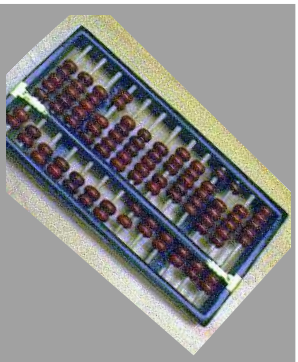
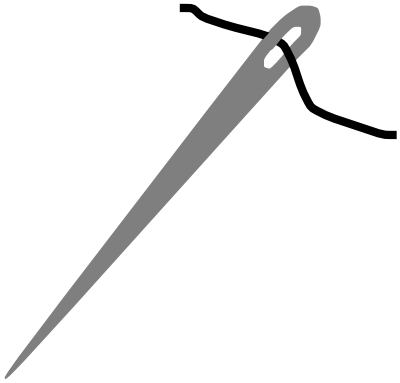


Arquitectura de Computadores



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

J.M. Llabería
E. Herrada
A. Olivé

Contenidos

Capítulo 2 Segmentación y replicación

2

SEGMENTACIÓN Y REPLICACIÓN

Un dispositivo serie no empieza a procesar una nueva tarea hasta que ha finalizado la tarea previa. Entonces, un objetivo en el diseño de un dispositivo serie es reducir el tiempo de procesado de una tarea.

La técnica de segmentación permite que un dispositivo empiece a procesar nuevas tareas antes de haber finalizado el procesado de las tareas previas. La segmentación tiene como objetivo incrementar el número de tareas procesadas por unidad de tiempo, en lugar de reducir el tiempo de una tarea individual.

La segmentación se sustenta en que el número de tareas que se quiere procesar es muy grande y lo que se reduce es el intervalo de tiempo entre tareas finalizadas. Por otro lado, cuando se aplica a dispositivos lógicos, el coste hardware que requiere la segmentación es pequeño y el incremento de rendimiento es muy significativo.

La técnica de replicación utiliza varias réplicas de un dispositivo serie. Entonces, cada réplica puede utilizarse para procesar una tarea distinta. A semejanza de la técnica de segmentación su objetivo es incrementar el número de tareas procesadas por unidad de tiempo. Sin embargo, el coste hardware de la replicación es significativo ya que se replica completamente el dispositivo serie.

Contenido

INTRODUCCIÓN

En este capítulo nos centramos en segmentaciones y replicaciones idealizadas, siendo una de las hipótesis que las tareas, de la secuencia de tareas que se procesa, son independientes entre sí. Esto es, una tarea no utiliza como dato el resultado calculado por una tarea previa.

La efectividad de las técnicas de segmentación y replicación se ilustra utilizando un sumador por su simplicidad.

Posteriormente se describe la segmentación del proceso de interpretación de instrucciones, centrándose exclusivamente en los recursos necesarios. La exposición detallada de un procesador segmentado, con su complejidad, se efectúa en los dos siguientes capítulos. Así mismo, se describe la segmentación del acceso a cache. También exclusivamente desde el punto de vista de recursos necesarios.

Para discutir y analizar el diseño de un dispositivo debe conocerse como funciona la lógica que lo implementa. Por ello, también se revisan algunas ideas básicas de lógica digital que se utilizan extensamente.

DISEÑO SERIE

Como ejemplo se utiliza un sumador de vectores de bits. En primer lugar se desarrolla un sumador de 1 bit, posteriormente se utilizan varios sumadores de 1 bit para construir un sumador de vectores de bits. Por último se muestra un dispositivo sumador serie que efectúa la suma de una secuencia de pares de números naturales representados mediante vectores de bits.

Sumador de 1 bit

La expresión algebraica correspondiente a la suma de bits con la misma ponderación es la siguiente:

$$x + y + c = 2 \times c_1 + s$$

El resultado son dos bits, uno de ellos es el bit denominado suma (s) y el otro bit se denomina acarreo de salida (c_1).

Expresiones lógicas. A partir de las entradas *(x, y)* y de un *acarreo* de entrada *(c_{en})*, unas expresiones lógicas para la suma (*s*) y el *acarreo* de salida (c_{sal}) son:

$$s = \langle x \oplus y \rangle \oplus c_{en}$$

$$c_{sal} = x \cdot y + c_{en} \cdot y + c_{en} \cdot x$$

Esquema lógico del sumador. El número de niveles de puertas del sumador depende del tipo de puertas disponible y del número de entradas de las puertas. Disponiendo de puertas AND, OR y XOR con 2 y 3 entradas, un circuito lógico que implementa el sumador se muestra en la Figura 2.1.

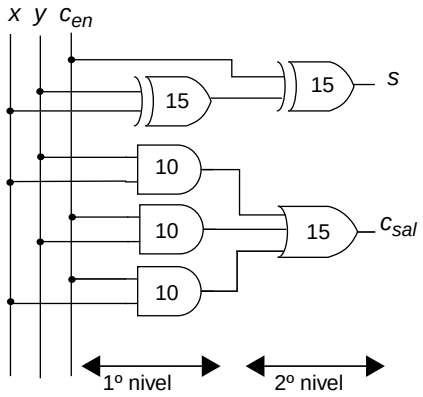


Figura 2.1 *Sumador de 1 bit. En cada puerta se especifica el retardo.*

Retardo del circuito Es el número de unidades de tiempo (u.t.), en el peor caso, que tarda en estabilizarse la última de las señales de salida, a partir del instante en que se han estabilizado todas las señales de entrada.

Supondremos que una puerta AND tiene un retardo de 10 u.t., una puerta XOR tiene un retardo de 15 u.t. y una puerta OR-3 tiene un retardo de 15 u.t.

El retardo del camino que calcula la señal *s* son 30 u.t. y el retardo del camino que calcula la señal c_{out} son 25 u.t. Por tanto, el retardo son 30 u.t.

Sumador de vectores de bits

Para representar un número natural (x) se utiliza un vector de bits $X = (x_{n-1}, \ldots, x_1, x_0)$ que se interpreta de forma ponderada:

$$\underline{x} = \sum_{i=0}^{n-1} x_i \times 2^i$$

donde $\underline{x}$ es valor numérico, que se calcula como la suma ponderada de los bits del vector de bits.

Expresiones algebraicas Dados los vectores de bit *(X, Y)* de entrada

$$X = (x_{n-1}, \ldots, x_1, x_0) \qquad Y = (y_{n-1}, \ldots, y_1, y_0)$$

que representan los números naturales x e y respectivamente, la operación suma se expresa como $\underline{s} = (\underline{x} + \underline{y}) mod 2^n$. El resultado s se representa mediante un vector de bits $S = (s_{n-1}, \ldots, s_1, s_0)$.

La suma de los vectores *X* e *Y* se efectúa sumando bit a bit los vectores de bits y propagando el acarreo.

$$x_i + y_i + c_i = 2 \times c_{i+1} + s_i \qquad 0 \le i < n$$

siendo $c_0 = 0$.

Esquema lógico del sumador En la Figura 2.2 se muestra el esquema de conexionado de 2 sumadores de 1 bit para construir un sumador de 2 bits con propagación serie del acarreo. El resultado es un vector de bits de tamaño 2 (s_1, s_0) y el bit c_2. El bit c_2 se utiliza para determinar situaciones de irrepresentabilidad.

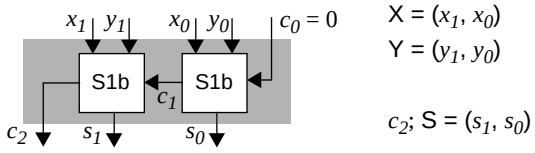


Figura 2.2 *Sumador de 2 bits con propagación serie del acarreo. S1b es un sumador de 1bit.*

El retardo del sumador de vectores de bits está determinado por la propagación serie del acarreo. Se tardan 30 u.t. en calcular la señal s_0 y 25 u.t. en calcular la señal c_1. En paralelo se ha calculado la señal $x_1 \oplus y_1$. Entonces, el cálculo de la señal s_1 experimenta el retardo de una puerta XOR (Figura 2.1). La señal c_2 experimenta un retardo de 25 u.t., después de los 25 u.t. necesarios para determinar c_1, lo cual es un retardo total de 50 u.t. Por tanto, el retardo del sumador de 2 bits es 50 u.t.

Módulo sumador

Un módulo sumador secuencial es un dispositivo que procesa secuencias de sumas de pares de números. El módulo tiene elementos de almacenamiento en la entrada y el sumador procesa las señales que hay en la salida de los elementos de almacenamiento. En la Figura 2.3 se muestra un módulo sumador. Los elementos de almacenamiento se representan mediante rectángulos con trama y no se muestra la entrada alimentada por la señal de reloj.

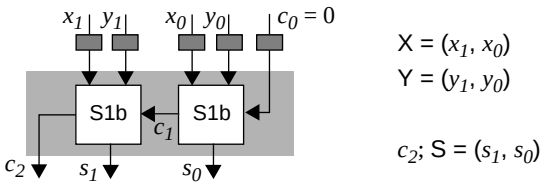


Figura 2.3 *Módulo sumador de vectores de 2 bits.*

Elemento de almacenamiento. Es un circuito secuencial que mantiene un estado. El elemento básico tiene dos entradas y una salida. Las entradas son el valor que se quiere almacenar y la señal de reloj. Esta última señal determina el instante en el cual se almacena el valor de la entrada.

Reloj. Es un dispositivo que produce una señal repetitiva de forma permanente y se caracteriza por el intervalo de tiempo de una repetición. A este intervalo de tiempo se le denomina periodo o tiempo de ciclo.

En un periodo de la señal de reloj (Figura 2.4) se distinguen dos subintervalos de tiempo consecutivos. En uno de ellos el nivel lógico es el cero y en el otro subintervalo de tiempo el nivel lógico es 1.

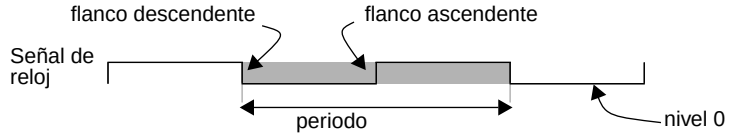


Figura 2.4 *Señal de reloj.*

El cambio de nivel lógico en la señal de reloj se denomina flanco y supondremos que es instantáneo. El flanco se denomina ascendente cuando se pasa de nivel 0 a nivel 1 y descendente en caso contrario.

Forma o aspecto de la señal de reloj. La forma de la señal de reloj se relaciona con la duración de los subintervalos de tiempo en que permanece en el nivel lógico 0 y en el nivel lógico 1 (Figura 2.5). Cuando los dos subintervalos de tiempos son de la misma duración se denomina una señal de reloj cuadrada. En caso contrario se denomina rectangular.

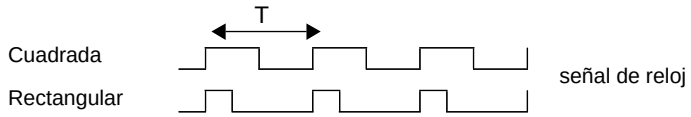


Figura 2.5 *Aspecto de la señal de reloj.*

El módulo sumador es un sistema secuencial y el funcionamiento es síncrono. La decisión del instante en el cual se actualiza el estado de los elementos de almacenamiento se denomina metodología de reloj.

Metodología de reloj. Define cuando los elementos de memorización pueden leerse y cuando pueden escribirse.

Es importante distinguir los instantes en que se lee un elemento de almacenamiento de los instantes en que se escribe. Si las dos acciones se efectuan de forma concurrente, el valor de la lectura puede ser el valor antiguo, el que se está escribiendo o una mezcla de ambos. Esta impredecibilidad no es tolerable y la metodología de reloj se diseña para prevenir estas circunstancia.

Nosotros supondremos una metodología de reloj que, para actualizar los elementos de memorización de un circuito secuencial, utiliza uno de los instantes de cambio de nivel lógico en la señal de reloj. En concreto, utilizaremos el flanco ascendente de la señal de reloj.

Registro

Un registro es un elemento básico de almacenamiento que actualiza su estado en el flanco ascendente (o descendente) de la señal de reloj. Un registro elemental almacena un bit. Entonces, para construir un registro de n bits se agrupan n registros elementales y se accede a ellos como una unidad. Esto es, cuando se accede al registro se leen o escriben los n bits.

En la Figura 2.6 la etiqueta D se corresponde con la entrada de datos del registro y la etiqueta Q se corresponde con la salida del registro. La señal de reloj se utiliza para sincronizar el instante de actualización del registro. Para simplificar los dibujos se representa usualmente un registro de 1bit (rebanada de 1 bit) y la parte asociada de la lógica combinacional. Si es necesario para la comprensión de la exposición se detallarán los n bits.

Lectura. La salida del registro (Q) puede utilizarse en cualquier instante de tiempo como entrada de un circuito combinacional y esta acción no modifica el estado del registro.

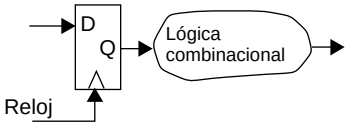


Figura 2.6 *Registro.*

Escritura. El valor de la señal de entrada (D) en el flanco ascendente de la señal de reloj se observa en la salida (Q) después de un retardo de propagación de la señal (t_p), que es función de los dispositivos lógicos utilizados para construir el registro.

La Figura 2.7 muestra la evolución temporal de las señales en una acción de escritura en un registro. Previo al flanco ascendente de la señal de reloj suponemos que el registro almacena el valor 0.

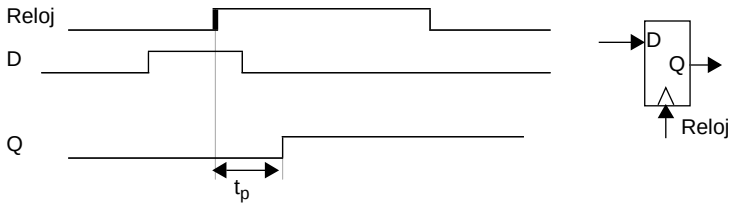


Figura 2.7 *Retardo de propagación de un registro.*

Retardo de propagación (t_p). Es el intervalo de tiempo desde el flanco ascendente de la señal de reloj hasta la estabilización del nuevo estado del registro. Como en el caso de puertas lógicas, el retardo de propagación depende de la carga (número de entradas de puertas lógicas) conectada a la salida del registro.

Funcionamiento

En un dispositivo serie hay que esperar a que finalice una tarea antes de iniciar una nueva tarea.

En la Figura 2.8 se muestra, mediante un diagrama temporal, la utilización del módulo sumador cuando se efectúa la suma de cuatro pares de vectores de bits (A+B, C+D, E+F, G+H). El inicio de cada operación se sincroniza con una señal de reloj cuyo periodo es T_{serie}.

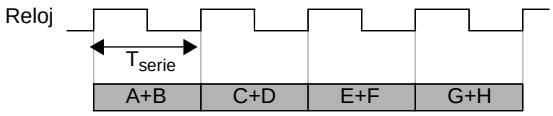


Figura 2.8 *Diagrama temporal del procesado de varias sumas en un sumador serie.*

El tiempo para procesar una secuencia de n tareas en un dispositivo serie es n x T_{serie}, donde T_{serie} es el periodo de la señal de reloj.

Productividad. Evalúa el número de operaciones por unidad de tiempo. Es el cociente entre el número de operaciones y el tiempo que tardan en efectuarse.

En un dispositivo serie, el número de tareas efectuadas por unidad de tiempo es $1/T_{serie}$.

Restricciones en el tiempo de ciclo

En un circuito secuencial se distinguen elementos de memorización y lógica combinacional. En la Figura 2.9 se muestra un esquema simplificado de un dispositivo que procesa una secuencia de tareas de forma serie, el periodo (T) de la señal de reloj y los retardos de los elementos.

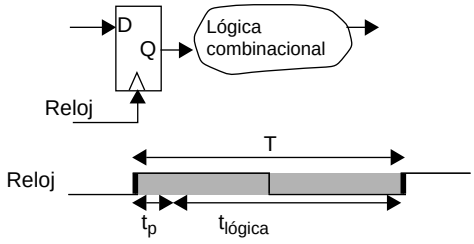


Figura 2.9 *Restricción en el periodo de la señal de reloj.*

Los elementos de almacenamiento actualizan el estado en el flanco ascendente de la señal de reloj y la salida (Q) es estable después de un retardo.

Antes de iniciar una nueva tarea, la lógica combinacional, una vez son estables las señales de entrada, debe disponer de tiempo suficiente para obtener señales estables en sus salidas. Esta restricción establece un valor mínimo al periodo de la señal de reloj.

El tiempo mínimo de ciclo (periodo) T, necesario para un funcionamiento correcto de un dispositivo serie, está expresado por la siguiente relación.

$$T \geq t_p + t_{\log ica}$$

donde t_p es el tiempo de propagación del registro y t_{logica} es el tiempo de retardo de la lógica combinacional.

El módulo sumador de 2 bits, suponiendo que el retardo de un registro es 2 u.t., requiere que el periodo de la señal de reloj sea mayor o igual que 52 u.t. Entonces, el número de operaciones efectuadas por unidad de tiempo es 1/52 = 0.02.

SEGMENTACIÓN

La técnica de segmentación permite diseñar un dispositivo que empieza a procesar una nueva tarea antes de haber finalizado el procesado de la tarea previa.

El primer paso para segmentar una tarea es distinguir una secuencia de fases. Seguidamente hay que identificar los módulos combinacionales que efectúan cada una de las fases o grupos de ellas. Posteriormente hay que establecer un secuenciamiento en la utilización de los módulos combinacionales que de lugar a la tarea.

Como ejemplo conductor de la exposición utilizaremos el módulo sumador de vectores de 2 bits. En este caso, hay dos fases y en cada una de ellas se suma un bit. Por tanto, necesitamos dos sumadores de 1 bit que denominaremos E0 y E1. En la Figura 2.10 se muestra la conexión lógica de los módulos para el caso del sumador de 2 bits.

En el módulo E0 se efectúa la suma de los dos bits menos significativos (x_0, y_0) y c_0. Una vez se conoce el bit de suma y el bit de acarreo (c_1, s_0), en el módulo E1 se efectúa la suma de los siguientes bits (x_1, y_1), junto con el acarreo de salida de la suma de los bits menos significativos (c_1) y se obtiene el bit de suma s_1 y el bit c2.

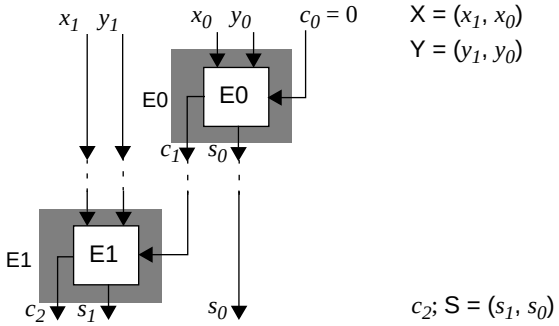


Figura 2.10 *Conexión lógica de sumadores de 1 bit (E0, E1) para obtener un sumador de vectores de 2 bits.*

Una vez identificados los módulos y las conexiones lógicas se insertan registros, denominados de desacoplo, entre las conexiones lógicas establecidas entre los módulos (Figura 2.11).

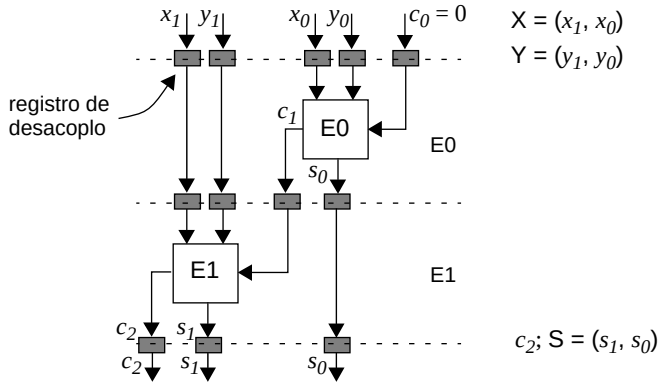


Figura 2.11 *Sumador de vectores de 2 bits segmentado en dos etapas.*

Etapa en un diseño segmentado. Está constituida por dispositivos lógicos y los registros de desacoplo conectados en sus entradas y salidas.

Posteriormente se ampliará el ámbito de utilización de la palabra etapa (pagina 138). Sin embargo, siempre estará asociada a la utilización de un dispositivo lógico aunque no se explicite.

Un ejemplo de etapa en la Figura 2.11 es el circuito combinacional E_0 junto con 9 registros conectados a sus entradas (x_0, y_0, c_0, x_1, y_1) y salidas (c_1, s_0, x_1, y_1).

Antes de proseguir con el ejemplo se describe la estructura y el funcionamiento de un dispositivo segmentado.

Funcionamiento y caracterización

Un registro de desacoplo transfiere la información de su entrada a la salida en instantes predeterminados de la señal de reloj. Nosotros utilizaremos el flanco ascendente de la señal de reloj.

Una metodología de reloj del tipo disparo por flanco permite leer el contenido de un registro, que sea utilizado como entrada en un circuito combinacional y que se escriba en otro registro (o el mismo registro) en el mismo ciclo de reloj. Entonces, las entradas en una etapa son valores que llegan de un ciclo previo, mientras que las salidas son valores que se utilizarán en el siguiente ciclo en otra etapa o en la misma etapa (Figura 2.12).

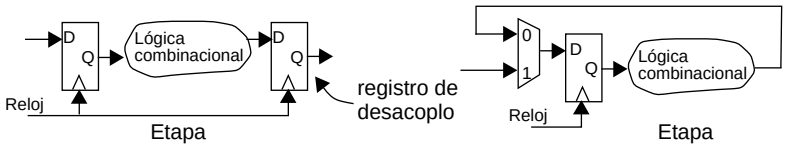


Figura 2.12 *Elementos de una etapa en un diseño segmentado.*

El funcionamiento de un dispositivo segmentado es síncrono. Esto es, en el flanco ascendente de la señal reloj todos los registros de desacoplo almacenan las salidas de la lógica combinacional. Por tanto, los registro de desacoplo aislan las etapas durante el periodo de la señal de reloj.

Después del flanco ascendente de la señal de reloj, la información en la salida del registro es utilizada como entrada por el módulo combinacional conectado a su salida. La duración del periodo es el tiempo utilizado por los módulos combinacionales para procesar las entradas y determinar las salidas.

En la Figura 2.11 se muestra el sumador de 2 bits segmentado junto con los registros de desacoplo. Las entradas de la etapa E0 son los vectores de bits X e Y y el bit c_0. El módulo combinacional en la etapa E0 suma los bits (x_0, y_0) y no procesa los bits (x_1, y_1). La salida de la etapa E0 es el conjunto de bits {x_1, y_1, c_1, s_0}. En el siguiente ciclo este conjunto de bits son las entradas de la etapa E1. Entonces la etapa E1 suma los bits (x_1, y_1, c_1) y no procesa el bit s_0. La salida de la etapa E1 es el vector de bits (s_1, s_0) y el bit c_2.

En la Figura 2.13 se muestra la utilización temporal y sincronizada con la señal de reloj de las etapas E0 y E1 cuando se efectúa la suma de los vectores de bits X= (x_1, x_0) e Y = (y_1,y_0).

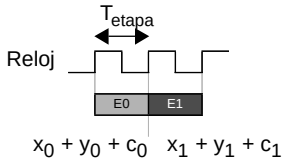


Figura 2.13 *Utilización de las etapas en un diseño segmentado al procesar una operación.*

Tabla de reserva. Es un diagrama que muestra la ocupación de las etapas de un dispositivo segmentado en función del tiempo expresado en ciclos.

Pueden utilizarse dos diagramas. Uno de ellos es bidimensional y el eje vertical representa las etapas del dispositivo mientras que el eje horizontal representa el tiempo en ciclos (Figura 2.14). Cada casilla del diagrama está identificada por una etapa y un ciclo determinado. Cada fila en la tabla de reserva se corresponde con la utilización de una única etapa a lo largo del tiempo. Para indicar el (los) ciclo(s) de ocupación se utilizan marcas en las casillas. Cada columna es una observación de la utilización de las etapas en un ciclo concreto.

La otra alternativa es utilizar un diagrama unidimensional y etiquetar cada ciclo con la etapa(s) que se utiliza(n). En la Figura 2.14 se muestran los dos diagramas para el caso del sumador de 2 bits segmentado en dos etapas E0 y E1.

Latencia de una tarea. Es el tiempo total de procesado de la tarea, expresado en ciclos de la señal de reloj.

En el ejemplo del sumador de 2 bits la latencia de suma son 2 ciclos.

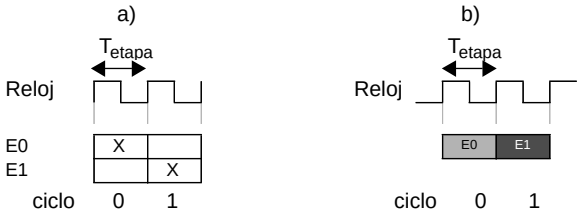


Figura 2.14 *Tabla de reserva de un sumador segmentado en dos etapas (E0, E1). a) bidimensional, b) unidimensional.*

En un funcionamiento serie del sumador segmentado, la suma de un par de números no se inicia hasta que ha finalizado la suma del par previo. Entonces al procesar una secuencia de sumas se observa el siguiente diagrama temporal.

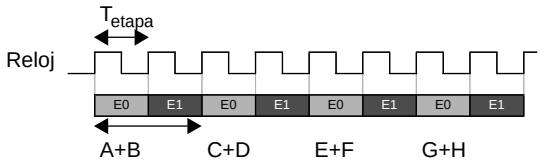


Figura 2.15 *Procesado serie de una secuencia de sumas en un sumador segmentado.*

Solapamiento en el procesado de tareas

En la Figura 2.15 se observa que mientras se utiliza la etapa E0 no se utiliza la etapa E1 y viceversa. Para extraer provecho de la segmentación se solapa el procesado de tareas. Se utilizan los ciclos en que una tarea no ocupa una etapa para procesar otra(s) tarea(s).

La Figura 2.16 muestra la utilización de las etapas al efectuar la suma de cuatro pares de números de forma solapada. En horizontal se muestra una operación etiquetando el ciclo en que se utiliza cada etapa. En el eje vertical se identifican las operaciones de más antigua a más joven en el sentido de arriba hacia abajo.

A partir de ahora se dará por sobreentendido el procesado de tareas de forma solapada cuando se mencione un dispositivo segmentado o se mencione la palabra segmentación.

En el flanco ascendente de la señal de reloj, las salidas de la etapa E0 se almacenan en los registros de desacoplo que hay entre las etapas E0 y E1 y en el siguiente ciclo son utilizadas como entradas en la etapa E1. En la Figura 2.16 se observa que concurrentemente, en un ciclo de la señal de reloj, se está efectuando la suma de dos pares de números. Un par está en la etapa E1 y otro par está en la etapa E0.

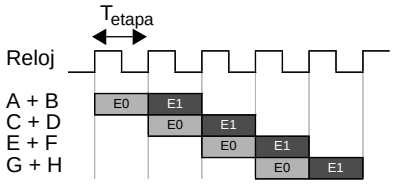


Figura 2.16 *Solapamiento del cálculo de varias sumas de vectores de 2 bits.*

En el sumador segmentado una operación tarda 2 ciclos en procesarse. Sin embargo, finaliza (inicia) una operación por ciclo. Respecto a un funcionamiento serie del diseño segmentado (Figura 2.15) se ha duplicado el número de sumas procesadas por ciclo.

Restricciones en el tiempo de ciclo

Para determinar el tiempo de ciclo debe calcularse el tiempo de retardo de cada una de las etapas del diseño segmentado y seleccionar el valor máximo (Figura 2.17). Esto es, el tiempo de ciclo del diseño segmentado debe cumplir la siguiente relación.

$$T \geq max(t_{Ei}) + t_p$$

siendo t_{Ei} el tiempo de retardo de la lógica combinacional en la etapa E_i y suponiendo que t_p es el mismo valor para todos los registros de desacoplo.

El sumador con propagación serie del acarreo se ha segmentado en 2 etapas denominadas E0 y E1 y el tiempo de retardo de la lógica combinacional en cada etapa en el mismo e igual $t_{E0} = t_{E1}$. Entonces, el periodo o tiempo de ciclo es

$$T_{etapa} = t_p + t_{E0} = t_p + t_{E1} = 2 + 30 = 32.$$

donde t_p es el tiempo de retardo de los registros de desacoplo.

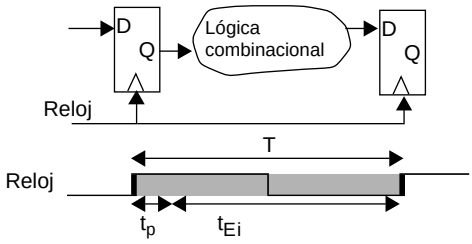
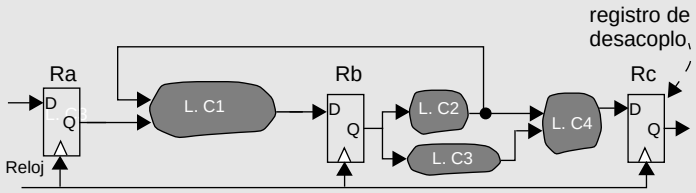


Figura 2.17 *Retardo de una etapa.*

Ejercicio

En la figura se muestra un dispositivo segmentado en dos etapas. Los elementos L.C1, L.C2, L.C3 y L.C4 representan lógica combinacional. El retardo de propagación de estos elementos es respectivamente t_{c1}, t_{c2}, t_{c3} y t_{c4} y el retardo de los registros de desacoplo es t_p. Calcule el tiempo de ciclo del diseño segmentado. Tenga en cuenta que la disposición espacial en el dibujo de un elemento no implica pertenencia sólo a una etapa concreta.



Respuesta

El elemento L.C2 se utiliza en las dos etapas. El retardo de la lógica de la etapa A es T1 = t_{c2} + t_{c1}, ya que los elementos L.C2 y L.C1 están conectados en serie. El retardo de la lógica de la etapa B es T2 = max (t_{c2}, t_{c3}) + t_{c4}.

Entonces, el tiempo de ciclo es

$$t_{ciclo} = t_p + \max(T1, T2)$$

Resumen

La idea de la segmentación es identificar fases en una tarea que efectúa un dispositivo. Entonces se diseñan o identifican módulos combinacionales para cada una de las fases o grupos de ellas. Estos módulos se conectan ente si, insertando registros de desacoplo en las conexiones entre módulos, y se establece una secuencia de utilización de los módulos que de lugar a la tarea

original. El sistema funciona de forma sincronizada con una señal de reloj y el estado de los registros de desacoplo se actualiza en el flanco ascendente de la señal de reloj.

El desacoplo de las etapas, soportado por los registros de desacoplo, permite solapar el procesado de varias tareas en el dispositivo segmentado. El resultado es que, respecto a un funcionamiento serie, se consigue incrementar el número de tareas procesadas por unidad de tiempo.

Ganancia

En un dispositivo que funciona de forma serie el tiempo de procesado de una tarea es (Figura 2.18)

$$T_{serie} = t_p + t_{\text{lógica}}$$

donde t_p es el retardo del registro en la entrada de la lógica combinacional y $t_{\text{lógica}}$ es el retardo de la lógica, siendo el tiempo de ciclo del reloj o período igual al tiempo de procesado.

