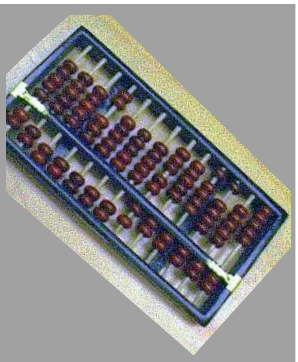
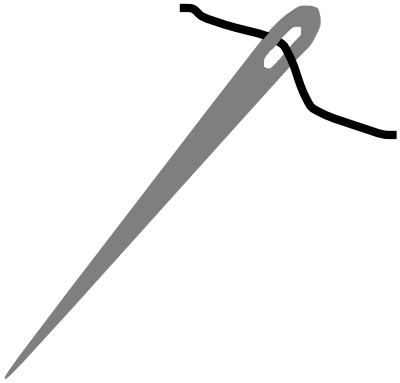


Arquitectura de Computadores



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

J.M. Llabería
E. Herrada
A. Olivé

Contenidos

Capítulo 3 Procesador segmentado lineal

3

PROCESADOR SEGMENTADO LINEAL

En este capítulo se utiliza la técnica de segmentación para incrementar el número de instrucciones que interpreta por ciclo el camino de datos de un procesador. La segmentación es una técnica de implementación del camino de datos y actualmente es una técnica clave en el diseño de procesadores.

El objetivo de aplicar la técnica de segmentación es reducir el tiempo de ejecución de los programas de forma transparente al programador de lenguaje máquina. Esto es, el código ejecutable de un programa que se interpreta en un procesador no segmentado se interpreta sin modificaciones en un procesador con el camino de datos segmentado y el resultado es el mismo.

La ganancia ideal en prestaciones al aplicar la técnica de segmentación se verá reducida por trabas. Una de ellas es la falta de recursos en el camino de datos que prohibe en ocasiones que varias instrucciones se puedan interpretar concurrentemente. La otra traba es que hay que acomodar el solapamiento en la interpretación de instrucciones con la semántica del lenguaje máquina para que el resultado sea el mismo que una interpretación serie.

Este capítulo está dedicado a la adecuación de un procesador segmentado para interpretar un lenguaje máquina que es de tipo imperativo.

Contenido

INTRODUCCIÓN

Un procesador serie no empieza a interpretar una instrucción hasta que ha finalizado la interpretación de la instrucción previa. En la Figura 3.1 se muestra la ejecución de varias instrucciones que efectúan la suma de dos números almacenados en memoria y almacena el resultado en memoria. Los acrónimos CP, BUS, D/L, ALU, M y ES se utilizan para indicar respectivamente las fases de determinación de la dirección de la instrucción, búsqueda de la instrucción, decodificación y acceso al banco de registros para leer los datos, utilización de la ALU, acceso a memoria para leer/escribir un dato y escritura del resultado en registros.

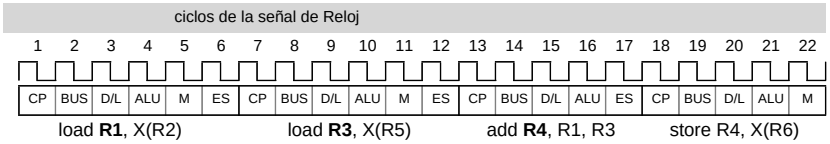


Figura 3.1 *Interpretación de una secuencia de instrucciones en un procesador serie que efectúa la suma de dos números. La instrucción interpretada se muestra en la parte inferior de la figura.*

El CPI en la secuencia de instrucciones de la Figura 3.1 es igual a 22/4 = 5.5 ciclos/instrucción.

En una secuencia de código en lenguaje ensamblador el registro destino se especifica a la izquierda, después del acrónimo de la instrucción y seguidamente se especifican los registros fuentes. Además, para ayudar a la legibilidad se utiliza negrita para el registro destino.

Segmentación

Para reducir el tiempo de ejecución de un programa se utiliza la técnica de segmentación en la implementación del camino de datos y se solapa la interpretación de instrucciones, con el objetivo ideal de empezar a interpretar una instrucción en cada ciclo de reloj.

La segmentación reduce el CPI de forma transparente al programador de lenguaje máquina. Esto es, los programas compilados para un procesador serie no necesitan modificarse. En términos de CPI la técnica de segmentación pretende que idealmente el CPI sea igual a 1.

Existen dos tipos de trabas que prohiben o dificultan conseguir un CPI igual a 1. Una de ellas son los recursos hardware disponibles y otra es que hay que respetar la semántica del lenguaje con que se expresa el programa.

Cuando se solapa la interpretación de instrucciones varias de ellas pueden querer utilizar el mismo recurso en el mismo ciclo. En estas condiciones, sólo una instrucción, de las que entran en conflicto, puede proseguir el proceso de interpretación y la otra u otras instrucciones deben esperarse. Los efectos debidos a la falta de recursos estructurales y su gestión en un dispositivo segmentado se han analizado en el Capítulo 2, no siendo por tanto objeto de estudio en este capítulo.

El lenguaje máquina que se utiliza para expresar un programa es un lenguaje de tipo imperativo. Mantener la semántica quiere decir mantener el orden, explicitado en el código, de las acciones de lectura y escritura en cada una de las posiciones de almacenamiento utilizadas (registros y memoria).

En un procesador serie una instrucción empieza a interpretarse cuando ha finalizado la interpretación de la instrucción previa. Entonces, el orden de lecturas y escrituras explicitado en el código no se modifica y además, siempre se conoce cual es la instrucción que debe interpretarse seguidamente.

Sin embargo, al segmentar el camino de datos y solapar la interpretación de varias instrucciones el orden de las lecturas y escrituras de una posición de almacenamiento de datos, especificada por el programador, puede modificarse. Cuando una de estas situaciones se produzca, habrá que retardar el proceso de interpretación de la instrucción que no respeta el orden y siguientes. Igualmente, la segmentación puede dar lugar a que se interprete una secuencia de instrucciones distinta de la que se interpreta en un procesador serie.

En la Figura 3.2 se muestra la segmentación en etapas del proceso de interpretación para los distintos tipos de instrucciones, la cual es la que se ha obtenido en el Capítulo 2 después de

eliminar los riesgos estructurales. Los acrónimos utilizados para identificar las etapas se refieren a las fases de interpretación de la instrucción. En la fase de ejecución se distingue la utilización de la ALU y el acceso a memoria (M). La lógica utilizada para evaluar la condición de secuenciamiento se incluye en la etapa ALU. Se dispone de dos caminos de lectura y un camino de escritura al banco de registros y en el mismo ciclo puede escribirse y leerse, en este orden, un registro.

Supondremos que la instrucción store finaliza después de 6 ciclos de interpretación. En estas condiciones todas la instrucciones tardan 6 ciclos en interpretarse.

		ciclos					
Tipo	instrucción	1	2	3	4	5	6
MEM	load	CP	BUS	D/L	ALU	M	ES
	store	CP	BUS	D/L	ALU	M	
ENT / IS		CP	BUS	D/L	ALU		ES

Figura 3.2 *Segmentación del proceso de interpretación de instrucciones. Una etapa sin etiqueta indica un retardo.*

En la Figura 3.3 se muestra un posible solapamiento al interpretar de forma segmentada el ejemplo de la Figura 3.1. En una fila de la figura se muestra la interpretación de una instrucción. En una columna, que se corresponde con un ciclo, se observan varias instrucciones interpretándose, cada una de ellas en una etapa distinta.

	ciclos												
instrucción	1	2	3	4	5	6	7	8	9	10	11	12	13
1. load **R1**, X(R2)	CP	BUS	D/L	ALU	M	ES							
2. load **R3**, X(R5)		CP	BUS	D/L	ALU	M	ES						
3. add **R4**, R1, R3			CP	BUS	D/L	D/L	D/L	ALU		ES			
4. store R4, X(R6)				CP	BUS	BUS	BUS	D/L	D/L	D/L	ALU	M	

Figura 3.3 *Interpretación de instrucciones en un camino de datos segmentado. La instrucción interpretada se muestra en la parte izquierda de la figura.*

Para calcular el CPI supondremos que la secuencia de instrucciones se ha extraído de una secuencia mayor de instrucciones. Para determinar el número de ciclos utilizados en ejecutar las instrucciones se contabiliza el número de ciclos desde que finaliza la primera instrucción hasta que finaliza la última instrucción, incluyendo estos dos ciclos. El número de ciclos es 8 y el número

de instrucciones es 4. Por tanto, el CPI es igual a 8/4 = 2.0, lo cual está alejado del valor ideal que es 1, debido a las trabas que retardan la interpretación de las instrucciones. Observe que algunas instrucciones permanecen varios ciclos en la misma etapa, siendo ello debido a las trabas y al mecanismo utilizado para gestionarlas.

En los siguientes apartados se analiza como respetar la semántica del código que se interpreta en un procesador segmentado: a) orden de lecturas y escrituras a posiciones de almacenamiento de datos y b) secuenciamiento de instrucciones. Por ejemplo, en la segmentación de la Figura 3.2 todas las instrucciones leen los operandos del banco de registros en la tercera etapa (D/L) y escriben el resultado, en la posición de almacenamiento especificada, en la última o penúltima etapa de interpretación. En el ejemplo de la Figura 3.3 la 3ª instrucción utiliza como operandos el resultado de las instrucciones 1ª y 2ª. La 2ª instrucción escribe en el banco de registros en el ciclo 7. Entonces, la 3ª instrucción permanece tres ciclos en la etapa D/L para adecuar las semánticas del procesador y del lenguaje.

Este capítulo está dedicado a la construcción de un procesador segmentado para interpretar un lenguaje máquina que es de tipo imperativo.

En el próximo capítulo se presentan técnicas de compilación e implementación del camino de datos que reducen o eliminan la pérdida de rendimiento debido a las trabas impuestas por la semántica del lenguaje.

Control de riesgos estructurales

En el Capítulo 2 se ha tratado en detalle la gestión de los riesgos estructurales. Aquí efectuamos un resumen.

Cuando dos instrucciones quieren utilizar un recurso en el mismo ciclo, el control decide que utiliza el recurso la instrucción que hace más tiempo que ha iniciado el proceso de interpretación.

La ocupación de recursos por parte de una instrucción se conoce como muy pronto en la etapa D/L, cuando se decodifica el tipo de instrucción. En cada ciclo el control conoce el ciclo en el cual las instrucciones, que están en etapas posteriores a la etapa de

decodificación, utilizan los recursos. Si el control determina que se producirá un conflicto no permite que la instrucción en la etapa de decodificación prosiga. Tampoco prosiguen el proceso de interpretación instrucciones posteriores.

Actuación cuando se detecta un riesgo estructural. El control no permite que la instrucción en la etapa de decodificación y posteriores prosigan el proceso de interpretación.

La actuación del control emula una ejecución serie ya que serializa la interpretación de las instrucciones.

CAMINO DE DATOS SEGMENTADO LINEAL

El lenguaje máquina para el que se describe el camino de datos se ha presentado en el Capítulo 2. En la Figura 3.4 se muestra una segmentación unificada del proceso de interpretación para todas las instrucciones. Todas las instrucciones pasan por todas las etapas y de forma implícita se conocen los recursos que utiliza en cada etapa cada tipo de instrucción. En ocasiones una etapa representa exclusivamente un retardo, ya que no se transforma información.

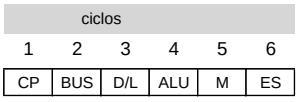


Figura 3.4 *Etapas de un procesador segmentado lineal.*

El resto de la sección está dedicado a la explicación del camino de datos segmentado mostrado en la Figura 3.5. Todos los registros de desacoplo (BUS_DL, DL_ALU, ALU_MEM, MEM_ES), menos el registro etiquetado CP, se han dibujado mediante una línea gruesa y no se muestra la señal de reloj. Para conocer cuando se usa un elemento del camino de datos hay que observar sus conexiones de entrada y salida (registros de desacoplo), no importa su ubicación espacial en el dibujo. Para visualizar los campos de una instrucción se muestran los campos de las instrucciones de tipo RR inmediatamente después del registro de desacoplo BUS_DL. Los campos de las otras instrucciones son similares al que se muestra como ejemplo.

Todos los registros de desacoplo entre etapas se actualizan en el flanco ascendente de la señal de reloj. Este es el instante en el cual consideramos que empieza un ciclo de la señal de reloj. Se dispone del tiempo que dura el periodo de la señal de reloj para procesar la información disponible en el(los) registro(s) de desacoplo de la entrada antes de que la información en las señales de salida de la etapa se almacene en el registro de desacoplo que existe en la salida de la etapa.

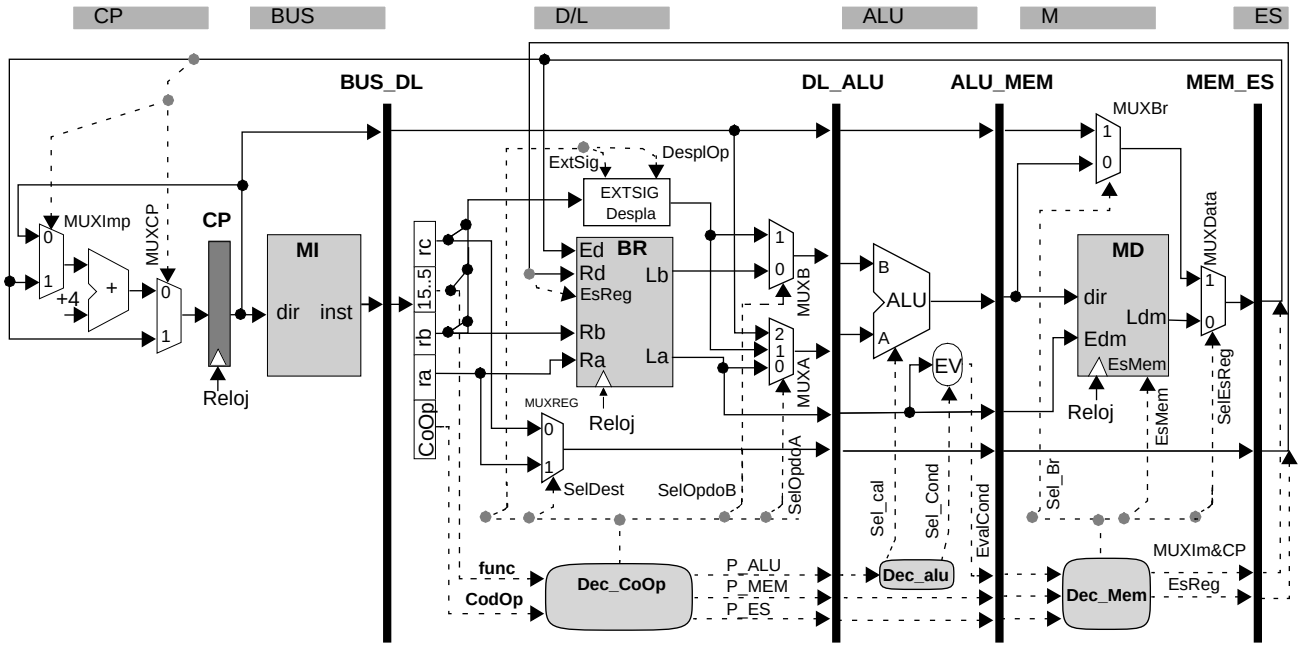


Figura 3.5 *Camino de datos segmentado.*

En las memorias MI y MD supondremos que nunca se producen fallos de cache. Esto es, MI y MD siempre almacenan el código o datos accedidos. Los elementos de memorización MI, BR y MD están sincronizados por la misma señal de reloj que los registros de desacoplo. Si es necesario, internamente se generan señales para efectuar, en un ciclo de la señal de reloj, las operaciones que se describen posteriormente. Por ejemplo, el banco de registros genera internamente señales para que en un ciclo de la señal de reloj se pueda escribir y leer el mismo registro, suponiendo que el tiempo de ciclo lo permite.

En el elemento MI no se explicitará la señal de reloj en las figuras ya que se utiliza exclusivamente para leer instrucciones. Los elementos BR y MD disponen de las señales EsReg y EsMem respectivamente que se utilizan para indicar cuando se efectúa una operación de escritura.

En la Figura 3.5 las señales con trazo continuo son datos o identificadores de registro y las señales con trazo discontinuo son información de control. Las señales de datos transportan 64 bits y las señales de control los bits necesarios para efectuar la función encomendada.

La mayoría de las señales van de izquierda a derecha, en el sentido de interpretación de una instrucción. Sin embargo, como al interpretar una instrucción se accede al banco de registros y al registro CP en etapas distintas, para leer y después escribir, existen señales de derecha a izquierda (parte superior de la figura). Estos elementos se han ubicado espacialmente en el dibujo en la etapa que se leen. Por razones de espacio en el dibujo, las señales que van de derecha a izquierda transportan datos e información de control. Esta información se agrupa después del registro de desacoplo MEM_ES y se separa en las etapas destino CP y D/L.

En cada ciclo todos las cables transportan información. La utilización de esta información en una etapa concreta depende del tipo de instrucción que utiliza la etapa y la fase de interpretación en la cual se utiliza la información. Cuando la información se obtiene o se genera en una etapa distinta a la que se consume hay que transportarla desde el lugar de origen al lugar de utilización atravesando etapas. Por ejemplo, el identificador del registro destino de una instrucción se obtiene en la etapa D/L y se utiliza en la etapa ES; transcurren 3 ciclos. Entonces, el identificador de registro destino viaja con la instrucción por la etapas hasta que se utiliza.

En el camino de datos presentado nos restringimos a instrucciones de secuenciamiento de tipo BR condicionales e incondicionales. Las de secuenciamiento condicional utilizan el elemento EV para evaluar la condición especificada en la instrucción. En un capítulo posterior se analizan los otros tipos de instrucciones de secuenciamiento.

Seguidamente se describe el flujo de la información de control. Posteriormente se describe la utilización del camino de datos por parte de cada tipo de instrucción.

Una vez analizado cada tipo de instrucción se describen las etapas empezando por la etapa CP y siguiendo en orden se finaliza en la etapa ES. Por último se muestran, en un ciclo de la señal de reloj, las interrelaciones entre etapas.

Control del camino de datos

El control del camino de datos segmentado se muestra en la parte inferior de la Figura 3.5. La información de control fluye entre etapas y en cada una de ellas se decodifica para encaminar y manipular los datos.

Control estacionario en los datos.

En cada etapa se decodifica la información de control suministrada por la etapa previa para determinar como hay que encaminar y manipular los datos.

En la etapa D/L el módulo Dec_CoOP, además de controlar la propia etapa, prepara la información para que sea decodificada y utilizada en las siguientes etapas (P_ALU, P_MEM, P_ES). Las señales de control se utilizan para encaminar la información, mediante los multiplexores, hacia los elementos del camino de datos e indicar a los circuitos combinacionales de cómputo o a los circuitos secuenciales (registros, memoria) la operación que debe efectuarse con la información disponible en sus entradas.

Notemos que la salida del elemento EV en la etapa ALU es información de control. Esta información se utiliza en la etapa M junto con el tipo de instrucción para seleccionar entre la dirección de la instrucción y el valor calculado en el elemento ALU en el ciclo previo.

Descripción del camino de datos utilizado por cada instrucción

En este apartado se describe la utilización de los elementos del camino de datos para cada tipo de instrucción. En los esquemas que se presentan del camino de datos, para facilitar la identifi-

cación de los elementos que utiliza cada instrucción, se han suprimido los elementos no utilizados por el tipo de instrucción que se analiza. Sin embargo, para constatar que la interpretación de instrucciones de distinto tipo utilizan partes comunes del camino de datos, se han dejado los multiplexores que permiten encaminar las señales en función del tipo de instrucción. Igualmente, para simplificar el dibujo, se han suprimido las señales de control y los elementos que las generan.

El secuenciamiento por defecto es interpretar las instrucciones en secuencia (secuenciamiento implícito). Esto es, suponiendo un formato fijo de 4 bytes, para calcular la dirección de una instrucción se suma 4 a la dirección de la instrucción previa (Figura 3.5 y Figura 3.6). En la etapa CP el registro CP se actualiza en cada ciclo utilizando esta hipótesis. Por tanto, se garantiza que se suministra una dirección en cada ciclo a la etapa BUS y ésta puede suministrar una instrucción por ciclo a la etapa D/L.

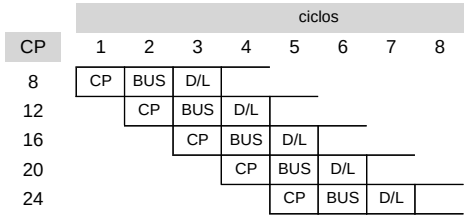


Figura 3.6 *Secuenciamiento implícito de instrucciones.*

En la descripción de todas las instrucciones excepto las instrucciones de secuenciamiento supondremos que el secuenciamiento es implícito.

Instrucciones tipo RR

Las instrucciones aritmético-lógicas tipo RR leen sus dos datos fuente del banco de registros (ra y rb) y almacenan el resultado en un registro destino (rc) (Figura 3.7).

Figura 3.7 *Formato de las instrucciones tipo RR.*

En la etapa CP se calcula la dirección de la instrucción suponiendo secuenciamiento implícito (Figura 3.8).

En el siguiente ciclo se utiliza la dirección calculada para acceder a MI y obtener la instrucción. En el tercer ciclo de interpretación se accede al banco de registros para leer los datos fuente y decodificar la instrucción. Los identificadores de los registros que se leen se obtienen de los campos de la instrucción ra y rb. Las entradas del banco de registros correspondientes a los identificadores de los caminos de lectura son Ra y Rb y los caminos que transportan los datos son La y Lb respectivamente.

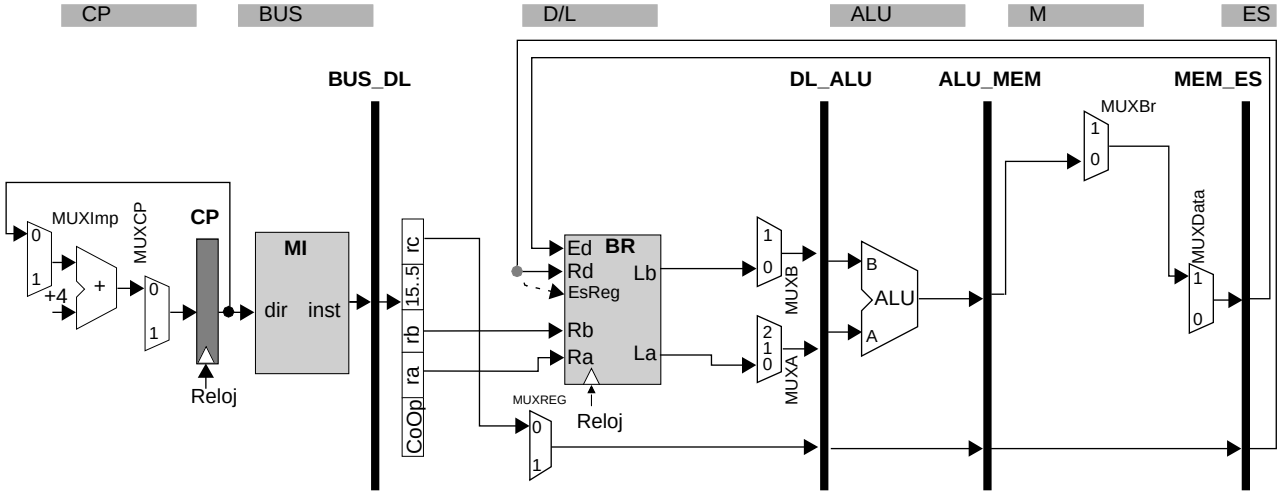


Figura 3.8 *Camino de datos utilizado por las instrucciones de tipo RR.*

En el cuarto ciclo se inicia la fase de ejecución de la instrucción en la etapa ALU. En el quinto ciclo no se transforma información y en el siguiente ciclo se escribe en el banco de registros utilizando el identificador del registro destino (rc) que ha viajado con la instrucción a través de las etapas ALU y MEM desde la etapa D/L.

Instrucciones tipo RI

Las instrucciones aritmético-lógicas de tipo RI leen un dato fuente del banco de registros (ra) y el otro dato es el campo literal de 8 bits especificado en la instrucción, el cual se interpreta como un número natural. El resultado se almacena en un registro (rc) (Figura 3.9).

	campos de la instrucción																			
	31	. . .	26	25	...	21	20	...	16	15	...	12	11	...	5	4	...	0
tipo	6			5			5			4			7			5		
RI	CoOp			ra			literal						1	func		rc		

Figura 3.9 *Formato de las instrucciones tipo RI.*

La única diferencia con la instrucción de tipo RR es que un dato es el campo literal. Entonces, en la etapa D/L se selecciona el campo literal como dato, después de formatearlo, mediante el multiplexor MUXB en lugar del contenido de un registro (Figura 3.10).

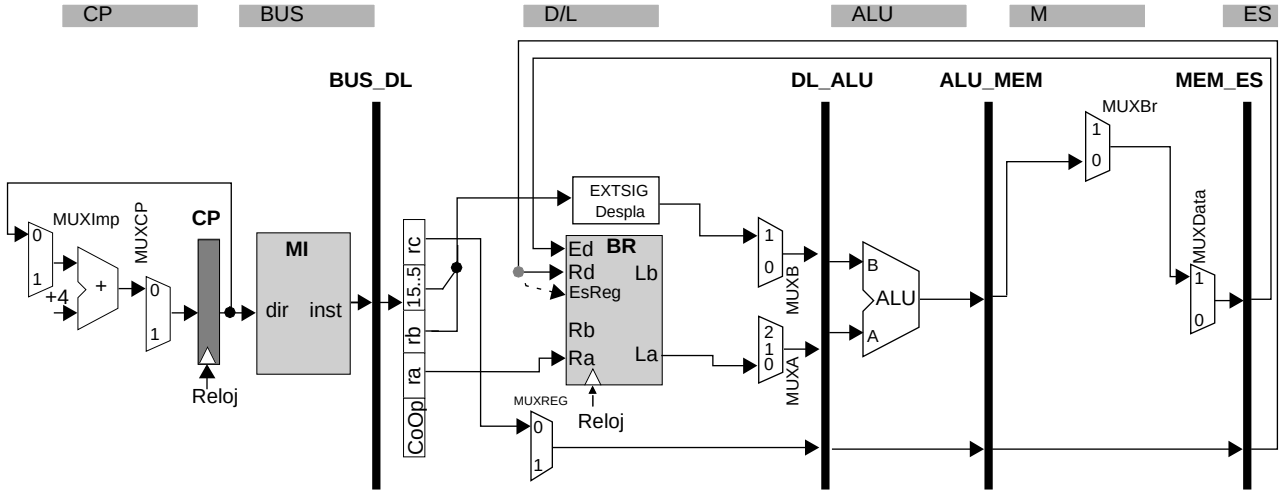


Figura 3.10 *Camino de datos utilizado por las instrucciones de tipo RI.*

LOAD

La instrucción load utiliza el contenido del registro rb y el campo literal especificado en la instrucción para calcular la dirección efectiva de la posición de memoria accedida. El campo literal es de 16 bits y se interpreta como un número entero. El valor leído de memoria se almacena en el registro ra (Figura 3.11).

	campos de la instrucción			
	31 . . . 26	25 ... 21	20 ... 16	15 ... 12 11 ... 5 4 ... 0
	6	5	5	16
Load	CoOp	ra	rb	literal

Figura 3.11 *Formato de la instrucción load.*

En las etapas CP y BUS se efectúan las mismas acciones que en los tipos de instrucciones descritos previamente. En la etapa D/L el dato que se lee del banco de registros está disponible en el camino Lb y el campo desplazamiento de la instrucción es el otro dato (Figura 3.12). El campo ra de la instrucción se encamina como identificador del registro destino. En la etapa ALU se calcula la dirección efectiva utilizando el elemento ALU y en la etapa M se utiliza la dirección calculada para acceder a MD. En la etapa ES se actualiza el banco de registros.

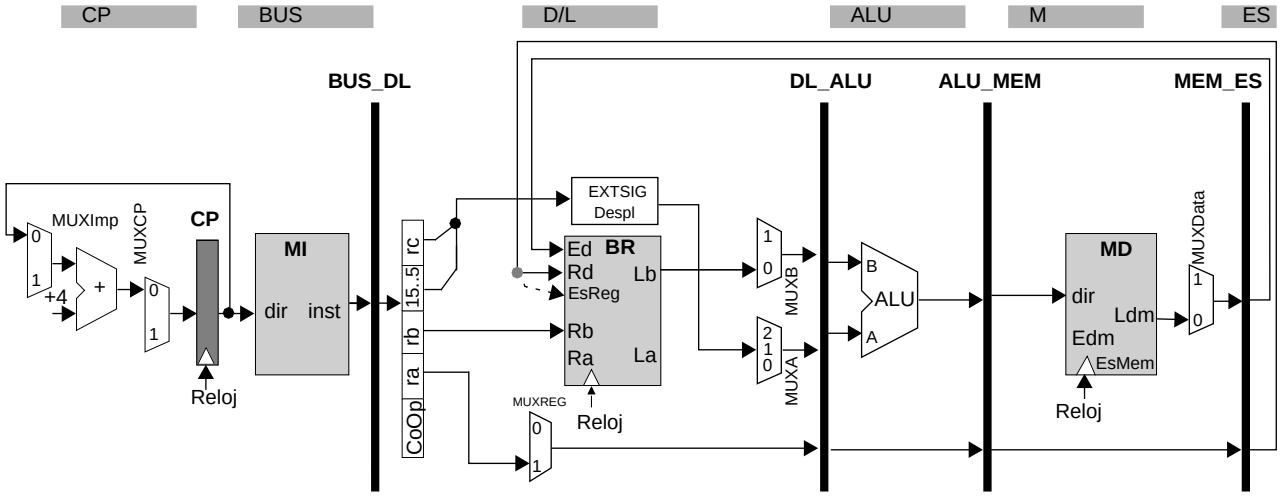


Figura 3.12 *Camino de datos utilizado por la instrucción load.*

STORE

El formato es idéntico al de la instrucción load y se interpreta de la misma forma con la excepción del campo ra. Este campo se utiliza como identificador del registro cuyo contenido se almacena

en memoria. En la etapa D/L se lee el contenido del registro ra, el cual se propaga en la etapa ALU (Figura 3.13) para ser utilizado en la etapa M.

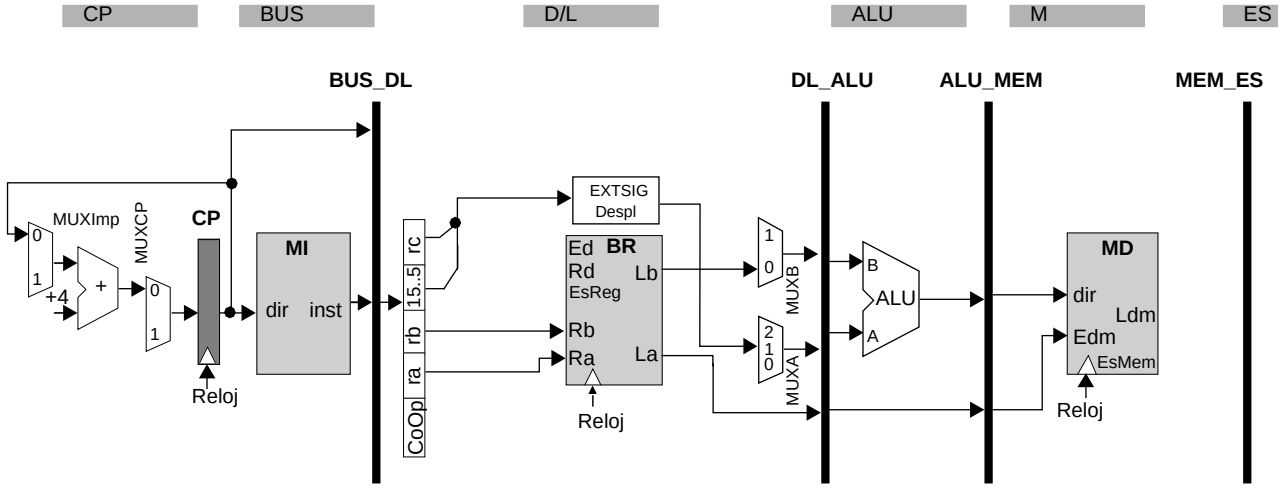


Figura 3.13 *Camino de datos utilizado por la instrucción store.*

BR

Este tipo de instrucción de secuenciamiento calcula una dirección efectiva utilizando la dirección de la instrucción de secuenciamiento y el literal especificado en la instrucción. El campo literal es de 21 bits y se interpreta como un número entero múltiplo de 4.

