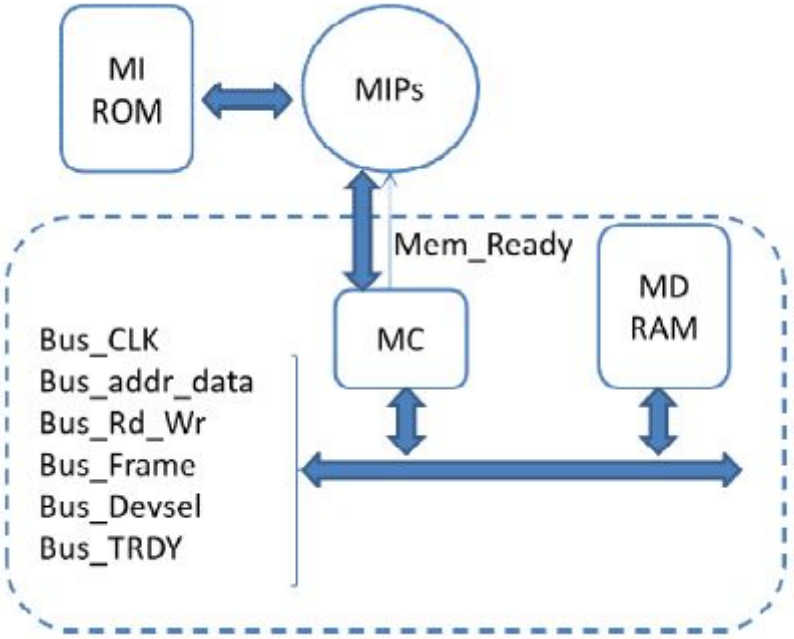
Proyecto Optativo 2:
MIPS segmentado

Arquitectura y Organización de Computadores II



Enrique Ruiz Flores 766685
Alba Vallés Esteban 760099

ÍNDICE

Explicación breve de nuestro diseño

Con el procesador MIPS de la práctica anterior, se sustituye la memoria principal por un componente que aúna la memoria caché y la memoria principal. Cuando el MIPS realiza una instrucción que accede a memoria(LW o SW) se envía a la caché y se espera a que la señal Mem_ready se active.

Si hay riesgo en memoria se hacen paradas en todas las etapas, impidiendo que se carguen los bancos de registros. Para compatibilizar este tipo de riesgo con los anteriores, la salida de avanza_ID y load_PC dependerá de la misma señal de la que dependían antes además de Mem_ready y los diferentes bancos solo avanzarán cuando ambas señales estén activas (comprobado mediante una puerta AND).

La Unidad de Control de la memoria caché se encarga de gestionar las instrucciones Load y Store que le llegan del procesador tal como se indica en la figura 1.

En el primer estado (*Start*) la caché se encuentra preparada para realizar la operación que le pide mientras que se mantengan a cero las señales que indican que se ha solicitado realizar una lectura o escritura en memoria llamadas RE y WE (read y write enable). También se mantiene en este estado en el caso de que se solicite realizar una lectura de una palabra ya cargada en caché.

El segundo estado (*Wait*) se encarga de esperar a que la memoria principal se encuentre lista para la transmisión de datos, para ello se le envía la dirección , en caso de una lectura también la dirección del bloque a traer a caché (en caso de escritura no ya que es *Write-around)* , también el tipo de operación requerida y por último se activa la señal Bus_Frame (bus ocupado).

Cuando la memoria principal está lista para recibir o transmitir datos se pasa al tercer estado (*Transfer*) donde se transfieren tantos datos como sean necesarios es decir ,1 palabra en el caso de escritura ya sea acierto o fallo, o 3 de las 4 palabras en el caso de fallo en lectura.

Si hay un fallo en lectura, tras el estado *Transfer* se pasa al estado 4 (*LastWord*) en el cual se recibe la última palabra del bloque y se avisa en el ciclo siguiente de que ya está listo. Si es una escritura se enviará la palabra a memoria principal y en el ciclo siguiente se pondrá ready a 1 para avisar de que la memoria ha terminado de atender la petición y está lista para atender la siguiente.

Se han añadido además registros contadores en la memoria caché para llevar la cuenta de los fallos en lectura (read miss), aciertos en escritura (write hit) y los fallos en escritura (write miss).
Estos contadores se incrementarán en el primer ciclo, es decir cuando se evalúa si es una lectura o escritura (RE o WE) y si es un acierto (Hit).