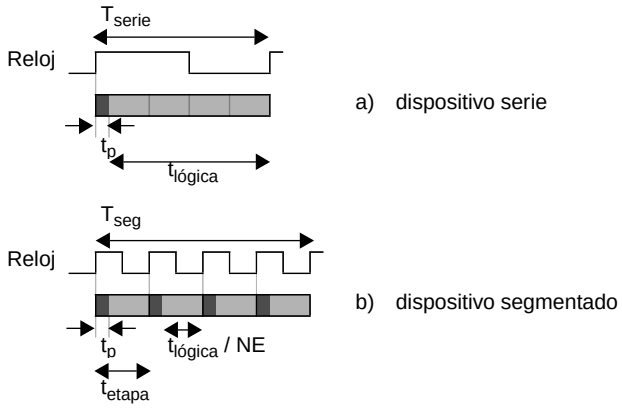


Figura 2.18 *Tiempo de procesado en: a) dispositivo serie y b) dispositivo segmentado.*

Al segmentar el dispositivo supondremos idealmente que el retardo de la lógica del dispositivo serie se distribuye uniformemente entre la lógica de las etapas del diseño segmentado. Esto es,

$$t_{\text{lógica}}\big|_{segmentado} = t_{\text{lógica}} / \text{NE}$$

donde NE es el número de etapas.

Entonces, teniendo en cuenta el retardo de los registros de desacoplo, el tiempo de procesado de una tarea es (Figura 2.18)

$$T_{seg} = NE \times (t_p + t_{\text{lógica}}/\text{NE}) = NE \times t_p + t_{\text{lógica}}$$

y el tiempo de ciclo del reloj o etapa en el diseño segmentado es

$$t_{etapa} = t_p + t_{\text{lógica}}/\text{NE}$$

Notemos que el tiempo en procesar una tarea individual en el dispositivo segmentado es mayor que en el dispositivo que funciona de forma serie.

$$T_{serie} = t_p + t_{\text{lógica}} < NE \times t_p + t_{\text{lógica}} = T_{seg}$$

La ganancia del dispositivo segmentado respecto del dispositivo serie se observa en el número de tareas procesadas por unidad de tiempo. El tiempo para procesar una secuencia de n tareas en los dos dispositivos es (Figura 2.19).

$$T_{serie}\big|_n = n \times (t_p + t_{\text{lógica}})$$

$$T_{seg}\big|_n = n \times (t_p + t_{\text{lógica}}/\text{NE}) + (NE - 1) \times (t_p + t_{\text{lógica}}/\text{NE})$$

donde suponemos de forma idealizada que en el dispositivo segmentado se puede iniciar una nueva operación en cada ciclo.

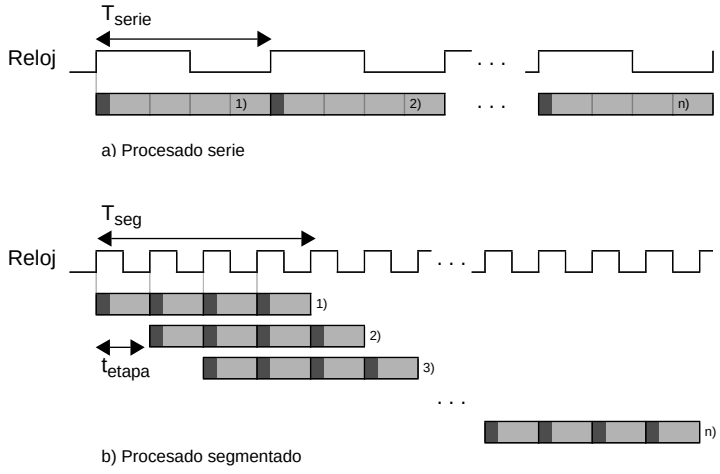


Figura 2.19 *Procesado de n tareas en: a) dispositivo serie y b) dispositivo segmentado.*

Supongamos que en el diseño segmentado n >> NE. Entonces,

$$T_{seg}\big|_{\mathrm{n}} = n \times (t_p + t_{\mathrm{lógica}}/\mathrm{NE})$$

El número de tareas procesadas por unidad de tiempo es

$$Productividad_{serie} = \frac{n}{n \times (t_p + t_{\mathrm{lógica}})} = \frac{1}{t_p + t_{\mathrm{lógica}}}$$

$$Productividad_{seg} = \frac{n}{n \times (t_p + t_{\mathrm{lógica}}/\mathrm{NE})} = \frac{1}{t_p + t_{\mathrm{lógica}}/\mathrm{NE}}$$

Por tanto, la ganancia de productividad es

$$Ganancia = \frac{Productividad_{seg}}{Productividad_{serie}} = \frac{T_{serie}\big|_n}{T_{seg}\big|_{\mathrm{n}}} = \frac{t_p + t_{\mathrm{lógica}}}{t_p + t_{\mathrm{lógica}}/\mathrm{NE}}$$

Suponiendo que el retardo de los registros de desacoplo es despreciable frente al retardo de la lógica ($t_p << t_{lógica}$) tenemos que el valor máximo de la ganancia es

$$Ganancia \leq NE$$

Por tanto, la productividad en el caso ideal se multiplica por el número de etapas cuando se utiliza un diseño segmentado.

Discusión de las idealizaciones

En general, al segmentar un dispositivo el retardo del dispositivo serie no se distribuye de forma uniforme entre las etapas. Entonces, el tiempo mínimo de ciclo o etapa en el diseño segmentado es

$$t_{etapa} = t_p + max(t_{Ei})$$

donde t_{Ei} es el retardo de la lógica en la etapa E_i y t_p es el retardo de un registro de desacoplo. En estas condiciones la ganancia es menor que la ideal.

Por ejemplo, la ganancia del sumador de 2 bits segmentado respecto del sumador serie es: ganancia = 52 / 32 = 1.62.

Esto es, el sumador segmentado procesa un 62% más rápido la secuencia de operaciones. La mejora es menor que 2 debido al retardo de los registros de desacoplo y a que el tiempo de retardo

de los módulos combinacionales (E0, E1) es el máximo de las dos señales que calcula. En contraposición, en el sumador serie existe solapamiento en la determinación de las señales de salida.

En ocasiones también existe otra fuente de pérdida de rendimiento: no se puede iniciar una nueva tarea en cada ciclo por razones estructurales del camino segmentado.

ELEMENTOS DE DESACOPLO

Distinguiremos dos elementos de memorización sincronizados con la señal de reloj. Uno de ellos utiliza el nivel de la señal de reloj para determinar cuando se almacena la señal de entrada y el otro utiliza el flanco de la señal de reloj para efectuar la misma operación. El primer elemento de almacenamiento se denomina celda y se dice que es sensitivo al nivel. El segundo elemento de almacenamiento se denomina biestable o registro y se dice que se dispara por flanco de reloj. Tanto uno como el otro se han utilizado en la implementación de dispositivos segmentados.

Nosotros utilizaremos una metodología de reloj del tipo disparo por flanco de reloj. Por tanto utilizaremos elementos de almacenamiento del tipo registro. Ahora bien, como un registro de desacoplo se puede construir con 2 celdas, se describe en primer lugar la lógica y el funcionamiento de un elemento de almacenamiento tipo celda.

Celda D (latch D)

Una celda D tiene una entrada, una salida y la señal de reloj. El valor de la señal de entrada se almacena en función del nivel de la señal de reloj: cuando el nivel de la señal de reloj es 0 o 1.

En la Figura 2.20 se muestra una celda D implementada mediante un multiplexor. La celda es transparente cuando el nivel de la señal de reloj es 1, ya que la información de la entrada se transmite a la salida. Sin embargo, cuando el nivel de la señal de reloj es 0, la celda se cierra (opaca) y se mantiene en su salida el valor observado la última vez que la celda estuvo abierta (transparente).

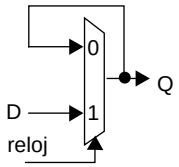


Figura 2.20 *Celda D.*

La Figura 2.21 muestra una secuencia de ciclos donde se observa el funcionamiento de la celda D. Los intervalos de tiempo en los cuales la celda es transparente se han marcado con línea más gruesa en la señal de reloj. Las transiciones de nivel en la señal Q, inducidas por una transición de nivel en la señal D, cuando la celda es transparente se han marcado con líneas de trazo discontinuo finalizadas con una flecha.

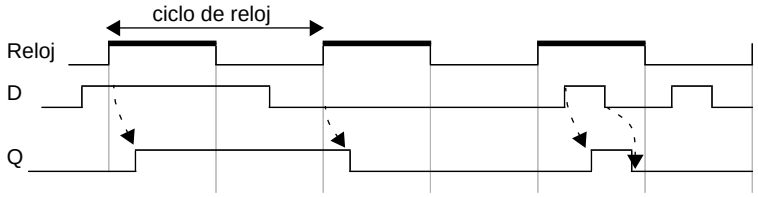


Figura 2.21 *Evolución temporal, en una celda D, de la señal de salida en función de la señal de entrada y la señal de reloj.*

Cuando el nivel lógico de la señal de reloj es 1, la salida Q sigue a la entrada D, después de un retardo de propagación de la celda. En el tercer ciclo de la señal de reloj (Figura 2.21) se produce un pulso en la entrada que se observa en la salida mientras el nivel de la señal de reloj es 1. Sin embargo, como la señal D toma el valor lógico 0, antes de que la señal de reloj cambie al nivel lógico 0, la salida tendrá el valor lógico 0 cuando el nivel de la señal de reloj es 0. Observar también que cuando el nivel de la señal de reloj es 0 no se modifica la salida aunque se modifique la entrada.

Registro D o Biestable D (Flip-flop D)

El número de entradas y salidas de un registro es el mismo que en la celda D. Sin embargo el registro D almacena la señal que hay en su entrada cuando se produce un flanco de la señal de reloj.

Un registro D se construye utilizando dos celdas D que actúan con niveles de la señal de reloj opuestos. La primera celda se denomina maestro y la segunda celda se denomina esclavo. En la Figura 2.22 se muestra una configuración de las celdas cuando el registro almacena la entrada en el flanco ascendente de la señal de reloj. Observemos que el inversor en el camino de la señal de reloj puede eliminarse si se conectan al revés las entradas del multiplexor en la celda maestro.

La celda maestro es transparente en el nivel 0 de la señal de reloj mientras que la celda esclavo está cerrada. Cuando el nivel de la señal de reloj cambia al valor 1 la celda maestro se cierra y la celda esclavo es transparente. Por tanto, el registro que forman las dos celdas almacena el valor de la entrada en el flanco ascendente de la señal de reloj.

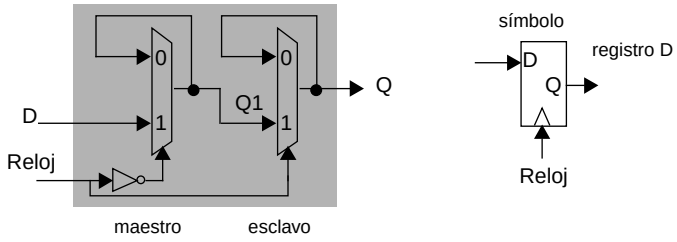


Figura 2.22 *Registro D.*

En la Figura 2.23 se muestra la evolución temporal de la señal de salida de un registro en función de la señal de entrada y la señal de reloj. El flanco ascendente de la señal de reloj se ha marcado con trazo grueso. Así mismo, se han oscurecido, mediante una trama en las señales Q y Q1, los intervalos de tiempo en los que esta cerrada respectivamente la celda esclavo o maestro. Una línea finalizada con una flecha indica una transición de nivel en una salida, inducida por una transición de nivel en una entrada del registro.

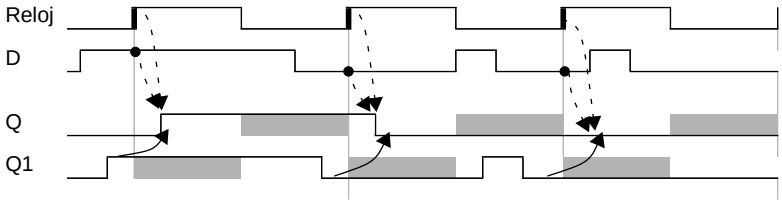


Figura 2.23 *Evolución temporal, en un registro D, de la señal de salida (Q) en función de la señal de entrada (D) y la señal de reloj.*

Cuando el nivel de la señal de reloj es 0 la celda maestro (Q1) sigue a la entrada D, después de un retardo de propagación. A su vez la celda esclavo mantiene su salida invariable (Q). Cuando se produce el cambio de nivel de la señal de reloj, la celda maestro se cierra y la celda esclavo pasa a ser transparente. Entonces, en la salida del registro (Q) se observa la salida de la celda maestro (Q1) que es invariable, ya que la celda maestro está cerrada.

En la Figura 2.23 se puede observar que los pulsos de la señal de entrada D, cuando el nivel de la señal de reloj es 1, no se transmiten a la salida, ya que la celda maestro está cerrada y su salida se mantiene invariable.

TIPOS DE UNIDADES FUNCIONALES SEGMENTADAS

En un apartado previo se ha analizado la segmentación de una unidad funcional o dispositivo que efectúa únicamente un tipo de tarea y además cada etapa se conecta exclusivamente a etapas sucesoras. Por etapa sucesora se entiende que no se ha utilizado previamente en el procesado de la tarea.

Dispositivo segmentado unifunción. Unicamente procesa un tipo de tarea.

Dispositivo segmentado lineal. Cada etapa se conecta exclusivamente a etapas sucesoras (Figura 2.24 a) y ninguna etapa tarda más de un ciclo.

Dispositivo multiciclo o seudolineal. Alguna etapa se utiliza de forma repetida en ciclos consecutivos (Figura 2.24 b).

Este último caso, desde el punto de vista de los instantes en que se puede iniciar una nueva tarea, es equivalente a una segmentación lineal, siendo el tiempo de ciclo de la segmentación lineal equivalente igual a un múltiplo entero de la segmentación de partida.

Dispositivo segmentado no lineal. Alguna etapa se utiliza más de una vez y al menos una de las utilizaciones es en ciclos no consecutivos (Figura 2.24 c).

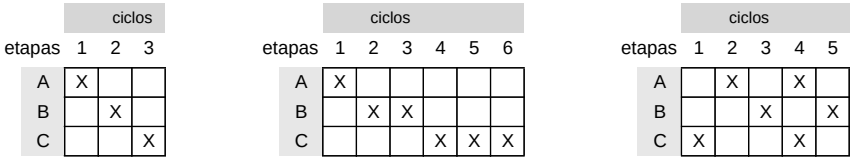


Figura 2.24 *Ejemplos de segmentación lineal y no lineal.*

También se pueden diseñar dispositivos que procesen varios tipos de tareas.

Dispositivo segmentado multifunción. Dispositivo que puede procesar tareas de varios tipos.

Dispositivo segmentado multifunción dinámico. Dispositivo que puede procesar de forma solapada varias tareas de distinto tipo.

Ejemplos de dispositivos segmentados multifunción dinámicos, entre otros, son: a) unidad funcional para efectuar operaciones con números representados en coma flotante, b) el camino de datos de un procesador segmentado que interpreta un conjunto de instrucciones y c) el acceso a la cache de datos.

REPLICACIÓN

La técnica de replicación utiliza varias réplicas de un dispositivo serie y su objetivo es incrementar el número de tareas procesadas por unidad de tiempo. Las réplicas procesan tareas de forma concurrente y cada réplica procesa tareas de forma serie. En la

Figura 2.25 se muestran dos sumadores de dos bits y en la Figura 2.26 se muestra la utilización de las dos réplicas para efectuar la suma de cuatro pares de números de forma solapada. En horizontal se muestra la secuencia de tareas que procesa una réplica y en vertical se muestran las réplicas. Cada réplica empieza a procesar una suma en el mismo instante de tiempo. Entonces, después de 2 ciclos se han procesado las 4 sumas. Por tanto, se tarda la mitad de tiempo respecto a utilizar únicamente un sumador.

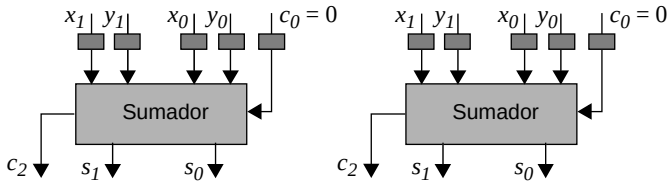


Figura 2.25 *Dos réplicas de un sumador serie.*

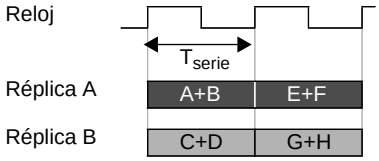


Figura 2.26 *Diagrama temporal que muestra el procesado de una secuencia de sumas cuando se utilizan dos sumadores serie.*

Ganancia

Seguidamente evaluamos el tiempo de procesar n tareas utilizando réplicas de un dispositivo. Supondremos que todas las réplicas empiezan a procesar tareas en el mismo instante de tiempo. Cuando un dispositivo finaliza el procesado de una tarea se inicia sin demora el procesado de otra tarea si quedan tareas pendientes (Figura 2.26).

El tiempo para procesar una secuencia de n tareas es:

$$T_{rep} = \left\lceil \frac{n}{ND} \right\rceil \times (t_p + t_{\text{lógica}})$$

siendo ND el número de dispositivos.

El número de tareas procesadas por unidad de tiempo es:

$$Productividad_{rep} = \frac{ND}{t_p + t_{\text{lógica}}}$$

donde suponemos que n es múltiplo de ND.

La ganancia es:

$$Ganancia = \frac{Productividad_{rep}}{Productividad_{serie}} = \frac{T_{serie}|_n}{T_{rep}|_n} = \frac{n \times (t_p + t_{\text{lógica}})}{\frac{n}{ND} \times (t_p + t_{\text{lógica}})}$$

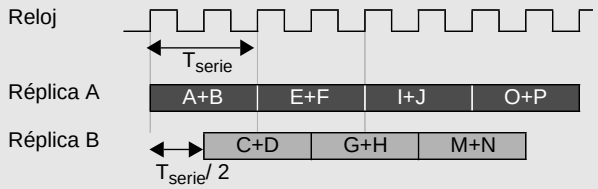
$$Ganancia = ND$$

Por tanto, la productividad en el caso ideal se multiplica por el número de réplicas cuando se utiliza replicación.

La ganancia de dos sumadores de dos bits respecto a un sumador serie sería: 52 / (52/2) = 2.

Ejercicio

En un sistema con varias réplicas se quiere procesar una secuencia de n tareas. Ahora bien, se quiere que al iniciar el procesado de la secuencia de tareas exista un intervalo constante de tiempo entre el inicio de tareas en réplicas distintas. Este intervalo es el periodo de una señal de reloj. Cuando una réplica finaliza una tarea empieza a procesar otra tarea, mientras queden tareas por procesar, en un instante de tiempo que es múltiplo del periodo de la señal de reloj (t_c). En la siguiente figura se muestra el caso de 2 réplicas y $t_c = T_{serie}/2$.



El tiempo de procesado de una tarea en una réplica es $T_{serie} = t_p + t_{lógica}$. ¿Cuál es el número mínimo de réplicas necesarias para iniciar una tarea cada t_c?.

Supongamos que ND es el número mínimo de réplicas necesaria para iniciar una tarea en cada periodo de la señal de reloj. Calcule el tiempo para procesar una secuencia de n tareas y la productividad.

Suponiendo que t_c es divisor de T_{serie}, calcule la ganancia del procesado propuesto en el ejercicio respecto de un procesado serie de las secuencia de tareas.

Respuesta

El siguiente cociente entre el tiempo de procesado y el periodo de la señal de reloj nos indica el número de ciclos que transcurren desde que un dispositivo se ocupa hasta que finaliza el procesado.

$$\left\lceil \frac{T_{serie}}{t_c} \right\rceil$$

Durante estos ciclos queremos iniciar el procesado de tareas en otras réplicas. Por tanto, el número de réplicas (ND) es

$$ND = \left\lceil \frac{T_{serie}}{t_c} \right\rceil$$

El tiempo para procesar una secuencia de n tareas es

$$T_{rep} = n \times t_c + (ND - 1) \times t_c$$

El número de tareas procesadas por unidad de tiempo es:

$$Productividad_{rep} = \frac{n}{T_{rep}} = \frac{n}{n \times t_c + (ND - 1) \times t_c}$$

Suponiendo que ND <<n tenemos

$$Productividad_{rep} = \frac{1}{t_c}$$

La ganancia respecto de un procesado serie es

$$Ganancia = \frac{T_{serie}\big|_n}{T_{rep}\big|_n} = \frac{n \times T_{serie}}{n \times t_c + (ND - 1) \times t_c}$$

Suponiendo, como dice el enunciado, que t_c es divisor de T_{serie} tenemos

$$Ganancia = \frac{n \times T_{serie}}{n \times t_c + (ND - 1) \times t_c} = \frac{n \times ND \times t_c}{n \times t_c + (ND - 1) \times t_c}$$

Suponiendo ND <<n.

$$Ganancia = ND$$

Por tanto, la productividad en el caso ideal se multiplica por el número de réplicas cuando se utiliza replicación.

INTERPRETACIÓN DE INSTRUCCIONES

En primer lugar se efectúa una descripción simplificada de un procesador que interpreta las instrucciones de forma serie. En concreto, se describe el formato de las instrucciones, el camino de datos y la utilización de los elementos (recursos) básicos del camino de datos en las fases de interpretación de una instrucción. Posteriormente se describe una segmentación del proceso de interpretación de instrucciones, se analizan los riesgos estructurales o de conflictos y se describen técnicas para eliminar algunos tipos de conflictos.

Lenguaje máquina

El lenguaje máquina que consideraremos tiene instrucciones para leer o escribir datos en memoria (MEM), instrucciones para efectuar cálculos aritmético-lógicos (ENT) e instrucciones que permiten modificar el secuenciamiento implícito (IS). En la Figura 2.27 se muestra el formato de las 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
	RI	CoOp	ra	literal	1	func	rc	
MEM	Lo | St	CoOp	ra	rb	literal			Acceso a memoria
IS	BR	CoOp	ra	literal				Saltos relativos
	JMP	CoOp	ra	rb	func	0 ... 0	0 ... 0	Saltos indexados

Figura 2.27 *Formato de las instrucciones de lenguaje máquina. Todas las instrucciones son de 32 bit de longitud con un campo de código de operación de 6 bits, ubicado en los bits <31:26> de la instrucción.*

En la Figura 2.28 se muestra la especificación semántica de los tipos de instrucciones RR, RI, Load y Store. Las instrucciones Load y Store utilizan el contenido de un registro y el campo literal de la instrucción para calcular la dirección efectiva. Las instrucciones de tipo RR y RI efectúan cálculos lógicos o aritméticos con enteros y sus datos fuente están en registros o en un campo literal y almacenan el resultado en un registro.

Tipo	Descripción	Operandos			Especificación semántica
RR	operación aritmético-lógica	ra	rb	rc	$rc^v = ra^v$ (CoOp, func) rb^v
RI		ra	literal	rc	$rc^v = ra^v$ (CoOp, func) #literal
Load	Lectura de memoria	ra	literal		ra^v = Mem [rb^v + ExtSig (literal)]
Store	Escritura en memoria	ra	rb	literal	Mem [rb^v + ExtSig (literal)] = ra^v

Figura 2.28 *Especificación semántica de los tipos de instrucciones RR, Ri, Load y Store. El superíndice v indica contenido del registro. Los campos CoOp y func determinan la operación que debe efectuarse.*

En la Figura 2.29 se muestra la especificación semántica de los tipos de instrucciones BR y JMP.

Tipo	Descripción	Operandos			Especificación semántica
BR	incondicional	ra	literal		$CP^v = CP^v + 4$ x ExtSig (literal)
	llamada a procedimiento	ra	literal		$ra^v = CP^v + 4$ $CP^v = CP^v + 4$ x ExtSig (literal)
	condicionales	ra	literal		if (f[Ra^v , CoOp]) then $CP^v = CP^v + 4$ x ExtSig (literal) else $CP^v = CP^v + 4$
JMP	indexado	ra	rb	0	$CP^v = rb^v$
	llamada a procedimiento	ra	rb	0	$ra^v = CP^v + 4$ $CP^v = rb^v$

Figura 2.29 *Especificación semántica de las instrucciones de tipo BR y JMP. El superíndice v indica contenido del registro y CP es el registro contador de programa que almacena la dirección de la instrucción. La función f[Ra^v, CoOp] indica que se evalúa el contenido del registro en función del código de operación de la instrucción.*

Las instrucciones de tipo BR calculan la dirección de la siguiente instrucción de forma relativa a la dirección de la instrucción de secuenciamiento. Esto es, a la dirección de la instrucción de secuenciamiento se le suma el campo literal especificado en la instrucción. Las instrucciones de secuenciamiento condicional 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 utilizan el contenido de un registro como dirección de la siguiente instrucción que debe interpretarse.

Los dos tipos de instrucciones de secuenciamiento incluyen instrucciones que almacenan la dirección de la instrucción siguiente a la de secuenciamiento en un registro especificado en la instrucción.

Camino de datos de un procesador serie

El camino de datos de un procesador se utiliza para interpretar las instrucciones de lenguaje máquina.

En la Figura 2.30 se muestran los recursos básicos del camino de datos de un procesador serie. En el camino de datos se distinguen los elementos memoria (MID), banco de registros (BR), unidad aritmético lógica (ALU), contador de programa (CP), módulo decodificador y control. Para simplificar el dibujo no se muestran algunos registros temporales necesarios para la interpretación de las instrucciones. Tampoco se muestran algunos elementos de lógica combinacional, como por ejemplo la lógica utilizada para evaluar la condición en una instrucción de secuenciamiento.

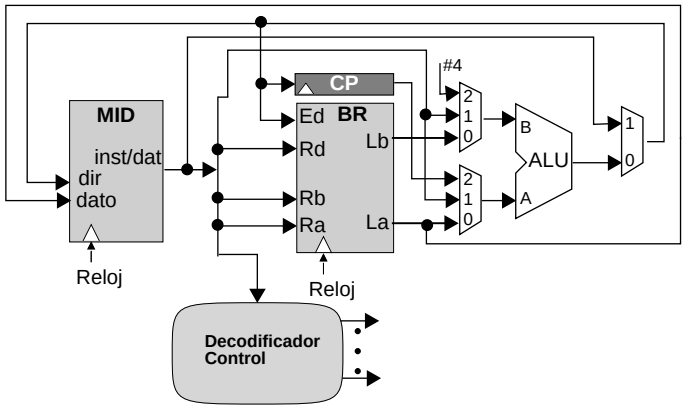


Figura 2.30 *Camino de datos simplificado de un procesador secuencial.*

La memoria almacena el código y los datos. El control del procesador se encarga de encadenar las fases de interpretación, las cuales se muestran en la Figura 2.31 junto con los recursos que se utilizan en cada fase.

Fases	Recursos utilizados
Determinar la dirección de la instrucción	ALU y CP
Búsqueda de la instrucción	memoria (MID)
Decodificación	decodificador
Lecturas de los registros fuente	banco de registros (BR)
Ejecución	ALU y memoria (MID)
Almacenamiento del resultado	banco de registros (BR) y memoria (MID)

Figura 2.31 *Fases de ejecución de una instrucción y utilización de recursos.*

En el camino de datos de la Figura 2.30 un elemento puede utilizarse varias veces cuando se interpreta una instrucción. Por ejemplo, el elemento memoria se utiliza para buscar la instrucción y en las instrucciones load/store para leer/escribir un dato. El elemento ALU se utiliza en: a) las operaciones aritmético lógicas, b) cálculo de la dirección efectiva de acceso a memoria en instrucciones load/store y c) cálculo de la dirección de la instrucción que se buscará en memoria.

Cada una de la fases de interpretación puede necesitar uno o varios ciclos de la señal de reloj. Nosotros supondremos que las fases de decodificación y lectura de operandos se efectuan en un ciclo de la señal de reloj y que todas las otras fases, menos ejecución, tienen una duración de un ciclo de reloj. El número de ciclos de la fase de ejecución depende de la instrucción que se interpreta. Por ejemplo, la fase de ejecución de las instrucciones de lenguaje máquina add y load son respectivamente 1 ciclo (ALU) y 2 ciclos (ALU, MID).

Un procesador serie no empieza a interpretar una instrucción hasta que ha finalizado la interpretación de la instrucción previa. En la Figura 2.32 se muestra la interpretación de varias instrucciones, necesitando la interpretación de cada instrucción varios ciclos. Los acrónimos CP, BUS, D/L, ALU y M se utilizan para indicar respectivamente las fases: determinación de la dirección de

una instrucción, búsqueda de la instrucción, decodificación y lectura de los registros fuente de la instrucción, utilización de la ALU, acceso a memoria para leer/escribir un dato. El acrónimo ES se utiliza para indicar escritura del resultado en el banco de registros y/o almacenar en registros temporales la dirección que determina una instrucción de secuenciamiento, además del resultado de evaluar la condición si es el caso.

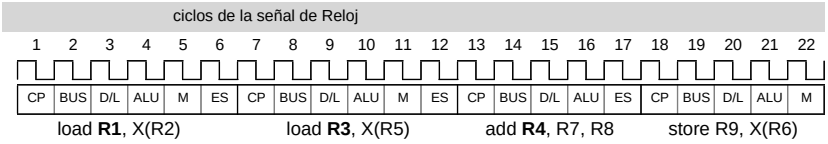


Figura 2.32 *Interpretación de una secuencia de instrucciones en un procesador serie. La instrucción interpretada se muestra en la parte inferior de la figura.*

Segmentación del proceso de interpretación de instrucciones

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 de empezar a interpretar una instrucción en cada ciclo de reloj. Esta técnica reduce el tiempo de ejecución de forma transparente al programador de lenguaje máquina. En otras palabras, cuando se ejecutan programas compilados para un procesador serie de la misma familia se obtiene el mismo resultado en menor tiempo.

Existen dos tipos de trabas que prohiben o dificultan procesar una instrucción por ciclo. 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. Esta última traba se desarrolla en el próximo capítulo. En este capítulo nos centramos en los riesgos estructurales debidos a los recursos hardware disponibles.

A la vista de la duración que hemos supuesto de las fases de interpretación de una instrucción, establecemos una correspondencia entre ciclos en el diseño serie multiciclo y ciclos en el diseño segmentado. En la Figura 2.33 se muestra la segmentación del proceso de interpretación para los distintos tipos de instrucciones. Los acrónimos utilizados para identificar los ciclos 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 y en la instrucciones ENT/IS ésta etapa se identifica como A/E. Cuando no requiera explicitarse o se sobreentienda se utilizará el acrónimo ALU.

Tipo	instrucción	ciclos 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	A/E	ES	

Figura 2.33 *Segmentación del proceso de interpretación de instrucciones.*

Distinguimos los casos de instrucciones load y store frente al de instrucciones ENT/IS. La diferencia entre instrucciones load y store está en la etapa de escritura en el banco de registros. La instrucción store no escribe en el banco de registros. Las instrucciones de tipo BR actualizan el registro CP en el ciclo de escritura (ES) y en función del código de operación pueden también actualizar el banco de registros.

En la Figura 2.34 se muestra, mediante tablas de reserva, la ocupación de los recursos del camino de datos de la Figura 2.30, una vez segmentado, por parte de las instrucciones. Los ciclos que se muestran se corresponden con los ciclos de la Figura 2.33.

ciclos

	1	2	3	4	5	6
MID		X			X	
BR			X			X
ALU	X			X		

load

ciclos

	1	2	3	4	5
MID		X			X
BR			X		
ALU	X			X	

store

ciclos

	1	2	3	4	5
MID		X			
BR			X		X
ALU	X			X	

ENT / IS

Figura 2.34 *Tablas de reserva que muestran el ciclo en que se utilizan los recursos del camino de datos.*

En la Figura 2.34 mediante los acrónimos MID, BR y ALU identificamos los elementos memoria, banco de registros y ALU respectivamente. Suponemos que en el banco de registros (BR), en un ciclo, se pueden efectuar dos lecturas o una escritura. En otras palabras, se pueden hacer una o dos lecturas y ninguna escritura, o bien una escritura y ninguna lectura. La memoria MID sólo tiene un camino de acceso y se puede efectuar una lectura o una

escritura. Tanto en la memoria como en el banco de registros los recursos son los caminos de acceso, que pueden ser de sólo lectura o escritura o permitir ambas operaciones.

