



Universidad Autónoma de Baja California



Facultad de Ingeniería Arquitectura y Diseño

Taller No. 11

Organización de computadoras.

Docente: Jonatan Crespo.

Alumna: Saenz Lopez Patricia

Grupo: 932

Ensenada, Baja California, 14 de noviembre de 2024.

Taller No. 11 Macros y estructuras de datos.

Desarrollar los siguientes puntos:

- ★ En tu cuaderno desarrolla lo siguiente: Investiga sobre el funcionamiento y aplicación de macros y saltos condicionales en ensamblador y da una conclusión de al menos dos párrafos que explique la diferencia entre ellos, haciendo énfasis en las funciones de cada una.

Taller No. 11 Macros y Estructuras de Datos.

>> Macros en Ensamblador:

Los macros en ensamblador son conjuntos de instrucciones agrupadas bajo un nombre específico. Cuando se invoca una macro, el ensamblador reemplaza su nombre por las instrucciones que contiene. Esto facilita la reutilización de código y mejora la legibilidad del código. Las macros pueden recibir parámetros, lo que permite una mayor flexibilidad y adaptabilidad. Por ejemplo, una macro para inicializar una pila puede ser reutilizada en diferentes partes del programa simplemente cambiando los parámetros.

>> Saltos Condicionales en Ensamblador:

Los saltos condicionales son instrucciones que alteran el flujo de ejecución del programa basándose en ciertas condiciones. Estos se utilizan para implementar estructuras de control como bucles y ramificaciones. Por ejemplo una instrucción de salto condicional puede comparar dos valores y dependiendo del resultado, transferir el control a una etiqueta específica. Esto permite la ejecución de diferentes caminos de código según las condiciones evaluadas.

>> Conclusión:

Los macros y los saltos condicionales en ensamblador tienen roles pero distintos en programación. Los macros permiten la reutilización de código al agrupar conjuntos de instrucciones bajo un nombre específico. Esto mejora la legibilidad y el mantenimiento del código, ya que se pueden definir una vez y luego utilizar múltiples veces en diferentes partes del programa. Por otro lado, los saltos condicionales son esenciales para controlar el flujo de ejecución del programa. Basándose en condiciones específicas, estos saltos permiten al programa ejecutar diferentes secciones de código.

En conjunto ambos conceptos son cruciales para escribir programas eficientes y funcionales en ensamblador.

★ Investiga la importancia del '%' en macros en ensamblador x86.
Por ejemplo:

- Definir macros con parámetros
- Llamar a una macro
- Macros con parámetros opcionales

D 15 M 11 A 24 Scribe®

> Importancia del '%' en macros en ensamblador x86:
El símbolo '%' en macros en ensamblador x86 desempeña un papel importante, especialmente en la manipulación de registros y direcciones de memoria.

> Definir macros con parámetros:
Los macros pueden recibir parámetros que se utilizan dentro del cuerpo de la macro, por ejemplo, una macro para mover un valor a un registro podría definirse de la siguiente manera:

```
MACRO MOV-REG, reg, val
    mov %reg, %val
END MACRO
```

En este caso, '%reg' y '%val' son marcadores de posición para los parámetros que se pasarán cuando se llama a la macro.

> Llamar a la macro:
Para llamar a una macro, simplemente se utiliza su nombre seguido de los parámetros necesarios. Por ejemplo, para mover el valor 'eax' al registro 'ebx', se llamaría a la macro de la siguiente manera:

```
MOV-REG ebx, eax
```

Macros con parámetros opcionales:
Los macros también pueden tener parámetros opcionales, que se pueden omitir al llamar a la macro. Esto se puede lograr utilizando directivas condicionales dentro de la macro. Por ejemplo, una macro para implementar un registro podría definirse de la siguiente manera:

```
MACRO INC-REG, reg, val
    inc %reg
    mov %val, %reg
ENDMACRO
```



- ★ En tu computadora o cuaderno, genera estructuras de datos para simular nuevos tipos de datos como fecha dd/mm/yyyy, correo electrónico, dirección completa (compuesto de datos como calle, número de casa y colonia), una cadena de texto tipo curp. Además, añade ejemplos de cómo podías acceder, manipular y objetivos de utilizar esos tipos de datos.

