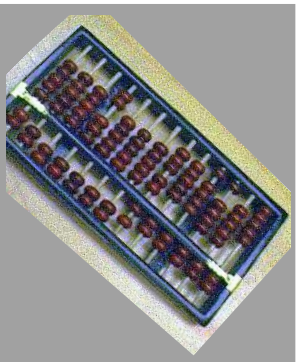
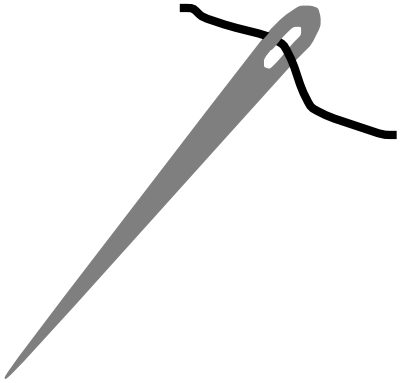


Arquitectura de Computadores



$$T = N \cdot CPI \cdot t_c$$

J.M. Llabería
E. Herrada
A. Olivé

Contenidos

Capítulo 4 Técnicas para tolerar o reducir la latencia efectiva de la segmentación

4

TÉCNICAS PARA TOLERAR O REDUCIR LA LATENCIA EFECTIVA DE LA SEGMENTACIÓN

En este capítulo se presentan técnicas para soportar o reducir la latencia efectiva de la segmentación. El objetivo es disminuir el tiempo de ejecución de una secuencia de instrucciones reduciendo los ciclos perdidos debido a riesgos de datos y secuenciamiento.

La primera técnica que se presenta se aplica de forma estática cuando se genera el código. El compilador se encarga de ordenar las instrucciones para tolerar la latencia efectiva, siendo su objetivo reducir los riesgos de datos.

Las siguientes técnicas que se presentan son mecanismos hardware. Para reducir la latencia efectiva de la segmentación de las instrucciones que manipulan datos se añaden, al camino de datos base, otros caminos de comunicación de datos entre etapas, que se denominan cortocircuitos y permiten disponer de un valor antes de que sea escrito en el elemento de almacenamiento.

Para reducir la latencia de la segmentación en las instrucciones de secuenciamiento, se modifica la segmentación diseñada en el Capítulo 3. También se presenta una técnica, denominada de predicción, que reduce los ciclos perdidos, aunque no disminuye la latencia efectiva de la segmentación.

Contenido

INTRODUCCIÓN

En el Capítulo 3 se ha mostrado la adecuación de la semántica del procesador a la semántica del lenguaje máquina, que es de tipo imperativo. Esta acomodación se efectúa emulando un funcionamiento secuencial cuando se detecta un riesgo. Los ciclos perdidos son debidos a la latencia efectiva de actualización de los elementos de almacenamiento. En la Figura 4.1 se muestra un ejemplo donde se muestra un riesgo de datos y un riesgo de secuenciamiento.

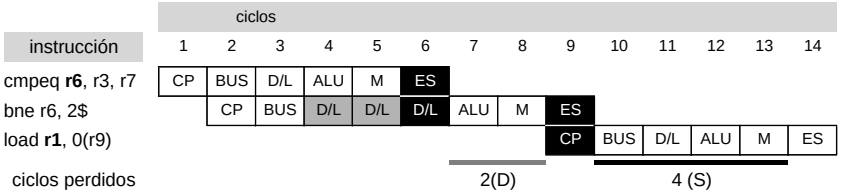


Figura 4.1 *Latencia efectiva en actualizar el banco de registros: 2 ciclos. Latencia efectiva en actualizar el registro CP: 4 ciclos.*

Por tolerar la latencia se entiende que una dependencia de datos entre dos instrucciones no ocasiona un riesgo de datos. En otras palabras, que entre la instrucción que produce el resultado y la instrucción que lo consume exista un número suficiente de ciclos de interpretación, para que la dependencia de datos no dé lugar a un riesgo de datos. En la Figura 4.2 se muestra una secuencia de 5 instrucciones y tres posibles formas de ordenarlas que calculan el mismo resultado. Cuando se interpreta la ordenación A, en el procesador descrito en el Capítulo 3, se detectan 2 riesgos de datos y se pierden 4 ciclos. Cuando se interpreta la ordenación B se detecta un riesgo de datos y se pierde 1 ciclo. Cuando se interpreta la ordenación C no se detecta ningún riesgo de datos y por tanto, no se pierde ningún ciclo.

A	B	C
add **r4**, r1, r3	add **r4**, r1, r3	add **r4**, r1, r3
sub **r5**, r2, r4	sub **r9**, r8, r6	sub **r9**, r8, r6
sub **r9**, r8, r6	sub **r5**, r2, r4	cmple **r12**, r13, r14
add **r10**, r0,r9	add **r10**, r0,r9	sub **r5**, r2, r4
cmple **r12**, r13, r14	cmple **r12**, r13, r14	add **r10**, r0, r9

Figura 4.2 *Tres ordenaciones de una secuencia de instrucciones que calculan el mismo resultado.*

Por reducir la latencia efectiva se entiende poder utilizar un resultado antes de que se escriba en el elemento de almacenamiento. Por ejemplo, en la Figura 4.3 se muestra la interpretación de dos instrucciones. La latencia efectiva de actualización del banco de registros es 3. Por ello, la siguiente instrucción detecta un riesgo de datos. Sin embargo, uno de los datos que quiere utilizar la segunda instrucción ya ha sido calculado durante el ciclo 4. Reducir la latencia efectiva es poner a disposición de la segunda instrucción el valor calculado. Para ello, se añaden caminos de comunicación de datos entre las etapas, que permiten utilizar un dato una vez ha sido producido y antes de que se escriba en el elemento de almacenamiento. Mediante uno de estos caminos de comunicación adicionales no se perderán ciclos en la interpretación de las dos instrucciones que se muestran en la Figura 4.3.

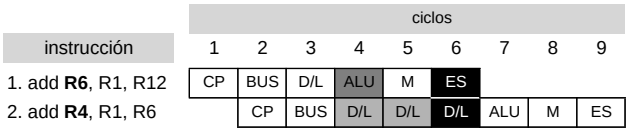


Figura 4.3 *Observación del instante en que se conoce el resultado calculado por una instrucción y el instante en que se actualiza el banco de registros.*

Por último, se describe una técnica donde se efectúa una hipótesis sobre el resultado de un cálculo, con el objetivo de no bloquear la interpretación de instrucciones cuando se detecta un riesgo. Esta técnica se denomina de predicción y se aplicará en las instrucciones de secuenciamiento condicional. En concreto se efectuará una hipótesis sobre el resultado de evaluar la condición.

PLANIFICACIÓN DE INSTRUCCIONES

Dada una secuencia ordenada de instrucciones pueden existir otras ordenaciones de la secuencia de instrucciones que obtengan el mismo resultado. Esta posibilidad se manifiesta construyendo el grafo de dependencias de la secuencia de instrucciones. Un grafo de dependencias expone los grados de libertad para ordenar las instrucciones y muestra la ordenación mínima para que el cálculo sea correcto.

En la Figura 4.4 puede verse el grafo de dependencias de la secuencia de instrucciones de la Figura 4.2. Las instrucciones a y c son fuente de una dependencia de datos verdadera debido a registros y las instrucciones destino son respectivamente b y d. Algunos ejemplos de ordenación de estas instrucciones se han mostrado en la Figura 4.2.

a. add **r4**, r1, r3
b. sub **r5**, r2, r4
c. sub **r9**, r8, r6
d. add **r10**, r0,r9
e. cmple **r12**, r13, r14

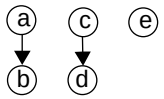


Figura 4.4 *Grafo de dependencias de una secuencia de instrucciones.*

En este apartado nos restringiremos a la ordenación de instrucciones en bloques básicos estáticos. Entendemos por estático que los bloques básicos se construyen cuando se compila el código.

Bloque básico estático (BB). Secuencia de instrucciones que satisface dos propiedades: a) ninguna instrucción, excepto tal vez la primera, es el destino de una instrucción de secuenciamiento y b) todas las instrucciones se interpretan si se interpreta la primera instrucción.

Para construir los BB de un código se siguen los siguientes pasos:

1. Encontrar los líderes, los cuales son la primera instrucción de un BB. Los líderes se identifican de la siguiente forma: a) la primera instrucción del código, b) la instrucción destino de una instrucción de secuenciamiento y c) una instrucción que sigue a una instrucción de secuenciamiento.
2. Dado un líder, el BB es el líder y todas las instrucciones que siguen en secuencia excluyendo el próximo líder.

Como las instrucciones de un BB se interpretan en secuencia, las operaciones que efectúan pueden representarse mediante un grafo de dependencias acíclico y dirigido.

En la Figura 4.5 se muestra una secuencia de instrucciones y el grafo de dependencias de datos del BB que constituye el cuerpo del bucle. En primer lugar describimos las dependencias debidas a registros. Las instrucciones a y b son fuente de una dependencia

de datos verdadera y la instrucción destino es la c. La instrucción c es fuente de una dependencia de datos verdadera y la instrucción d es el destino de la dependencia. La instrucción i es destino de una dependencia de datos verdadera cuya fuente es la instrucción h. Las instrucciones a, b y d son fuente, cada una de ellas, de una antidependencia siendo las instrucciones e, f y g respectivamente destino de la dependencia.

```
1$:  a. load r1, 0(r2)
     b. load r3, 0(r4)
     c. add r5, r1, r3
     d. store r5, 0(r6)
     e. add r2, r2, #8
     f. add r4, r4, #8
     g. add r6, r6, #8
     h. sub r9, r9, #1
     i. bne r9, 1$
```

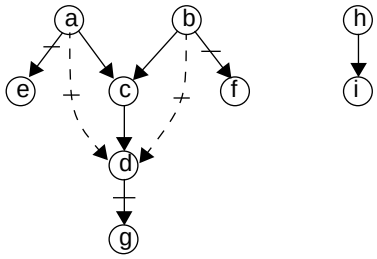


Figura 4.5 *Secuencia de instrucciones y grafo de dependencias de datos.*

Cuando no se conoce el contenido de los registros utilizados para calcular las direcciones efectivas de memoria, hay que ser conservador al determinar las dependencias debidas a memoria. Así, como no tenemos información sobre el contenido de los registros r2, r4 y r6 hay que suponer que las instrucciones a y b son fuente de una antidependencia y el destino es la instrucción d. Para identificarlas se han dibujado con una línea de trazos en el grafo de dependencias.

La instrucción de secuenciamiento siempre debe interpretarse al final de la secuencia de instrucciones del BB al que pertenece. Entonces, hay que añadir un nuevo tipo de dependencia en el grafo de dependencias, que se denomina de control. Nosotros usualmente no mostraremos explícitamente esta dependencia en el grafo de dependencias, aunque hay que tenerla en cuenta cuando se reordenen las instrucciones de un bloque básico que tiene como última instrucción una instrucción de secuenciamiento .

Cualquier ordenación topológica del grafo de dependencias acíclico es una planificación correcta de la secuencia de instrucciones. Entonces, la idea es utilizar la ordenación parcial que expresa el grafo de dependencias, para construir una ordenación

total de la secuencia de instrucciones, con el objetivo de que al interpretarse, en un camino de datos segmentado, se pierda el menor número posible de ciclos por riesgos de datos.

Un ejemplo de reordenación en la secuencia de código de la Figura 4.5 es ubicar la instrucción h como primera instrucción de la secuencia de instrucciones reordenada.

Planificación de instrucciones. Ordenación de la secuencia de instrucciones con el objetivo de reducir el tiempo de ejecución.

La planificación de instrucciones la efectúa el compilador y su función es separar la instrucción productora de un dato de la instrucción consumidora del dato, una distancia que elimine o reduzca los ciclos perdidos. Por separar dos instrucciones se entiende que entre ellas se ubican otras instrucciones independientes.

Latencia o retardo productor-uso. Número de ciclos que deben transcurrir entre una instrucción que produce un resultado y una instrucción que lo consume para que no exista riesgo de datos.

En la Figura 4.6 se muestra, mediante un diagrama temporal, el retardo productor-uso. Suponemos el procesador descrito en el Capítulo 3, donde en el mismo ciclo se puede escribir y leer, en este orden, un registro del banco de registros, siendo el retardo productor-uso de 3 ciclos.

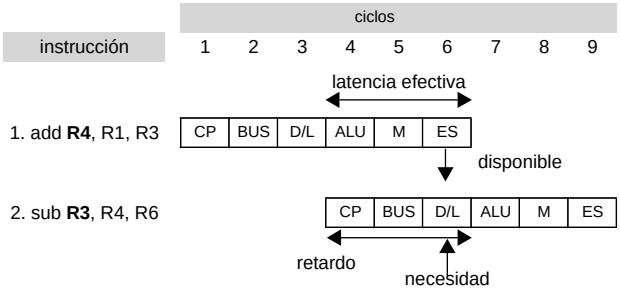


Figura 4.6 *Latencia efectiva y retardo productor-uso.*

Si la instrucción consumidora sigue en secuencia a la instrucción productora, el retardo productor-uso menos uno indica el número de ciclos perdidos.

Algoritmo de planificación

El problema de encontrar una planificación óptima en el caso de restricción de recursos es NP-completo. Entonces, se utilizan heurísticas para encontrar planificaciones que se aproximen a las óptimas en un tiempo prudencial.

El algoritmo que se describe supone un procesador segmentado lineal en el que se puede iniciar la fase de ejecución de una instrucción en cada ciclo.

Planificación mediante lista. Planifica las instrucciones en pasos. Utilizando una regla de selección, en cada paso se crea una lista de instrucciones que son elegibles para ser planificadas. Posteriormente, se aplica una segunda regla con el objetivo de efectuar la mejor selección, de una instrucción de la lista, con vistas a la próxima planificación.

Para efectuar la planificación se parte del grafo de dependencias (GD) del BB y los arcos se etiquetan con el retardo productor-uso menos uno en el caso de dependencias de datos. Las antidependencias y dependencias de salida se etiquetan con el valor cero, ya que el procesador lineal garantiza que no se producen riesgos si el destino de la dependencia se interpreta inmediatamente después de la fuente de la dependencia.

La instrucción de secuenciamiento, si existe, siempre debe planificarse en último lugar. Por ello, se añaden arcos adicionales etiquetados con el valor cero cuando la instrucción de secuenciamiento no depende directa o indirectamente de una instrucción debido a una dependencia de datos.

En la Figura 4.7 se muestra el grafo de dependencias de la secuencia de instrucciones mostrada en la Figura 4.5 con los arcos etiquetados como se ha descrito. Se han eliminado las antidependencias debidas a posiciones de memoria ya que por transitividad están cubiertas por otras dependencias. Recordemos que el retardo producto-uso son 3 ciclos para todas las instrucciones que actualizan el banco de registros.

La mayoría de bloques básicos tienen como última instrucción una instrucción de secuenciamiento. En caso contrario, el BB puede tener varias hojas (nodos de los cuales no depende ningún otro nodo). Entonces, se crea una hoja ficticia y se hacen confluir todas las hojas en la hoja ficticia.

```
1$:  a. load r1, 0(r2)
     b. load r3, 0(r4)
     c. add r5, r1, r3
     d. store r5, 0(r6)
     e. add r2, r2, #8
     f. add r4, r4, #8
     g. add r6, r6, #8
     h. sub r9, r9, #1
     i. bne r9, 1$
```

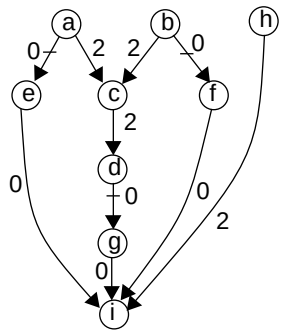


Figura 4.7 *Grafo de dependencias con los arcos etiquetados con el retardo producto-uso menos uno.*

En la Figura 4.8 se muestra un esquema del algoritmo de planificación de instrucciones que se describe seguidamente.

```
0. t = 0
1. Construir una lista de elegibles
        Condiciones de inclusión
            a) No tiene predecesor (raíces del árbol): incluir
            b) Tiene predecesor: incluir si todos los predecesores han sido planificados y si t ≥ TMC
2. Heurística de selección de un elemento de la lista
        Cálculo de la longitud del camino hasta el nodo hoja del BB
            Suma de las etiquetas de los arcos. Elegir el de mayor longitud.
3. Eliminar del grafo de dependencias el nodo seleccionado
4. Si quedan nodos por planificar
        t = t + 1
        Etiquetar con el TMC los sucesores directos del nodo seleccionado
            max [etiqueta actual (si existe), t + etiqueta del arco que los conecta]
        Volver a 1
```

Figura 4.8 *Esquema de un algoritmo de planificación de instrucciones. TMC se refiere al instante de tiempo más cercano en que se puede planificar un nodo.*

Construir una lista de elegibles. Para incluir un nodo en la lista debe cumplir una de las siguientes dos condiciones.

1. No debe tener predecesores. Esto es, no depende de nadie; son las raíces del árbol.
2. Si tiene predecesores, el dato producido por el predecesor está disponible en el caso de una dependencia verdadera

de datos o en los otros tipos de dependencias de datos la instrucción fuente ha sido planificada.

Las dos condiciones son obvias ya que el objetivo del proceso de planificación de instrucciones es no seleccionar una instrucción que hace perder ciclos.

La segunda condición requiere efectuar el cálculo del tiempo más cercano (TMC) en que se puede planificar un nodo sucesor directo de un nodo planificado, para que no se detecte un riesgo de datos. Para efectuar este cálculo se utiliza un contador de tiempo (t).

Para calcular el TMC de un sucesor directo de un nodo planificado se suma el contador de tiempo (t) y el valor de la etiqueta del arco que conecta el nodo planificado con el sucesor directo. El nodo sucesor directo se etiqueta con este tiempo si no estaba etiquetado y en caso contrario se etiqueta con el mayor de los valores, el valor calculado o el que había.

Entonces, un nodo se incluye en la lista de elegibles si todos los predecesores han sido planificados y el tiempo actual (t) es igual o mayor que el instante de tiempo más cercano en que puede planificarse.

Heurística de selección. Si hay más de un nodo elegible la idea es mirar el futuro. Se planificará la instrucción que puede producir más bloqueos si se retrasa su planificación. Para ello, se utiliza una aproximación del camino crítico (heurística). Se considera crítico el camino más largo hasta el nodo hoja del BB. Como longitud del camino se utiliza la suma de los valores de las etiquetas de los arcos que conectan los nodos a lo largo del camino considerado. Si, durante el cálculo, en un nodo confluyen varios valores de longitud se elige el mayor. El nodo que se planifica es el que tiene el camino más largo y si hay empate se planifica el nodo que aparece primero en el orden de programa original.

Una vez planificado un nodo se elimina del GD y el algoritmo incrementa en una unidad el contador de tiempo (t).

En la Figura 4.9 se muestra el GD en el instante t=0. En la parte derecha se muestra la lista de elegibles y la longitud del camino. En t = 0 se planifica el nodo a ya que tiene uno de los caminos más largos y es el primero en orden de programa.

El nodo a se elimina del GD, se calcula el TMC de los nodos sucesores directos y se incrementa en una unidad el contador de tiempo (t=1). Los nodos sucesores directos se etiquetan con el TMC calculado.

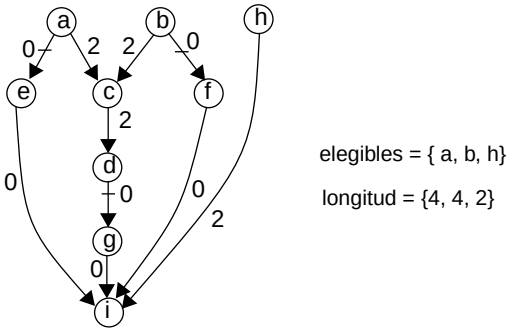


Figura 4.9 *Grafo de dependencias con los arcos etiquetados con el retardo producto-uso menos uno. Instante t=0 de la planificación.*

En la Figura 4.10 se muestra el etiquetado de los nodos después de haber planificado en t=0 el nodo a. En t=1 la lista de elegibles es {b, e, h} y las longitudes respectivas son {4, 0, 2}. Por tanto se planifica el nodo b. En t=2 la lista de elegibles es {e, f, h} y las longitudes respectivas son {0, 0, 2}. Por tanto se selecciona el nodo h. En t=3 la lista de elegibles es {e, f} y las longitudes son iguales. Por tanto se elige el nodo e.

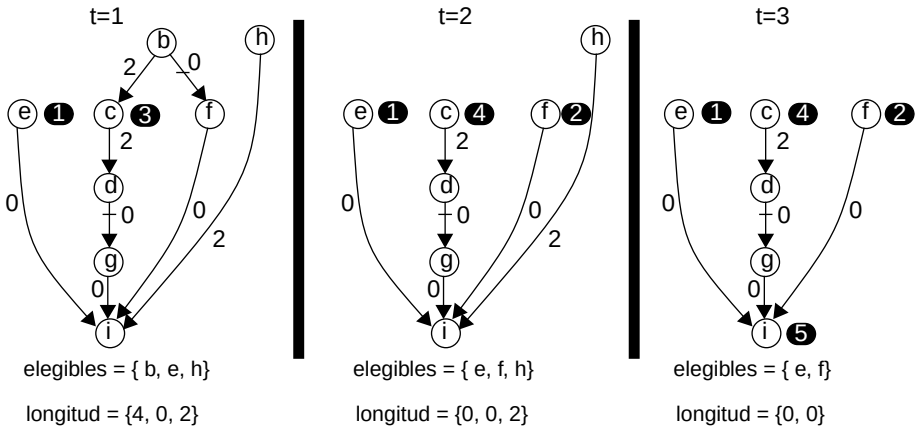


Figura 4.10 *Pasos desde t=1 hasta t=3 en el algoritmo de planificación de instrucciones.*

En t=4, Figura 4.11, la lista de elegibles es {c, f} y las longitudes respectivas son {2, 0}. Por tanto se planifica el nodo c. En t=5 la lista de elegibles es {f}. Por tanto se selecciona el nodo. En t=6, la lista de elegibles está vacía y no se planifica ningún nodo.

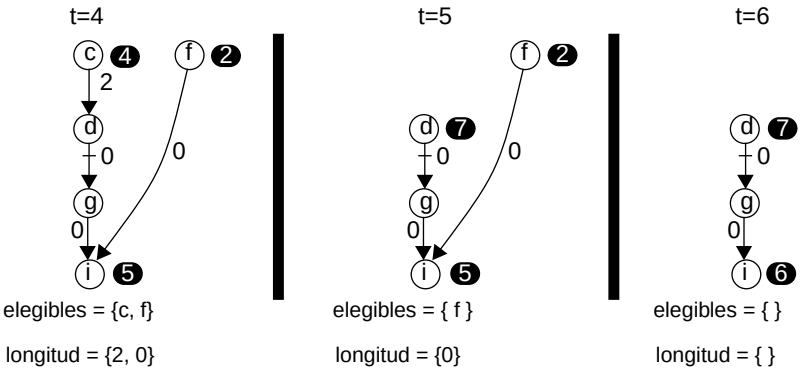


Figura 4.11 *Pasos desde t=5 hasta t=6 en el algoritmo de planificación de instrucciones.*

En t=7, Figura 4.12, la lista de elegibles es {d}. Por tanto se selecciona el nodo d. En t=8 se planifica el nodo g y en t=9 se planifica el nodo i.

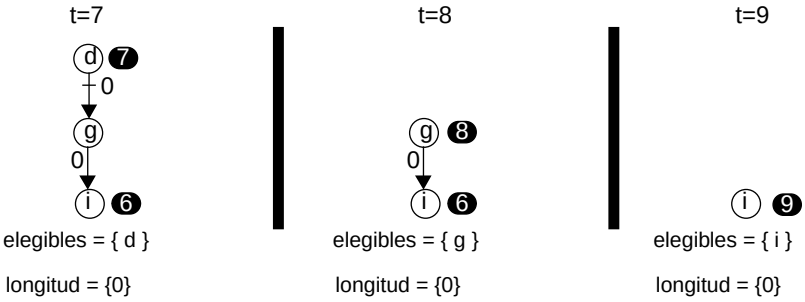


Figura 4.12 *Pasos desde t=7 hasta t=9 en el algoritmo de planificación de instrucciones.*

La secuencia de instrucciones después de planificar es: a, b, h, e, c, f, d, g, i. El número de ciclos en los que no se ha planificado ninguna instrucción es el número de ciclos perdidos por riesgos de datos, cuando se interpreta la secuencia de instrucciones, siendo en este ejemplo 1 ciclo.

Ejercicio

Muestre en sendos diagramas temporales la interpretación de una iteración de la secuencia de instrucciones de la Figura 4.5 utilizando: a) la ordenación original y b) la ordenación obtenida con el algoritmo de planificación. Calcule la ganancia cuando se utiliza el algoritmo de planificación.

Respuesta

En los siguientes diagramas temporales se muestra la interpretación de las dos secuencias de instrucciones.

A) Secuencia original.

	instrucción	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
		ciclos																				
a.	load **r1**, 0(r2)	CP	BUS	D/L	ALU	M	ES															
b.	load **r3**, 0(r4)		CP	BUS	D/L	ALU	M	ES														
c.	add **r5**, r1, r3			CP	BUS	D/L	D/L	D/L	ALU	M	ES											
d.	store r5, 0(r6)				CP	BUS	BUS	BUS	D/L	D/L	D/L	ALU	M	ES								
e.	add **r2**, r2, #8					CP	CP	CP	BUS	BUS	BUS	D/L	ALU	M	ES							
f.	add **r4**, r4, #8								CP	CP	CP	BUS	D/L	ALU	M	ES						
g.	add **r6**, r6, #8											CP	BUS	D/L	ALU	M	ES					
h.	sub r9, r9, #1												CP	BUS	D/L	ALU	M	ES				
i.	bne r9, 1$													CP	BUS	D/L	D/L	D/L	ALU	M	ES	
1$:	load **r1**, 0(r2)																				CP	BUS
	ciclos perdidos								2(D)			2(D)							2(D)			4(S)

Los ciclos de ejecución de una iteración del bucle son: 19

B) Secuencia reordenada.

	instrucción	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
		ciclos																		
a.	load **r1**, 0(r2)	CP	BUS	D/L	ALU	M	ES													
b.	load **r3**, 0(r4)		CP	BUS	D/L	ALU	M	ES												
h.	sub r9, r9, #1			CP	BUS	D/L	ALU	M	ES											
e.	add **r2**, r2, #8				CP	BUS	D/L	ALU	M	ES										
c.	add **r5**, r1, r3					CP	BUS	D/L	ALU	M	ES									
f.	add **r4**, r4, #8						CP	BUS	D/L	ALU	M	ES								
d.	store r5, 0(r6)							CP	BUS	D/L	D/L	ALU	M	ES						
g.	add **r6**, r6, #8								CP	BUS	BUS	D/L	ALU	M	ES					
i.	bne r9, 1$									CP	CP	BUS	D/L	ALU	M	ES				
1$:	load **r1**, 0(r2)															CP	BUS	D/L	ALU	M
	ciclos perdidos												1(D)						4(s)	

Los ciclos de ejecución de una iteración del bucle son: 14

La ganancia es

Ganancia = 19 / 14 = 1.36

CORTOCIRCUITOS

En el apartado anterior se ha descrito la técnica de reordenación de código, que explota el paralelismo a nivel de instrucción (instrucciones independientes entre sí), con el objetivo de reducir el número de veces que se detectan riesgos $R(i) \cap D(i+k) \neq \varnothing$, y por tanto, se reduce el CPI.

La aportación de la técnica de reordenación de código descrita es limitada, ya que se circunscribe a bloques básicos. El tamaño típico de un bloque básico es reducido (entre 4 y 9 instrucciones) y también suele existir poco paralelismo a nivel de instrucción. Entonces conviene utilizar otras técnicas que reduzcan o eliminen la penalización por riesgo $R(i) \cap D(i+k) \neq \varnothing$.

La idea es reducir la latencia efectiva de la segmentación y la técnica se denomina cortocircuito. Un cortocircuito es un hardware, adicional al camino de datos básico, que nos permite transportar datos y controlar la utilización de estos.

En este apartado nos limitamos a instrucciones que leen o escriben en el banco de registros. El caso de instrucciones de secuenciamiento, que actualizan el registro CP se describirá en un apartado dedicado exclusivamente a estas instrucciones.

Al observar el camino de datos segmentado se constata que existe un retardo entre el ciclo en el cual un dato está disponible y el ciclo en el cual se escribe el dato en el banco de registros. Como ejemplo, en intrucciones ENT el dato está disponible durante la etapa ALU y se escribe 2 ciclos después.

En la Figura 4.13 se muestra la interpretación segmentada de dos secuencias de instrucciones. La primera de la instrucciones calcula el registro R1 y las siguientes instrucciones leen R1. Observemos, en la parte izquierda de la figura, que en el ciclo 4 la 2ª instrucción está en la etapa D/L y el valor del contenido del registro R1 se está calculando en la etapa ALU, siendo conocido antes de finalizar el ciclo. Igualmente, en la parte derecha de la figura, cuando la 3ª instrucción está en la etapa D/L el contenido del registro R1 es conocido y está en la etapa M. Por tanto, conexiones (cortocircuitos) que transporten el valor desde la salida de las etapas ALU y M hasta la salida de la etapa D/L, permitirán utilizar el valor calculado antes de escribir en el banco de registros.

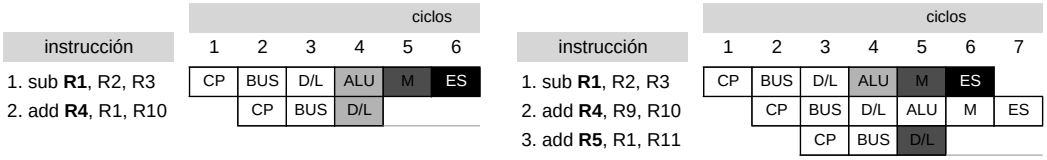


Figura 4.13 *Secuencias de instrucciones que muestran los instantes en que es conocido el valor que se almacena en un registro.*

Cortocircuito. Conexión (nivel físico) entre una etapa donde se dispone de un valor producido, y no escrito en el elemento de memorización del cual se lee, y una etapa donde el valor se utiliza como dato fuente de una instrucción.

Cortocircuito o atajo es el elemento hardware que se utiliza para efectuar una acción de avanzar o anticipar un dato a un consumidor, para que disponga del dato antes de que se escriba en el banco de registros o en la memoria de datos. Entonces, cualquiera de estas palabras se puede utilizar indistintamente para describir el concepto (nivel lógico).

En la Figura 4.14 se muestra una secuencia de instrucciones marcando con una flecha la utilización de un posible cortocircuito entre etapas y el ciclo que se efectuaría. El sentido de la flecha es de la etapa productora a la etapa consumidora. En el ciclo 4, el valor que se almacenará en el registro R1 se comunica desde la etapa ALU a la etapa D/L mediante un cortocircuito. Como la 2ª instrucción ya dispone del valor de los datos, se inicia la fase de ejecución en el siguiente ciclo. En el ciclo 5, el valor que se almacenará en el registro R1 se transporta desde la etapa M a la etapa D/L mediante otro cortocircuito y la 3ª instrucción inicia la fase de ejecución en el siguiente ciclo. Observe que las instrucciones 4ª y 5ª leen el dato del banco de registros.

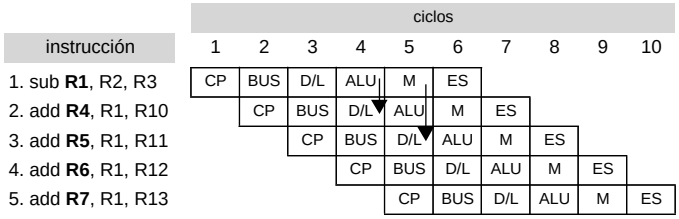


Figura 4.14 *Secuencia de instrucciones mostrando la comunicación de información cuando se utilizan cortocircuitos.*

Estas comunicaciones de datos se producen dentro del ciclo de la señal de reloj y con ellas se reduce la latencia efectiva de la segmentación en instrucciones que calculan el resultado en la ALU. La reducción de la latencia efectiva son 2 ciclos y en la secuencia de instrucciones utilizada como ejemplo no se pierden ciclos.

Cuando un cortocircuito resuelve un caso de riesgo de datos ya no existe situación de riesgo de datos.

Un cortocircuito crea un bucle hardware en el camino de datos, ya que se establece una comunicación entre una etapa avanzada en el proceso de interpretación y una etapa previa. Notemos que el objetivo de este tipo de bucles hardware es la eliminación de riesgos de datos y con ello la reducción de ciclos perdidos.

En un bucle hardware, creado para reducir la latencia efectiva de la segmentación, debe haber como mínimo un registro de desacoplo al recorrer el bucle. Además, como el dato que se suministra desde la etapa productora se utiliza en el mismo ciclo en la etapa consumidora, desde un punto de vista de rendimiento, el tiempo de ciclo debe incrementarse lo menos posible, si es que lo hace. Notemos que incrementar el retardo en una etapa no implica que deba incrementarse también el tiempo de ciclo.

En la Figura 4.15 se muestran ejemplos extremos de instantes de comunicación, pero no excluyen de otros: a) en el inicio del ciclo de reloj y b) finalizando el ciclo de reloj.

En el primer caso la señal, que comunica información entre etapas, puede experimentar casi todo el retardo de la etapa destino. En la Figura 4.15 este primer caso experimenta todo el retado de la etapa destino. En el segundo caso el retardo que

experimente la señal, que comunica información entre etapas, debe ser pequeño, en relación al tiempo de etapa, para que no se incremente el tiempo de ciclo.

En cualquier caso, notemos que añadir un cortocircuito requiere incluir multiplexores adicionales en el camino de datos, con el objetivo de efectuar una selección y estos multiplexores incrementan el retardo de la etapa, independientemente de que se utilice la señal que comunica información desde una etapa posterior. Recordemos que los multiplexores tienen una señal para efectuar la selección y la determinación del valor de esta señal influye en el retardo de la etapa.

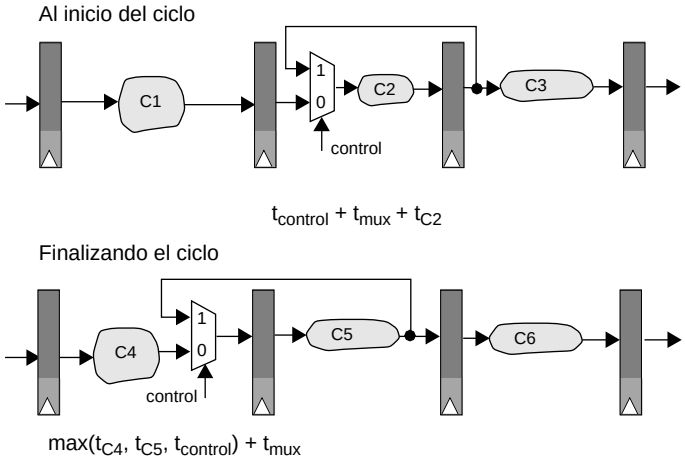


Figura 4.15 *Ejemplos de bucles hardware para comunicar datos. El retardo que se especifica se corresponde como la etapa destino del cortocircuito.*

En la Figura 4.16 se muestra parte del camino de datos segmentado descrito en el Capítulo 3, con 4 cortocircuitos que permiten reducir la latencia efectiva de escritura al banco de registros. Algunos de los cortocircuitos se han utilizado en el código de la Figura 4.14. Para identificar mejor los cortocircuitos se han sombreado los multiplexores de selección y las conexiones se han dibujado con trazo grueso.

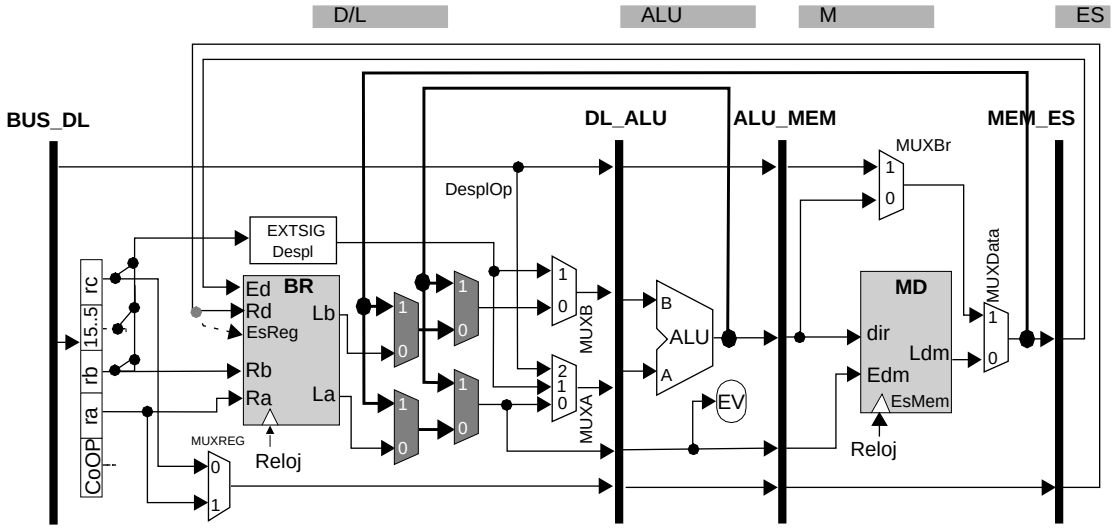


Figura 4.16 *Cortocircuitos añadidos al camino de datos para operaciones aritmético-lógicas.*

La comunicación de información se efectúa, en los cortocircuitos aquí incluidos, finalizando el ciclo de reloj. En la salida de un puerto del banco de registros se disponen dos mutiplexores que permiten seleccionar entre el valor leído del banco de registros o uno de los valores que se comunica mediante los cortocircuitos.

Análisis de la comunicación de datos entre etapas

En este apartado se efectúa un análisis de las comunicaciones de datos entre etapas que son fuente de datos (hay un dato producido) y etapas destino de datos (consumen datos). En una segmentación, la etapa destino es una etapa anterior a la etapa fuente.

En la Figura 4.17 se muestran dos tablas para identificar etapas desde las que se puede suministrar un dato (productoras) y etapas consumidoras, en el procesador descrito en el Capítulo 3. En la tabla de la izquierda se identifican las unidades funcionales productoras de dato (ALU y MD) y las etapas en que está disponibles el dato (ALU, M), antes de actualizarse el elemento de almacenamiento, suponiendo que la comunicación se efectúa

finalizando el ciclo de reloj. En la tabla de la derecha se muestra la etapa más tardía en que se necesita un dato, en función de su utilización.

		Productor de datos	
		ALU	MD
Disponible en etapa	ALU	sí	
	M	sí	sí

		Destino de datos	
		Entradas A y B de ALU o entrada en EV	Dato que se almacena en un store
Necesario en etapa	ALU	sí	
	M		sí

Etapas fuente	CP	BUS	D/L	ALU	M	ES

Etapas destino	CP	BUS	D/L	ALU	M	ES

Figura 4.17 *Tablas que muestran los productores de datos y las etapas destino de datos.*

La comunicación entre etapas fuente y destino se implementa mediante cortocircuitos. En la Figura 4.18 se muestran los bucles hardware creados por los cortocircuitos. La etiqueta de un cortocircuito sigue el formato “C_FuenteDestino”, donde Fuente y Destino son las unidades funcionales productora y consumidora respectivamente (ALU (A), MD (M)). Como unidad funcional fuente utilizaremos la unidad funcional más cercana a la fuente del cortocircuito. Como unidad funcional destino utilizaremos la unidad funcional más cercana al destino del cortocircuito.

