

C1 (1,5 punto)	C2 (1,5 punto)	C3 (1,5 punto)	C4 (1,5 punto)	C5 (1,5 punto)	P1 (2 puntos)	P2 (2 puntos)	TOTAL

## EPS - UAM - Arquitectura e Ingeniería de Computadores, Febrero 2005

Apellidos y Nombre:			
DNI :	Mañana1 (J.Garrido)	Mañana2 (S. López)	Tarde (S. López)

**ATENCION: Sólo se debe contestar a 4 de las 5 cuestiones. En el caso de contestar a las 5 sólo se considerarán las 4 primeras.**

**C1.-** Escribir el proceso equivalente a la siguiente asignación concurrente:

```

z <= a and not b when enable = '1' and sel = '0' else
  x or y when enable = '1' and sel = '1' else
  '0';

```

### SOLUCIÓN:

```

process(a,b,x,y,enable,sel)
begin
  if enable='1' and sel='0' then
    z <= a and not b;
  elsif enable='1' and sel='1' then
    z <= x or y;
  else
    z <= '0';
  end if;
end process;

```

**C2.-** Se tiene un sistema de memoria con paginación multinivel con las siguientes características:

- Espacio de direccionamiento tanto real como virtual de 4GB
- Páginas de 4KB
- Las tablas de traducción de todos los niveles ocupan una página
- Los descriptores de página de 32 bits incluyen entre otros, un bit de presencia, un bit de modificado y otro de uso.

Se pide:

a) Tamaño de los campos en que se divide la dirección virtual para ser traducida por la MMU

b) Tamaño máximo que pueden ocupar las tablas de traducción en memoria

c) Dibuja un esquema de cómo se realiza la traducción de direcciones y pon un ejemplo con una dirección virtual cualquiera de los pasos que se seguirían para traducirla. Indicar, justificando la respuesta, el número máximo de fallos de página que pueden ocurrir en un acceso a memoria.

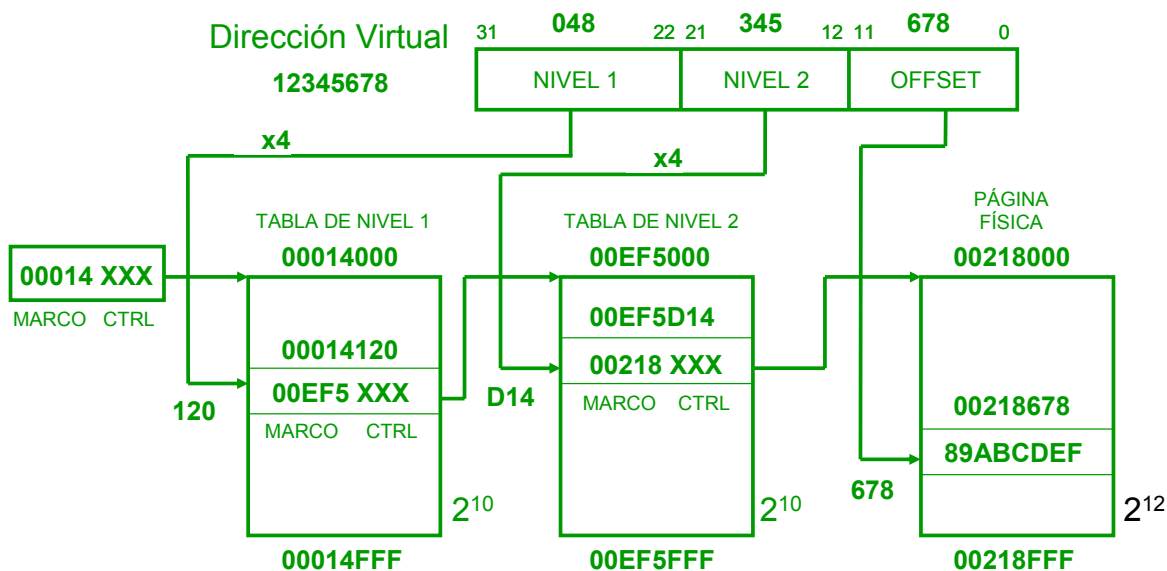
### **SOLUCIÓN:**

a) Las direcciones virtuales son de 32 bits (4 GB de direccionamiento virtual). El offset es de 12 bits (páginas de 4 KB). Si los descriptores de página ocupan 4 bytes (32 bits) y cada tabla es de 4KB, hay 1024 ( $2^{10}$ ) descriptores por tabla. Esto es, cada nivel necesita 10 bits por lo que hay dos niveles (10 bits de nivel 1 + 10 bits de nivel 2 + 12 bits de offset = 32 bits)

