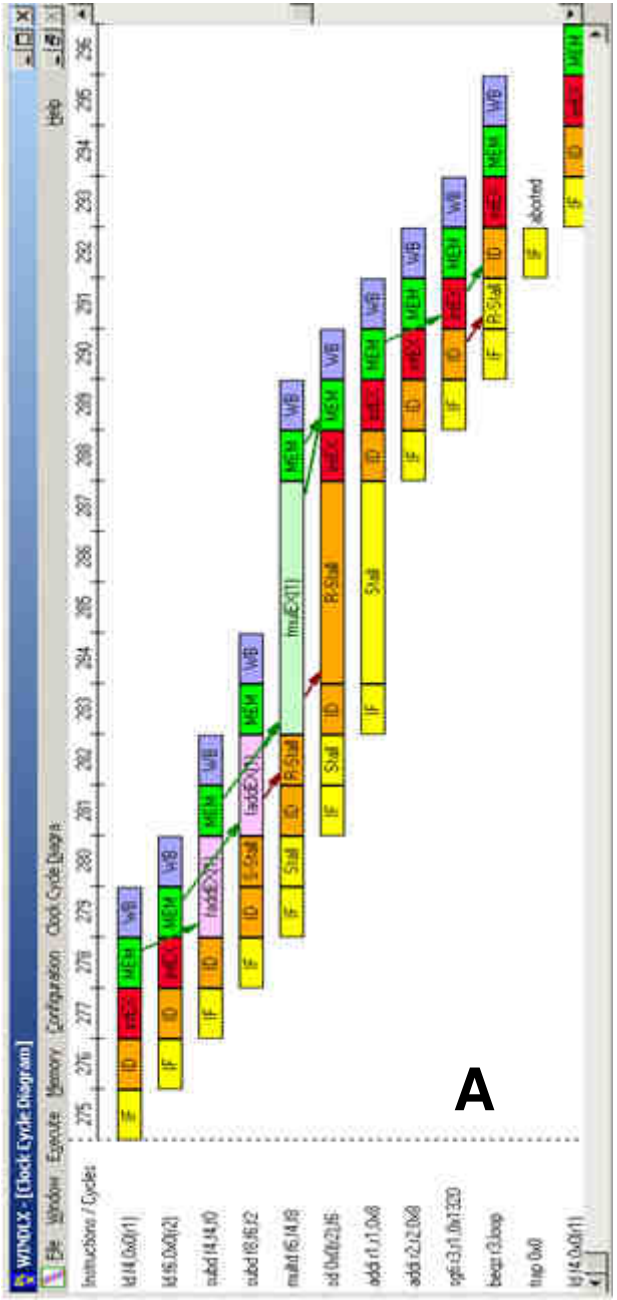
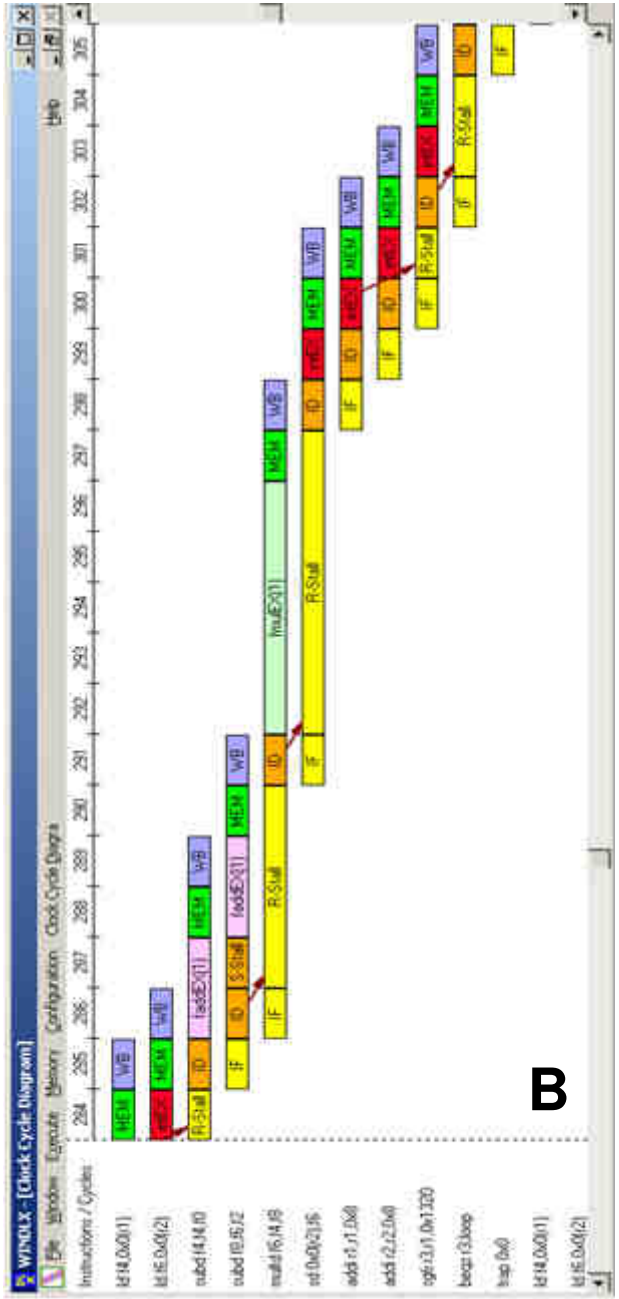
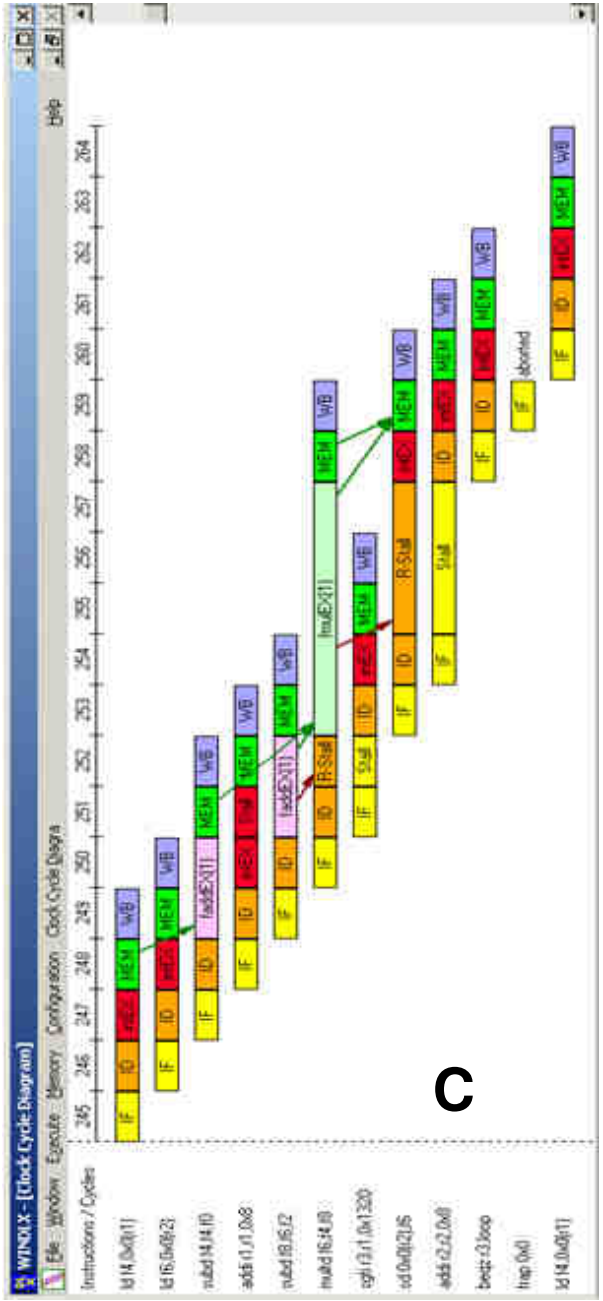
Las gráficas que vienen a continuación corresponden a tres simulaciones realizadas con Windlx para el bucle principal (100 iteraciones) de un cierto programa.







