DIARIO DE PRÁCTICAS

Luis Blanco de la Cruz
Roberto Gozalo Andrés

Primera Práctica 19/09/18

Ejercicio 1:

a)
En el simulador Mars, tras ejecutar nuestro programa vemos que es Little Endian porque en la dirección más baja del registro se guarda el bit más significativo, en este caso el 0, si fuera Big Endian se guardaría un 1. Ambas instrucciones son sintéticas, porque el compilador las cambia por otras básicas.

b)
Después de muchos errores, conseguimos compilar y ejecutar el programa correctamente.

c)
En nuestro programa utilizamos estas instrucciones:
-li: La cambia por addi.
-la: La cambia por lu+ori.
-b: La cambia por bgz.
-lw: Cuando se utiliza una etiqueta se transforma en lui.

d)
Todas las instrucciones sintéticas las marca como errores. Si, ahora el programa es mucho más difícil de leer que antes.

Ejercicio 2:

a)
No se traducen las instrucciones de la misma manera. la instrucción li se traduce como addiu en Mars y como ori en QtSPIM.

b)
bne no nos la marca como instrucción sintética pero probamos con beq y en Mars la traduce como addi+beq y en QtSPIM como ori+beq. Ambas son correctas, la traducción depende de cómo esté escrito el compilador de cada programa.

Ejercicio 3:

Como pone en el guión, al marcar la casilla de Bkpt se para el programa, si se marca en la casilla de un bucle el programa se para en cada iteración.

Segunda Práctica 26/09/18

Ejercicio 1:

Como se indica en el guión vamos a crear un programa para realizar el siguiente cálculo sobre matrices cuadradas:
B[i][j] = A[i][j] − A[j][i]
Dicho programa tiene como argumentos las direcciones de ambas matrices así como el número de filas de ellas (ya que son matrices cuadradas no necesitamos conocer el número de columnas porque es igual al de las filas).
Para comenzar con nuestro programa decidimos crear una función que hemos llamado calcula y que realizará el cálculo de la dirección
A[i][j] aplicando la fórmula obtenida en el guión.
A[i][j] = A +4 ∗ m ∗ i + 4 ∗ j
Durante la creación de dicha porción de código nos damos cuenta que nuestra función se puede optimizar sacando factor común quedando de la siguiente manera
A[i][j] = A +4 ∗ (m ∗ i + j)

Al principio nuestro programa no funciona correctamente, se nos había olvidado la gestión de la pila. Para solucionar esto primero debemos decrementar el valor de la pila, ya que esta apunta a la dirección más alta y antes de salir de la función devolver el apuntador a la dirección más alta.

Para recorrer las funciones necesitamos dos bucles for anidados, al primero lo llamaremos loopI y al otro loopJ. Tras pasar el profesor por nuestro puesto y solucionarnos algunas dudas que nos habían surgido conseguimos que se recorran las funciones correctamente. Lamentablemente la sesión de laboratorio llega a su fin y debemos de pausar nuestro trabajo.

Ya en nuestras casas y mediante videoconferencia retomamos el trabajo donde lo habíamos dejado en clase.

Nos damos cuenta de que nuestro programa no realiza el cálculo correctamente, ya que nos aparecen los valores correctos multiplicados por 4.

Tras revisar nuestro código nos damos cuenta de que no estábamos almacenando $t2 y $t3. Almacenamos mediante la instrucción lw dichos registros en $t6 y $t7 respectivamente. Ahora si nuestro programa realiza la operación correctamente para matrices de 4*4.

28/09/18

Nos ponemos ahora con la función que imprimirá la matriz, a la que llamamos

funcionB en previsión de que la que pida la matriz por pantalla sea la A.

Creamos dos cadenas ascii para imprimir un espacio y una tabulación para que quede mejor la matriz al imprimir.

Creamos dos bucles que iterarán hasta el tamaño de matriz que tengamos introducido en $a2, así podemos cambiar el tamaño más fácilmente cambiando solo un registro. Cogemos el código de la función calcula para calcular la posición a imprimir y lo introducimos en el bucle. Añadimos bifurcaciones para que imprima tabulaciones o saltos de línea cuando toque.

Al probarlo nos imprimía la matriz correctamente tabulada pero solo con 0s. Después de revisar el código, nos dimos cuenta que estábamos imprimiendo B, que estaba vacía ya que solo ejecutábamos la función de imprimir y no la de calcular. Al añadirla al bucle, ya funcionaba correctamente e imprimía los valores adecuados.

29/09/18

Nos disponemos a hacer la función de pedir los datos por pantalla. Después de echar un ojo a la referencia de mips conseguimos que almacene el dato introducido en la matriz.

Ahora creamos otro bucle y volvemos a copiar el codigo para calcular la posición en la que introducir el dato. Después de un par de errores por repetir registros en varias funciones, vemos que funciona correctamente.

Decidimos poner un poco más bonito la parte de introducción de datos y añadimos otras cadenas ascii. Ahora al pedirte los datos te mostrará un mensaje y la posición en la que introduces los datos; así será más fácil saber donde pones cada número.