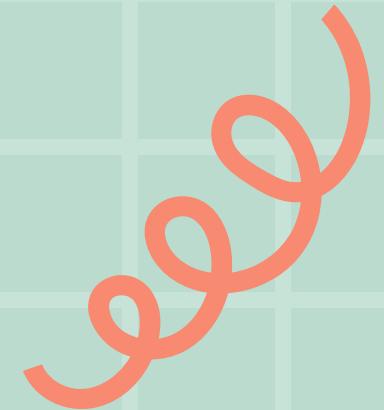
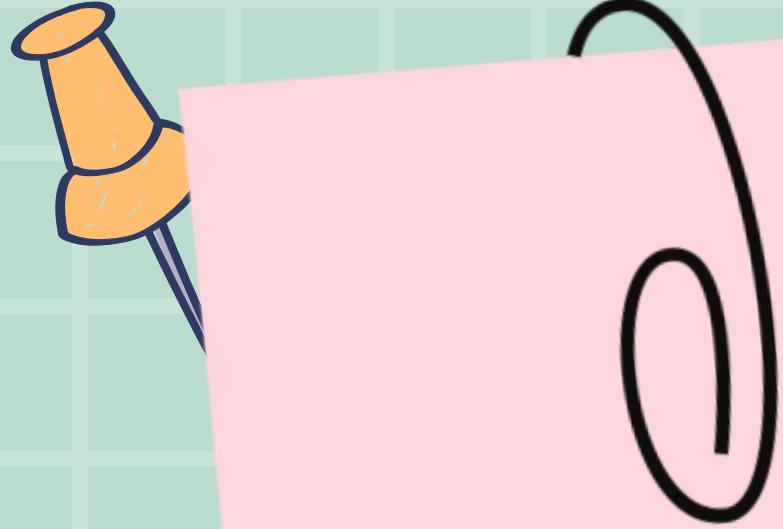


Manual de usuario para ensamblador

la mente de una computadora



Introducción

El lenguaje ensamblador es conocido por ser un lenguaje de bajo nivel el cual consiste en un conjunto de mnemónicos los cuales son representaciones de instrucciones básicas para microcontroladores, microprocesadores, computadoras, entre otros. Este lenguaje fue utilizado en los inicios del desarrollo del software y actualmente se utiliza en ámbitos académicos e investigaciones, o cuando se necesita manipular directamente al hardware o alto rendimiento. Además, también se utiliza para el desarrollo de controladores de dispositivo y en el desarrollo de sistemas operativos.

Es necesario tener en cuenta que el lenguaje ensamblador es importante ya que con él podemos escribir programas embebidos (con ellos se ejecutan tareas de control) ya que estos no requieren mucha memoria. Además, aún hay muchos dispositivos, como los microprocesadores, que solo se pueden manipular con lenguaje ensamblador.





Por otro lado, existen diversos tipos de lenguaje ensamblador, los cuales son:

- Ensamblador de una frase: Se leen en una línea del programa fuente y luego se traducen para producir una instrucción en lenguaje maquina o solo se ejecuta si es una pseudoinstrucción. Además, se construye la tabla de símbolos la cual trae las variables, etiquetas, etc.
- Ensamblador de dos frases: Esta se divide en dos etapas. En la primera se lee el programa fuente y se construye la tabla de signos. Y en la segunda lee nuevamente el programa fuente y va traduciendo en su totalidad. Actualmente estos son los más utilizados
- Ensambladores cruzados: Se utilizan en las computadoras que poseen un procesador diferente al que tiene la computadora en donde se va a ejecutar el programa objeto producido.
- Ensambladores residentes: Estos permanecen en la memoria principal de la computadora y cargan la ejecución del programa objeto producido. Su ventaja es que se puede comprobar de inmediato y programar sin necesidad de transportarlo a algún otro lugar.
- Macroensambladores: Estos permiten el uso de macros debido a que su potencia son programas robustos que ya no permanecen en la memoria una vez que se genera el programa objeto.
- Micro ensambladores: los procesadores que se utilizan en las computadoras tienen un repertorio fijo de instrucciones, ósea que el interprete de estas interpreta un determinado código de operación. El programa que indica el intérprete en las instrucciones de CPU, que es como debe actuar se llama micropograma y

