

## SOLUCIONES AL EXAMEN

**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.-** Se pide estudiar las mejoras del siguiente sistema de computación. El sistema original posee el siguiente uso promedio y CPI asociado a cada tipo de instrucciones.

Tipo Instrucción	Enteros	Salto incondicionales	Salto condicionales	Carga / Almacenamiento	Coma flotante
Promedio de uso	45 %	5 %	10 %	15 %	25 %
CPI	5	5	6	8	12

Dispone además de un sistema de memoria virtual paginado en dos niveles, en el que para el acceso a memoria se necesitan 4 ciclos (2 para la traducción de la dirección virtual-real y dos adicionales para el acceso a los datos). Es decir, de los CPI señalados, 4 ciclos corresponden a la fase de acceso a memoria y el resto a la ejecución propiamente dicha. El sistema original no posee ningún sistema de caché ó TLB.

Se plantean dos mejoras independientes, la primera consiste en añadir una unidad específica para las instrucciones de coma flotante que implica reducir el tiempo de ejecución a una cuarta parte. La segunda mejora pretende optimizar el acceso a memoria. Para ello se incorpora un TLB y una caché real. Las características de estas dos nuevas unidades son: tiempo de acceso al TLB 1 ciclo y frecuencia de fallos en el mismo del 4%. El acceso a la caché (1 palabra por bloque) es también de 1 ciclo y la frecuencia de aciertos de un 93%. Cada fallo se penaliza como si no hubiera habido mejora. Se quiere saber utilizando necesariamente la ley de Amdahl:

**a)** El CPI original antes de mejoras.

$$CPI_{\text{ORIG}} = 0,45 \cdot 5 + 0,05 \cdot 5 + 0,1 \cdot 6 + 0,15 \cdot 8 + 0,25 \cdot 12 = 2,25 + 0,25 + 0,6 + 1,2 + 3 = 7,3 \text{ ciclos}$$

**b)** La mejora obtenida en ambos casos por separado. ¿En cuánto es mejor una que la otra?

**b1)** Mejora debido a la unidad de punto flotante:

La mejora de la UCF supone una mejora de  $A_m = 4$  y se aplica en

$$F_m = 0,25 \cdot 8 / 7,3 = 2 / 7,3 = 0,274$$

$$A_g = 1 / \{(0,726) + (0,274/4)\} = 1,258; \text{ es decir } 25,8\%$$

*Comprobación del resultado usando el CPI:*

$$CPI_{\text{mej1}} = 0,45 \cdot 5 + 0,05 \cdot 5 + 0,1 \cdot 6 + 0,15 \cdot 8 + 0,25 \cdot 6 = 2,25 + 0,25 + 0,6 + 1,2 + 1,5 = 5,8 \text{ ciclos}$$

$$A_g = CPI_{\text{ORIG}} / CPI_{\text{mej1}} = 7,3 / 5,8 = 1,258.$$

**b2)** Mejora debido a TLB y caché:

La mejora realizada implica una mejora promedio que viene definida por la expresión:

$$t_{\text{acc}} = (t_{\text{TLB}} + F_{\text{TLB}} \cdot P_{\text{enTLB}}) + (t_{\text{CACHE}} + F_{\text{CACHE}} \cdot P_{\text{enCACHE}}) = (1 + 0,04 \cdot 2) + (1 + 0,07 \cdot 2) = 2,22 \text{ ciclos.}$$

Por lo tanto la aceleración mejorada será:  $A_m = 4 / 2,22 = 1,801$ .

