Estructura de Computadores

Problemas Tema 2: Segmentación

1. Sea el programa "suma1" dado por el siguiente código MIPS:

```
        sub $5, $0, $0
suma:   lw $10, 1000($20)
        add $5, $5, $10
        addi $20, $20, -4
        bne $20, $0, suma
```

Se pide:

a) Describir brevemente la tarea realizada por "suma1"

b) Detectar las dependencias que afectan a "suma1" en el MIPS segmentado en 5 etapas y clasificarlas en dependencias de datos y dependencias de control. Supondremos que todos los saltos se resuelven en la etapa de memoria.

c) Gestión de dependencias por parte del compilador (software):

I. Permutar el orden de las sentencias de "suma1" dando un nuevo programa, "suma2", en el que las dependencias de datos tengan el menor efecto posible sobre el rendimiento del MIPS segmentado.

II. Suponer un MIPS segmentado sin ningún tipo de soporte hardware para solventar los problemas derivados de los riesgos de datos que conlleva la segmentación. Insertar en "suma2" el mínimo número de códigos de no-operación (nop) para producir una nueva versión, "suma3", que pueda ejecutarse correctamente en este MIPS.

III. Evaluar la mejora obtenida hasta ahora con respecto al punto de partida. Para ello, comparar el rendimiento que ofrece "suma3" al ejecutarse sobre el MIPS anterior con respecto al de "suma1" sobre el MIPS sin segmentar de la implementación monociclo. Suponer un número N de iteraciones en ambos programas y un cauce con etapas perfectamente balanceadas para el caso segmentado. Para las dependencias de control, suponer que se predice salto no realizado.

d) Gestión hardware de las dependencias:

I. Realizar una traza de ejecución del programa suma1, en un MIPS con hardware para la detención del cauce por dependencias de datos, anticipación dentro del banco de registros y predicción de salto no realizado, ilustrando los conflictos que se producen, así como la forma de resolverlos.

Instrucción	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
sub $5, $0, $0																								
lw $10, 1000($20)																								
add $5, $5, $10																								
addi $20, $20, -4																								
bne $20, $0, suma																								
…																								
…																								

II. Ídem al apartado anterior, pero suponiendo un MIPS dotado con hardware para la habilitación de **todos** los caminos de anticipación de datos entre las etapas.

Instrucción	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
sub $5, $0, $0																		
lw $10, 1000($20)																		
add $5, $5, $10																		
addi $20, $20, -4																		
bne $20, $0, suma																		
…																		
…																		

III. Evaluar el rendimiento de los dos sistemas anteriores con respecto al MIPS sin segmentar de la implementación monociclo y compararlos entre sí. Suponer un número N de iteraciones en ambos programas y un cauce con etapas perfectamente balanceadas para el caso segmentado. Aplicar las mismas suposiciones que en el apartado c).III sobre el número de iteraciones y cauce con etapas balanceadas.

	suma1 monociclo	**suma1 detención**	**suma1 anticipación**
Número de ciclos			
CPI			

2. Considerar la estructura segmentada en cinco etapas del procesador MIPS, con hardware para la detección de riesgos por dependencias de datos e inserción de burbujas para garantizar la correcta ejecución de los programas. Respecto a las dependencias de control, el controlador implementa la suposición de salto no realizado. Todos los saltos se resuelven en la etapa de memoria. El banco de registros permite la lectura y escritura simultánea de un mismo registro sin conflicto. Sea el siguiente programa:

```
      addi $5, $0, 0
etiq: lw $11, 3000($22)
      add $5, $5, $11
      and $11, $11, $5
      sw $11, 3000($22)
      subi $22, $22, 4
      bne $22, $0, etiq
      subi $22, $22, 4
      sw $5, 3000($22)
```

Se pide:

a) Realizar un diagrama temporal multiciclo donde se vea la evolución del código a través del cauce segmentado. Mostrar tanto la posibilidad de éxito en el salto como la contraria.

b) Suponer que el bucle se efectúa 1000 veces, es decir, el salto tiene éxito 999 veces seguidas y a continuación no tiene éxito. Bajo estas condiciones, ¿cuál sería el CPI para ese trozo de código?

