ESTRUCTURA DE COMPUTADORES

GRADO EN INGENIERÍA INFORMÁTICA

Práctica 1 Introducción al lenguaje ensamblador

Curso 2019/2020

Jorge Rodríguez Fraile, 100405951, Grupo 81, 100405951@alumnos.uc3m.es
Carlos Rubio Olivares,100405834 , Grupo 81, 100405834@alumnos.uc3m.es

Índice

Ejercicio 1

inicializar

Respecto a este método, hemos empezado comprobando si M o N eran menores o iguales a 0 con 'ble', si esto ocurre, saltará a la etiqueta error, que le da a $v0 el valor -1 y termina el programa.

Respecto al resto del código, simplemente obtenemos el número de elementos de la matriz, creamos dos índices, uno que será el que indique la posición de cada elemento de memoria ($t5) y otro para el número de iteraciones ($t6). Para $t5 lo que haremos será sumarle la dirección de memoria inicial de la matriz, y dentro del bucle, se incrementará 4 (cada elemento de la matriz ocupa 4 bytes). Respecto a $t6, simplemente se incrementará una vez en cada iteración.

Para rellenar la matriz de 0's, simplemente creamos un bucle, cuando nuestro índice sea mayor que el número total de elementos, se irá a la etiqueta que finaliza el programa. En su interior, se copia el valor 0 donde esté apuntando $t5, y se incrementan nuestros índices como se ha indicado anteriormente.

Este método no nos ha dado muchos problemas, ya que la mayor dificultad residía en saber incrementar y usar de manera correcta nuestro índice para ir recorriendo la matriz.

Datos a introducir:	**Descripción de la prueba:**	**Resultado esperado:**	**Resultado obtenido:**
M = -2 N = 4	Vemos si al ser el número de filas negativo el programa nos devuelve un -1.	Esperamos que el programa nos devuelva un -1.	$v0 = -1: Correcto
M = 5 N = -2	Probamos si al tener las columnas un número negativo obtenemos un -1 como respuesta.	De nuevo, un -1 como valor en $v0.	$v0 = -1: Correcto
M = 5 N = 4	Prueba de una matriz con dimensiones correctas.	Devuelva 0 y la matriz esté completa de 0's.	$v0 = 0: Correcto Matriz A de 0's.

M = 0 N = 4	Prueba de si salta error al introducir un 0 en las columnas.	El registro $v0 contiene un -1	$v0 = -1: Correcto

sumar

En este método tenemos el argumento N (número de columnas) en pila, ya que el número de argumentos es 5, y según el convenio, sólo podríamos guardar 4 en registros.

De nuevo, debemos comprobar si M y N son menores o iguales a 0, por lo que usamos el mismo método que en inicializar. Para realizar la suma, debemos tener 3 índices que nos marquen en todo momento en que posición de memoria estamos en la matriz A, B y C, por lo que usaremos $t4, $t5 y $t6 respectivamente.

Después de crear estos índices, entramos en el loop, en cuanto nuestro iterador sea mayor que todos lo elementos de cualquiera de las matrices operandos (ya que deben tener la misma dimensión), salta a la etiqueta que termina el programa. En cada iteración vamos a cargar en $t0 y $t2 el valor i,j de cada matriz, los sumaremos en $t3, y copiaremos ese valor dentro de dónde esté apuntando $t4, el índice de la matriz resultado. Por último, incrementaremos el valor de los índices en 4, y del iterador en 1.

Datos a introducir:	**Descripción de la prueba:**	**Resultado esperado:**	**Resultado obtenido:**
Dos matrices cuadradas de 2x2 tal que: B = {{1,2},{3,4}} y C = {{4,3},{2,1}}	Vemos si la suma es correcta con datos sin ningún error.	Una matriz de 2x2 con todas sus posiciones de valor 5.	A = {{5,5},{5,5}} Correcto
M = 0 o N = 0	Vemos si salta error al ver que las filas o las columnas sean nulas.	No se realiza la suma y $v0 = -1.	$v0 = -1 Correcto
M o N negativos.	Vemos si nos aparece error al ver que las filas o columnas son negativas.	No se realiza la suma y $v0 = -1.	$v0 = -1 Correcto

extraerFila

En este método nos volvemos a encontrar otro valor en pila, ya que tenemos 5 argumentos de nuevo, en este caso será el número de la fila que queremos extraer.

Esta vez, además de hacer el mismo control de error que en los dos métodos anteriores, vamos a tener que hacer uno extra para j, así que utilizamos un 'bltz' (branch if lower than zero) y un 'bgt' (branch if greater than) para comprobar que no es menor que 0, ni mayor que el número de filas.

