
Introducción a Pilas



Pila

Una **Pila** (Stack) es una estructura de datos lineal. Es una lista ordenada en la cual la adición y remoción de datos se realiza solamente por un extremo, conocido como el **Tope** de la pila.

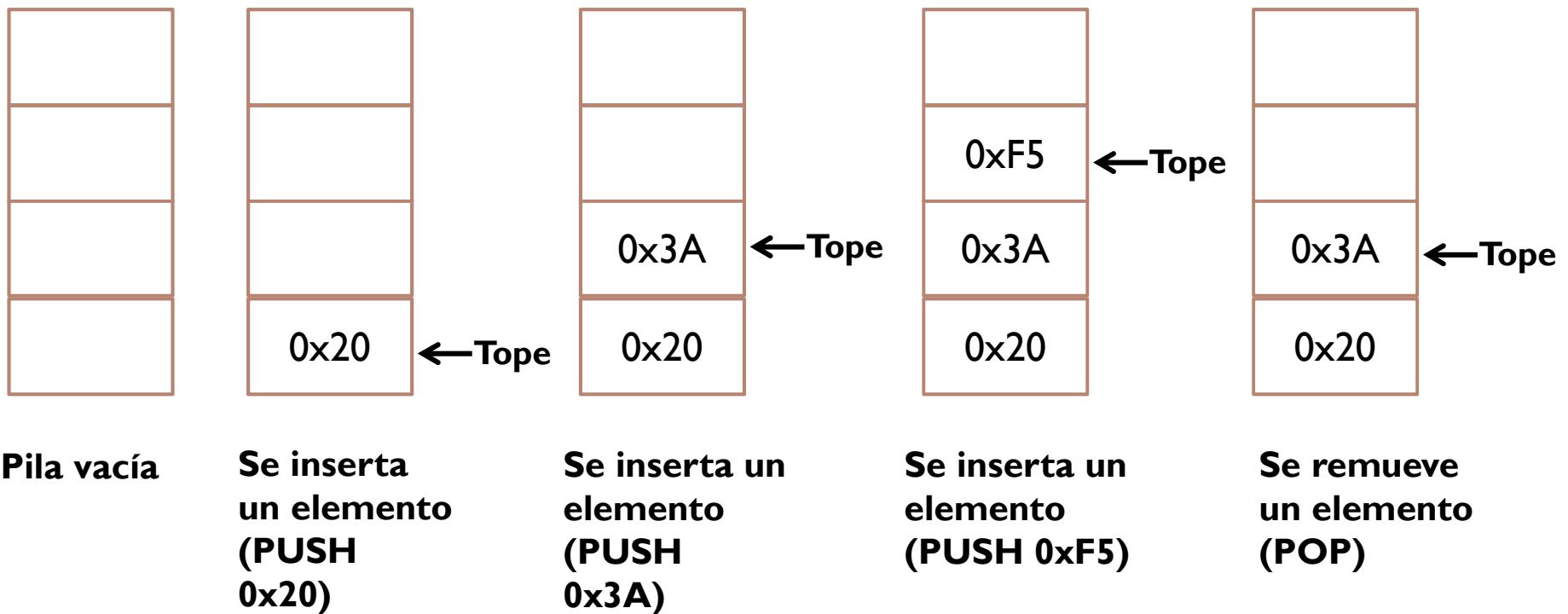
Debido a esto, el último elemento agregado a la pila será el primero que será removido. A esto se le conoce como **Last In First Out (LIFO)**.



Pila

Representación grafica:

Tope → SP (Stack Pointer)



Pila

Operaciones sobre las Pilas:

Las operaciones básicas que se pueden realizar en una pila son:

1. **PUSH:** Es el proceso de agregar un nuevo elemento al tope de la pila.
Se incrementa el apuntador de tope de pila y se inserta el elemento.

2. **POP:** Es el proceso de eliminar un elemento del tope de la pila.
Se retorna el elemento direccionado por el apuntador de tope de pila, y posteriormente se decrementa el apuntador.



Pila

La Pila del Procesador 8088:

Se dice que la Pila del 8088 crece a direcciones más bajas ya que:

Una operación **PUSH** causa que se decremente el apuntador de tope de pila (**SP**), y se inserte el dato en la nueva dirección apuntada por SP.

Y una operación **POP** causa que se retorne el dato direccionado por SP, y posteriormente se incremente SP.

Es decir, al estar insertando nuevos datos a la Pila estos se están almacenando en direcciones cada vez mas pequeñas; es por eso que se dice que "crece a direcciones más bajas".



Arquitectura RISC y CISC



Un aspecto muy importante de la arquitectura de computadoras es el diseño del conjunto de instrucciones para el procesador. El conjunto de instrucciones elegido para una computadora particular determina la manera en que se construyen los programas de lenguaje máquina.

Las primeras computadoras tenían conjuntos de instrucciones pequeños y simples, forzados sobre todo por la necesidad de minimizar la circuitería utilizada para implementarlos.

Conforme la circuitería digital se hizo más barata con la aparición de los circuitos integrados, las instrucciones tendieron a aumentar, tanto en cantidad como en complejidad.



CISC

Una computadora con una gran cantidad de instrucciones se clasifica como una **Computadora de Conjunto de Instrucciones Complejo, CISC** (Complex Instruction Set Computer).



RISC

Al principio de los 80, muchos diseñadores de computadoras recomendaron que las maquinas utilizaran menos instrucciones con fórmulas más sencillas que pudieran ejecutarse con mayor rapidez dentro del CPU, sin tener que utilizar la memoria con tanta frecuencia.

Este tipo de computadoras se clasifica como **Computadoras de Conjunto de Instrucciones Reducido, RISC** (Reduced Instructions Set Computers).

El concepto de la arquitectura RISC significa un intento para reducir el tiempo de ejecución al simplificar el conjunto de instrucciones de la computadora.



RISC

Características:

1. Relativamente pocas instrucciones.
2. Relativamente pocos modos de direccionamiento.
3. El acceso a memoria limitado a instrucciones de obtención y almacenamiento de datos.
4. Todas las instrucciones realizadas dentro de los registros del CPU.
5. Formatos de instrucciones decodificados con facilidad, de longitud fija.
6. Ejecución del ciclo de instrucción única. Capacidad de ejecutar una instrucción por ciclo
7. Control por circuitería en lugar de microprograma.



RISC

Una característica de los procesadores RISC es su capacidad para ejecutar **una instrucción por ciclo de reloj.**

Esto se logra al hacer simultáneamente las fases de recuperación (obtención de la instrucción), decodificación y ejecución de dos o tres instrucciones, utilizando **segmentación encauzada (pipeline)**.

Una instrucción de obtener o almacenar datos de memoria puede requerir dos ciclos de reloj porque el acceso a memoria toma más tiempo que las operaciones de registro.



RISC

Se suelen atribuir las siguientes características a la arquitectura RISC:

1. Una cantidad de registros en el procesador relativamente grande.
2. Uso de ventanas de registros traslapados para acelerar la llamadas y retorno de Procedimientos
3. Paralelismo de instrucciones eficiente.
4. Soporte de compilador para traducción eficiente de programas de lenguaje de alto nivel a programas de lenguaje máquina.



Filosofía RISC vs CISC

RISC	CISC
Instrucciones sencillas en un ciclo	Instrucciones complejas en varios ciclos
Solo LOAD/STORE hacen referencia a memoria	Cualquier instrucción puede referenciar a memoria
Procesamiento serie de varias etapas	Poco procesamiento en serie
Instrucciones ejecutadas por hardware	Instrucciones interpretadas por un microprograma
Instrucciones de formato fijo	Instrucciones de formato variable
Pocas instrucciones y modos	Muchas instrucciones y modos
La complejidad está en el compilador	La complejidad está en el microprograma
Varios conjuntos de registros	Un solo conjunto de registros



Arquitectura Harvard y Von Neumann



Arquitectura Harvard y Von Neumann

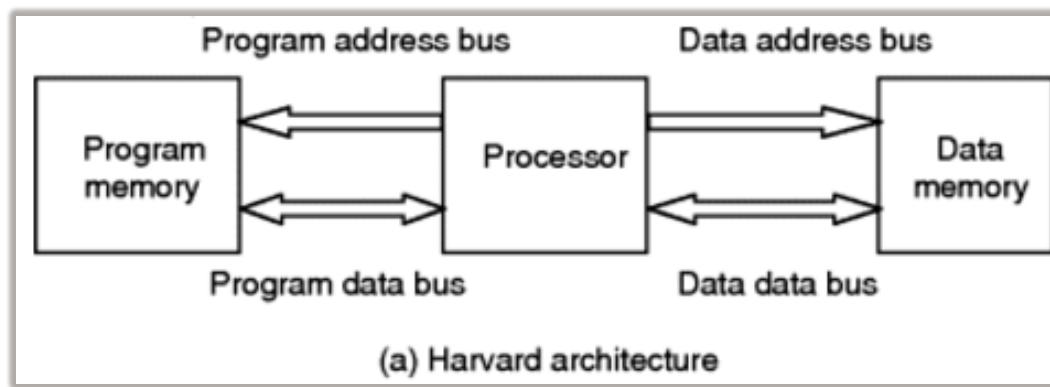
Las arquitecturas de los microprocesadores y microcontroladores generalmente recaen en una de estas dos categorías:

- Arquitectura **Harvard**
- Arquitectura **Von Neumann**



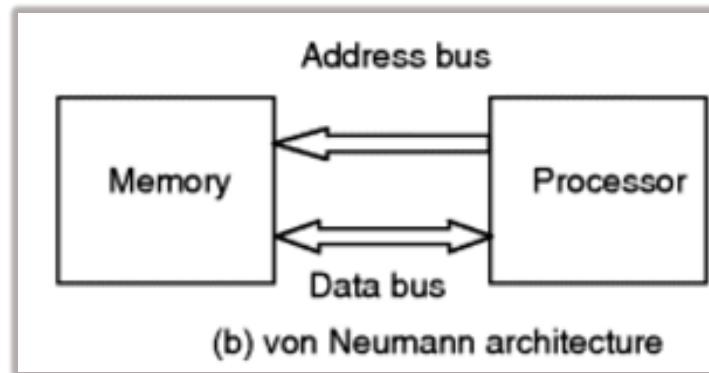
Arquitectura Harvard

En la arquitectura Harvard se manejan espacios de memoria separados para almacenar las instrucciones del programa y los datos, por lo que ambas memorias pueden ser accedidas simultáneamente.



Arquitectura Von Neumann

En la arquitectura Von Neumann el programa y los datos están almacenados en la misma memoria.



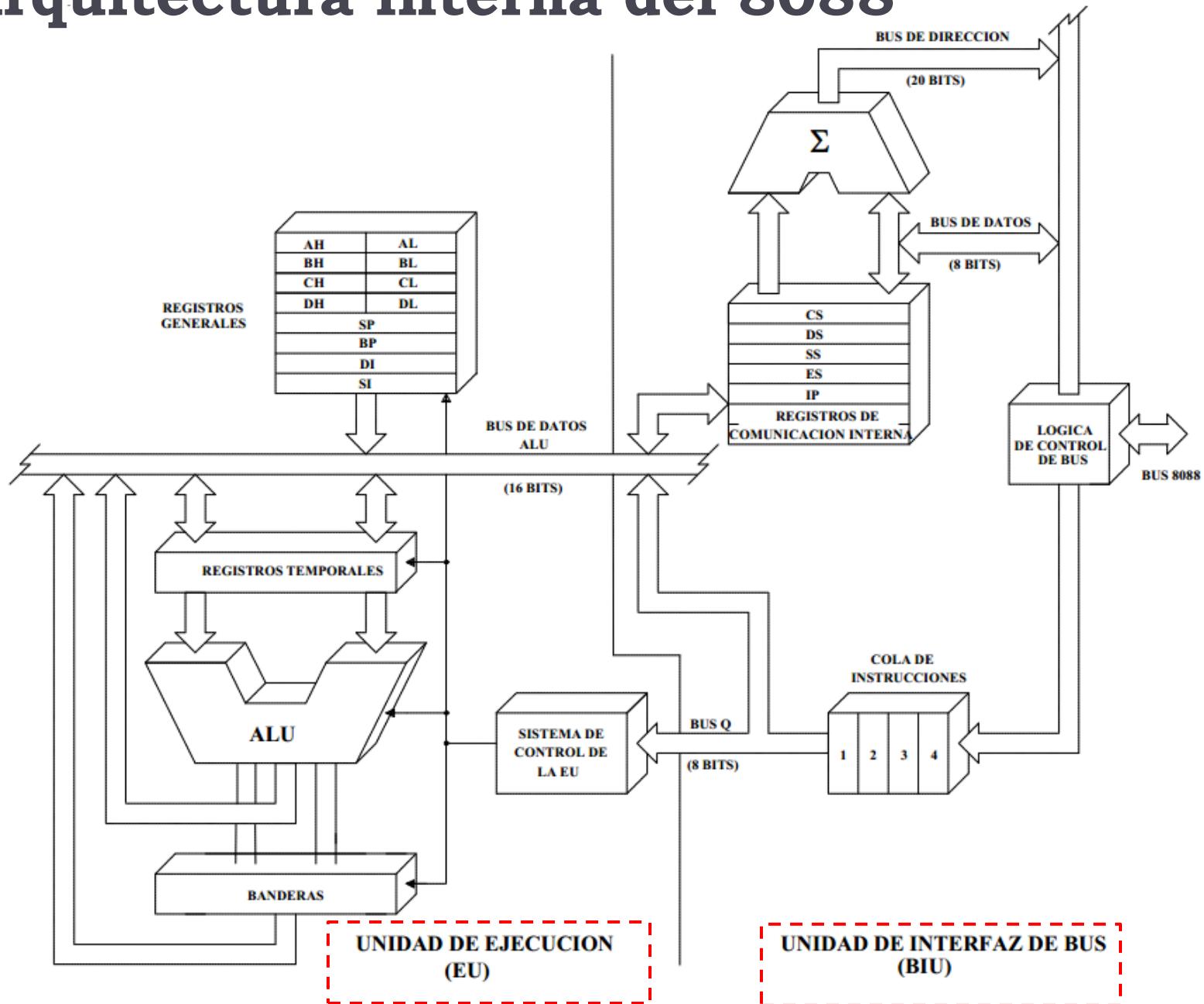
Arquitectura interna del 8088

Arquitectura interna del 8088

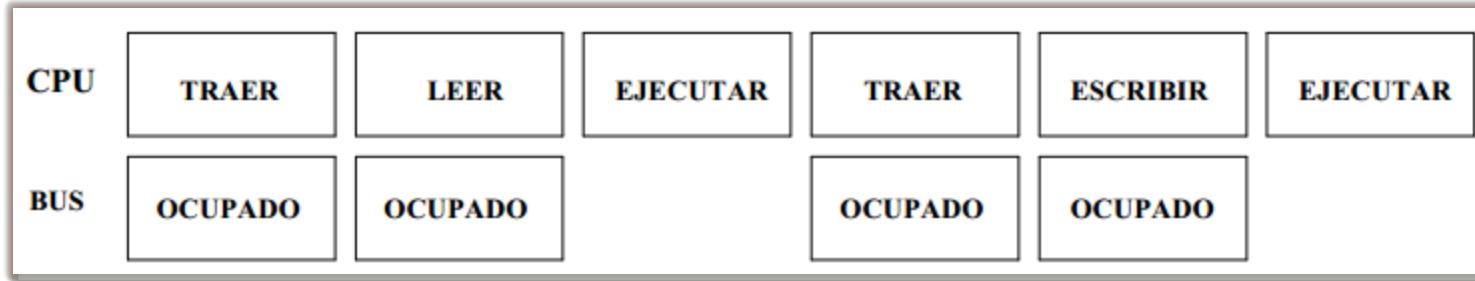
Intel diseño el 8088/8086 para realizar al mismo tiempo las principales funciones internas de transferencia de datos y búsqueda de instrucciones. Para conseguir esto, el 8088 y el 8086 constan de dos procesadores interconectados en el mismo circuito integrado.

Una unidad esta encarga de **buscar instrucciones (BIU)** y la otra de **ejecutarlas (EU)**. Esto diferencia al 8088 y 8086 de los procesadores anteriores (8080 y 8085).

Arquitectura interna del 8088



Comparación operacional entre el 8085 y el 8088



a) Operación del 8085 y actividad del bus



b) Operación de las unidades del 8088 y actividad del bus

Arquitectura interna del 8088

Unidad de Interfaz de Bus (BIU)

Contiene una **cola de instrucciones**, un **controlador de bus**, **registros de segmento** y el **puntero de instrucción** (IP: Instruction Pointer o Contador de Programa).

La principal función de la BIU es mantener llena la cola de instrucciones, generar y aceptar señales de control, proveer al sistema de direcciones de memoria y número de puerto de E/S, además de ser el mediador entre la Unidad de Ejecución (EU) y la memoria.

Arquitectura interna del 8088

Unidad de Interfaz de Bus (BIU)

La BIU asegura que la cola de instrucciones esté llena mediante la operación de traer la próxima instrucción si la cola tiene espacio.

Debido a que la próxima instrucción a ejecutar ya se va a encontrar dentro del microprocesador (en la cola de instrucciones), la velocidad de ejecución de los programas es mucho mas rápida en comparación a que si cada instrucción a ejecutar fuese traída directamente de memoria en el momento en que se va a ejecutar.

Arquitectura interna del 8088

Unidad de Ejecución (EU)

La función de la EU es sacar cada instrucción de la cola de instrucciones y ejecutarla.