En la parte izquierda de la Figura 4.18 la comunicación de información se efectúa finalizando el ciclo de reloj. El dato que se produce en la unidad funcional se utiliza en el mismo ciclo en la etapa consumidora. En la parte derecha de la figura se muestran los bucles harware cuando la comunicación se efectúa al principio del ciclo de reloj.

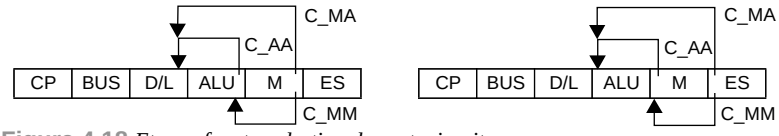


Figura 4.18 *Etapas fuente y destino de cortocircuitos.*

Los dos casos que se han mostrado no excluyen que, en un camino de datos, existan cortocircuitos que se utilizan al principio del ciclo de reloj y cortocircuitos que se utilizan finalizando el ciclo de reloj. La decisión depende, entre otros factores, de la influencia en el tiempo de ciclo y del número de multiplexores necesarios para implementar los cortocircuitos.

Metodología para enumerar los cortocircuitos

En esta sección se caracterizan y enumeran las comunicaciones de datos entre etapas, suponiendo que cada tipo de instrucción utiliza conexiones distintas en el camino de datos. Por ejemplo, el contenido del registro ra de una instrucción tipo RR se transporta a la etapa ALU por un camino distinto cuando la instrucción es del tipo BR (Figura 3.5).

Para caracterizar los riesgos de datos individualmente se requiere un análisis de los tipos de instrucciones, con el objetivo de identificar los registros fuente y destino. En la Figura 4.19 se muestran los campos de la instrucción que identifican los registros fuente y el registro destino para cada tipo de instrucción.

Clases	Tipos	campos			productor escribe	consumidor lee
ENT	RR	ra	rb	rc	rc	ra, rb
	RI	ra		rc	rc	ra
MEM	L (load)	ra	rb		ra	rb
	S (store)	ra	rb			ra, rb
IS	BR	ra				ra

Figura 4.19 *Tipos de instrucciones y registros de lectura y escritura utilizados.*

Para identificar el tipo de instrucción y registro utilizamos la notación “tipo.registro”. Analizando la Figura 4.19 hay 3 posibles productores (RR.rc, RI.rc y L.ra) y 7 posibles consumidores (RR.ra, RR.rb, RI.ra, L.rb, S.ra, S.rb y BR.ra).

Para determinar si es posible utilizar un cortocircuito hay que tener en cuenta: a) la latencia de la fase de ejecución, b) la distancia entre las instrucciones y c) la última etapa en que es necesario el dato en la instrucción consumidora.

Latencia de cálculo (o de la fase de ejecución). Número de ciclos que transcurren desde que se empiezan a utilizar los datos, para obtener un resultado, hasta que se obtiene el resultado.

Recordemos que las instrucciones ENT utilizan la ALU y las instrucciones load la ALU y MD en ciclos consecutivos. Entonces, la latencia de cálculo de las instrucciones ENT es 1 ciclo y la latencia de cálculo de las instrucciones load son 2 ciclos (Figura 4.20).

		Latencia de cálculo
Tipo de instrucción	RR.rc, RI.rc	1
	L.ra	2

Figura 4.20 *Latencia de cálculo por tipo de instrucción.*

Para enumerar los riesgos de datos utilizaremos un espacio de 3 dimensiones. Posteriormente se identifican las posibilidades de cortocircuito en este espacio de 3 dimensiones.

Coordenada P. Productor de dato.

Coordenada C. Consumidor de dato.

Coordenada D (distancia). Número de ciclos transcurridos entre el inicio de la fase de ejecución de una instrucción productora y la necesidad del dato por la instrucción consumidora, para iniciar la fase de ejecución o proseguirla.

En las Figura 4.21 y Figura 4.23 se muestran dos tablas y dos diagramas temporales. Cada tabla es para una distancia concreta (D). En cada tabla la coordenada vertical muestra los productores y la coordenada horizontal los consumidores. En el diagrama temporal de la derecha de la tabla se muestran, mediante una flecha, las comunicaciones entre etapas fuente y destino.

Supondremos que se inicia la interpretación de una instrucción por ciclo. Una marca en una casilla de la tabla de la izquierda indica que se puede utilizar un cortocircuito, ya que temporalmente se puede disponer del dato cuando se necesita; ha sido producido o está siendo producido en el mismo ciclo. Si no hay marca estamos en una situación de riesgo de datos, ya que el dato se produce en algún ciclo posterior.

En la Figura 4.21 se muestra el análisis para la distancia D=1. Todos los consumidores, excepto S.ra, necesitan como muy tarde el dato en la etapa D/L. Cuando los productores son instrucciones del tipo RR o RI la latencia de cálculo es un ciclo. Por tanto, se puede utilizar un cortocircuito, ya que el dato se está produciendo en el mismo ciclo que se quiere consumir y el consumo puede efectuarse al final del ciclo.

Figura 4.21 *Enumeración de los cortocircuitos considerando productor y consumidor de dato a distancia 1.*

Sin embargo, hay situaciones en que no existe la opción de utilizar un cortocircuito, ya que el dato todavía no ha sido producido y por tanto existe un riesgo de datos. Por ejemplo, el productor es L.ra y el consumidor es RR.ra (ciclo 4 en la Figura 4.22). Estos casos se producen debido a que la latencia de la fase de ejecución del load son 2 ciclos y se inicia la interpretación de una instrucción consumidora en el siguiente ciclo.

En la Figura 4.22 se muestra la interpretación de dos instrucciones con el comportamiento descrito. Al detectar el riesgo de datos se bloquea la interpretación de instrucciones 1 ciclo y ahora la instrucción consumidora está a distancia 2. En estas condiciones, el análisis de la posibilidad de utilizar un cortocircuito se traslada al caso D=2.

Cuando el productor es L.ra y el consumidor es S.ra siempre se puede utilizar un cortocircuito, ya que puede esperarse hasta la etapa ALU para consumir el dato.

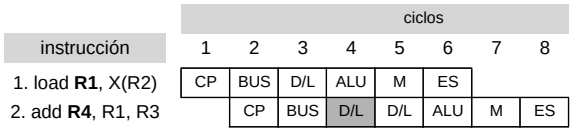


Figura 4.22 *Riesgo de datos. Productora L.ra y consumidora RR.ra.*

La Figura 4.23 muestra el análisis para la distancia D=2. En todos los casos el dato que se quiere consumir en la etapa D/L ya ha sido producido (RR.rc, RI.rc) o se está produciendo en el mismo ciclo (L.ra). La instrucción productora hace 2 ciclos que ha iniciado la fase de ejecución y la latencia de la fase de ejecución de las instrucciones es como máximo de 2 ciclos (load).

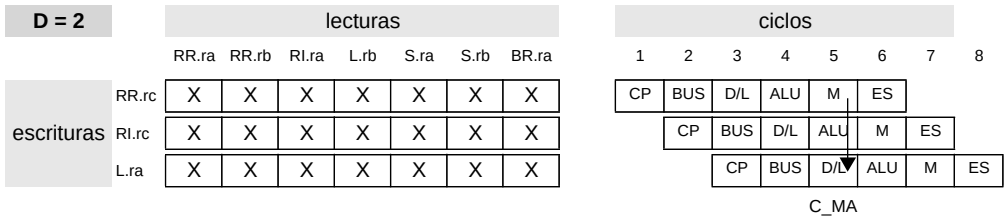


Figura 4.23 *Enumeración de los cortocircuitos considerando productor y consumidor de dato a distancia 2.*

No es necesario analizar distancias superiores, ya que la latencia de segmentación para actualizar el banco de registros son 3 ciclos para todos los tipos de instrucciones.

En resumen, la latencia efectiva de la segmentación, mediante los cortocircuitos, puede reducirse en todos los casos a 1 ciclo, menos en el caso de que el productor sea L.ra y el consumidor sea distinto de S.ra.

En el grupo de casos que la latencia efectiva son 2 ciclos y se detecta un riesgo de datos hay que bloquear la interpretación de instrucciones 1 ciclo. Después de un ciclo de bloqueo la distancia es 2 y ya se puede disponer del dato mediante un cortocircuito.

En general, cuando se necesita el dato en la etapa D/L, los cortocircuitos reducen la latencia efectiva de la segmentación a la latencia de cálculo. En estas condiciones el retardo productor-uso es igual a la latencia de cálculo de la instrucción productora.

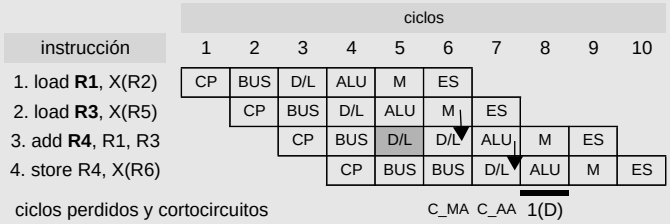
Ejercicio

Al procesador segmentado descrito en el Capítulo 3 se le añaden todos los cortocircuitos que acaban de describirse. Los cortocircuitos se utilizan finalizando el ciclo de reloj.

En un diagrama temporal, muestre la interpretación de la secuencia de instrucciones que está en la columna de la izquierda. Describa los ciclos de bloqueo y la utilización de los cortocircuitos.

Respuesta

En la siguiente figura se muestra el diagrama temporal. En la última fila del diagrama se muestra la etiqueta del cortocircuito y los ciclos de bloqueo. Así mismo, en el diagrama, se muestra mediante una flecha la comunicación entre etapas .



Los datos de la 3ª instrucción son calculados por la 1ª y 2ª instrucción. El valor del registro R1 está disponible en el ciclo 5, sin embargo, el otro dato (R3) aún no está disponible y por tanto se produce una situación de riesgo de datos. El valor del registro R3 no está disponible ya que la latencia de la fase de ejecución de la instrucción productora son 2 ciclos y ha transcurrido 1 ciclo desde el inicio de ejecución de la instrucción productora.

En el ciclo 6 están disponibles los dos datos. El contenido del registro R1 se lee del banco de registros y el valor que se almacenará en el registro R3 se obtiene mediante un cortocircuito desde la etapa M a la etapa D/L.

La 4ª instrucción tiene disponibles sus datos en el ciclo 7 y por tanto no se produce situación de riesgo de datos. El contenido del registro R6 se lee del banco de registros y el valor del registro R4 se obtiene mediante un cortocircuito desde la etapa ALU a la etapa D/L.

Camino de datos con cortocircuitos

No todas las posibilidades de avanzar la disponibilidad del dato (cortocircuito lógico), identificadas en el apartado anterior, requieren una conexión individualizada . Varias de ellas pueden utilizar la misma conexión física (cortocircuito físico). Este hecho depende del camino de datos segmentado del que se parte y del incremento del tiempo de etapa, que puede representar la ubicación de los multiplexores en puntos concretos del camino de datos. Para analizar el tiempo de etapa, además del retardo de la

lógica del camino de datos, hay que tener en cuenta el retardo en generar las señales de control de los elementos que implementan los cortocircuitos.

En este apartado la comunicación de datos entre etapas se efectuará finalizando el ciclo de reloj.

En el camino de datos descrito en el Capítulo 3, las instrucciones utilizan como máximo el contenido de dos registros como datos fuente. Por tanto, un lugar oportuno de ubicación de los multiplexores de cortocircuito en la etapa D/L es entre la salida del banco de registros y los multiplexores de encaminamiento. Recordemos que para determinar la selección en los multiplexores de encaminamiento (MUXA y MUXB) se tiene en cuenta el tipo de instrucción.

La alternativa elegida de ubicación de los multiplexores de cortocircuito $XDLAA_A$, $XDLAA_B$, $XDLMA_A$ y $XDLMA_B$ (Figura 4.24) reduce el número de multiplexores necesarios, aunque no se tiene en cuenta el posible incremento del tiempo de ciclo.

Alternativas para reducir la influencia en el tiempo de ciclo son por ejemplo, ubicar los multiplexores de cortocircuito más tarde en la etapa D/L o ubicar los multiplexores de encaminamiento en la etapa ALU. En general, en el segundo caso, el número de entradas y salidas en los registros de desacoplo puede incrementarse, lo cual incrementaría el consumo de potencia.

Por distancia de un cortocircuito entendemos la longitud del bucle hardware. Esto es, el número de etapas entre la etapa de inicio y la etapa final del bucle hardware, ésta última inclusive.

Un grupo de cortocircuitos (C_AA) está a distancia 1 y proviene de la salida de la etapa ALU; las instrucciones productoras son del tipo RR o RI. Los multiplexores que se utilizan para seleccionarlos son $XDLAA_A$ y $XDLAA_B$.

Otro grupo de cortocircuitos (C_MA) está a distancia 2 y proviene de la salida de la etapa M; la instrucción productora es del tipo RR, RI o L. Los multiplexores que se utilizan para efectuar la selección son $XDLMA_A$ y $XDLMA_B$.

Finalmente, el cortocircuito etiquetado como C_MM permite comunicar información entre la etapa M y la etapa ALU. La fuente del cortocircuito es L.ra y el destino es S.ra. La distancia es 1 y se utiliza el multiplexor XALMM, en la etapa ALU, para seleccionar

entre el dato leído del banco de registros y el dato proveniente de la etapa M. El valor seleccionado es el valor del registro S.ra, el cual se quiere almacenar en memoria.

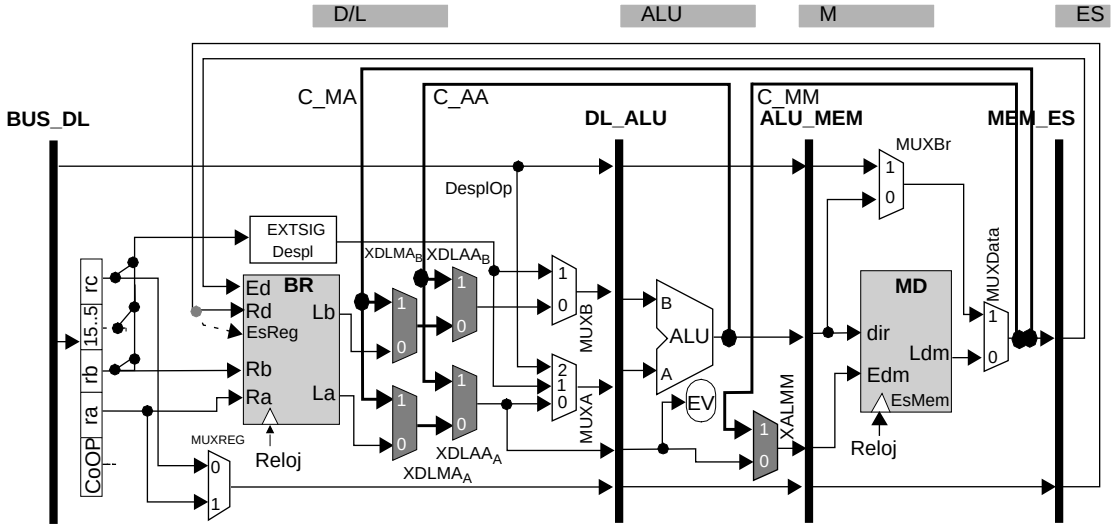


Figura 4.24 *Camino de datos con cortocircuitos.*

En las Figura 4.25 y Figura 4.26 se muestran las tablas de las Figura 4.21 y Figura 4.23, especificando en cada casilla el cortocircuito que se utiliza en el camino de datos de la Figura 4.24. Un cortocircuito se etiqueta con el nombre del multiplexor que permite seleccionarlo. Cuando no se indica ningún cortocircuito no existe esa posibilidad.

D = 1		lecturas						
		RR.ra	RR.rb	RI.ra	L.rb	S.ra	S.rb	BR.ra
escrituras	RR.rc	$XDLAA_A$	$XDLAA_B$	$XDLAA_A$	$XDLAA_B$	$XDLAA_A$	$XDLAA_B$	$XDLAA_A$
	RI.rc	$XDLAA_A$	$XDLAA_B$	$XDLAA_A$	$XDLAA_B$	$XDLAA_A$	$XDLAA_B$	$XDLAA_A$
	L.ra					XALMM		

Figura 4.25 *Asociación de cortocircuitos lógicos y físicos. Distancia 1.*

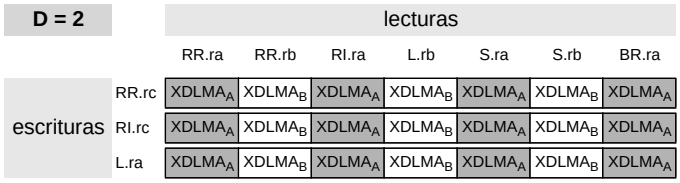


Figura 4.26 *Asociación de cortocircuitos lógicos y físicos. Distancia 2.*

Notemos que de igual forma que varios cortocircuitos lógicos se pueden implementar con un cortocircuito físico se puede dar la circunstancia inversa. Esto es, un cortocircuito lógico puede estar implementado mediante varios cortocircuitos físicos. En el diseño efectuado, este caso se produce cuando la instrucción productora es de tipo RR.rc o RI.rc y el consumo es S.ra. La información se puede comunicar en la etapa D/L mediante el cortocircuito $XDLAA_A$ o en la etapa M mediante el cortocircuito XALMM (Figura 4.27).

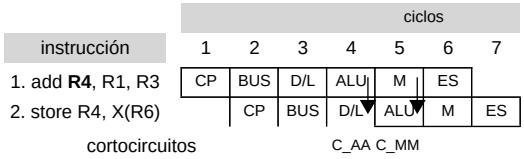


Figura 4.27 *Cortocircuitos físicos que implementan un mismo cortocircuito lógico.*

En la Figura 4.28 se muestra un dibujo vertical de las etapas para evidenciar las entradas de información de cada etapa en un ciclo de la señal de reloj. En la parte de la izquierda están los registros de desacoplo de entrada de las etapas y en la parte de la derecha los registro de desacoplo de salida. Observemos que algunos elementos utilizan como información de entrada señales provenientes de varios registros de desacoplo o de elementos de lógica de otras etapas (cortocircuitos). Este último caso puede representar un incremento del tiempo de ciclo respecto de un camino de datos segmentado sin cortocircuitos.

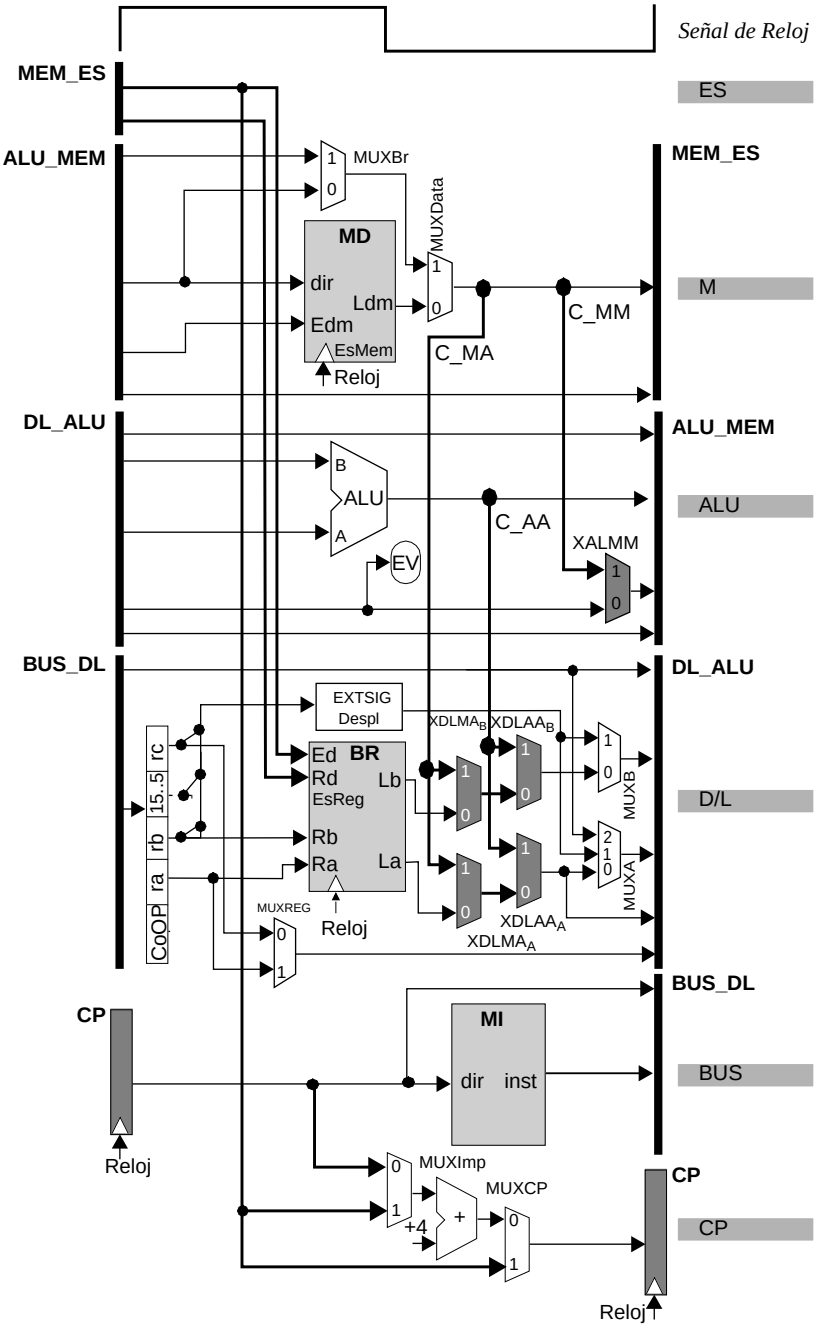


Figura 4.28 *Camino de datos con cortocircuitos. Transcurso en un ciclo de la señal de reloj.*

Ejercicio

Suponga la siguiente secuencia de instrucciones.

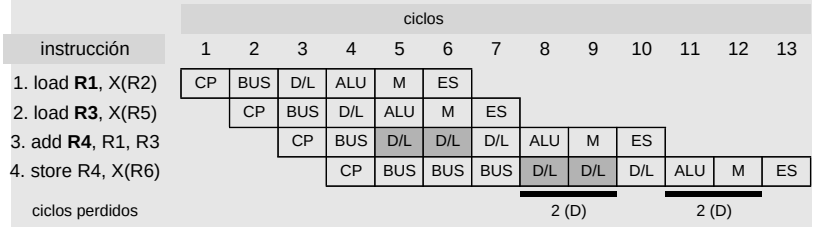
1. load **R1**, X(R2)
2. load **R3**, X(R5)
3. add **R4**, R1, R3
4. store R4, X(R6)

Disponemos de un procesador segmentado como el descrito en el Capítulo 3 con una frecuencia de operación de 1 GHz.

Cuando se añaden cortocircuitos es posible que se incremente el tiempo de ciclo. Calcule la frecuencia de operación mínima del procesador segmentado con cortocircuitos (Figura 4.28) para que el CPI de la anterior secuencia de instrucciones no se incremente.

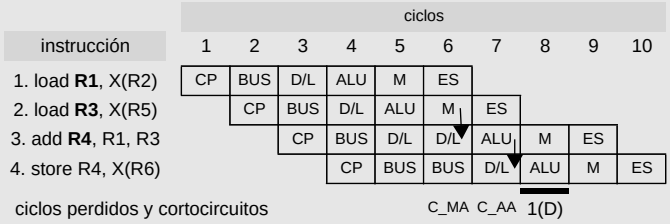
Respuesta

En el siguiente diagrama temporal se muestra la interpretación de la secuencia de instrucciones en el procesador sin cortocircuitos.



Los ciclos por instrucción son: $CPI_{sin} = 8/4 = 2$

En el siguiente diagrama temporal se muestra la interpretación de la secuencia de instrucciones en el procesador con cortocircuitos.



Los ciclos por instrucción son: $CPI_{con} = 5/4 = 1.25$

Tiene que cumplirse $T_{sin} \geq T_{con}$

$$CPI_{sin} \times t_c|_{sin} \geq CPI_{con} \times t_c|_{con}$$

$t_c|_{con} \leq (CPI_{sin} \times t_c|_{sin}) / CPI_{con} = (2 \times 1ns) / 1.25 = 1.6$ ns.

$$f_c^{con} \geq 1.25/2 = 0.625 \text{ Ghz}$$

Lógica de interbloqueos

En un camino de datos con cortocircuitos la lógica de interbloqueos gestiona la utilización de los cortocircuitos y de los bloqueos.

En la Figura 4.29 se muestra un camino de datos simplificado y la información utilizada por la lógica de interbloqueos.

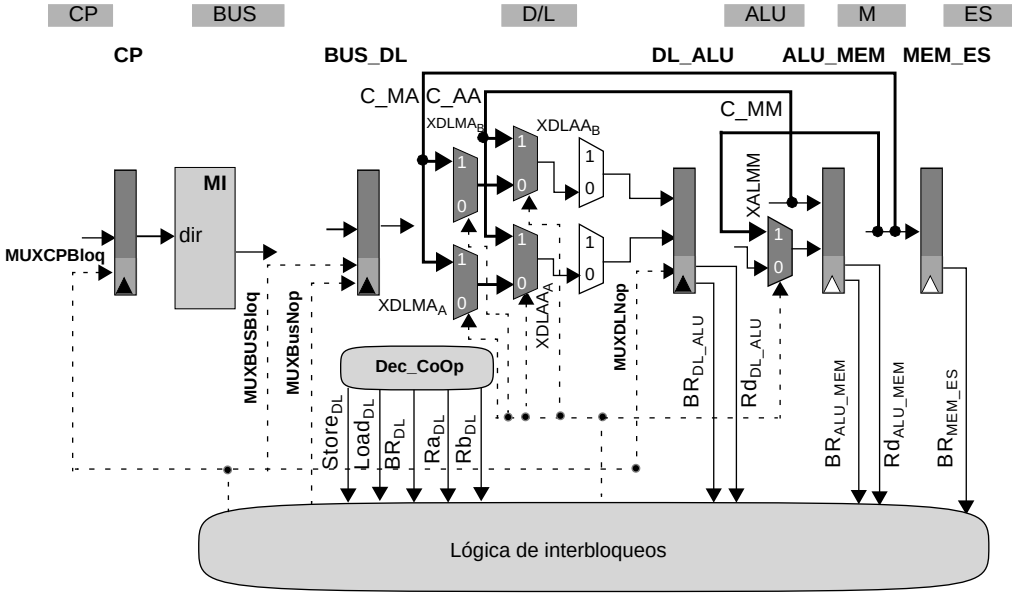


Figura 4.29 *Camino de datos simplificado con cortocircuitos y módulo de interbloqueos. No se muestran las señales de validación de la información suministrada a la lógica de interbloqueos.*

Los cortocircuitos eliminan riesgos de datos. Entonces, la lógica básica de activación de cortocircuitos es la misma que la utiliza para detectar los riesgos que eliminan. Esto es, se comparan los identificadores de registro destino de las instrucciones en las etapas ALU y M con los identificadores de los registros fuente de la instrucción que está en la etapa D/L.

En la Figura 4.30 se muestra la lógica de detección de igualdad de identificadores de registros fuente de la instrucción, que ocupa la etapa D/L, con los identificadores de registro destino de las instrucciones que ocupan las etapas ALU y M.

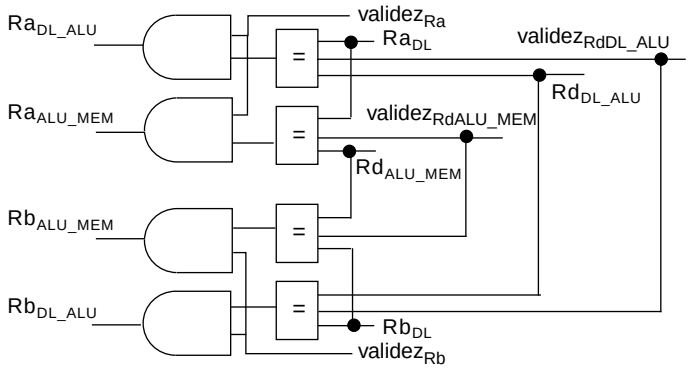


Figura 4.30 *Lógica de detección de igualdad de identificadores de los registros fuente de la instrucción, que ocupa la etapa D/L, con los identificadores de registro destino de las instrucciones que ocupan las etapas ALU y M.*

Las salidas de la lógica de detección de riesgos de datos Ra_{DL_ALU}, Ra_{ALU_MEM} y Rb_{DL_ALU}, Rb_{ALU_MEM} indican si el contenido del registro fuente ra o rb está siendo calculado por una instrucción más vieja que se está interpretando. Los subíndices identifican el registro de desacoplo.

Ahora bien, existe una situación que no era necesario distinguirla cuando se detectaba un riesgo de datos y se bloqueaba el proceso de interpretación y ahora para gestionar los cortocircuitos hay que distinguirla. Supongamos la secuencia de instrucciones mostrada en la Figura 4.31. La 1ª y 2ª instrucción escriben en el registro R1 y la 2ª y 3º instrucción leen el contenido del registro R1. La 2ª instrucción consume el dato que produce la 1ª instrucción y la 3ª instrucción debe consumir el dato que produce la 2ª instrucción.

	ciclos							
instrucción	1	2	3	4	5	6	7	8
1. add **R1**, R2, R3	CP	BUS	D/L	ALU	M	ES		
2. add **R1**, R1, R4		CP	BUS	D/L	ALU	M	ES	
3. load **R10**, 0 (R1)			CP	BUS	D/L	ALU	M	ES

Figura 4.31 *Secuencia de instrucciones que muestra la necesidad de priorizar la selección del cortocircuito.*

Cuando la 3ª instrucción está en la etapa D/L el identificador del registro destino de las etapas ALU y M es el mismo e igual R1. Ahora bien, como productora hay que seleccionar a la instrucción más cercana en orden de programa (instrucción menos vieja). Esto

es, hay que seleccionar a la instrucción que está en una fase menos avanzada del proceso de interpretación, con el objetivo de respetar la semántica del lenguaje máquina. En este caso es la instrucción que está en la etapa ALU. Por tanto, hay que jerarquizar o priorizar la selección del cortocircuito que se utiliza, lo cual tiene que quedar reflejado en el diseño del control.

Después de describir el control de los multiplexores de cortocircuito, se detalla la parte de actuación de la lógica de interbloqueos mediante dos ejercicios. En el primero se diseña la lógica de control de los multiplexores utilizados en los cortocircuitos y en el segundo ejercicio se diseña el control de riesgos.

Ejercicio

Diseñe un circuito combinacional que controle los multiplexores de cortocircuito $XDLAA_A$, $XDLAA_B$, $XDLMA_A$, $XDLMA_B$ y XALMM de la Figura 4.29.

Las salidas del circuito de detección de igualdad de los identificadores de registro fuente, de la instrucción que ocupa la etapa D/L, con los identificadores de los registros destino, de las instrucciones, que ocupan las etapas ALU y M, son: Ra_{DL_ALU}, Ra_{ALU_MEM}, Rb_{DL_ALU}, Rb_{ALU_MEM}, donde los subíndices identifican el registro de desacoplo.

Respuesta

Para nombrar a la señal de control de un multiplexor utilizaremos el nombre del multiplexor.

Los multiplexores $XDLAA_A$, $XDLAA_B$, $XDLMA_A$ y $XDLMA_B$ están ubicados en la etapa D/L y cada uno de ellos permite encaminar el dato transportado por un cortocircuito.

Si un identificador de registro fuente, de la instrucción que ocupa la etapa D/L, es igual a alguno de los identificadores de registro destino, de las instrucciones que ocupan las etapas ALU o M, hay que utilizar el cortocircuito.

El cortocircuito desde la etapa ALU es prioritario frente al cortocircuito desde la etapa M y cualquier cortocircuito es prioritario frente al camino de lectura del banco de registros. Notemos que la ubicación de los multiplexores en el camino de datos determina que esta prioridad sea por defecto. Entonces, las señales de control de los multiplexores son:

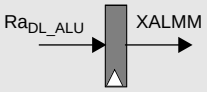
Control del multiplexor	Señal de control
$XDLAA_A$	Ra_{DL_ALU}
$XDLAA_B$	Rb_{DL_ALU}
$XDLMA_A$	Ra_{ALU_MEM}
$XDLMA_B$	Rb_{ALU_MEM}

En la etapa ALU está ubicado el multiplexor XALMM. En una instrucción store, si el identificador del registro fuente ra es igual al identificador de registro destino de la instrucción que está en la etapa M, hay que utilizar el cortocircuito desde la etapa M hacia la etapa ALU.

Notemos que no es necesario conocer el tipo de instrucción productora. Si la instrucción productora es una instrucción tipo RR o RI, la instrucción store obtiene el valor en la etapa D/L , utilizando el cortocircuito $XDLAA_A$ y en el siguiente ciclo se vuelve a obtener el valor utilizando el cortocircuito XALMM.

El identificador del registro fuente ra de una instrucción store no está disponible en la etapa ALU. Entonces la detección se efectúa en el ciclo en el cual la instrucción está en la etapa D/L y se retrasa la actuación hasta el siguiente ciclo, utilizando un registro de 1 bit. En estas circunstancias hay que comparar el identificador ra con el identificador del registro destino en la etapa ALU. Esta comparación ya la hemos efectuado para controlar el multiplexor $XDLAA_A$. La señal que nos interesa es Ra_{DL_ALU}.

Control del multiplexor	Señal de control
XALMM	Ra_{DL_ALU} retardada 1 ciclo



Notemos que no hace falta conocer si la instrucción en la etapa ALU es un load y tampoco hace falta conocer si la instrucción en la etapa D/L es un store, ya que la salida del multiplexor sólo se utiliza en la etapa M cuando la instrucción es un store. Para que esto ocurra ha habido que activar la señal EsMem que indica que se actualice MD.

Ejercicio

Diseñe un circuito combinacional que, en un procesador segmentado con cortocircuitos, detecte y gestione las situaciones de riesgo de datos y secuenciamiento, utilizando los multiplexores MUXCPBloq, MUXBUSBloq, MUXBusNop y el multiplexor MUXDLNop de la Figura 4.29.

La señal de salida Store_{DL} *del módulo Dec_CoOp indica si en la etapa D/L hay una instrucción store y la señal* Load_{DL_ALU} *disponible en el registro de desacoplo DL_ALU indica si en la etapa ALU hay una instrucción load.*

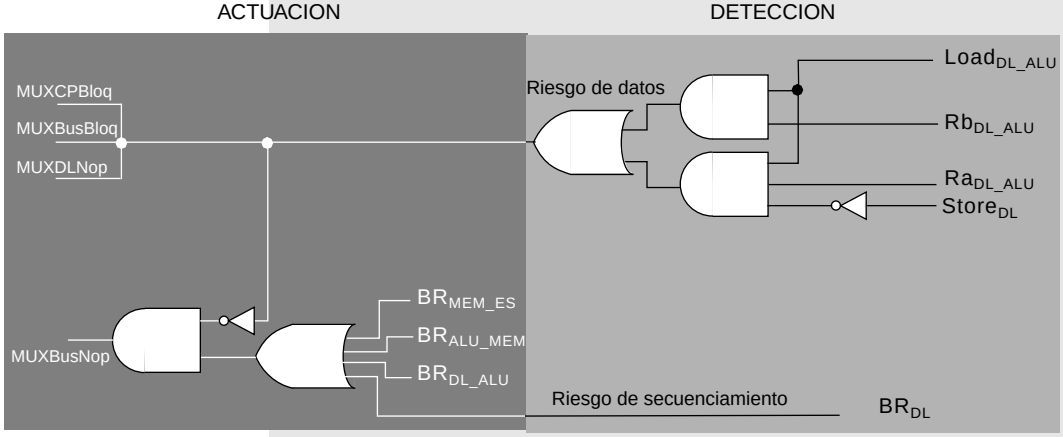
Las salidas del circuito de detección de igualdad de los identificadores de registro fuente, de la instrucción que ocupa la etapa D/L, con los identificadores de los registros destino, de las instrucciones en las etapas ALU y M, son: Ra_{DL_ALU} , Ra_{ALU_MEM}, Rb_{DL_ALU} , Rb_{ALU_MEM} , *donde los subíndices identifican el registro de desacoplo.*

Respuesta

Si la instrucción en la etapa ALU es una instrucción load y el identificador de registro destino es igual al identificador de registro fuente rb de la instrucción que ocupa la etapa D/L se produce un riesgo de datos.

La misma situación se produce con el identificador de registro fuente ra si la instrucción que está en la etapa D/L no es una instrucción store.

En el siguiente gráfico se muestra la lógica de interbloqueos. La parte etiquetada como detección incluye el módulo de detección de igualdad de identificadores, aunque éste no se muestra.



Cuando se detecta un riesgo de datos hay que inyectar una instrucción nop desde la etapa D/L y bloquear las instrucciones que ocupan las etapas D/L, BUS y CP durante 1 ciclo.

Cuando en la etapa D/L hay una instrucción de secuenciamiento, se detecta un riesgo de secuenciamiento (BR_{DL}). A partir de este ciclo (actuación) hay que inyectar instrucciones nop desde la etapa BUS, hasta que la instrucción de secuenciamiento llegue a la etapa ES inclusive.

REDUCCIÓN DE LA PENALIZACIÓN DEBIDA AL SECUENCIAMIENTO

Cuando se interpreta una secuencia de instrucciones sin instrucciones de secuenciamiento la etapa de búsqueda (BUS) suministra, en cada ciclo, instrucciones a la siguiente etapa. Sin embargo, cuando se interpreta una instrucción de secuenciamiento se produce una interrupción del flujo de instrucciones que suministra la etapa BUS a la siguiente etapa.

En al Figura 4.32 se muestran los bucles hardware que se utilizan para el secuenciamiento de instrucccciones.

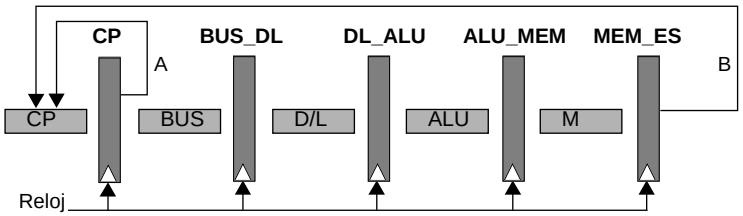


Figura 4.32 *Bucles hardware utilizados en el secuenciamiento de instrucciones. El bucle etiquetado como A se utiliza en el secuenciamiento implícito y el etiquetado como B en el secuenciamiento explícito.*

La actuación, cuando se detecta un riesgo de secuenciamiento, es suspender la interpretación de instrucciones y descartar del camino de datos las instrucciones posteriores a la instrucción de secuenciamiento. La penalización es el número de ciclos de retardo en determinar la dirección de la siguiente instrucción que

debe interpretarse. El número de ciclos de penalización en la Figura 4.32 es 4, que se corresponde con la longitud del bucle hardware B menos uno.

Cuando se interpreta una instrucción de secuenciamiento incondicional, la siguiente instrucción no se puede buscar antes de calcular la dirección destino.

Cuando se interpreta una instrucción de secuenciamiento condicional, para determinar el secuenciamiento hay que esperar el resultado de evaluar la condición y si el salto es tomado, debe también calcularse la dirección destino. Notemos que las acciones de evaluar la condición y calcular la dirección destino pueden efectuarse en paralelo, ya que son independientes entre sí.

