

# ARQUITECTURA DE COMPUTADORES

## EXAMEN DE PRÁCTICAS

- **Grados:** IST, ITT, IT, IT-ADE, IT-AEROESPACIAL, URJC
- **Fecha:** 29-Junio-2023
- **Convocatoria Extraordinaria** (PRESENCIAL)

## AVISO

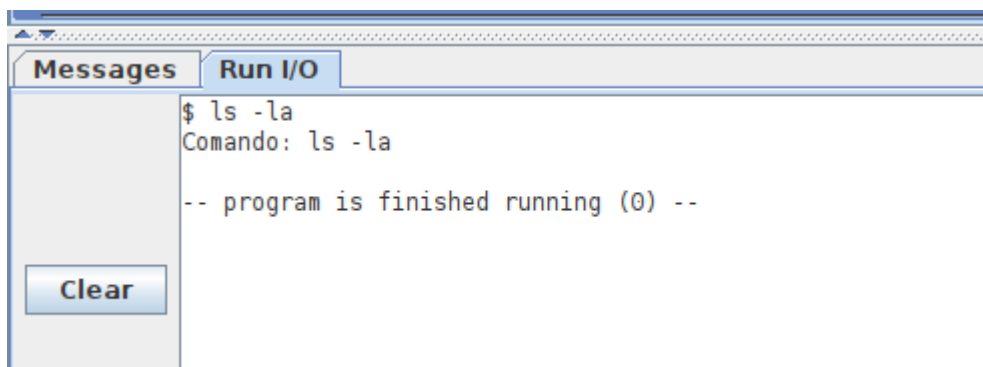
Asegúrate que tus programas cumplen con los siguientes criterios. Si no se cumple alguno de ellos **la nota será 0** en ese apartado

- **Cumplimiento de especificaciones.** Se deben cumplir las especificaciones indicadas en el enunciado: nombres de funciones, nombres de archivos, parámetros de las funciones, constantes, funcionalidad, etc. Compruébalo antes de entregar el examen
- **Respetar el convenio.** Resuelve las preguntas sin violar el convenio del uso de registros (ABI del RISC-V)
- **Sin errores en tiempo de ejecución (Runtime errors).** Tus programas no deben generar excepciones al ejecutarse
- **Sin errores al ensamblar.** Los ficheros entregados NO deben dar errores al ensamblarlos. Si una función la has dejado a medio hacer, asegúrate que al menos se ensambla sin errores

## Programa 1 (5 ptos)

Estamos desarrollando una aplicación para la línea de comandos, que se ejecutará en un RV32I en un sistema empotrado. La tenemos que desarrollar en lenguaje ensamblador para ahorrar memoria. Nuestro jefe de proyecto nos ha entregado este programa principal de pruebas (**prog1.s**), el cual pide un comando al usuario, lo copia a otra zona de memoria y luego lo imprime en la consola para comprobar que se ha copiado correctamente.

Tras su ejecución esta es la salida que obtenemos en la consola (el usuario ha introducido la cadena **ls -la** por teclado)



```
$ ls -la
Comando: ls -la

-- program is finished running (0) --
```

Nuestra misión es separar este programa en **dos partes**, manteniendo exactamente **la misma funcionalidad**. Por un lado hay que crear la función **strcpy(src, dest)** que tiene dos argumentos de entrada. El primero (src) es la **dirección de la cadena fuente** y el segundo (dest) **la dirección de la cadena**

**destino.** La función copia la cadena fuente en la destino y retorna. Esta función sólo realiza eso. No imprime nada en la consola, ni interactúa con el usuario. Sólo copia la cadena

Por otro lado hay que rehacer el programa principal para que use esta nueva función. Este programa principal interactúa con el usuario e imprime la misma salida en la consola, pero debe llamar a la función `strcpy` para realizar la copia de la cadena

Se pide:

**a)** (2.5) Implementa la función `strcpy` en el fichero **01\_strcpy.s**

**b)** (2.5) Implementa el nuevo programa principal en el fichero **01\_main.s**.

NOTA: No se te pide que inventes nada. Deberás modificar el código original para adaptarlo a lo pedido. Si el programa no es una adaptación del original, se considerará como violación de especificaciones