Objetivo

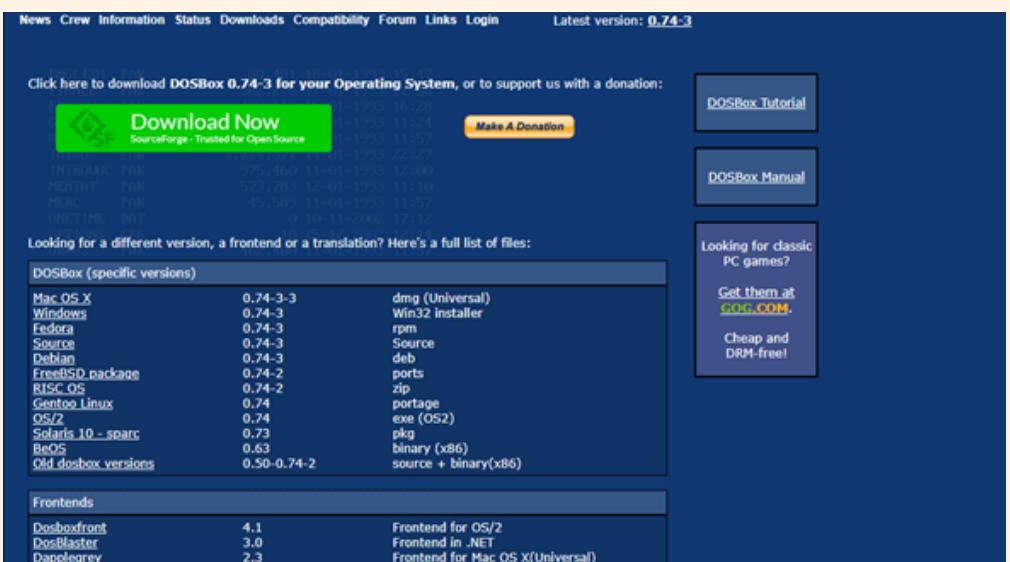
Debido a que el lenguaje ensamblador es un programa que permite llevar códigos de operación en forma de instrucciones que se pueden entender por el programador el objetivo es aprender este lenguaje, de esa manera se podrá contar con la capacidad de escribir programas de bajo nivel, así se tendrá una representación más directa con el código máquina.



Instrucciones de instalación del programa



- I. Ingresar al siguiente enlace:
<https://www.dosbox.com/download.php?main=1>



Le daremos click en el botón verde en donde dice Download Now y esperaremos hasta que el programa se instale en nuestras computadoras. Y una vez instalada se deberá ver de esta manera:



¡Vamos a ejecutar nuestro primer programa!



Se ejecutará el programa que va a imprimir el famoso "Hola Mundo". Para ello debemos:

- Abrir nuestro blog de notas
- Escribir el código que está en la siguiente diapositiva
- Vamos a abrir nuestros archivos y en el disco duro vamos a crear una carpeta con el nombre "Archivos_Arky"
- Dentro de esa carpeta se debe crear otra carpeta que se llame "arky"
- Dentro de la carpeta arky se debe crear otra carpeta que se llame "TASM"
- Dentro de TASM debe estar estos archivos:

📁	2sem21	27/10/2021 22:05	Carpeta de archivos
RAR	Programas-del-TASM (1) (1)	3/8/2021 22:47	Archivo WinRAR 342 KB
exe	TASM	27/10/1992 07:00	Aplicación 131 KB
exe	TD	27/10/1992 07:00	Aplicación 482 KB
exe	TLINK	29/10/1990 02:01	Aplicación 53 KB





datos segment

Rotulo db "Hola Mundo!\$"

datos endS

```
pila segment stack 'stack'  
dw 256 dup(?)  
pila endS
```

codigo segment

```
main: mov ax, pila  
      mov ss, ax
```

```
      mov ax, datos  
      mov ds, ax
```

```
      mov ah, 09h  
      lea dx, rotulo
```

int 21h

```
; mov ah, 4Ch  
; xor al, al  
; mov al, 0
```

```
      mov ax, 4C00h
```

int 21h

codigo ends

end main



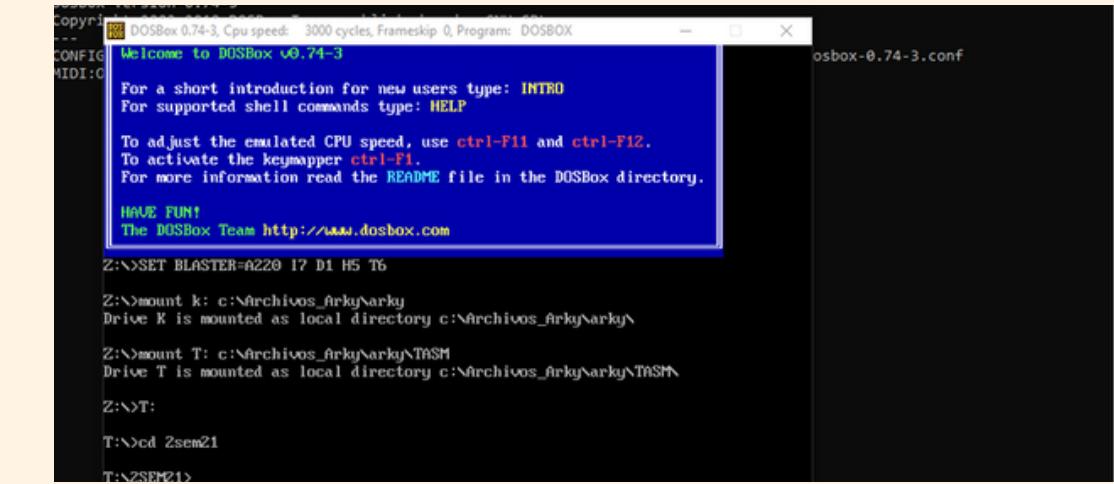
Por ultimo...

7. Vamos a ir de nuevo al blog de notas y vamos a guardar el archivo en la carpeta de 2sem2l y le vamos a poner el nombre de hmuno.asm



Ahora...