b) En el primer nivel hay una única tabla de 4 KB cuyos 1024 descriptores apuntan a otras tantas tablas de segundo nivel, como dice el enunciado también de 4KB. Por lo tanto, el tamaño máximo que pueden ocupar las tablas de traducción en memoria es:

$$\begin{array}{c} \text{nivel 2} \\ \downarrow \\ 4 \text{ KB} + 1024 * 4\text{KB} = 4 \text{ KB} + 4 \text{ MB} = 4100 \text{ KB} \\ \uparrow \\ \text{nivel 1} \end{array}$$

c) Supongamos los pasos que se seguirían para traducir la dirección 0x12345678:



En este ejemplo la dirección virtual 0x12345678 se traduce por la dirección física 0x00218678, cuyos contenidos son 0x89ABCDEF. Los descriptores de página incluyen un marco de página real de 20 bits y 12 bits de control (entre ellos los tres detallados en el enunciado)

En un acceso a memoria pueden ocurrir como máximo 3 fallos de página, dos si ambas tablas de traducción no están en memoria física, más otro si la posición final a la que hay que acceder tampoco lo está. Aunque el enunciado no lo indica así, si la tabla de primer nivel estuviera alojada en la MMU (como ocurre en muchos microprocesadores reales) nunca generaría una excepción y como mucho ocurrirían 2 fallos de página.

**C3.-** Se dispone de dos unidades cache para código, ambas de 256 bytes, con 32 bytes por bloque. Ambas disponen de una estrategia LRU para el reemplazamiento. La primera es completamente asociativa y la segunda asociativa de 2 vías. Partiendo de un estado con ambas caches vacías, se quiere saber **a)** la tasa de fallos en ambos casos y **b)** el estado de ambos directorios cache al final de la secuencia.

La secuencia en hexadecimal de referencias a memoria es:

25A2	3F85	4DA2	F503	2583	3F96	4DF0	F51D	3FA7	3F8F	2565	F57E	25B0
------	------	------	------	------	------	------	------	------	------	------	------	------

### SOLUCIÓN:

**a) Cache CA. Sistema con 8 entradas, todas ellas equivalentes. La dirección se divide en 11 bits para la etiqueta y 5 bits para diferenciar el byte en el bloque.**

	25 <sub>101</sub> 02	3F <sub>100</sub> 05	4D <sub>101</sub> 02	F5 <sub>000</sub> 03	25 <sub>100</sub> 03	3F <sub>100</sub> 16	4D <sub>111</sub> 10	F5 <sub>000</sub> 1D
A/F	F	F	F	F	F	A	F	A
	3F <sub>101</sub> 07	3F <sub>100</sub> 0F	25 <sub>011</sub> 05	F5 <sub>011</sub> 1E	25 <sub>101</sub> 10	Tasa de fallos 10/13 = 0,77		
A/F	F	A	F	F	F			

DIRECTORIO	
25 <sub>404</sub>	F5 <sub>011</sub>
3F <sub>100</sub>	
4D <sub>404</sub>	25 <sub>101</sub>
F5 <sub>000</sub>	
25 <sub>100</sub>	
4D <sub>111</sub>	
3F <sub>101</sub>	
25 <sub>011</sub>	

**b) Cache CA2V. Sistema con 8 entradas, todas ellas equivalentes. La dirección se divide en 9 bits para la etiqueta, 2 bits de índice y 5 bits para diferenciar el byte en el bloque.**

	25 <sub>1 01</sub> 02	3F <sub>1 00</sub> 05	4D <sub>1 01</sub> 02	F5 <sub>0 00</sub> 03	25 <sub>1 00</sub> 03	3F <sub>1 00</sub> 16	4D <sub>1 11</sub> 10	F5 <sub>0 00</sub> 1D
A/F	F	F	F	F	F	F	F	F
	3F <sub>1 01</sub> 07	3F <sub>1 00</sub> 0F	25 <sub>0 11</sub> 05	F5 <sub>0 11</sub> 1E	25 <sub>1 01</sub> 10	Tasa de fallos 12/13 = 0,92		
A/F	F	A	F	F	F			