Recordemos que sólo consideraremos instrucciones de secuenciamiento de tipo BR condicionales e incondicionales.

Las instrucciones de secuenciamiento condicional utilizan el contenido del registro ra para evaluar la condición especificada de forma implícita en el código de operación (CoOp) (Figura 3.14).

	campos de la instrucción		
	31 . . . 26	25 ... 21	20 ... 16	15 ... 12	11 ... 5	4 ... 0
tipo	6	5	21
BR	CoOp	ra	literal

Figura 3.14 *Formato de las instrucciones tipo BR.*

Dirección destino. Dirección de la siguiente instrucción que se interpreta después de una instrucción de secuenciamiento incondicional o después de una instrucción de secuenciamiento condicional cuando se cumple la condición.

Para efectuar el cálculo de la dirección destino, la dirección de la instrucción se propaga desde la etapa BUS hasta la etapa D/L (Figura 3.15). En la etapa D/L se selecciona esta dirección como dato para suministrarlo a la etapa ALU, donde se efectúa el cálculo de la dirección. Para el caso de instrucciones de secuenciamiento condicional también se transmite a la etapa ALU el contenido del registro ra, donde se evalúa la condición utilizando el módulo EV.

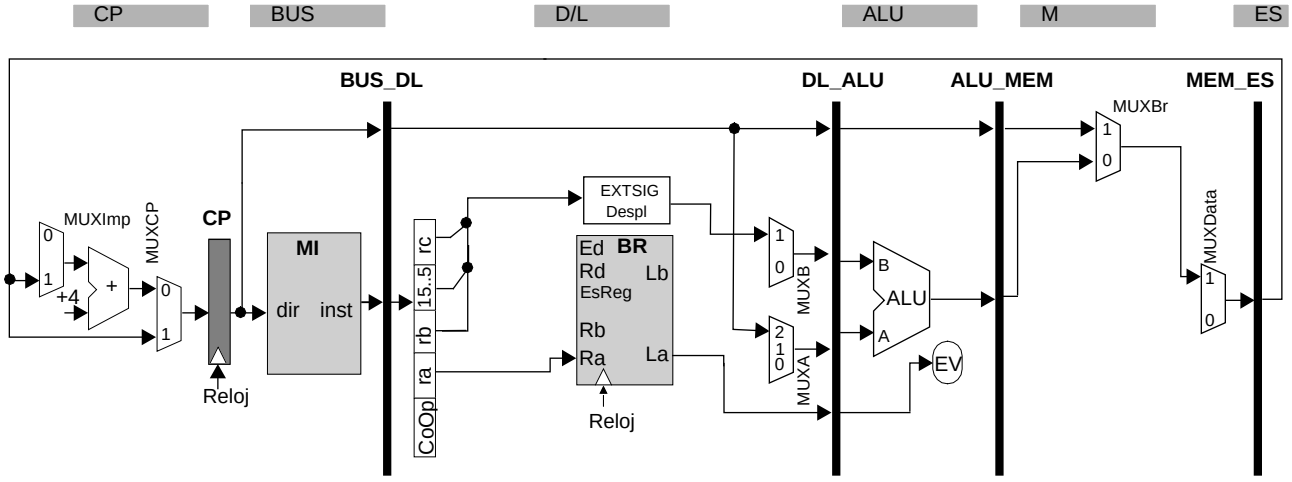


Figura 3.15 *Camino de datos utilizado por las instrucciones de tipo BR.*

La dirección de la instrucción de secuenciamiento se transmite hasta la etapa M, donde se efectúa la selección de la dirección que debe utilizarse para buscar la siguiente instrucción. Posteriormente, desde la etapa ES se alimenta a la lógica de la etapa CP para actualizar el registro CP.

En la etapa CP se suma 4 a la dirección suministrada desde la etapa ES. El multiplexor MUXCP se utiliza para seleccionar entre la dirección suministrada desde la etapa ES o la salida del sumador de la etapa CP. La dirección transmitida desde la etapa ES se selecciona si la instrucción es de secuenciamiento incondicional o se cumple la condición. En caso contrario, se selecciona la salida del sumador de la etapa CP.

Descripción de las etapas

En este apartado se describe cada una de las etapas individualmente. En concreto, se describen los registros de desacoplo de entrada y salida de cada etapa y la información que se transmite entre etapas.

Contador de programa (CP)

El registro de desacoplo en la salida de la etapa CP se denomina CP ya que almacena la dirección de la instrucción que se va a buscar en MI.

Las señales de entrada de la etapa CP provienen de los registros de desacoplo CP y MEM_ES (Figura 3.16). En la etapa se dispone de un sumador y dos multiplexores. Mediante el sumador se calcula la dirección de la siguiente instrucción en secuencia. El multiplexor MUXCP se utiliza para seleccionar entre la dirección de la siguiente instrucción en secuencia y la dirección efectiva seleccionada en una instrucción de secuenciamiento.

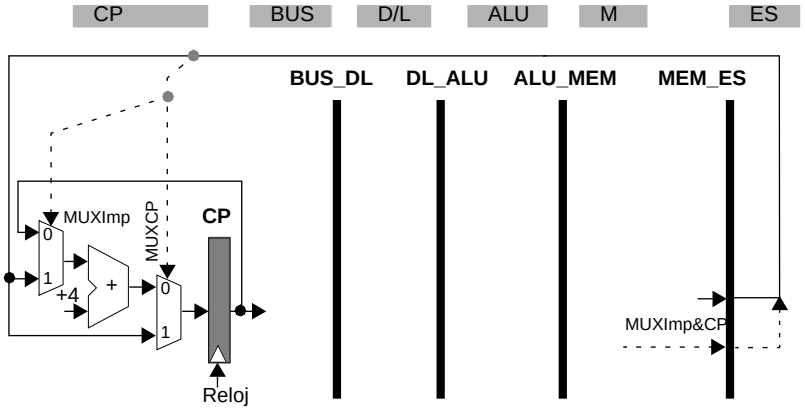


Figura 3.16 *Elementos de la etapa CP.*

El multiplexor MUXImp se utiliza para seleccionar entre la dirección que se está utilizando en la etapa BUS y la dirección suministrada desde la etapa ES. Si la instrucción es de secuenciamiento incondicional se selecciona siempre la entrada uno del multiplexor.

En una instrucción de secuenciamiento condicional se evalúa una condición y se selecciona la entrada uno del multiplexor MUXImp. Cuando no se cumple la condición, la dirección que se

recibe es la dirección de la instrucción de secuenciamiento, ya que se debe seguir buscando instrucciones en secuencia. Para ello, al valor recibido desde la etapa ES se le suma 4 (número de bytes de una instrucción) y se selecciona la entrada cero del multiplexor MUXCP. Si se cumple la condición se recibe la dirección destino (calculada en la ALU), ya que hay que modificar el secuenciamiento implícito, y se selecciona la entrada uno del multiplexor MUXCP.

El control de los multiplexores se efectúa con señales provenientes del registro de desacoplo MEM_ES.

Búsqueda (BUS)

En la etapa de búsqueda (Figura 3.17) tenemos el elemento MI que almacena el código. Para acceder a MI se utiliza la dirección almacenada en el registro CP.

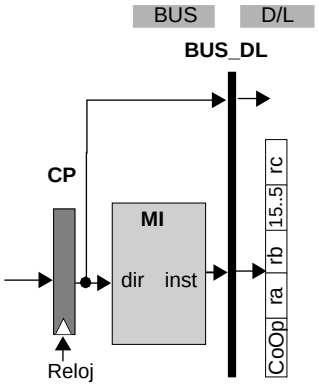


Figura 3.17 *Elementos de la etapa Búsqueda.*

El registro de desacoplo a la salida de esta etapa se denomina BUS_DL y almacena la instrucción que se ha leído de MI y la dirección de la instrucción.

Decodificación y lectura de registros fuente (D/L)

En la Figura 3.18 se muestran los elementos de la etapa D/L. Esta etapa tiene como entrada la información proveniente del registro de desacoplo BUS_DL. En la parte izquierda de la figura, después del registro de desacoplo BUS_DL, se muestran los campos de una instrucción.

El banco de registros es un circuito secuencial ya que mantiene memoria y como tal una de sus entradas es la señal de reloj. Mientras no se diga lo contrario supondremos que en un ciclo de la señal de reloj se puede escribir y leer, en este orden, el mismo registro del banco de registros, sin excluir que sean registros distintos. Los otros elementos incluidos en la etapa D/L son circuitos combinacionales.

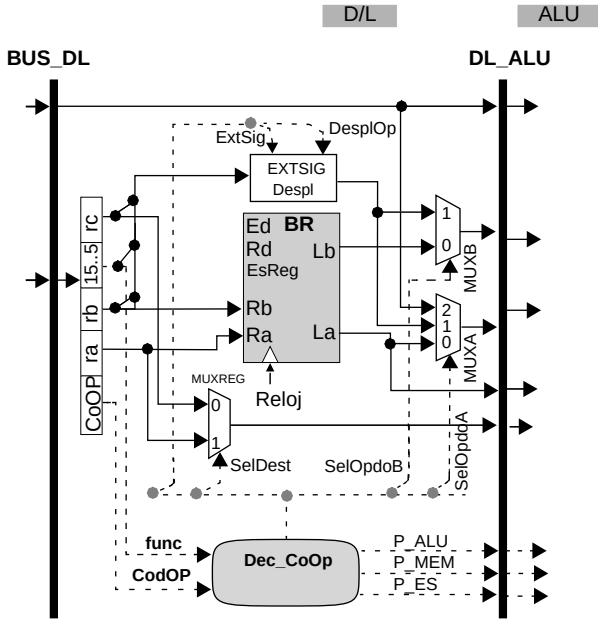


Figura 3.18 *Elementos de la etapa decodificación y lectura de los registros fuente.*

La información de lectura del banco de registros proviene directamente del registro de desacoplo BUS_DL. Las entradas del banco de registros correspondientes a los identificadores de los caminos de lectura son Ra y Rb y los caminos que transportan los datos son La y Lb respectivamente.

La instrucción se decodifica para conocer los datos fuente de la instrucción que pueden ser: a) el contenido de un registro, b) un campo literal o c) la dirección de la instrucción de tipo BR.

Las instrucciones tienen formato de tamaño fijo y los posibles identificadores de los registros fuente están siempre en la misma posición. Entonces, mientras se decodifica la instrucción se

accede en paralelo al banco de registros. Posteriormente, una vez se ha decodificado la instrucción, se utilizan o no los valores leídos.

Los multiplexores al final de la etapa seleccionan entre varias fuentes de información. El multiplexor MUXB permite seleccionar entre el contenido de un registro y el campo literal previamente formateado. En el caso del multiplexor MUXA la tercera entrada es la dirección de la instrucción de tipo BR.

El registro de desacoplo en la salida de la etapa se denomina DL_ALU. La información de datos que almacena es: a) los datos que procesará la ALU (salida de MUXA y MUXB), b) el contenido del registro leído por el camino etiquetado como La y c) la dirección de la instrucción.

El registro DL_ALU almacena la siguiente información de control: a) el identificador de registro destino de la instrucción (salida de MUXREG) y b) información de control para las siguientes etapas (P_ALU, P_MEM, P_ES).

Cálculo (ALU)

En la etapa ALU (Figura 3.19) se opera con los datos suministrados por la etapa D/L. Los elementos combinacionales son: a) ALU y b) EV. Este segundo elemento se utiliza para efectuar la evaluación de la condición en instrucciones de secuenciamiento condicional. La ALU efectúa los cálculos aritmético-lógicos especificados en la instrucción.

El registro de desacoplo en la salida de la etapa se denomina ALU_MEM. Los datos de salida de esta etapa son: a) el resultado del cálculo efectuado por la ALU, b) el contenido del registro leído en la etapa D/L por el camino etiquetado como La, c) la dirección de la instrucción.

La información de control incluye la generada en la etapa D/L (P_MEM, P_ES) y el resultado de la evaluación efectuada en EV.

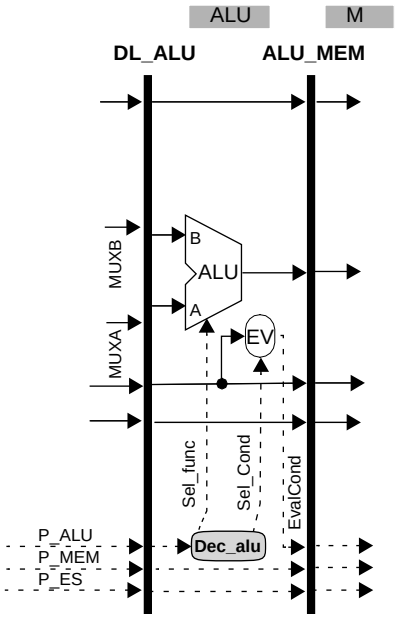


Figura 3.19 *Elementos de la etapa ALU.*

Memoria de datos (M)

En la etapa de memoria de datos (Figura 3.20) se accede a la memoria de datos (MD) si es una instrucción load/store o representa un retardo para otros tipos de instrucciones. El elemento MD es un circuito secuencial y como tal una de sus entradas es la señal de reloj. Otra entrada (EsMem) indica si hay que efectuar una operación de lectura (load) o una operación de escritura (store).

Esta etapa se utiliza para efectuar la selección de dirección en instrucciones de secuenciamiento condicional (MUXBr). La selección se efectúa utilizando el resultado de evaluar la condición (Sel_Br) y las posibles direcciones son: a) dirección de la instrucción y b) dirección destino calculada en la unidad ALU en el ciclo previo. En el caso de una instrucción de secuenciamiento incondicional se selecciona siempre la dirección calculada en la ALU.

Al final de la etapa hay un multiplexor (MUXData) que se utiliza para seleccionar entre la lectura efectuada en MD o la salida del multiplexor MUXBr.

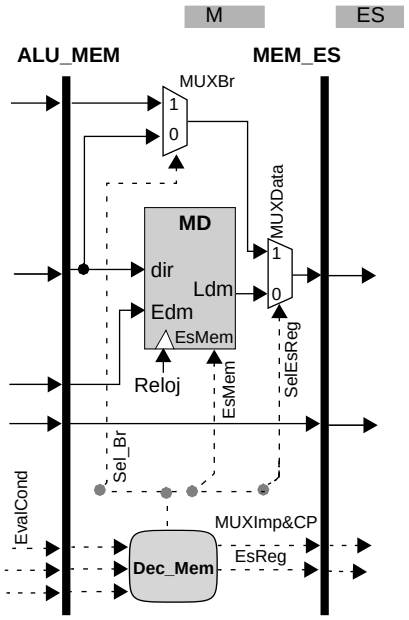


Figura 3.20 *Elementos de la etapa memoria de datos.*

El registro de desacoplo en la salida de la etapa se denomina MEM_ES.

Los información de salida de la etapa es:

a) Para instrucciones de secuenciamiento. Dirección de la instrucción de secuenciamiento o dirección destino. En este último caso se ha cumplido la condición evaluada. Además, se generan señales de control para los multiplexores (MUXCP, MUXImp) ubicados en la etapa CP.

b) Para las otras instrucciones excluyendo la instrucción store. Dato que debe almacenarse en el banco de registros y las señales que gestionan la escritura en el banco de registros (identificador del registro destino y señal de escritura en registros).

Escritura en el banco de registros (ES)

En el banco de registros la entrada correspondiente al identificador del registro de escritura es Rd, la entrada del dato es Ed y la señal que indica si hay que efectuar la operación de escritura se denomina EsReg (Figura 3.21).

La descripción detallada de las señales en el registro de desacoplo MEM_ES se ha efectuado al describir la etapa memoria de datos. La información de datos se encamina al banco de registros desde la etapa ES.

El identificador de registro y la señal EsReg también se encaminan al banco de registros desde la etapa ES.

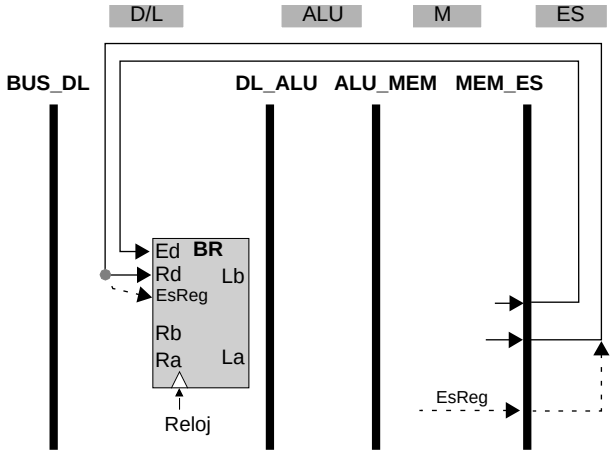


Figura 3.21 *Elementos de la etapa escritura.*

Ciclo de la señal de reloj

En la Figura 3.22 se muestran las etapas del procesador dibujadas en vertical para evidenciar que todas las etapas están procesando instrucciones de forma concurrente. La etapa ES está en la parte superior del dibujo y la etapa CP en la parte inferior. Esta visión se corresponde con cualquiera de las columnas de los diagramas temporales que se utilizan para mostrar la interpretación segmentada y solapada de una secuencia de instrucciones (Figura 3.23). Para simplificar el dibujo se han suprimido las señales de control y los elementos que las generan. Notemos que el registro de desacoplo de salida de una etapa se vuelve a dibujar como registro de desacoplo de entrada de la siguiente etapa.

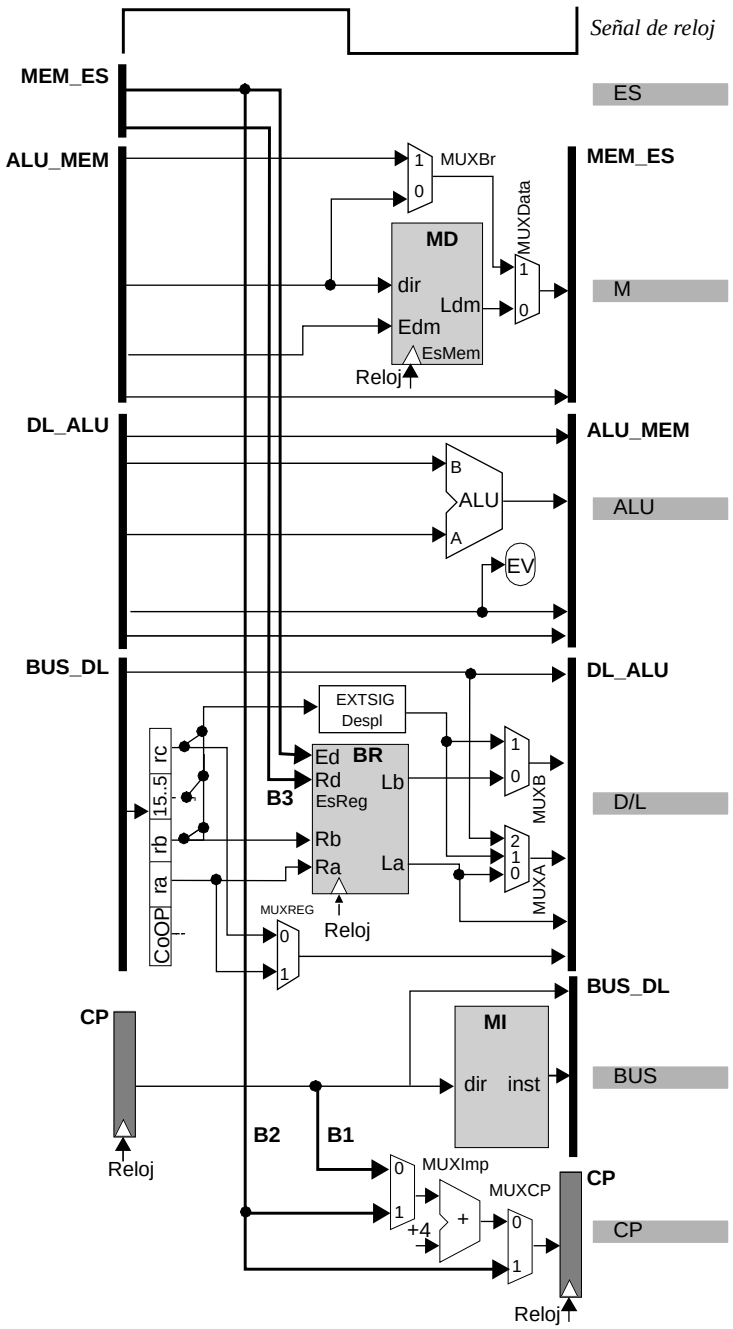


Figura 3.22 *Visión de las etapas en un ciclo de la señal de reloj.*

En el inicio de un ciclo de la señal de reloj todas las etapas empiezan a procesar la información que hay en los registros de desacoplo en la entrada de la etapa (parte izquierda de la Figura 3.22). Al final del ciclo de la señal de reloj, siguiente flanco ascendente, la información en las señales de salida de cada etapa se almacena en los registros de desacoplo en la salida de la etapa (parte derecha de la Figura 3.22).

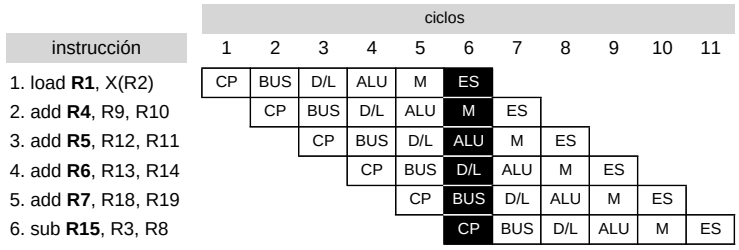


Figura 3.23 *Interpretación de una secuencia de instrucciones en un procesador segmentado.*

Observemos que la etapa CP tiene entradas provenientes de los registros de desacoplo MEM_ES y CP. También, el recurso banco de registros (BR) es accedido desde dos etapas (D/L y ES).

Lazo o bucle hardware. Es una comunicación entre etapas que permite que en una etapa se utilice información suministrada desde etapas posteriores. Por tanto, esta información está determinada por instrucciones en fases más avanzadas del proceso de interpretación.

Los bucles hardware son necesarios ya que en las secuencias de instrucciones que se interpretan: a) el resultado de una instrucción lo puede utilizar una instrucción posterior o b) una instrucción puede determinar cuál es la siguiente instrucción que debe interpretarse.

En la Figura 3.22 se observan, con líneas de trazo más ancho, 3 bucles hardware (B1, B2 y B3). Dos de ellos corresponden al secuenciamiento (B1 y B2). En el caso de secuenciamiento implícito la información proviene de la etapa BUS (B1) y en el caso de una instrucción de secuenciamiento la información proviene de la etapa ES (B2).

El otro bucle hardware es debido a la actualización del recurso banco de registros (B3). Notemos que una instrucción puede utilizar como dato el resultado que una instrucción previa ha almacenado en el banco de registros.

Estos bucles hardware determinan que sea necesario añadir control, en el camino de datos segmentado, para que se respete la semántica del lenguaje máquina. Esta adecuación de semánticas, la del camino de datos a la del lenguaje máquina, se desarrolla en los siguientes apartados.

SEMÁNTICA DEL PROCESADOR SEGMENTADO

En este apartado utilizamos dos ejemplos para mostrar que hay que adecuar la semántica del camino de datos segmentado a la semántica del lenguaje máquina. Para denotar los casos en que la semántica es distinta utilizaremos la palabra riesgo.

En la Figura 3.24 se muestra, mediante un diagrama temporal, la interpretación de una secuencia de instrucciones en el camino de datos segmentado de la Figura 3.22. La primera de las instrucciones calcula un resultado que se almacena en el registro R1. La 2ª instrucción utiliza el contenido del registro R1 como dato.

En un camino de datos no segmentado la 2ª instrucción no empieza a interpretarse hasta que ha finalizado la interpretación de la 1ª instrucción. Por tanto, el registro R1 ha sido escrito y la 2ª instrucción lo utiliza como dato.

Sin embargo, en el camino de datos segmentado, la interpretación de la 2ª instrucción puede dar lugar a un resultado incorrecto, ya que cuando se accede al banco de registro en el ciclo 4 la primera instrucción aún no ha actualizado el registro R1. Igualmente ocurre con la 3ª instrucción. En cambio la 4ª instrucción lee el valor calculado por la 1ª instrucción. Por tanto, hay que diseñar mecanismos de control para, en situaciones como la descrita, adecuar las semánticas.

Observe que transcurren 3 ciclos desde el inicio de la fase de ejecución de una instrucción ENT o load hasta que el valor está disponible en el banco de registros. A este número de ciclos se le denomina latencia efectiva de la segmentación.

Latencia efectiva de la segmentación. Número de ciclos que transcurren desde el ciclo en que se leen los datos hasta el primer ciclo en que se puede utilizar el resultado de un cálculo.

Notemos que en el caso de instrucciones de secuenciamiento uno de los datos es la dirección de la instrucción, el resultado es una dirección y la posición de almacenamiento es el registro CP.

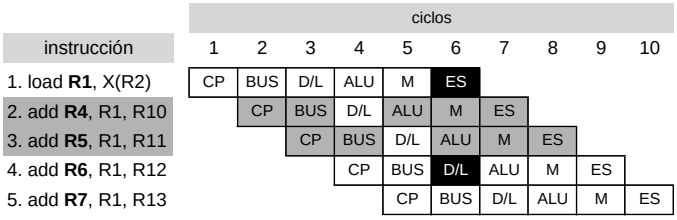


Figura 3.24 *Riesgo de datos. La segmentación modifica el orden, especificado en el código, de las escrituras y lecturas sobre una posición de almacenamiento.*

En la Figura 3.25 se muestra una secuencia de instrucciones donde la primera instrucción es una instrucción de secuenciamiento condicional.

dirección	instrucción
8	beq R1, 1$
12	load **R3**, X(R5)
16	add **R4**, R9, R3
20	store R7, X(R6)
24	sub **R10**, R12, R13
. . .	. . .
60	1$: load **R17**, X(R15)
64	add **R4**, R6, R17

Figura 3.25 *Secuencia de instrucciones que incluye una instrucción de secuenciamiento condicional.*

En un camino de datos no segmentado no se empieza a interpretar la instrucción que sigue a beq R1,1$ hasta que ésta ha finalizado. Por tanto, se conoce la dirección de la instrucción que debe interpretarse. Si no se cumple la condición, la instrucción interpetada es load R3,X(R5) y en caso contrario, la instrucción interpretada es load R17, X(R15).

En la Figura 3.26 se muestra la interpretación de la secuencia de instrucciones de la Figura 3.25 en un camino de datos segmentado. Supongamos que se cumple la condición y por tanto se modifica el secuenciamiento implícito. La actualización del registro CP tiene lugar en el ciclo 6 (Figura 3.26). Mientras tanto, se han seguido interpretando instrucciones en secuencia que actualizan el estado del procesador, lo cual en un procesador serie no se habría efectuado. Por tanto, hay que diseñar mecanismos de control para adecuar las semánticas.

Observemos que transcurren 5 ciclos desde que se lee la dirección de una instrucción de secuenciamiento hasta que se actualiza el registro CP con la dirección de la siguiente instrucción que debe interpretarse. Estos ciclos son la latencia efectiva en actualizar el registro CP.

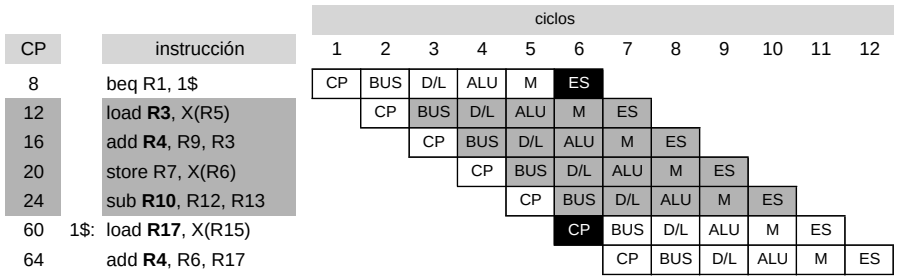


Figura 3.26 *Riesgo de secuenciamiento. La segmentación modifica el secuenciamiento especificado en el código.*

Las situaciones descritas previamente se denominan riesgos de datos y riesgos de secuenciamiento.

Riesgo de datos. Modificación del orden de lecturas y escrituras, especificado por el programador, sobre una posición de almacenamiento al interpretar de forma solapada una secuencia de instrucciones en un camino de datos segmentado.

Riesgo de secuenciamiento. Interpretación de una secuencia de instrucciones distinta de la especificada por el programador, cuando el código se interpreta de forma solapada en un camino de datos segmentado.

Para gestionar los riesgos hay que añadir al camino de datos segmentado un control de riesgos. Su función es detectarlos y actuar.

Lógica de interbloqueos de la segmentación (o lógica de interbloqueos). Hardware que detecta los riesgos debidos a la segmentación y actúa para asegurar que se respete la semántica del lenguaje máquina.

La detección de riesgo se efectúa en la etapa D/L, una vez se conoce el tipo de instrucción y los identificadores de los registros fuente. Una situación de riesgo de datos es con alguna instrucción en una fase posterior de interpretación. Una situación de riesgo de secuenciamiento es con instrucciones que deben interpretarse posteriormente.

El orden de actuación debe ser gestionar en primer lugar los riesgos de datos y posteriormente, una vez no existen riesgos de datos, los riesgos de secuenciamiento, ya que al interpretarse una instrucción de secuenciamiento puede existir un riesgo de datos con una instrucción previa.

Actuación del control de riesgo. Cuando se detecta un riesgo de datos o de secuenciamiento se emula un funcionamiento secuencial.

Emular un funcionamiento secuencial. Es serializar la interpretación de instrucciones cuya interpretación segmentada daría lugar a resultados incorrectos.

En los próximos apartados se muestra como se emula un funcionamiento secuencial para cada tipo de riesgo.

RIESGOS DE SECUENCIAMIENTO

Recordemos que, en el 6º ciclo de interpretación de una instrucción de secuenciamiento, la salida del registro de desacoplo MEM_ES alimenta a la lógica de la etapa CP (Figura 3.22). El valor transmitido es la dirección de la instrucción de secuenciamiento o la dirección destino. El sumador suma 4 al valor transmitido. Entonces, utilizando las señales de control, provenientes del registro de desacoplo MEM_ES, se selecciona (multiplexor MUXCP) entre el valor transmitido o la salida del sumador.

En la Figura 3.27 se muestran los bucles hardware utilizados para el secuenciamiento de instrucciones. El bucle A se utiliza en el secuenciamiento implícito y el bucle B se utiliza cuando se interpreta una instrucción de secuenciamiento.

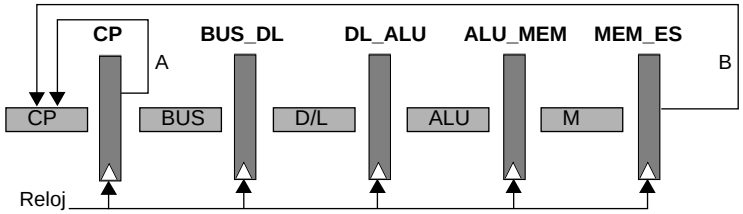


Figura 3.27 *Bucles hardware en el secuenciamiento de instrucciones. El bucle etiquetado como A se utiliza en el secuenciamiento implícito y el etiquetado como B en el secuenciamiento explícito.*

El bucle hardware A no produce ningún riesgo ya que se necesita calcular una dirección en cada ciclo y la latencia del bucle es 1 ciclo.

Sin embargo, el bucle B produce riesgos de secuenciamiento ya que la latencia del bucle es mayor que 1 ciclo. La etapa de inicio del bucle es la etapa CP, ya que es la etapa donde se lee una dirección (registro CP) que puede ser descartada, desde una etapa posterior, por una instrucción más vieja. La etapa desde donde se alimenta a la etapa de inicio, utilizando el bucle hardware, la denominaremos etapa final, siendo en este caso la etapa ES. Entre las etapas inicio y final están las etapas BUS, D/L, ALU y MEM. La latencia del bucle hardware es el número de etapas entre la etapa inicio y final, incluyendo esta última. Entonces, la latencia del bucle hardware B son 5 ciclos.

En la Figura 3.28 se muestra la interpretación de una secuencia de instrucciones donde la primera instrucción es de secuenciamiento. En la columna de la izquierda, el valor del CP que se muestra es el valor que se lee del registro CP en la etapa BUS y corresponde a la instrucción cuya interpretación se representa en la misma fila.

