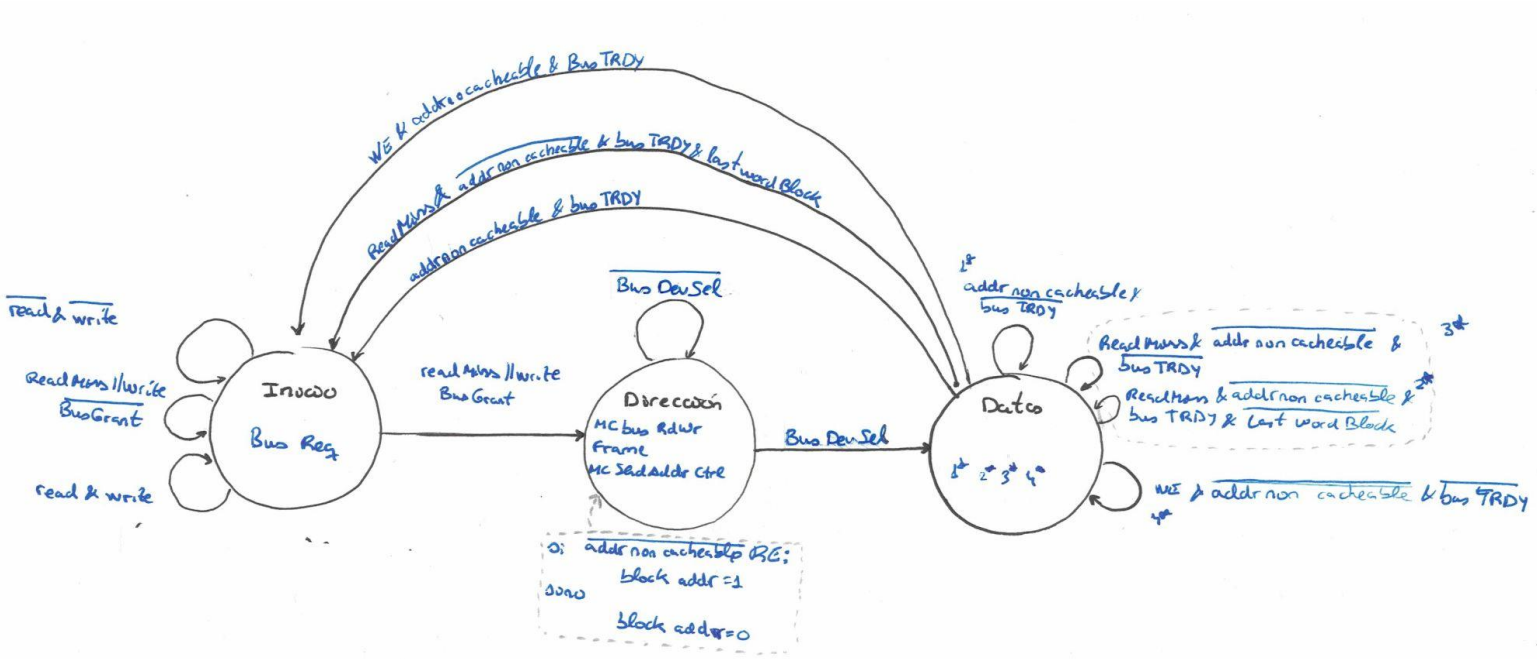
Proyecto 2

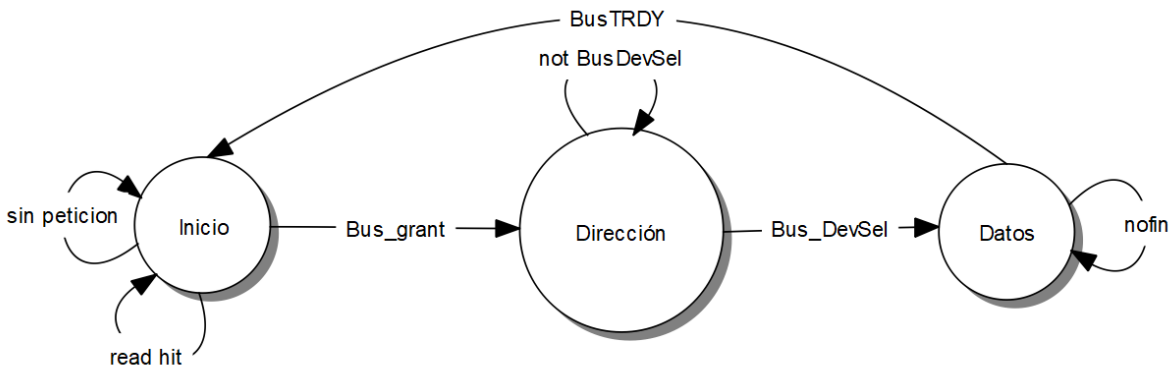
MEMORIA CACHÉ EN EL MIPS

Héctor Toral
Javier Pizarro

ÍNDICE

Diagrama de estados de la unidad de control

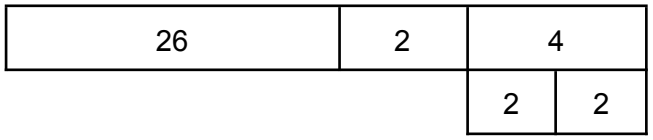




Este segundo autómata es uno más simplificado que intenta representar las partes características del anterior, de todas formas creemos que se ve más claro en el propio código vhdl que se encuentra más abajo.

Descomposición de la dirección

Los datos han sido obtenidos mediante la imagen proporcionada “mc_datos”



Tag: 26 bits
Conjunto: 2 bits
Palabras: 2 bits
Byte dentro de palabra: 2 bits

Descripción algorítmica de la jerarquía de memoria.

El autómata tiene 3 estados muy claramente divididos.
En el primero estado iteramos mientras tengamos la caché lista, en caso de que necesitemos traer datos o escribir solicitamos el uso del bus y si se nos concede avanzamos al siguiente estado.
En el segundo estado nos encargamos de lo referente a las direcciones y iteramos hasta que tengamos el bus para poder usarlo.
En el tercer estado se itera hasta tener el bus listo para enviar/recibir datos y hasta haber finalizado la lectura/escritura, dentro del estado se activan las señales según sea lectura con miss o escritura con hit o escritura con miss.

```
case state is
	when Inicio =>-- Estado Inicio
		if (RE = '0' and WE = '0') then -- si no piden nada no hacemos nada
			next_state <= Inicio;
			ready <= '1';
		elsif (RE = '1' and  hit = '1') then -- si piden y es acierto de lectura
mandamos el dato
			next_state <= Inicio;
			ready <= '1';
		elsif ((RE = '1' and hit = '0') or WE = '1') then
			Bus_req <= '1';
			if (Bus_grant = '1') then
				next_state <= Direccion;
				inc_w <= WE;
				inc_m <= not hit;
				MC_WE0 <= hit0 and WE;
				MC_WE1 <= hit1 and WE;
			else
				next_state <= Inicio;
			end if;
		end if;

	when Direccion =>
		-- salidas del estado
		MC_bus_Rd_Wr <= WE;
		Frame <= '1';
		MC_send_addr_ctrl <= '1';

		if (addr_non_cacheable = '0' and RE = '1') then block_addr <= '1';
		else block_addr <= '0';
		end if;

		-- transiciones
		if (Bus_DevSel = '0') then
			next_state <= Direccion;
