1. Comparador de Números: Escribir un programa que reciba dos números y determine si son iguales, si uno es mayor que el otro, o si son negativos.

```
; Leer A y B (código de lectura aquí)

mov eax, A
mov ebx, B
cmp eax, ebx

je SON_IGUALES
jg A_MAYOR
jl B_MAYOR

SON_IGUALES:
    ; mostrar "Son iguales"
    jmp FIN

A_MAYOR:
    ; mostrar "A es mayor"
    jmp FIN

B_MAYOR:
    ; mostrar "B es mayor"
    jmp FIN

; Verificar si alguno es negativo
FIN:
cmp eax, 0
jl A_NEGATIVO
cmp ebx, 0
jl B_NEGATIVO
jmp DONE

A_NEGATIVO:
    ; mostrar "A es negativo"
    jmp DONE

B_NEGATIVO:
    ; mostrar "B es negativo"

DONE:
```

2. Clasificación de Números: Leer un número y clasificarlo como positivo, negativo o cero.

```
; Leer N

mov eax, N
cmp eax, 0
je ES_CERO
jg ES_POSITIVO
jl ES_NEGATIVO

ES_POSITIVO:
    ; imprimir "positivo"
    jmp FIN

ES_NEGATIVO:
    ; imprimir "negativo"
    jmp FIN

ES_CERO:
    ; imprimir "cero"

FIN:
```

3. Par o Impar: Leer un número y determinar si es par o impar usando únicamente la bandera de paridad (PF).

```
; Leer N

mov eax, N
test eax, 1       ; prueba el bit 0

jp ES_PAR         ; PF = 1 → par
jnp ES_IMPAR      ; PF = 0 → impar

ES_PAR:
    ; imprimir "par"
    jmp FIN

ES_IMPAR:
    ; imprimir "impar"

FIN:
```

4. Simulación de Overflow: Pedir dos números y sumarlos, verificando si ocurre desbordamiento con la bandera OF (Overflow Flag). Imprimir un mensaje si se detecta overflow.

```
; Leer A y B

mov eax, A
mov ebx, B
add eax, ebx     ; puede activar OF

jo HAY_OVERFLOW
jno SIN_OVERFLOW

HAY_OVERFLOW:
    ; imprimir "Overflow detectado"
    jmp FIN

SIN_OVERFLOW:
    ; imprimir "No hubo overflow"

FIN:
```

5. Simulación de Acarreo: Realizar una suma entre dos números y verificar si hay un acarreo con la bandera CF (Carry Flag). Mostrar si se generó un acarreo o no.

```
; Leer A y B

mov eax, A
mov ebx, B
add eax, ebx     ; actualiza CF

jc HAY_ACARREO
jnc SIN_ACARREO

HAY_ACARREO:
    ; imprimir "Hubo acarreo"
    jmp FIN

SIN_ACARREO:
    ; imprimir "Sin acarreo"

FIN:
```

6. Mínimo y Máximo de Tres Números: Leer tres números e identificar el menor y el mayor.

```
; Leer A, B, C

mov eax, A
mov ebx, B

; Comparar A y B
cmp eax, ebx
jle A_ES_MENOR_O_IGUAL
mov min, ebx
mov max_temp, eax
jmp COMPARAR_CON_C

A_ES_MENOR_O_IGUAL:
mov min, eax
mov max_temp, ebx

COMPARAR_CON_C:
mov ecx, C

; Comparar con C para obtener mínimo
cmp ecx, min
jl C_ES_MENOR
jmp MIN_LISTO

C_ES_MENOR:
mov min, ecx

MIN_LISTO:

; Comparar C con max_temp para obtener máximo
cmp ecx, max_temp
jg C_ES_MAYOR
jmp MAX_LISTO

C_ES_MAYOR:
mov max, ecx
jmp MAX_LISTO

MAX_LISTO:
; imprimir min y max
```

7. Ordenamiento de Dos Números Leer dos números e intercambiarlos si no están en orden ascendente usando solo saltos condicionales.

```
; Leer A y B

mov eax, A
mov ebx, B
cmp eax, ebx

jle YA_ORDENADOS ; A <= B → nada que hacer

; Intercambio
mov ecx, eax
mov eax, ebx
mov ebx, ecx

YA_ORDENADOS:
; imprimir A, B en orden ascendente
```

8. Ciclo de Conteo sin Comparaciones: Implementar un contador de 0 a 9.

```
mov eax, 10   ; contador desde 10

CICLO:
dec eax
; imprimir eax
jns CICLO     ; mientras no sea negativo sigue

; termina cuando eax = -1
```