	VIA-2	VIA-1
INDICE	DIRECTORIO	DIRECTORIO
00	F5 <sub>0</sub> 3F <sub>1</sub>	3F <sub>4</sub> 25 <sub>4</sub> F5 <sub>0</sub>
01	4D <sub>4</sub> 25 <sub>1</sub>	25 <sub>4</sub> 3F <sub>1</sub>
10		
11	25 <sub>0</sub>	4D <sub>4</sub> F5 <sub>0</sub>

**C4.-** Se dispone de un procesador segmentado en cinco etapas, captura (**CI**), decodificación (**DI**), ejecución en la ALU (**EX**), acceso a memoria para datos (**ME**) y escritura de resultados (**WE**). La dirección de salto se sabe en (DI) y la condición se resuelve en (EX). En un supuesto sistema ideal (sin riesgos), todas las instrucciones tardan 1 ciclo, excepto las de coma flotante que tardan 3 ciclos. Las únicas pérdidas del rendimiento ideal, son debidas a los riesgos de control y se sabe que el porcentaje de saltos condicionales efectivos es del 70%.

TIPO	JMP	Bcc	C.FLOTANTE	OTRAS
% USO	5	20	20	55
CPI_TOTAL (ciclos)	1	1	2	1

Se pide conocer cuantitativamente la pérdida de rendimiento frente al sistema ideal, si la estadística de instrucciones es como la que se indica en la tabla superior y se diseñan las siguientes estrategias para el tratamiento de saltos:

a) Parar el sistema hasta resolver el salto.

b) Predecir no efectivo.

c) Predicción histórica sin BTB, para la secuencia **NE-E-E-NE-E-E-E-E** suponiendo dos bits de control que cambia la predicción cada dos fallos y que por defecto comienza prediciendo efectivo.

### SOLUCIÓN:

$$CPI^{IDEAL} = 0,8*1 + 0,2*3 = 1,4$$

#### Detiene el procesador

CI	JMP	Next	Tgt			Bcc	Next	--	N+1/T				
DI(T)		JMP					Bcc	--	N				
EX(cc)			JMP					Bcc	--	N			
ME				JMP					Bcc	--	N		
WE					JMP	---	Tgt			Bcc	--	N	N+1/T

#### Predice no efectivo

CI	Bcc	Next	N+1	N+2/T									
DI(T)		Bcc	N										
EX(cc)			Bcc	N									
ME				Bcc	N								
WE					Bcc	N	N+1	N+2/T					

#### Predice efectivo

CI	Bcc	Next	Tgt	T+1/N+1									
DI(T)		Bcc	N										
EX(cc)			Bcc	N									
ME				Bcc	N								
WE					Bcc	N	Tgt	T+1/N+1					

Por cada JMP (5%) se pierde 1 ciclo.

Por cada Bcc (20%), si se para se pierden 1 ciclos si NE (30%) y 2 ciclos si E (70%).

Por cada Bcc (20%), si se predice NE se pierden 0 ciclos si se acierta (30%) y 2 ciclos si falla (70%).

Por cada Bcc (20%), si se predice E se pierde 1 ciclo si se acierta (6/8) y 1 ciclo si falla (2/8).

a)  $CPI^{RETARDO} = 0,05*1 + 0,2*(0,3*1 + 0,7*2) = 0,39$

Pérdida de rendimiento  $CPI^{IDEAL} / CPI^{RETARDO} + CPI^{IDEAL} = 1,40 / 1,79 = 0,78$ . **22% de pérdida**

b)  $CPI^{RETARDO} = 0,05*1 + 0,2*(0,3*0 + 0,7*2) = 0,33$

Pérdida de rendimiento  $CPI^{IDEAL} / CPI^{RETARDO} + CPI^{IDEAL} = 1,40 / 1,73 = 0,81$ . **19% de pérdida**

c) Si se comienza con predicción efectiva y cambia cada dos fallos, ante una secuencia de saltos como la dada **NE-E-E-NE-E-E-E-E**, siempre se predice efectivo y se falla 2/8 veces y se acierta 6/8 veces.

$$CPI^{RETARDO} = 0,05*1 + 0,2*\{(2/8)*1 + (6/8)*1\} = 0,05*1 + 0,2*\{0,25*1 + 0,75*1\} = 0,25$$

Pérdida de rendimiento  $CPI^{IDEAL} / CPI^{RETARDO} + CPI^{IDEAL} = 1,40 / 1,65 = 0,85$ . **15% de pérdida**

**C5.-** Se tiene el siguiente algoritmo:

```

y = 0;
for (i=0; i<4; i++)
{
    y += a[i]*c[i];
}

```

Que se desea ejecutar en procesador VLIW con una unidad para LD/ST, una para saltos, una para operaciones lógicas y tres para operaciones aritméticas (multiplicación y suma). La latencia de todas las unidades es de un ciclo de reloj (no hay esperas) y el procesador tiene 32 registros de propósito general.

