Organización y Arquitectura de Computadoras
2018-1
Práctica 8: Llamadas al sistema

Profesor: José Galaviz Casas
Ayudante de laboratorio: Luis Soto Martínez

1. Objetivos

Generales:

- Introducir al alumno a los conceptos básicos de operaciones de entrada/salida, cubriendo la interacción entre el usuario, las rutinas, 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.
- **Número de colaboradores:**
  Individual.

```
        .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.

- **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 dispostivos 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. En esta práctica se desarrollará un intérprete de comandos en donde se aprovecharán algunas de las llamadas al sistema simuladas por MARS.

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.

Código	Servicio	Argumeto	Retorno
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* `$a1` = Tamaño	
9	Asignar memoria	`$a0` = Cantidad	`$v0` = Dirección
10	Terminar ejecución		
11	Imprimir caracter	`$a0` = Caracter	
12	Leer caracter		`$v0` = Caracter
13	Abrir archivo	`$a0` = Nombre* ** `$a1` = Banderas `$a2` = Modo	`$v0` = Descriptor
14	Leer del archivo	`$a0` = Descriptor `$a1` = Buffer* `$a2` = Tamaño	`$v0` = Número de caracteres leidos
15	Escribir al archivo	`$a0` = Descriptor `$a1` = Buffer* `$a2` = Tamaño	`$v0` = Número de caracteres escritos
16	Cerrar archivo	`$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 caracter nulo al final.

4.2. Intérprete de comandos

Un intérprete de comandos es un progama que ejecuta interactivamente órdenes introducidas por un usuario. El intérprete recibe una cadena de texto con la orden del usuario, la analiza, determina la tarea a realizar y la ejecuta. El intérprete que simularemos, cumplirá con las siguientes características:

1. La entrada y la salida será por medio del simulador de terminal de MARS.
2. El intérprete imprimirá en la terminal un *prompt* (un caracter o secuencia de caracteres) para indicar al usuario que espera un comando, el usuario podrá escribir un comando en la terminal, el intérprete ejecutará dicho comando, mostrará el resultado en la terminal y repetirá el proceso.
3. Si el usuario solicita un comando no implementado, el intérprete debe mostrar un mensaje indicando el error.
4. Los comandos que se deben implementar:

   a) **help**
   Descripción.
   Imprime información de los comandos disponibles y sus opciones. Si se llama sin argumentos, imprime una lista de los comandos disponibles.
   Sinópsis:
   `help [arg]`
   Argumentos:
   `arg` - Imprime la descripción y opciones del comando `arg`.

   b) **rev**
   Descripción.
   Imprime la reversa de una cadena. Si no se especifica un archivo, se utiliza la entrada estándar (sólo una línea).
   Sinópsis:
   `rev [file]`
   Argumentos:
   `file` - Archivo de texto a revertir.

   c) **cat**
   Descripción.
   Concatena los archivos y los muestra en pantalla.
   Sinópsis:
   `cat file [file]`
   Argumentos:
   `file` - Archivo a concatenar.

   d) **exit**
   Descripción.
   Termina la ejecución del interprete de comandos, terminando así la simulacion de MARS.

5. Los comandos se implementarán como subrutinas, el valor de retorno de cada una será un apuntador a una cadena, la cual contiene el resultado de la ejecución.

6. Si el usuario comete un error al dar los argumentos de un comando, la subrutina del comando será la encargada de informar al usuario el error.

5. Entrada

El intérprete será interactivo, el usuario introducirá la entrada o comando en el simulador de terminal de MARS. El tamaño máximo de los archivos de enterada será de 1 KB.

6. Salida

El intérprete mostrará el resultado en el simulador de terminal de MARS.

7. Variables libres

El alumno deberá definir un comando para el intérprete.

8. Procedimento

Escribe las rutinas necesarias en lenguaje ensamblador de MIPS respetando los siguientes lineamientos:

- Usa la convención de llamadas a subrutinas.
- Agrega una rutina *main* en donde se carguen de la memoria los argumentos en los registros adecuados y se llame a la subrutina que resuelva el ejercicio.
- El programa debe terminar con la llamada al sistema *exit*.
- Deberás entregar un único archivo de código fuente.
- No olvides documentar el código.

9. Ejercicios

1. Desarrolla el intérprete de comandos descrito en la sección 4.2.

2. Agrega un comando extra, debe ser de una complejidad similar a los otros. No olvides documentarlo para el comando `help`.

10. Preguntas

1. Considerando los componentes básicos de una computadora moderna, además de las llamadas al sistema descritas en la práctica, ¿qué otras resultarían necesarias?

2. ¿Qué comandos son básicos para un intérprete de comandos de un sistema operativo?

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

11. Bibliografía