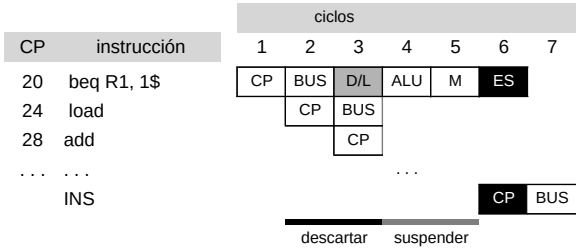


Figura 3.28 *Información en las etapas CP, BUS y D/L cuando se detecta una instrucción de secuenciamiento en la etapa D/L. INS indica instrucción que debe interpretarse después de la instrucción de secuenciamiento.*

En el ciclo que se detecta una instrucción de secuenciamiento (ciclo 3 en la Figura 3.28) en la etapa BUS se está buscando la siguiente instrucción (dirección 24) en secuencia y en la etapa CP se está calculando la dirección 28. Entonces, para respetar la semántica del lenguaje máquina es suficiente:

- Descartar las dos instrucciones posteriores que han iniciado el proceso de interpretación. Esto es, hay que eliminarlas del camino de datos.
- Suspender la interpretación de nuevas instrucciones hasta el ciclo en el cual la instrucción de secuenciamiento llega a la etapa ES (ciclo 6), ya que no se conoce la dirección de la siguiente instrucción que debe interpretarse.

Mientras se suspende la interpretación de nuevas instrucciones hay una secuencia de ciclos en los cuales las etapas (D/L, ALU, M, ES) no procesan información válida. Este último hecho hay que indicárselo a las etapas para que no se actualice el estado del procesador y a partir de ahora, cuando hablemos de suspender la interpretación de nuevas instrucciones lo consideraremos implícito.

Una forma de conseguir que no se actualice el estado del procesador es utilizar circuitería que inyecte, en el flujo de instrucciones, instrucciones nop durante los ciclos necesarios.

Actuación del control de riesgo de secuenciamiento. Se descartan de las etapas previas a D/L todas las instrucciones que han empezado a interpretarse y se suspende la interpretación de nuevas instrucciones hasta que se conoce la dirección de la instrucción que debe interpretarse.

Seguidamente se describe como se efectúan las acciones descritas. En primer lugar se muestra el efecto de las acciones en un diagrama temporal y posteriormente se describen las modificaciones necesarias en el camino de datos y su control.

Visualización de las acciones en un diagrama temporal

Para descartar las instrucciones posteriores a la instrucción de secuenciamiento tendremos en cuenta las siguientes consideraciones.

- El registro CP se actualiza con el valor correcto en el ciclo 6 (Figura 3.28).
- En la etapa BUS sólo se efectuan operaciones de lectura en el elemento MI, las cuales no modifican el estado del procesador.

Por tanto, no es estrictamente necesario eliminar de forma explícita las instrucciones de estas etapas, es suficiente con que la información que procesan no se transmita a la etapa D/L. Entonces, utilizaremos la salida de la etapa BUS para inyectar instrucciones nop, con el objetivo de que no se actualice el estado del procesador durante la latencia de actualización del registro CP.

En el diagrama temporal de la Figura 3.29 se muestra la actuación del control de riesgo de secuenciamiento. Suponemos que se cumple la condición y se modifica el secuenciamiento implícito. En el margen izquierdo, el valor que se muestra del registro CP es el valor que se lee del registro CP en la etapa BUS y corresponde a la instrucción cuya interpretación se representa en la fila.

El valor del registro 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. Sin embargo, una vez se ha detectado el riesgo de secuenciamiento y hasta el ciclo 6, en la etapa D/L se está interpretando una instrucción nop, la cual ha sido inyectada por el hardware. En el diagrama temporal cuando se inyecta una instrucción nop se explicita mediante el acrónimo nop en lugar del acrónimo de la etapa.

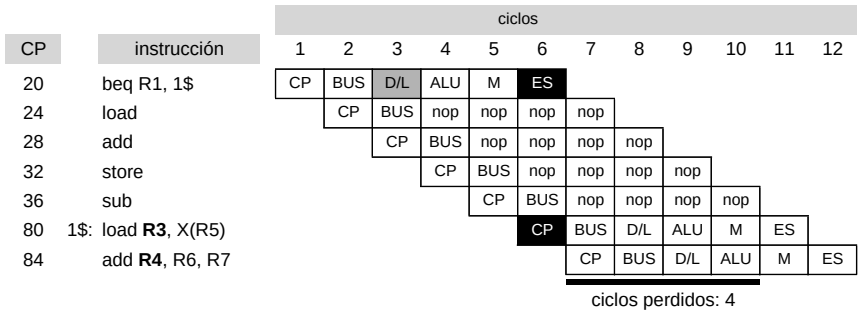


Figura 3.29 *Diagrama temporal que muestra la actuación del control de riesgos en un riesgo de secuenciamiento.*

Ciclos perdidos en un riesgo de secuenciamiento o penalización por instrucción de secuenciamiento. Latencia efectiva de la segmentación en actualizar el registro CP menos uno.

Notemos que durante varios ciclos la etapa D/L no decodifica instrucciones que pertenezcan al flujo de instrucciones que se interpretaría en un camino de datos serie.

En un diagrama temporal se identificará ciclo perdido con el ciclo en que finaliza la interpretación de una instrucción nop inyectada por el hardware. En el diagrama temporal de la Figura 3.29 los ciclos perdidos son desde el ciclo 7 hasta el ciclo 10.

En la Figura 3.30 se muestra una forma equivalente y más abreviada de representar el diagrama temporal de interpretación de instrucciones. Para ello, no se explicita el inicio de interpretación de instrucciones posteriores a la instrucción de secuenciamiento, ni la inyección de instrucciones nop. Esto es, durante la latencia de interpretación de la instrucción de secuenciamiento, no se explicitan acciones en las etapas que deja de utilizar la instrucción de secuenciamiento.

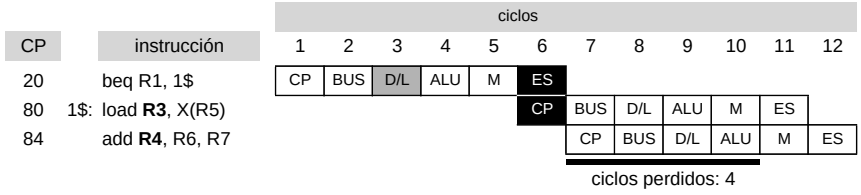
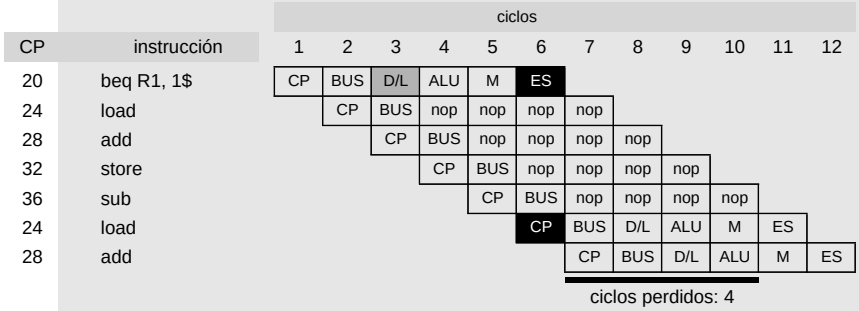


Figura 3.30 *Diagrama temporal simplificado que muestra la actuación del control en un riesgo de secuenciamiento.*

Ejercicio

Utilice la secuencia de instrucciones de la Figura 3.29 y suponga que no se modifica el secuenciamiento, ya que no se cumple la condición explicitada en la instrucción de secuenciamiento. Muestre en un diagrama temporal la interpretación segmentada de la secuencia de instrucciones cuando actúa el control de riesgo. Además, explique de forma detallada, utilizando el camino de datos, el flujo de información en la lógica de la etapa CP en el ciclo 6 (Figura 3.22).

Respuesta



En el ciclo 6 en la salida del registro de desacoplo MEM_ES se dispone de la dirección de la instrucción de secuenciamiento (20). Entonces, mediante el sumador se suma 4 a esta dirección y en el siguiente ciclo se busca en MI la instrucción que está en la dirección 24.

Control de los riesgos de secuenciamiento

La actuación descrita para gestionar un riesgo de secuenciamiento requiere modificar la salida de la etapa BUS (Figura 3.31), ya que hay que inyectar instrucciones nop. Para ello, se añade un multiplexor (MUXBusNop) que permite seleccionar entre la información transmitida en funcionamiento sin riesgo (instrucción leída en MI) y la instrucción nop. En los registros de desacoplo se distinguen dos partes mediante la tonalidad del tramado. El tramado más oscuro corresponde a datos y el tramado más claro corresponde a señales de control.

Para controlar el multiplexor MUXBusNop se utiliza la información de tipo de instrucción (BR) suministrada por el módulo Dec_CoOp y la información de tipo de instrucción que hay en los registros de desacoplo de entrada en cada etapa. En concreto nos interesan las etapas desde D/L hasta la etapa ES. Si en cualquiera

de estas etapas hay una instrucción de secuenciamiento hay que inyectar una instrucción nop en la etapa D/L al finalizar el ciclo (Figura 3.29).

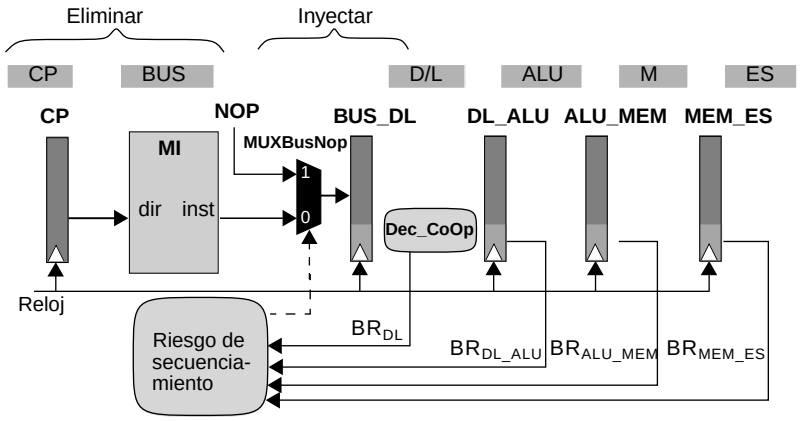


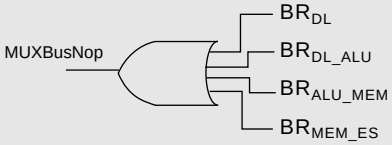
Figura 3.31 *Modificación de la etapa de búsqueda para gestionar los riesgos de secuenciamiento. Las señales BR_x indican si la etapa x está ocupada por una instrucción de secuenciamiento.*

Ejercicio

Diseñe un circuito combinacional que, a partir de la información de tipo de instrucción suministrada por el módulo Dec_CoOp y la disponible en los registros de desacoplo, controle el multiplexor MUXBusNop de la Figura 3.31. En la información de tipo de instrucción hay un bit que indica si la instrucción que ocupa la etapa es una instrucción de secuenciamiento. La señal de tipo proveniente de Dec_CoOp se denomina BR_{DL} y en los registros de desacoplo se denominan BR_{DL_ALU}, BR_{ALU_MEM} y BR_{MEM_ES}, donde el subíndice indica el registro de desacoplo.

Respuesta

Si en la etapa D/L o en alguna de las etapas desde la etapa ALU hasta la etapa ES se está interpretando una instrucción de secuenciamiento se selecciona la instrucción nop en el multiplexor MUXBusNop de la etapa BUS.



Ejercicio

En el camino de datos segmentado descrito previamente. ¿Cuál es la latencia efectiva de una instrucción de secuenciamiento?.

Supongamos que un programa tiene un 25% de instrucciones de secuenciamiento. Estas instrucciones crean situaciones de riesgo que se gestionan emulando una interpretación serie. ¿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 son 5 ciclos.

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

ciclos perdidos = f x cp = 0.25 x 4 =1

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

$CPI = CPI_{ideal}$ + ciclos perdidos = 1 + 1 = 2

Otra forma de efectuar el cálculo es utilizar directamente el CPI de cada tipo de instrucción y la proporción de instrucciones de cada tipo.

$CPI = CPI_{no\_secu} + CPI_{secu} = 0.75 \times 1 + 0.25 \times 5 = 2$

DEPENDENCIAS DE DATOS

El orden de lecturas y escrituras a las posiciones de almacenamiento explicitado en un programa se especifica mediante el concepto de dependencia de datos.

Dependencia de datos. Existe una dependencia de datos entre dos instrucciones que acceden a la misma posición de almacenamiento (registro, memoria) si como mínimo una de ellas escribe en la posición de almacenamiento.

Rango de una instrucción (R). Conjunto de posiciones de almacenamiento que escribe una instrucción.

Dominio de una instrucción (D). Conjunto de posiciones de almacenamiento que lee una instrucción.

Orden de programa. Es el orden de interpretación de las instrucciones en un procesador serie.

En la Figura 3.32 se muestran los tres tipos de dependencias y formas de nombrarlas. Para determinar el tipo de una dependencia entre dos instrucciones se utiliza el orden de programa y la acción que se realiza (lectura o escritura).

Tipo de dependencia	Ejemplos	formas de nombrarlas		siglas en inglés
$R(i) \cap D(i+k) \neq \varnothing$	i. **R1** = R2 + R3 . . . i+k. R7 = R4 + **R1**	dependencia verdadera	lectura después de escritura	RaW
$D(i) \cap R(i+k) \neq \varnothing$	i. R1 = **R2** + R3 . . . i+k. **R2** = R4 + R7	antidependencia	escritura después de lectura	WaR
$R(i) \cap R(i+k) \neq \varnothing$	i. **R1** = R2 + R3 . . . i+k. **R1** = R4 + R7	dependencia de salida	escritura después de escritura	WaW

Figura 3.32 *Tipos de dependencias y formas de nombrarlas. D y R representan respectivamente el dominio y rango de una instrucción. Se cumple k > 0, indicando que la instrucción i+k sigue a la instrucción i en orden de programa. Como ejemplos se muestran operaciones con datos en registros.*

Dependencia $R(i) \cap D(i+k) \neq \varnothing$. Una instrucción (i) escribe una posición de almacenamiento que es leída por una instrucción posterior (i+k) en orden de programa.

Dependencia $D(i) \cap R(i+k) \neq \varnothing$. Una instrucción (i) lee una posición de almacenamiento que es escrita por una instrucción posterior (i+k) en orden de programa.

Dependencia $R(i) \cap R(i+k) \neq \varnothing$. Una instrucción (i) escribe una posición de almacenamiento que es escrita por una instrucción posterior (i+k) en orden de programa.

Instrucción vieja. Teniendo en cuenta el orden de programa, una instrucción (p) se denomina vieja respecto de otra instrucción (q) si se ha especificado previamente (p < q).

Instrucción joven. Teniendo en cuenta el orden de programa, una instrucción (p) se denomina joven respecto de otra instrucción (q) si se ha especificado posteriormente (p > q).

Las dependencias de datos establecen un orden de interpretación de las instrucciones y una forma de representarlas es mediante un grafo de dependencias.

Grafo de dependencias. Grafo dirigido donde los nodos son instrucciones y los arcos representan relaciones de dependencia. El sentido de los arcos es desde la instrucción vieja (fuente de la dependencia) hacia la instrucción joven (destino de la dependencia).

En la Figura 3.33 se muestra el tipo de arco utilizado para representar cada tipo de dependencia en un grafo de dependencias.

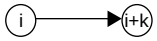
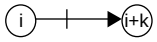
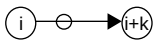
Tipo de dependencia	Ejemplos	tipo de arco
$R(i) \cap D(i+k) \neq \varnothing$	i. **R1** = R2 + R3 . . . i+k. R7 = R4 + **R1**	
$D(i) \cap R(i+k) \neq \varnothing$	i. R1 = **R2** + R3 . . . i+k. **R2** = R4 + R7	
$R(i) \cap R(i+k) \neq \varnothing$	i. **R1** = R2 + R3 . . . i+k. **R1** = R4 + R7	

Figura 3.33 *Tipos de arcos para representar las dependencias en un grafo de dependencias.*

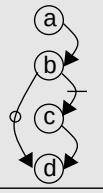
El grafo de dependencias expone de forma explícita un orden parcial entre las instrucciones, en contraposición al orden total (serie) del código. El sentido del arco en el grafo indica que la instrucción destino de la dependencia debe efectuar las lecturas o escrituras, en la posición de almacenamiento común, después de la instrucción fuente de la dependencia.

Ejercicio

Construya el grafo de dependencias de la secuencia de instrucciones que se especifica en el margen izquierdo.

Respuesta

La instrucción a actualiza uno de los registros fuente de la instrucción b. Por tanto, es una relación de dependencia $R(i) \cap D(i+k) \neq 0$. El mismo tipo de relación de dependencia existe entre las instrucciones c y d.

secuencia de instrucciones	grafo de dependencias
a. add **r1**, r2, r3	
b. sub **r4**, r1, r7	
c. add **r7**, r5, r0	
d. add **r4**, r7, r9	

El registro r7 accedido en la instrucción b determina una relación de antidependencia con la instrucción c. El registro r4 determina que las instrucciones b y d tengan una relación de dependencia de salida.

Ejercicio

Construya el grafo de dependencias de la secuencia de instrucciones que se especifica en el margen izquierdo.

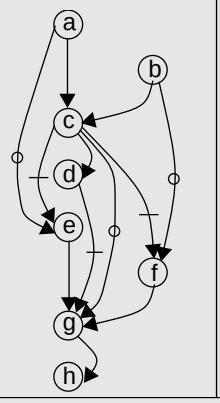
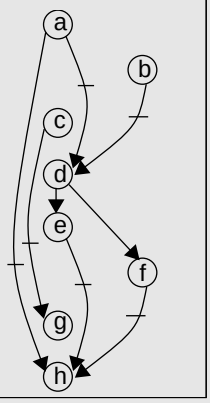
Respuesta

Para ayudar a la legibilidad del grafo de dependencias, debido al número de dependencias que tiene el código, se han dibujado dos grafos de dependencias parciales cuya unión es el grafo de dependencias. En el primer grafo de dependencias parcial se tienen en cuenta únicamente las dependencias debidas a registros y en el segundo grafo de dependencias parcial se tienen en cuenta las dependencias debidas a posiciones de memoria.

Las instrucciones a y b actualizan los registros fuente de la instrucción c. Por tanto son dependencias $R(i) \cap D(i+k) \neq \emptyset$. El mismo tipo de dependencia existe entre las instrucciones e y f y la instrucción g. También existe una dependencia verdadera entre cada uno de los pares de instrucciones (c, d) y (g, h).

Los registros r1 y r3 accedidos en la instrucción c determinan una antidependencia con las instrucciones e y f respectivamente. Estos registros junto con el registro r4 determinan que las instrucciones a, b y c tengan dependencia de salida con las instrucciones e, f y g respectivamente.

No tenemos información sobre el contenido de los registros r2, r7 y r9. Por tanto, hay que ser conservador y suponer que las posiciones de memoria accedidas por las instrucciones load y store son la misma. Entonces, hay que suponer una antidependencia de datos entre las instrucciones a y b y las instrucciones d y h.

secuencia de instrucciones	grafo de dependencias	
	registros	memoria
a. load **r1**, 0 (r2)		
b. load **r3**, 0 (r7)		
c. add **r4**, r3, r1		
d. store r4, 0 (r9)		
e. load **r1**, 4(r2)		
f. load **r3**, 4(r7)		
g. add **r4**, r3, r1		
h. store r4, 4 (r9)		

También hay que suponer dependencias verdaderas entre la instrucción d y las instrucciones e y f. Así mismo, hay que suponer una antidependencia de datos entre las instrucciones e y f y la instrucción h. Por último, no hay dependencia de salida entre las instrucciones d y h, ya que las instrucciones store utilizan el contenido del mismo registro para calcular la dirección efectiva, los campos inmediato son distintos y entre las dos instrucciones no existe ninguna instrucción que actualice el contenido del registro r9.

RIESGOS DE DATOS DEBIDOS A REGISTROS

La segmentación del camino de datos puede modificar el orden de las lecturas y escrituras sobre registros respecto de un camino de datos serie. Entonces, es necesario analizar los posibles riesgos de datos para que el control de riesgos actúe y garantice que se respeta el orden especificado por el programador.

Riesgo de datos. Detección de una dependencia de datos entre instrucciones que se ejecutan concurrentemente y además la segmentación no respeta el orden fuente-destino de la dependencia.

Cuando se interpreta una instrucción, los riesgos de datos debidos a registros se detectan en la etapa D/L. En esta etapa se dispone de los identificadores de los registros fuente y se conocen los identificadores de los registros destino de las instrucciones más viejas.

Riesgo de datos R(i)∩ D(i+k) ≠ ∅

Existe un riesgo de datos $R(i) \cap D(i+k) \neq \varnothing$ si alguno de los registros fuente de la instrucción que está en la etapa D/L lo actualiza una instrucción más vieja y aún no ha sido actualizado.

En la Figura 3.34 se muestra el bucle hardware que produce riesgos de datos debidos a registros. La etapa de inicio del bucle es la etapa D/L, ya que es la etapa donde se puede leer un dato que se actualiza desde una etapa posterior por una instrucción más vieja. La etapa final es la etapa ES. Entre estas dos etapas están las etapas ALU y MEM. La latencia del bucle hardware es el número de etapas entre la etapa inicio y final, incluyendo esta última. Entonces, la latencia del bucle hardware son 3 ciclos, ya que un mismo registro se puede escribir y leer, en este orden, en el mismo ciclo.

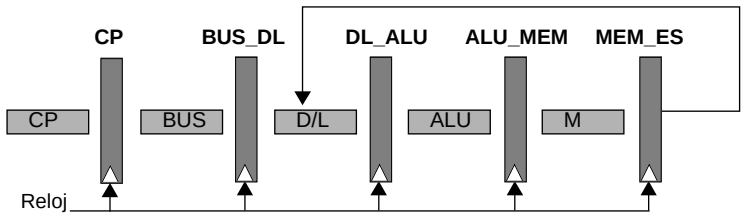


Figura 3.34 *Bucle hardware que produce riesgos de datos debidos a registros.*

En la Figura 3.35 se muestra la interpretación de una secuencia de instrucciones donde se detecta un riesgo de datos cuando se interpreta la segunda instrucción. El riesgo se detecta en la etapa D/L que es cuando se conocen los identificadores de registro de una instrucción. Para respetar la semántica del lenguaje máquina es suficiente:

- Bloquear la interpretación de la instrucción que está en la etapa D/L y las dos instrucciones más jóvenes que han iniciado el proceso de interpretación.

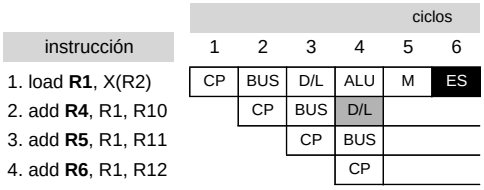


Figura 3.35 *Información en las etapas CP, BUS y D/L cuando se detecta un riesgo de datos en la etapa D/L*

Mientras se suspende o bloquea la interpretación de nuevas instrucciones hay una secuencia de ciclos en los cuales las etapas (ALU, M, ES) no procesan información válida. Este último hecho hay que indicárselo a las etapas para que no se actualice el estado del procesador. Para ello se inyectan, mediante circuitería, instrucciones nop desde la etapa D/L, mientras perdura el riesgo. De esta forma no se actualiza el estado del procesador en estos ciclos.

Actuación del control en un riesgo de datos. Se emula un funcionamiento secuencial, mediante el bloqueo del proceso de interpretación de instrucciones, a partir de la instrucción que detecta el riesgo.

Seguidamente se explica como se efectúan las acciones descritas. En primer lugar se muestra el efecto de las acciones en un diagrama temporal y posteriormente se describen las modificaciones necesarias en el camino de datos y su control.

Visualización de las acciones en un diagrama temporal

El número de ciclos de bloqueo es función de la latencia efectiva y el número de ciclos transcurridos desde que la instrucción fuente del riesgo de datos ha iniciado la fase de ejecución. La latencia efectiva en actualizar el banco de registros son 3 ciclos ya que un registro se puede escribir y leer, en este orden, en el mismo ciclo.

En la Figura 3.36 se muestra la interpretación de una secuencia de instrucciones y la actuación del control de riesgos. Para representar una instrucción con riesgo de datos se utilizan varias filas. Cada vez que se bloquea (retiene) una instrucción en una etapa se utiliza una nueva fila. Cuando la instrucción en la etapa D/L se bloquea, en el siguiente ciclo y posteriores se utiliza el acrónimo nop, en la misma fila, para indicar que se inyecta una instrucción nop.

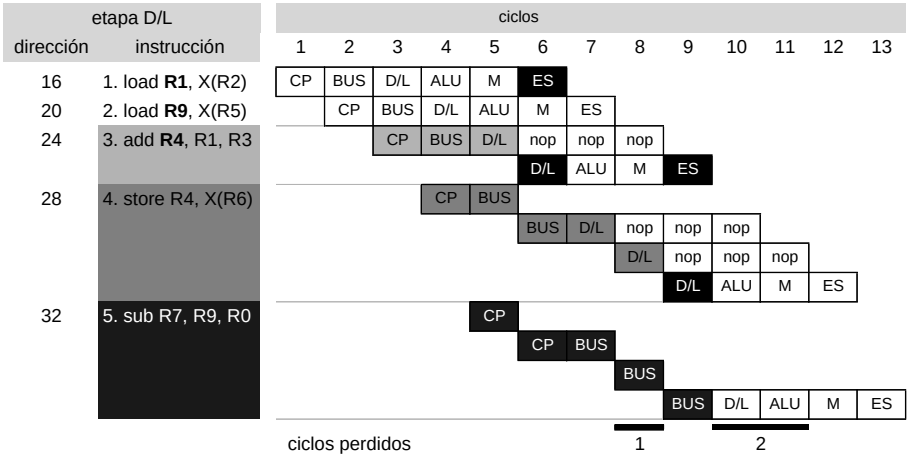


Figura 3.36 *Diagrama temporal que muestra la actuación del control de riesgos en un riesgo de datos.*

En la Figura 3.36 se observan dos instrucciones que detectan riesgos de datos. La primera de ellas es la 3ª instrucción que detecta un riesgo de datos en el ciclo 5. El control de riesgos bloquea a las instrucciones que están en las etapas CP, BUS y D/L durante un ciclo. Además, el control inyecta una instrucción nop desde la etapa D/L. En el siguiente ciclo no se detecta riesgo de datos, ya que ha transcurrido 1 ciclo desde que se ha iniciado la fase de ejecución (ciclo 4) de la instrucción fuente de la dependencia (1ª instrucción).

La segunda instrucción que detecta riesgos de datos es la 4ª. Los ciclos en que se detecta riesgo de datos son el 7 y el 8. Ello es debido a que la instrucción productora ha iniciado la fase de ejecución en el primer ciclo que la 4ª instrucción ocupa la etapa D/L. Como la latencia efectiva son 3 ciclos, deben de transcurrir 2 ciclos antes de que la instrucción productora actualice el registro.

Ciclos perdidos en un riesgo $R(i) \cap D(i+k) \neq \varnothing$ o Penalización por riesgo de datos. Latencia efectiva de la segmentación menos el número de ciclos que han transcurrido desde que se inició la fase de ejecución de la instrucción productora.

En un diagrama temporal se identificará ciclo perdido con el ciclo en que finaliza la interpretación de una instrucción nop inyectada por el hardware. En el diagrama temporal de la Figura 3.36 los ciclos perdidos son el 8, 10 y 11.

En la Figura 3.37 se muestra una forma equivalente y más abreviada de representar el diagrama temporal de interpretación de instrucciones. Para ello no se explicita la inyección de instrucciones nop y en los ciclos de bloqueo se repite el acrónimo de la etapa en la cual está retenida la instrucción. Esto es, todas las filas que se utilizan en la Figura 3.36 para representar la interpretación de una instrucción se colapsan en una sola fila y no se especifica el acrónimo nop.

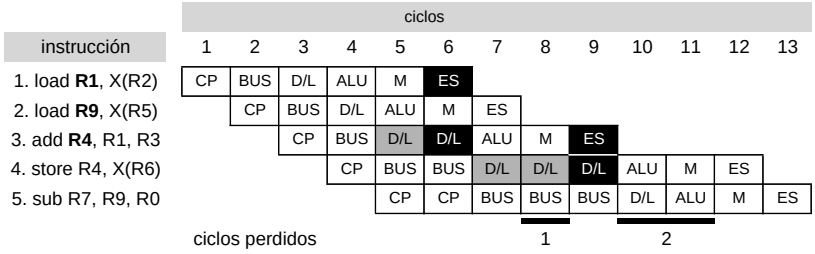


Figura 3.37 *Diagrama temporal abreviado que muestra la actuación del control de riesgos en un riesgo de datos*

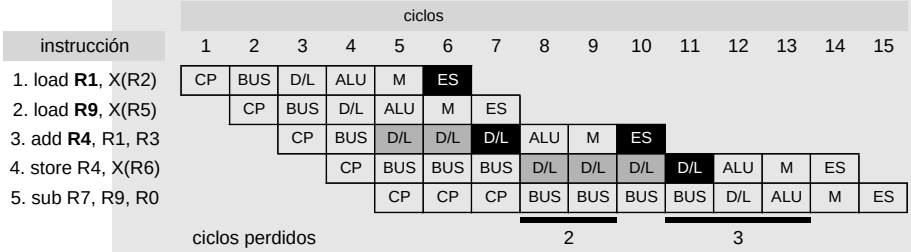
Ejercicio

Suponga un procesador con la misma segmentación que la descrita hasta ahora, excepto que es necesario todo el ciclo de la señal de reloj para escribir o leer un registro.

Muestre en un diagrama temporal la interpretación de la secuencia de instrucciones de la Figura 3.36. Así mismo, indique el número de ciclos perdidos.

Respuesta

En la siguiente figura se muestra un diagrama temporal abreviado con los ciclos perdidos al interpretar las secuencia de instrucciones.



La latencia efectiva son 4 ciclos. Entonces, el número de ciclos perdidos al interpretar la 3ª instrucción son 2 ciclos y el número de ciclos perdidos al interpretar la 4ª instrucción son 3 ciclos.

Control de los riesgos de datos

Por ahora obviaremos que el lenguaje máquina incluye instrucciones de secuenciamiento, ya que tendríamos que discutir las interacciones del bucle hardware correspondiente con el bucle hardware que produce riesgos de datos. Estas interacciones se analizarán en la última sección del capítulo. Este apunte es debido a que el registro de desacoplo MEM_ES es un registro de entrada de la etapa CP junto con el registro de desacoplo CP. En resumen, por ahora supondremos que la etapa CP tiene como único registro de desacoplo de entrada el registro CP.

Para bloquear el proceso de interpretación de instrucciones en la etapa D/L se utilizan multiplexores en la entrada de los registros de desacoplo de entrada de las etapas BUS y D/L. Una entrada del multiplexor es la información que se transmite cuando no hay riesgo y la otra entrada es la salida del registro de desacoplo.

Cuando se quiere retener una instrucción en una etapa, la información en los registros de desacoplo de entrada de la etapa no se actualiza y por tanto, en el siguiente ciclo la etapa vuelve a procesar la misma información. Con esta actuación, en el siguiente ciclo se vuelve a comprobar, en la etapa D/L, si la instrucción retenida produce algún tipo de riesgo.

En la Figura 3.38 se muestra la ubicación del multiplexor MUXCPBloq en la etapa CP, para retener respectivamente la instrucción buscada en MI y la que ocupa la etapa CP. Así mismo, se utiliza el multiplexor MUXBUSBloq para retener a la instrucción que ocupa la etapa D/L. En cada etapa sólo se muestra un multiplexor para reciclar la información de la salida del registro de desacoplo, aunque es necesario un multiplexor por cada una de las señales que se transmite entre etapas. Tampoco se muestran las señales de validación de la información suministrada al módulo Riesgo de datos. Estas señales de validación se generan en el módulo Dec_CoOp cuando se decodifica una instrucción.

En la etapa D/L (Figura 3.38) se ha ubicado el multiplexor MUXDLNop para inyectar la instrucción nop. Notemos que en este caso, a diferencia de un riesgo de secuenciamiento, la instrucción nop debe estar decodificada. Esto es, hay que transmitir a la siguiente etapa las señales de control correspondientes a una instrucción nop en lugar de las salidas del módulo Dec_CoOp (P_ALU, P_MEM, P_ES).