Abrimos la aplicación DOSBox



```
Copyrig... DOSBox 0.74-3. Cpu speed: 3000 cycles, Frameskip: 0, Program: DOSBOX
...
CONFIG Welcome to DOSBox v0.74-3
MIDI:
For a short introduction for new users type: INTRO
To supported shell commands type: HELP

To adjust the emulated CPU speed, use ctrl-F11 and ctrl-F12.
To activate the keymapper ctrl-F1.
For more information read the README file in the DOSBox directory.

HOW FUN!
The DOSBox Team http://www.dosbox.com

Z:\>SET BLASTER=A220 I7 D1 H5 T6
Z:\>mount k: c:\Archivos_Arkyarky
Drive K is mounted as local directory c:\Archivos_Arkyarky
Z:\>mount T: c:\Archivos_Arkyarky\NTASM
Drive T is mounted as local directory c:\Archivos_Arkyarky\NTASM
Z:\>T:
T:\>cd Zsem21
T:\>ZSEM21>
```

Y vamos a escribir: \tasm hmundo y le damos enter

Luego vamos a escribir: \tlink hmundo, y le damos enter

Por ultimo vamos a escribir: hmundo, y le damos enter

Y así vamos a ver en la pantalla de la consola el "Hola mundo"

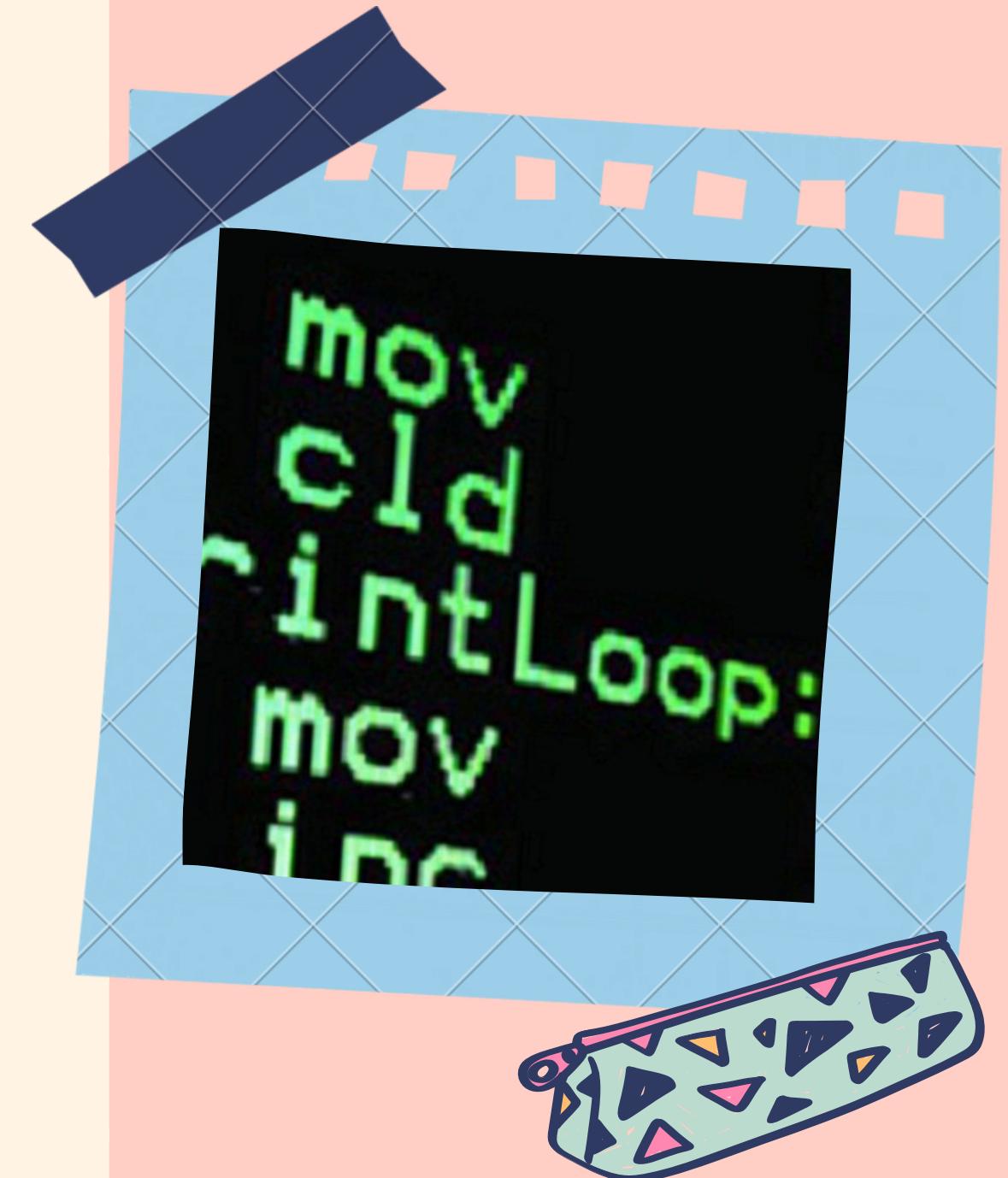




¿Qué debemos saber para empezar a programar?

