Proyecto MIPS – Parte 2

Memoria

Eduardo Gimeno Soriano

721615

Curso 2017-2018

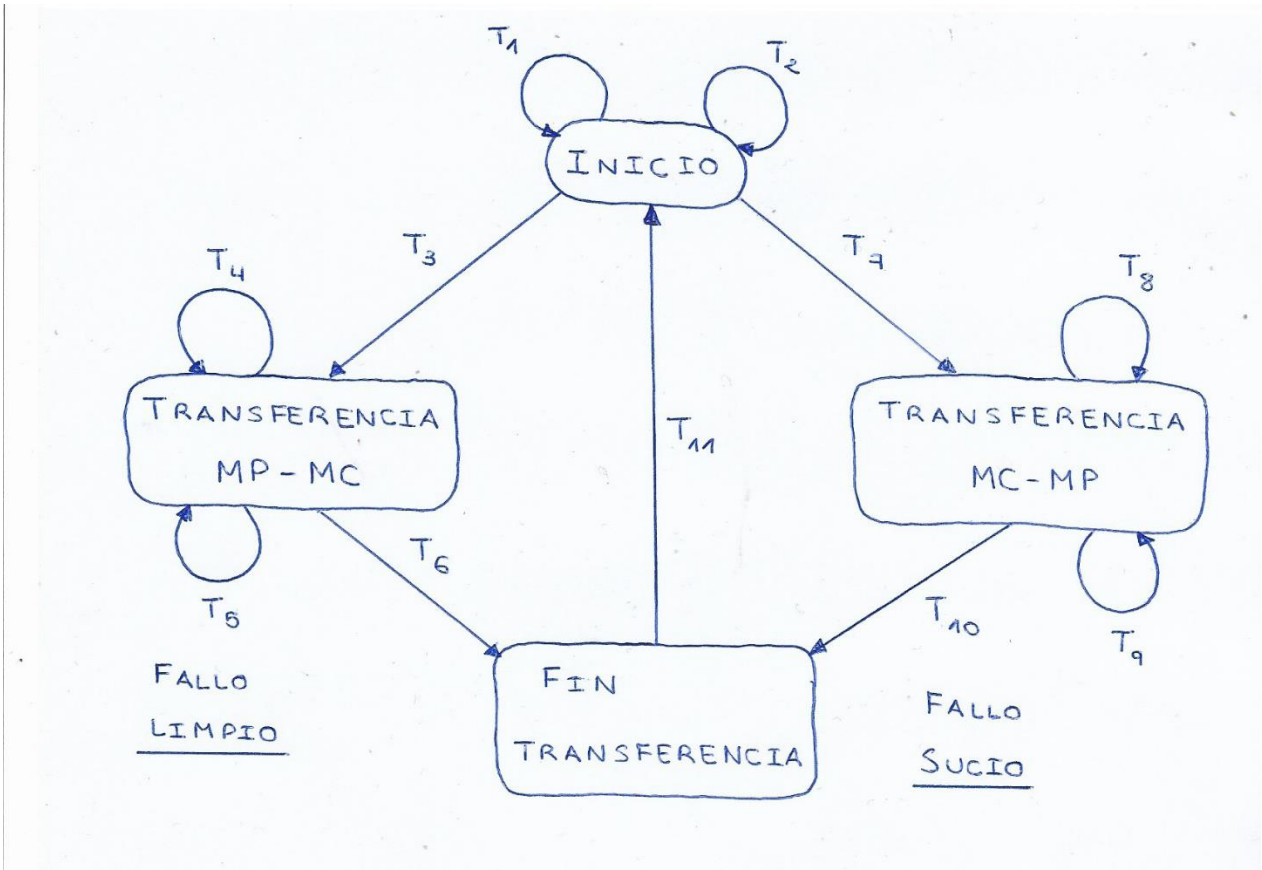
Índice

1. Explicación del diseño

El diagrama de estados para el controlador de la memoria cache consta de cuatro estados. Un estado inicial en el que se observa si hay acierto y en caso de fallo, si es limpio o sucio. Para cada tipo de fallo hay dos estados. En el primero se manda la dirección correspondiente a memoria principal y se activa la transferencia. En el segundo estado se procede a traer/llevar el bloque correspondiente a memoria o a enviar de nuevo la dirección y el tipo de operación si la memoria principal no puede realizarla en el ciclo correspondiente. Cuando se lleva/llega el último dato del bloque se actualiza el bloque sucio/se escribe el nuevo tag. Por último, se indica que ha finalizado la transferencia y se vuelve al estado inicial. En todos los estados correspondiente a fallo se indica que hay que parar.

En el MIPS se han añadido cuatro contadores, uno para cada tipo de parada, un nuevo riesgo en la unidad de detención y un mux para propagar 0 en vez de la señal RegWrite en la etapa MEM.

2. Diagrama de estados



- Estado Inicio: comprobar si hay acierto o fallo.
- Transferencia MP-MC: traer bloque nuevo de memoria principal a cache.
- Transferencia MC-MP: llevar bloque sucio a memoria principal de cache.
- Fin transferencia: se ha llevado/traído todo el bloque.

Acciones realizadas en cada transición:

- T1:
  Si RE = 1 && hit = 1
  MC_RE = 1
  mux_origen = 0
  ready = 1

  Acierto de lectura.

- T2:
  Si WE = 1 && hit = 1
  MC_WE = 1
  mux_origen = 0
  ready = 1

  Acierto de escritura.

- T3:
  Si hit = 0 && dirty_bit = 0
  Send_dirty = 0
  MC_send_addr = 1
  Frame = 1
  ready = 0

  Ha habido fallo y el bloque está limpio, se activa la transferencia, se envía la dirección y se avisa de que la operación no va a poder realizarse en un ciclo.

- T4:
  Si Bus_DevSel = 0 || bus_TRDY = 0
  Si Bus_DevSel = 0
  MC_send_addr = 1
  Send_dirty = 0
  Frame = 1
  bus_RE = 1
  ready = 0

  Si no se activa la señal Bus_DevSel se vuelve a enviar la dirección y para ambos casos se vuelve a enviar el tipo de operación.

