PROBLEMAS: SEGMENTACIÓN

■ 1. Sea una máquina no segmentada con 4 etapas de ejecución en la que cada etapa tiene la siguiente duración: 500ps, 500ps, 650ps y 450ps. La segmentación del procesamiento en esta máquina provoca que cada etapa incremente su duración en 45ps. Calcula la aceleración que obtenemos al introducir la mejora.

■ 2. Sea una máquina segmentada de 4 etapas (IF, ID, EX y WB). Desde el punto de vista de los accesos a memoria, la etapa IF realiza un acceso en lectura al área de código mientras que la etapa WB realiza accesos en lectura o escritura al área de datos en los casos de operaciones de tipo *load* o *store* respectivamente. Supongamos que tenemos una máquina cuya interfase con memoria dispone de un único puerto, es decir, en cada ciclo sólo puede realizar un acceso a memoria. Describe el parón estructural que provoca una instrucción de *load*.

■ 3. Sabemos que para un conjunto de programas de prueba, la frecuencia de uso de las instrucciones de *load*/*store* es del 45% del recuento total. Calcula la aceleración real de un cauce de 4 etapas con un único puerto de memoria asumiendo que no existen parones por dependencias de datos ni de control.

■ 4. Sea una máquina segmentada de 4 etapas (IF, ID, EX y WB). Todas las operaciones de enteros consumen un ciclo en la etapa de ejecución (EX). La multiplicación en coma flotante se procesa también en la etapa de ejecución (EX) pero con un operador no segmentado que tarda 5 ciclos. Determinar cómo es el parón estructural provocado por la aparición de una instrucción de multiplicación en coma flotante en una secuencia de instrucciones de enteros si la etapa de ejecución entera y la de coma flotante:

a) NO pueden avanzar en paralelo;
b) SÍ pueden avanzar en paralelo.

■ 5. Si un programa de prueba tiene un 5% de instrucciones de multiplicación en coma flotante, determinar el impacto del parón estructural descrito en el problema anterior. Discutir qué mejoras proporcionaría la segmentación en 5 etapas de la operación de multiplicación en coma flotante.

■ 6. Supongamos que en un programa de prueba se encuentra que el 45% de las instrucciones tienen referencias a datos. De ellas, el 30% provocan un parón por dependencias de datos en un cauce de 4 etapas. Suponiendo que no existen parones estructurales ni de control, ¿cuál es la aceleración respecto a la máquina no segmentada?

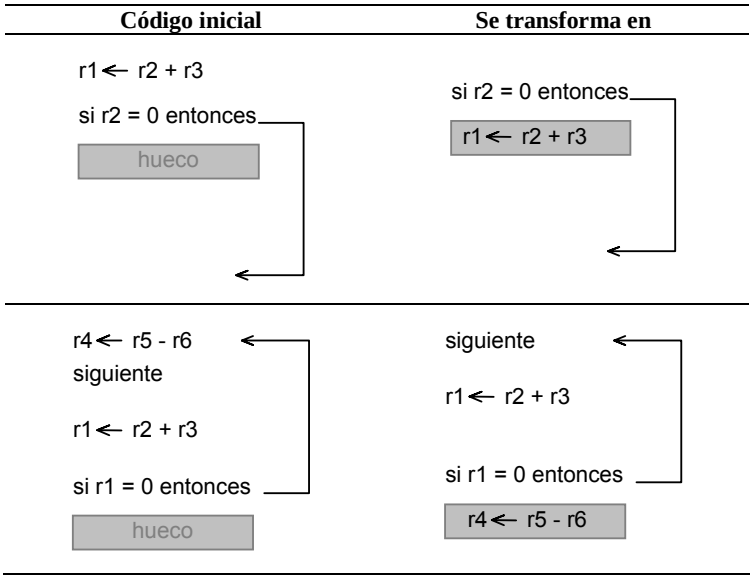
■ 7. En un conjunto de programas de prueba se ha encontrado que el 20% de las instrucciones son cargas y que en el 50% de las ocasiones la instrucción que sigue a la carga depende del resultado de la carga. Si este hecho provoca un parón de 1 ciclo, calcular la aceleración real respecto al caso no segmentado.

■ 8. Se han realizado una serie de *test* y se ha calculado que el 18% del recuento total de instrucciones pertenece a instrucciones de bifurcación. De ellas, el 85% son saltos condicionales y de ellos, el 65% alteran el contador de programa (saltos condicionales tomados). Si ejecutamos estos programas de *test* en un procesador segmentado, calcular la aceleración respecto al procesador no segmentado en función del esquema implementado para manejar los parones por dependencias de control:

a) Parón hasta conocer el resultado de la evaluación de la condición de salto;
b) Introducción de instrucción NOP,
c) Predecir siempre como no tomado,
d) Predecir siempre como tomado.

■ 9. Sea una máquina con un cauce de 4 etapas. La frecuencia con la que se producen detenciones entre la instrucción *i* y la *i*+1 es del 20% y provoca 2 parones. La frecuencia con la que se producen detenciones entre la instrucción *i* y la *i*+2 es del 5% y provoca 1 parón. Entre la instrucción *i* y la *i*+3 no se producen parones nunca.

■ 10. La bifurcación retardada es una técnica consistente en reordenar el código en tiempo de compilación con el fin de colocar detrás del salto condicional (hueco de retardo) una instrucción que ha de ejecutarse independientemente del destino del salto. No siempre es fácil encontrar una instrucción con esa característica. Discutir los casos presentados a continuación:



NOTA: casos tomados de "Arquitectura de computadores. Un enfoque cuantitativo". Hennessy Patterson. McGraw Hill, 1993.

■ 11. Algunos procesadores tienen la posibilidad de abortar la instrucción en el hueco de retardo si las circunstancias de la ejecución de requieren. Normalmente, el formato de la instrucción de salto contiene un bit que el compilador activa o desactiva indicando si la instrucción del hueco de retardo es anulable o no. Discute qué ventajas e inconvenientes concurren este mecanismo.