



**Universidad Autónoma de Baja California**

**Facultad de Ingeniería, Arquitectura y Diseño**

**Ángel Gabriel Zúñiga López**



**Prof. Jonatan Crespo Ragland**

**Práctica Repositorios**

**Organización de Computadoras**

**3er. Semestre**

**Ensenada B.C a 28 de Noviembre del 2024**

## Taller 5

```
1 section .data
2 msg db 'imprimir input del teclado: ';Input: &#39;, 0 ; Mensaje que se mostrara antes de la entrada
3
4 section .bss
5 input resb 1 ; Espacio para almacenar el caracter ingresado
6 sum resb 1 ; Espacio para almacenar la suma
7
8 section .text
9 global _start
10
11 _start:
12 ; Mostrar mensaje en consola
13 mov eax, 4
14 mov ebx, 1
15 mov ecx, msg ; direccion del mensaje
16 mov edx, 28 ; longitud del mensaje
17 int 0x80
18
19 ; Leer un caracter desde el teclado
20
21 mov eax, 3
22 mov ebx, 0
23 mov ecx, input ; direccion para almacenar la entrada
24 mov edx, 100 ; leer 1 byte (1 caracter)
25 int 0x80
26
27 ; Mostrar el caracter ingresado
28 mov eax, 4 ; syscall numero 4 es write (sys_write)
29 mov ebx, 1 ; descriptor de archivo 1 es stdout
30 mov ecx, input ; direccion del caracter
31 mov edx, 100 ; longitud del caracter
32 int 0x80 ; llamada al sistema
33
34 ; Calcular la suma de los caracteres
35 mov al, [input]
36 add al, [input]
37 mov [sum], al ; almacenar la suma en la variable sum
38
39 ; Mostrar la suma
40 mov eax, 4
41
42 mov ebx, 1
43 mov ecx, sum ; direccion de la suma
44 mov edx, 1 ; longitud de la suma
45 int 0x80
46
47 ; Terminar el programa
48 mov eax, 1
49 xor ebx, ebx ; codigo de salida 0
50 int 0x80
```

STDIN

organizacion

Output:

imprimir input del teclado: organizacion

## Taller 7

```
1 section .data
2
3 num1 db 5 ;se modifica el numero y al sumarlo con num2 se puede buscar el numero ascii que desea
4 num2 db 11;esto tomando en cuenta que se comienza en el 0 o el "48" en ascii como se declara en add eax, 48
5 result db 0 ;define una variable para almacenar el resultado
6 msg db 'Resultado:', 0
7
8 section .bss
9 buffer resb 4;reserva un espacio de 4 bytes en el buffer para el resultado
10
11 section .text
12 global _start
13
14 _start:
15
16 mov al, [num1]
17 add al, [num2]
18 mov [result], al
19 ;convertir el resultado a ASCII
20 movzx eax, byte [result]
21 add eax, 48 ;convertir el valor numerico en su correspondiente ASCII ('0' = 48)
22 mov [buffer], al ; almacenar el caracter ASCII en el buffer
23
24 mov eax, 4
25 mov ebx, 1
26 mov ecx, msg
27 mov edx, 11
28 int 0x80
29
30 mov eax, 4
31 mov ebx, 1
32 mov ecx, buffer
33 mov edx, 1
34 int 0x80
35
36 mov eax, 1
37 xor ebx, ebx
38 int 0x80
```

```

1 * section .data
2     num1 db 'A'           ; Guardar el carácter 'A'
3     num2 db '\'          ; Guardar el carácter '\'
4     num3 db '$'          ; Guardar el carácter '$'
5     num4 db '&'          ; Guardar el carácter '&'
6     num5 db 'i'          ; Guardar el carácter 'i'
7     msg db 'Resultado: ', 0 ; Mensaje que se imprimirá
8 * section .bss
9     buffer resb 5         ; Reservar 5 bytes en el buffer para los caracteres
10 section .text
11 global _start
12 * _start:
13     ; Cargar y almacenar cada carácter en el buffer
14
15     mov al, [num1]        ; Cargar 'A' en el registro AL
16     mov [buffer], al     ; Almacenar 'A' en el primer byte del buffer
17
18     mov al, [num2]        ; Cargar '\' en el registro AL
19     mov [buffer+1], al   ; Almacenar '\' en el segundo byte del buffer
20
21     mov al, [num3]        ; Cargar '$' en el registro AL
22     mov [buffer+2], al   ; Almacenar '$' en el tercer byte del buffer
23
24     mov al, [num4]        ; Cargar '&' en el registro AL
25     mov [buffer+3], al   ; Almacenar '&' en el cuarto byte del buffer
26
27     mov al, [num5]        ; Cargar 'i' en el registro AL
28     mov [buffer+4], al   ; Almacenar 'i' en el quinto byte del buffer
29
30     ; Imprimir el mensaje 'Resultado: '
31     mov eax, 4
32     mov ebx, 1
33     mov ecx, msg
34     mov edx, 11
35     int 0x80
36
37     ; Imprimir el buffer con los caracteres A\ $&i
38     mov eax, 4
39     mov ebx, 1
40     mov ecx, buffer
41     mov edx, 5
42     int 0x80
43
44     ; Terminar el programa
45     mov eax, 1
46     xor ebx, ebx
47     int 0x80

