Considera el siguiente código como parte de un bucle principal (10^7 iteraciones) de un cierto programa que se ejecuta sobre un procesador MIPS 3000 que no incorpora adelantamiento.

```
        (...)

    (1) subui r30, r30, 1
    (2) add r2, r2, r6
    (3) lw r4, 0(r2)
    (4) add r2, r4, r2
    (5) sw 0(r4), r2

        (...)
```

Responder a las cuatro siguientes cuestiones:

— Cuántos ciclos perdidos por iteración tendríamos al ejecutar este código?

6 ciclos

Explicación: Existen tres dependencias entre datos que provocan 2 ciclos perdidos cada uno.

— Si ahora se activa el adelantamiento, ¿Cuántos ciclos perdidos por iteración tendríamos al ejecutar en este caso?

1 ciclo

Explicación: El adelantamiento elimina todos los ciclos perdidos por dependencias de datos excepto en las provocadas por la instrucción de carga que pasa a perder 1 ciclo por iteración

— Proponer una reordenación del código que minimice la perdida de ciclos sin incrementar el número de registros utilizados. Responder con una lista de 5 números identificadores de instrucción separados por comas.

2, 3, 1, 4, 5

Explicación: La única instrucción que podemos mover es la (1). Se coloca entre la instrucción de carga y la siguiente para aprovechar el ciclo de perdida que provoca la dependencia crítica entre ambas.

— Si la ejecución del código sin adelantamiento consume un tiempo de 4,25 segundos y tiene una perdida de 6 ciclos por iteración y con adelantamiento y reordenación se eliminan las perdidas de ciclos y tarda 3 segundos ¿cuál será la frecuencia del procesador?

48 MHz. ($6*10^7$ / 1.25)

Explicación: Dado que la diferencia es de 6 ciclos por iteración, y con ello se ahorran 1,25 segundos en el total de las 10^7 iteraciones, se calcula la frecuencia como: NºCiclosTotales/Tiempo.

❑ Durante un exótico viaje por la India, un profesor del Departamento de Informática adquirió en un bazar un lote de procesadores segmentados a cambio de 2 camellos. Este procesador, el Khali V, venía sin documentación alguna. Aunque se sabe que su arquitectura es muy similar a la MIPS R3000, desconocemos alguna de sus características que nos gustaría averiguar. Para ello, compilamos (sin optimizaciones) y ejecutamos los siguientes programas anotando el tiempo de ejecución empleado para cada uno de ellos:

```
1    .text
2    .globl main
3  main:
4    add   $30, $0, 10000000
5  bucle:
6    add.d $f0, $f2, $f4
7    sub.d $f6, $f8, $f10
8    nop
9    nop
10   nop
11   nop
12   nop
13   nop
14   subu $30, $30, 1
15   bne  $30, 0, bucle
16   j    $31
17
18
```

Listado 1.

```
1    .text
2    .globl main
3  main:
4    add   $30, $0, 10000000
5  bucle:
6    add.d $f0, $f2, $f4
7    nop
8    sub.d $f6, $f8, $f10
9    nop
10   nop
11   nop
12   nop
13   nop
14   subu $30, $30, 1
15   bne  $30, 0, bucle
16   j    $31
17
18
```

Listado 2.

```
1  loop:
2    .text
3    .globl main
4  main:
5    add   $30, $0, 10000000
6  bucle:
7    add.d $f0, $f2, $f4
8    nop
9    nop
10   sub.d $f6, $f8, $f10
11   nop
12   nop
13   nop
14   nop
15   subu $30, $30, 1
16   bne  $30, 0, bucle
17   j    $31
18
19
```

Listado 3.

```
1    .text
2    .globl main
3  main:
4    add   $30, $0, 10000000
5  bucle:
6    add.d $f0, $f2, $f4
7    nop
8    nop
9    nop
10   sub.d $f6, $f8, $f10
11   nop
12   nop
13   nop
14   subu $30, $30, 1
15   bne  $30, 0, bucle
16   j    $31
17
18
```

Listado 4.

```
1    .text
2    .globl main
3  main:
4    add   $30, $0, 10000000
5  bucle:
6    add.d $f0, $f2, $f4
7    nop
8    nop
9    nop
10   nop
11   sub.d $f6, $f8, $f10
12   nop
13   nop
14   subu $30, $30, 1
15   bne  $30, 0, bucle
16   j    $31
17
18
```

Listado 5.

```
1    .text
2    .globl main
3  main:
4    add   $30, $0, 10000000
5  bucle:
6    add.d $f0, $f2, $f4
7    nop
8    nop
9    nop
10   nop
11   nop
12   sub.d $f6, $f8, $f10
13   nop
14   subu $30, $30, 1
15   bne  $30, 0, bucle
16   j    $31
17
18
```

Listado 6.

Los tiempos de ejecución obtenidos son:

Programa	Tiempo de ejecución
1	14s
2	13s
3	12s
4	11s
5	11s
6	11s

— ¿Cuántos ciclos de latencia tiene la unidad de suma flotante si sabemos que NO está segmentada?

4

Explicación: Como la unidad de suma flotante no está segmentada, una instrucción de suma (o resta) no podrá hacer uso de ella si está siendo utilizada por otra instrucción. En los programas 4, 5 y 6 se supone que estas instrucciones están lo suficientemente alejadas como para que la segunda ya no tenga que esperar para hacer uso de la unidad de suma de flotantes. Siendo el programa 4 el primero que no necesita esperar, se puede determinar que el número de ciclos de latencia es 4 (el ciclo en que la instrucción addd entra en la etapa floatEXE y los tres siguientes ocupados por instrucciones de relleno nop).

— ¿Está activado el salto retardado? ¿Por qué?

No. Si estuviera activado, la instrucción que sigue al salto condicional se ejecutaría y finalizaría el programa. De este modo, se ejecutaría una sola iteración del bucle, lo que daría lugar a un tiempo de ejecución despreciable (el comando time nos devolvería un tiempo de ejecución de 0,0s).

— ¿Cuál es la frecuencia del reloj del procesador?

10MHz

Explicación: Según las ejecuciones de los programas, un ciclo adicional en el bucle, es decir, 10 millones de ciclos adicionales en toda la ejecución, requiere 1 segundo. No hace falta hacer ninguna cuenta, efectuar 10 millones de ciclos por segundo implica una frecuencia de procesador de 10Mhz.