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

Organización del Computador 2
Primer parcial – 10/05/18

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.

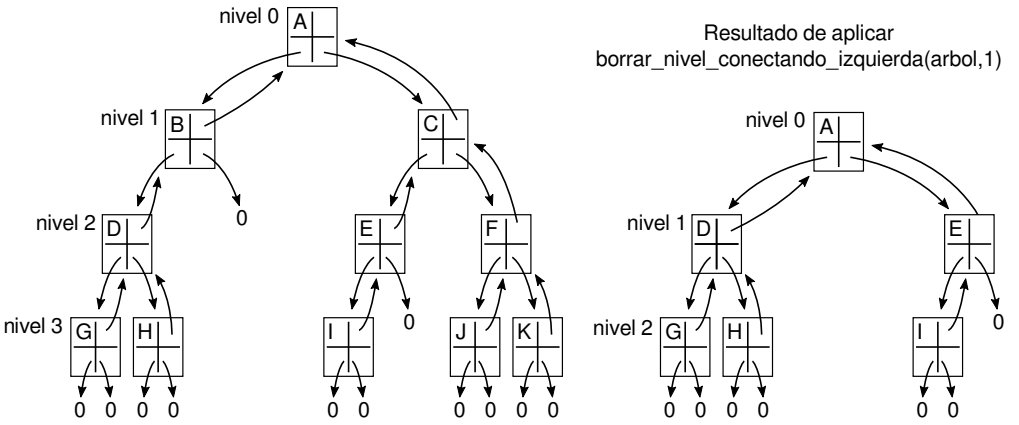
Ej. 1. (40 puntos)

Sea un árbol binario doblemente enlazado que respeta la siguiente estructura:

```
struct nodo {
            struct nodo* derecho,
            struct nodo* izquierdo,
            struct nodo* padre,
            void* data,
            void (*borrar)(void*)
            }
```

(15p) a. Programar en ASM la función `ContarPorNivel` que dado un puntero a nodo y un número de nivel, cuenta la cantidad de nodos que hay en el nivel indicado. El resultado es retornado en un puntero a entero denominado `cantidad`. Su aridad es: `void contar_por_nivel(struct nodo* arbol, unsigned int nivel, unsigned int* cantidad)`.

(25p) b. Programar en ASM la función `borrar_nivel_conectando_izquierda` que dado un doble puntero a nodo y un número de nivel, borra todos los nodos del nivel indicado, conectando al padre solamente los hijos izquierdos. El subarbol derecho debe ser eliminado. Su aridad es: `void borrar_nivel_conectando_izquierda(struct nodo** arbol, unsigned int nivel)`. Considerar que el puntero al primer nodo puede cambiar y debe ser retornado en el parámetro `arbol`.



Nota: Para borrar `data` se debe llamar a la función almacenada en el nodo en `borrar`. La misma tienen la misma aridad que la función `free`.

Ej. 2. (40 puntos)

Considerar un vector de 16 números enteros con signo de 24 bits almacenados en big-endian.

(25p) a. Construir una función en ASM utilizando SIMD que dado un puntero al vector de números mencionado, retorne la suma de los números del vector como un entero de 4 bytes.

(15p) b. Modificar la función anterior para que a los números pares los multiplique por π. En este caso el resultado debe ser retornado como *double*.

Ej. 3. (20 puntos)

Una nueva opción de compilación de `gcc` hace que luego de cualquier instrucción `call` se generen exactamente `k` bytes con *null*. Este lugar es utilizado por herramientas de *debugging* para almacenar información especial. El código resultante sería de la forma:

```
...|inst|inst|inst|call| k bytes |inst|inst|inst|...
```

El problema aparece cuando las funciones finalizan. Cuando se ejecute la instrucción `ret` se retornará a la dirección siguiente al `call`. En el código con la nueva opción de compilación, esto resultaría en la ejecución del área reservada de `k` bytes.

Para evitar este problema se pide construir una función que "arregle" la dirección de retorno. Esta función será ejecutada dentro de cada función llamada, en cualquier momento luego de construir el *stack-frame*.

(6p) a. Programar en ASM la función `arregla_retorno` que modifica la dirección de retorno de una función para evitar ejecutar el área reservada. El valor `k` es un parámetro para `arreglar_retorno`.

(7p) b. Programar en ASM una función alternativa (`modificar_area_reservada`), que a diferencia de modificar la dirección de retorno, escribe instrucciones de *no-operacion* en el área reservada a fin de continuar con la correcta ejecución en caso de retornar sobre el área reservada. La instrucción `NOP` se codifica como `0x90` y ocupa 1 byte.

(7p) c. Decidir verdadero o falso y justificar: Desensamblar un archivo binario sin ningún tipo de información de debug, de código generado con la nueva opción de compilación, no presenta ningún problema adicional que desensamblar código que no utilice la mencionada opción.

Nota: `arreglar_retorno` y `modificar_area_reservada` son llamadas como cualquier otra función.