Práctica. Paso de parámetros entre subrutinas

1. Objetivo de la práctica

El objetivo de esta práctica es que el estudiante se familiarice con la programación en ensamblador y el convenio de paso de parámetros a subrutinas en el ensamblador del MIPS32.

2. Descripción

El alumno deberá realizar la implementación en ensamblador de un conjunto de funciones:

1. Funciones para imprimir datos
2. Imprimir el valor ASCII de los caracteres de una cadena
3. Funciones que operan con números reales y enteros
4. Impresión de valores en big-endian y little-endian
5. Funciones de manipulación de arrays

NOTA: Las funciones implementadas deben tener el nombre indicado en las siguientes secciones, se ha de tener en cuenta que un nombre con mayúsculas es diferente de otro con minúsculas. Por ejemplo, los siguientes nombres de funciones corresponden a subprogramas diferentes:

imprimir_entero, imprimir_entero, imprimir_ENTERO, etc.

Observe que los nombres de función que se piden en este trabajo tienen todas sus letras en **minúsculas**.

3. Consideraciones sobre el paso de parámetros

El paso de parámetros a subrutina se hace mediante registros. La pila se usa solo cuando es necesario. El paso de parámetros se realiza de la siguiente forma:

- Los parámetros que no sean números en coma flotante se pasan usando los registros $a0, $a1, $a2 y $a3 del banco general de registros. Si hay más de cuatro parámetros de este tipo, los parámetros adicionales se pasan en la pila.
- Los parámetros de valores en coma flotante en simple precisión se pasan usando los registros $f12, $f13, $f14 y $f15 del coprocesador matemático. Si hay más de cuatro parámetros de este tipo, los parámetros adicionales se pasan en la pila.
- Los parámetros de valores en coma flotante de doble precisión se pasan usando los registros $f12 y $f14. Si hay más de dos parámetros de este tipo, los parámetros adicionales se pasan en la pila.
- Los resultados que no sean valores de coma flotante se devuelven en los registros $v0 y $v1. Si se devuelven más valores, éstos se han de devolver en la pila.
- Los resultados de valores en coma flotante se devuelven en el registro $f0.

Así, para una función similar a la siguiente:

```
int funcion (int n0, int n1, int n2, int n3, int n4, int n5)
```

- El parámetro n0 se pasará en el registro $a0
- El parámetro n1 se pasará en el registro $a1

- El parámetro n2 se pasará en el registro $a2
- El parámetro n3 se pasará en el registro $a3
- Los parámetros n4 y n5 se colocan en la pila, primero n5 y luego n4, quedando en la cima de la pila n4.

Para una función como la siguiente:

```
float funcion (float n0, float n1)
```

- El parámetro n0 se pasará en el registro $f12 del coprocesador matemático
- El parámetro n1 se pasará en el registro $f13 del coprocesador matemático
- El resultado se devolverá en el registro $f0

3.1. Paso de un array como parámetro

Para pasar como parámetro un array de valores (independientemente del tipo de los valores que haya en el array), se han de pasar dos parámetros:

- En el primero se pasa la dirección de memoria de comienzo del array.
- En el segundo se pasa el número de elementos del array.

Así, una función que procese un array podrá ser similar a la siguiente:

```
int funcion (int vector[], int n)
```

Esta función procesa un array, denominado vector, de números enteros. El número de componentes del array se pasa en el segundo parámetro (n). Para esta función, vector, representa la dirección de inicio y n el número de elementos. Al utilizar ensamblador la dirección se ha de pasara en el registro $a0 y el tamaño (el segundo parámetro) en el registro
$a1, tal y como se ha indicado anteriormente.

En el caso de cadenas de caracteres, no es necesario pasar el segundo argumento que indica la longitud, ya que se puede conocer el fin de la cadena, puesto que todas las cadenas finalizan con el código 0. En este caso el prototipo de la función es como el siguiente:

```
void función (char cadena[]);
```

Para información más detallada sobre el desarrollo de esta práctica consulte la bibliografía propuesta para la asignatura. Recuerde que uno de los objetivos de la práctica es fomentar la búsqueda de información por parte de los alumnos.

