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

Organización del Computador 2
Recuperatorio del Primer Parcial – 29/11/2016

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)

Considerar un árbol binario de búsqueda con la siguiente estructura.

```
typedef struct node_t {
   node *izquierda;
   int value;
   node *derecha;
} node;
```

y un conjunto de funciones con la siguiente aridad: `typedef int func(node *a, node *b)`

1. (15p) Construir en ASM una función `void expandir(node* arbol, func f)` que evalue `f` para cada par padre-izq del árbol, y agregue un nodo entre ambos con el resultado.

2. (25p) Construir en ASM una función `void comprimir(node *nodo)` para recorra todo el árbol realizando la siguiente acción: si el hijo de un nodo tiene exactamente un hijo, entonces será eliminado (conectando al nodo con su nieto).
   Recomendación: utilizar una función auxiliar `void comprimir_aux(node **ppHijo)`.

Ej. 2. (40 puntos)

Dado un arreglo de 64 ints, se desea implementar una función en ASM usando SIMD que genere un hash de los valores. La función de hash a implementar es la siguiente:

$$f(A_i, k) = \begin{cases} 3A_i + 1 & \text{si } k\pi A_i > 0 \\ \left\lfloor \frac{A_i}{2} \right\rfloor & \text{si no} \end{cases}$$

1. (25p) Programar en ASM usando SSE la función `hashear(int *arreglo, uint k)`
2. (15p) Cambiar el código anterior para que la condición sea si $k\pi A_i > 100$

Ej. 3. (20 puntos)

Los creadores del estandar de C están intentando agregar una nueva funcionalidad al lenguaje. La misma es una función `void restart(int n)`, que al ser llamada reinicia la ejecución de la n-ésima función encontrada en el stack. Para ello, la función restart asumirá que todas las funciones al llamar a otra pasarán todos los argumentos por la pila, y que las llamadas a función se harán con las siguientes instrucciones:

```
mov rdx, func
push rdx
call rdx
```

1. (5p) Escribir una función `void ciclo(int k)` que imprima desde k hasta 0 sin usar jumps, solo llamando a `restart`.
2. (15p) Programar la función `restart`.