Como veremos posteriormente, el diseño descrito se puede mejorar sustancialmente añadiendo recursos. Sin embargo, será de utilidad para introducir conceptos, terminología y algunas técnicas que permiten mejorar un diseño segmentado.

Comentario sobre el término etapa utilizado a partir de ahora. Debido a que en un modelo lineal de segmentación las etapas se utilizan un sólo ciclo, usualmente también se denomina etapa al ciclo donde se utiliza una etapa. Por ejemplo, se dice que el proceso de interpretación de una instrucción son cinco etapas en lugar de que tarda 5 ciclos. Además, la etapa suele etiquetarse con la acción que se efectúa en lugar del recurso que se utiliza y subyacentemente se supone el conocimiento del recurso utilizado. Por ejemplo, el segundo ciclo se denomina etapa de búsqueda, siendo el recurso que se utiliza MID. Observe también que al banco de registros y a la memoria se accede en dos ciclos distintos. Sin embargo, el recurso que hay que considerar en estos casos es el número y tipo de los posibles accesos por ciclo.

Término etapa. Describiremos una etapa por la acción que se realiza y cuando haya que analizar riesgos estructurales se explicitará el recurso en los casos que su omisión dificulte la comprensión.

También a partir de ahora y únicamente en este capítulo obviaremos las instrucciones de secuenciamiento, ya que determinan cual es la siguiente instrucción que debe interpretarse y ello es una relación de dependencia, lo cual se ha comentado en la introducción que no es objeto de este capítulo y se desarrolla en los siguientes capítulos.

Riesgo estructural

Cuando se procesa una tarea utilizando una segmentación lineal, la ocupación de un recurso una única vez permite iniciar una tarea por ciclo. Sin embargo, cuando un recurso se utiliza varias veces se crean situaciones denominadas de riesgo estructural (o conflicto de recurso) con otras tareas que se han empezado a procesar previamente.

Riesgo estructural o conflicto de recursos. Se produce cuando en un dispositivo segmentado tareas concurrentes pretenden utilizar un mismo recurso en el mismo ciclo.

Ocupación de un recurso en la tabla de reserva. Número de marcas en la fila correspondiente de la tabla de reserva.

Un caso de utilización repetida de un recurso se produce en una segmentación multiciclo o seudolineal: los recursos que se utilizan varias veces se ocupan de forma contigua en el tiempo. Sea k el número de veces que se utiliza el recurso que más se ocupa. Entonces, se puede iniciar una nueva tarea cada k ciclos. Este caso es equivalente, desde el punto de vista del inicio de operaciones, a multiplicar por k el tiempo de ciclo.

El caso general es que un recurso, cuando se utiliza varias veces, se ocupe de forma no contigua en el tiempo. Por ejemplo, si en ciclos consecutivos iniciamos la interpretación de instrucciones de tipo ENT, en el ciclo 5, se produciría un conflicto en la utilización del recurso BR (Figura 2.35). Esta situación no es permisible y el control del camino de datos segmentado debe actuar para que no se produzca.

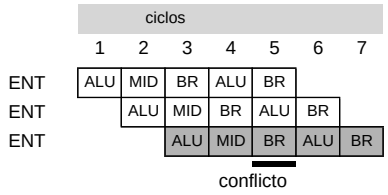


Figura 2.35 *Iniciación de una secuencia de instrucciones de tipo ENT en ciclos consecutivos. Conflicto de recursos en el ciclo 5 debido al recurso BR.*

Latencia de inicio o repetición. Número de ciclos que hay que esperar entre el inicio de dos tareas consecutivas en una unidad funcional segmentada para que no se produzca un riesgo estructural.

En general, cuando se procesa una secuencia de tareas, la latencia de inicio entre iniciaciones sucesivas puede ser distinta, obteniéndose una secuencia de latencias.

Latencia media de inicio (LMI). Dada una secuencia de latencias de inicio se calcula el valor medio.

La inversa de la latencia media de inicio es la productividad (en ciclos) que es un racional menor que uno.

La tabla de reserva de las instrucciones de tipo ENT muestra que hay dos recursos (BR y ALU) que se utilizan dos veces cuando se interpreta una instrucción. Entonces, como la ocupación de un recurso como máximo puede ser del 100%, tenemos que no se puede iniciar el proceso de interpretación de una instrucción cada ciclo.

Valor mínimo de la latencia media de iniciación. Es mayor o igual que el máximo número de marcas en cualquiera de las filas de la tabla de reserva, ya que la ocupación de un recurso no puede ser superior al 100%.

Seguidamente se analizan los riesgos entre instrucciones que tienen el mismo patrón de ocupación de las etapas y posteriormente se analizan los riesgos entre instrucciones que tiene distintos patrones de utilización de las etapas. Por patrón entendemos la misma tabla de reserva.

Además, en los dos casos se muestran técnicas que permiten eliminar los riesgos estructurales que se producen en el diseño efectuado. La decisión de eliminar un riesgo estructural es función de la frecuencia de ocurrencia del riesgo, de la influencia en el tiempo de ciclo y de la capacidad de integración. Por ejemplo, una forma de eliminar un riesgo estructural, debido a un recurso construido con lógica combinacional, es replicar el recurso tanta veces como sea necesario.

Conflictos entre instrucciones con el mismo patrón

En la 1ª fila de la Figura 2.36 se muestra la segmentación del proceso de interpretación de una instrucción ENT. En la 2ª fila se muestra la ocupación de los recursos por parte de la instrucción.

		ciclos				
		1	2	3	4	5
ENT	etapas	CP	BUS	D/L	ALU	ES
	recursos	ALU	MID	BR	ALU	BR

Figura 2.36 *Etapas y ocupación de recursos al interpretar una instrucción ENT.*

Para determinar una secuencia de latencias de inicio utilizaremos un diagrama temporal donde cada fila identifica una tarea distinta y las columnas identifican ciclos. A partir del ciclo en que se inicia una tarea cada casilla se etiqueta con el recurso o recursos utilizados.

El diagrama temporal se rellena suponiendo que se inicia una tarea en cada ciclo. Posteriormente se analiza ciclo a ciclo la 1ª y 2ª iniciación. Si la 2ª tarea pretende utilizar, en el mismo ciclo, un recurso que utiliza la 1ª tarea se produce un conflicto.

Si se detecta un conflicto se considera que no se ha iniciado la 2ª tarea. Si no se detecta ningún conflicto, los recursos utilizados en un ciclo se corresponden con el conjunto de recursos utilizados por las dos operaciones.

El análisis prosigue con las siguientes operaciones, teniendo en cuenta los recursos utilizados por las iniciaciones previas que se permiten. Esto es, las iniciaciones que no han dado lugar a riesgos estructurales.

El proceso finaliza cuando se observa una repetición de una secuencia de latencias.

En la Figura 2.37 se muestra un diagrama temporal para determinar una secuencia de latencias de inicio cuando se interpreta una secuencia de instrucciones ENT.

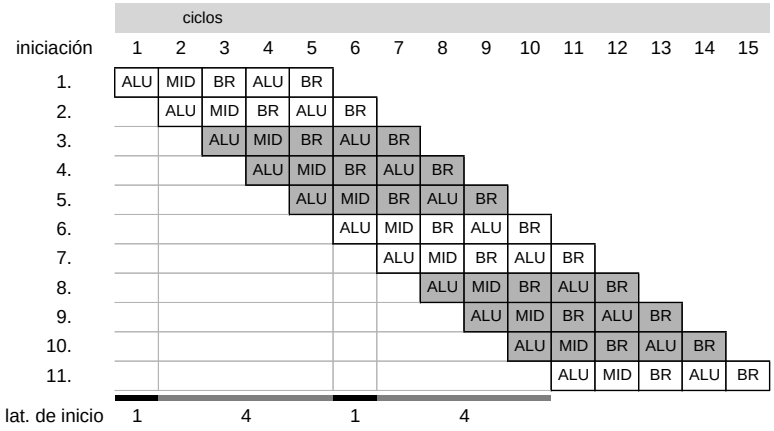


Figura 2.37 *Diagrama temporal para determinar una secuencia de latencias de inicio cuando se procesa una secuencia de instrucciones ENT.*

Para facilitar la observación se han tramado las instrucciones que dan lugar a conflictos. La 3ª iniciación no está permitida ya que se produciría un conflicto debido al recurso BR en el ciclo 5. Las iniciaciones 4ª y 5ª no están permitidas ya que se produciría un conflicto debido al recurso ALU en los ciclos 4 y 5 respectivamente. Indicamos sólo el primer conflicto, posteriormente se podrían producir otros conflictos. Los siguientes conflictos siguen el mismo patrón de riesgos estructurales.

Iniciando una nueva instrucción lo antes posible, la secuencia de latencias de inicio para las instrucciones de tipo ENT es (1, 4). Entonces, la latencia media de inicio es LMI = (1+4)/2 = 2.5 ciclos y por tanto, la productividad es 0.4 instrucciones/ciclo.

Rendimiento

Los riesgos estructurales representan una pérdida de rendimiento respecto del caso ideal, ya que no se puede iniciar una instrucción por ciclo.

Considerando la latencia media de inicio, suponiendo una distribución uniforme del tiempo de retardo de la lógica del dispositivo serie entre las etapas del dispositivo segmentado y despreciando el retardo de los registros de desacoplo, la ganancia de un diseño segmentado respecto de un diseño serie se calcula como

$$Ganancia = \frac{numero\ \ de\ \ etapas \times t_{ciclo}}{t_{ciclo} + (LMI - 1) \times t_{ciclo}}$$

donde t_{ciclo} es el tiempo de ciclo o periodo del diseño segmentado y el numerador es la latencia de inicio en el diseño serie del dispositivo.

En el segundo término del denominador, el factor que multiplica a t_{ciclo} se denomina ciclos perdidos o ciclos de bloqueo por riesgos estructurales, el cual es un valor medio y sus unidades son ciclos por instrucción.

$$ciclos\ \ perdidos = LMI - 1$$

Simplificando la expresión de la ganancia obtenemos:

$$Ganancia = \frac{NE}{LMI}$$

En el ejemplo de la Figura 2.37 la ganancia respecto de una interpretación serie es 5/2.5 = 2, siendo el número de ciclos perdidos 1.5 ciclos/instrucción. Obsérvese que, aún con un número muy significativo de conflictos de recursos, se interpretan el doble de instrucciones en el mismo tiempo. Sin embargo, este valor está bastante alejado del valor ideal que es 5; el número de etapas que tarda en interpretarse una instrucción ENT.

Ejercicio

Supongamos una secuencia de instrucciones load y que una instrucción se empieza a interpretar tan pronto como es posible (política avariciosa). Muestre una secuencia de latencias de inicio y calcule la ganancia respecto de una interpretación serie de la secuencia de instrucciones.

Respuesta

En la 1ª fila de la siguiente figura se muestra la segmentación del proceso de interpretación de una instrucción load. En la 2ª fila se muestra la ocupación de los recursos por parte de la instrucción.

		ciclos					
		1	2	3	4	5	6
load	etapas	CP	BUS	D/L	ALU	M	ES
	recursos	ALU	MID	BR	ALU	MID	BR

El siguiente diagrama temporal se utiliza para determinar las latencias de inicio. En cada ciclo se supone que se inicia una instrucción load. Posteriormente se efectúa el análisis de conflictos de recursos. Las instrucciones que tienen conflicto de recursos con instrucciones previas se han tramado.

	ciclos														
iniciación	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1.	ALU	MID	BR	ALU	MID	BR									
2.		ALU	MID	BR	ALU	MID	BR								
3.			ALU	MID	BR	ALU	MID	BR							
4.				ALU	MID	BR	ALU	MID	BR						
5.					ALU	MID	BR	ALU	MID	BR					
6.						ALU	MID	BR	ALU	MID	BR				
7.							ALU	MID	BR	ALU	MID	BR			
8.								ALU	MID	BR	ALU	MID	BR		
9.									ALU	MID	BR	ALU	MID	BR	
10.										ALU	MID	BR	ALU	MID	BR
11.											ALU	MID	BR	ALU	MID
lat. de inicio	1	1		4			1	1		4					

La 4ª iniciación no está permitida ya que se produciría un conflicto debido al recurso ALU en el ciclo 4. Las iniciaciones 5ª y 6ª no están permitidas ya que se produciría un conflicto debido al recurso ALU en los ciclos 5 y 6 respectivamente. Los siguientes conflictos siguen el mismo patrón de riesgos estructurales.

La latencia media de inicio es LMI = (1+1+4)/3 = 2 y la ganancia respecto de una interpretación serie es 6/2 = 3, lo cual representa un 200%.

Eliminación de conflictos

Una forma de eliminar un riesgo estructural, debido a lógica combinacional, es replicar el recurso tantas veces como sea necesario si la capacidad de integración lo permite. Otra posibilidad es que el retardo del recurso sea inferior al tiempo de ciclo y que el recurso se pueda utilizar varias veces en un ciclo.

La decisión de eliminar un riesgo estructural es función de la frecuencia de ocurrencia del riesgo y de la influencia en el tiempo de ciclo.

La ALU es un recurso que produce bastantes conflictos. Por un lado, en cada ciclo se utiliza la ALU para calcular la dirección de la instrucción que se va a buscar a memoria en el siguiente ciclo. Por otro lado, todas las instrucciones, excluyendo las de tipo JMP, utilizan la ALU para efectuar cálculos aritméticos o lógicos.

Para eliminar los riesgos estructurales, debidos al recurso ALU, es suficiente añadir un sumador en el camino de datos segmentado. Este sumador se utiliza para calcular la dirección de la siguiente instrucción que debe interpretarse, suponiendo secuenciamiento implícito. Por secuenciamiento implícito se entiende que después de utilizar la dirección i para buscar una instrucción en memoria, la siguiente instrucción que se busca en memoria se almacena en la dirección i+4.

En la Figura 2.38 se muestra la ocupación de los recursos por parte de las instrucciones cuando se ha añadido el sumador descrito.

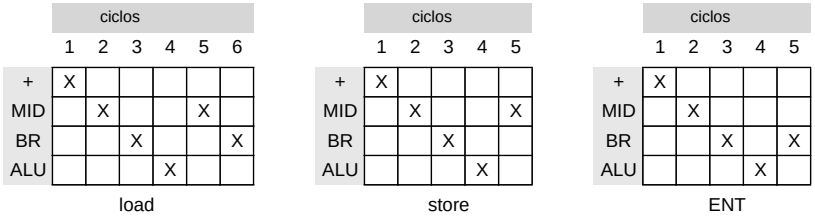


Figura 2.38 *Ocupación de recursos cuando se dispone de un sumador para el secuenciamiento implícito.*

Ejercicio

Supongamos las tablas de reserva de la Figura 2.38. ¿Cuál es la mínima latencia de inicio constante que garantiza que no se producen riesgos estructurales?.

Si sólo se interpretan instrucciones de tipo MEM (load o store) ¿Cuál es la mínima latencia de inicio constante que garantiza que no se producen riesgos estructurales?.

Respuesta

Utilizaremos la descripción efectuada en la sección anterior para determinar una secuencia de latencias de inicio. En este ejercicio se pide que la latencia de inicio sea constante. Para abreviar, en lugar de decir latencia de inicio constante diremos latencia de inicio.

La secuencia de instrucciones puede ser cualquier entrelazado de instrucciones de tipo ENT o MEM. En este ejercicio empezaremos analizando los conflictos cuando en la secuencia de instrucciones todas las instrucciones son del mismo tipo.

En la siguiente tabla se relacionan las latencias de inicio que producen conflictos y por conveniencia en este ejercicio se indica el primer recurso que produce conflicto en una iniciación y la primera iniciación en que se produce el conflicto. También se indican algunas latencias de inicio que no dan lugar a conflictos. La tabla ENT-ENT se corresponde con una secuencia de instrucciones ENT y la tabla MEM-MEM con una secuencia de instrucciones de tipo load o store.

latencia de inicio	ENT - ENT recurso	ENT - ENT iniciación	MEM - MEM recurso	MEM - MEM iniciación
1	BR	3ª	MID	4ª
2	BR	2ª	-	-
3	-	-	MID	2ª
4	-	-	-	-

La primer latencia de inicio que no da lugar a conflictos es 4.

Seguidamente analizamos si después de iniciar una instrucción de tipo ENT podemos iniciar una instrucción de tipo MEM con una latencia de inicio igual a 4 y viceversa. Utilizando las tablas de reserva vemos que es posible.

No es necesario proseguir el análisis ya que con latencia de inicio igual a 4 la 3ª iniciación no puede tener conflictos con la 1ª iniciación debido a que ha finalizado.

Para contestar a la 2ª pregunta es suficiente analizar la tabla MEM-MEM. Latencia de inicio sin conflictos es 2.

El banco de registros se utiliza en el 3º ciclo para leer los operandos fuente y en el 5º ciclo para escribir. Entonces, si se dispone de dos caminos de lectura (dos registros fuente) y un camino de escritura al banco de registros no se produce riesgo estructural. En la Figura 2.39 se muestra la ocupación de recursos cuando se dispone de dos caminos de lectura (BRL) y un camino de escritura (BRE). Recordemos que los recursos son los caminos de acceso.

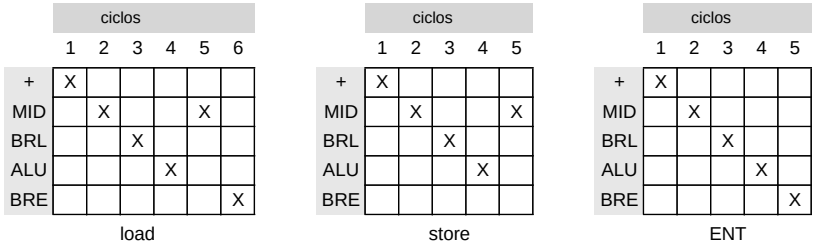


Figura 2.39 *Replicación de caminos de acceso al banco de registros para reducir riesgos estructurales.*

Con estas modificaciones, cuando se interpreta una secuencia de instrucciones tipo ENT no se producen conflictos, ya que los recursos sólo se utilizan una vez durante el proceso de interpretación. Sin embargo, las instrucciones tipo load/store aún producen conflictos ya que utilizan el recurso memoria dos veces.

La utilización de dos caminos de acceso a memoria para eliminar el riesgo es costosa. En su lugar, se tiene en cuenta que el código de un programa usualmente no se modifica y que la memoria de un procesador tiene una organización jerarquizada en niveles. Entonces, en el nivel de la jerarquía de memoria más cercano al procesador se distingue una cache de instrucciones (MI) y una

cache de datos (MD) y cada una de las cache tiene un camino de acceso (arquitectura Harvard). El siguiente nivel de la jerarquía de memoria almacena instrucciones y datos.

En una arquitectura Harvard, en la etapa de búsqueda de instrucciones se accede a la cache de instrucciones (MI) y en la fase de acceso a memoria de las instrucciones load/store se accede a la cache de datos (MD). En la Figura 2.40 se muestra la ocupación de los recursos después de esta modificación.

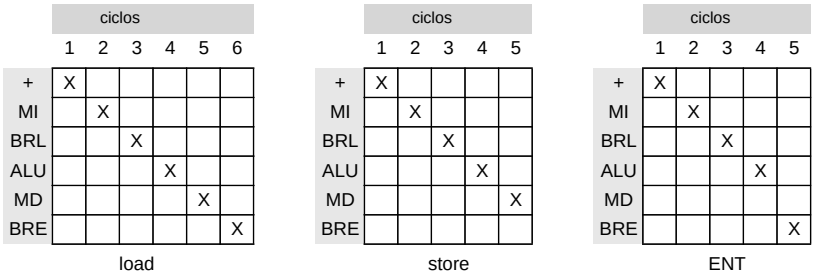


Figura 2.40 *Recursos y ocupación para que no existan conflictos entre instrucciones con el mismo patrón de ocupación de los recursos.*

Conflictos entre instrucciones con distinto patrón

En una secuencia de instrucciones que entremezcla los dos tipos de patrones (load, ENT) se producen riesgos estructurales, ya que la escritura al banco de registros se efectúa en ciclos distintos. En la Figura 2.41 se muestra una secuencia de instrucciones y la ocupación de las etapas. Después de una instrucción tipo ENT puede iniciarse una instrucción load o store sin que se produzcan conflictos. Sin embargo, después de iniciar una instrucción tipo load (ciclo 2) no puede iniciarse una instrucción de tipo ENT ya que se produce un conflicto en la utilización del recurso BRE (ciclo 7).

Figura 2.41 *Riesgos estructurales al interpretar una secuencia entrelazada de instrucciones load y ENT.*

Eliminación de conflictos

Las instrucciones de tipo ENT y load representan del orden del 50-60% de las instrucciones interpretadas y lo usual es que se interpreten de forma entrelazada. Por tanto, interesa eliminar el riesgo.

La alternativa de replicar el recurso, añadiendo un camino de escritura al banco de registros, es costosa (área ocupada) y posiblemente incremente el tiempo de acceso al banco de registros.

Entonces, se opta por otra alternativa que no requiere modificar el banco de registros. Se modifica el instante de escritura en las instrucciones ENT, con el objetivo de conseguir el mismo patrón de ocupación de etapas en las instrucciones ENT y load. Para ello, en las instrucciones ENT se retarda 1 ciclo la utilización del camino de escritura al banco de registros.

Retardo en una tabla de reserva. Ciclo en el cual no se procesa información.

En la Figura 2.42 se muestra el patrón de ocupación de los recursos después de introducir el retardo en la tabla de reserva de las instrucciones ENT.

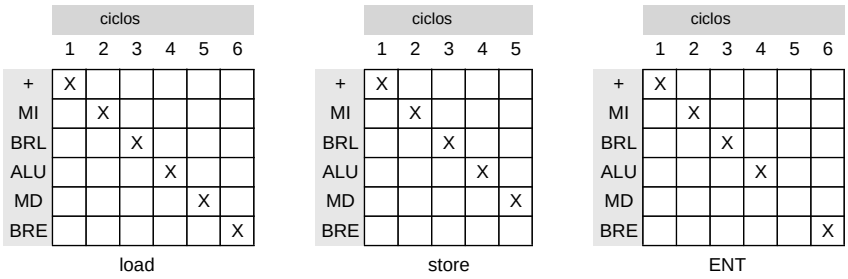


Figura 2.42 *Mismo patrón de ocupación de las etapas para todas las instrucciones.*

El diseño tiene una segmentación unificada para todos los tipos de instrucciones; todas las instrucciones utilizan los recursos, si ha lugar, en el mismo ciclo. Además, un recurso se utiliza una sola vez. Como efecto lateral se simplifica el control, ya que no es necesario detectar riesgos estructurales. En estas condiciones se dice que el procesador está segmentado de forma lineal.

Ejercicio

Supongamos que la memoria MID sólo tiene un camino de acceso. Las siguientes tablas de reserva muestran el ciclo en que se ocupan los recursos por parte de las instrucciones.

load

	ciclos					
	1	2	3	4	5	6
+	X					
MID		X			X	
BRL			X			
ALU				X		
BRE						X

store

	ciclos				
	1	2	3	4	5
+	X				
MID		X			X
BRL			X		
ALU				X	
BRE					

ENT

	ciclos					
	1	2	3	4	5	6
+	X					
MID		X				
BRL			X			
ALU				X		
BRE						X

Queremos determinar la pérdida de rendimiento que representa utilizar una MID con un camino de acceso respecto a utilizar una MID con dos caminos de acceso independientes. En este último caso no se producen conflictos y la latencia de inicio es 1.

En la siguiente secuencia de instrucciones se entremezclan instrucciones load y ENT.

1. load **R1**, X(R2)
2. store R9, X(R6)
3. add **R4**, R9, R10
4. load **R3**, X(R5)

Determine una secuencia de latencias de inicio para la anterior secuencia de instrucciones.

En un código el 30% de las instrucciones que se interpretan son del tipo load/store. Además, supondremos idealmente que las instrucciones no utilizan datos calculados por instrucciones previas y que tampoco hay instrucciones de secuenciamiento.

Calcule el número medio de instrucciones procesadas por ciclo.

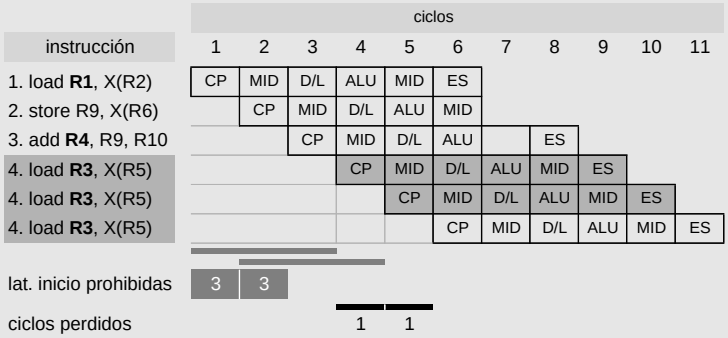
¿Cuál es la pérdida de rendimiento cuando se utiliza un camino de acceso respecto del caso de disponer de 2 caminos de acceso?.

Respuesta

En el siguiente diagrama se muestra la interpretación de la secuencia de instrucciones. En la etapa BUS se especifica el recurso MID y en la etapa M se especifica el recurso MID cuando se utiliza en la interpretación.

La 4ª instrucción no puede iniciarse en los ciclos 4 y 5 ya que se produce un riesgo estructural debido al recurso MID.

Observamos que siempre que se interpreta una instrucción de acceso a memoria al cabo de 3 ciclos no se puede iniciar una instrucción. Por tanto, se pierde 1 ciclo.



En un código con un 30% de instrucciones de tipo MEM la latencia media de inicio es 1 + 0.3 x 1 = 1.30. Esto es, la latencia de inicio ideal (1) más los ciclos de bloqueo determinados por las instrucciones de tipo MEM. Este último término es la proporción de instrucciones de tipo MEM por los ciclos de bloqueo que se producen cuando se interpreta una instrucción de tipo MEM.

El número medio de instrucciones procesadas por ciclo o IPC es

$$IPC = 1/CPI = 1/1.30 = 0.77$$

Un procesador con un camino de acceso es un 30% más lento que un procesador con 2 caminos de acceso.

CONTROL DE UN DISPOSITIVO SEGMENTADO

Un dispositivo segmentado requiere un control y en el control distinguimos dos partes (Figura 2.43): a) determinar cuando una nueva tarea se inicia o prosigue a la siguiente etapa y b) controlar la lógica de cada etapa y el encaminamiento entre las etapas. En el dibujo, la parte tramada más oscura de los registros de desacoplo representa datos y la parte más clara información de control.

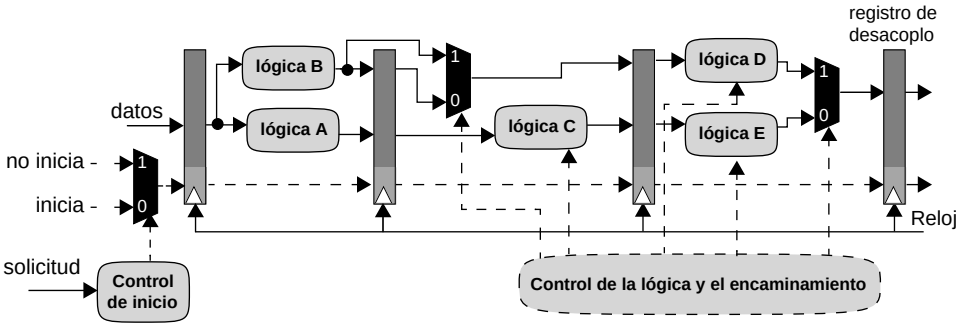


Figura 2.43 *Control de un dispositivo segmentado. Las lógicas A y B siempre efectúan la misma función.*

Control de iniciaciones

Un dispositivo segmentado lineal es un caso especial desde el punto de vista de control de iniciaciones. En cada ciclo se puede iniciar una tarea.

El control de iniciación utiliza las tablas de reserva y la señal de solicitud de tarea pendiente. Una tabla de reserva de una tarea permite caracterizar las latencias de inicio permitidas y prohibidas.

Latencia prohibida. Latencia de inicio que da lugar a un riesgo estructural.

Actuación cuando se detecta un futuro riesgo estructural. El control no permite que se inicie o prosiga la tarea. Las tareas previas siguen utilizando las etapas en función del tipo de tarea. El bloqueo de la tarea, que produciría un riesgo estructural, perdura hasta que se determina que no se producirá riesgo estructural.

Entonces, la actuación del control emula un procesado serie de las tareas, ya que en caso de riesgo estructural serializa el procesado de las tareas.

Como ejemplo, seguidamente se describe el diseño del control en un procesador segmentado en el cual pueden producirse riesgos estructurales.

Procesador segmentado

Si lo permite la capacidad de integración y el tiempo de ciclo, en los procesadores se intenta minimizar la posibilidad de conflicto de recursos. Ahora bien, esto no siempre es posible desde un punto de vista coste/rendimiento o no lo facilita la tecnología utilizada.

Un caso que requiere atención, desde el punto de vista de recursos, son los caminos de acceso al banco de registros. La técnica descrita de retardos, que ha permitido eliminar el conflicto de escritura entre instrucciones load e instrucciones ENT (Figura 2.42), no siempre es útil. Notemos que se produce un riesgo debido a que la latencia de cálculo es distinta. En las instrucciones ENT es 1 ciclo (ALU) y en las instrucciones load son 2 ciclos (ALU más MD).

Latencia de cálculo. Número de ciclos desde que se dispone de los datos para efectuar un cálculo hasta que finaliza el cálculo.

La latencia de cálculo de una operación de coma flotante es mayor que la latencia de cálculo de las instrucciones load o ENT. Entonces, en un procesador que disponga de unidades funcionales de coma flotante (UF_{CF}) el conflicto debido al camino de escritura al banco de registros, se produce entre más tipos de instrucciones.

En la Figura 2.44 se muestra una posible interpretación de una instrucción que opera con números en coma flotante. El cálculo se ha segmentado en 4 etapas (E1, E2, E3, E4) y la latencia de interpretación de la instrucción son 8 ciclos, escribiéndose en el banco de registros en el ciclo 8.

	ciclos							
Tipo de instrucción	1	2	3	4	5	6	7	8
C.F	CP	BUS	D/L	E1	E2	E3	E4	ES

Figura 2.44 *Interpretación segmentada de una instrucción de coma flotante. La unidad de cálculo está segmentada en 4 etapas (E1, E2, E3, E4).*

Si se aplica la técnica de retardos a la segmentación de la Figura 2.39, en las instrucciones ENT la fase de escritura debe retardarse 3 ciclos y en las instrucciones load 2 ciclos.

Latencias de cálculo de 4 ciclos son usuales en operaciones de suma y multiplicación de coma flotante. Sin embargo, otras instrucciones como la de división en coma flotante, además de no estar segmentadas, pueden tener latencias de cálculo superiores a 10 ciclos, siendo la frecuencia de utilización bastante pequeña y en

algunas programas cero. Por tanto, en estos procesadores, es recomendable controlar el recurso de escritura al banco de registros en lugar de utilizar la técnica de retardos, ya que en la mayoría de instrucciones se incrementa la latencia de escritura en el banco de registros.

La gestión o control del recurso de escritura al banco de registros requiere conocer el tipo de instrucción. El tipo de instrucción se conoce en la etapa D/L, que es donde se decodifica y el conflicto se produciría después de la etapa D/L. Por ello, en lugar de controlar el inicio de interpretación de una instrucción, lo que se controla es si una instrucción que está en la etapa D/L prosigue la interpretación en el siguiente ciclo o permanece en la etapa D/L. Esto es, la instrucción se retiene en la etapa y se bloquea la interpretación segmentada de esta instrucción hasta que desaparecen los riesgos.

Etapa de retención. Etapa en la que se retiene una instrucción.

Por ahora supondremos en los diseños segmentados que una etapa previa a la etapa de retención no utiliza información producida por una instrucción que ya ha pasado por la etapa de retención. En estas condiciones, para retener o bloquear una instrucción en una etapa, es suficiente con no actualizar los registros de desacoplo de entrada de la etapa.

