

Programación en Ensamblador
Ing. Marcelo Tosini - 2001

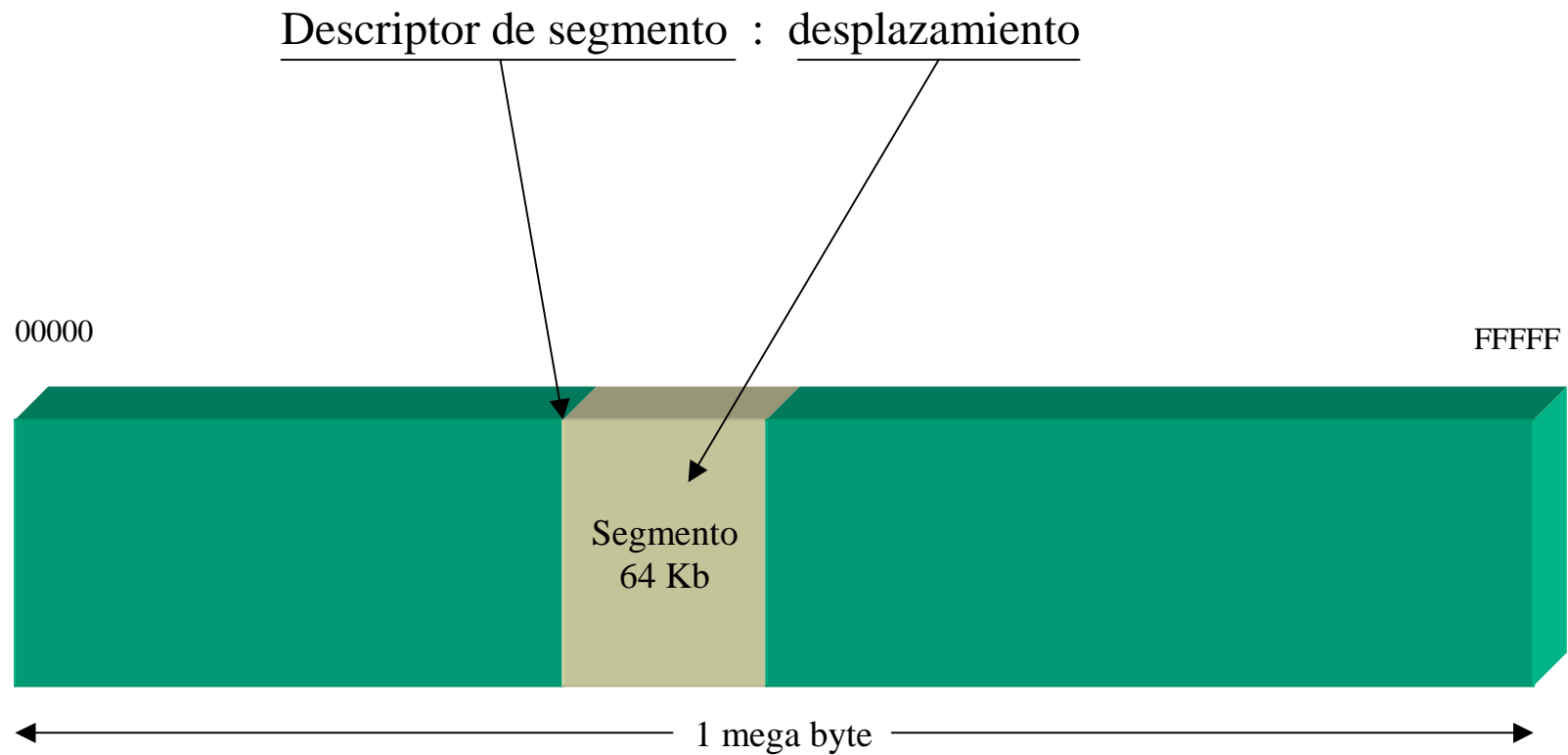
Características generales

- Procesador de 16 bits
 - Bus de direcciones de 20 bits : 1 Mbyte
 - Bus de datos interno de 16 bits
 - Bus de datos externo de
 - 16 bits en el 8086
 - 8 bits en el 8088
- Original del IBM PC/XT
- 89 instrucciones
- No tiene coprocesador