- T5:
  Si Bus_DevSel = 1 && bus_TRDY = 1 && last_word = 0
  MC_WE = 1
  mux_origen = 1
  bus_RE = 1
  Frame = 1
  count_enable = 1
  ready = 0

  Escribir en cache datos 0, 1 y 2 del bloque.

- T6:
  Si bus_TRDY = 1 && last_word = 1
  MC_WE = 1
  mux_origen = 1
  bus_RE = 1
  Frame = 1
  ready = 0
  count_enable = 1
  MC_tags_WE = 1

  Escribir último dato del bloque y escribir el nuevo tag.

- T7:
  Si hit = 0 && dirty_bit = 1
  Send_dirty = 1

MC_send_addr = 1
Frame = 1
ready = 0

Ha habido fallo y el bloque está sucio, se activa la transferencia, se envía la dirección y se avisa de que la operación no va a poder realizarse en un ciclo.

- T8:
  Si Bus_DevSel = 0 || bus_TRDY = 0
  Si Bus_DevSel = 0
  MC_send_addr = 1
  Send_dirty = 0
  Frame = 1
  bus_WE = 1
  ready = 0

  Si no se activa la señal Bus_DevSel se vuelve a enviar la dirección y para ambos casos se vuelve a enviar el tipo de operación.

- T9:
  Si Bus_DevSel = 1 && bus_TRDY = 1 && last_word = 0
  MC_RE = 1
  MC_send_data = 1
  mux_origen = 1
  bus_WE = 1
  count_enable = 1
  Frame = 1
  ready = 0

  Llevar a memoria principal los datos 0, 1 y 2 del bloque.

- T10:
  Si bus_TRDY = 1 && last_word = 1
  MC_RE = 1
  MC_send_data = 1
  mux_origen = 1
  bus_WE = 1
  count_enable = 1
  Frame = 1
  ready = 0
  Update_dirty = 1
  raplace_block = 1

  Llevar el último dato del bloque y actualizar el bit sucio.

- T11:
  Frame = 0
  ready = 0

Fin de la transferencia.

3. Hardware añadido

En primer lugar, se ha sustituido la memoria de datos por el componente MD_mas_MC, tal y como se indicaba en el enunciado.

Se ha añadido un nuevo riesgo a la unidad de detención, el cual para el MIPS todas las etapas hasta MEM en caso de fallo de lectura/escritura, hasta que la cache proporciona el dato solicitado. Para poder solventar este nuevo riesgo la unidad de detención cuenta con una nueva entrada, Mem_ready, proporcionada por el nuevo componente MD_mas_MC y una nueva salida Parar_MEM.

Parar_MEM <= Mem_ready;

Con esta nueva señal se ha modificado la señal load de los bancos de registros de las etapas ID, EX y MEM y del registro de pc, quedando de la siguiente manera.

load_ID <= parar_ID and parar_EX and parar_MEM
load_EX <= parar_EX and parar_MEM
load_MEM <= parar_MEM
load_PC <= parar_ID and parar_EX and parar_MEM

Finalmente se han añadido cuatro contadores, para contabilizar los ciclos para tipo de parada.

Contador para paradas por riesgo de datos.
ID_cn_en <= (not parar_ID) and (parar_EX) and (parar_MEM);
ID_counter: counter port map (clk => clk, reset => reset, count_enable =>
ID_cn_en, load => '0', D_in => "00000000", count => paradas_datos);

Contador para paradas por riesgo de FP.
EX_cn_en <= (not parar_EX) and (parar_MEM);
EX_counter: counter port map (clk => clk, reset => reset, count_enable =>
EX_cn_en, load => '0', D_in => "00000000", count => paradas_FP);

Contador para paradas por riesgo de memoria.
MEM_cn_en <= not parar_MEM;
MEM_counter: counter port map (clk => clk, reset => reset, count_enable =>
MEM_cn_en, load => '0', D_in => "00000000", count => paradas_memoria);

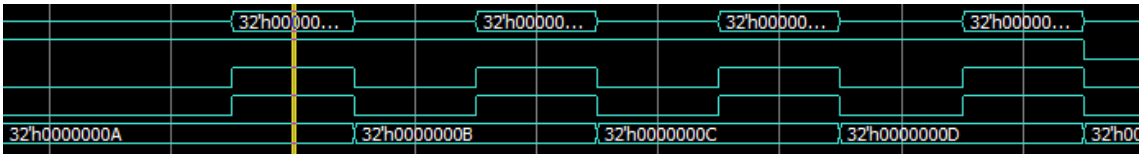
Contador para paradas por riesgo de control.
B_cn_en <= kill_IF and parar_ID and parar_EX and parar_MEM;

```
B_counter: counter port map ( clk => clk, reset => reset, count_enable =>
B_cn_en, load => '0', D_in => "00000000", count => paradas_control);
```

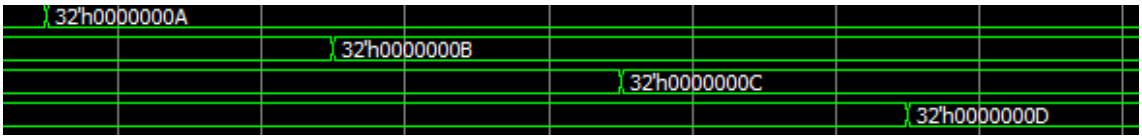
4. Pruebas

Pruebas locales de la memoria cache

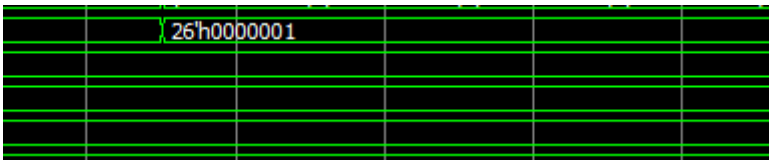
1. Fallo limpio de lectura en el conjunto 0 (@64)



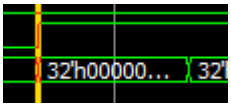
Memoria principal proporcionando los datos correspondientes a las direcciones 60, 64, 68 y 72 por fallo en cache.