También en el MIPS contabilizamos los ciclos totales, los ciclos de parada por riesgo de datos, riesgo de control y riesgo en memoria, y el número de lecturas y escrituras que se hacen a memoria. El contador de ciclos se incrementa con la señal de reloj, los contadores de paradas se incrementarán cuando la señal de riesgo esté activa y los de lectura y escritura en memoria cuando se detecte una instrucción LW o ST (señales Mem_Write y Mem_Read) y que no haya parada en memoria, es decir, cuando la señal Mem_ready sea 1.

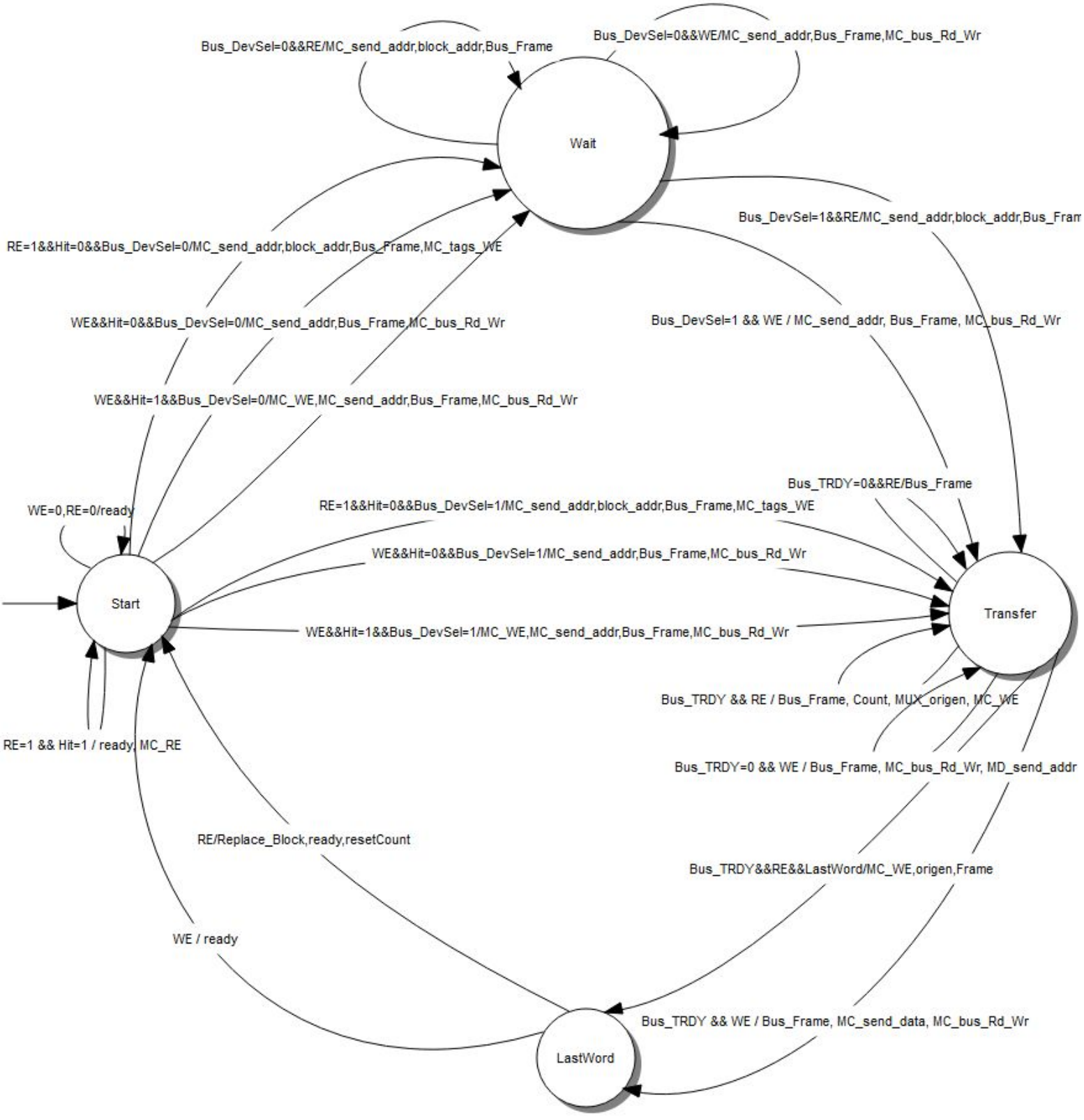


Figura 1 . Diagrama de estados de la Unidad de control de la memoria caché

Descripción algorítmica de la MC

Especificación	Ciclos
look-up(@x); if WE && Hit {Mc+x; }	1
send_addr(@x); wait(Bus_DevSel);	1
if WE {	
Mp(wW,@x); waitfor Mp;	CwW
}	
elsif miss(@x) && RE {Mp(rB,@x); waitfor Mp; Mc+X;}	CrB + 1
MRU(@x);	0
switch(proc_r/w)	
caseproc_r:ret x';	0
caseproc_w:ret;	0

CrB(Mp) = 4 + 2*3 = 10
CwW(Mp) = 5

Cef = 1+ 1 + mr * (CrB+1) + w*CwW = 1 + 1 + mr * 11 + w * 5
mr: tasa de fallo en lectura = Miss reads/instrucciones totales
w: tasa de escrituras = Total writes (wh + wm)/instrucciones totales