3.2. Variables locales

Las variables locales de un programa, deben mantener su valor durante todo el tiempo de vida de la función. Tenga en cuenta las siguientes recomendaciones:

- Las variables locales de una función se asignarán a registros siempre que sea posible. Para las variables que no se puedan almacenar en un registro, ya sea por su tamaño (por ejemplo, un array) o porque no queden registros disponibles, se asignará memoria en el

marco de pila de la función. Para estas variables se pueden utilizar los registros $s o $t y para el coprocesador en coma flotante algunos $f.

En el MIPS existe un convenio respecto al uso de los registros. Se pueden considerar dos tipos de registros:

- Registros que deben preservar (registros preservados) su valor entre llamadas a funciones (registros $s, $sp, $fp y $ra).
- Registros para los que no se garantiza el mantenimiento de su valor entre llamadas a funciones (todos los registros excepto los anteriores).

Para las variables que no se puedan almacenar en un registro, ya sea por su tamaño o porque no queden registros disponibles, se asignará memoria en el marco de pila de la función.

4. Funciones a desarrollar

A continuación se detallan las funciones que hay que desarrollar.

Recuerde que para todos los ejercicios que las rutinas implementadas tienen que seguir el convenio de paso de parámetros que se ha indicado anteriormente. En caso contrario, la práctica tendrá la calificación de 0.

4.1. Funciones para imprimir datos

Se desea realizar un programa denominado **ejercicio1.s** que implemente las siguientes funciones.

- `imprimir_cadena`. Esta función recibe como parámetro la dirección de inicio de una cadena y la imprime por pantalla. La función no devuelve ningún valor.
- `imprimir_entero.` Esta función recibe como parámetro un entero y lo imprime. No devuelve ningún resultado.
- `imprimir_doble`. Esta función recibe como parámetro un número en coma flotante de doble precisión y lo imprime por pantalla. La función no devuelve ningún resultado.

Para cada una de estas funciones se deberá indicar los registros que se han usado para el paso de parámetros. Además, el fichero **ejercicio1.s,** deberá incluir también el código main (principal) que incluirá ejemplos de llamadas a cada una de las funciones anteriores.

4.2. Imprimir el valor ASCII de una cadena

Se desea realizar un programa denominado **ejercicio2.s** que lea por teclado una cadena de caracteres e imprima el carácter leído y su código ASCII correspondiente seguidamente. Para ello se deberán implementar en dicho programa las tres siguientes funciones:

- `imprimir_letra`. Esta función recibe como parámetro un carácter. La función debe imprimir la letra, un espacio y a continuación el código ASCII de la letra pasada. La función no devuelve nada.
- `longitud_cadena`. Esta función recibe como parámetro una cadena de caracteres y la función devuelve la longitud de la cadena.

El alumno deberá también realizar un código principal (main) que:

- Lea por teclado una cadena de caracteres.
- Llame a continuación a la función `longitud_cadena` para conocer su longitud. El programa deberá imprimir la longitud por pantalla.
- Deberá recorrer la cadena y usar para cada carácter de la cadena la función `imprimir_letra`. Cada letra junto con su código ASCII aparecerá en una línea distinta.

Así, por ejemplo, si se lee la cadena "hola", el programa debe mostrar la siguiente salida:

Longitud de Hola: 4
Caracteres de Hola:
H 100
O 90
L 78
A 97

4.3. Funciones que operan con números reales y enteros

Se desea realizar un programa denominado **ejercicio3.s** que lea por teclado dos valores: un entero y un número real, y posteriormente imprima el resultado de la suma, resta, multiplicación y división (siempre que sea posible) de ambos valores.
Para ello deberá implementar las siguientes funciones:

- suma. Esta función recibe dos parámetros. El primero es un entero y el segundo un float (número en coma flotante de simple precisión). La función devolverá el valor de la suma de los números. El resultado será un float.

- resta. Esta función recibe dos parámetros. El primero es un entero y el segundo un float (número en coma flotante de simple precisión). La función devolverá el valor de la resta de los números. El resultado será un float.