Escritura de los datos en el conjunto 0 de la memoria cache.

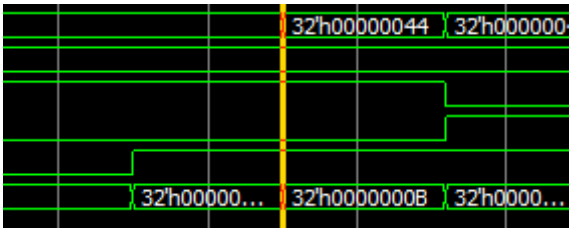


Escritura del tag en la memoria cache.



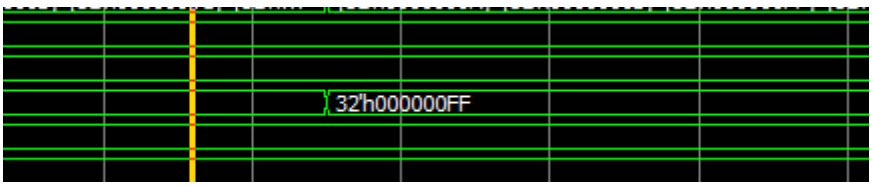
Memoria cache proporcionando el dato correspondiente.

2. Acierto de lectura en el conjunto 0 (@68)



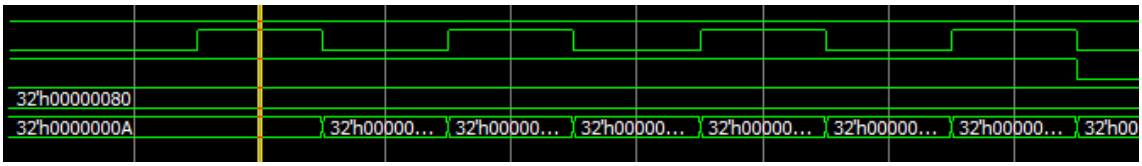
Memoria cache proporcionando el dato correspondiente.

3. Acierto de escritura en el conjunto 0 (@72)

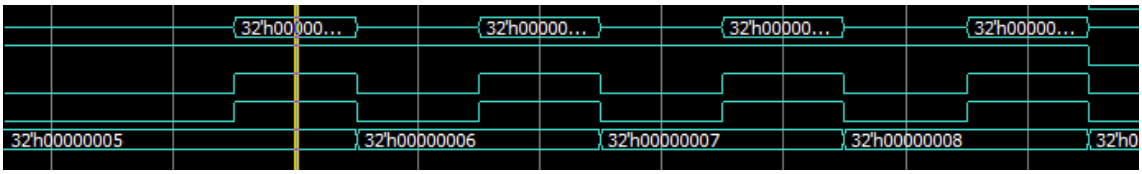


Escritura del dato en memoria cache.

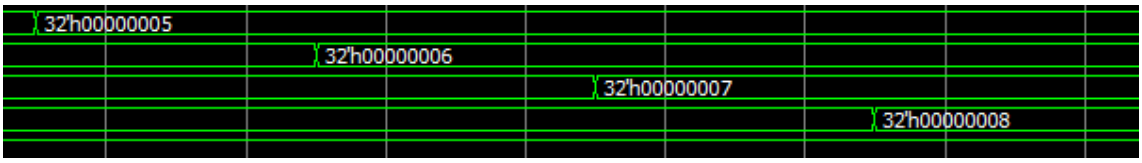
4. Fallo sucio de lectura en el conjunto 0 (@128)



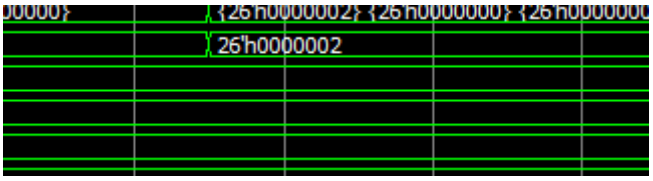
Envío de los datos desde cache a memoria principal por fallo sucio.



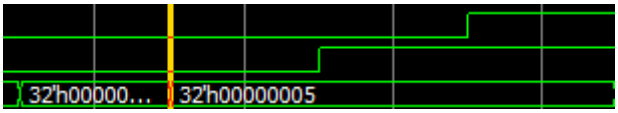
Memoria principal proporcionando los datos correspondientes a las direcciones 128, 132, 136 y 140.



Escritura de los datos en el conjunto 0 de la memoria cache.

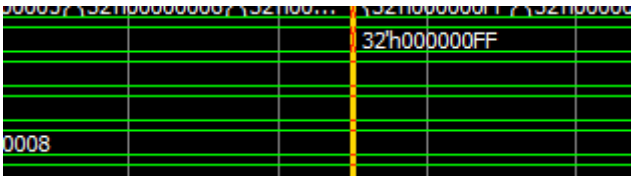


Escritura del tag en la memoria cache.



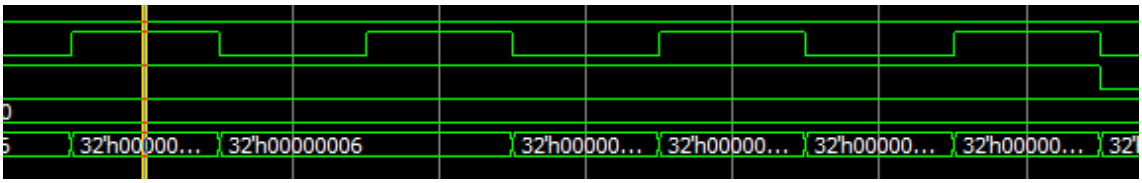
Memoria cache proporcionando el dato correspondiente.

5. Acierto de escritura en el conjunto 0 (@128)

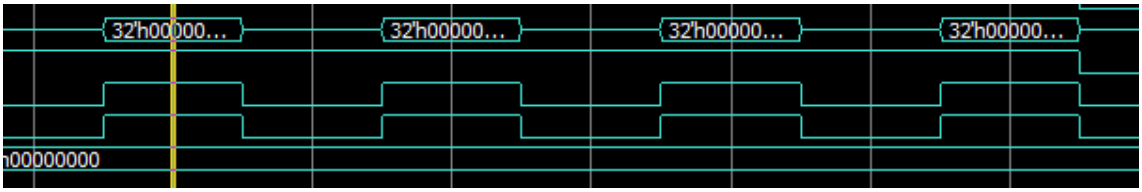


Escritura del dato en memoria cache.