La matriz **a[]** está guardada en memoria, y las constantes **c[]** se pueden codificar como dato inmediato en la instrucción a ejecutar (no hace falta ir a memoria para obtenerlas).

Con esta información crea las instrucciones VLIW optimizadas para que el código se ejecute lo más rápido posible, usando para ello la siguiente tabla:

LD/ST	SALTOS	LOGICA	ARIT1	ARIT2	ARIT3
LD a[0] → r1	NOP	NOP	NOP	NOP	NOP
LD a[1] → r2	NOP	NOP	r1 * c[0] → r10	NOP	NOP
LD a[2] → r3	NOP	NOP	r2 * c[1] → r11	NOP	NOP
LD a[3] → r4	NOP	NOP	r3 * c[2] → r12	r11 + r10 → r20	NOP
NOP	NOP	NOP	r4 * c[3] → r13	r12 + r20 → r20	NOP
NOP	NOP	NOP	NOP	r13 + r20 → r20	NOP
[ST r20 → y]	NOP	NOP	NOP	NOP	NOP
NOP	NOP	NOP	NOP	NOP	NOP

**La variable y se almacena en el registro r20. Opcionalmente se podría guardar en una posición de memoria, pero no es estrictamente necesario.**

NOTA: Para mayor comodidad utiliza pseudocódigo como el de estos ejemplos:

- LD a[0] → r1
- r2 \* c[1] → r1
- r2 + r3 → r1
- NOP

**P1.-** Un sistema ordenador dispone de una unidad cache unificada de correspondencia directa con una tasa de acierto del 80% y una estrategia de post-escritura (PE) aunque carece de bits de modificación. La estadística permite señalar el porcentaje de uso para cada tipo instrucciones. Cada instrucción gasta parte de su tiempo en acceder al sistema de memoria y la otra parte en la ruta de datos. En la tabla se indica para cada tipo de instrucción, los ciclos totales, así como el porcentaje de estos ciclos para uso del sistema de memoria. Una versión mejorada del sistema dispone de una cache no unificada, con porcentajes de acierto para código y datos del 95% y 92% respectivamente y además duplica la velocidad de la ruta de datos para las instrucciones en coma flotante. Se pide, utilizando necesariamente la ley de Amdahl **a)** el efecto parcial de ambas mejoras y **b)** la mejora global de la nueva versión.

**DATOS:** Se sabe que antes de la mejora, el tiempo de acceso medio por palabra a la memoria es de 3 ciclos. En la evolución no cambian ni el tiempo de acceso a las caches (1 ciclo), ni se modifica la penalización por bloque, tampoco se incorporan nuevos bits de control ni cambia, en su caso, la estrategia para escrituras.

**NOTA:** Tomar como una única mejora la evolución en el sistema cache.

TIPO	LOAD	STORE	C.FLOTANTE	OTRAS
% USO	10	15	20	55
CPI_TOTAL (ciclos)	8	8	10	5
%CPI_MEM	75	75	30	60

## SOLUCIÓN:

### a1) Mejora del sistema cache:

Antes de la mejora:	Después de la mejora:
$t_{acc}^A = \%A_M \{t_c + (1+w_M) (1-H) t_B\} = 3 \text{ ciclos}$	$t_{acc}^D = \%A_M^I \{t_c + (1-H) t_B\} + \%A_M^D \{t_c + (1+w_M) (1-H) t_B\}$
$t_B = (t_{acc}^A - \%A_M t_c) / \%A_M (1 + w_M) (1-H)$	$t_{acc}^D = 1 \{1 + (1-0,95)*3,5\} + 0,25 \{1 + 2 (1-0,92)*3,5\}$
$t_B = (3 - 1,25*1) / 1,25*(1+1)*(1-0,8) = 3,5 \text{ ciclos}$	$t_{acc}^D = 1,175 + 0,39 = 1,565 \text{ ciclos}$

De acuerdo con la ley de Amdahl, la aceleración mejorada es  $A_m = t_{acc}^A / t_{acc}^D = 3 / 1,565 = 1.92$

