Proyecto 2 – 2023 – Jerarquía de memoria de datos

Naranjo Ventura, Elizabeth Lilai (840091)

Moncasi Gosá, Carlota (839841)

Breve resumen

En este proyecto, el sistema se compone de un periférico de entrada/salida llamado IO_input conectado a IO_Master que actúa como maestro del bus, el procesador MIPS del proyecto anterior y un nuevo subsistema de memorias: MD, MD_Scratch (solo admite direcciones no cacheables) que actúan como servidores y MC (direcciones cacheables) que actúa como maestro, todas se envían y reciben datos mediante un método de arbitraje. La memoria cache es asociativa con dos vías y cada una tiene 4 bloques de 4 palabras. Funciona con escritura directa (write-through), la política de fallo en escritura es AF y la política de reemplazo es FIFO.

Si el MIPS pide un dato mal alineado, o el controlador intenta comunicarse por el bus como maestro, y ningún servidor responde a la instrucción en ese ciclo, la dirección de la palabra culpable se almacena en el registro de error y en cuanto MIPS la lee, se sigue con la transferencia.

Si la instrucción, ya sea lectura o escritura, produce un fallo, hay que traer el bloque entero a memoria cache en la vía y conjunto correspondientes y ya realizar la lectura o escritura en MD.

Si por el contrario, es un acierto porque su tag ya se encuentra en la cache, entonces se realiza la lectura de MD directamente. Pero en caso de ser escritura, se tiene que pasar por la fase de arbitraje para poder escribir la palabra en concreto en MD.

Además del diseño de la UC de MC, también se hace uso de contadores, tales como *data_stalls* que gestiona riesgos de datos, *control_stalls* que se ocupa de riesgos de control y *paradas_mem* que contabiliza las paradas por memoria. Ahora se tiene en cuenta si memoria no está preparada para realizar la instrucción de la etapa MEM y se detienen todas las etapas, incluida WB incrementando paradas_mem pero no data_stalls, ya que la causa no ha sido un riesgo de datos.

Modificaciones MIPS proyecto 1

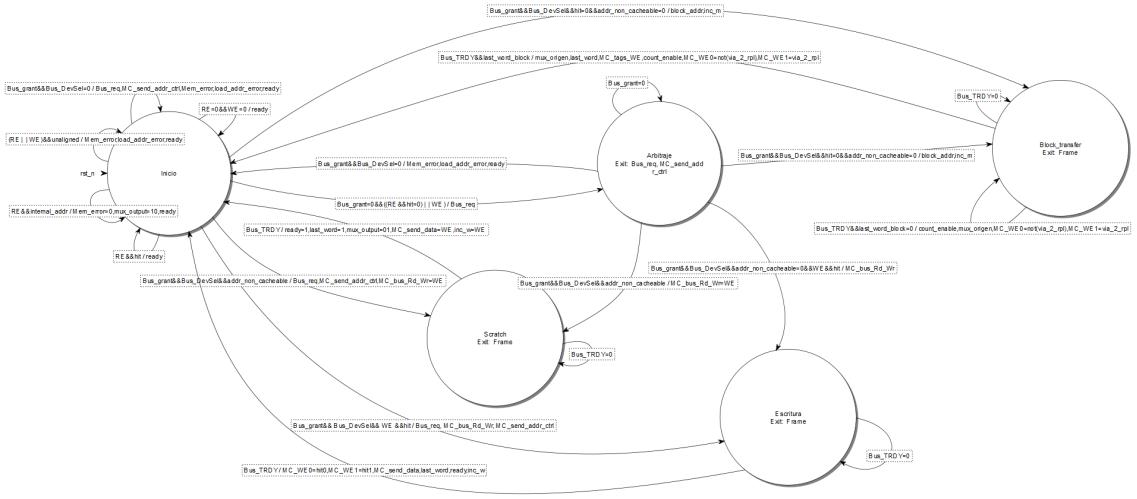
Cuando MC no puede realizar la operación solicitada en el ciclo actual baja la señal Mem_ready, y el procesador del MIPS detiene su ejecución hasta que esa señal vuelve a activarse. Si ocurre esto, no se deben cargar nuevas instrucciones en ninguna de las etapas (IF, ID, EX, MEM, WB) por lo que en la UD se ha modificado lo siguiente para que eso ocurra:

```
parar_EX_internal <= not(Mem_ready);
```