```

```

1 section .data
2     msg db 'Resultado: ', 0 ; Mensaje que se imprimirá
3 section .bss
4     buffer resb 1 ; Reservar un byte en el buffer para el carácter
5 section .text
6 global _start
7 _start:
8     ; Usar direccionamiento inmediato para cargar el carácter '@'
9     mov al, '@' ; Cargar directamente el carácter '@' en AL
10    mov [buffer], al ; Almacenar el carácter '@' en el buffer
11
12    ; Imprimir el mensaje 'Resultado: '
13    mov eax, 4
14    mov ebx, 1
15    mov ecx, msg
16    mov edx, 11 ; Longitud del mensaje
17    int 0x80 ; Interrupción para realizar la llamada al sistema
18
19    ; Imprimir el carácter '@'
20    mov eax, 4
21    mov ebx, 1
22    mov ecx, buffer ; Dirección del buffer que contiene '@'
23    mov edx, 1 ; Imprimir un solo carácter (1 byte)
24    int 0x80 ; Interrupción para realizar la llamada al sistema
25
26    ; Terminar el programa
27    mov eax, 1
28    xor ebx, ebx
29    int 0x80 ; Interrupción para realizar la llamada al sistema
30

```

```

1 section .data
2     char_at db '@' ; Almacenar el carácter '@' en la memoria
3     msg db 'Resultado: ', 0 ; Mensaje que se imprimirá
4 section .bss
5     buffer resb 1 ; Reservar un byte en el buffer para el carácter
6 section .text
7 global _start
8 _start:
9     ; Usar direccionamiento indirecto para cargar el carácter '@'
10    mov esi, char_at ; Cargar la dirección de la variable 'char_at' en ESI
11    mov al, [esi] ; Cargar el valor de la dirección almacenada en ESI en AL
12    mov [buffer], al ; Almacenar el carácter '@' en el buffer
13
14    ; Imprimir el mensaje 'Resultado: '
15    mov eax, 4
16    mov ebx, 1
17    mov ecx, msg ; Dirección del mensaje
18    mov edx, 11 ; Longitud del mensaje
19    int 0x80
20
21    ; Imprimir el carácter '@'
22    mov eax, 4
23    mov ebx, 1
24    mov ecx, buffer ; Dirección del buffer que contiene '@'
25    mov edx, 1 ; Imprimir un solo carácter (1 byte)
26    int 0x80
27
28    ; Terminar el programa
29    mov eax, 1
30    xor ebx, ebx
31    int 0x80 ; Interrupción para realizar la llamada al sistema
32