CPI resultante:

Instrucción	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
addi $5, $0, 0																							
lw $11, 3000($22)																							
add $5, $5, $11																							
and $11, $11, $5																							
sw $11, 3000($22)																							
subi $22, $22, 4																							
bne $22, $0, etiq																							
subi $22, $22, 4																							
sw $5, 3000($22)																							

Instrucción	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
addi $5, $0, 0																							
lw $11, 3000($22)																							
add $5, $5, $11																							
and $11, $11, $5																							
sw $11, 3000($22)																							
subi $22, $22, 4																							
bne $22, $0, etiq																							
subi $22, $22, 4																							
sw $5, 3000($22)																							

c) ¿Cómo influiría la existencia de una unidad de anticipación en la evolución del cauce? ¿Cuál sería el nuevo CPI?

Instrucción	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
addi $5, $0, 0																								
lw $11, 3000($22)																								
add $5, $5, $11																								
and $11, $11, $5																								
sw $11, 3000($22)																								
subi $22, $22, 4																								
bne $22, $0, etiq																								
subi $22, $22, 4																								
sw $5, 3000($22)																								

CPI resultante:

3. Sobre la misma arquitectura MIPS del ejercicio anterior, considerar esta vez el siguiente segmento de código:

```
      or $3, $0, $2
ite:  add $4, $2, $3
      and $7, $3, $4
      beq $3, $2, fin
      bne $4, $7, ite
fin:  lw $3, 100($7)
```

Responder a las mismas cuestiones del ejercicio anterior, pero esta vez analizando la semántica del programa para determinar el número de veces que salta. Las respuestas pueden completarse rellenando las tablas que aparecen a continuación:

Sin anticipación

Instrucción	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
or $3, $0, $2																							
add $4, $2, $3																							
and $7, $3, $4																							
beq $3, $2, fin																							
bne $4, $7, ite																							
lw $3, 100($7)																							

CPI resultante sin anticipación:

Con anticipación

Instrucción	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
or $3, $0, $2																							
add $4, $2, $3																							
and $7, $3, $4																							
beq $3, $2, fin																							
bne $4, $7, ite																							
lw $3, 100($7)																							

CPI resultante con anticipación:

4. Partimos de la estructura del procesador MIPS vista en clase, a la que se le incorpora una unidad para la detección de riesgos de datos, que se solventarán insertando burbujas. Establezcamos la modificación de fusionar la memoria de datos e instrucciones en una memoria común. Teniendo en cuenta este cambio, responder a las siguientes cuestiones:

   a) ¿Qué nuevos riesgos y/o conflictos aparecen durante la ejecución simultánea de instrucciones en las distintas etapas del cauce?
   b) ¿Cómo modificarías la unidad de detección de riesgos y el camino de datos para solventar estos nuevos riesgos?
   c) Mostrar, para este nuevo sistema, la evolución del cauce cuando se ejecuta el siguiente código y calcular su CPI.

```
lw   $8, 3000($7)
sub  $4, $4, $5
sw   $5, 3000($8)
add  $3, $4, $5
and  $5, $4, $5
sub  $9, $4, $0
or   $8, $8, $5
muli $5, $8, 100
```

5. Sea una arquitectura RISC con un conjunto de instrucciones similar al del procesador MIPS visto en clase. Su camino de datos presenta una segmentación en 3 etapas y una única vía de acceso a memoria, común para instrucciones y datos. Esto hace que las instrucciones se ejecuten según se muestra a continuación:

Instrucción	1	2	3	4	5
I1	IFD	REX	MEW		
I2		IFD	REX	MEW	
I3			IFD	REX	MEW

donde:

- IFD es la etapa de búsqueda y decodificación de instrucción.
- REX es la etapa de búsqueda de operandos, ejecución de operación ó resolución de condiciones de salto y cálculo de dirección para operando destino ó salto.
- MEW es la etapa de acceso a memoria para instrucciones de carga/almacenamiento, escritura de resultados en registros para instrucciones aritmético/lógicas y actualización del PC para instrucciones de salto.

