Laboratorio 1, Introduccion a MIPS

Alejandro Cortés (6 horas)
Departamento de Ingeniería Informática
Universidad de Santiago de Chile, Santiago, Chile
alejandro.cortes.c@usach.cl

***Resumen*—En este informe verás como se implementan operaciones básicas sobre vectores, pero en un lenguaje de bajo nivel similar a "ensamblador", en una arquitectura en particular, llamada MIPS, para lograr esto se trabajara con memoria, teniendo que leer y escribir en ella, como también con entradas por teclado, para así lograr introducirnos en este mundo.**

***Palabras claves*—Introducción, Arquitectura, Memoria, MIPS y Consola.**

I. Introducción

Se debe trabajar en el desarrollo de un módulo lógico-matemático para una nave no tripulada, que será utilizada en experimentos para explorar el océano, para ello se trabajara con una arquitectura en MIPS, por ende es necesario implementar un programa que maneje vectores con este sistema. Los vectores serán tridimensionales y estarán presentados de la forma:

$$v = [x, y, z] \tag{1}$$

Por lo que se espera generar un programa en MIPS capaz de trabajar con estos vectores, realizando operaciones vectoriales básicas, como la suma, multiplicación escalar, producto punto y producto cruz.

II. Antecedentes

Para poder realizar esto, primero hay que saber como funcionan estas operaciones en los vectores, teniendo así distintas operaciones, las cuales deben ser realizadas mediante los procedimientos que serán descritos.

II-A. Suma de vectores

Para realizar la suma de dos vectores, denominados:

$$\overrightarrow{v} = (x_1, y_1, z_1) \tag{2}$$

$$\overrightarrow{u} = (x_2, y_2, z_2) \tag{3}$$

debemos sumar independientemente cada uno de sus componentes, obteniendo así un vector $\overrightarrow{r}$ de la misma dimensión que los dos vectores anteriores.

$$\overrightarrow{r} = (x_1 + x_2, y_1 + y_2, z_1 + z_2) \tag{4}$$

II-B. Multiplicación por un escalar

Para realizar la multiplicación de un escalar denominado s, por un vector denominado $\overrightarrow{v}$, de la forma:

$$\overrightarrow{v} = (x_1, y_1, z_1) \tag{5}$$

debemos Multiplicar el escalar por cada uno de los componentes del vector, obteniendo así un vector $\overrightarrow{r}$ de la misma dimensión que el vector anterior.

$$\overrightarrow{r} = (s * x_1, s * y_1, s * z_1) \tag{6}$$

II-C. Producto escalar (Producto punto)

Para realizar el producto punto entre dos vectores, denominados:

$$\overrightarrow{v} = (x_1, y_1, z_1) \tag{7}$$

$$\overrightarrow{u} = (x_2, y_2, z_2) \tag{8}$$

debemos multiplicar cada uno de los componentes con su homólogo del otro vector, para luego sumarlos entre sí, obteniendo así un escalar.

$$r = x_1 * x_2 + y_1 * y_2 + z_1 * z_2 \tag{9}$$

II-D. Producto cruz

Para realizar el producto punto entre dos vectores, denominados:

$$\overrightarrow{v} = (x_1, y_1, z_1) \tag{10}$$

$$\overrightarrow{u} = (x_2, y_2, z_2) \tag{11}$$

debemos calcular el determinante de una matriz donde están contenidos ambos vectores de la forma:

$$det \begin{vmatrix} \hat{i} & \hat{j} & \hat{i} \\ x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \end{vmatrix} \tag{12}$$

Y para calcular su determinante, usamos la siguiente fórmula:

$$r = (y_1 * z_2 - z_1 * y_2)\hat{i} - (x_1 * z_2 - z_1 * x_2)\hat{j} + (x_1 * y_2 - y_1 * x_2)\hat{k} \tag{13}$$

Lo cual después puede ser escrito como vector, ya que está en función de ($\hat{i}$, $\hat{j}$, $\hat{k}$).

III. Materiales y métodos

En esta sección tendremos dos subsecciones: materiales y métodos.

III-A. Materiales

Además de lo matemático detrás de esto, se debe tener en cuenta que trabajaremos con un simulador llamado MARS, el cual funciona a través de JAVA y simula una arquitectura MIPS, que además ocupa su propio lenguaje tipo "ensamblador" (de bajo nivel, lo más cercano a la máquina posible) también llamado MIPS, donde principalmente se tiene en consideración 2 cosas importantes, funciona a través de la consola, por ende todo el I/O es por consola, además, tiene sus propios registros (que son la manera más rápida de acceder a datos) y tiene una memoria, a la cual se puede acceder, pero es un proceso más lento.

