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

Organización del Computador 2
Primer parcial – 12 de mayo de 2009

Normas Generales

- Numere las hojas entregadas. Complete en la primera hoja la cantidad total de hojas entregadas.
- Está permitido tener los manuales y los apuntes con las listas de instrucciones durante el exámen. Está prohibido compartir manuales o apuntes entre alumnos en 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.
- Por favor entregar esta hoja junto al examen.

Ej. 1. (40 puntos)

Existen varios métodos para transformar una imagen color en formato RGB a escala de grises. La función más sencilla para hacer ésto consiste en sustituir el valor de cada componente RBG de cada pixel según la siguiente función:

$$f(x) = \sqrt[\epsilon]{\alpha R(x)^\epsilon + \beta G(x)^\epsilon + \gamma B(x)^\epsilon}, \quad (1)$$

donde α, β y γ son coeficientes entre 0 y 1 y ϵ es un exponente entre 1 e ∞.

Una imagen PNG consiste en una tira de pixels de 4 bytes cada uno (RGB + 1 byte de transparencia). Suponer una imagen provista con ancho mútliplo de 4 pixels, bien alineado.

Escriba una función en lenguaje ensamblador que dada la imagen en formato PNG, la monocratice siguiendo la ecuación descripta en **??**, sin importar el valor del byte de transparencia (puede valer cualquier cosa). Para una primera implemenatción se fija $\epsilon = 2$ y $\alpha = \beta = \gamma = 1/4$. Para mejorar la performance de la rutina se pide utilizar instrucciones SIMD que permitan procesar varios pixels en paralelo. El prototipo de la función es el siguiente:

```
void monocromatizar(char* imagen, int alto, int ancho);
```

Nota: se puede considerar que la imagen dada no tiene bytes de relleno al final de cada linea

Ej. 2. (40 puntos)

Dado un File System sencillo cuya definición viene dada por las siguientes estructuras:

```
enum entry_type { TYPE_FILE, TYPE_DIR };

typedef struct dir_entry {
  enum entry_type this_entry_type;
  char name[8+3];
  void *first_entry;
  void *next_entry;
} __attribute__((packed)) *ptr_dir_entry;

typedef struct file_entry {
  enum entry_type this_entry_type;
  char name[8+3];
  unsigned int attr;
  unsigned int size;
```

```
  void *data;
  void *next_entry;
} __attribute__((packed)) *ptr_file_entry;
```

donde el árbol de directorios se arma de la siguiente manera:

- Se tiene un `dir_entry` inicial.
- `first_entry` apunta al primer elemento (archivo o directorio) incluido en él. `NULL` si no corresponde.
- `next_entry` apunta al siguiente elemento dentro del directorio actual. `NULL` si no corresponde.
- `data` apunta a los datos del archivo, guardados en binario.
- `size` indica el tamaño del archivo en bytes

Se desea implementar una función que recorra un File System pasado como parámetro y calcule el tamaño promedio de los archivos. El prototipo de la función es el siguiente: `unsigned int calcularPromedioSize(ptr_dir_entry root_folder);`

Se pide:

1. Escribir el pseudo-código de la función.
2. Escribir la función en lenguaje ensamblador comentando el código, haciendo referencias claras al pseudo-código del ítem anterior.

Nota: tener en cuenta que la sumatoria de los tamaños de todos los archivos no necesariamente entra en 32 bits.

Ej. 3. (20 puntos)

Sea el siguiente código C,

```
int main(int argc, char* argv[]) {
	char nombre[10];
	copiar(nombre, argv[1]);
	return 0;
}

void copiar(char* buffer, char* data) {
	char* _data = data;
	char* _buffer = buffer;
	while( *_data != 0 ) {
		*_buffer = *_data;
		_data++;
		_buffer++;
	}
}
```

Buffer overflow: se produce cuando se copia una cantidad de datos sobre un área de memoria que no es lo suficientemente grande como para contenerlos, sobrescribiendo de esta manera otras zonas de memoria.

1. Siguiendo la convención C, ¿cómo queda el stack luego entrar en la función copiar? Indicar donde se almacenan las variables locales de la función `copiar`.
2. Proponga y explique un ejemplo de una posible entrada para el programa anterior, de manera que se produzca un buffer overflow.
3. *opcional* ¿Cuál sería la entrada para el programa si deseo que la dirección de retorno del main sea `0xBABA` (Hint: piense en el stack completo incluyendo la llamada al `main`)?