La mejora en el sistema de memoria afecta a la captura de instrucciones (1) y a las instrucciones con acceso para datos (0,15).  $F_m = (1+0,15) \cdot 4 / 7,3 = 0,63$

$$A_g = 1 / \{(0,37) + (0,63 / 1,801)\} = 1,389; \text{ es decir } 38,9\%$$

*Comprobación del resultado usando el CPI:*

$$CPI_{\text{mej2}} = 2,22 + (0,45 \cdot 1 + 0,05 \cdot 1 + 0,1 \cdot 2 + 0,15 \cdot 2,22 + 0,25 \cdot 8) = 2,22 + (0,45 + 0,05 + 0,2 + 0,333 + 2) = 5,253 \text{ ciclos}$$

$$A_g = CPI_{\text{ORIG}} / CPI_{\text{mej2}} = 7,3 / 5,253 = 1,389$$

**c) ¿Cuál será el rendimiento de la mejora conjunta del sistema?**

Para tratar ambas mejoras al tiempo partimos de la segunda ya realizada y se aplica la primera sobre los resultados ya mejorados.

$$CPI_{mej2} = 2,22 + (0,45*1+0,05*1+0,1*2+0,15*2,22+0,25*8) = 2,22+(0,45+0,05+0,2+0,333+2) = 5,253 \text{ ciclos}$$

La mejora de la UCF supone una mejora de  $A_m = 4$  y se aplica en:

$$F_m = 0,25*8 / 5,253 = 0,38$$

$$A_g = 1 / \{(0,62) + (0,38/4)\} = 1,398.$$

**La mejora global será de  $1,389 * 1,398 = 1,94$ . Es decir de 94 %.**

Igualmente para tratar ambas mejoras podemos partir de la mejora 1 (en la UCF) y luego aplicar la mejora 2 (caché + TLB). De ser así tiene:

$$CPI_{mej1} = 0,45*5 + 0,05*5 + 0,1*6 + 0,15*8 + 0,25*6 = 2,25 + 0,25 + 0,6 + 1,2 + 1,5 = 5,8 \text{ ciclos}$$

$$\text{Dado el } t_{acc} = (t_{TLB} + F_{TLB} * Pen_{TLB}) + (t_{CACHE} + F_{CACHE} * Pen_{CACHE}) = 2,22 \text{ ciclos, la } A_m = 4 / 2,22 = 1,801.$$

$$F_m = (1+0,15)*4 / 5,8 = 0,793$$

$$A_{g2tras1} = 1 / \{(0,207) + (0,793/1,801)\} = 1,544.$$

**La mejora global será:  $A_{gtotal} = A_{g1} * A_{g2tras1} = 1,258 * 1,544 = 1,94$ . Es decir 94 %.**

**C2.-** Considere un máquina con una memoria principal de 64 kbytes, direccionable por bytes y con un tamaño de bloque de 8 bytes. Suponga que con esta máquina se utiliza una caché de 32 líneas y correspondencia directa. Responda justificando en cada caso brevemente la respuesta:

- a) ¿Señalar cómo se divide la dirección de memoria entre etiqueta, número de línea o índice y número de byte en el bloque?

ETIQUETA	INDICE	BYTE EN BLOQUE
8	5	3

JUSTIFICACION del apartado a:

Si la memoria es de 64 kB =  $2^{16}$  bytes, la dirección es de 16 bits.  
 Si un bloque contiene 8 bytes, el campo byte en bloque es de 3 bits.  
 Si la caché de CD tiene 32 líneas, el campo índice tiene 5 bits.  
 Luego el campo etiqueta es de  $16 - 3 - 5 = 8$  bits.

- b) ¿En qué líneas se almacenarían los bytes que se encuentran en las siguientes direcciones dadas en binario?

- 1.- 0001 0001 0001 1011.....Se almacena en la línea: 4ª (índice 00011)  
 2.- 1100 0011 0011 0100.....Se almacena en la línea: 7ª (índice 00110)  
 3.- 1101 0000 0001 1101.....Se almacena en la línea: 4ª (índice 00011)  
 4.- 1010 1010 1010 1010.....Se almacena en la línea: 22ª (índice 10101)

JUSTIFICACION del apartado b:

**Con la respuesta**

- c) Suponga que se almacena en la caché el byte de dirección 0001 1010 0001 1010. ¿Cuáles son las direcciones (en hexadecimal) de los bytes que se almacenan junto con él?

