FAC. DE INGENIERÍA – UNMdP **16/DIC/2024** Arquitectura de Computadoras

Apellido y Nombre: Delgado, Facundo NOTA FINAL 5 (CINCO) Hojas: 0

1	2						TOTAL
40%	60%						100%
B	R					6 (Seis)	B

1) Escriba las microinstrucciones de la arquitectura horizontal, necesarias para implementar la macroinstrucción POPI (Pop Indirect) cuya función es sacar el valor que está en el tope de la pila y almacenarlo en la celda de memoria apuntada por el registro AC.

AMUX	COND	ALU	SH	MBR	MAR	RD	WR	ENC	C	B	A	ADDR
0	0	2	0	0	1	1	0	0	0	2	0	0 mbr := [sp]
0	0	0	0	0	0	1	0	0	0	0	0	0
1	0	2	0	1	1	0	1	0	0	1	0	0 [ac] := mbr
0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	1	2	2	6	0 sp := sp+1

AMUX
0 = Buffer A
1 = MBR

COND
0 = No salta
1 = Si N==1
2 = Si Z==1
3 = Salta siempre

ALU
0 = A + B
1 = A & B
2 = A
3 = ~A

SH
0 = No desplaza
1 = alu >> 1
2 = alu << 1
3 = (no usado)

MBR, MAR, RD, WR, ENC
0 = No
1 = Si

REGISTROS
PC=0, AC=1, SP=2
IR=3, TIR=4.
K_0=5, K_1=6, K_{-1}=7
AMask=8, SMask=9
AX=10... FX=15

① No hace falta tomar el valor de MBR y volver a cargarlo, se puede usar directamente para el WR

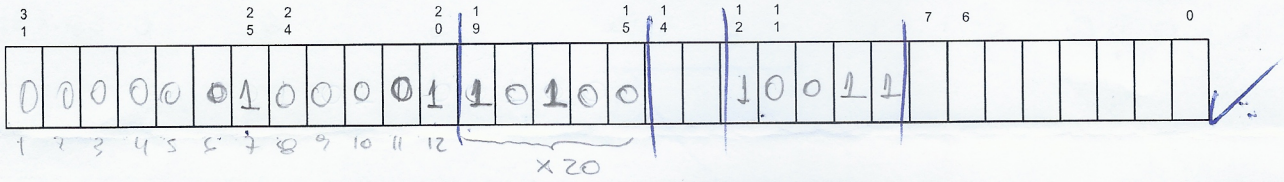
NOTA: Se exige un mínimo del 40% de desarrollo correcto de cada tema para comenzar a tener puntaje en dicho tema.
Responder cada tema en forma completa, precisa y con letra legible.

2) Para una arquitectura RISC **RV32I**, codifique los argumentos de las siguientes instrucciones utilizando el formato adecuado. En cada caso, **justifique** la elección del formato empleado.
Los registros se indican como Xnn (por ejemplo: X19) y las constantes se expresan en hexadecimal (por ejemplo: %4). Los campos **funct7, funct3 y opcode**, se deberán dejar en blanco.

Formato R	funct7	rs2	rs1	f3	rd	opcode
Formato I	imm		rs1	f3	rd	opcode
Formato L	imm		rs1	f3	rd	opcode
Formato S	imm	rs2	rs1	f3	imm	opcode
Formato B	imm	rs2	rs1	f3	imm	opcode

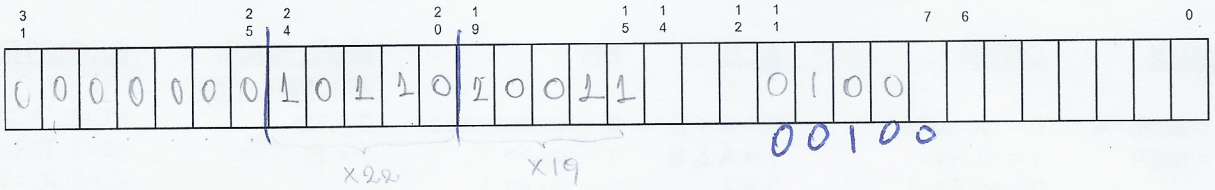
a) Addi X19,X20,%21 (Suma destino,op,op) Formato usado: I

