

Nº Orden	Apellido y nombre	L.U.	Cantidad de hojas

## Organización del Computador 2

### Primer parcial – 11/05/17

1 (35)	2 (35)	3 (30)	
--------	--------	--------	--

#### Normas generales

- Numere las hojas entregadas. Complete en la primera hoja la cantidad total de hojas entregadas.
- Entregue esta hoja junto al examen, la misma **no** se incluye en la cantidad total de hojas entregadas.
- Está permitido tener los manuales y los apuntes con las listas de instrucciones en el examen. Está prohibido compartir manuales o apuntes entre alumnos durante el examen.
- Cada ejercicio debe realizarse en hojas separadas y numeradas. Debe identificarse cada hoja con nombre, apellido y LU.
- La devolución de los exámenes corregidos es personal. Los pedidos de revisión se realizarán por escrito, antes de retirar el examen corregido del aula.
- Los parciales tienen tres notas: I (Insuficiente): 0 a 59 pts, A- (Aprobado condicional): 60 a 64 pts y A (Aprobado): 65 a 100 pts. No se puede aprobar con A- ambos parciales. Los recuperatorios tienen dos notas: I: 0 a 64 pts y A: 65 a 100 pts.

### Ej. 1. (35 puntos)

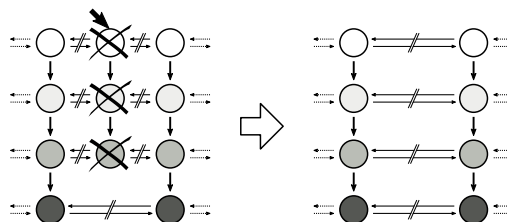
Sea la siguiente estructura de listas doblemente enlazadas encadenadas entre si.

```
struct supernode {
    supernode* abajo,
    supernode* derecha,
    supernode* izquierda,
    int dato };
```

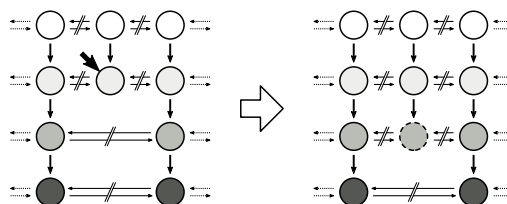
- todos los nodos pertenecen a una lista doblemente enlazada
- todos los nodos son referenciados desde algun otro nodo en otra lista (excepto en la primera)
- todas las listas respetan el orden de los nodos que las apuntan

Implementar en ASM las siguientes funciones.

- (20p) a. `void borrar_columna(supernode** sn)`: Dada un doble puntero a nodo dentro de la primera lista, borra una columna de nodos. Modifica el doble puntero dejando un nodo valido de la primer lista.



- (15p) b. `void agregar_abajo(supernode** sn, int d)`: Agrega un nuevo nodo a la lista inmediata inferior del nodo apuntado. Considerar que el nodo donde agregar puede no tener vecinos inmediatos en la lista inferior.



## Ej. 2. (35 puntos)

Un pixel es codificado como 3 componentes RGB.

En una codificación particular se utiliza 5 bits para R, 6 para G y 5 para B, almacenados en 2 bytes.

R					G						B				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Se tiene una imagen de  $m \times n$  pixeles almacenados en esta codificación, con  $n$  y  $m$  multiplos de 8.

Implementar en ASM usando instrucciones de SIMD las siguientes funciones:

- (20p) a. `void to24(img16* src, int m, int n, img24** dst)`: Convierte una imagen almacenada en `src` a pixeles de 24 bits (1 byte por componente en orden RGB). Almacenando el resultado en `dst`.  
Cada componente debe ser escalado según se indica a continuación:

Componente R o Componente B					0		
7	6	5	4	3	2	1	0

Componente G						0	
7	6	5	4	3	2	1	0

- (15p) b. `void toBN(img16* src, int m, int n, img8** dst)`: Convierte una imagen almacenada en `src` a pixeles de una sola componente de 8 bits, cada pixel corresponde al promedio de cada componente escalada a 8bits según muestra el ejercicio anterior, es decir  $(R + G + B)/3$ . Almacenando el resultado en `dst`.

En ambas funciones se debe solicitar memoria donde guardar el resultado y retornar el puntero por `dst`.

## Ej. 3. (30 puntos)

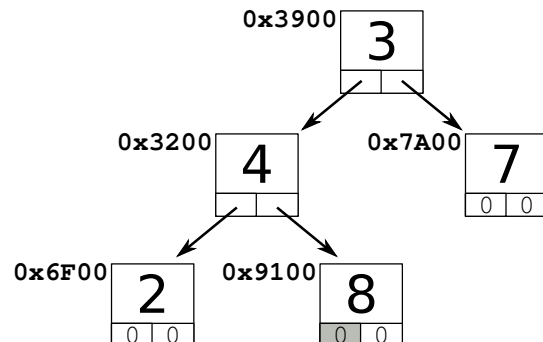
Considerando el código a continuación, que realiza la sumatoria de todos los valores dentro de un arbol binario, respetando el siguiente `struct`: `struct node { node* izq, node* der, int var }`

```

0xA000| suma: push    rbp
0xA001|             mov    rbp, rsp
0xA004|             push   rbx
0xA005|             push   r12
0xA007|             mov    rbx, rdi
0xA00a|             mov    eax, 0
0xA00f|             cmp    rdi, 0
0xA013|             je     .fin
0xA015|             mov    r12d, [rbx + 16]
0xA019|             mov    rdi, [rbx]
0xA01c|             call   suma
0xA021|             add    r12d, eax
0xA024|             mov    rdi, [rbx + 8]
0xA028|             call   suma
0xA02d|             add    eax, r12d
0xA030| .fin: pop     r12
0xA032|             pop    rbx
0xA033|             pop    rbp
0xA034|             ret

```

Caso:



Notar que cada caja es un nodo, los numeros en hexadecimal corresponden a sus direcciones en memoria.

- (15p) a. Dibujar el estado de la pila en **hexadecimal** para la ejecución del algoritmo `suma` sobre el arbol de la figura. Se debe dibujar la pila hasta el momento en que el algoritmo es llamado con el puntero al calor en gris de la figura.
- (15p) b. Construir una función en ASM que dado un puntero al tope de la pila, devuelve la profundidad del arbol recorrido hasta el momento. Considerar que se ejecuta la función `suma` para cualquier arbol y que el puntero fue obtenido obtenido luego de crear el `stack frame`.