Además no podrá leerse en un mismo ciclo un registro que va a ser escrito en dicho ciclo (es decir, no hay anticipación en el banco de registros). Bajo estas condiciones, responder a las siguientes cuestiones:

a) ¿Qué tipo de riesgos pueden darse? Poner ejemplos.
b) Considérese que se introduce una unidad de detección de riesgos. ¿Qué efecto causa la inserción de esta unidad en la evolución del cauce? Justificar la respuesta y poner ejemplos.
c) ¿Qué ocurriría con las instrucciones de salto?
d) Supóngase que al procesador se le dota de una unidad de detección de riesgos y el controlador aplica la suposición de salto no realizado. Representar en un diagrama multiciclo la evolución del cauce para el siguiente trozo de código. Mostrar las dos posibilidades de resolución de salto. Calcular el CPI de ese trozo de código en ambos casos.

```
        sw  $5, 3000($7)
        or  $5, $4, $5
        sub $8, $4, $7
        lw  $4, 3000($8)
        bne $7, $0, etiq
        add $8, $8, $5
etiq:   sw  $5, 3000($8)
```

6. Betta, una empresa dedicada a la fabricación de microprocesadores segmentados tipo MIPS, quiere saber si tendría éxito un nuevo procesador antes de empezar a fabricarlo. Hasta ahora, con el modelo M1, trataban los riesgos de control con detención del cauce. En el nuevo modelo, el M2, han introducido una mejora en el hardware, la de suponer los saltos no realizados. Ambas estrategias se corresponden fielmente con las vistas en clase para el MIPS.

   De los programas que usan como bancos de prueba, tienen la siguiente información en cuanto a la probabilidad con lo que aparece cada tipo de instrucción:

   - 4% de llamadas a procedimientos.
   - 3% de saltos incondicionales.
   - 25% de saltos condicionales, de los cuales en el 42% la condición que se evalúa es cierta.
   - El resto son operaciones aritmético-lógicas y de transferencias de datos.

   Se ha utilizado el mismo compilador tanto para M1 como para M2, pero en M2 se ha conseguido doblar la frecuencia de funcionamiento del procesador. Los riesgos de datos no hay que tenerlos en cuenta, ya que van resueltos en el mismo procesador.

   Con todos estos datos, responder a las siguientes cuestiones:

   a) ¿Qué funcionalidad debe realizar el hardware que se ha añadido en M2 para el tratamiento de saltos?
   b) Teniendo en cuenta el modo en que se resuelven los riesgos de datos en M1 y M2, ¿cuáles son las estrategias que se pueden haber implementado?
   c) Calcular el CPI de ambos procesadores para los programas de prueba.
   d) Por una encuesta de mercado, dados los costes adicionales que tiene la fabricación de M2, éste solo tendría éxito si ejecutara los programas al menos en la mitad de tiempo que su antecesor, el M1. Suponiendo que no te interesa la ruina de Betta, ¿aconsejarías a ésta la fabricación de M2 o esperar a que se mejore el diseño? Razonar la decisión tomada (si faltan datos, indicar cuáles y decir para qué valores convendría fabricar y para cuáles no).

7. Sobre la misma arquitectura MIPS de los ejercicios anteriores, con **todos** los caminos de anticipación ya incluidos, considerar esta vez el siguiente segmento de código:

```
cambia:    xor $4, $4, $5
           xor $5, $4, $5
           xor $4, $4, $5
           jr $31
inicio:    lw $4, 1000($0)
           lw $5, 1004($0)
           beq $4, $5, sigue
           jal cambia
sigue:     sw $4, 1008($0)
fin:       # fin del programa
```

   y asumir que las posiciones de memoria 1000, 1004 y 1008 almacenan palabras cuyo valor es 0000BEB0, 0000CAFE y 0000DED0, respectivamente (en hexadecimal). Realizar el diagrama de ciclos y calcular su CPI, teniendo en cuenta que el programa comienza su ejecución en la línea etiquetada con inicio y termina cuando se completa la última instrucción antes de alcanzar la posición etiquetada con fin.

