


33

EXPERIMENTALES Y
TECNOLOGÍA

POE
004.2
PRO
pro


www.dykinson.com

Javier Castillo
Emilio Castillo
José Luis Bosque
Pabo Huerta

PROBLEMAS RESUELTOS DE ARQUITECTURA DE COMPUTADORES

Universitat d'Alacant



500487056

UNIVERSIDAD REY JUAN CARLOS

ISBN 978-84-9849-341-2



9 788456 493412

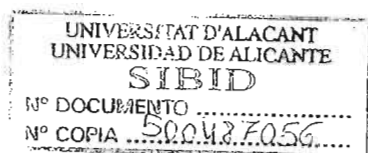
Problemas Resueltos de Arquitectura de
Computadores

FE COV. 2/PRO/pro

Problemas Resueltos de Arquitectura de Computadores

Javier Castillo, Emilio Castillo, José Luis Bosque, y Pablo Huerta

Todos los derechos reservados. Ni la totalidad ni parte de este libro, incluido el diseño de la cubierta, puede reproducirse o transmitirse por ningún procedimiento electrónico o mecánico. Cualquier forma de reproducción, distribución, comunicación pública o transformación de esta obra solo puede ser realizada con la autorización de sus titulares, salvo excepción prevista por la ley. Dirijase a CEDRO (Centro Español de Derechos Reprográficos, www.cedro.org) si necesita fotocopiar o escanear algún fragmento de esta obra.



© Copyright by
Los Autores
Madrid, 2008

Editorial DYKINSON, S.L. Meléndez Valdés, 61 - 28015 Madrid
Teléfono (+34) 91 544 28 46 - (+34) 91 544 28 69
e-mail: info@dykinson.com
<http://www.dykinson.es>
<http://www.dykinson.com>

ISBN: 978-84-9849-341-2
Depósito Legal: SE-6773-2008

Preimpresión realizada por el autor

Printed by Publidisa

Índice general

1. Introducción	9
2. Análisis de rendimiento	11
3. Segmentación	23
4. Planificación Dinámica	57
5. Jerarquía de Memoria	75
6. Sistema de E/S	95

Capítulo 1

Introducción

El presente libro es un compendio de ejercicios resueltos de las asignaturas de Organización de Computadores de segundo de Ingeniería Informática de la Universidad de Cantabria y de Arquitectura e Ingeniería de Computadores de cuarto de la carrera de Ingeniería Informática de la Universidad Rey Juan Carlos.

Cada capítulo corresponde a un bloque temático dentro de estas asignaturas. El primer capítulo presenta una serie de ejercicios relacionados con análisis de rendimiento en sistemas procesadores de propósito general. El segundo capítulo introduce el concepto de segmentación y segmentación avanzada, incluyendo operaciones de punto flotante en el cauce. Los problemas relacionados con planificación dinámica de instrucciones y predicción de saltos se pueden encontrar en el capítulo 3. En el capítulo 4 se resuelven una colección de problemas relacionados con la jerarquía de memoria. Finalmente el capítulo 5 presenta y resuelve una selección de ejercicios sobre dispositivos de E/S y buses.

Los autores desean que este libro sea de utilidad para el estudio de dichas asignaturas y quedan a disposición de los alumnos para cualquier comentario o corrección que deseen enviar.

Los autores

Móstoles, 1 de Agosto de 2007

Capítulo 2

Análisis de rendimiento

2.1. Problema 1

En un determinado procesador las instrucciones enteras se ejecutan en 1 ciclo de reloj, mientras que las de coma flotante necesitan 5 ciclos de reloj para completarse. Los programas que se van a ejecutar en este procesador incluyen un 20 % de operaciones en coma flotante. Desde el punto de vista coste/prestaciones ¿sería interesante rediseñar el procesador para que las operaciones de coma flotante sean 5 veces más rápidas a costa de duplicar el coste total del procesador?

Solución:

Para decidir si vale la pena implementar la mejora calcularemos el speedup obtenido:

$$S = \frac{t_{anterior}}{t_{nuevo}} = \frac{I * CPI_{anterior} * T}{I * CPI_{nuevo} * T} = \frac{CPI_{anterior}}{CPI_{nuevo}} = \frac{0,8 * 1 + 0,2 * 5}{0,8 * 1 + 0,2 * 1} = 1,8$$

El speedup es 1.8 con lo cual elevar al doble el coste del procesador no es rentable.

2.2. Problema 2

El coprocesador de un computador mejora en un factor 4 el procesamiento de números en coma flotante. El tiempo de ejecución de un determinado procesador es de 1 minuto cuando se utiliza el coprocesador y de 2 minutos y medio cuando no se utiliza. Calcular cuál es el porcentaje del tiempo de ejecución que el programa pasa realizando operaciones en coma flotante cuando no hay coprocesador.

