1. **(6 puntos)** Se dispone de un procesador MIPS superescalar de **dos vías** y que posee gestión dinámica de instrucciones con especulación hardware. El lanzamiento de las instrucciones se realiza en orden y alineado. Las instrucciones atraviesan las siguientes etapas: IF (búsqueda de instrucciones), I (decodificación y lanzamiento de las instrucciones), En (ejecución en el operador multiciclo correspondiente), WB (escritura en los buses comunes de datos; duración de la transferencia 1 ciclo) y C (confirmación de las instrucciones y actualización del predictor, en su caso). El ROB tiene 32 entradas.

   Las características de las unidades funcionales son los siguientes:

	Nº Operadores	Latencia	Características
Carga/Almacenamiento	2	2	4 buffers de lectura y 4 de escritura, **no segmentados**
Cálculo de direcciones	2	1	Utiliza los mismos buffers de lectura y escritura
Suma/Resta CF	1	2	6 estaciones de reserva, **segmentado**
Multiplicador CF.	1	4	4 estaciones de reserva, **segmentado**
Enteros/Saltos	2	1	8 estaciones de reserva

   Se dispone de un predictor de saltos del tipo *Branch Target Buffer* (BTB) de dos bits que ofrece la predicción al final del ciclo de búsqueda de la instrucción.

   Se pretende evaluar el comportamiento del procesador ante el siguiente fragmento de código, correspondiente a la multiplicación de un vector $\vec{V}_{1\times3}$ por una matriz $\vec{M}_{3\times2}$, $\vec{Z} = \vec{V} \times \vec{M}$.

```
    dadd r5, r4, 16
loop_i:
    l.d f0, zero(r0)
    dadd r2, r0, r3
    dadd r1, r0, v
    dadd r6, r1, #24
loop_j:
    l.d f1, 0(r1)
    l.d f2, 0(r2)
    mul.d f3, f1, f2
    add.d f0, f0, f3
    dadd r2, r2, #16
    dadd r1, r1, #8
    bne r1, r6, loop_j
end_j:
    s.d f0, 0(r4)
    dadd r4, r4, #8
    dadd r3, r3, #8
    bne r4, r5, loop_i
end_i:
    trap #0
```

   Al comenzar la ejecución del bucle, todas las estructuras internas del procesador se encuentran vacias, salvo la tabla del predictor BTB que contiene el salto `bne r1,r6,loop_j` en estado *Weakly Not Taken (01)* y el salto `bne r4,r5,loop_i` en estado *Strongly Taken (11)*. Todos los registros valen `0`, salvo los registro `r3` y `r4` que apuntan a M y Z respectivamente. El contenido de la memoria es el siguiente:

M	M + 8	M + 16	M + 24	M + 32	M + 40	V	V + 8	V + 16	Z	Z + 8	zero
1.0	4.0	2.0	5.0	3.0	6.0	2.0	4.0	6.0	0.0	0.0	0.0

   Se solicita:

   a) Dibujar el diagrama instrucciones-tiempo de **la primera iteración** hasta el ciclo en que la instrucción de salto `bne r1,r6,loop_j` alcanza la etapa Commit, inclusive. Mostrar sólo las instrucciones que quepan en el diagrama adjunto. Utilizad la siguiente notación para las etapas de ejecución: load/store, AC, L1 y L2; suma/resta c.f, A1 y A2; multiplicación c.f. M1, M2, M3 y M4; enteras y saltos, E1.

   b) Determinar la penalización por fallo para el salto `bne r1,r6,loop_j`, indicando el número de ciclos de penalización y desde qué ciclo a qué ciclo se podrían considerar *ciclos de parada* o ciclos perdidos.

   c) Teniendo en cuenta que el ROB comienza con la entrada `#0` y que la etapa Issue siempre utiliza la primera estación de reserva o buffer disponible de cada tipo, determinar el contenido de los campos `value` y `rob`

de los registros `f0`, `f1`, `f2` y `f3` al finalizar el ciclo en que la instrucción `mul.d f3,f1,f2` hace commit. Al final de dicho ciclo, ¿cuántas entradas del *reorder buffer* estarán ocupadas? ¿Qué estaciones de reserva, buffers de lectura y escritura estarán ocupadas? Indicad en la tabla adjunta con una **X** las que estén ocupadas.