Además, se ha añadido un nuevo contador al MIPS para contabilizar el número de ciclos por paradas de memoria. Para activarlo, se ha añadido la señal inc_paradas_mem y, para no contabilizar dos veces la misma parada, esta señal habilita el nuevo contador mientras que deshabilita los de riesgos de datos y de riesgos de control de la siguiente manera:

```
if (Mem_ready = '0')
        inc_paradas_mem = '1';
        inc_data_stalls = '0';
        inc_control_stalls = '0';
```

Diagrama de estados de la unidad de control



Como se puede observar, el autómata tiene 5 estados: Inicio, Arbitraje, Scratch, Block_transfer y Escritura.

El primer estado gestiona los siguientes casos:

- No se realiza ninguna transferencia (RE=0 and WE=0).
- Se produce un acierto de lectura de MD (RE=1 and hit=1). En este caso, el dato ya se encuentra en MC y se envía directamente al MIPS mux_output=00.
- Se producen lecturas o escrituras de direcciones no alineadas ((RE=1 or WE=1) and unaligned=1). En este caso se pasa a estado de error (Mem_error=1) y se carga la

dirección que ha producido este error en el registro interno de MC (load_addr_error=1).

- Lecturas del registro interno de MC (RE=1 and internal_addr=1). Estas lecturas siempre producen hit, se sale del estado de error (Mem_error=0) y se envía el contenido del registro interno al MIPS (mux_output=10).

- Concesiones del bus para realizar una transferencia que lo necesite (fallos de lectura de MD, escrituras o transferencias con MD Scratch) pero sin respuesta de ningún servidor (Bus_DevSel=0). Como nadie nos ha respondido, pasamos a un estado de error (Mem_error=1) y se carga la dirección que ha producido este error en el registro interno de MC (load_addr_error=1).

Para todos los casos anteriores, nos quedamos en el estado de Inicio e indicamos ready=1 para esperar a la siguiente transferencia.

Además, el estado de **Inicio** gestiona otros casos que implican pasar a otro estado:

- Solicitud del bus para una transferencia pero no te lo conceden aún (Bus_grant=0). Si ocurre esto pasamos al estado Arbitraje.

- Lectura o escritura de una dirección no cacheable (addr_non_cacheable=1) y te han concedido el bus (Bus_grant=1). En este caso pasamos al estado Scratch indicando, además, si se trata de una lectura o una escritura (MC_send_addr_ctrl=WE).

- Aciertos de escritura en MD (WE=1 and hit=1 and addr_non_cacheable=0). En este caso pasamos al estado de Escritura, indicando que se trata de una escritura (MC_send_addr_ctrl=1).

- Fallos tanto de lectura como de escritura en MD (hit=0 and addr_non_cacheable=0). En este caso pasamos al estado Block_transfer para traernos el bloque correspondiente a MC. Indicamos que la dirección es la del bloque y no la de la palabra (block_addr=1) y al ser fallo incrementamos ese contador (inc_m=1).

Respecto al estado **Arbitraje**, este gestiona las ocasiones en las que has solicitado el uso del bus pero no te lo han concedido aún. Mientras no te lo concedan nos quedamos en este estado y, una vez te lo han concedido, vamos a los estados Scratch, Escritura o Block_transfer dependiendo del tipo de transferencia. Estos cambios de estado se dan con las mismas condiciones y salidas que los casos anteriormente explicados del estado Inicio a los estados Scratch, Block_transfer y Escritura. Además, si una vez concedido el bus ningún servidor te responde, pasamos del estado Arbitraje a Inicio pasando a un estado de error (Mem_error=1), cargando la dirección que ha producido este error en el registro interno de MC (load_addr_error=1) y poniendo ready=1 para indicar que esperamos la siguiente transferencia.