		elsif (Bus_DevSel = '1') then
			next_state <= Datos;
		end if;
```

```
        when Datos =>
                Frame <= '1';

                --No cacheable
                if (addr_non_cacheable = '1') then
                        if (bus_TRDY = '1') then
                                next_state <= Inicio;
                        else
                                next_state <= Datos;
                        end if;

                        MC_send_data <= WE;
                        ready <= bus_TRDY;
                        mux_output <= bus_TRDY;
                        last_word <= bus_TRDY;
                end if;

                --Read miss cacheable
                if (RE = '1' and hit = '0' and addr_non_cacheable = '0') then
                        count_enable <= bus_TRDY;
                        mux_origen <= bus_TRDY;
                        if (bus_TRDY = '1') then
                                if (last_word_block = '0') then
                                        next_state <= Datos;
                                else
                                        next_state <= Inicio;
                                end if;

                                last_word <= last_word_block;
                                MC_WE0 <= not via_2_rpl;
                                MC_WE1 <= via_2_rpl;
                                MC_tags_WE <= last_word_block;
                        else
                                next_state <= Datos;
                        end if;

                --Write cacheable
                elsif (WE = '1' and addr_non_cacheable = '0') then
                        if (bus_TRDY = '1') then
                                next_state <= Inicio;
                        else
                                next_state <= Datos;
                        end if;

                        MC_send_data <= '1';
                        ready <= bus_TRDY;
                        last_word <= bus_TRDY;
                        if (bus_TRDY = '1' and hit = '1') then
                                MC_WE0 <= hit0;
                                MC_WE1 <= hit1;
                        end if;
                end if;
end case;
```