```

## Taller 8

```
1 section .data
2     msg db 'Resultado: ', 0
3     newline db 0xA
4
5 section .bss
6     res resb 4 ; Espacio para el resultado
7
8 section .text
9     global _start
10
11 _start:
12
13     ; Instrucciones aritméticas
14     mov eax, 10 ; se coloca los números a sumar
15     mov ebx, 5  ; se coloca los números a sumar
16     add eax, ebx ; se suma eax+ebx, se almacena en eax
17
18     ; Instrucción Lógica (AND)
19     and eax, 0xF ; Realiza AND bit a bit con 0xF (15 en dec.)
20
21     ; Instrucciones de manipulación de bits
22     shl eax, 1   ; aquí la sumatoria se por 2
23
24     ; Guardar el resultado en la sección .bss
25     mov [res], eax ; almacena el valor de la sumatoria en res
26
27     ; Llamar a la rutina para imprimir el resultado
28     mov eax, 4 ; Syscall para escribir
29     mov ebx, 1 ; Usar la salida estándar (pantalla)
30     mov ecx, msg ; Dirección del mensaje a imprimir
31     mov edx, 11 ; Longitud del mensaje
32     int 0x80 ; Interrupción para imprimir el mensaje
33
34     ; Imprimir el número (resultado almacenado en 'res')
35     mov eax, [res] ; Cargar el resultado en EAX
36     add eax, '0' ; Convertir el número en carácter (ASCII)
37     mov [res], eax ; Almacenar el carácter convertido
38     mov eax, 4 ; Syscall para escribir
39     mov ebx, 1 ; Usar la salida estándar
```

```

1  section .data
2  msg db 'Resultado: ', 0
3  newline db 0xA
4  section .bss
5  res resb 4 ; Espacio para el resultado
6
7  section .text
8  global _start
9  _start:
10 ; Cargar un valor en EAX
11 mov eax, 0xF0F0 ; EAX = 1111 0000 1111 0000 (binario)
12
13 ; Aplicar máscara usando AND
14 mov ebx, 0xF0F ; EBX = 0000 1111 0000 1111 (binario)
15 and eax, ebx ; EAX = EAX AND EBX
16 | | | | | | | | ; Resultado en EAX = 0000 0000 0000 0000 (binario)
17
18 ; Guardar el resultado en la sección .bss
19 mov [res], eax ; Guarda el valor en 'res' para impresión
20
21 ; Imprimir el mensaje "Resultado: "
22 mov eax, 4
23 mov ebx, 1
24 mov ecx, msg
25 mov edx, 11
26 int 0x80
27
28 ; Imprimir el valor en 'res' como carácter
29 mov eax, [res]
30 add eax, '0' ; Convertir a carácter ASCII (para fines de demostraci
31 mov [res], eax
32 mov eax, 4
33 mov ebx, 1
34 mov ecx, res
35 mov edx, 1
36 int 0x80
37
38 ; Imprimir nueva línea
39 mov eax, 4
40 mov ebx, 1
41 mov ecx, newline
42 mov edx, 1
43 int 0x80
44
45 ; Terminar el programa
46 mov eax, 1
47 xor ebx, ebx
48 int 0x80
49

```

```

1  section .data
2  msg db 'Resultado: ', 0 ; Mensaje para imprimir
3  newline db 0xA ; Nueva línea (salto de línea)
4
5  section .bss
6  res resb 4 ; Espacio para el resultado
7
8  section .text
9  global _start
10
11 _start:
12
13 ; Instrucciones aritméticas
14 mov eax, 100 ; Coloca 100 en el registro EAX
15 mov ebx, 8 ; Coloca 8 en el registro EBX (100 + 8 = 108)
16 add eax, ebx ; Suma EAX + EBX, el resultado se almacena en EAX (100 + 8 = 108)
17
18 ; Guardar el resultado en la sección .bss
19 mov [res], eax ; Almacena el valor de EAX (108) en la memoria reservada (res)
20
21 ; Llamar a la rutina para imprimir el resultado
22 mov eax, 4 ; Syscall para escribir
23 mov ebx, 1 ; Usar la salida estándar (pantalla)
24 mov ecx, msg ; Dirección del mensaje a imprimir
25 mov edx, 11 ; Longitud del mensaje
26 int 0x80 ; Interrupción para imprimir el mensaje
27
28 ; Imprimir el carácter correspondiente a 108 ('L')
29 mov eax, [res] ; Cargar el resultado en EAX
30 ; No necesitamos hacer ninguna conversión aquí, simplemente lo movemos a [res]
31 mov [res], al ; Almacenar el carácter 'L' en res
32
33 mov eax, 4 ; Syscall para escribir
34 mov ebx, 1 ; Usar la salida estándar
35 mov ecx, res ; Dirección del resultado
36 mov edx, 1 ; Longitud de 1 carácter
37 int 0x80 ; Interrupción para imprimir el carácter 'L'
38
39 ; Imprimir nueva línea
40 mov eax, 4 ; Syscall para escribir
41 mov ebx, 1 ; Usar la salida estándar
42 mov ecx, newline ; Dirección de la nueva línea
43 mov edx, 1 ; Longitud de 1 carácter
44 int 0x80 ; Interrupción para imprimir nueva línea
45
46 ; Terminar el programa
47 mov eax, 1 ; Syscall para salir
48 xor ebx, ebx ; Código de salida 0
49 int 0x80 ; Interrupción para terminar el programa
50