Las acciones necesarias para gestionar riesgos estructurales son: a) control del inicio o proseguir y b) indicar la validez de la información procesada por una etapa.

El lenguaje máquina tiene instrucciones que no modifican el estado del procesador. Nosotros utilizaremos la palabra nop para indicar una instrucción de este tipo.

Cuando en un ciclo se retenga la interpretación de una instrucción, utilizaremos la codificación de una instrucción nop en lugar de una señal de validez explícita, para indicar a la siguiente etapa que no procesa información válida. Notemos que una instrucción nop no utiliza los recursos MD y BR y por tanto, no modifica el estado del procesador. A esta acción la denominaremos inyección de una instrucción nop.

Como la etapa de retención es D/L también hay que retener a las instrucciones en las etapas CP y BUS ya que en el siguiente ciclo, debido a la retención en D/L, la etapa previa de cada una de ellas estará ocupada (Figura 2.45).

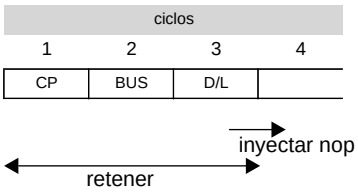


Figura 2.45 *Acciones cuando se detecta un riesgo estructural.*

En la Figura 2.46 se muestra, en un diagrama temporal, la interpretación de una secuencia de instrucciones donde se observa la acción de retención o bloqueo. La segmentación de las instrucciones se corresponde con la especificada en la Figura 2.40, donde las instrucciones load y ENT actualizan el banco de registros en ciclos distintos respecto del inicio de interpretación de la instrucción.

Una instrucción se representa utilizando una fila del diagrama cuando no está afectada por un riesgo estructural. Para dejar patente en el diagrama situaciones de retención y para representar la inyección de una instrucción nop, en cada situación de retención utilizaremos una nueva fila en las instrucciones afectadas. En esta nueva fila se inicia la representación a partir de la etapa en que se retiene la instrucción. La inyección de una instrucción nop se explícita mediante el acrónimo nop en lugar del acrónimo de la etapa.

Los ciclos de bloqueo los contabilizaremos en la salida del dispositivo segmentado. Entonces, en un procesador se contabiliza un ciclo de bloqueo en el ciclo que finaliza una instrucción nop.

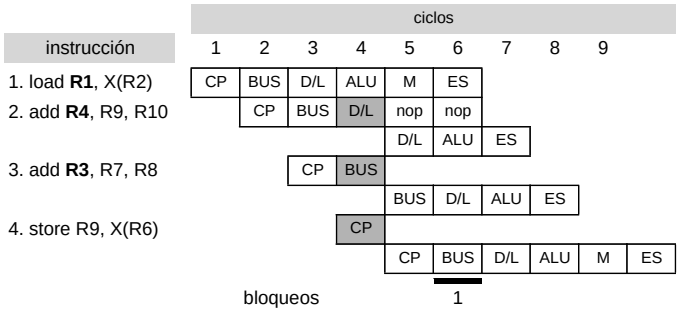


Figura 2.46 *Bloqueo de la interpretación segmentada de instrucción para controlar un riesgo estructural.*

En el ciclo 3 se decodifica una instrucción de tipo load. Entonces, se conoce que en el ciclo 6 se utiliza el camino de escritura al banco de registros. En el ciclo 4 se decodifica la 2ª instrucción que es de tipo ENT y utilizaría el camino de escritura al banco de registros también en el ciclo 6. Por tanto, la 2ª instrucción se retiene en la etapa D/L en el ciclo 4. Adicionalmente en este ciclo hay que retener, en la etapa que ocupan, a las instrucciones 3ª y 4ª. También, al finalizar el ciclo 4 se inyecta una instrucción nop desde la etapa D/L hacia la etapa ALU.

MODOS EN UN ELEMENTO DE DESACOPLO

La necesidad de retener instrucciones en algunas etapas e inyectar una instrucción nop cuando se produce una retención requiere arropar a los registros de desacoplo con lógica que permita controlar estas acciones.

En la Figura 2.47 se muestran tres posibles modos de funcionamiento de los elementos de desacoplo. Un elemento de desacoplo consta de un registro de desacoplo y multiplexores. Los multiplexores se utilizan para implementar los modos de funcionamiento: normal, bloqueo y burbuja.

Modo normal. En este modo la información en la entrada del registro de desacoplo se transfiere a la salida cuando se produce un flanco ascendente de la señal de reloj.

Modo bloqueo. La salida del registro de desacoplo no se modifica cuando se produce un flanco ascendente de la señal de reloj. Para implementar este modo se ha añadido el multiplexor MUXBloq en la entrada del registro de desacoplo. Una entrada del multiplexor es la salida del propio registro de desacoplo (datos y control) y la otra entrada está conectada a la salida de la lógica de la etapa. La salida del multiplexor MUXBloq es la entrada del registro de desacoplo si no se utiliza el modo que se describe seguidamente.

Modo burbuja. En la salida del registro de desacoplo hay una nop después de que se produzca un flanco ascendente de la señal de reloj. Para implementar este modo se ha añadido el multiplexor MUXNop en la entrada del registro de desacoplo. Este multiplexor puede estar en el camino de datos o en el camino de las señales de control. El primer caso se utiliza cuando la instrucción aún no ha sido decodificada (Figura 2.47 a) y el segundo caso se utiliza

cuando la instrucción ya ha sido decodificada (Figura 2.47 b). Una entrada del multiplexor MUXNop es la instrucción nop (NOP, NOPDEC) y la otra entrada es la salida del multiplexor MUXBloq o MUXBloqB.

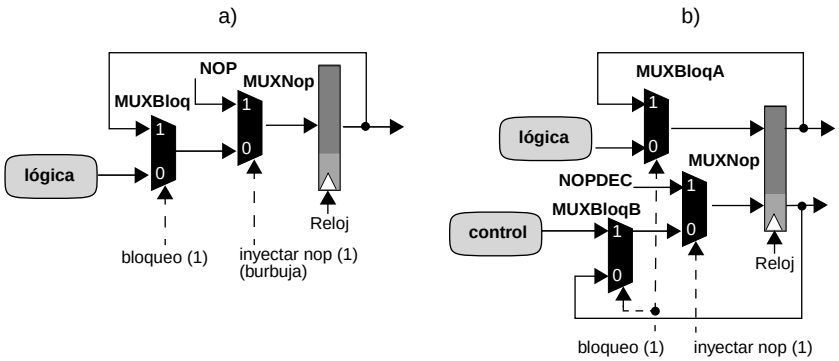


Figura 2.47 *Lógica adicional en los registros de desacoplo para implementar los modos: normal, bloqueo y burbuja.*

CONTROL DE RIESGOS ESTRUCTURALES

En la Figura 2.48 se muestra un esquema simplificado de las etapas CP, BUS y D/L. Los registros de desacoplo entre las etapas CP y BUS, las etapas BUS y D/L y las etapas D/L y ALU se han etiquetado respectivamente como CP, BUS_DL y DL_ALU.

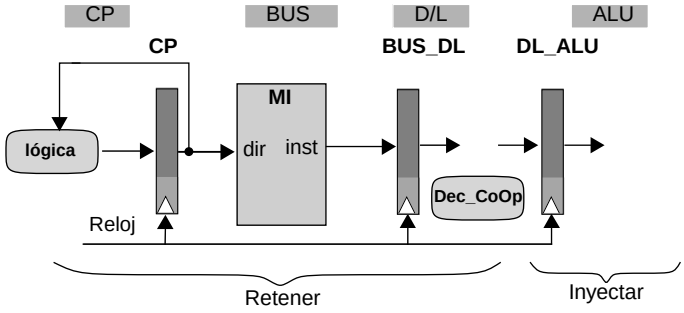


Figura 2.48 *Esquema simplificado de las etapas CP, BUS y D/L.*

Para retener una instrucción en una etapa se utilizan multiplexores en algunos de los registros de desacoplo de las etapas afectadas.

En la Figura 2.49 se muestran de forma esquemática las etapas CP, BUS y D/L de un camino de datos segmentado y la lógica utilizada para retener la información en la etapa CP, BUS y D/L e inyectar una instrucción nop desde la etapa D/L.

El multiplexor MUXCPBloq permite seleccionar entre la información procesada en la etapa o la información almacenada en el registro de desacoplo CP. Observemos que la etapa CP tiene como registro de desacoplo de entrada y salida el registro CP.

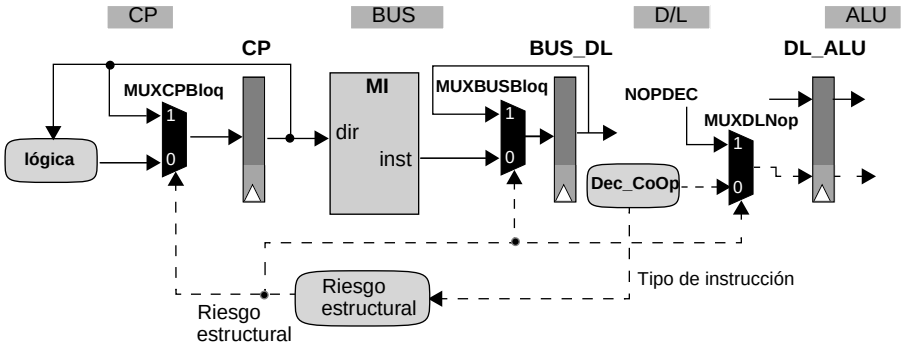


Figura 2.49 *Lógica de retención e inyección de instrucciones nop.*

Del análisis de la Figura 2.41 habíamos concluido que después de una instrucción load la latencia de inicio 1 está prohibida si la instrucción es de tipo ENT.

Recordemos que una latencia prohibida indica el número de ciclos, a partir del ciclo actual, en el que iniciar una tarea produce un conflicto.

Una implementación hardware para detectar una latencia prohibida es utilizar un registro de desplazamiento de profundidad igual a la latencia prohibida. Cuando se quiere prohibir una latencia se inyecta un 1 por la cola del registro de desplazamiento. En caso contrario se inyecta un cero. La cabeza del registro de desplazamiento es analizada por el control de riesgos y cuando se observa un 1 se actúa en consonancia. En la Figura 2.50 se muestra una implementación hardware para detectar una latencia prohibida de valor 3.

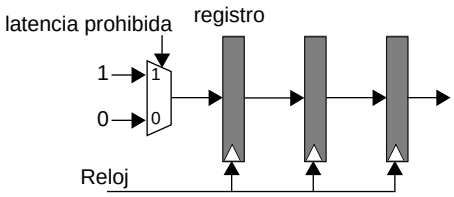


Figura 2.50 *Circuito para detectar una latencia prohibida igual a tres.*

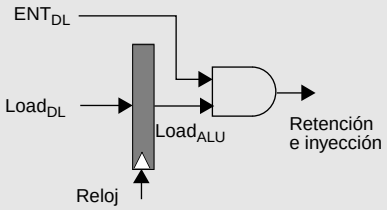
En el procesador que se está analizando, la latencia prohibida que hay que gestionar es 1. Por tanto, es suficiente con un registro. Cuando en la etapa D/L se decodifica una instrucción tipo load se inyecta un uno en el circuito de detección de latencias prohibidas. Adicionalmente en la etapa D/L se observa la salida del circuito. Entonces, si la instrucción que se está decodificando es de tipo ENT y se detecta una latencia prohibida hay que bloquear la interpretación de la instrucción.

Ejercicio

En la etapa D/L se generan señales que indican el tipo de instrucción que está ocupando la etapa (ENT_{DL}, $Load_{DL}$). Diseñe un circuito para controlar la latencia prohibida 1 entre una instrucción load y una instrucción ENT.

Respuesta

Como la latencia prohibida es 1, es suficiente un registro en el circuito de detección. La entrada de este registro es la señal $Load_{DL}$, que indica que en la etapa D/L hay una instrucción load.



La salida del circuito es la operación AND de la señal del circuito de detección de latencia prohibida y la señal ENT_{DL} que indica si en la etapa D/L hay una instrucción ENT.

Control del camino de datos

Una vez se ha iniciado una operación los datos fluyen a través de las etapas del dispositivo segmentado utilizando el encaminamiento y secuenciamiento que indica la tabla de reserva.

En un ciclo determinado es posible que una o varias etapas no procesen información. Por ello, se dispone de una señal que indica a las etapas si están procesando información válida (Figura 2.51). Un ejemplo de utilización de esta señal es en etapas que contienen elementos de memorización. Estos elementos no deben actualizarse si no se procesa información válida. Para efectuar esta acción de control es suficiente asociar un bit a la información de control y propagar este bit por las mismas etapas que se propagan los datos.

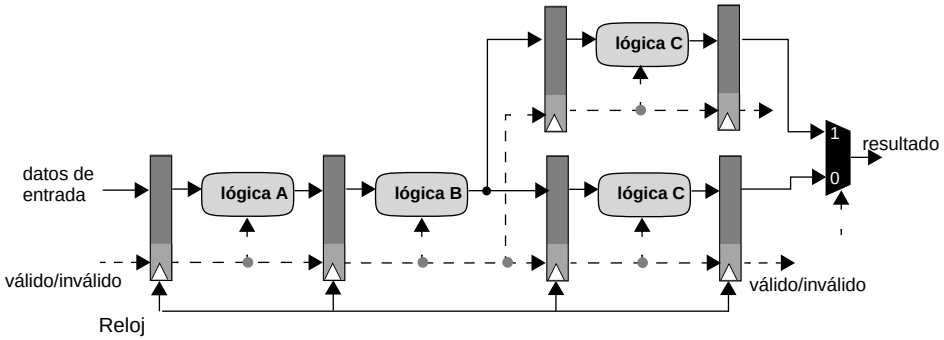


Figura 2.51 *Señal de validación de información.*

También hay que controlar el encaminamiento de información entre etapas y en lo que respecta a la lógica hay que seleccionar, en función de la operación, el procesado en cada etapa. Un ejemplo de este último caso es cuando en una etapa se dispone de un sumador y lógica para efectuar la operación AND bit a bit.

En el diseño del control de la lógica de cada etapa y del encaminamiento entre etapas se distinguen dos casos extremos: a) control estacionario en el tiempo y b) control estacionario en los datos.

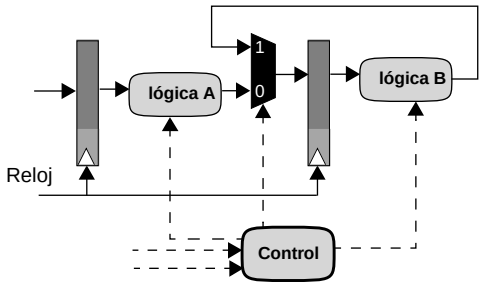


Figura 2.52 *Control estacionario en el tiempo.*

Control estacionario en el tiempo. Un elemento centralizado que en cada ciclo envía las señales a todos los elementos del dispositivo segmentado que lo requieren (Figura 2.52).

Control estacionario en los datos. La información de control fluye con los datos y el control está integrado en la segmentación. Cada vez que se inicia una operación se generan las señales necesarias, para que en cada etapa sean interpretadas y controlen la lógica y el encaminamiento. Una etapa a su vez puede generar o procesar información de control para la siguiente etapa (Figura 2.53).

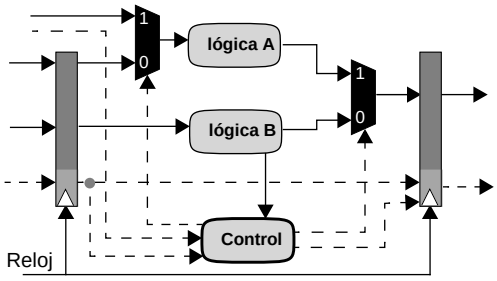


Figura 2.53 *Control estacionario en los datos.*

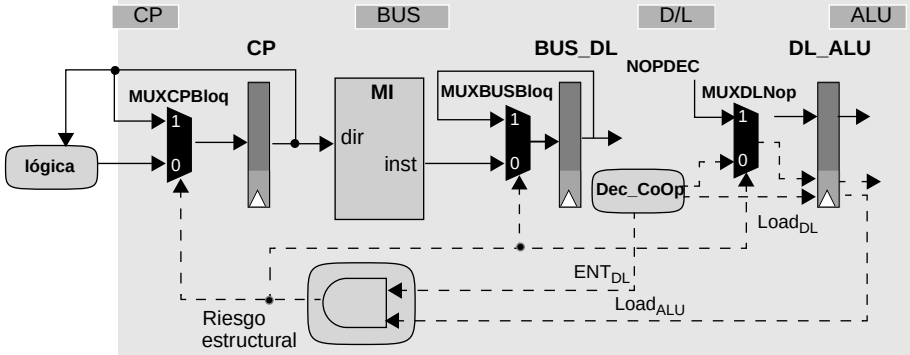
EL control estacionario en los datos se utiliza extensamente en dispositivos segmentados complejos ya que permite descentralizar el control y ello simplifica el diseño. También es de utilidad cuando la frecuencia de funcionamiento es elevada puesto que el tiempo de propagación de las señales por los cables se hace más significativo.

Ejercicio

Integre en el camino de datos de la Figura 2.49 el control de riesgos diseñado en el ejercicio previo.

Respuesta

La integración es simple ya que se pueden utilizar los registros de desacoplo ente las etapas para detectar latencias prohibidas. En la siguiente figura se muestra el diseño resultante, donde se puede decir que el control es estacionario en los datos.



MEMORIA CACHE

En los procesadores se utiliza una organización jerárquica de memoria para adecuar las necesidades contrapuestas de latencia reducida de acceso a memoria y gran capacidad de almacenamiento.

Cuando el procesador efectúa una referencia a memoria se accede en primer lugar a la cache. Si la cache almacena el contenido de la dirección accedida finaliza el acceso. En caso contrario, el controlador de la jerarquía de memoria gestiona el acceso a los siguientes niveles.

Organización de la cache

La memoria cache consta de posiciones de almacenamiento denominadas contenedores y en cada contenedor se distinguen, en un primer nivel de detalle, el campo de etiqueta y el campo dato. Un contenedor de cache se identifica mediante una dirección

que denominaremos identificador del contenedor o entrada. Este identificador se obtiene a partir de la dirección de acceso a memoria.

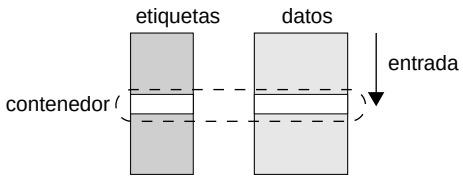


Figura 2.54 *Organización de la cache.*

Bloque o bloque de memoria. Conjunto contiguo de posiciones de memoria direccionable.

El espacio de direcciones de memoria se divide en bloques disjuntos de igual tamaño y cualquier dirección tiene un bloque asociado. Entonces, la dirección de un bloque se determina a partir de la dirección accedida utilizando la siguiente expresión:

$$\text{dirección}_{\text{bloque}} = \text{dirección} / B$$

donde B es el tamaño de bloque en bytes y es una potencia de dos.

El bloque es la unidad de mapeo en cache. Esto es, un contenedor almacena un bloque de datos y la etiqueta.

El número de contenedores de cache es

$$NC = C / B$$

donde C es la capacidad de datos de la cache medida en bytes y es una potencia de dos.

Dependiendo de la función de correspondencia (traducción) utilizada para asociar una dirección de memoria a un contenedor de cache se distinguen distintas organizaciones de cache. Nosotros supondremos la organización denominada mapeo directo, en la cual el contenedor donde se puede almacenar el contenido de una dirección de memoria está unívocamente determinado.

Cuando se utiliza mapeo directo, la expresión que relaciona la dirección de un bloque con el contenedor de cache donde se almacena es:

$$\text{entrada} = \text{dirección}_{\text{bloque}} \ \textbf{mod} \ NC$$

En estas condiciones, direcciones de bloque múltiplos del número de contenedores se mapean en el mismo contenedor. Esta característica determina que se produzcan conflictos. Entendemos por conflicto que un bloque sea expulsado de cache porque se accede a otro bloque que debe almacenarse en el mismo contenedor. Se dice que se ha producido un fallo debido a un conflicto cuando posteriormente se vuelve a cargar en cache el bloque expulsado (fallo por conflicto).

Para poder identificar de forma biunívoca un bloque se utiliza el campo etiqueta del contenedor de cache. El contenido de este campo es el siguiente valor.

$$\text{etiqueta} = \text{dirección}_{\text{bloque}} \ / \ NC$$

Dada una dirección, el elemento accedido, del campo de datos del contenedor, se determina mediante la siguiente expresión.

$$\text{elemento} = (\ \text{dirección}\ \textbf{mod}\ B\ ) \ / \ TE$$

donde TE es el tamaño del elemento en bytes y es potencia de dos, el tamaño de bloque es múltiplo del tamaño de un elemento y los bytes de un elemento no están en dos bloque adyacentes.

Como $C = 2^c$, $B = 2^b$ y $TE = 2^e$, las operaciones anteriores son simples de implementar (Figura 2.55). Dada una dirección, los e bits menos significativos no se utilizan, los siguientes b-e bits se utilizan para determinar el elemento accedido (EL) y los siguientes c-b bits indican el contenedor (EN). Los restantes bits (ETIQ) se almacenan en el campo etiqueta del contenedor.

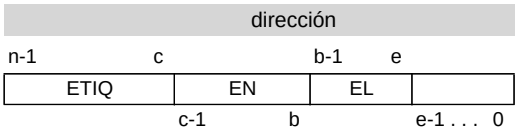


Figura 2.55 *Campos de la dirección de memoria utilizados para acceder a cache.*

En la Figura 2.56 se muestra un esquema lógico del acceso a cache. Suponemos que un acceso por defecto lee o escribe 8 bytes. También, suponemos que un bloque contiene dos elementos de 8 bytes cada uno.

Un banco de memoria es un conjunto de posiciones de almacenamiento y supondremos que tiene un único camino de acceso, que se utiliza para leer o escribir. Los elementos cuyo símbolo es

un triángulo funcionan como conmutadores y permiten el paso de la señal únicamente en un sentido. Se utilizan para desconectarse o conectarse al puerto de entrada/salida de cada banco de memoria.

Para almacenar el campo datos del contenedor se utilizan dos bancos de memoria y en cada banco de memoria se almacena un elemento del bloque. Para almacenar el campo etiquetas se utiliza otro banco de memoria.

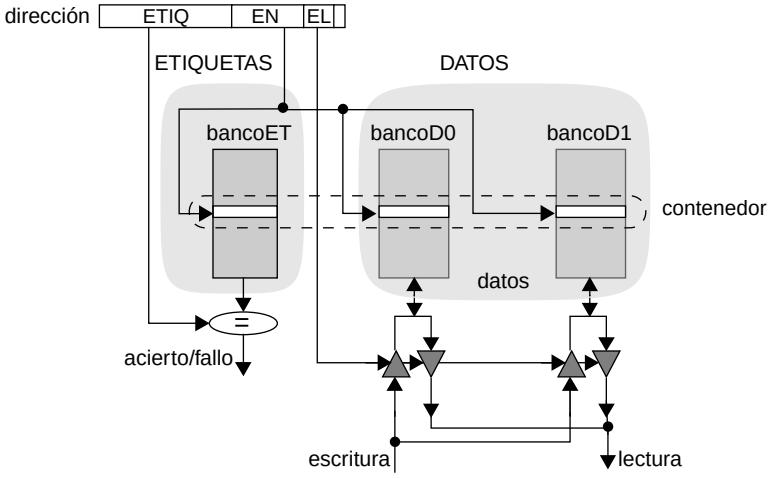


Figura 2.56 *Esquema lógico del acceso a una cache con mapeo directo.*

Utilizando el campo EN de la dirección se conoce el contenedor donde puede estar el bloque. Se lee el campo etiqueta del contenedor y su contenido se compara con el campo ETIQ de la dirección. Si son iguales se ha detectado un acierto. En caso contrario el bloque no está almacenado en cache y se dice que se ha producido un fallo.

Si se detecta un acierto los bits del campo EL indican el elemento accedido.

Segmentación del acceso

El acceso a cache, tanto en lecturas como en escrituras, consta de 2 fases: a) comprobación de la etiqueta del contenedor y b) acceso al campo de datos del contenedor. La segunda fase hay que efectuarla cuando se detecta un acierto.

En la fase de comprobación de etiquetas se lee el campo etiqueta del contenedor de cache y se compara con el campo etiqueta de la dirección (ETIQ). Si la comparación indica igualdad se ha producido un acierto y en la segunda fase se lee/escribe el campo de datos del contenedor. En caso contrario se accede a memoria para obtener el dato.

Las fases de acceso a cache pueden efectuarse en un ciclo o requerir varios ciclos de reloj. En este apartado supondremos que cada fase requiere un ciclo y analizaremos el acceso a una cache de datos tanto en accesos de lectura y como de escritura. No analizaremos las acciones que deben efectuarse cuando se produce un fallo de cache. Por tanto, obviaremos el campo del contenedor que almacena información sobre el estado del bloque: indica si el contenedor no almacena información válida o si, en un cache con escritura retardada, el contenido del contenedor ha sido actualizado. También obviaremos la conexión e intercambio de información con memoria.

En la Figura 2.57 se muestra la segmentación del acceso a cache en un acceso a memoria. En el primer ciclo se efectúa una lectura en el banco que contiene el campo etiquetas y se compara el valor leido con el campo ETIQ de la dirección (etapa ET). En el segundo ciclo, si se ha producido un acierto, se accede a uno de los bancos que almacena el campo datos (etapa DAT). En una instrucción load se lee una posición de almacenamiento del campo datos y en una instrucción store se escribe una posición de almacenamiento del campo datos. Notemos que en un acceso sólo es necesario leer/escribir en uno de los bancos que almacena el campo datos.

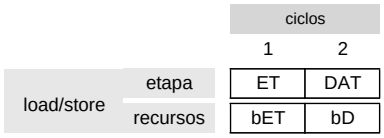


Figura 2.57 *Segmentación lineal en dos etapas del acceso a cache para las instrucciones load y store. bET indica banco de etiquetas y bD banco de datos (bD0, bD1).*

Como las dos instrucciones de acceso a memoria tienen el mismo patrón de utilización de los recursos, cuando se accede con latencia de inicio igual a 1 no se producen conflictos.

Camino de datos segmentado

En la Figura 2.58 se muestra el camino de datos utilizado para acceder a los recursos que constituyen la cache. Por claridad del dibujo no se muestra la señal de reloj que actualiza los registros de desacoplo ni la entrada de reloj en los bancos de memoria que almacenan el campo etiqueta y el campo datos. También, en los bancos de datos se dibuja de forma separada el camino que transporta el dato que se lee ($dato_L$) y el camino que transporta el dato que se escribe ($dato_E$).

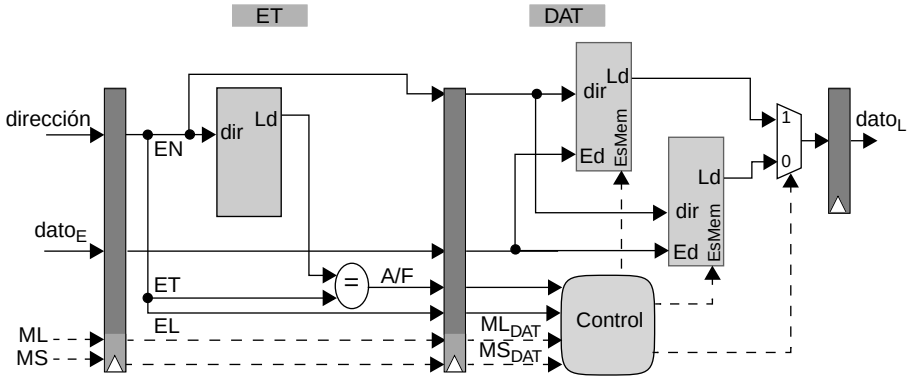


Figura 2.58 *Camino de datos segmentado en dos etapas del acceso a cache.*

Cuando se detecta un acierto, el control se encarga de seleccionar, mediante el campo EL de la dirección, el banco que se utiliza. Las señales ML y MS indican respectivamente si la instrucción es un load o un store. El subíndice en estas señales indica la etapa. La señal EsMem de un banco se activa cuando se escribe en el banco.

EJEMPLOS

En este apartado se presentan 4 ejemplos de segmentación. El primer ejemplo describe la segmentación de una unidad funcional de coma flotante cuando los recursos disponibles son escasos y se analiza la ganancia respecto de una implementación serie. El segundo ejemplo es la segmentación del proceso de interpretación de instrucciones cuyos operandos están en memoria. El tercero es la segmentación del acceso a cache reduciendo la latencia de interpretación de las instrucciones load. El último ejemplo es la utilización de un bus segmentado y el acceso a los bancos de memoria por parte del controlador de memoria.

Unidad funcional de coma flotante

Un número en coma flotante se representa mediante dos componentes, el significando y el exponente y de forma implícita se supone una base. El significando es un número con signo y se representa en signo y magnitud. Entonces, un número en coma flotante se expresa como

$$x = (-1)^{S_x} \times M_x \times 2^{E_x}$$

donde 2 es la base, E_x es el exponente, $(-1)^{S_x}$ es el signo, perteneciendo S_x al conjunto {0,1}, y M_x denota la magnitud del significando.

En la representación de un número en coma flotante existen grados de libertad, como por ejemplo el número de bit utilizados para representar cada componente y la forma de efectuar el redondeo, que dan lugar a que un programa calcule resultados distintos en función del computador que se utiliza. Por ello, se ha desarrollado la norma IEEE Floating-point Standard 754 que es ampliamente utilizada. En la norma se incluye el tratamiento, por ejemplo de situaciones de irrepresentabilidad, cálculos inválidos y división por cero.

En la siguiente tabla se describen las fases de las operaciones de suma y multiplicación de números representados en coma flotante.

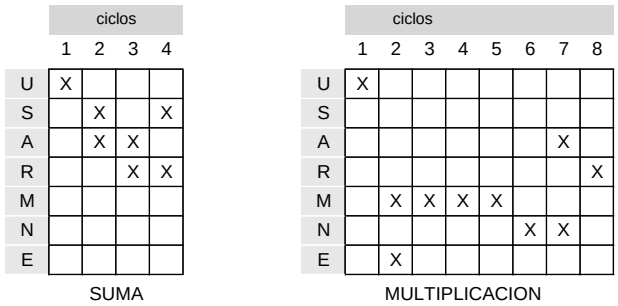
SUMA	
Fases	Recursos utilizados
Resta de exponentes	sumador pequeño
Alineación del significando	lógica de desplazamiento
Suma de mantisas	sumador grande
Normalización	lógica de desplazamiento
Redondeo	lógica de redondeo

MULTIPLICACION	
Fases	Recursos utilizados
Multiplicación de magnitudes	matriz de sumadores sumador con propagación del acarreo
Suma de exponentes	sumador pequeño
Normalización	lógica de desplazamiento
Redondeo	lógica de redondeo

En la siguiente tabla se relacionan las etapas de una unidad funcional segmentada de coma flotante y la función que realizan.

Etapa	Función
A	suma
R	redondeo
S	desplazamiento
M	primera etapa del multiplicador
N	segunda etapa del multiplicador
U	formateo
E	comprobación de las condiciones de excepción antes de multiplicar

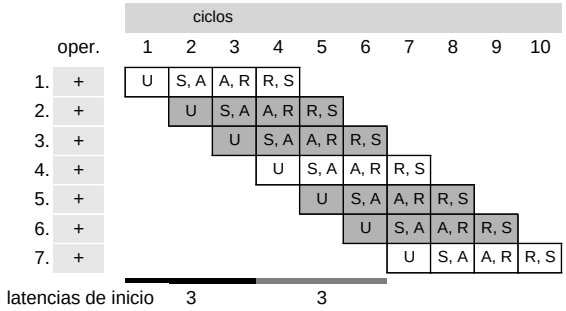
La utilización de las etapas para efectuar las operaciones de suma y multiplicación de números representados en coma flotante se muestra en las siguientes tablas de reserva. La latencia de la operación suma son 4 ciclos y la latencia de la operación multiplicación son 8 ciclos.