Ciclos efectivos

Cef = Ta + Tf x Penalización

Ta = 2
TF = 20/41 fallos/ accesos
Penalización = 18

2+20/41 *18= 9,780

Programas de prueba

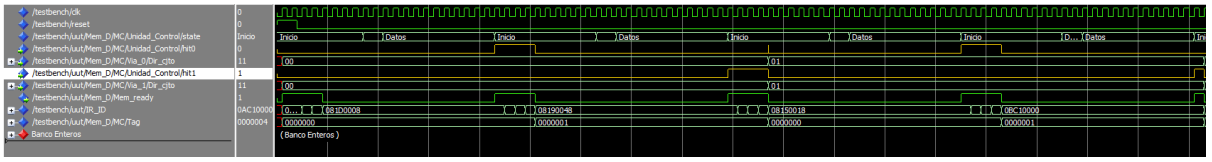
**Lo que pone a la derecha en el caso de cargar registros (al principio) es lo que se supone que debe hacer la dirección que va a ser cargada en el registro cuando se emplee, no lo que ocurre al cargar el registro.
Para realizar las pruebas primero cargamos todas las direcciones a emplear en los registros comenzando por el 31, una vez cargadas las direcciones y en la situación inicial en la que todos los conjuntos apuntan a la via0.
Una vez cargadas todas las direcciones procedemos a hacer loads con todas de manera que llenamos todos los conjuntos en ambas vías con misses en todos los casos, una vez tenemos toda la caché cargada con los datos probamos a hacer un load en cada conjunto el cual da siempre hit, alternando para cada conjunto la vía.
Posteriormente para probar los stores primero hacemos todos los posibles stores en todas las vías de todos los conjuntos que dan hit y luego hacemos otra tanda de stores que van a dar miss para todas las vías y todos los conjuntos.
Adicionalmente a esta prueba y la prueba del test bench específico de la MC, hemos empleado el código de test1 del proyecto anterior(TestMIPS) y su correcto funcionamiento.

```
#cargamos los registros con las direcciones de memoria a probar cargar direcciones para misses lectura
####################################################
#conjunto 0 via 0 miss
lw        r31  #0  r0;#direccion conjunto 0 miss via 0
lw        r30  #4  r0;#direccion conjunto 1 miss via 0
lw        r29  #8  r0;#direccion conjunto 2 miss via 0
lw        r28  #12  r0;#direccion conjunto 3 miss via 0
#conjunto 0 via 1 miss
lw        r27  #64  r0;#direccion conjunto 0 miss via 1
lw        r26  #68  r0;#direccion conjunto 1 miss via 1
lw        r25  #72 r0;#direccion conjunto 2 miss via 1
lw        r24  #76  r0;#direccion conjunto 3 miss via 1
#conjunto 1 via 0 miss
#cargar direcciones para hits de lectura
lw        r23  #16  r0;#direccion conjunto 0 hit via 0
lw        r22  #20  r0;#direccion conjunto 1 hit via 1
lw        r21  #24  r0;#direccion conjunto 2 hit via 0
lw        r20  #28  r0;#direccion conjunto 3 hit via 1
#conjunto 1 via 1 miss(para tener ambos conjuntos apuntando a la via 0)
lw        r19  #80  r0;
```