La EU contiene una ALU, un registro de instrucción y un arreglo de registros.

Arquitectura interna del 8088

Unidad de Ejecución (EU)

La **ALU** realiza operaciones aritméticas y lógicas sobre la memoria o sobre registros.

El **registro de instrucción** recibe instrucciones de la cola de instrucción y son decodificadas a operaciones directas para la unidad de ejecución.

El **arreglo de registros** mantiene información temporalmente. También contiene registros índices y punteros utilizados para direccionar operandos localizados en memoria.

Arquitectura interna del 8088

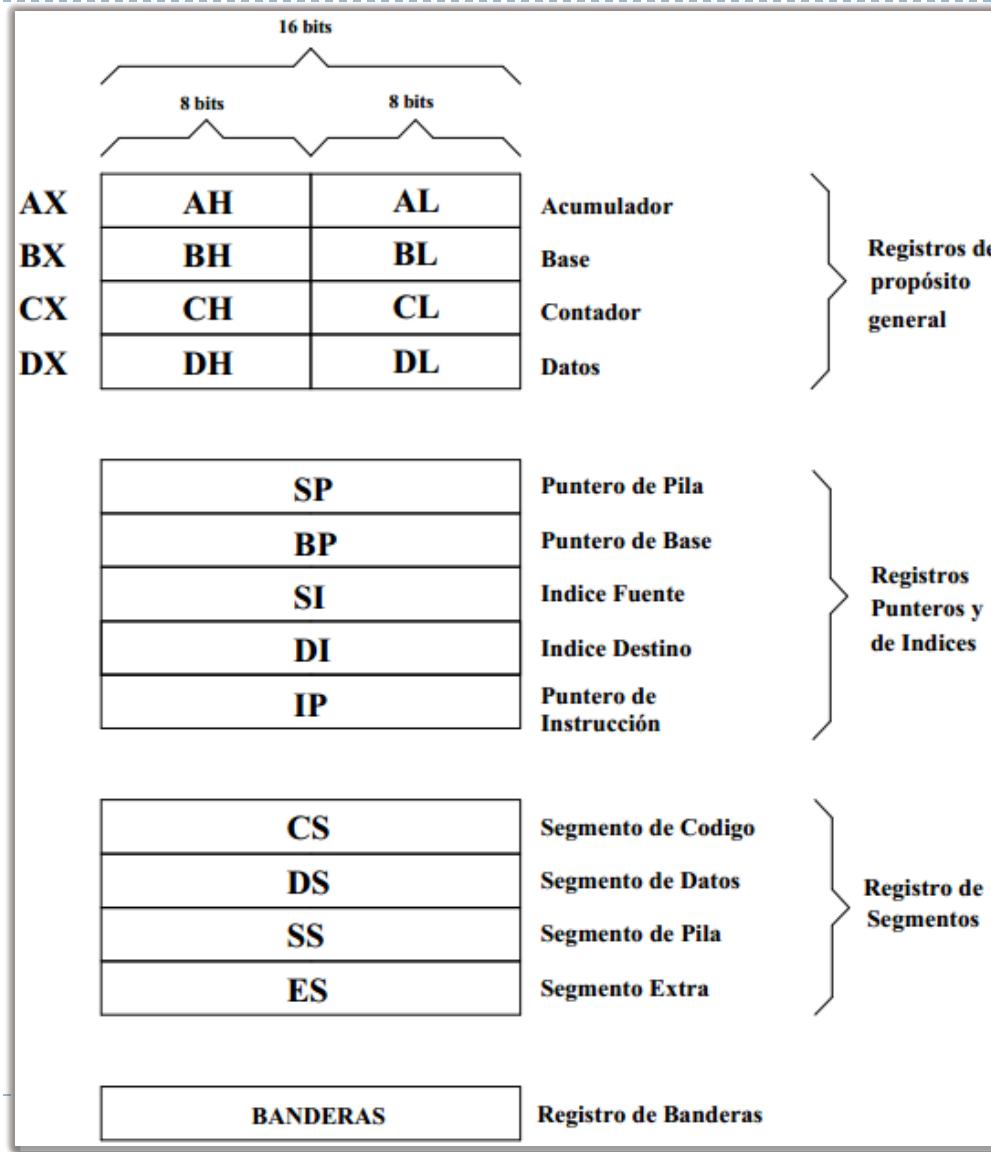
Conjunto de registros del 8088

El 8088 contiene 14 registros de 16 bits que se asocian a tres grupos:

- **Registros de Propósito General**
- **Registros Punteros y de Índice**
- **Registros de Segmentos**

Además de contiene un **Registro de Banderas** que indica el estado de la última operación realizada por la ALU.

Conjunto de registros del 8088



Conjunto de registros del 8088

Registros de Propósito General

Estos registros se pueden utilizar de cualquier manera que el programador desee (siempre y cuando sea una operación permitida).

Pueden manejarse como registros de 16 bits: **AX**, **BX**, **CX** y **; o cada uno como dos registros independientes de 8 bits: **AH**, **AL**, **BH**, **BL**, **CH**, **CL**, **DH** y **DL**.**

La principal función de los registros de propósito general se describe a continuación:

Conjunto de registros del 8088

Registros de Propósito General

AX (Acumulador): Generalmente utilizado para mantener temporalmente resultados después de una operación aritmética o lógica.

BX (Base): Generalmente utilizado para mantener la dirección base de un dato localizado en memoria.

CX (Contador): Contador en algunas instrucciones, tales como ciclos, corrimientos y rotaciones.

DX (Dato): Mantiene los bits mas significativos del producto después de una multiplicación de 16 bits, también los bits mas significativos del dividendo antes de una división, y el numero del puerto de E/S en una instrucción de E/S.

Conjunto de registros del 8088

Registros Punteros y de Índice

Son utilizados como índice o punteros a una localidad de memoria.

SP (Puntero de Pila): Direcciona datos de una pila de memoria tipo LIFO (Last in, First Out). Se modifica su valor cuando se ejecuta una instrucción PUSH o POP, o cuando se invoca una subrutina mediante una instrucción CALL y cuando se retorna de ella mediante un RET.

BP (Puntero de Base): Se utiliza para direccionar datos en la pila de memoria, teniendo el programador total control sobre el valor de este registro.

Conjunto de registros del 8088

Registros Punteros y de Índice

SI (Índice Fuente): Se utiliza para direccionar indirectamente datos fuente mediante el uso de instrucciones con cadenas.

DI (Índice Destino): Se utiliza para direccionar indirectamente datos destino mediante el uso de instrucciones con cadenas.

IP (Puntero de Instrucción): Direcciona la próxima instrucción a ejecutar. La dirección de la próxima instrucción a ejecutar esta formada por: **IP + CS x 10h**

Conjunto de registros del 8088

Segmentos de Memoria y Registros de Segmento

Un **segmento de memoria** es un bloque de 64 Kbytes de memoria direccionado por un **registro de segmento**.

Cuatro segmentos diferentes pueden existir simultáneamente en el espacio de memoria:

- **Segmento de Código**
- **Segmento de Datos**
- **Segmento de Pila**
- **Segmento Extra**

Conjunto de registros del 8088

Segmentos de Memoria y Registros de Segmento

Los **datos** que se encuentran en cada uno de los segmentos son apuntados o indexados por los registros de punteros e índices, registro base o registro de instrucción:

Segmento de Memoria	Datos apuntados por:
Código	IP
Datos	BX, SI y DI
Pila	SP y BP
Extra	SI y DI

Conjunto de registros del 8088

Segmentos de Memoria y Registros de Segmento

La dirección de inicio de cada segmento de memoria es almacenada por un **registro de segmento**.

Los registros de segmento almacenan 16 de los 20 bits de la dirección del segmento.

La dirección de inicio de cada segmento consiste en el valor de su registro de segmento asociado, multiplicado por 10h.

