

# Organización y Arquitectura de Computadoras

## 2020-1

### Práctica 7: Llamadas al sistema

Profesor: José de Jesús Galaviz Casas

17 de octubre de 2019

## 1. Objetivos

### Generales:

- Introducir al alumno a los conceptos básicos de operaciones de entrada/salida, cubriendo la interacción entre el usuario, el sistema operativo y el hardware.

### Particulares:

El alumno continuará practicando la programación en el lenguaje ensamblador de MIPS y al finalizar la práctica:

- Estará familiarizado con las llamadas al sistema.
- Será capaz de manipular cadenas de caracteres con lenguaje ensamblador MIPS-32.

## 2. Requisitos

- **Conocimientos previos:**
  - Instrucciones aritméticas, lógicas y de manejo de flujo de programas del lenguaje ensamblador MIPS.
  - Convención de llamadas a subrutinas.
  - Manejo de archivos en sistemas POSIX.
  - Codificación ASCII.
- **Tiempo de realización sugerido:**  
5 horas.

```

        .data
cad:    .asciiz "Hola mundo"
        .text
        la      $a0, cad          # Cargar argumento
        li      $v0, 4            # Cargar código imprimir cadena
        syscall
        li      $v0, 10           # Cargar código terminar prog
        syscall

```

Figura 1: Ejemplo de llamada al sistema en MARS.

- **Número de colaboradores:**  
Individual.
- **Software a utilizar:**
  - *Java Runtime Environment* versión 5 o superior.
  - El paquete MARS [Mars].

### 3. Planteamiento

En una computadora moderna, el sistema operativo es el encargado de gestionar la comunicación con los dispositivos de entrada y salida; administrar la memoria, otorgando nuevos segmentos a los programas que lo requieran; y dirigir la ejecución de los programas, entre otras muchas cosas. Los programas solicitan estos servicios al sistema operativo a través de un mecanismo denominado **llamada al sistema**, una instrucción especial que transfiere el control del procesador al sistema operativo, éste realiza la tarea que le fue solicitada y al finalizar, regresa los resultados y el control al programa que lo solicitó. En nuestro caso, al no contar con un sistema operativo en MARS, éste simula un pequeño subconjunto de llamadas.

## 4. Desarrollo

### 4.1. Llamadas al sistema en MIPS

Para solicitar un servicio en el simulador MARS, un programa debe cargar el código de la llamada en el registro `$v0`, los argumentos en los registros `$a0-$a2` ó `$f12` en el caso de punto flotante y ejecutar la instrucción `syscall`. Si la llamada regresa algún valor, éste se encontrará en el registro `$v0` ó `$f0` para datos de punto flotante y dobles. En la tabla 1 se encuentran las llamadas necesarias para completar la práctica, el resto las puedes encontrar en el menú ayuda del simulador. En la figura 1 se muestra un fragmento de código como ejemplo, el programa imprime un mensaje en la terminal de MARS.

| <b>Código</b> | <b>Servicio</b>     | <b>Argumento</b>                                     | <b>Retorno</b>                          |
|---------------|---------------------|--|---|
| 1             | Imprimir entero     | \$a0 = Entero  |   |
| 2             | Imprimir flotante   | \$f12 = Flotante                                     |   |
| 3             | Imprimir doble      | \$f12 = Doble  |   |
| 4             | Imprimir cadena     | \$a0 = Cadena* **                                    |   |
| 5             | Leer entero         |  | \$v0 = Entero                           |
| 6             | Leer flotante       |  | \$f0 = Flotante                         |
| 7             | Leer doble          |  | \$f0 = Doble                            |
| 8             | Leer cadena         | \$a0 = Buffer*<br>\$a1 = Tamaño                      |   |
| 9             | Asignar memoria     | \$a0 = Cantidad                                      | \$v0 = Dirección                        |
| 10            | Terminar ejecución  |  |   |
| 11            | Imprimir carácter   | \$a0 = Carácter                                      |   |
| 12            | Leer carácter       |  | \$v0 = Carácter                         |
| 13            | Abrir archivo       | \$a0 = Nombre* **<br>\$a1 = Banderas<br>\$a2 = Modo  | \$v0 = Descriptor                       |
| 14            | Leer del archivo    | \$a0 = Descriptor<br>\$a1 = Buffer*<br>\$a2 = Tamaño | \$v0 = Número de<br>caracteres leídos   |
| 15            | Escribir al archivo | \$a0 = Descriptor<br>\$a1 = Buffer*<br>\$a2 = Tamaño | \$v0 = Número de<br>caracteres escritos |
| 16            | Cerrar archivo      | \$a0 = Descriptor                                    |   |

Tabla 1: Códigos de llamadas al sistema en MARS.

\* El argumento es un apuntador, es decir la dirección de memoria en donde comienza el dato.

\*\* Cadena con carácter nulo al final.

## 5. Procedimiento

Escribe los programas lenguaje ensamblador de MIPS respetando los siguientes lineamientos:

- Puedes hacer uso de macros
- Debe ejecutarse sin errores para ser evaluado
- Debe terminar con la llamada al sistema *exit*.
- Debe desplegar un mensaje al inicio de su ejecución que indique al usuario como éste se usa
- Debe desplegar un mensaje que indique cual es el resultado que entrega el programa y que terminó la ejecución
- Deberás entregar un archivo por cada ejercicio
- No olvides documentar el código.

## 6. Ejercicios

1. Escribe un programa que reciba una cadena y escriba la reversa de dicha cadena en la terminal
2. Escribe un programa que cree copie el contenido de un archivo remplazado cada aparición de un carácter por otro determinado El programa recibirá por la terminal de MARS el nombre del archivo seguido de los dos caracteres y crear un archivo de nombre *salida.txt* que contendrá la copia con remplazo del archivo de entrada.

## 7. Preguntas

1. Investiga y describe con tus propias palabras ¿cómo resuelve una llamada al sistema el sistema operativo?