Las dos operaciones utilizan recursos de forma repetida debido a que los recursos hardware disponibles son limitados.

Pregunta 1: *Suponiendo que una operación se inicia lo más pronto posible, calcule una secuencia de latencias de inicio para una secuencia de operaciones suma.*

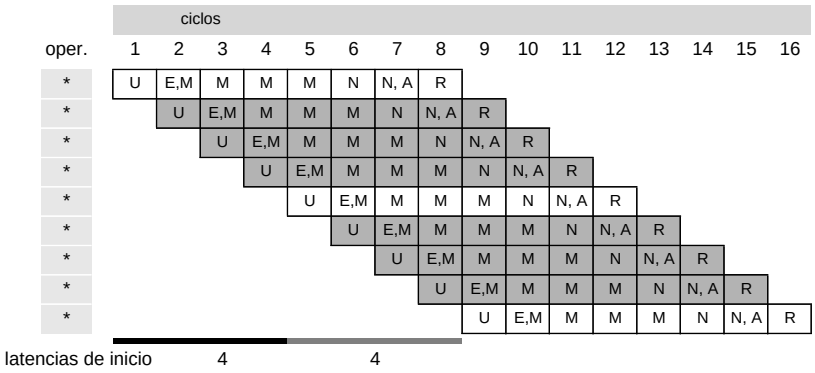
Respuesta: En la siguiente figura se muestra el diagrama temporal donde se han tramado las operaciones que dan lugar a conflictos. El conflicto de la 2ª y 5ª iniciación es debido al recurso A y el conflicto de la 3ª y 6ª iniciación es debido al recurso S.



Iniciando una nueva operación lo antes posible, la secuencia de latencias de inicio para la operación suma es (3) y la latencia media de inicio es 3.

Pregunta 2: *Suponiendo que una operación se inicia lo más pronto posible, calcule una secuencia de latencias de inicio para una secuencia de operaciones producto.*

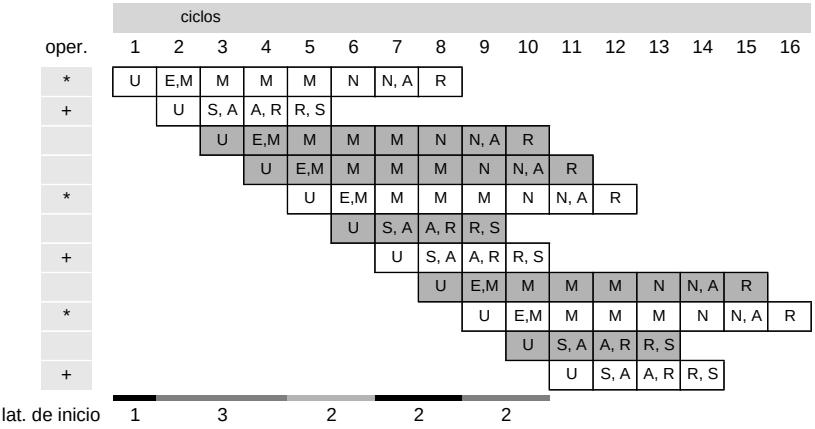
Respuesta: En el diagrama temporal se muestran las iniciaciones que no son factibles tramando la operación. En todos los casos el conflicto es debido al recurso M.



La secuencia de latencias es (4) y la latencia de inicio es 4.

Pregunta 3: *Suponiendo que una operación se inicia lo más pronto posible, calcule una secuencia de latencias de inicio para una secuencia entrelazada una a una de operaciones producto y suma.*

Respuesta: En la figura se muestra una secuencia de latencias de inicio cuando se entrelazan una a una operaciones de multiplicación y suma.



En los ciclos 3 y 4 no se puede iniciar una operación producto, ya que se producen conflictos en la utilización de la etapa M en los ciclos 4 y 5 respectivamente. En el ciclo 6 no se puede iniciar una

operación suma, ya que se produce un conflicto en la utilización del recurso A en el ciclo 7. Por último, en los ciclos 8 y 10 no se pueden iniciar operaciones de multiplicación y de suma respectivamente debido a conflictos en la utilización de los recursos M y A.

Suponiendo que el número de operaciones de la secuencia es grande, la secuencia de latencias de inicio que se repite es (2) y la latencia media de inicio es igual a 2.

Pregunta 4: *En la UF de coma flotante descrita previamente se procesan las siguientes secuencias de operaciones: a) sumas, b) multiplicaciones y c) entrelazado una a una de ambas operaciones. Calcule la ganancia que se obtiene con el diseño segmentado respecto del diseño serie, en los tres casos.*

No tenga en cuenta en la evaluación el retardo de los registros de desacoplo y suponga que el tiempo de procesado de la lógica del dispositivo serie se ha distribuido uniformemente entre las etapas del dispositivo segmentado.

Respuesta: a) Secuencia de operaciones de SUMA

Ganancia = 4 / 3 = 1.33

ya que el dispositivo segmentado tiene 4 etapas y finaliza una operación de suma cada 3 ciclos.

El dispositivo segmentado es un 33% más rápido que el dispositivo serie en procesar una secuencia de operaciones de suma.

b) Secuencia de operaciones de MULTIPLICACION

Ganancia = 8 / 4 = 2

ya que el dispositivo segmentado tiene 8 etapas y la latencia media de inicio son 4 ciclos.

El dispositivo segmentado tarda la mitad de tiempo que el dispositivo serie en procesar una secuencia de operaciones de multiplicación.

c) Secuencia entrelazada de operaciones de suma y de multiplicación

Ganancia = (8 + 4) / (2 + 2) = 3

ya que el encadenamiento de una operación de multiplicación y suma procesadas de forma secuencial son 12 ciclos y la latencia media de inicio en el dispositivo segmentado son 2 ciclos para cada operación. O sea 4 ciclos para cada pareja suma-multiplicación.

El dispositivo segmentado procesa 3 veces más rápido una secuencia de operaciones multiplicación-suma de coma flotante que el dispositivo serie.

Instrucciones con datos y resultados en memoria

El lenguaje máquina de algunos procesadores incluye instrucciones aritmético-lógicas donde los datos fuente y/o el resultado se almacenan en memoria. Ejemplos de estos tipos de instrucciones son:

1) addR ax, 4 [di] 2) addM 8 [di], ax

La primera instrucción (addR) tiene como datos fuente el contenido del registro ax y el contenido de una posición de memoria, cuya dirección se calcula como 4 más el contenido del registro di,

$$ax^v = ax^v + M[4 + di^v]$$

donde el superíndice v indica contenido del registro y M posición de memoria. El resultado de la suma se almacena en el registro ax.

En la segunda instrucción (addM) los datos fuente son el contenido de la posición de memoria $8 + di^v$ y el contenido del registro ax. El resultado de la suma se almacena en la posición de memoria $8 + di^v$.

$$M[8 + di^v] = ax^v + M[8 + di^v]$$

Pregunta 1: *Las fases de una instrucción son: determinar una dirección, búsqueda de la instrucción, decodificación, lectura de los datos fuente, ejecución y escritura del resultado.*

Para las instrucciones addR y addM especifique en una tabla cuales son las acciones que deben efectuarse en cada fase.

Respuesta: En la siguiente tabla se especifican las acciones efectuadas en las fases de interpretación.

La diferencia entre las dos instrucciones está en las acciones que se efectúan en la fase de escritura del resultado.

	addR ax, 4 [di]	addM 8 [di], ax
Fases	Acciones	Acciones
Determinar la dirección de la instrucción	incrementar el CP	
Búsqueda de la instrucción	acceder a memoria	
Decodificación	decodificar	
Lectura de los datos fuente	leer del banco de registros calcular la dirección efectiva acceder a memoria	
Ejecución	efectuar el cálculo	
Escritura del resultado	escribir en el banco de registros	escribir en memoria

Para construir un camino de datos segmentado disponemos de los recursos mostrados en la siguiente tabla. Suponga que el retardo de cada uno de los recursos es similar y que la lectura del banco de registros se puede efectuar en el mismo ciclo que la decodificación de instrucciones. Además supondremos que un cálculo de la ALU no se puede utilizar como dirección para acceder a memoria.

Recursos	Funcionalidad
+	sumador que se utiliza para el secuenciamiento implícito
+@	sumador para calcular direcciones efectivas
MI	memoria que almacena instrucciones, con un sólo camino de acceso
BRL	dos caminos de lectura al banco de registros
BRE	un camino de escritura al banco de registros
ALU	unidad aritmético-lógica
MD	memoria que almacena datos, con un sólo camino de acceso
D	decodificador

Pregunta 2: *Utilizando la contestación de la pregunta anterior, los recursos disponibles y su retardo que es similar en todos ellos, proponga una segmentación en etapas del proceso de interpretación de las instrucciones addR y addM.*

Respuesta: En la figura se muestra una segmentación en etapas del proceso de interpretación de las instrucciones.

		ciclos						
		1	2	3	4	5	6	7
addR	etapas	CP	BUS	D/L	CDE	ML	ALU	ME/ES
	recursos	+	MI	D / BRL	+@	MD	ALU	BRE
addM	etapas	CP	BUS	D/L	CDE	ML	ALU	ME/ES
	recursos	+	MI	D / BRL	+@	MD	ALU	MD

En la primer fila se identifican las etapas y en la segunda fila los recursos utilizados. En primer lugar están las etapas CP y BUS donde se utilizan los recursos + y MI. Posteriormente se decodifica la instrucción y se leen los datos almacenados en registros (D/L). Seguidamente se calcula la dirección del dato almacenado en memoria (CDE) y se accede a la memoria de datos (ML). En el siguiente ciclo se efectúa el cálculo especificado en la instrucción (ALU) y en el último ciclo se actualiza la posición de almacenamiento donde debe almacenarse el resultado (ME o ES).

Pregunta 3: *¿Existe posibilidad de riesgos estructurales cuando se interpreta una secuencias de instrucciones addR, addM o una secuencia entrelazada de ambas?.*

Respuesta: En las siguientes tablas de reserva se muestran los recursos utilizados en cada ciclo por cada tipo de instrucción. La instrucción tipo addR tiene una segmentación lineal.

	ciclos						
	1	2	3	4	5	6	7
+	X						
MI		X					
BRL			X				
+@				X			
ALU						X	
MD					X		
BRE							X

addR

	ciclos						
	1	2	3	4	5	6	7
+	X						
MI		X					
BRL			X				
+@				X			
ALU						X	
MD					X		X
BRE							

addM

En la tabla de reserva de la instrucción addM se observa la ocupación repetida del recurso MD. Entonces, se producen conflictos de recursos cuando se interpretan secuencias de instrucciones addM y cuando se entrelaza la interpretación de los dos tipos de instrucciones.

El lenguaje máquina también dispone de instrucciones aritmético-lógicas que operan con datos almacenados en registros e instrucciones de acceso a memoria que almacenan en un registro el contenido de una posición de memoria o almacenan en una posición de memoria el contenido de un registro. Ejemplos de ellas, son las siguiente tres instrucciones:

1) add bx, ax 2) load ax, X [di] 3) store bx, X [di]

que efectúan las siguientes operaciones:

1) $bx^v = bx^v + ax^v$ 2) $ax^v = M[X + di^v]$ 3) $M[X + di^v] = bx^v$

Otra instrucción es

push ECX

Esta instrucción tiene como dato implícito el contenido del registro ESP, el cual se decrementa en 4 y se almacena el resultado en el mismo registro. Además, se almacena el contenido del registro ECX en la posición de almacenamiento $M[ESP^v - 4]$.

1ª) $M[ESP^v - 4] = ECX^v$ 2ª) $ESP^v = ESP^v - 4$

Pregunta 4: *Proponga una segmentación para interpretar las instrucciones "add bx, ax", "load ax, X [di]", "store bx, X [di]" y "push ECX" que minimice los riesgos estructurales. Tenga en cuenta en el diseño que el lenguaje máquina también incluye las instrucciones de la segunda pregunta. Recuerde que un cálculo efectuado en la ALU no se puede utilizar como dirección para acceder a memoria.*

Respuesta: En la figura se muestran las etapas y los ciclos en que se ocupan los recursos. Los ciclos en que no se especifica ningún acrónimo son de retardo.

		ciclos						
		1	2	3	4	5	6	7
	etapas	CP	BUS	D/L	CDE	ML	ALU	ME/ES
add bx, ax	recursos	+	MI	D / BRL			ALU	BRE
load ax, X[di]	recursos	+	MI	D / BRL	+@	MD		BRE
store bx, X[di]	recursos	+	MI	D / BRL	+@			MD
push ECX	recursos	+	MI	D / BRL	+@		ALU	MD/BRE

Para minimizar los riesgos estructurales, se intenta utilizar los recursos en el mismo orden y en el mismo ciclo que en la segmentación de las instrucciones addR y addM (respuesta de la 2ª pregunta).

Los tres primeros ciclos de interpretación son idénticos para todas las instrucciones. Para calcular direcciones efectivas se utiliza el recurso “+@” y para efectuar operaciones aritméticas, cuyo resultado debe almacenarse en un registro, se utiliza el recurso ALU. La actualización del banco de registros o memoria se efectúa en el último ciclo de interpretación.

En las instrucciones store bx, X[di] y push ECX se puede considerar otra segmentación que se diferencia en el ciclo en el cual se actualiza memoria. En lugar de actualizar la memoria en el ciclo 7 se actualiza memoria en el ciclo 5. La denominación de las etapas no se modifica y el 5º ciclo se sigue denominando ML.

		ciclos						
		1	2	3	4	5	6	7
	etapas	CP	BUS	D/L	CDE	ML	ALU	ME/ES
store bx, X[di]	recursos	+	MI	D / BRL	+@	MD		
push ECX	recursos	+	MI	D / BRL	+@	MD	ALU	BRE

En ocasiones el lenguaje máquina también dispone de instrucciones cuyos datos fuente y destino están almacenados en posiciones de memoria. Además, en general todas las posiciones de almacenamiento accedidas en la instrucción son distintas. Un ejemplo de este tipo de instrucciones es:

addMM X[di], Y[ei], Z[hi]

que efectúa la siguiente operación:

$$M[X+di^V] = M[Y+ei^V] + M[Z+hi^V]$$

Para interpretar este tipo de instrucciones se añade un camino de datos de lectura al banco de registros.

Pregunta 5: *Proponga una segmentación, sin añadir recursos, para interpretar la instrucción addMM que minimice la latencia de interpretación.*

Respuesta: En la figura se muestran los recursos utilizados y los ciclos en que se ocupan.

	ciclos							
	1	2	3	4	5	6	7	8
etapas	CP	BUS	D/L	@	ML	ML	ALU	ME/ES
recursos	+	MI	D / BRL	+@	MD	MD	ALU	MD
					+@		+@	

En el tercer ciclo se leen los 3 registros ya que se dispone de 3 caminos de lectura. Los ciclos 4 y 5 se utilizan para obtener el dato M[Z+hi^{V}]. Los ciclos 5 y 6 se utilizan para obtener el dato M[Y+ei^{V}]. En el ciclo 7 se calcula el resultado y la dirección de memoria donde se almacenará. Por último, en el ciclo 8 se almacena el resultado en memoria.

En algunos ciclos se utilizan dos recursos concurrentemente y los recursos “+@” y MD se utilizan tres veces.

Reducción de la latencia del load en el acceso a cache

En un diseño de partida el acceso a cache de las instrucciones load y store se segmenta en 2 etapas: a) comprobación de la etiqueta del contenedor y b) acceso al campo datos del contenedor para leer o escribir el elemento accedido. La siguiente figura muestra las etapas descritas y los recursos utilizados: banco de etiquetas (bET) y bancos de datos (bD, bD0, bD1).

		ciclos	
		1	2
load/store	etapa	ET	DAT
	recursos	bET	bD

En el caso de las instrucciones load, la latencia de acceso a cache puede reducirse en un ciclo. Para ello, se accede concurrentemente al campo etiquetas y al campo datos de un contenedor, recursos bET y bD respectivamente. Entonces, en función de si se produce acierto o fallo se suministra el dato.

En el caso de una instrucción store no es posible reducir la latencia, ya que antes de actualizar el campo datos es necesario comprobar si el dato está almacenado en cache. Si no se comprueba podríamos estar escribiendo en otra dirección.

Teniendo en cuenta la posibilidad descrita, la segmentación del acceso a cache para las instrucciones load y store se muestra en la siguiente figura. En la instrucción load etiquetamos a la única etapa como ET / DAT.

		ciclos	
		1	2
Load	etapa	ET / DAT	
	recursos	bET / bD	

		ciclos	
Store	etapa	ET	DAT
	recursos	bET	bD

La segmentación del proceso de interpretación de las instrucciones se segmenta de la forma que muestra la siguiente figura.

	ciclos					
instrucción	1	2	3	4	5	6
load	CP	BUS	D/L	ALU	ET/DAT	ES
store	CP	BUS	D/L	ALU	ET	DAT

Pregunta 1: *Muestre si se producen conflictos en la interpretación de una secuencia entrelazada de instrucciones load y store. Justifique la contestación.*

Respuesta: Como los patrones de utilización de los recursos son distintos, se producen conflictos cuando se entrelazan instrucciones load y store. En la figura se muestra una secuencia de instrucciones donde se produce un conflicto en el ciclo 7, ya que una instrucción store accede al recurso bD y una instrucción load posterior también requiere acceso al recurso bD.

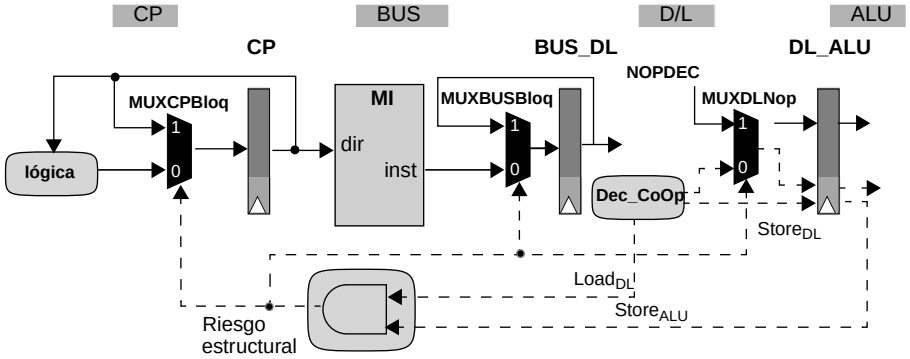
	ciclos							
instrucción	1	2	3	4	5	6	7	8
1. load **R1**, X(R2)	CP	BUS	D/L	ALU	ET/DAT	ES		
2. store R9, X(R6)		CP	BUS	D/L	ALU	ET	DAT	
3. load **R3**, X(R5)			CP	BUS	D/L	ALU	ET/DAT	ES
							conflicto	

Pregunta 2: *Diseñe un control que permita gestionar el riesgo estructural desde la etapa D/L. Esto es, en la etapa D/L y previas se retiene a las instrucciones hasta que no se produzcan riesgos si se prosigue la interpretación de la instrucción que ocupa la etapa D/L.*

Respuesta: Necesitamos un hardware que nos permita detectar una latencia prohibida igual a 1 después de que en la etapa D/L se decodifique una instrucción store. El hardware es simplemente un registro.

En la etapa D/L se observa la salida del circuito de detección de latencia prohibida. Entonces, cuando se detecta una latencia prohibida y en la etapa D/L hay una instrucción load hay que retener a las instrucciones en las etapas D/L, BUS y CP e inyectar una instrucción nop desde la etapa D/L.

En la siguiente figura se muestra el camino de datos con el control del riesgo. Para detectar la latencia prohibida 1 se utiliza el registro de desacoplo DL_ALU. Entonces, mediante una puerta AND se detecta que se producirá un conflicto si en la etapa D/L hay una instrucción load ($Load_{DL}$) y en la etapa ALU hay una instrucción store ($Store_{ALU}$).



Pregunta 3: *Supongamos que en la traza de ejecución de un código hay un 9% de instrucciones store y el 25% de las veces la instrucción que sigue a una instrucción store es una instrucción load. Además, supondremos idealmente, que las instrucciones no utilizan datos calculados por instrucciones previas y que tampoco hay instrucciones de secuenciamiento. Calcule el número medio de instrucciones procesadas por ciclo (IPC).*

Respuesta: La latencia de inicio ideal es 1. Esta latencia se incrementa en 1 ciclo en los casos que una instrucción store está seguida inmediatamente por una instrucción load.

Como hay un 25% de instrucciones store, la latencia media de inicio es

$$1 + 0.09 \times 0.25 = 1.02$$

Entonces, el IPC es 1/1.02 = 0.98

Para mantener la latencia de la instrucción load y eliminar los conflictos una solución es disponer de dos caminos de acceso a los bancos que almacenan el campo datos del contenedor. Un camino se utiliza para efectuar una lectura y el otro camino para efectuar una escritura. Sin embargo, esta alternativa es bastante costosa y puede incrementar el tiempo de ciclo.

Otra alternativa es retardar el acceso a datos en las instrucciones store de forma variable (segmentación flexible). En la siguiente figura se muestra la segmentación del acceso a cache de una instrucción store cuando se puede retardar de forma variable el acceso al campo datos del contenedor. El instante en que una instrucción store actualiza el campo datos depende de si se produce conflicto con otra instrucción load posterior. Si se produce un conflicto se retrasa el acceso. En caso contrario se accede al recurso bD. Por tanto, el número de ciclos de retardo dependerá del número de instrucciones load posteriores que acceden a cache. En la figura el acrónimo R/DAT en el 2º ciclo indica que se intenta actualizar el campo datos del contenedor y en caso de no poderse actualizar, debido a un conflicto, se retarda la actualización mediante un buffer (r).

		ciclos				
		1	2	. . .		
Store	etapa	ET	R/DAT	. . .	R/DAT	DAT
	recursos	bET	r ó bD	. . .	r ó bD	bD

Pregunta 4: *En la siguiente figura se muestra en la columna de la izquierda una secuencia de instrucciones. Mediante un diagrama temporal muestre el uso de la segmentación flexible en las instrucciones store. Suponga que antes de la etapa ET/DAT en la instruc-*

ciones load y antes de la etapa ET en las instrucciones store están las etapas CP, BUS D/L y ALU y recuerde que la instrucción load actualiza el banco de registros (ES).

Respuesta: En los ciclos 7 y 8 se producen conflictos entre la 2ª instrucción (store) y dos instrucciones load más jovenes (3ª y 4ª). En caso de conflicto se da prioridad a una instrucción load que es más joven que la instrucción store. En el ciclo 9 la 3ª instrucción (store) puede acceder al recurso bD. Esta acción se efectúa concurrentemente con el acceso al recurso bET efectuado por la 5ª instrucción (store). En el ciclo 10 entre la 5ª y 6ª instrucción se observa el mismo tipo de comportamiento.

	ciclos											
instrucción	1	2	3	4	5	6	7	8	9	10	11	12
1. load **R1**, X(R2)	CP	BUS	D/L	ALU	ET/DAT	ES						
2. store R9, X(R6)		CP	BUS	D/L	ALU	ET	R	R	DAT			
3. load **R3**, X(R5)			CP	BUS	D/L	ALU	ET/DAT	ES				
4. load **R4**, X(R8)				CP	BUS	D/L	ALU	ET/DAT	ES			
5. store R10, X(R7)					CP	BUS	D/L	ALU	ET	DAT		
6. store R11, X(R12)						CP	BUS	D/L	ALU	ET	DAT	

Pregunta 5: *¿Cuál es el número máximo de instrucciones store pendientes de acceder al campo datos del contenedor?. ¿Cuántas posiciones de almacenamiento son necesarias para implementar la segmentación flexible?.*

El número máximo de instrucciones pendientes de actualizar el campo datos es una, ya que una instrucción store, pendiente de actualizar el campo datos, siempre puede actualizar el campo datos en el 5º ciclo de interpretación de una instrucción store más joven.

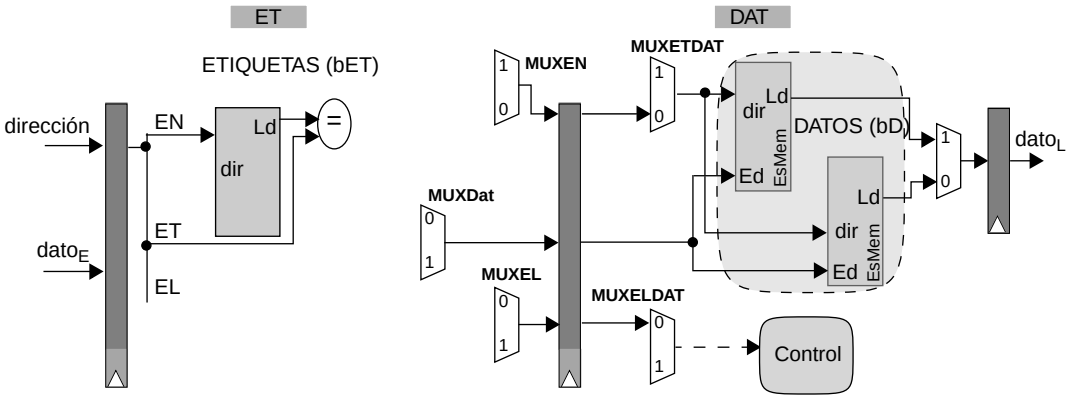
El número de posiciones de almacenamiento necesarias para implementar la segmentación flexible es una, ya que el retardo en actualizar el recurso bD sólo se produce si concurrentemente se interpreta una instrucción load más joven. Cuando se interpreta una instrucción store, si existiera una instrucción store pendiente de actualizar el recurso bD, el store viejo accede al recurso bD mientras el store actual accede al recurso bET. Este comportamiento se observa en los ciclos 9 y 10 del diagrama temporal utilizado para contestar a la 4ª pregunta.

Pregunta 6: *En la siguiente figura se muestra un camino de datos que permite reducir la latencia de la instrucción load y utiliza segmentación flexible en instrucciones store. En el camino de datos, por razones de claridad en el dibujo no se incluyen las señales de control.*

Los multiplexores MUXETDAT y MUXELDAT se utilizan para acceder concurrentemente a los recursos bET y bD en una instrucción load.

Cuando una instrucción store requiere retardos en el acceso al recurso bD se utiliza el registro de desacoplo entre las etapas ET y DAT como elemento de almacenamiento. Para ello se dispone de tres multiplexores (MUXEN, MUXDat, MUXEL) en la etapa ET.

Establezca las conexiones en las entradas de los multiplexores descritos para que el camino de datos funcione utilizando segmentación flexible en las instrucciones store y la latencia de las instrucciones load sea 1 ciclo.

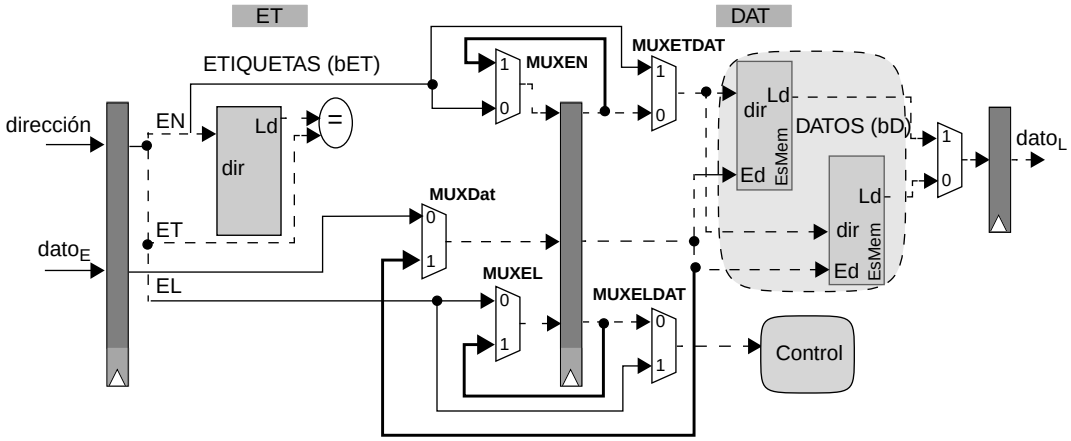


Respuesta: Los multiplexores MUXETDAT y MUXELDAT se utilizan para seleccionar componentes de la dirección. Una de las entradas de cada uno de los multiplexores es la componente de una instrucción load y la otra entrada es la componente de una instrucción store. Notemos que cuando la dirección es de una instrucción load se accede a los recursos bET y bD en el mismo ciclo.

Los multiplexores MUXEN, MUXEL y MUXDat se utilizan en el caso de instrucciones store. Permiten utilizar el registro de desacoplo en la entrada de la etapa DAT como buffer. Una entrada

de los multiplexores proviene del registro de desacoplo de entrada de la etapa ET y la otra entrada proviene del registro de desacoplo de entrada de la etapa DAT.

Para distinguir fácilmente las conexiones que se añaden, las conexiones del dibujo original se dibujan con líneas a trazos.



Controlador de memoria

Cuando se produce un fallo de cache el controlador de memoria se encarga de las transferencias de información necesarias entre memoria y cache o viceversa. Servir un fallo de cache requiere traer un bloque de memoria y en ocasiones actualizar memoria con el bloque expulsado de cache. Este último caso se produce cuando se utiliza escritura retardada para mantener la coherencia en la jerarquía de memoria.

La comunicación entre el controlador de memoria y la memoria se efectúa mediante transferencias que utilizan un bus. En un bus se distinguen tres grupos de señales: a) órdenes, b) dirección y c) datos. Cada grupo de señales representa un recurso (cables) que se gestiona de forma independiente.

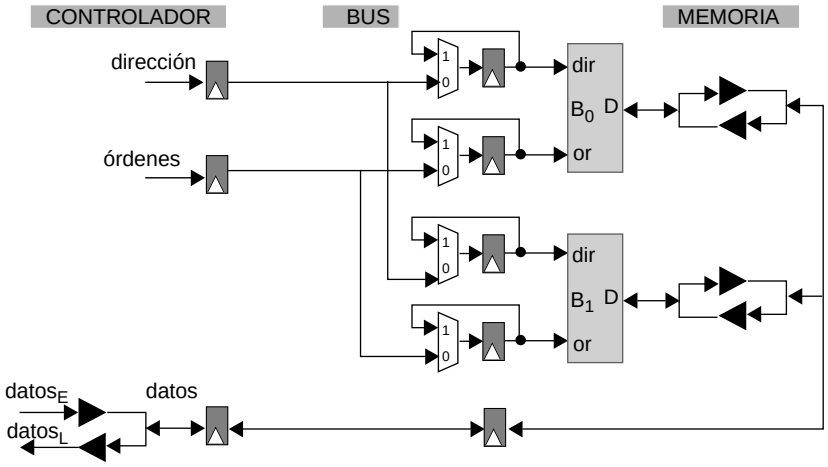
El rendimiento del conjunto bus y memoria se caracteriza por los parámetros: latencia y ancho de banda. En primer lugar nos centraremos en el ancho de banda.

Una forma de incrementar el ancho de banda es que el conjunto bus y memoria pueda estar sirviendo varios accesos de forma concurrente.

Para soportar concurrencia en la memoria se utilizan varios bancos de memoria con un sólo camino de acceso a cada banco y acceso independiente a cada uno de ellos. Mientras los accesos concurrentes sean a bancos distintos se pueden servir concurrentemente. En caso contrario hay que serializarlos y es tarea del controlador de memoria.

La concurrencia en el bus se soporta gestionando de forma segmentada los recursos que constituyen un bus, ya que no es necesario ocupar todos los recursos durante todo el tiempo que tarda una transferencia de información. Al gestionar los recursos del bus de forma segmentada, cada uno de ellos se utiliza exclusivamente durante los ciclos en que se transfiere información. Los ciclos en que una transferencia no utiliza un recurso pueden ser utilizados por otra transferencia.

