	0	1				
22	1	0	1	1	0				
23	1	0	1	1	1				
24	1	1	0	0	0				
25	1	1	0	0	1				
26	1	1	0	1	0				
27	1	1	0	1	1				
28	1	1	1	0	0				
29	1	1	1	0	1				
30	1	1	1	1	0				
31	1	1	1	1	1				



Aplicación de los dos algoritmos

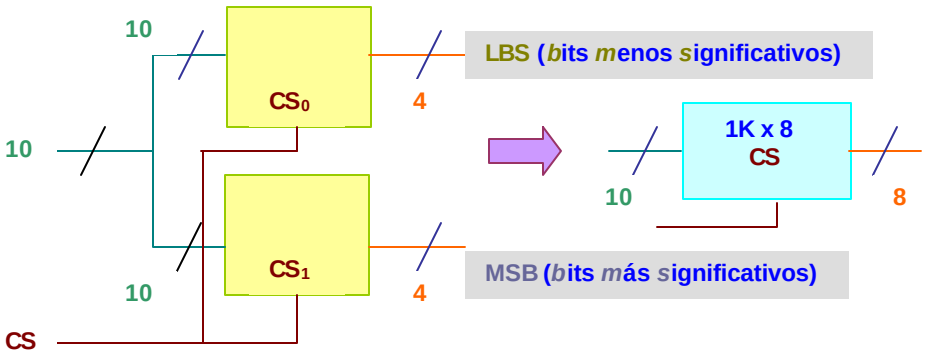
EJEMPLO: A partir de un circuito integrado de **1K x 4**, obtener un módulo de memoria de **4K x 8**.



$$\text{Número de circuitos integrados} = \frac{\text{densidad del módulo}}{\text{densidad de un IC}} = \frac{\text{4K x 8}}{\text{1K x 1}} = 8$$

Para interconectar los circuitos integrados (*IC*) aplicamos los ***2 algoritmos de expansión de memoria***.

a. Para el algoritmo de ***expansión de longitud*** de palabra:



b. Aplicando el algoritmo de ***expansión de localidades*** de memoria:

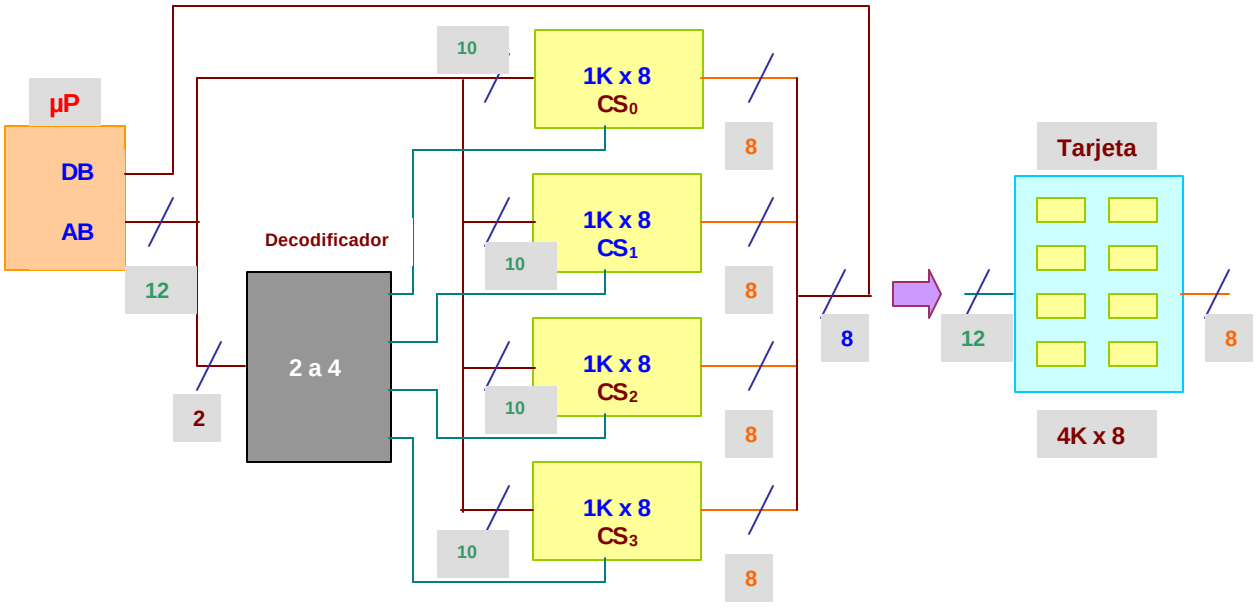
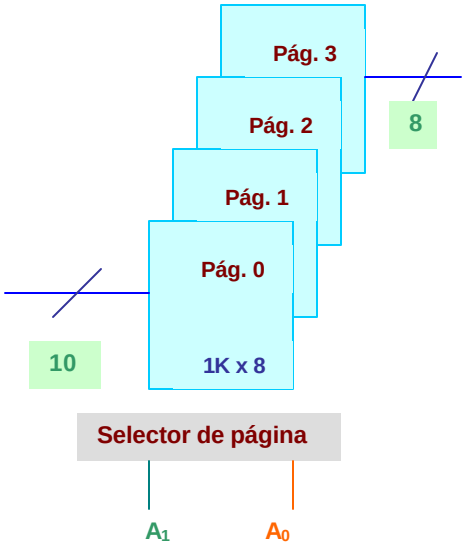


Tabla de memoria:

DIRECCIÓN					CONTENIDO	
A_{11}	A_{10}	A_9	¼	A_0	N_2	N_1
0	0	0 : 1	¼ : ¼	0 : 1	IC_1	IC_0
0	1	0 : 1	¼ : ¼	0 : 1	IC_3	IC_2
1	0	0 : 1	¼ : ¼	0 : 1	$1C_5$	IC_4
1	1	0 : 1	¼ : ¼	0 : 1	$1C_7$	IC_6

Un concepto didáctico es representar a la memoria como si fuese un cuaderno compuesto por páginas. La página se asigna de alguna manera lógica correspondiente al arreglo de *hardware*, como se muestra en la siguiente figura:



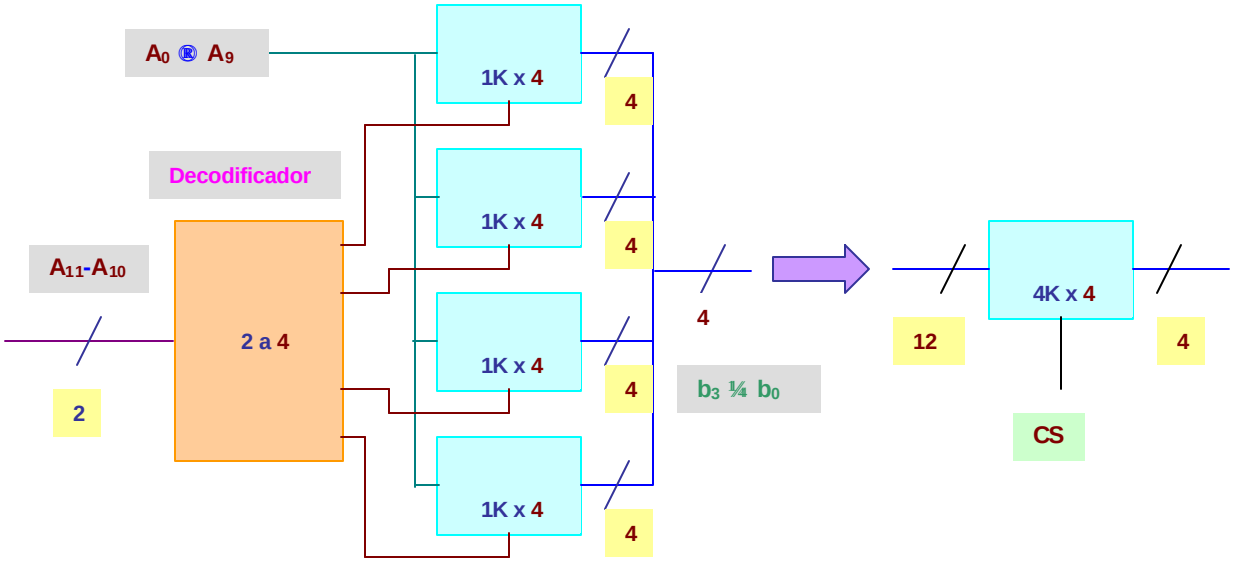
El cuaderno contiene 4 páginas de 1K x 8

Otra manera de representación es por medio de una tabla:

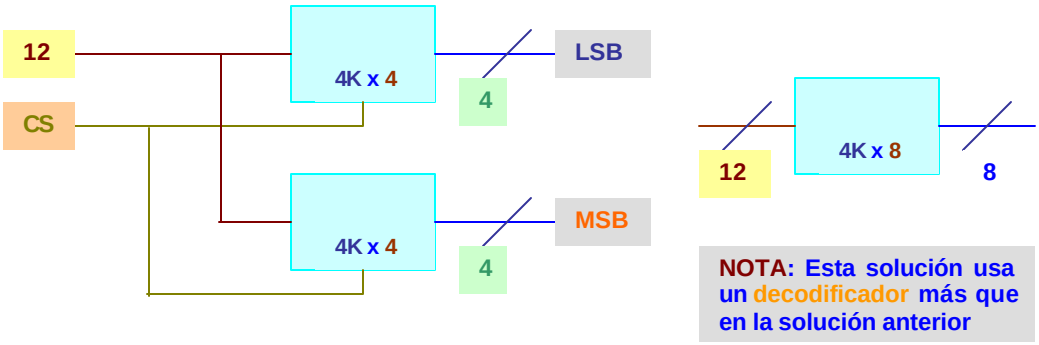
	b_7						b_0
0							
1							
2							
3							
:							
1023							
	N_1						N_0

La ***hoja*** es un espacio cuadriculado, los ***renglones*** son las ***direcciones*** y las ***columnas*** son los **bits de las palabras**. En cada hoja están representados **2** circuitos integrados de **1K x 4**.