Tarda lo mismo un acierto que un fallo en escritura de la caché porque en caso de acierto se escribe en el mismo ciclo de búsqueda y luego se realiza la transmisión de la palabra a Memoria principal por ser Write Through

Bancos de pruebas

Código	*Comentarios*
Prueba1:	
LW R7,0(R0)	Read Miss conjunto 1
LW R8,16(R0)	Read Miss conjunto 2
LW R9,32(R0)	Read Miss conjunto 3
LW R10,48(R0)	Read Miss conjunto 4
LA R1,16(R0)	R1 = 16
LA R6,32(R1)	R6 = 48
ST R1,64(R0)	Write Miss
ST R1,80(R0)	Write Miss
ST R1,96(R0)	Write Miss
ST R1,112(R0)	Write Miss
LW R11,0(R0)	Read hit conjunto 1
LW R12,16(R0)	Read hit conjunto 2
LW R13,32(R0)	Read hit conjunto 3
LW R14,48(R0)	Read hit conjunto 4
buc ST R1,0(R31)	Write Hit en conj 1-3
ADD R31,R31,R1	R31 = R31 + 16
BNE R31,R6,buc	if R6 != R31 salto a
ST R1,0(R31)	Write Hit conjunto 4
Prueba 2:	
LA R5,0(R0)	R5 = 0
LA R6,1(R0)	R6 = 1
LA R7,4(R0)	R7 = 4
LA R0, 4(R5)	R0=4
LA R8, 10(R1)	R8= 16
buc ST R0, 4(R1)	@1=4 fallo en escritura
LA R1, 4(R0)	R1=8
ST R1, 0(R1)	@2=8 fallo en escritura
LW R2, 0(R0)	R2=4(@1) fallo en lectura
LW R3, 0(R1)	R3=8(@2) acierto en lectura
ADD R4,R3,R2	R4= 4+8 =12
ST R4,0(R4)	@3= 12
ADD R0,R8,R0	R0= 4+16 = 20
ADD R5,R6,R5	R5++
BNE R5,R7, buc	itera 4 veces(en cada iteración traerá un bloque distinto) (FFF6)