El estado **Scratch** gestiona todas las transferencias que se refieren a la memoria de datos Scratch (direcciones no cacheables). Mientras la transferencia no termine (Bus_TRDY=0) nos quedamos en este estado y, cuando el servidor nos indique que ya puede hacerla en el ciclo

actual, volvemos a Inicio indicando que es la última palabra (last_word=1), mandando el dato pedido si se trata de una lectura (mux_output=01), incrementando el contador de escrituras (inc_w=WE) y mandando el dato al bus (MC_send_data=WE) cuando sea una escritura y activando ready para indicar que estamos listos para la siguiente transferencia.

El estado de **Escritura** gestiona cuándo hay un acierto en escritura de una dirección cacheable. Mientras la transferencia no termine (Bus_TRDY=0) nos quedamos en este estado y, cuando el servidor nos indique que ya puede hacerla en el ciclo actual, volvemos a Inicio indicando que es la última palabra (last_word=1), escribiendo en MC donde corresponde (MC_WE0=hit0, MC_WE1=hit1), incrementando el contador de escrituras (inc_w=1), mandando el dato al bus (MC_send_data=WE) y activando ready para indicar que estamos listos para la siguiente transferencia.

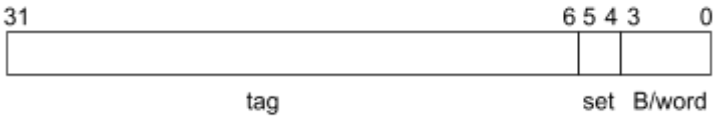
Por último, el estado **Block_transfer** gestiona los fallos tanto de escritura como de lectura de una dirección cacheable. Mientras la transferencia no termine (Bus_TRDY=0) nos quedamos en este estado y, cuando el servidor nos indique que ya puede hacerla en el ciclo actual, dividimos en dos casos:

- Si no se trata de la última palabra nos quedamos en este estado aumentando el contador de palabras (count_enable=1), indicando que la dirección no viene del MIPS (mux_origen=1) y escribiendo en MC donde corresponde (MC_WE0=not(via_2_rpl), MC_WE1=via_2_rpl).

- Si se trata de la última palabra, volvemos al estado de Inicio activando las mismas señales que cuando no es la última: indicando que es la última palabra (last_word=1) y escribiendo el tag (MC_tags_WE=1). Al volver a Inicio, aún no has terminado toda la transferencia ya que solo has traído el bloque a MC por lo que no se activa ready aún. Este se activará cuando se haga el acierto en lectura o el acierto en escritura (pasando al estado Escritura) como ya se ha explicado anteriormente.

Descomposición de la dirección

La memoria cache con la que trabajamos es asociativa con dos vías y, a su vez, cada vía tiene 4 bloques de 4 palabras. Teniendo esto en cuenta y comprobándolo con las separaciones del fuente MC_datos.vhd podemos verificar que la descomposición de la dirección para el direccionamiento de la MC es la siguiente:



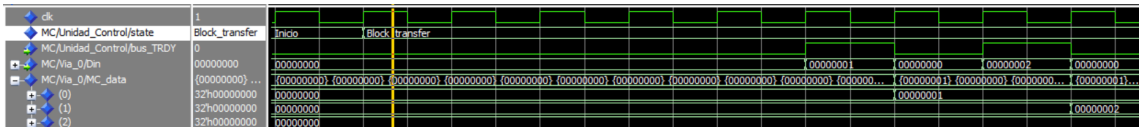
4 bits para identificar el byte dentro de bloque, 2 bits para identificar el conjunto y 26 bits para identificar el tag. Además, de los 4 bits para identificar el byte dentro de bloque, los bits de mayor peso identifican la palabra.