Salto tomado. Cuando al interpretar una instrucción de secuenciamiento se modifica el contador de programa con la dirección destino.

Salto no tomado. Cuando al interpretar una instrucción de secuenciamiento condicional no se modifica el secuenciamiento implícito. Es decir, se sigue en secuencia.

En un ejercicio hemos calculado que un 25% de instrucciones de secuenciamiento en un código representa un incremento de CPI igual a 1. Este valor determina que el procesador segmentado obtiene la mitad del rendimiento que se obtendría en el caso ideal. Esto es, los programas tardan el doble de tiempo en ejecutarse. Por ello, interesa desarrollar técnicas que reduzcan la latencia efectiva de la segmentación en instrucciones de secuenciamiento.

Reducción de la latencia de segmentación

Utilizando cortocircuitos hemos visto que se puede reducir la latencia efectiva de la segmentación en actualizar el banco de registros. En el caso de instrucciones de secuenciamiento observamos que toda la información necesaria para efectuar el secuenciamiento está disponible en la etapa M (Figura 4.28). Entonces, si desde la etapa M se alimenta a la etapa CP con la dirección seleccionada se reduce en 1 ciclo la latencia efectiva de segmentación; se pasa de 5 ciclos a 4 ciclos.

Secuenciamiento condicional

Para reducir en mayor medida la latencia efectiva podemos trasladar el multiplexor MUXBr (Figura 4.28) a la etapa ALU (Figura 4.36). En estas condiciones, en la etapa ALU se dispone de toda la información para efectuar el secuenciamiento. Entonces, alimentando la etapa CP desde la etapa ALU se reduce en 2 ciclos la latencia efectiva de segmentación; se pasa de 5 ciclos a 3 ciclos. En este caso, el control del camino de datos se encarga de que en las etapas M y ES no se efectúe ninguna acción cuando se interpreta una instrucción de secuenciamiento.

En la Figura 4.33 se muestra una secuencia de instrucciones que incluye una instrucción de secuenciamiento condicional. La instrucción de secuenciamiento finaliza ahora en la etapa ALU. Para explicitar este hecho, no se muestran en los diagramas temporales los acrónimos de las etapas siguientes a la etapa ALU. En el diagrama temporal suponemos que se cumple la condición y se modifica el secuenciamiento implícito. En el margen izquierdo se muestra el valor de la salida del registro CP correspondiente a la instrucción que se busca en la misma fila. El valor de CP se actualiza en cada ciclo siguiendo el secuenciamiento implícito y en el siguiente ciclo se va a buscar en MI la instrucción que indica la dirección almacenada en CP. Una vez se ha detectado el riesgo de secuenciamiento (ciclo 3) y hasta el ciclo 4 inclusive se inyectan instrucciones nop desde la etapa BUS.

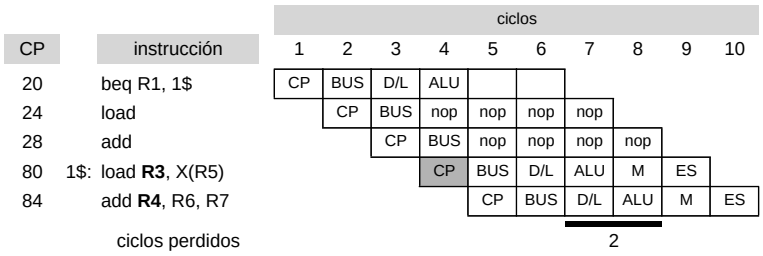


Figura 4.33 *Secuenciamiento de instrucciones cuando una instrucción de secuenciamiento finaliza en la etapa ALU.*

Secuenciamiento incondicional

En el caso de una instrucción de secuenciamiento incondicional la latencia de segmentación puede reducirse en un ciclo, respecto a la latencia de una instrucción de secuenciamiento condicional, si añadimos un sumador en la etapa D/L. Notemos que para el

cálculo de la dirección destino no se requiere el acceso al banco de registros y se dispone de la información necesaria (dirección de la instrucción y literal) en el inicio del ciclo de reloj (Figura 4.36).

En la Figura 4.34 se muestra una secuencia de instrucciones que incluye una instrucción de secuenciamiento incondicional. Como la latencia efectiva de la segmentación son 2 ciclos, la penalización es 1 ciclo.

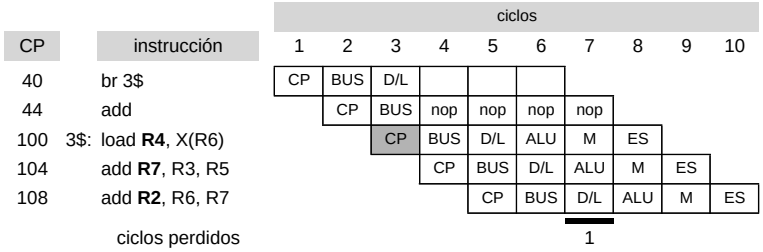


Figura 4.34 *Secuenciamiento de instrucciones cuando una instrucción de secuenciamiento incondicional finaliza en la etapa D/L.*

Bucles hardware

En la Figura 4.35 se muestran los nuevos bucles hardware que se utilizan para el secuenciamiento explícito de instrucciones.

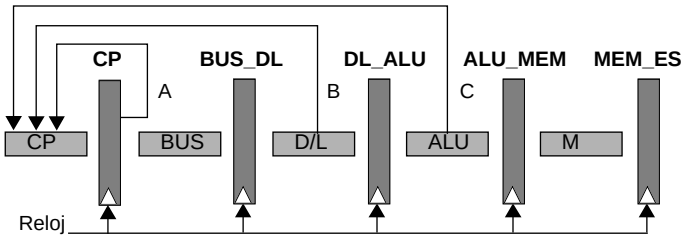


Figura 4.35 *Bucles hardware que permiten reducir la latencia efectiva de una instrucción de secuenciamiento. La etapa CP se alimenta desde las etapas BUS, D/L y ALU.*

El bucle B se utiliza en las instrucciones de secuenciamiento incondicional y el bucle C en las instrucciones de secuenciamiento condicional. Las longitudes de estos bucles hardware son 2 y 3 cilos respecivamente.

Suponemos que las modificaciones efectuadas no modifican el tiempo de ciclo.

Camino de datos y control

En la Figura 4.36 se muestra el camino de datos modificado para que las instrucciones de secuenciamiento incondicional y condicional finalicen respectivamente en las etapas D/L y ALU. Los cortocircuitos, utilizados para eliminar algunos riesgos de datos, no se muestran explícitamente ya que no están afectados. También, para facilitar la observación de los cambios efectuados, se han dibujado con trazo grueso los elementos añadidos o que han modificado su ubicación y las señales correspondientes.

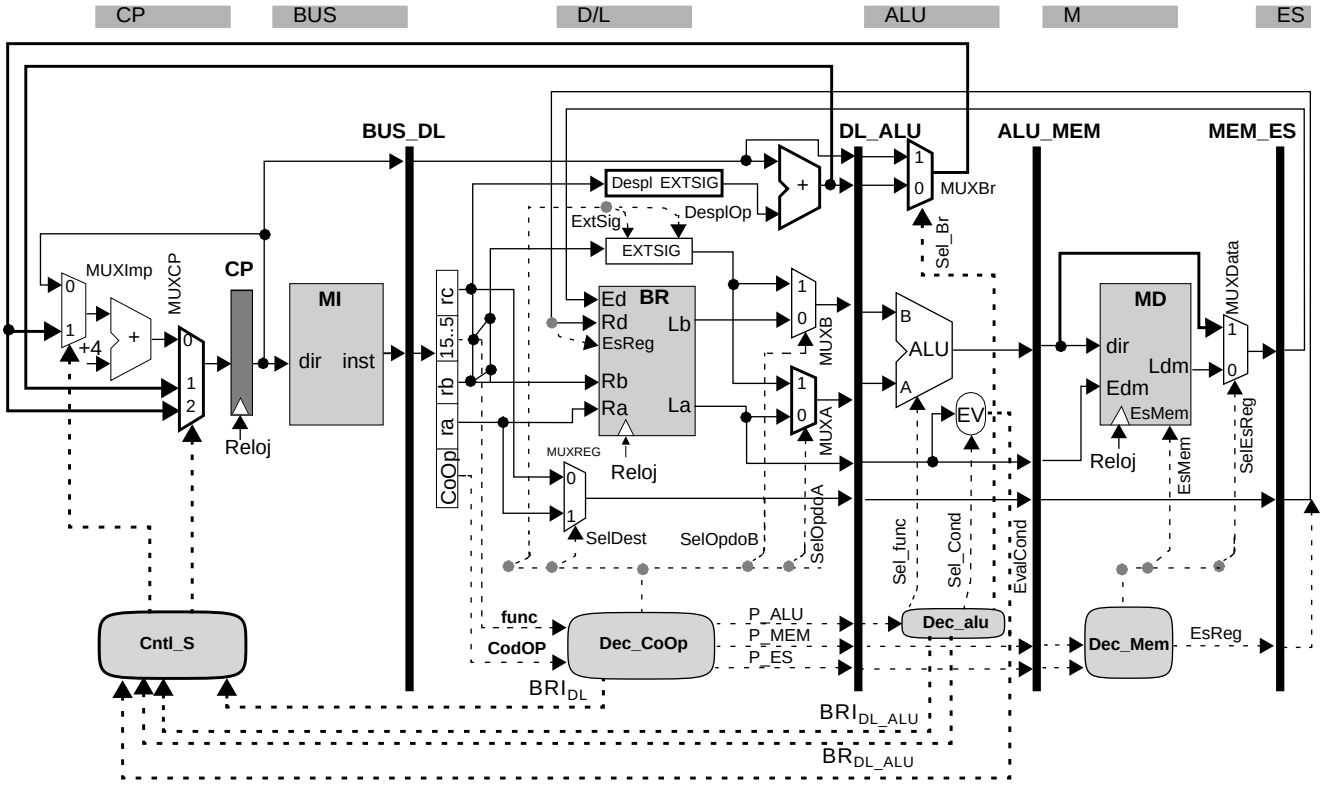


Figura 4.36 *Modificación del camino de datos para que una instrucción de secuenciamiento incondicional finalice en la etapa D/L y una instrucción de secuenciamiento condicional finalice en la etapa ALU. No se muestran los cortocircuitos utilizados para eliminar algunos riesgos de datos.*

El cálculo de la dirección destino de una instrucción de secuenciamiento se efectúa en la etapa D/L. Para ello se añade en la etapa D/L un sumador y un módulo que extiende el signo y

desplaza el campo literal. Ninguno de los dos componentes añadidos requiere señales de control ya que siempre efectúan la misma acción.

Si la instrucción es de secuenciamiento incondicional, en el mismo ciclo, en la etapa CP, se utiliza la dirección destino que se está calculando en la etapa D/L. Si la instrucción es de secuenciamiento condicional se transporta la dirección destino y la dirección de la instrucción de secuenciamiento a la siguiente etapa (ALU). Como la ALU no se utiliza para calcular la dirección destino, el multiplexor MUXA, de la etapa D/L, sólo requiere 2 entradas.

En la etapa ALU se evalúa la condición en el módulo EV (EvalCond), después de leer en el ciclo previo el dato fuente de la instrucción. El módulo Dec_alu utiliza la señal EvalCond y el tipo de instrucción para controlar el multiplexor MUXBr mediante la señal Sel_Br. El multiplexor se utiliza para seleccionar entre la dirección de la instrucción de secuenciamiento y la dirección destino.

La salida del multiplexor MUXBr alimenta a la etapa CP, que en el mismo ciclo determina la dirección de la instrucción que debe buscarse en MI en el siguiente ciclo.

En la etapa CP se dispone de un multiplexor de 3 entradas para seleccionar entre el secuenciamiento implícito y las direcciones suministradas desde las etapas D/L y ALU.

Como ahora las instrucciones de secuenciamiento incondicional y condicional tienen latencias de segmentación distintas, el control de los multiplexores MUXImp y MUXCP requiere información de las etapas D/L y ALU. En la Figura 4.37 se muestra el circuito combinacional de control del secuenciamiento (Cntl_S).

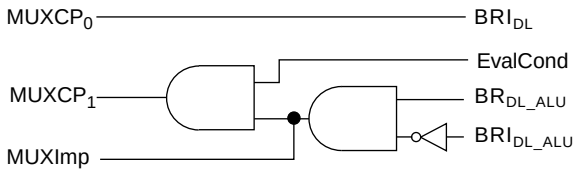


Figura 4.37 *Lógica combinacional del módulo de control de secuenciamiento Cntl_S.*

La señal de salida BRI_{DL}, del módulo Dec_CoOp, indica si la instrucción en la etapa D/L es de secuencionamiento incondicional. Las señales BR_{DL_ALU} y BRI_{DL_ALU} disponibles en la etapa

ALU indican respectivamente si la instrucción que ocupa la etapa es de secuenciamiento y si es incondicional. La condición evaluada se cumple cuando EvalCond=1.

Si en la etapa D/L hay una instrucción de secuenciamiento incondicional se selecciona la entrada uno del multiplexor MUXCP. Cuando en la etapa ALU hay una instrucción de secuenciamiento condicional se selecciona la entrada uno del multiplexor MUXImp y la entrada cero del multiplexor MUXCP si no se cumple la condición. En cambio, si se cumple la condición se selecciona la entrada dos del multiplexor MUXCP.

Cuando en las etapas D/L y ALU no hay instrucciones de secuenciamiento se seleccionan las entradas cero de los multiplexores MUXImp y MUXCP.

Notemos que no pueden coexistir concurrentemente una instrucción de secuenciamiento condicional en la etapa ALU y una instrucción de secuenciamiento incondicional en la etapa D/L, ya que el control de riesgo descarta las instrucciones, buscadas en secuencia, posteriores a la instrucción de secuenciamiento. La gestión del primer riesgo determina que la detección del segundo riesgo, en el mismo ciclo, no sea posible.

Control de los riesgos de secuenciamiento

La actuación descrita, que reduce la latencia efectiva de las instrucciones de secuenciamiento, requiere modificar el control del multiplexor MUXBusNop para gestionar riesgos de secuenciamiento (Figura 4.38).

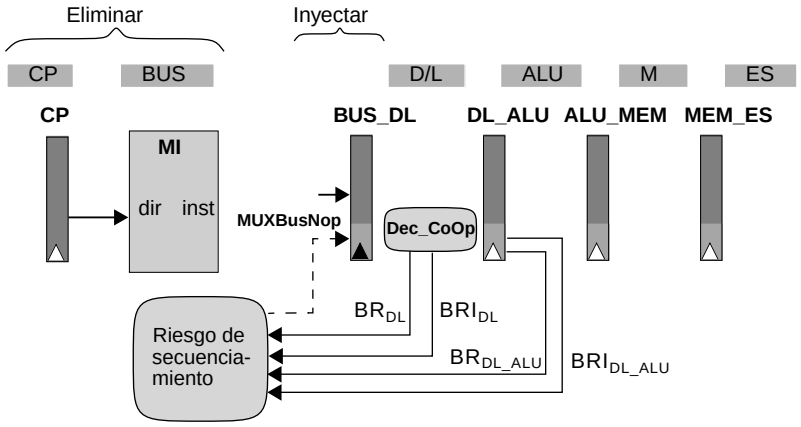


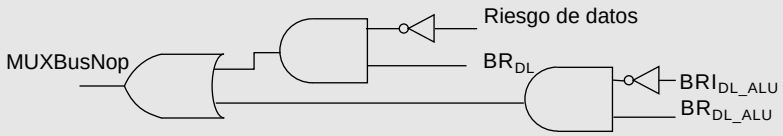
Figura 4.38 *Etapa de búsqueda y multiplexor para gestionar riesgos de secuenciamiento cuando una instrucción de secuenciamiento establece el secuenciamiento en la etapa D/L o en la etapa ALU.*

Ejercicio

Diseñe un circuito combinacional que a partir de la información de tipo de instrucción controle el multiplexor MUXBusNop de la Figura 4.38. El módulo DecCoOp suministra los bits BR_{DL} y BRI_{DL} que indican respectivamente si la instrucción en la etapa D/L es una instrucción de secuenciamiento y si es incondicional. En el registro de desacoplo DL_ALU, se dispone de la misma información de tipo pero relativa a la instrucción que ocupa la etapa (BR_{DL_ALU}, BRI_{DL_ALU}).

Respuesta

Cuando en la etapa D/L hay una instrucción de secuenciamiento, debe seleccionarse la instrucción nop en el multiplexor MUXBusNop de la etapa BUS. Esta acción se lleva a cabo si en la instrucción de secuenciamiento condicional no se detecta un riesgo de datos, el cual debe gestionarse en primer lugar, si existe. Aunque las instrucciones de secuenciamiento incondicional no producen riesgos de datos, se condiciona la señal de instrucción de secuenciamiento (BR_{DL}).



También debe seleccionarse la instrucción nop en el multiplexor MUXBusNop de la etapa BUS cuando en la etapa ALU hay una instrucción de secuenciamiento condicional.

Ejercicio

En el procesador que se acaba de describir, la interpretación de una instrucción de secuenciamiento condicional finaliza en la etapa ALU y la interpretación de una instrucción de secuenciamiento incondicional finaliza en la etapa D/L. ¿Cuál es la penalización al interpretar cada tipo de instrucción de secuenciamiento?.

Suponga que un programa tiene un 25% de instrucciones de secuenciamiento y de ellas el 8% son incondicionales. Estas instrucciones crean situaciones de riesgo que se gestionan emulando una interpretación secuencial. ¿Cuál es el incremento de CPI que representan los riesgos de secuenciamiento respecto del caso ideal de $CPI|_{ideal}$ =1?

Respuesta

La latencia efectiva de una instrucción de secuenciamiento y los ciclos perdidos son

	latencia	ciclos perdidos
incondicional	2	1
condicional	3	2

El incremento de CPI respecto del caso ideal son los ciclos perdidos.

ciclos perdidos = f x cp = 0.25 (0.92 x 2 + 0.08 x 1) = 0.48

donde f es la fracción de instrucciones que produce riesgo de secuenciamiento y cp son los ciclos perdidos por instrucción.

El CPI medio del programa es

$CPI = CPI|_{base}$ + ciclos perdidos = 1 + 0.48 = 1.48

Ejercicio

Suponga que en el camino de datos de la Figura 4.36 el elemento EV se ubica en la etapa D/L, se suprime el multiplexor MuxBr de la etapa ALU, la señal que transporta la dirección de la instrucción que ocupa la etapa D/L se conecta en la entrada 1 del multiplexor MUXImp y se elimina la entrada 2 del multiplexor MUXCP. Indique la penalización por instrucción de secuenciamiento.

Respuesta

La penalización en instrucciones de secuenciamiento incondicional no se modifica y es 1 ciclo. En cambio, la penalización en instrucciones de secuenciamiento condicional se reduce 1 ciclo. Por tanto, ahora la penalización es 1 ciclo.

PREDICCIÓN FIJA DEL SENTIDO

En la Figura 4.39 se muestran los bucles hardware utilizados en el secuenciamiento de instrucciones cuando las instrucciones de secuenciamiento finalizan la interpretación en la etapa D/L (incondicional) o ALU (condicional). Cuando se interpreta una instrucción de secuenciamiento, se produce un riesgo de secuenciamiento debido a que la latencia de los bucles hardware B y C es mayor que 1 ciclo.

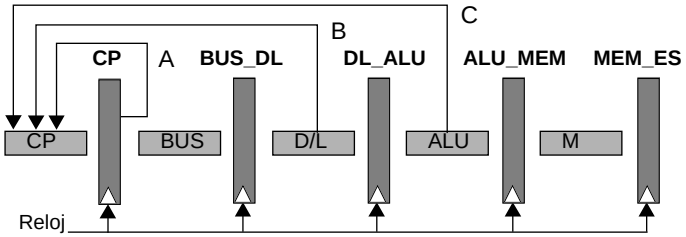


Figura 4.39 *Bucles hardware en el secuenciamiento de instrucciones. La etapa CP se alimenta desde las etapas BUS, D/L y ALU.*

La actuación que se ha descrito hasta ahora para gestionar un riesgo de secuenciamiento es interrumpir desde la etapa BUS el flujo de instrucciones hacia la etapa D/L, hasta que se conoce la dirección de la siguiente instrucción que debe interpretarse.

En este apartado se describe la utilización de predicción en la interpretación de instrucciones de secuenciamiento condicional.

Una técnica de predicción reduce la penalización sin reducir la latencia efectiva de la segmentación. Es una técnica que efectúa una hipótesis sobre un resultado, que puede ser correcta o errónea. Cuando la hipótesis es errónea la penalización no se reduce y en ocasiones puede ser mayor que si no se efectuara ninguna predicción.

Por tanto, una técnica de predicción cuya penalización, en caso de error de predicción, es mayor que si no se efectúa predicción tiene que utilizar buenos predictores, ya que en caso contrario la suma de penalizaciones por errores de predicción podría ser mayor que la reducción de penalización debida a todos los aciertos en la predicción.

Por otro lado, hay que considerar el coste, tanto en componentes como en consumo de potencia, del hardware que se añade para efectuar la predicción, comprobarla o verificarla y efectuar la posterior corrección si la predicción ha sido errónea.

Una técnica de predicción es un mecanismo que se añade a un procesador segmentado que funciona correctamente sin dicha técnica.

Estudios experimentales muestran que el comportamiento de las instrucciones de secuenciamiento condicional es predecible. En particular, se puede aprovechar la predictabilidad del resultado de evaluar la condición.

Sentido. Es el resultado de evaluar la condición en una instrucción de secuenciamiento condicional: seguir en secuencia o modificar el secuenciamiento.

Predicción de sentido. Efectuar una hipótesis sobre el sentido.

La interrupción del flujo de instrucciones hacia la etapa D/L siempre representa una pérdida de rendimiento. Entonces la idea es utilizar, en instrucciones de secuenciamiento condicional, una predicción de sentido con el objetivo de seguir interpretando instrucciones. Si la predicción es correcta puede eliminarse o reducirse la penalización. En caso contrario, se pierde el trabajo efectuado dependiente de la predicción y hay que reanudar el flujo de instrucciones desde la dirección correcta, lo cual podría representar un coste en ciclos.

Predicción fija. Dada una instrucción de secuenciamiento condicional, siempre se efectúa la misma hipótesis sobre su sentido, que puede estar codificada en la instrucción.

A partir de ahora, por instrucción predicha se entenderá que para iniciar la interpretación de esa instrucción se ha utilizado una predicción.

Las necesidades/actuaciones al efectuar una predicción son:

1. Debe comprobarse la predicción (verificación).
2. Las fases que se interpreten de las instrucciones predichas no deben modificar el estado del procesador, ya que en caso de error de predicción las instrucciones predichas deben poderse anular.
3. Hay que restaurar el flujo correcto de instrucciones cuando se detecta un error de predicción.

Las instrucciones que son objeto de una predicción sobre su resultado se siguen interpretando y el resultado que calculan se utiliza para comprobar la corrección de la predicción.

Comprobación de la predicción de sentido. Comparar el valor de la predicción con el resultado de evaluar la condición especificada en la instrucción.

En el diseño descrito, la comprobación de la predicción se puede efectuar, como muy pronto, en la etapa ALU. Recordemos que en la etapa ALU está ubicado el elemento EV, el cual se utiliza para evaluar la condición especificada en la instrucción.

Las fases donde se actualiza el estado del procesador son M y ES en función del tipo de instrucción. Por tanto, antes de que se pueda actuar en un error de predicción, las instrucciones predichas no deben efectuar estas fases, ya que es posible que deban descartarse.

Aunque el registro CP pertenece al estado del procesador, su actualización es la que permite suministrar instrucciones. En un error de predicción de sentido, el encargado de restaurar el valor correcto del registro CP es el mecanismo de recuperación. Por ello, cuando se efectúa una predicción, la dirección que no se ha utilizado en la predicción, o información que permita calcularla, se guarda hasta que se verifica la predicción.

Restaurar el flujo correcto de instrucciones. Después de un error de predicción de sentido, suministrar a la etapa CP una dirección, para que ésta alimente a la etapa BUS con la dirección de la siguiente instrucción que debe interpretarse.

En este apartado supondremos que la restauración del flujo de instrucciones se efectúa en el mismo ciclo que se comprueba la predicción. Recordemos que la comprobación de la predicción se efectúa en la etapa ALU.

Para que una predicción de sentido sea de utilidad hay que disponer de la dirección de la instrucción que hay que buscar en memoria. Predecir seguir en secuencia es simple ya que es el secuenciamiento por defecto. En cambio predecir que se modifica el secuenciamiento implícito requiere conocer la dirección destino y en el camino de datos utilizado se calcula en la etapa D/L (Figura 4.36).

Para efectuar la predicción de sentido utilizamos el signo del literal, el cual está disponible en la etapa D/L. Si el literal es positivo se predice seguir en secuencia. En caso contrario se predice modificar el secuenciamiento implícito. Esta decisión se sustenta en que las estructuras iterativas de un lenguaje de alto nivel (bucles) se traducen en una secuencia de instrucciones de lenguaje máquina donde la última instrucción (final del bucle) es una instrucción de secuenciamiento con un literal negativo.

En primer lugar desarrollaremos el caso de predecir que se sigue en secuencia. Posteriormente analizaremos el caso de predecir que se modifica el secuenciamiento.

Predicción de seguir en secuencia

Utilizaremos el acrónimo CPre, en la etapa ALU de los diagramas temporales, para indicar la acción de comprobación de la predicción de forma explícita.

En la Figura 4.40 se muestra la interpretación de una instrucción de secuenciamiento condicional donde se predice seguir en secuencia y se comprueba que la predicción es correcta. En el ciclo 3 se predice seguir en secuencia y la instrucción que está en la etapa BUS se corresponde con el flujo de instrucciones predicho. En el 4 ciclo no se toma ninguna acción de recuperación, ya que se comprueba que la predicción es correcta y por tanto, la penalización es cero.

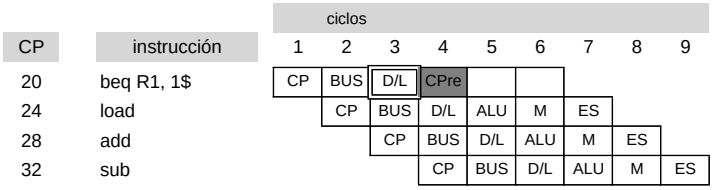


Figura 4.40 *Predicción de seguir en secuencia en una instrucción de secuenciamiento condicional y comprobación de que la hipótesis es correcta.*

En la Figura 4.41 se muestra la interpretación de una instrucción de secuenciamiento condicional donde se predice seguir en secuencia y en la etapa ALU se detecta que la predicción es errónea. En este caso hay que descartar las instrucciones predichas y restaurar el flujo correcto de instrucciones.

La comprobación de la predicción se efectúa en el ciclo 4 (CPre). En este ciclo, las instrucciones predichas no han llegado a fases de interpretación en las que se actualiza el estado del procesador (CP=24 y CP=28). Por tanto, de forma sencilla, pueden eliminarse del camino de datos. Para ello, se inyectan instrucciones nop desde las etapas BUS y D/L, al final del ciclo en el cual se detecta el error de predicción (ciclo 4). La reanudación del flujo correcto de instrucciones se efectúa suministrando la dirección destino desde la etapa ALU a la etapa CP en el 4º ciclo. Notemos que esta dirección ha sido calculada en el ciclo previo en la etapa D/L.

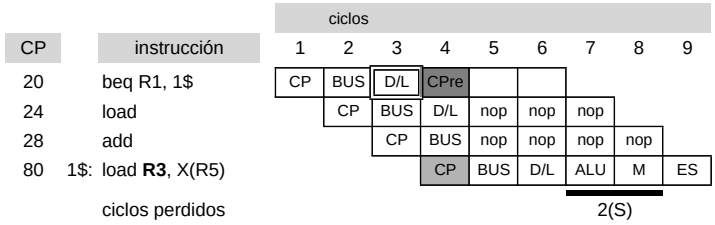


Figura 4.41 *Predicción de seguir en secuencia en una instrucción de secuenciamiento condicional y comprobación de que la hipótesis ha sido errónea.*

En la tabla de la Figura 4.42 se muestran las penalizaciones en ciclos cuando se interpreta una instrucción de secuenciamiento condicional y se predice seguir en secuencia en la etapa D/L.

	comprobación	
	correcta	errónea
Predicción de seguir en secuencia	0	2

Figura 4.42 *Penalizaciones de una instrucción de secuenciamiento condicional cuando se predice seguir en secuencia.*

Predicción de que se modifica el secuenciamiento implícito

En la Figura 4.43 se muestra la interpretación de una instrucción de secuenciamiento condicional prediciendo que se modifica el secuenciamiento implícito y detección en la etapa ALU de que la predicción ha sido correcta.

Mientras no se dispone de información del tipo de instrucción se siguen buscando instrucciones en secuencia. En la etapa D/L se efectúa la predicción (ciclo 3). Como se predice modificar el secuenciamiento se descartan las instrucciones entre la instrucción de secuenciamiento y la instrucción predicha. En este ejemplo es la 2ª instrucción (CP=44). Para eliminar esta instrucción se inyecta una instrucción nop desde la etapa BUS (final del ciclo 3).

En la etapa ALU (ciclo 4) se comprueba la predicción efectuada sobre el sentido de la instrucción de secuenciamiento. Como la predicción es correcta sólo se pierde 1 ciclo.

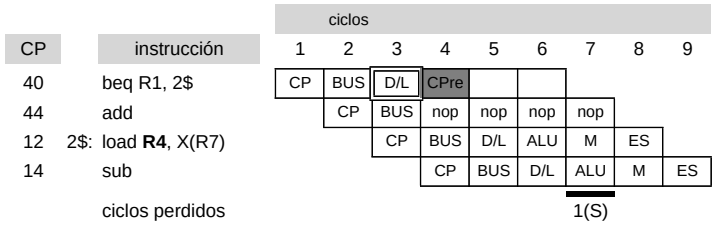


Figura 4.43 *Predicción de modificar el secuenciamiento implícito y comprobación de que la hipótesis ha sido correcta.*

En la Figura 4.44 se muestra la interpretación de una instrucción de secuenciamiento condicional prediciendo que se modifica el secuenciamiento implícito. La predicción es errónea y hay que descartar las instrucciones predichas. En este ejemplo se descarta la 3ª instrucción (CP = 12) inyectando una instrucción nop desde la etapa BUS (ciclo 4). Recordemos que en el 3º ciclo, cuando se predice que se modifica el secuenciamiento, se descarta la 2ª instrucción.

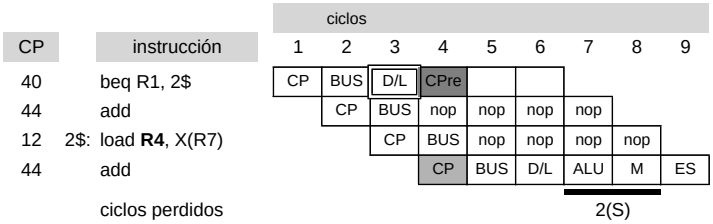


Figura 4.44 *Predicción de modificar el secuenciamiento implícito y comprobación de que la hipótesis ha sido errónea.*

Además de anular las instrucciones que no deben interpretarse hay que restaurar el secuenciamiento. En la Figura 4.44 la dirección de restauración es 44 y la información para restablecer el flujo correcto de instrucciones se suministra desde la etapa ALU. En concreto, se suministra la dirección 40 y en la etapa CP se suma 4 a este valor.

En la tabla de la Figura 4.45 se muestran las penalizaciones en ciclos cuando se interpreta una instrucción de secuenciamiento condicional y se predice modificar el secuenciamiento en la etapa D/L.

	comprobación	
	correcta	errónea
Predicción de modificar el secuenciamiento	1 + 0	1 + 1

Figura 4.45 *Penalizaciones de una instrucción de secuenciamiento condicional cuando se predice en la etapa D/L modificar el secuenciamiento.*

En los ciclos de penalización se puede distinguir el primer ciclo del resto. El primer ciclo es debido a que no se dispone de la dirección destino y también se le denomina penalización por retardo en la búsqueda. El resto de ciclos es debido a la penalización en caso de error en la predicción.

Situaciones de riesgo con instrucciones predichas

En el diseño del control de secuenciamiento hay que tener en cuenta nuevas circunstancias en la interpretación solapada de instrucciones. Estas circunstancias son debidas a situaciones de riesgo en instrucciones predichas. Cualquier situación de riesgo en una instrucción predicha está condicionada a que no se inicie una acción de recuperación en el mismo ciclo.

En el mismo ciclo que se pretende restaurar el secuenciamiento, después de detectar un error de predicción de sentido, una instrucción más joven predicha, que ocupa la etapa D/L

1. Puede ser una instrucción de secuenciamiento.
2. Puede detectar un riesgo de datos.

Modificación del secuenciamiento en el ciclo de recuperación

En la Figura 4.46 se muestra una secuencia de instrucciones donde la primera de ellas es una instrucción de secuenciamiento condicional y se ha predicho seguir en secuencia. La segunda instrucción es una instrucción predicha y es de secuenciamiento incondicional.

La ruptura de secuenciamiento determinada por la instrucción predicha está condicionada por una posible acción de recuperación en el mismo ciclo.

En la Figura 4.46 se supone un error de predicción de sentido Por tanto, hay que descartar las instrucciones predichas, que en este caso es la instrucción br 3$, y seleccionar en la etapa CP la dirección suministrada desde la etapa ALU. En consecuencia, la modificación de secuenciamiento determinada por la 2ª instrucción no se efectúa.

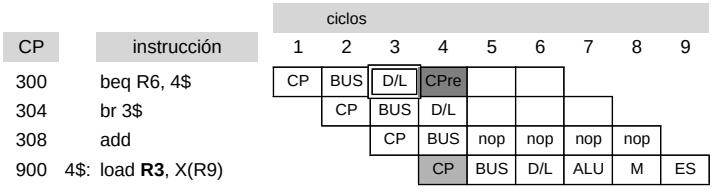


Figura 4.46 *La ruptura de secuenciamiento establecida por una instrucción de secuenciamiento incondicional predicha está condicionada a que en el mismo ciclo no se esté efectuando una acción de recuperación del secuenciamiento.*

Notemos que la 2ª instrucción de la Figura 4.46 puede ser una instrucción de secuenciamiento condicional.

Detección de un riesgo de datos en el ciclo de recuperación

En la Figura 4.47 se muestra una secuencia de instrucciones. En este ejemplo supondremos que no hay cortocircuitos en el camino de datos que permitan disponer del dato antes de escribirlo en el banco de registros.

En el 5º ciclo se detecta un riesgo de datos entre la 1ª y 3ª instrucción, siendo ésta última una instrucción predicha. Si en el mismo ciclo hay que recuperarse de un error de predicción hay que descartar a las instrucciones predichas. Por tanto, la actuación en el caso de detectar un riesgo de datos está condicionada por una posible acción de recuperación del secuenciamiento en el mismo ciclo.

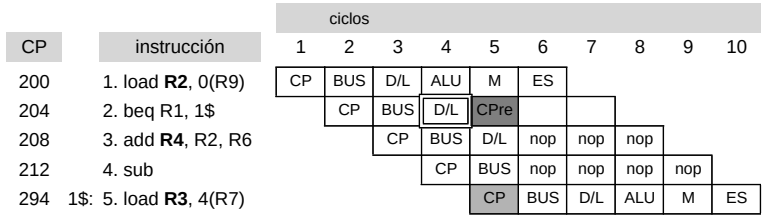


Figura 4.47 *En este ejemplo se supone que no hay cortocircuitos. La actuación cuando se detecta un riesgo de datos está supeditada a que no se efectúe una acción de recuperación del secuenciamiento en el mismo ciclo.*

Camino de datos y control

En la Figura 4.48 se muestra el camino de datos de la Figura 4.36, modificado para efectuar la predicción y recuperar el flujo correcto de instrucciones, después de un error de predicción de sentido. Las modificaciones se muestran en la parte inferior de la figura y están incluidas dentro de una trama.

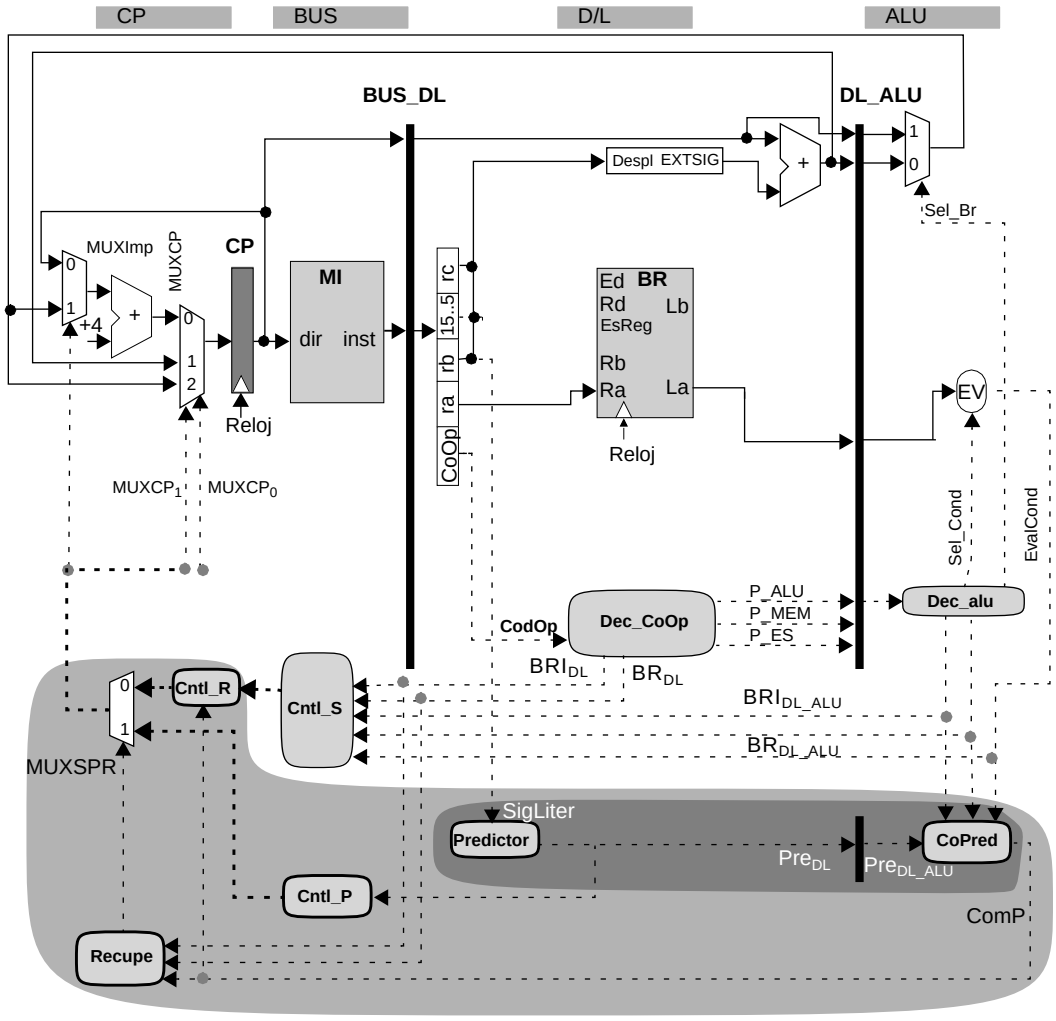


Figura 4.48 *Camino de datos modificado para utilizar predicción de sentido en las instrucciones de secuenciamiento.*

Los módulos Predictor y CoPred están en el camino de datos y los otros módulos añadidos pertenecen al control (Figura 4.49).