≡  OneCompiler

HelloWorld.asm

```
1 ;Patricia Saenz Lopez: 1290667.
2 ;Organizacion de computadoras.
3 ;Taller No. 11, Ejercicio: 1.
4 ;15/11/24.
5
6 ;fecha (dd/mm/yyyy)
7
8 * section .data
9   |  fecha db 25, 12, 1990      ; Día, mes, año
10
11 * section .text
12   |  global _start
13
14 * _start:
15   |  ; Acceder a la fecha
16   |  mov al, [fecha]           ; Día
17   |  mov bl, [fecha + 1]        ; Mes
18   |  mov cl, [fecha + 2]        ; Año
19
20
```

☰ </> OneCompiler

HelloWorld.asm

```
1 ;Patricia Saenz Lopez: 1290667.  
2 ;Organizacion de computadoras.  
3 ;Taller No. 11, Ejercicio: 2.  
4 ;15/11/24.  
5  
6 ;correo electronico  
7  
8 section .data  
9     usuario db "usuario123", 0  
10    dominio db "dominio.com", 0  
11  
12 section .text  
13     global _start  
14  
15 _start:  
16     ; Acceder al correo electrónico  
17     lea esi, [usuario]  
18     lea edi, [dominio]  
19  
20
```

☰ </> OneCompiler

HelloWorld.asm

```
1 ;Patricia Saenz Lopez: 1290667.  
2 ;Organizacion de computadoras.  
3 ;Taller No. 11, Ejercicio: 3.  
4 ;15/11/24.  
5  
6 ;direccion  
7  
8 section .data  
9     calle db "Avenida Diamante", 0  
10    numero db "123", 0  
11    colonia db "Hidalgo", 0  
12  
13 section .text  
14     global _start  
15  
16 _start:  
17     ; Acceder a La dirección completa  
18     lea esi, [calle]  
19     lea edi, [numero]  
20     lea ebx, [colonia]  
21  
22
```

☰ </> OneCompiler

HelloWorld.asm

```
1 ;Patricia Saenz Lopez: 1290667.
2 ;Organizacion de computadoras.
3 ;Taller No. 11, Ejercicio: 4.
4 ;15/11/24.
5
6 ;curp
7
8 section .data
9   curp db "SALP021126MCHNPTA8", 0
10
11 section .text
12   global _start
13
14 _start:
15   ; Acceder a La CURP
16   lea esi, [curp]
17
18
```

☰ </> OneCompiler

HelloWorld.asm

42y875nqy

```
1 ;Patricia Saenz Lopez: 1290667.
2 ;Organizacion de computadoras.
3 ;Taller No. 11, Ejercicio: 5.
4 ;15/11/24.
5
6 section .data
7   fecha db 25, 12, 1990           ; Día, mes, año
8   usuario db "usuario123", 0
9   dominio db "dominio.com", 0
10  calle db "Avenida Diamante", 0
11  numero db "123", 0
12  colonia db "Diamante", 0
13  curp db "SALP021126MCHNPTA8", 0
14
15 section .text
16   global _start
17
18 _start:
19   ; Acceder y manipular la fecha
20   mov al, [fecha]             ; Día
21   mov bl, [fecha + 1]          ; Mes
22   mov cl, [fecha + 2]          ; Año
23   ; Aquí podrías realizar operaciones con al, bl, y cl
24
25   ; Acceder y manipular el correo electrónico
26   lea esi, [usuario]
27   lea edi, [dominio]
28   ; Aquí podrías realizar operaciones con esi y edi para manipular el correo
29
30   ; Acceder y manipular la dirección completa
31   lea esi, [calle]
32   lea edi, [numero]
33   lea ebx, [colonia]
34   ; Aquí podrías realizar operaciones con esi, edi y ebx para manipular la dirección
35
36   ; Acceder y manipular la CURP
37   lea esi, [curp]
38   ; Aquí podrías realizar operaciones con esi para manipular la CURP
39
40   ; Terminar el programa
41   mov eax, 60                ; Código de salida para "exit"
42   xor edi, edi               ; Código de salida 0
43
44
```

★ Compila el código de taller11 adjunto y documenta en tu taller mediante comentarios en el código el funcionamiento del programa.