Solución:

Este ejercicio se soluciona aplicando la ley de Amdahl.

$$S = \frac{1}{(1 - F) + \frac{F}{G}}$$

Dado el speedup y la ganancia podemos calcular la fracción de tiempo que se aplica dicha mejora de la siguiente forma:

$$S = \frac{2,5}{1} = \frac{1}{(1-F) + \frac{F}{4}}$$

Despejando F, obtenemos que la fracción de tiempo que se utiliza la mejora es de 0.8

2.3. Problema 3

En el análisis teórico del efecto de una cierta mejora en un sistema, se obtiene que la aceleración global del mismo vale -2. Indicar brevemente cuáles pueden ser los posibles significados de este resultado.

Solución:

Este resultado implica necesariamente un error de cálculo, no una reducción en el rendimiento.

Utilizando la ley de Amdhal:

$$S = \frac{1}{(1-F) + \frac{F}{G}} = -2$$

Es decir que:

$$(1-F) + \frac{F}{G} < 0$$

Tanto F como G son siempre positivas, por tanto una aceleración negativa sólo puede conseguirse si $(1-F) < 0$, es decir cuando la fracción de tiempo en el que se debe aplicar la mejora sea mayor del 100 %, lo cual es imposible.

2.4. Problema 4

Supóngase una máquina en donde el CPI promedio es de 4 ciclos. Un 10 % son multiplicaciones y tardan 10 ciclos. Si sólo se mejoran estas operaciones en un factor 2.5, calcular el nuevo CPI promedio.

Solución:

$$CPI_{antes} = 0,9 * N + 0,1 * 10 = 4$$

Siendo N el CPI de las otras instrucciones, podemos despejar $0,9 * N = 3,1$

$$CPI_{despues} = 0,9 * N + 0,1 * \frac{10}{2,5} = 3,4$$

2.5. Problema 5

En un procesador se ejecuta una aplicación que consume un D % de su tiempo realizando accesos a disco y un F % de su tiempo ejecutando instrucciones de coma flotante. El resto del tiempo se ejecutan instrucciones enteras, que tienen un CPI de 1.5. Se dispone de un coprocesador para cálculos en coma flotante que alcanza un CPI de 5. Se evalúan dos posibles modificaciones para mejorar el rendimiento del sistema:

1. Sustituir el disco por otro el doble de rápido. El speedup obtenido es de un 17.6 %.
 2. Sustituir el disco por otro un 25 % más rápido y el coprocesador por uno con CPI de 2.5. El speedup obtenido es de 1.351.
- Calcular los porcentajes de tiempo D y F.
 - Calcular el porcentaje de instrucciones de coma flotante ejecutadas por el programa.

Solución:

Para calcular D aplicaremos directamente la ley de Amdhal:

$$S = 1,176 = \frac{1}{(1-D) + \frac{D}{2}}$$

Despejando D obtenemos que el porcentaje de tiempo que se realizan accesos al disco es del 30 %.

Para calcular la fracción de tiempo empleada en ejecutar instrucciones de punto flotante, utilizaremos la siguiente expresión

$$S = 1,351 = \frac{t_{anterior}}{t_{nuevo}} = \frac{D * t_{anterior} + F * t_{anterior} + E * t_{anterior}}{\frac{D * t_{anterior}}{1,25} + \frac{F * t_{anterior}}{5/2,5} + E * t_{anterior}}$$

Conociendo que $D=0.3$ y que la suma de los porcentajes de tiempo $D+F+E=1$, obtenemos que $E=1-D-F=0.7-F$. Sustituyendo todos los valores en la ecuación, obtenemos que $F=0.4$ y por lo tanto $E=0.3$.

Para calcular el porcentaje de instrucciones de coma flotante ejecutadas, primero calcularemos que porcentaje de tiempo del que se están ejecutando instrucciones se ejecutan instrucciones de coma flotante.

Como el 30 % del tiempo se está accediendo al disco duro el 70 % se están ejecutando instrucciones. Si sobre el 100 % del tiempo el 40 % se están ejecutando instrucciones de punto flotante, sobre el 70 %, podemos hacer una regla de tres que nos da que un 57 % del tiempo que ejecuta instrucciones, se están ejecutando instrucciones de punto flotante.

Por lo tanto el tiempo que se ejecutan instrucciones de punto flotante partido por el tiempo total que se ejecutan instrucciones es 0.57

$$0,57 = \frac{I_f * CPI_f}{I_f * CPI_f + I_e * CPI_e}$$

Donde el total de instrucciones de enteros es igual al número total de instrucciones menos las instrucciones de punto flotante.

$$I_e = I - I_f$$

Sustituyendo y despejando:

$$0,57 = \frac{5 * I_f}{3,5 * I_f + 1,5 * I}$$

$$\frac{I_f}{I} = 0,28$$

Por lo tanto el programa tiene un 28 % de instrucciones de punto flotante y un 72 % de instrucciones de enteros.