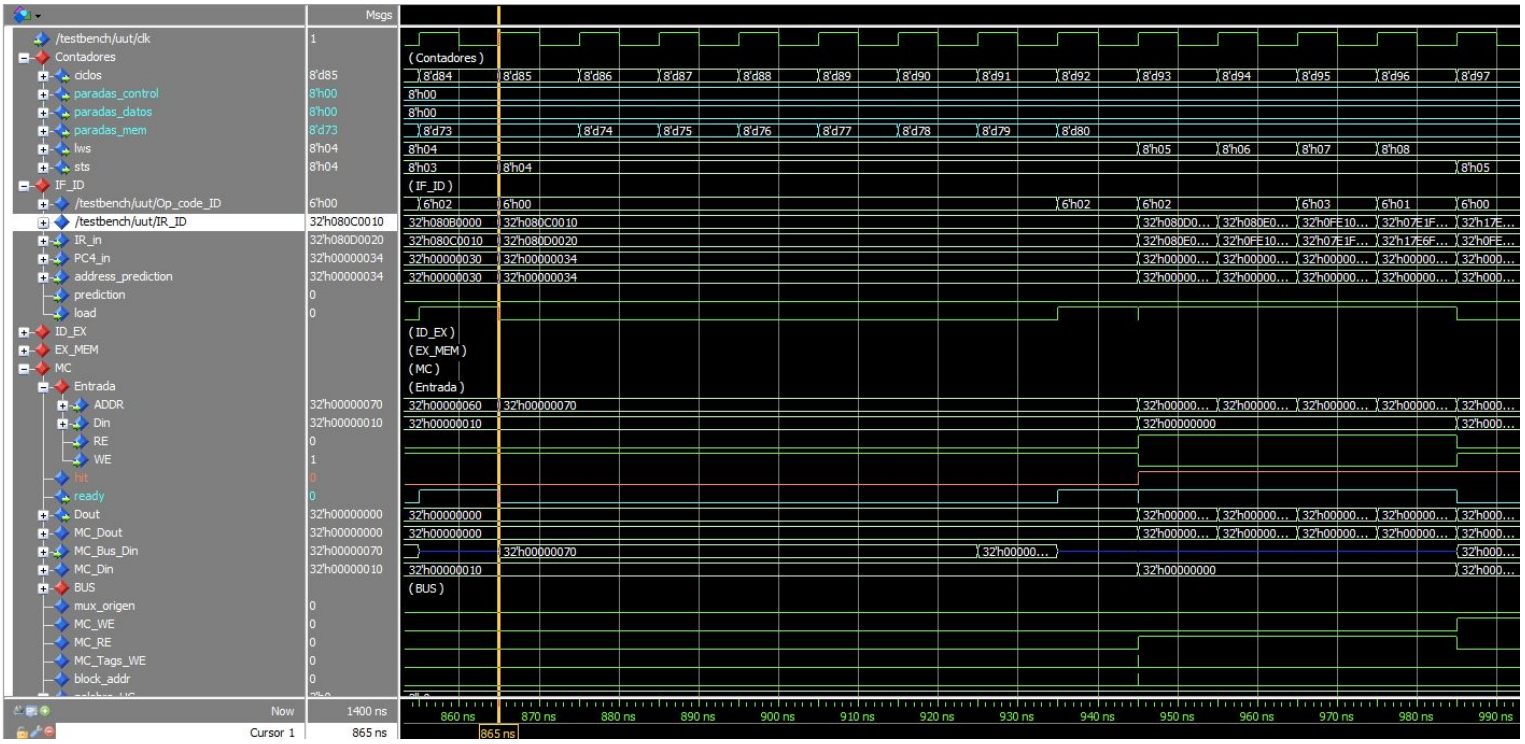


Figura 2. Banco de pruebas 1 - Momento en el que termina un SW que ha sido write miss y comienzan 4 LWs de bloques en caché que producen hit