Lo primero que debemos saber es que nuestro programa se divide en 3 partes:

- datos segment: en este se guardaran las etiquetas. En las etiquetas se guardaran los mensajes que se deseen enviar
- pila segment stack 'stack': es una sección de la memoria que se usa durante la ejecución, se usa con frecuencia para almacenar parámetros de funciones.
- código segment: aquí es donde empezaremos a ejecutar nuestro código





Directivas

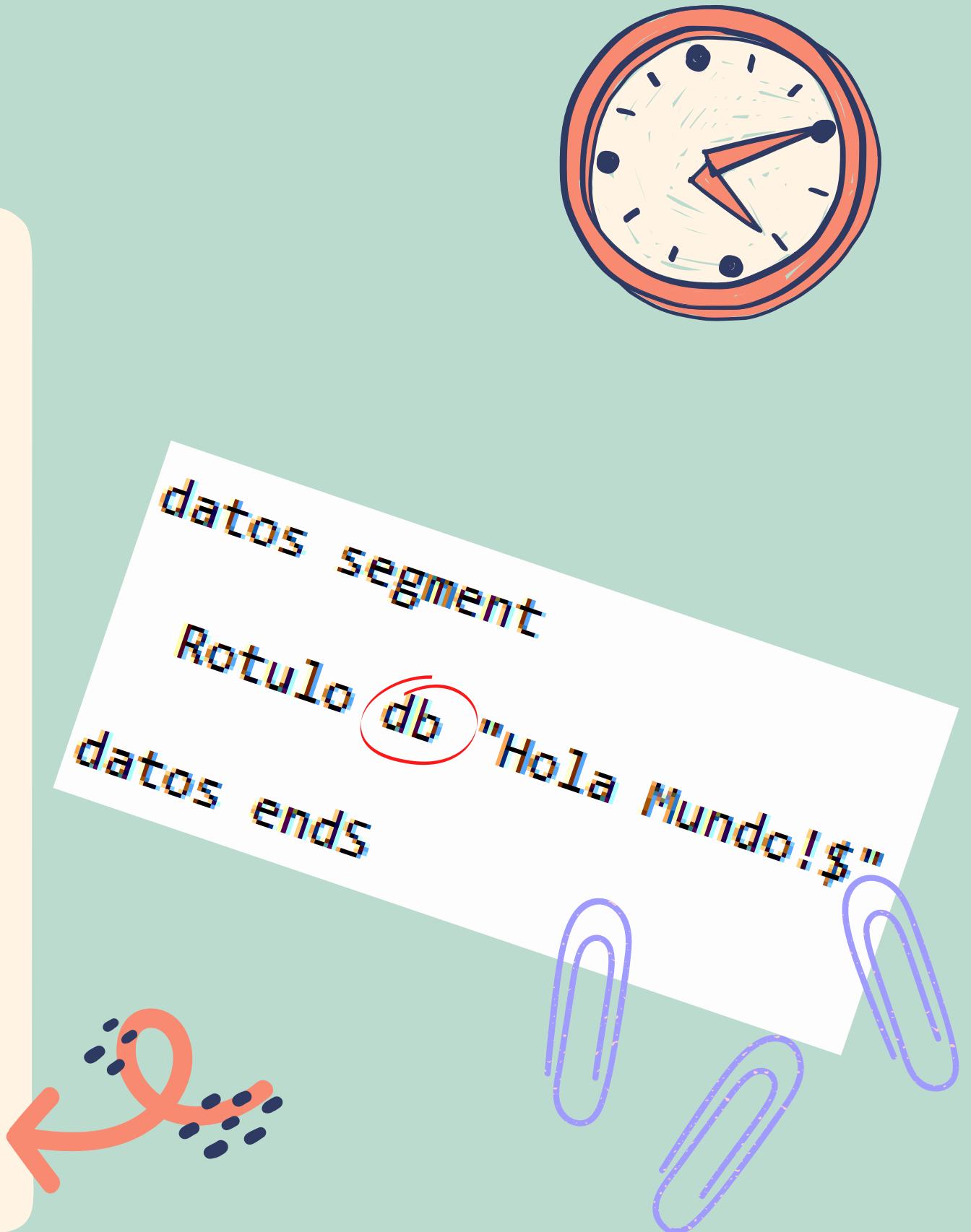
Se utilizan para definir segmentos, símbolos, procedimientos o subrutinas, reservar memoria, etc

Existen diversos tipos de directivas:

- DB (Define Byte): Reservar datos de tamaño byte (8 bits)
- DW (Define Word): Reservar datos de tipo palabra (16 bits)
- DD (Define Doubleword): Reservar datos de tipo doble palabra (32 bits)
- DQ (Define Quadword): Reservar datos de tipo quádruple palabra (64 bits)
- DT (Define Terabyte): Reservar datos de tipo terabyte (80 bits o 10 bytes)

También está la directiva segment:

Las directivas SEGMENT y ENDS marcan el principio y el final del segmento cuyo nombre se especifica.



REGISTROS INTERNAOS DE LA UNIDAD CENTRAL DE PROCESO (UCP):

Registros de datos:

1

AX: Registro acumulador. Es el principal empleado en las operaciones aritméticas.