En la siguiente figura se muestra un esquema simplificado de los elementos bus y memoria. En la memoria se distinguen dos bancos y en el bus se distinguen los tres grupos de señales comentadas previamente. Por el cable etiquetado datos la información puede fluir en los dos sentidos.

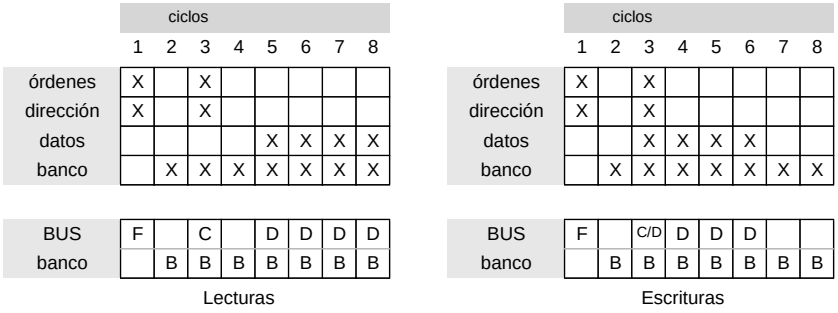


Los multiplexores de la figura se utilizan para retener la información si es necesario y están controlados utilizando la información que indica cual es el banco accedido, la cual se transmite por el grupo de señales denominado órdenes.

La memoria está organizada en bancos y una transferencia de lectura o escritura requiere acceder exclusivamente a un banco. El acceso a cada banco es independiente. Los elementos cuyo símbolo es un triángulo funcionan como conmutadores y permiten el paso de la señal únicamente en un sentido. Se utilizan para desconectarse o conectarse al bus direccional denominado datos. En caso de conexión se tiene en cuenta el sentido de las señales: transmitir o recibir información.

Todos los bancos están conectados al controlador de memoria mediante el bus segmentado descrito previamente. En una operación de escritura de un bloque los datos del bloque fluyen en el sentido de controlador de memoria a memoria. En una operación de lectura los datos del bloque fluyen en sentido contrario.

En la siguiente figura se muestra la utilización del bus segmentado y la ocupación del banco accedido en una operación de lectura y en una operación de escritura de un bloque.



En la parte superior de la figura previa se muestra una tabla de reserva detallada por recurso y en la parte inferior una tabla de reserva resumida. En esta última tabla los acrónimos F y C representan que se utilizan los recursos órdenes y dirección conjuntamente en dos ciclos distintos. La utilización de los recursos datos y banco se especifica mediante los acrónimos D y B respectivamente.

Antes de efectuar el acceso a los bancos el controlador de memoria determina el banco a partir de la dirección recibida. La información del banco al que se accede se transmite por el grupo de señales denominado órdenes. A partir de ahora cuando se utilice la palabra dirección se entenderá la dirección que se utiliza para acceder a un banco concreto.

La transmisión de una dirección se efectúa en dos partes: a) fila (F) y b) columna (C). En ambos casos, junto con cada componente de la dirección, en el mismo ciclo, se utiliza el recurso órdenes para transmitir, entre otras informaciones, el tipo de operación (lectura, escritura) que se quiere efectuar y el identificador de banco.

Los bits correspondientes a la fila son los más significativos de la dirección y los bits correspondientes a la columna son los menos significativos. La componente fila de la dirección se transmite en primer lugar y se inicia el acceso al banco de memoria referenciado. Al cabo de dos ciclos se transmite la componente columna de la dirección.

El recurso dato se ocupa durante 4 ciclos para transmitir el bloque de memoria. En cada ciclo se transmite un número fijo de bytes. Nosotros supondremos que en cada ciclo que se ocupa el recurso dato se transmiten 16 bytes y el tiempo de ciclo es 2ns.

El recurso banco está ocupado durante 7 ciclos de los 8 que dura la transferencia. Accesos que se sirven utilizando bancos distintos sólo pueden tener conflictos debido a los recursos del bus.

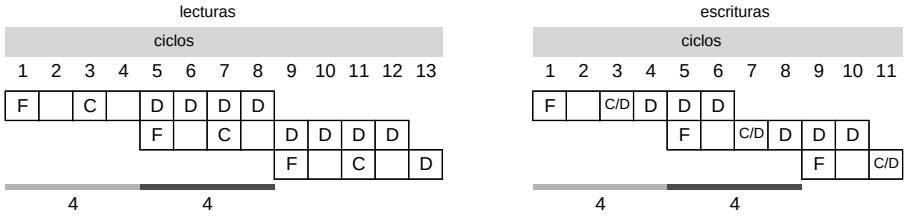
Pregunta 1: *¿Cuál es el ancho de banda (número medio de datos transmitidos por unidad de tiempo) máximo del recurso datos?.*

Respuesta: Como máximo se pueden transmitir 16 bytes cada 2ns. Entonces, 16bytes/2ns = 8 GBytes/s.

Pregunta 2: *Suponga que cada acceso es a un banco de memoria distinto en los siguientes casos de ráfagas de operaciones: a) lecturas, b) escrituras y c) entrelazado una a una de lecturas y escrituras. Calcule: 1) una secuencia de latencias de inicio, 2) la latencia media de la secuencia de latencias de inicio, 3) la latencia de n transferencias, 4) el número de transferencias por ciclo y 5) el ancho de banda en el recurso datos del bus. Finalmente calcule la*

ganancia respecto de un funcionamiento serie. Esto es, desde el controlador de memoria no se inicia un nuevo acceso hasta que ha finalizado el anterior.

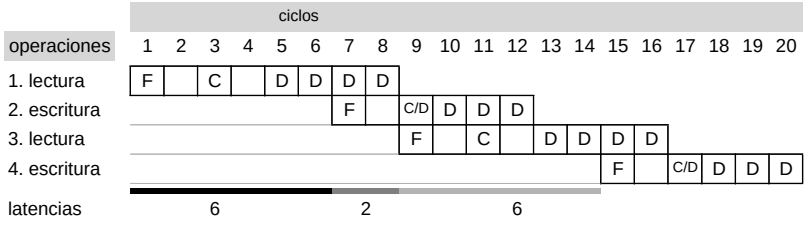
Respuesta: En la figura se muestran diagramas para determinar las latencias permitidas en ráfagas de lecturas y escrituras respectivamente. Como se accede a bancos distintos no se especifica el recurso banco.



En la siguiente tabla se recopilan las contestaciones a las preguntas.

lecturas / escrituras	Banco distinto
Secuencia de latencias de inicio	(4)
Latencia media de inicio	4 ciclos
Latencia	n x 4 + 4 ciclos
Transferencias por ciclo	n / (nx4+4) ~ 1/4 = 0.25
Ancho de banda	16 Bytes/2ns= 8 GBytes/s
Ganancia	lecturas: n x 8 / (n x 4 + 4) ~ 2 escrituras: n x 6/ (n x 4 + 4) ~ 1.5

En la figura se muestra un diagrama temporal con una secuencia entrelazada de operaciones de lectura y escritura a bancos distintos.

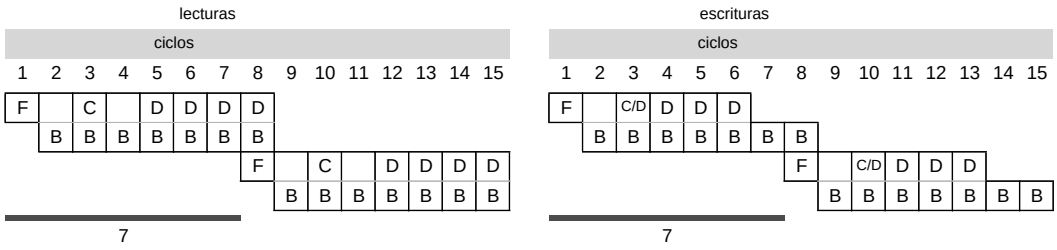


En la siguiente tabla se recopilan las contestaciones a las preguntas.

secuencia entrelazada de lecturas/ escrituras	Banco distinto
Secuencia de latencias de inicio	(6, 2)
Latencia media de inicio	(6+2) / 2 = 4 ciclos
Latencia	n x 4 + 4 ciclos
Transferencias por ciclo	n / (nx4+4) ~ 0.25
Ancho de banda	16 Bytes/2ns= 8 GBytes/s
Ganancia	[(n/2) x (8 + 6)] / (n x 4 + 4) = 1.75

Pregunta 3: *Para las siguientes ráfagas de operaciones que acceden al mismo banco: a) lecturas, b) escrituras y c) entrelazado una a una de lecturas y escrituras, calcule: 1) una secuencia de latencias de inicio, 2) la latencia media de la secuencia de latencias de inicio, 3) la latencia de n transferencias, 4) el número de transferencias por ciclo y 5) el ancho de banda en el recurso datos del bus.*

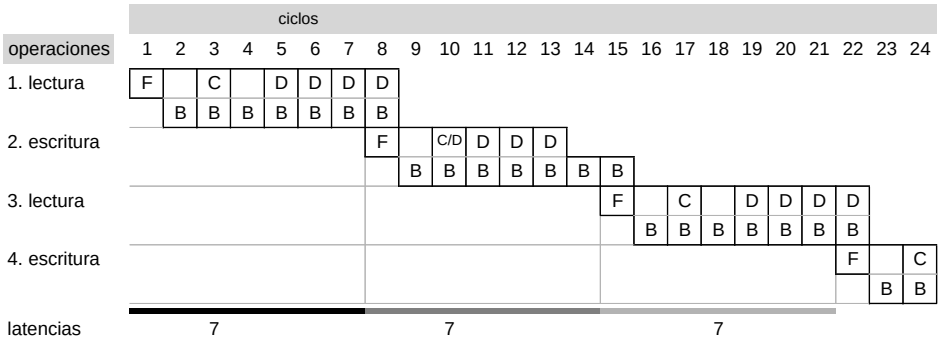
Respuesta: En la figura se muestran diagramas para determinar las latencias permitidas en ráfagas de operaciones de lectura y escritura cuando se accede al mismo banco.



En la siguiente tabla se recopilan las contestaciones a las preguntas. Note que el recurso datos está ocupado 4 veces cada 7 ciclos.

lecturas / escrituras	Mismo banco
Secuencia de latencias de inicio	(7)
Latencia media de inicio	7 ciclos
Latencia	n x 7 + 1 ciclos
Transferencias por ciclo	n / (nx7+1) ~ 0.14
Ancho de banda	(16 Bytes x 4)/ (7 x 2ns) = 4.57 GBytes/s

En la figura se muestra un diagrama temporal con una secuencia entrelazada una a una de operaciones de lectura y escritura al mismo banco.

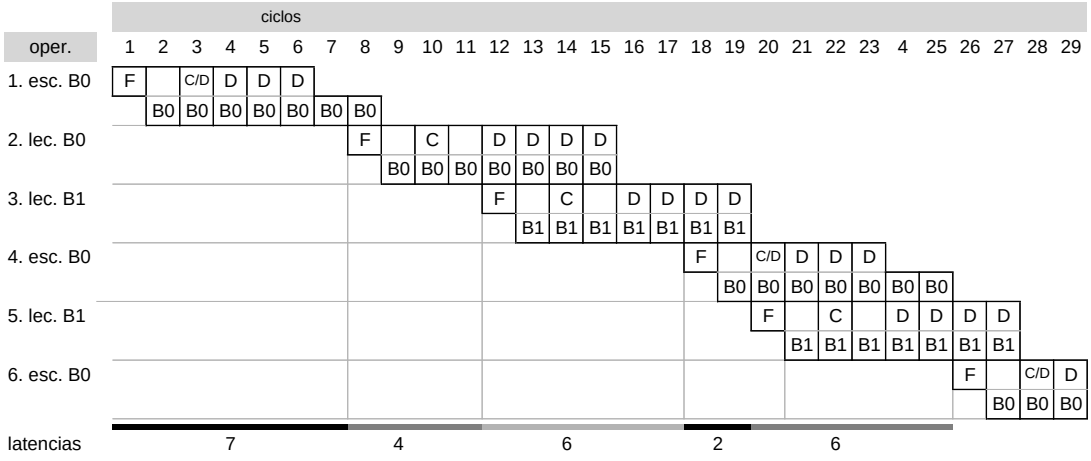


En la siguiente tabla se recopilan las contestaciones a las preguntas. Note que el recurso datos está ocupado 4 veces cada 7 ciclos.

secuencia entrelazada de lecturas/ escrituras	Mismo banco
Secuencia de latencias de inicio	(7)
Latencia media de inicio	7 ciclos
Latencia	n x 7 + 1 ciclos
Transferencias por ciclo	n / (nx7+1) ~ 0.14
Ancho de banda	(16 Bytes x 4)/ (7 x 2ns) = 4.57 GBytes/s

Pregunta 4: *Utilizando un diagrama temporal, muestre la evolución temporal de la secuencia de accesos que se muestra en la columna izquierda de la figura. Junto con el tipo de transferencia (lec. esc.) se indica el banco al que se accede.*

Respuesta: La 2ª transferencia debe esperar a que se desocupe el banco B0. La 3ª, 4ª, 5ª y 6ª transferencia deben esperar a que el recurso datos quede libre.



Pregunta 5: *Para contestar a la siguiente pregunta analice las respuestas a las preguntas 2ª, 3ª y 4ª . ¿Qué característica debe cumplir una secuencia de accesos a bancos de memoria para extraer el máximo rendimiento del sistema de memoria?.*

Respuesta: Una secuencia extrae el máximo rendimiento cuando 2 accesos contiguos son a bancos distintos.

Pregunta 6: *Llamaremos secuencia a una secuencia de 4 accesos consecutivos a bancos de memoria. Además, supondremos que la solicitud de acceso a cualquier banco es equiprobable.*

Cuando se dispone de sólo 2 bancos de memoria, calcule el número de secuencias que extraen el máximo rendimiento del sistema de memoria y el número total de secuencias distintas.

Supongamos que se utilizan 4 bancos de memoria. El número de secuencias que extraen el máximo rendimiento del sistema de memoria es 108 y el número total de secuencias distintas es 256.

¿Es de utilidad disponer de más de 2 bancos de memoria en el conjunto bus-memoria descrito?.

Respuesta: Las secuencias que extraen el máximo rendimiento con 2 bancos son las siguientes dos:

0, 1, 0, 1 y 1, 0, 1, 0

El número total de secuencias distintas es $2^4 = 16$. Entonces, la proporción de secuencias que extraen el máximo rendimiento es 2/16.

Para contestar a la segunda pregunta hay que comparar las proporciones de secuencias que extraen el máximo rendimiento en los dos casos:

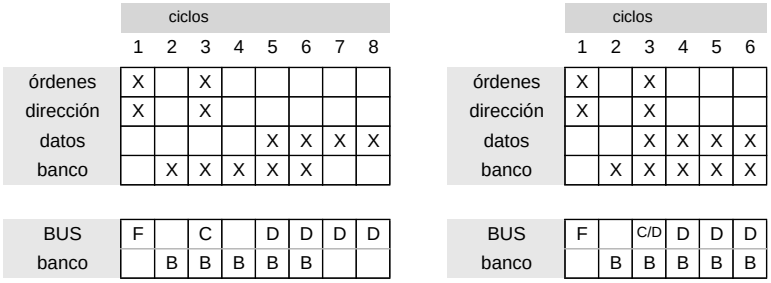
2 bancos 2/16 = 32/256 = 0.12

4 bancos 108/256 = 0.42

Utilizando 4 bancos de memoria la proporción de secuencias que extraen el máximo ancho de banda es mayor; más del triple. Por tanto, es de utilidad disponer de más de 2 bancos de memoria para secuencias de 4 accesos consecutivos.

En un banco de memoria se distinguen agrupaciones de posiciones contiguas de almacenamiento denominadas páginas. Una página se identifica mediante la componente fila de una dirección y las distintas posiciones de almacenamiento en una página son accesibles mediante la componente columna de la dirección.

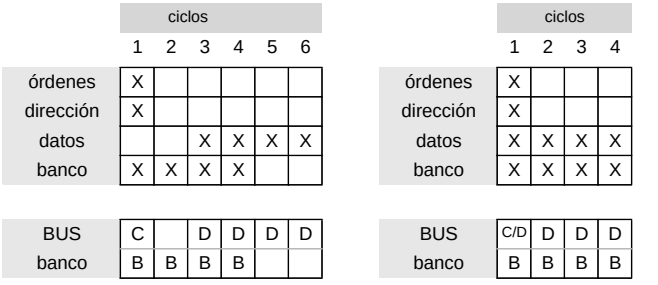
Cuando se accede a una página de un banco se puede dejar abierta. Esto significa que un acceso posterior a la misma página del banco no requiere transmitir la componente fila de la dirección. También, un acceso que deja abierta la página no requiere ocupar el banco los últimos 2 ciclos. En la siguiente figura se muestra la ocupación de recursos al acceder a un banco cuando se deja la página abierta.



Lectura: abre página cerrada y la deja abierta

Escritura: abre página cerrada y la deja abierta

En la siguiente figura se muestra la ocupación de recursos cuando se accede a una página abierta y se deja abierta. No es necesario dedicar ciclos para transmitir la componente fila de la dirección ni para abrir la página.

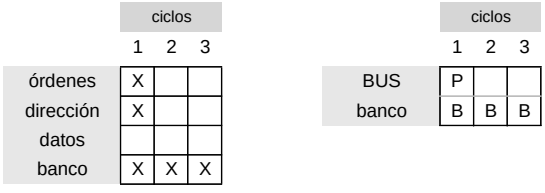


Lectura con página abierta y la deja abierta

Escritura con página abierta y la deja abierta

El mecanismo descrito reduce la latencia de acceso a memoria cuando una secuencia de accesos accede a la misma página de un banco. Se pasa de una latencia de acceso de 8 ciclos a una latencia de acceso de 6 ciclos en operaciones de lectura. En operaciones de escritura se pasa de 8 ciclos a la mitad.

Para poder acceder en un banco a una página distinta de la página que está abierta hay que cerrar previamente la página abierta. En la siguiente figura se muestra la ocupación de los recursos cuando se quiere cerrar una página en un banco.



Cerrar página

En la figura el acrónimo P indica precarga y el primer ciclo de la transferencia de cerrar página se puede solapar con el antepenúltimo ciclo de una transferencia de lectura previa al mismo banco, que dejaba la página abierta, sin que exista conflicto en el recurso banco. En el caso de una operación de escritura se puede solapar con el último ciclo. En la siguiente figura se muestran las acciones cuando se cierra una pagina inmediatamente después de un acceso previo.

	ciclos					
	1	2	3	4	5	6
órdenes	X			X		
dirección	X			X		
datos			X	X	X	X
banco	X	X	X	X	X	X
BUS	C		D	D/P	D	D
banco	B	B	B	B	B	B

Lectura con página abierta y la deja cerrada

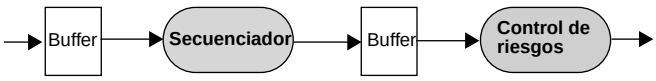
	ciclos					
	1	2	3	4	5	6
órdenes	X					
dirección	X					
datos	X	X	X	X		
banco	X	X	X	X	X	X
BUS	C/D	D	D	D/P		
banco	B	B	B	B	B	B

Escritura con página abierta y la deja cerrada

La ocupación de recursos descrita antes de la 1ª pregunta abre una página y cierra la página después de servirse el acceso.

El controlador de memoria dispone de un módulo denominado secuenciador. Este módulo, partiendo de las peticiones de acceso que recibe, genera las secuencias de acceso a cada banco que considera mejor con el objetivo de reducir el tiempo de servicio. Esto es, si una página está abierta y hay que acceder a otra página del banco, el módulo secuenciador genera una transferencia para cerrar la página. También, si el módulo secuenciador observa que tiene que procesar varios accesos a la misma página no cierra la página, ya que se reduce el tiempo de servicio de la secuencia de accesos.

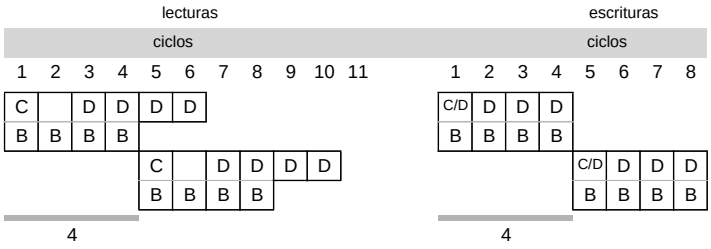
Las secuencias de referencias generadas por el secuenciador son las que procesa el módulo de control de riesgos estructurales del controlador.



Pregunta 7: *Para las siguientes ráfagas de operaciones que acceden a la misma página: a) lecturas, b) escrituras y c) entrelazado una a una de lecturas y escrituras, calcule: 1) una secuencia de latencias de inicio, 2) la latencia media de la*

secuencia de latencias de inicio, 3) la latencia de servicio de n transferencias, 4) el número de transferencias por ciclo y 5) el ancho de banda en el recurso datos del bus. Suponga en todos los casos que antes de la ráfaga y al finalizar la página está abierta.

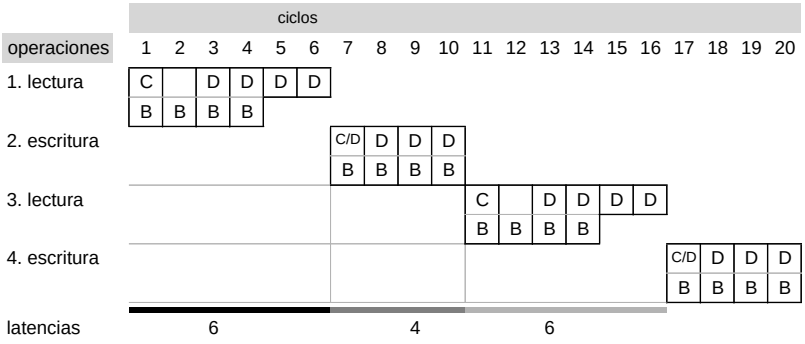
Respuesta: En la figura se muestran diagramas para determinar las latencias permitidas cuando las secuencias de transferencias de lectura y escritura acceden a la misma página.



En la siguiente tabla se recopilan las contestaciones a las preguntas.

lecturas / escrituras	Misma página
Secuencia de latencias de inicio	(4)
Latencia media de inicio	4 ciclos
Latencia	lecturas: n x 4 + 2 ciclos escrituras: n x 4 ciclos
Transferencias por ciclo	n / (nx4+2) ~ 0.25
Ancho de banda	16 Bytes/2ns= 8 GBytes/s

En la siguiente figura se muestra un diagrama temporal con una secuencia entrelazada una a una de operaciones de lectura y escritura a la misma página.



En la siguiente tabla se recopilan las contestaciones a las preguntas. Note que cada 10 ciclos se ocupa el recurso dato 8 veces.

secuencia entrelazada de lecturas/ escrituras	Misma página
Secuencia de latencias de inicio	(6,4)
Latencia media de inicio	(6 + 4) / 2 = 5 ciclos
Latencia	n par: n x 5 ciclos n impar: n x 5 + 1 ciclos
Transferencias por ciclo	n / (n x 5) ~ 0.20
Ancho de banda	(16 Bytes x 8)/ (10 x 2ns) = 6.4 GBytes/s

Pregunta 8: *Compare la latencia de una secuencia de operaciones de lectura cuando: a) se accede a bancos distintos, b) se accede al mismo banco pero a distinta página y c) se accede al mismo banco y a la misma página.*

Suponga que la página (o páginas) accedida está inicialmente cerrada y después de la secuencia de transferencias queda cerrada.

Respuesta: En la siguiente tabla se recopilan las contestaciones de las preguntas anteriores: 2ª, 3ª y 7ª.

secuencia de lecturas	Latencia
Distinto banco	n x 4 + 4 ~ n x 4 ciclos
Mismo banco y distinta página	n x 7 + 1 ~ n x 7ciclos
Mismo banco y página	2 + (n x 4 + 2) + 3 ~ n x 4 ciclos

En el último caso se ha sumado a la contestación de la 7ª pregunta el tiempo de abrir y cerrar la página.

Las latencias en el primer y último caso son aproximadamente iguales. La latencia del segundo caso es

(n x 7) / (n x 4) = 1.75 veces mayor que la de los otros dos casos.

Pregunta 9: *Calcule, en los siguientes casos, el tiempo de las dos transferencias requeridas en un reemplazo de cache cuando se expulsa un bloque actualizado:*

a) las dos referencias acceden a bancos distintos.

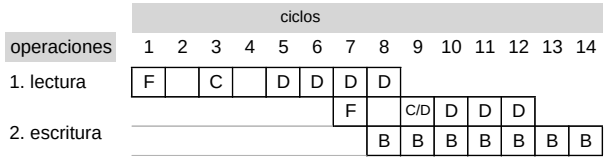
b) las dos referencias acceden al mismo banco pero la página es distinta.

c) las dos referencias acceden al mismo banco y a la misma página.

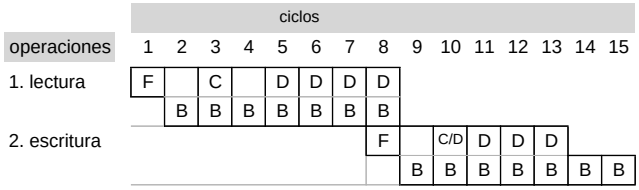
Suponga que la página (o páginas) accedida está inicialmente cerrada, después de las dos transferencias debe quedar cerrada y entre las dos transferencias no se efectúan otras transferencias. Para reducir la latencia de lectura observada por el procesador, el orden de las transferencias solicitadas desde cache es en primer lugar lectura y después escritura.

Respuesta: En los siguientes diagramas temporales se muestra la ocupación de los recursos y la duración de las transferencias.

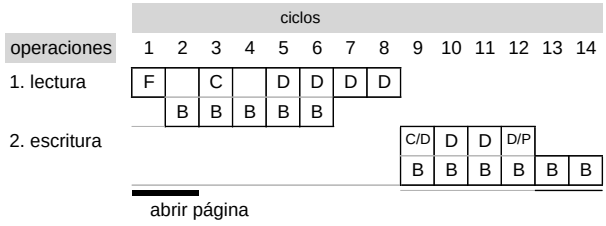
Distinto banco:



Mismo banco y distinta página:



Mismo banco y página:



El tiempo de las dos transferencias para cada caso es:

secuencia lectura/escritura	Latencia de servicio
Distinto banco	14 ciclos
Mismo banco y distinta página	15 ciclos
Mismo banco y página	2 + 10 + 2 = 14 ciclos

En el último caso se distingue explícitamente en la suma los ciclos utilizados para abrir y cerrar la página.

EJERCICIOS

Ejercicio 2.1

Un procesador segmentado tiene tres etapas (I , M y A). Existen, únicamente dos tipos de instrucciones X e Y y su interpretación utiliza las etapas en los ciclos especificados a continuación.

	ciclos		
instrucción	1	2	3
X	I	A	
Y	I	M	A

Dos instrucciones colisionan, o se produce un riesgo estructural, si durante su interpretación intentan utilizar la misma etapa en el mismo ciclo.

Pregunta 1: *Compruebe que es posible que exista colisión al ejecutar un programa si la latencia de iniciación de instrucciones es de un ciclo.*

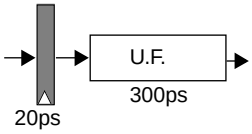
Pregunta 2: *¿Cuál es la latencia de iniciación mínima constante que garantiza que no existe colisión en la ejecución de un programa?.*

Pregunta 3: *Rediseñe los dos tipos de instrucciones para que la latencia de iniciación constante sea la menor posible. (NOTA: analice el número de ciclos y utilización de las etapas).*

Pregunta 4: *Razone si el nuevo diseño aumenta o disminuye el tiempo de ejecución de los programas al compararlo con los dos casos anteriores. Para ello, calcule la productividad en cada caso.*

Ejercicio 2.2

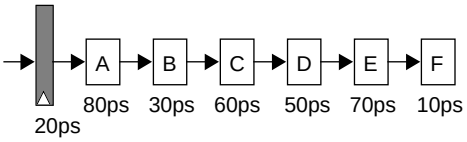
Supongamos que se dispone de una unidad funcional (UF) no segmentada con un retardo de 300ps. El módulo secuencial utilizado para procesar secuencias de operaciones es el siguiente.



Pregunta 1: *¿Cuál es el periodo mínimo de la señal de reloj (tiempo de ciclo)?.*

Pregunta 2: *Calcule la productividad del módulo secuencial.*

Supongamos que la UF anterior se ha dividido en 6 bloques (A, B, C, D, E y F) que se conectan como se muestra en la figura y con los tiempos de retardo que se indican.



Pregunta 3: *Utilizando 1 registro de desacoplo segmente la UF para maximizar la productividad (2 etapas). Indique la productividad.*

Pregunta 4: *¿Cuál es la latencia de la operación en el diseño de la 3ª pregunta?.*

Pregunta 5: *Utilizando 2 registros de desacoplo segmente la UF para maximizar la productividad (3 etapas). Indique la productividad.*

Pregunta 6: *¿Cuál es la latencia de la operación en el diseño de la 5ª pregunta?.*

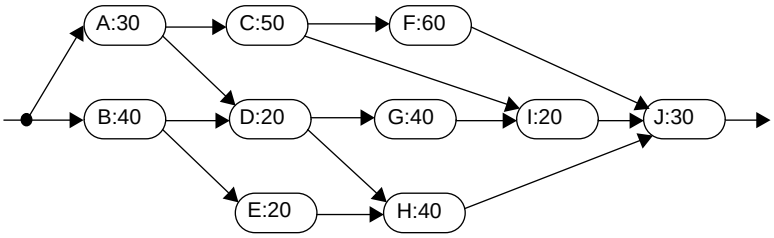
Pregunta 7: *¿Cuál es el número de etapas mínimo para maximizar la productividad?.*

Supongamos que un mejor particionado de la UF consigue que todos los bloques tengan el mismo retardo y que el número de bloques puede ser el que queramos.

Pregunta 8: *Calcule la productividad en el caso de que el retardo de cada uno de los bloques (t_B), en que se divide la UF, es despreciable frente al retardo que representa el registro de desacoplo (t_{RD}) entre etapas ($t_B << t_{RD}$).*

Ejercicio 2.3

Considere el siguiente circuito donde se muestran bloques combinacionales y el flujo de información entre bloques. Así mismo, dentro de cada bloque, se ha anotado la latencia en picosegundos.



Pregunta 1: *¿Cuál es la latencia del circuito?. Marque con trazo grueso las conexiones del camino crítico.*

Suponga que el retardo de almacenamiento en un registro de desacoplo es cero.

Pregunta 2: *¿Cuál es la máxima productividad, en operaciones por segundo, que se puede obtener al segmentar el circuito?.*

Pregunta 3: *Muestre una organización segmentada que maximice la productividad.*

Pregunta 4: *Muestre un organización segmentada que minimice el número de etapas y mantenga una productividad de al menos una operación cada 85 ps.*

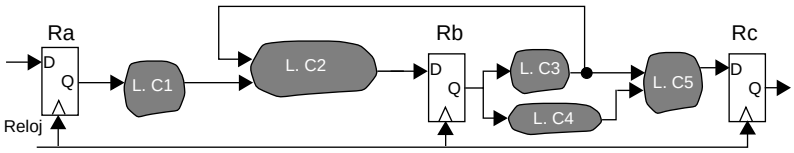
Entre dos etapas puede haber varios flujos de información. Cada uno de ellos requiere de un registro de desacoplo. La información que hay en la entrada de un registro de desacoplo se transfiere a la salida en el flanco ascendente de la señal de reloj. Suponga la siguiente segmentación.

	bloques
1ª etapa	A, B, D, G
2ª etapa	C, F, E
3ª etapa	H, I, J

Pregunta 5: *Muestre en un esquema los registros de desacoplo numerándolos como 1 o 2 para indicar respectivamente si son registro de salida de la 1ª etapa o de la 2ª etapa.*

Ejercicio 2.4

En la figura se muestra un dispositivo segmentado en dos etapas. Los elementos L.C1, L.C2, L.C3, L.C4 y L.C5 representan lógica combinacional. El retardo de propagación de estos elementos es respectivamente t_{c1}, t_{c2}, t_{c3}, t_{c4} y t_{c5} y el retardo de los registros de desacoplo es t_p.



