PRINCIPIOS DE COMPUTADORES.
PRÁCTICA 5

Descripción.

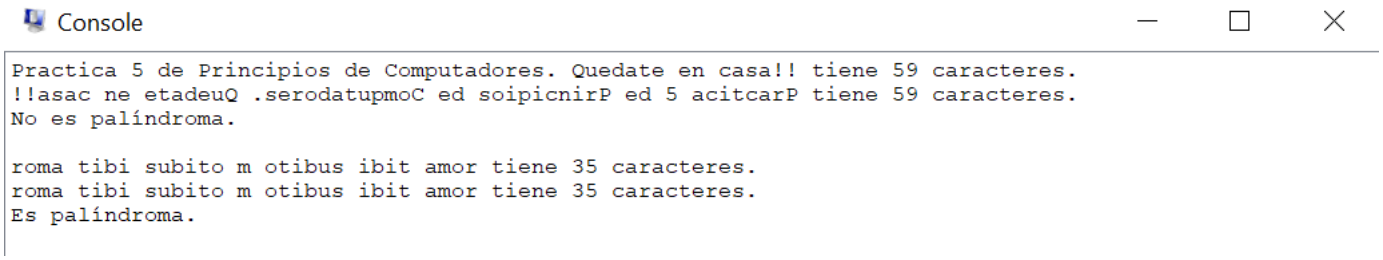
Te proponemos realizar en ensamblador un programa que contenga una función que invierta una cadena de caracteres y compruebe si es palíndroma. En concreto debe implementarse lo siguiente:

1) Una versión recursiva de la función con el nombre ***reverse_r*** que reciba dos parámetros. En el primer parámetro $a0 debe recibir la dirección de memoria donde se encuentra la cadena a invertir, y en el segundo parámetro $a1 el número de caracteres de la cadena. Debe devolver en $v0 un '1' si la cadena es palíndroma y un '0' si no lo es.
2) Una versión iterativa de la función con el nombre ***reverse_i*** que reciba dos parámetros. En el primer parámetro $a0 debe recibir la dirección de memoria donde se encuentra la cadena a invertir, y en el segundo parámetro $a1 el número de caracteres de la cadena. Debe devolver en $v0 un '1' si la cadena es palíndroma y un '0' si no lo es.
3) Una función ***strlen*** que calcule el número de caracteres que tiene una cadena de caracteres del tipo asciiz (es decir, que termine con la cadena con un byte 0). Esta función recibirá en $a0 la dirección de memoria donde se encuentra la cadena de la cual tiene que calcular la longitud, y devolverá en $v0 el número de caracteres de la misma.

Con este guion se te suministra un esqueleto del programa *practica5_esqueleto.s* que debes usarlo como base para realizar tu programa (ya que tiene las cadenas para probar la ejecución e imitar la salida propuesta).

```
    .data
cadena:     .asciiz "Practica 5 de Principios de Computadores. Quedate en casa!!"
cadena2:    .asciiz "roma tibi subito m otibus ibit amor"
cadtiene:   .asciiz " tiene "
cadcarac:   .asciiz " caracteres.\n"
cadespal:   .asciiz "Es palíndroma.\n\n"
cadnoespal: .asciiz "No es palíndroma.\n\n"
    .text

strlen:  # numero de caracteres que tiene una cadena sin considerar el '\0'
         # la cadena tiene que estar terminada en '\0'
         # $a0 tiene la direccion de la cadena
         # $v0 devuelve el numero de caracteres

         # INTRODUCE AQUÍ EL CÓDIGO DE LA FUNCIÓN strlen SIN CAMBIAR LOS ARGUMENTOS

reverse_i:  # funcion que da la vuelta a una cadena
            # $a0 cadena a la que hay que dar la vuelta
            # $a1 numero de caracternes que tiene la cadena
            # $v0 1 Si es palíndroma 0 si no lo es

            # INTRODUCE AQUÍ EL CÓDIGO DE LA FUNCIÓN reverse_i SIN CAMBIAR LOS ARGUMENTOS


reverse_r:  # funcion que da la vuelta a una cadena
            # $a0 cadena a la que hay que dar la vuelta
            # $a1 numero de caracternes que tiene la cadena
            # $v0 1 Si es palíndroma 0 si no lo es

            # INTRODUCE AQUÍ EL CÓDIGO DE LA FUNCIÓN reverse_r SIN CAMBIAR LOS ARGUMENTOS


main:
            # INTRODUCE AQUÍ EL CÓDIGO DE LA FUNCIÓN main QUE REPRODUZCA LA SALIDA COMO EL GUIÓN
            # INVOCANDO A LA FUNCIÓN strlen DESPUÉS DE CADA MODIFICACIÓN DE LAS CADENAS
```

Es importante que invoques a las funciones con los argumentos que hemos propuestos, y no otros. Además, antes de imprimir los mensajes por la pantalla debes calcular la longitud de la cadena con la función ***strlen*** para comprobar que no ha sido modificada. El programa debe producir una salida como la siguiente:



La primera línea impresa es la primera cadena original y su longitud. La segunda línea impresa corresponde con la cadena una vez se ha invocado a la función ***reverse_i***. La tercera línea impresa indica que la primera cadena no es palíndroma.
A continuación lo que se observa corresponde con la segunda cadena según se ha definido originalmente. Justo debajo aparece esta segunda cadena una vez se ha invocado a la función ***reverse_r*** (obviamente, si se invierte una cadena palíndroma debe quedar igual que la original). Finalmente se muestra que se ha detectado que la cadena es palíndroma.

Si tienes dificultad para diseñar el algoritmo (ya sea el recursivo o el iterativo) debes preguntar a los profesores de prácticas. Junto con este guion también te adjuntamos un programa en C++ que implementa la funcionalidad que te estamos pidiendo, aunque es posible que el uso de punteros para tratar las cadenas pueda confundirte (en esta ocasión, las funciones en ensamblador son incluso más claras).

Debes apoyarte en las trasparencias de la tutoría académica número 6.

La evaluación de este ejercicio será realizada por los profesores de prácticas, y se resolverá según los siguientes criterios:

1) Para aprobar la práctica con un 5, el código debe ejecutarse sin errores, y como mínimo deben implementarse las funciones ***strlen*** y ***reverse_r*** (versión recursiva). Se debe hacer un uso correcto de la pila y respetarse los argumentos propuestos.

2) Para sacar hasta un 8, el código debe ejecutarse sin errores, y como mínimo debe hacer lo anteriormente expuesto, pero además implementarse la función ***reverse_i*** (versión iterativa). Deberá además imitar la salida propuesta respetando todas las especificaciones.

3) Para sacar hasta un 10, el código debe incluir lo anteriormente expuesto y además implementar alguna función adicional (como por ejemplo que el usuario pueda introducir la cadena por el teclado u otras funcionalidades adicionales que tengan relación con el tratamiento de las cadenas).