```



```

1 section .data
2 msg db 'Resultado: ', 0 ; Mensaje para imprimir
3 newline db 0xA ; Nueva línea (salto de línea)
4
5 section .bss
6 res resb 4 ; Espacio para el resultado
7
8 section .text
9 global _start
10
11 _start:
12
13 ; Instrucciones aritméticas
14 mov eax, 30 ; Coloca 30 en el registro EAX
15 mov ebx, 38 ; Coloca 38 en el registro EBX (30 + 38 = 68)
16 add eax, ebx ; Suma EAX + EBX, el resultado se almacena en EAX (30 + 38 = 68)
17 ; Instrucción lógica (AND)
18 and eax, 0xFF ; Realiza AND bit a bit con 0xFF (255 en decimal),
19 ; EAX seguirá siendo 68 (0x44 en hexadecimal)
20 ; Guardar el resultado en la sección .bss
21 mov [res], eax ; Almacena el valor de EAX (68) en la memoria reservada (res)
22 ; Llamar a la rutina para imprimir el resultado
23 mov eax, 4 ; Syscall para escribir
24 mov ebx, 1 ; Usar la salida estándar (pantalla)
25 mov ecx, msg ; Dirección del mensaje a imprimir
26 mov edx, 11 ; Longitud del mensaje
27 int 0x80 ; Interrupción para imprimir el mensaje
28 ; Imprimir el carácter correspondiente a 68 ('D')
29 mov eax, [res] ; Cargar el resultado en EAX
30 movzx eax, byte [res] ; Asegurarse de que EAX contiene el valor correcto
31 mov [res], al ; Almacenar el valor ASCII correspondiente en res
32 mov eax, 4 ; Syscall para escribir
33 mov ebx, 1 ; Usar la salida estándar
34 mov ecx, res ; Dirección del resultado
35 mov edx, 1 ; Longitud de 1 carácter
36 int 0x80 ; Interrupción para imprimir la letra 'D'
37 ; Imprimir nueva línea
38 mov eax, 4 ; Syscall para escribir
39 mov ebx, 1 ; Usar la salida estándar
40 mov ecx, newline ; Dirección de la nueva línea
41 mov edx, 1 ; Longitud de 1 carácter
42 int 0x80 ; Interrupción para imprimir nueva línea
43 ; Terminar el programa
44 mov eax, 1 ; Syscall para salir
45 xor ebx, ebx ; Código de salida 0
46 int 0x80 ; Interrupción para terminar el programa
47

```

```

1  section .data
2  msg db 'Resultado: ', 0 ; Mensaje para imprimir
3  newline db 0xA ; Nueva línea (salto de línea)
4
5  section .bss
6  res resb 4 ; Espacio para el resultado
7
8  section .text
9  global _start
10
11 _start:
12
13 ; Instrucciones aritméticas
14 mov eax, 30 ; Coloca 30 en el registro EAX
15 mov ebx, 36 ; Coloca 36 en el registro EBX (30 + 36 = 66)
16 add eax, ebx ; Suma EAX + EBX, el resultado se almacena en EAX (30 + 36 = 66)
17 ; Instrucción lógica (AND)
18 and eax, 0xFF ; Realiza AND bit a bit con 0xFF (255 en decimal),
19 ; EAX seguirá siendo 66 (0x42 en hexadecimal)
20 ; Guardar el resultado en la sección .bss
21 mov [res], eax ; Almacena el valor de EAX (66) en la memoria reservada (res)
22 ; Llamar a la rutina para imprimir el resultado
23 mov eax, 4 ; Syscall para escribir
24 mov ebx, 1 ; Usar la salida estándar (pantalla)
25 mov ecx, msg ; Dirección del mensaje a imprimir
26 mov edx, 11 ; Longitud del mensaje
27 int 0x80 ; Interrupción para imprimir el mensaje
28 ; Imprimir el carácter correspondiente a 66 ('B')
29 mov eax, [res] ; Cargar el resultado en EAX
30 movzx eax, byte [res] ; Asegurarse de que EAX contiene el valor correcto
31 mov [res], al ; Almacenar el valor ASCII correspondiente en res
32 mov eax, 4 ; Syscall para escribir
33 mov ebx, 1 ; Usar la salida estándar
34 mov ecx, res ; Dirección del resultado
35 mov edx, 1 ; Longitud de 1 carácter
36 int 0x80 ; Interrupción para imprimir la letra 'B'
37 ; Imprimir nueva línea
38 mov eax, 4 ; Syscall para escribir
39 mov ebx, 1 ; Usar la salida estándar
40 mov ecx, newline ; Dirección de la nueva línea
41 mov edx, 1 ; Longitud de 1 carácter
42 int 0x80 ; Interrupción para imprimir nueva línea
43 ; Terminar el programa
44 mov eax, 1 ; Syscall para salir
45 xor ebx, ebx ; Código de salida 0
46 int 0x80 ; Interrupción para terminar el programa
47