Para tener una segunda solución, aplicamos el **algoritmo de expansión de localidades** de memoria:



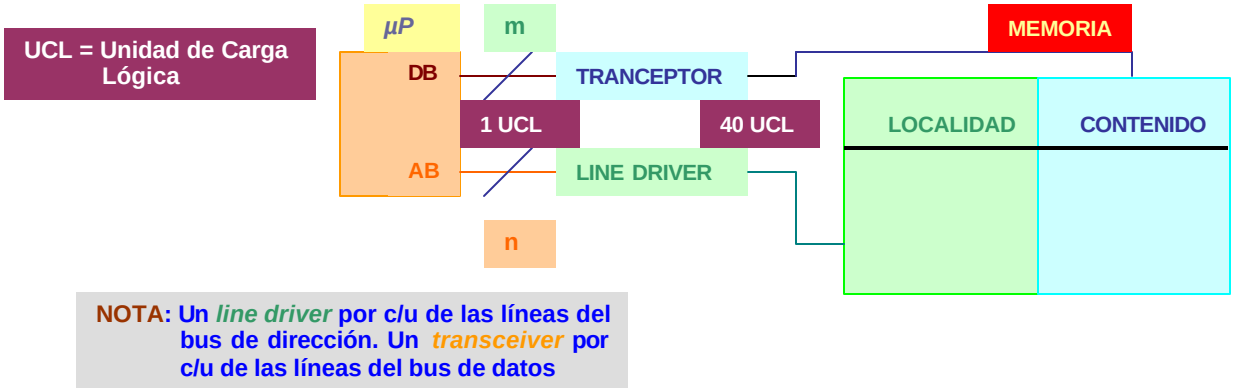
Después aplicamos el algoritmo de **expansión de longitud de la palabra**:



EJEMPLOS de interconexión ***µP – Memoria***.

Para propósito de interconexión ***µP*** – Memoria, será necesario el concepto de amortiguar el bus de dirección con circuitos ***line drivers*** (***excitadores de línea***) y el bus de datos con elementos bidireccionales tranceptores (***transceivers***), con el fin de no sobrecargar las líneas de dirección y de datos del ***µP***.

Esto, cuando se conecta mucha memoria a sistemas de propósito general

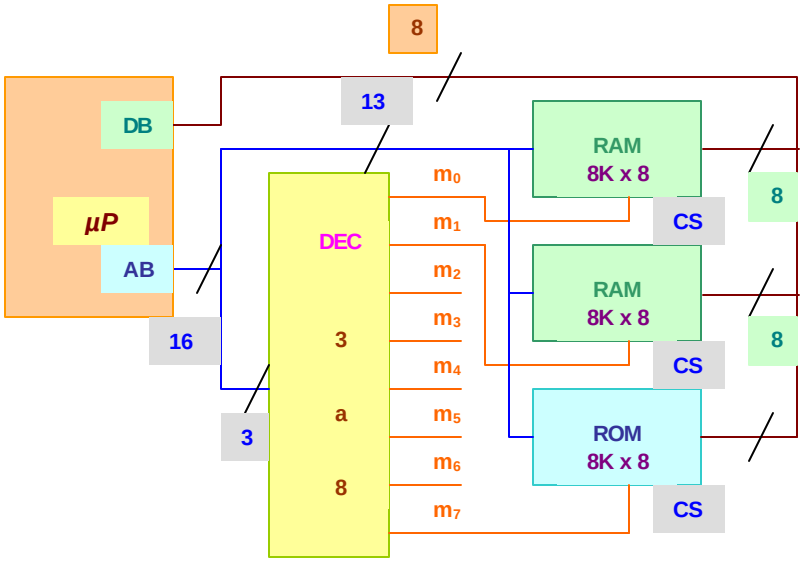


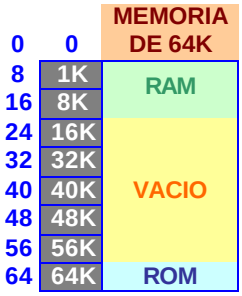
EJEMPLO: **Interconexión *mP*-memoria**. Se tiene un ***µP*** con **16** líneas en el ***bus de dirección*** (***AB***, *por sus siglas en inglés*) y **8** líneas en el ***bus de datos*** (***DB***, *por sus siglas en inglés*) Tenemos **2** módulos de ***memoria de acceso aleatorio*** (***RAM***, por sus siglas en inglés) de **8K x 8** y un módulo de ***memoria de sólo lectura*** (***ROM***, por sus siglas en inglés) de **8K x 8**. Ubicar los módulos de ***RAM*** en los primeros **16K** de direccionamiento del ***µP***. Ubicar el módulo de **ROM** en los últimos **8K** de direccionamiento del ***µP***.

La ***Figura 1.14***, muestra la solución.

EJEMPLO: **Interconexión *mP*-memoria**. Se tiene un ***µP*** con **20** líneas en el ***AB*** y **16** líneas en el ***DB***. Tenemos 2 módulos de memoria ***RAM*** de **8K x 16** y un módulo de memoria ***ROM*** de **8K x 16**. Ubicar los módulos de **RAM** en las direcciones **32K** a **48K** del ***µP*** y ubicar el módulo ***ROM*** en los penúltimos **8K** de direccionamiento del ***µP***.

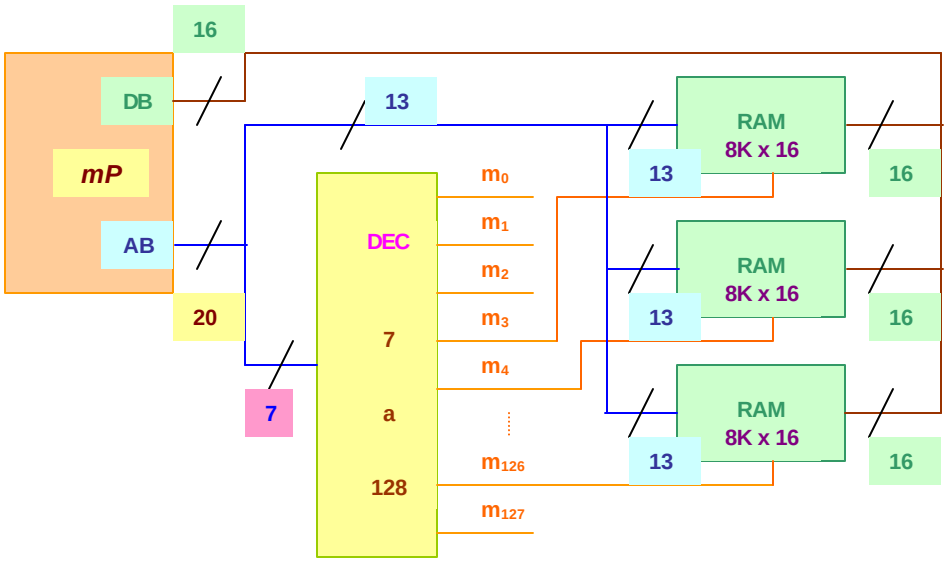
La ***Figura 1.15***, presenta la solución.