2

BX: Registro base. Se usa para indicar un desplazamiento.

3

CX: Registro contador. Se usa como contador en los bucles.

4

DX: Registro de datos. También se usa en las operaciones aritméticas.

```
main: mov ax, pila
      mov ss, ax

      mov ax, datos
      mov ds, ax

      mov ah, 09h
      lea dx, rotulo

      int 21h

;     mov ah, 4Ch
;     xor al, al      ; mov al, 0

      mov ax, 4C00h

      int 21h
```



REGISTROS INTERNOS DE LA UNIDAD CENTRAL DE PROCESO (UCP):

- Registros de segmentos:

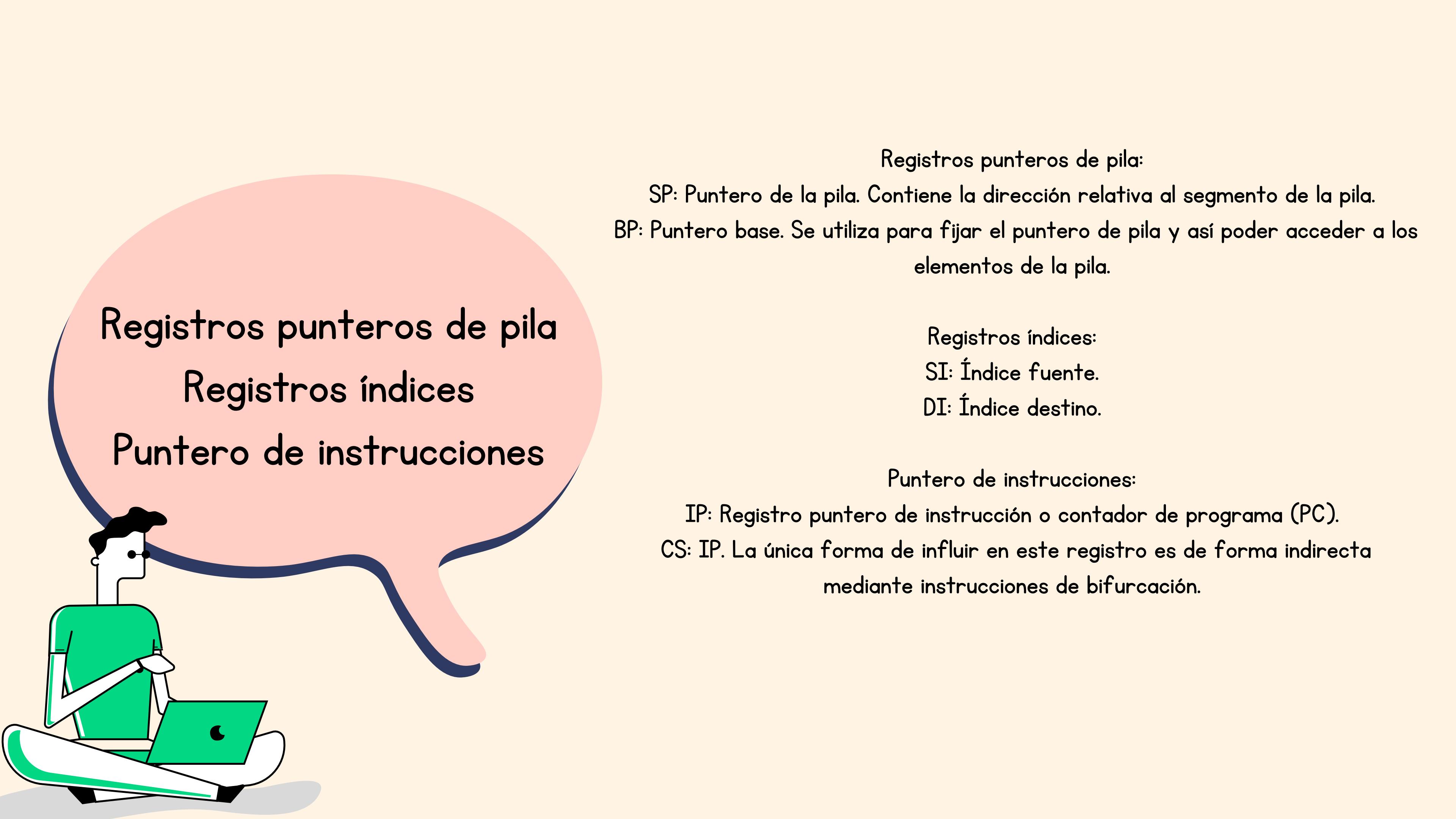
- CS: Registro de segmento de código.

- Contiene la dirección de las instrucciones del programa.

- DS: Registro segmento de datos. Contiene la dirección del área de memoria donde se encuentran los datos del programa.

- SS: Registro segmento de pila. Contiene la dirección del segmento de pila. La pila es un espacio de memoria temporal que se usa para almacenar valores de 16 bits (palabras).

- ES: Registro segmento extra. Contiene la dirección del segmento extra.



Registros punteros de pila

Registros índices

Puntero de instrucciones

Registros punteros de pila:

SP: Puntero de la pila. Contiene la dirección relativa al segmento de la pila.