Las direcciones son:

**1A1F<sub>16</sub> ; 1A1E<sub>16</sub> ; 1A1D<sub>16</sub> ; 1A1C<sub>16</sub> ; 1A1B<sub>16</sub> ; 1A1A<sub>16</sub> ; 1A19<sub>16</sub> y 1A18<sub>16</sub>**

JUSTIFICACION del apartado c:

**Las direcciones que se guardan en un mismo bloque comparten etiqueta e índice.**

- d) ¿Cuántos bytes de memoria puede almacenar en total la caché?

El número total de bytes en la caché es: **32 (bloques) x 8 (bytes/bloque) = 256 bytes**

JUSTIFICACION del apartado d:

**Con la respuesta**

**C3.-** En un procesador segmentado de cinco etapas, S1: captura de instrucción, S2: decodificación y captura de operandos, S3: ejecución, S4: acceso a memoria de datos y S5: escritura en registro. La dirección de salto se conoce en la segunda y la resolución del mismo en la tercera etapa. Para un determinado programa se ha detectado para cada 100 instrucciones, se repite la siguiente secuencia para los saltos condicionales NE-NE-E-E-E-NE-E-NE-E-E-NE-NE-E-E-E (E: Efectivo; NE: No Efectivo). Además se sabe que un 5% de las instrucciones son saltos incondicionales. Si en el sistema sólo se producen riesgos de control, se quiere conocer el efecto en el rendimiento sobre un sistema que retiene el proceso hasta que se detecta la condición de salto, para las siguientes arquitecturas:

a) Predicción estática efectiva.

b) Predicción dinámica histórica con control de 2 bits (cambia la predicción tras 2 fallos consecutivos).

Justifique cada una de las respuestas con los correspondientes cronogramas. Por defecto, cuando sea necesario, suponer que se predice efectivo.

### SOLUCION:

Se detiene el proceso hasta conocer la condición.

<b>S1</b>	JMP	N	T				
<b>S2 (T)</b>		JMP		T			
<b>S3 (cc)</b>			JMP		T		
<b>S4</b>				JMP		T	
<b>S5</b>					JMP		T

Los saltos incondicionales penalizan siempre con 1 ciclo.

<b>S1</b>	Bcc	N		N+1/T				
<b>S2 (T)</b>		Bcc		Nc	N+1/T			
<b>S3 (cc)</b>			Bcc		N	N+1/T		
<b>S4</b>				Bcc		N	N+1/T	
<b>S5</b>					Bcc		N	N+1/T

Los saltos condicionales penalizan con 1 ciclo si es no efectivo (NE) y con 2 ciclos si es efectivo (E).

$$CPI = CPI_{IDEAL} + Retardo_{JMP} + Retardo_{BCC} = 1 + 0,05 * 1 + 0,06 * 1 + 0,09 * 2 = 1,29 \text{ ciclos.}$$

a) Predicción estática efectiva (E)

<b>S1</b>	Bcc	N	T	N+1/T+1				
<b>S2 (T)</b>		Bcc	N	T	N+1/T+1			
<b>S3 (cc)</b>			Bcc	N	T	N+1/T+1		
<b>S4</b>				Bcc	N	T	N+1/T+1	
<b>S5</b>					Bcc	N	T	N+1/T+1

Los saltos condicionales penalizan con 1 ciclo tanto si es no efectivo (NE) como si es efectivo (E).

$$CPI = CPI_{IDEAL} + Retardo_{JMP} + Retardo_{BCC} = 1 + 0,05 * 1 + 0,15 * 1 = 1,20 \text{ ciclos.}$$

La predicción mejora el rendimiento en un 7,5% ( $1,29/1,20=1,075$ ).

b) Predicción dinámica efectiva, equivale a la predicción estática (a), penaliza con 1 ciclo tanto si acierta (E) como si falla (NE).

Predicción dinámica no efectiva (NE).

<b>S1</b>	Bcc	N	N+1	N+2/T				
<b>S2 (T)</b>		Bcc	N	N+1	N+2/T			
<b>S3 (cc)</b>			Bcc	N	N+1	N+2/T		
<b>S4</b>				Bcc	N	N+1	N+2/T	
<b>S5</b>					Bcc	N	N+1	N+2/T

Los saltos condicionales NO penalizan si acierta (NE) y penaliza con 2 ciclos si falla (E).