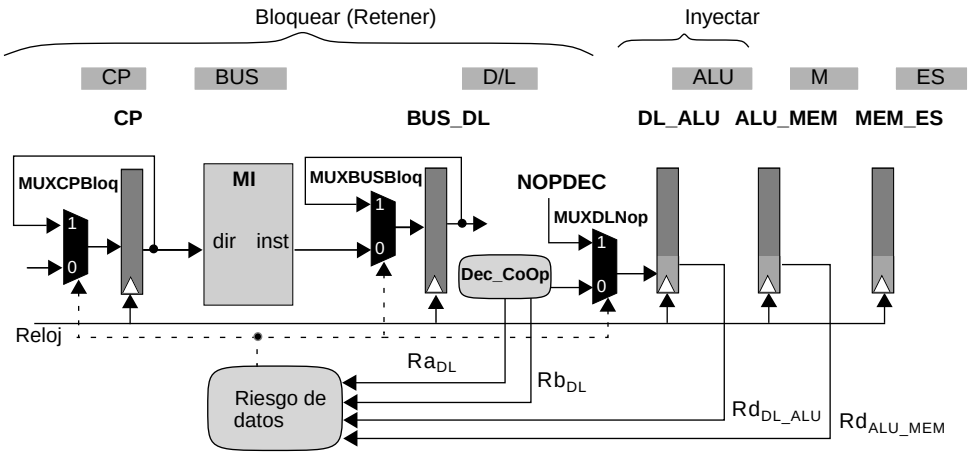


Figura 3.38 *Modificación del camino de datos para gestionar los riesgos de datos. Se bloquean las instrucciones en las etapas CP, BUS y D//L y se inyectan instrucciones nop desde la etapa D/L mientras perdura el riesgo.*

El control de los tres multiplexores (MUXCPBloq, MUXBUSBloq y MUXDLNop) lo efectúa el circuito combinacional de control de riesgos de datos. Este circuito utiliza como entradas los identificadores de los registros fuente de la instrucción en la etapa D/L y los identificadores de los registros destino de las instrucciones en las etapas ALU y M.

Los identificadores de los registros fuente de una instrucción se obtienen directamente de los campos de la instrucción que se ha leído de MI, ya que siempre, si hay que utilizarlos, están en la misma posición. Ahora bien, para cada tipo de instrucción hay que determinar si los campos ra y rb deben interpretarse como identifiadores de registro. Para ello el decodificador, en función del tipo de instrucción, genera unas señales, que denominaremos de validez, que indican si el campo debe interpretarse como un identificador de registro. El mismo razonamiento se aplica al campo rc de la instrucción.

En la Figura 3.38 no se muestran las señales de validez de los identificadores de registro de forma explícita. Sin embargo, para dejar constancia de que la interpretación de los campos ra y rb depende del decodificador, las señales Ra_{DL} y Rb_{DL}, que codifican los identificadores de registro, son salidas del módulo Dec_CoOP.

El circuito de control de riesgo indica riesgo de datos si cualquiera de los identificadores de los registros fuente de la instrucción en la etapa D/L es igual a alguno de los identificadores de registro destino de las instrucciones en las etapas ALU o M.

Ejercicio

Diseñe un circuito combinacional que, a partir del identificador de registro destino en los registros de desacoplo DL_ALU y ALU_MEM y los identificadores de los registros fuente de la instrucción en la etapa D/L, controle los multiplexores MUXCPBloq, MUXBUSBloq y MUXDLNop de la Figura 3.38.

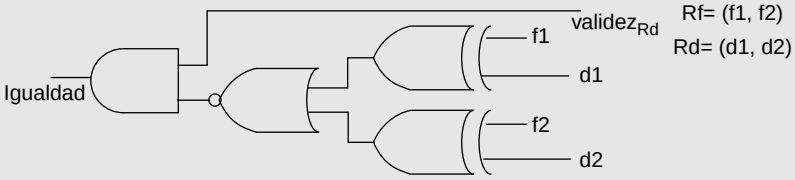
Los identificadores de los registros fuente de la instrucción son las señales Ra_{DL} y Rb_{DL} *y las señales de validación de los identificadores son respectivamente* $validez_{Ra}$ *y* $validez_{Rb}$.

En los registros de desacoplo DL_ALU y ALU_MEM las señales de los identificadores de registro destino de la instrucción que utiliza la etapa se denominan Rd_{DL_ALU} , Rd_{ALU_MEM} , *donde el subíndice indica el registro de desacoplo y las señales de validación se denominan* $validez_{RdDL\_ALU}$ *y* $validez_{RdALU\_MEM}$ *respectivamente.*

Respuesta

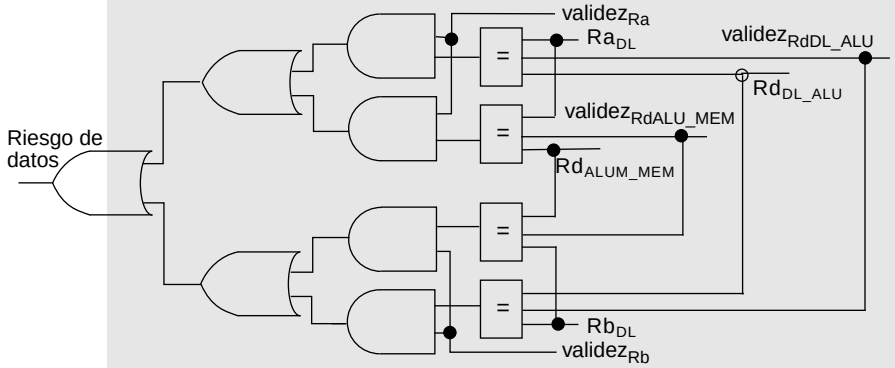
El elemento base del circuito de detección de riesgos de datos debidos a registros es el comparador. Mediante comparadores se determina la posible igualdad de los identificadores de los registros fuente de la instrucción, que está en la etapa D/L, y los identificadores de los registros destino de instrucciones más viejas.

El circuito combinacional para comparar la igualdad de dos vectores de 2 bits, teniendo en cuenta la señal de validez del registro destino, es el siguiente.



Cada identificador de registro fuente de la instrucción en la etapa D/L debe compararse con el identificador de los registro destino de las instrucciones que están en las etapas ALU y M. Si la salida de alguno de los comparadores indica igualdad existe un riesgo. Las comparaciones se validan con las señales de validación de los

identificadores de registro fuente. En la siguiente figura los comparadores se identifican con un rectángulo y el símbolo de igualdad en su interior.



En la parte superior de la figura se comparan el identificador de registro Ra_{DL} con los identificadores de registro destino en las etapas ALU y MEM (Rd_{DL_ALU} y Rd_{ALU_MEM}). En la parte inferior de la figura se compara el identificador de registro Rb_{DL} con los mismos identificadores de registro destino.

La señal Riesgo de datos controla los multiplexores MUXCPBloq, MUXBUSBloq y MUXDLNop.

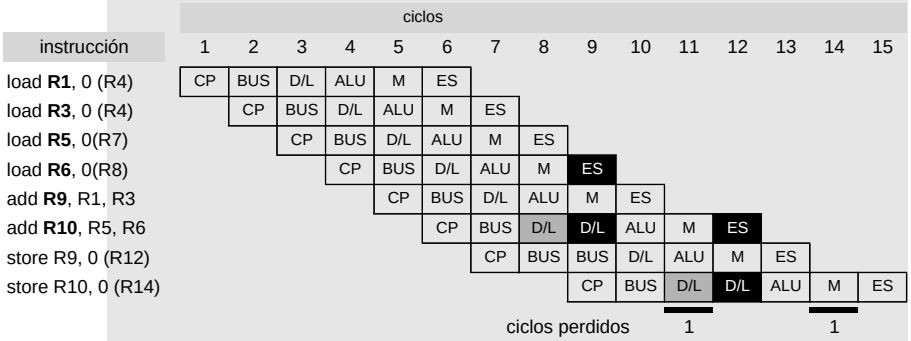
Ejercicio

Muestre en un diagrama temporal la interpretación de la secuencia de instrucciones que se muestra en la columna de la izquierda. Identifique en el diagrama temporal los ciclos perdidos.

Calcule el CPI de esta secuencia de instrucciones suponiendo que se ha extraído de una secuencia de mayor tamaño y la última instrucción antes de la secuencia analizada finaliza un ciclo antes que la instrucción load R1, 0(R2).

Respuesta

En la siguiente figura se muestra el diagrama temporal y se identifican los ciclos perdidos cuando finaliza una instrucción nop inyectada por el hardware. La instrucción nop no se muestra explícitamente ya que se utiliza el diagrama simplificado.



En el ciclo 8 se detecta un riesgo de datos debido al registro R6 y en el ciclo 11 el riesgo de datos es debido al registro R10.

Si no existen riesgos de datos finaliza una instrucción en cada ciclo y $CPI|_{ideal}$ =1. El CPI de la secuencia de instrucciones es el $CPI|_{ideal}$ más los ciclos perdidos por instrucción.

$$CPI = CPI|_{ideal} + \text{ciclos perdidos por instrucción}$$

$$CPI = 1 + \frac{2}{8} = 1.25$$

Otra forma de efectuar el cálculo es contabilizar el número de ciclos y dividir por el número de instrucciones.

$$CPI = \frac{\text{número de instrucciones + ciclos perdidos}}{\text{número de instrucciones}}$$

$$CPI = \frac{8+2}{8} = 1.25$$

Riesgo de datos $D(i) \cap R(i+k) \neq \emptyset$ y $R(i) \cap R(i+k) \neq \emptyset$

En la Figura 3.39 se muestra la interpretación de una secuencia de instrucciones con dependencias de datos $D(i) \cap R(i+k) \neq \emptyset$ y $R(i) \cap R(i+k) \neq \emptyset$ debidas a registros.

Todas las instrucciones tardan el mismo tiempo en interpretarse y pasan por las mismas etapas en el mismo orden. Como caso particular, la etapa de acceso a la memoria de datos sólo representa un retardo para las instrucciones que no acceden a memoria.

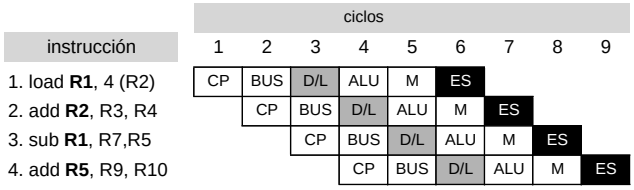


Figura 3.39 *Secuencia de instrucciones que permite observar que no se detectan riesgos* $D(i) \cap R(i+k) \neq \emptyset$ y $R(i) \cap R(i+k) \neq \emptyset$.

Todas las instrucciones leen los operandos fuente en la misma etapa (D/L) y la actualización del banco de registros se efectúa en la etapa de escritura (ES), al final del proceso de interpretación de una instrucción. Por tanto, una instrucción siempre lee antes de que escriba una instrucción posterior en orden de programa y no se produce riesgo $D(i) \cap R(i+k) \neq \emptyset$. Igualmente una instrucción siempre escribe antes que escriba una instrucción posterior en orden de programa y no se produce riesgo $R(i) \cap R(i+k) \neq \emptyset$.

Por otro lado, el control de riesgos diseñado hasta ahora garantiza que una instrucción no puede iniciar su fase de ejecución mientras no la haya iniciado la instrucción anterior en orden de programa. Esto es, las instrucciones inician la ejecución en orden de programa.

En resumen, el diseño efectuado de la segmentación y el control de riesgos utilizado no produce riesgos $D(i) \cap R(i+k) \neq \emptyset$ y $R(i) \cap R(i+k) \neq \emptyset$.

RIESGOS DE DATOS DEBIDOS A MEMORIA DE DATOS

La segmentación del camino de datos puede modificar el orden de programa de las lecturas y escrituras sobre una posición de almacenamiento en la memoria de datos. Entonces, es necesario analizar los posibles riesgos de datos para que el control actúe y garantice que se respeta el orden especificado por el programador.

En este contexto, el rango o el dominio de una instrucción es la posición de memoria de datos accedida mediante una dirección efectiva. En el caso de una instrucción load la posición de memoria accedida pertenece al dominio de la instrucción. En cambio, la posición de memoria accedida en una instrucción store pertenece al rango de la instrucción. Por tanto, lo que se compara son direcciones efectivas de acceso a la memoria de datos.

En la Figura 3.40 se muestra la ejecución de una secuencia de instrucciones de acceso a memoria. El acceso a memoria siempre se efectúa en el mismo ciclo de interpretación en cualquier instrucción que accede a memoria. Por tanto

- una instrucción store siempre escribe antes de que lea una instrucción load más joven.
- una instrucción load siempre lee antes de que escriba una instrucción store más joven.
- una instrucción store siempre escribe antes de que escriba una instrucción store más joven.

Por otro lado, el control de riesgos diseñado garantiza que las instrucciones inician la ejecución en orden de programa.

Entonces, el diseño efectuado de la segmentación y el control de riesgos utilizado no produce riesgos al acceder a la memoria de datos.

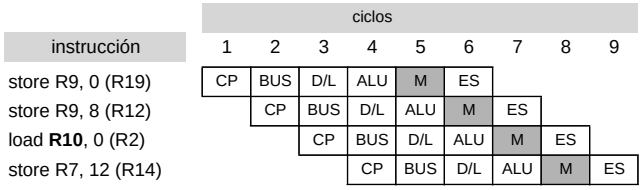


Figura 3.40 *Secuencia de instrucciones para observar que no se detectan riesgos de datos debidos a posiciones de almacenamiento en la memoria de datos.*

LÓGICA DE INTERBLOQUEOS DE LA SEGMENTACIÓN

En este apartado se presenta un esquema de la lógica necesaria para controlar los riesgos.

La diferencia de semánticas entre el camino de datos y el lenguaje máquina requiere adecuar la semántica del primero al segundo. La situación en la cual se produciría una diferencia de semántica se denomina situación de riesgo y requiere un control.

El control se denomina lógica de interbloqueos de la segmentación y gestiona los riesgos emulando un funcionamiento secuencial.

La lógica de interbloqueos garantiza que una instrucción no puede iniciar su fase de ejecución mientras no la haya iniciado la instrucción anterior en orden de programa. Esto es, las instrucciones inician la ejecución en orden de programa. Teniendo en cuenta esta característica, se dice que el procesador es en orden o que ejecuta las instrucciones en orden de programa. Adicionalmente, como todas las instrucciones tienen la misma latencia de interpretación se dice que finalizan en orden.

En la Figura 3.41 se muestran los bucles hardware que producen riesgos de datos y de secuenciamiento.

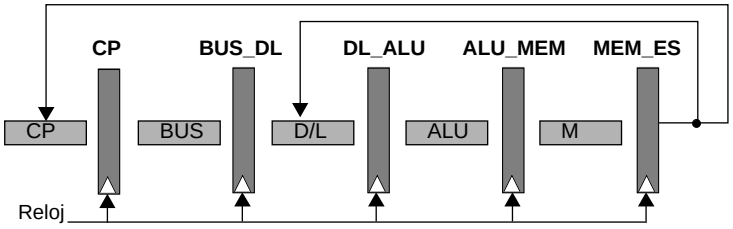


Figura 3.41 *Bucles hardware que producen riesgos de datos y riesgos de secuenciamiento.*

La detección de riesgos se efectúa en la primera etapa donde se tiene información sobre la instrucción que se quiere interpretar, siendo ésta la etapa D/L. Para detectarlos se utiliza información de la instrucción en la etapa D/L e información de las instrucciones más viejas.

En la Figura 3.42 se muestra un esquema del camino de datos utilizado para controlar los riesgos de datos y secuenciamiento.

En un funcionamiento sin riesgos, en cada ciclo de reloj se transfiere la información procesada en una etapa a la etapa o etapas conectadas en su salida. En cambio, en una situación de riesgo se bloquea la interpretación de instrucciones en algunas etapas. Este bloqueo se dice que crea una burbuja en el flujo de instrucciones que se ejecutan y por tanto ocasiona una pérdida de rendimiento.

Conceptualmente una acción de bloqueo determina que no se transfiere información a la siguiente etapa y ésta no tiene que procesar información. Sin embargo, en cada ciclo se actualizan los registros de desacoplo y se procesa la información que hay en sus entradas. Este hecho no es importante en etapas donde sólo hay lógica combinacional (ej. ALU), pero no es permisible en etapas donde se actualiza el estado del procesador (ej. MEM, ES). Para solventar esta situación, mediante hardware, se inyecta en el flujo de instrucciones una instrucción que no actualice el estado del procesador (ej. instrucción nop).

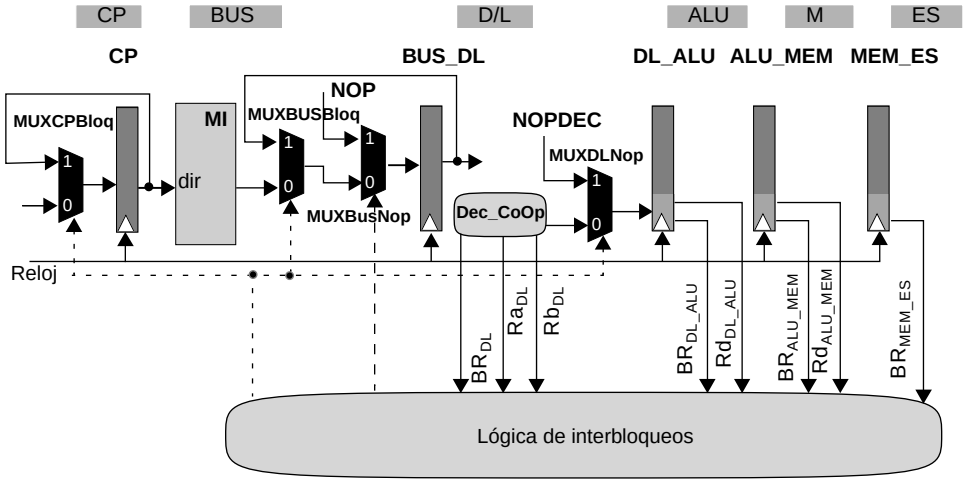


Figura 3.42 *Modificación del camino de datos para gestionar todos los riesgos. No se muestran las señales de validación de la información suministrada a la lógica de interbloqueos.*

En resumen, las acciones que se efectúan para gestionar los riesgos, utilizando los registros de desacoplo y la lógica adicional (multiplexores) que se ha añadido, se muestran en la Figura 3.43. Estas acciones ya han sido descritas en el Capítulo 2 cuando se

describía el control de riesgos estructurales y se ha detallado la circuitería que arropa a los registros de desacoplo para efectuar las acciones descritas. Notemos que la lógica asociada a los registros de desacoplo de la Figura 3.42 se corresponde con la lógica de la Figura 2.47, la cual se muestra en la parte izquierda de la Figura 3.44.

Acciones	Funcionalidad
Normal	transferir la entrada a la salida
Bloqueo	retener la instrucción en la etapa
Inyección (Burbuja)	Inyectar una instrucción nop para indicar a las etapas que los datos de entrada son inválidos

Figura 3.43 *Acciones que se efectúan utilizando los registros de desacoplo y la lógica asociada para gestionar los riesgos.*

A partir de ahora para simplificar los dibujos utilizaremos el gráfico que se muestra en la parte derecha de la Figura 3.44 en lugar del registro de desacoplo y los multiplexores.

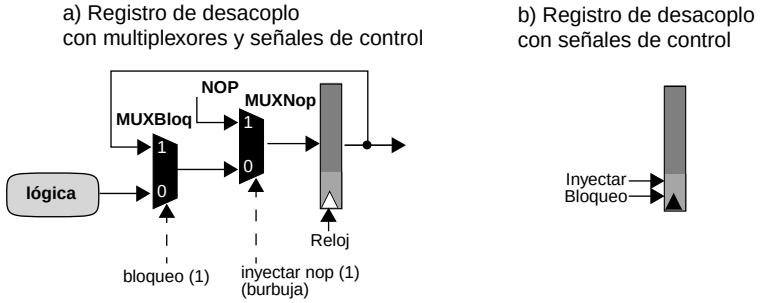


Figura 3.44 *Gráfico para representar un registro de desacoplo y la lógica asociada que se utiliza para implementar los tres modos que permiten controlar los riesgos.*

En el nuevo gráfico se suprimen los multiplexores y el conexionado asociado. Para diferenciarlos de un registro de desacoplo que sólo tiene el modo normal de funcionamiento la marca de la señal de reloj se tinta de negro. En el nuevo gráfico, sólo se muestran las señales de control de los multiplexores que se utilizan para una acción de bloqueo o una acción de inyección. Tampoco se distinguirá explícitamente si la instrucción nop que se inyecta está o no decodificada ni la señal de reloj en el registro de

desacoplo. En todos los casos en que se utilice el nuevo gráfico se supondrá que el conexionado de los multiplexores es el representado en la parte izquierda de la Figura 3.44, si éste no fuera el caso se indicará explícitamente.

Interacciones entre riesgos de datos y secuenciamiento

En las secciones anteriores se han descrito los controles de riesgos de secuenciamiento y de datos de forma independiente, sin considerar posibles interacciones para simplificar la descripción. Seguidamente se analizan estas interacciones.

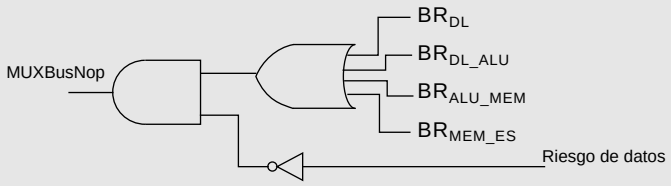
Una de las interacciones es que una instrucción de secuenciamiento condicional puede detectar un riesgo de datos en la etapa D/L. En estas condiciones, mientras no se resuelve el riesgo de datos no se toman las acciones que determinaría la instrucción de secuenciamiento.

Ejercicio

Diseñe el circuito combinacional de control de riesgos de secuenciamiento que tenga en cuenta los riesgos de datos en la instrucción de secuenciamiento. La señal de salida del circuito de control debe controlar el multiplexor MUXBusNop de la Figura 3.31. Suponga que disponemos de la señal Riesgo de datos que está activada cuando se ha detectado un riesgo de datos.

Respuesta

En primer lugar hay que gestionar el posible riesgos de datos debido al dato fuente que se lee del banco de registros. Una vez no existe riesgo de datos se gestiona el riesgo de secuenciamiento que produce la instrucción de secuenciamiento en la fase de ejecución.



Ahora analizaremos si, cuando se está gestionando un riesgo de secuenciamiento, es posible detectar un riesgo de datos en una instrucción más joven.

En la Figura 3.45 se vuelve a mostrar la Figura 3.29. La primera instrucción es de secuenciamiento. La gestión del riesgo determina que la siguiente instrucción se empiece a interpretar cuando finaliza la instrucción de secuenciamiento. Por tanto, cuando se está gestionando un riesgo de secuenciamiento, no es posible detectar un riesgo de datos en una instrucción más joven.

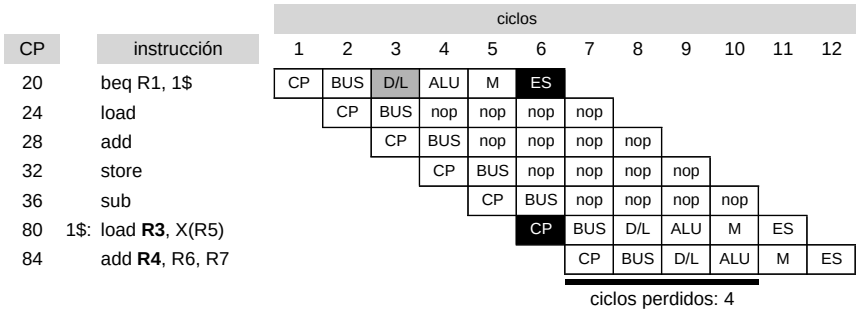


Figura 3.45 *Diagrama temporal que muestra que mientras se gestiona un riesgo de secuenciamiento no se puede detectar un riesgo de datos en un instrucción más joven.*

Ejercicio

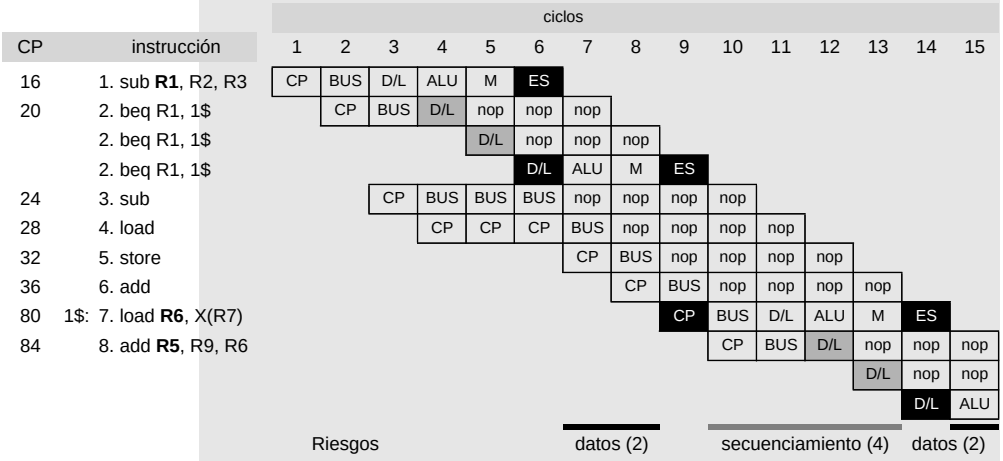
Muestre en un diagrama temporal la interpretación de la siguiente secuencia de instrucciones.

dirección	instrucción
16	sub **R1**, R2, R3
20	beq R1, 1$
24	sub
28	load
32	store
36	add
. . .	. . .
80	1$: load **R6**, X(R7)
84	add **R5**, R9, R6

Suponga que al evaluar la condición en la instrucción de secuenciamiento condicional se determina que se modifica el secuenciamiento implícito.

Respuesta

En la siguiente figura se muestra el diagrama temporal.



Entre la 1ª instrucción y 2ª la instrucción se detecta un riesgo de datos $R(i) \cap D(i+k) \neq 0$. Mientras no se resuelve este riesgo no se toman las acciones que determinaría la instrucción “beq” en la etapa D/L si prosiguiera la interpretación.

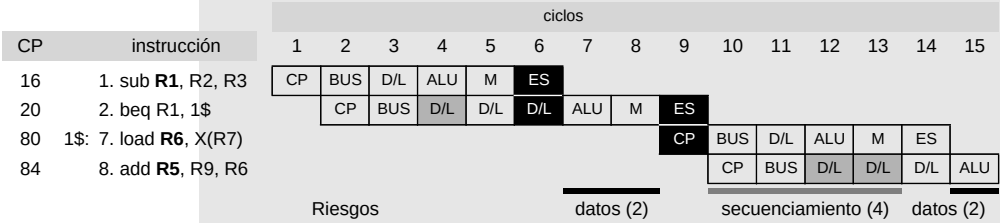
En el ciclo 5 se detecta la condición de riesgo de secuenciamiento establecida por la 2ª instrucción. Esta instrucción bloquea la búsqueda de instrucciones hasta el ciclo 8, ya que es en este ciclo cuando se conoce la dirección de la siguiente instrucción que debe interpretarse.

En el ciclo 12, después de haber gestionado el riesgo de secuenciamiento se detecta un riesgo de datos.

Ciclos perdidos por riesgo de datos: 2.

Ciclos perdidos por riesgo de secuenciamiento: 4.

En la siguiente figura se muestra el diagrama temporal abreviado.



Esquema de puertas lógicas

En la Figura 3.46 se muestra la lógica de interbloqueos, la cual es centralizada. En ella se diferencia la parte correspondiente a la detección de situación de riesgo y la parte correspondiente a la actuación.

En la parte superior se muestra la lógica utilizada para la detección de riesgos de datos y en la parte inferior la lógica utilizada para la detección de riesgos de secuenciamiento.

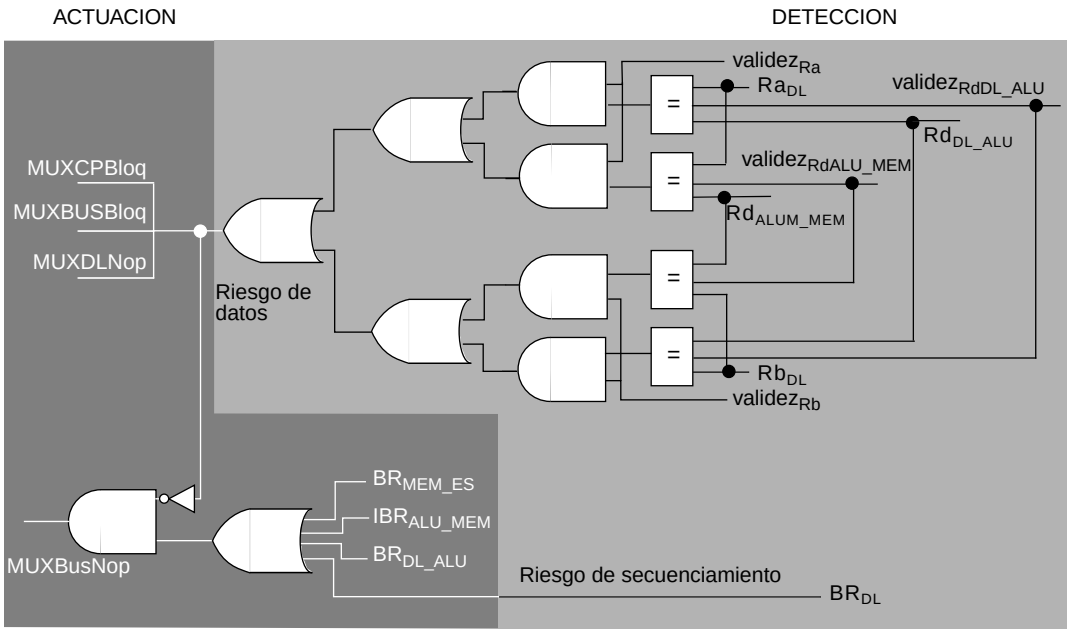


Figura 3.46 *Lógica de interbloqueos.*

EJEMPLOS

En este apartado se utilizan varios ejemplos de código para analizar el rendimiento de un procesador segmentado. El primero efectúa la ordenación lexicográfica de los elementos almacenados en un vector. El segundo es un código que inserta un elemento en una lista ordenada.

En las secuencias de instrucciones que se interpretan se utilizan las instrucciones cmple y cmpeq que son de tipo ENT.

- **cmple rc**, ra, rb: comparación de menor o igual.

  cmple — if ($ra^v \leq rb^v$) then $rc^v = 1$ else $rc^v = 0$

- **cmpeq rc**, ra, rb: comparación de igualdad.

  cmpeq — if ($ra^v = rb^v$) then $rc^v = 1$ else $rc^v = 0$

El contenido del registro ra se compara con el contenido del registro rb, interpretando ambos contenidos como números enteros. Si la relación especificada en la instrucción se cumple, se escribe el valor 1 en el registro rc, en caso contrario se escribe el valor 0 en el registro rc.