2.6. Problema 6

Un computador tiene un CPI de 4 para las instrucciones de coma flotante, excepto para las raíces cuadradas que el CPI es de 20. El resto de instrucciones tienen un CPI de 1.33. Un determinado benchmark tiene un 25 % de operaciones en coma flotante (sin incluir a la raíz cuadrada) y un 2 % de raíces cuadradas. Se proponen dos mejoras que suponen el mismo coste:

1. Modificar la organización de la arquitectura para aumentar la frecuencia de reloj un 15 % a costa de aumentar a 23 el CPI de las raíces cuadradas.
2. Mejorar el hardware que realiza las operaciones en coma flotante (sin incluir a la raíz cuadrada) para reducir su CPI a 2.

¿Qué mejora es la más acertada?

Solución:

En este ejercicio se nos da el porcentaje de instrucciones de cada tipo que contiene el programa, no el porcentaje de tiempo que se utiliza la mejora, por lo tanto no podemos utilizar la ley de Amdahl. En su lugar calcularemos el speedup como el cociente de los tiempos de ejecución con y sin la mejora.

$$S = \frac{t_{anterior}}{t_{nuevo}} = \frac{I * CPI_{anterior} * T}{I * CPI_{nuevo} * T}$$

La primera modificación mejora la frecuencia de reloj a costa de empeorar el CPI de las raíces cuadradas.

$$S = \frac{(0,25 * 4 + 0,02 * 20 + 0,73 * 1,33) * T}{(0,25 * 4 + 0,02 * 23 + 0,73 * 1,33) * \frac{T}{1,15}} = 1,12$$

La segunda reduce el CPI de las operaciones de punto flotante a 2.

$$S = \frac{0,25 * 4 + 0,02 * 20 + 0,73 * 1,33}{0,25 * 2 + 0,02 * 20 + 0,73 * 1,33} = 1,26$$

Por tanto la mejora mas acertada sería la segunda.

2.7. Problema 7

Una empresa que realiza software para control en tiempo real quiere mejorar las prestaciones de uno de sus clientes. El programa en cuestión consume 2 segundos en un procesador convencional. El 16 % de dicho programa es paralelizable sobre 4 procesadores y el 80 % es totalmente paralelizable. Este programa debe controlar un proceso cada 0,2 segundos. Calcular el mínimo número de procesadores necesarios para alcanzar el rendimiento deseado.

Solución:

Del enunciado deducimos que podemos acelerar el 16 % del programa en un factor de 4 y el 80 % en un factor n, donde n es el número de procesadores que debemos utilizar.

Si deseamos ejecutar el programa en 0.2 segundos el speedup necesario es de $2/0.2=10$ por lo tanto aplicando la ley de Amdahl:

$$S = 10 = \frac{1}{(1 - 0,16 - 0,8) + \frac{0,16}{4} + \frac{0,8}{n}}$$

De ahí deducimos que $n=40$ procesadores.

2.8. Problema 8

Nos encargan realizar la paralelización de un software de tratamiento de ficheros. Para ello nos dicen que el 20 % del tiempo se consume en leer ficheros, el 5 % del tiempo en escribir ficheros y el resto del tiempo se consume realizando cálculos, de los cuales el 80 % es paralelizable. ¿Cuál es el speedup máximo que podemos prometer?

Solución:

Para calcular el speedup máximo aplicaremos la ley de Amdahl suponiendo que podemos obtener una ganancia infinita en la parte paralelizable. Para ello calcularemos la fracción de tiempo que se puede utilizar la mejora, teniendo en cuenta que el 75 % del tiempo se realizan cálculos y que de ese 75 % el 80 % es paralelizable. Por tanto la mejora se puede aplicar el 80 % del 75 %, esto es el 60 %.

$$S = \frac{1}{(1 - 0,6) + \frac{0,6}{\infty}} = 2,5$$

Por tanto en este programa se puede obtener un speedup máximo de 2.5.

2.9. Problema 9

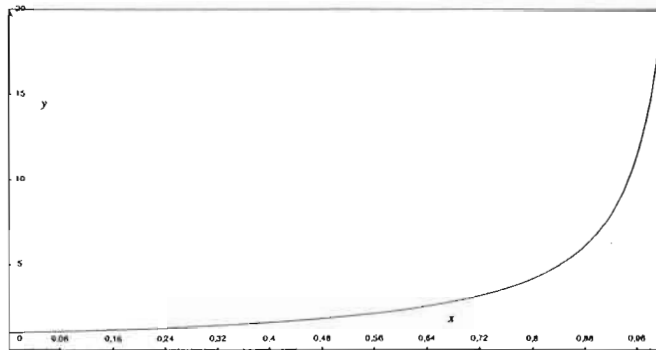
Supongamos que se está considerando mejorar una máquina añadiéndole un modo vectorial. La ejecución de un cálculo utilizando el modo vectorial es 20 veces más rápida que en el modo normal de ejecución.

