TABLA 1: El procesador PROCEXA de 32 bits, puede captar TRES instrucciones por ciclo, DECODIFICAR TRES instrucciones por ciclo, y dispone de una ventana de instrucciones centralizada desde donde se produce la EMISIÓN DESORDENADA de hasta DOS instrucciones por ciclo.

También dispone de un buffer de reordenamiento (ROB) donde implementa el renombramiento y la finalización ordenada, pudiendo retirarse desde el ROB, TRES instrucciones por ciclo para escribir sus resultados en la etapa WB del cauce. El procesador incluye además DOS unidades de Suma/Resta de UN ciclo; UN multiplicador de TRES ciclos, y UNA unidad de Carga/Almacenamiento de memoria de DOS ciclos.

Por tanto, las etapas del cauce del procesador son: IF (captación de instrucciones), ID (Decodificación de instrucciones), EX (ejecución en unidad funcional de la operación codificada por la instrucción), y WB (retirada de la instrucción del ROB y escritura de resultado en los registros del procesador).

No se considera etapa ROB explícita en el cauce porque se supone que en el último ciclo de ejecución de las unidades funcionales, el resultado de la operación queda almacenado en el ROB.

Para este procesador y la secuencia de instrucciones siguiente

```
(1)  ld r1,0(r3)      // r1=M(r3)
(2)  ld r2,8(r3)      // r2 = M(r3+8)
(3)  add r2, r1, r2   // r2=r1+r2
(4)  mul r4, r1, r2   // r4= r1*r2
(5)  sub r4, r1, r4   // r4=r1-r4
(6)  sub r5, r1, r3   // r5=r1-r3
```

TABLA SOLUCIONADA:

		1	2	3	4	5	6	7	8	9	10	11	12
(1)	ld r1,0(r3) ; r1=M(r3)	IF	ID	EX	EX	WB							
(2)	ld r2,8(r3) ; r2=M(r3+8)	IF	ID			EX	EX	WB					
(3)	add r2,r1,r2 ; r2=r1+r2	IF	ID					EX	WB				
(4)	mul r4,r1,r2 ; r4=r1*r2		IF	ID					EX	EX	EX	WB	
(5)	sub r4,r1,r4 ; r4=r1-r4		IF	ID								EX	WB
(6)	sub r5,r1,r3 ; r5=r1-r3		IF	ID		EX							WB

Para el procesador PROCEXA de la **TABLA 1**:

EJERCICIO 1: ¿En qué ciclo se empieza a ejecutar la instrucción (6)? (los ciclos se numeran empezando en 1).
SOLUCIÓN: Ciclo 5.

EJERCICIO 2: ¿En qué ciclo se empieza a ejecutar la instrucción (2)? (los ciclos se numeran empezando en 1).
SOLUCIÓN: Ciclo 5.

EJERCICIO 3: ¿En qué ciclo se empieza a ejecutar la instrucción (4)? (los ciclos se numeran empezando en 1).
SOLUCIÓN: Ciclo 8.

EJERCICIO 4: ¿Cuántos ciclos tarda en procesarse la secuencia de seis instrucciones indicada?
SOLUCIÓN: 12 ciclos.

TABLA 2: El cauce de un superescalar tiene las siguientes etapas:

a) IF (1 ciclo = 1 c para cada instrucción) capaz de procesar 4 instrucciones por ciclo (i/c).
b) ID (1 c) capaz de procesar 4 i/c.
c) EX (de 1c a 4c) dependiendo de la unidad.
d) WB (1c) capaz de retirar del buffer de reorden (ROB) 4 i/c.

Dispone de las siguientes unidades:

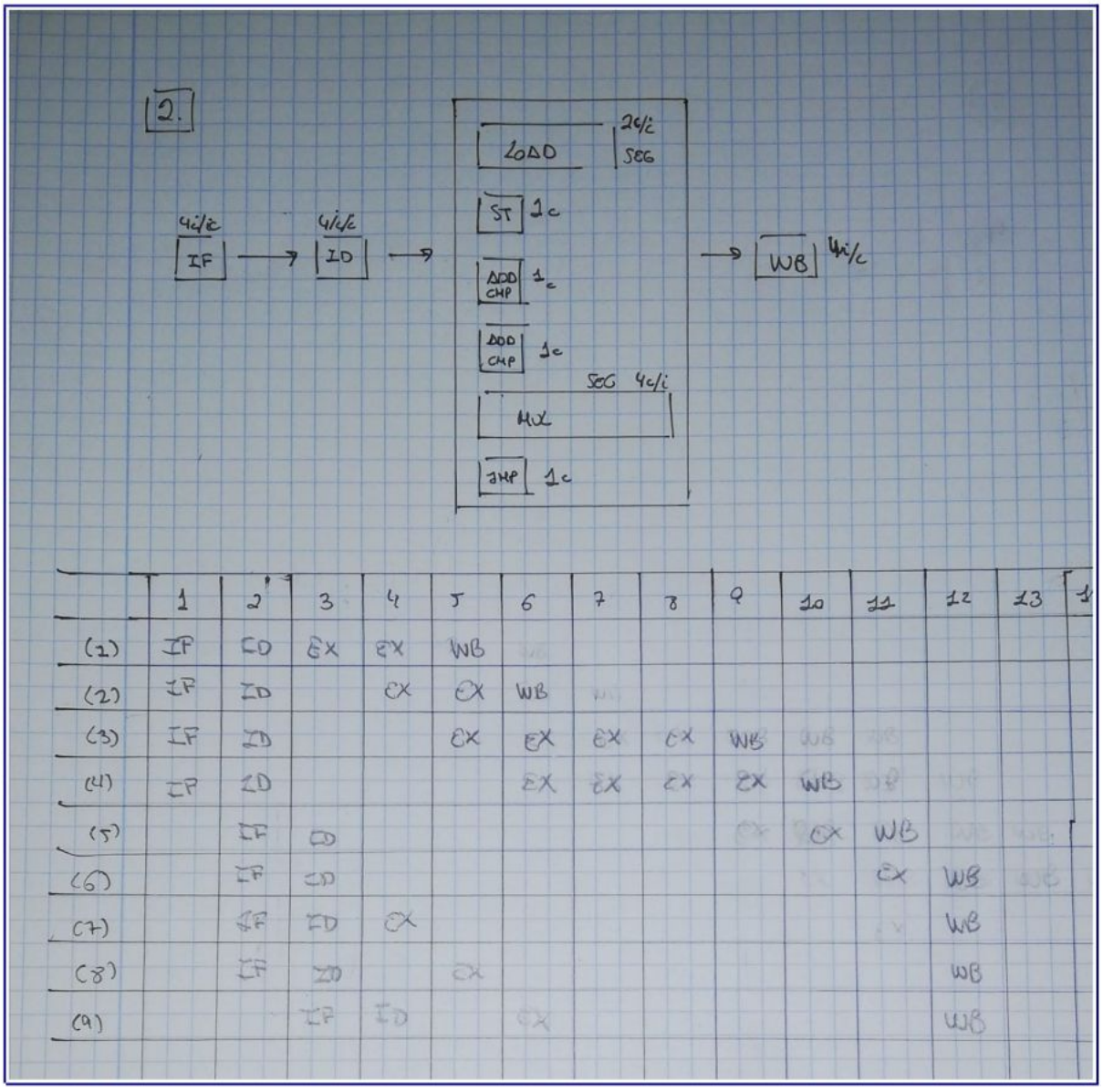
A) 1 unidad para carga de memoria segmentada en dos etapas de 1c cada una.
B) 1 unidad de almacenamiento en memoria (1 c).
C) 2 unidades para *addq* y *cmpl* (1 c).
D) 1 unidad para *addsd* (1 c).
E) 1 unidad para *mulsd* segmentada en 4 etapas de 1c cada una.
F) 1 unidad para saltos (1c).

Tenga en cuenta que no hay límite en las entradas del ROB y de la ventana de instrucciones centralizada y que se pueden emitir un máximo de 4 i/c. ¿El siguiente código tarda en procesarse 14 ciclos en el cauce descrito?

```
.L6
(1) movsd  (%r12,%rax,8), %xmm2        ;    xmm2=M[r12+rax*8]
(2) movsd  0(%rbp,%rax,8), %xmm4       ;    xmm4=M[rbp+rax*8]
(3) mulsd  %xmm1, %xmm2                ;    xmm2=xmm2*xmm1
(4) mulsd  %xmm3, %xmm4                ;    xmm4=xmm4*xmm3
(5) addsd  %xmm4, %xmm2                ;    xmm2=xmm2+xmm4
(6) movsd  %xmm2, 0(%r13,%rax,8)       ;    M[r13+rax*8]=xmm2
(7) addq   $1, %rax                    ;    rax=rax+1
(8) cmpl   %eax, %ebx                  ;    eax-ebx
(9) jg     .L6                         ;    Salto si eax-ebx>0
```

