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

Organización del Computador 2
Primer parcial – 01/10/2015

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 desea aplicar un filtro laplaciano para detección de bordes sobre una imagen en escala de grises y un filtro de desenfoque en una imagen en colores. La imagen en escala de grises se representa como una matriz de pixeles, donde cada pixel contiene una única componente de 8 bits que representa la intensidad de blanco. Un valor de 255 indica el color blanco, mientras que el valor 0 representa el color negro. La imagen en colores también se representa como una matriz de pixeles, pero estos contienen 4 componentes, cada una con los valores de intesidad de Rojo (R), Verde (G), Azul (B) y Transparencia (A).

Los filtros se aplican utilizando un kernel de convolución.

Realizamos la convolución por separado para cada componente k.

$$O[i,j,k] = (I * K)[i,j,k] = \sum_{x=-r}^{r} \sum_{y=-r}^{r} I[i+x, j+y, k]\, K[r-x, r-y] \tag{1}$$

donde $I_{w \times h}$ es una imagen de ancho w y alto h, $K_{n \times n}$ es el *kernel* de convolución y r es el radio del *kernel* de convolución ($r = \frac{n-1}{2}$)

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

1) Operador Laplaciano

$$\begin{bmatrix} 0{,}0625 & 0{,}125 & 0{,}0625 \\ 0{,}125 & 0{,}25 & 0{,}125 \\ 0{,}0625 & 0{,}125 & 0{,}0625 \end{bmatrix}$$

2) Operador de desenfoque

Se desea aplicar estos filtros utilizando instrucciones SSE y programando de acuerdo al modelo SIMD. Suponiendo que en `rax` se encuentra el puntero al elemento `[0, 0]` del kernel de convolución, en `rbx` el puntero al elemento `[1, 1]` de la imágen y en `rcx` el ancho de la imagen. Se pide responder la siguiente pregunta justificando *detalladamente* e incluyendo el código ASM que realiza la operación:

¿Cuántos elementos procesa en simultaneo en cada uno de los siguientes casos? ¿Podría procesar más?

a) Aplicar una iteración del operador laplaciano indicado en 1) a una imagen en escala de grises.

b) Aplicar una iteración del operador de desenfoque indicado en 2) a una imagen en colores.

Importante

Desarrolle en detalle y justifique su respuesta a cada caso. La justificación debe realizarse anteponiendo una breve explicación a cada etapa del código ASM (analisis de tipo/tamaño de los datos, empaquetado/despempaquetado, conversiones, operaciones aritmeticas, etc.)

Ej. 2. (40 puntos)

Considerar la estructura `listaDeNombres` que representa la lista de todos los nombres registrados en el RE.NA.PER. como lista de strings almacenados en el campo `nombre` de sus respectivos nodos. La lista se implementa a partir de enlazar de forma simple los nodos entre sí. La única estructura que compone a la lista será la siguiente:

```
typedef struct {
    nodo* siguiente;
    char* nombre;
} __attribute__((__packed__)) nodo;
```

(a) **(15p)** Implementar en ASM la función `char* mayusculizar(nodo *n)`, que realiza la siguiente acción. Tomando como base el nombre almacenado en el nodo pasado, crea dos nuevos strings: uno que contiene las letras mayusculas y por otro las minúsculas (los dos nuevos strings mantienen el orden de los caracteres). Las letras mayusculas deben reemplazar al nombre original en el nodo y las minusculas devolverse como resultado de la función.

Los caracteres posibles son letras mayúsculas o minúsculas. Las minúsculas se encuentran entre los números 97 (`a`) y 122 (`z`). Las mayeusculas entre los números 65 (`A`) y 90 (`Z`)

(b) **(25p)** Implementar en ASM la función `void mayusculizarLista(nodo *n)`, que mayusculiza cada nodo de la lista, insertando entre cada uno un nuevo nodo que contenga el string con las letras minúsculas, siempre y cuando éste no sea vacío.

Nota: Todos los strings utilizan memoria dinámica. Recordar liberar la memoria si es necesario.

Ej. 3. (20 puntos)

Sea la estructura `listaDeDatos`, definida como una lista simplemente enlazada de nodos que contienen punteros a una estructura desconocida en sus respectivos nodos. Se sabe que la estructura desconocida guarda valores de punto flotante.

```
typedef struct {
    nodo* siguiente;
    short codigo;
    char tipo;
    int id;
    float valor;
    estructura_magica* nombre;
} nodo;
```

(a) **(20p)** Suponiendo que existe la función
`void minmaxsumcount(estructura_magica *, double * max, double * min, double * sum, short * n)`
que itera sobre la estructura desconocida y reemplaza los valores señalados por `max`, `min`, `sum` y `n` con el valor máximo, mínimo, la suma y la cantidad de elementos. Escriba una rutina `minmaxprom` que itere sobre `listaDeDatos` y muestre por pantalla el máximo valor, mínimo valor y el promedio de todos los valores que se pueden encontrar en la lista y las estructuras desconocidas. Debe mostrar un único máximo, mínimo y promedio. Para realizar esta función, no se puede utilizar memoria dinámica ni declarar variables en la sección `.data`. Sólo puede utilizar la sección `.text`.
La aridad de la función debe ser `void minmaxprom(nodo * listaDeDatos)`