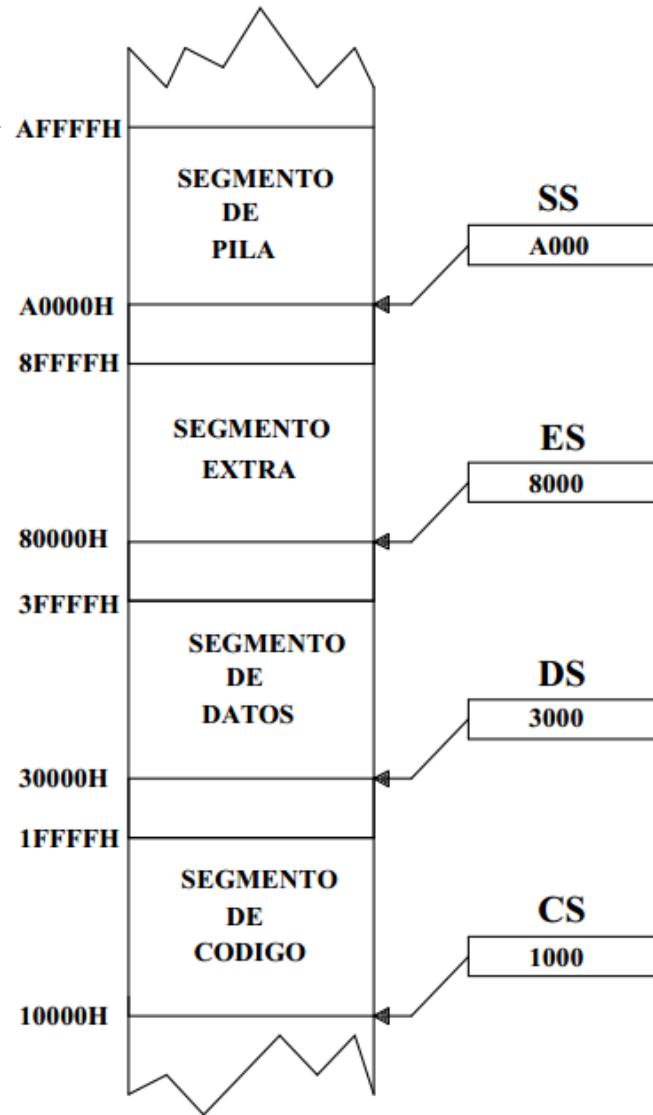
Conjunto de registros del 8088

Segmentos de Memoria y Registros de Segmento

Segmentos de memoria y sus registros de segmento asociados:

Segmento de Memoria	Direccionado por:
Código	CS
Datos	DS
Pila	SS
Extra	ES

Memoria



Ejemplo que muestra el contenido de cada registro de segmento y la dirección de inicio de cada segmento de memoria

Conjunto de registros del 8088

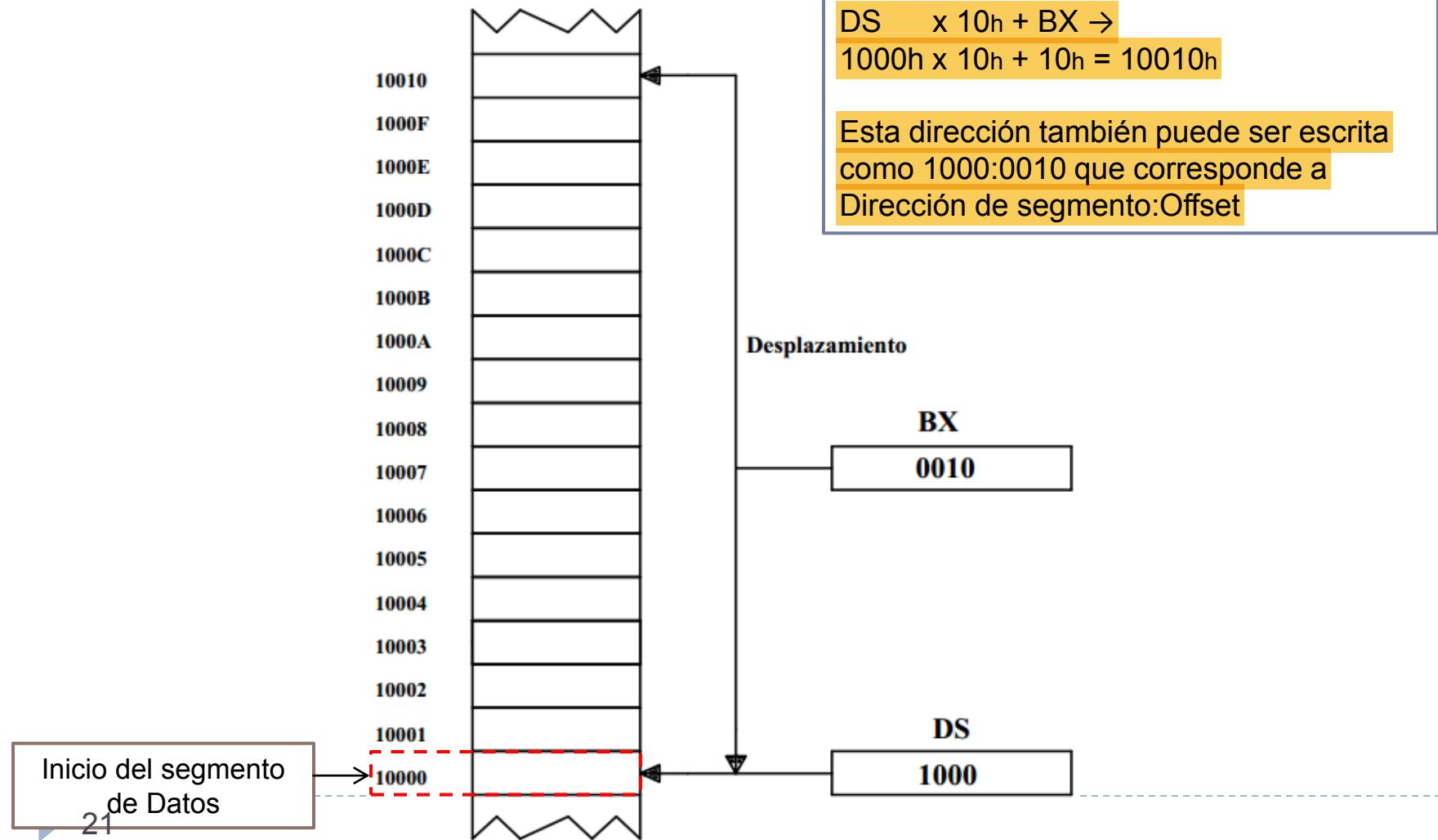
Segmentos de Memoria y Registros de Segmento

Para direccionar un dato en un segmento de memoria, el contenido del registro de segmento asociado contiene la parte de la dirección del segmento, y a este se le incrementa un registro índice o puntero, el cual contiene el desplazamiento (**offset**).

La **dirección efectiva** es la suma de la dirección del **segmento** y el **offset**.

Conjunto de registros del 8088

Ejemplo:



Conjunto de registros del 8088

Segmentos de Memoria y Registros de Segmento

Cada segmento de memoria tiene una función especial.

Segmento de Código: Una sección de memoria de 64 Kbytes que contiene el programa o código. La dirección de la próxima instrucción a ejecutar es generada por la suma del contenido del puntero de instrucción y el contenido de CSx10h.

Segmento de Datos: Una sección de memoria de 64 Kbytes que contiene los datos direccionados por todas las instrucciones y modos de direccionamiento. La dirección efectiva de un dato es generada por la suma del contenido de uno de los registros índice o puntero (BX, SI o DI) y el contenido de DSx10h.

Conjunto de registros del 8088

Segmentos de Memoria y Registros de Segmento

Segmento de Pila: Una sección de memoria de 64 Kbytes usada por la pila tipo LIFO. La dirección efectiva de la pila es una combinación del contenido del Puntero de Pila mas el contenido de SS. El registro BP también puede direccionar datos de la pila.

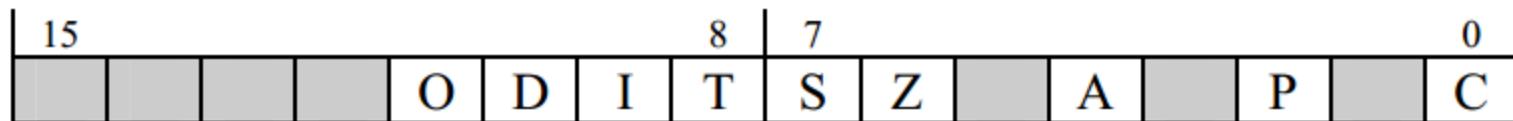
Segmento Extra: Un segmento especial que normalmente se usa por instrucciones de cadenas. Cuando una instrucción de cadena es ejecutada, la localidad del destino es direccionada por el registro índice destino (DI) más ESx10h, y la dirección fuente esta direccionada por el registro índice fuente (SI) más DSx10h.

Conjunto de registros del 8088

Registro de Banderas

También llamado registro de estado, es un registro de 16 bits el cual contiene información sobre el estado de la última operación realizada por la ALU.

En la siguiente imagen se muestra la posición relativa de cada bit de bandera.



Conjunto de registros del 8088

Registro de Banderas

C (Acarreo): Indica un acarreo, o un préstamo en el bit mas significativo después de una operación aritmética. Esta bandera también se modifica por algunas instrucciones de corrimiento y rotación.

P (Paridad): Se refiere a la paridad del resultado de una operación aritmética o lógica. Esta bandera es un 1 si el resultado contiene un numero par de unos, de lo contrario es 0.

A (Acarreo Auxiliar): Representa un acarreo o préstamo entre medio-bytes (nibbles) de una operación aritmética o lógica entre registros de 8 bits.