En función del comportamiento observado, responder a las tres cuestiones que siguen:

— Indica el nombre de las técnicas implícitas utilizadas en cada una de las simulación anteriores que justifican las mejoras en los respectivos tiempos de ejecución

	Técnica 1	Técnica 2
Simulación A	Adelantamiento	
Simulación B		
Simulación C	Adelantamiento	Reordenación

Explicación: Aplicando lo estudiado en las practicas realizadas con el simulador Windlx y con las imágenes de cada simulación, se pueden deducir las técnicas que se usan en cada una de ellas

— Cuántos ciclos por iteración de parada hay en la simulación original (sin técnica alguna aplicada) debido a cada tipo de riesgo?

Dependencias de Datos	14
Estructurales	1
Riesgos de Control	1

Explicación: Aplicando lo estudiado en las practicas realizadas con el simulador Windlx y con las imágenes de cada simulación, se calculan los ciclos de parada provocados por cada tipo de riesgo. Hay que tener en cuenta que la parada provocada por los riesgos estructurales queda incluida en la pérdida global por dependencia de datos de la instrucción mult f4,f6,f8

— ¿Cuántos ciclos se ahorran en el total de iteraciones debido exclusivamente a la técnica hardware?

700

Explicación: El adelantamiento se aplica de manera exclusiva en la simulación A. El número de ciclos que se pierden por iteración en ella es 7, con lo que ahorramos 14 – 7 = 7 ciclos / iteración y por lo tanto 700 ciclos en total

— ¿Cuántos ciclos se ahorran en el total de iteraciones debido exclusivamente a la técnica software (una vez aplicada la técnica hardware)?

300

Explicación: En la simulación C se realiza reordenamiento. Hay 4 ciclos de perdida por iteración y respecto a la simulación A ahorramos 3 ciclos / iteración, es decir, 300 ciclos en total

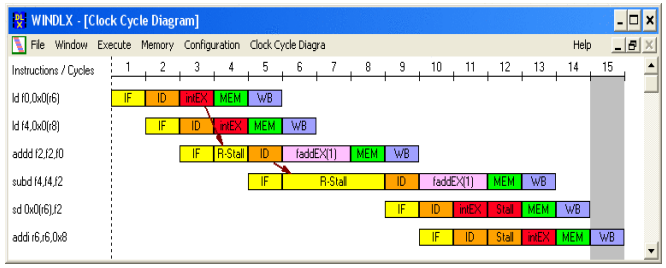
— Si el tiempo de ejecución en la simulación más rápida es de 1270 ciclos ¿qué ganancia final se ha conseguido con la aplicación conjunta de las citadas técnicas?

1,787

Explicación: La simulación más rapida es la C, y la original la B. La diferencia de perdida de ciclos entre la simulación B y la simulación C es de 10 ciclos por iteración según las gráficas. Así pues, la simulación B tendrá un tiempo de ejecución total de 2270 ciclos y la ganancia será entonces 2270/1270

A continuación se muestra el código interno de un bucle ejecutado durante 10^7 iteraciones por un procesador segmentado como el estudiado en clase (unidad flotante NO segmentada) y también un cronograma correspondiente a su simulación en ciertas condiciones.

```
1  ld f0, 0(r6)
2  ld f4, 0(r8)
3  addd f2, f2, f0
4  subd f4, f4, f2
5  sd 0(r6), f2
6  addi r6,r6, 8
```



En función del comportamiento observado responder a las tres cuestiones que siguen:

— ¿Cuántos ciclos se pierden por iteración al ejecutar el citado código

A) sin adelantamiento?
B) con adelantamiento?

A: 4 B: 1

Explicación: El cronograma representado corresponde al caso sin adelantamiento y en el se observa la perdida todal de 4 ciclos por iteración, 1 por la dependencia entre las instrucciones 1 y 3 y otros 3 por la dependencia entre las instrucciones 3 y 4 (el ciclo de parada en la instrucción 5 no supone un ciclo perdido, ya que no impide que la instrucción finalice un ciclo después de la que le precede). En caso de adelantamiento desaparecerían los ciclos perdidos por dependencias y tan solo persistiría un ciclo de pérdida debido al problema estructural que suponen las dos intrucciones flotantes consecutivas con unidades de ejecución de dos ciclos de latencia.

— Proponer una reordenación del código que, en combinación con el adelantamiento, elimine la perdida de ciclos sin incrementar el número de registros utilizados. Responder con una lista de 6 números identificadores de instrucción separados por comas.

1,2,3,5,4,6

Explicación: La única reordenación "legal" (que no adultera los resultados de la ejecución) capaz de eliminar el ciclo perdido aún con adelantamiento, es aquella que sitúa la penúltima instrucción entre las dos flotantes.

— Si el procesador es un MIPS 3000 a 50 MHz. ¿cuánto tiempo de ejecución extra supone cada ciclo perdido por iteración?

0,2 seg.

Explicación: El tiempo extra será el resultado de multiplicar 1 ciclo por el número total de iteraciones y por el tiempo que supone cada uno de esos ciclos.

❑ A continuación se muestran las instrucciones que componen el cuerpo de un bucle dedicado a calcular el producto escalar de dos vectores residentes en memoria (el registro r10 sólo se utiliza como contador del bucle)

```
1 lw    r4,   0(r1)
2 lw    r5,   0(r2)
3 multu r6,   r4,   r5
4 addu  r3,   r3,   r6
5 addui r1,   r1,   4
6 addui r2,   r2,   4
7 subui r10,  r10,  1
```

Según el cronograma mostrado en la figura, la primera iteración del bucle terminaría en el ciclo 23 utilizando el simulador WinDLX configurado SIN adelantamiento.