≡  OneCompiler

HelloWorld.asm

42y875nqy 

```
1 ;Patricia Saenz Lopez: 1290667.
2 ;Organizacion de computadoras.
3 ;Taller No. 11, Ejercicio: 6.
4 ;15/11/24.
5
6 - section .data
7     num1 db 5           ; Primer número (5)
8     num2 db 11          ; Segundo número (11)
9     result db 0          ; Variable para almacenar el resultado
10    message db "Resultado: ", 0 ; Mensaje para imprimir antes del resultado
11
12 - section .bss
13     buffer resb 4       ; Reservar 4 bytes para un buffer temporal
14
15 - section .text
16     global _start
17
18 - %macro PRINT_STRING 1
19     mov eax, 4           ; Código de sistema para escribir (sys_write)
20     mov ebx, 1           ; File descriptor para salida estándar (stdout)
21     mov ecx, %1          ; Dirección del mensaje a imprimir
22     mov edx, 13          ; Longitud del mensaje
23     int 0x80             ; Llamada al sistema
24 - %endmacro
25
26 - %macro PRINT_NUMBER 1
27     mov eax, %1           ; Cargar el número en eax
28     add eax, '0'          ; Convertir el número a su representación ASCII
29     mov [buffer], eax      ; Almacenar el carácter ASCII en el buffer
30     mov eax, 4           ; Código de sistema para escribir (sys_write)
31     mov ebx, 1           ; File descriptor para salida estándar (stdout)
32     mov ecx, buffer        ; Dirección del buffer
33     mov edx, 1           ; Longitud del carácter
34     int 0x80             ; Llamada al sistema
35 - %endmacro
36
37 - _start:
38     mov al, [num1]         ; Cargar el primer número en al
39     add al, [num2]         ; Sumar el segundo número al valor en al
40     mov [result], al       ; Almacenar el resultado en la variable result
41
42     PRINT_STRING message   ; Imprimir el mensaje "Resultado: "
43     PRINT_NUMBER [result]  ; Imprimir el resultado de la suma
44
45     ; Salir del programa
46     mov eax, 1           ; Código de sistema para salir (sys_exit)
47     mov ebx, 0           ; Código de salida 0
48     int 0x80             ; Llamada al sistema
```

★ Completa los %macros en el código de taller 112 adjunto para que se ejecute de manera correcta con el objetivo de imprimir la suma de 3 números que pertenecen a una estructura de datos.

☰ < OneCompiler

🔍⚙️⚠️EDITOR CHALL

HelloWorld.asm

42y875nqy

```

1 ;Patricia Saenz Lopez: 1290667.
2 ;Organización de computadoras.
3 ;Taller No. 11, Ejercicio: 7.
4 ;15/11/24.
5
6 section .data
7     message db "La suma de los valores es: ", 0
8     newline db 10, 0           ; Nueva línea para la salida
9
10 section .bss
11     buffer resb 4          ; Buffer para convertir números a caracteres
12
13 section .text
14     global _start
15
16 %macro DEFINE_VALUES 3
17     ; Define una "estructura" con tres valores
18     val1 db %1              ; Primer valor
19     val2 db %2              ; Segundo valor
20     val3 db %3              ; Tercer valor
21 %endmacro
22
23 %macro PRINT_STRING 1
24     mov eax, 4                ; Syscall número para 'write'
25     mov ebx, 1                ; File descriptor para stdout
26     mov ecx, %1              ; Dirección del mensaje
27     mov edx, 25               ; Longitud del mensaje
28     int 0x80
29 %endmacro
30
31 %macro PRINT_NUMBER 1
32     ; Convierte un numero en eax a caracteres ASCII y lo imprime
33     mov ecx, buffer + 3        ; Apunta al final del buffer
34     mov byte [ecx], 0          ; Coloca el terminador nulo al final
35     mov eax, %1              ; Cargar el número en eax
36
37 .next_digit:
38     xor edx, edx            ; Limpia edx para la división
39     div dword ebx            ; Divide eax entre 10, cociente en eax, residuo en edx

```

☰ < OneCompiler

🔍⚙️⚠️EDITOR CHALL

HelloWorld.asm

42y875nqy

```

40     add dl, 0                ; Convierte el dígito a ASCII
41     dec ecx
42     mov [ecx], dl            ; Mueve hacia atrás en el buffer
43     test eax, eax            ; Almacena el dígito en el buffer
44     jnz .next_digit          ; Verifica si quedan dígitos
45
46     ; Imprime el número
47     mov eax, 4                ; Syscall para write
48     mov ebx, 1                ; Salida estándar
49     mov edx, buffer + 4 - ecx ; Longitud real del número
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
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

```