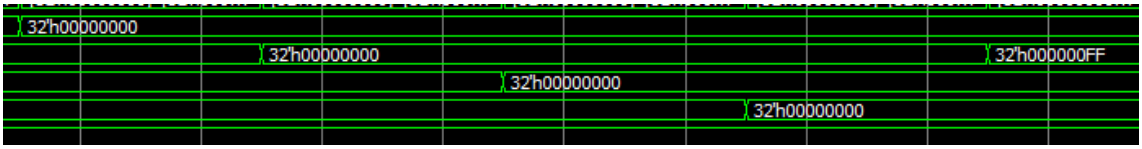
6. Fallo sucio de escritura en el conjunto 0 (@196)



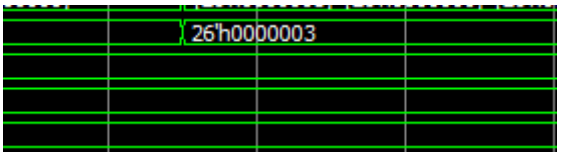
Envío de los datos desde cache a memoria principal por fallo sucio.



Memoria principal proporcionando los datos correspondientes a las direcciones 192, 196, 200 y 204.

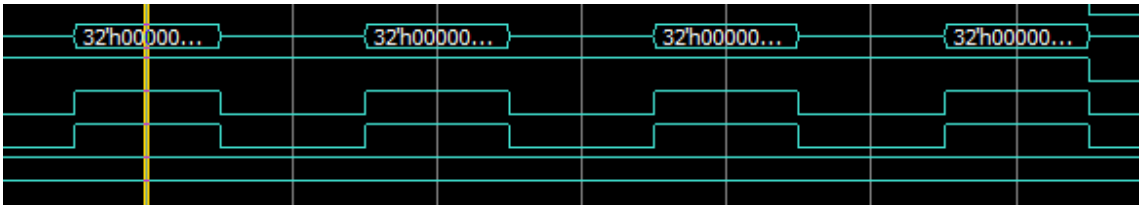


Escritura de los datos en el conjunto 0 de la memoria cache y del nuevo dato.

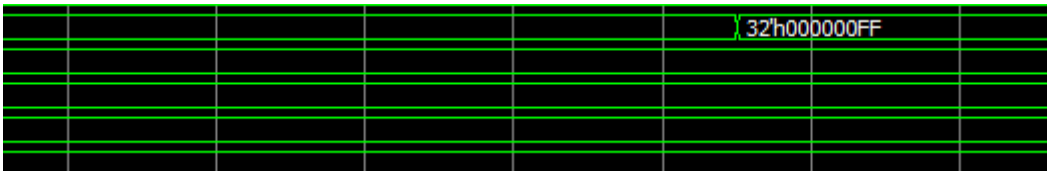


Escritura del tag en la memoria cache.

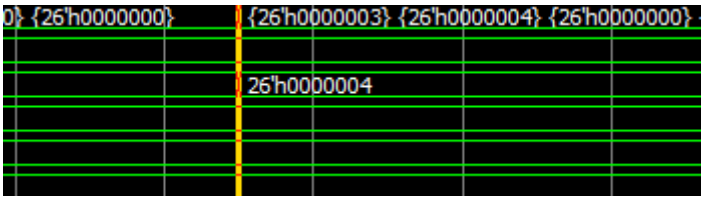
7. Fallo limpio de escritura en el conjunto 1 (@272)



Memoria principal proporcionando los datos correspondientes a las direcciones 272, 276, 280 y 284.

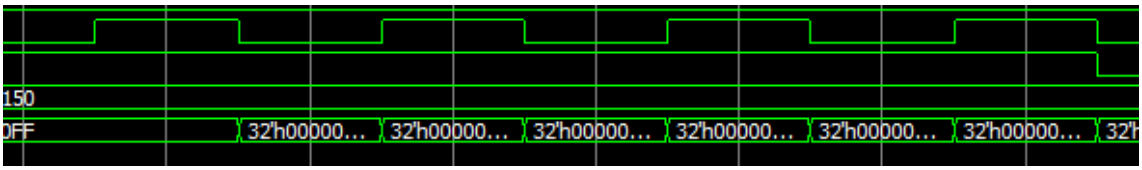


Escritura de los datos en el conjunto 1 de la memoria cache y del nuevo dato.

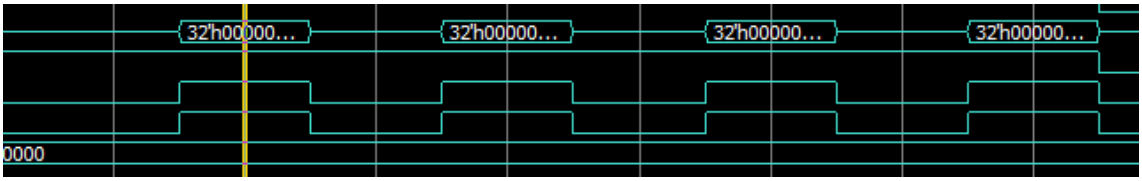


Escritura del tag en la memoria cache.

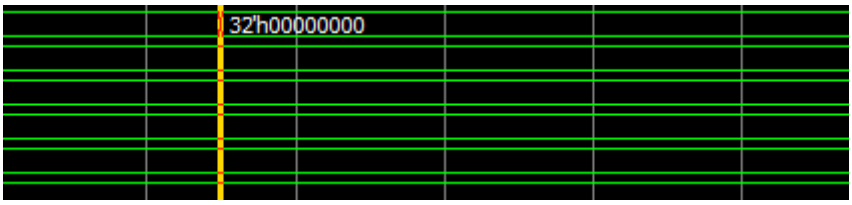
8. Fallo sucio de lectura en el conjunto 1 (@336)



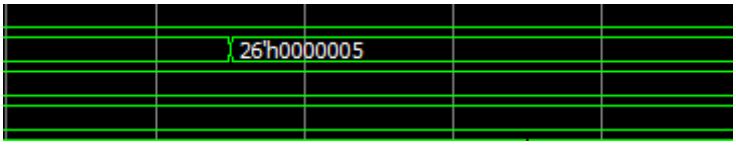
Envío de los datos desde cache a memoria principal por fallo sucio.