```

```

1  section .data
2  msg db 'Resultado: ', 0 ; Mensaje para imprimir
3  newline db 0xA ; Nueva línea (salto de línea)
4
5  section .bss
6  res resb 4 ; Espacio para el resultado
7
8  section .text
9  global _start
10
11 _start:
12
13 ; Instrucciones aritméticas
14 mov eax, 2 ; Coloca 2 en el registro EAX
15 ; Aquí no se necesita una suma adicional, simplemente usamos 2 directamente.
16
17 ; Guardar el resultado en la sección .bss
18 mov [res], eax ; Almacena el valor de EAX (2) en la memoria reservada (res)
19
20 ; Llamar a la rutina para imprimir el resultado
21 mov eax, 4 ; Syscall para escribir
22 mov ebx, 1 ; Usar la salida estándar (pantalla)
23 mov ecx, msg ; Dirección del mensaje a imprimir
24 mov edx, 11 ; Longitud del mensaje
25 int 0x80 ; Interrupción para imprimir el mensaje
26
27 ; Imprimir el carácter correspondiente a 50 ('2')
28 mov eax, [res] ; Cargar el resultado en EAX
29 add eax, '0' ; Convertir el número en carácter ASCII (2 + 48 = 50)
30 mov [res], al ; Almacenar el carácter correspondiente en res
31
32 mov eax, 4 ; Syscall para escribir
33 mov ebx, 1 ; Usar la salida estándar
34 mov ecx, res ; Dirección del resultado
35 mov edx, 1 ; Longitud de 1 carácter
36 int 0x80 ; Interrupción para imprimir el carácter '2'
37
38 ; Imprimir nueva línea
39 mov eax, 4 ; Syscall para escribir
40 mov ebx, 1 ; Usar la salida estándar
41 mov ecx, newline ; Dirección de la nueva línea
42 mov edx, 1 ; Longitud de 1 carácter
43 int 0x80 ; Interrupción para imprimir nueva línea
44
45 ; Terminar el programa
46 mov eax, 1 ; Syscall para salir
47 xor ebx, ebx ; Código de salida 0
48 int 0x80 ; Interrupción para terminar el programa
49

```

```

1  section .data
2  msg db 'Resultado: ', 0 ; Mensaje para imprimir
3  newline db 0xA ; Nueva línea (salto de línea)
4
5  section .bss
6  res resb 4 ; Espacio para el resultado
7
8  section .text
9  global _start
10
11 _start:
12
13 ; Instrucciones aritméticas
14 mov eax, 4 ; Coloca 4 en el registro EAX
15 ; Guardar el resultado en la sección .bss
16 mov [res], eax ; Almacena el valor de EAX (4) en la memoria reservada (res)
17
18 ; Llamar a la rutina para imprimir el resultado
19 mov eax, 4 ; Syscall para escribir
20 mov ebx, 1 ; Usar la salida estándar (pantalla)
21 mov ecx, msg ; Dirección del mensaje a imprimir
22 mov edx, 11 ; Longitud del mensaje
23 int 0x80 ; Interrupción para imprimir el mensaje
24
25 ; Imprimir el carácter correspondiente a 4 ('4')
26 mov eax, [res] ; Cargar el resultado en EAX
27 add eax, '0' ; Convertir el número en carácter ASCII (4 + 48 = 52)
28 mov [res], al ; Almacenar el carácter correspondiente en res
29
30 mov eax, 4 ; Syscall para escribir
31 mov ebx, 1 ; Usar la salida estándar
32 mov ecx, res ; Dirección del resultado
33 mov edx, 1 ; Longitud de 1 carácter
34 int 0x80 ; Interrupción para imprimir el carácter '4'
35
36 ; Imprimir nueva línea
37 mov eax, 4 ; Syscall para escribir
38 mov ebx, 1 ; Usar la salida estándar
39 mov ecx, newline ; Dirección de la nueva línea
40 mov edx, 1 ; Longitud de 1 carácter
41 int 0x80 ; Interrupción para imprimir nueva línea
42
43 ; Terminar el programa
44 mov eax, 1 ; Syscall para salir
45 xor ebx, ebx ; Código de salida 0
46 int 0x80 ; Interrupción para terminar el programa
47