Mapa de memoria

FIGURA 1.14. Interconexión *mP*-memoria



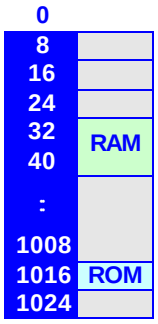
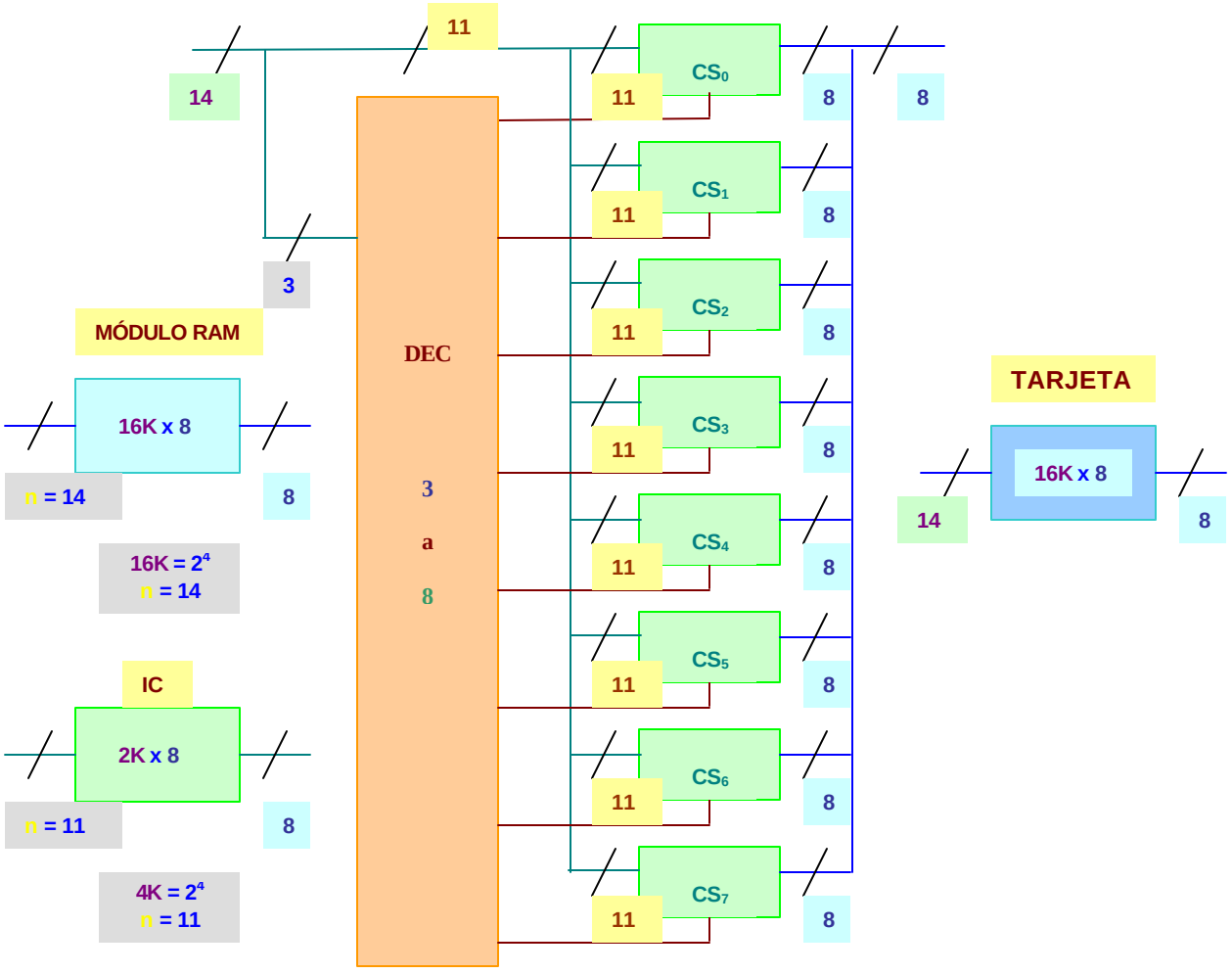


FIGURA 1.15. Interconexión *mP*-memoria

EJEMPLO: Tenemos el ***µP*** **8085** con **16** líneas en el ***bus de dirección*** (***AB***) y **8** líneas en el ***bus de datos*** (***DB***) Tenemos circuitos integrados de memoria ***RAM*** de **2K x 8** y ***PROMS*** de 1K **x 8**. Configurar en ***RAM*** un módulo de 16K **x 8** y en ***ROM*** un módulo de 16K **x 8**. Ubicarlos, la ***RAM*** en los primeros 16K y el ***ROM*** en los últimos 16K de direccionamiento del ***µP***.

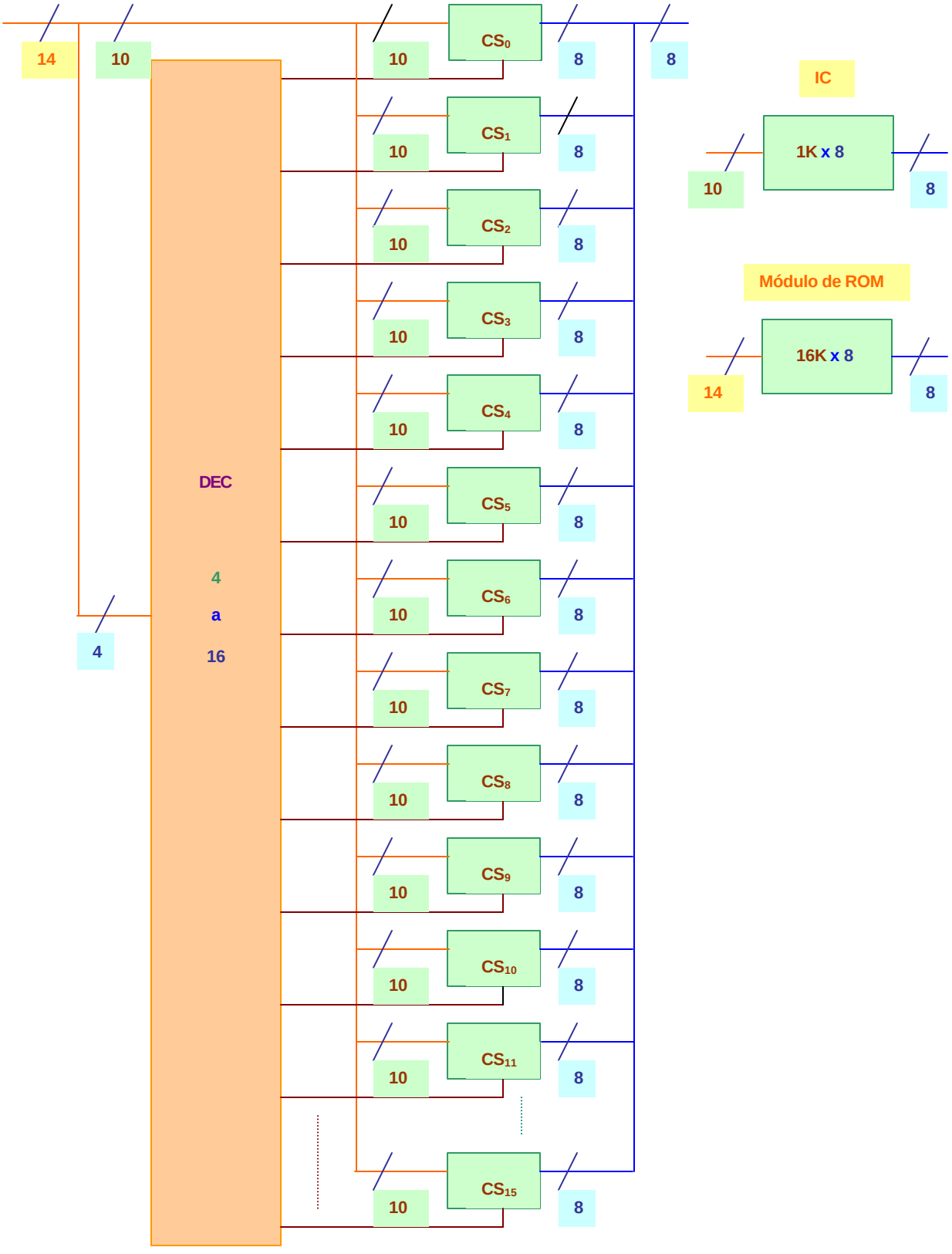
1. Módulo de *RAM*.

$$\text{Número de IC} = \frac{\text{densidad del módulo}}{\text{densidad del IC}} = \frac{16K \times 8}{2K \times 8} = 8$$



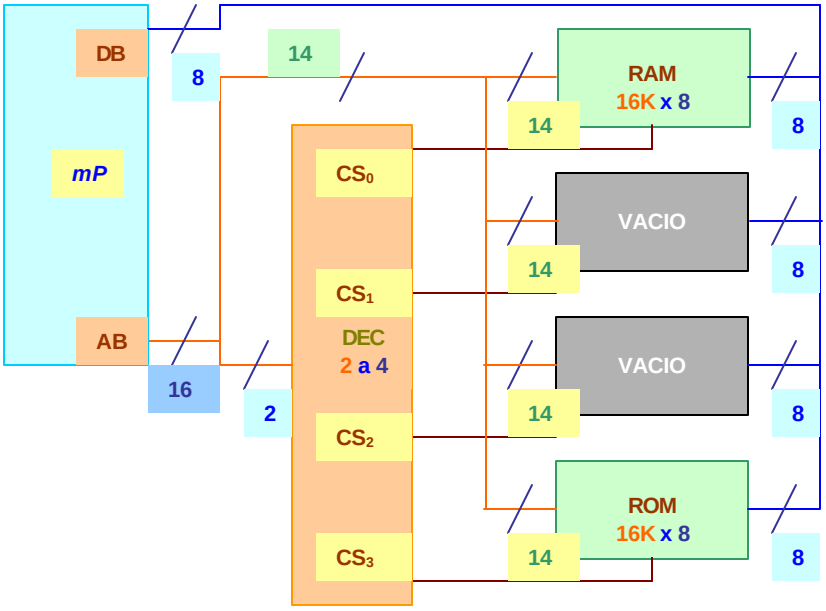
2. Módulo de *ROM*.

$$\text{Número de IC} = \frac{\text{densidad del módulo}}{\text{densidad del IC}} = \frac{16K \times 8}{1K \times 8} = 16$$



3. Interconectar el *µP* con módulos.

$$\text{Número de módulos} = \frac{\text{espacio de direccionamiento del } mP}{\text{densidad del módulo}} = \frac{64K \times 8}{16K \times 8} = 4$$



NOTA: En general cada *IC* contiene más de una línea de *CS*

Los *µP* 8086/8088: **Intel** diseñó los ***µP* 8086/8088** para realizar al mismo tiempo las principales funciones internas de transferencia de datos y búsqueda de instrucciones. Para lograrlo ambos microprocesadores constan de dos procesadores interconectados en el mismo *IC*. Una unidad está encargada de buscar instrucciones y la otra de ejecutarlas. Además, la unidad encargada de buscar instrucciones utiliza un método llamado de ***estructura tubular*** (***pipeline*** <por cola>) en la cual entran las instrucciones.