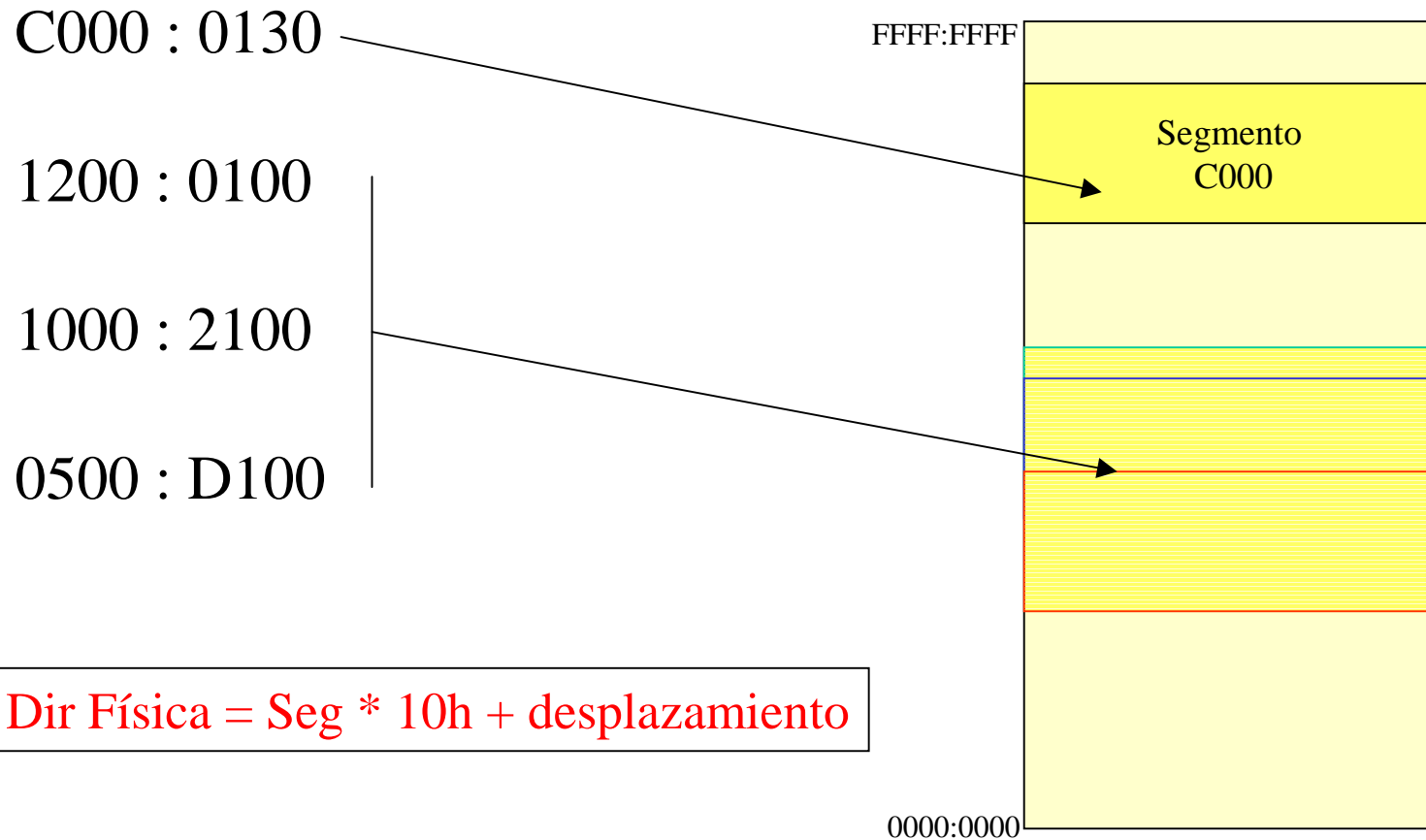
Tipos de datos

- ASCII
- BCD
- Entero sin signo
 - 8 bits 0..255
 - 16 bits 0..65535
- Entero con signo
 - 8 bits -128..127
 - 16 bits -32768..32767
- Cadenas secuencia de bytes o palabras

Manejo de memoria



Manejo de memoria (ejemplos)