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.

```
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$:  load r8, (r7)          load A(I)
     load r9, 4 (r7)        load A(I+1)
     cmple r10, r8, r9      A(I) ≤ A(I+1)
     bne r10, 2$            se intercambia si A(I) > A(I+1)
     store r9, (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 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.

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.

En todas la preguntas supondremos que al interpretar la instrucción bne r11,1$ se cumple la condición.

El código se ejecuta en un procesador serie y en el procesador segmentado con control de riesgos descrito en este capítulo. En el procesador serie supondremos que la latencia de todas la instrucciones son 6 ciclos y la frecuencia de funcionamiento es 200 Mhz. La frecuencia de funcionamiento del procesador segmentado son 0.5 GHz.

Pregunta 1: *En un procesador serie calcule los ciclos de ejecución de una iteración del bucle en los dos posibles supuestos al interpretar la instrucción bne r10,2$.*

Respuesta: Los ciclos de ejecución de una iteración son el número de instrucciones por los ciclos de interpretación.

	condición (bne r10, 2$)	
	salto no tomado (no se cumple)	salto tomado (se cumple)
T_{serie}	11 x 6 = 66 ciclos	8 x 6 = 48 ciclos

Pregunta 2: *En el procesador segmentado, calcule los ciclos de ejecución por iteración, en los dos posibles supuestos al interpretar la instrucción bne r10,2$. Así mismo, indique los ciclos perdidos por tipo de riesgo y calcule el CPI por iteración, en cada uno de los dos casos.*

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

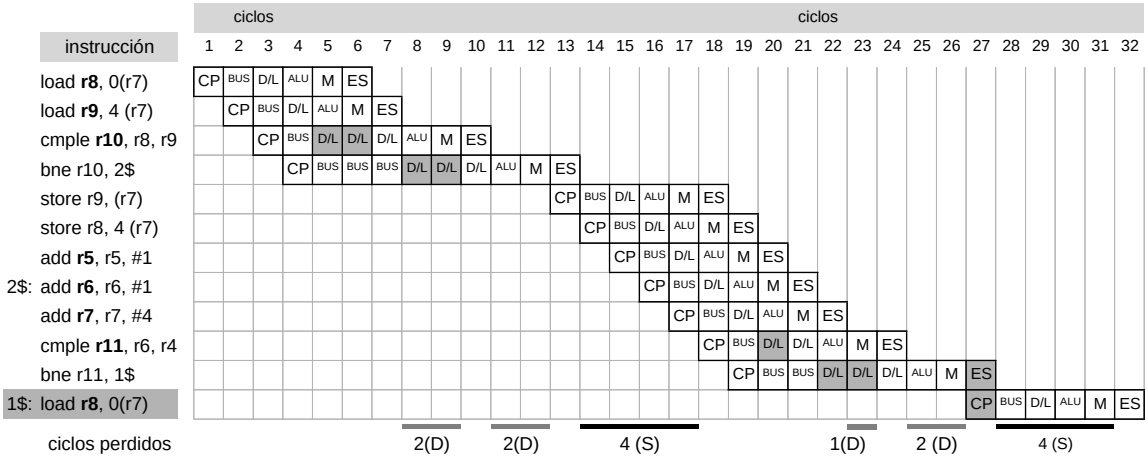
A) Salto no tomado (NT): No se cumple la condición de salto en la instrucción bne r10, 2$.

En la siguiente figura se muestra mediante un diagrama temporal la interpretación segmentada de una iteración del código descrito y la 1ª instrucción de la siguiente iteración.

En el ciclo 5 se detectan dos riesgos de datos debidos a los registros r8 y r9. El riesgo debido al registro r8 desaparece en el ciclo 6 y el debido al registro r9 desaparece en el ciclo 7.

En el ciclo 8 se detecta un riesgo de datos debido al registro r10. El riesgo desaparece en el ciclo 10 y entonces se detecta un riesgo de secuenciamiento. Este último riesgo hace que se pierdan 4 ciclos.

En el ciclo 20 se detecta un riesgo de datos debido al registro r6 y se pierde 1 ciclo. En los ciclos 22 y 23 se detecta un riesgo de datos debido al registro r11 y en el ciclo 24 se detecta un riesgo de secuenciamiento. Este último riesgo hace que se pierdan 4 ciclos.



Los ciclos perdidos son

	Riesgos	
	datos	secuenciamiento
ciclos perdidos$\|_{NT}$	7	8

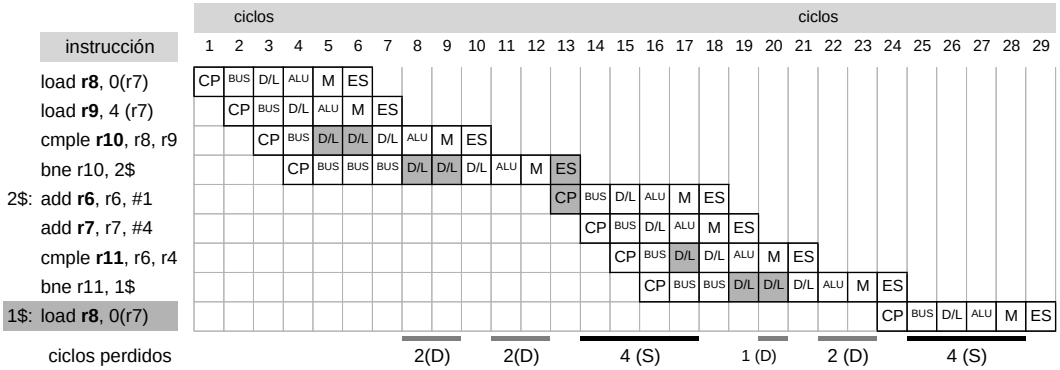
Los ciclos de ejecución en un camino de datos segmentado y el CPI cuando no se toma el salto son:

En una iteración	salto no tomado
$T_{seg}\|_{NT}$ = número de instrucciones + ciclos perdidos	11 + 15 = 26 ciclos
$CPI_{NT} = T_{seg}\|_{NT}$ / (número de instrucciones)	26 / 11 = 2.36

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 se modifica el flujo de interpretación a partir de la instrucción de secuenciamiento ble r10, 2$.

Analizando a partir de la instrucción bne r10, 2$ tenemos que se detecta un riesgo de datos en el ciclo 17 debido al registro r6. También se detecta un riesgo de datos en el ciclo 19 que perdura durante el siguiente ciclo debido al registro r11. Finalmente, en el ciclo 21 se detecta un riesgo de secuenciamiento cuya penalización es de 4 ciclos.



Los ciclos perdidos son

	Riesgos	
	datos	secuenciamiento
ciclos perdidos$\|_T$	7	8

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	8 + 15 = 23 ciclos
$CPI_T = T_{seg}\|_T$ / (número de instrucciones)	23 / 8 = 2.88

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

Pregunta 3: *Determine el CPI medio en el procesador segmentado.*

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

$$CPI = \frac{0.3 \times T_{seg}\big|_{NT} + 0.7 \times T_{seg}\big|_{T}}{0.3 \times N_{NT} + 0.7 \times N_{T}} = \frac{0.3 \times 26 + 0.7 \times 23}{0.3 \times 11 + 0.7 \times 8} = \frac{7.8 + 16.1}{3.3 + 5.6} = 2.69$$

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.

$$CPI = 0.3 \times \frac{N_{NT}}{0.3 \times N_{NT} + 0.7 \times N_T} \times CPI_{NT} + 0.7 \times \frac{8}{0.3 \times 11 + 0.7 \times 8} \times 2.875$$

Sustituyendo los restantes valores

$$CPI = 0.88 + 1.81 = 2.69$$

Pregunta 4: *Calcule la ganancia de la interpretación segmentada respecto de una interpretación serie.*

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

$$Ganancia = \frac{T_{serie}}{T_{seg}} = \frac{0.3 \times 66 + 0.7 \times 48}{0.3 \times 26 + 0.7 \times 23} \times \frac{0.5}{0.2} = \frac{19.8 + 33.6}{7.8 + 16.1} \times 2.5 = 5.59$$

Por tanto, el camino de datos segmentado es un 459% más rápido que el serie.

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 r9 es el contenido de la variable q. 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$                  break
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.

El código se ejecuta en un procesador serie y en el procesador segmentado con control de riesgos descrito en este capítulo. En el procesador serie suponga que la latencia de todas la instrucciones son 6 ciclos y que la frecuencia de operación es la misma que en el procesador segmentado.

Pregunta 1: *En un procesador serie, calcule los ciclos de ejecución de una iteración del bucle, en los dos posibles supuestos al interpretar la instrucción bne r5, 1$.*

Respuesta: Los ciclos de ejecución de una iteración son el número de instrucciones por los ciclos de interpretación.

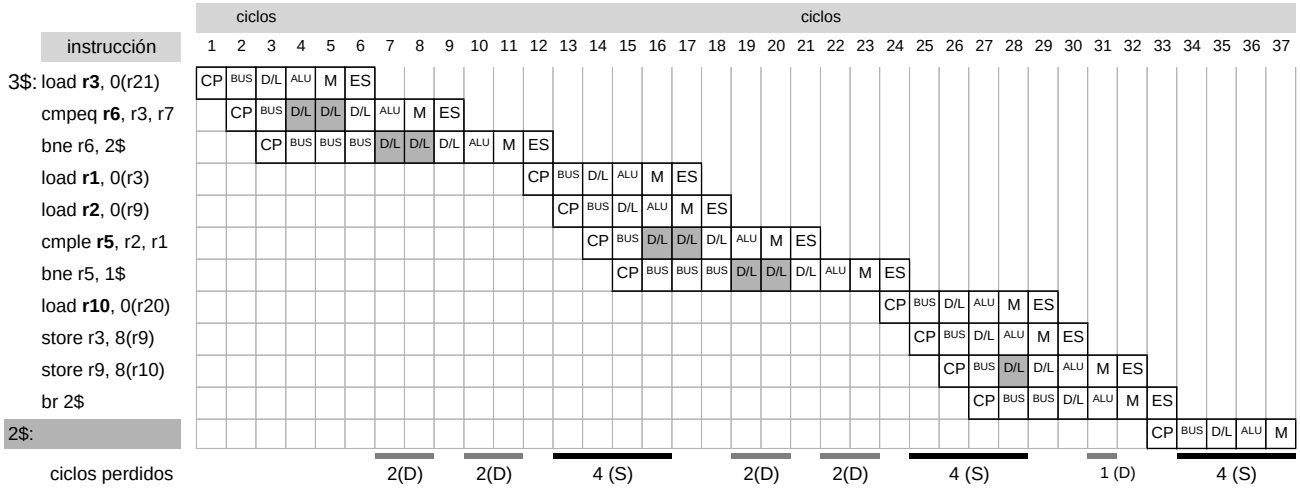
	condición (bne r10, 2$)	
	salto no tomado (no se cumple)	salto tomado (se cumple)
T_{serie}	11 x 6 = 66 ciclos	11 x 6 = 66 ciclos

Pregunta 2: *En el procesador segmentado, 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: El elemento se inserta en la lista cuando al ejecutar la instrucción bne r5,1$ no se cumple la condición.

En la siguiente figura se muestra, mediante un diagrama, temporal la interpretación segmentada de una iteración del código descrito.

En el ciclo 4 se detecta un riesgo de datos debido al registro r3, que perdura durante dos ciclos. En el ciclo 7 se detecta otro riesgo de datos que desaparece en el ciclo 9. En el ciclo 9 se detecta un riesgo de secuenciamiento. Este último riesgo hace que se pierdan 4 ciclos.



En el ciclo 16 se detecta un riesgo de datos debido al registro r2 y se pierden 2 ciclos. En el ciclo 19 el riesgo de datos es debido al registro r5 y se pierden 2 ciclos. En el ciclo 21 se detecta un riesgo de secuenciamiento y hace que se pierdan 4 ciclos.

En el ciclo 28 se detecta un riesgo de datos debido al registro r10 y se pierde 1 ciclo. En el ciclo 30 se detecta un riesgo de secuenciamiento y se pierden 4 ciclos.

Los ciclos perdidos son

	Riesgos	
	datos	secuenciamiento
ciclos perdidos$\|_{NT}$	9	12

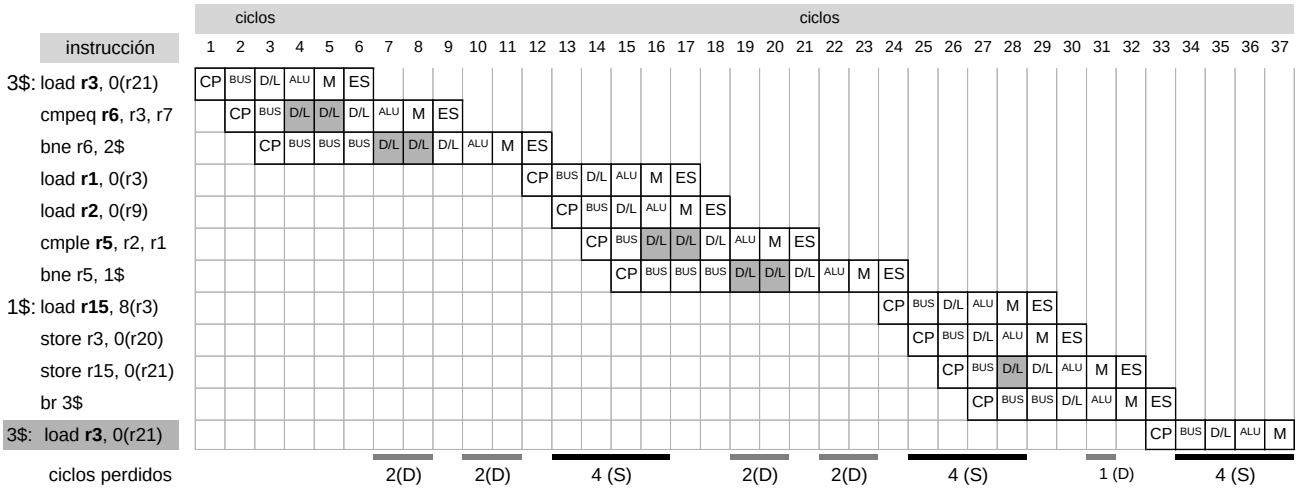
Los ciclos de ejecución en un camino de datos segmentado y el CPI cuando no se toma el salto son:

En una iteración	salto no tomado
$T_{seg}\|_{NT}$ = número de instrucciones + ciclos perdidos	11 + 21 = 32 ciclos
$CPI_{NT} = T_{seg}\|_{NT}$ / (número de instrucciones)	32 / 11 = 2.91

Pregunta 3: *En el procesador segmentado, 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: En la siguiente figura se muestra, mediante un diagrama temporal, la interpretación segmentada de una iteración del código descrito.

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$.



Analizando a partir de la instrucción bne r5, 1$ tenemos que se detecta un riesgo de datos en el ciclo 28 debido al registro r15. En el ciclo 30 se detecta un riesgo de secuenciamiento cuya penalización es de 4 ciclos.

Los ciclos perdidos son

	Riesgos	
	datos	secuenciamiento
ciclos perdidos$\|_T$	9	12

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 + 21 = 32 ciclos
$CPI_T = T_{seg}\|_T$ / (número de instrucciones)	32 / 11 = 2.91

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 cola.

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

Respuesta: Como el CPi es el mismo en los dos casos, el CPI medio es independiente de la ponderación de cada caso. Por tanto, el CPI es 2.91.

Si efectuamos el cálculo podemos utilizar la expresión que se describe seguidamente. 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 inserta el elemento y el caso en que no se inserta 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,

$$CPI = \frac{80 \times 0.4 \times 11}{363} \times 2.91 + \frac{11}{363} \times 2.91 = 2.91$$

Pregunta 5: *Calcule la ganancia de la interpretación segmentada respecto de una interpretación serie.*

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 ciclos por instrucción.

$$Ganancia = \frac{CPI_{serie}}{CPI_{seg}} = \frac{6}{2.91} = 2.06$$

Por tanto, el camino de datos segmentado es 106% más rápido que el serie.

EJERCICIOS

Ejercicio 3.1

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.

Pregunta 1: *En un procesador que interpreta las instrucciones de forma serie, calcule los ciclos de ejecución de una iteración del bucle. Para ello suponga que la latencia de todas las instrucciones son 6 ciclos.*

Pregunta 2: *En el procesador segmentado con control de riesgos descrito en este capítulo calcule los ciclos de ejecución por iteración. Así mismo, indique los ciclos perdidos por tipo de riesgo y calcule el CPI.*

Pregunta 3: *Suponga que las frecuencias de funcionamiento del procesador serie y segmentado son iguales. Calcule la ganancia de una interpretación segmentada respecto de una interpretación serie al ejecutar una iteración del bucle.*

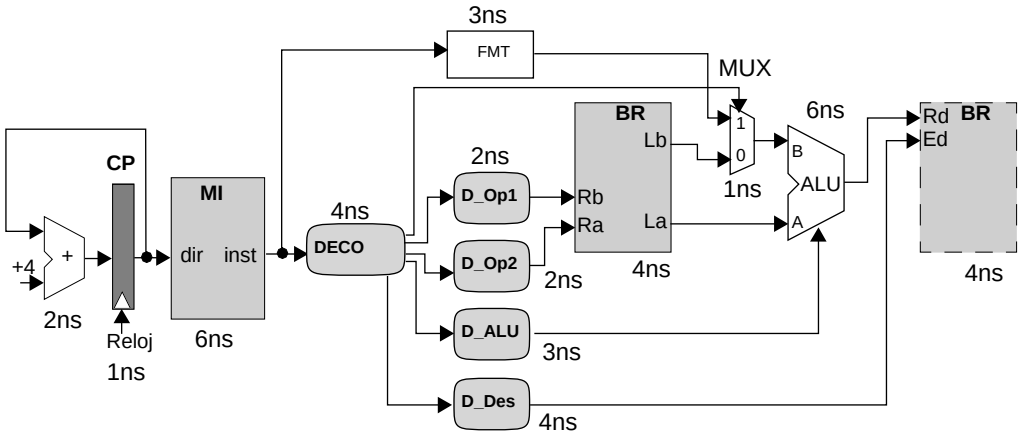
Una nueva versión del procesador segmentado tiene una frecuencia de funcionamiento 1.5X mayor. En estas condiciones, es necesario todo el ciclo de reloj para escribir o leer un registro del banco de registros.

Pregunta 4: *En esta versión del procesador, calcule los ciclos de ejecución por iteración. Así mismo, indique los ciclos perdidos por tipo de riesgo y calcule el CPI.*

Pregunta 5: *Calcule la ganancia de la nueva versión del procesador segmentado respecto de la versión previa.*

Ejercicio 3.2

En el siguiente dibujo se muestra el camino de datos de un procesador que únicamente interpreta instrucciones que utilizan la ALU.



Para simplificar el análisis supondremos que tanto la memoria de instruciones (MI) como el banco de registros son circuitos combinacionales en lugar de secuenciales. Así mismo, para facilitar la observación del camino de datos y su análisis, la escritura en el banco de registros se representa mediante un símbolo con el contorno a trazos en la parte derecha del camino de datos. El banco de registros permite efectuar concurrentemente 2 operaciones de lectura y 1 operación de escritura. Esto es, tiene 2 caminos de lectura y 1 camino de escritura.

Pregunta 1: *Represente en un diagrama temporal, con tantas filas como dispositivos lógicos, los retardos de los dispositivos lógicos (CP, +, MI, DECO, FMT, D_Op1, D_Op2, D_ALU, D_Des, BR_lec,*

MUX, ALU y BR_esc). El retardo de un dispositivo lógico se muestra cuando todas las señales de entrada son estables (peor caso). Determine el tiempo de ciclo del procesador descrito.

El diseño previo se quiere segmentar. Ahora bien, los bloques lógicos no se pueden dividir en subbloques.

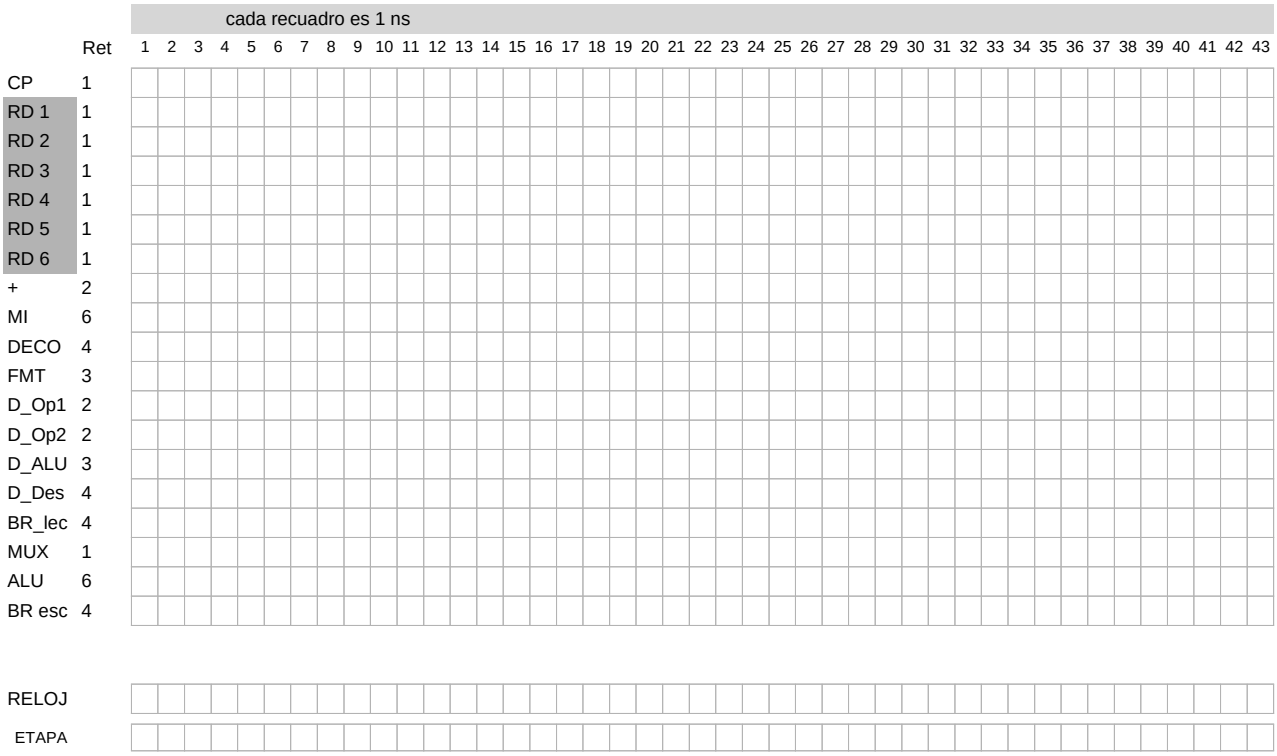
Algunos bloques se pueden ubicar indistintamente en varias etapas sin afectar al tiempo de ciclo. Si éste es el caso hay que ubicar el bloque en la etapa más cercana al inicio de la segmentación.

Los registros de desacoplo que se utilizan en el diseño segmentado tienen un tiempo de propagación de la señal de 1 ns.

Denomine RD1, RD2, etc a los registros de desacoplo entre etapas. Recuerde que la señal de reloj puede tener intervalos de tiempo distintos a nivel cero y a nivel uno. Al mostrar la señal de reloj en las siguientes preguntas siga suponiendo que la memoria de instrucciones y el banco de registros son circuitos combinacionales.

Pregunta 2: *Cuando se tiene en cuenta la interpretación de una instrucción aislada, proponga una implementación segmentada del procesador en la que el tiempo de ciclo sea el menor posible. Esto es, máxima productividad cuando se interpreta una secuencia de instrucciones independientes. Para ello, represente en un diagrama temporal los retardos de los dispositivos lógicos utilizados en cada ciclo (CP, +, MI, DECO, FMT, D_Op1, D_Op2,*

D_ALU, D_Des, BR_lec, MUX, ALU, BR_esc y los registros de desacoplo utilizados). Muestre la señal de reloj e indique el tiempo de ciclo.



Pregunta 3: *Muestre en el camino de datos, para cada flujo de información entre dispositivos lógicos, los registros de desacoplo que hay que añadir cuando se segmenta el camino de datos.*

Pregunta 4: *Indique el número de ciclos que deben transcurrir entre una instrucción que almacena un valor en un registro (productora) y una instrucción (consumidora) que quiere utilizar el valor calculado por la instrucción productora, para que no se produzca un riesgo de datos. Esto es, la latencia productor uso.*

Ejercicio 3.3

En un procesador segmentado con 7 etapas (CP, BUS, D/L, ALU, ET, DT, ES), el proceso de interpretación de instrucciones se ha segmentado de la forma que se muestra en la siguiente figura.

	1	2	3	4	5	6
load	CP	BUS	D/L	ALU	ET/DT	ES
store	CP	BUS	D/L	ALU	ET	DT
ENT	CP	BUS	D/L	ALU		ES

La única diferencia con el procesador segmentado lineal descrito en el capítulo 3 es la segmentación del acceso a memoria de datos. Los posibles riesgos de datos debidos a registros y riesgos de secuenciamiento se gestionan de la misma forma que en el capítulo 3, dando lugar a que el inicio de la fase de ejecución de las instrucciones sea en orden de programa.

El procesador dispone en el primer nivel de la jerarquía de memoria de una cache para instrucciones y una cache para datos. Nosotros supondremos que un acceso a cualquiera de las dos cache de primer nivel siempre es un acierto.

Tanto el acceso al campo etiqueta como al campo dato de un contenedor de cache requiere un ciclo.

En una instrucción load los campos etiqueta y dato de un contenedor se leen en paralelo (ciclo 5) y en el siguiente ciclo el dato leído se escribe en el banco de registros (ciclo 6).

En el caso de una instrucción store es necesario en primer lugar acceder al campo etiqueta para comprobar si el dato está almacenado en cache antes de actualizar el campo dato. Como es necesario un ciclo para acceder a cada campo de información de un contenedor de cache y hay que efectuarlo secuencialmente son necesarios dos ciclos (ciclos 5 y 6).

Supondremos que no existen riesgos estructurales cuando se interpretan de forma solapada varias instrucciones. En particular supondremos que la cache de datos dispone de un puerto de lectura y un puerto de escritura. Estos puertos de acceso son independientes y una acción de lectura o escritura en cualquiera de los campos del contenedor requiere todo el ciclo de la señal de reloj.

Pregunta 1: *Analice los posibles riesgos de datos al acceder a la memoria de datos.*

Pregunta 2: *Indique las acciones que deben efectuarse y el ciclo en el cual pueden efectuarse, para detectar el riesgo de datos al acceder a memoria.*

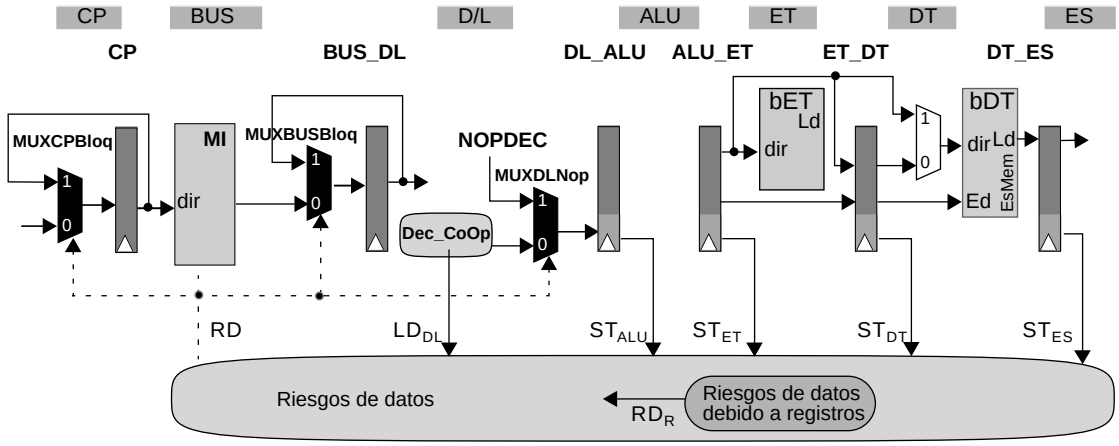
En el procesador segmentado que se utiliza, la última etapa del proceso de interpretación de una instrucción donde se puede retener una instrucción es la etapa D/L.

Pregunta 3: *Describa un mecanismo para controlar los posibles riesgos de datos al acceder a memoria. Posteriormente muestre en un diagrama temporal, utilizando la siguiente secuencia de instrucciones, el mecanismo de control de riesgo descrito. Justifique de forma detallada las retenciones o bloqueos en la etapa D/L.*

instrucciones	instrucciones
1. load **R1**, 4(R2)	4. add **R3**, R3, R4
2. store R4, 8(R7)	5. store R23, 0(R7)
3. load **R9**, 4(R1)	6. load **R9**, 0(R1)

El mecanismo utilizado puede ser conservador en la gestión de riesgos, aunque con un mecanismo de este tipo usualmente se pierden más ciclos de los necesarios.

En la siguiente figura se muestran las señales disponibles para gestionar riesgos de datos en la memoria de datos. La memoria cache de datos son los módulos bET y bDT.



La activación de la señal ST_k indica que la etapa k está ocupada por una instrucción store. La activación de la señal LD_k indica que la etapa k está ocupada por una instrucción load. La activación de

la señal RD_R, salida del módulo Riesgo de datos debido a registros, indica que se ha producido un riesgo de datos debido a registros.

Pregunta 4: *Diseñe un circuito combinacional que, con las señales descritas, controle los multiplexores MUXCPBloq, MUXBUSBloq y MUXDLNop de la figura previa, en el caso de que se detecte un riesgo de datos, ya sea debido a registros o a memoria.*

La segmentación descrita ha sido necesaria debido a que se ha querido en un nuevo diseño multiplicar por 2 la frecuencia de reloj de un diseño previo del procesador. En el diseño previo del procesador, que funciona a una frecuencia de 500 Mhz, se podía acceder secuencialmente en un ciclo al campo etiqueta y dato de un contenedor.

En el diseño previo, al ejecutar un conjunto de programas de prueba que se utilizan para evaluar el rendimiento, se ha medido un CPI igual a 2.6. Además se ha medido que hay un 15% de instrucciones store y el 60% de las veces la instrucción store está seguida inmediatamente por una instrucción load. Por otro lado, las instrucciones load que se interpretan inmediatamente después de la instrucción store cuando llegan a la etapa D/L no detectan ningún riesgo de datos debido a registros.

Pregunta 5: *Calcule el CPI medio en el nuevo procesador que funciona a una frecuencia de 1 Ghz.*

Pregunta 6: *Calcule la ganancia del nuevo procesador respecto del diseño previo.*

Del 60% de casos que una instrucción load sigue inmediatamente a una instrucción store se ha comprobado que sólo en el 30% de las veces las direcciones del store y del load son coincidentes.

Pregunta 7: *Calcule la pérdida de rendimiento por ser conservadores en la gestión del riesgo de datos en memoria respecto a un caso ideal de conocer en la etapa D/L, antes de calcular la dirección del load, si la dirección del load es igual a la dirección del store previo (oráculo).*

Ejercicio 3.4

La adecuación de la semántica del procesador segmentado con el lenguaje máquina también puede efectuarse modificando el código que debe interpretarse.

La idea es emular un funcionamiento secuencial utilizando instrucciones que no modifiquen el estado del procesador. El lenguaje máquina dispone de la instrucción nop que al interpretarse no actualiza el estado del procesador y el tamaño de todas las instrucciones es 4 bytes.

Durante los ciclos que se conoce que existe un riesgo debemos conseguir que el camino de datos esté interpretando instrucciones nop. Para ello hay que modificar el código original e insertar instrucciones nop cuando conocemos que se produciría un riesgo en una interpretación segmentada. Note que al crear un nuevo código se modifica la ubicación de las instrucciones en el espacio lógico. Esto es, la dirección de las instrucciones en el código modificado no será la misma que en el código original.

Pregunta 1: *Modifique los códigos que se muestran seguidamente para que al interpretarse en el camino de datos segmentado sin control de riesgos, descrito en el capítulo 3, se obtengan los mismos resultados que en un camino de datos que interpreta las instrucciones de forma serie.*

Código A

dirección	instrucción
0	load **R1**, X(R2)
4	add **R4**, R1, R10
8	add **R5**, R1, R11
12	add **R6**, R1, R12
16	add **R7**, R1, R13
. . .	
60	
64	

Código B

dirección	instrucción
8	beq R1, 1$
12	load **R3**, X(R5)
16	add **R4**, R9, R13
20	store R7, X(R6)
24	sub **R10**, R12, R13
. . .	. . .
60	1$: load **R17**, X(R15)
64	add **R4**, R6, R7

Pregunta 2: *Muestre en un diagrama temporal la interpretación de las instrucciones. Para ello suponga la segmentación lineal del Capítulo 3.*

Ejercicio 3.5

Un procesador segmentado lineal, con las 6 etapas típicas:

ciclos					
1	2	3	4	5	6
CP	BUS	D/L	ALU	M	ES

sin riesgos estructurales, que NO puede escribir y leer un registro en el mismo ciclo y pone el siguiente valor en el CP en el PENULTIMO ciclo cuando interpreta una instrucción de salto, ejecuta el siguiente código:

```
L :   r8     <---  M [ r7 + 0 ]
      r9     <---  M [ r7 + 4 ]
      si (r9)  saltar a T
      M[ r7 + 0 ]  <---   r9
      M [ r7 + 4 ]  <---   r8
T :   r6      <--- r6 + 1
      r7      <--- r7 + 4
      si (r6) saltar a L
```

Pregunta 1: *Presente el cronograma de una iteración, suponiendo que las dos instrucciones de salto condicional rompen la secuencia (siempre saltan).*

Pregunta 2: *Presente el cronograma de una iteración , suponiendo que la primera instrucción de salto no rompe la secuencia (no salta a T) y que la segunda instrucción de salto rompe la secuencia (siempre salta a L).*

Pregunta 3: *Calcule el número de ciclos perdidos en una iteración, indicando cuántos son por Riesgo de Datos y cuántos son por Riesgo de Secuenciamiento, en cada uno de las dos preguntas anteriores.*

Pregunta 4: *Calcule el valor del CPI medio en una iteración, sabiendo que el primer salto condicional rompe la secuencia el 20% de las veces.*

Pregunta 5: *Calcule la ganancia de este procesador sobre otro procesador con la misma frecuencia de reloj, pero que opera en serie porque no está segmentado, y que interpreta este mismo código suponiendo que el primer salto condicional rompe la secuencia el 30% de las veces.*

Ejercicio 3.6

Un procesador segmentado lineal, con las 6 etapas típicas:

ciclos					
1	2	3	4	5	6
CP	BUS	D/L	ALU	M	ES

sin riesgos estructurales, que SI puede escribir y leer un registro en el mismo ciclo y pone el siguiente valor en el CP en el ULTIMO ciclo cuando interpreta una instrucción de salto, ejecuta el siguiente código:

```
L :   r2 <---  M [ r7 + 0 ]
      si (r2)  saltar a T
      r1   : =  r2 + 0
T :   r3   : =  r3 + 1
      r7   : =  r7 + 4
      si  (r3)  saltar a L
```

Ejecuta el bucle L un número ilimitado de iteraciones sucesivas, en la mitad de las cuales la segunda instrucción acaba rompiendo la secuencia (saltando a T), y en la otra mitad no.

Pregunta 1: *Presente el cronograma de una iteración completa y el comienzo de la siguiente, para el caso en que la segunda instrucción no rompe la secuencia.*

Pregunta 2: *Exprese la cantidad de ciclos perdidos en la iteración del cronograma a causa de Riesgos de Datos (RdD) y a causa de Riesgos de Secuenciamiento (RdS), y la cantidad total de ciclos que tarda la iteración (los que hay desde inicio iteración hasta inicio iteración siguiente).*

Pregunta 3: *Sin presentar el cronograma, calcule el número de ciclos perdidos en una iteración, indicando cuántos son por RdD y cuántos son por RdS, en el caso en que la segunda instrucción rompe la secuencia.*

Pregunta 4: *Calcule el valor del CPI medio que consigue este procesador al interpretar este código.*

Pregunta 5: *Este procesador consume una potencia P = 55 vatios a una tensión V = 5 voltios, y la batería que lo alimenta tiene una carga Q = 11 x 10^6 Amperios x segundo.*
Calcule el tiempo T, en segundos, que tarda en agotarse completamente la batería.

Pregunta 6: *Calcule la velocidad en MIPS de este procesador ejecutando este código, sabiendo que ha completado 10^{14} iteraciones antes de agotar la batería.*

Pregunta 7: *Calcule la ganancia G que obtiene este procesador respecto a otro equivalente pero que es NO segmentado y que tiene tiempo de ciclo la mitad, al ejecutar este mismo código.*

Pregunta 8: *Calcule el valor de la frecuencia, en GHz, del procesador NO segmentado.*

Ejercicio 3.7

Un procesador, que interpreta las siguientes instrucciones

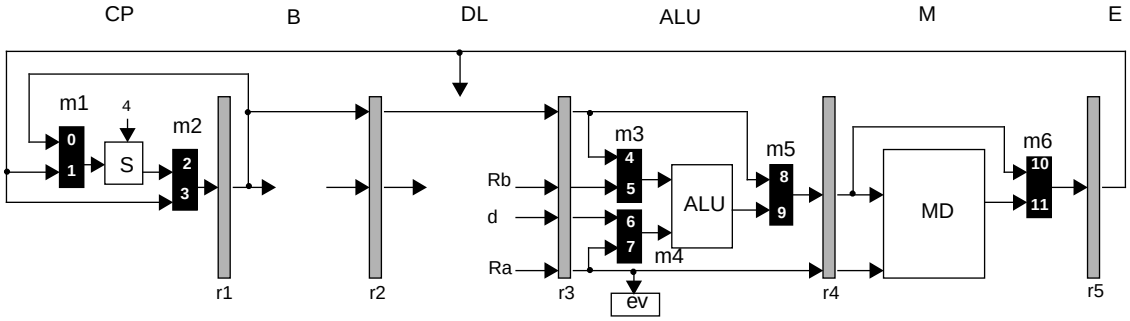
INT:	Rc = Ra op Rb
Load:	Rc = Mem[Rb + d]
Store:	Mem[Rb + d] = Ra
Br:	if cond (Ra) then PC = PC + d; else CP = CP + 4

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
ALU	operación aritmético-lógica (INT), o cálculo de la dirección efectiva (Load/store), o evaluación de la condición, cálculo de la dirección destino y selección de la dirección (Br)
M	acceso a memoria (Load, Store)
E	escritura en el banco de registros (INT, Load) y en el CP (Br)

El banco de registros permite la escritura y la lectura, en este orden, de un mismo registro en un ciclo de reloj. 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 en detalle las etapas B, DL y E). Tampoco 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 hay que seleccionar para cada tipo de instrucción en los ciclos 4, 5 y 6 del proceso de interpretación. Por ejemplo, para una instrucción Store, donde la marca x denota indistinto:*

	ciclo 4			5	6	
Instrucción	m3	m4	m5	m6	m1	m2
Store	5	6	9	x	0	2

Supongamos que el retardo de propagación (en ps) de los componentes mostrados en el camino de datos es

Componente	retardo
S:	150 (sumador)
ALU:	200 (unidad aritmético-lógica)
EV:	100 (evaluador de condiciones)
MD:	350 (memoria de datos)
mx:	100 (multiplexor)
ri:	50 (registro de desacoplo, donde r1 es el registro CP)
decodificadores y circuitos que generan señales de control: 0	

Pregunta 2: *Calcule el tiempo de etapa máximo y mínimo de CP, ALU y M.*
Considerando sólo las etapas CP, ALU y M, calcule el tiempo de ciclo de reloj.

En la etapa DL se detectan los riesgos de datos y de secuenciamiento. Cuando se detecta un riesgo de datos, la lógica de control bloquea la interpretación de las instrucciones que estan en las etapas DL, B i CP mientras perdura el riesgo. En caso de riesgo de secuenciamiento, la lógica de control descarta las instrucciones buscadas hasta que se actualiza el Contador de Programa con la dirección de la siguiente instrucción.

El procesador ejecuta el siguiente programa, que localiza el elemento máximo de una lista.

```
for (; p!=NULL; p=p->next)
{
   if (p->dat > max)
   {
      max=p->dat;
      pos=p;
   }
}
```

```
1$:  load r2, 8(r0)      ;r2 ← mem[r0+8]
     cmpgt r3, r2, r1    ;r3 ← (r2 > r1)
     beq r3, 2$          ;si (r3=0) salta a 2$
     add r1, r2, r10     ;r1 ← r2
     add r5, r0, r10     ;r5 ← r0
2$:  load r0, 0(r0)      ;r0 ← mem[r0+0]
     bne r0, 1$          ;si (r0≠0) salta a 1$

     Valores iniciales:
     r0 = dirección del primer elemento de la lista
     r1 = 0; r10 = 0
```

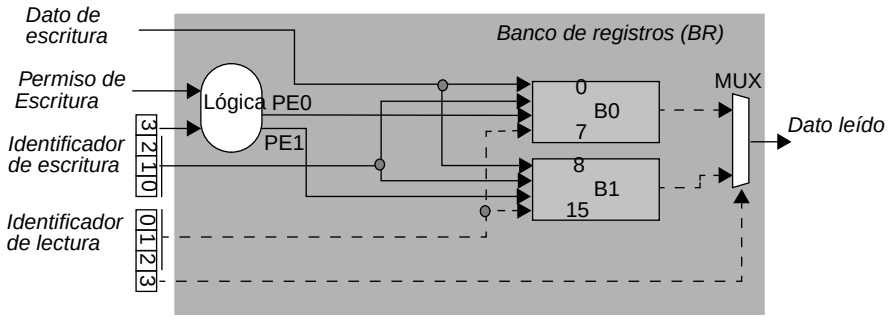
Pregunta 3: *Muestre el cronograma de ejecución de las instrucciones de una iteración entera del bucle y de la primera instrucción de la siguiente iteración, en el caso en que* `p->dat > max`*. Identifique los ciclos perdidos en el cronograma e indique cuál es el motivo.*

Pregunta 4: *Calcule el CPI medio suponiendo que la condición* `p->dat > max` *se cumple en el 10% de las iteraciones.*

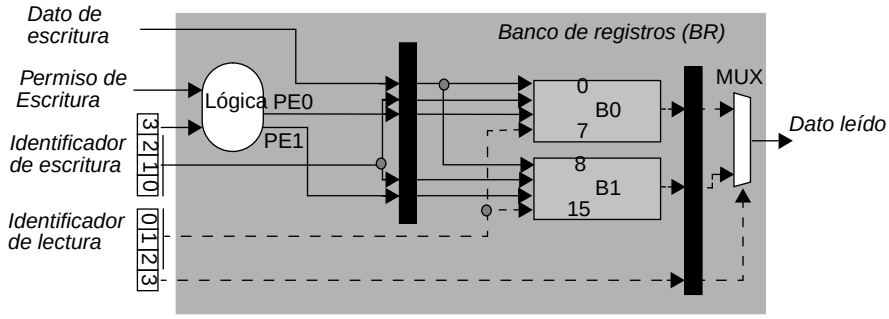
Ejercicio 3.8

El banco de registros de un procesador se ha construido utilizando subbancos de menor tamaño y lógica combinacional. En esta organización, registros con identificadores de registros consecutivos están ubicados en el mismo subbanco mientras sea posible. En la siguiente figura se muestra un esquema del banco de registros, donde se puede observar el camino de datos en una operación de escritura (líneas contínuas) y el camino de datos en una operación de lectura (líneas discontínuas).

En una operación de lectura se lee un registro de cada subbanco (B0, B1) utilizando los bits menos significativos del identificador de registro. Posteriormente se selecciona uno de los valores leidos utilizando el bit más significativo del identificador de registro (MUX). En una operación de escritura se determina en primer lugar el subbanco donde está ubicado el registro (Lógica). Para ello se utiliza el bit más significativo del identificador de registro. Posteriormente se actualiza el registro en el subbanco de registros seleccionado y el registro del subbanco se determina utilizando los bits menos significativos del identificador de registro.



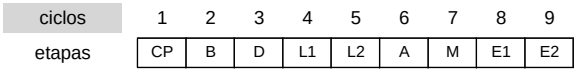
La frecuencia de funcionamiento del procesador requiere segmentar el acceso al banco de registros, tanto en operaciones de lectura como en operaciones de escritura. En la siguiente figura se muestra la segmentación en los dos casos y los recursos utilizados en cada ciclo, donde los rectángulos negros indican registros de desacoplo.



Lectura	etapas	. . .	L1	L2	. . .
	recursos	. . .	BR	MUX	. . .

Escritura	etapas	. . .	E1	E2
	recursos	. . .	Lógica	BR

En la siguiente figura se muestra la segmentación en etapas del proceso de interpretación de las instrucciones en el procesador segmentado que estamos considerando.



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

ETAPA	FUNCIONALIDAD	RECURSOS
CP	determinar el CP	sumador
B	búsqueda de la instrucción	memoria de instrucciones (MI)
D	decodificación	decodificador
A	operación aritmético-lógica	ALU
M	acceso a memoria de datos	memoria de datos (MD)

La funcionalidad de las restantes etapas ya ha sido descrita previamente de forma parcial y seguidamente describimos el resto. Supondremos que en un ciclo se puede escribir y leer, en este orden, un registro del banco de registros. También, supondremos, por ahora, que en el banco de registros se pueden efectuar, en el mismo ciclo, 2 operaciones de lectura y 1 operación de escritura.

Suponga que el procesador no utiliza ningún tipo de control para gestionar los riesgos de datos.

Pregunta 1: *Dada la siguiente secuencia de 9 instrucciones, muestre mediante un diagrama temporal las situaciones de riesgos de datos. Indíquelas en el margen derecho del diagrama temporal.*

instrucción	instrucción	instrucción
1. add **R3**, R3, R1	4. add **R7**, R3, R2	7. xor **R10**, R3, R11
2. sub **R4**, R3, R1	5. add **R8**, R3, R2	8. and **R12**, R3, R13
3. add **R5**, R3, R6	6. sub **R9**, R3, R10	9. load **R14**, 0(R15)

Después de segmentar el acceso al banco de registros, el tiempo de ciclo del procesador sigue limitado por el acceso al banco de registros. Por ello se decide restringir de forma drástica el número de accesos por ciclo. En un ciclo determinado se pueden efectuar o sólo 2 lecturas o sólo 1 escritura en el banco de registros.

Suponga una secuencia de instrucciones donde todas las instrucciones escriben en el banco de registros.

Pregunta 2: *Utilice un diagrama temporal para mostrar la secuencia de latencias permitidas. Indique también las latencias prohibidas que determina una instrucción que escribe en el banco de registros.*

Pregunta 3: *Calcule la latencia media de inicio (LMI).*

Pregunta 4: *Calcule el IPC si suponemos que sólo se producen riesgos estructurales debidos a los caminos de acceso al banco de registros y no se producen riesgos de datos.*

Un análisis en mayor detalle del banco de registros permite relajar la solución adoptada previamente en cuanto al número de accesos concurrentes en un ciclo. Como el banco de registros se ha construido utilizando 2 subbancos de registros, se puede estar escribiendo en un subbanco y efectuar 2 lecturas en el otro subbanco en un mismo ciclo. La explotación de esta característica es tarea del compilador, el cual se encarga de asignar los registros de forma que se reduzcan las situaciones de riesgo estructural. Ahora bien, las situaciones de riesgo estructural aún perduran y por tanto es necesario controlarlas.

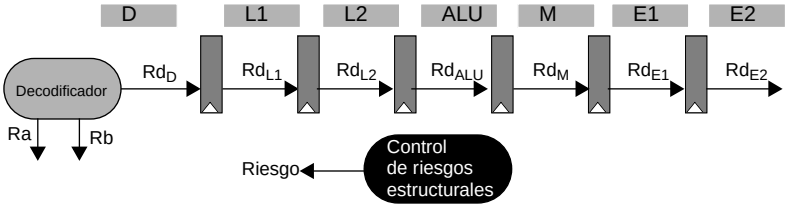
Como etapa de retención utilizaremos la etapa D. En la etapa D disponemos de los identificadores de registros fuente de la instrucción que ocupa ésta etapa y, en cualquiera de las etapas posteriores a la etapa D, disponemos del identificador de registro destino de la instrucción que ocupa la etapa. Cada uno de los identificadores de registro está acompañado por una señal de validación que no tendremos en cuenta. Los bits de los identificadores de registro se numeran de la siguiente forma:

registros fuente	Ra	a_3	a_2	a_1	a_0
	Rb	b_3	b_2	b_1	b_0

registro destino	Rd	d_3	d_2	d_1	d_0

donde el subíndice indica la ponderación del bit cuando el vector de bits se interpreta como un número natural.

Pregunta 5: *Muestre en el esquema que se adjunta las entradas al módulo Control de riesgos estructurales y diseñe la lógica del módulo, indicando explícitamente los bits de cada identificador de registros que se utilizan.*



El procesador emula un funcionamiento serie cuando detecta un riesgo estructural o de datos.

Pregunta 6: *Muestre en un diagrama temporal la interpretación de la siguiente secuencia de 5 instrucciones. Cuando se produzcan acciones de retención hay que mostrar la inyección de instrucciones nop de forma explícita e indicar la causa de la retención en la parte inferior del diagrama temporal (ciclos perdidos).*

instrucción	instrucción	instrucción
1. add **R4**, R0, R1	3. add **R5**, R4, R9	5. add **R8**, R3, R2
2. sub **R8**, R3, R5	4. add **R12**, R11, R12	

Ejercicio 3.9

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

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

ETAPA	FUNCIONALIDAD	RECURSOS
CP	determinar el CP	sumador
B	búsqueda de la instrucción	memoria de instrucciones (MI)
D/L	decodificación, lectura de registros	decodificador, 2 caminos de lectura al banco de registros
ALU	operación aritmético-lógica	ALU
M	acceso a memoria de datos, si es el caso	memoria de datos (MD)
ES	escritura en el banco de registros	1 camino de escritura al banco de registros

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.

Las instrucciones de secuenciamiento actualizan el registro CP en la etapa ES.

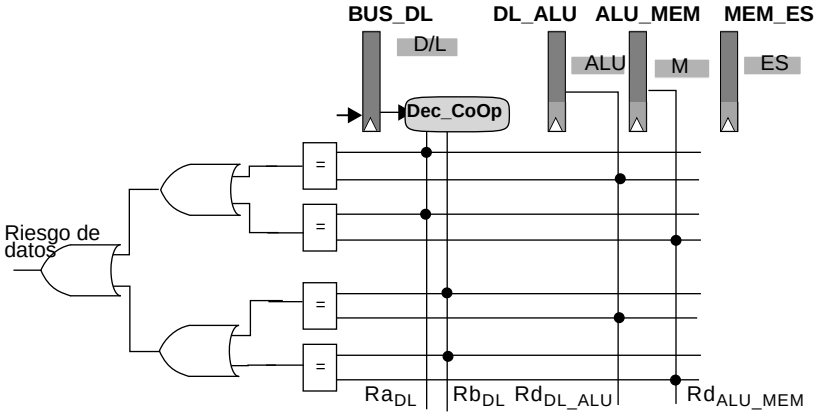
El conjunto de instrucciones del procesador se amplía con la siguiente instrucción.

Tipo	Descripción	Operandos	Semántica
Load con actualización loadA	Lectura de memoria y actualización del registro base	ra rb lite.	ra^v = Mem [rb^v + ExtSig (literal)] rb^v = rb^v + ExtSig (literal) si ra = rb el valor almacenado es indeterminado

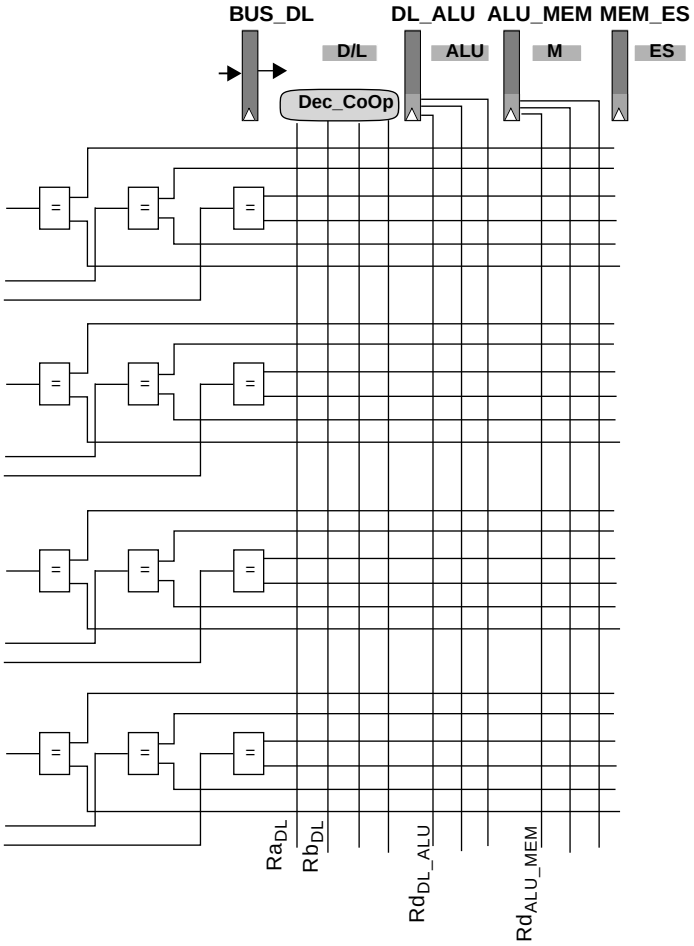
El superíndice v indica contenido del registro

Pregunta 1: *Indique los recursos adicionales (del tipo descrito previamente) que deben añadirse al camino de datos para interpretar la instrucción descrita. Justifique la respuesta.*

En la siguiente figura se muestra el circuito que detecta riesgos de datos antes de añadir la instrucción descrita. Se compara el identificador de registro destino de las instrucciones que están en las etapas ALU y M con los identificadores de registro fuente de la instrucción en la etapa D/L. En la figura Ra y Rb son los identificadores de los registros fuente de la instrucción que está en la etapa D/L. Rdx es el identificador del registro destino de la instrucción que ocupa la etapa cuyo registro de desacoplo de entrada es x. Los puntos negros en la matriz de cables indican conexión del cable vertical y horizontal. En el circuito no se explicitan las señales de validación de los identificadores de registro.



Pregunta 2: *Proponga un circuito de detección de riesgo de datos cuando se añade la instrucción descrita. Justifique la respuesta. En la figura que se suministra indique las conexiones mediante puntos negros y añada las puertas necesarias. Etiquete las señales e indique el significado de las mismas.*



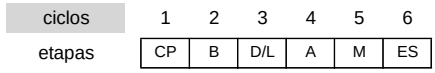
Suponga la siguiente secuencia de instrucciones.

```
1$:  loadA R1, 8(R2)        load con actualización
     cmpeq R4, R2, R6
     add R3, R3, R1
     beq R4, 1$
```

Pregunta 3: *Dibuje un diagrama temporal que muestre la interpretación de una iteración del bucle, suponiendo que el salto es tomado. Calcule el CPI de una iteración.*

Ejercicio 3.10

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	RECURSOS
CP	determinar el CP	sumador
B	búsqueda de la instrucción	memoria de instrucciones (MI)
D/L	decodificación, lectura de registros	decodificador, 2 caminos de lectura al banco de registros
ALU	operación aritmético-lógica	ALU
M	acceso a memoria de datos	memoria de datos (MD)
ES	escritura en el banco de registros	1 camino de escritura al banco de registros

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.

Las instrucciones de secuenciamiento actualizan el registro CP en la etapa ES.

Suponga la siguiente secuencia de instrucciones.