Conjunto de registros del 8088

Registro de Banderas

Z (Cero): Indica si el resultado de una operación aritmética o lógica es cero, en caso de que si sea, entonces $Z=1$.

S (Signo): Indica el signo del resultado de una operación aritmética o lógica. Si el resultado es negativo, $S=1$.

T (Atrapar): Causa que el 8088 entre a un estado de operación paso a paso para depuración.

I (Habilitar interrupciones): Habilita o deshabilita la terminal INTR (requerimiento de interrupción). Si $I=1$ entonces INTR esta habilitada.

Conjunto de registros del 8088

Registro de Banderas

D (Dirección): Selecciona el modo de operación de auto-incremento o auto-decremento para los registros DI y SI en operaciones de cadena. Si D=0, entonces DI y SI son incrementados durante la ejecución de una instrucción de cadena.

O (Sobreflujo): Se activa después de que una operación aritmética de suma o resta a ocurrido un sobreflujo. Por ejemplo, si 7Fh (+127) se suma a 01h (+1), y se esta operando con números con signo, el resultado es 80h (-128). Debido a que -128 no es un resultado correcto, la bandera es puesta en 1 para indicar un sobreflujo.

Modos de Direccionamiento



El *código de operación* de una instrucción especifica la instrucción a ejecutar, la cual va a actuar sobre ciertos operandos que consisten en datos almacenados en registros o en memoria.

El **modo de direccionamiento** especifica la forma de interpretar la información contenida en el campo operando de la instrucción, para poder localizar en base a esta información el/los operando(s) sobre los que va a actuar la instrucción.

ADD AX , BX



Operando

Modos de Direccionamiento

Generalmente se utiliza la instrucción **MOV** (movimiento de dato) para describir los modos de direccionamiento de datos, aunque estos aplican para el resto de las instrucciones.

La instrucción **MOV** transfiere bytes o palabras de datos entre los registros o entre la memoria y los registros.

En el caso del 8088, una palabra corresponde a 16 bits (2 bytes).



Modos de Direccionamiento

Instrucción MOV

Copia un byte o palabra de un registro a otro, o entre memoria y un registro.



MOV Destino , Fuente
MOV AX , BX

Esta instrucción copia la palabra almacenada en BX al registro AX. El contenido de BX no se modifica tras esta operación.



Modos de Direccionamiento

Instrucción MOV

Ejemplos:

- **MOV DX, F4E3**

Asigna el valor 0xF4E3 al registro DX

- **MOV CL,AL**

Copia el byte almacenado en AL al registro CL. El contenido de AL no se modifica.

- **MOV DH,[BX]**

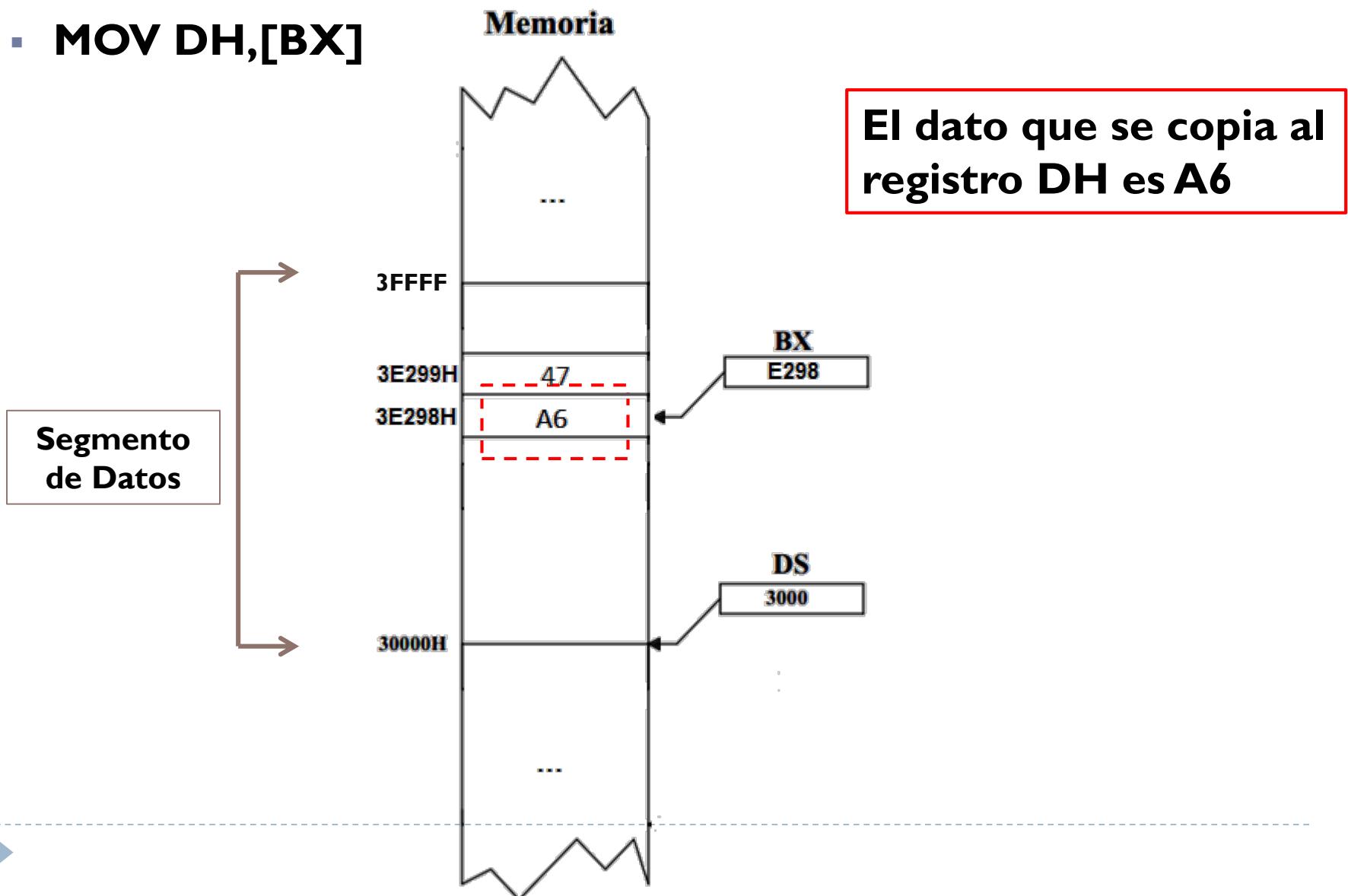
Copia el byte almacenado en memoria en la dirección apuntada por BX en el registro DH.

Copia el byte almacenado en la posición DS:BX



Modos de Direccionamiento

- **MOV DH,[BX]**

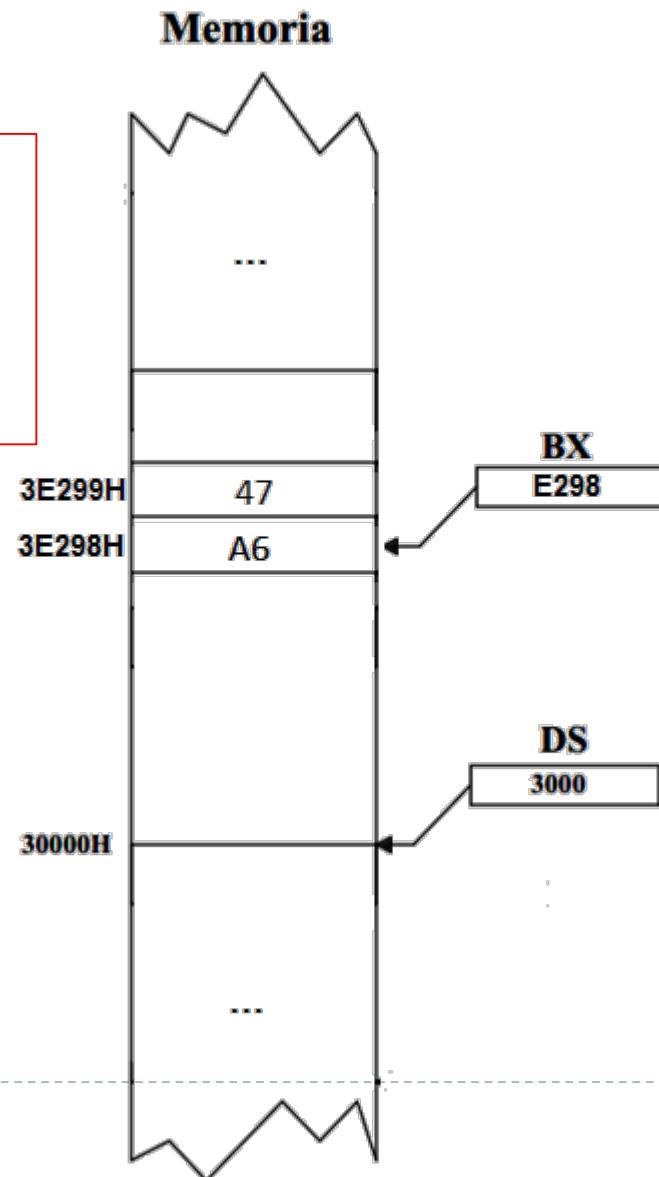


Modos de Direccionamiento

- **MOV AX,[BX]**

¿Cuál es el resultado de esta instrucción?