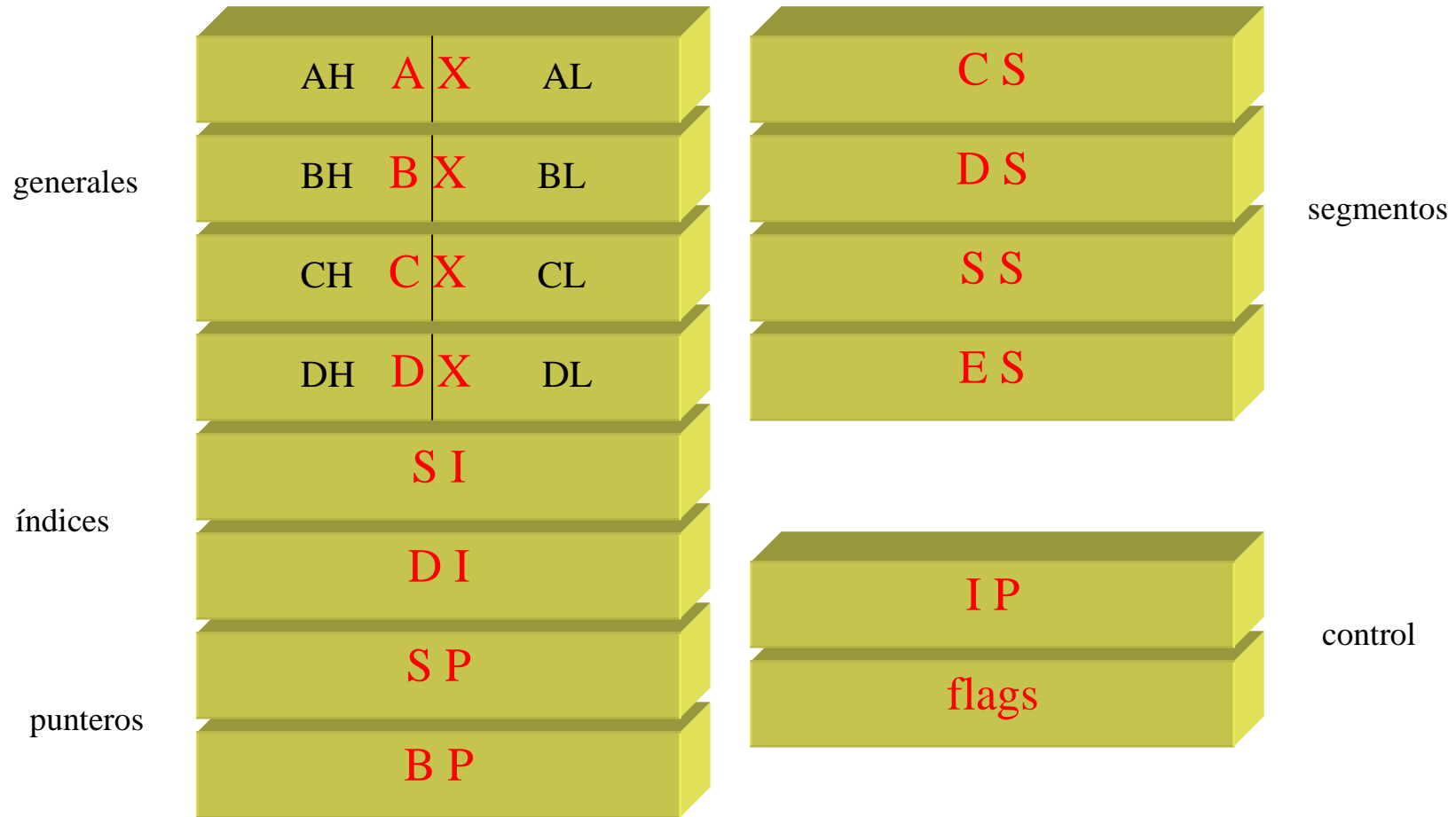


Juego de registros

14 registros de 16 bits

- 4 generales: **AX, BX, CX, DX**
- 2 índices: **SI, DI**
- 2 punteros: **SP, BP**
- 4 segmentos: **DS, CS, ES, SS**
- 1 estado: **Flag**
- 1 contador de programa: **IP**

Juego de registros



Registro de flags



C : acarreo en la suma y arrastre en la resta

P : paridad del dato (0, impar y 1, par)

A : acarreo auxiliar. Indica el acarreo o arrastre entre los bits 3 y 4

Z : indicación de resultado igual a cero

S : indicador de signo del resultado. 0, positivo y 1, negativo

T : trampa. Habilita la característica de depuración del procesador

I : habilitación de interrupciones de hardware

D : selección de incremento o decremento en los índices

O : sobreflujo.