- producto. Esta función recibe dos parámetros. El primero es un entero y el segundo un float (número en coma flotante de simple precisión). La función devolverá el valor del producto de los números. El resultado será un float.

- division. Esta función recibe dos parámetros. El primero es un entero y el segundo un float (número en coma flotante de simple precisión). La función devolverá el valor de la división de los números. El resultado será un float.

Observe que en este caso a las funciones se pasan valores de tipo diferentes. Los enteros se pasan en los registros $a0-$a3 y los valores de tipo float en $f12-$f15.

El programa main deberá leer por teclado un valor entero y un valor de tipo float. A continuación, llamará a cada una de las funciones anteriores con los parámetros adecuados, imprimiendo por pantalla cada uno de los resultados.

4.4. Impresión de valores en big-endian y little-endian

Se debe implementar un programa llamado **ejercicio4.s** que lee por teclado un valor entero (4 bytes). El objetivo de la función es implementar una función que invierta el orden de los bytes, de forma que se pueda pasar de little endian a big endian y viceversa. Para ello, se deberán

implementar las siguientes funciones:

- invertir_bytes. Esta función recibe un valor entero como parámetro. La función devuelve un resultado entero. El resultado se obtiene invirtiendo el orden de los bytes del número pasado como parámetro, de forma que sea posible convertir de little endian a big endian y viceversa utilizando esta función.

- imprimir_binario. Esta función recibe un valor entero como parámetro (recuerde que un entero ocupa 32 bits). La función deberá imprimir su representación binaria. La función deberá imprimir los 32 dígitos binarios. Esta función no devuelve ningún resultado

El programa deberá incluir una función main que haga lo siguiente:

- Lea por teclado un número entero.

- Invierta el orden de los bytes del número leído e imprima su valor en binario.

- Vuelva a invertir el número obtenido en la inversión anterior. Para este número vuelva a imprimir su valor en binario.

4.5. Funciones de manipulación de arrays

Se debe implementar un programa llamado **ejercicio5.s** que implemente las siguientes funciones:

```
void sumar_int (int a[], int b[], int c[], int n)
void sumar_float (float a[], float b[], float c[], int n)
```

Para ello deberá implementar las siguientes funciones:

- La función `sumar_array_int` suma dos vectores de números enteros (el a y el b) y almacena el vector suma en c.

- La función `sumar_array_float` suma dos vectores de números de tipo float (el a y el b) y almacena el vector suma en c.

- La función `imprimir_array_int`, que imprime todas las componentes de un array pasado por parámetro.

- La función `imprimir_array_float`, que imprime todas las componentes de un array pasado por parámetro.

Las funciones no devuelven ningún resultado y asumen que todos los vectores tienen dimensión n.

Para estas funciones, se hará a su vez uso de estas 2 funciones, que también han de implementarse.

- La función `sumar_int`, suma dos números enteros pasados por parámetro y devuelve como resultado su suma.

- La función `sumar_float`, suma dos números de tipo float pasados por parámetro y devuelve como resultado su suma.

El programa debe incluir el programa principal (main). En el segmento de datos se declararán:

- 3 arrays de 5 elementos de tipo entero
- 3 arrays de 10 elementos de tipo entero
- 3 arrays de 5 elementos de tipo float.
- 3 arrays de 10 elementos de tipo float

El programa deberá probar las funciones anteriores con los arrays declarados. Es decir, para cada uno de los puntos anteriores, realizará la suma y dejará el resultado en el otro array. Además se deberá imprimir los vectores a sumar y el resultado.

5. Realización de la práctica

Para cada uno de las funciones solicitadas se deben realizar los siguientes pasos:

- Asignación de parámetros de entrada y parámetros de salida.
- Asignación de variables locales si las hubiera.
- Especificación del marco de pila, es decir, qué elementos se almacenan en el marco de pila de las funciones.
- Descripción de cómo se realiza el paso de parámetros para cada función y cómo se obtiene el resultado de cada función.
- Escritura del cuerpo de la función.
- Escritura de un programa de prueba, con casos de prueba.

Estos pasos han de quedar definidos en la documentación a entregar.