BP: Puntero base. Se utiliza para fijar el puntero de pila y así poder acceder a los elementos de la pila.

Registros índices:

SI: Índice fuente.

DI: Índice destino.

Puntero de instrucciones:

IP: Registro puntero de instrucción o contador de programa (PC).

CS: IP. La única forma de influir en este registro es de forma indirecta mediante instrucciones de bifurcación.

Registro de banderas (flags)

Cada bandera es un bit y se usa para registrar la información de estado y de control de las operaciones del microprocesador.

Banderas de estado:

CF: Bandera de acareo.

OF: Bandera de desbordamiento (aritmético).

ZF: Bandera de resultado 0 o comparación igual.

SF: Bandera de resultado o comparación negativa.

PF: Bandera de paridad (número par de bits).

AF: Bandera auxiliar. Indica si hay necesidad de ajuste en las operaciones aritméticas con números BCD.

Banderas de control:

DF: Bandera de dirección. Controla la dirección de las operaciones con cadenas de caracteres incrementando o decrementando automáticamente los registros índices (SI y DI) **IF:** Bandera de interrupciones. Indica si están permitidas o no las interrupciones de los dispositivos externos. 3

TF: Bandera de atrape. Controla la operación de modo paso a paso

SEGMENTOS Y REGISTROS ASOCIADOS

Se usan para manejar la información, el tamaño de estos segmentos es de 64kb

El ensamblador accede a esta información tomando en cuenta la localización del segmento, dada por los registros DS, ES, SS y CS.

La forma de indicarle al ensamblador con cuales de los segmentos se va a trabajar es por medio de las directivas .CODE, .DATA y .STACK

```
mov ax, datos  
mov ds, ax  
mov ah, 09h  
lea dx, rotulo
```

Un programa consta de 4 tipos de segmentos.

Segmento código: Cada instrucción se direcciona mediante el registro segmento de código y el registro de desplazamiento.

Segmento de datos: Los datos se direccionan mediante el registro de segmento de dato y un registro de desplazamiento.

Segmento de pila: Los datos se direccionan mediante el registro segmento de pila y un registro de desplazamiento.

Segmento extra: Sustituye el registro de segmento, por ejemplo ES:BX.



TÉCNICAS DE CODIFICACIÓN EN ENSAMBLADOR

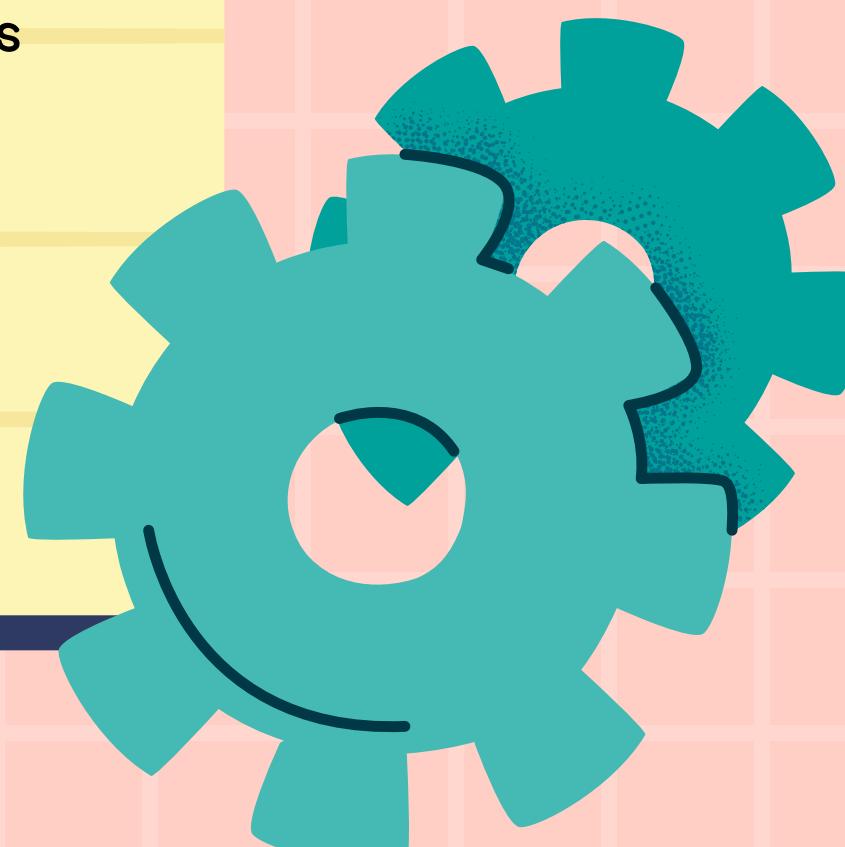
BUCLLES: Inicialización de un registro al nº de vueltas del bucle

SALTOS CONDICIONALES: compara la condición y si se cumple saltar a la etiqueta prefijada saltar fuera del condicional

Etiqueta: instrucción del bucle que decrementa el contador y salta a la etiqueta si el contador es mayor que cero