RESPUESTA: FALSO

SOLUCIÓN TABLA:



	1	2	3	4	5	6	7	8	9	10	11	12	13
(1)	IF	ID	EX	EX	WB								
(2)	IF	ID		EX	EX	WB							
(3)	IF	ID			EX	EX	EX	EX	WB	WB			
(4)	IF	ID				EX	EX	EX	EX	WB			
(5)		IF	ID							EX	WB		
(6)		IF	ID								EX	WB	
(7)		IF	ID	EX								WB	
(8)		IF	ID		EX							WB	
(9)			IF	ID		EX						WB	

EJERCICIO 5: Suponga que en un mismo ciclo se decodifican las cuatro instrucciones siguientes.
(i) add r2,r2,r1 //r2=r2+r1
(i+1) mul r3,r4,r1 //r3=r4*r1
(i+2) add r4,r4,r1 //r4=r4+r1
(i+3) sub r5,r2,r1 //r5=r2-r1

(entre paréntesis se indica el orden en el que están en el código), y pasan a una ventana de instrucciones única desde la que se emiten a las distintas unidades funcionales. El procesador puede emitir (con emisión DESORDENADA) CUATRO instrucciones por ciclo y tiene DOS unidades de suma/resta (con un retardo de 1 ciclo) y UN multiplicador (con un retardo de 3 ciclos).
Las instrucciones (i) e (i+3) se pueden emitir en el mismo ciclo.
RESPUESTA: FALSO. Hay dependencia RAW entre ellas.

EJERCICIO 6: En la misma situación de la pregunta anterior para la misma secuencia de instrucciones. Las cuatro instrucciones se pueden emitir en un único ciclo.
RESPUESTA: FALSO. Hay dependencia RAW entre la primera y la última.

EJERCICIO 7: Un procesador de 32 bits (4 bytes) NO podría adelantar la instrucción i+1 a la i (i precede a i+1 en el código) aunque implemente adelantamiento de LOADs a STORES ESPECULATIVO.

```
(i)     sw 0(r6), r2     // M(r6)<-- r2
(i+1)   lw r4, 8(r6)     // r4 <--M(r6+8)
```

RESPUESTA: FALSO. El dato que se escribe y el que se lee no se solapan y no hay riesgo RAW. No hace falta que el procesador tenga adelantamiento especulativo.

EJERCICIO 8: Un procesador de 32 bits (4 bytes) que implementa adelantamiento de loads a stores ESPECULATIVO podría adelantar la instrucción i+1 a la i (i precede a i+1 en el código)

```
(i)     sw 0(r6), r2     // M(r6)<-- r2
(i+1)   lw r4, 8(r6)     // r4 <--M(r6+8)
```

RESPUESTA: VERDADERO. El dato que se escribe y el que se lee no se solapan y no hay riesgo RAW. No hace falta que el procesador tenga adelantamiento especulativo.

EJERCICIO 9: Un procesador de 32 bits (4 bytes) que NO implementa adelantamiento de loads a stores ESPECULATIVO podría adelantar la instrucción i+1 a la i (i precede a i+1 en el código)

```
(i)     sw 0(r6), r2     // M(r6)<-- r2
(i+1)   lw r4, 8(r6)     // r4 <--M(r6+8)
```

RESPUESTA: VERDADERO. El dato que se escribe y el que se lee no se solapan y no hay riesgo RAW. No hace falta que el procesador tenga adelantamiento especulativo.

EJERCICIO 10: Un procesador podría adelantar la instrucción i+1 a la i (i precede a i+1 en el código) si implementa adelantamiento de LOADs a STORES ESPECULATIVO

```
(i)     sw 0(r5), r2     // M(r5)<-- r2
(i+1)   lw r4, 0(r6)     // r4 <--M(r6)
```

RESPUESTA: VERDADERO. No se puede saber si r5 y r6 apuntan a direcciones solapadas hasta que no se obtengan los correspondientes valores al ejecutar el código. Por eso para que se pueda producir el adelantamiento, el procesador debe implementar adelantamiento especulativo.