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

Organización del Computador 2
Primer parcial – 17/05/2011

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 examen. 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.
- Los parciales tienen 3 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 2 notas: I: 0 a 64 pts y A: 65 a 100 pts.
- Entregue esta hoja junto al examen. Esta hoja se numera como hoja 0.

Ej. 1. (40 puntos)

En álgebra lineal, una matriz de rotación es una matriz que se utiliza para realizar una rotación en el espacio euclideo. Por ejemplo, la matriz

$$\begin{pmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{pmatrix}$$

rota los puntos (x, y) del espacio cartesiano en sentido antihorario con un ángulo θ. Para realizar la rotación, se debe multiplicar esta matriz por las coordenadas (x, y) obteniendo las nuevas coordenadas (x', y') del punto. De esta manera, quedan definidas las siguientes ecuaciones:

$$x' = x * cos(\theta) - y * sin(\theta)$$
$$y' = x * sin(\theta) + y * cos(\theta)$$

Modelaremos un punto en el espacio cartesiano con el siguiente el tipo de datos:

```
typedef struct _punto {
    short x;
    short y;
} __attribute__(( __packed__)) punto;
```

Escriba una función en lenguaje ensamblador que dado un vector de puntos de longitud n (n de 16 bits, múltiplo de 2)y un ángulo θ, modifique el vector rotando cada punto del mismo en sentido antihorario θ grados. El prototipo de la función es: `void rotar(punto* v, unsigned short n, float θ)`.

Notas: Se deben procesar al menos dos puntos por vez utilizando registros SSE. Las cuentas deben realizarse en punto flotante de precisión simple.

Ej. 2. 40 puntos

La computación gráfica es un campo de las Ciencias de la Computación que estudia métodos para sintetizar y manipular contenido visual digitalmente. Para algunas aplicaciones en este campo, se requiere guardar los puntos de un objeto en alguna estructura de datos y luego aplicarle diferentes transformaciones a todos los puntos o a algunos subcojuntos de ellos.

En este caso, para guardar los puntos de un objeto utilizamos un árbol binario que en cada nodo guarda un subconjunto de los puntos en un vector. Además, se cuenta con una lista de rotaciones que deberán aplicarse a algunos puntos del objeto. La primer rotación debe aplicarse a todos los puntos en el árbol, la segunda debe aplicarse luego de la primera y sólo a los puntos que están en ambos subárboles

(no a los puntos de la raíz), la tercera rotación debe aplicarse luego de las dos primeras y sólo a los puntos que están en los subárboles a partir del tercer nivel del árbol y así sucesivamente hasta que se agoten las rotaciones a aplicar o el árbol.

Para modelar este problema, se cuenta con las siguientes estructuras de datos:

```
typedef struct _punto {
    short x;
    short y;
} __attribute__(( __packed__)) punto;

typedef struct nodo_t {
    punto* v;            //puntero al vector de puntos
    unsigned short n;    //longitud del vector v
    struct nodo_t * izq;
    struct nodo_t * der;
} __attribute__(( __packed__)) nodo_arbol;
```

La lista de rotaciones se modela con una lista de los siguientes nodos:

```
typedef struct nodo_l {
    float theta;
    struct nodo_l * proximo;
} nodo_lista;
```

donde `theta` es el ángulo θ según en cuál deben rotarse los puntos seleccionados. Un árbol es un puntero a `nodo_arbol` y una lista es un puntero a `nodo_lista`.

1. **(10 ptos)** Escriba el pseudocódigo de la función que dado un árbol y una lista de rotaciones, aplique las rotaciones a los diferentes subconjuntos del árbol según lo explicado anteriormente. El prototipo de la función es: `void rotar_sucesivamente(nodo_arbol* arbol, nodo_lista* lista)`
2. **(30 ptos)** Escriba la función en lenguaje ensamblador.

Notas:

- Para aplicar una rotación con ángulo θ_1 de la lista de rotaciones a un vector de puntos v_1 de longitud n_1 de un nodo particular del árbol puede usarse la función que programaron en el Ejercicio 1 realizando la siguiente llamada: `rotar(`v_1`,` n_1`,` θ_1`)`
- Recuerden que existe la recursión.

Ej. 3. (20 puntos)

En C es posible pasar una función como párametro, la misma se pasa como un puntero a la memoria donde está definida la función. Se desea construir la función `f1_to_f2` que dada la función `f1` devuelve un puntero a una nueva función que llamaremos `f2`. Los prototipos de las funciones son:

- `void* f1_to_f2( void* f1)`
- `int f1( int i, int j, int k)`
- `int f2( int j, int i)`

La función `f1_to_f2` debe solicitar memoria donde construir una nueva función `f2`. La nueva función utilizará la función `f1` para calcular el resultado, utilizando para esto un llamado a la función `f1`. Se debe respetar el prototipo de la función `f1` y para el caso del parámetro `k` se fija en 10.

Suponer que los opcode de las siguientes instrucciones son:

`opcodepushimm`	= `PUSH <imm>`	(2bytes, 4bytes para el inmediato)
`opcodepushmem`	= `PUSH [ESP+<imm>]`	(4bytes, 2bytes para el inmediato)
`opcodecall`	= `CALL <addr>`	(3bytes, 4bytes para la dirección)
`opcodeaddesp12`	= `ADD ESP,12`	(4bytes)
`opcoderet`	= `RET`	(1byte)