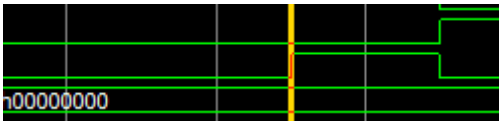
Memoria principal proporcionando los datos correspondientes a las direcciones 336, 340, 344, 348.



Escritura de los datos en el conjunto 1 de la memoria cache.

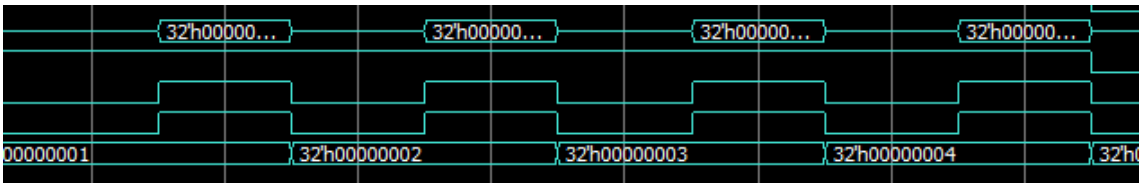


Escritura del tag en la memoria cache.

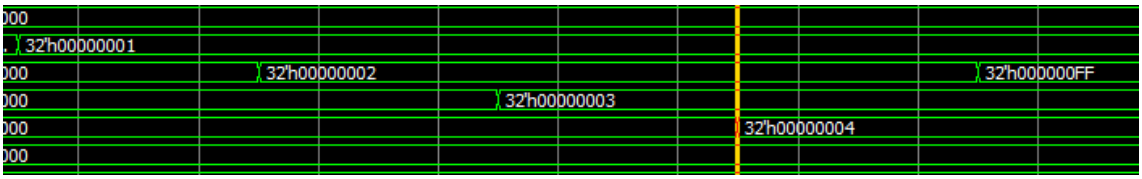


Memoria cache proporcionando el dato correspondiente.

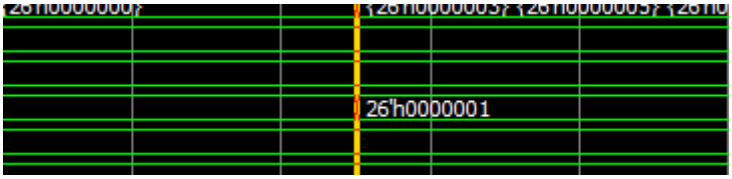
9. Fallo limpio de escritura en el conjunto 2 (@100)



Memoria principal proporcionando los datos correspondientes a las direcciones 96, 100, 104, 108 por fallo en cache.

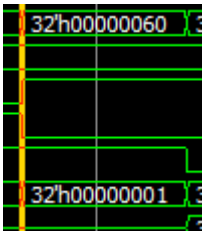


Escritura de los datos en el conjunto 2 de la memoria cache.



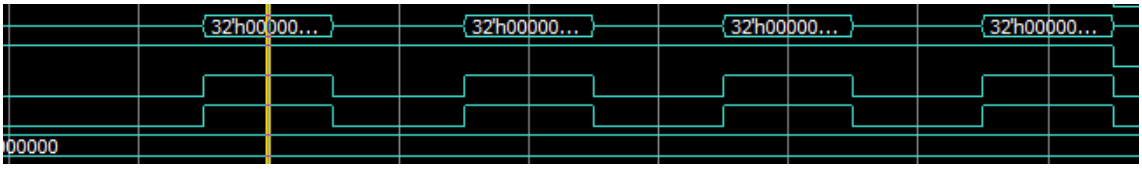
Escritura del tag en la memoria cache.

10. Acierto de lectura en el conjunto 2 (@96)

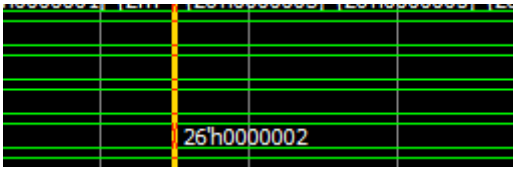


Memoria cache proporcionando el dato correspondiente.

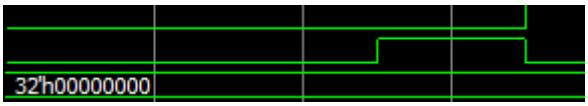
11. Fallo limpio de escritura en el conjunto 3 (@176)



Memoria principal proporcionando los datos correspondientes a las direcciones 176, 180, 184, 188 por fallo en cache.

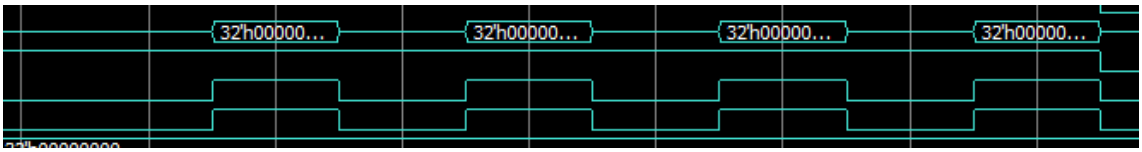


Escritura del tag en la memoria cache.

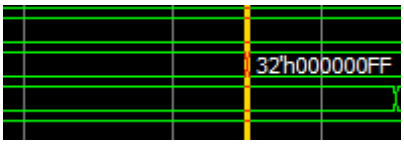


Memoria cache proporcionando el dato correspondiente.