FICHEROS A ENTREGAR: `01_strcpy.s` y `01_main.s` (Tu programa debe funcionar sólo con esos dos ficheros, sin depender de ninguno adicional)

## Programa 2 (5 ptos)

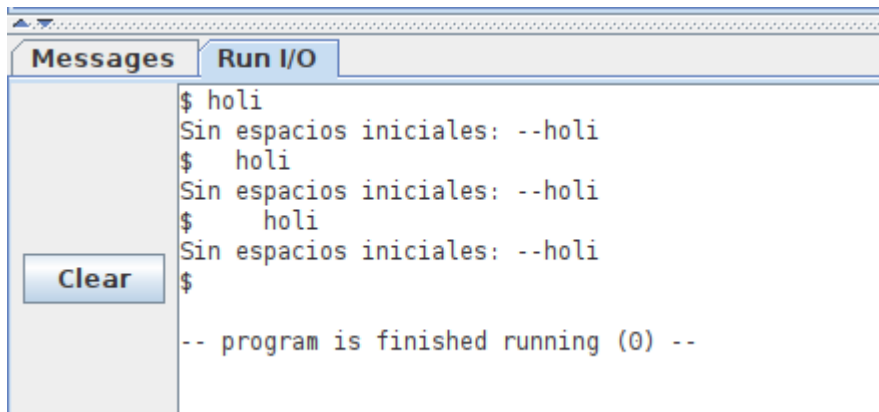
Seguimos con el desarrollo de una aplicación en la línea de comandos, escrita en lenguaje ensamblador para un RV32I. Nuestro jefe de proyecto nos ha proporcionado la función `skip(cad, car)`, en el fichero `02_skip.s`

Esta función se usa para ignorar los caracteres iniciales de una cadena. El primer argumento (`cad`) es un puntero a la cadena, y el segundo (`car`) es el caracter a ignorar. Así, por ejemplo, si le pasamos el puntero a la cadena `***holi` y llamamos a la función `skip` indicando `*` como segundo argumento, la función devolverá el puntero a la parte de la cadena donde NO están los asteriscos iniciales: `holi`

Nuestro jefe de proyecto nos pide crear **la nueva función** `skip_spaces(cad)` que simplemente llama a la función `skip()` pasando como argumento la misma cadena (`cad`) pero usando un espacio para el argumento `car`. Es decir, su única misión es llamar, a su vez, a la función `skip(cad, ' ')`. Otro ingeniero del equipo añadirá en el futuro más funcionalidad. Nuestra misión es simplemente crear el esqueleto.

Además, tenemos que crear un programa principal para comprobar que la función `skip_spaces()` funciona correctamente. Este programa debe imprimir el prompt (`$`), esperar a que el usuario introduzca una cadena, llamar a la función `skip_spaces()`, imprimir el mensaje "Sin espacios iniciales: --" y luego la cadena sin los espacios iniciales. Esto se repite en un bucle hasta que el usuario apriete enter sin introducir nada más. En ese momento se termina el programa

En este pantallazo se muestra un ejemplo de ejecución del programa principal en la que el usuario ha introducido tres veces la cadena `ho li`, con 0, 2 y 4 espacios iniciales, y finalmente aprieta enter sin introducir nada



```
$ holi
Sin espacios iniciales: --holi
$ holi
Sin espacios iniciales: --holi
$ holi
Sin espacios iniciales: --holi
$
-- program is finished running (0) --
```

Comprobamos que en todos los casos, efectivamente, se imprime la cadena "holi" sin los espacios iniciales. Es la prueba de que nuestra función `skip_spaces()` funciona correctamente

Se pide:

- a) (2.5 ptos) Implementar la función `skip_spaces()` en el fichero `02_skip_spaces.s`
- b) (2.5 ptos) Implementar el **programa principal** en el fichero `02_main.s`

**NOTA:** Este programa está formado únicamente por tres ficheros: `02_skip.s`, `02_skip_spaces.s` y `02_main.s`. Debe funcionar sin añadir ninguno más

FICHEROS A ENTREGAR: `02_skip_spaces.s` y `02_main.s`