PRÁCTICA 4. Subrutinas.

Realizar un programa en ensamblador MIPS para operaciones con vectores de números flotantes en simple precisión. En el segmento de datos tendremos almacenados dos vectores `v1` y `v2` que podrán ocupar 160 bytes (máximo 40 elementos flotantes simple precisión). El número actual de elementos de cada uno de los vectores estará igualmente almacenado en memoria bajo las etiquetas `n1` y `n2`.

El programa deberá tener implementada las siguientes subrutinas:

1. Realizar una subrutina que se llame **print_vec** para imprimir un vector de flotantes simple precisión por la consola. Esta subrutina recibirá tres parámetros enteros y no devolverá ningún valor. El primer parámetro será la dirección base del vector a imprimir, y el segundo parámetro será el número de elementos que tiene el vector. El tercer argumento será la dirección de memoria de una cadena que sirva como separador al imprimir los elementos del vector. Esta subrutina deberá ser invocada cada vez que se desee imprimir un vector por la consola. **NOTA**: La función únicamente imprime el vector como una lista de números. El encabezado previo con el número de elementos del vector y el salto de línea final deberá realizarse desde la función que llama a ésta.
2. Realiza una subrutina que llame **change_elto** para modificar un elemento concreto de un vector de flotantes de simple precisión. Esta subrutina recibe tres parámetros. El primer parámetro es la dirección base del vector. El segundo parámetro es un entero que corresponde con el índice del elemento que queremos modificar **(este entero debe ser un número mayor o igual que cero y menor que el número de elementos del vector, pero la subrutina no debe comprobar este hecho, sino que debe hacerse desde el programa principal).** El tercer parámetro es un flotante de simple precisión que corresponde con el dato que queremos almacenar en la posición indicada.
3. Realiza una subrutina que llame **swap** para intercambiar dos elementos de un vector de flotantes de simple precisión. Esta subrutina recibe tres parámetros. El primer parámetro es la dirección base del vector. El segundo parámetro es un entero que corresponde con el índice del primer elemento que queremos intercambiar, y el tercer parámetro es el índice del segundo elemento que queremos intercambiar **(los índices deben ser números mayor o igual que cero y menor que el número de elementos del vector, pero la subrutina no debe comprobar este hecho, sino que debe hacerse desde el programa principal).**
4. Realizar una **subrutina RECURSIVA** llamadas **mirror** y que servirá para invertir los elementos de un vector. La función recibirá dos argumentos enteros. El primer argumento es la dirección de memoria del vector. El segundo argumento es el número de elementos del vector. ***Esta función debe hacer uso de la función swap para realizar el intercambio de elementos.*** La rutina intercambiará el primer elemento con el último elemento del vector y se llamará a sí misma pasando como dirección base del vector la del segundo elemento y como tamaño 2 menos que el tamaño recibido. El caso base (no habrá que hacer más llamadas) cuando el vector tenga un tamaño menor o igual a 1.

5. Realiza una subrutina que se llame **mult_add** que recibe como argumentos tres flotantes en simple precisión, y devuelve un flotante de simple precisión. Debe realizar la operación de multiplicar los dos primeros argumentos y sumarle el tercer argumento, devolviendo este valor como resultado de la subrutina.
6. Realiza una subrutina que se llame **prod_esc** que recibe como argumentos tres enteros: el primer argumento es la dirección de memoria del primer vector, el segundo argumento la dirección de memoria del segundo vector, el tercer argumento es el número de elementos de cada vector. Esta subrutina devolverá un flotante en simple precisión que corresponde con el producto escalar de los dos vectores pasados como argumentos. Esta subrutina debe usar **mult_add** en su operación. **Solo se puede hacer el producto escalar de dos vectores con la misma dimensión. Esta subrutina supone que los dos vectores tienen la misma dimensión, y la comprobación de que la tienen deberá hacerse desde el programa principal.**

Inicialmente, el programa principal debe cargar los vectores `v1` y `v2` con 40 valores cada uno. El vector `v1` empezará con el valor 10, y el siguiente elemento será el anterior más 1. El vector `v2` empezará con el valor 40, y el siguiente elemento será el anterior menos 1. Esta carga inicial se hace únicamente una vez (como es lógico debes hacerlo en un bucle … no elemento a elemento).