Pregunta 1: *Identifique los elementos combinacionales de cada etapa. Tenga en cuenta que la disposición espacial en el dibujo de un elemento no implica pertenencia sólo a una etapa concreta.*

Pregunta 2: *Calcule el tiempo de ciclo del diseño segmentado.*

Ejercicio 2.5

Un procesador segmentado no lineal tiene 5 etapas: CP, M, DL, A y E, tiene una memoria (M) única para instrucciones y datos, tiene un solo camino (E) de escritura en el banco de registros y no se producen riesgos estructurales debido a los caminos de lectura al banco de registros. Interpreta instrucciones de esta manera:

	ciclos					
instrucciones	1	2	3	4	5	6
load	CP	M	DL	A	M	E
store	CP	M	DL	A	M	
ent	CP	M	DL	A	E	

Para medir sus prestaciones este procesador ejecuta repetida e indefinidamente un código con 4 instrucciones que son:

```
store
load
load
ent
```

Sabemos que nunca hay riesgo de datos entre ellas.

Este procesador, por ser no lineal, puede tener riesgos estructurales. Este procesador puede resolverlos, a base de controlar el inicio de la interpretación de instrucciones.

Pregunta 1: *Calcule una secuencia óptima de latencias de inicio.*

Pregunta 2: *Calcule el valor en ciclos de la latencia media mínima de inicio.*

Pregunta 3: *Calcule el número de ciclos que tarda en interpretar 40.000 instrucciones.*

Pregunta 4: *Calcule el valor medio del IPC.*

Pregunta 5: *Calcule la ganancia sobre un procesador serie que tenga el mismo tiempo de ciclo.*

Pregunta 6: *Calcule la cantidad de ciclos perdidos por tener un solo camino de escritura.*

Ejercicio 2.6

Una Unidad Funcional multifunción segmentada en 5 etapas (A, B, C, D, E) procesa de forma solapada 4 tipos de operaciones según las tablas de reserva:

	ciclos					
	1	2	3	4	5	6
A	X					
B		X				
C			X			
D				X	X	
E						X

op1

	ciclos				
	1	2	3	4	5
A	X				
B		X			
C			X		
D				X	X
E					

op2

	ciclos			
	1	2	3	4
A	X			
B		X		
C			X	
D				
E				X

op3

	ciclos		
	1	2	3
A	X		
B		X	
C			X
D			
E			

op4

Se quiere analizar el rendimiento al procesar dos secuencias de operaciones ilimitadas y periódicas:
S1 = (op2, op1, op4, op3) ; es decir: op2, op1, op4, op3, op2, op1, op4, op3, op2, ...
S2 = (op3, op1, op4, op2)

Pregunta 1: *Calcule para S1 y S2 una secuencia periódica de latencias de inicio. Justifique la respuesta mostrando en un cronograma el procesado de las cinco primeras operaciones de la secuencia.*
Calcule la latencia media de inicio.
Calcule el número de operaciones procesadas por ciclo.

Pregunta 2: *Calcule para S1 y S2 la ganancia que se obtiene respecto del caso en que la unidad funcional procese las operaciones sin solapamiento.*

Pregunta 3: *Calcule para S1 y S2 el tiempo en finalizar las 100 primeras operaciones de la secuencia.*

Ejercicio 2.7

Tenemos una Unidad Funcional Segmentada (UFS) que no es lineal, tiene 3 etapas (A, B y C) y hace una operación que tarda 5 ciclos, de manera repetitiva y sin dependencias.

En cada uno de los 5 ciclos se usa una y sólo una de las etapas.

Pregunta 1: *Presente una Tabla de Reservas (TdR) que muestre la actividad o no de cada etapa en cada ciclo de la operación. Debe tener en cuenta que sabemos que la secuencia periódica de latencias de inicio <2 , 3> no causa ningún conflicto, aunque no sabemos si es la secuencia óptima (máxima productividad) o no lo es.*

Pregunta 2: *Calcule la latencia media mínima (LMM) con la TdR que ha presentado.*
Justifique ahora si <2, 3> es óptima o no lo es.

Pregunta 3: *Calcule el tiempo T, en segundos, que tarda esta UFS en completar 10000 operaciones, sabiendo que cada ciclo dura 20 ns.*

Pregunta 4: *Calcule la ganancia G que obtiene respecto a usar la UF sin segmentar y con idéntico tiempo de ciclo.*

Pregunta 5: *Muestre con un cronograma si la secuencia periódica < 2 > permite usar la UFS , con la TdR que ha presentado, sin causar conflictos.*
Idem para la secuencia periódica <3>.

Ejercicio 2.8

El lenguaje máquina de un procesador tiene instrucciones para leer o escribir datos en memoria (MEM), instrucciones para efectuar cálculos aritmético-lógicos (ENT) entre otros tipos de instrucciones que no consideraremos. En la figura se muestra el formato de las 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															
	RI	CoOp			ra			literal					1	func			rc															Cálculo
MEM	Lo | St	CoOp			ra			rb			literal																					Acceso a memoria

En la siguiente figura se muestra la especificación semántica de los tipos de instrucciones RR, RI, Load y Store. Las instrucciones Load y Store utilizan el contenido de un registro y el campo literal de la instrucción para calcular la dirección efectiva. Las instrucciones de tipo RR y RI efectuan cálculos lógicos o aritméticos con enteros y sus datos fuente están en registros o en un campo literal y almacenan el resultado en un registro. El superíndice v indica contenido del registro. Los campos CoOp y func determinan la operación que debe efectuarse.

Tipo	Descripción	Operandos			Especificación semántica
RR	operación aritmético-lógica	ra	rb	rc	$rc^v = ra^v$ (CoOp, func) rb^v
RI		ra	literal	rc	$rc^v = ra^v$ (CoOp, func) #literal
Load	Lectura de memoria	ra	rb	literal	ra^v = Mem [rb^v + ExtSig (literal)]
Store	Escritura en memoria	ra	rb	literal	Mem [rb^v + ExtSig (literal)] = ra^v

Pregunta 1: *Las fases de una instrucción son: determinar la dirección, búsqueda de la instrucción, decodificación, lectura de los datos fuente, ejecución y escritura del resultado.*
Para las instrucciones especificadas describa mediante una tabla cuales son las acciones que deben efectuarse en cada fase.

Para construir un camino de datos segmentado disponemos de los recursos mostrados en la siguiente tabla. En el banco de registros se dispone de un camino de lectura y de otro camino de lectura o escritura.

Recursos	Funcionalidad	retardo
+	sumador que se utiliza para el secuenciamiento implícito	10 ns
MID	memoria que almacena instrucciones y datos	50 ns
BusMID	bus desde la ALU al recurso MID para transmitir la dirección. Se utiliza en las instrucciones load y store	15 ns
BR	acceso al banco de registros. Lectura o escritura	25 ns
ALU	unidad aritmético-lógica	35 ns
D	decodificador	0 ns

Un recurso puede utilizarse varias veces en un ciclo si su retardo o tiempo de acceso lo permite.

Pregunta 2: *Utilizando la contestación de la pregunta anterior, los recursos disponibles y su retardo, proponga una segmentación en etapas del proceso de interpretación de las instrucciones descritas e indique el tiempo de ciclo. En la contestación minimice la latencia de interpretación, el tiempo de ciclo y los posibles riesgos estructurales.*

Pregunta 3: *¿Cuántos caminos de acceso son necesarios a la memoria MID para que no se produzcan riesgos estructurales?.*

Pregunta 4: *Justifique si son suficientes los caminos de accesos descritos al banco de registros para que no se produzcan conflictos.*

Ejercicio 2.9

En la figura se muestra una segmentación del proceso de interpretación de instrucciones.

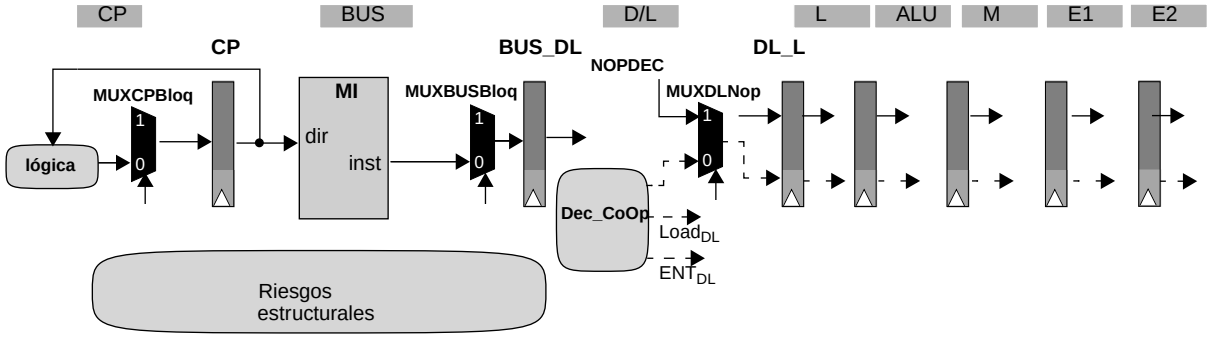
		ciclos							
Tipo	instrucción	1	2	3	4	5	6	7	8
MEM	load	CP	BUS	D/L	L	ALU	M	E1	E2
	store	CP	BUS	D/L	L	ALU	M		
ENT		CP	BUS	D/L	L	ALU	E1	E2	

Las fases de lectura y escritura al banco de registros requieren 2 ciclos y están segmentadas. El número de caminos de lectura y escritura al banco de registros son 2 y 1 respectivamente. En las etapas BUS y M se accede a memorias distintas (MI y MD). Las situaciones de riesgo se detectan en la etapa D/L y la instrucción no prosigue su interpretación hasta que desaparece el riesgo.

Pregunta 1: *Dada la siguiente secuencia de instrucciones, muestre en un diagrama temporal las acciones que se toman en el camino de datos para resolver riesgos estructurales. En el diagrama temporal utilice una nueva línea cuando una instrucción se bloquea en una etapa. Así mismo, indique la inserción de instrucciones nop y su avance en las etapas del procesador.*

instrucción
1. load **R1**, X(R2)
2. add **R4**, R9, R10
3. add **R3**, R7, R8
4. store R9, X(R6)

En la figura se muestran, de forma esquemática, algunas etapas de un camino de datos segmentado que interpreta las instrucciones descritas previamente y la lógica utilizada para retener la información en las etapas CP, BUS y D/L e inyectar una instrucción nop desde la etapa D/L. En la etapa D/L se generan señales que indican el tipo de instrucción que está ocupando la etapa (ENT_{DL}, $Load_{DL}$).



Pregunta 2: *Añada las conexiones necesarias en el camino de datos, para gestionar riesgos estructurales. Diseñe el circuito de control de riesgos estructurales, especificando las entradas y salidas.*

Ejercicio 2.10

Un procesador segmentado en 7 etapas interpreta instrucciones de acceso a memoria (Load, Store):

ciclos	1	2	3	4	5	6	7
etapas	CP	B	DL	ALU	T/D	C/F	E/D

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

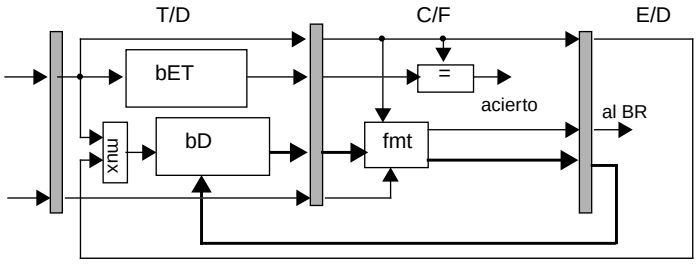
ETAPA	FUNCIONALIDAD
CP	determinar el CP
B	búsqueda de la instrucción
DL	decodificación, detección de riesgos, lectura de operandos en registros
ALU	cálculo de la dirección efectiva
T/D	lectura de la etiqueta y el bloque de datos de la cache
C/F	comparación para determinar acierto/fallo y formateo de los datos
E/D	Escritura en un registro (Load) o escritura del bloque de datos en cache (Store)

El banco de registros tiene 2 caminos de lectura y 1 camino de escritura.

La cache de datos es de mapeo directo y dispone de los siguientes recursos:

RECURSOS	FUNCIONALIDAD
bET	banco de etiquetas
bD	banco de datos con 1 camino de acceso para leer o escribir un vector de B bits (longitud de un bloque de memoria)
=	comparador
fmt	lógica que selecciona un campo de E bits de un vector de B bits o que modifica un campo de E bits (tamaño de un elemento) de un vector de B bits

La siguiente figura muestra el camino de datos correspondiente a las 3 últimas etapas:



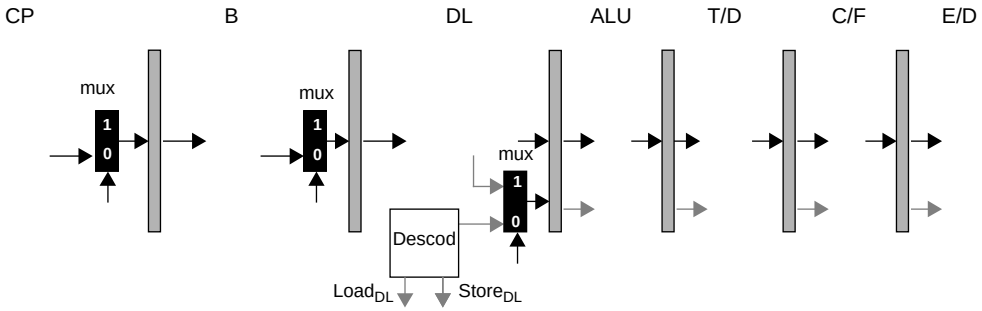
El acceso a la cache de datos está segmentado. Seguidamente se detallan las acciones que se efectúan en las 3 últimas etapas del proceso de interpretación:

- en las instrucciones Load, se lee concurrentemente el banco de etiquetas y el banco de datos (ciclo 5), se determina acierto/fallo y concurrentemente se selecciona el elemento del bloque solicitado mediante la lógica fmt (ciclo 6). Finalmente, se actualiza el banco de registros (ciclo 7).
- en las instrucciones Store, se lee concurentemente el banco de etiquetas y el banco de datos (ciclo 5), se determina acierto/fallo y concurrentemente se modica el elemento del bloque solicitado utilizando la lógica fmt (ciclo 6). Finalmente, se actualiza el banco de datos de la cache (ciclo 7).

Supondremos que todos los accesos a memoria son aciertos en la cache de datos. También supondremos que no existen dependencias de datos entre las instrucciones que se interpretan.

Pregunta 1: *Analice los posibles riesgos estructurales durante la interpretación de una secuencia de instrucciones de acceso a memoria. Indique las latencias de inicio prohibidas, los tipos de instrucciones y los recursos involucrados.*

Pregunta 2: *Diseñe la lógica para detectar los riesgos estructurales en la etapa DL. Añada en el camino de datos las conexiones necesarias para gestionar los riesgos estructurales.*



Pregunta 3: *Calcule el IPC medio cuando el procesador interpreta una secuencia ilimitada de instrucciones Load, una secuencia ilimitada de instrucciones Store, una secuencia ilimitada de instrucciones Store, Load.*

Ejercicio 2.11

Una unidad funcional segmentada multifunción (UFS) tiene 4 etapas y puede realizar tres tipos diferentes de operaciones, siguiendo las Tablas de Reserva (TdR) que se muestran:

	ciclos			
etapas	1	2	3	4
A	X			
B		X		
C			X	
D				X

OPa

	ciclos						
etapas	1	2	3	4	5	6	7
A	X			X			
B		X					X
C			X		X		
D						X	

OPb

	ciclos					
etapas	1	2	3	4	5	6
A	X					
B					X	
C			X	X		
D		X				X

OPc

El retardo de la lógica combinacional de cada etapa es, respectivamente, de 154 ns, 196 ns, 180 ns y 170 ns.

El retardo de cada registro separador de desacoplo es de 4 ns.

Pregunta 1: *Calcule la secuencia óptima de latencias de inicio, cuando UFS realiza una sucesión ilimitada de operaciones, todas del tipo OPb.*

Pregunta 2: *Calcule la latencia media mínima, cuando UFS realiza una sucesión ilimitada de operaciones, todas del tipo OPc.*

Pregunta 3: *Calcule el tiempo de ciclo en ns. y la frecuencia en MHz, de UFS.*

Pregunta 4: *Calcule la velocidad de UFS, cuando realiza una sucesión ilimitada de operaciones repitiendo la terna OPa OPb OPc siempre en este orden, medida en MOPS (millones de operaciones por segundo).*

Pregunta 5: *Calcule la ganancia de UFS sobre una unidad funcional no segmentada construida con la misma lógica combinacional, cuando realizan una sucesión ilimitada de operaciones repitiendo la terna OPa OPb OPc siempre en este orden.*

Pregunta 6: *Calcule el tiempo (en seg.) que tarda UFS en realizar 4 millones de operaciones, todas del tipo OPc.*

Pregunta 7: *Calcule la productividad de UFS, en función del número (n) de operaciones, cuando realiza una sucesión de operaciones repitiendo la pareja OPb OPc siempre en este orden. Considere n par.*

Pregunta 8: *Calcule la productividad ideal de UFS, cuando realiza una sucesión ilimitada de operaciones, que son todas del tipo OPa.*

Pregunta 9: *Calcule el número mínimo (NM) de operaciones, todas del tipo OPa, que debe realizar UFS para alcanzar el 90% de la productividad ideal calculada en el anterior apartado.*

Otra unidad funcional segmentada multifunción (UFSlinc) es igual que la UFS pero con dos etapas más, ya que se añaden copia de la C y copia de la D. De esta manera consigue realizar de forma lineal (sin conflictos, 6 ciclos y 6 etapas) cualquier sucesión de operaciones OPc. Las Tablas de Reserva de las operaciones se modifican para usar las 6 etapas de la forma más eficaz posible y sin cambiar la cantidad de ciclos (4 para las OPa y 7 para las OPb).

Pregunta 10: *Calcule la latencia media mínima, cuando UFSlinc realiza una sucesión ilimitada de operaciones, todas del tipo OPb.*

Pregunta 11: *Calcule la energía (en julios) para que UFSlinc realice tres mil millones de operaciones, todas del tipo OPb, sabiendo que su potencia es de 50 w.*

Pregunta 12: *Calcule la relación entre tensiones de alimentación V(UFSlinc) / V(UFS) si ambas tienen igual potencia. Sabemos que UFSlinc tiene la misma frecuencia y tiene el doble de capacidad efectiva equivalente que UFS.*

Ejercicio 2.12

Suponga un procesador segmentado con la siguiente estructura de etapas:

ciclos	1	2	3	4	5
	CP	B	DL	M / ALU	E
			registros	lectura/ escritura	registros

Los tipos de instrucciones que se ejecutan en este procesador segmentado son los siguientes:

instrucción	registros	descripción
operación	Ri Rk Rj	Ri = Rk + Rj
load	Ri Rk Rj	Ri = Mem(Rk) $Rk_{nuevo} = Rk_{viejo} + Rj$
store	Ri Rk Rj	Mem(Ri) = Rj $Ri_{nuevo} = Ri_{viejo} + Rk$
br	R0 lit. 1 lit. 2 cond	si se cumple la condición que ha establecido una instrucción anterior saltar a la instrucción cuya dirección está en R0viejo, sino seguir en secuencia. En cualquier caso R0 = CP + lit.1 R1 = R0 + lit.2

Pregunta 1: *Indique cuántos recursos son necesarios y en qué etapas se utilizan para que en cualquier ciclo se pueda estar ejecutando cualquier combinación de instrucciones. Los registros R0, R1 y CP no pertenecen al banco de registros generales.*

Recursos a considerar:

- *Caminos de acceso a memoria.*
- *Caminos de acceso al banco de registros (lectura y escritura).*
- *Caminos de acceso a los registros R0, R1 y CP (lectura y escritura).*
- *ALU y sumadores.*

NOTA: un recurso se ocupa durante todo un ciclo.

Ejercicio 2.13

Tenemos tres clases de bloques A, B y C, de lógica combinacional, cuyos retardos respectivos son t(A) = 50 ns, t(B) = 90 ns. y t(C) = 80 ns. De cada clase de bloque disponemos de cualquier cantidad.

Tenemos registros cuyo retardo es de 10 ns.

Con estos elementos se realiza cierta operación, que necesita usarlos siguiendo este patrón:

A B A C B

Queremos realizar una sucesión ilimitada de estas operaciones, que no tienen ninguna dependencia entre ellas.

Pregunta 1: *Calcule la productividad en millones de operaciones por segundo y calcule el tiempo de ciclo en nanosegundos, si usamos una Unidad Funcional no segmentada que tiene 5 bloques.*

Pregunta 2: *Calcule la productividad en millones de operaciones por segundo y calcule el tiempo de ciclo en nanosegundos, si usamos una Unidad Funcional segmentada no lineal con tres etapas, en cada una de las cuales ponemos uno de los bloques disponibles.*

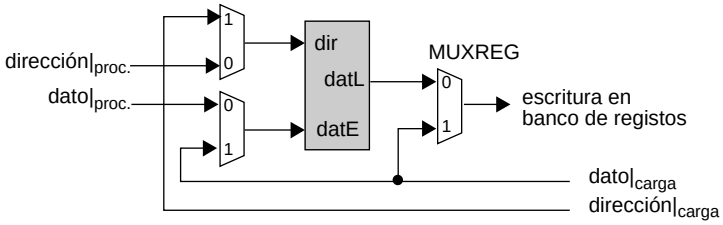
Pregunta 3: *Calcule la productividad en millones de operaciones por segundo y calcule el tiempo de ciclo en nanosegundos, si usamos una Unidad Funcional segmentada lineal con cinco etapas, en cada una de las cuales ponemos uno de los bloques disponibles, y son iguales entre sí los de la 3ª y la 1ª, y los de la 5ª y la 2ª.*

Pregunta 4: *Cierta Unidad Funcional segmentada no lineal utiliza un reloj de 4 MHz, y tiene una productividad de 1.6 millones de operaciones por segundo cuando realiza una sucesión ilimitada de operaciones iguales e independientes entre sí. Sabemos que, desde que comienza la 2ª operación, en cada ciclo hay siempre DOS operaciones sucesivas usando esta Unidad Funcional simultáneamente. Calcule la secuencia de latencias de inicio, que está formada por dos valores diferentes. Calcule el número de ciclos que dura cada operación.*

Ejercicio 2.14

Un procesador dispone de dos niveles de cache. El segundo nivel de cache lo etiquetaremos como L2 y el primer nivel de cache de datos lo etiquetaremos como L1. En la siguiente figura se muestra un esquema simplificado del camino de datos a la cache L1. Sólo existe un camino de acceso a la cache. Este camino se utiliza para lecturas y escrituras y está compartido por los accesos efectuados desde el procesador y las acciones de carga desde el

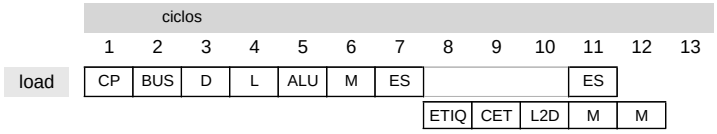
segundo nivel de cache. Observe que existe un camino que permite que mientras se está efectuando la carga del bloque en cache se puede escribir el dato accedido en el banco de registros (MUXREG). En el dibujo, para simplificar el trazado de los cables, se han representado separados el camino de datos de escritura (datE) y de lectura (datL). Tampoco se muestra en el dibujo la lógica necesaria para actualizar la cache L2.



En este procesador se ha implementado el concepto de cache no bloqueante en L1 y load no bloqueante. Esto es, la cache L1 sigue sirviendo accesos después de un fallo de cache y el procesador sigue interpretando instrucciones, después de que una instrucción load falle en cache, mientras no sea necesario como dato fuente de una instrucción el dato que carga el load en un registro.

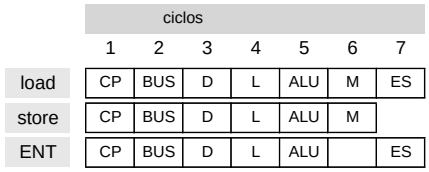
Supondremos que esta última característica (load no bloqueante) está limitada y sólo se permite un load en curso de interpretación que haya fallado en la cache L1. Esto es, si se está sirviendo un fallo de cache debido a una instrucción load y un load posterior falla en la cache L1 se bloquea el procesador.

En la siguiente figura se muestra la segmentación de una instrucción load. Los siete primeros ciclos son la segmentación usual con la diferencia de que decodificación y lectura del banco de registros se efectúan en etapas separadas. Cuando se detecta un fallo en la cache L1 no se actualiza el banco de registros en el ciclo 7 y se inicia el acceso a la cache L2. En el ciclo 8 se accede al campo etiqueta de la cache L2 y en el ciclo 9 se comprueba si es acierto o fallo.Si es un acierto en la cache L2, en el ciclo 10 se accede al campo datos y en los ciclos 11 y 12 se actualiza la cache L1. Adicionalmente en el ciclo 11 se actualiza el banco de registros con el dato accedido por la instrucción load. Note que desde la cache L2 se transfiere en primer lugar el trozo del bloque que contiene el elemento accedido por la instrucción load.



El número de caminos de escritura al banco de registros es 1.

Suponga que todas las instrucciones utilizan la siguiente segmentación. En las instrucciones de acceso a memoria se supone acierto en la cache L1. Una casilla en blanco indica que la información en los registros de desacoplo de entrada de la etapa se transfiere, sin procesar, a los registros de desacoplo de salida.



Pregunta 1: *Muestre, mediante un diagrama temporal, si se pueden producir riesgos estructurales después de que un load falle en la cache L1 y acierte en la cache L2. Indique las latencias prohibidas junto con el tipo de instrucción y los recursos que producen conflicto.*

La etapa L es una etapa de retención y mediante ella se pueden gestionar posibles conflictos en etapas posteriores. Por tanto, se puede utilizar para gestionar riesgos estructurales que se pueden producir debido a un fallo de cache de primer nivel.

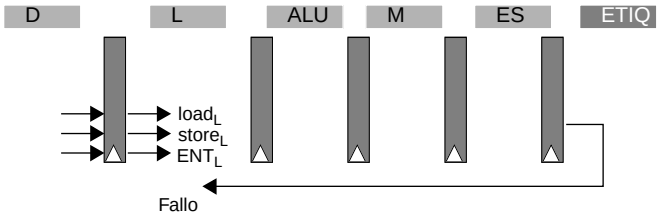
Sin embargo, en la etapa L sólo se dispone de información de acierto/fallo relativa a la cache L1, pero no se dispone de información de acierto/fallo relativa a la cache L2. La información de acierto/fallo en L1 está disponible en la etapa L en el ciclo siguiente al de de escritura de una instrucción load (ciclo 8).

Pregunta 2: *Proponga un mecanismo que gestione los riesgos estructurales que se pueden producir cuando se detecta un fallo en la cache L1. Note que el mecanismo debe ser conservador (hay que considerar el peor caso), ya que no se dispone de información de acierto o fallo de la cache L2. Por tanto, hay que suponer que se acierta en la cache L2.*

Pregunta 3: *Muestre en un diagrama temporal la interpretación segmentada de la secuencia de instrucciones que se muestra seguidamente. Suponga que la 1ª instrucción load provoca un fallo en la cache L1 y en la cache L2 está almacenado el bloque.*

1. load	5. add
2. add	6. store
3. load	7. load
4. add	8. add

Pregunta 4: *En la etapa L se dispone de las señales $load_L$, $store_L$ y ENT_L que indican el tipo de instrucción que ocupa la etapa. Además, se dispone de la señal Fallo que indica que se ha producido un fallo en la cache L1.*



Diseñe un circuito que indique que se producirá un riesgo estructural si la instrucción que está en la etapa L prosigue su interpretación.

Usualmente cuando el lenguaje máquina de un procesador incluye instrucciones de coma flotante se dispone de un banco de registros para números representados en coma flotante. Este banco es disjunto del banco que utilizan las instrucciones de aritmética entera.

Un caso particular son las instrucciones de acceso a memoria de coma flotante (Fload y Fstore) que utilizan los dos bancos de registros. Del banco de registros de enteros se leen los datos utilizados para calcular la dirección efectiva. El banco de registros de coma flotante se utiliza para escribir el dato que se trae de memoria en instrucciones Fload o leer el dato que se almacena en memoria en instrucciones Fstore.

Para soportar la disparidad de latencia de cálculo de las instrucciones de cálculo de coma flotante y la instrucción Fload se dispone de 2 caminos de escritura al banco de registros de coma flotante. Un camino es exclusivo para la instrucciones Fload y el otro camino lo comparten las unidades funcionales que se utilizan para efectuar los cálculos de coma flotante.

En este contexto, los riesgos estructurales que pueden producirse después de que una instrucción Fload falle en la cache L1 son sólo debidos a instrucciones de acceso a memoria.

Pregunta 5: *Muestre, mediante un diagrama temporal, si se pueden producir riesgos estructurales después de que un Fload falle en la cache L1 y acierte en la cache L2. Indique las latencias prohibidas junto con el tipo de instrucción y los recursos que producen conflicto.*

En la cache L1 se utiliza escritura inmediata y no se asigna un contenedor si se produce un fallo de cache al interpretar una instrucción store. Además, los dos niveles de cache que estamos suponiendo se gestionan de forma que el contenido de la cache L1 es un subconjunto del contenido de la cache L2 (propiedad de inclusión). Esto es, todo bloque almacenado en la cache L1 está almacenado en la cache L2.

La mayor parte de datos que manipulan los programas denominados numéricos o de coma flotante están representados en coma flotante y los conjuntos de trabajo de datos a los que acceden ocupan bastante memoria. Supondremos que estos conjuntos de trabajo son bastante mayores que la capacidad del primer nivel de cache.

En estas condiciones ser conservador, para gestionar los riesgos estructurales cuando se detecta un fallo en la cache L1 no es una idea afortunada, ya que existe una alta probabilidad de que la intrucción Fload que se retiene en la etapa L también produzca un fallo en la cache L1.

En su lugar, lo que puede hacerse es tomar las siguientes decisiones cuando en la etapa L se determina que se producirá un conflicto:

- una instrucción Fload produciría el conflicto. Se fuerza un fallo en la cache L1. La instrucción no accede a la cache L1 y directamente se encamina a la cache L2. Para ello se utiliza la misma segmentación que en un fallo en la cache L1, pero en los ciclos que se accedería a la cache L1 no se ocupan los recursos.
- una instrucción Fstore produciría el conflicto. El Fload o load previo con quien entra en conflicto almacena el dato en el banco de registros pero no se actualiza la cache L1 con el bloque que se ha traido de la

cache L2.

Pregunta 6: *Dadas las siguientes secuencias de instrucciones utilice un diagrama temporal para mostrar su interpretación segmentada. Suponga que el primer Fload provoca un fallo en el primer nivel de cache.*

Fload	Fload	Fload
add	add	add
add	add	add
add	add	add
add	add	add
Fload	Fstore	add
add	add	Fstore

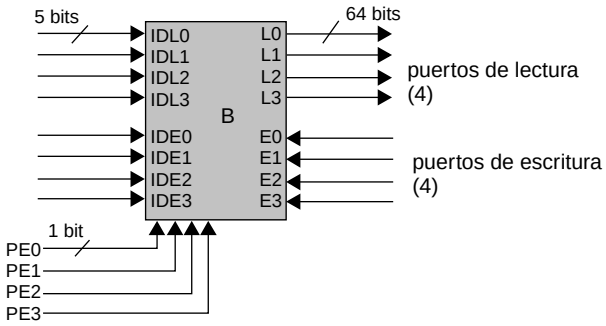
Ejercicio 2.15

En un procesador superescalar (IPC>1) el ancho de banda del banco de registros es bastante grande. Por ejemplo, si se interpretan 4 instrucciones de forma paralela son necesarios 8 caminos de lectura y 4 caminos de escritura como mínimo para que no se produzcan conflictos.