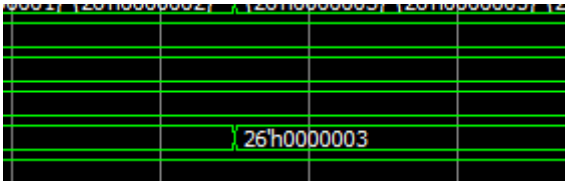
12. Fallo limpio de escritura en el conjunto 3 (@244)



Memoria principal proporcionando los datos correspondientes a las direcciones 240, 244, 248, 252 por fallo en cache.

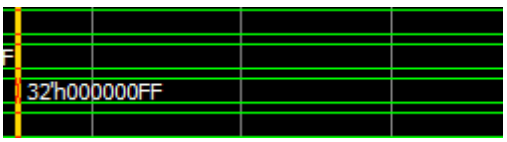


Escritura del dato en el conjunto 3 de la memoria cache.



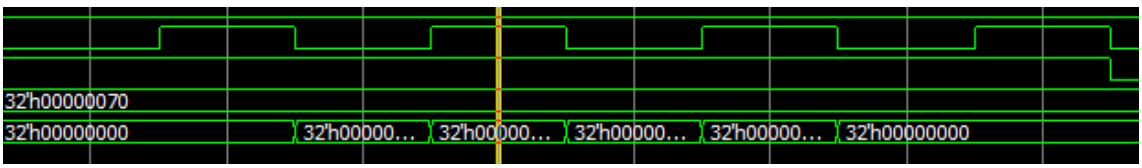
Escritura del tag en la memoria cache.

13. Acierto de escritura en el conjunto 3 (@248)

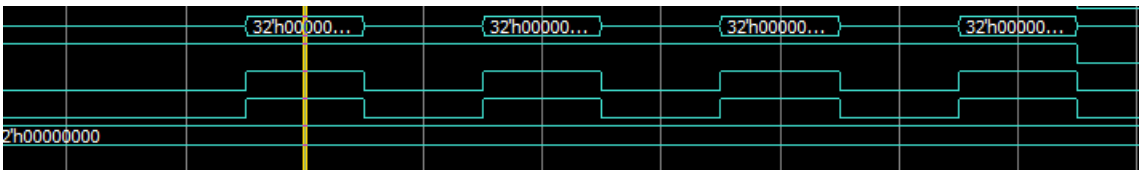


Escritura del dato en memoria cache.

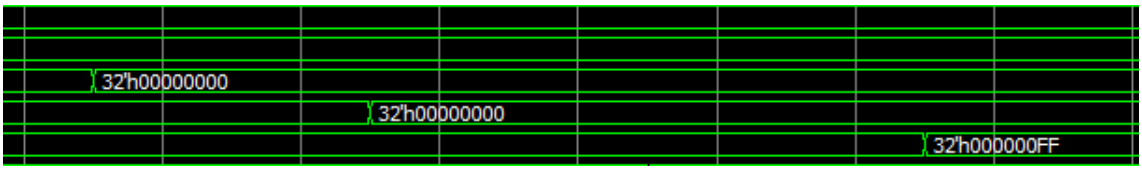
14. Fallo sucio de escritura en el conjunto 3 (@124)



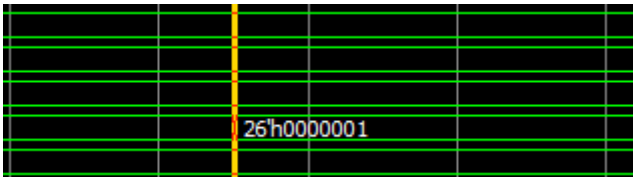
Envío de los datos desde cache a memoria principal por fallo sucio.



Memoria principal proporcionando los datos correspondientes a las direcciones 112, 116, 120, 124.



Escritura de los datos en el conjunto 3 de la memoria cache.



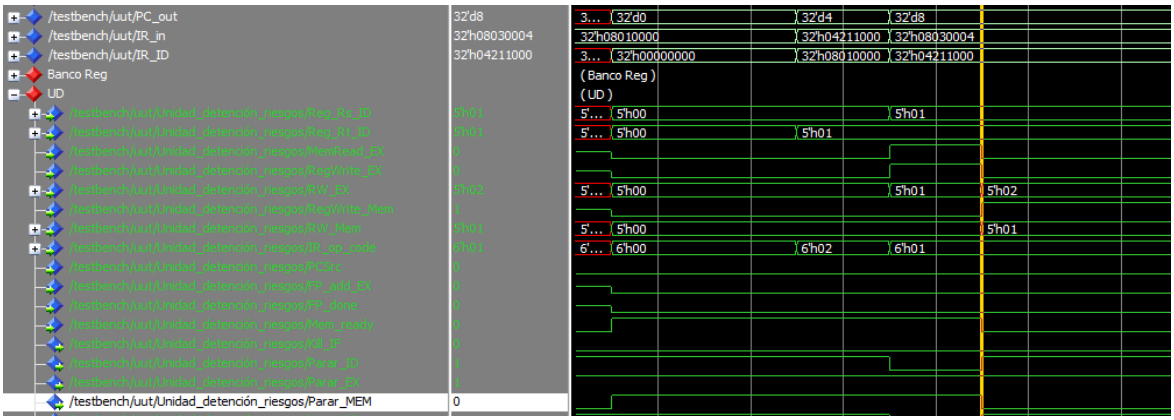
Escritura del tag en la memoria cache.