	Módulos	Funcionalidad
Camino de datos	Predictor	Efectúa una predicción utilizando el signo del literal
	CoPred	Comprueba la predicción utilizando el resultado de evaluar la condición
Control de los multiplexores de secuenciamiento (MUXImp y MUXCP)	Cntl_S y Cntl_R	Señales de control de los multiplexores cuando se interpreta una instrucción de secuenciamiento incondicional o hay que recuperarse en un error de predicción
	Cntl_P	Señales de control de los multiplexores cuando se efectúa una predicción
	Recupe	Control que permite efectuar una predicción o iniciar una acción de recuperación

Figura 4.49 *Tabla resumen de las funcionalidades de los elementos añadidos al camino de datos para efectuar predicción en las instrucciones de secuenciamiento condicional.*

En la etapa D/L se dispone del módulo Predictor que utiliza el signo del literal (SigLiter) para determinar el valor de la señal de predicción Pre_{DL} (Figura 4.50).

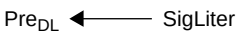


Figura 4.50 *Módulo Predictor.*

En la etapa ALU se comprueba la predicción en el módulo CoPred y se valida con las señales que permiten determinar que la instrucción que ocupa la etapa es una instrucción de secuenciamiento condicional (Figura 4.51). La señal de salida del módulo CoPred es ComP y se activa cuando la predicción ha sido errónea.

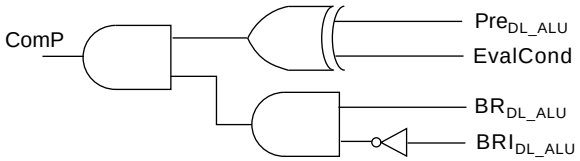


Figura 4.51 *Módulo CoPred.*

Control de la predicción y recuperación

Las salidas de los módulos Cntl_S, Cntl_R y Cntl_P son 3 señales para controlar los multiplexores MUXImp y MUXCP. En el nuevo multiplexor MUXSPR la salida y cada entrada, excluyendo la de control, transporta 3 señales (MUXImp, $MUXCP_0$ y $MUXCP_1$). En la figura estas señales no se muestran individualmente. Ahora bien, como ayuda se ha utilizado un trazo grueso discontínuo para dibujar las líneas.

Los módulos Cntl_S, Cntl_R y Recupe y el multiplexor MUXSPR se utilizan para determinar el secuenciamiento cuando se efectúa una predicción y cuando hay que recuperarse de un error de predicción.

El módulo Cntl_P determina las señales de control de los multiplexores MUXImp y MUXCP en función de la predicción (Figura 4.52). Cuando se predice modificar el secuenciamiento se selecciona la entrada uno del multiplexor MUXCP. En caso contrario hay que seleccionar la entrada cero de los multiplexores MUXCP y MUXImp.

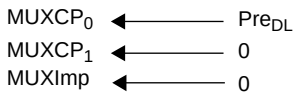


Figura 4.52 *Módulo Cntl_P*

El módulo Cntl_S se utiliza en instrucciones de secuenciamiento incondicional y cuando hay que recuperarse en un error de predicción de sentido. Ahora bien, el módulo Cntl_S mostrado en la Figura 4.37 hay que modificarlo para tener en cuenta que pueden coexistir dos instrucciones de secuenciamiento en la segmentación, siendo la 2ª de ellas una instrucción predicha.

Este caso no se tuvo en cuenta previamente porque no era posible, ya que se descartaban las instrucciones más jóvenes cuando se detectaba el riesgo de secuenciamiento. La modificación de las señales de salida del módulo Cntl_S se efectuará externamente, utilizando el módulo Cntl_R.

El caso que interesa es que la instrucción predicha sea de secuenciamiento incondicional y en el mismo ciclo haya que recuperarse de un error de predicción. La instrucción predicha ocupa la etapa D/L y la recuperación se inicia en la etapa ALU.

En la tabla izquierda de la Figura 4.53 se muestran las salidas del módulo Cntl_S cuando en D/L hay una instrucción de secuenciamiento incondicional predicha y en el mismo ciclo se inicia una acción de recuperación, habiendose predicho seguir en secuencia. Notemos que en el multiplexor MUXCP se seleccionaría la 4ª entrada cuando el multiplexor tiene 3 entradas.

En la tabla derecha de la Figura 4.53 se muestra el valor que deben tener las señales de salida del módulo Cntl_R, cuando se produce la circunstancia descrita, ya que la acción de recuperación requiere utilizar la dirección destino. Ahora bien, en los dos casos de error de predicción la señal $MUXCP_0$ debe tomar el valor cero. El valor en el bit $MUXCP_1$ depende de si hay que seguir en secuencia o modificar el secuenciamiento.

Salidas de Cntl_S	
Señal	**Valor**
MUXImp	1
$MUXCP_0$	1
$MUXCP_1$	1

Salidas de Cntl_R	
Señal	**Valor**
MUXImp	{0, 1}
$MUXCP_0$	0
$MUXCP_1$	{0, 1}

Figura 4.53 *Izquierda: Valor de las señales de salida del módulo Cntl_S cuando, en el mismo ciclo que se inicia una acción de recuperación de un error de predicción de sentido, hay una instrucción de secuenciamiento incondicional predicha en la etapa D/L. Derecha: Valor de las señales de salida del módulo Cntl_R.*

El módulo Cntl_R sólo tiene que modificar las salidas del módulo Cntl_S cuando se produce una acción de recuperación. Entonces, utilizaremos la señal ComP, que indica que hay que recuperarse, para obtener las señales de salida del módulo Cntl_R. En la Figura 4.54 se muestra el módulo Cntl_R. Si hay que recuperarse el bit de menor peso que controla el multiplexor MUXCP se pone a cero.

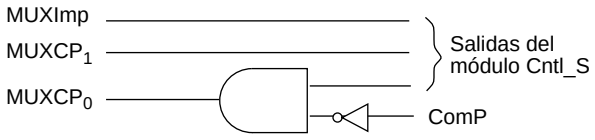


Figura 4.54 *Módulo Cntl_R*

El multiplexor MUXSPR se utiliza para seleccionar entre modificación del secuenciamiento predicho o no y acciones de recuperación.

Cuando se interpreta una instrucción de secuenciamiento incondicional se selecciona la entrada cero del multiplexor MUXSPR. Sin embargo, cuando la instrucción es de secuenciamiento condicional se selecciona la entrada uno del multiplexor.

Cuando se detecta un error de predicción, se selecciona en el multiplexor MUXSPR la entrada cero .

En la Figura 4.55 se muestra una tabla resumen de la entrada del multiplexor MUXSPR que debe seleccionarse en cada caso.

	MUXSPR
Incondicional	0
Predicción	1
Recuperación	0

Figura 4.55 *Entrada que debe seleccionarse del multiplexor MUXSPR en función de la actuación.*

En la Figura 4.56 se muestra la lógica combinacional del módulo Recupe, que se encarga de controlar el multiplexor MUXSPR.

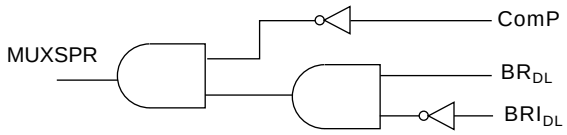


Figura 4.56 *Módulo Recupe.*

Ejercicio

Suponga la siguiente secuencia de instrucciones.

CP		instrucción
4	2$:	load **R7**, X(R9)
8		
...		. . .
16		
20		1. beq R1, 1$
24		2. bne R1, 2$
28		add
. . .		. . .
80	1$:	load **R3**, X(R5)
84		add **R4**, R6, R7

Cuando se interpreta una instrucción de secuenciamiento condicional se predice el sentido en la etapa D/L utilizando el signo del desplazamiento. La condición se evalúa y comprueba en la etapa ALU. La recuperación, en caso de error de predicción de sentido, se efectúa cuando la instrucción está en la etapa ALU.

Empezando con la instrucción beq R1 *1$ muestre, en un diagrama temporal, la interpretación segmentada de la secuencia de instrucciones y describa las actuaciones que se llevan a cabo en cada ciclo. Suponga que la predicción de sentido es errónea.*

Respuesta

En la 1ª instrucción de secuenciamiento se predice que el secuenciamiento no se modifica (ciclo 3) ya que el literal es positivo.

En el ciclo 4 se detecta que la predicción ha sido errónea. En este ciclo, la 2ª instrucción de secuenciamiento predice que debe modificarse el secuenciamiento ya que el literal es positivo. El control no tiene en cuenta la predicción, ya que la instrucción no debe interpretarse, y se restaura el secuenciamiento en la dirección 80.

			ciclos									
CP		instrucción	1	2	3	4	5	6	7	8	9	10
20		1. beq R1, 1$	CP	BUS	D/L	CPre						
24		2. bne R1, 2$		CP	BUS	D/L	nop	nop	nop			
28		3. add			CP	BUS	nop	nop	nop	nop		
80	1$:	4. load **R3**, X(R5)				CP	BUS	D/L	ALU	M	ES	
84		5. add **R4**, R6, R7					CP	BUS	D/L	ALU	M	ES
		ciclos perdidos							2 (S)			

Las instrucciones cuya interpretación se ha predicho se descartan inyectando instrucciones nop desde las etapas D/L y BUS. El secuenciamiento de instrucciones se restaura seleccionado en la etapa CP la dirección destino suministrada desde la etapa ALU.

Control de riesgos de secuenciamiento y recuperación

En la Figura 4.57 se muestra el camino de datos con los elementos necesarios para efectuar el control de riesgos de secuenciamiento. Este módulo de control debe actuar cuando se establece o predice un secuenciamiento y cuando hay que recuperarse de un error de predicción. Respecto del control de riesgo de

secuenciamiento sin predicción (Figura 4.38) se ha añadido el multiplexor MUXDLNop en la etapa D/L. Notemos que utilizamos la simbología simplificada descrita en el Capítulo 3.

El módulo de control de riesgos de secuenciamiento recibe información de los módulos Dec_CoOp, Predictor y CoPred.

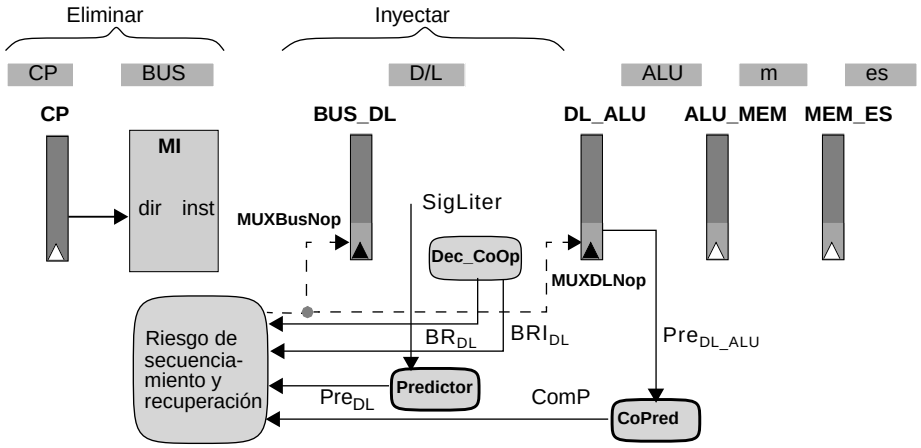


Figura 4.57 *Control de riesgos de secuenciamiento cuando se predice el sentido de las instrucciones de secuenciamiento condicional.*

Por un lado, en la etapa D/L se dispone de información del tipo de instrucción que la ocupa. En la información de tipo de instrucción, el bit BR_{DL} indica si la instrucción que ocupa la etapa es una instrucción de secuenciamiento y el bit BRI_{DL} indica si es incondicional.

Por otro lado, se dispone de la señal de comprobación de predicción ComP, suministrada por el módulo CoPred y de la señal de predicción $Pre_{DL,}$ suministrada por el módulo Predictor.

En la Figura 4.58 se muestra la lógica de control de riesgos de secuenciamiento cuando se utiliza predicción de sentido. Si la instrucción en la etapa D/L es de secuenciamiento incondicional o si se predice modificar el secuenciamiento hay que inyectar una instrucción nop desde la etapa BUS. Esta última acción, está condicionada a que en la instrucción de secuenciamiento condicional no se detecte un riesgo de datos.

Las actuación previa está condicionada a que no se haya iniciado una recuperación en el mismo ciclo. Sin embargo, como hay que efectuar la misma acción si se inicia una acción de recuperación no es necesario condicionarla.

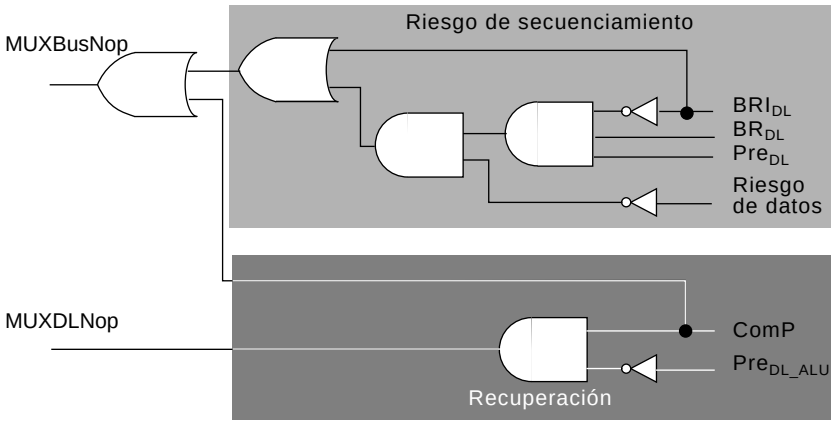


Figura 4.58 *Lógica de control de riesgos de secuenciamiento y recuperación.*

Si la predicción de sentido es errónea (ComP=1) hay que inyectar una instrucción nop, en el ciclo de recuperación, desde la etapa BUS. Si además la predicción había sido seguir en secuencia (Pre_{DL_ALU} = 0) también hay que inyectar una instrucción nop desde la etapa D/L.

Ejercicio

Suponga que la interpretación de una instrucción incondicional modifica el secuenciamiento en la etapa D/L. En la interpretación de una instrucción de secuenciamiento condicional se predice el sentido en la etapa D/L, utilizando el signo del literal. La condición se evalúa en la etapa ALU y se comprueba la predicción en la etapa ALU. La acción de recuperación, en caso de error de predicción de sentido, se efectúa cuando la instrucción está en la etapa ALU.

Un programa tiene un 25% de instrucciones de secuenciamiento y de ellas el 8% son incondicionales. En las instrucciones de secuenciamiento condicional el signo del literal es negativo el 75% de los casos. Cuando el signo del literal es negativo la predicción es correcta un 98% de las veces y cuando el signo del literal es positivo la predicción es correcta un 75% de las veces.

¿Cuál es el incremento de CPI que representan los riesgos de secuenciamiento respecto del caso ideal de $CPI|_{ideal} = 1$*?*

Respuesta

La penalización al interpretar una instrucción de secuenciamiento condicional es función de la predicción efectuada y del resultado de verificar la predicción (correcta, errónea).

		comprobación		
		correcta	errónea	ciclos perdidos
Predicción	seguir en secuencia	0	0.25 x 2	0.5
	modificar el secuenciamiento	0.98 x 1	0.02 x 2	1.02

Los ciclos perdidos al interpretar una instrucción de secuenciamiento condicional son:

ciclos perdidos$|_{cond}$ = 0.25 x 0.5 + 0.75 x 1.02 = 0.89

ya que un 25% de las veces el signo del literal es positivo y el resto es negativo.

Los ciclos perdidos al interpretar una instrucción de secuenciamiento incondicional son:

ciclos perdidos$|_{incond}$ = 1

El número total de ciclos perdidos debido a las instrucciones de secuenciamiento son:

ciclos perdidos = 0.25 x (0.89 x 0.92 +1 x 0.08) = 0.22

donde se tiene en cuenta la fracción de instrucciones de secuenciamiento (25%) y la fracción de instrucciones de secuenciamiento incondicional (8%) y condicional (92%).

El CPI medio del programa es

CPI = CPI$|_{base}$ + ciclos perdidos = 1 + 0.22 = 1.22

Ejercicio

Suponga la siguiente secuencia de instrucciones.

CP		instrucción
4	2$:	load **R7**, X(R9)
8		
...		. . .
16		
20		1. beq R1, 1$
24		2. bne R1, 2$
28		add
. . .		. . .
80	1$:	load **R3**, X(R5)
84		add **R4**, R6, R7

Cuando se interpreta una instrucción de secuenciamiento condicional se predice el sentido en la etapa D/L, utilizando el signo del literal. La condición se evalúa en la etapa ALU y se comprueba la predicción en la etapa ALU. La acción de recuperación, en caso de error de predicción de sentido, se efectúa cuando la instrucción está en la etapa M. Utilice el acrónimo CPre en la etapa ALU para indicar que se comprueba la condición y el acrónimo Rec en la etapa M para indicar una acción de recuperación.

Empezando con la instrucción beq R1,*1$ muestre, en un diagrama temporal, la interpretación segmentada de la secuencia de instrucciones y describa las actuaciones que se llevan a cabo en cada ciclo. Suponga que la predicción de sentido es errónea.*

Respuesta

En la 1ª instrucción de secuenciamiento se predice que el secuenciamiento no se modifica (ciclo 3) ya que el literal es positivo.

En el 4º ciclo, al interpretar la 2ª instrucción, se predice que se modifica el secuenciamiento ya que el literal es negativo.

CP		instrucción	ciclos 1	2	3	4	5	6	7	8	9	10	11
20		1. beq R1, 1$	CP	BUS	D/L	CPre	Rec						
24		2. bne R1, 2$		CP	BUS	D/L	ALU	nop	nop				
28		3. add			CP	BUS	D/L	nop	nop	nop			
4	2$	4. load **R7**, X(R9)				CP	BUS	nop	nop	nop	nop		
80	1$:	5. load **R3**, X(R5)					CP	BUS	D/L	ALU	M	ES	
84		6. add **R4**, R6, R7						CP	BUS	D/L	ALU	M	ES
		ciclos perdidos							3 (S)				

En el ciclo 4 se detecta que la predicción en la instrucción beq R1,1$ ha sido errónea. En el ciclo 5 el control activa la recuperación del secuenciamiento y se descartan las instrucciones 2ª, 3ª y 4º.

Las instrucciones cuya interpretación se ha predicho se descartan inyectando instrucciones nop desde las etapas BUS, D/L y ALU (ciclo 5). El secuenciamiento de instrucciones se restaura seleccionando en la etapa CP, la dirección destino suministrada desde la etapa M.

CAMINO DE DATOS CON PREDICCIÓN DE SENTIDO Y CORTOCIRCUITOS

En la Figura 4.59 se muestra la segmentación en etapas del proceso de interpretación de las instrucciones. En las instrucciones de secuenciamiento condicional la posible acción de recuperación se inicia en el mismo ciclo que se comprueba la predicción (CPre).

		ciclos					
		1	2	3	4	5	6
	ENT / MEM	CP	BUS	D/L	ALU	M	ES
BR	incondicional	CP	BUS	D/L			
BR	condicional	CP	BUS	D/L	CPre		

Figura 4.59 *Etapas y funcionalidad por tipo de instrucción en un procesador segmentado lineal.*

La utilización de cortocircuitos en el camino de datos reduce la latencia efectiva de segmentación de las instrucciones ENT y load. Sólo se detecta riesgo de datos cuando la instrucción productora es un load y en el siguiente ciclo se inicia la interpretación de una instrucción que necesita el dato en la etapa ALU (elementos ALU y EV). En estos casos se pierde 1 ciclo.

Cuando se interpreta una instrucción de secuenciamiento incondicional la latencia efectiva de segmentación son 2 ciclos; por tanto se pierde 1 ciclo. En la interpretación de una instrucción de secuenciamiento condicional se predice el sentido, utilizando el signo del desplazamiento, para reducir la latencia efectiva de la

segmentación. Si la predicción es correcta no se pierde ningún ciclo cuando se sigue en secuencia y se pierde 1 ciclo cuando se toma el salto. En caso de error de predicción se pierden 2 ciclos en los dos casos.

En la Figura 4.61 se muestra un dibujo vertical de las etapas del camino de datos de un procesador segmentado con cortocircuitos y predicción de sentido. Este esquema permite visualizar fácilmente las entradas de información en cada etapa en un ciclo de reloj. Para simplificar el dibujo, los dos multiplexores de cortocircuito en cada salida del banco de registros se han agrupado en un único multiplexor, siendo la funcionalidad y control de este multiplexor idéntica a la descrita en Figura 4.29. En la Figura 4.60 se muestra la equivalencia.

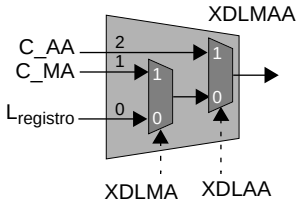


Figura 4.60 *Equivalencia de multiplexores de cortocircuito.*

Las líneas de trazo grueso identifican las señales relacionadas con la interpretación de una instrucción de secuenciamiento. Observe que se ha creado un camino de datos exclusivo para las instrucciones de secuenciamiento, excluyendo la transmisión de información al módulo que evalúa la condición (EV).

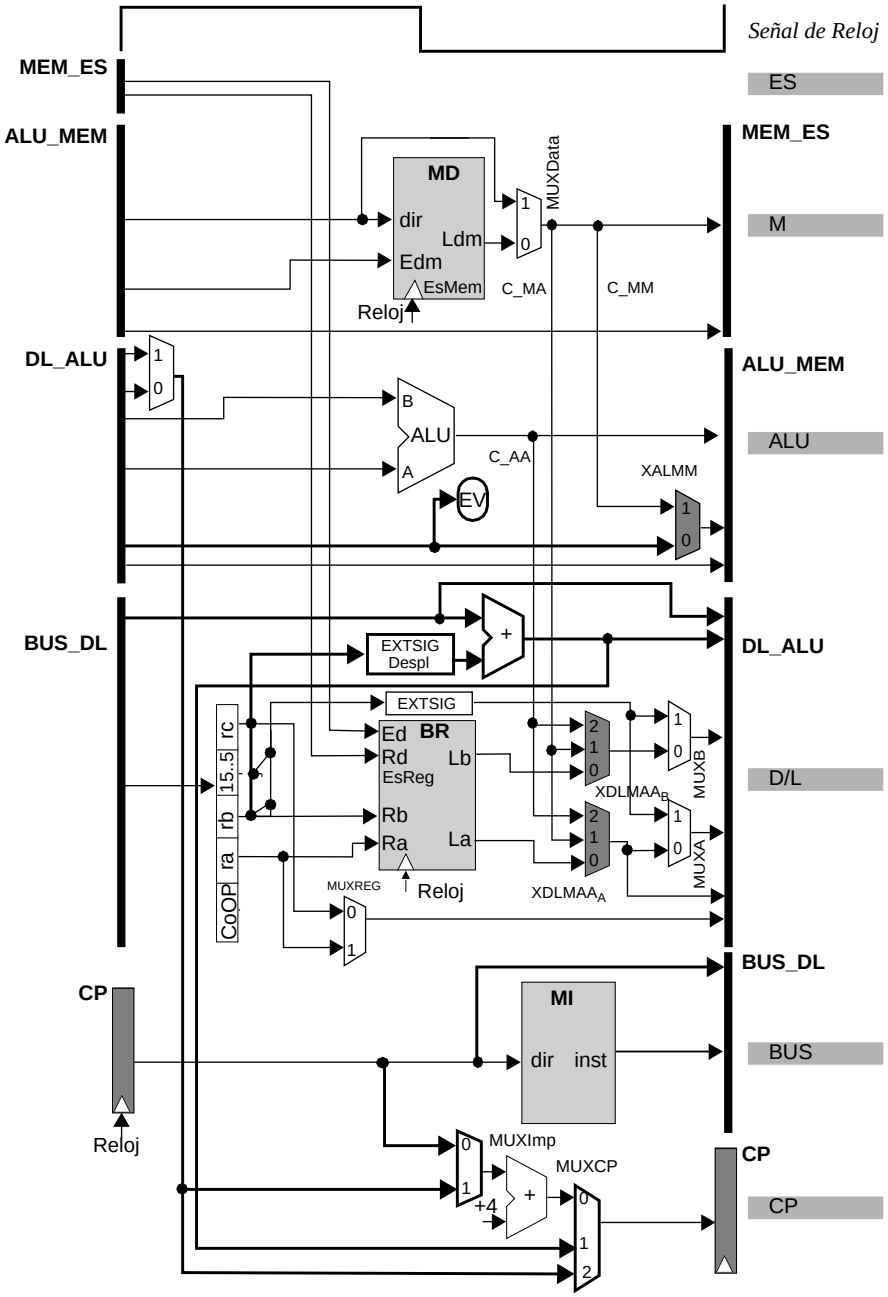


Figura 4.61 *Camino de datos con cortocircuitos y predicción de sentido. Visión de las etapas en un ciclo de la señal de reloj*

Lógica de interbloqueos

En un camino de datos con cortocircuitos y predicción de sentido en las instrucciones de secuenciamiento condicional, la lógica de interbloqueos gestiona la utilización de los cortocircuitos, los riesgos de datos, los riesgos de secuenciamiento y efectúa las actuaciones necesarias para descartar instrucciones en una acción de recuperación.

En la Figura 4.62 se muestra un camino de datos simplificado donde se explicita la información utilizada por la lógica de interbloqueos.

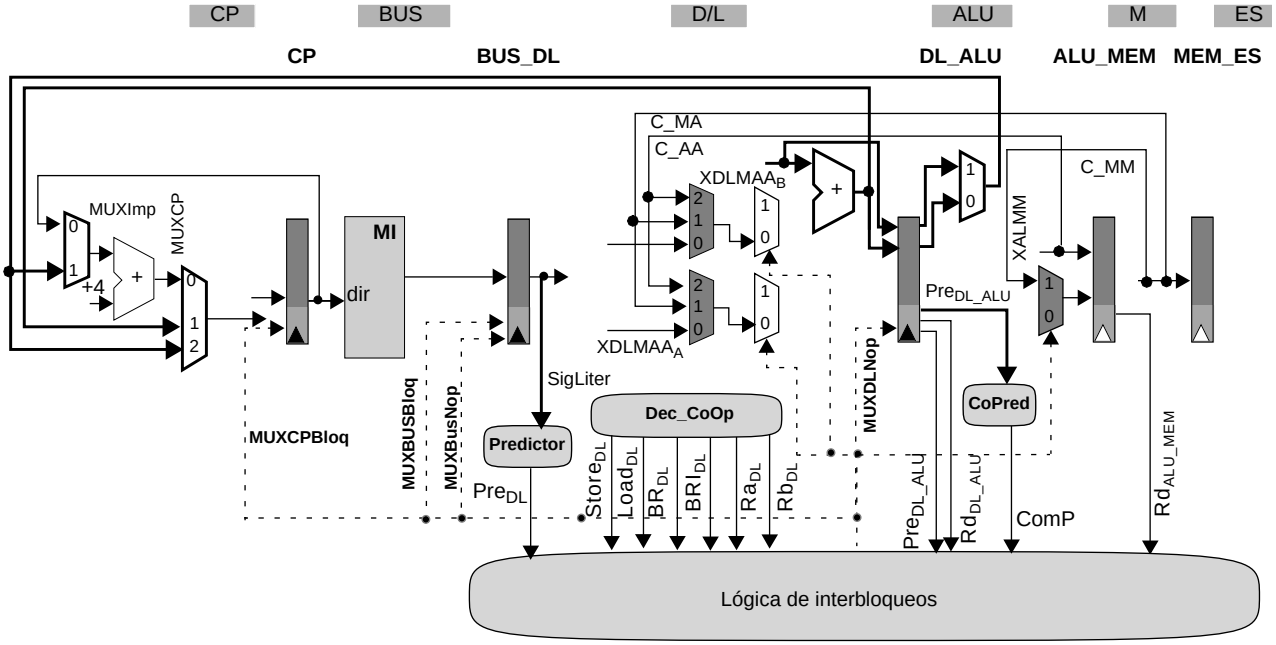


Figura 4.62 *Camino de datos simplificado con cortocircuitos, predicción de sentido en instrucciones de secuenciamiento condicional y módulo de interbloqueos. No se muestran las señales de validación de la información suministrada a la lógica de interbloqueos.*

La lógica de detección de igualdad de identificadores de registro se ha mostrado en la Figura 4.30. La lógica para controlar los cortocircuitos es la misma que cuando no se utiliza predicción de sentido en las instrucciones de secuenciamiento condicional.

En Figura 4.63 se muestra la lógica de interbloqueos. La parte etiquetada como detección incluye el módulo de detección de igualdad de identificadores, aunque éste no se muestra.

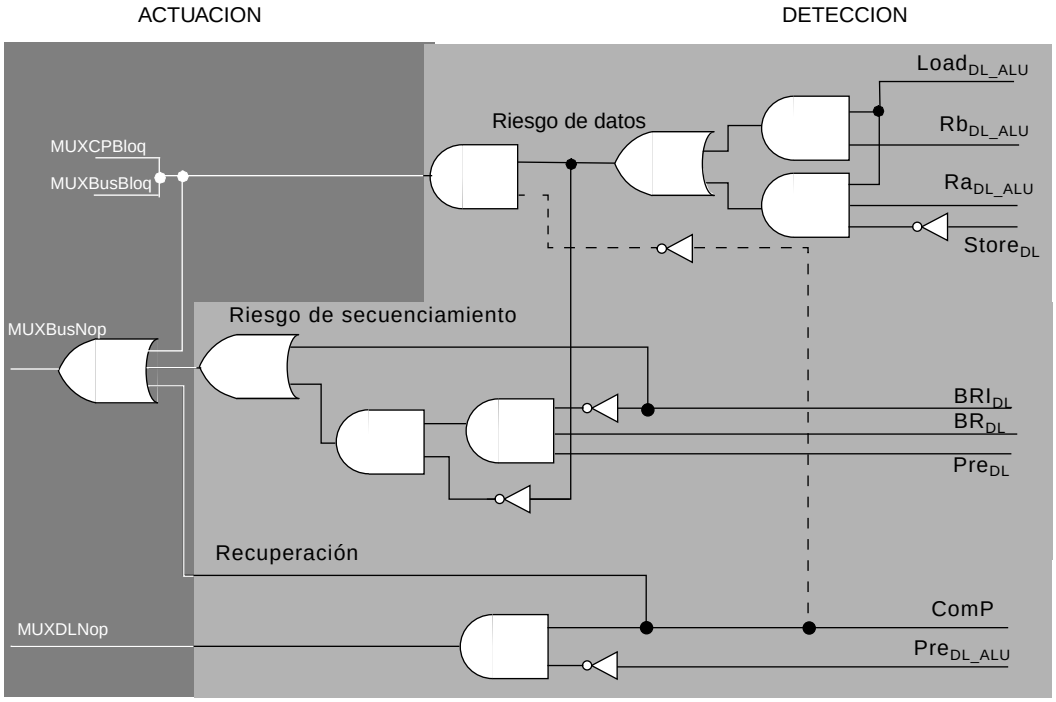


Figura 4.63 *Lógica de interbloqueos en un camino de datos con cortocircuitos y predicción de sentido, en función del signo del desplazamiento, cuando se interpreta una instrucción de secuenciamiento condicional.*

Cuando se detecta un riesgo de datos hay que inyectar una instrucción nop desde la etapa D/L y bloquear las instrucciones que ocupan las etapas D/L, BUS y CP durante 1 ciclo.

Si la instrucción en la etapa ALU es una instrucción load y el identificador de registro destino es igual al identificador de registro fuente rb de la instrucción que ocupa la etapa D/L se produce un riesgo de datos.

La misma situación se produce con el identificador de registro fuente ra si la instrucción que está en la etapa D/L no es una instrucción store.

En general esta actuación está condicionada a que no se inicie la recuperación de un error de predicción de sentido en el mismo ciclo (señal ComP). Ahora bien, en el camino de datos diseñado no se puede detectar un riesgo de datos e iniciar una acción de recuperación en el mismo ciclo (Figura 4.47). Sin embargo, para dejar constancia de ello, en el diseño de la lógica de interbloqueos se ha tenido en cuenta (Figura 4.63, línea a trazos).

Si la instrucción en la etapa D/L es de secuenciamiento incondicional o si se predice modificar el secuenciamiento hay que inyectar una instrucción nop desde la etapa BUS. Esta última acción, está condicionada a que en la instrucción de secuenciamiento condicional no se detecte un riesgo de datos.

Las actuación previa está condicionada a que no se haya iniciado una recuperación en el mismo ciclo. Sin embargo, como hay que efectuar la misma acción si se inicia una acción de recuperación no es necesario condicionarla.

Si la predicción de sentido es errónea (ComP=1) hay que inyectar una instrucción nop, en el ciclo de recuperación, desde la etapa BUS. Si además la predicción había sido seguir en secuencia (Pre_{DL_ALU} = 0) también hay que inyectar una instrucción nop desde la etapa D/L.

Ejercicio

En un programa se interpreta la siguiente distribución de instrucciones.

ENT	Load	Store	BR
42%	23%	15%	20%

La instrucción que sigue en secuencia a una instrucción load necesita utilizar, en la etapa ALU , el dato que lee el load un 25% de las veces.

Un 10% de las instrucciones de secuenciamiento son incondicionales. Un 17% de las veces, el signo del literal en las instrucciones de secuenciamiento condicional es negativo.

Las instrucciones de secuenciamiento condicional con literal positivo siguen en secuencia el 60% de las veces y las que tienen desplazamiento negativo toman el salto el 93% de las veces.

Calcule el CPI del programa cuando se utiliza el procesador descrito en este capítulo, con cortocircuitos y predicción de sentido.

Respuesta

Los ciclos perdidos por riesgos de datos son debidos a instrucciones load seguidas por una instrucción que necesita el dato en la etapa ALU. El número de ciclos perdidos en cada ocurrencia es 1. Entonces,

$$\text{ciclos perdidos}|_{\text{load}} = 0.25 \times 1 = 0.25$$

La distribución de las instrucciones de secuenciamiento es

		proporciones	no tomado	tomado
condicional	literal +	90%	83%	60%	40%
	literal -		17%	7%	93%
incondicional		10%			

Las penalizaciones de las instrucciones de secuenciamiento son las siguientes

			comprobación	
			acierto	error
condicional	literal +		0	2
	literal -		1	2
incondicional		1		

La penalización de las instrucciones de secuenciamiento teniendo en cuenta la distribución es

				comprobación	
		proporciones		acierto	error
condicional	literal +	0.9 x	83 x	0	0.4 x 2
	literal -		17 x	0.93 x 1	0.07 x 2
incondicional		0.1 x 1			

Los ciclos perdidos por instrucción de secuenciamiento son

$$0.1 \times 1 + 0.9\,[0.83 \times 0.4 \times 2 + 0.17\,(0.93 \times 1 + 0.07 \times 2)\,] = 0.86$$

Los ciclos perdidos por las instrucciones de secuenciamiento son

$$\text{ciclos perdidos}|_{\text{secu}} = 0.2 \times 0.86 = 0.17$$

El CPI medio del programa es

$$CPI = 1 + \text{ciclo perdidos}|_{load} + \text{ciclos perdidos}|_{secu}$$

$$CPI = 1 + 0.23 \times 0.25 + 0.17 = 1.23$$

Ejercicio

Por razones de frecuencia de reloj la segmentación del proceso de interpretación de las instrucciones de acceso a memoria requiere dos etapas (ET y DAT). En la etapa ET se accede al campo etiqueta del contenedor y en la etapa DAT se accede al campo dato del contenedor de cache. En estas condiciones la segmentación de los diferentes tipos de instrucción es la siguiente.

Tipo	instrucción	ciclos 1	2	3	4	5	6	7
MEM	load	CP	BUS	D/L	ALU	ET	DAT	ES
	store	CP	BUS	D/L	ALU	ET	DAT	
ENT		CP	BUS	D/L	ALU			ES
BR	incondicional	CP	BUS	D/L				
	condicional	CP	BUS	D/L	CPre			

Al camino de datos se le añaden todos los cortocircuitos necesarios para reducir la latencia efectiva de segmentación cuando se escribe en el banco de registros.

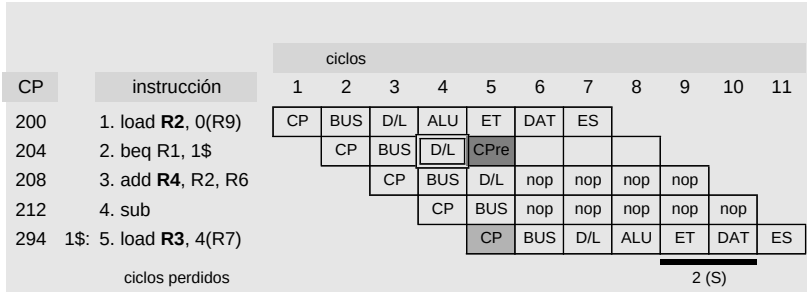
Muestre, en un diagrama temporal, la interpretación de la siguiente secuencia de instrucciones. Suponga que cuando se interpreta la instrucción de secuenciamiento se predice seguir en secuencia y se detecta un error de predicción.

CP		instrucción
200		1. load **R2**, 0(R9)
204		2. beq R1, 1$
208		3. add **R4**, R2, R6
212		4. sub
. . .		. . .
294	1$:	5. load **R3**, 4(R7)

Describa en cada ciclos los riesgos que se detectan y las acciones que se efectúan en el ciclo que se inicia la recuperación del error de predicción.

Respuesta

En el ciclo 4 se predice seguir en secuencia. En el ciclo 5 se detecta un error de predicción y también se detecta un riesgo de datos en la 3ª instrucción debido al registro R2.



Como se ha producido un error de predicción se inicia la recuperación y no se tiene en cuenta el riesgo de datos en la instrucción predicha.

EJEMPLOS

En este apartado se utilizan varios ejemplos de código para analizar el rendimiento de un procesador segmentado con cortocircuitos y predicción de sentido. Uno de ellos es la suma de dos vectores elemento a elemento, otro es la ordenación lexicográfica de los elementos almacenados en un vector y por último se analiza la inserción de un elemento en una lista ordenada. En cuanto a planificación de instrucciones, se analiza en un ejemplo la determinación del camino crítico en un grafo de dependencias.

En los ejercicios, cuando se diga procesador descrito en este capítulo entenderemos: procesador segmentado con cortocircuitos, predicción de sentido, utilizando el signo del literal, en instrucciones de secuenciamiento condicional, reducción de latencia efectiva en instrucciones de secuenciamiento incondicional y el control de riesgos descrito en este capítulo.

Cuando se soliciten ciclos perdidos por riesgos de secuenciamiento se computarán los ciclos perdidos cuando la predicción de sentido es errónea y los ciclos debidos al riesgo de secuenciamiento cuando se predice modificar el secuenciamiento en instrucciones de secuenciamiento condicional o cuando se interpreta una instrucción de secuenciamiento incondicional. A estos últimos ciclos se les denomina ciclos de retardo en la búsqueda de instrucciones.