Modos de direccionamiento

7 modos de direccionamiento básicos

- ✓ **Modo registro:** el operando es un registro.
- ✓ **Modo inmediato:** el operando es una constante.
- ✓ **Modo directo:** el operando es una dirección efectiva (explícita).
- ✓ **Modo registro indirecto:** similar al anterior pero la dirección efectiva está contenida en un registro (BX, BP, SI, DI).
- ✓ **Modo relativo a base:** la dirección efectiva se encuentra sumando un desplazamiento a BX o BP.
- ✓ **Modo indexado directo:** igual al anterior usando SI o DI.
- ✓ **Modo indexado a base:** combinación de los dos anteriores. La dirección efectiva se calcula como la suma de un registro base, un registro índice y, opcionalmente, un decalaje o desplazamiento.

Modos de direccionamiento

Ejemplo de uso

- ✓ **Modo registro:** `add ax, bx`
- ✓ **Modo inmediato:** `add ax, 5`
- ✓ **Modo directo:** `add ax, [100]`
- ✓ **Modo registro indirecto:** `add ax, [bx]`
- ✓ **Modo relativo a base:** `add ax, [bp + 100]`
- ✓ **Modo indexado directo:** operaciones de cadena : `movsb`
- ✓ **Modo indexado a base:** `add ax, [bx + si + 100]`

Modos de direccionamiento

En general:

BASE + **INDICE** + **DESPLAZAM**

Ninguno Ninguno Ninguno

BX o BP + SI o DI + 8 bits

BX o BP + SI o DI + 16 bits

Juego de instrucciones

- **Cero operandos:** trabajan sobre algún operando explícito, puede ser un registro o un flag

CLC **pone carry en 0**

- **Un operando:** el único operando es fuente y destino de la operación

INC AX **AX := AX + 1**

- **Dos operandos:** el primer operando es destino de la operación entre los dos operandos

ADD AX, BX **AX := AX + BX**

Grupos de instrucciones

- **Transferencia de datos (14):** movimiento de datos entre registros y/o memoria
- **Aritméticas (20):** operaciones aritméticas de enteros
- **Manipulación de bits (10):** operaciones lógicas
- **Cadenas (5):** movimiento, búsqueda y comparación de cadenas de datos
- **Transferencia de programa (29):** saltos, llamadas...
- **Control del procesador (11):** detención, depuración, IRQs,...

Transferencia de datos

IN	carga el acumulador desde un dispositivo de I/O
LAHF	carga los flags en AH
LEA	carga una dirección efectiva
LDS	carga DS y un registro de 16 bits con datos de memoria de 32 bits
LES	carga ES y un registro de 16 bits con datos de memoria de 32 bits
MOV	carga byte o palabra o doble palabra
OUT	saca datos del acumulador a un puerto de I/O
POP	recupera una palabra de la pila
POPF	recupera los flags de la pila
PUSH	almacena una palabra en la pila
PUSHF	almacena los flags en la pila
SAHF	carga AH en los flags
XCHG	intercambia bytes o palabras
XLAT	emplea AL para entrar a una tabla de conversión

Instrucciones aritméticas

AAA, AAD, AAM, AAS	ajuste ASCII para suma, división, producto y resta
ADD	suma datos entre registros o la memoria y otro registro
ADC	suma con acarreo
CBW	convierte byte a palabra
CMP	compara los datos
CWD	convierte palabra a doble palabra
DAA, DAS	ajuste decimal en AL para una suma/resta en BCD
DEC	decrementa operando en 1
DIV	división sin signo
IDIV	división con signo
IMUL	multiplicación con signo
INC	incrementa operando en 1
MUL	multiplicación sin signo
NEG	cambia el signo
SBB	resta con acarreo
SUB	resta datos entre los registros y la memoria u otro reg.

Manipulación de bits

AND	Y lógica
NOT	invertir (complemento a 1)
OR	O lógica
SAR	desplazamiento aritmético a derecha
SHL/SAL	desplazamiento a izquierda
SHR	desplazamiento lógico a derecha
RCL	rotación a la izquierda con acarreo
ROR	rotación a izquierda
RCR	rotación a derecha con acarreo
ROR	rotación a derecha
TEST	operación con el AND lógico pero sólo afecta banderas
XOR	O exclusivo