Justificación: Se utiliza este formato de operación con inmediatos, para formar el inmediato se utiliza funct7 + rs2 (12 bits)



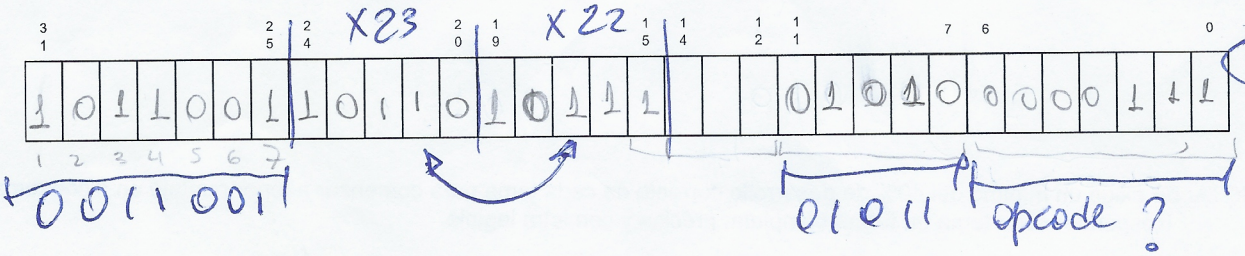
b) Sb X22,%4(X19) (Almacenamiento origen,destino) Formato usado: S

Justificación: Se utiliza este formato de almacenamiento en memoria
Se forma con el inmediato como offset (funct7 + rd) 12 bits y r1 como base



c) Beq X22,X23,%B2A (Salto op,op,offset) Formato usado: B

Justificación: Se utiliza el formato branch parecido al Storage
Solo que se usa el inmediato con offset (saltos relativos)



Apellido y Nombre: Delgado, Facundo Hojas: 2

1	2						TOTAL
4/10	6/10						10/10
R	R						4 (cuatro)

Totalizador Assembler 16/DIC/2024

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.
AYUDA: los structs son referencias, no se traducen a assembler.

```c
typedef struct _node{
    short int value;
    void* data;
    struct _node *next;
} node;
```

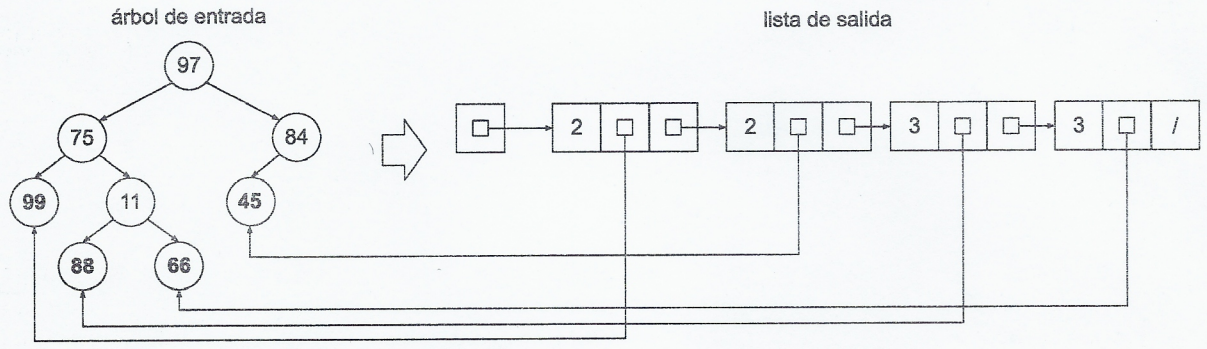
```
;UTILIZAR y NO DESARROLLAR:

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

```c
node* node_new (short int value, void* data){
    register node* new = (node*)malloc(sizeof(node));
    if (new != NULL){
        new->value = value;
        new->data = data;
        new->next = NULL;
    }
    return new;
}

void add_sort (node** head, node* new){
    while (*head != NULL && (*head)->value < new->value){
        *head = &(*head)->next;
    }
    new->next = *head;
    *head = new;
}
```

Ejercicio 2: Utilizando las subrutinas del ejercicio 1, desarrollar la subrutina "**GET_LEAFS**", y todas las necesarias, para que dado un árbol binario (**NO ordenado**), **devuelva una lista (nodo*) que contenga los punteros a todas las hojas del árbol y se encuentre ordenada por el nivel de cada hoja.** Ejemplo:



Formato del nodo del árbol:

	2 bytes	4 bytes	4 bytes
A	clave	puntero a izquierda	puntero a derecha
	Entero	Dir. Mem. tipo A	Dir. Mem. tipo A

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.

EL PARÁMETRO MÁS CERCANO ES EL ÚLTIMO PUSH ⊛

HOJA N° P1

FECHA

Facundo Delgado

ALREVÉS

1)