Volvemos a usar dos índices de memoria para A y B ($t1,$t2) y para localizar en memoria la fila a la que vamos a querer acceder usamos: dir.inicial.B + (fila*columnas*4). Ya que por ejemplo, si queremos llegar a la fila 2 en una matriz de 3 columnas, pasaremos por 3 valores, 2 veces. Este valor se almacenará en $t2, sustituyendo al valor de la dirección inicial de B, que no necesitaremos.

En el bucle, simplemente copiamos el valor de B[i,j] en A[j] hasta que nuestro iterador sea más grande que el número de columnas. Incrementamos en 4 el valor de $t1 y $t2, y en 1 el de nuestro iterador.

Datos a introducir:	**Descripción de la prueba:**	**Resultado esperado:**	**Resultado obtenido:**
Matriz de 3x3 con datos {{4,6,7},{87,5,76}, {543,23,12}}, hemos pedido la fila 2.	Comprobar si el resultado está bien con datos introducidos correctamente.	A = {87,5,76}	A = {87,5,76} Correcto
j>M 7>3	Pedir una fila que exceda el número de columnas existentes.	$v0 = -1	$v0 = -1 Correcto
M o N negativo y/o nulos.	Introducimos valores negativos y nulos en las filas y columnas para comprobar que no se realiza la copia de la fila y el programa para.	$v0 = -1	$v0 = -1 Correcto

j negativo.	Damos valores erróneos a j para ver si no se realiza la copia en el vector A.	$v0 = -1	$v0 = -1 Correcto
j=0	Probar si acepta el valor 0 como fila a copiar.	Copiar la primera fila.	Copia la primera fila, A = {4,6,7} Correcto

masCeros

Para la función masCeros, tenemos 4 argumentos, por lo que no hay nada almacenado en pila, pero, en cambio, usaremos 3 valores en pila: $s6, $s3 y $s4, que usaremos para intercambiar direcciones de A y B, pudiendo usar las funciones respetando el convenio.

Para empezar, guardamos espacio en pila para estos valores, y redireccionamos los argumentos para utilizar calcular con A, guardando la dirección inicial de B en $s4. También se debe tener en cuenta que tenemos que guardar $ra en pila, ya que si no, el programa se repetirá infinitamente.

Al llamar a la función calcular con A, utilizando como argumento de valor 0, guardamos el resultado en $s6. Volvemos a redireccionar los argumentos, esta vez la dirección de A se guarda en $s4. Llamamos a la función calcular con B, y el resultado lo guardamos en $s3.

Por último, simplemente comparamos $s3 y $s6, si $s6>$s3, devolvemos un 0, si $s3>$s6, devolvemos un 1, y si son iguales,un 2. Al terminar esto volvemos a vaciar espacio en pila, y termina el programa.

En esta función hemos tenido problemas a la hora de almacenar valores en pila, ya que la función calcular nos modificaba incluso algunos registros $s, lo hemos podido solucionar almacenando los valores en los registros que no se modificaban en calcular.

Datos a introducir:	**Descripción de la prueba:**	**Resultado esperado:**	**Resultado obtenido:**
M y N menores o iguales a 0	De nuevo, vemos si este programa no realiza ninguna operación y sólo devuelve un -1	$v0 = -1	$v0 = -1 Correcto

Una matriz A con 4 0's. Una matriz B con 3 0's	El caso de que la matriz A tenga más 0's que la B	$v0 = 0	$v0 = 0 Correcto
Una matriz B con 3 0´s Una matriz A con 4 0´s	El caso de que la matriz B tenga más 0's que la A	$v0 = 1	$v0 = 1 Correcto
Dos matrices con el mismo número de 0's (4)	El caso de que ambas matrices tengan el mismo número de 0´s	$v0 = 2	$v0 = 2 Correcto

Ejercicio 2

extraer valores

En esta función hemos empezado haciendo el mismo control de errores que en todos los anteriores, y después creamos espacio en pila para 3 valores, $s0 (lo utilizaremos para saber si hay un NaN), $s1 (iterador de las direcciones de memoria en la matriz A) y $s2 (valor a comparar). Creamos nuestros contadores, que más tarde irán a las posiciones del vector en su correspondiente orden y empezamos el bucle.

Para empezar, copiamos el valor que hay en el registro de nuestro iterador de direcciones de $s2 y empezamos a comparar con máscaras para ver en cuál de las categorías cae nuestro valor:

-Para compararlo con 0, simplemente utilizamos 0x0000 0000, si coincide, se incrementa el contador correspondiente.

-En cuanto a el infinito negativo utilizamos 0xFF80 0000 (1 1111 1111 000...00) y realizamos lo mismo que con el caso anterior.