	NE	NE	E	E	E	NE	E	NE	E	E	NE	NE	E	E	E
Predicción	E	E	NE	NE	E	E	E	E	E	E	E	E	NE	NE	E
A/F	F	F	F	F	A	F	A	F	A	A	F	F	F	F	A

Hay 11 predicciones E (6 fallos y 5 aciertos) y 4 predicciones NE (todas fallan)

$$CPI = CPI_{IDEAL} + Retardo_{JMP} + Retardo_{BCC} = 1 + 0,05 * 1 + 0,11 * 1 + 0,04 * 2 = 1,24 \text{ ciclos.}$$

La predicción mejora el rendimiento en un 4,0% ( $1,29/1,24=1,040$ ).

**C4.-** Suponer que la señal *s*, de tipo *std\_logic*, tiene actualmente el valor '0'. ¿Cuál es el valor de las variables *std\_logic* *v1* y *v2* después de la ejecución de las siguientes sentencias dentro de un proceso?

```
s <= '1';

if s = '1' then
    v1 := '1';
else v1 := '0';
end if;

wait on s;

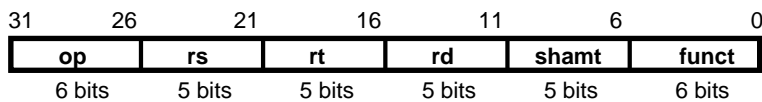
if s = '1' then
    v2 := '1';
else v2 := '0';
end if;
```

Razona brevemente tu respuesta.

**v1 = '0' y v2 = '1'**

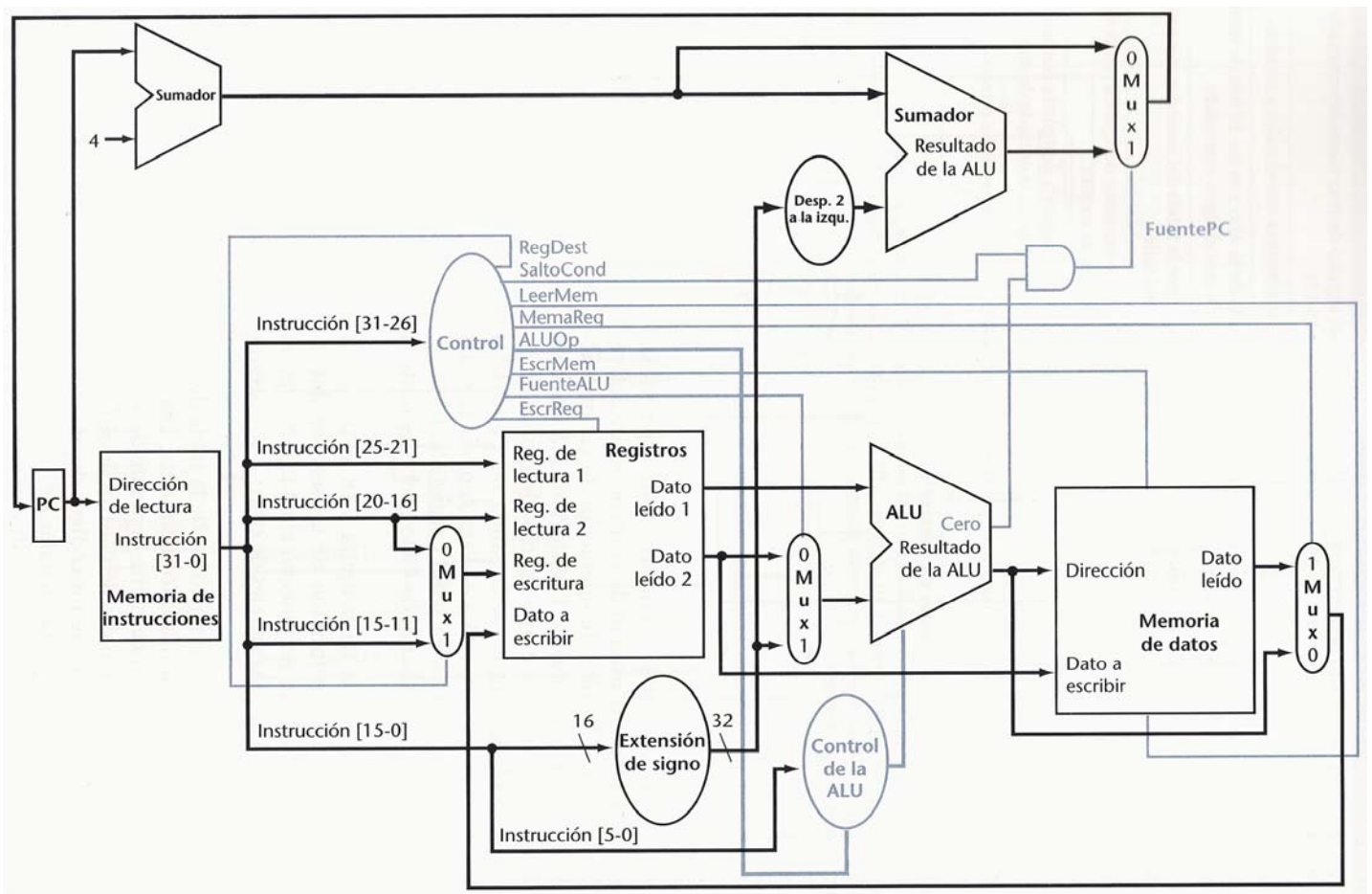
Al no haberse suspendido el proceso, el valor de la señal *s* todavía es '0' en el primer *if*; por lo tanto a la variable *v1* se le asigna '0'. Con la sentencia *wait*, se suspende el proceso y se actualiza el valor de la señal *s* a '1'. De esta manera a la señal *v2* se le asigna '1' en el segundo *if*.