```
1$:  load R1, 0(R2)
     cmpeq R4, R1, R6
     add R2, R2, #8
     beq R4, 1$
```

La instrucción cmpeq es de tipo ENT y utiliza la ALU para efectuar la comparación. La semántica es la siguiente:

- **cmpeq rc**, ra, rb: comparación de igualdad.

  cmpeq if ($ra^v = rb^v$) then $rc^v = 1$
  else $rc^v = 0$

El contenido del registro ra se compara con el contenido del registro rb, interpretando ambos contenidos como números enteros. Si la relación especificada en la instrucción se cumple, se escribe el valor 1 en el registro rc, en caso contrario se escribe el valor 0 en el registro rc.

Pregunta 1: *Dibuje un diagrama temporal que muestre la interpretación de una iteración del bucle. En el diagrama temporal debe mostrarse la inyección de instrucciones nop cuando se gestiona un riesgo. Indique los ciclos perdidos por cada tipo de riesgo y calcule el CPI de una iteración.*

El computador donde se ejecuta el programa funciona a una frecuencia de 500 Mhz y consume una potencia de 30 W (vatios). La batería que alimenta al procesador suministra 1 A · H (amperios por hora) a 5 voltios.

Pregunta 2: *Calcule la energía de la batería.*

Pregunta 3: *Calcule la energía consumida por el procesador en un ciclo.*

Pregunta 4: *¿Cuántas iteraciones del bucle se pueden ejecutar antes de que la carga de la batería se reduzca a la mitad?. Exprese el resultado en millones de iteraciones.*

Ejercicio 3.11

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

ciclos	1	2	3	4	5	6
etapas	CP	B	D/L	A	M	ES

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

ETAPA	FUNCIONALIDAD	RECURSOS
CP	determinar el CP	sumador
B	búsqueda de la instrucción	memoria de instrucciones (MI)
D/L	decodificación, lectura de registros	decodificador, 2 caminos de lectura al banco de registros
ALU	operación aritmético-lógica	ALU
M	acceso a memoria de datos	memoria de datos (MD)
ES	escritura en el banco de registros	1 camino de escritura al banco de registros

El conjunto de instrucciones del procesador puede interpretarse sin que se produzcan riesgos estructurales cuando se acierta en la cache de datos.

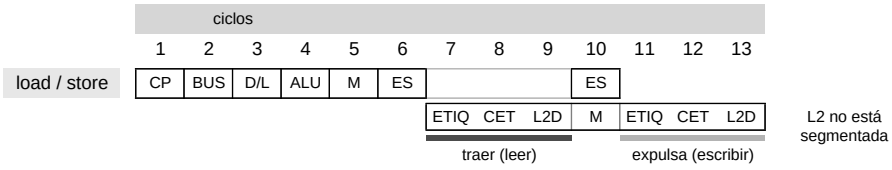
En el mismo ciclo se puede escribir y leer, en este orden, un registro del banco de registros.

Las instrucciones de secuenciamiento actualizan el registro CP en la etapa ES.

La memoria de datos es una cache y por tanto, se pueden producir fallos de cache. La cache utiliza como mecanismo de coherencia actualización retardada y asignación de bloque en cache tanto en fallos de lectura como en fallos de escritura.

En un fallo de cache, el algoritmo de reemplazo, al determinar el contenedor de cache que se utilizará para almacenar el bloque de datos, puede decidir expulsar un bloque que ha sido actualizado. En este caso son necesarios 2 accesos al siguiente nivel. En primer lugar se trae el bloque de datos del siguiente nivel a la cache y posteriormente se escribe en el siguiente nivel el bloque que se reemplaza (expulsión), si éste ha sido actualizado. El bloque que se reemplaza se almacena en un buffer hasta el instante en que se actualiza el siguiente nivel.

La segmentación descrita previamente se extiende, cuando se produce un fallo de cache, de la forma que se muestra en la siguiente figura. En la segunda fila de la figura se muestra el acceso al siguiente nivel de la jerarquía, en el cual supondremos que siempre está el bloque de datos.



Cuando se produce un fallo de cache y en el caso de una instrucción load, en el ciclo 6 no se actualiza el banco de registros. El acceso al siguiente nivel no está segmentado y se requieren 3 ciclos para acceder (ETIQ, CET, L2D).

En una operación de lectura la cache se carga (M) en el ciclo 10 y en paralelo se actualiza el banco de registros (ES). En una operación de escritura en el ciclo 10 sólo se carga la cache. Los

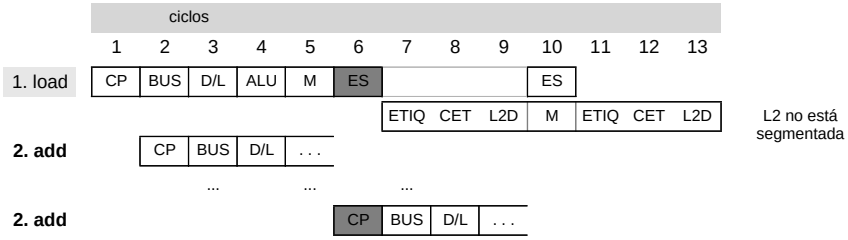
ciclos 11, 12 y 13 se utilizan para transferir y almacenar, en el siguiente nivel de la jerarquía, el bloque que se ha expulsado de cache.

Supondremos que la cache es bloqueante. Así mismo, si durante el proceso de interpretación de instrucciones es necesario retener una instrucción en una etapa, estas sólo pueden ser las etapas D/L, BUS y CP. Bloquear instrucciones en otras etapas es más complicado, ya que, entre otras cuestiones, la cantidad de información que se transmite entre etapas es mayor y parte de la información debe actualizarse para proseguir la interpretación, mientras que la otra parte de información no debe actualizarse.

La detección de un fallo de cache se efectúa en la etapa M y está lo suficientemente tardía en el proceso de interpretación que otra instrucción más joven puede estar en la etapa ALU, pudiendo ser de acceso a memoria. En estas circunstancias no se pueden retener las instrucciones. Por ello, una alternativa es utilizar un mecanismo que anule las instrucciones a partir de la instrucción que sigue a la que produce el fallo de cache y que vuelva a iniciar el proceso de interpretación de las instrucciones anuladas. Al mecanismo lo denominaremos “argucia de re-interpretación”, ya que se ha predicho que sería un acierto en cache, se ha detectado un fallo y la forma de gestionarlo es volver a interpretar las instrucciones. Notemos que esta decisión de anular todas las instrucciones que siguen a la instrucción que produce el fallo de cache es conservadora.

En caso de un fallo de cache, la señal de fallo de cache se activa (valor 1) en el ciclo 5, cuando la instrucción de acceso a memoria está en la etapa M y deja de estar activa al finalizar el ciclo.

La re-interpretación se empieza actualizando el registro CP con la dirección de la 1ª instrucción que debe re-interpretarse. Según se muestra en la siguiente figura, esta acción se produce en el ciclo en el cual la instrucción de acceso a memoria, que produce un fallo de cache, se encuentra en la etapa ES (ciclo 6). En los ciclos 2 ... 5 se ha seguido utilizando el secuenciamiento que determina la interpretación de instrucciones. La anulación de las instrucciones más jóvenes que la instrucción de acceso a memoria, cuando se detecta un fallo de cache, se efectúa en el instante que pasan a la etapa M.



En los diagramas temporales que se solicitan, muestre la anulación de las instrucciones posteriores al fallo de cache utilizando la palabra nop, en lugar del acrónimo de la etapa, a partir del ciclo que la instrucción está anulada. Así mismo indique las latencias prohibidas, teniendo en cuenta el primer riesgo estructural que se produciría.

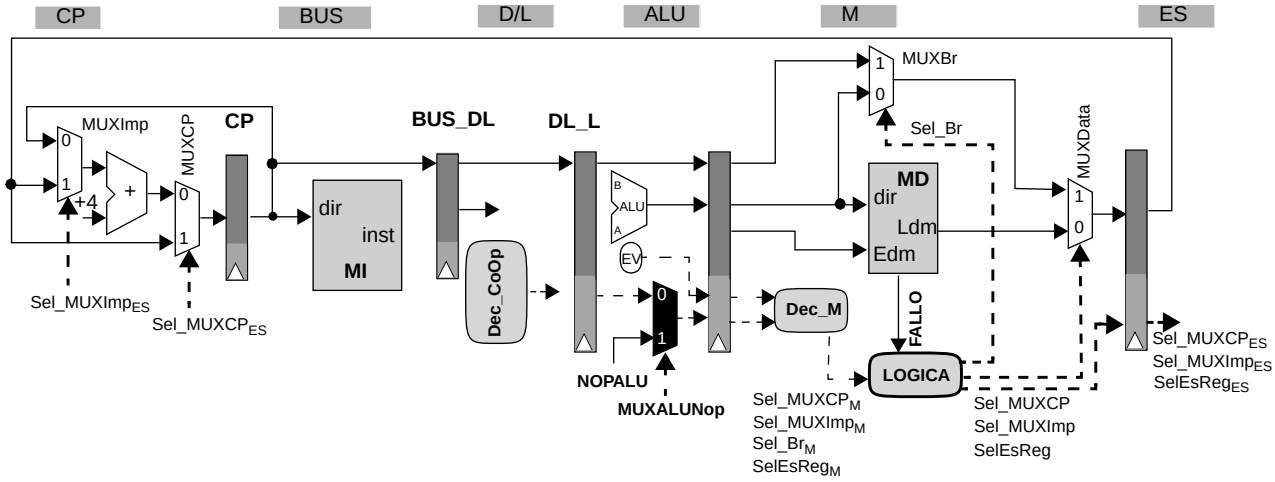
En las siguientes 2 preguntas muestre en un diagrama temporal, teniendo en cuenta el mecanismo de actuación al detectar un fallo de cache, los posibles riesgos estructurales:

Pregunta 1: *Cuando un fallo de cache no requiere la actualización del siguiente nivel de la jerarquía.*

Pregunta 2: *Cuando un fallo de cache requiere la actualización del siguiente nivel de la jerarquía.*

En la siguiente figura se muestra parte del camino de datos y en particular las partes de interés para gestionar un fallo de cache. Se muestra con detalle la parte de secuenciamiento, la etapa de memoria y el multiplexor que permite anular las instrucciones en curso cuando se detecta un fallo de cache.

La señal MUXALUNop se utiliza para inyectar una instrucción nop en la etapa M en los ciclos posteriores al fallo de cache, con el objetivo de anular las instrucciones en curso de interpretación. La señal FALLO toma el valor 1 cuando se produce un fallo de cache.



Las señales de control Sel_MUXImp, Sel_MUXCP, Sel_Br y SelESReg permiten encaminar la información de secuenciamiento. En un procesador sin el mecanismo descrito de re-interpretación se calculan en el módulo Dec_M. Ahora bien, cuando se añade el mecanismo de re-interpretación es necesario que estas señales sean modificadas en el módulo LOGICA, el cual también tiene como entrada la señal FALLO. Las dos últimas señales (Sel_Br y SelESReg) se utilizan en el mismo ciclo que se calculan y las dos primeras señales en el siguiente ciclo.

Pregunta 3: *Indique la función lógica que determina el valor de la señal MUXALUNop utilizando la señal FALLO y las que estime oportunas. En el diseño de la lógica se pueden utilizar puertas lógicas y registros.*

Pregunta 4: *Indique cómo el módulo LOGICA modifica las señales de salida del módulo Dec_M para obtener las señales Sel_MUXImp, Sel_MUXCP, Sel_Br y SelESReg que permiten gestionar un fallo de cache de la forma descrita. En el diseño de la lógica se pueden utilizar puertas lógicas y registros. Utilice el subíndice M para nombrar a las señales de entrada del módulo LOGICA.*

En el diseño propuesto con gestión del fallo de cache, además de riesgos estructurales se pueden producir riesgos de datos, ya que las instrucciones load son no bloqueantes.

Pregunta 5: *Muestre en un diagrama temporal la interpretación de la siguiente secuencia de instrucciones. En concreto, los riesgos de datos debidos a registros antes y después de empezar la re-interpretación de las instrucciones anuladas. No es necesario mostrar la inyección del instrucciones nop cuando se actúa al detectar un riesgo de datos, pero si es necesario en la anulación de instrucciones debido a un fallo de cache.*

1. load R1, 0(R3)	Fallo de cache
2. add R4, R1, R2	
3. load R7, 8(R12)	Acierto en cache
4. add R11, R12, R5	
5. sub R19, R20, R30	
6. store R11, 12(R9)	Acierto en cache

La detección de uno de los riesgos de datos que se ha observado en la respuesta de la pregunta anterior no es factible con la lógica (comparadores de identificador de registro) que se utiliza cuando se supone que siempre se acierta en L1, ya que no se tiene en cuenta la posiblidad de fallo de cache.

Pregunta 6: *Suponga que se dispone de la señal RD que indica si existe un riesgo de datos cuando se supone que siempre se acierta en L1. Proponga un mecanismo conservador que gestione los posibles riesgos de datos cuando existe un fallo de cache en curso de servicio. En el diseño de la lógica se pueden utilizar puertas lógicas y registros, pero no se pueden utilizar comparadores de identificador de registro.*

Ejercicio 3.12

En el lenguaje máquina de un procesador podemos distinguir el siguiente formato, clases y tipo de instrucciones:

		Campos de la instrucción																				
		31	. . .	26	25	...	21	20	...	16	15	...	12	11	...	5	4	...	0		
Clases	**Tipo**	6			5			5			4			7			5				**Descripción**
ENT	RR	CoOp			ra			rb			0	...	0	func			rc				Cálculo
IS	BR	CoOp			ra			literal													Saltos relativos
	JMP	CoOp			ra			rb						0	...	0	0	...	0		Saltos indexados

En la siguiente tabla se describe la semántica de cada tipo de instrucción:

Clase	Tipo	Descripción	Operandos	Especificación semántica
ENT	RR	cálculo	ra rb rc	$rc^v = F(ra^v, rb^v)$
IS	BR	llamada a procedimiento	ra literal	$ra^v = CP^v$ $CP^v = CP^v + 4 \times ExtSig(literal) + 4$
		incondicional	ra literal	$CP^v = CP^v + 4 \times ExtSig(literal) + 4$
		condicionales	ra literal	if $(f[ra^v, CoOp])$ then $CP^v = CP^v + 4 \times ExtSig(literal) + 4$ else $CP^v = CP^v + 4$
	JMP	llamada a procedimiento	ra rb 0	$ra^v = CP^v$ $CP^v = rb^v + 4$
		indexado	ra rb 0	$CP^v = rb^v + 4$

donde el superíndice v indica el valor almacenado en el registro.

Las instrucciones de la clase ENT leen el contenido de dos registros, efectúan un cálculo y almacenan el resultado en un registro.

Las instrucciones de tipo BR calculan la dirección destino de la siguiente forma: a la dirección de la instrucción siguiente a la de secuenciamiento se le suma el campo literal especificado en la instrucción (el tamaño de una instrucción es de 4 bytes). Las instrucciones de secuenciamiento condicional evalúan además el contenido de un registro. El resultado de la evaluación se utiliza para seleccionar entre seguir en secuencia o modificar el secuenciamiento implícito. Las instrucciones de secuenciamiento de tipo JMP suman 4 al contenido de un registro y el resultado se utiliza como dirección de la siguiente instrucción que debe interpretarse. Los dos tipos de instrucciones de secuenciamiento (BR y JMP) incluyen instrucciones que almacenan la dirección de la instrucción de secuenciamiento en un registro especificado en la instrucción.

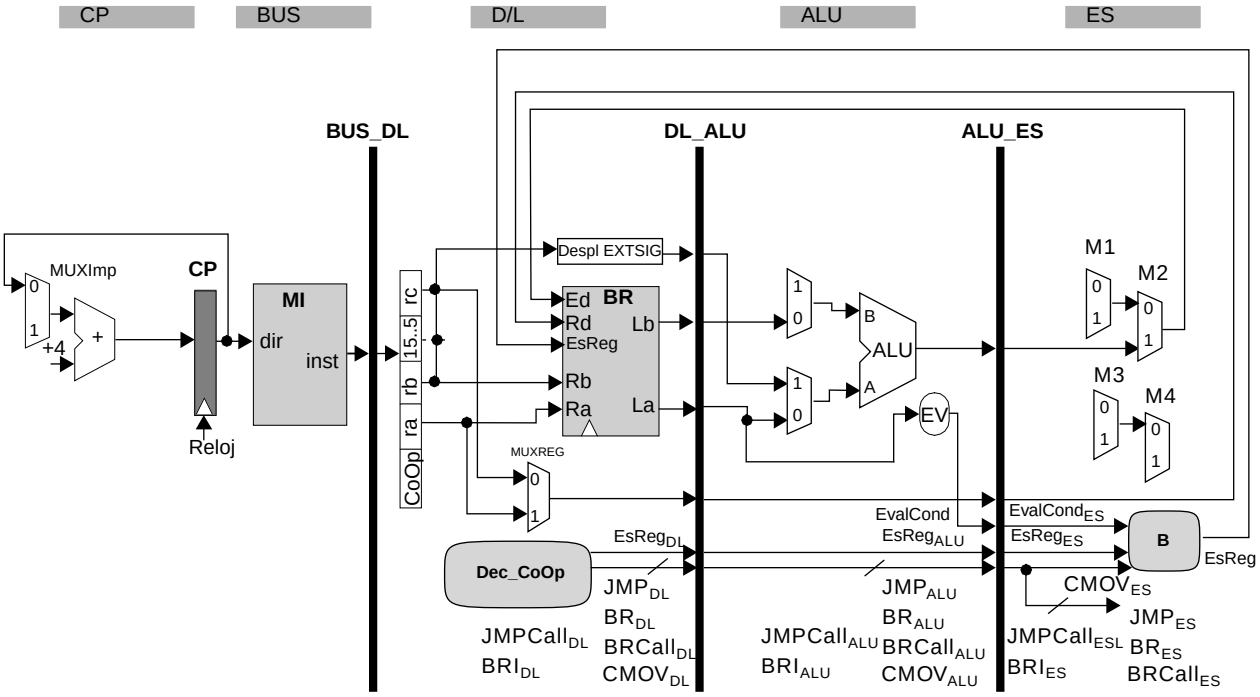
El lenguaje máquina también dispone de instrucciones para acceder a memoria. Ahora bien, nosotros nos centraremos en las instrucciones que se han descrito. Para interpretar las instrucciones descritas supondremos que el procesador utiliza la siguiente segmentación.

ciclos	1	2	3	4	5
	CP	BUS	D/L	ALU	ES

La actualización de un registro del banco de registros y del registro CP en instrucciones de secuenciamiento se efectúa en el ciclo 5 relativo al inicio de interpretación.

En la siguiente figura se muestran los elementos del camino de datos del procesador segmentado y las conexiones necesarias para interpretar las instrucciones de la clase ENT.

La señal EvalCond toma el valor 1 cuando se cumple la condición. Las señales BR y JMP indican tipo de instrucción. Las señales BRI, BRCall o JMPCall indican respectivamente instrucción de secuenciamiento incondicional o llamada a procedimiento de tipo BR o JMP. La señal CMOV sólo hay que utilizarla para contestar a la pregunta e) e indica instrucción de movimiento condicional. Todas estas señales cuando se activan toman el valor 1.



En las siguientes preguntas se solicita establecer conexiones entre los elementos del camino de datos para interpretar las instrucciones de secuenciamiento. Cuando se contesta una pregunta debe tenerse en cuenta que las conexiones que se establezcan deben ser compatibles con las conexiones estable-

cidas en contestaciones anteriores. Del mismo modo la contestación de la primera pregunta debe tener en cuenta las conexiones que se muestran en el camino de datos para interpretar las instrucciones de la clase ENT.

Pregunta 1: *Añada en el camino de datos anterior las conexiones necesarias para interpretar las instrucciones de tipo BR.*

Pregunta 2: *Añada en el camino de datos mostrado previamente, exclusivamente, las conexiones necesarias para interpretar las instrucciones de tipo JMP.*

Pregunta 3: *Especifique el control de los multiplexores MUXImp, M2, M3 y M4 teniendo en cuenta todas las instrucciones de secuenciamiento.*

En el registro de desacoplo BUS_DL se pueden distinguir en general varias señales o campos. Una de las señales se corresponde con los bits que codifican la instrucción que se ha leido de memoria y que se decodifica en la etapa D/L; a este campo lo denominaremos inst (ver figura). Suponga que, cuando en un riesgo de secuenciamiento desde la etapa BUS se inyecta una instrucción NOP, sólo se actúa sobre el campo inst. Las otras señales o campos transmiten la misma información que se transmitiría si no se hubiera inyectado la instrucción NOP.

Pregunta 4: *Modifique el camino de datos para que todas las instrucciones de secuenciamiento tengan un ciclo menos de penalización debido al riesgo de secuenciamiento. Para ello sólo puede utilizar multiplexores de 2 entradas en la etapa CP. Suponiendo que el tiempo de ciclo está determinado por la etapa CP, la solución propuesta debe reducir el tiempo de ciclo.*

Algunos procesadores disponen de instrucciones cuya operación es condicional. El objetivo de incluir estas instrucciones en el lenguaje máquina es eliminar las instrucciones de secuenciamiento condicional en algunas secuencias de código. Con ello se pretende reducir los ciclos perdidos. Una instrucción condicional típica es la siguiente instrucción de movimiento condicional.

- **cmovge**: mover si mayor igual a cero.

El formato de estas instrucciones es el especificado para la clase ENT y la semántica es la siguiente:

Clase	Descripción	Operandos	Especificación semántica
ENT	mov. condicional	ra | rb | rc	if (f[ra^v , CoOp]) then $rc^v = rb^v$

El contenido del registro ra se evalúa utilizando la condición especificada en la instrucción. Si la condición especificada se cumple, se escribe el contenido del registro rb en el registro rc. En caso contrario no se modifica el contenido del registro rc.

Un registro del banco de registros se puede escribir y leer, en este orden, en el mismo ciclo. Suponga que en las instrucciones de movimiento condicional la señal $EsReg_{DL}$ siempre se activa en la etapa D/L, ya que se efectúa la hipótesis de que se actualizará el registro destino.

Pregunta 5: *Añada al camino de datos mostrado previamente, exclusivamente, las conexiones necesarias para interpretar una instrucción de movimiento condicional. Indique la función lógica que hay que implementar en el módulo B. Así mismo indique los ciclos perdidos, en función del resultado de evaluar la condición, si la instrucción consumidora del resultado producido por una instrucción de movimiento condicional está a distancia 1, 2 o 3.*

Ejercicio 3.13

En un procesador el tamaño de las instrucciones de lenguaje máquina es fijo e igual a 2 bytes. En el lenguaje máquina podemos distinguir los siguientes tipos de instrucciones de ruptura de secuencia (o saltos). Esto es, que pueden modificar el secuenciamiento implícito.

- *Saltos relativos al contador de programa*.

El formato contiene un código de operación de 4 bits, un campo de 3 bits para identificar un registro, un campo de 3 bits para especificar la función y un campo de 6 bits para especificar el desplazamiento. Cuando el campo registro no se utiliza, los bits correspondientes son igual a cero.

campos de la instrucción

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
4				3			3			6						
CodOp				Ra			func			desplazamiento						formato
0	1	1	1	0	0	0	0	0	0	desplazamiento						**br**
0	1	1	1	Ra			0	0	1	desplazamiento						**bsr**
0	1	1	1	Ra			func			desplazamiento						**bcond**

instrucciones

En todos los casos el desplazamiento se interpreta como un desplazamiento con signo, en número de instrucciones, desde la dirección de la instrucción. Esto es, se desplaza a la izquierda un bit (para alinear la dirección a 2 bytes), se extiende el signo a 16 bits, y se suma al contador de programa (CP) para calcular la dirección lógica destino.

- **bcond**: salto condicional.

El campo func de la instrucción indica la evaluación que debe efectuarse del contenido del registro Ra. Se comprueba si el registro Ra cumple la relación especificada en la instrucción. Si se cumple la relación el CP se carga con la dirección lógica calculada; en caso contrario la ejecución continua con la siguiente instrucción en secuencia.

bcond

$$\text{if } (Ra^v == \text{func}) \text{ then } CP^v = CP^v + 2 \times \text{ExtSig}(\text{despl})$$
$$\text{else } CP^v = CP^v + 2$$

donde el superindice v indica el valor almacenado en el registro.

- **br**: salto incondicional.

El CP se carga con la dirección lógica calculada (destino del salto).

br

$$CP^v = CP^v + 2 \times \text{ExtSig}(\text{despl})$$

- **bsr**: llamada a procedimiento. Salto incondicional y guardar la dirección de retorno.

El CP de la siguiente instrucción se almacena en el registro Ra y el CP se carga con la dirección destino del salto.

bsr

$$Ra^v = CP^v + 2$$
$$CP^v = CP^v + 2 \times \text{ExtSig}(\text{despl})$$

- *Salto indexado*.

El formato contiene un código de operación de 4 bits y un campo de registro de 3 bits, siendo todos los otros bits igual a cero.

4				3			3	6						
CodOp					Ra		func							formato
0	0	1	1	0	0	0	Rb	0	0	0	0	0	0	**jmp** instrucción

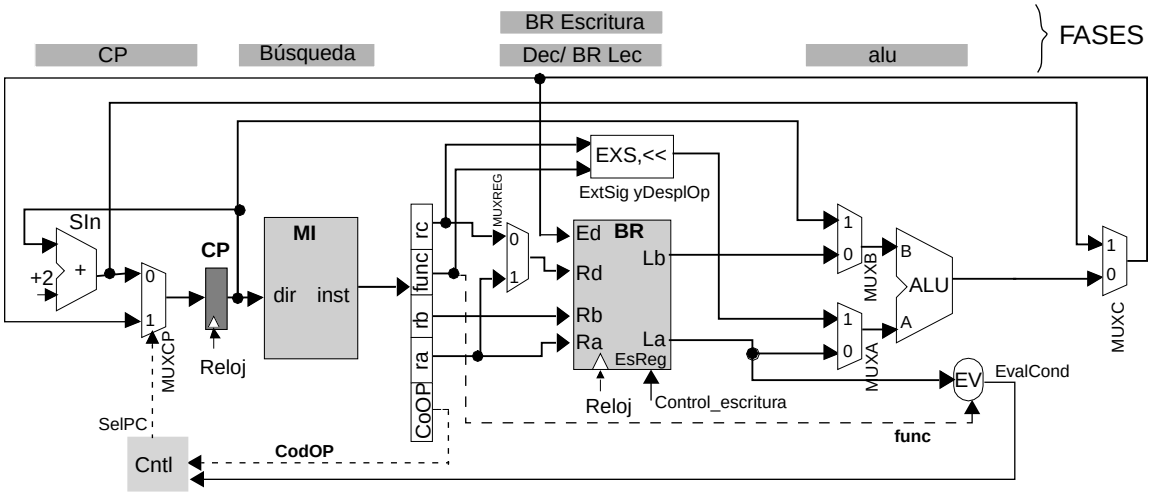
- **jmp**: salto incondicional. (ej.: retorno de procedimiento).

EL CP se carga con el contenido del registro Rb, ignorando el bit de menor peso.

$$CP^{v} = Rb^{v}$$

Camino de datos simplificado.

Un camino de datos simplificado donde se interpretan las instrucciones de ruptura de secuenciamiento se muestra en la siguiente figura. En cuanto al control sólo se muestra la parte que controla el multiplexor MUXCP. Este multiplexor selecciona entre secuenciamiento implícito o la dirección que se establece mediante la instrucción de ruptura de secuenciamiento. Seguidamente se efectúa una descripción del camino de datos relacionada con las instrucciones de ruptura de secuenciamiento.



Los registros del banco de registros y el registro CP se actualizan en el flanco ascendente de la señal de Reloj.

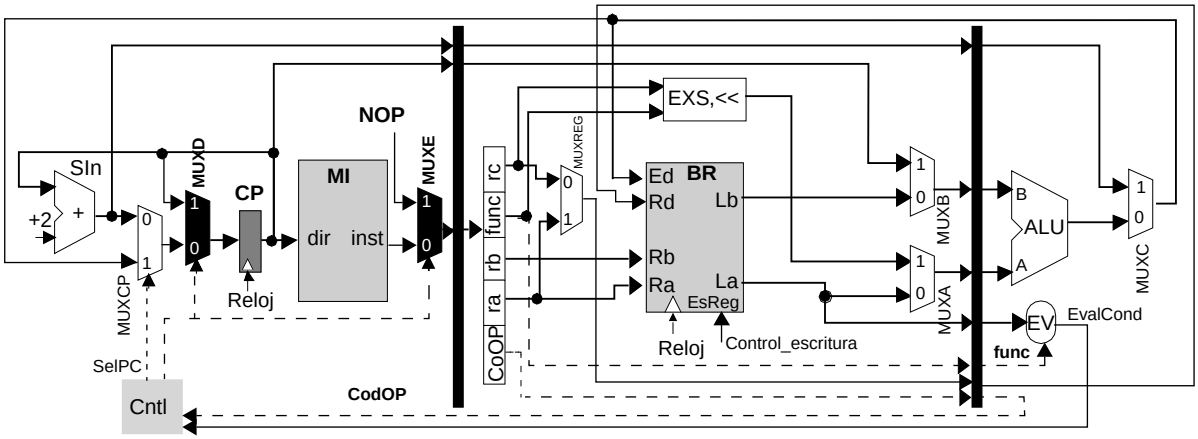
Las instrucciones de ruptura de secuencia relativas al contador de programa utilizan la ALU para efectuar el cálculo de la dirección lógica especificada en la instrucción.

El registro del banco de registro cuyo identificador es el 0 (R0) está cableado al valor cero. Cuando una instrucción lo utiliza como registro fuente se lee el valor 0 y en caso de utilizarlo como registro destino no se actualiza.

Observemos que en las instrucciones de ruptura de secuencia tipo **jmp** los bits del campo de la instrucción denominado Ra son cero. Este valor se corresponde con el identificador del registro R0 y su contenido se puede sumar al contenido del registro Rb sin que se modifique el valor del registro Rb.

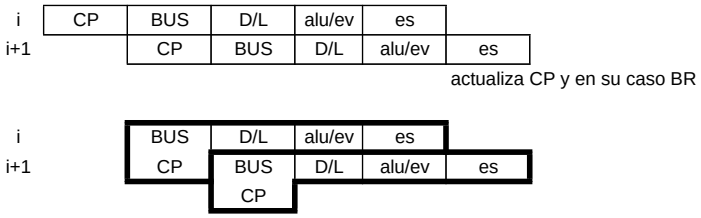
$$CP^v = Rb^v + R0^v = Rb^v$$

Una segmentación “ingenua” del camino de datos se muestra en la siguiente figura. Las barras verticales negras representan registros de desacoplo, que se actualizan siempre en el flanco ascendente de la señal de Reloj y no tienen señales adicionales de puesta a cero ni de permiso de escritura.



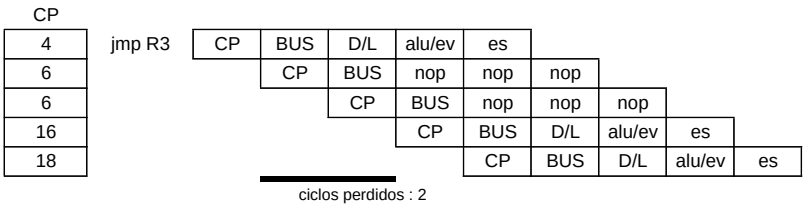
El siguiente diagrama temporal muestra la utilización de las etapas por parte de una instrucción. Observemos que en este diseño segmentado, la acción de escribir en el banco de registros en el flanco ascendente de la señal de Reloj permite que el propio registro del banco haga de registro de desacoplo entre la etapa de alu/ev y la etapa de escritura. En este diagrama temporal se muestra el cálculo del CP de una instrucción en la misma línea que las etapas (búsqueda, etc) en las cuales se interpreta la misma

instrucción (arriba). Otra representación equivalente (abajo) incluye el cálculo del CP de la siguiente instrucción en la etapa de búsqueda de la instrucción previa.



El control del procesador (Cntl) se encarga de bloquear el inicio de interpretación de las instrucciones posteriores al salto. Así mismo, alimenta con la instrucción NOP a la etapa D/L, una vez se ha detectado el riesgo de secuenciamiento, durante los ciclos necesarios hasta que se puede actualizar el CP con la dirección que determina la instrucción de salto. Mediante el multiplexor MUXD se controla que el CP no se actualice y mediante el multiplexor MUXE se inserta una instrucción NOP en la etapa D/L.

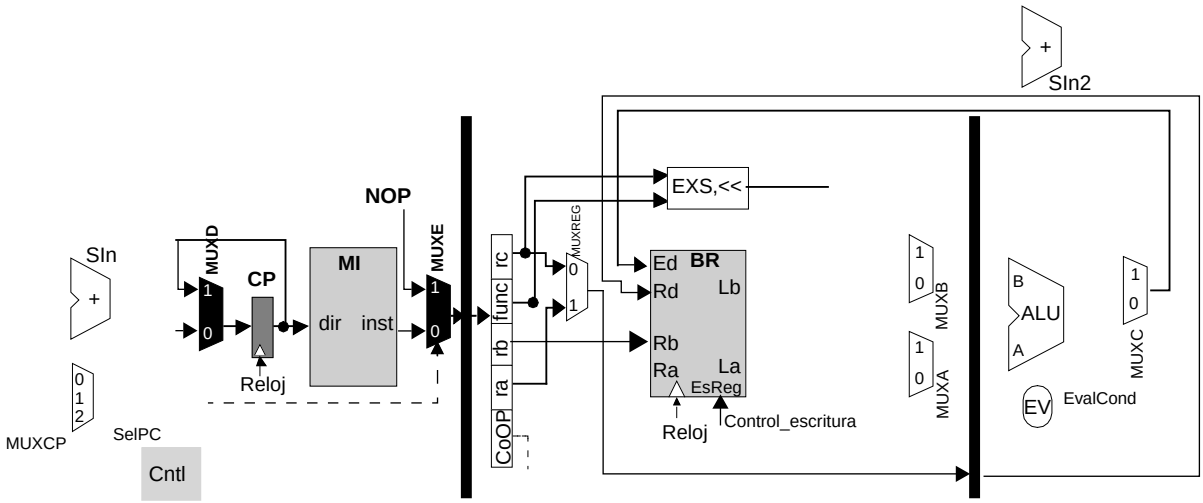
En estas condiciones la ejecución de cualquier instrucción de ruptura de secuencia representa una penalización de 2 ciclos. En la siguiente figura se muestra como ejemplo la ejecución de una instrucción del tipo **jmp**, donde el contenido del registro R3 es 16 y la dirección lógica de la instrucción **jmp** es 4.



En todas las preguntas que siguen se supone un control por bloqueo en los riesgos de secuenciamiento. Se piden modificaciones del camino de datos para reducir la penalización debida a los riesgos de secuenciamiento y piden evaluar la mejora respecto al diseño “ingenuo”, que denominaremos diseño base.

En las contestaciones sólo deben añadirse las conexiones necesarias, en el camino de datos y en el control (Cntl) que determinan la salida del elemento MUXCP, para interpretar el tipo de salto que se pregunta.

Para indicar las modificaciones del camino de datos debe utilizarse el siguiente dibujo. Los circuitos están en la etapa en la cual se USAN (conexión de sus entradas y salidas), sin importar su ubicación espacial en el dibujo. Para contestar una pregunta pueden no ser necesarias las 3 entradas del multiplexor MUXCP.



Para mostrar la penalización de un riesgo de secuenciamiento se utilizará un cronograma que muestre una secuencia de instrucciones, donde la 1ª instrucción es un salto. En el cronograma, a partir del ciclo en que actúa el control de riesgo de secuenciamiento, debe visualizarse la palabra NOP en las instrucciones afectadas. También se especificará el valor del CP al final de cada ciclo; el tamaño de una instrucción es de 2 bytes y una dirección identifica una posición de memoria que almacena un byte.

Para evaluar la mejora de rendimiento supondremos que, en un conjunto de programas de prueba, el CPI del diseño base es 1.7 y que la distribución de las instrucciones de ruptura de secuencia es la siguiente.

	tipo de salto	frecuencia
condicional	bcond	12%
incondicional	br	3%
llamada a procedimiento	bsr	2%
salto indexado	jmp	3%

En todos los casos, la mejora se calcula como el cociente $Tiempo|_{base}$ / $Tiempo|_{modificado}$ y sólo se considera la reducción de penalización en el riesgo de salto que se pregunta.

Considere que cualquier modificación del camino de datos deja inalterado el valor del tiempo de ciclo.

Pregunta 1: *Modifique el camino de datos del diseño base para reducir la penalización de los riesgos debidos a los saltos indexados. Indique la penalización en ciclos.*

Pregunta 2: *Calcule la mejora que se obtiene, respecto al diseño base, debido a las modificaciones efectuadas en el camino de datos en la pregunta 1).*

Pregunta 3: *Modifique el camino de datos del diseño base para reducir la penalización de los riesgos debidos a los saltos relativos al CP e indique la penalización en ciclos. Utilice el elemento SIn2.*

Pregunta 4: *Calcule la mejora que se obtiene, respecto al diseño base, debido a las modificaciones efectuadas en el camino de datos en la pregunta 3).*

Suponga que se modifica la especificación de las instrucciones de salto relativas al CP. La nueva especificación del cálculo de la dirección efectiva es la siguiente.

bcond, br, bsr $CP^v = (CP^v + 2) + 2 \times ExtSig (despl)$

Esto es, el CP destino del salto se calcula como la suma del CP de la instrucción que sigue al salto más el desplazamiento especificado en la instrucción, interpretado como un desplazamiento con signo, en número de instrucciones, desde la dirección de la instrucción.

Pregunta 5: *Suponga que no se dispone del elemento SIn2. Modifique el camino de datos del diseño base para reducir la penalización de los riesgos debidos a los saltos relativos al CP.*

Ejercicio 3.14

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

ciclos	1	2	3	4	5	6
etapas	CP	B	D/L	A	M	ES

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

ETAPA	FUNCIONALIDAD	RECURSOS
CP	determinar el CP	sumador (+)
B	búsqueda de la instrucción	memoria de instrucciones (MI)
D/L	decodificación, lectura de registros	decodificador, 2 caminos de lectura al banco de registros (BR2L)
A	operación aritmético-lógica	ALU
M	acceso a memoria de datos	memoria de datos (MD)
ES	escritura en el banco de registros	1 camino de escritura al banco de registros (BRE)

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. Las instrucciones de secuenciamiento actualizan el registro CP en la etapa ES. Un riesgo se gestiona en la fase de decodificación, que en el procesador descrito previamente se corresponde con la etapa D/L.

Suponga la siguiente secuencia de instrucciones.

dirección	instrucción
100	load **R1**, 0(R2)
104	add **R3**, R1, R3
108	add **R2**, R2, #8
112	sub **R4**, R6, R5
116	beq R4, 100

El número de iteraciones del bucle es 10000

Pregunta 1: *Dibuje un diagrama temporal que muestre la interpretación de una iteración del bucle y la 1ª instrucción de la siguiente iteración. En el diagrama temporal debe mostrarse en la columna de la izquierda, en cada fila, la dirección en la entrada del registro de desacoplo CP (la que se utiliza para efectuar la búsqueda). También debe mostrarse la inyección de instrucciones nop cuando se gestiona un riesgo. Indique los ciclos perdidos por cada tipo de riesgo y calcule el CPI de una iteración.*

El computador donde se ejecuta el programa funciona a una frecuencia de 500 Mhz y consume una potencia de 30 W (vatios).

Pregunta 2: *Calcule la energía consumida al ejecutar todas las iteraciones del bucle.*

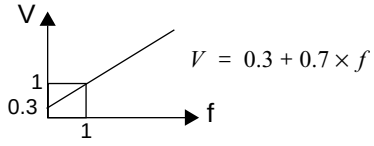
En la siguiente tabla se muestra el retardo de los componentes que deben tenerse en cuenta en cada una de las etapas. El retardo de los otros componentes, incluidos los registros de desacoplo se considera despreciable.

etapa	CP	B	D/L	A	M	ES
recurso	+	MI	BR2L	ALU	MD	BRE
retardo en ns	1.5	2	1	2	2	1

Pregunta 3: *Proponga una segmentación del procesador con menos etapas, y sin añadir lógica, que permita reducir la frecuencia de funcionamiento y que reduzca los ciclos perdidos por riesgos de datos y de secuenciamiento. Indique los recursos que se utilizan en cada etapa. Así mismo, evalúe la frecuencia de funcionamiento, la cual debe ser mayor que 200 Mhz y menor que 260Mhz.*

Pregunta 4: *Utilizando el procesador diseñado en la contestación de la 3ª pregunta, dibuje un diagrama temporal simplificado que muestre la interpretación de una iteración del bucle y la 1ª instrucción de la siguiente iteración. Indique los ciclos perdidos por cada tipo de riesgo y calcule el CPI de una iteración.*

Una aproximación lineal que relaciona la frecuencia y la tensión de alimentación en un dispositivo CMOS es la siguiente, donde la frecuencia y la tensión de alimentación están normalizadas respectivamente a la frecuencia y a la tensión de alimentación de trabajo.



Esta relación nos indica que una reducción del 50% en la frecuencia reducirá la tensión de alimentación un 35% (0.65 x V_0), donde V_0 es la tensión de alimentación original. Esto es, $V = 0.3 + 0.7 \times 0.5 = 0.65$.

De la misma forma, una reducción de la tensión de alimentación a la mitad (0.5 x V_0) reducirá la frecuencia de funcionamiento un 70% (0.3 x f_0).

Pregunta 5: *Cuando se utiliza la segmentación propuesta en la contestación de la 3ª pregunta, calcule la potencia.*

Pregunta 6: *Cuando se utiliza la segmentación propuesta en la contestación de la 3ª pregunta, calcule la energía consumida al ejecutar todas las iteraciones del bucle.*

Pregunta 7: *Calcule la ganancia en energía respecto de la segmentación en 6 etapas.*

El procesador dispone de un mecanismo que permite conmutar de la segmentación en 6 etapas (original, a la que denominamos modo A), a la segmentación propuesta en la contestación de la 3ª pregunta, a la que denominamos modo B.

Este mecanismo de conmutación de modo evalúa el CPI del procesador cada 1170 ciclos y si el CPI es mayor que 2.5 o menor que 2.0 reorganiza la segmentación con el objetivo de reducir o incrementar la frecuencia de funcionamiento. Para reorganizar la segmentación espera a que finalice la ejecución de la última instrucción que se ha empezado a interpretar en el modo actual.

En concreto, cuando el CPI es mayor que 2.5 pasa de utilizar el modo A a utilizar el modo B. Cuando el CPI es menor que 2.0 pasa de utilizar el modo B a utilizar el modo A.

Al finalizar un periodo de 1170 ciclos vuelve a efectuarse una evaluación del CPI partiendo de cero. Esto es, como si fuera el primer periodo. En cada periodo de evaluación no se tienen en cuenta los ciclos necesarios para efectuar el cambio de modo (descarga). Tampoco se tienen en cuenta los ciclos de carga de la segmentación.

En los cálculos que se solicitan seguidamente no contabilice los ciclos de descarga y carga de la segmentación en un cambio de modo. Suponga que el procesador empieza a ejecutar en modo A.

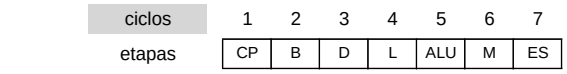
Pregunta 8: *Calcule el número de iteraciones que se ejecutan en cada modo. Así mismo, calcule el tiempo de ejecución.*

Pregunta 9: *Calcule la energía media consumida por iteración cuando se ejecutan todas las iteraciones del bucle y actúa el mecanismo de cambio de modo.*

Pregunta 10: *Calcule la potencia media cuando se ejecutan todas las iteraciones del bucle y actúa el mecanismo de cambio de modo.*

Ejercicio 3.15

Tenemos un procesador segmentado lineal en 7 etapas.



ETAPA	FUNCIONALIDAD
CP	determinar la dirección de la instrucción
B	búsqueda de la instrucción
D	decodificación y detección de riesgos
L	lectura de operandos en el banco de registros
ALU	operación o cálculo de la dirección efectiva o evaluación de la condición y cálculo de la dirección destino
M	acceso a memoria de datos
ES	escritura del resultado en el banco de registros y el contador de programa

El camino de datos dispone de recursos suficientes para que no se produzcan riesgos estructurales. El banco de registros permite la escritura y lectura de un registro en el mismo ciclo.

Los riesgos de datos y de secuenciamiento se detectan en la etapa D. En los riesgos de datos se retienen las instrucciones en las etapas D, B y CP y se inyectan NOPs hacia la etapa L. En un riesgo de secuenciamiento se descartan las instrucciones buscadas (inyectando NOPs hacia la etapa D) hasta que se conoce la dirección de la siguiente instrucción que debe interpretarse (etapa ES).

Pregunta 1: *Indique la penalización de un riesgo de secuenciamiento. Justifique la respuesta mediante un cronograma.*

Pregunta 2: *Muestre el cronograma de ejecución de la secuencia de instrucciones e indique la cantidad total de ciclos perdidos.*