1. Dibujar un gráfico donde se muestre la evolución del speedup obtenido con la mejora frente al porcentaje de tiempo que puede emplearse el modo vectorial.
2. ¿Qué porcentaje de vectorización se necesita para conseguir una aceleración de 2?
3. ¿Qué porcentaje de vectorización se necesita para conseguir la mitad de la aceleración máxima que se alcanza utilizando el modo vectorial?

Solución:

Para calcular el speedup utilizamos la ley de Amdhal:

$$S = \frac{1}{(1 - F) + \frac{F}{20}} = \frac{1}{1 - 0,95F}$$



En la gráfica podemos observar que cuando la fracción de tiempo que utilizamos la mejora es 0, es decir que no la utilizamos, el speedup obtenido es 1, tardamos lo mismo que sin la mejora. De igual forma podemos observar que cuando utilizamos la mejora el 100 % del tiempo obtenemos un speedup de 20, ya que no podemos mejorar más globalmente que la mejora individual.

En el segundo apartado se nos pide calcular el tiempo que tenemos que aplicar la mejora para obtener un speedup de 2. Utilizando la ley de Amdhal:

$$S = 2 = \frac{1}{(1 - F) + \frac{F}{20}}$$

Despejando F obtenemos que debemos aplicar la mejora un 52 % del tiempo.

Por último nos piden hallar la fracción de tiempo para conseguir la mitad del speedup máximo. Como el speedup máximo que podemos conseguir es de 20, lo que nos piden es obtener la fracción de tiempo que debemos utilizar la mejora para obtener un speedup de 10. Sustituyendo en la ecuación anterior obtenemos que $F=0.94$.

2.10. Problema 10

Se proponen tres mejoras para una nueva arquitectura, con ganancias $G1=30$, $G2=20$ y $G3=15$, respectivamente. Estas mejoras no se pueden utilizar simultáneamente.

1. Si las mejoras 1 y 2 se pueden utilizar cada una el 25 % del tiempo, ¿qué fracción del tiempo se debe utilizar la mejora 3 para que el speedup total sea de 10?
2. Supongamos que para un determinado benchmark la fracción de uso de las mejoras es del 15 % para la 1 y la 2 y del 70 % para la 3. Si queremos maximizar el rendimiento del sistema y sólo podemos incluir una mejora ¿cuál deberíamos escoger?

Solución:

Para solucionar este ejercicio aplicaremos de nuevo la ley de Amdhal. En el primer apartado nos piden la fracción de tiempo que debemos aplicar la mejora 3 para obtener un speedup de 10, por tanto:

$$S = 10 = \frac{1}{(1 - 0,25 - 0,25 - F) + \frac{0,25}{30} + \frac{0,25}{20} + \frac{F}{15}}$$

Despejando F, obtenemos que la fracción de tiempo que debemos aplicar esta mejora es del 45 %.

En el apartado 2 debemos evaluar cada mejora por separado para decidir cual es la mejor. Para ello utilizaremos la ley de Amdhal y veremos cual de ellas por separado nos da un mayor speedup.

$$S_1 = \frac{1}{(1 - 0,15) + \frac{0,15}{30}} = 1,169$$

$$S_2 = \frac{1}{(1 - 0,15) + \frac{0,15}{20}} = 1,166$$

$$S_3 = \frac{1}{(1 - 0,7) + \frac{0,7}{15}} = 2,88$$

Con lo cual la mejora que ofrece mejor resultado es la tercera.

2.11. Problema 11

La Universidad acaba de adquirir un computador para ejecutar una aplicación de simulación el 20 % del tiempo y para gestionar una base de datos el resto del tiempo. La aplicación de simulación dedica el 25 % del tiempo a realizar operaciones en coma flotante y el 5 % del tiempo a realizar accesos a disco. La gestión de la base de datos realiza accesos a disco el 50 % del tiempo que se está ejecutando. Todas las instrucciones tienen un CPI=1 y para ejecutar una operación en coma flotante se necesitan 200 instrucciones enteras. Nos proponen las siguientes mejoras:

1. Adquirir un coprocesador que sólo necesite una instrucción para realizar una operación en coma flotante y que tenga para estas instrucciones un CPI=10.
2. Adquirir un nuevo disco duro con el doble de velocidad.

Si la Universidad sólo se puede permitir una de estas mejoras ¿cuál debería escoger?

Solución:

Para tomar la decisión calcularemos el speedup que obtenemos introduciendo cada una de las mejoras en cada una de las aplicaciones y ponderaremos la mejora en función del porcentaje de tiempo que se ejecuta cada aplicación.

Aplicación A

En el caso de la aplicación de simulación el 25 % del tiempo se realizan operaciones de punto flotante y el 5 % del tiempo accesos a disco.

