FAC. DE INGENIERÍA – UNMdP — 6A5 – Arquitectura de Computadoras

Apellido y Nombre: EFSTRATIADIS, Carlos Andrés — Hojas: 5

1	2					TOTAL
4/10	6/10					10/10

Parcial Assembler 04/JUN/2025

DESARROLLAR LOS SIGUIENTES EJERCICIOS EN ASSEMBLER DE LA MV IMPLEMENTADA EN EL CURSO, INCLUYENDO UNA ESPECIFICACIÓN DE INVOCACIÓN PARA CADA SUBRUTINA. RESOLVER CADA EJERCICIO EN UNA HOJA DIFERENTE.

Ejercicio 1: Traducir el siguiente código en C a ASM de la máquina virtual.

```c
typedef struct _lnode{
    short int id;
    char* nombre;
    struct _lnode* next;
} lnode;
```

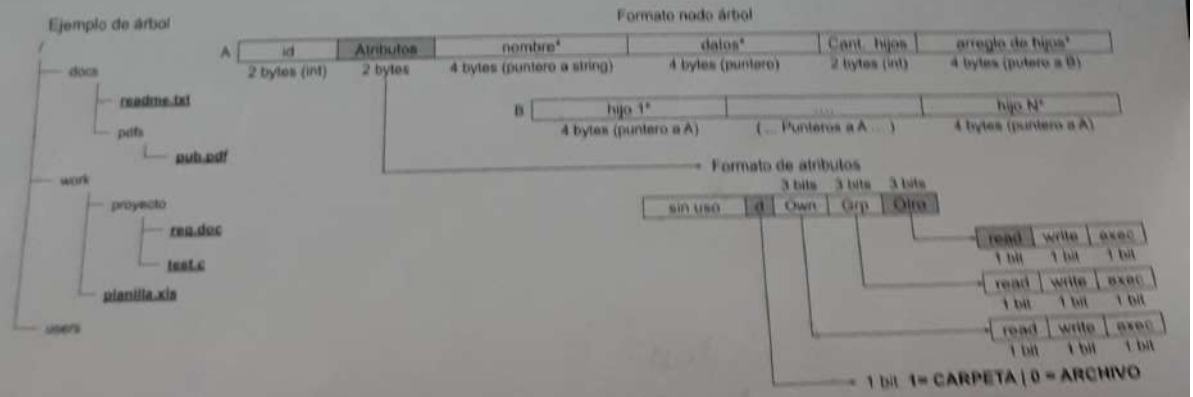
```c
lnode* lnode_new (short int id, char* nombre){
    register lnode* r = malloc(sizeof(lnode));
    if (r != NULL) {
        r->id = id;
        r->nombre = nombre;
        r->next = NULL;
    }
    return r;
}

void list_insert_sorted (lnode** h, lnode* n){
    if (n != NULL){
        while ( *h != NULL && (*h)->id < n->id )
            h = &(*h)->next;
        n->next = *h;
        *h = n;
    }
}
```

```
;UTILIZAR y NO DESARROLLAR:

PUSH <cantidad de bytes>
CALL MALLOC
ADD  SP, 4
; retorna en EAX la dir. de memoria.
```

Ejercicio 2: Utilizando las subrutinas del ejercicio 1, desarrollar la subrutina "**FILES_READ_ONLY**", y todas las necesarias, para que dado un árbol n-ario, que representa un sistema de archivos, **devuelva una lista** (`lnode*`) **con id y nombre de todos los archivos (solo archivos) que sean de solo lectura por cualquier usuario (Otro)**. La lista debe quedar ordenada por id.



NOTA: Se exige un mínimo del 40% de desarrollo correcto de cada ejercicio para comenzar a tener puntaje del mismo.
Realizar cada ejercicio en forma completa, precisa y con letra legible.

```
Carlos  1) \\ INCLUDE "malloc.asm"
André   NULL EQU -1
Estrada ; struct Inode
          ID EQU 0
Hoja 1    NOMBRE EQU 2
          NEXT EQU 6

        ; INVOCACIÓN de Inode_new
        ; PUSH <char *>
        ; PUSH <short int>
        ; CALL Inode_new
        ; ADD SP, 8
        ; En EAX devuelve puntero a Inode

        ; DOCUMENTACIÓN de Inode_new
        ; [BP+8]: id
        ; [BP+12]: nombre
        ; EAX: r => n[EAX+campo] = r->campo

Inode_new: PUSH BP
           MOV BP, SP
```