Al procesador principal de se llama unidad de ejecución, el cual está encargado de decodificar y ejecutar todas las instrucciones. Esta unidad es idéntica en ambos microprocesadores.

A la otra unidad se le llama ***unidad de interfaz de bus*** (***BIU***, *por sus siglas en inglés*), la cual está encargada de localizar las instrucciones y de transferir todos los datos entre los registros internos y el mundo exterior.

Dispositivos de Entrada/Salida (E/S)

La *Figura 1.16*, muestra la *arquitectura* de *3 buses* de un sistema de una *microcomputadora*:

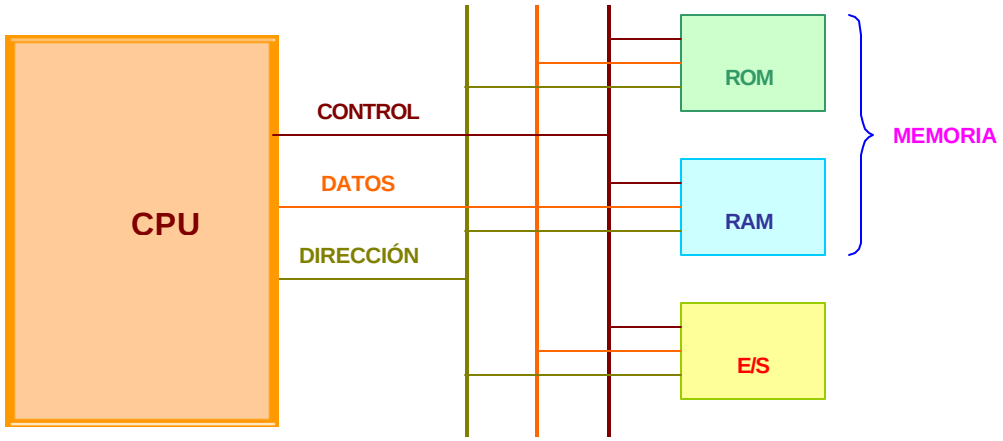


FIGURA 1.16. Arquitectura **de** *3 buses* **de un sistema** *µC*.

Se tienen dos técnicas para la *Entrada/Salida*:

- *E/S mapeada en memoria*
- *E/S mapeada en E/S*

Algunos *microprocesadores* asignan parte del espacio disponible en memoria para la *entrada/salida*. Los *µP* que usan parte del espacio de memoria para *E/S* se dice que usan *E/S mapeada en memoria*. Los *µP* **6800**, **6802**, **6805**, **6809** y el **68000** usan esta técnica.

El **8086/8088** no utiliza esta técnica. Todo el espacio de memoria del sistema puede ser utilizado para memoria. El sistema de *E/S* tiene su propio espacio de direccionamiento. A una arquitectura de *E/S* como esta se le conoce como *E/S mapeada en E/S*.

Una operación de *E/S* puede definirse de la siguiente manera:

INPUT Cuando el *µP lee datos* desde una fuente que no es la memoria del sistema

OUTPUT Cuando el *µP* **escribe datos** a un destino que no es la memoria del sistema

El bus de control define el tipo de comunicación que ocurre. En otras palabras, si el sistema utiliza *E/S mapeada en E/S*, hay líneas de control separadas para la *E/S* y la memoria del sistema.

Por ejemplo, el sistema de memoria utiliza las líneas de control etiquetadas como *memory read* y *memory write* mientras que el sistema de *E/S* usa las líneas de control etiquetadas como *input read* y *output write*, como lo muestra la *Figura 1.17*.

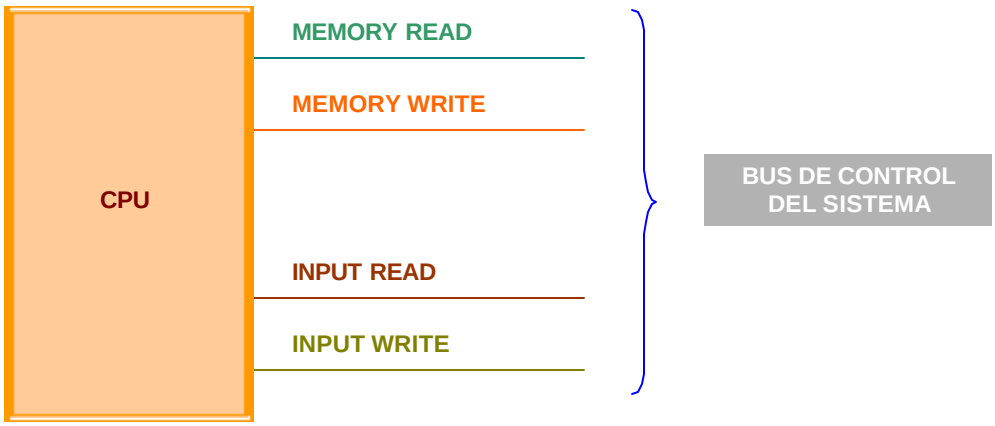


***FIGURA 1.17*. Señales de control del sistema para la *memoria* y la *E/S*.**

Direccionamiento de *E/S*.

Puerto: Es un ***lugar único*** utilizado para ***leer*** o ***escribir*** ***datos***, no es la memoria del sistema.

De hecho, un ***puerto*** es similar a una localidad única en el sistema de memoria desde el cual el ***dato*** debe ser ***leído*** o ***escrito*** durante una operación de memoria.

A cada ***puerto*** en el sistema de ***E/S*** se le da una dirección única, llamada ***código de selección del puerto***.

Para generar un ***código de selección del puerto***, las ***líneas de dirección*** A_0 – A_{15} son decodificadas con circuitería lógica para que respondan a una combinación específica.

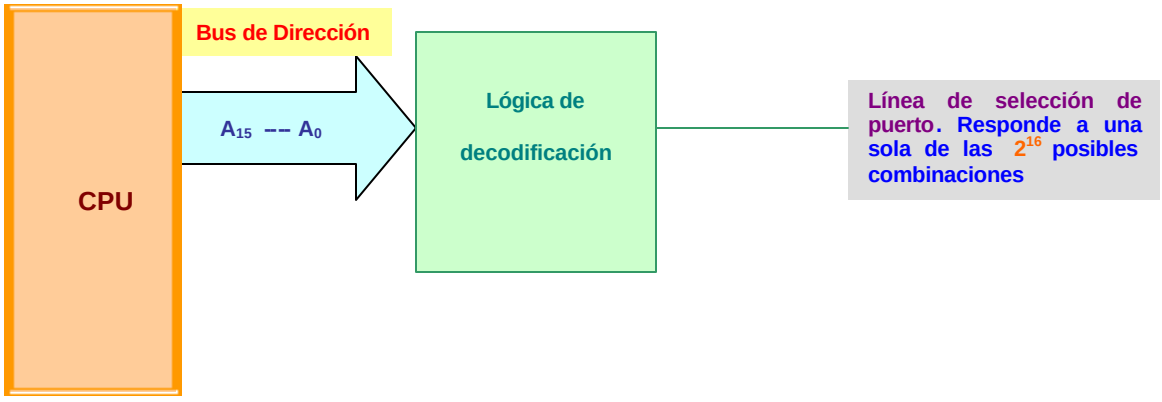


FIGURA 1.18. Cada **puerto** de ***E/S*** está diseñado para responder a una sola combinación de las 2^{16} posibles, del ***bus de dirección*** del sistema. Recuerde para ***E/S***, se toman **16** líneas de las **20** que tiene el ***bus de dirección*** (***AB***)

Por ejemplo, un puerto puede tener un ***código de selección de puerto*** igual a **0507H**, como se muestra en la ***Figura 1.19***:

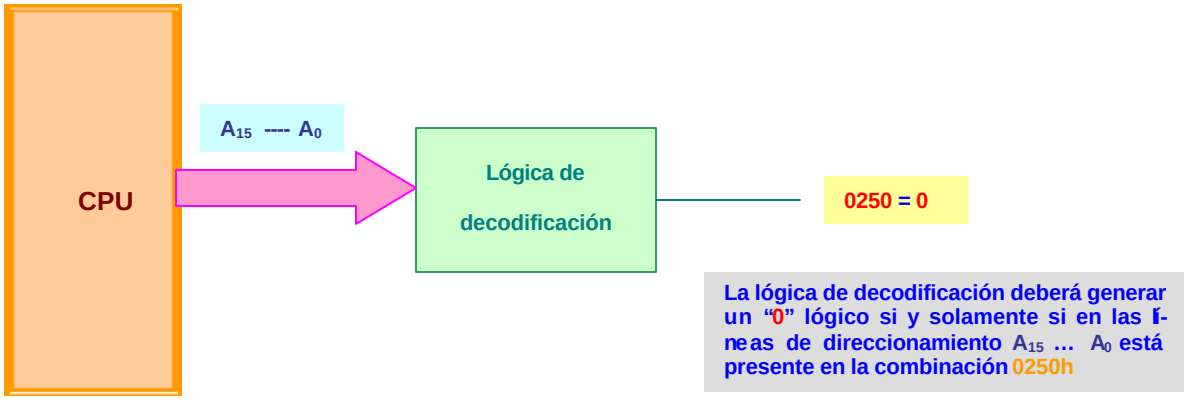


FIGURA 1.19. El puerto de *E/S* en este ejemplo, responde a la dirección **0250h**

Para los *µP* **8086/8088** se utilizan **16** de las **20** líneas de direccionamiento para direccionar la *E/S*. Esto da un total de **65,535** puertos de *E/S* disponibles en el sistema.