La mejora se aplica en las fases de acceso a memoria para todas las instrucciones es decir:

$$F_m = (0,1*0,75*8 + 0,15*0,75*8 + 0,2*0,3*10 + 0,55*0,6*5) / (0,1*8 + 0,15*8 + 0,2*10 + 0,55*5) = \\ (0,60 + 0,90 + 0,60 + 1,65) / (0,8 + 1,2 + 2 + 2,75) = 3,75/6,75 = 0.56$$

$$A_G = 1 / \{(1-F_m) + (F_m / A_m)\} = 1 / \{(0,44) + (0,56/1,92)\} = 1 / 0,732 = 1,366. \text{ Mejora el } 36,6\%$$

### a1) Mejora de la ruta de datos:

De acuerdo con la ley de Amdahl, la aceleración mejorada es  $A_m = 2$

La mejora se aplica en todas las instrucciones es decir:

$$F_m = (0,2*0,7*10) / (0,1*8 + 0,15*8 + 0,2*10 + 0,55*5) = 1,4 / (0,8 + 1,2 + 2 + 2,75) = 1,4 / 6,75 = 0.21$$

$$A_G = 1 / \{(1-F_m) + (F_m / A_m)\} = 1 / \{(0,79) + (0,21/2)\} = 1 / 0,895 = 1,117. \text{ Mejora el } 11,7\%$$

### b) Ambas mejoras conjuntas:

	Ciclos sin mejora		Ciclos si R. datos		Ciclos si Memoria	
	Mem	R. dat	Mem	R. dat	Mem	R. dat
LOAD/STORE (0,25%)	$0,75*8 = 6$	$0,25*8 = 2$	6	2	3,125	2
C. FLOTANTE (0,20%)	$0,3*10 = 3$	$0,7*10 = 7$	3	3,5	1,562	7
OTRAS (0,55%)	$0,6*5 = 3$	$0,4*5 = 2$	3	2	1,562	2

Se calcula como primera mejora la ruta de datos:  $A_G^1 = 1,117$

Para la memoria, la aceleración mejorada es la misma que antes  $A_m = 1.92$

La mejora se aplica en todas las instrucciones pero con diferente fracción mejorada:

$$F_m = (0,25*6 + 0,2*3 + 0,55*3) / (0,25*8 + 0,2*6,5 + 0,55*5) = \\ (1,5 + 0,60 + 1,65) / (2 + 1,30 + 2,75) = 3,75 / 6,05 = 0.620$$

$$A_G^2 = 1 / \{(1-F_m) + (F_m / A_m)\} = 1 / \{(0,38) + (0,62/1,92)\} = 1 / 0,703 = 1,422.$$

La mejora total será:  $A_G^1 \times A_G^2 = 1,117 \times 1,422 = 1,589$ . La mejora global es del 58,9%

Se calcula como primera mejora la memoria:  $A_G^1 = 1,366$

Para la ruta de datos, la aceleración mejorada es la misma que antes  $A_m = 2$

La mejora se aplica en todas las instrucciones pero con diferente fracción mejorada:

$$F_m = (0,2*7) / (0,25*5,125 + 0,2*8,562 + 0,55*3,562) = \\ 1,4 / (1,281 + 1,712 + 1,959) = 1,4 / 4,952 = 0.283$$

$$A_G^2 = 1 / \{(1-F_m) + (F_m / A_m)\} = 1 / (0,717) + (0,283/2) = 1 / 0,859 = 1,165.$$

La mejora total será:  $A_G^1 \times A_G^2 = 1,366 \times 1,165 = 1,591$ . La mejora global es del 59,1%

**P2.-** Tenemos un procesador RISC con arquitectura Harvard de cauce único. Está segmentado en cinco etapas: **IF** captura de instrucción, **DE** decodificación, obtención de la dirección destino en saltos y lectura de operandos, **EX** operación en la ALU y comprobación de la condición en saltos, **MEM** acceso a la memoria de datos y **WR** escritura del resultado en el banco de registros. Todas las instrucciones pasan por todas estas etapas, aunque haya alguna etapa en la que no tienen que hacer ninguna operación.

En la siguiente tabla se tiene un fragmento de código junto con el diagrama del pipeline para su ejecución (en los mnemónicos el registro destino siempre va primero). Considera que los dos saltos son efectivos, y que tanto I5 como I14 siempre son válidas y terminan su ejecución completamente. Las detenciones no justificadas por riesgo de datos, son debidas a fallos en la cache.