Para calcular la ganancia sabemos que sin coprocesador se necesitan 200 instrucciones enteras con un CPI de 1 cada una de ellas, por lo tanto las instrucciones de coma flotante sin coprocesador tienen un CPI de 200. Con el coprocesador se necesita una sola instrucción con CPI=10, de esta forma podemos calcular la ganancia que tiene un valor de $200/10=20$

$$S_{\text{coprocesador A}} = \frac{1}{(1 - 0,25) + \frac{0,25}{20}} = 1,31$$

En el caso de introducir un disco con el doble de velocidad, la mejora podemos aplicarla el 5 % del tiempo y obtenemos una ganancia de 2.

$$S_{\text{disco A}} = \frac{1}{(1 - 0,05) + \frac{0,05}{2}} = 1,025$$

Aplicación B

En el caso de la aplicación de simulación el 50 % del tiempo se accede al disco, y no se nos da información de que porcentaje de tiempo se ejecutan instrucciones de punto flotante así que asumiremos que dicha aplicación consta únicamente de operaciones con enteros. De esta forma:

$$S_{\text{disco B}} = \frac{1}{(1 - 0,5) + \frac{0,5}{2}} = 1,33$$

Para decidir cual de las mejoras adquirir, evaluaremos la mejora total obtenida con cada una de ellas, sabiendo que la aplicación A se ejecuta el 20 % del tiempo y la B el 80 %.

Con el coprocesador de coma flotante la mejora que obtenemos es:

$$S_{\text{coprocesador}} = 0,2 * S_{\text{coproccsador A}} + 0,8 * S_{\text{coprocesador B}} = 0,2 * 1,31 + 0,8 * 1 = 1,06$$

Con el disco el doble de rápido obtenemos:

$$S_{\text{disco}} = 0,2 * S_{\text{disco A}} + 0,8 * S_{\text{disco B}} = 0,2 * 1,025 + 0,8 * 1,33 = 1,269$$

Por lo tanto optaremos por adquirir un disco duro más rápido

2.12. Problema 12

Se quiere mejorar el rendimiento de un programa de simulación dinámica de fluidos. El programa invierte un 30 % del tiempo en realizar lecturas de fichero y un 5 % en escrituras. El resto del tiempo se emplea en realizar operaciones en coma flotante de las cuales el 15 % es paralelizable en 2 procesadores y el 65 % es paralelizable en N procesadores. Para la mejora del rendimiento se proponen dos posibilidades:

1. Cambiar el subsistema de I/O sustituyendo los discos IDE actuales por discos SCSI con un rendimiento 7 veces superior.
 2. Realizar una implementación paralela de la aplicación sobre un sistema multiprocesador con 4 procesadores.
- Determinar cual de las dos alternativas ofrece una mayor mejora (speedup).

Solución:

Para resolver el ejercicio evaluaremos cada una de las alternativas.

La alternativa 1 consiste en cambiar los discos duros, ejecutando el programa sobre un sistema monoprocesador. Del enunciado obtenemos que la aplicación está el 35 % del tiempo accediendo al disco duro, y además sabemos que la ganancia de cambiar el disco es de 7, por lo tanto:

$$S_1 = \frac{1}{(1 - 0,35) + \frac{0,35}{7}} = 1,42$$

La mejora 2 consiste en implementar el programa en 4 procesadores. Como sabemos que el programa invierte el 35 % del tiempo en acceder al disco sabemos que el 65 % del tiempo está ejecutando instrucciones. De ese 65 % restante el 65 % es paralelizable en N procesadores mientras que el 15 % lo es en 2 y el resto no lo es.

Por lo tanto el $65\% * 65\% = 42\%$ del tiempo que se ejecuta el programa puede ser paralelizado sobre N procesadores y el $65\% * 15\% = 9\%$ puede ser paralelizado sobre 2.

Aplicando la ley de Amdhal:

$$S_2 = \frac{1}{(1 - 0,42 - 0,09) + \frac{0,09}{N} + \frac{0,42}{3 - N}}$$

Por lo tanto debemos buscar la asignación óptima de procesadores que nos den el mayor speedup. Suponiendo que la parte no paralelizable debe ejecutarse en un procesador, y las otras partes deben ejecutarse también en al menos un procesador, el speedup máximo se obtendrá cuando $N=1$, siendo su valor 1.38.

Por lo tanto la mejor alternativa es cambiar el subsistema de I/O.

2.13. Problema 13

Tenemos una arquitectura de carga/almacenamiento con los siguientes porcentajes:

Operación	% Utilización	CPI
ALU	43	1
LOAD	21	2
STORE	12	2
Salto	24	2

El 25% de las operaciones de ALU tienen un operando que no se reutiliza. Se propone añadir al repertorio de esta arquitectura instrucciones LOAD/ALU que tengan un operando fuente en memoria y $CPI=2$. ¿Sería rentable si a cambio se incrementa el CPI de las instrucciones de salto hasta 3?