Aunque el sistema de *E/S* y el sistema de ***memoria*** son completamente separados, estos dos sistemas utilizan las mismas líneas de direccionamiento. Esto significa que podemos tener:

- La dirección de ***memoria*** 004Fh
- La dirección del ***puerto*** 004Fh

Una distinción puede hacerse entre estos dos sistemas revisando las líneas en el ***bus*** de control del sistema:

- Durante una operación de memoria de una de las dos líneas del ***bus*** de control *memory read* o ***memory write*** está activa.
- Durante una operación de *E/S* una de las dos líneas del ***bus*** de control *input read* o ***output write*** está activa.

Espacio de *E/S* reservado.

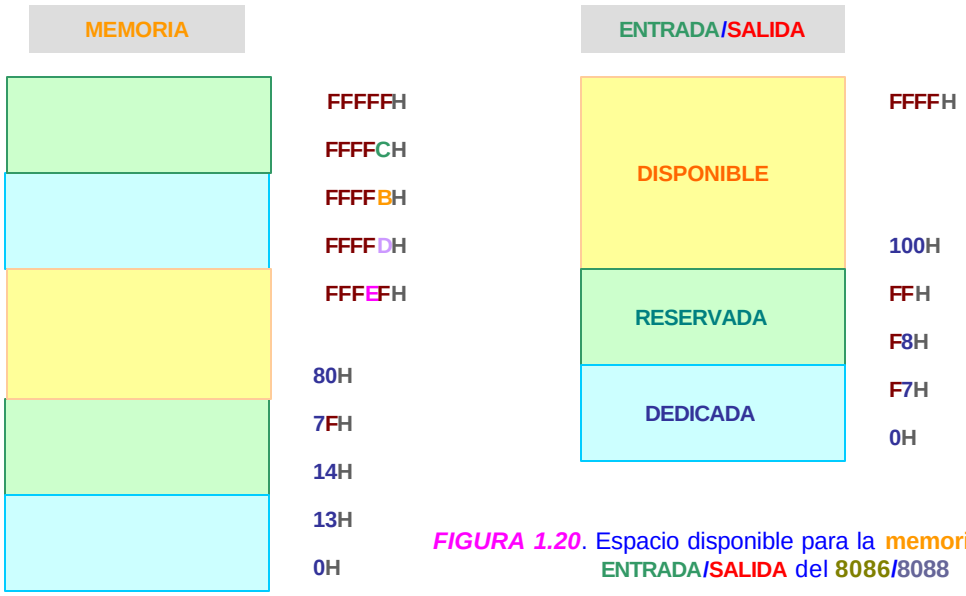


FIGURA 1.20. Espacio disponible para la memoria y la ENTRADA/SALIDA del 8086/8088

Las direcciones 00F8h a 00FFh no debemos utilizarlas ya que Intel las ha reservado para desarrollos de futuros productos. Si las usamos, no podremos utilizarlos posteriormente con los desarrollos de Intel.

¿Qué es un dispositivo de *E/S*?

Un dispositivo de ***E/S*** puede ser definido como ***hardware*** de ***E/S*** controlado por el sistema. Un dispositivo tal puede tener uno o más **puertos** de ***E/S*** o **direcciones** de ***E/S*** asociadas.

Ejemplos de tales dispositivos son los *circuitos de baja escala de integración* (***LSI***, *por sus siglas en inglés*), tales como el ***controlador*** de discos flexibles o un ***generador*** de base de tiempo.

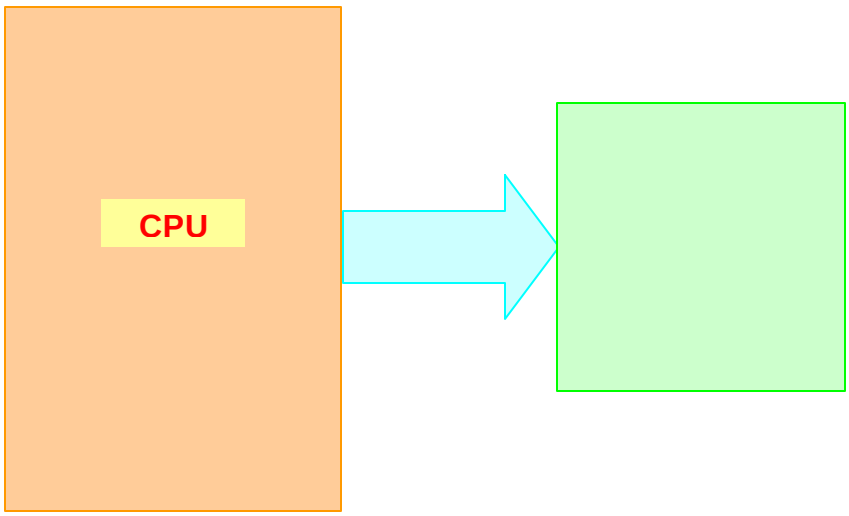


FIGURA 1.21. Un **dispositivo** de ***E/S*** debe responder eléctricamente a una o más **direcciones** de **puertos** de ***ENTRADA/SALIDA***. En este ejemplo, el dispositivo de ***E/S*** responde a las **direcciones** 00008h a la 00010h inclusive.

Software

Instrucción INPUT

Se utiliza para leer datos desde un puerto de entrada. Su estructura es:

IN acumulador A, puerto

Si el puerto es de **8** bits, el dato captado queda en el registro ***AL***. Si el puerto es de 16 bits, el dato captado queda en el registro ***AX***.

Instrucción OUTPUT

Se utiliza para transferir datos del ***µP*** a un puerto de salida. Su estructura es:

OUT puerto, acumulador A

Direccionamiento de un puerto de *E/S*.

Hay dos formas de utilizar las instrucciones anteriores:

a. Dirección a través de una **constante**

Dando la instrucción (1 byte) y la dirección del puerto (1 byte) Con este método podemos acceder un puerto ubicado en las primeras 256 localidades.

```
IN      AL, 03h
OUT     04h, AL
```

b. A través del registro **DX.**

Con esto podemos direccionar un puerto ubicado en cualquiera de las **65,536** localidades asignadas a puertos.

```
MOV     DX, 03h
IN      AL

MOV     DX, 04h
IN      AL
```

Donde: **03h** es la dirección del puerto de ***entrada*** y **04h** es la dirección del puerto de ***salida***.

Dispositivo *de E/S paralelo* (*PIO, por sus siglas en inglés*) 8255.

Es un dispositivo de ***E/S*** **programable** utilizado en un sistema ***microcomputador*** para controlar *hardware* **periférico**.

Descripción del 8255.

Es un circuito ***LSI*** encapsulado en un ***CI*** de **40** patas. Está diseñado para realizar una gran variedad de funciones de interfase en el medio ambiente de ***microprocesadores***. La ***Figura 1.G***, muestra su diagrama a bloques.

La función de cada bloque es la siguiente:

En el diagrama hay **4** bloques que conectan físicamente el **8255** al *hardware* externo. Estos bloques tienen líneas etiquetadas como PA_0 ® PA_7, PB_0 ® PB_7, PC_0 ® PC_3, PC_4 ® PC_7.

Los grupos de señales de esos bloques son divididos lógicamente en tres puertos conocidos como puerto ***A*** (***PA***), puerto ***B*** (***PB***) y puerto ***C*** (***PC***)

Estos **4** puertos (***PC*** está dividido en dos grupos) son conectados a un ***bus*** interno de datos en el **8255**. Es, vía este ***bus*** interno de datos, que los puertos son programados.

Hay dos bloques de control etiquetados como grupo de control **A** y grupo de control **B**, que definen cómo van a operar los tres puertos de ***E/S*** en el sistema.

Hay varios modos distintos de operación del **8255**, los cuales deben ser definidos por la ***palabra de control*** que él escribe en el dispositivo.

