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

Organización del Computador 2
Primer parcial – 9/10/2012

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.
- Existen tres notas posibles para los parciales: 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. Para los recuperatorios existen sólo dos notas posibles: I: 0 a 64 pts y A: 65 a 100 pts.

Ej. 1. (40 puntos)

El fin de Buenos Aires se avecina; una plaga de mosquitos se acerca desde la selva húmeda del Congo con mucha sed de sangre latina. Para combatirla, el *Ministerio de control de plagas*, propuso esperar la llegada de la nube asesina de mosquitos con un gran conjunto de sapos. Por esta razón se le encargó a un grupo de estudiantes de biología la clonación a mansalva de los anfibios salvadores.

Los estudiantes de biología comenzaron a trabajar de inmediato en el proyecto pero confundieron el orden de la cadena de ADN del sapo y como resultado, obtuvieron ovejas verdes saltarinas.

Una cadena de ADN está formada por eslabones. Cada eslabón contiene una base nitrogenada, un puntero al eslabón pareja y otro puntero al eslabón inferior.

Ayudemos al grupo de biología a reordenar la cadena implementando una función que dada una cadena de ADN la reordene. El ordenamiento deberá respetar el resultado de la función `cmpBase()` (ya dada por los biólogos) de forma tal que **duplique** o **destruya** un par de eslabones, según corresponda.

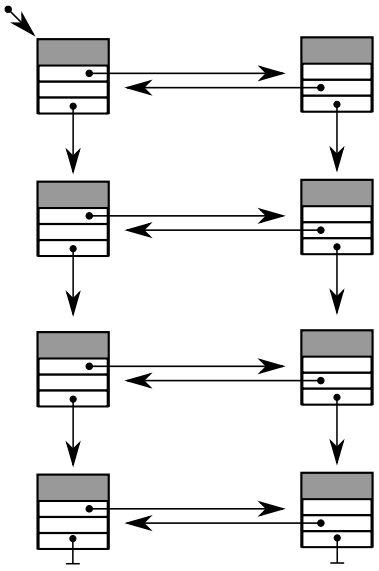

Figura 1: Ejemplo de cadena de ADN con 8 eslabones.

La estructura es:

```
typedef enum action_e { borrar=0, duplicar=1 } action;

typedef struct eslabon_t{
    char base;
    struct eslabon_t* eslabon_pareja;
    struct eslabon_t* eslabon_inferior;
} __attribute__((__packed__)) eslabon;
```

La función a implementar es:

```
eslabon* ordenarCadena(eslabon* primer_eslabon,enum action_e (*cmpBase)(char* base1,
char* base2));
```

La función pasada por parámetro tiene la aridad `enum action_e cmpBase(char* eslabon1, char* eslabon2)` y retorna un valor de tipo enumerado, representando la acción a seguir. La función `ordenarCadena` debe retornar el primer eslabon, ya que este puede cambiar.

(10p) 1. Escribir el pseudo-código de la función `ordenarCadena`.
(30p) 2. Implementar en ASM de 64 bits la función `ordenarCadena`.

Ej. 2. (40 puntos)

Se tiene una matriz de $n \times m$ números enteros de 16 bits con signo. Resultado de una simulación de un complejo modelo climático. Para utilizar estos resultados como entrada de otra de las etapas de procesamiento se deben aplicar determinadas transformaciones a los datos. Estas transformaciones están definidas por la función g. La transformación depende del valor original. Para el caso de `0xff` y `0x0`, no se debe realizar ninguna acción, ya que estos números corresponden a valores no válidos para el modelo. Para cualquier otro caso, se aplica una transformación de escala. El objetivo es aplicar a cada valor de la matriz la función g.

La función esta definida por:

$$g(x_{i,j}) = \begin{cases} x_{i,j} & \text{si } x_{i,j} = \texttt{0xff} \text{ o } x_{i,j} = \texttt{0x00} \\ \dfrac{x_{i,j} \cdot \pi}{35} & \text{en otro caso} \end{cases}$$

La aridad de la función a implementar será: `void aplicar_g(short* datos, int n, int m)`

(10p) 1. Explicar en palabras: ¿cuál es la operatoria para resolver la función pedida? ¿Qué propiedad deben tener los valores de m y n para que se pueda resolver sin repetir calculos?
(30p) 2. Implementar en ASM de 64 bits, usando SIMD, la función pedida; considerando las restricciones del item anterior. Se deberán procesar al menos 4 datos por ciclo.

Nota: Para cada operación que use registros `xmm` debe indicar el estado del registro destino mediante un dibujo de su contenido.

Ej. 3. (20 puntos)

Sea el siguiente código en C:

```
void rprint(int** d, char* t)
{
    int i = 0;
    char[100] text;
    while( t[i] != 0 )
    {
        text[98-i] = t[i];
        i = i + 1;
    }
    *(d[0]) = i;
    text[99] = 0;
    printf("%s",&(text[98-i+1]));
}
```

(12p) 1. Implementar en ASM de 64 bits la función mencionada.
(8p) 2. ¿Qué sucede si la string (`t`) pasada por parámetro supera los 100 caracteres? ¿Es posible modificar la dirección de retorno de `rprint` si se pasa un buffer lo suficientemente grande? ¿De qué tamaño sería este buffer? Justifique detalladamente cada respuesta.