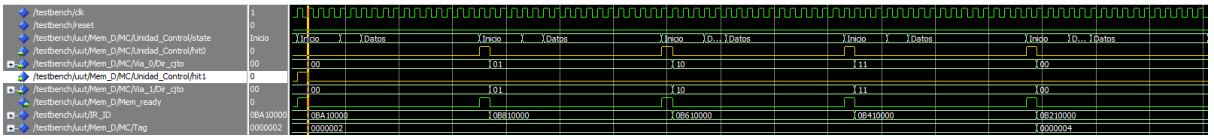
```
#####################################################
#PRUEBAS lectura
lw r1 #0 r31;#conjunto 0 miss via 0
lw r1 #0 r30;#conjunto 1 miss via 0
lw r1 #0 r29;#conjunto 2 miss via 0
lw r1 #0 r28;#conjunto 3 miss via 0
lw r1 #0 r27;#conjunto 0 miss via 1
lw r1 #0 r26;#conjunto 1 miss via 1
lw r1 #0 r25;#conjunto 2 miss via 1
lw r1 #0 r24;#conjunto 3 miss via 1
lw r1 #0 r23;#conjunto 0 hit via 0
lw r1 #0 r22;#conjunto 1 hit via 1
lw r1 #0 r21;#conjunto 2 hit via 0
lw r1 #0 r20;#conjunto 3 hit via 1
#####################################################
#PRUEBAS escritura hits
sw r0 #0 r31;#conjunto 0 hit via 0
sw r0 #0 r30;#conjunto 1 hit via 0
sw r0 #0 r29;#conjunto 2 hit via 0
sw r0 #0 r28;#conjunto 3 hit via 0
sw r0 #0 r27;#conjunto 0 hit via 1
sw r0 #0 r26;#conjunto 1 hit via 1
sw r0 #0 r25;#conjunto 2 hit via 1
sw r0 #0 r24;#conjunto 3 hit via 1
#PRUEBAS escritura misses
sw r0 #64 r31;#conjunto 0 miss via 0
sw r0 #64 r30;#conjunto 1 miss via 0
sw r0 #64 r29;#conjunto 2 miss via 0
sw r0 #64 r28;#conjunto 3 miss via 0
sw r0 #64 r27;#conjunto 0 miss via 1
sw r0 #64 r26;#conjunto 1 miss via 1
sw r0 #64 r25;#conjunto 2 miss via 1
sw r0 #64 r24;#conjunto 3 miss via 1
#####################################################
```

Ejemplo de código

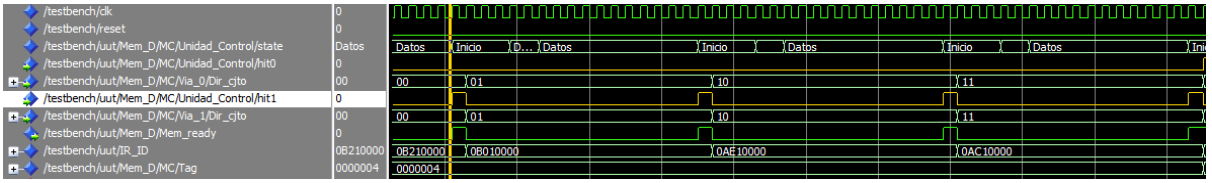
**Las señales amarillas son los hits 0 y 1, la línea amarilla indica la última posición de la gráfica anterior.
Carga de los registros:
Read en via 0 conj 0
Read en via 1 conj 0
Read en via 0 conj 1
Read en via 1 conj 1



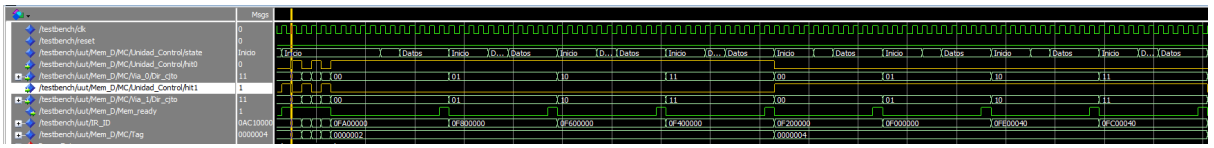
Carga de los conjuntos 0,1,2,3, en vía 0