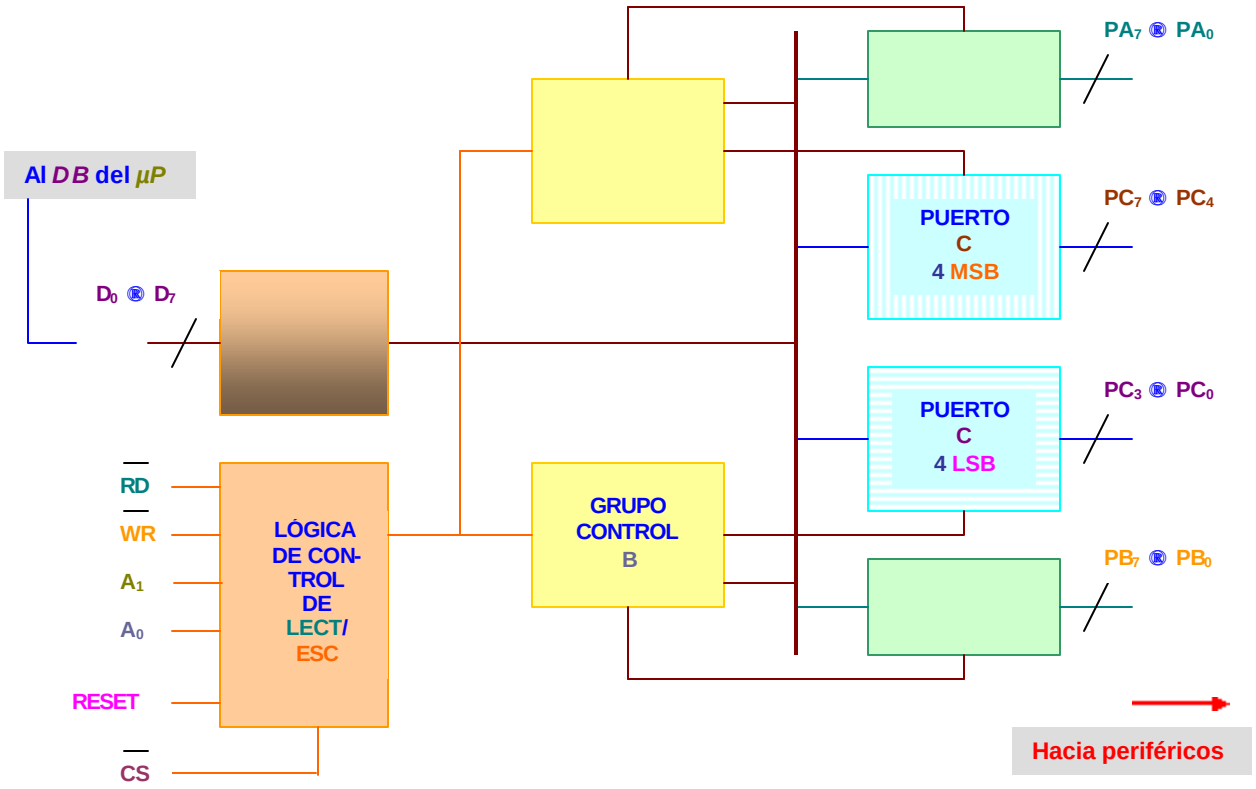


FIGURA 1.22. Diagrama a bloques del *PIO* 8255

Note que el grupo *C* consiste de dos puertos de **4** bits. Uno de los grupos de **4** bits es asociado con el grupo de control *A* y el otro con el grupo de control *B*.

Los bloques finales del diagrama a bloques son:

- *Buffer* del *bus* de datos
- Lógica de control de *LECTURA*/*ESCRITURA*

Estos dos bloques proporcionan la interfase eléctrica entre el *µP* y el **8255**.

El *buffer* del *bus* de datos del **8255** se conecta al *DB* del *µP*.

La lógica de control de *LECTURA*/*ESCRITURA* enruta el dato *hacia*/*desde* el registro interno (*AL*, *AX*) con la sincronización adecuada.

Conexión física con el *µP*.

Hay **4** *puertos internos* accesados por las líneas de dirección *A1*, *A0*. Para explicar su funcionamiento, usaremos las direcciones **10h**, **11h**, **12h** y **13h**.

Registros de lectura y escritura del 8255.

Examinaremos la definición de registros y direcciones de puertos mostrados en la *Figura 1.23*.

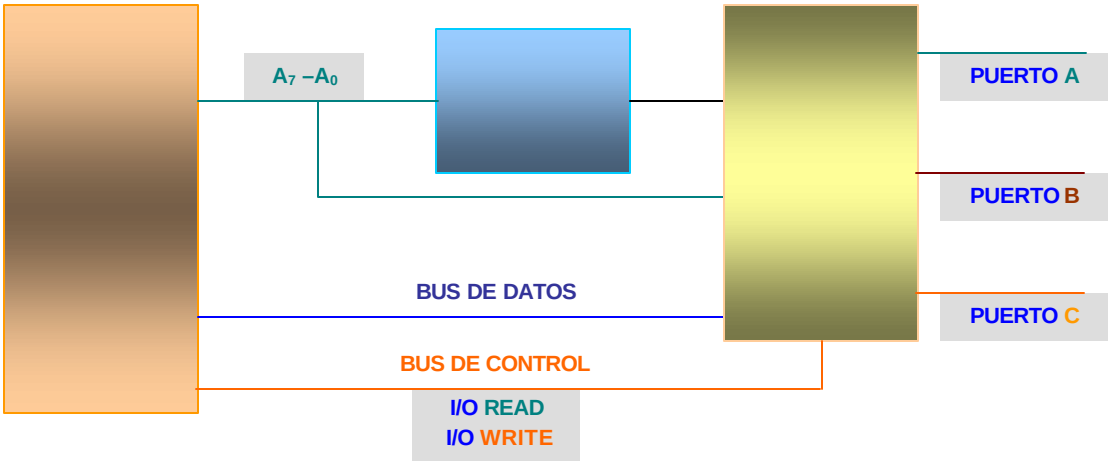
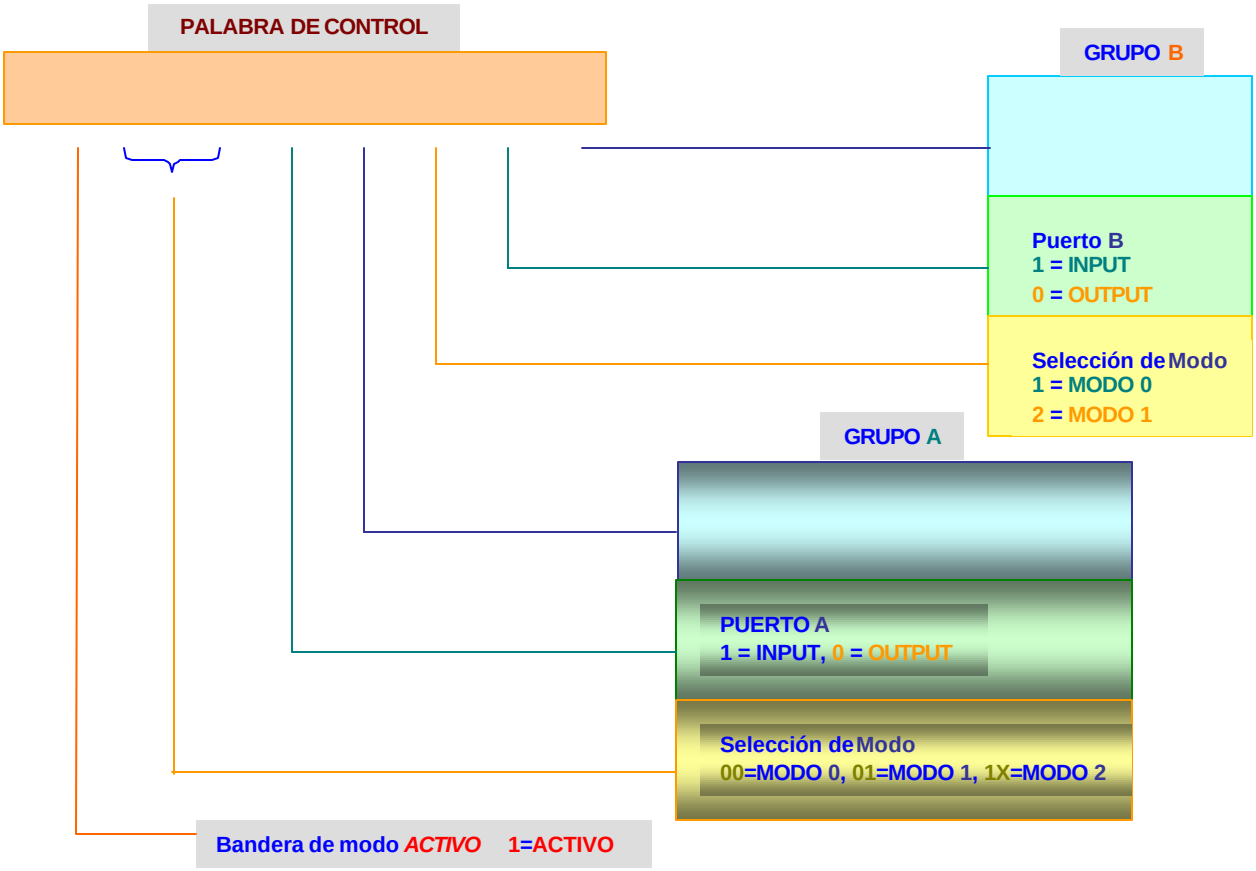


FIGURA 1.23. **Interconexión entre el *µP* y el *PIO***

DIRECCIÓN	# DE PUERTO	DEFINICIÓN
10H	0	Puerto A como SALIDA Puerto A como ENTRADA
11H	1	Puerto B como SALIDA Puerto B como ENTRADA
12H	2	Puerto C como SALIDA Puerto C como ENTRADA
13H	3	*Palabra de control* para definir la función de los puertos anteriores

La función de los registros **0** al **2** es definida por la ***palabra escrita*** en el registro **3**.