```

## Taller 10

```
1  CMP AX, BX    ; Compara el valor de AX con el valor de BX
2  JE  Etiqueta  ; Salta a "Etiqueta" si AX == BX (si ZF = 1)
```

```
CMP AX, BX    ; Compara el valor de AX con el valor de BX
JZ  Etiqueta  ; Salta a "Etiqueta" si AX == BX (si ZF = 1)
```

```
CMP AX, BX    ; Compara el valor de AX con el valor de BX
JNE Etiqueta  ; Salta a "Etiqueta" si AX != BX (si ZF = 0)
```

```
CMP AX, BX    ; Compara el valor de AX con el valor de BX
JNZ Etiqueta  ; Salta a "Etiqueta" si AX != BX (si ZF = 0)
```

```
CMP AX, BX    ; Compara el valor de AX con el valor de BX
JGE Etiqueta  ; Salta a "Etiqueta" si AX >= BX (para números con signo)
```

```
CMP AX, BX    ; Compara AX con BX
JL  Etiqueta  ; Salta a "Etiqueta" si AX < BX (en números con signo)
```

```
CMP AX, BX    ; Compara el valor de AX con el valor de BX
JLE Etiqueta  ; Salta a "Etiqueta" si AX <= BX (para números con signo)
```

```
CMP AX, BX    ; Compara el valor de AX con el valor de BX
JS  Etiqueta  ; Salta a "Etiqueta" si el resultado de la comparación es negativo (si SF = 1)
```

```
MOV AX, 0 ;Inicializa sum a 0
MOV CX, 1 ;Inicializa count a 1
```

```
while_loop:
CMP CX,10 ; Compara count con 10
JG end_while ; Salta al final si count > 10
ADD AX, CX ; Suma count a sum
INC Cx ;Incrementa count
JMP while_loop ; Repite el ciclo
end_while:
```

```
MOV AX, 0 ;Inicializa sum a 0
MOV SI, 0 ;puntero a la lista
```

```
do_while_loop:
MOV BX,[SI] ;LEE EL NUMERO DE LA lista
ADD AX, BX ;SUMA EL NUMERO A SUM
ADD SI, 2 ;MUEVE EL PUNTERO AL SIGUIENTE NUMERO
CMP BX, 0 ;VERIFICA SI EL NUMERO ES NEGATIVO
JS end_do_while ; salta si el numero es NEGATIVO
JMP do_while_loop ;repite el ciclo
end_do_while
```



```
MOV AX,1 ;INICIALIZA PRODUCT A 1
MOV CX,1 ;Inicializa i a 1
```

```
for_loop:
CMP CX,5 ; Compara i con 5
JG end_for ; Salta si i>5
MUL CX ; Multiplica product por i
INC CX;INCREMENTA i
JMP for_loop ; Repite el ciclo.
end_for:
```

```
MOV AX, num ;Carga el valor de num
TEST AX,1 ;Prueba si el bit menos significativo es 0 (par)
JZ es_par ;Salta a es_par si es 0 (num es par)
MOV result_odd, 1; Almacena el resultado en result_odd
JMP end_if_else
es par:
MOV result_even, 1; Almacena el resultado en result_even
end_if_else:
```

```
MOV CX, 10 ; Inicializa count en 10
```

```
for_loop_dec:
CMP CX,1 ; Compara count con 1
JL end_for_dec ;Salta si count <1
;Aquí se imprime o almacena el valor de count
DEC CX ;Decrementa count
JMP for_loop_dec ; Repite el ciclo.
end_for_dec:
```