Solución:

a) Diagrama instrucciones-tiempo de **la primera iteración** hasta que el salto `bne r1,r6,loop_j` corrije el estado de la predicción.

```
PC     Instruc.          1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19
0      dadd r5,r4,#16    IF I  E1 WB C
loop_i l.d f0,zero(r0)   IF I  AC L1 L2 WB C
8      dadd r2,r0,r3        IF I  E1 WB    C
12     dadd r1,r0,#48       IF I  E1 WB       C
16     dadd r6,r1,#24          IF I     E1 WB C
loop_j l.d f1,0(r1)            IF I     AC L1 L2 WB C
24     l.d f2,0(r2)               IF I  AC L1 L2 WB C
28     mul.d f3,f1,f2             IF I              M1 M2 M3 M4 WB C
32     add.d f0,f0,f3                IF I                           A1 A2 WB C
36     dadd r2,r2,#16                IF I  E1 WB                          C
40     dadd r1,r1,#8                    IF I  E1 -  WB                       C
44     bne r1,r6,loop_j                 IF I           E1 WB                 C
end_j  s.d f0,0(r4)                        IF I  AC                          x
52     dadd r4,r4,#8                       IF I  E1 WB                       x
56     dadd r3,r3,#8                          IF I  E1 WB                    x
60     bne r4,r5,loop_i                       IF I     E1 WB                 x
0      dadd r5,r4,#16                            IF X
loop_i l.d f0,zero(r0)                           IF I  AC L1 L2 WB          x
8      dadd r2,r0,r3                                IF I  E1 WB               x
12     dadd r1,r0,#48                               IF I  E1 WB               x
16     dadd r6,r1,#24                                  IF I     E1 WB         x
loop_j l.d f1,0(r1)                                    IF I     AC L1 L2 WB   x
24     l.d f2,0(r2)                                       IF I  AC L1 L2 -  WB x
28     mul.d f3,f1,f2                                     IF I                 X
32     add.d f0,f0,f3                                        IF I              x
36     dadd r2,r2,#16                                        IF I  E1 WB       x
40     dadd r1,r1,#8                                            IF I  E1 -  WB x
44     bne r1,r6,loop_j                                         IF I          X
end_j  s.d f0,0(r4)                                                IF I  AC    x
52     dadd r4,r4,#8                                               IF I  E1 -  X
```

b) Determinar el contenido de los campos `value` y `rob` de los registros `f0`, `f1`, `f2` y `f3` al finalizar el ciclo en que la instrucción `mul.d f3,f1,f2` hace commit. Al final de dicho ciclo, ¿cuántas entradas del *reorder buffer* estarán ocupadas? ¿Qué estaciones de reserva estarán ocupadas?

Registro	F0	F1	F2	F3
value	0.0	2.0	1.0	2.0
rob	# 23	# 20	# 21	# 22

El número de entradas del ROB ocupadas serían 19: desde la 8 con la instrucción `add.d f0,f0,f3` a la 26 con la instrucción `bne r1,r6,loop_j`.
Las estaciones de reserva y buffers ocupado serán:

Est. Reserva	e1	e2	e3	e4	e5	e6	e7	e8	a1	a2	a3	a4	m1	m2	m3	m4
ocupado	S	S	S						S	S				S		

Buffer Lect. / Escrit.	l1	l2	l3	l4	s1	s2	s3	s4
ocupado		S	S		S			

c) Determinar la penalización por fallo para el salto `bne r1,r6,loop_j`, justificando la respuesta.

```
PC     Instruc.          1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19
40     dadd r1,r1,#8                    IF I  E1 -  WB                       C
44     bne r1,r6,loop_j                 IF I           E1 WB                 C
end_j  s.d f0,0(r4)                        IF I  AC                          x
52     dadd r4,r4,#8                       IF I  E1 WB                       x
                                         |                                   |
                                         |-------------- 13 ciclos ----------|
```

Los ciclos desde el 7 al 19 se pueden considerar ciclos de parada.

2. **(3.5 puntos)** El siguiente diagrama i–t muestra la ejecución de un fragmento de bucle que ha sido optimizado con la técnica de *loop unrolling*:

```
                        1  2  3  4  5  6  7  8  9  10 11 . . . 39 40 41 42 43 44 45 46 47 48 49 50
loop:   l.d f1, 0(r1)   IF I  AC L1 L2 L3 L4 L5 L6 L7 L8       L36L37WB C
        l.d f2, 8(r1)      IF I  AC                                  L1 WB C
        l.d f3,16(r1)         IF I  AC                               L1 WB C
        l.d f4,24(r1)            IF I  AC                               L1 WB C
        mul.d f1, f0, f1            IF I                                M1 M2 M3 WB C
        mul.d f2, f0, f2               IF I                                M1 M2 M3 WB C
        mul.d f3, f0, f3                  IF I                                M1 M2 M3 WB C
        mul.d f4, f0, f4                     IF I                                M1 M2 M3 WB C
        .                                       .
        .                                          .
        .                                             .
```

El procesador tiene soporte *hardware* para ejecución fuera de orden y especulación, puede lanzar una instrucción por ciclo de reloj y dispone de un operador de carga no segmentado y uno de multiplicación segmentado para operandos de coma flotante. Dispone de un único nivel de cache L1. La memoria principal está basada en tecnología DDR3 y su frecuencia de reloj coincide con la del núcleo del procesador.