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18
<b>I1</b> ADD R3, R5, R7	IF	DE	EX	MEM	WR													
<b>I2</b> SUB R5, R4, R2		IF	IF	IF	DE	EX	MEM	WR										
<b>I3</b> ADD R6, R5, R0					IF	DE	EX	MEM	WR									
<b>I4</b> BCC R7, L0						IF	DE	EX	MEM	WR								
<b>I5</b> LOAD R4, (R3)							IF	DE	EX	MEM	MEM	MEM	WR					
...																		
<b>I10</b> L0: LOAD R1, (R5+10)								IF	DE	EX	EX	EX	MEM	WR				
<b>I11</b> ADD R2, R1, R10									IF	DE	DE	DE	DE	DE	EX	MEM	WR	
<b>I12</b> MUL R5, R2, R11																		
<b>I13</b> JMP L1																		
<b>I14</b> STORE (R10), R2																		
...																		
<b>I20</b> L1: AND R1, R0, R0																		

a) Señalar los potenciales riesgos de datos presentes en este fragmento de código (independientemente del procesador). Indicar cuáles de ellos causan esperas en este procesador.

**RAW:**

I1/I5 por R3  
I2/I3 por R5  
I2/I10 por R5  
I10/I11 por R1  
I11/I12 por R2  
I11/I14 por R2

**WAR:**

I1/I2 por R5  
I1/I12 por R5  
I2/I5 por R4  
I1/I11 por R2  
I3/I12 por R5  
I10/I12 por R5  
I11/I20 por R1

**WAW:**

I2/I12 por R5  
I10/I20 por R1

Adicionalmente, se pueden producir riesgos WAR en procesadores con planificación estática (VLIW) entre I5 e I14 si R3 es igual a R10, y entre I10 e I14 si R5+10 es igual a R10. De todos estos errores, sólo el riesgo RAW entre I10 e I11 por R1 causa esperas en este procesador.

Responder a las siguientes tres preguntas justificadamente. No se considerarán las repuestas sin justificar.

b) ¿Cuál es la penalización por fallo en la caché?

En I2 hay una parada no justificada de dos ciclos en el estado IF que sólo puede ser debida a un fallo en la caché. Lo mismo ocurre en la etapa MEM de I5, otra parada de injustificada de 2 ciclos. Por lo tanto, la penalización de la caché es de 2 ciclos tanto para instrucciones como para datos.

c) ¿Existe algún camino para el adelantamiento de datos? ¿Si es así, entre que etapas? ¿Permite el banco de registros acceso concurrente en un mismo ciclo para leer y escribir en un mismo registro?

Si, en el diagrama se puede observar que el riesgo RAW entre I2 e I3 por R5 no causa esperas, lo que sólo puede ser debido a que existe un camino de adelantamiento de datos entre etapas de ejecución. No existe adelantamiento de datos entre la etapa MEM y la EXE porque el riesgo entre I10 e I11 por R1 si que causa esperas. Sin embargo, en esta espera se puede observar que en el ciclo 14 al mismo tiempo que I10 escribe (etapa WR) el registro R1, I11 puede leerlo (etapa DE). Por lo tanto, si que se puede leer y escribir concurrentemente el mismo registro.

d) ¿Qué significa que las instrucciones I5 e I14 siempre sean válidas? ¿Existe predicción en el salto I4? Si es así, ¿es efectiva o no efectiva?

Esto significa que el procesador emplea la técnica de ejecución retardada (delayed slot). Si que existe predicción en este salto, y es efectiva, puesto que en el ciclo 8, cuando se está averiguando si el salto es efectivo o no, ya se está cargando la instrucción del destino del salto I10.

Finalmente, e) Terminar de rellenar en la siguiente tabla el diagrama del pipeline. Suponer que no hay fallos en la caché.

	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20	C21	C22	C23	C24	C25	C26
I1 ADD R3, R5, R7																		
I2 SUB R5, R4, R2																		
I3 ADD R6, R5, R0	WR																	
I4 BCC R7, L0	MEM	WR																
I5 LOAD R4, R3	EX	MEM	MEM	MEM	WR													
...																		
I10 L0: LOAD R1, (R5+10)	DE	EX	EX	EX	MEM	WR												
I11 ADD R2, R1, R10	IF	DE	DE	DE	DE	DE	EX	MEM	WR									
I12 MUL R5, R2, R11		IF	IF	IF	IF	IF	DE	EX	MEM	WR								
I13 JMP L1							IF	DE	EX	MEM	WR							
I14 STORE (R10), R2								IF	DE	EX	MEM	WR						
...																		
I20 L1: AND R1, R0, R0									IF	DE	EX	MEM	WR					