Suma de dos vectores elemento a elemento

En la parte izquierda de la siguiente figura se muestra el código de la operación suma de dos vectores elemento a elemento en un lenguaje de alto nivel. En la parte derecha se muestra una traducción a lenguaje máquina.

```
do I =1, N
   A(I) = B(I) + C(I)
enddo
```

```
1$:  load r1, 0(r2)      load C(I)
     load r3, 0(r4)      load B(I)
     add r5, r1, r3      B(I) + C(I)
     store r5, 0(r6)     store A(I)
     add r2, r2, #8      índice del vector C
     add r4, r4, #8      índice del vector B
     add r6, r6, #8      índice del vector A
     sub r9, r9, #1      contador de iteraciones
     bne r9, 1$
```

El tamaño de un dato son 8 bytes. El registro r9 se ha inicializado con el número de iteraciones y los registros r2, r4 y r6 se han inicializado con la dirección base de los vectores C, B y A respectivamente.

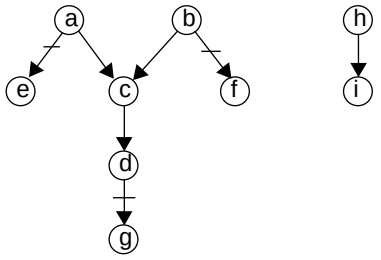
Cuando este código se interpreta en el procesador segmentado con control de riesgos descrito en el Capítulo 3, los ciclos de ejecución por iteración, los ciclos perdidos por tipo de riesgo y el CPI son:

Tiempo de ejecución	Ciclos perdidos		CPI
19 ciclos	Datos	6 ciclos	2.11
	Secuenciamiento	4 ciclos	
	Total	10 ciclos	

Pregunta 1: *Construya el grafo de dependencias de datos del cuerpo del bucle. El compilador conoce que los vectores A, B y C ocupan posiciones disjuntas de memoria.*

Respuesta: En la siguiente figura se muestra la secuencia de instrucciones de lenguaje máquina que efectúa la suma de dos vectores elemento a elemento y el grafo de dependencias.

```
1$:  a. load r1, 0(r2)
     b. load r3, 0(r4)
     c. add r5, r1, r3
     d. store r5, 0(r6)
     e. add r2, r2, #8
     f. add r4, r4, #8
     g. add r6, r6, #8
     h. sub r9, r9, #1
     i. bne r9, 1$
```



Las instrucciones a y b son fuente de una dependencia de datos verdadera y la instrucción destino es la c. La instrucción c es fuente de una dependencia de datos verdadera, siendo la instrucción d la instrucción destino de la dependencia.

La instrucción i es destino de una dependencia de datos verdadera cuya fuente es la instrucción h.

Las instrucciones a, b y d son fuente, cada una de ellas, de una antidependencia con las instrucciones e, f y g respectivamente.

Como las posiciones de memoria que ocupan los vectores fuente (A, B) son distintas de las posiciones de memoria que ocupa el vector destino (C) no hay dependencias debidas a las posiciones de memoria.

Un análisis de las instrucciones nos muestra que algunas dependencias están relacionadas con el cálculo de la dirección efectiva en las instrucciones de acceso a memoria.

La dirección efectiva se calcula sumando el campo literal y el contenido de un registro. La dependencia es entre la instrucción de acceso a memoria y otra instrucción que actualiza el contenido del registro con un valor conocido.

Modificando el campo literal podemos reordenar la instrucción de acceso a memoria y la que modifica el contenido del registro. Por ejemplo, podemos mover la instrucción g antes de la instrucción d si sustituimos esta última por store r5, -8(r6). Notemos que ahora la instrucción g es fuente de una dependencia de datos verdadera y la instrucción d es el destino de la dependencia.

La posibilidad que se acaba de describir se puede utilizar para reducir el número de niveles del grafo de dependencias y con ello reducir la longitud del camino de planificación más largo. Por longitud del camino se entiende la suma de los valores de los arcos desde una raiz a la hoja.

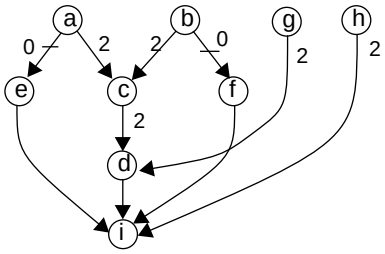
Mover las instrucciones e y f antes de las instrucciones a y b incrementa la profundidad del grafo de dependencias. En cambio, mover la instrucción g antes de la instrucción d reduce la profundidad del grafo y no incrementa la longitud del camino. Teniendo en cuenta estas consideraciones una secuencia de instrucciones que efectúa el mismo calculo es la siguiente.

```
1$:  a. load r1, 0(r2)
     b. load r3, 0(r4)
     c. add r5, r1, r3
     g. add r6, r6, #8
     d. store r5, -8(r6)
     e. add r2, r2, #8
     f. add r4, r4, #8
     h. sub r9, r9, #1
     i. bne r9, 1$
```

Pregunta 2: *Planifique la secuencia de instrucciones que se acaba de mostrar con el objetivo de reducir los ciclos perdidos por riesgos de datos. Recuerde que la latencia o retardo productor-uso en el procesador del Capítulo 3 es 3.*

Respuesta: En la siguiente figuran se muestra la secuencia de código y el grafo de dependencias.

```
1$:  a. load r1, 0 (r2)
     b. load r3, 0 (r4)
     c. add r5, r1, r3
     g. add r6, r6, #8
     d. store r5, -8 (r6)
     e. add r2, r2, #8
     f. add r4, r4, #8
     h. sub r9, r9, #1
     i. bne r9, 1$
```

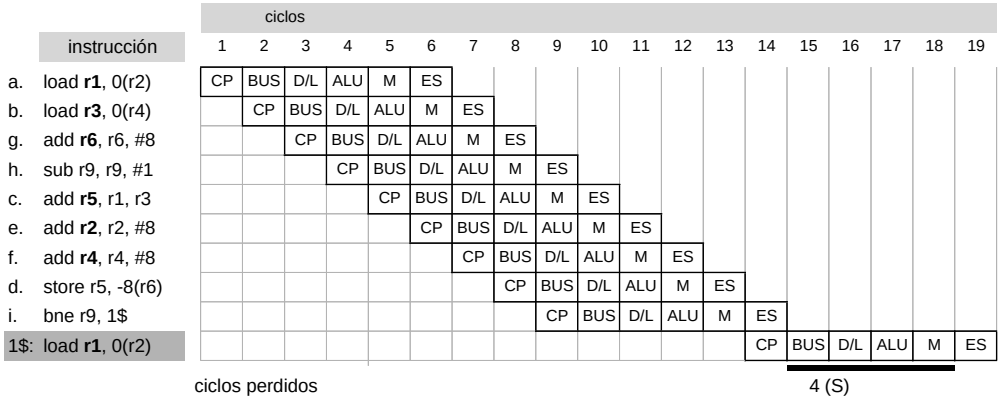


Utilizando el algoritmo de planificación descrito, las dos primeras instrucciones que se planifican son a y b. Posteriormente las instrucciones seleccionadas son g, h.

La 5ª instrucción seleccionada es la c. Posteriormente seleccionamos las instrucciones e, f, d, e i.

Pregunta 3: *Para interpretar el código se utiliza un procesador segmentado con el control de riesgos descrito en el Capítulo 3. Calcule los ciclos de ejecución por iteración del bucle reordenado obtenido en la pregunta 2. Así mismo, indique los ciclos perdidos por tipo de riesgo y calcule el CPI.*

Respuesta: En la siguiente figura se muestra mediante un diagrama temporal la interpretación segmentada de una iteración del código modificado y planificado. En el ciclo 11 se detecta un riesgo de secuenciamiento y se pierden los ciclos del 15 al 18 inclusive.



Los ciclos perdidos son

	Riesgos	
	datos	secuenciamiento
ciclos perdidos	0	4

Los ciclos de ejecución son

$$T_{Pla} = \text{número de instrucciones} + \text{ciclos perdidos}$$

$$T_{Pla} = 9 + 4 = 13 \text{ ciclos}$$

Los ciclos por instrucción son

$$CPI = T_{Pla} / (\text{número de instrucciones}) = 13 / 9 = 1.44$$

Pregunta 4: *Calcule la ganancia cuando se interpreta el código planificado respecto de cuando se interpreta el código sin planificar.*

Respuesta: La ganancia se calcula como el cociente de tiempos de ejecución. Como la frecuencia de funcionamiento es la misma, calculamos el cociente de ciclos de ejecución.

$$\text{Ganancia} = T_{seg} / T_{Pla} = 19 / 13 = 1.46$$

Por tanto, la reordenación de código permite ejecutar el código un 46% más rápido.

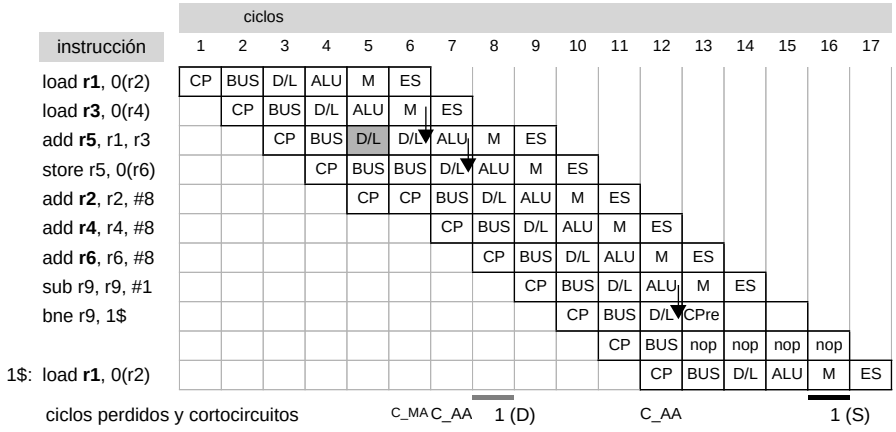
Seguidamente evaluamos el rendimiento que se obtiene cuando se utiliza el procesador descrito en este capítulo y se interpreta el código original, sin reordenar instrucciones.

Además, supondremos que la predicción en la instrucción bne r9, 1$ es correcta.

Pregunta 5: *Calcule los ciclos de ejecución por iteración. Así mismo, indique los ciclos perdidos por tipo de riesgo y calcule el CPI.*

Respuesta: En la siguiente figura se muestra, mediante un diagrama temporal, la interpretación segmentada de una iteración del código. Los cortocircuitos utilizados se muestran mediante flechas desde la etapa fuente a la etapa destino. Además, en la última fila del diagrama se muestra la etiqueta del cortocircuito utilizado y los ciclos perdidos.

En el ciclo 5 se detecta un riesgo de datos debido al registro r3 y se pierde 1 ciclo. En el ciclo 6 se utiliza un cortocircuito desde la etapa M a la etapa D/L para obtener el valor que almacenará r3. En el ciclo 7 se utiliza un cortocircuito desde la etapa ALU a la etapa D/L para obtener el valor que se almacenará en r5.



En el ciclo 12 se utiliza un cortocircuito desde la etapa ALU a la etapa D/L para obtener el valor que se almacenará en el registro r9. En el mismo ciclo se predice modificar el secuenciamiento ya que el campo literal de la instrucción de secuenciamiento es negativo. Se produce un riesgo de secuenciamiento y se pierde 1 ciclo. En el ciclo 13 se verifica la predicción y se determina que ha sido correcta.

Los ciclos perdidos son

	Riesgos	
	datos	secuenciamiento
ciclos perdidos	1	1

El número de ciclos de ejecución en un camino de datos segmentado con cortocircuitos y predicción de sentido en instrucciones de secuenciamiento condicional es

$$T_{RLPS} = \text{número de instrucciones} + \text{ciclos perdidos}$$

$$T_{RLPS} = 9 + 2 = 11 \text{ ciclos}$$

Los ciclos por instrucción son

$$CPI = T_{RLPS} / (\text{número de instrucciones}) = 11 / 9 = 1.22$$

Una ligera reordenación de código elimina el riesgo de datos. Por ejemplo, ubicar la instrucción add **r2**, r2, #8 después de la 2ª instrucción load.

Supongamos que los cortocircuitos y la predicción de sentido no incrementan el tiempo de ciclo.

Pregunta 6: *Calcule la ganancia que se obtiene utilizando la versión del procesador descrita en este capítulo respecto de la versión del procesador descrita en el Capítulo 3, suponiendo que no se reordena el código. Así mismo, calcule la ganancia cuando en la versión del procesador descrita Capítulo 3 se interpreta el código reordenado obtenido en la 2ª pregunta.*

Respuesta: La ganancia se calcula como el cociente de tiempos de ejecución. Como la frecuencia de funcionamiento es la misma efectuamos el cociente de ciclos de ejecución.

$$\text{Ganancia}|_{\text{sin pla}} = T_{seg} / T_{RLPS} = 19 / 11 = 1.73$$

donde T_{seg} es el tiempo cuando se utiliza el procesador segmentado del Capítulo 3.

Por tanto, la reducción de latencia efectiva y la predicción de sentido permite ejecutar el código un 73% más rápido.

$$\text{Ganancia}|_{\text{con pla}} = T_{Pla} / T_{RLPS} = 13 / 11 = 1.18$$

donde T_{Pla} es el tiempo cuando se utiliza el procesador segmentado del Capítulo 3 y se interpreta el código reordenado.

Por tanto, la reducción de latencia efectiva y la predicción de sentido permite ejecutar el código un 18% más rápido.

Ordenación lexicográfica de los elementos de un vector

Como algoritmo de ordenación utilizaremos el denominado burbuja y nos centraremos en el bucle interno. Este bucle recorre los elementos de un vector y los compara dos a dos. En el caso de estar desordenados efectúa un intercambio del contenido de las posiciones que ha comparado.

En la siguiente figura se muestra el código en alto nivel y la traducción a lenguaje máquina del bucle interno del algoritmo de la burbuja. El bucle externo, el cual no se muestra, utiliza la variable cambios para determinar si ha finalizado la ordenación. Para ello inicializa con el valor cero esta variable antes de iniciar cada ejecución del bucle interno.

```
do I =1, N
    if (A(I) > A(I+1)) then
        temp = A(I)
        A(I) = A(I+1)
        A(I+1) = temp
        cambios = cambios +1
    endif
enddo
```

```
1$:  load r8, 0(r7)        load A(I)
     load r9, 4(r7)        load A(I+1)
     cmple r10, r8, r9     A(I) ≤ A(I+1)
     bne r10, 2$           se intercambia si A(I) > A(I+1)
     store r9, 0(r7)       store A(I)
     store r8, 4(r7)       store A(I+1)
     add r5, r5, #1        contador de cambios
2$:  add r6, r6, #1        contador de iteraciones
     add r7, r7, #4        índice del vector A
     cmple r11, r6, r4     r6 ≤ r4
     bne r11, 1$           iterar si aún no ha finalizado
```

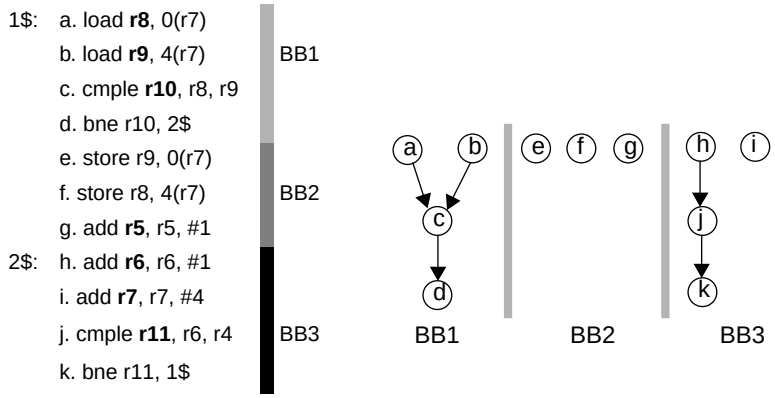
El registro r4 se inicializa con el número de iteraciones y el registro r6 con el valor uno. El registro r7 se inicializa con la dirección base del vector A y el registro r5 se inicializa con el valor cero.

Cuando este código se interpreta en el procesador segmentado con control de riesgos descrito en el Capítulo 3, los ciclos de ejecución por iteración, los ciclos perdidos por tipo de riesgo y el CPI son:

sentido del salto	Tiempo de ejecución	CPI	Ciclos perdidos		
NT	26 ciclos	2.36	Datos	NT / T	7 ciclos
T	23 ciclos	2.88	Secuenciamiento	NT / T	8 ciclos
			Total	NT / T	15 ciclos

Pregunta 1: *Construya el grafo de dependencias de datos del cuerpo del bucle.*

Respuesta: En la siguiente figura se muestra el grafo de dependencias de los tres bloques básicos.



En el BB1 todas las instrucciones son dependientes entre si. Por tanto no se pueden reordenar. En el BB2 todas las instrucciones son independientes. Por tanto, se puede utilizar cualquier orden. En el BB3 sólo la instrucción i no está en la cadena de dependencias.

Pregunta 2: *Planifique las instrucciones en cada BB con el objetivo de reducir los ciclos perdidos por riesgos de datos, si ello es posible, cuando se interpretan en el procesador descrito en el Capítulo 3. Justifique la respuesta.*

Respuesta: En el BB1 sólo se pueden reordenar las instrucciones a y b. En el BB2 se puede utilizar cualquier orden. En el BB3 sólo se puede reordenar la instrucción i.

Ninguna de las posibles reordenaciones reduce los ciclos perdidos por riesgos de datos.

Seguidamente evaluamos el rendimiento que se obtiene en el procesador descrito en este capítulo, cuando el código que se intepreta es el original, sin reordenar instrucciones.

En todas la preguntas supondremos que al interpretar la instrucción bne r11,1$ la predicción es correcta.

Pregunta 3: *Calcule los ciclos de ejecución por iteración. Así mismo, indique los ciclos perdidos por tipo de riesgo y calcule el CPI.*

Efectúe los cálculos en los dos posibles supuestos al interpretar la instrucción bne r10,2$.

Respuesta: Distinguimos las dos posibles situaciones al interpretar la instrucción bne r10,2$.

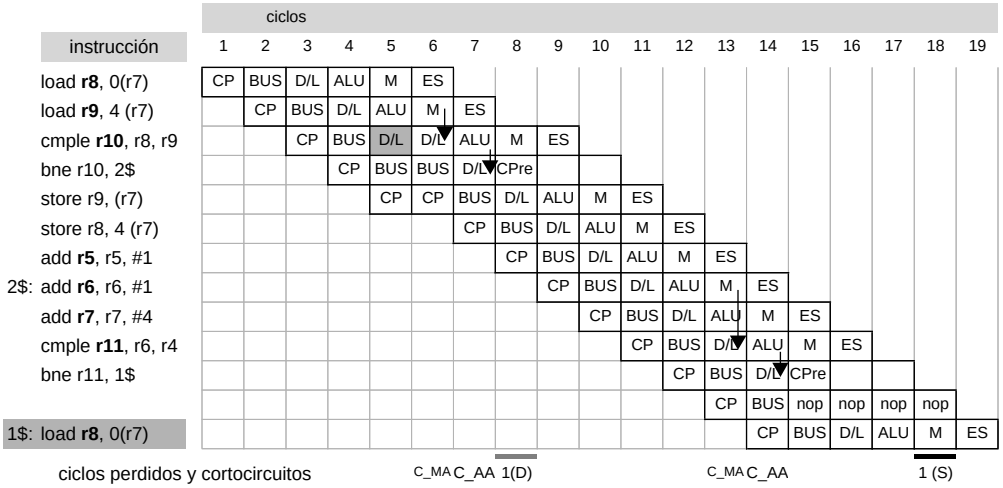
A) Salto no tomado (NT): No se cumple la condición de salto en la instrucción bne r10, 2$.

En la siguiente figura se muestra la interpretación segmentada de una iteración del código. Se predice seguir en secuencia, ya que el literal es positivo. Por tanto, se acierta en la predicción de sentido, ya que no se cumple la condición evaluada.

En el ciclo 5 se detecta un riesgo de datos debido al registro r9. En el ciclo 6 se utiliza el cortocircuito desde la etapa M a la etapa D/L y se prosigue la interpretación de instrucciones.

En el ciclo 7 se utiliza el cortocircuito desde la etapa ALU a la etapa D/L y se predice seguir en secuencia ya que el signo del desplazamiento es positivo. En el ciclo 8 se detecta un acierto de predicción de sentido y por tanto, no se pierde ningún ciclo.

En los ciclos 13 y 14 se utilizan respectivamente cortocircuitos desde las etapas M y ALU a la etapa D/L. En el ciclo 14 se predice modificar el secuenciamiento ya que el literal es negativo y en el ciclo 15 se comprueba que la predicción es correcta. Por tanto, sólo se pierden 1 ciclo por riesgo de secuenciamiento.



Los ciclos perdidos son

	Riesgos	
	datos	secuenciamiento
ciclos perdidos$\|_{NT}$	1	1

Los ciclos de ejecución y el CPI cuando se toma el salto son:

En una iteración	salto tomado
$T_{seg}\|_{NT}$ = número de instrucciones + ciclos perdidos	11 + 2 = 13 ciclos
$CPI_T = T_{seg}\|_{NT}$ / (número de instrucciones)	13 / 11 = 1.18

B) Salto tomado (T): Se cumple la condición de salto en la instrucción bne r10, 2$.

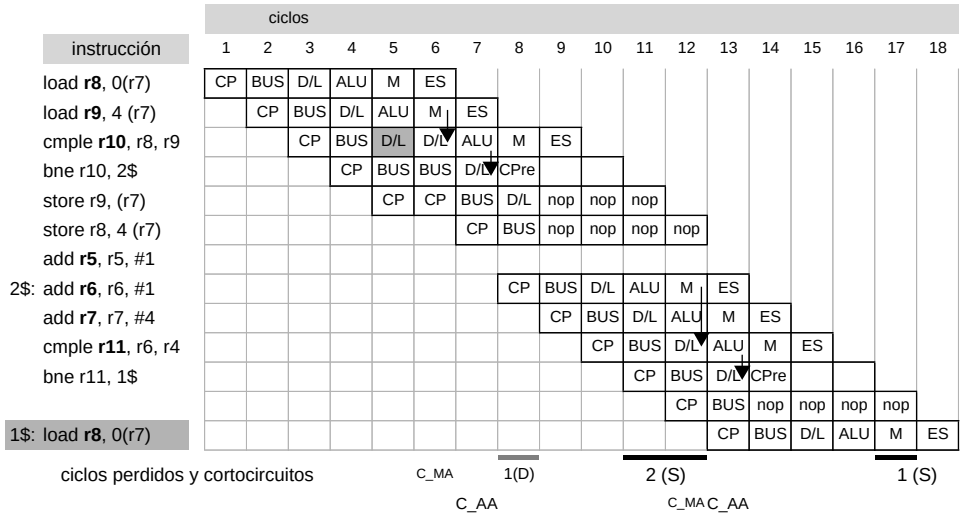
La diferencia con el caso anterior radica en que la predicción de sentido en la instrucción bne r10, 2$ es errónea.

En la siguiente figura se muestra la interpretación segmentada de una iteración del código. En la instrucción bne r10, 2$ se predice seguir en secuencia, ya que el literal es positivo. Como se cumple la condición se produce un error de predicción de sentido.

En el ciclo 5 se detecta un riesgo de datos debido al registro r9. En el ciclo 6 se utiliza un cortocircuito desde la etapa M a la etapa D/L y se prosigue la interpretación de instrucciones.

En el ciclo 7 se utiliza un cortocircuito desde la etapa ALU a la etapa D/L y se predice seguir en secuencia ya que el signo del desplazamiento es positivo. En el ciclo 8 se detecta un error de predicción de sentido y por tanto, se han perdido los ciclo 11 y 12.

En los ciclos 12 y 13 se utilizan respectivamente cortocircuitos desde las etapas M y ALU a la etapa D/L. En el ciclo 13 se predice modificar el secuenciamiento ya que el literal es negativo y en el ciclo 14 se comprueba que la predicción es correcta. Por tanto, sólo se pierden 1 ciclo por riesgo de secuenciamiento.



Los ciclos perdidos son

	Riesgos	
	datos	secuenciamiento
ciclos perdidos$\|_T$	1	3

Los ciclos de ejecución y el CPI cuando se toma el salto son:

En una iteración	salto tomado
$T_{seg}\|_T$ = número de instrucciones + ciclos perdidos	8 + 4 = 12 ciclos
$CPI_T = T_{seg}\|_T$ / (número de instrucciones)	12 / 8 = 1.5

Suponga que al ejecutar las N iteraciones del bucle interno del algoritmo de la burbuja se producen un 30% de intercambios.

Pregunta 4: *Determine el CPI medio en el procesador segmentado descrito en este capítulo.*

Respuesta: Una forma de efectuar el cálculo es dividir los ciclos totales ponderados por el número de instrucciones ponderado.

$$CPI = \frac{\sum f_i \times T_i}{\sum f_i \times N_i}$$

Sustituyendo tenemos

$$CPI = \frac{0.3 \times T_{seg}\big|_{NT} + 0.7 \times T_{seg}\big|_T}{0.3 \times N_{NT} + 0.7 \times N_T} = \frac{0.3 \times 13 + 0.7 \times 12}{0.3 \times 11 + 0.7 \times 8} = \frac{3.9 + 8.4}{3.3 + 5.6} = 1.38$$

Otra forma de efectuar el cálculo es desarrollar la expresión anterior para efectuar el cálculo con el CPI.

$$CPI = \frac{\sum f_i \times T_i}{\sum f_i \times N_i} = \sum f_i \times \frac{N_i}{N} \times \frac{T_i}{N_i} = \sum f_i \times \frac{N_i}{N} \times CPI_i$$

donde N es la suma de todos los N_i, el número de veces que hay que tenemos que tenerlos en cuenta (f_i). Esto es, $N = \sum f_i \times N_i$.

Entonces, para calcular el CPI medio hay que tener en cuenta la proporción de instrucciones en cada uno de los casos respecto del total y la fracción de veces que se toma o no se toma el salto.

$$CPI = \sum f_i \times \frac{N_i}{N} \times CPI_i$$

donde dado un CPI_i, f_i es la fracción de veces que hay que considerarlo, N_i es el número de instrucciones cuando se considera.

En este ejercicio hay dos valores de CPI. En la siguiente expresión se muestra el desarrollo de la expresión anterior donde se han sustituido los valores en el caso de que el salto sea tomado.

$$CPI = 0.3 \times \frac{N_{NT}}{0.3 \times N_{NT} + 0.7 \times N_T} \times CPI_{NT} + 0.7 \times \frac{8}{0.3 \times 11 + 0.7 \times 8} \times 1.5$$

Sustituyendo los restantes valores

$$CPI = 0.44 + 0.94 = 1.38$$

Pregunta 5: *Calcule la ganancia de la versión del procesador descrito en este capítulo respecto de la versión del procesador descrita en el Capítulo 3, suponiendo que los cortocircuitos incrementan el tiempo de ciclo en un 2%.*

Respuesta: La ganancia se calcula como el cociente de tiempos. El tiempo se calcula como los ciclos por la inversa de la frecuencia.

$$Ganancia = \frac{T_{seg3}}{T_{seg4}} = \frac{0.3 \times 26 + 0.7 \times 23}{0.3 \times 13 + 0.7 \times 12} \times \frac{1}{1.02} = \frac{7.8 + 16.1}{3.9 + 8.4} \times \frac{1}{1.02} = 1.90$$

donde T_{seg3} es el tiempo cuando se utiliza el procesador segmentado descrito en el Capítulo 3 y T_{seg4} es el tiempo cuando se utiliza el procesador segmentado descrito en este capítulo.

Por tanto, el camino de datos segmentado con cortocircuitos y predicción de sentido es un 90% más rápido que el camino de datos segmentado sin cortocircuitos y sin predicción de sentido.

Inserción de un elemento en una lista ordenada

En la parte izquierda de la siguiente figura se muestra un trozo de código que inserta un elemento en una lista ordenada de mayor a menor. La lista tiene como mínimo dos elementos y el elemento que se inserta es menor que el primero y mayor que el último elemento de la lista. Antes de iterar el bucle la variable q apunta al elemento que se quiere insertar y las variables p y prev apuntan al primer elemento de la lista.

En la parte derecha de la figura se muestra una traducción a lenguaje ensamblador. Los registros r9, r20 y r21 contienen las variables q, prev y p respectivamente. El registro r7 contiene el valor cero que se utiliza como codificación de null.

```
While (p!= null)
{    if (p->valor < q->valor)
     {    q->siguiente = p;
          prev->siguiente = q ;
          break ;
     }
     prev = p;
     p = p->siguiente ;
}
```

```
3$:  load r3, 0(r21)        contenido de p
     cmpeq r6, r3, r7       p!= null
     bne r6, 2$             ¿final de lista?
     load r1, 0(r3)         p->valor
     load r2, 0(r9)         q->valor
     cmple r5, r2, r1       p->valor ≥ q->valor
     bne r5, 1$             insertar si p->valor < q->valor
     load r10, 0(r20)       contenido de prev
     store r3, 8(r9)        q->siguiente = p
     store r9, 8(r10)       prev->siguiente = q
     br 2$                  salir del bucle
1$:  load r15, 8(r3)        contenido de p->siguiente
     store r3, 0(r20)       prev = p
     store r15, 0(r21)      p = p->siguiente
     br 3$                  iterar
2$:
```

En todas las preguntas supondremos que al interpretar la instrucción bne r6, 2$ no se cumple la condición.

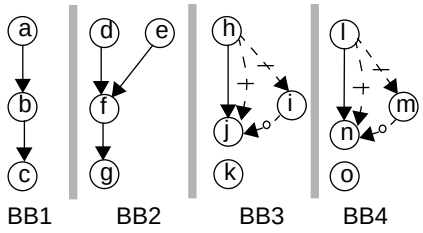
Cuando este código se interpreta en el procesador segmentado con control de riesgos descrito en el Capítulo 3 los ciclos de ejecución por iteración, los ciclos perdidos por tipo de riesgo y el CPI son:

sentido del salto bne r5, 1$	Tiempo de ejecución	CPI	Ciclos perdidos		
NT / T	32 ciclos	2.91	Datos	NT / T	9 ciclos
			Secuenciamiento	NT / T	12 ciclos
			Total	NT / T	21 ciclos

Pregunta 1: *Construya el grafo de dependencias de datos del cuerpo del bucle.*

Respuesta: En la siguiente figura se muestra el grafo de dependencias de cada bloque básico y las dependencias entre BB. Las líneas a trazos son dependencias debidas a memoria.

```
3$:  a. load r3, 0(r21)
     b. cmpeq r6, r3, r7      BB1
     c. bne r6, 2$
     d. load r1, 0(r3)
     e. load r2, 0(r9)        BB2
     f. cmple r5, r2, r1
     g. bne r5, 1$
     h. load r10, 0(r20)
     i. store r3, 8(r9)       BB3
     j. store r9, 8(r10)
     k. br 2$
1$:  l. load r15, 8(r3)
     m. store r3, 0(r20)      BB4
     n. store r15, 0(r21)
     o. br 3$
2$:
```



Seguidamente evaluamos el rendimiento en el procesador descrito en este capítulo.

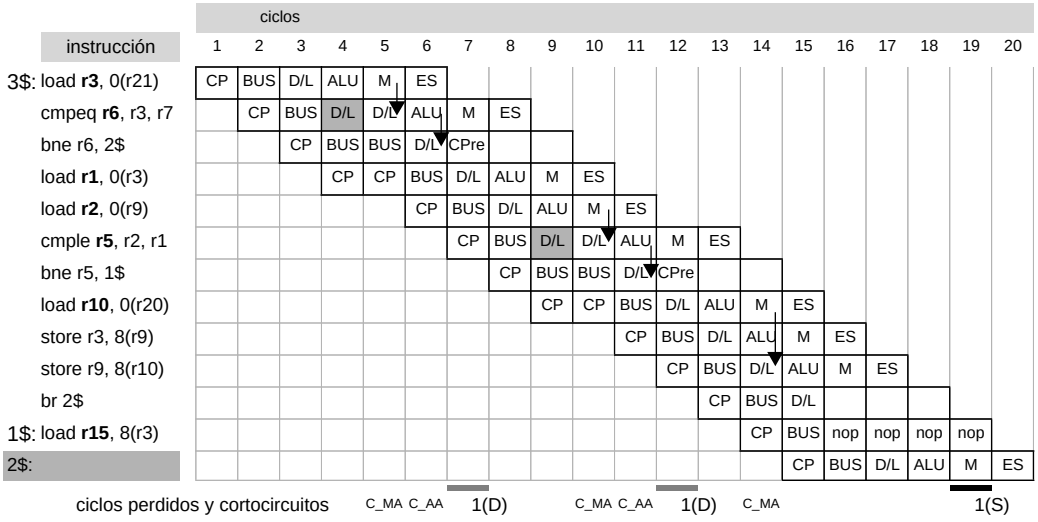
Pregunta 2: *Calcule los ciclos de ejecución en la iteración que se inserta el elemento. Así mismo, indique los ciclos perdidos por tipo de riesgo y calcule el CPI por iteración.*

Respuesta: Cuando el elemento se inserta en la lista, al interpretar la instrucción bne r5,1$ no se cumple la condición. En la siguiente figura se muestra mediante un diagrama temporal la interpretación segmentada de este caso.

En el ciclo 4 se detecta un riesgos de datos debidos al registro r3 que dura 1 ciclo. En el ciclo 5 se utiliza un cortocircuito desde la etapa M a la etapa D/L para obtener el valor que se almacena en r3. En el ciclo 6 se predice seguir en secuencia ya que el signo del literal es positivo. En el ciclo 7 se comprueba que la predicción ha sido correcta.

En el ciclo 9 se detecta un riesgo de datos debido al registro r2 y se pierde 1 ciclo. En el ciclo 10 se utiliza un cortocircuito desde la etapa M a la etapa D/L para obtener el valor que se almacena en r2. En el ciclo 11 se predice seguir en secuencia y se comprueba en el siguiente ciclo que la predicción es correcta.

En el ciclo 14 se utiliza un cortocircuito desde la etapa M a la etapa D/L para obtener el valor del registro r10. En el ciclo 16 se produce un riesgo de secuenciamiento y se pierde 1 ciclo.



Los ciclos perdidos son

	Riesgos	
	datos	secuenciamiento
ciclos perdidos$\|_{NT}$	2	1

Los ciclos de ejecución en un camino de datos segmentado y el CPI cuando no se toma el salto son:

En una iteración	salto no tomado
$T_{seg}\|_{NT}$ = número de instrucciones + ciclos perdidos	11 + 3 = 14 ciclos
$CPI_{NT} = T_{seg}\|_{NT}$ / (número de instrucciones)	14 / 11 = 1.27

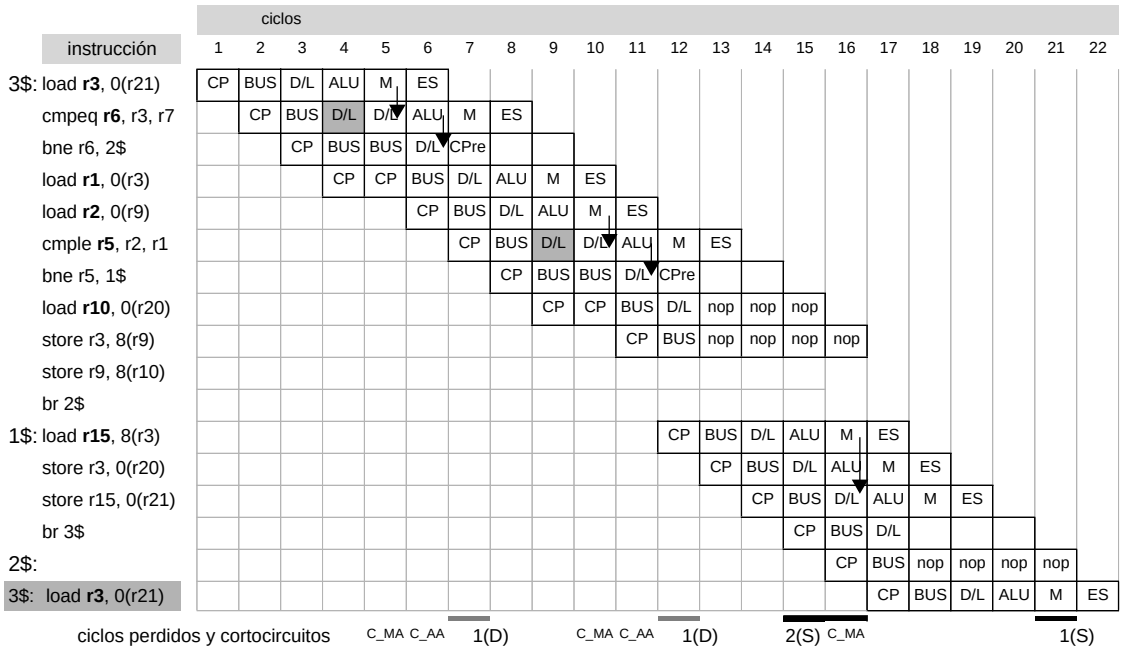
Pregunta 3: *Calcule los ciclos de ejecución en una iteración en la cual no se inserta el elemento. Así mismo, indique los ciclos perdidos por tipo de riesgo y calcule el CPI por iteración.*

Respuesta: Cuando el elemento no se inserta en la lista, al interpretar la instrucción bne r5,1$ se cumple la condición. Esto es, se modifica el secuenciamiento. En la siguiente figura se muestra mediante un diagrama temporal la interpretación segmentada de este caso.

La diferencia con la pregunta anterior radica en que se modifica el flujo de interpretación a partir de la instrucción de secuenciamiento bne r5, 1$. Entonces, como la predicción es seguir en secuencia se produce un error de predición y hay que recuperarse.

Analizando a partir de la instrucción bne r5, 1$ tenemos que se detecta un error de predicción de sentido en el ciclo 12. La acción de recuperación requiere descartar las dos siguientes instrucciones. En el ciclo 16 se obtiene el valor del registro r15 mediante un cortocircuito desde la etapa M a la etapa D/L.

En el ciclo 17 se produce un riesgo de secuenciamiento, siendo la penalización 1 ciclo.



Los ciclos perdidos son

	Riesgos	
	datos	secuenciamiento
ciclos perdidos$\|_T$	2	3

Los ciclos de ejecución en un camino de datos segmentado y el CPI cuando se toma el salto son:

En una iteración	salto tomado
$T_{seg}\|_T$ = número de instrucciones + ciclos perdidos	11 + 5 = 16 ciclos
$CPI_T = T_{seg}\|_T$ / (número de instrucciones)	16 / 11 = 1.45

Antes de empezar la inserción de un elemento el tamaño de la lista son 80 elementos y la inserción se efectúa después de haber recorrido en media un 40% de la lista.

Pregunta 4: *Determine el CPI medio en el procesador segmentado descrito en este capítulo.*

Respuesta: Para calcular el CPI medio hay que tener en cuenta la proporción de instrucciones en cada uno de los casos respecto del total y la fracción de veces que se recorre la lista y se inserte el elemento y el caso en que no se inserte el elemento.

$$CPI = \frac{N_{NT}}{N} \times CPI_{NT} + \frac{N_T}{N} \times CPI_T$$

donde N_{NT} y N_T son respectivamente el número de instrucciones cuando se inserta el elemento y cuando no se inserta y N es igual a $N = N_{NT} + N_T$, siendo $N_{NT} = 80 \times 0.4 \times 11 = 352$ y $N_T = 11$.