**¿AX = A647 ó
AX = 47A6 ?**



Endianness

Se refiere al orden en que se almacenan los bytes en memoria.

Little Endian: El byte **menos** significativo **del dato** se almacena en la posición **menos** significativa.

CH (+ significativo)

CL (- significativo)

Si CX = 30B6h, MOV [1237],CX

Entonces:



DS = 2000h

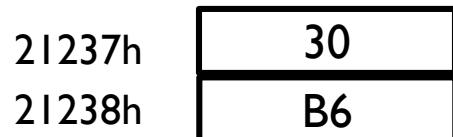


Endianness

Big Endian: El byte **más significativo** del dato se almacena en la posición **menos** significativa.

Si CX = 30B6, MOV [1237],CX

Entonces:



DS = 2000h



Endianness

Otro Ejemplo:

Dado el dato **12345678h** de 32 bits (4 bytes) que se quiere almacenar en la posición **01000h**, este valor se puede almacenar en memoria de dos distintas maneras dependiendo la arquitectura del procesador:

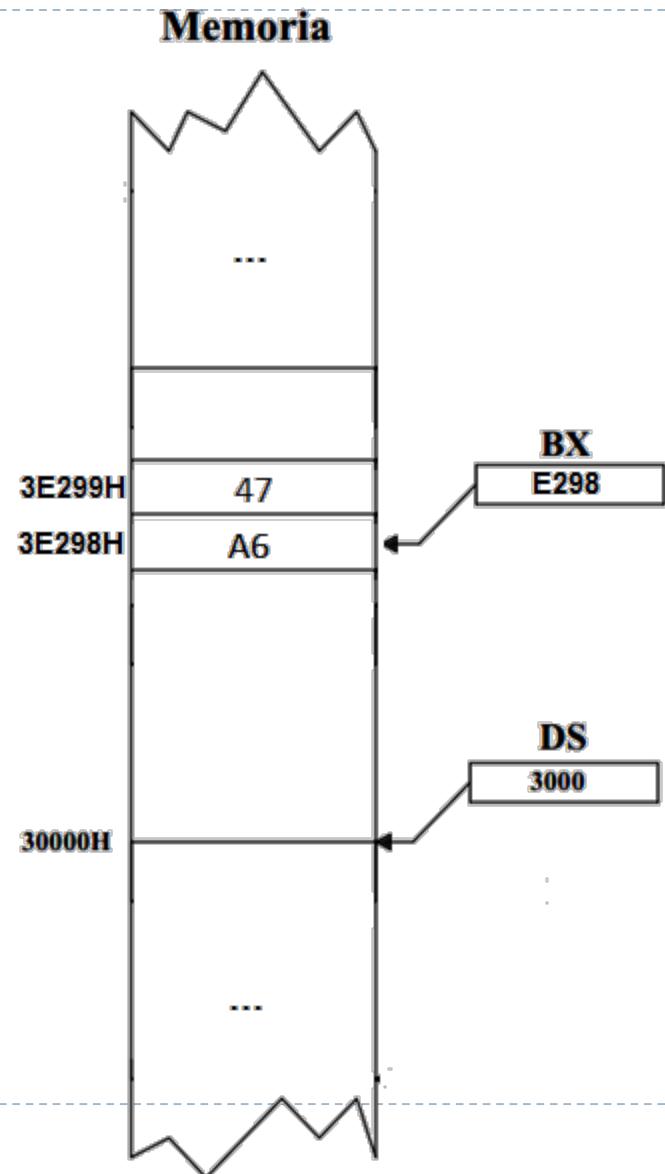
Address	00001000h	00001001h	00001002h	00001003h
Big-endian	12h	34h	56h	78h
Little-endian	78h	56h	34h	12h



Endianness

Respuesta:

El 8088 es little endian, por lo tanto el resultado de instrucción **MOV AX,[BX]** es **AX = 47A6**



Modos de Direccionamiento

Modos de direccionamiento del 8088:

1. Direccionamiento a Registro
2. Direccionamiento Inmediato
3. Direccionamiento Directo
4. Direccionamiento de Registro Indirecto
5. Direccionamiento Base mas Índice
6. Direccionamiento Relativo a Registro
7. Direccionamiento Relativo a Base mas Índice



Modos de Direccionamiento

A continuación se describe cada uno de ellos



1. Direccionamiento a Registro

Consiste en la transferencia de un byte o palabra desde un registro fuente hacia un registro destino.

Ejemplo:

AX = 18F5 BX = 0A45 CX = 3218 DX = 98E4



MOV DL,AH →

AX = 18F5 BX = 0A45 CX = 3218 DX = 98I8



1. Direccionamiento a Registro

Ejemplo:

AX = 18F5 BX = 0A45 CX = 3218 DX = 9818

MOV BX,CX →

AX = 18F5 BX = **3218** CX = **3218** DX = 9818



1. Direccionamiento a Registro

Importante:

- ▶ No se permite transferir datos de un registro de 8 a uno de 16 bits o viceversa.
- ▶ No se permite transferir datos (modificar) el valor del registro de segmento de Código (**CS**).
- ▶ No se permite transferir datos de un registro de segmento a otro registro de segmento, por ejemplo, MOV DS,ES **X**

Si es permitido transferir entre otro tipo de registro y uno de segmento (siempre y cuando no sea el registro CS). Ejemplo:

MOV DS,BP o MOV AX,SS ✓

1. Direccionamiento a Registro

Más ejemplos:

Tabla 1. Ejemplos de instrucciones de Direccionamiento a registros

<i>Lenguaje Ensamblador</i>	<i>Operación</i>
MOV AL,BL	BL → AL
MOV CH,CL	CL → CH
MOV AX,CX	CX → AX
MOV SP,BP	BP → SP
MOV DS,AX	AX → DS
MOV SI,DI	DI → SI
MOV DI,SI	SI → DI
MOV BX,ES	ES → BX
MOV CS,DS	No permitido
MOV BL,BX	No permitido



2. Direccionamiento Inmediato

Transfiere un dato de tamaño byte o palabra a un registro o localidad de memoria. En la instrucción se especifica el dato a transferir.

Ejemplo:

AX = 18F5 BX = 0A45 CX = 3218 DX = 98E4

MOV CX,89B5 →

AX = 18F5 BX = 0A45 CX = 89B5 DX = 9818

MOV AL,32 →

AX = 1832 BX = 0A45 CX = 89B5 DX = 9818



2. Direcccionamiento Inmediato

Importante:

- ▶ No se permite asignar un valor inmediato a un registro de segmento, por ejemplo, MOV DS,8B45 



2. Direccionamiento Inmediato

Más ejemplos:

Tabla 2. Ejemplos de instrucciones con Direccionamiento inmediato

<i>Lenguaje Ensamblador</i>	<i>Operación</i>
MOV AX,44	0044H → AX
MOV SI,0	0000H → SI
MOV CH, 64	64H → CH
MOV SP,3000	3000H → SP



3. Direccionamiento Directo

Transfiere un dato de tamaño byte o palabra entre un registro y una localidad de memoria del segmento de Datos, siendo el desplazamiento un valor constante.

Ejemplo:

MOV BL,[3456] → Copia el byte almacenado en $(DS \times 10h + 3456h)$, a BL

MOV AX,[1267] → Copia la palabra almacenada en $(DS \times 10h + 1267)$, al registro AX



Copia una palabra porque se está usando un registro de 16 bits (AX), si se usara uno de 8 bits se copiaría solo un byte

3. Direccionamiento Directo

Importante:

- ▶ No se permite transferir datos de una posición de memoria a otra posición de memoria directamente.

Es decir:

MOV [1234],[6789] 

Para llevar a cabo esa acción, se puede realizar:

MOV AX,[6789] 

MOV [1234],AX 

4. Direccionamiento de Registro Indirecto

Permite direccionar o apuntar a datos almacenados en cualquier localidad de memoria del segmento de Datos, de Pila o del segmento Extra, por medio de los registros BX, BP, SI y DI.