## Taller 11

```
1 section .data
2     ; Estructura de fecha (dd/mm/yyyy)
3     fecha dd 0      ; Día
4         dd 0      ; Mes
5         dd 0      ; Año
6
7     ; Estructura de correo electrónico
8     correo db "usuario@dominio.com", 0 ; Dirección de correo electrónico terminada en NULL
9
10    ; Estructura de dirección completa
11    direccion_calle db "Calle Ejemplo", 0
12    direccion_numero db "123", 0
13    direccion_colonia db "Colonia Centro", 0
14
15    ; Cadena CURP (18 caracteres + NULL)
16    curp db "ABCD010101HDFLRC01", 0
```

```
1 section .text
2     global _start
3
4 _start:
5     ; Acceso y manipulación de la fecha
6     mov eax, [fecha]      ; Cargar el día
7     add eax, 1            ; Incrementar el día
8     mov [fecha], eax      ; Guardar el día actualizado
9
10    mov eax, [fecha + 4]   ; Cargar el mes
11    mov [fecha + 4], 2     ; Establecer el mes a febrero
12
13    mov eax, [fecha + 8]   ; Cargar el año
14    mov [fecha + 8], 2024 ; Cambiar el año a 2024
15
16    ; Modificar la dirección
17    mov esi, direccion_calle
18    mov edi, calle_nueva
19    call cambiar_cadena
20
21    ; Terminar el programa
22    mov eax, 60            ; syscall: exit
23    xor edi, edi          ; status 0
24    syscall
25
26 cambiar_cadena:
27    ; Esta función copia una nueva cadena en una dirección específica
28    ; Entradas: ESI -> origen, EDI -> destino
29    cld                  ; Asegurar dirección de incremento
30    rep movsb            ; Copiar byte a byte
31    ret
```



```

1 section .data
2     num1 db 5           ; Define un byte con el valor 5
3     num2 db 11          ; Define otro byte con el valor 11
4     result db 0         ; Espacio para almacenar el resultado (1 byte)
5     message db "Resultado: ", 0 ; Mensaje de texto terminado en NULL para impresión
6
7 section .bss
8     buffer resb 4        ; Reserva 4 bytes en memoria para el buffer (para números)
9
10 section .text
11     global _start        ; Punto de entrada principal del programa
12
13 macro PRINT_STRING 1     ; Macro para imprimir cadenas
14     mov eax, 4           ; syscall: write
15     mov ebx, 1           ; descriptor de archivo (1 = stdout)
16     mov ecx, %1          ; dirección de la cadena a imprimir
17     mov edx, 13          ; longitud de la cadena (13 caracteres en este caso)
18     int 0x80            ; llamada al sistema
19 %endmacro
20
21 macro PRINT_NUMBER 1     ; Macro para imprimir un número (convertido a carácter)
22     mov eax, %1          ; Carga el número en eax
23     add eax, '0'         ; Convierte el número a su carácter ASCII
24     mov [buffer], eax    ; Almacena el carácter en el buffer
25     mov eax, 4           ; syscall: write
26     mov ebx, 1           ; descriptor de archivo (1 = stdout)
27     mov ecx, buffer      ; dirección del buffer a imprimir
28     mov edx, 1           ; longitud del dato a imprimir (1 carácter)
29     int 0x80            ; llamada al sistema
30 %endmacro
31
32 _start:
33     ; Realizar la suma
34     mov al, [num1]       ; Carga el valor de num1 en el registro AL
35     add al, [num2]       ; Suma el valor de num2 al contenido de AL
36     mov [result], al    ; Almacena el resultado en la variable result
37
38     ; Imprimir el mensaje
39     PRINT_STRING message ; Llama al macro PRINT_STRING para imprimir "Resultado: "
40
41     ; Imprimir el número resultado
42     PRINT_NUMBER [result] ; Llama al macro PRINT_NUMBER para imprimir el valor de result
43
44     ; Salir del programa
45     mov eax, 1           ; syscall: exit
46     mov ebx, 0           ; código de salida (0 = éxito)
47     int 0x80            ; llamada al sistema para terminar el programa
48

```

```

1 section .data
2     message db "La suma de los valores es: ", 0 ; Mensaje inicial
3     newline db 10, 0 ; Nueva línea para la salida
4
5 section .bss
6     buffer resb 10 ; Buffer para convertir números a caracteres (hasta 10 bytes)
7
8 section .text
9     global _start
10
11 %macro DEFINE_VALUES 3
12     ; Define una "estructura" con tres valores
13     val1 db %1 ; Primer valor
14     val2 db %2 ; Segundo valor
15     val3 db %3 ; Tercer valor
16 %endmacro
17
18 %macro PRINT_STRING 1
19     ; Macro para imprimir cadenas de texto
20     mov eax, 4 ; Syscall número para 'write'
21     mov ebx, 1 ; File descriptor para stdout
22     mov ecx, %1 ; Dirección del mensaje
23     mov edx, 25 ; Longitud fija del mensaje (ajustar si es necesario)
24     int 0x80
25 %endmacro
26
27 %macro PRINT_NUMBER 1
28     ; Convierte un número en eax a caracteres ASCII y lo imprime
29     mov eax, %1 ; Carga el número en EAX
30     xor ecx, ecx ; Limpia ECX (contador de longitud)
31     mov edi, buffer + 9 ; Apunta al final del buffer (último byte)
32     mov ebx, 10 ; Divisor para obtener dígitos decimales
33
34 .convert_to_ascii:
35     xor edx, edx ; Limpia EDX para la división
36     div ebx ; Divide EAX entre 10, cociente en EAX, residuo en EDX
37     add dl, '0' ; Convierte el residuo (dígito) a ASCII
38     dec edi ; Mueve el puntero hacia atrás en el buffer
39     mov [edi], dl ; Almacena el carácter en el buffer
40     inc ecx ; Incrementa la longitud del número
41     test eax, eax ; Verifica si quedan dígitos
42     jnz .convert_to_ascii ; Si quedan dígitos, continúa
43
44     ; Imprime el número desde el buffer
45     mov eax, 4 ; Syscall para 'write'
46     mov ebx, 1 ; Salida estándar
47     mov ecx, edi ; Apunta al inicio del número en el buffer
48     mov edx, ecx ; Calcula la longitud real
49     add edx, ecx ; Ajusta el tamaño de la impresión
50     int 0x80
51 %endmacro
52
53 %macro PRINT_SUM 0
54     ; Realiza la suma de tres valores y la imprime