Análisis de las latencias en el bus

A partir de la simulación hemos obtenido los siguientes retardos, sin tener en cuenta los debidos al arbitraje.

CrB(MD): ciclos para leer un bloque de MD

CrB(MD) = L ciclos (para la primera palabra) + 3*R ciclos (para las palabras restantes)



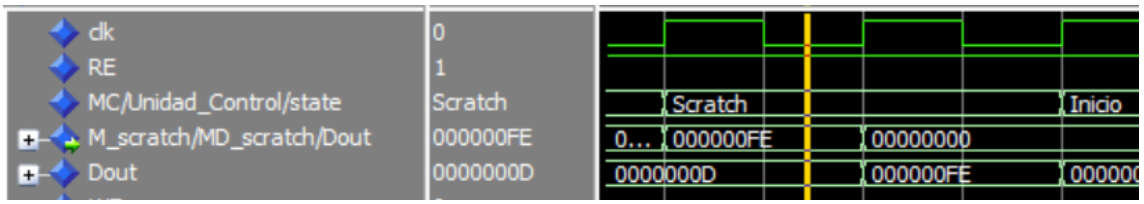
En el cronograma se puede observar la etapa Block_transfer (trae un bloque de MP a MC) y como para la primera palabra se tarda 5 ciclos (L=5) y para las demás se tardan 2 ciclos (R=2).

Por lo tanto, CrB(MD) = 5 + 3*2 = 11 ciclos

CwW(MD): ciclos para escribir una palabra en MD

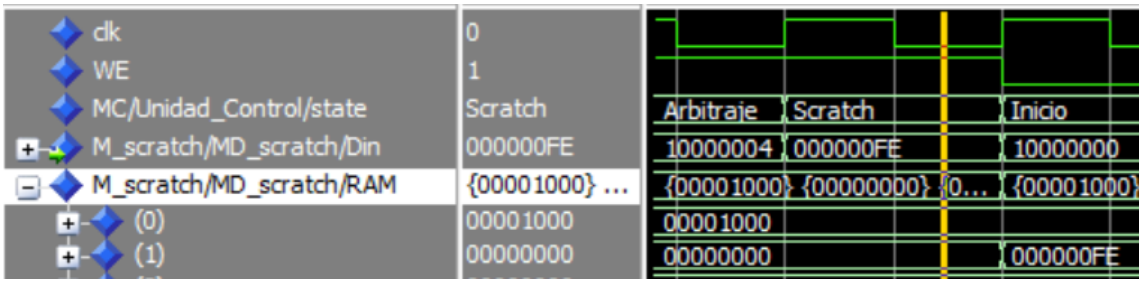
En el cronograma se puede observar la etapa Escritura y que se tarda en escribir en MD 5 ciclos.

CrW(MDscratch): ciclos para leer una palabra de MD Scratch



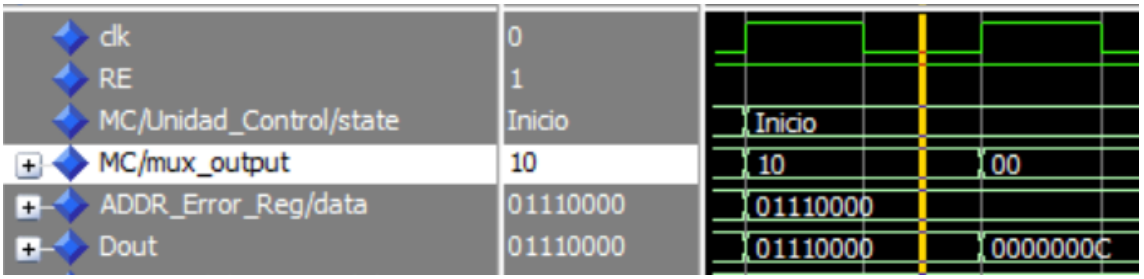
En el cronograma se puede observar la etapa Scratch, que se trata de una lectura y que se tarda en leer 2 ciclos.