— ¿En qué ciclo terminaría dicha iteración si activáramos el adelantamiento?

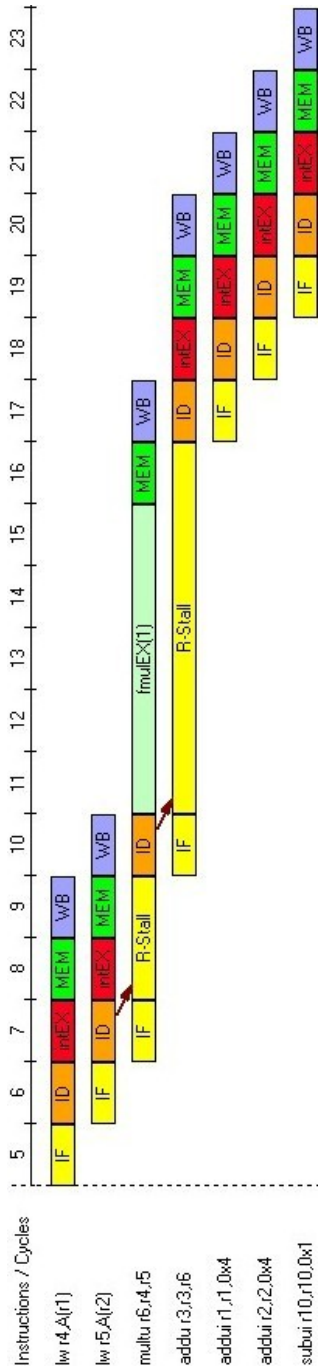
Terminaría en el ciclo 20

Explicación: La primera instrucción que detiene el cauce en la figura, presenta un riesgo de dependencia provocada por una instrucción de carga, que detiene el cauce 2 ciclos. Mediante la técnica de adelantamiento, se reduce la detención de 2 a 1 ciclo. En el segundo caso, la instrucción addu sólo tendría que esperar a que finalice la etapa EXE de la instrucción mult para obtener el valor adelantado en su etapa EXE, con lo que pasaría de perder 6 ciclos a perder solo 4. Estos 2 ciclos de ahorro sumados con el ciclo ahorrado por la primera dependencia hacen un total de 3 ciclos menos de ejecución por iteración. Si antes la instrucción subui terminaba en el ciclo 23, ahora terminara 3 ciclos antes, es decir, en el ciclo 20.

— Propón una reordenación del código que, con el adelantamiento activado, pierda 3 ciclos menos por iteración. Responder con una lista de 7 números identificadores de instrucción separados por comas

(1, 2, 7, 3, 5, 6, 4) ó (1, 2, 3, 5, 6, 7, 4)

Explicación: Para evitar perder el ciclo provocado por la dependencia existente entre las instrucciones 2 y 3, se inserta entre ellas una instrucción que no provoque nuevos riesgos (la última, que puede ser ejecutada en cualquier parte del bucle). Además, también se pierden ciclos entre las instrucciones 3 y 4, debido a que addu necesita el nuevo valor de r6. Se puede evitar perder dos ciclos insertando ahí las dos instrucciones encargadas de incrementar los punteros de los vectores, que la única restricción que tienen es que se deben ejecutar detrás de las instrucciones 1 y 2 respectivamente. Otra posible solución sería insertar solo las tres últimas instrucciones entre la 3 y la 4.



❑ La configuración de un procesador con arquitectura DLX (como la correspondiente a los simuladores vistos en prácticas) es la siguiente:

- Implementa la técnica de adelantamiento
- MEM y WB soportan 2 instrucciones simultaneas
- Unidad de suma flotante SEGMENTADA de 2 ciclos de latencia

Dado el siguiente código fuente:

```
1 or    r2,   r3,   r4
2 addd  f0,   f2,   f4
3 subd  f6,   f0,   f8
4 and   r5,   r2,   r6
5 xor   r8,   r2,   r10
```

— Dibujar el cronograma que resultaría al ejecutarlo en dicho procesador hasta el ciclo en que se ejecute la etapa WB de la instrucción **xor**.

	1	2	3	4	5	6	7	8	9	10	11	12
or	IF	ID	EXE	MEM	WB							
addd		IF	ID	fEXE1	fEXE2	MEM	WB					
subd			IF	ID	fEXE1	fEXE2	MEM	WB				
and				IF	ID	EXE	MEM	WB				
xor					IF	ID	EXE	MEM	WB			

— Propón una reordenación del código que evite la pérdida de ciclos en el caso de que la Unidad flotante no fuese segmentada. Responder con una lista de 5 números identificadores de instrucción separados por comas.

Cualquiera que separe las dos intrucciones flotantes