Solución:

Al introducir esta nueva instrucción varían los porcentajes de utilización de las operaciones, así como sus CPI. El 25% de las operaciones de la ALU van a convertirse en operaciones del tipo LOAD/ALU, por lo tanto la nueva tabla de operaciones con sus nuevos porcentajes es:

Operación	% Utilización	CPI
ALU	$43 \cdot 0.75 = 32.25$	1
LOAD	21	2
STORE	12	2
Salto	24	3
LOAD/ALU	$43 \cdot 0.25 = 10.75$	2

Para comprobar si la modificación es rentable calculamos el speedup como el cociente entre el CPI antes y después:

$$S = \frac{0,43 \cdot 1 + 0,21 \cdot 2 + 0,12 \cdot 2 + 0,24 \cdot 2}{0,3225 \cdot 1 + 0,21 \cdot 2 + 0,12 \cdot 2 + 0,24 \cdot 3 + 0,1075 \cdot 2} = 0,82$$

Por lo tanto perdemos rendimiento y la mejora no es acertada.

2.14. Problema 14

Considerar dos implementaciones diferentes P1 y P2 del mismo repertorio de instrucciones. Hay cinco clases de instrucciones con el CPI dado en la tabla adjunta. P1 tiene una frecuencia de reloj de 4 Ghz y P2 de 6 Ghz. Se define el rendimiento de pico como la velocidad más rápida a la que cualquiera de los computadores puede ejecutar cualquier secuencia de instrucciones. ¿Cuál es el rendimiento de pico de P1 y P2 expresado en instrucciones por segundo? Si el número de instrucciones de un programa se divide de forma equivalente entre todas las clases salvo la clase A que ocurre el doble que las demás, ¿Cuánto más rápido es P2 que P1?

Instrucciones	P1	P2
A	1	2
B	2	2
C	3	2
D	4	4
E	3	4

Solución:

El rendimiento pico es el número de instrucciones por segundo de la secuencia más rápida que puede ejecutar el procesador.

Para el procesador P1 la secuencia más rápida que puede ejecutar es una compuesta por todas las instrucciones tipo A. En ese caso:

$$Rendimiento_A = \frac{f}{CPI_{secuencia}} = \frac{4 \cdot 10^9}{1} = 4 \cdot 10^9 \frac{ins}{s}$$

Para el procesador P2 la secuencia más rápida está compuesta por instrucciones del tipo A, B y C, que todas tardan 2 ciclos:

$$Rendimiento_B = \frac{f}{CPI_{secuencia}} = \frac{6 \cdot 10^9}{2} = 3 \cdot 10^9 \frac{ins}{s}$$

Por lo tanto el procesador P1 tiene un mayor rendimiento de pico.

Para calcular la segunda cuestión, podemos generar una secuencia de instrucciones tal que siempre se ejecuta AABCDE, de forma repetitiva N veces. Esta secuencia divide la ejecución de forma equivalente entre todas las instrucciones, salvo la clase A que se ejecuta el doble.

Para esta secuencia de instrucciones el CPI de cada procesador será:

$$CPI_1 = \frac{ciclos}{instrucciones} = \frac{N \cdot (2 \cdot 1 + 1 \cdot 2 + 1 \cdot 3 + 1 \cdot 4 + 1 \cdot 3)}{N \cdot 6} = 2,33$$

$$CPI_2 = \frac{ciclos}{instrucciones} = \frac{N \cdot (2 \cdot 2 + 1 \cdot 2 + 1 \cdot 2 + 1 \cdot 4 + 1 \cdot 4)}{N \cdot 6} = 2,66$$

Por tanto para esta secuencia el rendimiento de pico será:

$$\text{Rendimiento}_A = \frac{f}{CPI_{\text{secuencia}}} = \frac{4 * 10^9}{2,33} = 1,71 * 10^9 \frac{\text{ins}}{\text{s}}$$

$$\text{Rendimiento}_B = \frac{f}{CPI_{\text{secuencia}}} = \frac{6 * 10^9}{2,66} = 2,25 * 10^9 \frac{\text{ins}}{\text{s}}$$

Por lo tanto el procesador P2 es más rápido para esa secuencia.

2.15. Problema 15

En la tabla adjunta se muestra el tiempo de ejecución de tres sistemas ejecutando cinco benchmarks. Comparar el rendimiento relativo de cada uno de estos sistemas utilizando la media aritmética y la geométrica. Analizar los resultados y explicarlos.

Programa	Tiempo Sistema A	Tiempo Sistema B	Tiempo Sistema C
V	150	200	75
W	200	250	150
X	250	175	200
Y	400	800	500
Z	1000	1200	1100

Solución:

Para comparar el rendimiento de estos procesadores calcularemos la media aritmética y geométrica de los tiempos de ejecución de los programas del benchmark.