Pruebas MIPS

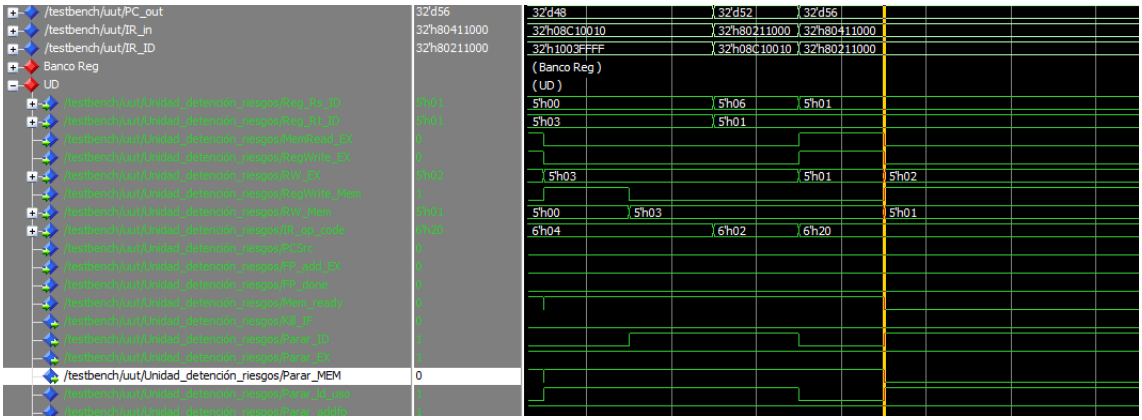
1.

```
lw r1, 0(r0)
add r1, r1, r2
lw r3, 4(r0)
add r2, r2, r4
add r3, r4, r5
sw r5, 8(r0)
beq r0, r5, inm
add r0, r0, r0
lw r0, 0(r1)
nop
lw r0, 12(r6)
beq r3, r0, LO
lw r1, 16(r6)
addfp r1, r1, r2
addfp r1, r2, r2
sw r2, 20(r6)
beq r0, r0, fin
```

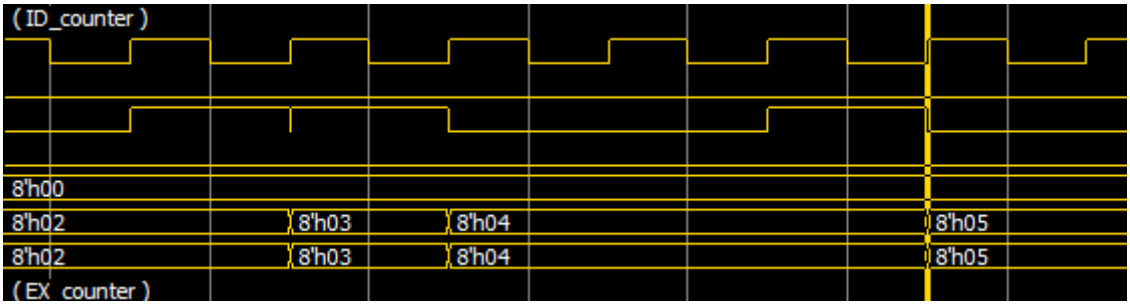
Este es el programa de prueba de la parte 1, además de los riesgos ya presentes, ahora hay dos paradas más por memoria.
Las cinco primeras instrucciones de acceso a memoria acceden al conjunto 0 de la cache, la primera de ellas falla por estar esta vacía (fallo limpio), el resto son aciertos de lectura (3) y escritura (1). Las dos últimas acceden al conjunto 1, la primera de ellas falla (fallo limpio) y la segunda es acierto de escritura.



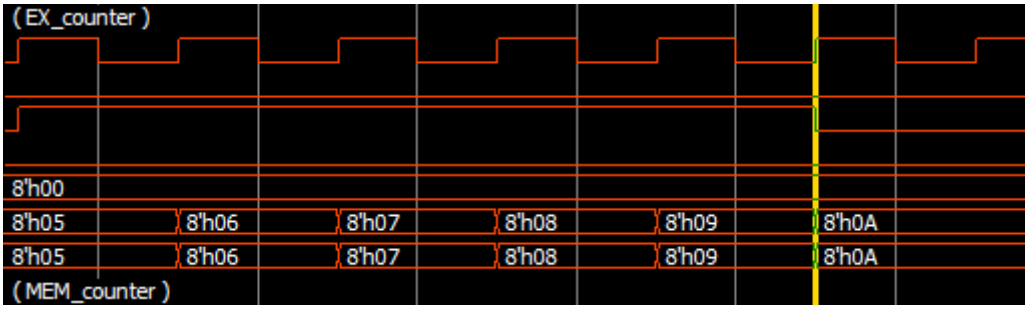
Primera parada por fallo limpio de lectura en el conjunto 0 provocado por el primer lw.



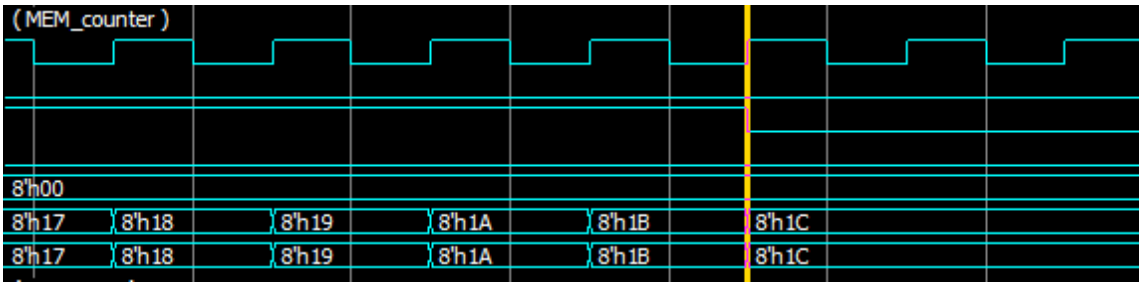
Segunda parada por fallo limpio de lectura en el conjunto 1 provocado por el quinto lw.



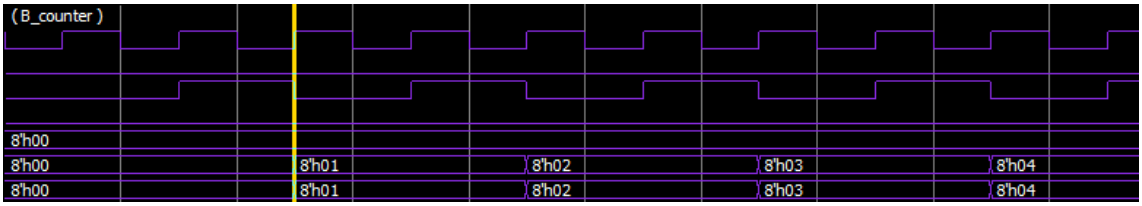
Ciclos de parada por riesgo de datos.