Por tanto,

$$CPI = \frac{80 \times 0.4 \times 11}{363} \times 1.27 + \frac{11}{363} \times 1.45 = 1.28$$

Pregunta 5: *Calcule la ganancia que se obtiene cuando se utiliza, para interpretar las instrucciones, el procesador descrito en este capítulo respecto del procesador descrito en el Capítulo 3. La frecuencia de reloj es la misma en los dos procesadores.*

Respuesta: La ganancia se calcula como el cociente de tiempos. Sin embargo, como el número de instrucciones que se ejecutan y la frecuencia de reloj es la misma la calculamos como el cociente de CPI.

$$Ganancia = \frac{CPI_{seg}}{CPI} = \frac{2.91}{1.27} = 2.27$$

donde CPI_{seg} es el CPI cuando se utiliza el procesador segmentado descrito en el Capítulo 3.

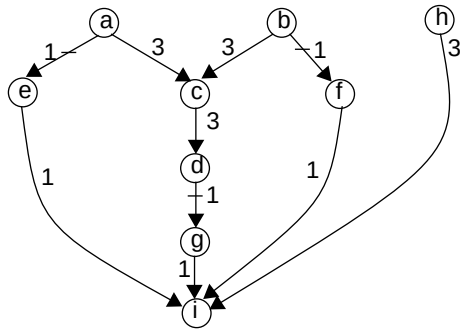
Por tanto, el camino de datos segmentado con cortocircuitos y predicción de sentido es 127% más rápido que el camino de datos segmentado sin cortocircuitos y sin predicción de sentido.

Planificación sin limitación de recursos

En el procesador descrito en este capítulo el retardo productor-uso de todas las instrucciones que actualizan el banco de registros es 3.

En la siguiente figura se muestra un código y el grafo de dependencias. Los arcos del grafo de dependencias se han etiquetado con el retardo productor-uso de las instrucciones que producen el resultado. Las antidependencias se etiquetan con latencia 1, ya que se inicia la interpretación de una instrucción por ciclo y si se mantiene el orden de programa esta dependencia no produce riesgo en un camino de datos segmentado lineal.

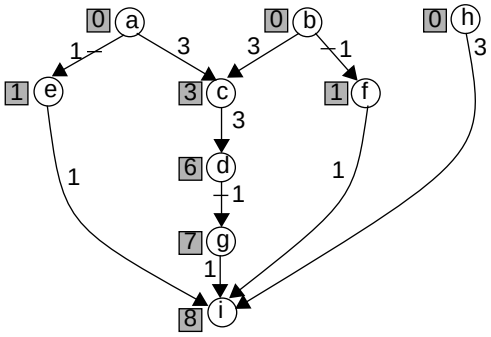
1$: a. load **r1**, 0(r2)
b. load **r3**, 0(r4)
c. add **r5**, r1, r3
d. store r5, 0(r6)
e. add **r2**, r2, #8
f. add **r4**, r4, #8
g. add **r6**, r6, #8
h. sub **r9**, r9, #1
i. bne r9, 1$



Pregunta 1: *Calcule el tiempo más pronto posible (TMC) en el cual puede planificarse cada uno de los nodos del grafo de dependencias. En otras palabras, calcule el instante de tiempo, desde el*

inicio de interpretación del código, en el cual se dispone de los datos para ejecutar la instrucción representada por el nodo. Para ello empiece por los nodos sin dependencias (raíces) y vaya acumulando el valor de las etiquetas de los arcos a medida que recorre el árbol hasta la hoja.

Respuesta: El árbol se recorre desde los nodos libres de dependencias hasta el nodo hoja. En el recorrido se etiquetan los nodos acumulando desde la raíz elegida el valor de las etiquetas de los arcos. Cuando en un nodo confluyen varios caminos se elige el valor máximo de los valores acumulados.



Pregunta 2: *El tiempo de ejecución mínimo de la secuencia de instrucciones es el tiempo de ejecución suponiendo recursos ilimitados.*

Suponga que el tiempo de ejecución de la instrucción i son 4 ciclos. ¿Cuál es el tiempo mínimo?.

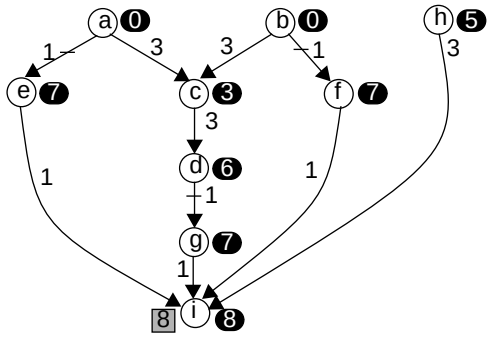
Así mismo, indique el orden y el instante de tiempo en el que se iniciará la ejecución de cada una de las instrucciones.

Respuesta: El valor con el cual se etiqueta la hoja del árbol es el tiempo mínimo de ejecución, con recursos ilimitados, hasta la hoja. Como la latencia de ejecución de la instrucción i son 4 ciclos, el tiempo de ejecución es 8 + 4 = 12.

Las instrucciones se ejecutan en paralelo de la siguiente forma (a, b, h), (e, f), c, d, g, i, en los instantes de tiempo t = 0, 1, 3, 6, 7 y 8 respectivamente.

Pregunta 3: *Calcule el tiempo más tardío (TMT) en el cual puede planificarse cada uno de los nodos del grafo de dependencias para que no se incremente el tiempo de interpretación. En otras palabras, calcule el instante de tiempo más tarde posible, desde el inicio de interpretación del código, en el cual hay que interpretar una instrucción para que no se incremente el tiempo de ejecución. Para ello recorra el árbol de dependencias desde la hoja hasta las raíces y utilice como tiempo de la hoja el TMC calculado previamente.*

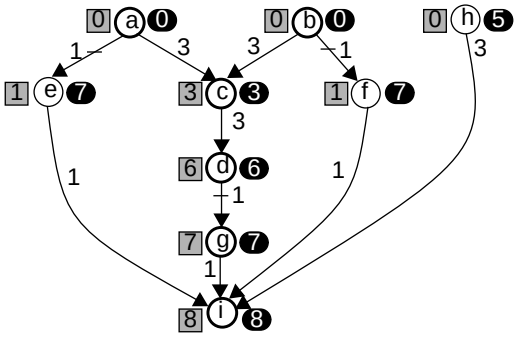
Respuesta: Utilizando como valor inicial el TMC del nodo hoja se empiezan a etiquetar los nodos restando las latencias que etiquetan los arcos. Cuando en un nodo confluyen varios caminos se elige el valor mínimo. En la figura los nodos están etiquetados con TMT.



Pregunta 4: *Describa como determinar el camino crítico en el grafo de dependencias etiquetado con TMC y TMT. Por camino crítico se entiende una secuencia de instrucciones en las que el valor de TMC y TMT es el mismo.*

Indique también el intervalo de tiempo disponible para planificar las instrucciones que no están en el camino crítico.

Respuesta: Cuando en un nodo las etiquetas TMT y TMC tienen el mismo valor el nodo está en el camino crítico.



En el ejemplo el camino crítico lo constituye la secuencia de instrucciones (a, b, c, d, g, i).

La diferencia (TMT - TMC) indica un margen de tiempo durante el cual puede iniciarse la interpretación del nodo sin que se incremente el tiempo de ejecución del código. Las instrucciones e, f y h tienen intervalos de inicio de ejecución de 6, 6 y 5 unidades de tiempo respectivamente.

EJERCICIOS

Ejercicio 4.1

Un procesador, que interpreta las siguientes instrucciones

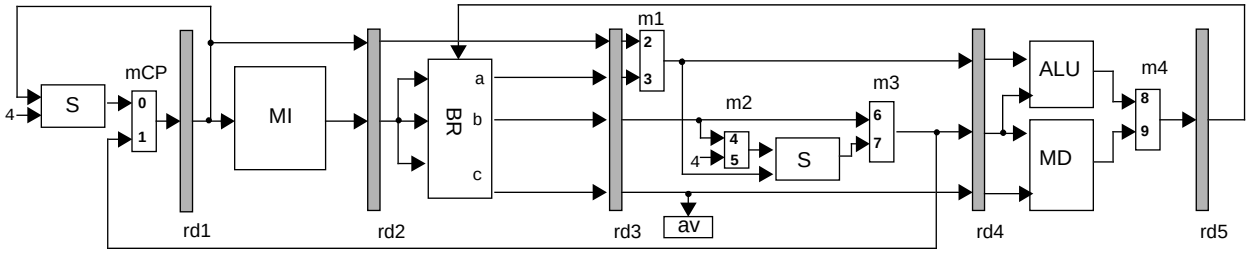
Tipo	Descripción
INT	Rc = Ra op Rb
Load	Rc = Mem[Ra + Rb]
Store	Mem[Ra + Rb] = Rc
Br	si cond (Rc) entonces CP = Rb en caso contrario CP = CP + 4

está segmentado linealmente en 6 etapas:

ETAPA	FUNCIONALIDAD
CP	determinar la dirección de la instrucción
B	búsqueda de la instrucción
DL	decodificación, detección de riesgos, lectura de operandos en registros
X1	cálculo de la dirección efectiva (Load/Store). Evaluación de la condición y selección de la dirección de la siguiente instrucción que se interpreta (BR)
X2	operación aritmético-lógica, acceso a memoria de datos (Load, Store)
E	escritura en el banco de registros (INT, Load)

El banco de registros permite la escritura y la lectura, en este orden, de un mismo registro en un ciclo de reloj. No hay cortocircuitos. El camino de datos dispone de recursos suficientes para que no se produzcan riesgos estructurales.

La figura muestra el Camino de Datos del procesador (no se muestran los multiplexores que permiten retener la información o inyectar una instrucción nop en los registros de desacoplo).



Pregunta 1: *Indique las entradas de los multiplexores del camino de datos que se han de seleccionar para cada tipo de instrucción en el ciclo 4 (m1, m2, m3, mCP) y en el ciclo 5 (m4) del proceso de*

interpretación.
Por ejemplo, para una instrucción Store, donde la marca x denota indistinto:

				ciclo 4		ciclo 5
Instrucción	m1	m2	m3	mCP	m4
Store	3	4	7	0	x

En el camino de datos, las etapas que determinan el tiempo de ciclo del reloj son CP, X1 y X2. El retardo (en ps) de cada componente es:

Componente	retardo
S	250 (sumador)
ALU	400 (unidad aritmético-lógica)
MD	450 (memoria de datos)
AV	150 (evaluador de condiciones)
mx	150 (multiplexor)
rd_i	100 (registro de desacoplo), donde rd1 es el registro CP

el decodificador y los circuitos que generan las señales de control: 0

Pregunta 2: *Indique el camino crítico del camino de datos enumerando los componentes que lo constituyen. Calcule el periodo mínimo de la señal de reloj.*

En la etapa DL se detectan los riesgos de datos y los riesgos de secuenciamiento. Cuando se detecta un riesgo de datos, el procesador bloquea la interpretación de las instrucciones que están en las etapas DL, B y CP mientras persiste el riesgo. En caso de riesgo de secuenciamiento, el procesador descarta las instrucciones buscadas hasta que se actualiza el Contador de Programa con la dirección de la siguiente instrucción (etapa X1).

El procesador ejecuta el siguiente programa, que suma los elementos de una lista y cuenta el número de elementos iguales a cero.

```
for (; p!=NULL; p=p->next)
{
   sum=sum+p->dat
   if (p->dat == 0)
   {
      z=z+1;
   }
}
```

```
1$:  load r1, (r0 + r10)     ;r1 ← mem[r0+r10]
     add r2, r2, r1          ;r2 ← r2 + r1
     bne r1, r11             ;si (r1≠0) saltar a 2$
     add r3, r3, r12         ;r3 ← r3 + r12
2$:  load r0,(r0 + r14)      ;r0 ← mem[r0+r14]
     bne r0, r13             ;si (r0≠0) saltar a 1$
```

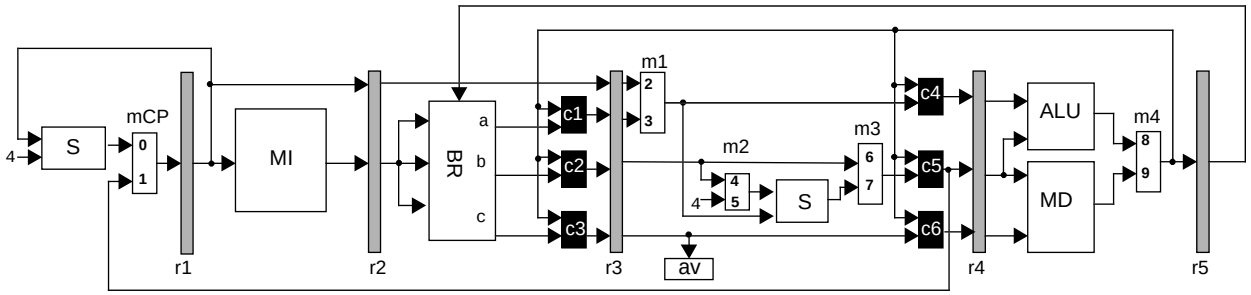
Valores iniciales:

r0 = dirección del primer elemento de la lista
r2 = 0; r3 = 0; r10 = 0; r11 = 2$; r12 = 1; r13 = 1$; r14 = 8.

Pregunta 3: *Muestre el cronograma de ejecución de las instrucciones de una iteración entera del bucle y de la primera instrucción de la siguiente iteración, en el caso en que p->dat == 0. Identifique los ciclos perdidos en el cronograma e indique el motivo.*

Pregunta 4: *Calcule el CPI medio suponiendo que la lista consta de 100 elementos, de los cuales 10 son cero.*

Modificamos el camino de datos añadiendo los cortocircuitos que se muestran en la figura.



Pregunta 5: *Para cada una de las cinco parejas de instrucciones consecutivas, determine el multiplexor del mejor cortocircuito mediante el cual la segunda instrucción obtiene el operando fuente r0 y el número de ciclos que se ha de bloquear esta instrucción.*

Add r0,r1,r2 Add r1,r0,r1	Add r0,r1,r2 Store r0,(r1+r2)	Add r0,r1,r2 Store r1,(r0+r2)	Load r0,(r1+r2) Bne r0,r1	Load r0,(r1+r2) Bne r1,r0

Pregunta 6: *Indique el camino crítico del nuevo camino de datos enumerando los componentes que lo constituyen. Suponga que el retardo de los circuitos de control de los multiplexores de cortocircuitos es cero. Calcule el periodo mínimo de la señal de reloj.*

El procesador con cortocircuitos ejecuta el programa anterior.

Pregunta 7: *Muestre el cronograma de ejecución de las instrucciones de una iteración entera del bucle y de la primera instrucción de la siguiente iteración en el caso en que p->dat == 0. Identifique los cortocircuitos utilizados, los ciclos perdidos en el cronograma e indique el motivo.*

Pregunta 8: *Calcule el CPI medio suponiendo que la lista consta de 100 elementos, de los cuales 10 son cero.*

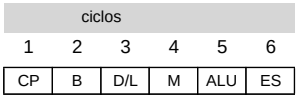
Pregunta 9: *Calcule la ganancia obtenida.*

Ejercicio 4.2

Se quiere diseñar un procesador segmentado para ejecutar, entre otras, las siguientes instrucciones:

Tipo	Descripción
cálculo	Rk = Ri op Rj
Load	Rk = Mem[Ri] Ri = Ri + Rj ; siempre Ri ≠ Rk
Store	Mem[Ri] = Rk Ri = Ri + Rj

El procesador está segmentado en 6 etapas:



donde

ETAPA	FUNCIONALIDAD
CP	determinar la dirección de la instrucción
B	búsqueda de la instrucción
D/L	decodificación, detección de riesgos, lectura de operandos en registros
M	acceso a memoria (Load, Store)
ALU	operación aritmético-lógica
ES	escritura en el banco de registros (al final del ciclo)

Cuando se produce un riesgo el procesador bloquea la interpretación de instrucciones en las etapas D/L, B y CP hasta que desaparece el riesgo.

Pregunta 1: *Diseñe el camino de datos para poder ejecutar las 3 clases de instrucciones (no es necesario especificar las etapas CP y B). El camino de datos tendrá que disponer de los recursos suficientes para evitar riesgos estructurales. En este apartado no considere posibles cortocircuitos.*

Pregunta 2: *Indique los casos en que el procesador se bloquea debido a un riesgo por dependencias de datos. Muestre con un ejemplo de código cada situación junto con un cronograma de ejecución. En este apartado no considere posibles cortocircuitos.*

Pregunta 3: *Para cada uno de los casos del apartado anterior, especifique los cortocircuitos que resuelven los riesgos de datos. Indique para cada cortocircuito las etapas fuente y destino y la reducción en ciclos de bloqueo que se obtiene.*

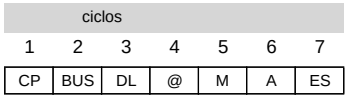
Ejercicio 4.3

Tenemos que ejecutar el código siguiente:

valores iniciales: r0 = 100, r1 = 200, r2 = 5

1	r4 := M[r0 + 0]	LD	ra : = M[rb + cte]
2	r3 := r2 - M[r0 + 4]	OPRM	rc := ra op M[rb + cte]
3	r0 := r0 + 8	OP	rc := ra op cte
4	r5 := r3 - r4	OP	rc := ra op rb
5	M[r1 + 0] := r5	ST	M[rb + cte] := ra
6	r1 := r1 - 4	OP	rc := ra op cte
7	saltar a 1 si (r1 > r0)	BRC	saltar si condición (ra , rb)

Tenemos un procesador segmentado lineal con 7 etapas, y sin riesgos estructurales



- la verificación de riesgos y, en su caso, el bloqueo, se hace en la etapa DL
- el cálculo de la dirección efectiva [rb + cte] se hace en la etapa @
- el acceso a la cache de datos se hace en la etapa M

- el cálculo de cualquier op se hace en la etapa A
- la evaluación de condición y la escritura del CP por salto se hace en la etapa A

Este procesador interpreta las instrucciones BRC con predicción fija No Saltar.

Este procesador puede escribir (E) y leer (L) un registro durante el mismo ciclo.

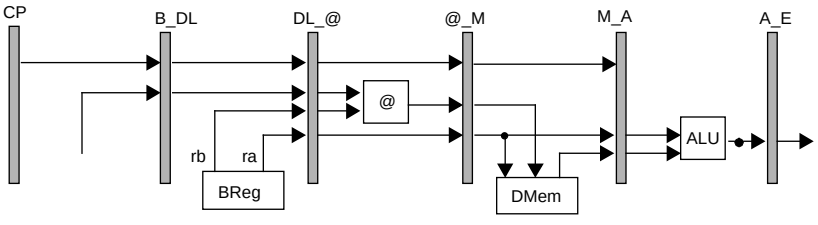
Pregunta 1: *Calcule, mediante expresiones aritméticas, el número de iteraciones de este bucle.*

Pregunta 2: *Presente el Grafo de Dependencias completo de las 7 instrucciones de este bucle.*

Pregunta 3: *La cache de instrucciones es ideal, pero la cache de datos tiene un fallo después de cada 5 aciertos consecutivos en los accesos. Consideramos un acceso cada lectura o cada escritura en una dirección efectiva. Suponga que el primer acceso es un acierto.*
Calcule el cociente entre accesos fallados y accesos totales (tasa de fallos) tras completarse todas las iteraciones del bucle.
Calcule cuantos fallos ocurren al ejecutarse instrucciones ST.

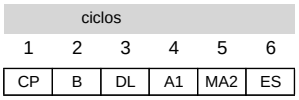
Pregunta 4: *El procesador no tiene ningún cortocircuito. Cada fallo en la cache de datos provoca bloqueo durante 10 ciclos. Calcule, sin presentar ningún cronograma, el número total de ciclos que tarda en ejecutar todas las iteraciones del bucle.*

Pregunta 5: *Ahora tenemos que añadir un cortocircuito, que debe acabar al final de la etapa DL. Presente el cronograma de una iteración y el inicio de la siguiente, para mostrar el efecto de usar ese cortocircuito. Suponga aquí que no hay ningún fallo en la cache de datos. Calcule el número de ciclos en que se reduce ahora la ejecución de cada iteración del bucle. Dibuje ese cortocircuito sobre el esquema parcial de Camino de Datos que se le facilita.*



Ejercicio 4.4

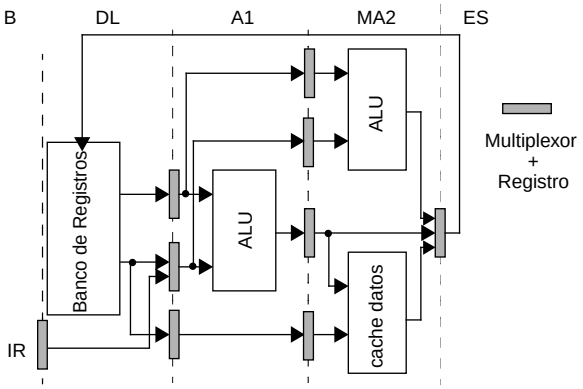
Tenemos un procesador segmentado lineal de 6 etapas:



donde:

ETAPA	FUNCIONALIDAD
CP	determinar la dirección de la instrucción
B	búsqueda de la instrucción
DL	decodificación, detección de riesgos, lectura de operandos en registros en la segundo semiciclo
A1	cálculo de la dirección de memoria, operación aritmética o lógica
MA2	acceso a memoria, operación aritmético-lógica
ES	escritura en el banco de registros en el primer semiciclo

La figura muestra el esquema del camino de datos, que no incluye las etapas CP y B ni los cortocircuitos. El registro IR es el registro de desacoplo entre la etapa B y DL.



Las instrucciones aritmético-lógicas efectúan el cálculo en la etapa A1 si los dos operandos fuente están disponibles. En el caso de que algún operando no esté calculado, el procesador puede enviar a ejecutar la instrucción y realizar el cálculo en la etapa MA2.

Dada la siguiente secuencia de instrucciones:

```
load r0, 0 (r0)      ;r0 ← mem[r0+0]
load r1, 0 (r1)      ;r1 ← mem[r1+0]
add r2, r0, r1       ;r2 ← r0 + r1
add r1, r1, r2       ;r1 ← r1 + r12
store r1, 0 (r0)     ;mem[r0+0] ← r1
```

Pregunta 1: *Muestre el cronograma de ejecución de las 5 instrucciones, deduciendo de forma clara los cortocircuitos utilizados. Etiquete con un identificador cada cortocircuito y muéstrelo en el camino de datos. Los cortocircuitos pueden tener cualquier etapa destino.*

Pregunta 2: *Repita el apartado anterior, sustituyendo las dos últimas instrucciones del código por las siguientes instrucciones*

load **r2**, 0 (r2)
store r2, 0 (r1)

Ejercicio 4.5

Tenemos un procesador segmentado lineal que, entre otras instrucciones, tiene los siguientes tres tipos de instrucciones:

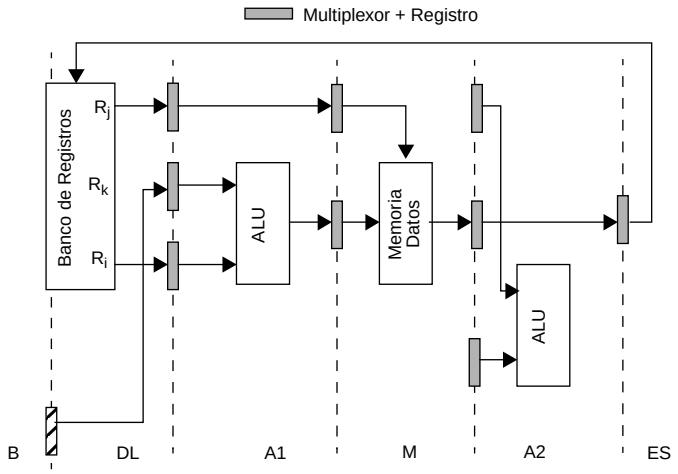
Descripción	Especificación	Semántica
Una instrucción para guardar en memoria el resultado de una operación aritmética	OPSTORE Rk, Ri, Rj, X	Mem[Ri + X]= (Rj op Rk)
Una instrucción para efectuar operaciones aritméticas con un operando en memoria	OPM Rk, Ri, Rj, X	Rk= Mem[Ri + X] op Rj
Una instrucción de acceso a memoria con post-incremento del registro base	LOADINC Rk, Ri, Rj, X	Rk = Mem[Ri + X] Ri= Ri + Rj

Todas las instrucciones pasan por las mismas 7 etapas:

ciclos						
1	2	3	4	5	6	7
CP	B	DL	A1	M	A2	ES

La escritura en el banco de registros ocupa todo el ciclo. Los riesgos se analizan en la etapa DL, si no se pueden resolver el procesador se bloquea.
Conocemos que las instrucciones OPM realizan la operación en la etapa A2, las LOADINC efectúan el post-incremento en la etapa A2 y las de tipo OPSTORE realizan todos los cálculos en la etapa A1.

Parte del camino de datos del procesador se muestra en la siguiente figura.



Pregunta 1: *Complete el camino de datos para que permita ejecutar los tres tipos de instrucciones descritas sin riesgos estructurales. Sólo hay que tener en cuenta las etapas mostradas en el camino de datos. Indique así mismo, el número de multiplexores y registros añadidos, el número de ALU añadidas y el número de caminos de acceso al banco de registros añadidos.*

Pregunta 2: *Ejecute el siguiente código en el procesador descrito utilizando todos los cortocircuitos que reduzcan el tiempo de ejecución. Indique de forma clara las acciones que se efectúan en cada ciclo y etiquete los cortocircuitos añadidos al camino de datos.*

(OPM)	R1 = Mem[R2 + X0] + R0
(LOADINC)	R3 = Mem[R2 + X1]
	R2 = R2 + R1
(OPM)	R4 = Mem[R2 + X2] + R3
(OPSTORE)	Mem[R4 + x3] = R3 + R2

Suponga que el procesador dispone de cortocircuitos desde cualquier etapa a cualquier etapa.

Pregunta 3: *¿ En qué etapa sería mejor efectuar el post-incremento para las operaciones de tipo* `LOADINC`*?. ¿Cuál es la razón?.*

Ejercicio 4.6

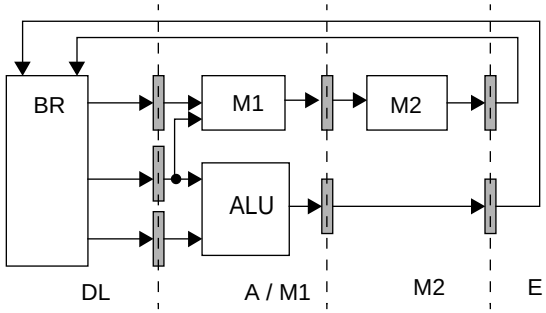
Tenemos un procesador segmentado lineal, en el que los accesos a memoria tardan 2 ciclos y están totalmente segmentados, con las siguientes etapas.

ETAPA	FUNCIONALIDAD
CP	determinar la dirección de la instrucción
B	búsqueda de la instrucción
DL	decodificación, detección de riesgos, lectura de operandos en registros
A / M1	operación aritmética o lógica, 1ª etapa del acceso a memoria
M2	2ª etapa del acceso a memoria
E	escritura en el banco de registros

Distinguiremos los siguientes 3 tipos de instrucciones de lenguaje máquina.

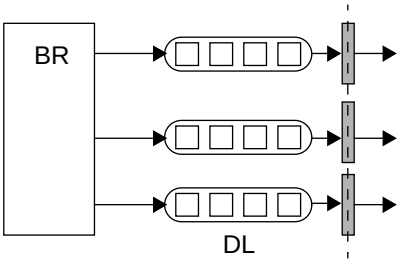
Especificación	Semántica
oper Ri, Rk, Rj	Ri <- Rk op Rj
load Ri, Rk, Rj (Ri< > Rk)	1) Ri <- Mem (Rk) 2) Rk <- Rk + Rj
store Ri, Rk, Rj	1) Mem (Rk) <- Ri 2) Rk <- Rk + Rj

La siguiente figura muestra el camino de datos, que no incluye las etapas CP y B.

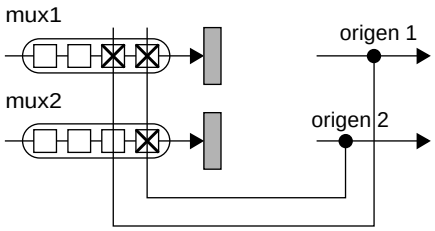


- La unidad funcional ALU efectúa todas la operaciones de cálculo.
- La unidad memoria ejecuta las instrucciones load y store y está segmentada en 2 etapas M1 y M2.
- El banco de registros (BR) dispone de 3 caminos de lectura y 2 caminos de escritura. El BR permite escribir y leer, en este orden, un registro en un mismo ciclo.

La situación de los multiplexores de cortocircuito está preestablecida al final de la etapa DL, tal como se muestra en la siguiente figura.



Cuando se solicite, es necesario indicar claramente el origen (•) y el destino (**X**) de los cortocircuitos. En el siguiente ejemplo: mux1 es un multiplexor donde llegan 2 cortocircuitos (de origen 1 y origen 2) y mux 2 es un multiplexor donde sólo llega el cortocircuito con origen 2. Cada multiplexor selecciona entre la salida del banco de registros y hasta 4 cortocircuitos.



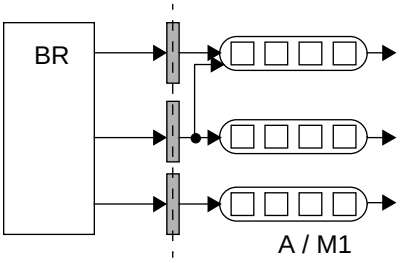
Disponemos de la siguiente información de retardos:

- El banco de registros suministra los datos necesarios 55 nanosegundos después de iniciarse el ciclo.
- Un registro de desacoplo tiene un retardo de 10 nanosegundos.
- Un multiplexor tiene un retardo de 15 nanosegundos.
- La ALU tiene un retardo de 30 nanosegundos.
- La lógica M1 de la memoria tiene un retardo de 40 nanosegundos.
- La lógica M2 de la memoria tiene un retardo de 50 nanosegundos.
- La detección y control de riesgos se efectúa en la etapa DL en paralelo con el acceso al banco de registros y no afecta al tiempo de ciclo.

Pregunta 1: *Dibuje los cortocircuitos necesarios para minimizar los bloqueos debidos a dependencias de datos.*

Pregunta 2: *Las etapas que influyen en el tiempo de ciclo son DL, A/M1 y M2. Calcule el tiempo que necesita cada etapa y el tiempo de ciclo mínimo del procesador.*

Para reducir el tiempo de ciclo, se propone situar los multiplexores de cortocircuito al principio de la etapa A/M1, tal como se muestra en la siguiente figura.



Pregunta 3: *Para la nueva organización, dibuje los cortocircuitos necesarios para minimizar los bloqueos debidos a dependencias de datos.*

Pregunta 4: *Calcule el nuevo tiempo de cada etapa y el nuevo tiempo de ciclo mínimo.*

Ejercicio 4.7

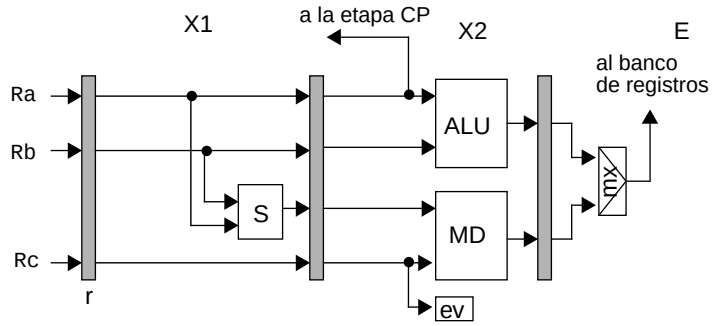
El lenguaje máquina contiene 4 tipos de instrucciones:

Tipo	Descripción
Load	Rc = M[Ra+Rb]
Store	M[Ra+Rb] = Rc
Op	Rc = Ra op Rb
Br	si cond (Rc) entonces CP = Ra

Segmentamos el camino de datos del procesador en las siguientes etapas:

ETAPA	FUNCIONALIDAD
CP	determinar la dirección de la instrucción
B	búsqueda de la instrucción
DL	decodificación, detección de riesgos, lectura de operandos en registros
X1	cálculo de la dirección efectiva (Load/Store)
X2	operación aritmético lógica (Op), acceso a memoria (Load/Store), evaluación de la condición y modificación del CP con la dirección destino del salto si se cumple la condición (Br)
E	escritura en el banco de registros

La figura muestra parte del camino de datos del procesador (no incluye las etapas CP, B y DL). El banco de registros permite la escritura y lectura de un mismo registro en un ciclo de reloj.

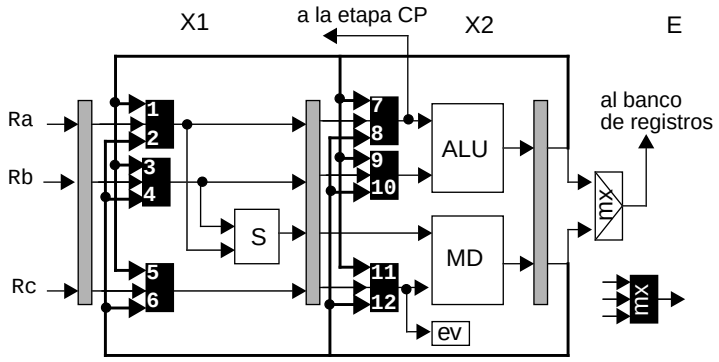


En este procesador, las etapas que determinan el tiempo de ciclo de reloj son X1 y X2. El tiempo (nanosegundos) de cada componente es:

recurso	retardo (ns)
S	2 (sumador)
ALU	3 (unidad arit-lógica)
MD	4.5 (memoria de datos)
EV	1 (evaluador de condiciones)
mx	1 (multiplexor)
r	0.5 (registro de desacoplo)

Pregunta 1: *Indique para cada tipo de instrucción el tiempo mínimo que se requiere en las etapas X1 y X2.*
Calcule la frecuencia máxima de la señal de reloj.

Modificamos el camino de datos añadiendo 12 cortocircuitos tal como se muestra en la figura. Observe que los multiplexores para encaminar los datos de los cortocircuitos están ubicados al principio de las etapas.



Pregunta 2: *Evalúe el retardo de las etapas X1 y X2 y la frecuencia máxima de la señal de reloj en el procesador con cortocircuitos.*

Para las instrucciones de salto condicional se utiliza predicción fija No Saltar. La verificación de la predicción se efectúa en la etapa X2.

Pregunta 3: *Calcule las penalizaciones (ciclos perdidos) cuando se ejecuta un salto condicional predicho correctamente y cuando se ejecuta un salto mal predicho.*

El programa de prueba opera sobre una lista de 100 elementos:

```
for (; p!=NULL; p=p->next)
{
    s=s+p->dat ;
    p->dat = s ;
}
```

```
1$:  load r4, (r0 + r1)      ;r4 ← mem[r0+r1]
     add r3, r4, r3          ;r3 ← r4 + r3
     store r3, (r0 + r1)     ;mem[r0+r1] ← r3
     load r0, (r0 + r2)      ;r0 ← mem[r0+r2]
     bne r0, r9              ;si (r0≠0) saltar a 1$
```

Valores iniciales:

r0 = dirección del primer elemento de la lista
r1 = 0; r2 = 8; r3 = 0; r9 = 1$

Pregunta 4: *Calcule los ciclos de ejecución por iteración. Dibuje primero el cronograma de ejecución de una iteración, indicando en cada ciclo de reloj los cortocircuitos utilizados. Para las instrucciones de salto condicional se utiliza predicción fija No Saltar.*

Pregunta 5: *Calcule el tiempo de ejecución del programa en el procesador con cortocircuitos y la ganancia que se obtiene respecto del procesador sin cortocircuitos. Los dos procesadores utilizan predicción fija No Saltar.*

Ejercicio 4.8

Un procesador segmentado lineal consta de las 9 etapas siguientes:

ETAPA	FUNCIONALIDAD
CP	determinar la dirección de la instrucción
B1	1º ciclo de búsqueda de la instrucción
B2	2º ciclo de búsqueda de la instrucción
D	decodificación, detección de riesgos
L	lectura de operandos en registros
O	operación, cálculo de la dirección efectiva o dirección destino del salto, evaluación de la condición y escritura del CP
M1	1º ciclo del acceso a la memoria de datos
M2	2º ciclo del acceso a la memoria de datos
E	escritura en el banco de registros

La lectura y escritura de un registro dura un ciclo de reloj. La cache de instrucciones y la cache de datos están segmentadas y tienen 2 ciclos de latencia.

El procesador ejecuta el siguiente programa:

```
for (; p!=NULL; p=p->next)
{
    if ( p->dat == N) z ++ ;
}
```

```
1$:  load r1, 0 (r0)      ;r1 ← mem[r0+0]
     sub r2, r1, r5       ;r2 ← r1 - r5
     bnez r2, 2$          ;si (r2≠0) salta a 2$
     add r6, r6, #1       ;r6 ← r6 + #1
2$:  load r0, 8 (r0)      ;r0 ← mem[r0+8]
     bnez r0, 1$          ;si (r0≠0) salta a 1$

     Valores iniciales:
     r0 = dirección del primer elemento de la lista
     r5 = N; r6 = 0
```

donde la lista es de 100 elementos, de los cuales el 10% son igual a la constante N.

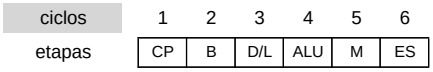
Pregunta 1: *El procesador se bloquea cuando detecta un riesgo de datos o un riesgo de secuenciamiento. Cuando se detecta un riesgo, el procesador inhibe la búsqueda y decodificación de nuevas instrucciones hasta que se resuelve el riesgo. El camino de datos no tiene cortocircuitos. Calcule el CPI.*

Pregunta 2: *El procesador dispone de los cortocircuitos necesarios para reducir o eliminar los ciclos de bloqueo por riesgos de datos y se bloquea al detectar un riesgo de secuenciamiento. Calcule el CPI.*

Pregunta 3: *Se añade en el camino de datos del procesador un sumador en la etapa D para calcular la dirección destino de un salto y se utiliza el siguiente mecanismo para ejecutar un salto: predice modificar el secuenciamiento implícito si el salto es hacia atrás; predice seguir en secuencia si el salto es hacia adelante. La verificación de la predicción se efectúa en la etapa O.*
El procesador dispone de los cortocircuitos necesarios. Calcule el CPI.

Ejercicio 4.9

En la siguiente figura se muestra la segmentación en etapas del proceso de interpretación de las instrucciones en un procesador.



La funcionalidad de las etapas y recursos básicos se describen en la siguiente tabla.

ETAPA	FUNCIONALIDAD
CP	determinar la dirección de la instrucción
B	búsqueda de la instrucción
D/L	decodificación, lectura de registros
ALU	operación aritmético-lógica
M	acceso a memoria de datos
ES	escritura en el banco de registros o en el registro CP

Las instrucciones de secuenciamiento, condicionales e incondicionales, actualizan el registro CP en la etapa ES. El conjunto de instrucciones del procesador puede interpretarse sin que se produzcan riesgos estructurales. En el mismo ciclo se puede escribir y leer, en este orden, un registro del banco de registros.

Riesgos y actuación

En la etapa D/L se detectan los riesgos de datos y de secuenciamiento y se actúa para respetar la semántica del lenguaje máquina.

Cuando se produce un riesgo de datos se bloquea la interpretación de la instrucción que ocupa la etapa D/L y esta retención se propaga a las instrucciones que están en las etapas previas. Durante los ciclos de bloqueo se inyectan, mediante circuitería, instrucciones NOP desde la etapa D/L hacia la etapa ALU.

