Taller 12 — Organización de Computadoras (Resuelto)

Objetivo: Desarrollar el uso de macros y estructuras de datos en ensamblador x86. A continuación se entregan respuestas claras, ejemplos comentados y código de muestra (con macros) para los puntos solicitados.

1) Importancia de '%' en macros en ensamblador x86 En ensamblador (NASM/YASM/MASM según sintaxis) el símbolo '%' se usa dentro de macros para indicar expresiones o para concatenación de nombres y para referirse a parámetros locales en algunos ensambladores (por ejemplo en MASM/TASM hay diferentes convenciones). En NASM, el pre-procesador utiliza '%' en directivas como `%define`, `%macro` y `%endmacro` para diferenciar instrucciones del preprocesador. Usos prácticos: - Definir macros con parámetros: `%macro NOMBRE nparams` ... `%endmacro`. - Evitar colisiones de etiquetas creando nombres únicos: usando concatenación `label%_suffix` o `%$` para generar sufijos locales. - Evaluar expresiones del preprocesador: `%assign`, `%if`, `%endif`. Ejemplo corto: %macro PUSH32 1 push dword %1 %endmacro Aquí `%1` se refiere al primer parámetro del macro.

Ejemplos de macros con parámetros y opcionales: 1) Macro con parámetros obligatorios: %macro ADD_REGS 2 ; suma el segundo operando al primero: ADD dest, src add %1, %2 %endmacro Uso: ADD_REGS eax, ebx 2) Macro con parámetro opcional (NASM permite definir cantidad variable de parámetros): %macro MOV_IF 2-3 ; MOV_IF dest, src [, condicion] mov %1, %2 %if %0 = 3 ; si hay tercer parámetro, lo usamos como comentario/condición ; condición: %3 %endif %endmacro Uso: MOV_IF eax, 5 MOV_IF ebx, eax, ;este es un tercer parámetro opcional

2) Estructuras de datos en ensamblador (simulación de tipos compuestos) A continuación se muestran ejemplos de cómo representar y acceder a tipos compuestos: fecha dd/mm/yyyy, correo electrónico, dirección y CURP. Se usan etiquetas y offsets y ejemplos de acceso. Estructura: Fecha (dd/mm/yyyy) — 3 bytes + 2 bytes para año (como palabra) fecha: dd db '05' ; ejemplo como ASCII, pero también puede ser un byte numérico slash1 db '/' mm db '11' slash2 db '/' yyyy dw 2025 ; palabra little-endian Acceso (ejemplo conceptal en código): ; cargar día (si es ASCII) mov al, [fecha] ; al = '0' o '5' ; si fuese numérico: ; mov al, [fecha_num.dd] Estructura: Correo electrónico (cadena terminada en 0) email_addr: db 'erick.l@ejemplo.com',0 Para acceder se usa la dirección de la etiqueta y operaciones con offsets. Estructura: Dirección completa (calle, número, colonia) direccion: calle db 'Av. Reforma',0 numero db '123',0 colonia db 'Centro',0 Estructura: CURP (cadena fija de 18 caracteres) curp: db 'ABCD850101HDFXXX09',0 Uso con arreglos/matrices: ; definir un arreglo de 3 fechas (cada fecha ocupa 8 bytes en el formato anterior) fechas_array: fecha0: dd 5, '/', 11, '/', 2025 ; formato mixto ilustrativo (no ensamblador literal) ; En práctica se usan offsets y etiquetas para cada campo. Observaciones: - En ensamblador se manejan offsets y punteros para simular structs. - Para manipular campos compuestos se calcula la dirección base + offset del campo. - Para strings terminados en 0 se recorren hasta encontrar 0.

3) Documentación del código en el enlace (onecompiler) Se requiere compilar el enlace https://onecompiler.com/assembly/4458pwk48. Como no se puede ejecutar aquí, se documenta qué buscar y cómo comentar: - Añade comentarios en cada sección indicando: * Qué hacen las secciones .data, .bss y .text. * Qué registros se usan y por qué. * Qué hace cada función/macro. - Si el programa usa syscalls, documenta la convención (Linux x86 int 0x80). - Indica dependencias del ensamblador (NASM, MASM, TASM) y la plataforma objetivo. Ejemplo de comentario de cabecera: ; Programa de ejemplo: realiza X ; Ensamblador: NASM ; Compilar: nasm -f elf32 prog.asm && ld -m elf_i386 -s -o prog prog.o

4) Código de ejemplo: estructura x/x/x y suma de elementos (incluye macro) Descripción: - Representamos una estructura que contiene 3 números (X/X/X). - Usamos un macro para iterar/sumar elementos y mostrar la idea. - Código en estilo NASM (comentado). No es obligatorio que compile sin cambios; está orientado a la explicación y uso didáctico.

```
; Ejemplo NASM (32-bit conceptual)
; Macro para sumar n elementos a partir de una etiqueta base
%macro SUM_ARRAY 2
    ; %1 = etiqueta_base, %2 = contador (n)
    mov esi, %1          ; puntero base en ESI
    xor eax, eax         ; acumulador
    mov ecx, %2          ; contador
.sum_loop_%=:
    cmp ecx, 0
    je .sum_end_%=
    ; suponemos elementos de 4 bytes (dd)
    mov ebx, [esi]       ; cargar elemento actual
    add eax, ebx
    add esi, 4
    dec ecx
    jmp .sum_loop_%=
.sum_end_%=:
    ; resultado en EAX
%endmacro

section .data
array_dd dd  10, 20, 30    ; estructura X/X/X ejemplo (3 elementos)
msg     db 'Suma total: ', 0

section .text
global _start
_start:
    ; llamar macro: SUM_ARRAY array_dd, 3
    SUM_ARRAY array_dd, 3

    ; aquí EAX tiene la suma (10+20+30 = 60)
    ; para propósitos de demo, terminamos el programa
    mov ebx, eax
    mov eax, 1
    int 0x80
```

Conclusión y entrega - Se incluyeron explicaciones del uso de '%' en macros, ejemplos de macros con parámetros y opcionales, estructuras de datos simuladas para fecha, correo, dirección y CURP, y un código de ejemplo que suma una estructura X/X/X usando un macro. - Si quieres que lo ajuste para un ensamblador específico (MASM, NASM, TASM) o que lo deje listo para compilar y ejecutar en Linux x86-32, lo adapto y lo pruebo. - Si necesitas diagramas, estructuras en tabla o que inserte directamente estos contenidos dentro del archivo Word/DOCX original en las posiciones exactas del taller, lo puedo pegar ahí también y devolvértelo en PDF. Entregado por: Tu compa — respuestas bien lanza, ya queda chido.