Ciclos de parada por addfp.



Ciclos de parada por memoria.



Ciclos con riesgo de control (seguiría indefinidamente por bucle final).

2.

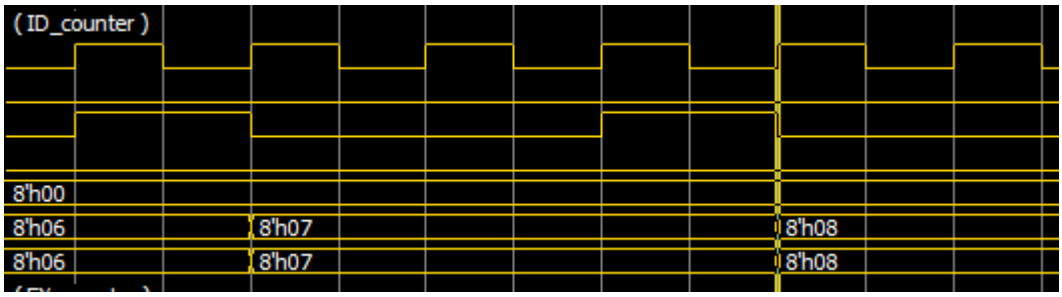
in lw r1, 0(r0)
lw r2, 4(r0)
add r2, r4, r4
lw r5, 16(r0)
add r5, r5, r6
sw r6, 128(r0)
beq r4, r1, fin
beq r0, r0, in

fin lw r1, 0(r0)

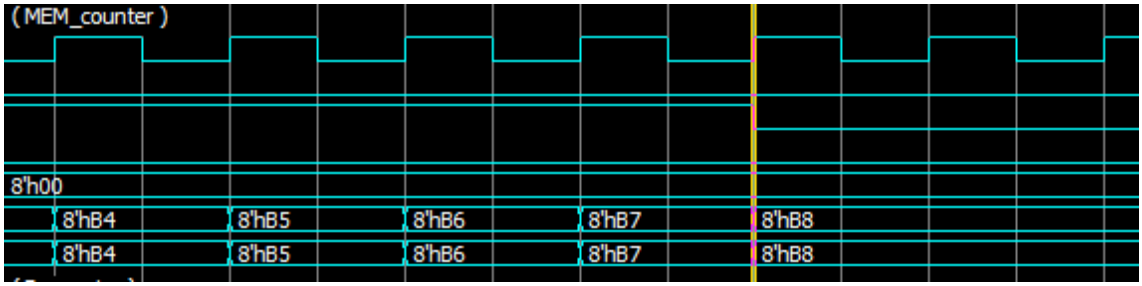
Este programa realiza cuatro iteraciones con los siguientes resultados en los accesos de memoria cache.

Iteración	1	2	3	4
lw r1, 0(r0)	Fallo limpio	Fallo sucio	Fallo sucio	Fallo sucio
lw r2, 4(r0)	Acierto	Acierto	Acierto	Acierto
lw r5, 16(r0)	Fallo limpio	Fallo limpio	Fallo limpio	Fallo limpio
sw r6, 128(r0)	Fallo limpio	Fallo limpio	Fallo limpio	Fallo limpio
lw r1, 0(r0)				Fallo sucio

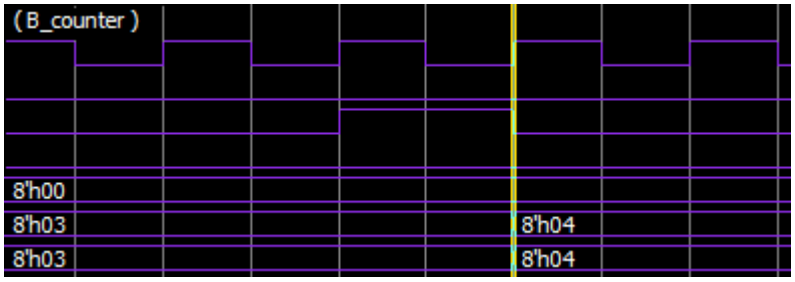
Además, en este programa existen dos riesgos de datos (lw-uso) y riesgos de control provocados por los saltos.



Ciclos de parada por riesgo de datos.



Ciclos de parada por memoria.



Ciclos con riesgo de control.

5. Conclusiones

Una vez vistos todos los riesgos que pueden afectar a la ejecución de un programa en un MIPS, los fallos en memoria cache son los que más penalizan en cuanto a ciclos, sobre todo los fallos sucios, pero si en un programa se acceden a direcciones consecutivas se reduce el tiempo de ejecución.

En cuanto a la elaboración de esta parte del proyecto, para el estudio de los fuentes y comprensión de lo que se pedía se han dedicado 2 horas aproximadamente, pero esta parte requirió repasar algunos conceptos de teoría, por tanto, habría que añadirle 1:30 más. En cuanto al diseño inicial de la unidad de control, como se pasó por varias versiones hasta llegar a la definitiva, se han dedicado 4 horas aproximadamente. Para la realización de pruebas la mayor parte del tiempo se ha dedicado a realizar pruebas locales sobre la cache, unas 11 horas, y 3 horas ha probar con programas completos, sobre todo para realizar pruebas sobre los contadores. Finalmente, a la memoria se le han dedicado unas 4 horas.

	Estimado (horas)	Real (horas)
Estudio de los fuentes	2	3,5
Diseño inicial de la UC	2	4
Depuración y ajustes	16	14
Memoria	3	3,5

¿Crees que has cumplido los objetivos de la asignatura?
He entendido el funcionamiento del MIPS y de los distintos componentes que lo conforman, así como las distintas métricas de rendimiento utilizadas a lo largo de la asignatura y el funcionamiento de las memorias cache y sus políticas de reemplazo por fallo.

¿Qué nota te pondrías si te tuvieses que calificar a ti mismo?
Me calificaría en torno al 6 – 6.5, ya que en la primera parte del proyecto tuve que realizar correcciones porque no entendí bien algunos aspectos de primeras y porque en esta parte no he podido realizar ninguna de las mejoras optativas propuestas.