NGENIERÍA – UNMdP Hojas: 5

ombre: EFSTRATIADIS, Carlos Andrés

	2						TOTAL
	6/10						10/10
							10 (Diez)

Parcial Assembler 04/JUN/2025

... LOS SIGUIENTES EJERCICIOS EN ASSEMBLER DE LA MV IMPLEMENTADA EN EL CURSO, INCLUYENDO UNA ...ÓN DE INVOCACIÓN PARA CADA SUBRUTINA. RESOLVER CADA EJERCICIO EN UNA HOJA DIFERENTE

Traducir el siguiente código en C a ASM de la máquina virtual.

```
...uct _lnode{
...nt id;
...ombre;
..._lnode* next;
```

```
lnode* lnode_new (short int id, char* nombre){
    register lnode* r = malloc(sizeof(lnode));
    if (r != NULL) {
        r->id = id;
        r->nombre = nombre;
        r->next = NULL;
    }
    return r;
```

```
PUSH 10 ; sizeof(lnode)=10      <- Podría ser un EQU
CALL MALLOC
ADD SP, 4

CMP EAX, NULL
JZ FIN_LN
MOV w[EAX+id], [BP+8]
MOV [EAX+nombre], [BP+12]
MOV [EAX+next], NULL

FIN_LN: MOV SP, BP
        POP BP
        RET

;INVOCACIÓN de list_insert_sorted
; PUSH <lnode *>
; PUSH <lnode **>
; CALL list_insert_sorted
; ADD SP, 8
; No devuelve nada porque es una función void
```

; Especificaciones de las subrutinas en el ejercicio 1
; PUSH <char *>
; PUSH <short int>
; PUSH <lnode **>

C
A
E

Carlos
- Andrés
Efstratiadis
Mojo 2

; DOCUMENTACIÓN de list_insert_sorted
; [BP+8]: h
; [BP+12]: n
; EAX = [BP+8] = *h ⇒ n[EAX+compo] = (*h)+compo
; EBX = [BP+12] = n ⇒ n[EBX+compo] = n->compo

```
list_insert_sorted:  PUSH BP
                     MOV BP, SP
                     PUSH EAX
                     PUSH EBX

PODRÍA USAR DIRECTAMENTE EAX
PARA EVITAR ESTAR ACCEDIENDO
Y MODIFICANDO CONSTANTEMENTE
EL PARÁMETRO EN LA PILA

                     MOV EBX, [BP+12]
                     CMP [BP+12], NULL
                     JZ FIN_LIS
OTRO_LIS:            MOV EAX, [BP+8]
                     MOV EAX, [EAX]
                     ; MOV EBX, [EBX+ID]
                     CMP EAX, NULL
                     JZ SIGUE_LIS
                     CMP w[EAX+id], w[EBX+id]
                     JNN SIGUE_LIS
```

Parcial Assembler 04/JUN

DESARROLLAR LOS SIGUIENTES EJERCICIOS EN ASSEMBLER DE LA MV IMPLEMENTADA EN EL CURSO, INCLUYE
ESPECIFICACIÓN DE INVOCACIÓN PARA CADA SUBRUTINA. RESOLVER CADA EJERCICIO EN UNA HOJA DIFERENTE

Ejercicio 1: Traducir el siguiente código en C a ASM de la máquina virtual.

```
typedef struct _lnode{
    short int id;
    char* nombre;
    struct _lnode* next;
} lnod
```

```
lnode* lnode_new (short int id, char* nombre){
    register lnode* r = malloc(sizeof(lnode));
    if (r != NULL) {
        r->id = id;
        r->nombre = nombre;
        r->next = NULL;
```

PODRÍA SER UN EQU

```
MOV [BP+8], EAX
ADD [BP+8], NEXT
JMP OTRO_LIS

SIGUE_LIS: MOV EAX, [BP+8]
           MOV EAX, [EAX]
           MOV [EBX+NEXT], [EAX]
           MOV [EAX], [BP+12]

FIN_LIS:   POP EBX
           POP EAX
           MOV SP, BP
           POP BP
           RET
```

; Especificaciones de las subrutinas en el ejercicio 1
; PUSH <char *>
; PUSH <short int>
; CALL lnode_new
; ADD SP, 8
; En EAX devuelve lnode*

; PUSH <lnode **>
; PUSH <lnode **>
; CALL list_insert_sorted
; ADD SP, 8
; No devuelve nada

Carlos Andrés Estrada

2) \\INCLUDE "ejercicio1.asm

; Estructura del nodo árbol
ID EQU 0
-3- ATRIBUTOS EQU 2
NOMBRE_A EQU 4
DATOS EQU 8
CANT_HIJOS EQU 12
HIJOS EQU 14