```
Load r3,0(r0)
Load r5,0(r3)
Add r3,r3,#8
Store r5,8(r1)
```

Pregunta 3: *Suponga que el procesador ejecuta una secuencia ilimitada de instrucciones que no son de secuenciamiento. Determine el número mínimo (n) de instrucciones de la secuencia que el procesador puede estar interpretando concurrentemente. Suponga que el procesador ha ejecutado al menos 7 instrucciones de la secuencia.*
Justifique la respuesta mostrando en qué etapas están las n instrucciones (i_1, i_2, …, i_n).

Pregunta 4: *Utilizando comparadores y puertas lógicas, diseñe el circuito para detectar riesgos de datos. Debe mostrar la utilización de las señales de validez.*

Dado el siguiente código, que elimina los elementos con valor impar de una lista con al menos 2 elementos:

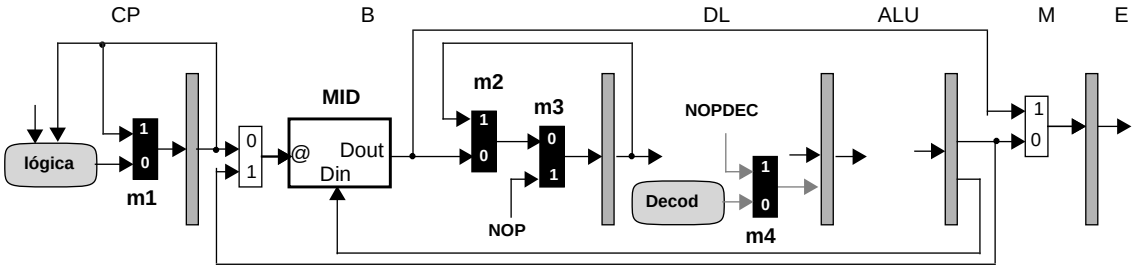
```
do                                  1$:  Load r3, 0(r0)     ;i1 r3 ← mem[r0+0]
{                                        Load r5,8(r1)      ;i2 r5 ← mem[r1+8]
   if (c->dat % 2) != 0) {               And r4, r3, #1     ;i3 r4 ← r3 & 1
      p->next = c->next }                Beq r4, 2$         ; i4 si (r4 = 0) saltar a 2$
      p = c;                             Store r5, 8(r0)    ;i5 mem[r0+8] ← r5
      c = c->next;                  2$:  Add r0, r1, #0     ;i6 r0 ← r1
}                                        Add r1,r5,#0       ; i7 r1 ← r5
while (c <> NULL)                        Bne r5, 1$         ;i8 si (r5≠0) saltar a 1$

                                         Valores iniciales:
                                         r0 = dirección del primer elemento de la lista
                                         r1 = dirección del segundo elemento de la lista
```

Pregunta 5: *Indique el ciclo de reloj en que cada instrucción de una iteración y la primera de la siguiente iteración ocupa la etapa ES. Considere los dos casos en función de la condición (c->dat % 2) != 0.*

Pregunta 6: *Sea p la probabilidad de que se cumpla la condición (c->dat % 2) != 0. Calcule el CPI medio en función de p.*

Ejercicio 3.16

El camino de datos de un procesador con frecuencia de reloj de 2 GHz, segmentado en 6 etapas, dispone de un único puerto para acceder a la memoria de instrucciones y de datos (MID), tal como se muestra en la figura.



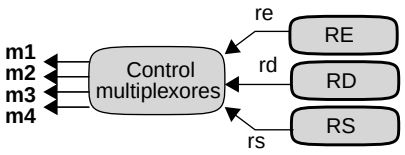
Para contestar a las preguntas 1) y 2) suponga que los riesgos estructurales se pueden resolver retardando el inicio de la interpretación de las instrucciones. También, supondremos que no hay riesgos de datos y de secuenciamiento entre las instrucciones.

Pregunta 1: *Analice los posibles riesgos estructurales durante la interpretación de una secuencia de instrucciones.*
Indique la latencia de inicio prohibida.

Pregunta 2: *Deduzca una secuencia de instrucciones periódica (i1, i2, i3, i4) e ilimitada que produzca un rendimiento de 1600 MIPS.*

En este procesador, la latencia efectiva de las instrucciones que actualizan el banco de registros es de 3 ciclos; la latencia efectiva de las instrucciones que modifican el secuenciamiento implícito es de 5 ciclos.

La figura muestra el esquema de la lógica para controlar los riesgos.



El módulo RE detecta los riesgos estructurales debidos al único camino de acceso a memoria. El módulo RD detecta riesgos de datos y el módulo RS detecta riesgos de secuenciamiento.

Para resolver un conflicto estructural, el control de la segmentación bloquea las etapas B y CP durante un ciclo de reloj e inyecta una NOP en la salida de la etapa B.

Pregunta 3: *Indique el número de ciclos que se puede conocer anticipadamente un riesgo estructural debido al único camino de acceso a memoria.*
Diseñe el módulo detector de riesgos estructurales RE.

Pregunta 4: *Muestre el cronograma de interpretación de la secuencia de instrucciones independientes*
Load, Load, Load, Add.

Cuando se produce un riesgo de datos, el control bloquea las etapas CP, B y DL; durante los ciclos de bloqueo se inyectan instrucciones NOP decodificadas en la salida de la etapa DL.

Para resolver un riesgo de secuenciamiento, el control descarta las instrucciones buscadas hasta que se actualiza el contador de programa con la dirección de la siguiente instrucción que debe interpretarse.

Pregunta 5: *Muestre los cronogramas de interpretación de las 2 secuencias de instrucciones:*

1ª secuencia	2ª secuencia
`Load r2,0(r0)`	`Load r0,8(r0)`
`Add r3,r3,#1`	`Bne r0,for`
`Add r1,r1,r2`	`Load r2,0(r0)`
`Load r0,8(r0)`	

Pregunta 6: *Diseñe la lógica de control de los multiplexores m1, m2, m3 y m4.*

Este procesador ejecuta el siguiente programa de prueba:

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

```
1$:  Load r2, 0(r0)     ;r2 ← mem[r0+0]
     Add r3, r3, #1     ;r3 ← r3 + 1
     Add r1, r1, r2     ;r1 ← r1 + r2
     Load r0, 8(r0)     ;r0 ← mem[r0+8]
     Bne r0, 1$         ;si (r0≠0) saltar a 1$

     Valores iniciales:
     r0 = dirección del primer elemento de la lista
     r1 = 0; r3 = 0
```

Pregunta 7: *Calcule el número de ciclos para ejecutar una iteración, indicando los ciclos perdidos por riesgos de datos, secuenciamiento y estructurales.*
Calcule el rendimiento del procesador en MIPS al ejecutar el programa de prueba.

Ejercicio 3.17

Un procesador, que interpreta las siguientes instrucciones

INT:	Rc = Ra op Rb
Load:	Rc = Mem[Rb + extsgn(literal)]
Store:	Mem[Rb + extsgn(literal)] = Ra
Br:	if cond (Ra) then CP = CP + extsgn(literal); else 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 y detección de riesgos, lectura de operandos en registros
ALU	cálculo aritmético-lógico (INT), dirección efectiva (Load/Store), evaluación de la condición (Br)
M	acceso a memoria de datos (Load/Store)
ES	escritura del resultado en el banco de registros (INT, Load) y el contador de programa (Br)

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

En la etapa DL se detectan los riesgos de datos y 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 perdura 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 ES).

Este procesador ejecuta el siguiente fragmento de código, que elimina los elementos repetidos de una lista ordenada que contiene al menos dos elementos.

```
for (; p!=NULL; p=p->next)
{
   if (p->dat == q->dat)
   {
      q->seg = p->seg;
      q = p;
   }
}
```

```
1$:  load r11, 0(r1)        ;r11 ← mem[r1+0]
     load r10, 0(r0)        ;r10 ← mem[r0+0]
     cmpeq r12, r11, r10    ;si (r11=r10) r12 ← 1
                            ;en otro caso   r12 ← 0
     beq r12, 2$            ;si (r12=0) saltar a 2$
     load r1, 8(r1)         ;r1 ← mem[r1+8]
     store r1, 8(r0)        ;mem[r0+8] ← r1
     add r0, r1, #0         ;r0 ← r1
2$:  bne r1, 1$             ;si (r1≠0) saltar a 1$
```

donde los valores iniciales de las variable son:

Valores iniciales:

r0 = dirección del primer elemento de la lista
r1 = dirección del segundo elemento de la lista

Pregunta 1: *Muestre el cronograma de ejecución de las instrucciones de una iteración entera del bucle y de la primera instrucción de la siguiente iteración, para cada uno de los resultados de la instrucción beq r12, 2$. Identifique en el cronograma los ciclos perdidos e indique la causa.*

Pregunta 2: *Calcule los ciclos por iteración y el CPI para cada resultado de la instrucción beq r12, 2$.*

Pregunta 3: *Calcule el CPI medio suponiendo que la lista contiene 101 elementos, por tanto el bucle itera 100 veces, y la lista contiene 20 elementos repetidos.*

Pregunta 4: *Calcule la ganancia, suponiendo un procesador serie en el que la latencia de cualquier instrucción es 6 ciclos y suponiendo que el procesador segmentado funciona con una frecuencia de reloj 1.5 X la del procesador serie.*

Ejercicio 3.18

Un procesador segmentado lineal, con las 6 etapas típicas.

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

sin riesgos estructurales, que SI puede escribir y leer un registro en el mismo ciclo, que no tiene ningún cortocircuito, que no hace predicciones y pone el siguiente valor en el CP en el ULTIMO ciclo cuando interpreta una instrucción de salto, interpreta el siguiente código:

```
1   load     P:   r5 <---  M [ r0 + 0 ]
2   load          r6 <---  M [ r1 + 0 ]
3   ent           r7 <--- r5 - r6
4   br            si (r7 = 0) saltar a N
5   load          r4 <---  M [ r1 + 0 ]
6   store         M [ r0 + 8 ] <--- r4
7   ent           r0 <--- r0 + 8
8   ent      N:   r1 <--- r1 + 8
9   br            si  (r1 < n)  saltar a P
```

Suponga que son iguales los contenidos de r5 y de r6, para responder las 3 siguientes preguntas.

Pregunta 1: *Presente el Grafo de Dependencias (GdD si r7 =0), poniendo en cada nodo el número de la instrucción y diferenciando los arcos según el tipo de dependencia (RaW o WaR o WaW).*

Pregunta 2: *Sin presentar el cronograma, calcule el número de ciclos perdidos en una iteración completa (desde empezar a interpretar la instrucción 1 hasta volver a empezar a interpretar la instrucción 1) , indicando cuántos son por RdD y cuántos son por RdS.*

Pregunta 3: *Calcule el valor de la métrica IPC en una iteración completa.*

Suponga que son diferentes los contenidos de r5 y de r6, para responder las 3 siguientes preguntas.

Pregunta 4: *Presente el Grafo de Dependencias (GdD si r7 <>0), poniendo en cada nodo el número de la instrucción y diferenciando los arcos según el tipo de dependencia (RaW o WaR o WaW).*

Pregunta 5: *Sin presentar el cronograma, calcule el número de ciclos perdidos en una iteración completa (desde empezar a interpretar la instrucción 1 hasta volver a empezar a interpretar la instrucción 1) , indicando cuántos son por RdD y cuántos son por RdS.*

Pregunta 6: *Calcule el valor de la métrica IPC en una iteración completa.*

Ejercicio 3.19

Un procesador segmentado lineal, con las 6 etapas típicas.

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

sin riesgos estructurales, que SI puede escribir y leer un registro en el mismo ciclo, que no tiene ningún cortocircuito, que no hace predicciones y pone el siguiente valor en el CP en el PENULTIMO ciclo cuando interpreta una instrucción de salto, interpreta el siguiente código:

```
1    load     P:   r1 <---  M [ r2 + 4 ]
2    br            si (r1 >3) saltar a S
3    load          r6 <---  M [ r1 + 4 ]
4    ent      S:   r7 <--- r6 + r1
5    load          r8 <---  M [ r7 + 4 ]
6    ent           r9 <--- r1 + r8
7    ent           r2 <--- r2 + 1
8    br            si  (r2 < 10)  saltar a P
```

Sabemos que , antes de entrar en este bucle, r2 contiene el valor cero y M [v] contiene el valor v - 3 .

Pregunta 1: *Presente el cronograma de la 1ª iteración y las tres primeras instrucciones de la 2ª iteración , sabiendo que todo RdD y todo RdS produce bloqueo durante los ciclos que permanece el riesgo.*

Pregunta 2: *Calcule el número de ciclos perdidos en una iteración completa (desde empezar a interpretar la instrucción 1 hasta volver a empezar a interpretar la instrucción 1) , indicando cuántos son por RdD y cuántos son por RdS.*

Pregunta 3: *Calcule el valor de la métrica CPI cuando se hayan completado TODAS las iteraciones*

Pregunta 4: *El procesador tiene f = 170 MHz y Pot = 100 w. Cuando completa la ejecución de todas las iteraciones ha consumido el 0.01 % de la energía de su batería. Sabiendo que la tensión de alimentación es de 0.1 v. , calcule la carga total de la batería, expresada en A · s .*

Pregunta 5: *Suponga que las instrucciones de acceso a memoria tienen CPI = 4.8 . Aplicamos una mejora de diseño que consigue que ese valor se reduzca a 3.6 . No hay ninguna otra modificación en el procesador. Calcule la Ganancia obtenida con esa mejora, cuando se ejecuta completamente el código anterior (todas las iteraciones del bucle P).*

Ejercicio 3.20

Sabiendo estos valores iniciales r3 = r4 = 100 ; r5 = r6 = 0 ; M[100] = M[104] = 1; considere el código siguiente:

```
1   load     B:   r1 <---  M [r3 + 0]
2   load          r2 <---  M [r3 + 4]
3   ent           r5 <---  r5 + r1
4   ent           r6 <---  r6 + r2
5   ent           r5 <---  r5 + r6
6   store         M [r3 + 0]  <--- r4
7   brc           si (r5 = 0) saltar a B
8   store         M [r3 + 8]  <--- r5
```

Un procesador segmentado lineal (PSL) tiene las seis etapas.

ETAPA	FUNCIONALIDAD
CP	determinar la dirección de la instrucción
B	búsqueda de la instrucción
DL	decodificación y detección de riesgos, lectura de operandos en registros
A	opera con enteros y evalúa condición de salto
M	acceso a memoria de datos, escribe CP según salto condicional
ES	escritura resultado en registro, al final del tiempo de ciclo

Las etapas del procesador tienen recursos suficientes para evitar cualquier Riesgo Estructural. Cualquier otro Riesgo se detecta en la 3ª etapa, y se trata durante los ciclos que dure, con los mecanismos estudiados en el tema 3.

Pregunta 1: *Presente el Grafo de Dependencias de Datos (GdD) que muestre , solamente, las que son del tipo RaW .Identifique cada nodo del grafo con el número (del 1 al 8) que tiene la instrucción del código antes considerado.*

Pregunta 2: *Presente otro GdD que muestre todas las demás dependencias de datos que tiene este código.*

Pregunta 3: *Calcule el número total de ciclos que el PSL emplea en interpretar este código. Como es un bucle, tenga en cuenta la cantidad de iteraciones efectuadas, que debe calcular. Presente el cronograma de ejecución de una iteración completa.*

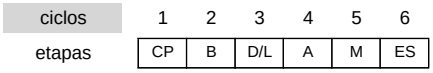
Pregunta 4: *Calcule cuantos ciclos se pierden cuando el PSL interpreta este código. Presente la respuesta separando la cantidad perdida por riesgos de secuenciamiento (RdS) y por riesgos de datos (RdD), e identificando por su número (del 1 al 8) la instrucción en la que se produce cada pérdida.*

Pregunta 5: *Si el PSL tiene f = 9.41 GHz, calcule el valor de la f (en MHz) de un procesador serie no segmentado (PSNS) que tarde en interpretar este mismo código el triple de tiempo que el PSL, sabiendo que el PSNS ocupa 6 ciclos para cada instrucción load o store, y cinco ciclos para cada instrucción ent o brc.*

Pregunta 6: *Un nuevo procesador segmentado lineal (PSL2) se diferencia del PSL solamente en que realiza la acción "escribe CP según salto condicional" en la 4ª etapa (A) y no en la 5ª (M). Calcule la ganancia, expresada en porcentaje, que obtiene PSL2 sobre PSL al interpretar este código.*

Ejercicio3.21

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 se describe en la siguiente tabla.

ETAPA	FUNCIONALIDAD
CP	determinar el CP.
B	búsqueda de la instrucción.
D/L	decodificación, lectura de registros (2 caminos de lectura).
A	operación aritmético-lógica.
M	acceso a memoria de datos.
ES	escritura en el banco de registros (1 camino de escritura).

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. Las instrucciones de secuenciamiento actualizan el registro CP en la etapa ES. Un riesgo se gestiona en la fase de decodificación, que en el procesador descrito previamente se corresponde con la etapa D/L.

Suponga la siguiente secuencia de instrucciones.

dirección	instrucción
100	load **R1**, 0(R2)
104	add **R3**, R1, R3
108	add **R2**, R2, #8

dirección	instrucción
112	sub **R4**, R6, R5
116	beq R4, 100

Pregunta 1: *Dibuje un diagrama temporal que muestre la interpretación de una iteración del bucle y la 1ª instrucción de la siguiente iteración. En el diagrama temporal debe mostrarse en la columna de la izquierda, en cada fila, la dirección en la entrada del registro de desacoplo CP (la dirección que se utilizaría en el siguiente ciclo para efectuar la búsqueda). También debe mostrarse la inyección de instrucciones nop cuando se gestiona un riesgo. Indique los ciclos perdidos por cada tipo de riesgo y calcule el CPI de una iteración.*

En una nueva versión del procesador se incrementa la frecuencia de funcionamiento y ello da lugar a que se incremente la latencia de interpretación. En la siguiente figura se muestra la segmentación en etapas del proceso de interpretación de las instrucciones en el nuevo procesador segmentado.

ciclos	1	2	3	4	5	6	7	8	9	10
etapas	CP	B1	B2	D/L1	L2	A	M1	M2	E1	E2

Los accesos a memoria y al banco de registros se han segmentado en 2 etapas. La lectura de un registro del banco de registros se efectúa en la etapa D/L1, utilizándose la etapa L2 para encaminar los datos. La escritura en el banco de registros se efectúa en la etapa E2, utilizándose la etapa E1 para determinar el registro que quiere escribirse (decodificación del identificador de registro). El número de caminos al banco de registros del nuevo procesador es el mismo que en el procesador original. Igualmente en un ciclo se puede efectuar una acción de escritura y una de lectura, en este orden, sobre un registro del banco de registros. En cuanto al acceso a la memoria de datos supondremos que la acción propiamente dicha de lectura o escritura en memoria se efectúa en el ciclo 8. Las instrucciones de secuenciamiento actualizan el registro CP en la etapa E2. Por último, un riesgo se gestiona en la etapa D/L1.

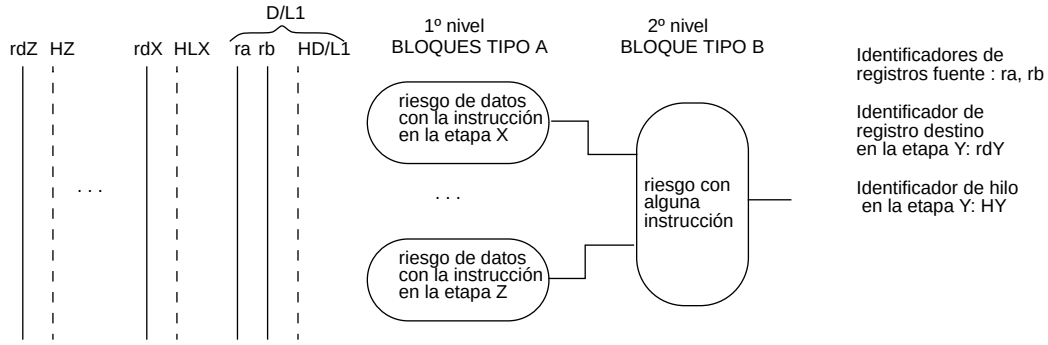
Un mecanismo utilizado para aumentar la productividad de un procesador es utilizar la técnica multihilo de grano fino. Esto es, se entrelaza la interpretación de instrucciones de varios hilos con el objetivo de reducir los ciclos perdidos por riesgos. Para ello se replica el registro contador de programa y el banco de registros tantas veces como hilos se quieran interpretar concurrentemente. Otros recursos del procesador como la memoria, tanto de instrucciones como de datos, o la ALU se comparten.

Suponga que el nuevo procesador segmentado, especificado previamente, tiene la funcionalidad multihilo y se puede iniciar en cada ciclo la interpretación de una instrucción de un hilo distinto al del ciclo previo.

Pregunta 2: *¿Cuál debe ser el número mínimo de hilos que deben interpretarse concurrentemente para que no se detecte ningún tipo de riesgo entre instrucciones del mismo hilo.*

En el procesador segmentado multihilo, además de la información que se transmite entre etapas en un procesador segmentado sin esta funcionalidad, se transmite información sobre el hilo al que pertenece la instrucción que ocupa la etapa. Para identificar esta información utilizaremos el acrónimo H y como sufijo el acrónimo de la etapa.

En la siguiente figura se muestra a nivel de bloques la lógica que se utiliza en el procesador multihilo para determinar riesgos de datos. Tenemos un 1º nivel de bloques donde se detecta la posibilidad de riesgo de datos de una instrucción en la etapa X (productora) con una instrucción en la etapa D/L1. En el 2º nivel tenemos un bloque que indica si en alguno de los bloques de tipo A se ha detectado riesgo de datos.



Pregunta 3: *¿Cuántos bloque del tipo A son necesarios en el procesador segmentado multihilo?.*

Pregunta 4: *Suponga que el número de bits utilizados como identificador de registro y para identificar un hilo es el mismo. Muestre la lógica dentro del bloque A para detectar la posibilidad de riesgo entre una instrucción que ocupa la etapa ALU y una instrucción que ocupa la etapa D/L1. Para ello dispone de módulos*

comparadores de 2 entradas (2C), que indican si existe coincidencia entre las 2 entradas, y de puertas lógicas. Así mismo muestre la lógica utilizada en el bloque B. En el diseño de la lógica no tendremos en cuenta las señales de validez. Las instrucciones pueden tener como máximo un registro destino y también, como máximo, dos registros fuente.

Para no incrementar el tiempo de ejecución de un hilo se toma la decisión de que se cambia de hilo cuando se detecta un riesgo de datos o de secuenciamiento. Para ello, cuando se detecta un riesgo en la etapa D/L1 se indica a la etapa CP que seleccione la dirección de otro hilo. En los dos casos hay que descartar algunas instrucciones que se están interpretando.

En la contestación de las siguientes preguntas supondremos un número ilimitado de hilos en el procesador.

Pregunta 5: *Para el caso de detección de un riesgo de datos e independientemente del número de ciclos que han transcurrido desde que la instrucción productora ha estado en la etapa D/L1, ¿cuántas instrucciones hay que descartar?.*

Pregunta 6: *Respecto a un procesador segmentado que no utiliza la técnica multihilo, ¿en cuánto se reducen o incrementan los ciclos perdidos si la instrucción consumidora está a una distancia D = {1, 2, 3, 4, 5, 6} de la instrucción productora?. En este contexto, entendemos por distancia el número de ciclos transcurridos desde que la instrucción productora está en la etapa D/L1 hasta el ciclo en que la instrucción consumidora está en la etapa D/L1.*

Pregunta 7: *Para el caso de detección de un riesgo de secuenciamiento, ¿cuántas instrucciones hay que descartar?.*

Pregunta 8: *Respecto de un procesador segmentado que no utiliza la técnica multihilo, ¿en cuánto se reducen los ciclos perdidos por riesgos de secuenciamiento cuando se cambia de hilo al detectar un riesgo de secuenciamiento?.*

Ejercicio 3.22

Disponemos de un procesador segmentado linealmente en 6 etapas.

ciclos	1	2	3	4	5	6
etapas	CP	B	D/L	A	M	ES

La funcionalidad de las etapas se describe en la siguiente tabla.

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.
A	operación aritmético-lógica, o cálculo de la dirección efectiva o evaluación de la condición.
M	acceso a memoria de datos.
ES	escritura en el banco de registros o en CP.

El camino de datos dispone de los recursos necesarios para que no se produzcan riesgos estructurales. Un mismo registro del banco de registros se puede escribir y leer, en este orden, en el mismo ciclo de reloj.

En la etapa DL se detectan los riesgos de datos y secuenciamiento. Cuando se detecta un riesgo de datos, la lógica de control impide el progreso de las instrucciones que ocupan las etapas DL, B y CP e inyecta NOPs hacia la etapa ALU mientras perdura el riesgo. En caso de riesgo de secuenciamiento, la lógica de control descarta las instrucciones buscadas, inyectando NOPs hacia la etapa DL, hasta que se actualiza el Contador de Programa con la dirección de la siguiente instrucción que debe interpretarse (etapa E).

El procesador ejecuta el siguiente código, del cual desconocemos algunos identificadores de registro, con un CPI=2.

```
100  Load r1,0(r0)    i1
104  Add r0,ri,#8     i2
108  Sub r2,r2,#1     i3
112  Add rj,r1,r3     i4
116  Bne rk,100       i5
```

Pregunta 1: *Deduzca los identificadores de registro ri, rj, rk. Sólo se pueden utilizar los registros r0, r1, r2 y r3.*

Pregunta 2: *Dibuje el grafo de dependencias de datos.*

Pregunta 3: *Dibuje el cronograma simplificado de ejecución de 8 instrucciones (una iteración completa y las 3 primeras instrucciones de la siguiente iteración). Determine el número de ciclos de ejecución por iteración.*

Pregunta 4: *Indique qué etapas están procesando NOPs inyectadas por la lógica de control en los ciclos 6 y 15.*

El tiempo de ciclo del procesador es de 3 ns. El procesador consume 100nJ en cada ciclo que las 6 etapas están procesando información válida. En cambio, en los ciclos en que M etapas están procesando NOPs inyectadas por la lógica de control, el procesador consume (100 - 10•M) nJ.

Pregunta 5: *Calcule la potencia media consumida durante la ejecución del programa. Suponga que el número de iteraciones ilimitado.*

Ejercicio 3.23

Disponemos de un procesador segmentado linealmente en 6 etapas.

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.
A	operación aritmético-lógica, o cálculo de la dirección efectiva o evaluación de la condición.
M	acceso a memoria de datos.
ES	escritura en el banco de registros o en el registro CP.

El camino de datos dispone de los recursos necesarios para que no se produzcan riesgos estructurales. Un mismo registro del banco de registros se puede escribir y leer, en este orden, en el mismo ciclo de reloj.

En la etapa DL se detectan los riesgos de datos y secuenciamiento. Cuando se detecta un riesgo de datos, la lógica de control impide el progreso de las instrucciones que ocuapn las etapas DL, B y CP e inyecta NOPs hacia la etapa ALU mientras perdura el riesgo. En caso de riesgo de secuenciamiento, la lógica de control descarta las instrucciones buscadas, inyectando NOPs hacia la etapa DL, hasta que se actualiza el Contador de Programa con la dirección de la siguiente instrucción que debe interpretarse (etapa E).

La cache de instrucciones (MI) no es ideal. Esto es, se producen fallos de cache al ir a buscar instrucciones. La MI dispone de un puerto de lectura, para suministrar una instrucción al procesador, y un puerto de escritura, para escribir el bloque transferido desde el siguiente nivel de la jerarquía de memoria en caso de fallo. El acceso al siguiente nivel de la jerarquía está segmentado con una latencia de inicio de 1 ciclo. La MI puede soportar un número ilimitado de fallos.

En la siguiente figura se muestra la interpretación de una instrucción que detecta un fallo en MI.

		ciclo									
Dirección	Instrucción	1	**2**	**3**	**4**	5	6	7	8	9	10
124	`Add r0,r1,r2`	CP	B	nop	nop	nop	nop				
				B	nop	nop	nop	nop			
					B	nop	nop	nop	nop		
						B	DL	ALU	M	E	
128	`Sub r5,r6,r7`		CP								
128				CP							
128					CP						
128						CP	B	DL	ALU	M	E

En el 2º ciclo, la instrucción Add detecta fallo en MI y se inicia el acceso al siguiente nivel de la jerarquía de memoria. En el 3º y 4º ciclo se repite el acceso y la cache indica repetidamente fallo. En el 5º ciclo se actualiza el contenedor de MI con el bloque transferido desde el siguiente nivel de la jerarquía de memoria, se accede a MI (acierto) y se suministra la instrucción. Observe que en los ciclos que se detecta fallo la lógica de control bloquea la interpretación de las instrucciones que ocupan las etapas B y CP e inyecta NOPs hacia la etapa DL. La penalización por fallo es de 3 ciclos.

En las siguientes preguntas se han de analizar algunas de las interacciones entre los riesgos de datos, riesgos de secuenciamiento y fallos en MI. En todos los casos las respuestas se justificarán mediante un cronograma que muestre la actuación de la lógica de control. En las situaciones donde se detecten concurrentemente diversos tipos de riesgos se habrá de indicar como se priorizan.

Pregunta 1: *Calcule el número total de ciclos perdidos cuando el procesador ejecuta la siguiente secuencia de instrucciones.*

```
124  Load r1,0(r0)
128  Add r3,r1,r3
132  Sub r2,r2,#1
```

donde la instrucción Add detecta fallo en MI.

Pregunta 2: *Calcule el número total de ciclos perdidos cuando el procesador ejecuta la siguiente secuencia de instrucciones.*

```
120  Load r1,0(r0)
124  Add r3,r1,r3
128  Sub r2,r2,#1
132  Store r5,8(r0)
```

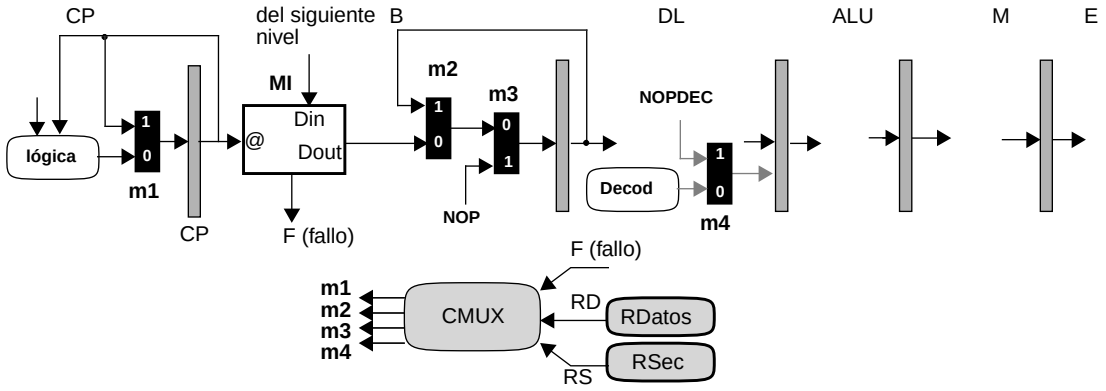
donde la instrucción Sub detecta fallo en MI.

Pregunta 3: *Calcule el número total de ciclos perdidos cuando el procesador ejecuta la siguiente secuencia de instrucciones.*

124	Bne r1, 200
128	Add r0, r0, #8
132	Sub r2,r2,#1
136	Load r1,0(r1)
140	Load r3,8(r0)

Suponga que no se cumple la condición en la instrucción de salto y la instrucción Add detecta fallo en MI.

En la siguiente figura se muestra un esquema simplificado del camino de datos y de la lógica de bloqueo. El módulo RDatos detecta riesgos de datos mientras que el módulo RSsec detecta riesgos de secuenciamiento. El módulo CMUX genera las señales de control de los multiplexores que retienen las instrucciones o inyectan NOPs en los registros de desacoplo. La señal F es la salida del comparador de etiquetas de MI y se activa (F = 1) en los ciclos en que se detectan fallos.



Pregunta 4: *Diseñe la tabla de verdad del módulo CMUX.*