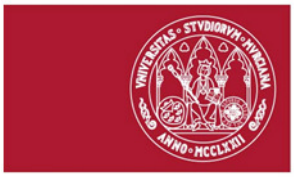

Estructura y Tecnología de Computadores

Examen final de prácticas (24 de mayo de 2019)

En el archivo `examen-2019-junio-prácticas.tar.gz` se incluyen dos versiones incompletas de un programa «progexamen»: una en C y otra en ensamblador MIPS. El programa, una vez completo, mostrará un menú donde se nos ofrecerán varias opciones para probar los ejercicios del examen que se describen a continuación.

Los códigos C y ensamblador que se proporcionan incluyen el menú y algún código auxiliar y de prueba, además de la cabecera de los procedimientos que se deberán implementar en ensamblador durante el examen. Sin embargo, no se incluye en la versión en C el código de algunos procedimientos que deberán ser diseñados e implementados durante el examen, por lo que no se puede compilar el programa dado. Por ello, también **se incluye una versión precompilada del programa completo para poder comprobar cuál debe ser su funcionamiento exacto una vez terminado**.

Teniendo en cuenta lo anterior, haga lo que piden los siguientes apartados, implementando los procedimientos que se piden en el fichero `progexamen.s`:

1. (1 punto) **Implemente** el procedimiento `preguntar_cadena`, que recibe tres parámetros: la dirección de comienzo de una cadena de caracteres que contiene el mensaje que se le debe mostrar al usuario para pedir la cadena, la dirección de comienzo de un buffer en el que se debe almacenar el resultado y un entero indicando el número máximo de caracteres que se deben leer. La función debe imprimir el mensaje y a continuación leer una cadena por teclado que debe de quedar almacenada en el buffer dado, respetando el máximo número de caracteres indicado por el tercer parámetro. Se aconseja utilizar la función `print_string` que se proporciona ya implementada.

2. (2,25 puntos) **Traduzca** el procedimiento `compara_cadena_prefijos`, que recibe tres argumentos (dos cadenas y un número `num_caracteres`) y compara los primeros `num_caracteres` de las dos cadenas, devolviendo `true` si y solo si sus primeros `num_caracteres` son iguales o si ambas cadenas son iguales y tienen menos de `num_caracteres`.

3. (2,25 puntos) **Traduzca** el procedimiento `contar_apariciones`, que recibe dos cadenas como argumentos (*texto* y *nombre*). El texto representa un fichero de agenda que contiene varias líneas, en cada una de las cuales aparece un nombre y su número de teléfono separados por '`:`'. Devuelve el número de líneas de *texto* que contienen *nombre* como subcadena.

4. (2,25 puntos) **Implemente** el procedimiento `contar_vocales` el cual recibe dos argumentos: la dirección de una cadena de texto y la dirección de una estructura de tipo `CuentaVocales` llamada `resultado`. El procedimiento debe contar cuántas veces aparece cada una de las cinco vocales minúsculas y sin tilde en el texto (caracteres «`aeiou`»). El resultado de la cuenta de cada vocal debe de quedar almacenado en el campo correspondiente de la estructura `resultado`.

5. (2,25 puntos) **Implemente** el procedimiento `obtener_saldo`, el cual recibe el nombre de un posible cliente de un banco y lo busca en el array `cuentas`, el cual contiene un elemento con la información de cada una de las cuentas del banco. El número de elementos del array `cuentas` está almacenado en la variable `num_cuentas`. El procedimiento debe devolver el saldo de la cuenta asociada al titular buscado, o 0 si no es cliente del banco.

   Pista: puede utilizar el procedimiento `compara_cadena_prefijos` para comprobar si el titular de una cuenta coincide con el buscado. Limite la comparación a un máximo de 100 caracteres.

Notas importantes a tener en cuenta para la realización de los ejercicios:

- Los procedimientos auxiliares y el menú están bien implementados y no es necesario ni recomendable gastar tiempo en entender cómo están implementados. Sólo hay que usarlos para comprobar el funcionamiento de los procedimientos implementados durante el examen.

- En el caso de los ejercicios en los que se pide que se traduzca una función, la traducción debe de ser lo más literal posible y se debe evitar cualquier tipo de optimización (en particular, debe respetar fielmente la estructura de los bucles originales).
- En el caso de los ejercicios en los que se pide que se implemente alguna función, se pueden utilizar funciones auxiliares siempre que se considere oportuno.
- A la hora de evaluar, sólo se tendrá en cuenta el código ensamblador (se ignorará cualquier modificación de la versión en C).
- Puede comprobar si su solución se comporta como la solución correcta comparando la salida de su programa con la versión precompilada, pero tenga en cuenta que el código de la solución puede ser incorrecto aún cuando se comporte correctamente.