**C5.-** Se tiene un sencillo sistema RISC con control en un único ciclo (similar al descrito en la asignatura). Las instrucciones son de 32 bits y los bits se distribuyen como en la figura.



La unidad de control decodifica la instrucción en función de los 6 bits de mayor peso (31:26) generando las 9 señales de control que se muestran en el gráfico. La acción de cada señal se detalla a continuación:

Señal	Acción
<b>RegDst</b>	Control del multiplexor de registro de destino
<b>SaltoCond (Branch)</b>	Señal de habilitación del multiplexor de salto. Si además el resultado de la operación es cero carga los 16 bits de menor peso de la instrucción desplazados en el contador de programa PC
<b>LeerMem (MemRead)</b>	Señal de habilitación de lectura de la memoria de datos.
<b>MemaReg (Mem2Reg)</b>	Control del multiplexor de memoria (1, dato de mem; 0, dato directo de ALU)
<b>ALUOp</b>	2 bits que indican que hará la ALU. 00 se utiliza la ALU para cálculo de direcciones, 10: operación entre registros (depende de los bits 5:0 de la instrucción); 01 si es una operación de salto; 11 prohibido.
<b>EscrMem (MemWrite)</b>	Habilitación de escritura en memoria de datos
<b>FuenteALU (ALUScr)</b>	Multiplexor de la segunda entrada a la ALU
<b>EscrReg (RegWrite)</b>	Habilitación de escritura en los registros.



- a. Complete en el cuadro adjunto la salida del bloque combinacional “control” dependiendo de los tipos de instrucciones que se ejecuten. (Utilice el símbolo X para indicar “no importa”)

Instrucción	RegDst	ALUSrc	Mem2Reg	RegWrite	MemRd	MemWr	Branch	ALUOp1	ALUOp0
ADD	1	0	0	1	0	0	0	1	0
LOAD	0	1	1	1	1	0	0	0	0
STORE	X	1	X	0	0	1	0	0	0
BEQ	X	0	X	0	0	0	1	0	1
AND	1	0	0	1	0	0	0	1	0

Nota: Descripción de las instrucciones:

<b>ADD rs, rt, rd</b>	Suma el contenido de los registros R(rs) y R(rt) y lo guarda en el registro R(rd)
<b>LOAD rs, rt, inmediato</b>	carga en el registro R(rt) el contenido de memoria de la posición $MEM(\text{sign\_ext}(\text{inmediato}) + R(rs))$
<b>STORE rs, rt, Inmediato</b>	guarda en la posición $MEM(\text{sign\_ext}(\text{inmediato}) + R(rs))$ , el contenido del registro R(rt)
<b>BEQ rs, rt, Inmediato</b>	si $R(rs) = R(rt)$ , el PC (programm counter) se carga con $PC + \text{sign\_ext}(\text{inmediato}) * 4$ , Si no con $PC + 4$
<b>AND rs, rt, rd</b>	Realiza el AND bit a bit del contenido de los registros R(rs) y R(rt) y guarda el resultado en el registro R(rd)

- b. ¿Cuales es la principal desventaja del control uniclo que motiva la utilización de arquitecturas multiciclo?

Típicamente tiempo de ciclo muy largo. Todas las instrucciones utilizan, sin necesidad, tanto tiempo como la instrucción más lenta.

- c. ¿Cuál es el CPI en una arquitectura con control uniclo sin considerar detenciones en la memoria?

Uno.

**P1.-** Se quiere conocer el tiempo de acceso medio por instrucción en un sistema de memoria virtual con las siguientes características.

1.- Una caché unificada virtual de correspondencia directa, con estrategia de post-escritura (PE) y con 4 palabras por bloque. El tiempo de acceso por palabra es de 10 nsec, el porcentaje de bloques modificados es del 40% y la tasa de fallos es del 5%.

2.- Sistema MMU, dotado de un TLB completamente asociativo de 32 entradas, un acierto se gestiona en 15 nsec y un fallo se penaliza con 120 nsec necesarios para obtener la dirección real. En el TLB se acierta el 98% de las veces que se accede a él.

3.- La memoria real, formada por una unidad SDRAM cuyo tiempo de acceso medio por palabra es de 100 nsec.

La estadística en el uso del sistema indica que el 25% de las instrucciones utilizadas son de acceso a datos, de la cuales el 35% son escrituras.

## **SOLUCION**

El tiempo para leer/escribir un bloque desde/a memoria es:  $t_B = 4 \cdot t_m = 4 \cdot 100 = 400$  nsec.

Para cada acceso a memoria:

1. La lectura/escritura (acierto o fallo) en caché es:  $t_C = 10$  nsec
2. Si fallo en la caché (virtual), antes de ir a memoria se debe traducir la DV=>DR. ( $F_{CACHE} = 5\%$ )
  - a. Tiempo para traducir DV => DR.
    - i. Acierto en el TLB:  $(1 - H_{CACHE}) \cdot t_{TLB}$ :  $0,05 \cdot 15 = 0,75$  nsec
    - ii. Fallo en el TLB ( $F_{TLB} = 2\%$ ):  $(1 - H_{CACHE}) \cdot (1 - H_{TLB}) \cdot t_P = 0,05 \cdot 0,02 \cdot 120 = 0,12$  nsec
  - b. Penalización para escribir en memoria los bloques sucios (PE):
 
$$(1 - H_{CACHE}) \cdot w_M \cdot t_B \Rightarrow (0,05) \cdot (0,4) \cdot 400 = 8$$
 nsec.
  - c. Penalización para leer el bloque demandado:
 
$$(1 - H_{CACHE}) \cdot t_B \Rightarrow (0,05) \cdot (1) \cdot 400 = 20$$
 nsec.

Tiempo promedio por cada acceso:

$$/t_{ACC} = t_C + (1 - H_{CACHE}) \cdot [t_{TLB} + (1 - H_{TLB}) \cdot t_P] + \{(1 - H_{CACHE}) \cdot (1 + w_M) \cdot t_B\}$$

$$/t_{ACC} = 10 + 0,75 + 0,12 + 28 = 38,87$$
 nsec.