Después trás la impresión de un título, deberá repetirse un bucle que tras imprimir los dos vectores (con una cabecera que indica su tamaño) ofrece un menú de opciones como el siguiente:

(1) Cambiar dimensión de un vector
(2) Cambiar un elemento de un vector
(3) Invertir un vector
(4) Calcular el producto escalar de dos vectores
(0) Salir

En las opciones 1,2 y 3 el usuario podrá elegir el vector sobre el que realizar la operación correspondiente, usando el siguiente mensaje: “Elija vector para realizar la operación (1) para v1 o (2) para v2: “ y el usuario podrá seleccionar así el vector. En caso de introducir una opción incorrecta imprimirá por consola el error correspondiente y volverá al bucle principal.

Como es lógico, todas las opciones usarán siempre que sea necesario las funciones que se han descrito al principio de este guión. Las comprobaciones sobre rangos y tamaños deberán realizarse en el programa principal. En caso de ocurrir errores, deberá imprimirse por consola el error correspondiente y volver al bucle principal.

Debes tener en cuenta todas las normas que se han visto en relación al desarrollo de subrutinas, utilizando los registros adecuados tanto en el paso de parámetros como en la devolución de resultados. Pon especial atención en el tratamiento de la pila, y en el convenio de uso de los registros (salvados vs temporales). **La nota de la práctica dependerá en gran medida de esto, aparte de que funcione correctamente**.

Junto con este enunciado te suministramos el segmento de datos del programa.

Cada subrutina tendrá que tener una etiqueta de inicio que coincida con el nombre que figura en el enunciado (<nombre>) y al final de la misma (después de todas sus instrucciones) otra con el mismo nombre pero terminada en _fin. Es decir tendrán que estar definidas las etiquetas: `print_vec` y `print_vec_fin`, `change_elto` y `change_elto_fin`, `swap` y `swap_fin`, `mirror` y `mirror_fin`, `mult_add` y `mult_add_fin`, `prod_esc` y `prod_esc_fin`. Además los parámetros de entrada y salida se asignarán siguiendo estrictamente el orden en que aparecen en el enunciado dentro del grupo de registros correspondiente según el convenio visto en las tutorías TA2 y TA3.

Recuerda que valoramos muchos aspectos aparte del correcto funcionamiento del programa: que se pueda identificar correctamente el autor y la fecha de la última modificación, que esté bien comentado, que su lectura sea fácil (indentación, correcto uso de identificadores y etiquetas, etc), y que esté documentado el uso de los registros. Además debes incluir el código en C++ (o lenguaje de alto nivel) que resuelve la práctica, añadiéndolo como comentarios al principio de la misma y por partes en las secciones correspondientes.

Pasaremos un test de similitud entre las prácticas, y serán suspendidas con un cero las prácticas con un alto grado de coincidencia, independientemente de quiénes sean los autores originales y los plagiadores. Puedes ayudar a tus compañeros con explicaciones, pero es muy mala idea por este motivo dejarles código o partes del mismo. Invítalo a que pregunte sus dudas al profesor de prácticas correspondiente.

En las siguientes páginas se muestran algunos ejemplos de ejecución.