Ejemplo:

MOV AX,[BX] → Copia la palabra almacenada en DS x 10h + BX al registro AX

MOV AH,[BX] → Copia el byte almacenado en DS x 10h + BX al registro AH

MOV CX,[BP] → Copia la palabra almacenada en SS x 10h + BP al registro CX



4. Direccionamiento de Registro Indirecto

MOV byte ptr[BX], 1A → Copia el byte 0x1A a la dirección de memoria
DS x 10h + BX

MOV word ptr[BP], 1F3 → Copia la palabra 0x01F3 a la dirección de memoria SS x 10h + BP

MOV word ptr[DI], 9 → Copia la palabra 0x0009 a la dirección de memoria DS x 10h + DI



4. Direccionamiento de Registro Indirecto

Recordatorio:

"Los datos que se encuentran en cada uno de los segmentos de memoria son apuntados o indexados por los registros de punteros e índices, registro base o registro de instrucción"

Segmento de Memoria	Datos apuntados por:
Datos	BX, SI y DI
Pila	BP y SP
Extra	SI y DI



5. Direccionamiento Base mas Índice

Es similar al direccionamiento de Registro Indirecto, pero en este tipo de direccionamiento se utiliza un registro base (BX o BP) más un registro índice (DI o SI) para direccionar a memoria.

Ejemplo:

MOV DX,[BX+DI] → Copia la palabra almacenada en DS x 10h + BX + DI al registro DX

MOV CX,[BP+DI] → Copia la palabra almacenada en SS x 10h + BP + DI al registro CX

MOV [BP+SI], CX → Copia la palabra almacenada en el registro CX a la dirección de memoria SS x 10h + BP + SI

5. Direccionamiento Base mas Índice

MOV [BP+DI], CL → Copia el byte almacenado en el registro CL a la dirección de memoria SS x 10h + BP + DI

MOV [BX+SI], SP → Copia el valor del Apuntador de Pila a la dirección de memoria DS x 10h + BX + SI



5. Direccionamiento Base mas Índice

Direccionamiento de Arreglos de Datos

El mayor uso del direccionamiento base mas índice es para direccionar elementos de un arreglo en memoria.

Suponga que necesitamos direccionar los elementos en un arreglo localizado en el segmento de datos en la localidad de memoria ARREGLO.

Para cumplir esto, necesitamos cargar el registro BX con la dirección inicial del arreglo, y DI (o SI) con el número de elemento a ser accedido:

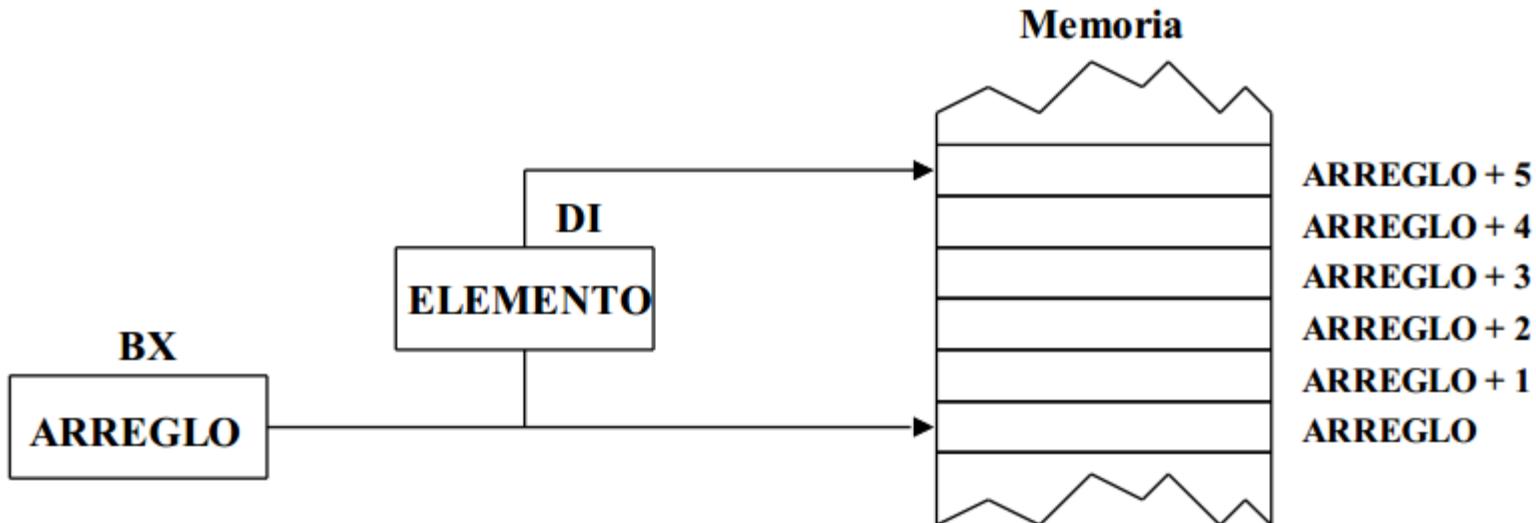
MOV BX,OFFSET ARREGLO	;direcciona el arreglo
MOV DI,5	;elemento 5
MOV AL,[BX +DI]	;toma el dato



5. Direccionamiento Base mas Índice

Direccionamiento de Arreglos de Datos

```
MOV BX,OFFSET ARREGLO ;direcciona el arreglo  
MOV DI,5 ;elemento 5  
MOV AL,[BX +DI] ;toma el dato
```



6. Direccionamiento Relativo a Registro

Es similar al direccionamiento Base más Índice, pero en este tipo de direccionamiento se utiliza un registro base o índice (BX, BP, SI, DI) más una constante.

Ejemplo:

MOV AL,[BX+8] → Copia el byte almacenado en DS x 10h + BX + 8 al registro AL

MOV [SI+F342],DX → Copia la palabra almacenada en el registro DX a la posición de memoria DS x 10h + SI + F342

MOV CH,[BP+A6] → Copia el byte almacenado en SS x 10h + BP + A6 al registro CH

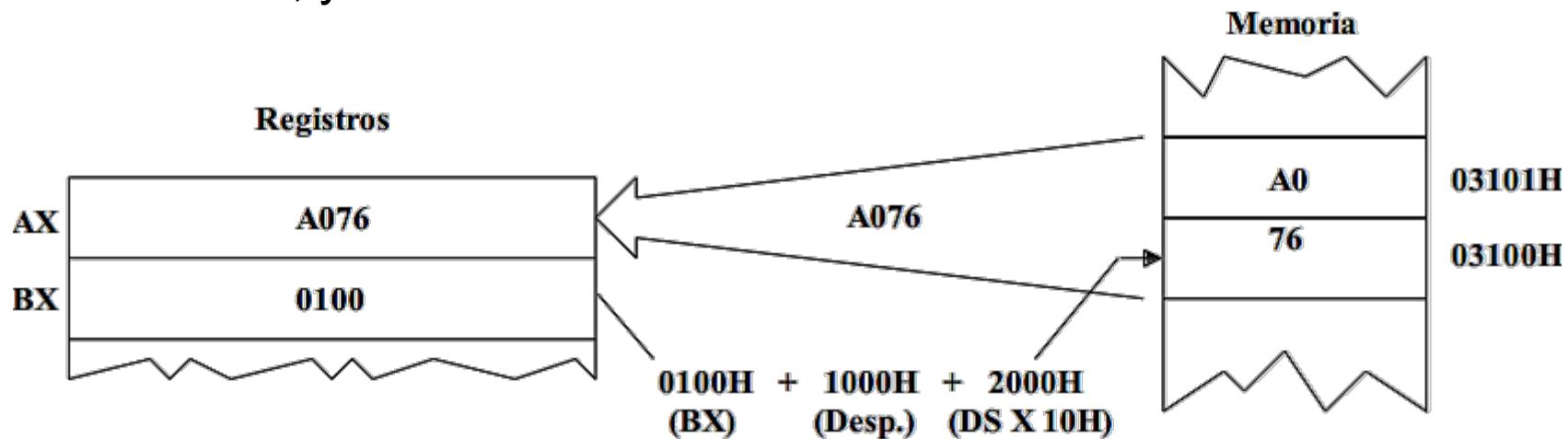


6. Direcccionamiento Relativo a Registro

Otro ejemplo:

MOV AX,[BX+1000]

Si DS = 0200H, y BX = 0100H:



6. Direccionamiento Relativo a Registro

Direccionamiento de Arreglos de Datos

Al igual que con el direccionamiento Base más Índice, con el Relativo a Registro también se pueden direccionar arreglos de datos.