-Con el infinito positivo es exactamente igual, solo que nuestra máscara será 0x7F80 0000 (0 1111 1111 000...0)

-En cuanto a NaN, el proceso es algo más complicado, primero copiamos nuestro valor en $s0, para poder rotar lógicamente una vez hacia la izquierda para eliminar nuestro signo, y 24 veces a la derecha para eliminar la mantisa. Con esto, nos quedamos sólo con el exponente, si este coincide con 0xFF (1111 1111) obviamente es un NaN

-Con los no normalizados, nos quedamos con el exponente de nuestro número, ($s0), si coincide con 0x00, por eliminación podemos categorizarlo como no normalizado.

-Por último, si no es ninguna de los anteriores, tiene que ser un número normalizado válido.

-Iteramos tanto nuestro índice 1 vez como nuestro iterador de dirección de memoria 4 veces.
Una vez terminado la cuenta de todos los elementos de la matriz, simplemente introducimos cada contador en su posición asignada y volvemos a restablecer las posiciones de pila.

Datos a introducir:	**Descripción de la prueba:**	**Resultado esperado:**	**Resultado obtenido:**
M y N menores o iguales a 0. (M= 0, N = -2)	Probar que el programa salta a error_extraer y no calcula nada.	$v0 = -1	$v0 = -1 Correcto
Matriz A = {{1,2,3}; {4,5,6}; {7,8,9}}	Probar que la condicional de comprobar un número normalizado funciona.	V = {0,0,0,0,0,9}	V = {0,0,0,0,0,9} Correcto
Matriz A = {{0,0,0}; {0,0,0}; {0,0,0}}	Probar que la condicional de comprobar que un número es 0 funciona.	V = {9,0,0,0,0,0}	V = {9,0,0,0,0,0} Correcto
Matriz A = {{0xFF800000,0xFF800000,0xFF800000}, {0xFF800000,0xFF800000,0xFF800000}, {0xFF800000,0xFF800000,0xFF800000}}	Probar que la condicional de comprobar que hay un infinito positivo funciona .	V = {0,9,0,0,0,0}	V = {0,9,0,0,0,0} Correcto
Matriz A = {{0x7F800000,0x7F800000,0x7F800000}, {0x7F800000,0x7F800000,0x7F800000}, {0x7F800000,0x7F800000,0x7F800000}}	Probar que la condicional de comprobar que hay un infinito negativo funciona	V = {0,0,9,0,0,0}	V = {0,0,9,0,0,0} Correcto

Matriz A = {{0x7F806700,0xFF800120,0xFF821050}, {0x7F810100,0x7F829190,0xFF801350}, {0x7F811110,0xFF813020,0x7F856100}	Probar que la condicional de comprobar que es NaN funciona	V = {0,0,0,9,0,0}	V = {0,0,0,9,0,0} Correcto
Matriz A = {0x00000403,0x00001402,0x00050510},{0x00012232,0x00000411,0x00030333}, {0x00010202,0x00000578,0x00020323}}	Probar que la condicional de comprobar que es un número no normalizado funciona	V = {0,0,0,0,9,0}	V = {0,0,0,0,9,0} Correcto
Matriz A = {{0xFF800000,2,0x7F800000}; {0x7F806700,0x00000403,0x7F806700}; {0,0xFF800000,0x00020323}}	Probar que juntando todos los casos anteriores el programa funciona correctamente	V = {1,2,1,2,2,1}	V = {1,2,1,2,2,1} Correcto

sumar

En la última función de la práctica, tenemos otro argumento en pila, por lo que lo copiamos en el registro $t0 y volvemos a hacer nuestro control de error. Multiplicamos las filas y las columnas para saber donde para nuestro loop, y lo guardamos en $t1, nuestro índice será $t2. Copiamos las direcciones iniciales de A,B y C en $t3, $t4 y $t5 respectivamente, y empezamos el loop.

La estructura del bucle es muy parecida a la función sumar del ejercicio 1:

1. Copiamos en $f4 (valor de A) y en $f5(valor de B) el valor que se encuentra en el registro al que apuntan nuestras direcciones actuales de A y B.
2. Sumamos $f4 y $f5 y lo almacenamos en $f6
3. Copiamos $f6 en el registro donde se encuentre el iterador de direcciones de C
4. Incrementamos $t2 en 1 y $t3,$t4 y $t5 en 4.

Datos a introducir:	**Descripción de la prueba:**	**Resultado esperado:**	**Resultado obtenido:**
M y N menores o iguales a 0	Probar que el programa detecta el error y no calcula nada	$v0 = -1	$v0 = -1 Correcto
Matriz A = {{0x3FC00000,0x40B33333},{0x4059999A,0x400CCCCD}} Matriz B = {0x3FC00000,0x4019999A},{0x40666666,0x40F9999A}}	Probar que se realiza la suma correctamente	Matriz C = {{0x40400000,0x41000000},{0x40E00000,0x41200000}}	Matriz C = {{0x40400000,0x41000000},{0x40E00000,0x41200000}} Correcto