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

Organización del Computador 2
Primer parcial – 05-05-2015

1 (40)	2 (40)	3 (20)	

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.

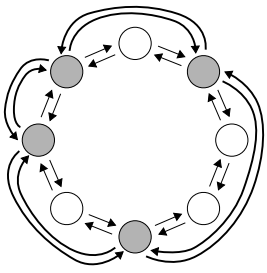
Ej. 1. (40 puntos)

Se tiene una estructura de lista circular que almacena información sobre procesos. La misma es una lista doblemente encadenada entre un conjunto de nodos, que a su vez contienen otra lista doblemente encadenada entre algunos de estos nodos. Esta última también es circular.

```
struct node {
  int   id;
  node* next;
  node* prev;
  node* superNext;
  node* superPrev;
  info* data;
}
```

La estructura contiene dos pares de punteros: `next` y `prev` que construyen la lista circular entre todos los nodos, y `superNext` y `superPrev` que construyen la otra lista entre algunos de estos nodos denominada *super*.

Además, los nodos contienen un puntero `data` que es utilizado para almacenar la información sobre el proceso, y un identificador denominado `id`.



(10p) (a) Implementar en ASM la función borrar (`info* borrar(node* n)`), que borra el nodo indicado manteniendo el invariante de las dos listas y retorna el puntero a la información que el nodo contenia.

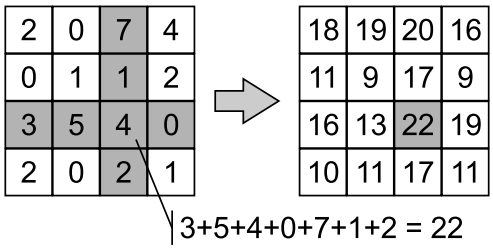
(15p) (b) Implementar en ASM la función agregarSuper (`void agregarSuper(node* n)`), que toma un nodo `n` que pertenece a la lista y lo agrega a la lista *super*.

(15p) (c) Implementar en ASM la función estanTodos (`int estanTodos(node* n)`), que retorna 1 si la lista *super* contiene a todos los nodos.

Nota: La lista circular tiene al menos 31 nodos y la lista *super* contiene al menos 21 nodos antes de llamar a cualquiera de las funciones.

Ej. 2. (40 puntos)

Se tiene una matriz de 4×4 enteros sin signo de 32bits. Se desea implementar una función denominada `sumatorias` que se encarga de generar una nueva matriz que contiene por cada elemento la sumatoria de todos los elementos en la fila y columna de la matriz original, contando solo una vez cada uno de los números.



(30p) (a) Implementar en ASM utilizando instrucciones de SIMD la función `sumatorias` (`unsigned int* sumatorias(unsigned int* matriz)`). Procesar la mayor cantidad posible de elementos, justificar.

(10p) (b) Considerando que la matriz de entrada no contiene enteros, sino *float*s y que en la nueva matriz resultado se deben almacenar enteros de 32bits, modifique el código anterior para reflejar este cambio (puede reescribir todo el código o una parte del mismo).

Ej. 3. (20 puntos)

Por cuestiones de seguridad se busca implementar una nueva convención para la utilización de la pila. En esta nueva convención se poseen dos pilas independientes que almacenan por un lado direcciones de retorno y por otro, datos.

(10p) (a) Explique como implementaría una convención con estas características. ¿Utilizaría variables globales?. Explique detalladamente como construir y deshacer el *stack-frame* y cómo llamar a una función. Utilice gráficos y código según corresponda. Recordar que se debe respetar la convención C y la alineación de memoria.

(10p) (b) Implementar en ASM la siguiente función C, respetando la convención explicada en el punto anterior.

```
int convertAndAdd(int a, int b) {
   int x, y, z;
   x = convert(a);
   y = convert(b);
   z = x + y;
   return z;
}
```