

```

1 section .data
2     num1 db 5          ; Define la variable num1 con el valor de un byte: 5 (decimal)
3     num2 db 11         ; Define la variable num2 con el valor de un byte: 11 (decimal)
4     result db 0        ; Define la variable result (1 byte) inicializada a 0
5     msg db 'Resultado: ', 0 ; Define la cadena de texto a imprimir (incluye terminador 0)
6
7 section .bss
8     buffer resb 4      ; Reserva 4 bytes en memoria no inicializada para un buffer
9
10 section .text
11     global _start      ; Declara la etiqueta _start como punto de entrada global
12
13 _start:
14
15
16     mov al, [num1]      ; Carga el valor de num1 (5) en el registro AL (registro de 8 bits)
17     add al, [num2]      ; Suma el valor de num2 (11) a AL. AL = 5 + 11 = 16 (decimal)
18     mov [result], al    ; Almacena el resultado (16) en la variable result
19
20     movzx eax, byte [result] ; Extiende con ceros el byte de [result] (16) a EAX (32 bits). EAX = 16
21     add eax, 48         ; Suma 48 (código ASCII de '0'). EAX = 16 + 48 = 64 (decimal)
22     ; El código ASCII 64 es el carácter '@'.
23
24
25     mov eax, 4          ; Syscall número 4: write
26     mov ebx, 1          ; File descriptor 1: stdout (salida estándar)
27     mov ecx, msg        ; Puntero al inicio de la cadena a imprimir
28     mov edx, 11         ; Longitud de la cadena a imprimir (incluyendo el espacio)
29     int 0x80            ; Ejecuta la llamada al sistema
30
31
32     mov eax, 4          ; Syscall número 4: write
33     mov ebx, 1          ; File descriptor 1: stdout
34     mov ecx, buffer     ; Puntero al carácter ASCII a imprimir ('@')
35     mov edx, 1          ; Longitud: 1 byte
36     int 0x80            ; Ejecuta la llamada al sistema
37
38
39     mov eax, 1          ; Syscall número 1: exit
40     xor ebx, ebx        ; Código de retorno 0
41     int 0x80            ; Ejecuta la llamada al sistema

```

PROGRAMA CON DIRECCIONAMIENTO INMEDIATO

section .data

; num1 db 5 ; YA NO SE USAN

; num2 db 11 ; YA NO SE USAN

result db 0

msg db 'Resultado: ', 0

section .bss

buffer resb 4

section .text

global _start

_start:

; Uso de Inmediato: Carga el valor 5

mov al, 5 ; Mueve el valor inmediato 5 a AL

; Uso de Inmediato: Suma el valor 11

add al, 11 ; Suma el valor inmediato 11 a AL (AL = 16)

mov [result], al

; ... (resto del código sin cambios, ya usa inmediato en add eax, 48)

movzx eax, byte [result]

add eax, 48

mov [buffer], al

mov eax, 4

mov ebx, 1

mov ecx, msg

mov edx, 11

int 0x80

```

mov eax, 4
mov ebx, 1
mov ecx, buffer
mov edx, 1
int 0x80
mov eax, 1
xor ebx, ebx
int 0x80

```

; PROGRAMA CON DIRECCIONAMIENTO INDIRECTO (usando ESI)

```

section .data
num1 db 5
num2 db 11
result db 0
msg db 'Resultado: ', 0
section .bss
buffer resb 4
section .text
global _start
_start:
; Cargamos la dirección base (num1) en ESI
mov esi, num1 ; ESI = dirección de num1
; Uso de Indirecto: [ESI] accede a num1 (5)
mov al, [esi] ; Carga num1 (5) a AL
; Uso de Indirecto con Desplazamiento: [ESI+1] accede a num2 (11)
add al, [esi+1] ; Suma num2 (11) a AL (AL = 16)
; Uso de Indirecto: Usamos la etiqueta 'result'
mov [result], al
; ... (resto del código sin cambios)
movzx eax, byte [result]
add eax, 48
mov [buffer], eax
mov eax, 4
mov ebx, 1

mov ecx, msg
mov edx, 11
int 0x80
mov eax, 4
mov ebx, 1
mov ecx, buffer
mov edx, 1
int 0x80
mov eax, 1
xor ebx, ebx
int 0x80

```