

SICAM

Práctica 4 de Arquitectura de Computadores

José Márquez Doblas GG2

El programa realizado en la práctica es un programa sencillo que consiste en un bucle que hace diversos accesos a memoria mediante las instrucciones LW y SW (Load Word y Save Word respectivamente).

TRADUCTOR DE INSTRUCCIONES MIPS					
D. Hexad	Etiqueta	Instrucción	Rd/Rt	Rs/Off	Rt/Imm
00000300	LOOP	ADDI	\$1	\$1	4
00000304		LW	\$2	0	\$1
00000308		SW	\$3	24	\$1
0000030C		BNEQ	\$1	\$4	LOOP
00000310		NOP			
00000314		NOP			
00000318		NOP			
0000031C		NOP			
00000320		NOP			
00000324		NOP			
00000328		NOP			
0000032C		NOP			
00000330		NOP			
00000334		NOP			
00000338		NOP			
0000033C		NOP			
00000340		NOP			
00000344		NOP			
00000348		NOP			
0000034C		NOP			
00000350		NOP			
00000354		NOP			
00000358		NOP			

REGISTROS			
R0	00000000	R1	00000000
R2	00000000	R3	000000FF
R4	00000024	R5	00000000
R6	00000000	R7	00000000
R8	00000000	R9	00000000
R10	00000000	R11	00000000
R12	00000000	R13	00000000
R14	00000000	R15	00000000
R16	00000000	R17	00000000
R18	00000000	R19	00000000
R20	00000000	R21	00000000
R22	00000000	R23	00000000
R24	00000000	R25	00000000
R26	00000000	R27	00000000
R28	00000000	R29	00000000
R30	00000000	R31	00000000
RHi	00000000	RLo	00000000

El funcionamiento del programa es el siguiente:

ADDI va sumando 4 al contenido del registro \$1 y guarda el resultado en ese mismo registro, LW carga en el registro \$2 la palabra cuya dirección es el contenido del registro \$1, de igual forma SW guarda el registro \$3, inicializado a FF por nosotros, en la palabra direccionada por el contenido del registro \$1 más 24. Finalmente BNEQ bifurcará a ADDI si el registro \$1 es diferente al registro \$4, inicializado a 24 por nosotros, terminando el programa cuando estos sean iguales, es decir, cuando el contenido del registro \$1 sea 24.

El contenido del registro \$2 no debería variar de 0 puesto que la primera dirección que modifica SW es la 28 y la última que carga LW es 24. Por lo tanto si nos quedase FF en el registro \$2 sabríamos que algo ha fallado en el bucle.

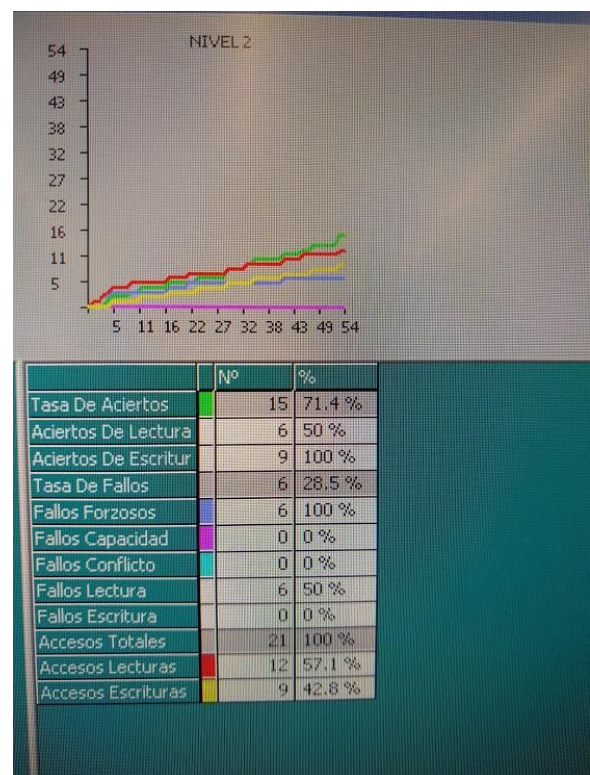
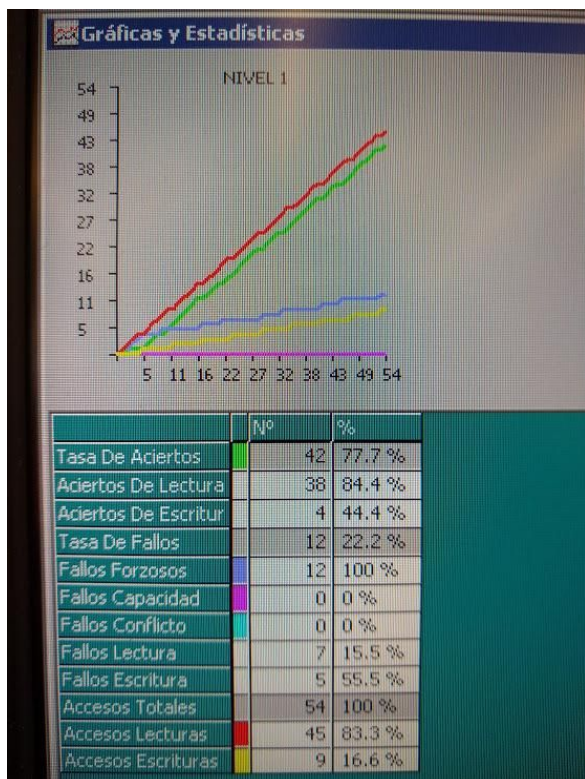
Tras realizar la simulación obtenemos los siguientes estados de los registros y la memoria principal:

REGISTROS			
R0	00000000	R1	00000024
R2	00000000	R3	000000FF
R4	00000024	R5	00000000
R6	00000000	R7	00000000

Dirección y Contenido de MP					
Dirección	Contenido	Dirección	Contenido	Dirección	Contenido
00000028	000000FF	0000002C	000000FF	00000030	000000FF
00000034	000000FF	00000038	000000FF	0000003C	000000FF
00000040	000000FF	00000044	000000FF	00000048	000000FF
00000300	20210004	00000304	8C220000	00000308	AC230024
0000030C	1424FFF0				

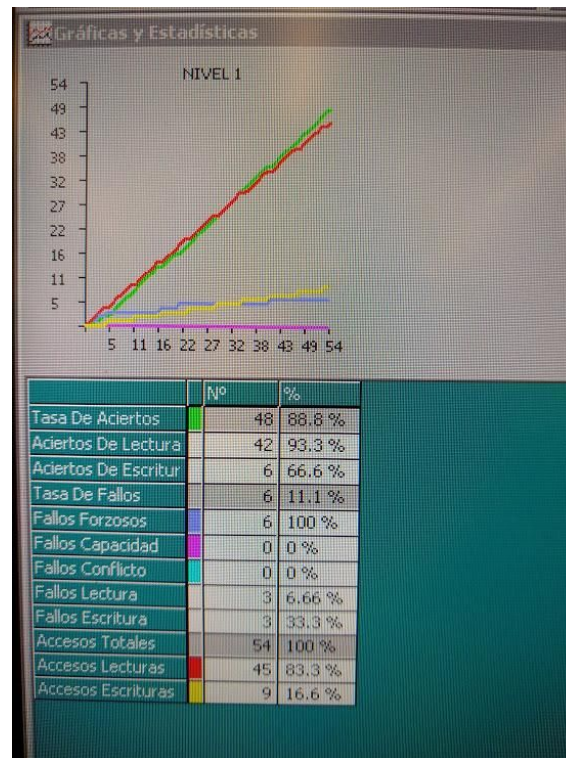
Tal y como he descrito antes, el contenido del registro \$2 quedará a 0 al finalizar el proceso.

Compilamos y ejecutamos el programa de simulación con su configuración por defecto, obteniendo los siguientes resultados:



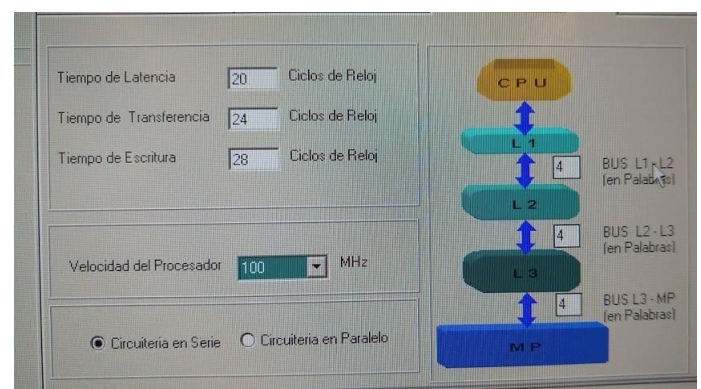
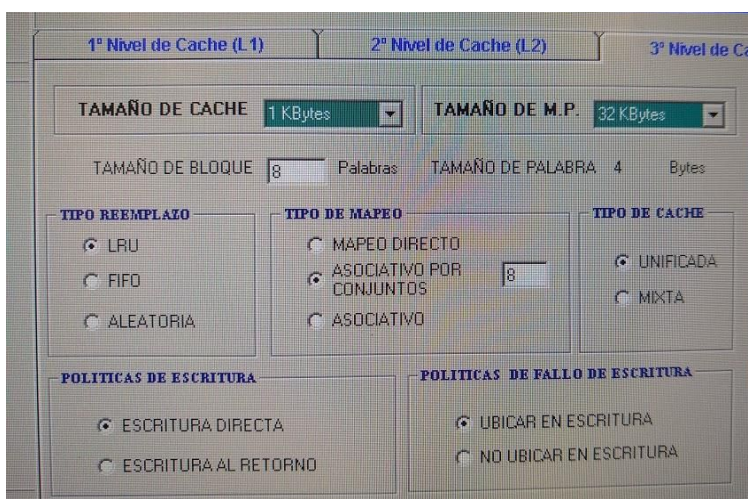
Podemos concluir de los resultados obtenidos que tanto la tasa de aciertos de L1 como la de L2 no son remarcables con un 77,7% y un 71,4% respectivamente. Si podemos analizar la tasa de aciertos en lectura y escritura por separado. En L2 vemos que la tasa de aciertos en lectura es del 50% mientras que en L1 es de 84,4%, esto nos dice que L1 es más eficiente en lectura que L2. Sin embargo, L2 tiene una tasa de aciertos en escritura del 100%, es decir, que todas las veces que ha querido escribir lo ha hecho, por el otro lado L1, con una tasa de aciertos en escritura del 44,4%, denota su baja eficiencia en escritura frente a L2, mucho más que su comparación en lectura.

Si cambiamos la configuración de memoria caché a un solo nivel y cambiamos su tamaño de bloque a 4 palabras, obtenemos los siguientes resultados:

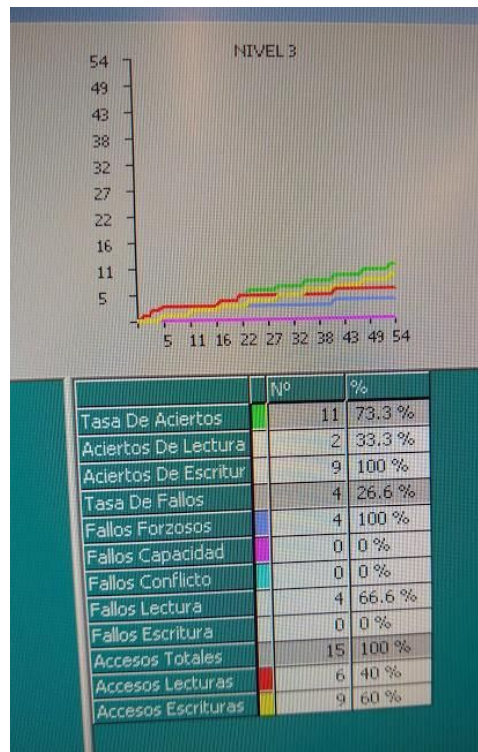


Dado que nuestro problema modifica direcciones de memoria próximas las unas a las otras era fácil concluir que iban a salir estos resultados. Al aumentar el tamaño de bloque, cada vez que copia un bloque se trae más palabras de memoria haciendo que la tasa de aciertos sea mayor, aunque sigue sin ser remarcable, aumentando además tanto la tasa de aciertos en lectura como en escritura, a pesar de que esta última sigue siendo bastante baja.

Por último cambiaremos nuestra configuración de memoria a un memoria caché de 3 niveles dejando la configuración de la primera simulación a L1 y L2 y teniendo L3 con la siguiente:



Y obtenemos los mismos resultados en L1 y L2 que en el primer ejemplo y los siguientes resultados en L3:



Podemos concluir de estos resultados que L3 es todavía menos eficiente que L2 en lectura pero mantiene su tasa de aciertos en escritura del 100%, sin embargo, tiene una tasa de errores menor ya que el número de accesos totales es menor también.