Media	Sistema A	Sistema B	Sistema C
Aritmética	400	525	405
Geométrica	312.9	384.4	262.1

Donde la media aritmética de A se ha calculado como:

$$\text{Media}_{\text{aritmética}} A = \frac{150 + 200 + 250 + 400 + 1000}{5} = 400$$

Y su media geométrica:

$$\text{Media}_{\text{geométrica}} A = \sqrt[5]{150 * 200 * 250 * 400 * 1000} = 312,9$$

Sabemos que para analizar rendimientos la media más adecuada es la aritmética. Al analizar dicha media comprobamos que el sistema A es el más rápido seguido por el C. La media geométrica nos da un resultado diferente. No obstante sabemos que a menudo se utiliza la media geométrica en lugar de la aritmética para evaluar el rendimiento de los computadores debido a que si normalizamos los datos, la media geométrica es independiente de la máquina sobre la que normalicemos.

Capítulo 3

Segmentación

3.1. Problema 1

Dado un procesador no segmentado con un tiempo de ciclo de 10 ns y registros intermedios con un retardo de 0,5 ns.

1. ¿Cuáles son los tiempos de ciclo de las versiones segmentadas del procesador con 2, 4, 8 y 16 etapas si la lógica del camino de datos se divide equitativamente entre las etapas del cauce?
2. En el procesador del último ejercicio, ¿cuántas etapas de segmentación son necesarias para lograr un tiempo de ciclo de 2ns? ¿Y para 1 ns?
3. Para el mismo procesador y los mismos retardos de los registros intermedios del apartado 1, ¿cuál es el tiempo de ciclo mínimo posible con un cauce de 4 etapas si se transfiere lógica a la etapa final para equilibrar el retardo de los registros intermedios del resto de etapas?

Solución:

1. Se sabe que el tiempo de ciclo viene dado por la expresión: $T_c = \frac{T_{\text{enosegm.}}}{N_{\text{etapas}}} + \text{retardo}$.
En los casos 2,4,8,16 obtenemos los tiempos siguientes:

Nº Etapas	T. ciclo
2	$10/2 + 0,5 = 5,5\text{ns}$
4	$10/4 + 0,5 = 3\text{ns}$
8	$10/8 + 0,5 = 1,75\text{ns}$
16	$10/16 + 0,5 = 1,125\text{ns}$

2. Aplicando la misma expresión del apartado anterior, podemos obtener el número de etapas necesario:

- a) Para 2ns. $2 = \frac{10}{N} + 0,5 \rightarrow N = 6,67$, redondeando a entero superior, obtenemos 7 etapas

b) Para $1n.s. 1 = \frac{10}{N} + 0,5 \rightarrow N = 20$ etapas.

3. El retardo total en este caso, vendría dado por la distribución de los tiempos de retardo a lo largo de las etapas. Partiendo de esa idea, se puede deducir que en este caso el retardo total vendrá dado por: $10 + 3 * 0,5 = 11,5$ (Es el tiempo del procesador no segmentado en este caso), y el tiempo de ciclo final en un procesador de 4 etapas será: $\frac{11,5}{4} = 1,875ns$

3.2. Problema 2

Suponga que un procesador no segmentado tiene un tiempo de ciclo de 25 ns, y que el camino de datos está constituido por módulos con latencias de 2, 3, 4, 7, 3, 2, y 4 ns (en ese orden). Al segmentar este procesador, no es posible cambiar el orden de los módulos (por ejemplo, colocando la etapa de lectura de registros antes de la etapa de decodificación) ni dividir un módulo en varias etapas (por razones de complejidad). Suponiendo registros intermedios con 1 ns de retardo:

1. ¿Cuál es el tiempo de ciclo mínimo que puede alcanzarse segmentando el procesador?
2. Si el procesador se divide en el número menor de etapas que permita lograr la latencia mínima del apartado 1, ¿cuál es la latencia del cauce?
3. Si se limita el número de etapas del cauce a 2, ¿cuál es el tiempo de ciclo mínimo?
4. ¿Cuál es la latencia del cauce del apartado c?

Solución:

1. Si no existe límite para el número de etapas, la latencia viene marcada por el módulo más lento más el registro de segmentación, resultando en: $7ns + 1ns = 8ns$.
2. Se agrupan módulos adyacentes con una latencia total de 7ns o menos. De esta forma se obtienen 5 etapas, un ejemplo es:
(Módulos en función de los tiempos de retardo)

2,3	4	7	3	2,4
-----	---	---	---	-----

resultando ahora en una latencia de: $5 * 8 = 40ns$.

3. Limitando las etapas a 2, surgen 2 posibles combinaciones:

2,3,4,7	3,2,4
2,3,4	7,3,2,4

- a) En total las dos etapas son, 16 y 9, resultando en un $T_c = 16 + 1 = 17ns$.
 - b) Este caso es simétrico al anterior, 9 y 16, con un T_c también de 17ns.
4. La latencia de ese cauce es: $16 * 2 = 32ns$.