CwW(MDscratch): ciclos para escribir una palabra en MD Scratch



En el cronograma se puede observar la etapa Scratch, que se trata de una escritura y que se tarda en escribir en la MD Scratch 1 ciclo.

CrW(RegInterno): ciclos para leer el registro interno de la MC



En el cronograma se puede observar la etapa Inicio, que se trata de una lectura del registro interno de la MC, el contenido de este registro y que la salida al MIPS es ese contenido. Toda esta lectura tarda 1 ciclo.

Expresión de cálculo de los ciclos efectivos

Los eventos que pueden ocurrir a nuestra MC junto con sus costes en ciclos son:

- **Acierto de lectura:** rh(MD) = 1 ciclo
- **Fallo de lectura:** rm(MD) = 1 ciclo + CrB(MD) = 1 + 11 = 12 ciclos
- **Acierto de escritura:** wh(MD) = 1 ciclo + CwW(MD) = 1 + 5 = 6 ciclos
- **Fallo de escritura:** wm(MD) = 1 ciclo + CrB(MD) + wh(MD) = 1 + 11 + 6 = 18 ciclos
- **Lectura de MD scratch:** rh(MDscratch) = 1 ciclo + CrW(MDscratch) = 1 + 2 = 3 ciclos
- **Escritura de MD scratch:** wh(MDscratch) = 1 ciclo + CwW(MDscratch) = 1 +1 = 2 ciclos
- **Lectura del registro interno:** rh(RegInterno) = CrW(RegInterno) = 1 ciclo

Teniendo esto en cuenta junto con los ciclos de arbitraje (1,5 ciclos) en aquellos casos que se podría necesitar el uso del bus, calculamos los ciclos efectivos mediante la siguiente fórmula:

$$Ceff = 1 + \frac{\#rm \cdot (CrB_{MD} + 1)}{\#refs} + \frac{\#wh \cdot CwW_{MD}}{\#refs} + \frac{\#wm \cdot CwW_{MD}}{\#refs} + \frac{\#rm_{Scratch} \cdot CrW_{Scratch}}{\#refs} + \frac{\#wm_{Scratch} \cdot CwW_{Scratch}}{\#refs}$$

$$= 1 + \frac{\#rm \cdot (1,5+11+1)}{\#refs} + \frac{\#wh \cdot (1,5+5)}{\#refs} + \frac{\#wm \cdot (1,5+5)}{\#refs} + \frac{\#rm_{Scratch} \cdot (1,5+2)}{\#refs} + \frac{\#wm_{Scratch} \cdot (1,5+1)}{\#refs}$$

$$= 1 + \frac{\#rm \cdot 13,5}{\#refs} + \frac{\#wh \cdot 6,5}{\#refs} + \frac{\#wm \cdot 6,5}{\#refs} + \frac{\#rm_{Scratch} \cdot 3,5}{\#refs} + \frac{\#wm_{Scratch} \cdot 2,5}{\#refs}$$

Aquellos componentes que llevan delante el # significa el número de ejecuciones lo correspondiente en el programa. Por ejemplo, #rm significa el número de fallos de escritura que se dan en el programa.

Programas de prueba

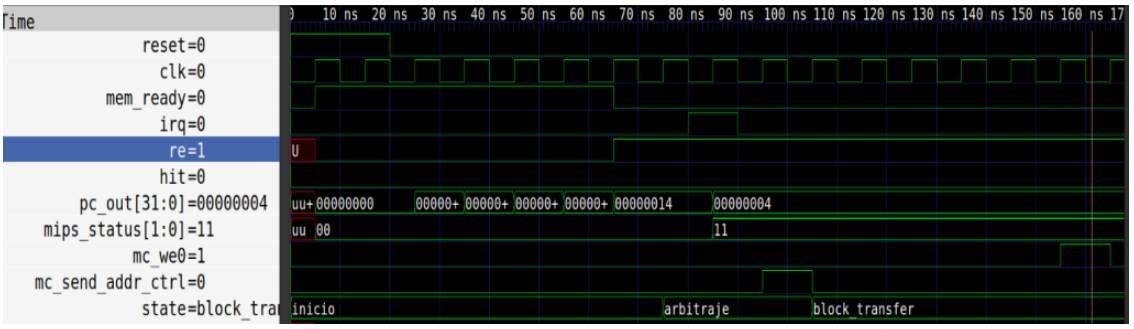
Para probar el correcto funcionamiento del sistema hemos creado dos programas de prueba. Uno para comprobar solo el correcto funcionamiento de los accesos a memoria y otro para comprobar que la parte del proyecto 1 (anticipaciones, detenciones, interrupciones) se siguen ejecutando bien también. Los programas se encuentran en el fichero pruebas.asm y para el primero, además, hemos creado un nuevo banco de pruebas para probar su funcionamiento que se ha llamado testbench_pruebas.vhd. Ambos ficheros se han adjuntado en la entrega.