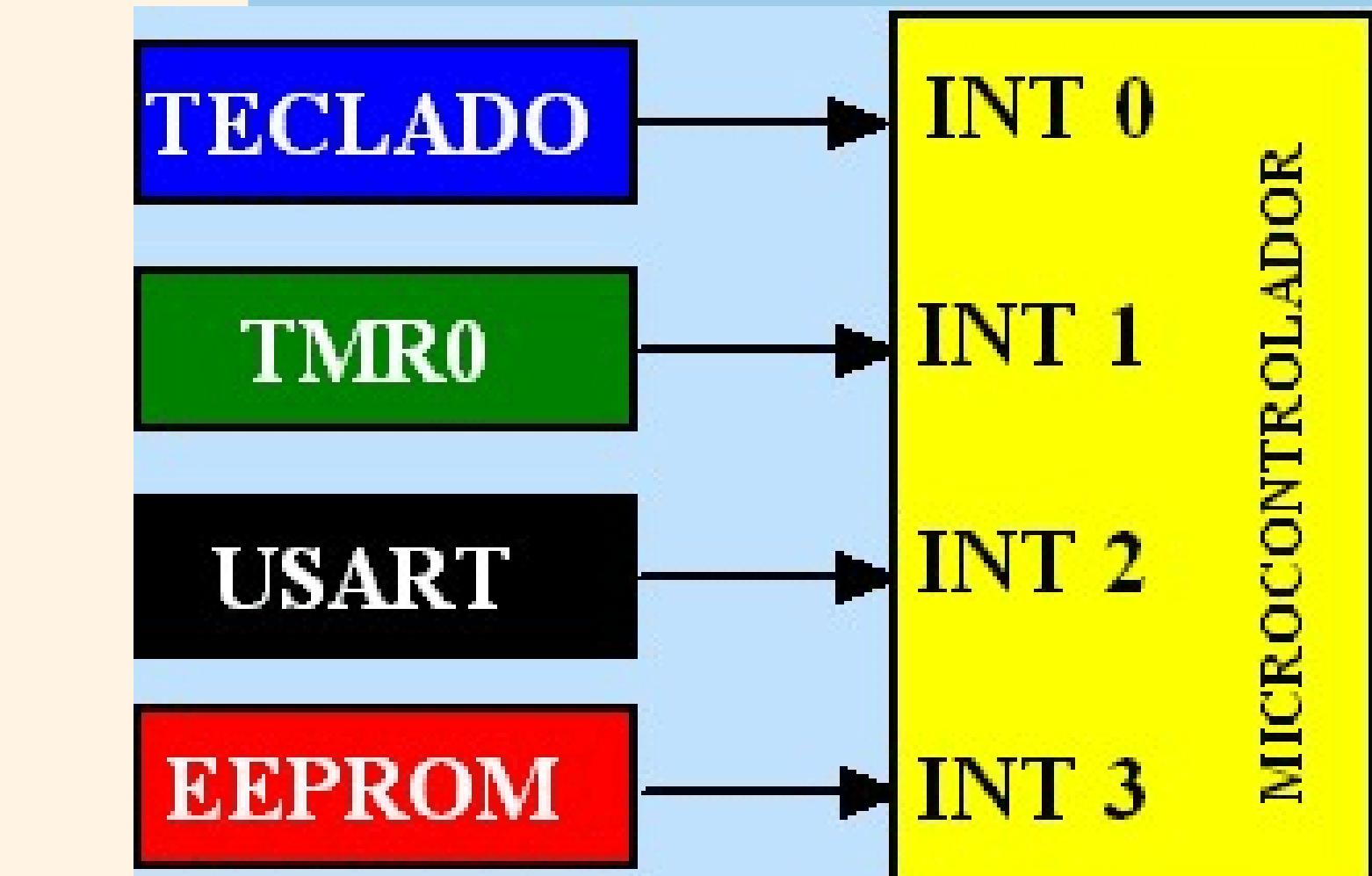
Declaración: PROC RET ENDP

Llamada: CALL



Interrupciones

Estas determinan la ejecución de un programa para permitir el uso de la UCP a un proceso prioritario.

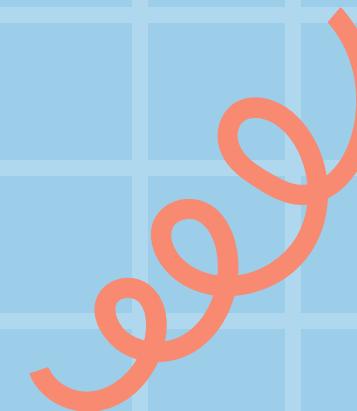


SALTOS, CICLOS Y PROCEDIMIENTOS

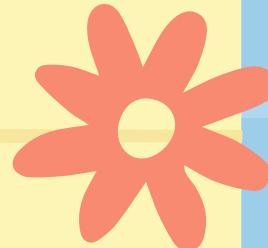


• TEST	verifica
• CMP	compara
• JMP	salta
• JE, JZ	salta si es <i>igual</i> a cero
• JNE, JNZ	salta si <i>no igual</i> a cero
• JS	salta si <i>signo negativo</i>
• JNS	salta si <i>signo no negativo</i>
• JP, JPE	salta si <i>paridad par</i>
• JNP, JOP	salta si <i>paridad impar</i>
• JO	salta si <i>hay capacidad excedida</i>
• JNO	salta si <i>no hay capacidad excedida</i>
• JB, JNAE	salta si <i>por abajo</i> (no encima o igual)
• JNB, JAE	salta si <i>no está por abajo</i> (encima o igual)
• JBE, JNA	salta si <i>por abajo o igual</i> (no encima)
• JNBE, JA	salta si <i>no por abajo o igual</i> (encima)
• JL, JNGE	salta si <i>menor que</i> (no mayor o igual)
• JNL, JGE	salta si <i>no menor que</i> (mayor o igual)
• JLE, JNG	salta si <i>menor que o igual</i> (no mayor)
• JNLE, JG	salta si <i>no menor que o igual</i> (mayor)