Console

```
Practica 4 de Principios de Computadores. Subrutinas.

Vector con dimension 40
10.00000000 11.00000000 12.00000000 13.00000000 14.00000000 15.00000000 16.00000000 17.00000000 18.00000000 19.00000000 20.00000000 21.00000000 22.00000000 23.00000000 24.00000000 25.00000000
26.00000000 27.00000000 28.00000000 29.00000000 30.00000000 31.00000000 32.00000000 33.00000000 34.00000000 35.00000000 36.00000000 37.00000000 38.00000000 39.00000000 40.00000000 41.00000000
42.00000000 43.00000000 44.00000000 45.00000000 46.00000000 47.00000000 48.00000000 49.00000000

Vector con dimension 40
40.00000000 39.00000000 38.00000000 37.00000000 36.00000000 35.00000000 34.00000000 33.00000000 32.00000000 31.00000000 30.00000000 29.00000000 28.00000000 27.00000000 26.00000000 25.00000000
24.00000000 23.00000000 22.00000000 21.00000000 20.00000000 19.00000000 18.00000000 17.00000000 16.00000000 15.00000000 14.00000000 13.00000000 12.00000000 11.00000000 10.00000000 9.00000000
8.00000000 7.00000000 6.00000000 5.00000000 4.00000000 3.00000000 2.00000000 1.00000000

(1) Cambiar dimension de un vector
(2) Cambiar un elemento de un vector
(3) Invertir un vector
(4) Calcular el producto escalar de dos vectores
(0) Salir

Elija opcion: 1

Elija vector para realizar la operacion (1) para v1 (2) para v2: 1

Introduzca nueva dimension para el vector (1-40): 4

Vector con dimension 4
10.00000000 11.00000000 12.00000000 13.00000000

Vector con dimension 40
40.00000000 39.00000000 38.00000000 37.00000000 36.00000000 35.00000000 34.00000000 33.00000000 32.00000000 31.00000000 30.00000000 29.00000000 28.00000000 27.00000000 26.00000000 25.00000000
24.00000000 23.00000000 22.00000000 21.00000000 20.00000000 19.00000000 18.00000000 17.00000000 16.00000000 15.00000000 14.00000000 13.00000000 12.00000000 11.00000000 10.00000000 9.00000000
8.00000000 7.00000000 6.00000000 5.00000000 4.00000000 3.00000000 2.00000000 1.00000000

(1) Cambiar dimension de un vector
(2) Cambiar un elemento de un vector
(3) Invertir un vector
(4) Calcular el producto escalar de dos vectores
(0) Salir

Elija opcion: 1

Elija vector para realizar la operacion (1) para v1 (2) para v2: 2

Introduzca nueva dimension para el vector (1-40): 4

Vector con dimension 4
10.00000000 11.00000000 12.00000000 13.00000000

Vector con dimension 4
40.00000000 39.00000000 38.00000000 37.00000000

(1) Cambiar dimension de un vector
(2) Cambiar un elemento de un vector
(3) Invertir un vector
(4) Calcular el producto escalar de dos vectores
(0) Salir

Elija opcion:
```

```
Elija opcion: 6

Error: opcion incorrecta.

Vector con dimension 4
10.00000000 11.00000000 12.00000000 13.00000000

Vector con dimension 4
40.00000000 39.00000000 38.00000000 37.00000000

(1) Cambiar dimension de un vector
(2) Cambiar un elemento de un vector
(3) Invertir un vector
(4) Calcular el producto escalar de dos vectores
(0) Salir

Elija opcion: 2

Elija vector para realizar la operacion (1) para v1 (2) para v2: 3

Error: opcion incorrecta.

Vector con dimension 4
10.00000000 11.00000000 12.00000000 13.00000000

Vector con dimension 4
40.00000000 39.00000000 38.00000000 37.00000000

(1) Cambiar dimension de un vector
(2) Cambiar un elemento de un vector
(3) Invertir un vector
(4) Calcular el producto escalar de dos vectores
(0) Salir

Elija opcion: 2

Elija vector para realizar la operacion (1) para v1 (2) para v2: 1

Elija el indice del elemento a cambiar: 3

Introduce nuevo valor para el elemento elegido: 99.99

Vector con dimension 4
10.00000000 11.00000000 12.00000000 99.98999786

Vector con dimension 4
40.00000000 39.00000000 38.00000000 37.00000000

(1) Cambiar dimension de un vector
(2) Cambiar un elemento de un vector
(3) Invertir un vector
(4) Calcular el producto escalar de dos vectores
(0) Salir

Elija opcion:
```

```
(1) Cambiar dimension de un vector
(2) Cambiar un elemento de un vector
(3) Invertir un vector
(4) Calcular el producto escalar de dos vectores
(0) Salir

Elija opcion: 3

Elija vector para realizar la operacion (1) para v1 (2) para v2: 2

Vector con dimension 4
10.00000000 11.00000000 12.00000000 99.98999786

Vector con dimension 4
37.00000000 38.00000000 39.00000000 40.00000000

(1) Cambiar dimension de un vector
(2) Cambiar un elemento de un vector
(3) Invertir un vector
(4) Calcular el producto escalar de dos vectores
(0) Salir

Elija opcion: 4

El producto escalar de los vectores es: 5255.59960938

Vector con dimension 4
10.00000000 11.00000000 12.00000000 99.98999786

Vector con dimension 4
37.00000000 38.00000000 39.00000000 40.00000000

(1) Cambiar dimension de un vector
(2) Cambiar un elemento de un vector
(3) Invertir un vector
(4) Calcular el producto escalar de dos vectores
(0) Salir

Elija opcion:  0

FIN DEL PROGRAMA.
```