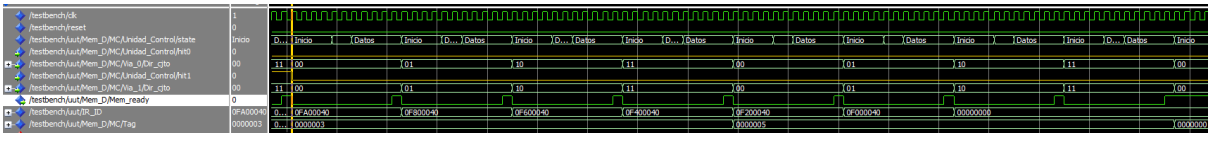
Carga de los conjuntos 0,1,2,3 en via 1



Read hit conj 0 via 0
Read hit conj 1 via 1
Read hit conj 2 via 0
Read hit conj 3 via 1
Write hit conjuntos 0,1,2,3 via 0
Write hit conjuntos 0,1,2,3 via 1



Write miss conjuntos 0,1,2,3 via 0
Write miss conjuntos 0,1,2,3 via 1
Fin



SpeedUp = 170 / 129 = 1,31 , lo que implica que el rendimiento con esta jerarquía de memoria incrementa el rendimiento en un 31% respecto al Mips con la antigua jerarquía de memoria.

Gestión del trabajo

Fecha	Tiempo	Autor/es	Tarea
20/05	2h	Javier Pizarro Héctor Toral	Lectura de enunciado y código base
20/05	2h	Javier Pizarro, Héctor Toral	Plantear primer autómata a papel
20/05	1h	Héctor Toral Javier Pizarro	Añadir caché al mips Añadir paradas en mips y contador
21/05	5h	Javier Pizarro, Héctor Toral	Programar autómata y entender del todo las señales
22/05	4h	Javier Pizarro, Héctor Toral	Pasar los tests de la caché
23/05	3h	Javier Pizarro, Héctor Toral	Pasar los tests del mips con caché
24/05	1h	Javier Pizarro	Elaborar modelo tests propios
25/05	2h	Héctor Toral	Comprobar resultados tests
26/05	3h	Javier Pizarro, Héctor Toral	Comprobar más tests
27/05	3h	Javier Pizarro, Héctor Toral	Elaborar documentación
28/05	4h	Javier Pizarro, Héctor Toral	Elaborar documentación y repasar tests/cuentas

Autoevaluación

Francisco Javier Pizarro:
¿Crees que has cumplido los objetivos de la asignatura?
A lo largo del cuatrimestre he tratado de realizar todos los problemas, faltando únicamente a las 2 últimas sesiones por distintos motivos, corrigiendo en bastantes ocasiones(5-7 no recuerdo el número exacto) los problemas dado el reducido tamaño del grupo de mañanas de 10-11, adicionalmente he sido capaz de llevar todas las prácticas de la asignatura al día sin dificultad alguna, si bien es cierto que por faltar a un par de sesiones de 2 h de los viernes, por distintos motivos, esto me hizo perder un poco el hilo de la asignatura pero a pesar de ello pude recuperarlo y realizar satisfactoriamente ambos proyectos.

¿Qué nota te pondrías si te tuvieses que calificar a ti mismo?
7.75
Principalmente porque por motivos de tiempo no hemos podido realizar los apartados optativos de este proyecto y porque en el primero tuvimos que realizar una pequeña subsanación además de mis 4 faltas de asistencia y las notas de los controles.

Héctor Toral:
¿Crees que has cumplido los objetivos de la asignatura?
En mi opinión, creo sinceramente que he cumplido los objetivos de la asignatura, he asistido a todas las sesiones de problemas salvo a la última por problemas de tiempo con otro proyecto participando en todas ellas activamente y saliendo voluntariamente varias veces a corregir los ejercicios propuestos, además he realizado todas las prácticas satisfactoriamente en los plazos establecidos, Es cierto que no he podido acudir a todas las sesiones teóricas debido a problemas de salud pero siempre he conseguido volver a reengancharme gracias a los vídeos subidos en moodle como a mis compañeros.
También he sido capaz de realizar ambos proyectos y de ayudar a varios compañeros a entender algunos de los problemas.

¿Qué nota te pondrías si te tuvieras que calificar a ti mismo?
8