Instrucción	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
xor $4, $4, $5																							
xor $5, $4, $5																							
xor $4, $4, $5																							
jr $31																							
lw $4, 1000($0)																							
lw $5, 1004($0)																							
beq $4, $5, sigue																							
jal cambia																							
sw $4, 1008($0)																							
fin:																							

CPI resultante:

8. Considerar la estructura segmentada en cinco etapas del procesador MIPS, con hardware para la detección de riesgos por dependencias de datos, inserción de burbujas para garantizar la correcta ejecución de los programas, y con **todos** los caminos de anticipación posibles habilitados. Respecto a las dependencias de control, los saltos condicionales e incondicionales se resuelven por completo durante la etapa de **decodificación** y el controlador implementa la suposición de salto no realizado. El banco de registros permite la lectura y escritura simultánea de un mismo registro sin conflicto. Para el siguiente segmento de código:

```
      ori  $2, $0, 1000h
loop: lw   $1, 2800h($2)
      sub  $4, $1, $0          rotar: add  $10, $4, $4
      jal  rotar                      muli $7, $10, 2
      sw   $7, 7800h($2)              jr $31
      sw   $1, C800h($2)
      subi $2, $2, 4
      bne  $2, $0, loop
```

   a) Señalar las dependencias de datos y control que pueden producir riesgos en un MIPS segmentado con anticipación en el banco de registros, bajo la hipótesis de salto no realizado. Para cada dependencia de datos indicar entre que instrucciones se produce y cuál es el operando implicado.
   b) Mostrar en un diagrama de ciclos la evolución del cauce segmentado desde el comienzo (instrucción `ori`) hasta que la instrucción `lw` termina de ejecutarse completamente por segunda vez.

9. Queremos mejorar el rendimiento del procesador MIPS visto en clase, reduciendo el impacto de los riesgos de control. Para ello introducimos un comparador en la etapa ID que se encarga de realizar la comparación de los valores de los registros que se lean en el banco de registros, para el caso de instrucciones de salto condicional. Además, en esa misma etapa añadimos otra ALU que se encarga de calcular la dirección efectiva de salto, en caso de que hubiera que saltar. Por lo tanto en esta arquitectura los saltos se resuelven completamente al final de la etapa ID.

a) Con esta estrategia, ¿se eliminan totalmente los riesgos de control? Razona la respuesta.
b) En esta nueva arquitectura hemos habilitado todos los caminos de anticipación de la arquitectura MIPS vista en clase: EXE-EXE, MEM-MEM, MEM-EXE. ¿Existe alguna situación en la que sea conveniente introducir un nuevo camino de anticipación? Pon un ejemplo.
c) Para evaluar el rendimiento de esta nueva arquitectura utilizamos un benchmark de prueba en el que el 15% de las instrucciones son saltos condicionales, de los cuales el 80% son saltos realizados. El 5% de las instrucciones son saltos incondicionales y llamadas a procedimientos. El 20% son instrucciones tipo load el 10% instrucciones tipo store y el resto instrucciones entre registros. Ignorando los riesgos por dependencias de datos, calcula la aceleración que se consigue con este benchmark, comparando la nueva arquitectura con la arquitectura del MIPS vista en clase, cuando en ambas se aplica predicción de salto tomado.

10. Los diseñadores del MIPS han decidido incorporar un nuevo modo de direccionamiento para especificar los operandos fuente de las instrucciones aritmético-lógicas. En concreto, se van a permitir instrucciones aritmético-lógicas en las que uno de los operandos fuente puede especificar una posición de memoria. Por ejemplo, es válida la instrucción:

    add $2, $3, 5Ch($4)

    donde comprobamos que el operando destino es el registro 2, el primer operando fuente es el registro 3 y el segundo operando fuente es el dato que está en la posición de memoria dada por $4+5Ch, es decir la instrucción anterior realiza la operación $2 ← $3 + MEM[5Ch+$4].