El primer programa de prueba suma las componentes de un vector (componentes consecutivas en memoria) para comprobar que se realizan correctamente aciertos y fallos de lectura. Posteriormente, guarda el resultado de la suma anterior y también prueba lecturas y escrituras en MD Scratch. Por último, se hace una lectura de una dirección no alineada para comprobar la correcta lectura del registro interno. A continuación se explican los accesos a memoria.

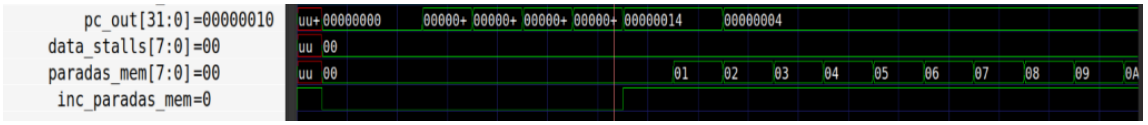
PC	Inst.	Operación	Tag	Set	Vía
0x10	lw r1, 144(r0)	rm (MD)	2	01	0
0x14	lw r2, 160(r0)	rm (MD)	2	10	0
0x18	lw r3, 176(r0)	rm (MD)	2	11	0
0x1C	lw r4, 192(r0)	rm (MD)	3	00	0
0x20	lw r5, 160(r0)	rh (MD)	2	10	0
0x24	lw r7, 160(r0)	rh (MD)	2	10	0
0x2C	lw r6, r5	rm (MD) 1ª iteración	0	00	1
		rh (MD) 2ª iteración	0	00	1
		rh (MD) 3ª iteración	0	00	1
		rh (MD) 4ª iteración	0	00	1
		rm (MD) 5ª iteración	0	01	1
		rh (MD) 6ª iteración	0	01	1
		rh (MD) 7ª iteración	0	01	1
		rh (MD) 8ª iteración	0	01	1
0x44	sw r2, 128(r0)	wm (MD)	2	00	0
0x48	lw r4, 240(r0)	rm (MD)	3	11	1
0x4C	lw r1, r4	rh (Scratch)	4194304	—	—
0x54	lw r3, 256(r0)	rm (MD)	4	00	0

0x58	sw r2, r3	wh (Scratch)	4194304	—	—
0x5C	lw r0, 5(r0)	dirección desalineada	—	—	—
0x64	lw r2, 224(r0)	rm (MD)	3	10	1
0x60	lw r1, r2	rh (RegInterno)	—	—	—