Cadenas

CMPS

comparación entre 2 cadenas en memoria

LODS

cargar el acumulador con un dato de una cadena

MOVS

mover cadena de memoria a memoria

SCAS

comparación entre los datos de una cadena y el acumulador

STOS

almacenar el acumulador

Transferencia de programa

CALL	llamada a subrutina
INT	interrupción de software
INT 3	interrupción 3
INTO	interrupción si hay overflow
IRET	retorno de una rutina de interrupción
JA, JAE, JB, JBE	saltar si mayor, mayor o igual, menor, menor o igual
JE/JZ	saltar si es cero o igual
JG, JGE, JL, JLE	saltar si mayor, mayor o igual, menor, menor o igual
JMP	salto incondicional
JNE/JNZ	saltar si no es igual o no es cero
JNC, JNO, JNP, JNS	saltar si no acarreo, overflow, paridad, signo
JC, JO, JP, JS	saltar si acarreo, overflow, paridad, signo
LOOP	repite un ciclo CX veces
LOOPE, LOOPNE	igual a la anterior pero termina prematuramente por Z=1, 0
JCXZ	saltar si CX es 0
RET	retorno de subrutina

Control del procesador

CLC	borrar acarreo
CLD	habilitar incremento automático
CLI	deshabilitar terminal INTR
CMC	complementar acarreo
HLT	alto hasta que se reinicialice o exista interrupción
NOP	no operación
STC	activa acarreo
STD	habilitar decremento automático
STI	habilitar interrupciones
WAIT	espera a que el terminal TEST=0
LOCK	controla el terminal LOCK

Ejemplos de instrucciones

<code>in al, dx</code>	carga en AL el byte del puerto direccionado por DX
<code>in ax, dx</code>	carga en AX la palabra del puerto direccionado por DX
<code>mov ax, bx</code>	copia BX en AX
<code>mov ch, 5</code>	Carga CH con 5d
<code>mov bx, [bx+2]</code>	carga BX con la palabra de memoria apuntada por BX+2
<code>mov [bp+si], al</code>	almacena AL en la posición de memoria BP+SI
<code>adc al,bl</code>	suma AL + BL + acarreo
<code>daa</code>	Ajuste decimal de AL (si A=1 se suma 16h a AL)
<code>inc [bx+di]</code>	incrementa en 1 la posición de memoria BX + DI

Ejemplos de instrucciones

And ax, cx	operación Y lógico entre AX y CX
xor bx, bx	operación o exclusivo de BX y BX (notar que deja BX en cero)
sar AX, 5	shift aritmetico a derecha 5 lugares
mul cl	multiplica CL * AL y el resultado queda en AX
mul cx	multiplica CX * AX y el resultado queda en DX:AX
cmp al, bl je, otro_lado	compara los datos en AL y BL si son iguales salta a otro_lado
div cl	divide AX / CL y el resultado queda en AL y resto en AH
div cx	divide AX:DX / CX y el resultado queda en AX y resto en DX

Ejemplos de instrucciones

Mov si, 100	carga puntero de cadena de origen
mov di, 200	carga puntero de cadena de destino
mov cx, 50	carga longitud de cadenas
rep	repite hasta CX=0
cmps	compara [DS:SI] con [ES:DI]

	Mov si, 100	carga puntero de cadena de origen
	mov di, 200	carga puntero de cadena de destino
	mov cx, 50	carga longitud de cadenas
ciclo:	lods	carga AL con contenido de [DS:SI]
	out dx, al	saca AL por el puerto DX
	jcxz siga	si CX=0 termina de recorrer la cadena
	jmp ciclo	sigue recorriendo la cadena
sig:	-----	

Notas de interés

- Una referencia a memoria se forma con un segmento y un desplazamiento

ES:BX , CS:IP , DS:BX+SI+5

- Algunos registros tienen asociado un segmento por defecto (por lo que no es necesario ponerlos explícitamente)