El número de registros y el número de caminos de acceso a un banco de registros determinan la latencia de acceso además del área de chip ocupada. En ocasiones el tiempo de acceso al banco de registros puede limitar el tiempo de ciclo de un procesador segmentado. Una alternativa para adecuarse al tiempo de ciclo del procesador es la segmentación, la replicación o ambas.

Un banco de registros está compuesto por decodificadores y una matriz de celdas de almacenamiento. Hay un decodificador por camino de acceso al banco de registros y se utilizan para seleccionar el registro al que se quiere acceder y la lógica es combinacional. Sin embargo, la matriz de celdas de almacenamiento tiene componentes cuyo funcionamiento es analógico (no digital). En estas condiciones la técnica de segmentación que utiliza registros de desacoplo no se puede utilizar, ya que no se puede segmentar el acceso a la matriz de celdas de almacenamiento.

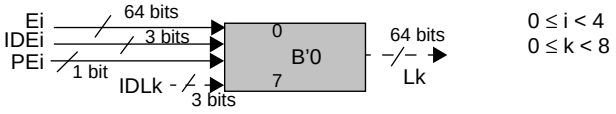
Para adecuarse al tiempo de ciclo del procesador se dispone de bancos de registros con 4 puertos de lectura y 4 puertos de escritura y 32 registros. A este banco de registros lo denominaremos elemento base. En la siguiente figura se muestra un banco de registros con los puertos de lectura y escritura.



El par Li-IDLi es un camino de lectura. El cable etiquetado Li transporta el dato leído y el cable etiquetado IDLi se utiliza para suministrar al banco de registros el identificador del registro que se quiere leer. La interpretación de la tripleta Ei-IDEi-PEi es pareja con la salvedad de que Ei transporta el dato que se quiere escribir y la señal PEi es el permiso de escritura.

Pregunta 1: *Proponga un diseño de un banco de registros con 8 puertos de lectura y 4 puertos de escritura con el mismo tiempo de acceso que el elemento base.*

Debido a un incremento de la frecuencia de funcionamiento del procesador el nuevo elemento base es un banco de registros con 8 registros, 8 caminos de lectura y 4 caminos de escritura. En la siguiente figura se muestra el nuevo elemento base.



Pregunta 2: *Utilizando el nuevo elemento base, multiplexores, decodificadores y puertas lógicas diseñe un camino de acceso a un banco de registros con 32 registros. Registros con identificadores de registros consecutivos deben almacenarse en el mismo elemento base mientras sea posible. El identificador de registro en el banco de 32 registros debe seccionarse para construir un identificador de registro para un banco de registros de 8 registros (IDEi) y establecer el valor de la señal PEi que indica si debe actualizarse el banco de registros de 8 registros en una acción de escritura.*

Pregunta 3: *El tiempo de acceso al elemento base son 2 ns y el retardo de un multiplexor de 2 entradas es 0.5 ns. Segmente la acción de lectura al banco de registros para adecuarse al tiempo de ciclo (2ns).*

Ejercicio 2.16

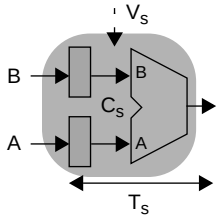
Una reducción en la tensión de alimentación da lugar a un ahorro cuadrático en la potencia consumida y por tanto, es una alternativa muy atractiva para reducir el consumo de potencia.

$$Potencia = C \times V^2 \times f$$

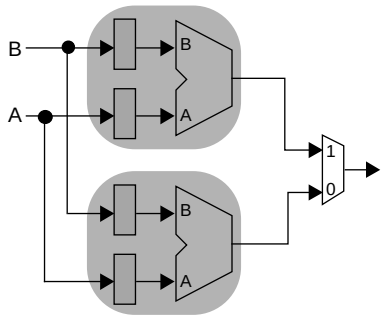
Sin embargo, dentro de un rango de funcionamiento, el retardo (r = 1/f) de las puertas CMOS se incrementa inversamente con la tensión de alimentación. En la siguiente tabla las filas muestran relaciones entre la tensión de alimentación y el retardo. La tensión de alimentación y el retardo original se denominan V_0 y r_0 respectivamente.

tensión de alimentación	retardo
0.58 x V_0	2 x r_0
0.40 x V_0	4 x r_0

Disponemos de un módulo sumador alimentado con una tensión de alimentación V_s, una capacidad efectiva equivalente C_s y con un retardo $T_s = 1/f_s$, donde f_s es la frecuencia de funcionamiento.



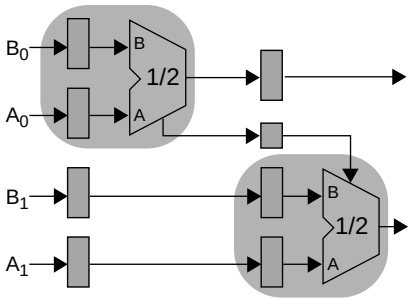
Suponga que se replica el sumador. La capacidad efectiva equivalente aumenta por un factor de 2. Además, este factor se incrementa en un 7% debido al necesario encaminamiento y multiplexación de los datos.



Pregunta 1: *Calcule la potencia consumida respecto del original cuando el sistema con dos réplicas tiene la misma productividad que el dispositivo original.*

Pregunta 2: *El área del diseño con dos réplicas se incrementa en un factor de 3. Calcule la densidad de potencia respecto del original.*

El sumador original se puede seccionar en dos sumadores cada uno de tamaño mitad y segmentar el diseño.



Esta alternativa incrementa la capacidad efectiva un 15% debido a los registros adicionales necesarios.

Pregunta 3: *Calcule la potencia consumida respecto del original cuando el sistema segmentado tiene la misma productividad que el dispositivo original.*

Pregunta 4: *Suponga un sistema donde se utiliza la técnica de replicación y segmentación: 2 réplicas y 2 etapas en cada réplica. Calcule la potencia consumida respecto del original cuando este sistema tiene la misma productividad que el dispositivo original.*

Ejercicio 2.17

Sea UFNS una Unidad Funcional no segmentada con las siguientes características:

Parámetros	Descripción
t_R	retardo de propagación del registro
t_L	retardo de propagación de la lógica combinacional
E_R	energía imputable al registro
E_L	energía imputable a la lógica combinacional

Sea UFSeg una versión segmentada lineal en N etapas de UFNS (N>1) con las siguientes características:

Parámetros	Descripción
t_{Li}	distribución uniforme del retardo de la lógica combinación, es decir $t_{Li} = t_L \times N^{-1}$
E_{Li}	la energía imputable a la lógica combinacional de cada etapa es $E_{Li} = E_L \times N^{-1/2}$
t_R	retardo de propagación de un registro de desacoplo
E_R	energía de un registro de desacoplo

Calcule, para UFNS y UFSeg, en función de t_R, t_L, E_R, E_L y N:

Pregunta 1: *La productividad al realizar una secuencia ilimitada de operaciones independientes.*

Pregunta 2: *La energía consumida en un ciclo de reloj.*

Pregunta 3: *La potencia consumida.*

Pregunta 4: *La métrica de eficiencia productividad / potencia.*

Suponga que t_R=0, E_R=0 y que UFNS y UFseg se alimentan mediante baterías con la misma carga inicial.

Pregunta 5: *¿Qué unidad funcional descarga la batería en menos tiempo?.*

Pregunta 6: *¿Qué unidad funcional realiza más operaciones antes de agotar la batería?.*

Ejercicio 2.18

Una cache de instrucciones es capaz de suministrar 6 instrucciones cada T_{cache}. El decodificador de instrucciones que es alimentado por esta cache es capaz de procesar 3 instrucciones cada T_{deco}.

La relación entre los tiempos de acceso a la cache de instrucciones y el decodificador es

$$T_{cache} = T_{deco} \times 2$$

Pregunta 1: *Diseñe un sistema con una cache de instrucciones y un decodificador que funcione con la máxima productividad posible. Si es necesario puede utilizar un buffer de almacenamiento.*

Ejercicio 2.19

Un procesador segmentado utiliza la siguiente segmentación y recursos, describiéndose la funcionalidad de las etapas en la tabla que se adjunta. En este ejercicio no se consideran las instrucciones de secuenciamiento.

clases / tipos		ciclos 1	2	3	4	5	6
ENT	etapas	CP	B	D/L	A	ES	
	recursos	+	MID	BRL	ALU	BRE	
load	etapas	CP	B	D/L	A	M	ES
	recursos	+	MID	BRL	ALU	MID	BRE
store	etapas	CP	B	D/L	A	M	
	recursos	+	MID	BRL	ALU	MID	

ETAPA	FUNCIONALIDAD
CP	determinar el CP
B	búsqueda de la instrucción
D/L	decodificación, lectura de registros
A	operación aritmética
M	acceso a memoria de datos
ES	escritura en el banco de registros

El número de caminos de acceso al banco de registros es 1 para lecturas y 1 para escrituras. El número de caminos de acceso a la memoria MID es uno de lectura/escritura. Los recursos cuyo retardo se considera distinto de cero son los siguientes:

recursos	registro de desacoplo	+	MID	BRL	ALU	BRE
retardo en ns	1	1	5	2	5	2

el tiempo de ciclo es de 6 ns

El número máximo de operandos fuente de una instrucción que deben leerse del banco de registros es 2. Una instrucción actualiza como máximo una posición de memoria o un registro del banco de registros.

Pregunta 1: *Cuando se interpreta una secuencia de instrucciones, el recurso MID puede producir riesgos estructurales. Determine las latencias prohibidas.*

Pregunta 2: *¿Existen en el camino de datos otros recursos que puedan producir riesgos estructurales?. Justifique la respuesta.*

En la siguiente figura se muestra la gestión de riesgos estructurales debido al recurso MID.

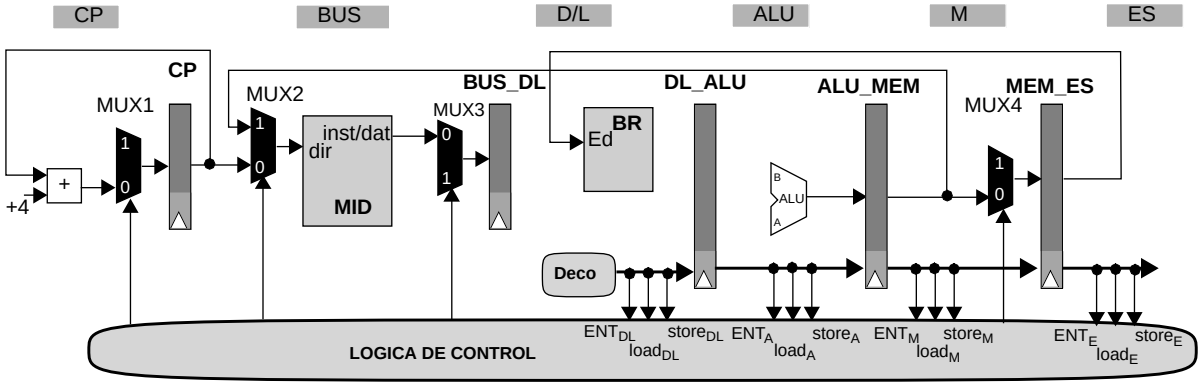
dirección	instrucción	ciclos 1	2	3	4	5	6	7	8	9	10	11	12
16	load	CP	BUS	D/L	ALU	M	ES						
20	store		CP	BUS	D/L	ALU	M						
24	add			CP	BUS	D/L	ALU	ES					
28	sub				CP		nop	nop	nop				
28						CP		nop	nop	nop			
28							CP	BUS	D/L	ALU	ES		
32	load							CP	BUS	D/L	ALU	M	ES
36	add								CP	BUS	D/L	ALU	ES

La 1ª columna indica la dirección que se calcula en la etapa CP en la fila correspondiente

Una casilla en blanco indica que no se accede al recurso utilizado en la etapa. El recurso está siendo utilizado por otra instrucción

El riesgo estructural se detecta en la etapa D/L y la etapa de retención es la etapa CP. En el ciclo que se produce el riesgo la instrucción más vieja accede a la memoria MID, la instrucción más joven se retiene en la etapa CP y se inyecta una instrucción nop a la etapa D/L.

En la siguiente figura se muestra un camino de datos simplificado e incompleto del procesador segmentado. La activación de una señal ENT_X, $load_X$, o $store_X$ indica que la etapa X está ocupada por una instrucción de la clase/tipo ENT, load, o store respectivamente.



Pregunta 3: *Añada al camino de datos, mostrado previamente, las conexiones que faltan para transportar datos y gestionar los riesgos debidos al recurso MID. Indique también desde dónde se inyectan las instrucciones nop. Un cable que cruce un registro de desacoplo se considerará afectado por el funcionamiento del registro de descoplo.*

Pregunta 4: *Diseñe la lógica de control de los multiplexores MUX1, MUX2, MUX3 y MUX4.*

Ejercicio 2.20

La Unidad Funcional Segmentada no lineal UFS 1 procesa un tipo de operaciones usando tres etapas A, B y C durante 6 ciclos, siguiendo este patrón:

ciclos					
1	2	3	4	5	6
A	B	B	C	B	C

La Unidad Funcional Segmentada no lineal UFS 2 procesa un tipo de operaciones usando tres etapas D, E y F durante 4 ciclos, siguiendo este patrón:

ciclos			
1	2	3	4
D	E	F	D y E

Pregunta 1: *Calcule, para la UFS 1.*
1/ una secuencia periódica de latencias de inicio permitidas
2/ el valor de la latencia media
3/ la productividad en operaciones por ciclo, siendo N el total de operaciones

Pregunta 2: *Calcule, para la UFS 2.*
1/ una secuencia periódica de latencias de inicio permitidas
2/ el valor de la latencia media
3/ la productividad en operaciones por ciclo, siendo N el total de operaciones

Formamos ahora la UFS 3, cuya entrada es la entrada de UFS 1 y cuya salida es la salida de UFS 2, conectando directamente la salida de UFS 1 como entrada para la UFS 2. Es decir, hemos

concatenado una tras la otra. De este modo UFS 3 procesa un tipo de operaciones usando seis etapas A, B, C, D, E y F durante 10 ciclos, de acuerdo con el siguiente patrón:

ciclos									
1	2	3	4	5	6	7	8	9	10
A	B	B	C	B	C	D	E	F	D y E

Pregunta 3: *Sabemos que UFS 1 sigue la secuencia periódica de latencias de inicio calculada en la 1ª pregunta. Justifique si UFS 2 puede o no puede seguir la secuencia periódica de latencias de inicio calculada en la 2ª pregunta. En caso de que no pueda, calcular cual debería seguir.*

Pregunta 4: *Calcule cuantos ciclos tarda UFS 3 en terminar las 5 primeras operaciones, y calcule cuantas operaciones ha terminado al cabo de los primeros 50 ciclos.*

Ejercicio 2.21

Tenemos que interpretar tres tipos diferentes de instrucciones

tipos	FUNCIONALIDAD	LATENCIA (ciclos)
ent	rc : = ra (op) rb	7
load	ra : = M [rb + lit]	8
store	M[rb + lit] : = ra	6

Los nombres ra , rb, y rc identifican registros del banco de uso general. El nombre lit identifica a una constante.

Un procesador segmentado (el PSM6) tiene 6 etapas. Interpreta las instrucciones siguiendo estas Tablas de Reservas:

	ciclos						
etapas	1	2	3	4	5	6	7
CP	X						
B		X					
DL			X	X			
A					X		
M							
E						X	X

ent

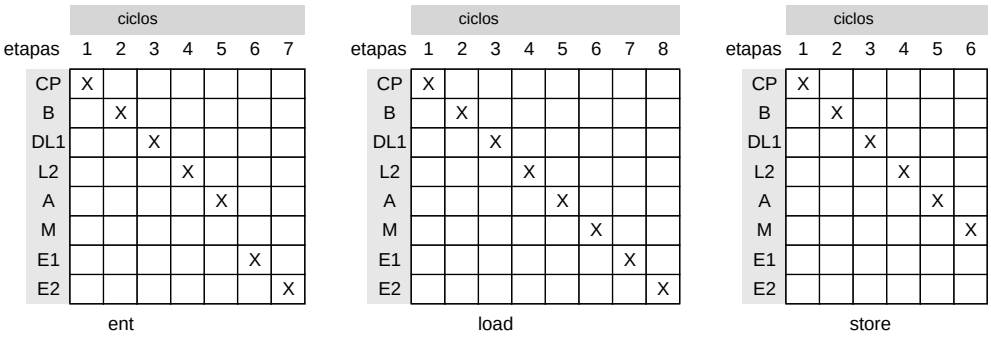
	ciclos							
etapas	1	2	3	4	5	6	7	8
CP	X							
B		X						
DL			X	X				
A					X			
M						X		
E							X	X

load

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

store

Otro procesador segmentado (el PS8) tiene 8 etapas. Interpreta las instrucciones siguiendo estas Tablas de Reservas :



Observe que la lectura (o la escritura) de un registro del banco emplea dos ciclos consecutivos en cada uno de los dos procesadores. Pero en el PS8 se ha segmentado en dos etapas y en el PSM6 no .

Cualquiera de estos dos procesadores puede tener conflictos que causen riesgo estructural (R. E.) en algunas etapas. Vamos a suponer que todo R. E. se resuelve siempre bloqueando el inicio de nuevas instrucciones durante los ciclos necesarios. Es decir que, cuando la latencia ideal de inicio igual a 1 sea prohibida, la aumentamos hasta el mínimo valor que sea permitido.

Cada uno de estos dos procesadores tiene que interpretar el mismo código, que es una secuencia de estas 6 instrucciones :
load load ent ent ent store
que se repite indefinidamente. No hay ningún Riesgo de Datos.

Pregunta 1: *Presente el cronograma cuando el PSM6 interpreta ese código, sin ninguna limitación en la cantidad de puertos de su Banco de Registros. Marque cada ciclo sin permiso de inicio mediante un cuadro en negro. Es suficiente representar 7 instrucciones, las 6 de una secuencia y la primera de la siguiente.*
Calcule el valor de la Latencia Media.
Calcule el mínimo número de puertos de lectura y de puertos de escritura que se necesitan.

Pregunta 2: *Calcule la frecuencia, en GHz, del PSM6, sabiendo que al interpretar ese código consigue una velocidad de 600 MIPS.*

Pregunta 3: *Presente el cronograma cuando el PS8 interpreta ese código, sin ninguna limitación en la cantidad de puertos de su Banco de Registros. Marque cada ciclo sin permiso de inicio*

mediante un cuadro en negro. Es suficiente representar 7 instrucciones, las 6 de una secuencia y la primera de la siguiente. Calcule el valor de la Latencia Media.

Pregunta 4: *Calcule el número de veces que el PS8 ha interpretado la secuencia de 6 instrucciones, sabiendo que durante ese tiempo consigue una Productividad media cuyo valor es de 0.75 instrucciones/ciclo.*

Pregunta 5: *Un procesador no segmentado (el PNS) tiene f = 150 MHz e interpreta esos mismos tres tipos diferentes de instrucciones y con las mismas diferentes latencias medidas en ciclos. Gasta una energía igual a 0,043 joules cuando interpreta 30000 instrucciones de ese mismo código. Calcule la Potencia, en watios, del PNS.*

Pregunta 6: *Otro procesador no segmentado (el PNSBis) tiene f = 120 MHz. Interpreta las instrucciones ent con los mismos ciclos de latencia que el PNS (7 ciclos). En cambio interpreta las load y las store con menos ciclos de latencia. Sabemos que tiene ganancia G = 1.075 sobre el PNS cuando interpreta ese mismo código (secuencias de 6 instrucciones load, load, ent, ent, ent, store). Calcule el cociente número de ciclos en el PNS entre número de ciclos en el PNSBis para interpretar la parte que no son instrucciones ent.*

Ejercicio 2.22

La Unidad Funcional Segmentada UFS procesa tres tipos de operaciones usando sus seis etapas A , B , C , D , E y F durante 7, 6 o 5 ciclos, siguiendo estos patrones:

	ciclos						
	1	2	3	4	5	6	7
op1	A	B	C	D	E	E	F
op2	A	B	C	D	E	E	
op3	A	B	C	D	F		

Pregunta 1: *Describa todos los riesgos estructurales (R. E.) por conflicto en el uso de cualquier etapa que puede sufrir esta UFS si se pretende iniciar una nueva operación a cada ciclo. Las operaciones pueden ser cualquier combinación de los tres tipos descritos, y suponemos que nunca presentan dependencias de datos entre ellas. La descripción consiste en indicar la secuencia de operaciones, formada por la correspondiente lista de nombres y marcando la etapa de conflicto.*

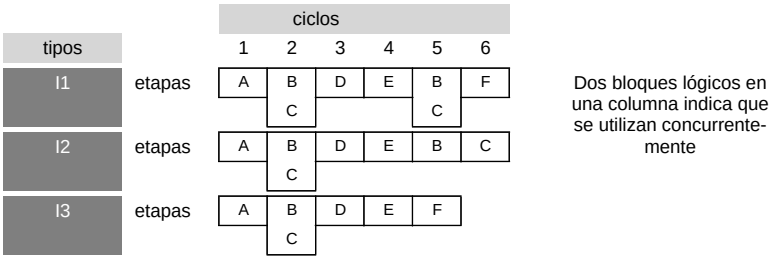
Pregunta 2: *Para resolver cualquier R. E. se aplica la solución de retrasar el comienzo de la nueva operación, aumentando así la latencia de inicio o repetición. Muestre un cronograma ciclo a ciclo de cada uno de los casos con R. E. del apartado anterior.*

Pregunta 3: *Tenemos que ejecutar una secuencia de nueve operaciones formada de la manera siguiente:*
op2 op3 op2 op1 op3 op3 op1 op1 op2
Presente el cronograma completo, ciclo a ciclo.

Pregunta 4: *Hacemos la secuencia de operaciones del apartado anterior durante diez repeticiones idénticas y consecutivas. Tenga en cuenta, naturalmente el solapamiento entre dos repeticiones sucesivas, para lo cual conviene que continúe durante algunos ciclos el cronograma anterior.*
Calcule la Productividad, expresada en operaciones por ciclo.

Ejercicio 2.23

En un procesador segmentado se distinguen 6 bloques de lógica (A, B, C, D, E y F) que se utilizan en los ciclos que se especifican seguidamente en función del tipo de instrucción.



Suponga que antes del 1º ciclo de interpretación de una instrucción se conoce su tipo. Por tanto, una instrucción no inicia la interpretación hasta que se conoce que no se producen riesgos estructurales.

Pregunta 1: *Indique la latencia media de inicio si se interpreta una secuencia ilimitada de instrucciones del mismo tipo. Así mismo, indique las latencias prohibidas que determina una instrucción con otras del mismo tipo y especifique el ciclo de la instrucción más joven, relativo a su posible y fallido inicio de interpretación, en el cual se produce el riesgo estructural.*

Pregunta 2: *Indique las latencias prohibidas que determina una instrucción de un tipo con una secuencia de instrucciones de otro tipo y especifique el ciclo de la instrucción más joven, relativo a su inicio de interpretación, en el cual se produce el riesgo estructural. Considere los siguientes 6 casos:*

a) (I1, I2, . . . , I2, . . . , I2)
b) (I1, I3, . . . , I3, . . . , I3)
c) (I2, I1, . . . , I1, . . . , I1)
d) (I2, I3, . . . , I3, . . . , I3)
e) (I3, I1, . . . , I1, . . . , I1)
f) (I3, I2, . . . , I2, . . . , I2)

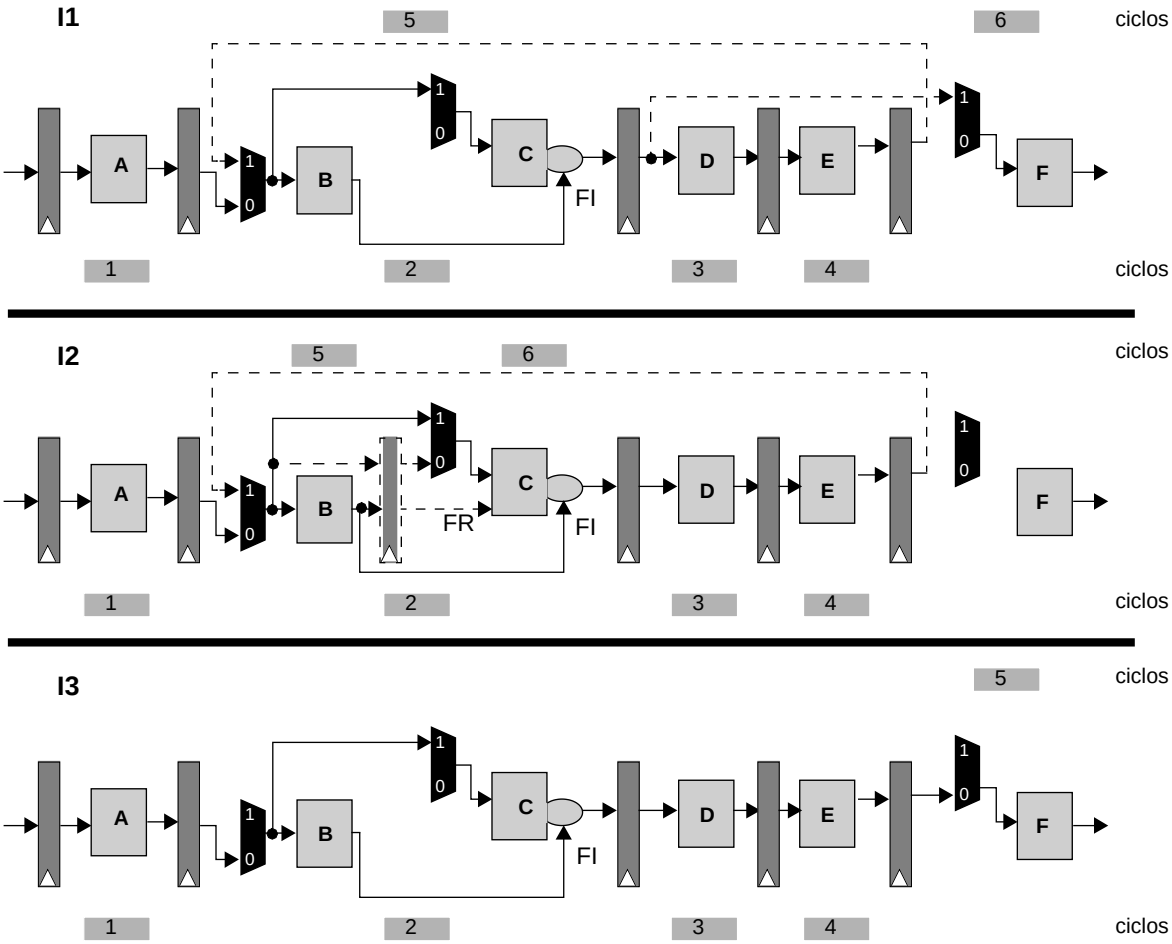
Pregunta 3: *Dada la siguiente secuencia de instrucciones (I2, I1, I3, I3, I3) muestre en un diagrama temporal la interpretación sin riesgos de las mismas.*

En el diseño segmentado descrito se quieren eliminar los riesgos estructurales. Para ello se puede incrementar el número de ciclos que tarda una instrucción en interpretarse, se pueden replicar recursos y/o se pueden insertar retardos. Para identificar los recursos replicados se utilizará el mismo acrónimo del bloque original con un sufijo numérico y un retardo se especificará con el acrónimo R.

Los bloques de lógica B y C se corresponden con una fase de interpretación y por ello, si se replica uno de ellos debe replicarse el otro. En el caso de que estos bloques de lógica se utilicen en ciclos distintos debe utilizarse el bloque B antes que el bloque C.

Pregunta 4: *Elimine todos los riesgos estructurales aplicando las técnicas relacionadas previamente, teniendo en cuenta que el número de recursos replicados de ser el mínimo posible y que la inserción de retardos tiene menor coste hardware. Aplique las técnicas en el siguiente orden: a) replicar bloques de lógica, b) incrementar la latencia de interpretación y c) inserción de retardos. Debe especificarse la aplicación de cada una de las técnicas de forma incremental y en el orden que se ha explicitado.*

En las siguientes figuras se muestra, de forma simplificada, un esquema del camino de datos utilizado por cada instrucción. En cada uno de ellos se han dejado sin conectar las entradas y bloques de lógica que permiten acomodar la interpretación de los 3 tipos de instrucciones en un mismo camino de datos.



En los esquemas del camino de datos, las líneas con trazo continuo muestran el flujo de información y utilización de las etapas en los 4 primeros ciclos de interpretación de todos los tipos de instrucciones y el 5º ciclo en las instrucciones de tipo I3. Las líneas con trazo discontinuo y registro de desacoplo con perfil discontinuo muestran el flujo de información y utilización de las etapas en el resto de ciclos en las instrucciones de tipo I1 e I2. En el bloque de lógica C uno de los flujos de información puede provenir directamente del bloque B (FI) o a través de un registro de desacoplo (FR). Ahora bien, en un ciclo determinado, en el bloque C, sólo se utiliza una de las señales (FI o FR). Cuando los 2 bloques trabajan concurrentemente en la misma etapa se utiliza la

señal FI y cuando los bloques trabajan en serie, en etapas distintas, se utiliza la señal FR. La dinámica de funcionamiento del primer caso puede modificarse de forma que se utilicen los bloques B y C en serie (2 ciclos). Ahora bien, en el segundo caso no se puede modificar la dinámica de funcionamiento (ciclos 5 y 6 en las instrucciones de tipo I2).

Pregunta 5: *Diseñe el camino de datos que se utilizaría en la respuesta de la pregunta 4). En el esquema deben identificarse de forma clara los recursos replicados. Muestre el camino para cada una de las instrucciones por separado, incluyendo los elementos que permitirán encaminar los datos para interpretar los 3 tipos de instrucciones. Si una línea cruza un registro de desacoplo se considera afectada por la señal de reloj que controla el registro de desacoplo.*

Ejercicio 2.24

Un dispositivo segmentado DS dispone de 5 bloques de lógica combinacional A, B, C, D y E. Este dispositivo efectúa dos tipos de operaciones y las siguientes tablas de reserva describen la utilización de los bloques en función del tiempo.

cicles	1	2	3	4	5	6	7
	A	B	C	D	E	B	D

OPa

cicles	1	2	3	4	5	6
	A	B	C	D	E	D

OPb

La frecuencia de reloj de DS es de 700 MHz.

Pregunta 1: *Determine las latencias de inicio prohibidas entre parejas de operaciones.*

Pregunta 2: *Muestre en un cronograma el procesado por DS de una secuencia ilimitada de operaciones OPa.*
Calcule el rendimiento en Millones de Operaciones Por Segundo (MOPS).
Calcule la mejora debida al solapamiento de operaciones.

Pregunta 3: *Muestre en un cronograma el procesado por DS de una secuencia ilimitada de operaciones OPb.*
Calcule el rendimiento en MOPS.
Calcule la mejora debida al solapamiento de operaciones.

Pregunta 4: *Muestre en un cronograma el procesado por DS de una secuencia ilimitada de operaciones siguiendo el patrón OPa, OPb, OPb, OPa.*
Calcule el rendimiento en MOPS.

Pregunta 5: *Suponga que se puede replicar un sólo bloque de lógica combinacional con el objetivo de mejorar el rendimiento. ¿Cuál de los 5 bloques replicaría?. Justifique la respuesta indicando, para este nuevo dispositivo segmentado, las latencias de inicio prohibidas entre parejas de operaciones.*