Cuando se detecta un riesgo de secuenciamiento se suspende la interpretación de instrucciones posteriores a la instrucción de secuenciamiento. La suspensión perdura hasta que se actualiza el registro CP con la dirección de la siguiente instrucción que debe interpretarse. Las etapas CP y B siguen de forma autónoma calculando una dirección y buscando una instrucción. La instrucción buscada en memoria se descarta, ya que durante los ciclos de suspensión se inyectan, mediante circuitería, instrucciones nop desde la etapa B hacia la etapa D/L.

Código

En el procesador segmentado descrito se interpreta la siguiente secuencia de instrucciones.

```
        20    load R1, X(R2)
        24    load R9, X (R5)
        28    beq R1 1$
        32    load R4, X(R1)
1$:     36    add R6, R4, R1
        40    sub R7, R6, R8
        44    load R10, X(R6)
        48    add R11, R1, R10
```

Supondremos en todas la preguntas que el salto es no tomado. Esto es, no se cumple la condición.

Visualización del progreso de las instrucciones en un diagrama temporal

Para representar una instrucción que se retiene en una etapa se utilizan varias filas. Cada vez que se retiene una instrucción en una etapa se utiliza una nueva fila. En esta nueva fila se empieza la representación en el ciclo siguiente al de retención, indicando la etapa en la cual está la instrucción. Si se sigue produciendo una acción de retención se utiliza una nueva fila. Desde la etapa más

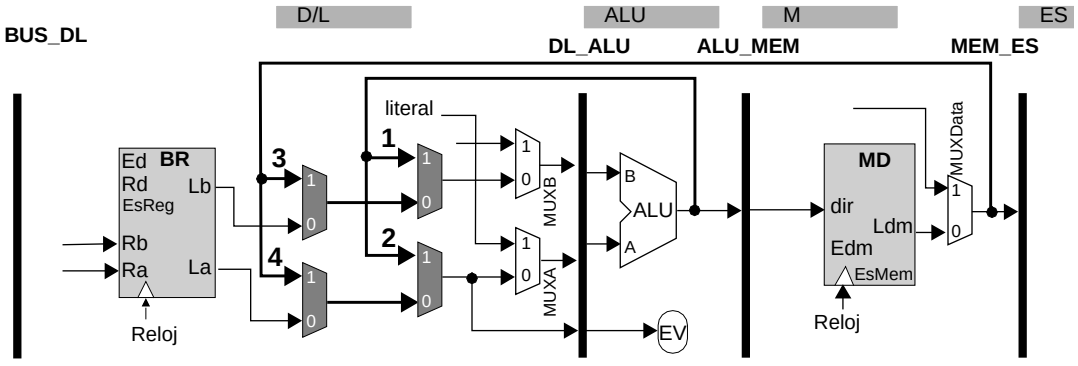
avanzada de la segmentación, en la cual se produce la retención, se inyecta una instrucción nop al finalizar el ciclo de retención. Esta instrucción nop se propaga por todas las restantes etapas. Para visualizar esta instrucción nop en el diagrama temporal se utilizará el acrónimo nop, en lugar del acrónimo de etapa, una vez inyectada la instrucción nop.

Para representar el descarte de instrucciones posteriores a una instrucción de secuenciamiento se utilizará el acrónimo nop, en lugar del acrónimo de la etapa, una vez se ha inyectado la instrucción nop.

Un ciclo perdido se identificará en el ciclo en que finaliza una instrucción nop inyectada por el hardware. Estos ciclos se mostraran en el diagrama temporal en la parte inferior indicando el tipo. Para los riesgos de datos utilice la etiqueta D y para los riesgos de secuenciamiento la etiqueta S. Así mismo, en la columna etiquetada con la palabra instrucción indique la instrucción cuya interpretación se está representando en la fila correspondiente.

Pregunta 1: *Muestre en un diagrama temporal la interpretación de la secuencia de instrucciones previa. Tenga en cuenta cómo se ha descrito la forma de visualizar la interpretación de las instrucciones. Así mismo, indique los ciclos perdidos por riesgos de datos y de secuenciamiento. La última instrucción cuya interpretación debe mostrarse, incluyendo la escritura en el banco de registros, es load* ***R10****, X(R6).*

En el procesador descrito se añaden cortocircuitos desde las etapas ALU y M a la etapa D/L. En la siguiente figura se muestra un diagrama con su identificación.



Además, se añade la característica de utilizar predicción fija en las instrucciones de secuenciamiento condicional y el cálculo de la dirección destino del salto, en instrucciones de secuenciamiento condicional e incondicional, se efectúa en la etapa D/L. Para predecir el sentido se utiliza el signo del literal y esta predicción se utiliza en la etapa D/L para establecer el secuenciamiento predicho.

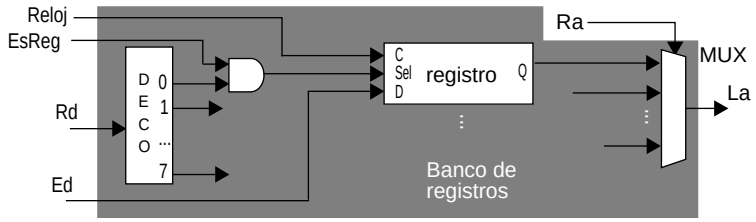
La comprobación de la predicción se efectúa en la etapa ALU (que etiquetamos CPre) y en caso de error de predicción, el registro CP se actualiza con la dirección correcta cuando la instrucción de secuenciamiento está en la etapa ES (que etiquetamos Rec).

ciclos	1	2	3	4	5	6
etapas	CP	B	D/L	CPre		Rec

Pregunta 2: *Muestre en un diagrama temporal simplificado, sin mostrar la inyección de instrucciones nop, la interpretación de TODA la secuencia de código del enunciado. Indique los ciclos perdidos por riesgos de datos y de secuenciamiento. Así mismo, en el ciclo que se utilice un cortocircuito indique el cortocircuito utilizado mediante el número que tiene en la figura previa. Considere que la codificación de las instrucciones es la siguiente.*

load rd, #literal (rb)	add/sub rd, ra, rb	br ra, #literal

En una operación de escritura a un banco de registros se suministra un identificador de registro y un dato. El identificador de registro (Rd) se decodifica por el decodificador y la salida del decodificador activa la señal de escritura en un registro del banco de registros (Sel), condicionada al permiso de escritura (EsReg). Entonces, cuando llega el flanco ascendente de la señal de reloj se almacena el dato (Ed) en el banco de registros.

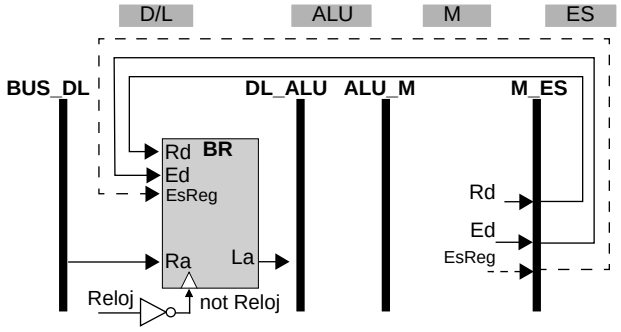


Para una operación de lectura, la salida de los registros se conecta a un multiplexor. Para leer un valor se utiliza el identificador de registro (Ra) que determina cuál de las entradas del multiplexor se selecciona (La). El retardo de los componentes del banco de registros se indica en la siguiente tabla.

COMPONENTE	RETARDO en u.t.
DECO	10
AND	1

COMPONENTE	RETARDO en u.t.
Escritura en el registro	6
MUX	10

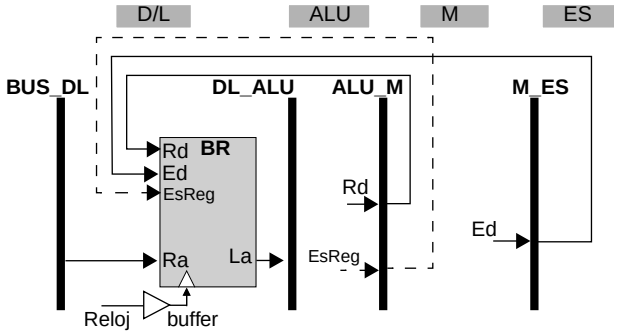
En el camino de datos de un procesador segmentado observamos las conexiones que se muestran en la siguiente figura. Las etiquetas Rd, Ed y EsReg indican respectivamente identificador de registro destino, dato y permiso de escritura. La etiqueta Ra es el identificador del registro que se quiere leer y “La” es la salida del multiplexor utilizado en operaciones de lectura en el banco de registros.



El retardo del inversor y de los registros de desacoplo es 1 u.t.

Pregunta 3: *En cualquier registro, queremos hacer posible la lectura del valor escrito en ese mismo ciclo. Muestre en un ciclo de la señal de reloj el retardo de los componentes (inversor, MUX, Retardo de escritura, DECO, AND, Registro de desacoplo) y de la señal not Reloj. Suponga que la señal de reloj es cuadrada y el periodo es 40 u.t.*

Con el objetivo de poder reducir el tiempo de ciclo, la actualización de un registro del banco de registros se efectúa utilizando el flanco ascendente de la señal de reloj y ello requiere modificar el conexionado de las señales utilizadas para efectuar una operación de escritura, lo cual se muestra en la siguiente figura. El retardo del buffer es 1 u.t.



Pregunta 4: *En cualquier registro, queremos hacer posible la lectura del valor escrito en ese mismo ciclo. Muestre en 2 ciclos consecutivos de la señal de reloj el retardo de los componentes (inversor, MUX, Retardo de escritura, DECO, AND, Registro de desacoplo) y de la señal en la salida del buffer. Suponga que la señal de reloj está 14 u.t. a nivel alto y 10 u.t. a nivel bajo.*

Ejercicio 4.10

Tenemos un procesador segmentado lineal con las siguientes etapas:

ciclos	1	2	3	4	5	6	7
etapas	CP	B	DL	M1	A	M2	E

y el siguiente lenguaje máquina:

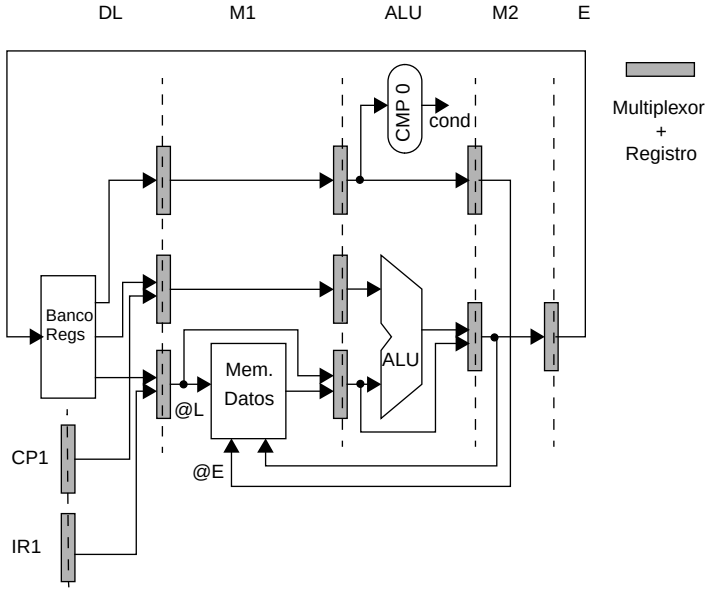
Descripción	Semántica
op Rd, Rf1, Rf2	Rd = Rf1 op Rf2
opm Rf1, Rf2, Rf3	M[Rf1] = M[Rf2] op Rf3
bnz Rf, #d	if (Rf <> 0) entonces CP = CP + #d

El procesador está diseñado garantizando que no se producen riesgos estructurales (la memoria de datos ha de permitir un acceso de lectura y un acceso de escritura en el mismo ciclo). El procesador no dispone de cortocircuitos. Cuando se produce un riesgo de datos, el procesador se bloquea hasta que desaparece el riesgo. En el caso de una instrucción de salto, el procesador se bloquea hasta que conoce el nuevo CP y la condición está evaluada.

La funcionalidad de las etapas y recursos básicos se describen en la siguiente tabla.

ETAPA	FUNCIONALIDAD
CP	determinar la dirección de la instrucción
B	búsqueda de la instrucción
DL	decodificación y lectura de los operandos en registros
M1	lectura de memoria para las instrucciones opm
A	ALU para las instrucciones op y opm. Cálculo del nuevo CP y evaluación de la condición para la instrucción bnz
M2	escritura en memoria para las instrucciones opm
E	escritura en el banco de registros (requiere todo el ciclo de reloj)

En la siguiente figura se muestran las 5 últimas etapas del camino de datos.



donde CP1 contiene la dirección de la instrucción que ocupa la etapa DL e IR1 contiene la instrucción buscada en el ciclo previo.

Pregunta 1: *Muestre en un cronograma la interpretación del siguiente bucle identificando los ciclos de bloqueo y el motivo o motivos. Calcule el número de ciclos para ejecutar todas las iteraciones del bucle.*

```
DO I= 1, N                1$:  M[R6] = M[R5] + R3     (opm)
  V(I)= X(I)+3                 M[R5] = M[R6] * R3     (opm)
  X(I)= V(I)*3                 R6 = R6 + R1           (op)
ENDDO                          R5 = R5 + R1           (op)
                               R4 = R4 - R1           (op)
                               bnz R4, 1$

                               Valores iniciales:
                               r6 = dirección del primer elemento del vector V
                               r5 = dirección del primer elemento del vector X
                               r1 = 1, r4 = N; r3 = 03
```

El procesador se rediseña con cortocircuitos con el objetivo de reducir los ciclos perdidos.

Pregunta 2: *Muestre en un cronograma la interpretación del bucle previo identificando los cortocircuitos utilizados, los ciclos de bloqueo y el motivo o motivos. Así mismo, dibuje en el camino de datos los cortocircuitos utilizados. Calcule el número de ciclos para ejecutar todas las iteraciones del bucle.*

Ejercicio 4.11

Tenemos un procesador segmentado (PA), que se ha diseñado como el modelo lineal explicado en el capítulo anterior y en este capítulo, con 6 etapas y con frecuencia de reloj 200 MHz.

ciclos	1	2	3	4	5	6
etapas	CP	B	DL	A	M	ES

La funcionalidad de las etapas es la siguiente:

ETAPA	FUNCIONALIDAD
CP	determinar la dirección de la instrucción
B	búsqueda de la instrucción
DL	decodificación, comprobación de riesgos y lectura de los operandos en registros
A	cálculos
M	acceso a memoria de datos
ES	escritura en el banco de registros

Se propone otro diseño (PB), segmentado en 9 etapas porque la búsqueda, el cálculo y el acceso a memoria se han segmentado con dos etapas cada uno. Esto permite usar reloj de frecuencia 350 MHz.

ciclos	1	2	3	4	5	6	7	8	9
etapas	CP	B1	B2	DL	A1	A2	M1	M2	ES

Tanto PA como PB bloquean la decodificación cada vez que se poduce un riesgo.

Para comparar prestaciones de ambos diseños, definimos Ganancia = TA / TB, siendo TA y TB el tiempo empleado en ejecutar un mismo código, respectivamente en PA y PB.

Pregunta 1: *Deduzca el valor de la latencia mínima de una instrucción, es decir, el tiempo mínimo desde inicio hasta el final de la instrucción, en cada uno de los dos diseños.*

Pregunta 2: *Deduzca el valor ideal (máximo teórico) de la Ganancia que se obtendría ejecutando un código con cuatro instrucciones independientes entre sí y sin rupturas de secuencia.*

Consideramos ahora el siguiente código:

```
F1 ← M[R1+d1]
F2 ← M[R3+d2]
F3 ← F1 • F2
M[R5+d3] ← F3
```

Pregunta 3: *Calcule el valor de la Ganancia, si no hay cortocircuitos ni en PA ni en PB, y la escritura de registro ocupa todo el ciclo en PA y en PB. Presente los cronogramas que justifican la respuesta.*

Pregunta 4: *Calcule el valor de la Ganancia, si no hay cortocircuitos ni en PA ni en PB, y la escritura de registro ocupa el primer semiciclo en PA, pero sigue ocupando todo el ciclo en PB. Presente el cronograma que justifica la respuesta.*

Pregunta 5: *Calcule el valor de la Ganancia, si hay cortocircuitos para cargar los registros que separan la etapa DL de la siguiente tanto en PA com en PB. Presente los cronogramas que justifican la respuesta.*

Ejercicio 4.12

Sea un procesador segmentado con las siguientes etapas:

ciclos	1	2	3	4	5	6	7
etapas	CP	B	DL	A1	M	A2	ES

Todas las instrucciones pasan por todas las etapas (un ciclo en cada etapa), aunque algunas instrucciones en algunas etapas no hagan nada. Todas las etapas se han implementado con hardware separado.

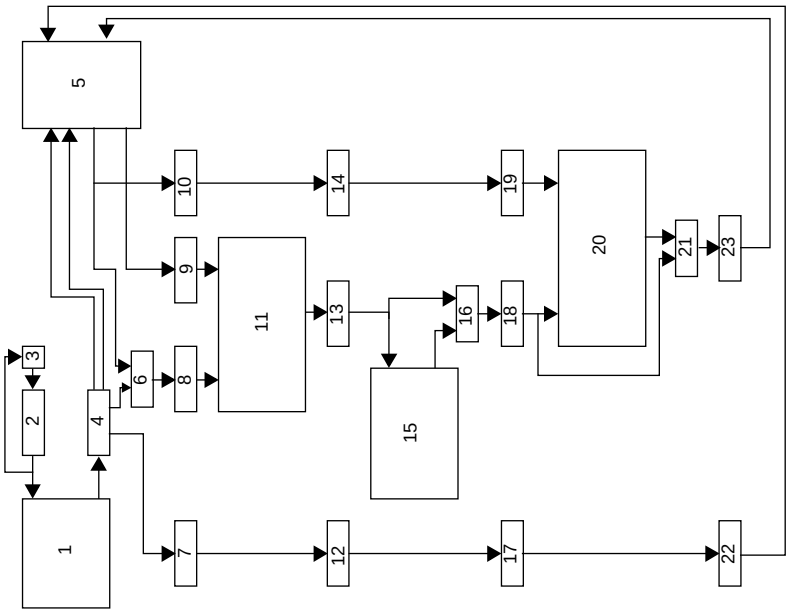
A efectos del problema nos interesan los siguientes tipos de instrucciones:

Tipo	Descripción
OPR:	Rd <---- Rf1 op Rf2
OPM:	Rd <---- M[Rf1+X] op Rf2
LOAD:	Rd <---- M[Rf1+X]

En las etapas se efectuan las siguientes operaciones:

ETAPA	FUNCIONALIDAD
CP	determinar la dirección de la instrucción
B	búsqueda de la instrucción
DL	decodificación y lectura de los operandos en registros En caso de riesgo por dependencias se bloquea la instrucción en esta etapa, hasta que se resuelve el riesgo. Ninguna otra instrucción posterior pasará a ejecutarse hasta que la instrucción bloqueada salga de la etapa DL (o sea, las instrucciones se ejecutan en orden).
A1	se efectúa la operación en las instrucciones OPR y se calcula la dirección de memoria [Rf1+X] en las instrucciones OPM y LOAD (X es un campo que está en la instrucción)
M	se accede a memoria en las instrucciones OPM y LOAD
A2	se efectúa la operación en las instrucciones OPM
ES	se escribe el resultado en el registro destino (al final del ciclo)

Pregunta 1: *Identifique los 23 elementos [banco de registros (BR), unidades aritméticas (ALUs), memorias (MEMs), multiplexores (MPXs), CP, RI y registros entre etapas (REGs)] de la figura que representa el camino de datos para poder ejecutar estos 3 tipos de instrucciones. Observe que no hay ningún cortocircuito.*



Rellene con números del 1 al 23:

BR	ALUs	MEMs	MPXs	CP	RI	REGs

En las preguntas siguientes se pide que indique la ejecución de un código mediante cronogramas. Se deben indicar también (si es el caso) los ciclos de bloqueo y los cortocircuitos entre las etapas que se usan en cada ciclo (con flechas etiquetadas). Cada etiqueta (a, b, c ...) se corresponde con un cortocircuito hardware, que deberá dibujar y etiquetar sobre el camino de datos de la pregunta anterior.

Pregunta 2: *Presente en un cronograma la ejecución del siguiente fragmento de programa, suponiendo que no existen cortocircuitos. Indique el número total de ciclos.*

```
CODIGO1
LOAD:   R1 <---- M[R2 + X]
OPR:    R2 <---- R1 op R2
LOAD:   R3 <---- M[R2 + X]
OPM:    R1 <---- M[R3 + X] op R2
OPM:    R1 <---- M[R2 + X] op R1
```

Pregunta 3: *Presente en un cronograma la ejecución del código 1, usando todos los cortocircuitos que sean necesarios para minimizar el tiempo de ejecución. Dibuje en el camino de datos los cortocircuitos. Indique el número total de ciclos.*

Pregunta 4: *¿Se puede reducir el tiempo de ejecución del código 1 respecto al de la pregunta 3) con alguna técnica software?. ¿Con qué cortocircuitos, en caso de poderse reducir el tiempo?. Razone la respuesta.*

Pregunta 5: *Repita la pregunta 3 si la última instrucción del código 1 se cambia por: OPM:R1 <---- M[R1+X] op R2*
Señale qué cortocircuitos sirven. Indique el número total de ciclos

Ejercicio 4.13

Sea un procesador segmentado con las siguientes etapas:

ciclos	1	2	3	4	5	6	7
etapas	CP	B	DL	A1	M	A2	ES

Todas las instrucciones pasan por todas las etapas (un ciclo en cada etapa), aunque algunas instrucciones en algunas etapas no hagan nada. Todas las etapas se han implementado con hardware separado.

A efectos del problema nos interesan los siguientes tipos de instrucciones:

Tipo	Descripción
OPR:	Rd <---- Rf1 op Rf2
OPM:	Rd <---- M[Rf1+X] op Rf2
LOAD:	Rd <---- M[Rf1+X]

En las etapas se efectuan las siguientes operaciones:

ETAPA	FUNCIONALIDAD
CP	determinar la dirección de la instrucción
B	búsqueda de la instrucción
DL	decodificación y lectura de los operandos en registros En caso de riesgo por dependencias se bloquea la instrucción en esta etapa, hasta que se resuelve el riesgo. Ninguna otra instrucción posterior pasará a ejecutarse hasta que la instrucción bloqueada salga de la etapa DL (o sea, las instrucciones se ejecutan en orden).
A1	se efectúa la operación en las instrucciones OPR y se calcula la dirección de memoria [Rf1+X] en las instrucciones OPM y LOAD (X es un campo que está en la instrucción)
M	se accede a memoria en las instrucciones OPM y LOAD
A2	se efectúa la operación en las instrucciones OPM
ES	se escribe el resultado en el registro destino (al final del ciclo)

Pregunta 1: *Dibuje el camino de datos (banco de registros, buses, ALUs, memorias, multiplexores, CPs, IRs, registros separadores de etapas, y todas sus interconexiones) para ejecutar los tres tipos de instrucciones. Suponga, en esta pregunta, que no existen cortocircuitos.*

En las preguntas siguientes se pide que indique la ejecución de fragmentos de programas mediante cronogramas. Se deben indicar tambien (si es el caso) los ciclos de bloqueo y los cortocircuitos entre las etapas que se usan en cada ciclo (con flechas etiquetadas). Cada etiqueta (a, b, c,...) se corresponde con un cortocircuito hardware, que deberá dibujar y etiquetar sobre el camino de datos de la pregunta anterior.

Pregunta 2: *Presente en un cronograma la ejecución del siguiente fragmento de programa suponiendo que no existen cortocircuitos.*

```
CODIGO1
OPR:    R2 <---- R1 op R2
LOAD:   R3 <---- M[R1 + X]
OPM:    R1 <---- M[R3 + X] op R2
```

Pregunta 3: *Presente en un cronograma la ejecución del código 1 si se dispone de los cortocircuitos que sean necesarios para reducir el tiempo de ejecución.*

Pregunta 4: *¿Se puede reducir el tiempo de ejecución del código 1, respecto al de la pregunta 3), con alguna técnica software?. ¿Son necesarios cortocircuitos?. Presente los resultados mediante un cronograma.*

Ejercicio 4.14

Sea un procesador (PR1) segmentado lineal de 6 etapas.

ciclos	1	2	3	4	5	6
etapas	CP	B	DL	A	M	ES

Este procesador interpreta el código (C1) que es el bucle de 16 instrucciones siguiente:

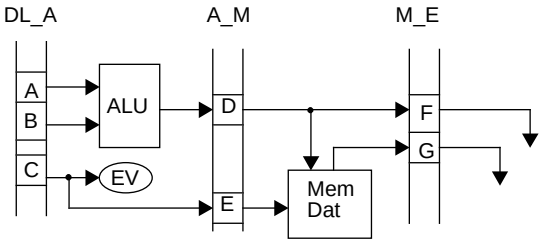
1.-	r2 := r1 - r2	9.-	nop
2.-	r3 := r3 + r1	10.-	r9 := r5 - 1
3.-	nop	11.-	r1 := r1 -2
4.-	nop	12.-	M[r3+0] = r4
5.-	r4 := r2 + r4	13.-	nop
6.-	r5 := r6 + r7	14.-	si (r9) saltar a 1.-
7.-	nop	15.-	nop
8.-	nop	16.-	nop

El PR1 no tiene lógica para bloquear y tampoco tiene ningún cortocircuito, y sabemos que el resultado de la ejecución de C1 sería incorrecto si se elimina o cambia de posición cualquiera de sus 16 instrucciones.

Pregunta 1: *Deduzca si puede o no puede escribir y leer un registro en el mismo ciclo.*
Deduzca la duración de cada riesgo de secuenciamiento (RdS) y en función de la distancia, la de cada riesgo de datos (RdD). Calcule cuántos ciclos tarda cada iteración del bucle. Calcule el CPI.

Pregunta 2: *Deduzca cuál es el cortocircuito que debemos añadir al PR 1 para que interprete correctamente un código (C2) con 13 instrucciones, que es el que resulta al retirar del C1 las tres instrucciones numeradas como 7.-, 8.- y 9.-. Dibújelo sobre la figura que se muestra después de la pregunta 3.*

Pregunta 3: *Deduzca cuál es el cortocircuito que debemos añadir al PR1 para que interprete correctamente un código (C3) con 15 instrucciones, que es el que resulta al retirar del C1 la instrucción numerada como 13.-. Dibújelo sobre la figura que se muestra seguidamente.*



Añadimos al PR1 lógica para bloquear y también los dos cortocircuitos deducidos en los dos apartados anteriores, y llamamos PR2 a este nuevo procesador, que tiene frecuencia de reloj de 2 Ghz.

PR2 ejecuta el código (C4) que es el bucle de 8 instrucciones que resulta al retirar de C1 todas las instrucciones nop.

Pregunta 4: *Presente el grafo de dependencias de C4. Presente el cronograma de ejecución de una iteración completa.*

Pregunta 5: *Calcule cuántos ciclos se bloquea por RdS y cuántos por RdD. Calcule el CPI.*

Pregunta 6: *Deduzca cuáles son las dos instrucciones de C4 cuyo orden debe intercambiarse para conseguir que los RdD no provoquen ningún ciclo de bloqueo. Resulta el código C5.*

Añadimos al PR2 lógica para predicción fija de saltos “salta siempre” y un sumador para calcular el nuevo CP en la tercera etapa, y llamamos PR3 a este nuevo procesador.

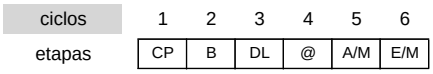
Pregunta 7: *Calcule cuántos ciclos se bloquea PR3 por RdS y cuántos por RdD cuando ejecuta C4. Calcule el CPI.*

Pregunta 8: *Calcule el tiempo de ciclo del PR3, sabiendo que para ejecutar C5 tarda el mismo tiempo que tarda el PR2 para ejecuta C4.*

Pregunta 9: *Calcule cuántas iteraciones del bucle C5 debe ejecutar el PR3 para tardar 3×10^{-5} s.*

Ejercicio 4.15

Sea el siguiente procesador segmentado lineal en 6 etapas:



donde

ETAPA	FUNCIONALIDAD
CP	determinar la dirección de la instrucción
B	búsqueda de la instrucción
DL	decodificación, comprobación de riesgos y lectura de los operandos en registros (segundo semiciclo)
@	cálculo de la dirección de memoria, o dirección destino de dalto y evaluación de la condición
A/M	operaciones aritméticas y lógicas, o acceso a memoria
E/M	escritura del resultado en registro (durante el primer semiciclo) o escritura en memoria

La memoria de datos no está segmentada y se mantiene ocupada durante 2 ciclos para las instrucciones `Store` y durante 1 ciclo para las instrucciones `Load`.

Cuando se se detecta un riesgo el procesador bloquea la búsqueda y decodificación de instrucciones hasta que desaparece el riesgo.

Cuando se interpreta una instrucción de secuenciamiento el procesador se bloquea hasta que se resuelve el salto. El cálculo de la dirección destino y la evaluación de la condición se efectúan en la etapa @ y al final del ciclo se actualiza el contador de programa.

Pregunta 1: *Indique las situaciones en que el procesador se bloquea debido a: 1) riesgos estructurales debidos a la memoria de datos, 2) riesgos de dependencias de datos y 3) riesgos de secuenciamiento. Ilustre con un ejemplo de código cada situación junto con un cronograma de ejecución.*

Pregunta 2: *Para cada una de las situaciones de la pregunta 2, especifique el cortocircuito que resuelve el riesgo por dependencias de datos. Para cada cortocircuito, especifique las etapas fuente y destino y la reducción de ciclos de bloqueo que se consigue.*

Queremos evaluar el rendimiento del procesador con diferentes mecanismos. Para ello utilizaremos el siguiente fragmento de código.

```
for ( i = 0 ; i < N ; i++) {
   b(i) = a (i);
   c(i) = a(i);
}
```

```
1$:  R4 ← Mem[R0 + 0]
     Mem[R1 + 0] ← R4
     Mem[R2 + 0] ← R4
     R0 ← R0 + 4
     R1 ← R1 + 4
     R2 ← R2 + 4
     R3 ← R3 - 1
     if R3 ≠ 0 then goto 1$
```

Para las siguientes cuatro preguntas supondremos que el procesador no incluye cortocircuitos.

Pregunta 3: *Calcule el tiempo de ejecución (en ciclos) del bucle. Justifique la respuesta con un cronograma de ejecución de las dos primeras iteraciones.*

Pregunta 4: *Reordene el código original para minimizar el tiempo de ejecución. Muestre el código resultante, el tiempo de ejecución y un cronograma de las dos primeras iteraciones.*

Suponga que el procesador no dispone de un mecanismo para actuar en el caso de riesgos de secuenciamiento y sigue interpretando instrucciones en secuencia hasta que se resuelve el salto. Notemos que estas instrucciones se interpretan siempre, independientemente de que se siga en secuencia o se modifique el secuenciamiento. Este tipo de actuación se denomina salto retardado.

Pregunta 5: *¿ Cuál es el número de instrucciones que se interpretan después de la instrucción de secuenciamiento y antes de que se empiece a interpretar la instrucción que determina la instrucción de secuenciamiento?.*

Pregunta 6: *Reordene el código resultante de la pregunta 4 para minimizar, si es posible, el tiempo de ejecución. Indique el nuevo código y calcule el tiempo de ejecución del bucle.*

Ejercicio 4.16

Sea un procesador segmentado lineal de 7 etapas.

ciclos	1	2	3	4	5	6	7
etapas	CP	B	D	L	A	M	E

donde

ETAPA	FUNCIONALIDAD
CP	determinar la dirección de la instrucción
B	búsqueda de la instrucción
D	decodificación, detección de riesgos y cálculo de la dirección destino de salto
L	lectura de los operandos en registros o de los cortocircuitos
A	operaciones aritméticas, cálculo de la dirección de memoria o evaluación de la condición en instrucciones de secuenciamiento
M	acceso a memoria
E	escritura del resultado en registro (ocupa todo el ciclo)

El procesador dispone de los cortocircuitos A/A, M/A y E/A. Cuando se produce un riesgo de datos el procesador bloquea la búsqueda y decodificación de nuevas instrucciones.

Sea el siguiente programa.

```
do I=1, N                    1$:  Load r1, 0(r0)
  if (X[i] ≠ 0) then              Beqz r1, 2$
    X[i] = X[i] + Y[i]            Load r3, 0(r2)
  endif                           Add  r1, r1, r3
ENDDO                             Store r1, 0(r0)
                             2$:  Add  r0, r0, #4
                                  Add  r2, r2, #4
                                  Sub  r5, r5, #1
                                  bnez r5, 1$

                                  Valores iniciales:
                                  r0 = dirección del primer elemento del vector X
                                  r2 = dirección del primer elemento del vector Y
                                  r5 = N
```

El 80% de los elementos del vector X son distintos de cero.

Se pide para las siguientes 3 preguntas, el número medio de ciclos por iteración. Justifique las respuestas mediante cronogramas, indicando cuándo se utilizan los 3 tipos de cortocircuitos (A/A, M/A y E/A).

Pregunta 1: *Las instrucciones de secuenciamiento provocan el bloqueo del procesador hasta que se ha evaluado la condición.*

Pregunta 2: *Predicción fija en función de si el salto es hacia adelante o hacia atrás. Si el salto condicional es hacia adelante, la predicción es no saltar; si el salto es hacia atrás, la predicción es saltar. En ambos casos, si el resultado del salto es contrario al predicho, el procesador descarta las instrucciones que se han empezado a interpretar de la rama incorrecta.*

Pregunta 3: *El mecanismo de ejecución de saltos condicionales es el mismo que en la 2ª pregunta. Se añade al repertorio de instrucciones del lenguaje máquina la instrucción Store Condicional.*

Instrucción	Descripción
Storenez rk, #X(ri) ,rj	Mem(ri+X) ← rk, si rj $\neq 0$ en otro caso no se escribe en memoria

Reescriba el código correspondiente, intentando reducir al mínimo los ciclos por iteración.

Ejercicio 4.17

Sea un procesador segmentado lineal de 6 etapas.

ciclos	1	2	3	4	5	6
etapas	CP	B	L	A	M	E

donde

ETAPA	FUNCIONALIDAD
CP	determinar la dirección de la instrucción
B	búsqueda de la instrucción
L	decodificación, comprobación de riesgos y lectura de operandos en registros (segundo semiciclo)
A	operaciones aritméticas, cálculo de la dirección de memoria o dirección destino y evaluación de la condición en instrucciones de secuenciamiento
M	acceso a memoria
E	escritura del resultado en registro (durante el primer semiciclo)

El procesador dispone de cortocircuitos A/A y M/A.

Para evaluar la eficiencia de varios mecanismos de interpretación de instrucciones de secuenciamiento condicional, se utilizará el siguiente fragmento de código.

```
j=0;                                1$:   Load r2,A(r0)
for ( i = 0 ; i < N ; i++) {              Beqz r2, 2$
   if (A[i] ≠ 0) then {                   Store r2, B(r1)
   B[j] = A[i];                           Add  r1, r1, 1
   j++;                             2$    Add r0, r0, 1
}                                         Sub r4, r3, r0
                                          Bnez r4, 1$
```

Se conoce que el 40% de los elementos del vector almacenan el valor cero.

Se solicita para las siguientes cuatro preguntas calcular el número medio de ciclos por iteración. Justifique las respuestas mediante un cronograma, indicando cuando se utilizan los 2 tipos de cortocircuitos (A/A y M/A).

Pregunta 1: *Las instrucciones de secuenciamiento condicional provocan el bloqueo del procesador durante 2 ciclos (latencia de una instrucción de salto 3 ciclos).*

Pregunta 2: *Predicción estática fija No Saltar, es decir, después de un instrucción de secuenciamiento condicional, el procesador sigue interpretando instrucciones en secuencia; en caso de que se modifique el secuenciamiento implícito descarta las instrucciones del camino secuencial.*

Suponga que el procesador no dispone de un mecanismo para actuar en el caso de riesgos de secuenciamiento y sigue interpretando instrucciones en secuencia hasta que se resuelve el salto. Notemos que estas instrucciones se interpretan siempre, independientemente de que se siga en secuencia o se modifique el secuenciamiento. Este tipo de actuación se denomina salto retardado.

Pregunta 3: *Las instrucciones de secuenciamiento condicional son del tipo salto retardado. En este caso, el compilador intenta reordenar el código para optimizar el rendimiento. Muestre cuál seria el código resultante después de aplicar la reordenación.*

Pregunta 4: *El mecanismo de interpretación de instrucciones de secuenciamiento condicional es el mismo que en la 1ª pregunta. Ahora bien, se extiende el repertorio de instrucciones de lenguaje*

máquina con una instrucción de Movimiento Condicional.

Instrucción	Descripción
Movne rk, ri, rj	rk ← ri, si rj ≠ 0 en otro caso, rk no se modifica

El código que genera el compilador cuando se dispone de esta instrucción es el siguiente.

```
j=0;                          1$:  Load r2,A(r0)
for ( i = 0 ; i < N ; i++) {       Add r0, r0, 1
   if (A[i] ≠ 0) then {            Store r2, B(r1)
   B[j] = A[i];                    Add r5, r1, 1
   j++;                            Movne r1, r5, r2
}                                  Sub r4, r3, r0
                                   Bnez r4, 1$
```

Ejercicio 4.18

Consideremos un procesador con un lenguaje máquina como el utilizado en este capítulo, excepto las instrucciones de secuenciamiento condicional.

Instrucción	Descripción
bnez Rf1,Rf2	si (Rf1<>0) entonces CP ← Rf2

El procesador está segmentado en 6 etapas.

ciclos	1	2	3	4	5	6
etapas	CP	B	DL	EX1	EX2	ES

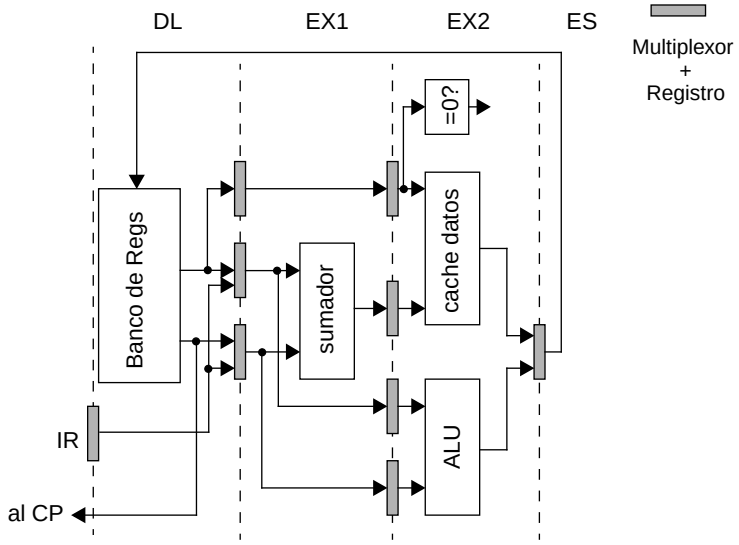
donde

ETAPA	FUNCIONALIDAD
CP	determinar la dirección de la instrucción
B	búsqueda de la instrucción
DL	decodificación, comprobación de riesgos y lectura de operandos en registros (segundo semiciclo)
EX1	cálculo de la dirección de memoria para acceder a la memoria de datos
EX2	operaciones aritméticas y lógicas, acceso a memoria, evaluación de la condición
ES	escritura del resultado en registro (durante el primer semiciclo)

Para la intepretación de las instrucciones de secuenciamiento condicional se utiliza *Predicción estática fija Saltar*, es decir, cuando el procesador detecta una instrucción de secuenciamiento condicional, inicia la búsqueda de instrucciones a partir de la

dirección destino del salto. En caso de error en la predicción, el procesador descarta las instrucciones del camino destino e inicia la búsqueda de instrucciones por el camino secuencial.