    Sin embargo, introducir nuevos modos de direccionamiento supone cambiar la arquitectura del procesador MIPS segmentado, así que sus diseñadores deciden modificarlo para que ahora incorpore 6 etapas, tal como se indica en la siguiente figura:

Instrucción	1	2	3	4	5	6	7
I1	IF	ID	A1	M	A2	WB	
I2		IF	ID	A1	M	A2	WB

    donde cada etapa tiene el siguiente significado:

- IF representa la etapa de búsqueda de instrucción.
- ID representa la etapa de decodificación y en ella se accede al banco de registros.
- A1 representa una nueva etapa con un sumador que se encarga de calcular una dirección efectiva de memoria, si la instrucción lo requiere. Esto ocurrirá en las instrucciones lw y sw, así como en las instrucciones aritmético lógicas en las que uno de los operandos fuentes es un dato que está en memoria (tal y como ocurre en la instrucción add de la figura anterior). En caso en que la instrucción no necesite acceder a memoria, esta etapa no tiene ningún efecto, aunque se consume el ciclo de reloj (tal y como ocurre en la instrucción or de esa misma figura);
- M representa la etapa de acceso a memoria.
- A2 representa la etapa de ejecución, donde se ubica la ALU del procesador.
- WB es la etapa de post-escritura, en la que se accede al banco de registros.

NOTA: Las etapas IF, ID, M, A2 y WB son idénticas a las etapas IF, ID, MEM, EX y WB del procesador MIPS visto en clase.

a) Si suponemos anticipación en el banco de registros, ¿qué cortocircuitos propondrías para reducir los riesgos por dependencias de datos? Pon un ejemplo de cada tipo.
b) Calcula el CPI del siguiente fragmento de código, suponiendo que se ejecuta en el nuevo procesador MIPS de 6 etapas con anticipación en el banco de registros y con el HW necesario para implementar los caminos de anticipación que hayas propuesto en el apartado anterior. Indica claramente los cortocircuitos.

Instrucción	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
`addi $9, $0, 0`																										
`add $10, $0, 58($9)`																										
`muli $11, $10, 4`																										
`lw $3, 0($11)`																										
`sw $3, 0($10)`																										
`sw $3, 4($10)`																										
`lw $4, 4($11)`																										
`sub $9, $9, 0($4)`																										

CPI resultante: [ ]

(exprésalo en forma fraccionaria)

c) Los cortocicuitos que has propuesto, ¿Eliminan totalmente los riesgos por dependencias de datos? ¿Hay algún caso en el que sea necesaria la detención del cauce?

11. Los diseñadores del procesador MIPS deciden intentar mejorar el rendimiento de la arquitectura. En concreto, se centran en el problema de los riesgos de control. Para ello proponen realizar la carga del PC con la dirección de la instrucción a la que se salta en la misma etapa en la que se sabe si se salta o no. Por ejemplo, las instrucciones de salto incondicional saben que se salta al final de la etapa ID (decodificación), mientras que las instrucciones de salto condicional pueden saber si se salta o no al final de la etapa EX (ejecución). Para evaluar el rendimiento de la nueva máquina, deciden utilizar un conjunto de benchmarks de prueba. De estos benchmarks conocen la siguiente información:
    - El 20% son instrucciones del tipo lw/sw
    - El 40% son instrucciones aritmético-lógicas
    - El 25% son instrucciones de salto condicional, de las que el 80% son saltos que se realizan.
    - El 5% son instrucciones de salto incondicional
    - El resto son instrucciones de llamadas a procedimientos

    Responder a las siguientes cuestiones:

    a) Con este rediseño en la arquitectura, ¿se eliminan los riesgos de control? Razona tu respuesta.

b) Calcula el rendimiento de la nueva arquitectura MIPS respecto de la arquitectura clásica del MIPS visto en clase, suponiendo que en ambos casos se aplica una estrategia de predicción de salto no realizado.

c) Calcula el rendimiento de la nueva arquitectura MIPS respecto de la arquitectura clásica del MIPS visto en clase, suponiendo ahora que en ambos casos se aplica una estrategia de predicción de salto realizado.

d) En vista de los resultados anteriores, ¿qué estrategia de predicción es la más efectiva? ¿Por qué?