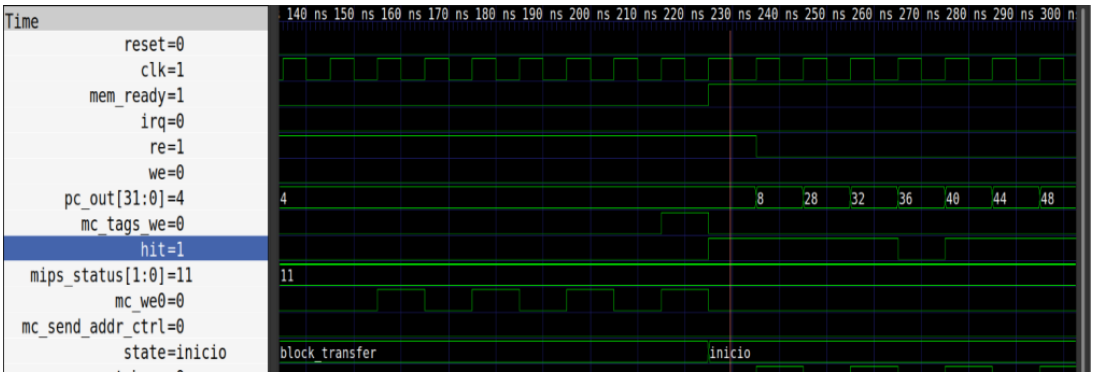
El segundo programa que hemos creado es para comprobar que, además de los accesos a memoria se realizan correctamente, la anticipación de operandos y las interrupciones se siguen ejecutando bien. Para ello, el programa ejecuta un load-uso.



Se puede comprobar que la instrucción 0xC: lw r3, 8(r0) se trata de un fallo en lectura (@0x14) y, durante el load-uso previo a ella, le llega una IRQ y se activa el modo interrupción mips_status y salta a la instrucción 0x4 en la cual se manda la dirección de lectura (mc_send_addr_ctrl) y se traen las palabras 1,2,3 y 4 al conjunto 0 de vía 0. Se observan todos los pasos de traer a MC el bloque entero por fallo de lectura (hit=0) de acuerdo a nuestra máquina de estados: Inicio, Arbitraje, Block_transfer.



En este extracto, se observa cómo, con el load-uso (add r2, r1, r2), durante la instrucción siguiente @0x14, el contador de riesgos de datos data_stalls se para, pero el contador de paradas_mem sigue aumentando ya que ante una parada en MEM, la causa principal es memoria, no el load-uso.



Aquí se ve cómo hay un acierto en lectura (hit=1), porque ya se ha realizado esta instrucción (@0x4) previamente en el conjunto 0 de la vía 0. Como es acierto de lectura, se mantiene en Inicio.

Speedup que aportan la MC y la MD Scratch

Una jerarquía como la que se utiliza en este proyecto es favorable sobre todo en aquellos casos en los que haya un gran número de aciertos de lectura, ya que el número de ciclos de acceso a memoria es 1.

Un ejemplo se ve en el primer programa de prueba diseñado ya que el bucle realiza lecturas, teniendo los dos fallos obligatorios mientras que los demás son aciertos y tardan solo 1 ciclo en procesarse. Si no hubiese MC tardarían bastantes más.

En cuanto a las escrituras, si se trata de un fallo se tardan más ciclos teniendo MC al traer 4 palabras mientras que si no hubiese MC solo se traería una. Respecto a las escrituras no hay mucha diferencia.

Cuantificación de horas dedicadas

	Carlota	Lilai
Estudio del enunciado y los fuentes	5h	3h
Modificaciones MIPS	1h	1h
Diagrama de estados de la unidad de control	4h	4h
Depuración, verificación y programas de prueba	7h	9h
Memoria	4h	6h

Conclusiones y Autoevaluación

Carlota: creo que he dedicado horas y esfuerzo y que al final ha dado sus frutos. He visto mi mejora en el aprendizaje con el tiempo aunque esta asignatura me parece de las más difíciles, pero muy interesante. Pienso que si tuviese que calificarme a mí misma, me pondría un 6.

Lilai: este proyecto me ha servido para afianzar conceptos de clase y de prácticas que parecían estar entendidos pero no era del todo así. Además, pese a no ser la parte que más me ha gustado de esta asignatura, haber podido probar y ver cómo se producían los fallos y aciertos al acceder a memoria me ha ayudado a aprender bastante. Yo creo que he cumplido con los objetivos de la asignatura tras finalizar este proyecto y todas las prácticas y problemas y creo que una nota adecuada sería un 6.