```

```

55     mov al, [val1]           ; Carga el primer valor en AL
56     add al, [val2]           ; Suma el segundo valor
57     add al, [val3]           ; Suma el tercer valor
58     movzx eax, al            ; Expande AL a EAX para asegurar un valor de 32 bits
59
60     ; Imprime el resultado de la suma
61     PRINT_NUMBER eax
62     PRINT_STRING newline
63 %endmacro
64
65 ; Definimos los tres valores con la macro DEFINE_VALUES
66 DEFINE_VALUES 3, 5, 7        ; Los valores son 3, 5 y 7
67
68 _start:
69     ; Imprime el mensaje inicial
70     PRINT_STRING message
71
72     ; Imprime la suma de los valores
73     PRINT_SUM
74
75     ; Salir del programa
76     mov eax, 1                ; Syscall para 'exit'
77     mov ebx, 0                ; Código de salida
78     int 0x80
79

```

## Trabajo en Clase Inline Assembly

```
1* section .data
2 | msg db "Hello, World!", 0xA ; Mensaje a imprimir seguido de un salto de línea
3 | len equ $ - msg ; Longitud del mensaje
4
5* section .text
6 | global _start ; Punto de entrada
7
8* _start:
9 | ; Llamar a write (syscall número 4 en Linux x86)
10 | mov eax, 4 ; Código de sistema para write
11 | mov ebx, 1 ; Descriptor de archivo (1 = salida estándar)
12 | mov ecx, msg ; Dirección del mensaje
13 | mov edx, len ; Longitud del mensaje
14 | int 0x80 ; Llamada al kernel
15
16 | ; Llamar a exit (syscall número 1 en Linux x86)
17 | mov eax, 1 ; Código de sistema para exit
18 | xor ebx, ebx ; Código de salida (0)
19 | int 0x80 ; Llamada al kernel
20 |
```

Output:

Hello, World!

## Taller 12

```

1 section .data
2     prompt db "Ingrese un numero: ", 0 ; Mensaje para pedir el número
3     result_msg db "El numero incrementado es: ", 0 ; Mensaje de resultado
4     newline db 10, 0 ; Nueva línea
5
6 section .bss
7     input resb 2 ; Reservar espacio para la entrada (número)
8
9 section .text
10    global _start
11
12 _start:
13     ; Mostrar mensaje para pedir el número
14     mov eax, 4 ; syscall write
15     mov ebx, 1 ; file descriptor (stdout)
16     mov ecx, prompt ; mensaje
17     mov edx, 18 ; longitud del mensaje
18     int 0x80 ; llamada al sistema
19
20     ; Leer la entrada del usuario
21     mov eax, 3 ; syscall read
22     mov ebx, 0 ; file descriptor (stdin)
23     mov ecx, input ; buffer de entrada
24     mov edx, 2 ; longitud máxima de la entrada (para un número de dos dígitos)
25     int 0x80 ; llamada al sistema
26
27     ; Convertir entrada de ASCII a número
28     movzx eax, byte [input] ; Cargar el primer byte (carácter) de la entrada
29     sub eax, '0' ; Convertir el carácter ASCII a número (restando '0')
30
31     ; Incrementar el número
32     inc eax ; Incrementar el número en 1
33
34     ; Mostrar mensaje de resultado
35     mov edx, 26 ; longitud del mensaje
36     mov ecx, result_msg ; mensaje de resultado
37     mov ebx, 1 ; file descriptor (stdout)
38     mov eax, 4 ; syscall write
39     int 0x80 ; llamada al sistema
40
41     ; Convertir el número incrementado de vuelta a ASCII
42     add eax, '0' ; Convertir el número de vuelta a ASCII
43
44     ; Mostrar el número incrementado
45     mov [input], al ; Guardar el resultado en el buffer de entrada (input)
46     mov edx, 1 ; longitud del número (un solo carácter)
47     mov ecx, input ; puntero al número incrementado
48     mov eax, 4 ; syscall write
49     int 0x80 ; llamada al sistema
50
51     ; Imprimir nueva línea
52     mov eax, 4 ; syscall write
53     mov ebx, 1 ; file descriptor (stdout)
54     mov ecx, newline ; nueva línea
55     mov edx, 1 ; longitud de la nueva línea
56     int 0x80 ; llamada al sistema
57
58     ; Salir del programa
59     mov eax, 1 ; syscall exit

```

```

59     mov eax, 1 ; syscall exit
60     xor ebx, ebx ; código de salida 0
61     int 0x80 ; llamada al sistema
62

```