12. Los diseñadores del procesador MIPS se han planteado reducir el coste del sistema. Para ello, deciden utilizar una memoria de datos más barata. El problema es que la latencia de esta memoria es de 2 ciclos de reloj. Para incorporar esta nueva memoria en el diseño segmentado, dividen la etapa de memoria original del MIPS (M) en dos etapas: M1 y M2. Por lo tanto, el camino de datos del nuevo procesador segmentado hace que las instrucciones se ejecuten según se muestra a continuación:

Instrucción	1	2	3	4	5	6	7
I1	IF	ID	EX	M1	M2	WB	
I2		IF	ID	EX	M1	M2	WB

En este nuevo diseño las instrucciones `lw` y `sw` completan su acceso a memoria al finalizar la etapa M2. Se ha decidido además activar todos los caminos de anticipación posibles, y que exista anticipación en el banco de registros (es decir, en un mismo ciclo puede escribirse y leerse un mismo registro). En cuanto a los saltos, se resuelven por completo durante la etapa de **decodificación** y el controlador implementa la suposición de salto no realizado. Bajo estas condiciones, responde a las siguientes cuestiones:

a) ¿Entre qué etapas pueden activarse cortocircuitos? Pon ejemplos ilustrativos.

b) Suponiendo que la unidad de control aplica la estrategia de predicción de salto no realizado, representa en la tabla la evolución del cauce para la ejecución completa del siguiente trozo de código. Calcular el CPI total de este código.

```
      addi $2, $0, 16
buc:  lw $3, 0($2)
      lw $4, 4($2)
      add $5, $3, $4
      sw $5, 100($2)
      subi $2, $2, 8
      bne $2, $0, buc
      add $6, $3, $4
```

Instrucción	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
`addi $2,$0,16`																										
`lw $3, 0($2)`																										
`lw $4, 4($2)`																										
`add $5,$3,$4`																										
`sw $5,100($2)`																										
`subi $2,$2,8`																										
`bne$2,$0,buc`																										
`add $6,$3,$4`																										
`---`																										

13. Los diseñadores del procesador MIPS se centran ahora en el problema de los riesgos de control. Se han percatado de que se pierde un alto número de ciclos cuando se implementa una estrategia de predicción estática, tanto salto tomado como salto no tomado. Por eso deciden implementar una estrategia de predicción dinámica. Para ello, se decide incorporar en el controlador una tabla llamada BTB. La BTB sólo

contiene información sobre las instrucciones de salto que se han ejecutado hasta ese momento. Esta tabla es una memoria asociativa con 3 campos: el primer campo identifica el PC de una instrucción de salto (PCInst), el segundo campo identifica la dirección destino de salto de esa instrucción (DirDest) y el tercer campo es un bit de predicción (Pred). Si este bit vale 1 es que la última vez que se ejecutó esa instrucción de salto, se saltó. Si vale 0 es porque la última vez que se ejecutó no se saltó. Se le llama bit de predicción porque depende del valor que tenga, se saltará o no al volver a ejecutarse esa instrucción.

