EXAMEN PRÁCTICAS

ARQUITECTURA DE COMPUTADORES

3º ST, TT, T y Teleco-ADE - URJC

Fuenlabrada, 2 de Diciembre de 2016

Se pide implementar (respetando el convenido de llamada a subrutina) el programa **lista_ordenada.asm** que crea una lista simplemente enlazada en la que todas las inserciones se hacen de mayor a menor valor. Por ejemplo, las inserciones de los nodos con valores 40, 12 y 50 tiene como resultado una lista en la que el primer nodo de la lista es el 50, después le sigue el 40 y por último el 12 (que apunta a null).

En el `main` se crea el primer nodo fuera del bucle y después, dentro del bucle, se realiza **`insert_in_order`** N veces (hasta que se introduzca un 0). Después, se debe imprimir la lista de menor a mayor, es decir, en nuestro ejemplo, se imprimen los valores 12, 40 y 50. El registro **$s0** se utilizará en el `main` para apuntar al primer nodo de la lista. `main` debe mantener actualizado el valor de **$s0** con los valores que le devuelve **`insert_in_order`** . Así, si **`insert_in_order`** le devuelve un 0 (un puntero nulo), significa que el primero de la lista sigue siendo el mismo. Por el contrario, si le devuelve una dirección distinta de cero, entonces será la del nodo que ahora pasa a ser el primero de la lista.

En C, la estructura de datos para representar un **nodo** de la lista sería la siguiente:

```
typedef struct _node_t {
      int val;              /* valor del nodo; tamaño palabra */
      struct _node_t *next; /* puntero al nodo siguiente */
} node_t;
```

El nodo siguiente (“next”) al último nodo de la lista es NULL. Se pide implementar tres funciones:

(obligatoria)`node_t * create(int val, node_t *next)`: crea un nuevo nodo e inicializa los campos del nodo con los argumentos recibidos. Devuelve la dirección del nuevo nodo creado.

(obligatoria)`node_t * insert_in_order(node_t *first, int val)`: recibe como parámetros la dirección del primer nodo de la lista y el valor del nodo a añadir. Crea un nuevo nodo por medio de la subrtuina **`create`** con el valor indicado y lo inserta en orden, de tal forma que los elementos de la lista queden ordenados de mayor a menor valor. Devuelve un puntero nulo si el primer nodo de la lista sigue siendo el mismo. Por el contrario, devuelve la dirección del nuevo nodo insertado cuando éste pasa a ser el primero de la lista.

(opcional)`void print(node_t *first)`: recorre la lista de forma **recursiva** imprimiendo los elementos de menor a mayor valor. Algoritmo:

```
if (first->next =! null) {
      print(first->next);
}
printf(“%d\n”, first->val);
return;
```