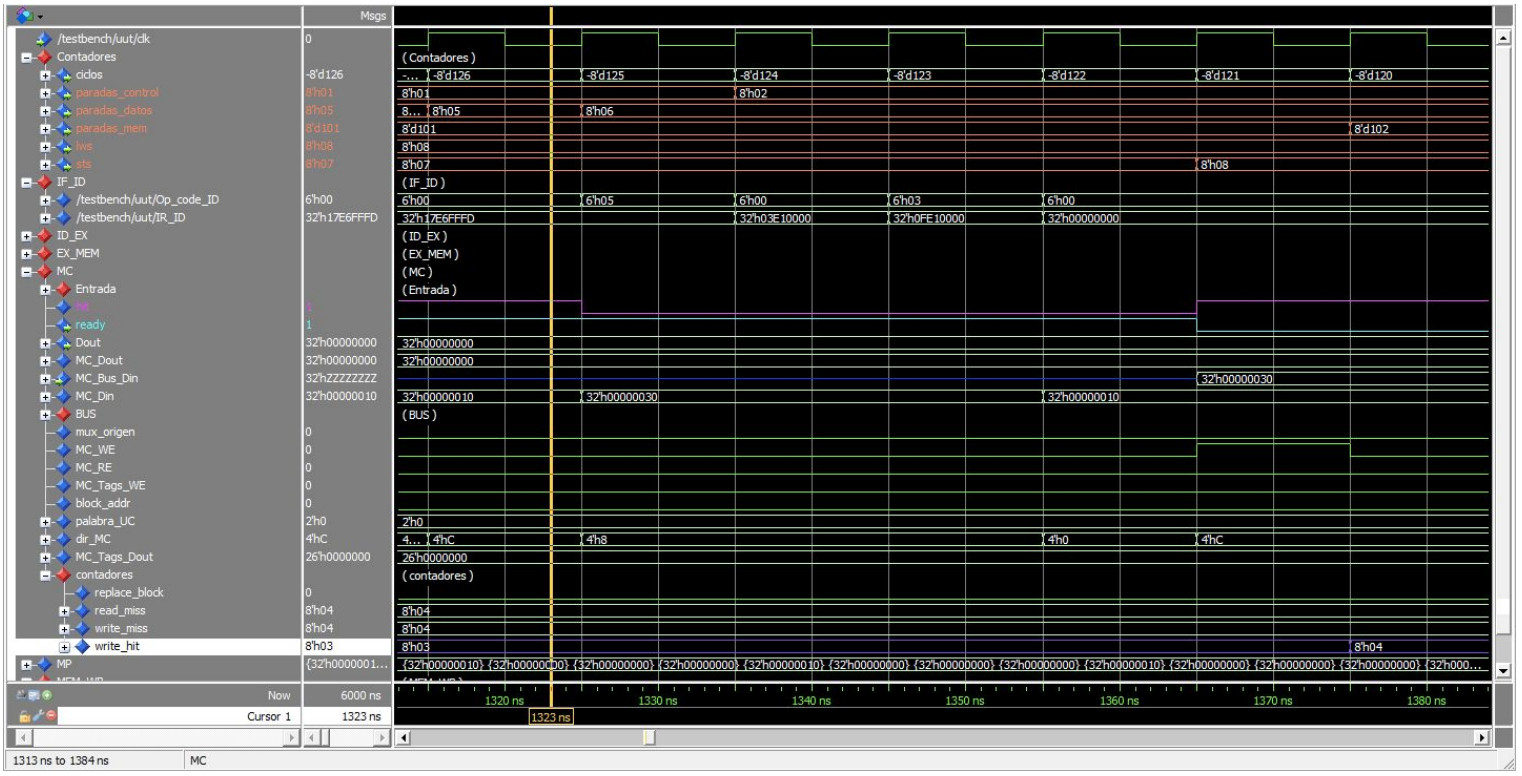


Figura 3. Banco de pruebas 1 - Write hit en el conjunto 4

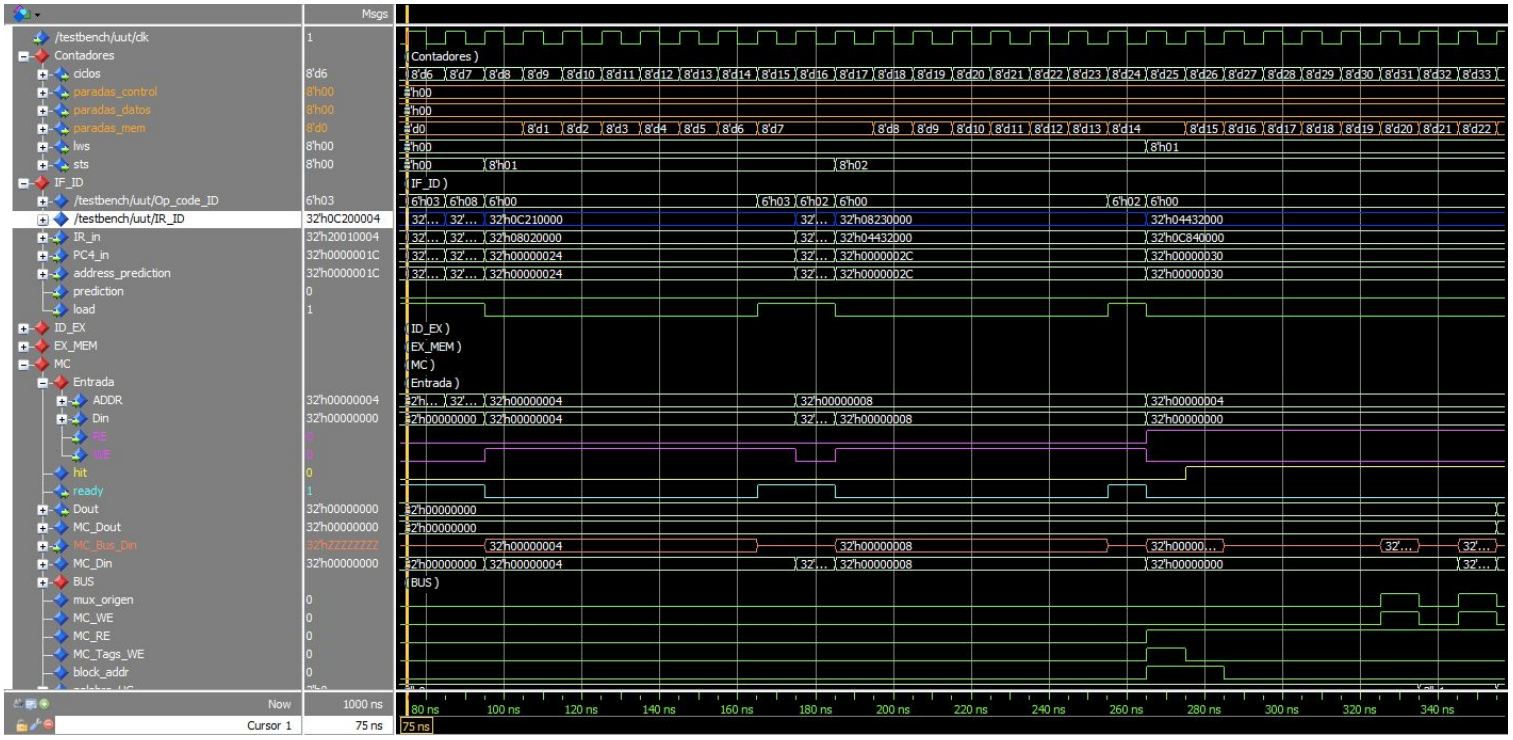


Figura 4. Banco pruebas 2.1 Comienzo del bucle y contabilización de las correspondientes paradas de memoria