Esta figura muestra las definiciones de bits del ***registro*** de control. La utilizaremos para escribir nuestros programas.

Dispositivo de E/S serial (***Receptor/Transmisor, Sincrónico/Asincrónico Universal 8251A, USART,*** *por sus siglas en inglés*)

Proporciona dos funciones básicas:

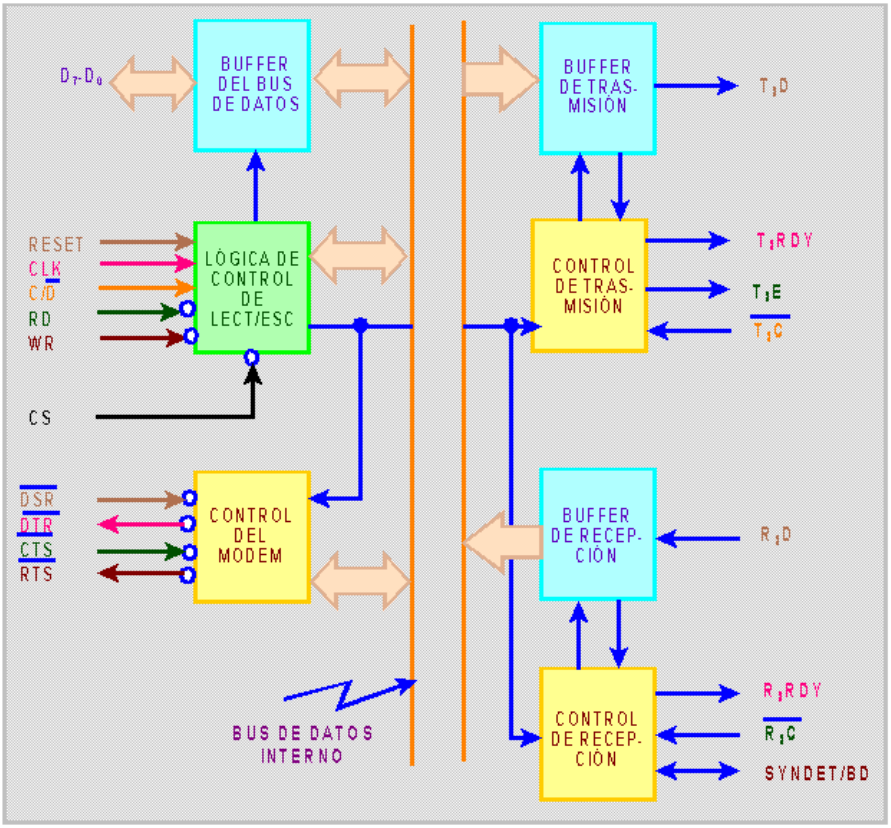
- En el modo de ***TRANSMISIÓN***, serializa el dato paralelo del ***mP*** insertando en forma automática bits de ***inicio***, ***parada*** y ***paridad*** (en el modo ***asincrónico***), o caracteres de ***sincronía*** (en el modo ***sincrónico***)
- En el modo ***RECEPTOR*** convierte el dato serie a paralelo y verifica errores de ***paridad***, ***encuadre*** (*frame errors*) y ***sobreescritura*** de datos (*overrun*)

Este circuito integrado (*IC*) tiene un bus de datos ***bidireccional***, el cual permite al ***mP*** programar su funcionamiento vía uno de tres bytes de control.

La ***Figura 1***, muestra un diagrama a bloques y descripción de patas del ***USART***. Nótese que el *IC* cuenta con un ***transmisor*** y un ***receptor*** separados, cada uno con sus propias entradas de reloj (***R_XD*** y ***T_XD***)

- ***R_XD*** es la entrada de datos serial (recibe datos)
- ***T_XD*** es la salida de datos serial (transmite datos)

Las señales de estatus ***R_XRDY*** y ***T_XRDY*** indican que los *buffers* de datos del ***receptor*** y ***transmisor*** están listos para leerse o escribirse, respectivamente.



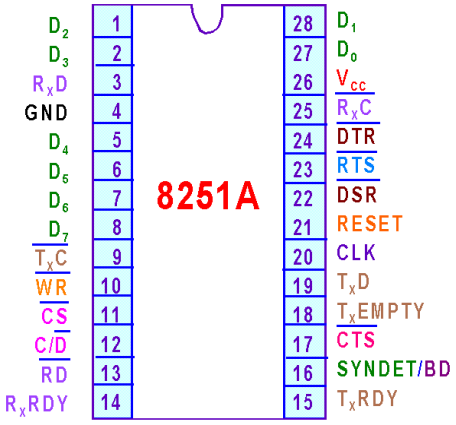


FIGURA 1. Diagrama a bloques y descripción de patas del **8251A**

La línea ***SYNDET/BD*** es una señal que va a **1** cuando el carácter de sincronía ha sido detectado cuando operamos en el modo ***sincrónico***.

En el modo ***asincrónico***, indica una condición de ***ruptura***. Esto es un nivel lógico continuo "**0**" en la línea receptora R_XD. Esta condición es enviada por un ***receptor*** al ***transmisor*** para solicitar ***ruptura*** de transmisión debida quizás a una condición de error. ***SYNDET/BD*** está disponible también vía de status del puerto.

El intercambio de datos paralelo entre el ***CPU*** y el ***USART*** viaja sobre las líneas del bus de datos ***bidireccional D7 - D_0***.

Las líneas $\overline{\mathbf{RD}}$ y $\overline{\mathbf{WR}}$ controlan la dirección del flujo de datos.

La línea $\overline{\mathbf{CS}}$ debe estar en "**0**" para que el *IC* sea seleccionado. El transmisor y el receptor continúan funcionando independientemente de la línea $\overline{\mathbf{CS}}$.

La ***Figura 2***, muestra la interfaz entre el ***mP*** **8086** y el ***USART*** **8251A**.

Cuando la línea **C/$\overline{\mathbf{D}}$** está en "**0**" selecciona al puerto de datos interno. Cuando es **1**, selecciona al puerto de control. Debido al ***MUX***, la línea AB_0 se utiliza para seleccionarlos.

Se utiliza decodificación completa para las **16** líneas de ***E/S*** en forma tal que el ***USART*** es mapeado en puertos consecutivos **0070h** y **0071h**, seleccionados por AB_0.

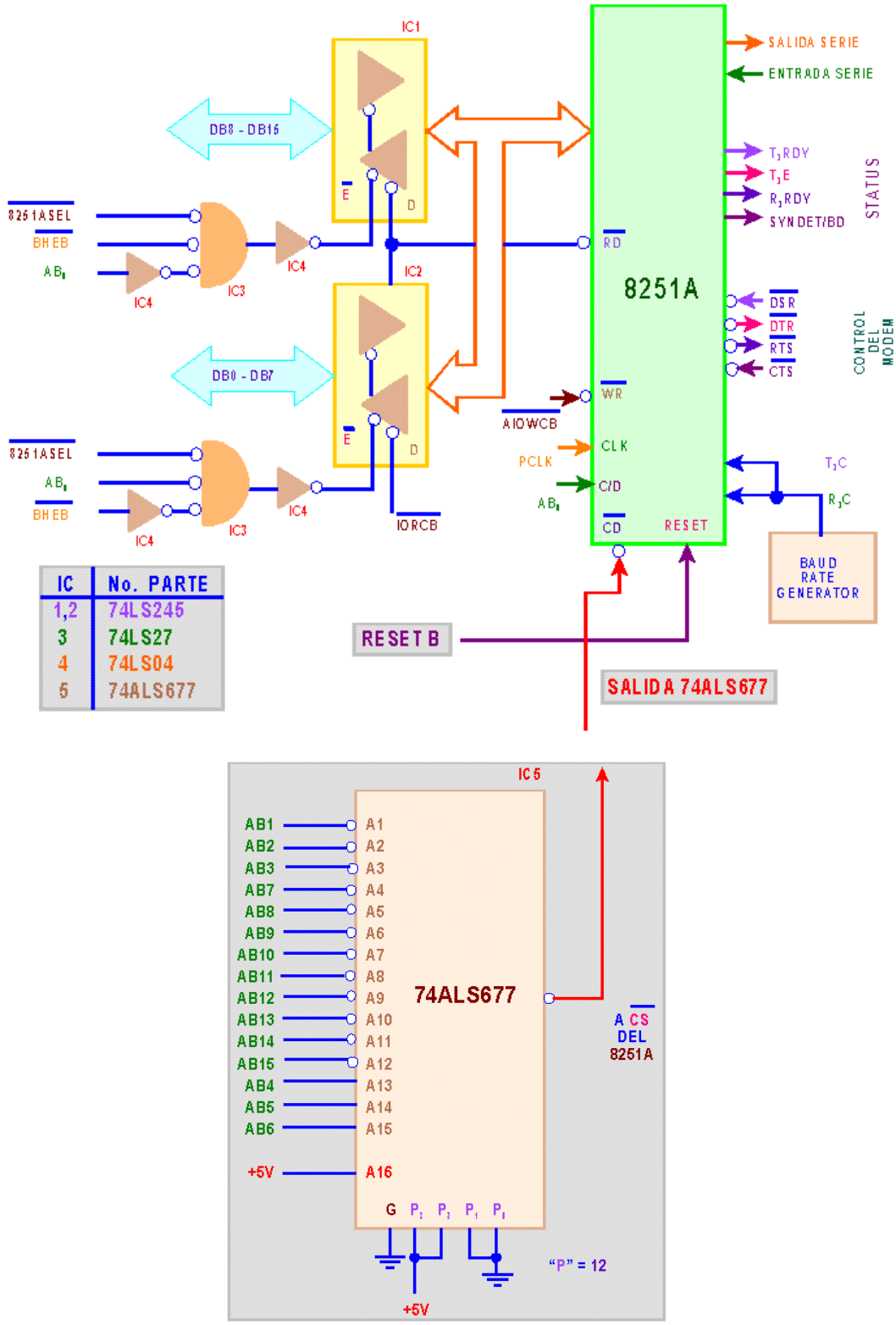


FIGURA 2. **Interfaz del *mP* 8086 y *USART* 8251A**

La siguiente tabla resume las funciones de ***lectura*** y escritura de cada uno de los ***puertos***:

C/D	$\overline{RD}$	$\overline{WR}$	Dirección del puerto	FUNCIÓN
0	0	1	70h	Lectura de un byte de datos
0	1	0	70h	Escritura de un byte de datos
1	0	1	71h	Lectura de un byte de datos
1	1	0	71h	Escritura de un byte de datos

La *Figura 2* también muestra las cuatro señales de control del *MODEM*: $\overline{DSR}$, $\overline{DTR}$, $\overline{RTS}$ y $\overline{CTS}$. Con excepción de $\overline{CTS}$, estas señales son de propósito general y no afectan la operación del circuito; sin embargo, $\overline{CTS}$ debe ser igual a **0** si el transmisor interno va a ser habilitado.

Se requiere la entrada de reloj **CLK** para sincronización interna, debe tener una frecuencia de al menos **30** veces la frecuencia del transmisor o receptor en el modo sincrónico y al menos **4.5** veces la frecuencia del transmisor o receptor en el modo asincrónico.

NOTA DE APLICACIÓN: En la mayoría de los casos, los niveles lógicos **TTL** en las líneas de entrada o salida serie deben trasladarse a especificaciones **RS-232C** (**EIA** – *Electronic Industries Associates*) Esto es, **-12V** para el nivel lógico "**1**" y **+12V** para el nivel lógico "**0**".

Después de trasladar los niveles **TTL** del *USART* a niveles **RS-232C**, el dato serial puede ser transmitido a miles de pies.

Programación del 8251A para modo asincrónico.

Utilice la siguiente secuencia:

1. **Desactivar** el circuito integrado (externamente o a través del byte de control interno) Externamente a través de la pata *RESET*.
2. **Escribir** la instrucción de *modo* en el puerto de control.
3. **Escribir** la instrucción de *comando* en el puerto de control.

La *Figura 3* describe la forma de las instrucciones de *modo* y *comando*.

Use un comando *RESET* para la secuencia de iniciación, el siguiente comando será interpretado como una instrucción de *MODO*. Después de que este byte ha sido escrito, todas las escrituras posteriores al registro de control deben ser interpretadas como instrucciones *COMANDO*. La única forma de retornar a la instrucción *MODO* es aplicando un pulso *RESET* o escribir un byte *COMANDO* con el bit **6** igual a **1**.

EJEMPLO: Escriba la rutina de *iniciación* para programar el **8251A** para transmisión *asincrónica* con **7** bits de *datos*, **2** bits de *parada* y *paridad impar*. Seleccione un *reloj* **16x** y programe $\overline{DTR}$ y $\overline{RTS}$ a "**0**".

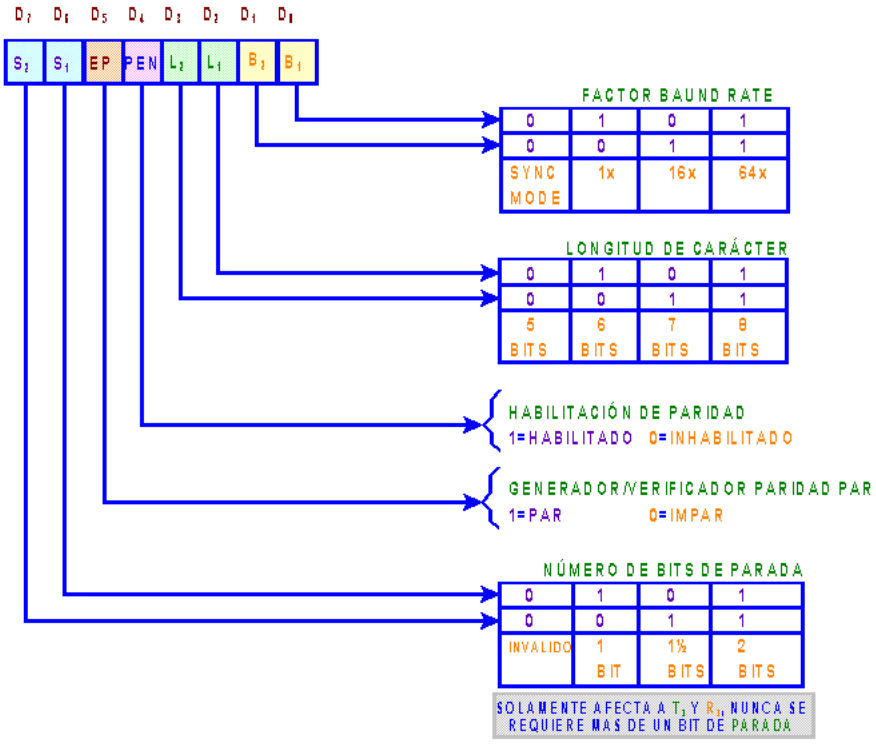


FIGURA 3.a. Formato de instrucción modo ***asincrónico***

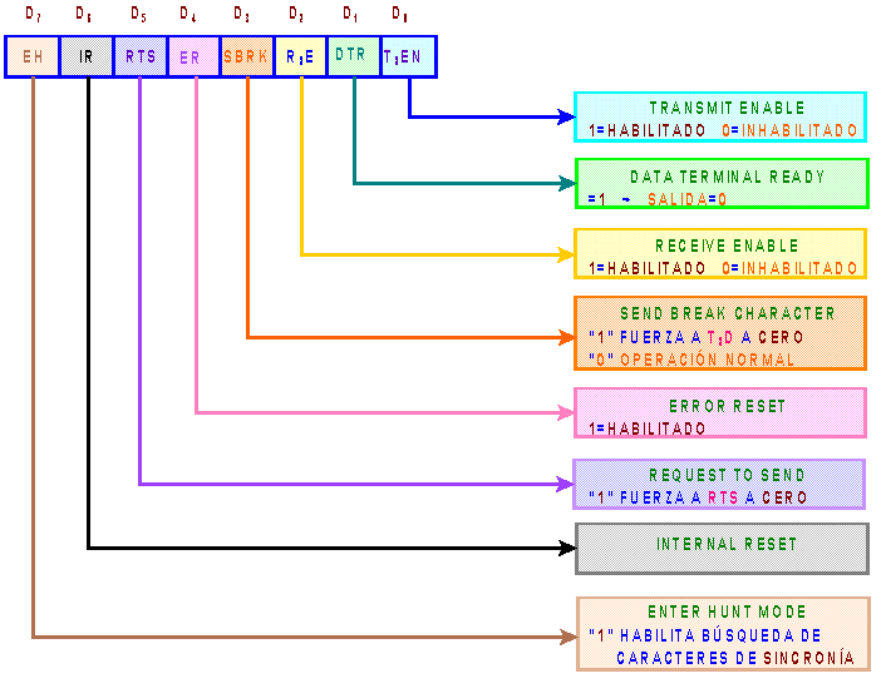


FIGURA 3.b. Formato de instrucción para **comando** ***sincrónico*** o ***asincrónico***

SOLUCIÓN

```
MOV   AL, 01000000B        ;comando RESET
OUT   71h, AL              ;puerto de control
```

```
MOV   AL, 11011010B      ;instrucción de modo:
                         ;7 datos, 2 bits de parada,
                         ;paridad impar, 16x
OUT   71h, AL            ;puerto de control

MOV   AL, 00110111B      ;instrucción comando:
                         ; RTS y DTR o error RESET habilitado
OUT   71h, AL            ;puerto comando
IN    AL, 70h            ;lectura ficticia para
                         ;borrar el receptor
```

NOTA: La habilitación del ***receptor*** (bit **2** de la instrucción ***comando***) solamente inhibe la bandera R_XRDY, no al ***receptor***. Así, es posible para el ***receptor*** haber capturado un carácter antes o durante la rutina de ***iniciación***. La lectura ***ficticia*** asegura que el ***receptor*** retenga un registro en ***blanco***.

TAREA

a. **Investigue** los pasos necesarios para programar el **8251A** para el modo ***sincrónico***. Muestre varios ejemplos.

b. **Muestre** el código necesario para el ***control*** de:

- Una ***impresora***
- Un ***MODEM***. Ayúdese de las rutinas del ***BIOS*** de **DOS**.