El funcionamiento de la BTB es el siguiente:

- Durante la etapa de búsqueda de instrucción (IF)) el controlador consulta la BTB.
    - Si no hay ninguna entrada en la BTB cuyo campo PCInst coincida con el PC de la instrucción que se está buscando actualmente, es porque no se trata de una instrucción de salto, o bien es un salto que todavía no se ha ejecutado. En este caso, en el siguiente ciclo de reloj, entrará en el cauce la siguiente instrucción del programa.
    - Si, por el contrario, hay alguna entrada en la BTB cuyo campo PCInst coincida con el PC actual, es porque se trata de una instrucción de salto que se ha ejecutado antes. En este caso se chequeará el bit de predicción, Pred.
        - Si Pred=0, el controlador asumirá que hay que aplicar la estrategia de salto no tomado, y por lo tanto en el siguiente ciclo de reloj, entrará en el cauce la siguiente instrucción del código.
        - Si Pred=1 el controlador asumirá salto tomado, y en el siguiente ciclo de reloj, entrará en el cauce la instrucción destino de salto (se sabe cuál es porque en la tabla está guardada la dirección destino DirDest).
- La instrucción continuará ejecutándose por las etapas ID, EX, M y WB. Si se trata de una instrucción de salto, en la etapa de EX se habrá calculado la condición de salto (es decir, si hay que saltar o no) y se podrá comprobar si la predicción ha sido correcta o no:
    - Si la predicción ha sido correcta, es decir, la predicción fue salto tomado (o salto no tomado) y la condición de salto indica que hay que saltar (o no saltar) entonces la predicción ha acertado y no hay que hacer nada más.
    - Por el contrario, si la predicción falló, es decir, se ejecutó salto tomado (o salto no tomado) y la condición de salto indica que no hay que saltar (o hay que saltar), en ese caso hay que actualizar el bit de predicción, Pred para que indique lo que ha ocurrido en esta ejecución de la instrucción de salto. Además, habrá que anular las instrucciones que han entrado en el cauce después de la instrucción de salto y cargar el PC con la dirección correcta. En el siguiente ciclo de reloj, por tanto, recomenzará la ejecución con la instrucción correcta que debe ejecutarse después de este salto.
    - Otra posibilidad es que sea la primera vez que se ejecuta esa instrucción de salto. Por tanto, la información de esta instrucción no estará en la BTB. En este caso, en la etapa de M habrá que actualizar la BTB con la información del PC de esta instrucción (PCInst), la dirección destino de salto (DirDest) y el bit de Predicción (Pred).

Suponiendo que se trabaja con el MIPS visto en clase, para el que se activan todos los caminos de anticipación posibles, y que en el banco de registros pueda leerse en un mismo ciclo el registro que

va a ser escrito en dicho ciclo (anticipación en el banco de registros), responde a las siguientes cuestiones:

a) Representa en la tabla adjunta la evolución del cauce hasta que se ejecute por tercera vez la instrucción `lw $3,0($2)` para el código que se muestra a continuación. Indica claramente los cortocircuitos.

```
            addi $2, $0, 1000
bucle:      lw $3, 0($2)
            add $4, $3, $3
            sw $4, 1000($2)
            subi $2, $2, 8
            bne $2, $0, bucle
```

Instrucción	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
`addi $2,$0,1000`																								
`lw $3,0($2)`																								
`add $4,$3,$3`																								
`sw $4,1000($2)`																								
`subi $2,$2,8`																								
`bne $2,$0,bucle`																								
…																								
…																								

b) Sabiendo que la instrucción `bne $2,$0,bucle` se encuentra en la dirección `000A5624h`, rellena la siguiente tabla indicando, para cada iteración del código anterior, el valor que los campos DirDest y Pred tendrán en la BTB cuando se comienza a ejecutar esa instrucción en la iteración correspondiente. Fíjate que en la primera iteración aún no se ha ejecutado esa instrucción, por lo tanto en la BTB no hay información sobre esa instrucción.

	PCInst	DirDest	Pred
1ª iteración	---	---	---
2ª iteración	000A5624h		
3ª iteración	000A5624h		
Última iteración	000A5624h		

c) Calcular el CPI de la ejecución completa del código

CPI resultante:

14. Sea la versión del procesador MIPS vista en clase, en la que la terminación de los saltos es en la etapa de memoria, evaluándose la condición de salto en ejecución, y que para la resolución de las dependencias de control implementa la suposición de salto no realizado. Además, para la resolución de los riesgos de datos, dispone de hardware para la implementación de las estrategias de detención y anticipación, y adicionalmente implementa anticipación en el banco de registros. Para el siguiente fragmento de código:

```
        ori $4, $0, 10
loop:   add $7, $4, $1
        beq $7, $4, fin
        lw $1, 100($7)
        j loop
fin:    sw $1, 200($4)
        sub $5, $1, $4
```

a) Completa un diagrama multiciclo donde se muestre la evolución del código a través del cauce segmentado, suponiendo que en el salto condicional la primera vez no salta y la segunda sí.
b) Calcula el CPI suponiendo que se realizan 100 iteraciones, es decir, las 99 primeras veces la instrucción beq no salta, y en la última iteración sí (deja indicada la forma en que lo calculas).

CPI resultante:

Se decide modificar la arquitectura del procesador para anticipar los saltos en HW, de forma que se terminen en la etapa de decodificación.

c) ¿Qué cambios habría que introducir para ello, en el caso de la instrucción beq? Haz un esquema.
d) Para esta arquitectura con saltos anticipados en el HW, ¿hace falta habilitar cortocicuitos adicionales? Justifica tu respuesta.
e) Vuelve a hacer el apartado a) para esta segunda implementación del MIPS.

CPI resultante:

15. Considerar el siguiente programa MIPS (NUM es una constante declarada previamente):

```
1       addi $9, $0, NUM
2       sub $10, $10, $10
3       add $11, $0, $0
4 SA:   j SB
5       lw $4, 8000($10)
6 SB:   lw $3, 400($10)
7       lw $4, 560($10)
8       sub $3, $3, $4
9       sw $3, 1200($10)
10      add $11, $11, $3
11      sw $11, 1600($10)
12      addi $10, $10, 4
13      bne $9, $10, SB
14      sw $11, 400($10)
```

Suponemos que dicho código se ejecuta sobre el MIPS segmentado visto en clase. Para resolver los riesgos de datos el procesador dispone de todos los cortocircuitos necesarios (entre las etapas EX-EX, MEM-MEM y MEM-EX), existiendo además anticipación dentro del banco de registros. Para los riesgos de control se usa la predicción estática de salto no tomado. La condición de salto y el cálculo de la dirección de salto se computan en la etapa de ejecución, por lo que los saltos se resuelven por completo en dicha etapa.

a) Suponiendo que NUM=8, muestra cómo será la evolución completa del código anterior en el MIPS utilizando la tabla adjunta TABLA-P.2. Indica claramente cómo se resolverán los riesgos: cortocircuitos utilizados, burbujas insertadas en caso necesario.

b) Calcula el CPI (expresarlo en forma fraccionaria):

16. Trabajamos con el MIPS de 5 etapas visto en clase para el que sabemos que hay anticipación en el banco de registros y además se han habilitado los siguientes caminos de anticipación: EX-EX, MEMMEM, MEM-EX. Para tratar los riesgos de control se ha decidido que la dirección de salto se calcule en ID, mientras que la condición de salto se calcule en EX. De esta manera, la terminación de los saltos incondicionales es en ID, mientras que la de los saltos condicionales es en EX. Además se ha optado por implementar la estrategia de predicción estática de Salto no Tomado.

a) Para la arquitectura propuesta, ¿son los cortocircuitos propuestos suficientes para resolver todos los riesgos por dependencias de datos? Si no es así, indica qué cortocircuitos adicionales son necesarios, poniendo ejemplos e indicando si es necesario la inserción de burbujas en ID.

b) Para el siguiente código, para el que sabemos que la instrucción j loop sólo llega a ejecutarse la 1a vez, realiza la traza de ejecución indicando claramente los cortocircuitos y las burbujas con las que se resuelven los riesgos:

```
main :  addi $1 , $0 , 0
        addi $2 , $0 , 100
loop :  lw $3 , 0( $2 )
        beq $1 , $3 , f i n
        add $1 , $1 , $3
        addi $2 , $2 , 4
        j loop
fin:    sw $1 , 100( $2 )
```

c) Calcula asímismo el CPI de la ejecución completa (en forma fraccionaria):