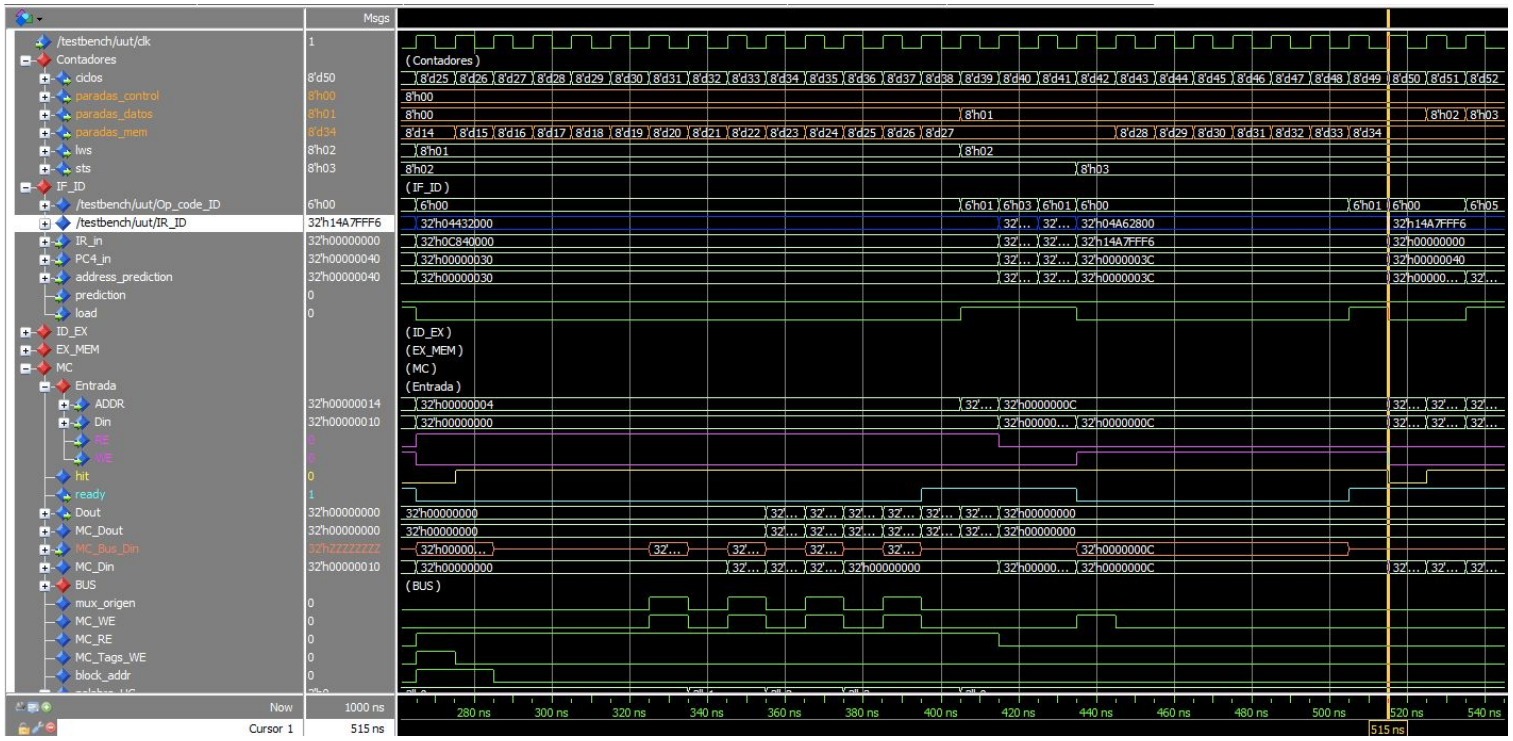


Figura 5. Banco pruebas 2.2 Últimas instrucciones del bucle e incremento de los correspondientes contadores