```
value equ 8
NULL EQU -1
data equ 10
node_new:
```

✗ *INCLUDE <MALLOC>

EL PUSH SIEMPRE AGREGA 4 BYTES

✗ 12

Invocación:

DEBERÍAN ESTAR AL REVÉS

```
push <numero 2 bytes>
push <puntero a un dato>
call node_new
add SP, 6
```

```
push bp
mov bp, sp

push EBX
push ECX

mov EBX, W[Bp + value]
mov ECX, [Bp + data]
```

USO INNECESARIO DE REGISTROS

```
mov EAX, NULL
```

✗ NO ES LO QUE HACE EL CÓDIGO C

```
push 10
call MALLOC
add SP, 4

CMP EAX, NULL
JZ fin_node_new
```

NO EXISTE

```
mov W[EAX], EBX
mov [EAX+2], ECX
mov [EAX+6], NULL

pop ECX
pop EBX
mov SP, bp
pop bp
RET
```

NOTA

```
head EQU,8
new  EQU,12
```

AL REVÉS → ÍDEM ⊛

```
add_sort:
    push bp
    mov bp,sp
    push EAX
    push EBX
    push ECX

    mov  EAX, [BP+ head]
    mov  EBX, [BP + data]

while_add_sort:
    CMP [EAX], NULL
    JZ  fin_while_add_sort
    mov ECX W[EAX]
    CMP W[ECX] , W[EBX]
    JNN fin_while_add_sort
    add ECX,6
    mov EAX, ECX
    JMP while_add_sort
fin_while_add_sort:
    mov [EBX+6], [EAX]
    mov [EAX], EBX

    pop ECX
    pop EBX
    pop EAX
    mov sp, bp
    pop bp
    RET
```

ES UN PUNTERO A UN PUNTERO
EL CONTENIDO TIENE 4 BYTES

Invocación

```
push <puntero a cabeza de la lista>
push <puntero nuevo nodo>
call add_sort
add sp, 8
```

NOTA

HOJA N° P2

FECHA

Facundo Delgado

$r_2 = r_1 - r_2$

R-

2) GET_LEAFS: (root, lista, nivel)

```
push BP
mov bp, sp

push EBX
push ECX
push EDX

mov EBX, [Bp + 8] ; root
mov ECX, [Bp + 12] ; lista
mov DX, W[Bp + 16] ; nivel
CMP [EBX], NULL
JZ fin_get_leafs
push ECX
push [EBX+2]
call GET_LEAFS
add SP, 8
push ECX
push [EBX + 6]
call GET_LEAFS
add SP, 8

CMP [EBX+2], NULL
JNZ fin_get_leaf
CMP [EBX+6], NULL
JNZ fin_get_leaf

push EBX ; puntero a la hoja
push DX ; nivel
call node_new
add sp, 6
push EAX ; *new
push ECX ; **head
call add_sort
add sp, 8

fin_get_leafs:
pop EDX
pop ECX
pop EBX
mov sp, bp
pop bp
```

main:

```
...
push 0
push <puntero al la cabeza de la lista>
push <puntero al arbol>
call get_leafs
add SP, 12
stop
```

debe ser Doble -

FALTA Push "Nivel"

No Aumento el Nivel en la ~~recursividad~~ ~~sobre~~ invocación Recursiva

FALTA Push

No Respeto Protocolo

Se debe computar primero si es una hoja Así No se hace ~~una~~ llamadas de más -

debe ser doble puntero

- No CREA LA LISTA (envio como parám un puntero simple)

Debería usar una subrutina que cree la lista llamar a otra subrutina ~~recursiva~~ RECURSIVA

NOTA