Como hay en promedio %Acc = 1,25 accesos a memoria por instrucción, el tiempo promedio por instrucción será:

$$/t^M_{ACC} = \%Acc \cdot /t_{ACC} = 1,25 \cdot 38,87 = 48,59$$
 nsec



**P2.-** Se dispone de dos sistemas con planificación dinámica de instrucciones. El primero utiliza la técnica del marcador estudiada para el CDC6600 (scoreboard) y el segundo está basado en la implementación de IBM del algoritmo de Tomasulo. Ambos sistemas disponen de 2 unidades para suma/resta y operaciones lógicas que tardan 3 ciclos, una unidad para multiplicar que consume 8 ciclos, una unidad para la división que utiliza 12 y una única unidad de LD/ST que accede a memoria en 4 ciclos. Existe un único puerto de escritura en los registros internos.

NOTA: la sintaxis de los operandos es en todos los casos  $R_{\text{DESTINO}}, R_{\text{FUENTE}}, R_{\text{FUENTE}}$ .

a) Indique los riesgos potenciales RAW, WAR y WAW del siguiente código

	Instrucción	RAW	WAR	WAW
I1:	LD R2, 200+R1	I2 con I1 por R2	I3 con I2 por R3	I6 con I2 por R7
I2:	MUL R7,R2,R3	I6 con I1 por R2	I4 con I1 por R1	
I3:	ADD R3,R4,R5	I4 con I3 por R3	I6 con I5 por R7	
I4:	DIV R1,R3,R4	I5 con I2 por R7		
I5:	SUB R6,R7,R8	I6 con I3 por R3		
I6:	ADD R7,R3,R2			

b) Complete la tabla adjunta correspondiente a la ejecución de la secuencia de instrucciones para el sistema por Marcador del CDC6600 (Scoreboard)

	Instrucción	EMISION	CAP. OPER.	EJECUCION	ESCRITURA
I1:	LD R2, 200+R1	1	2	3-6	7
I2:	MUL R7,R2,R3	2	8 <sup>(a)</sup>	9-16	17
I3:	ADD R3,R4,R5	3	4	5-7	9 <sup>(b)</sup>
I4:	DIV R1,R3,R4	4	10 <sup>(c)</sup>	11-22	23
I5:	SUB R6,R7,R8	5	18 <sup>(d)</sup>	19-21	22
I6:	ADD R7,R3,R2	18 <sup>(e)</sup>	19	20-22	24 <sup>(f)</sup>

#### Aclaraciones:

- Espera la escritura de R2 por parte de I1 en el ciclo 7.
- Espera la escritura en R3 hasta la lectura por parte de I2 en el ciclo 8. Riesgo WAR
- Espera la escritura en R3 en ciclo 9 por I3. Riesgo RAW
- Espera la escritura en R7 en ciclo 17 por I2. Riesgo RAW
- Espera que se libere alguna unidad de suma/resta (I3, ciclo 9) y que el registro destino esté libre (R7 de la I2 en 17, riesgo WAW)
- Si existe un solo puerto de escritura debe esperar. Se considera correcto colocar 23 ya que el enunciado no lo dice

c) Complete la tabla adjunta correspondiente a la ejecución de la secuencia de instrucciones para el sistema que implementa el algoritmo de Tomasulo

	Instrucción	EMISION	EJECUCION	ESCRITURA
I1:	LD R2, 200+R1	1	2-5	6
I2:	MUL R7,R2,R3	2	7-14 <sup>(a)</sup>	15
I3:	ADD R3,R4,R5	3	4-6	7 <sup>(b)</sup>
I4:	DIV R1,R3,R4	4	8-19 <sup>(c)</sup>	20
I5:	SUB R6,R7,R8	5	16-18	19
I6:	ADD R7,R3,R2	8 <sup>(d)</sup>	9-11	12 <sup>(e)</sup>

#### Aclaraciones:

- Espera la escritura de R2 por parte de I1 en el ciclo 6. (RAW)
- La misma arquitectura previene los riesgos WAR
- Espera la escritura en R3. Riesgo RAW
- no emite hasta que tiene una estación de suma/resta libre (ciclo 7 de I3)
- La arquitectura previene los riesgos WAW a través de las estaciones de reserva (renombramiento dinámico).