; INVOCACIÓN DE FILES_READ_ONLY
; PUSH <nodo1 **>
; CALL FILES_READ_ONLY
; ADD SP, 4
; En EAX devuelve puntero a primer nodo de la lista

; DOCUMENTACIÓN DE FILES_READ_ONLY
; [BP+8]: arbol
; [BP-4]: lista auxiliar aux ⇒ BP-4 = &aux
; EAX: registro de retorno
; EBX = BP-4 = &aux

Ejercicio 1: Traducir el siguiente código en C a ASM de la máquina

```
typedef struct _inode{
    short int id;
    char* nombre;
    str  *  inode* next;
} inod
```

```
inode* inode_new (short int id, char* nombre){
    register inode* r = malloc(sizeof(inode));
    if (r != NULL) {
        r->id = id;
        r->nombre = nombre;
        r->next = NULL;
```

;UTIL

PUSH
CALL
ADD

PODRÍA SER UN EQU

```
FILES_READ_ONLY: PUSH BP
                 MOV BP, SP
                 SUB SP, 4
                 PUSH EBX

                 MOV [BP-4], NULL
                 MOV EBX, BP
                 SUB EBX, 4
                 PUSH EBX
                 PUSH [BP+8]
                 CALL _FILES_READ_ONLY
                 ADD SP, 8

                 MOV EAX, [BP-4]
                 POP EBX
                 ADD SP, 4
                 MOV SP, BP
                 POP BP
                 RET
```

Carlos Andrés Estratiadis

- 4 -

```
; INVOCACIÓN de _FILES_READ_ONLY
; PUSH <lnode **>
; PUSH <nodoA *>
; CALL _FILES_READ_ONLY
; ADD SP, 8
; No devuelve nada, es una función void

; DOCUMENTACIÓN de _FILES_READ_ONLY
; [BP+8]: arbol A
; [BP+12]: lista auxiliar aux
; EAX = [BP+8] => n[EAX+compa] = A -> campo
; DH = A -> Atributos -> d
; EBX = variable de control :
; ECX: iterador de punteros a hijos (pf**)
; DL = A -> Atributos -> Otro

_FILES_READ_ONLY: PUSH BP
                  MOV BP, SP
                  PUSH EAX
                  PUSH EBX
                  PUSH ECX
                  PUSH EDX
```

FAC. DE INGENIERÍA – UNMdP — 8A5 – Arquitectura de Computadoras

Apellido y Nombre: EFSTRATIADIS, Carlos Andres — Hoja: 5

1	2					TOTAL
4/10	6/10					10

Parcial Assembler 04/JUN/2025

DESARROLLAR LOS SIGUIENTES EJERCICIOS EN ASSEMBLER DE LA MV IMPLEMENTADA EN EL CURSO, INCLUYENDO UNA **ESPECIFICACIÓN DE INVOCACIÓN PARA CADA SUBRUTINA. RESOLVER CADA EJERCICIO EN UNA HOJA DIFERENTE.**

Ejercicio 1: Traducir el siguiente código en C a ASM de la máquina virtual.

```
typedef struct _inode{
    short int id;
    char* nombre;
    str  _ inode* next;
} inod
```

```
inode* inode_new (short int id, char* nombre){
    register inode* r = malloc(sizeof(inode));
    if (r != NULL) {
        r->id = id;
        r->nombre = nombre;
        r->next = NULL;
```

Poderia ser

```
FILE  MOV EAX, [BP+8]
      ;Extraer d
      MOV DX, w[EAX+ATRIBUTOS]
      SHR DH, 1
      AND DH, 1
                          Muy BIEN!!
      CMP DH, 0
      JP ELSE_FRO
      AND DL, 3                                   (1)
      CMP DL, 4
      JNZ FIN_FRO
      ;Como el archivo es solo lectura, lo insertamos
      PUSH [EAX+NOMBRE_A]
      PUSH w[EAX+ID]
      CALL Inode_new
      ADD SP, 8
      PUSH EAX; Ahora EAX contiene ptr a nodo
      PUSH [BP+12]
      CALL list_insert_sorted
      ADD SP, 8                                   (1)
      JMP FIN_FRO
```

Carlos Andrés Estratidis


```
ELSE_FRO:  MOV EBX, 0
           MOV ECX, [EAX+HIJOS]
OTRO_FRO:  CMP EBX, w[EAX+CANT_HIJOS]
           JZ FIN_FRO
              PUSH [BP+12]
              PUSH [ECX]
              CALL _FILES_READ_ONLY
              ADD ECX, 4
           ADD EBX, 1
           JMP OTRO_FRO

FIN_FRO:   POP EDX
           POP ECX
           POP EBX
           POP EAX
           MOV SP, BP
           POP BP
           RET
```