El camino de datos se muestra seguidamente y no incluye las etapas B y CP ni los cortocircuitos.



Consideremos el siguiente programa de prueba.

```
1$:  Load r1, 0(r0)          ;r1 ← mem[r0+0]
     Load r2, 0(r1)          ;r2 ← mem[r1+0]
     Add  r2, r2, r4         ;r2 ← r2 + r4
     Store 0(r2), r3         ;mem[r2+0] ← r3
     Load r0, 8(r0)          ;r0 ← mem[r0+8]
     Bnez r0, r5      (1$)   ;si r0 <> 0 entonces cp ← r5

     Valores iniciales:
     r5 = 1$
     r4 = 1
```

Pregunta 1: *Determine los cortocircuitos con cualquier etapa como destino, que se utilizan en la ejecución del código de prueba, con el objetivo de reducir los ciclos perdidos por riesgos de datos. Añada estos cortocircuitos al camino de datos.*

Pregunta 2: *Calcule el número de ciclos por iteración. Justifique la respuesta con un cronograma de ejecución, indicando en cada ciclo la etapa que ocupa cada instrucción y los cortocircuitos utilizados.*

Pregunta 3: *Podemos reducir aún más los ciclos perdidos por riesgos de datos si observamos que en las instrucciones de acceso a memoria con desplazamiento igual a Cero no es necesaria la suma para determinar la dirección efectiva.*
Indique en el camino de datos los cortocircuitos que utilizarían las instrucciones Load o Store con desplazamiento Cero para no perder ciclos.
Calcule el número de ciclos por iteración del código anterior. Justifique la respuesta con un cronograma de ejecución.

Pregunta 4: *Queremos modificar el mecanismo de interpretación de las instrucciones de secuenciamiento utilizando predicción estática fija No Saltar. Determine la frecuencia máxima de saltos efectivos (que rompen el secuenciamiento implícito) que tendría que tener un programa para que la hipótesis No Saltar fuera mejor.*

Ejercicio 4.19

En la interpretación de una instrucción se pueden distinguir las siguientes suboperaciones:

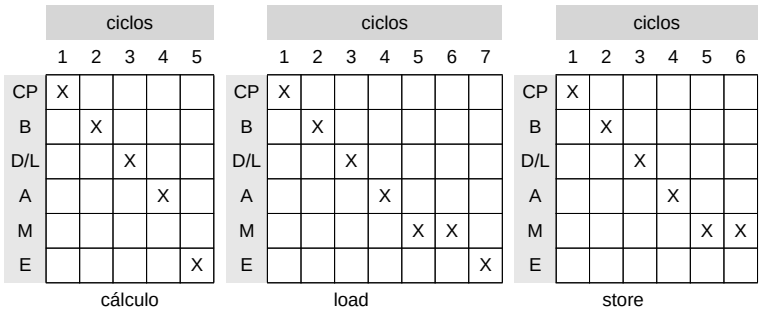
suboperación	retardo (ns)
búsqueda de la instrucción	50
decodificación y lectura de operandos	50
escritura de registros	50
acceso a memoria	100
unidad aritmético lógica	50

Partiendo de ellas se ha diseñado un procesador segmentado en 6 etapas

etapas	funcionalidad	ciclos
CP	determinación de la dirección de la instrucción	1
B	búsqueda de la instrucción	1
D/L	decodificación y lectura de operandos	1
E	escritura de registros	1
M	acceso a memoria	2
A	unidad aritmético lógica	1

La funcionalidad de las etapas es la misma que la del procesador lineal explicado en este capítulo.

En este procesador existen tres tipos diferentes de instrucciones en función de como se utilizan las etapas. En la figura se indican las tablas de reserva para los distintos tipos de instrucciones.



Nótese que las instrucciones tipo load y store utilizan durante dos ciclos consecutivos la etapa M. Esto es, la memoria tiene un tiempo de acceso de 2 ciclos y no está segmentada

La unidad de control detecta riesgos estructurales y riesgos de dependencias de datos y tiene capacidad de bloquear el procesador hasta que los riesgos desaparecen. Además, puede indicarle a la etapa D/L que efectúe la lectura de operandos tanto del banco de registros como de los cortocircuitos.

Hay dos buses para leer los registros y un bus para escribir los registros. La escritura se efectúa durante la primera mitad del ciclo y las lecturas se efectúan en la segunda mitad del ciclo.

Cualquier operación debe tener todos los datos que necesita al finalizar la etapa de D/L.

Pregunta 1: *Dibuje un esquema simplificado (registros, ALU, M, no son necesarias las etapas CP y B) del camino de datos de este procesador identificando claramente las etapas.*

Pregunta 2: *Indique en qué situaciones se debe bloquear el procesador y durante cuántos ciclos para eliminar los riesgos estructurales.*

Pregunta 3: *Indique en qué situaciones, en las que aparecen riesgos debidos a dependencias de datos, se debe bloquear el procesador. Determine el número de ciclos de bloqueo.*

Pregunta 4: *Indique el número mínimo de cortocircuitos necesarios para hacer desaparecer los riesgos debidos a dependencias de datos cuando sea posible. Justifique la respuesta indicando qué tipos de combinaciones de instrucciones los utilizan.*

Pregunta 5: *Dibuje sobre el esquema de la pregunta 1 de forma clara los cortocircuitos.*

Pregunta 6: *Modifique la ocupación de las etapas de los tres tipos de instrucciones para conseguir: a) una latencia de iniciación de instrucciones mínima y constante y b) que no se produzcan riesgos estructurales.*
Indique el valor de la latencia. Dibuje las tablas de reserva que son distintas al enunciado indicando qué se hace en cada etapa e indique qué riesgos estructurales y riesgos debidos a dependencias de datos se han eliminado.

Ejercicio 4.20

Suponga un procesador con las siguientes instrucciones de lenguaje máquina.

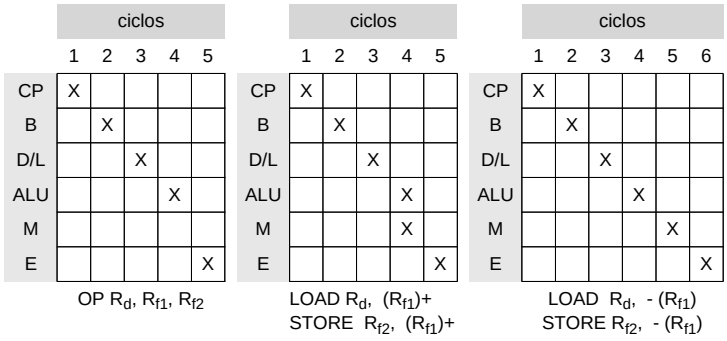
Tipo	Descripción
Cálculo	OP R_d, R_{f1}, R_{f2}
acceso a memoria	LOAD R_d, $(R_{f1})+$ LOAD R_d, $-(R_{f1})$ STORE R_{f2}, $(R_{f1})+$ STORE R_{f2}, $-(R_{f1})$

La funcionalidad de los modos de direccionamiento $(R_f)+$ y $-(R_f)$ es la siguiente.

Modo de direccionamiento	Descripción	Acciones sucesivas
$(R_{f1})+$	postincremento	M[R_{f1}] $R_{f1} = R_{f1} + 1$
$-(R_{f1})$	predecremento	$R_{f1} = R_{f1} - 1$ M[R_{f1}]

En las instrucciones LOAD, R_d siempre es un registro diferente a R_{f1}.

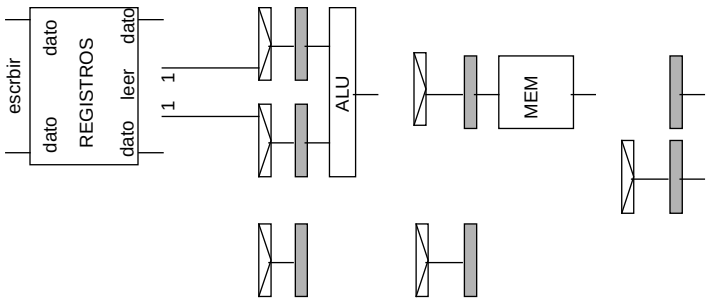
El procesador se segmenta en etapas cuya ocupación se describe en las siguientes tablas de reserva.



La memoria de instrucciones y la memoria de datos tienen caminos de acceso separados. Hay dos buses para leer de los registros y dos buses para escribir en los registros. La escritura se efectúa durante la primera mitad del ciclo y las lecturas se efectúan en la segunda mitad del ciclo. Cualquier operación debe tener todos los datos que necesita al finalizar la etapa de D/L.

Los riesgos se detectan en la etapa de decodificación. La instrucción se bloquea al detectar un riesgo de cualquier tipo en la etapa de decodificación y permanece en ella hasta que desaparece el riesgo.

Pregunta 1: *Dibuje en el esquema simplificado que se adjunta (que no incluye las etapas CP y B) el flujo de datos entre etapas para cada tipo de instrucción. En un último diagrama dibuje el camino de datos que sirve para cualquier instrucción. El valor constante 1 está cableado.*



Pregunta 2: *Indique en qué situaciones se debe bloquear el procesador y durante cuántos ciclos para eliminar los riesgos estructurales (falta de recursos).*

En un programa se han medido las siguientes proporciones de secuencias de 2 instrucciones.

		instrución i		
		- (R_{f1})	(R_{f1})+	OP
instrución i + 1	- (R_{f1})	0.36 %	1.44 %	4.2 %
	(R_{f1})+	1.44 %	5.76 %	16.8 %
	OP	4.2 %	16.8 %	49 %

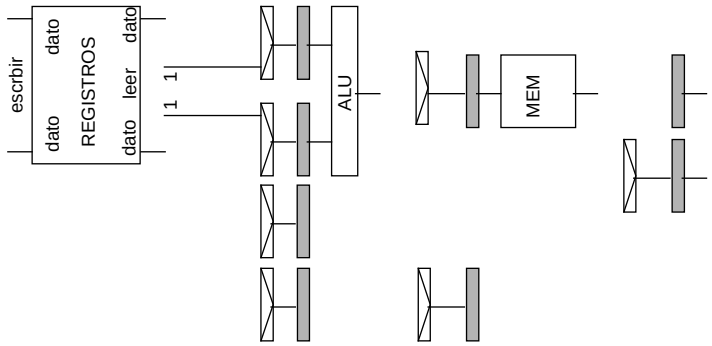
En la tabla - (R_{f1}) y (R_{f1})+ indican los modos de direccionamiento predecremento y postincremento respectivamente. Se conoce además que el 80% de las instrucciones de acceso a memoria son LOAD.

Pregunta 3: *Calcule el CPI suponiendo que sólo existen riesgos estructurales.*

Pregunta 4: *Indique el número mínimo de cortocircuitos necesarios para hacer desaparecer los riesgos debidos a dependencias de datos cuando sea posible. Justifique la respuesta indicando qué tipos de combinaciones de instrucciones los utilizan. Dibuje SOLO los cortocircuitos en el esquema simplificado del camino de datos que se ha utilizado en la 1ª pregunta, etiquetándolos para su identificación.*
NOTA: de una etapa puede salir más de un cortocircuito.

Pregunta 5: *Indique los bloqueos por riesgos de datos suponiendo que existen los cortocircuitos del apartado anterior.*

Pregunta 6: *Modifique la ocupación de las etapas de los tres tipos de instrucciones para conseguir. a) una latencia de iniciación de instrucciones mínima y constante y b) que no se produzcan riesgos estructurales.*
Indique el valor de la latencia. Dibuje las tablas de reserva que sean distintas al enunciado. Dibuje en el esquema simplificado que se adjunta el flujo de datos entre etapas.



Ejercicio 4.21

El lenguaje máquina de un procesador tiene instrucciones para leer o escribir datos en memoria (load, store), instrucciones para efectuar cálculos aritmético-lógicos (INT) e instrucciones que permiten modificar el secuenciamiento implícito (BRI, BRxx, JMP).

En la siguiente figura se muestra la especificación semántica de los tipos de instrucciones.

Las instrucciones de tipo INT efectuan cálculos aritméticos o lógicos y sus datos fuente están en registros y almacenan el resultado en un registro. Las instrucciones Load y Store utilizan el contenido de dos registros para calcular la dirección efectiva.

Tipo	Descripción	Operandos			Semántica
INT	operación aritmético-lógica	ra	rb	rc	$rc^v = ra^v$ Op rb^v
Load	Lectura de memoria	ra	rb	rc	ra^v = Mem [rb^v + rc^v]
Store	Escritura en memoria	ra	rb	rc	Mem [rb^v + rc^v] = ra^v
JMP	indexado		rb		$CP^v = rb^v$
BRI	incondicional	ra	literal		$CP^v = CP^v$ + 4 x ExtSig (literal)
BRxx	condicional	ra	literal		if (Ra^v != xx) then $CP^v = CP^v$ + 4 else $CP^v = CP^v$ + 4 x ExtSig (literal)

El superíndice v indica contenido del registro

Las instrucciones de tipo BRI y BRxx calculan la dirección destino de forma relativa a la dirección de la instrucción de secuenciamiento (CP). Esto es, a la dirección de la instrucción de secuenciamiento se le suma el campo literal especificado en la instrucción. Las instrucciones de secuenciamiento condicional

(BRxx, donde xx especifica la condición) evalúan además el contenido de un registro para seleccionar entre seguir en secuencia o modificar el secuenciamiento implícito (utilizar la dirección destino). Las instrucciones de tipo JMP utilizan el contenido de un registro como dirección de la siguiente instrucción que debe interpretarse.

En la siguiente figura se muestra la segmentación en etapas del proceso de interpretación para los distintos tipos de instrucciones.

	ciclos					
instrucción	1	2	3	4	5	6
load	CP	BUS	D/L	ALU	M	ES
store	CP	BUS	D/L	ALU	M	
INT	CP	BUS	D/L	ALU	ES	
BRI, BRxx, JMP	CP	BUS	D/L	ALU	M	ES

En la etapa CP se determina la dirección de la instrucción que se busca en memoria en el siguiente ciclo (etapa BUS) y se dispone de un sumador para efectuar el secuenciamiento implícito.

En la etapa D/L se decodifica la instrucción y se leen los registros fuente.

En la etapa ALU las instrucciones de tipo INT, load y store efectúan las siguientes acciones:

- Se efectúa el cálculo especificado en la instrucción: INT
- Se calcula una dirección efectiva: Load/Store

En la etapa M se accede a memoria (load, store).

En la etapa ES se actualiza el banco de registros (load, INT) y el CP (BRI, BRxx y JMP).
Una acción de lectura o escritura al banco de registros requiere todo el ciclo.

Pregunta 1: *¿Cuántos caminos de acceso, de lectura y de escritura, son necesarios en el banco de registros para que no se produzcan riesgos estructurales?. Justifique la respuesta*

En la etapa D/L se detectan los riesgos de datos y de secuenciamiento y se actúa para respetar la semántica del lenguaje máquina. Cuando se produce un riesgo de datos se bloquea la interpretación de la instrucción que está en la etapa D/L y de las

instrucciones que están en las etapas previas. Durante los ciclos de bloqueo se inyectan, mediante circuitería, instrucciones NOP desde la etapa D/L hacia la etapa ALU.

En el caso de detectar un riesgo de secuenciamiento se bloquea la interpretación de instrucciones posteriores a la instrucción de secuenciamiento. El bloqueo perdura hasta que se actualiza el registro CP con la dirección de la siguiente instrucción que debe interpretarse. La información en las etapas CP y BUS se descarta y durante los ciclos de bloqueo se inyectan, mediante circuitería, instrucciones NOP desde la etapa BUS hacia la etapa D/L.

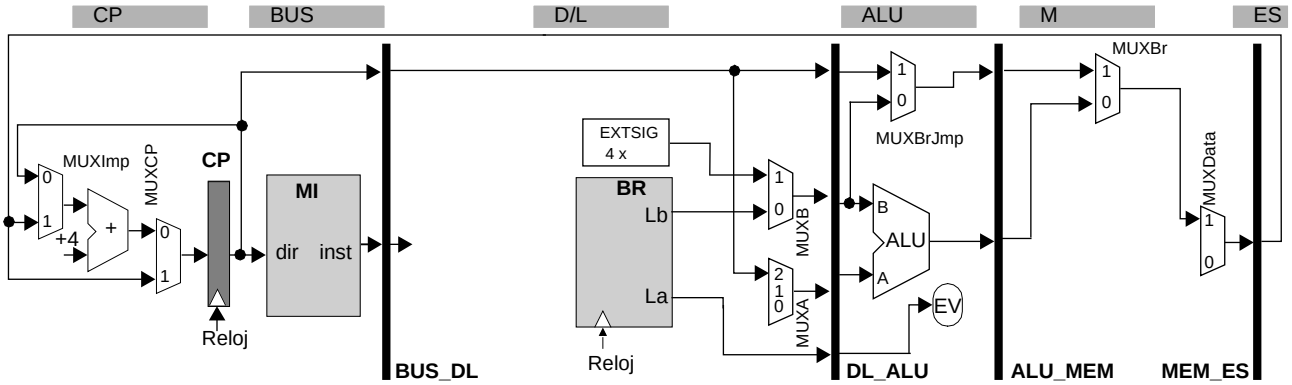
Pregunta 2: *¿Cuáles son las penalizaciones en ciclos, ciclos perdidos, cuando se detecta un riesgo de datos del tipo lectura después de escritura entre dos instrucciones consecutivas (*$R(i) \cap D(i+1) \neq 0$*)?. Para justificarlo, dibuje un diagrama temporal para cada una de las siguientes secuencias de instrucciones.*

1ª secuencia
r1 <- r2 + r3
r4 <- r1 + r4
load r9, r10, r11
r14 <- r21 + r31

2ª secuencia
load r9, r10, r11
r4 <- r9 + r4
load r3, r14, r17
r14 <- r21 + r31

Pregunta 3: *En una secuencia de instrucciones sin instrucciones de secuenciamiento, ¿cuál es el número mínimo de instrucciones independientes que debe haber entre una instrucción que produce un dato (escribe) y una instrucción que consume el dato (lee) para que no se produzca una situación de riesgo de datos?. Justifique la respuesta.*

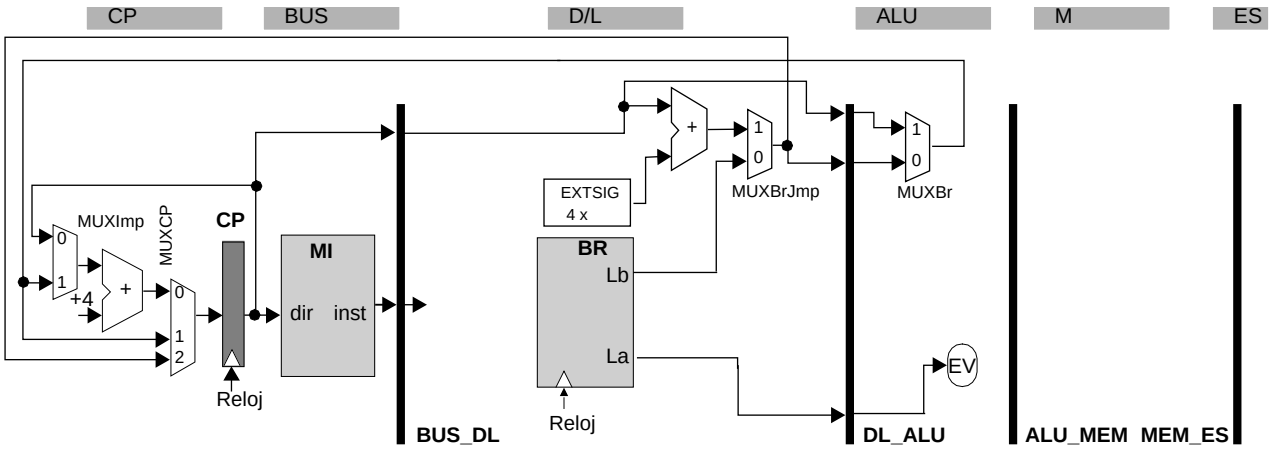
En la siguiente figura se muestra el camino de datos para las instrucciones de secuenciamiento. Los registros de desacoplo, excepto el registro CP, se muestran mediante una línea gruesa negra.



En la etapa ALU se calcula la dirección destino (BRI, BRxx), se evalúa la condición especificada (EV) y se selecciona (MUXBrJmp) entre el contenido de un registro leído en la etapa D/L (JMP) o la dirección de la instrucción. En la etapa M se selecciona (MUXBr) una dirección en función del tipo de instrucción (BRI, BRxx, JMP) y del resultado de evaluar la condición (BRxx). El registro de desacoplo CP se actualiza en la etapa ES.

Pregunta 4: *¿Cuál es la penalización en ciclos, ciclos perdidos, cuando se detecta un riesgo de secuenciamiento?. Justifíquelo mediante un diagrama temporal.*

Para reducir la latencia efectiva de actualización del registro CP se añade un sumador en la etapa D/L, que suma la dirección de la instrucción con el campo desplazamiento de la instrucción. El multiplexor MUXBrJmp se ubica en la etapa D/L y permite seleccionar entre el resultado de la suma y el contenido de un registro. La salida del multiplexor alimenta a la lógica de la etapa CP y se transfiere a la etapa ALU. En la etapa ALU se ubica el multiplexor MUXBr que originalmente estaba ubicado en la etapa M. Este multiplexor permite seleccionar entre la dirección de la instrucción y la salida del multiplexor MUXBrJmp. La salida del multiplexor MUXBr alimenta a la lógica de la etapa CP.



El diseño del camino de datos que se acaba de describir para las instrucciones de secuenciamiento, la inclusión en el mismo de todos los cortocircuitos necesarios para reducir la latencia efectiva de escritura en el banco de registros y la inclusión de los recursos necesarios para garantizar que no se producen riesgos estructurales lo denominaremos diseño modificado.

Pregunta 5: *¿Cuáles son las penalizaciones en ciclos, ciclos perdidos, cuando se detecta un riesgo de secuenciamiento en el diseño modificado?. Justifíquelo mediante diagramas temporales.*

Supongamos que en un programa se mide la siguiente proporción de instrucciones o de secuencias de dos instrucciones consecutivas con dependencias de datos.

instrucciones / secuencia de instrucciones	Proporción
BRI y JMP	4%
BRxx	17%
load **R1**, R2, R3 add **R4**, R1, R4	5%
load **R1**, R2, R3 store **R7**, R1, R4	3%

secuencia de instrucciones	Proporción
load **R1**, R2, R3 store **R1**, R4, R4	2%
add **R1**, R2, R3 add **R4**, R1, R4	10%
add **R1**, R2, R3 BRxx R1, X	10%
load **R1**, R2, R3 BRxx R1, X	6%

El registro destino se identifica en negrita. También se identifica en negrita, en una instrucción store, el registro cuyo contenido se almacena en memoria.
Debe entenderse que los identificadores de registro utilizados en las secuencias de instrucciones son exclusivamente para ilustrar la dependencia de datos.

Además, el compilador genera el código de forma que que no se detecten riesgos escritura después de escritura ($R(i) \cap R(i+k) \neq 0$).

Pregunta 6: *Cuando se interpreta el programa cuyas estadísticas se han descrito, ¿cuál es el número medio de ciclos por instrucción (CPI)?.*

Ejercicio 4.22

Para reducir el tiempo de ejecución de los programas que se ejecutan en un procesador se utiliza la técnica de segmentación en la implementación del camino de datos y se solapa la interpretación de instrucciones, con el objetivo de empezar a interpretar una instrucción en cada ciclo de reloj.

Segmentación y camino de datos

En la siguiente figura se muestra la segmentación en etapas del proceso de interpretación de las instrucciones.

ciclos	1	2	3	4	5	6
etapas	CP	B	DL	A	M	E

En la etapa CP se determina la dirección de la instrucción que se busca en memoria en el siguiente ciclo (etapa B) y se dispone de un sumador para efectuar el secuenciamiento implícito.

En la etapa DL se decodifica la instrucción y se leen los registros fuente.

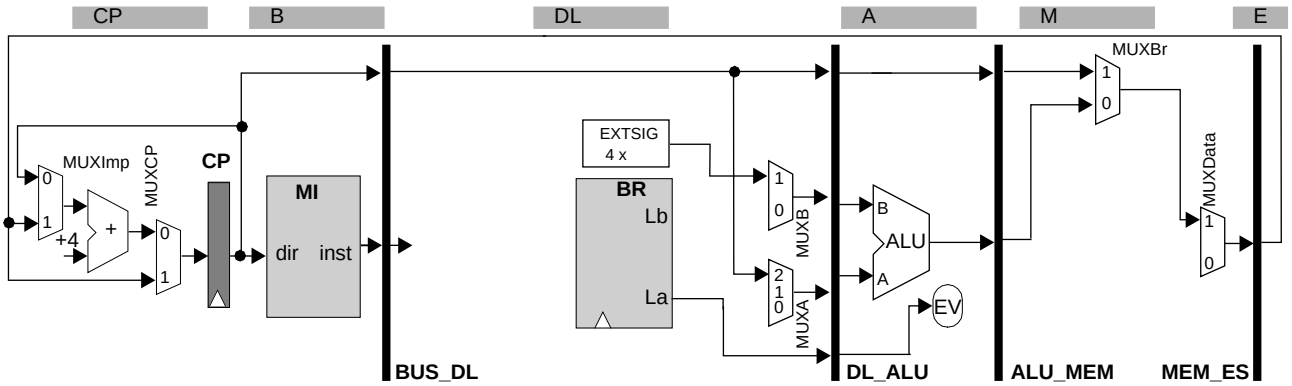
En la etapa A las instrucciones efectúan las siguientes acciones:

- Aritmético-lógicas: Se efectúa el cálculo especificado en la instrucción
- Load/Store: Se calcula la dirección efectiva.
- Secuenciamiento condicional: Se calcula la dirección destino y se evalúa el contenido de un registro.

En la etapa M las instrucciones efectúan las siguientes acciones:

- Aritmético-lógicas: No se efectúa ninguna acción.
- Load/Store: Se accede a memoria.
- Secuenciamiento condicional: En función del resultado de evaluar la condición especificada en la instrucción, se selecciona entre la

dirección de la instrucción de secuenciamiento y la dirección destino. En la siguiente figura se muestra el camino de datos para las instrucciones de secuenciamiento. Los registros de desacoplo, excepto el registro CP, se muestran mediante una línea gruesa negra.



En la etapa E las instrucciones efectúan las siguientes acciones:

- Aritmético-lógicas: Se actualiza el banco de registros.
- Load: Se actualiza el banco de registros.
- Store: No se efectúa ninguna acción.
- Secuenciamiento condicional: Se alimenta a la etapa CP con la dirección seleccionada en la etapa M. Al finalizar el ciclo se actualiza el registro CP con la dirección de la instrucción que debe buscarse en memoria.

Una acción de lectura o escritura al banco de registros requiere medio ciclo, efectuándose la escritura al principio del ciclo.

Riesgos y actuación

En la etapa DL se detectan los riesgos de datos y de secuenciamiento y se actúa para respetar la semántica del lenguaje máquina. Cuando se produce un riesgo de datos se bloquea la interpretación de la instrucción que está en la etapa DL y de las instrucciones que están en las etapas previas. Durante los ciclos de bloqueo se inyectan, mediante circuitería, instrucciones NOP desde la etapa DL hacia la etapa A.

Cuando se detecta un riesgo de secuenciamiento se bloquea la interpretación de instrucciones posteriores a la instrucción de secuenciamiento. El bloqueo perdura hasta que se actualiza el

registro CP con la dirección de la siguiente instrucción que debe interpretarse. La información en las etapas CP y B se descarta y durante los ciclos de bloqueo se inyectan, mediante circuitería, instrucciones NOP desde la etapa B hacia la etapa DL.

Código

En la siguiente figura se muestra el código en alto nivel y la traducción a lenguaje máquina del bucle interno del algoritmo de la burbuja. El algoritmo tiene dos bucles, el bucle externo no se muestra y el bucle interno recorre los elementos de un vector y los compara dos a dos. En el caso de estar desordenados se efectúa un intercambio del contenido de las posiciones que se han comparado. El bucle externo utiliza la variable cambios para determinar si ha finalizado la ordenación. Para ello inicializa con el valor cero esta variable antes de iniciar otra ejecución del bucle interno.

```
do I =1, N
    if (A(I) > A(I+1)) then
        temp = A(I)
        A(I) = A(I+1)
        A(I+1) = temp
        cambios = cambios +1
    endif
enddo
```

```
1$:  load r8, 0 (r7)        load A(I)
     load r9, 4 (r7)        load A(I+1)
     cmple r10, r8, r9      A(I) =< A(I+1)
     bne r10, 2$            se intercambia si A(I) > A(I+1)
     store r9, 0 (r7)       store A(I)
     store r8, 4 (r7)       store A(I+1)
     add r5, r5, #1         contador de cambios
2$:  add r6, r6, #1         contador de iteraciones
     add r7, r7, #4         índice del vector A
     cmple r11, r6, r4      r6 =< r4
     bne r11, 1$            iterar si aún no ha finalizado
```

El registro r4 se inicializa con el número de iteraciones y el registro r6 con el valor uno. El registro r7 se inicializa con la dirección base del vector A y el registro r5 se inicializa con el valor cero.

Para contestar a las siguiente preguntas dibuje un diagrama temporal que muestre la interpretación de las instrucciones de una iteración del código descrito previamente. Además, calcule el número de ciclos perdidos debidos a riesgos de secuenciamiento y a riesgos de datos y los ciclos por instrucción (CPI). Adicionalmente suponga en todos los casos que al interpretar la instrucción “bne r11, 1$“ el salto es tomado.

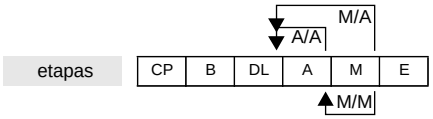
Pregunta 1: *No se cumple la condición en la instrucción "bne r10, 2$".*

Pregunta 2: *Se cumple la condición en la instrucción "bne r10, 2$".*

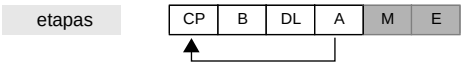
Pregunta 3: *Suponga que al ejecutar las N iteraciones del bucle interno del algoritmo de la burbuja se producen un 50% de intercambios. Determine el CPI medio.*

Mecanismos para reducir la latencia efectiva de las instrucciones

En el camino de datos se añaden cortocircuitos para reducir la latencia de uso de un valor calculado por una instrucción. En la siguiente figura se muestran los cortocircuitos disponibles indicando la etapa productora de valor y la etapa consumidora de valor.



También se modifica el camino de datos para reducir la latencia de las instrucciones de secuenciamiento condicional. En la figura se muestra que la etapa CP se alimenta desde la etapa ALU con el objetivo de reducir la latencia efectiva.



Adicionalmente, en los ciclos en los que aún no se conoce la dirección destino de la instrucción de secuenciamiento se predice seguir en secuencia. En consecuencia en la etapa ALU se comprueba la predicción efectuada.

Riesgos y actuación

Cuando se detecta un riesgo de secuenciamiento (etapa DL) se predice seguir en secuencia. Si la predicción ha sido errónea (se comprueba en etapa ALU) se descarta la información de las etapas previas y se inyectan, mediante circuitería, instrucciones NOP en las etapas oportunas.

Para contestar a las siguiente preguntas dibuje un diagrama temporal que muestre la interpretación de las instrucciones de una iteración del código descrito previamente. Además, calcule el número de ciclos perdidos debidos a riesgos de secuenciamiento y a riesgos de datos y los ciclos por instrucción (CPI). Adicionalmente suponga en todos los casos que al interpretar la instrucción "bne r11, 1$" el salto es tomado.

Pregunta 4: *No se cumple la condición en la instrucción "bne r10, 2$".*

Pregunta 5: *Se cumple la condición en la instrucción "bne r10, 2$".*

Pregunta 6: *Suponga que al ejecutar las N iteraciones del bucle interno del algoritmo de la burbuja se producen un 50% de intercambios. Determine el CPI medio.*

Pregunta 7: *Calcule la ganancia cuando se añaden al procesador los mecanismos que permiten reducir la latencia efectiva de las instrucciones.*

Ejercicio 4.23

Un procesador que interpreta el siguiente repertorio de instrucciones:

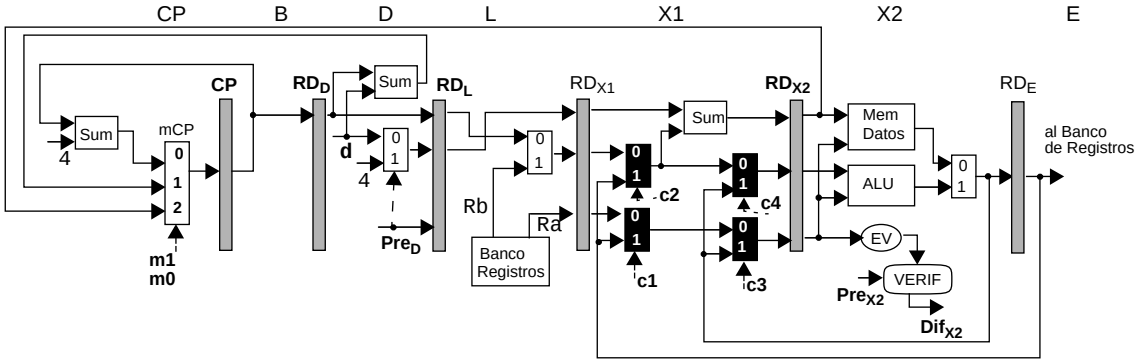
Tipo	Semántica	Lenguaje ensamblador
ENT	Rd = Ra op Rb	Add Rd, Ra,Rb
LD	Rd = M[Rb + d]	Load Rd,d(Rb)
ST	M[Rb + d] = Ra	Store Ra,d(Rb)
BC	si cond (Ra) entonces CP = CP + d en caso contrario CP = CP + 4	Bne Ra, dirección destino

está segmentado linealmente en 7 etapas:

ETAPA	FUNCIONALIDAD
CP	determinación de la dirección de la instrucción
B	búsqueda de la instrucción
D	decodificación, detección riesgos de datos, predicción de sentido y cálculo de la dirección destino (BC)
L	lectura de operandos en el banco de registros, cálculo de las señales de control de los cortocircuitos
X1	cálculo de la dirección efectiva (LD, ST), cálculo de la dirección de recuperación (BC)
X2	acceso a memoria (LD, ST), operación aritmético-lógica (ENT), evaluación de la condición y verificación de la predicción (BC)
E	escritura del resultado en el banco de registros (ENT, LD)

El procesador dispone de recursos suficientes para interpretar cualquier secuencia de instrucciones sin riesgos estructurales. Un registro del banco de registros se puede escribir y leer, en este orden, en el mismo ciclo de reloj. Los riesgos de datos se detectan en la etapa D. Cuando la lógica de control detecta un riesgo de datos, se detiene el flujo de instrucciones entre las etapas CP, B, D y se inyectará una NOP hacia la etapa L.

La figura muestra el esquema simplificado del camino de datos.



Hay 4 cortocircuitos. Observe que los multiplexores de encaminamiento de datos para cortocircuitos están ubicados en la etapa X1. El cálculo de las señales de control (c1, c2, c3, c4) de los multiplexores se efectúa en la etapa L. A partir de la etapa D, el camino de datos dispone de señales indicando el tipo de instrucción que se está procesando en cada etapa (señales ENT_Z, LD_Z, ST_Z y BC_Z, donde Z es el nombre de la etapa). Estas señales no se muestran en la figura.

Pregunta 1: *Deduzca los retardos Productor-uso según el tipo de instrucción. Tenga en cuenta que una pareja de instrucciones tipo X - tipo Y puede tener más de un retardo.*

Pregunta 2: *Dibuje el cronograma de ejecución de la siguiente secuencia de instrucciones:*

```
Load r0,0(r0)
Add r0,r0,r1
Sub r3,r2,r0
Load r0,8(r0)
```

Indique los valores que han de tomar las señales c1, c2, c3, c4 en los ciclos 5, 6, 7 y 8.

Pregunta 3: *Diseñe el módulo de control de los multiplexores de cortocircuito y de detección de riesgos de datos. No es necesario considerar las señales de validación de los identificadores de registro.*

Para interpretar las instrucciones de salto condicional, el procesador utiliza predicción de sentido y ésta es función del signo del desplazamiento (d). En la etapa D se calcula la dirección destino del salto y se modifica el secuenciamiento si el desplazamiento es negativo (Pre_D=1). La verificación de la predicción se efectúa en la etapa X2, modificando el secuenciamiento si el resultado de la evaluación de la condición no coincide con la predicción (Dif_{X2}=1). Para obtener la dirección de la siguiente instrucción que se interpreta después de un error de predicción, en la etapa D se selecciona la constante (4 o el desplazamiento) y en la etapa X1 se efectúa el cálculo.

Los registros de desacoplo en los cuales se pueden inyectar NOPs son los registros RD_D, RD_L y RD_{X2}. En caso de error de predicción, la lógica de control deberá actuar durante 2 ciclos de reloj para eliminar del camino de datos, lo más pronto posible, las instrucciones buscadas antes de detectar el error de predicción.

En las siguientes preguntas, el valor constante especificado en una instrucción de salto representa la dirección destino del salto.

Pregunta 4: *Dibuje el cronograma que muestre la actuación de la lógica de control cuando el procesador ejecuta la secuencia de instrucciones, suponiendo que se produce un error de predicción.*

```
200  Beq r0,400
204  Load r0,0(r4)
208  Load r1,8(r4)
212  Add r2,r1,r0
216  Store r2,8(r0)
220  Add r3,r3,r5
     . . .
400  Load r0,0(r10)
404  Load r1,8(r10)
```

Pregunta 5: *Indique las penalizaciones cuando se interpreta una instrucción de salto condicional.*

Pregunta 6: *Deduzca las expresiones lógicas de las señales de control del multiplexor de la etapa CP.*

Pregunta 7: *Suponga que en el ciclo 10 los registros del banco de registros tienen los valores r0=0, r1=1, r2=2, y el registro de desacoplo CP=200. Considerando el siguiente fragmento de código:*

```
100  Bne r2,400
104  Load r3,4(r5)
108  Sub r5,r6,r7
     . . .
200  Bne r0,400
204  Beq r1,100
208  Add r8,r8,r9
     . . .
400  Load r9,-4(r10)
404  Xor r7,r7,r7
408  Store r7,0(r9)
```

Indique, para cada instrucción de salto, en qué ciclo se efectúa la predicción, el sentido de la predicción, en que ciclo se verifica la predicción y el resultado de la verificación.
Determine la secuencia de valores que toma el registro CP entre los ciclos 11 y 16.

Pregunta 8: *Conociendo que entre los ciclos 10 y 15 (ambos incluidos) el registro de desacoplo CP toma los valores 200, 204, 208, 212, 212 y 400 cuando el procesador interpreta el siguiente trozo de código:*

```
200  Beq r0,400
204  Load ri,8(r10)
208  Load rk,0(rj)
212  Store r2,-4(rm)
216  Add r6,r7,r6
     . . .
400  Load r0,0(r10)
```

Deduzca unos posibles identificadores de registro ri, rj, rk y rm. Sólo se pueden utilizar los registros entre el r0 y el r5, ambos incluidos.