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

Organización del Computador 2
Recuperatorio del Primer parcial – 25-11-2014

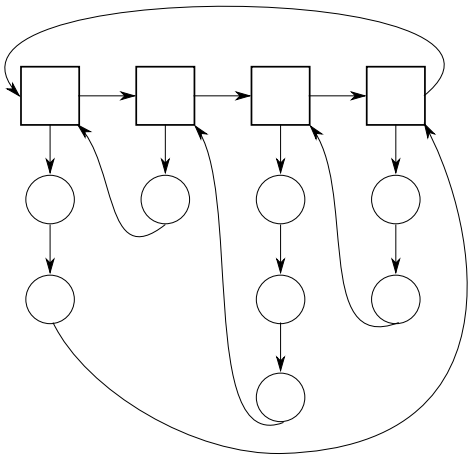
1 (40)	2 (40)	3 (20)	

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 las estructuras `struct cuadrado { circulo *hijo; cuadrado *siguiente}` y `struct circulo { void *siguiente, void *dato; bool es_ultimo };`. Las mismas se utilizan para formar listas de la forma ejemplificada en la siguiente figura.



Observar que el campo `siguiente` en `circulo` puede apuntar tanto a un `circulo` como a un `cuadrado` dependiendo del estado de `es_ultimo`.

(20p) (a) Implementar en ASM la función `void borrar_cuadrado_e_hijos(cuadrado *target)`, que borra tanto al cuadrado como a sus hijos

(20p) (b) Implementar en ASM la función `float recalcular(cuadrado *target, float(*fun)(circulo*))`, que recorre todos los circulos de todos los cuadrados aplicando `fun` y devuelve la suma de todos los resultados.

Ej. 2. (40 puntos)

Se posee una imagen de 512×512 pixeles, con los bytes de la forma B_0,G_0,R_0,A_0, B_1, G_1, R_1, A_1, ... y se desea aplicar un filtro usando SSE que modifique *solamente* los componentes verdes de la siguiente manera:

$$P_{g\_out}[i,j] = \begin{cases} Q \cdot P_{g\_in}[i,j] & \text{si } P_{g\_in}[i,j] > 128 \\ 128 & \text{si no} \end{cases}$$

donde Q es un parámetro de punto flotante entre 0.0 y 10.0, el resultado de la multiplicación debe saturarse en 255, y el resto de los píxeles debe quedar igual. La función a implementar es `modificar_verdes(unsigned char *in, unsigned char *out, float Q)`

(20p) (a) Escribir un fragmento de código de sólo una iteración del ciclo y explicar el estado inicial de los registros. Este código será citado en el próximo ejercicio.

(20p) (b) Escribir el código completo de la función `modificar_verdes`.

Ej. 3. (20 puntos)

Cuando se llama funciones sucesivamente estas modifican la pila construyendo el *stack frame* y almacenando información local. Considerando que todas las funciones respetan la siguiente estructura: `push rbp | mov rbp, rsp | ... | pop rbp | ret` y que además siempre son llamadas mediante la instrucción `call`.

(2p) (a) Mostrar en un gráfico la información almacenada en la pila luego de llamar sucesivamente a tres funciones anidadas.

(12p) (b) Construir una función en ASM que devuelva la dirección del stack frame que tiene más datos almacenados.

(6p) (c) Construir una función en ASM que obtenga el puntero al comienzo de la función que mas datos tiene almacenados en la pila.

Nota: Considerar que la instrucción `call` ocupa exactamente 9 bytes. El primero corresponde al *opcode* y los cuatro restantes a la dirección absoluta de la función a llamar.