Se pide:

a) Justificar el diferente comportamiento que tienen la primera y la segunda instrucciones de carga.

b) ¿Cuál es el tiempo de acceso (en ciclos de reloj) a la cache de datos L1 en caso de acierto (sin incluir el retardo de la etapa de calculo de la dirección de memoria, etapa AC)?

c) ¿Cuál es la penalización por fallo?

d) En la segunda iteración del bucle (no mostrada en el diagrama), la primera instrucción de carga (`l.d f1,0(r1)`) reduce el tiempo para ejecutarse completamente de 42 a 18 ciclos. ¿A qué se debe ese comportamiento? El resto de instrucciones consumen el mismo número de ciclos que los mostrados en el diagrama.

¿Cuál es el valor de $t_{RP} + t_{RCD}$ para los módulos DIMM de memoria?

e) ¿Cuál es el tamaño de bloque de cache en bytes?

Solución:

a) Justificar el diferente comportamiento que tienen la primera y la segunda instrucciones de carga.
La primera carga falla en el caché de datos de primer nivel y tiene que acceder al siguiente nivel en la jerarquía hasta que encuentre los datos. Luego, trae un bloque de caché a la memoria caché de datos de primer nivel. La segunda carga accede a la siguiente palabra de 64 bits en el mismo bloque de caché. Por lo tanto, acierta en el caché de primer nivel y la latencia de acceso es mucho más corta.

b) ¿Cuál es el tiempo de acceso (en ciclos de reloj) a la cache de datos L1 en caso de acierto (sin incluir el retardo de la etapa de calculo de la dirección de memoria, etapa AC)?
Desde la segunda y siguientes instrucciones de carga, se necesita un ciclo para acceder al caché de datos de primer nivel.

c) ¿Cuál es la penalización por fallo?
La primera carga requiere 37 ciclos para acceder a la memoria caché de datos de primer nivel y luego acceder a la memoria principal. Como el tiempo de acceso a la memoria caché es de un ciclo, se necesitan 36 ciclos para acceder a la memoria y llevar el bloque correspondiente a la memoria caché.

d) En la segunda iteración del bucle (no mostrada en el diagrama), la primera instrucción de carga (`l.d f1,0(r1)`) reduce el tiempo para ejecutarse completamente de 42 a 18 ciclos. ¿A qué se debe ese comportamiento? El resto de instrucciones consumen el mismo número de ciclos que los mostrados en el diagrama. ¿Cuál es el valor de $t_{RP} + t_{RCD}$ para los módulos DIMM de memoria?

En primer lugar, tenga en cuenta que el predictor de saltos predecirá incorrectamente la condición de salto en la primera iteración, por lo que tendrá que vaciar el ROB e iniciar de nuevo la búsqueda de instrucciones. Por lo tanto, la razón de la menor cantidad de ciclos se tiene que encontrar fuera del procesador.
Antes de ejecutar la primera iteración de bucle, el búfer de fila correspondiente en la memoria principal tenía la fila incorrecta abierta. Por lo tanto, la primera carga en la primera iteración tuvo que cerrar la fila incorrecta, abrir la correcta y acceder al bloque solicitado.
La primera carga en la segunda iteración falla en la memoria caché y tiene que acceder a la memoria principal. Sin embargo, esta vez, la fila correcta está abierta y la penalización por fallo es mucho más corta (12 ciclos en lugar de 36). La diferencia entre ambos accesos es $t_{RP} + t_{RCD}$. Por tanto, $t_{RP} + t_{RCD}$ es 24 ciclos.

e) ¿Cuál es el tamaño de bloque de cache en bytes?
Como la primera carga en la segunda iteración falla en la caché de nuevo, las cuatro cargas en la primera iteración consumieron todo el bloque de caché. Cada carga trae un valor de 64 bits (8 bytes). Por lo tanto, el tamaño del bloque es de 32 bytes.

□

3. **(0.5 puntos)**

La memoria principal de un computador, basada en tecnología DDR3, tiene la misma frecuencia de reloj que el núcleo del procesador. Tiene un sólo nivel de cache L1.

Calcular el tiempo medio de acceso a memoria si el tiempo en caso de acierto en la cache L1 es de 1 ciclo, la tasa de fallos de la cache L1 es del $10\,\%$, el parámetro *memory locality* es $ML = 40\,\%$ y la penalización por fallo es de 16 ciclos de reloj cuando la fila abierta es la buscada y de 40 ciclos en caso contrario.

Solución:

The average memory access time is:

$$T_{\text{access}} = Ht_{L1} + MR_{L1} \times MP_{L1}$$

$$T_{\text{access}} = 1 + 0{,}1 \times ((1 - 0{,}4) \times 40 + 0{,}4 \times 16) = 4{,}04 \text{ cycles}$$

□