Instrucción mov:



Transferencia de datos entre celdas de memoria, registros y acumulador.

Donde Destino es el lugar a donde se moverán los datos y fuente es el lugar donde se encuentran dichos datos. Los diferentes movimientos de datos permitidos para esta instrucción son:

Destino: registro. Fuente: registro

Destino: registro. Fuente: memoria

Destino: memoria. Fuente: registro

Destino: registro. Fuente: dato inmediato

Destino: memoria. Fuente: dato inmediato

Destino: memoria. Fuente: acumulador

Destino: acumulador. Fuente: memoria

Destino: registro de segmento. Fuente:
memoria/registro Destino: memoria/registro. Fuente:
registro de segmento

MOV AX,0006h

MOV BX,AX

MOV AH,4Ch

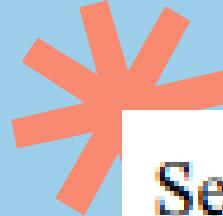
INT 21H

OPERACIONES ARITMETICA



ADD	Suma Acumulador
ADDC	Suma con Acarreo
DA	Ajuste decimal del acumulador
DEC	Decremento del operando
DIV	División Acumulador por B
INC	Incrementa la variable
MUL	Multiplicación Acumulador por B
SUBB	Resta con acarreo

OPERACIONES LOGICAS



Se usan para realizar operaciones a nivel de bits.

- **NOT operando:** cambia los bits 1 por 0 y viceversa y devuelve el resultado en el mismo operando.

AL = F2h AL 1111 0010

NOT AL; NOT AL 0000 1101 = Odh

- **OR destino, fuente:** operación *o lógico inclusivo*. El resultado se almacena en destino.

AX = FEDCh= 1111 1110 1101 1100

BX = 1234 h = 0001 0010 0011 0100

OR AX, BX 1111 1110 1111 1100 = FEFC h

- **AND destino, fuente:** la operación *Y lógica* entre 2 operandos, el resultado se deja en destino.

AX = FEDC h 1111 1110 1101 1100

BX = 1234 h 0001 0010 0011 0100

ADD AX, BX 0001 0010 0001 0100 = 1214 h

- **XOR destino, fuente:** la operación *o lógico exclusiva*; el resultado se deja en destino.

AX = FEDC h 1111 1110 1101 1100

BX = 1234 h 0001 0010 0011 0100

XOR AX, BX 1110 1100 1110 1000 = ECE8 h



INTERRUPCIONES, DIRECCIÓN Y FUNCIÓN



DIRECCION (hex)	INTERRUPCION (hex)	FUNCION
0-3	0	Division by zero
4-7	1	Single step trace
8-B	2	Non maskable interrupt
C-F	3	Break point instruction
10-13	4	Overflow
14-17	5	Print screen
18-1F	6,7	Reserved
20-23	8	Timer
24-27	9	Keyboard interrupt
28-37	A,B,C,D	Reserved
38-3B	E	Diskette interrupt
3C-3F	F	Reserved
40-43	10	Video screen I/O

44-47	11	Equipment check
48-4B	12	Memory size check
4C-4F	13	Diskette I/O
50-53	14	Communication I/O
54-57	15	Cassette I/O
58-5B	16	Keyboard input
5C-5F	17	Printer Output
60-63	18	ROM Basic entry code
64-67	19	Bootstrap loader
68-6B	1A	Time of day
6C-6F	1B	Get control on keyboard break
70-73	1C	Get control on timer interrupt
74-77	1D	Pointer to video initialization table
78-7B	1E	Pointer to diskette parameter table
7C-7F	1F	Pointer to table for graphics characters
ASCII 128-255		

80-83	20	DOS program terminate
84-87	21	DOS function call
88-8B	22	DOS terminate address
90-93	24	DOS fatal error vector
94-97	25	DOS absolute disk read
98-9B	26	DOS absolute disk write
9C-9F	27	DOS terminate, fix in storage
A0-FF	28-3F	Reserved for DOS
100-1FF	40-7F	Not used
200-217	80-85	Reserved by BASIC
218-3C3	86-F0	Used by BASIC interpreter
3C4-3FF	F1-FF	Not Used



A large grey rectangular note is pinned to a light blue wall with a yellow pushpin. The note has a light blue border and contains the text "GRACIAS POR SU ATENCIÓN" in bold black letters. A small black string hangs from the top edge of the note.

GRACIAS POR
SU ATENCIÓN