DS es segmento por defecto de BX y SI

ES es segmento por defecto de DI

SS es segmento por defecto de SP y BP

CS es segmento por defecto de IP

- En modos complejos se asume como segmento por defecto el de la base

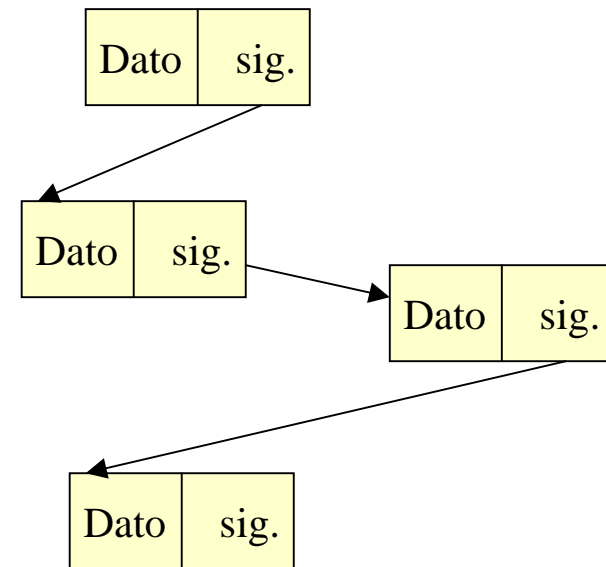
BX + DI por defecto DS

BP + SI por defecto ES

Programa de ejemplo I

Sumar los elementos de una lista vinculada apuntada por BX

```
      Mov bx, inicio_lista
      xor ax, ax
      clc
sumar : cmp bx, 0
      je fin
      adc ax, [bx]
      mov bx, [bx+2]
      jmp sumar
fin :
```



Programa de ejemplo II

Factorial (N)

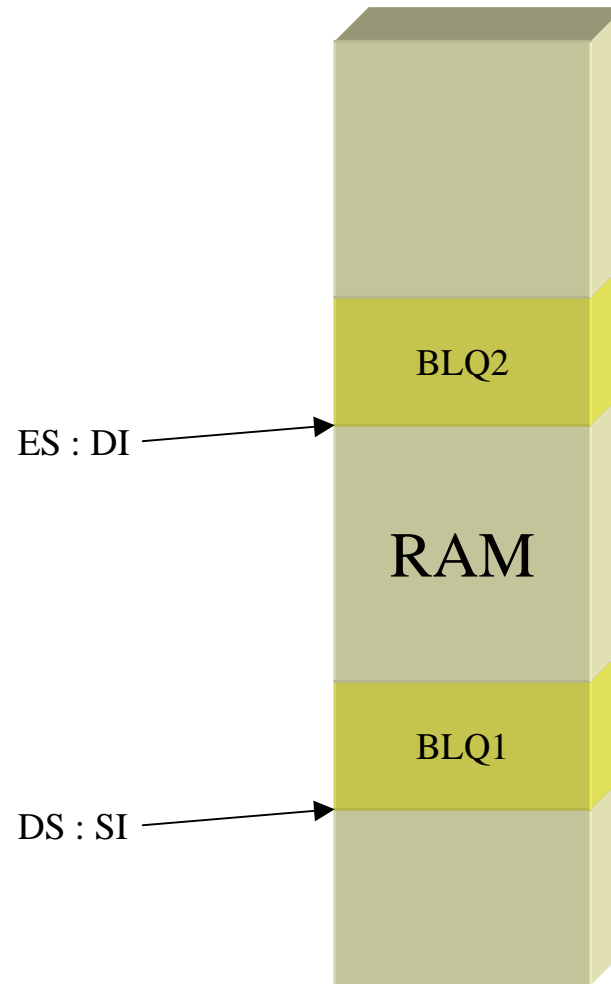
```
Fact PROC NEAR
    push bp
    mov bp,sp           ; bp apunta al tope de pila
    mov bx,[bp+4]      ; cargo el argumento pasado en bx
    cmp bx,1
    je fin             ; a los fines practicos corto la recursion en 1
    dec bx
    push bx            ; siguiente operando
    call fact
    pop bx             ; extraigo el resultado parcial de fact
fin:  mul [bp+4],bx    ; multiplico resultado parcial por argumento actual
    pop bp            ; saco bp de la pila
    RET               ; retorno
fact ENDP

start:  mov bx,N      ;cargo bx con N para obtener su factorial
    push,bx          ; apilo bx
    call fact
    pop bx           ; extraigo el resultado de la pila
    mov result,bx
```

Programa de ejemplo III

Copiar un bloque de memoria

```
Mov cx, longitud  
lds si, inicio_BLQ1  
les di, inicio_BLQ2  
rep  
movsb
```



Programa de ejemplo IV

Hacer una suma multibyte de 2 cadenas de 10 bytes

```
    lds si, operando1 + 10
    lds bx, operando2 + 10
    les di, resultado + 10
    mov cx, 10
    cld
sig: mov ax, [si]
    adc ax, [bx]
    mov [di], ax
    jcxz fin
    dec si
    dec bx
    dec di
    jmp sig
fin:
```