III-B. Métodos

Para la suma de dos vectores se requiere leer desde la memoria ambos vectores y devolver el resultado por consola, por ende en MIPS primero hay que escribir las instrucciones necesarias para leer desde la memoria los vectores (con la instrucción lw y una dirección de memoria del formato [0x100100XX], dejando esto dentro de un registro temporal $tx), luego se realizan los procedimientos de suma necesarios (como se muestra en: II-A) y se deja el resultado listo para ser mostrado en pantalla, de manera que sea entendible para un usuario.
Posteriormente, se realiza lo mismo para el producto de un vector por un escalar, donde hay que leer desde la memoria el vector, pero a diferencia de antes, ahora el usuario ingresa el escalar por consola, el cual debe ser un número entero por el que se multiplicará cada componente del vector (según se puede ver en: II-B)
Luego, hay que realizar el producto punto o producto escalar (expresado en II-C), leyendo también el primer vector desde consola, y el segundo se solicita por consola, paso a paso, solicitando en cada línea una componente distinta, para finalmente mostrar el resultado, que esta vez no será un vector, sino un escalar (número).
Por último, como bonus/extra se realiza el producto cruz, siendo este el más complejo, y con más procedimientos, por ende requiere más registros y potencia de cálculo, además de requerir ingresar ambos vectores por consola, para luego (siguiendo los cálculos de II-D) obtener un nuevo vector, que será mostrado en la consola.

IV. Resultados

Para confirmar el correcto funcionamiento del programa se probaron cada uno de los programas según la función que realizan, de acuerdo a los métodos planteados en III-B. Para el primer caso, la suma de vectores, se guardaron 2 vectores en la memoria, el primero de ellos con los valores (2, -3, 4) y el segundo de ellos con (3, 4, -2) y podemos observar el resultado entregado por consola.

```
El vector resultante es : ( 5, 1, 2 )

-- program is finished running (dropped off bottom) --
```

Figura 1: Resultado de la suma

Por ende, se concluye que dado el algoritmo planteado, funcionará para cualquier valor x entero, tanto positivo como negativo, la misma generalización se da para cualquiera de los procedimientos realizados. Continuando con la multiplicación de un vector con un escalar, en este caso se guardó en memoria un vector con los valores (2, -3, 4), luego se solicita un escalar por consola, para finalmente mostrar el resultado por ahí mismo.

```
Ingrese el escalar por el cual se multiplicara el vector : -2

El vector resultante es : ( -4, 6, -8 )

-- program is finished running (dropped off bottom) --
```

Figura 2: Resultado de la Multiplicación por escalar

Posteriormente, se comprueba el funcionamiento del producto punto, este lee el primer vector desde la memoria y luego solicita los componentes del segundo vector uno a uno, para finalmente devolver por consola el producto punto o producto escalar de ambos vectores.

```
Ingrese la componente X del vector: 3

Ingrese la componente Y del vector: -1

Ingrese la componente Z del vector: -5

El resultado es : -11
-- program is finished running (dropped off bottom) --
```

Figura 3: Resultado del producto punto

Para finalizar, solamente queda el producto cruz, y en este caso no es necesario leer nada desde la memoria, se solicitarán ambos vectores por consola, de la misma manera que antes, esta vez el resultado es un vector, así que se muestra como tal por consola.

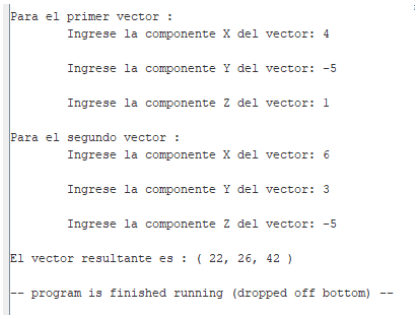

Figura 4: Resultado del producto cruz

V. Conclusiones

Efectúa una revisión del trabajo realizado, resumiendo los resultados obtenidos y la aportación de estos en el ámbito estudiado. Debe ser breve y concisa. Además, contiene los aspectos que se desprenden o quedaron fuera del alcance, que pueden ser considerados en actividades futuras. Como es posible observar, se cumplieron correctamente los objetivos planteados, fue posible realizar el programa requerido y realizo las operaciones solicitadas de manera correcta, entregando los resultados esperados para los ejemplos introducidos, como para otros ejemplos probados, además se logró generar conocimiento básico respecto a MIPS, tanto de sus instrucciones como de su arquitectura (registros, memoria e instrucciones, especialmente), de momento solo se planteó con números enteros, por lo que queda para el futuro implementar esto para todos los números racionales.[1]

Referencias

[1] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design MIPS Edition: The Hardware/Software Interface*, Amsterdam ; Boston, Oct. 2013.