Relativo a Registro:

MOV DI,5	;elemento 5
MOV AL,ARREGLO[DI]	;toma el dato

Base más Índice:

MOV BX,OFFSET ARREGLO	;direcciona el arreglo
MOV DI,5	;elemento 5
MOV AL,[BX +DI]	;toma el dato

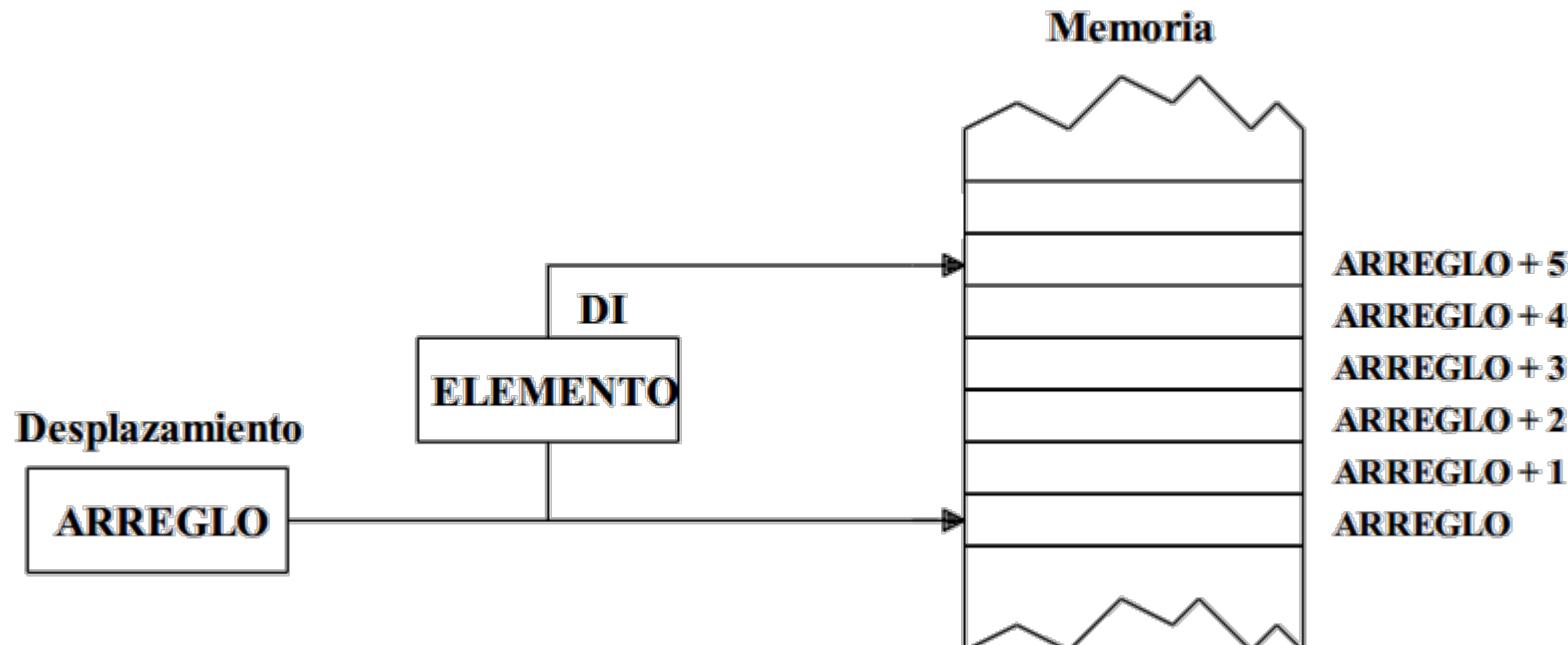


6. Direccionamiento Relativo a Registro

Direccionamiento de Arreglos de Datos

Relativo a Registro:

MOV DI,5	;elemento 5
MOV AL,ARREGLO[DI]	;toma el dato



7. Direccionamiento Relativo a Base mas Índice

Es similar al modo de direccionamiento Base más Índice, pero agrega un desplazamiento más.

Es el modo de direccionamiento menos usado en un programa, y el más complejo.

Frecuentemente direcciona un arreglo de dos dimensiones de datos en memoria.

Ejemplo:

MOV AX,[BX+SI+10] → Copia la palabra almacenada en $(DS \times 10h + BX + SI + 10h)$, al registro AX

MOV [BP+SI+F5],BL → Copia el byte almacenado en el registro BL a la posición de memoria $(SS \times 10h + BP + SI + F5h)$

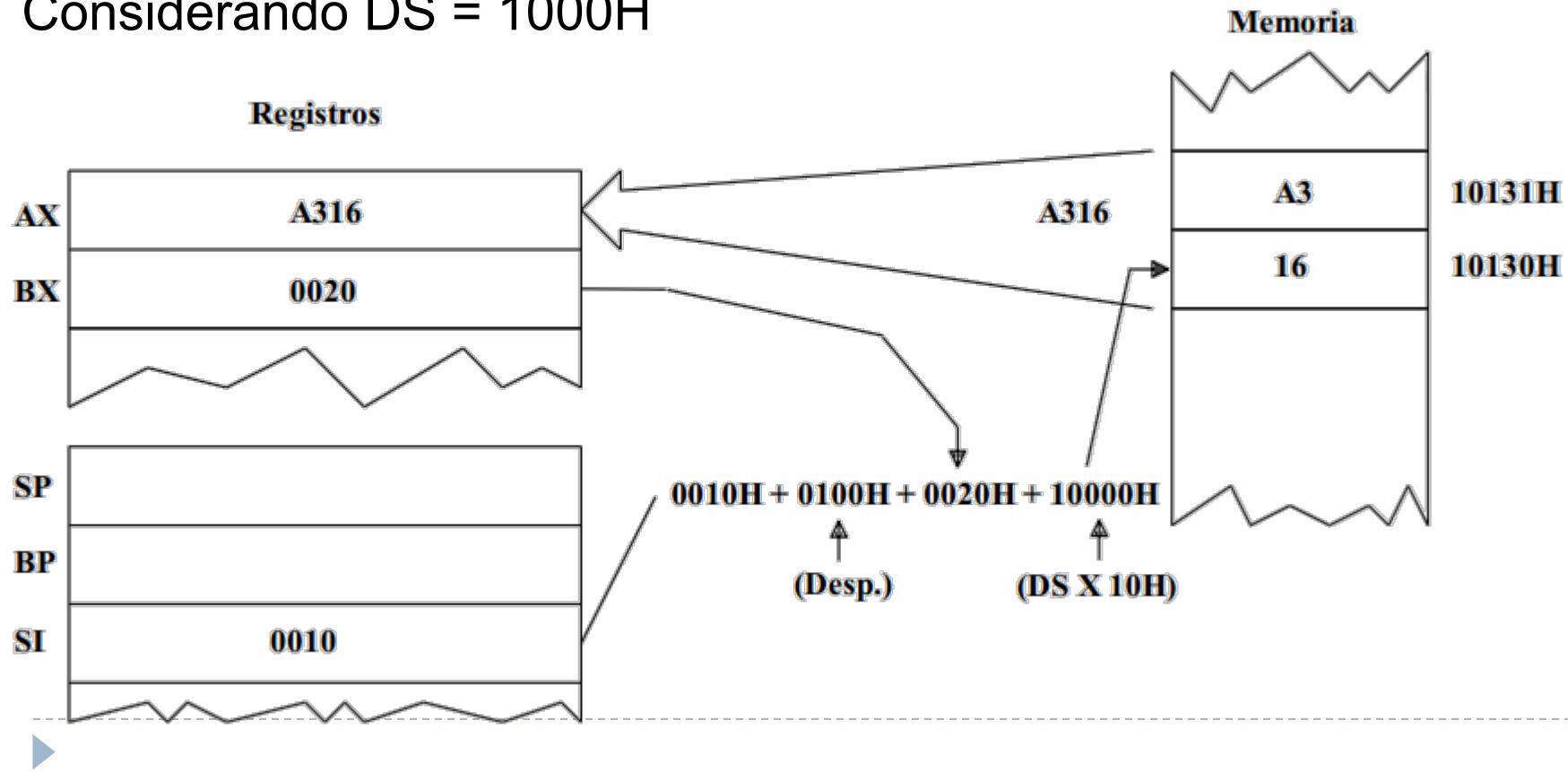


7. Direccionamiento Relativo a Base mas Índice

Ejemplo:

MOV AX,[BX+SI+100]

Considerando DS = 1000H



Resumen de Modos de Direccionamiento



Resumen

		<i>Fuente</i>	<i>Generación de Dirección</i>	<i>Destino</i>
Registro	MOV AX,BX	BX Registro		AX Registro
Inmediato	MOV BL,3AH	3AH Dato		BL Registro
Directo	MOV[1234] ,AX	AX Registro	(Desp) + (DS X 10H) 1234H + 10000H	11234H Memoria
Registro Indirecto	MOV [BX],AX	AX Registro	(BX) + (DS X 10H) 0300H + 10000H	10300H Memoria
Base mas Indice	MOV [BX+SI],AX	AX Registro	(BX) + (SI) + (DS X 10H) 0300H + 0200H + 10000H	10500H Memoria
Relativo a Registro	MOV [BX+4],AX	AX Registro	(BX) + 4 + (DS X 10H) 0300H + 4 + 10000H	10304H Memoria
Relativo a Base mas Indice	MOV ARRAY[BX+SI],AX	AX Registro	(BX) + ARRAY + (SI) + (DS X 10H) 0300H + 1000H + 0200H + 10000H	11500H Memoria