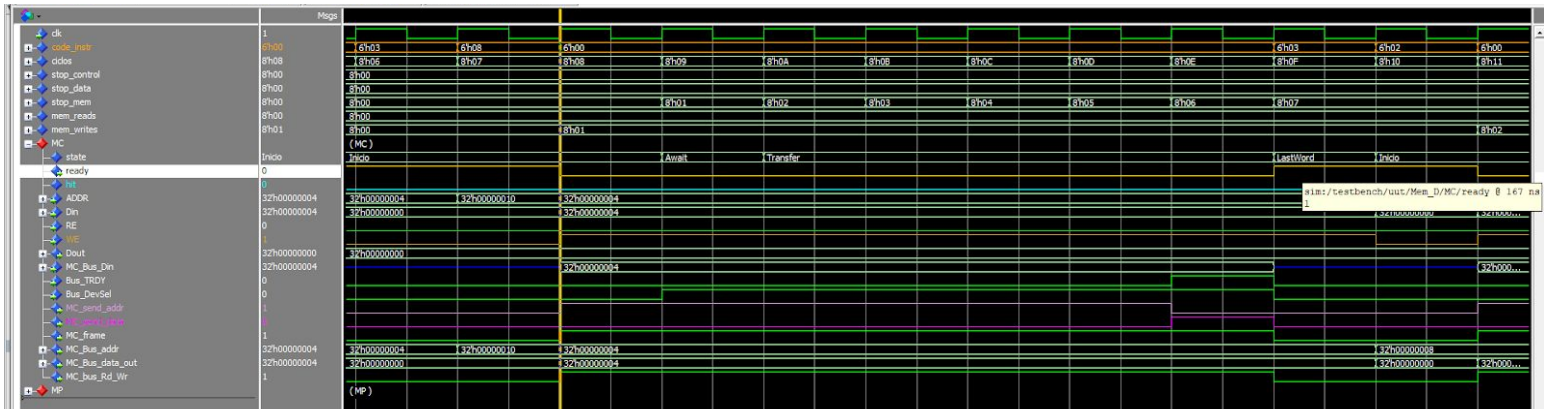


Figura 6. Banco pruebas 2. Write miss en el conjunto 0 palabra 1

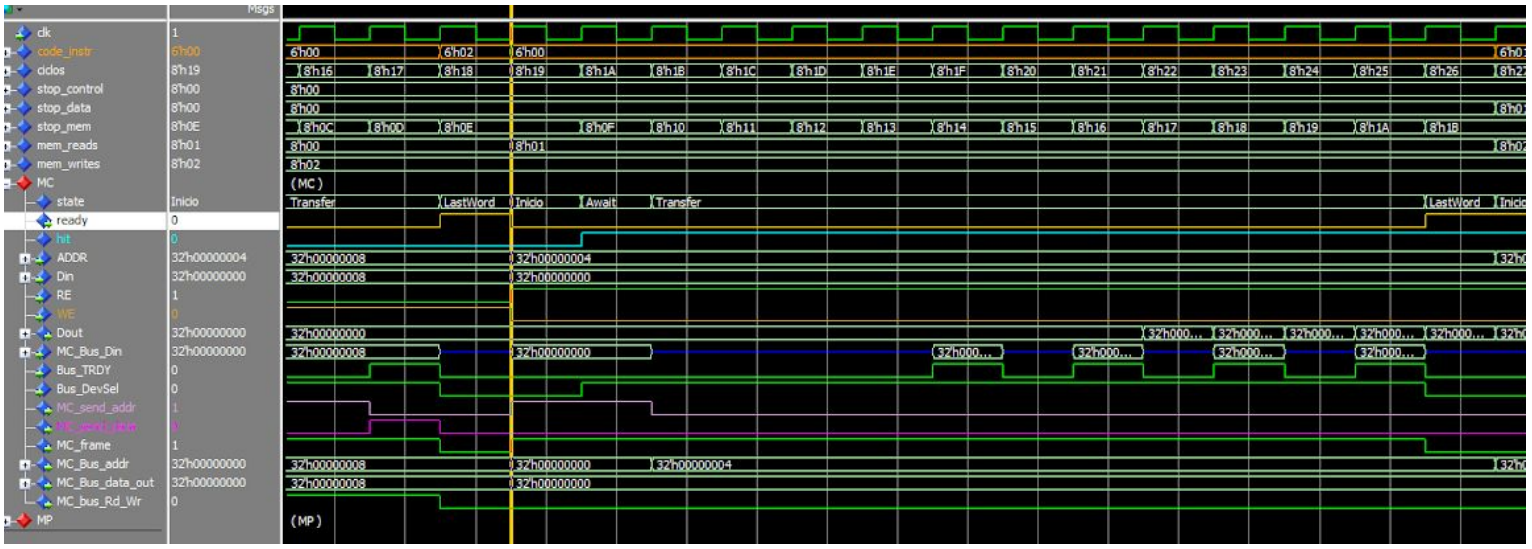


Figura 7. Banco pruebas 2. Read miss en el conjunto 0 palabra 1

Figura 8. Banco pruebas 2. Read hit en el conjunto 0 palabra 2

Conclusiones

Los accesos a memoria hacen que sea muy lento el procesador, con la memoria caché, si el programa tiene buena localidad espacial y temporal, hace que desaparezcan la mayoría de los ciclos de parada ya que al haber un fallo de lectura se llevaría el bloque entero de memoria a cache y si posteriormente hay lecturas en direcciones del mismo bloque que el anterior, al estar el bloque cargado ya en la cache la lectura sería mucho más rápida.

Al ser Write Around, al escribir una palabra no trae el bloque por lo que es mejor agrupar las lecturas por un lado y las escrituras por otro.

Tiempo dedicado

Día	Componentes	Descripción	Horas
27/05	Todos	Diseño del autómata de la unidad de control de la MC	2
27/05	Todos	Implementación de la UC_MC	2,5
28/05	Todos	Prueba testbench_MC+MD	2,5
28/05	Todos	Añadir contadores en el MIPS y en la MC	1
29/05	Todos	Debug y arreglo de la UC_MC	1
29/05	Todos	Prueba testbench proyecto 1	2
29/05	Enrique	Banco de pruebas: prueba 1	0,5
30/05	Alba	Banco de pruebas: prueba 2	1
30/05	Enrique	Banco de pruebas: prueba 1	0,5
30/05	Todos	Testbench con prueba 2	2,5
30/05	Todos	Testbench con prueba 1	1,5
31/05	Alba	Debug con prueba 2	2
31/05	Enrique	Redacción de la memoria	1,5
1/06	Alba	Redacción de la memoria	0,5
1/06	Enrique	Redacción de memoria	0’5
Total		Enrique: 18	
		Alba: 18’5	

Autoevaluación

¿Crees que has cumplido los objetivos de la asignatura?

Enrique: Si

Alba: Sí

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

Enrique: 7

Alba: 8