

1. OPCODES EN ENSAMBLADOR X86

a) MOV

Función: Mueve datos de un lugar a otro (registro, memoria o inmediato).

Uso: MOV destino, fuente

Ejemplo: MOV AX, BX ; Copia el valor de BX en AX

b) ADD

Función: Suma un valor al registro o memoria.

Ejemplo: ADD AX, 5 ; $AX = AX + 5$

c) SUB

Función: Resta un valor.

Ejemplo: SUB AX, 1 ; $AX = AX - 1$

d) INC

Función: Incrementa en 1.

Ejemplo: INC AX ; $AX = AX + 1$

e) DEC

Función: Decrementa en 1.

Ejemplo: DEC AX ; $AX = AX - 1$

f) CMP

Función: Compara dos valores (resta interna, no guarda resultado).

Ejemplo: CMP AX, BX

g) JMP

Función: Salto incondicional.

Ejemplo: JMP etiqueta

h) JE / JNE

Función: JE salta si son iguales; JNE si no son iguales.

Ejemplo: JE fin

i) JZ / JNZ

Función: Igual que JE/JNE (Z = Zero Flag).

Ejemplo: JZ inicio

j) PUSH / POP

Función: Manejo de pila.

Ejemplo: PUSH AX / POP AX

k) CALL / RET

Función: Llamada y retorno de subrutina.

Ejemplo: CALL funcion / RET

l) SHR / SHL

Función: Desplazamiento de bits.

Ejemplo: SHL AX, 1

2. BITS DEL REGISTRO EFLAGS (0 AL 21)

Bit 0 — CF (Carry Flag): Indica acarreo/borrow en sumas/restas. Usado por instrucciones adc/sbb y saltos jc/jnc.

Bit 1 — (reservado): No usar; históricamente reservado o con valor fijo.

Bit 2 — PF (Parity Flag): Paridad del byte de resultado (1 si número de bits 1 es par).

Bit 3 — (reservado): No usar.

Bit 4 — AF (Auxiliary Carry / Adjust Flag): Acarreo entre bit 3 y 4. Útil en operaciones BCD/decimal.

Bit 5 — (reservado): No usar.

Bit 6 — ZF (Zero Flag): ZF=1 si el resultado fue cero. Base para je/jne, jz/jnz.

Bit 7 — SF (Sign Flag): Signo del resultado (MSB). Indica resultado negativo en representación con signo.

Bit 8 — TF (Trap Flag): Single-step/debug (genera trap después de cada instrucción cuando TF=1).

Bit 9 — IF (Interrupt Enable Flag): Habilita/deshabilita interrupciones externas (sti/cli).

Bit 10 — DF (Direction Flag): Controla la dirección de instrucciones de cadena (cld limpia, std pone).

Bit 11 — OF (Overflow Flag): Indica overflow en operaciones con signo.

Bits 12–13 — IOPL (I/O Privilege Level): 2 bits que indican nivel de privilegio para I/O (0..3).

Bit 14 — NT (Nested Task): Indica tarea anidada; uso histórico en gestión de tareas/TSS.

Bit 15 — (reservado): Reservado.

Bit 16 — RF (Resume Flag): Usado al manejar excepciones de debug para reanudar sin re-trigger.

Bit 17 — VM (Virtual-8086 Mode): Indica ejecución en modo Virtual-8086 (compatibilidad 16-bit en protegido).

Bit 18 — AC (Alignment Check): Activa verificación de alineación (genera excepción por accesos mal alineados en ring 3 en CPUs que lo soportan).

Bit 19 — VIF (Virtual Interrupt Flag): Imagen virtual del IF para entornos virtualizados.

Bit 20 — VIP (Virtual Interrupt Pending): Indica interrupción virtual pendiente (virtualización).

Bit 21 — ID (ID Flag): Usado para detectar soporte de CPUID (se puede togglear para comprobar presencia de CPUID).

Modelo de referencia: Intel® 80386 Programmer's Manual.

3. FRAGMENTO DE CÓDIGO NASM

```
section .data
```

```
msg_ok db 'Los valores son iguales', 10  
len_ok equ $-msg_ok
```

```
msg_no db 'No son iguales', 10  
len_no equ $-msg_no
```

```
section .text  
global _start
```

```
_start:
```

```
; --- ejemplo: comparar dos números ---
```

```
mov eax, 5 ; primer operando
```

```
mov ebx, 7 ; segundo operando
```

```
cmp eax, ebx ; prepara flags: ZF = 1 si eax==ebx  
je iguales ; si ZF=1 salta a 'iguales'
```

```
; rama 'no iguales'
```

```
mov rax, 1 ; syscall: write (x86-64)
```

```
mov rdi, 1 ; fd = stdout
```

```
mov rsi, msg_no
```

```
mov rdx, len_no
```

```
syscall
```

```
jmp fin ; salto incondicional al final
```

```
iguales:
```

```
; rama 'iguales'
```

```
mov rax, 1
```

```
mov rdi, 1
```

```
mov rsi, msg_ok
```

```
mov rdx, len_ok
```

```
syscall
```

```
fin:
```

```
; salir del programa
```

```
mov rax, 60 ; syscall: exit
```

```
xor rdi, rdi ; status 0
```

```
syscall
```