3.3. Problema 3

Considerar la siguiente secuencia de instrucciones para el procesador MIPS.

```
DIV R2, R5, R8
SUB R9, R2, R7
ASH R5, R14, R6
MUL R11, R9, R5
BEQ R10, #0, R12
OR R8, R15, R2
```

1. Identifique todos los riesgos RAW de la siguiente secuencia de instrucciones.
2. Identifique todos los riesgos WAR de la secuencia de instrucciones anterior.
3. Identifique todos los riesgos WAW de la secuencia de instrucciones.
4. Dibujar el diagrama de ejecución en el cauce segmentado del procesador MIPS.

Solución:

1. RIESGOS RAW (Lectura después de escritura):
 - a) DIV y SUB en R2
 - b) ASH y MUL en R5
 - c) SUB y MUL en R9
 - d) DIV y OR en R2
2. RIESGOS WAR (Escritura después de lectura):
 - a) DIV y ASH en R5
 - b) DIV y OR en R8
3. RIESGOS WAW (Escritura después de escritura): No hay.

Instrucción/ciclo	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
DIV R2, R5, R8	IF	ID	EX	ME	WB												
SUB R9, R2, R7		IF	ID	ID	ID	EX	ME	WB									
ASH R5, R14, R6			IF	IF	IF	ID	EX	ME	WB								
MUL R11, R9, R5						IF	ID	ID	ID	EX	ME	WB					
BEQ R10, #0, R12							IF	IF	IF	ID	EX	ME	WB				
OR R8, R15, R2													IF	ID	ME	EX	WB

3.4. Problema 4

Identificar todas las dependencias de datos en el siguiente código. ¿Cuáles de ellas se pueden resolver usando anticipación?

```
ADD R2, R5, R4
ADD R4, R2, R5
SW R5, 100(R2)
ADD R3, R2, R4
```

Solución:

▪ DEPENDENCIAS:

- ADD 1º y ADD 2º en R2
- SW y ADD 1º en R2
- ADD 3º y ADD 1º en R2
- ADD 3º y ADD 2º en R4

Todas pueden ser resueltas con anticipación.

3.5. Problema 5

Suponga que en el cauce segmentado en 5 etapas del procesador MIPS se ejecuta el siguiente fragmento de código. Suponiendo que no existe anticipación de resultados, dibujar un diagrama de ejecución temporal y calcular el número de ciclos que se emplearía para ejecutarlo.

```
ADD R1, R2, R3
SUB R4, R5, R6
MUL R8, R9, R10
DIV R12, R13, R14
```

Solución:

Instrucción/ciclo	1	2	3	4	5	6	7	8
ADD R1, R2, R3	IF	ID	EX	ME	WB			
SUB R4, R5, R6		IF	ID	EX	ME	WB		
MUL R8, R9, R10			IF	ID	EX	ME	WB	
DIV R12, R13, R14				IF	ID	EX	ME	WB

El código tarda 8 ciclos en ejecutarse.

3.6. Problema 6

Suponga que en el cauce segmentado en 5 etapas del procesador MIPS se ejecuta el siguiente fragmento de código. Suponiendo que no existe anticipación de resultados dibujar un diagrama de ejecución temporal y calcular el número de ciclos que se emplearía para ejecutarlo. Comparar los resultados del ejercicio 4 y 5 y explicar a qué se deben las diferencias.

```
ADD R1, R2, R3
SUB R4, R5, R6
MUL R8, R9, R4
DIV R12, R13, R14
```

Solución:

Los únicos riesgos que pueden dar problemas en la arquitectura MIPS son los de tipo RAW de los cuales, en las instrucciones se identifica uno entre SUB y MUL en R4 que provoca una parada del cauce.

Instrucción/ciclo	1	2	3	4	5	6	7	8	9	10
ADD R1, R2, R3	IF	ID	EX	ME	WB					
SUB R4, R5, R6		IF	ID	EX	ME	WB				
MUL R8, R9, R4			IF	IF	IF	ID	EX	ME	WB	
DIV R12, R13, R14						IF	ID	EX	ME	WB

Tarda 10 ciclos.

3.7. Problema 7

Suponga que en el cauce segmentado en 5 etapas del procesador MIPS se ejecuta el siguiente fragmento de código. Suponiendo que no existe anticipación de resultados dibujar un diagrama de ejecución temporal y calcular el número de ciclos que se emplearía para ejecutarlo. Suponga que el salto representado por la instrucción BEQ no se produce.

```
ADD R1, R2, R3
SUB R4, R5, R6
BEQ R2, #0, R9
DIV R12, R13, R14
```

Solución:

El problema en este caso está localizado en la instrucción BEQ la cual generará un bloqueo hasta que se termine de ejecutar puesto que hasta esa etapa no se sabe la dirección efectiva de salto.

Instrucción/ciclo	1	2	3	4	5	6	7	8	9	10	11
ADD R1, R2, R3	IF	ID	EX	ME	WB						
SUB R4, R5, R6		IF	ID	EX	ME	WB					
BEQ R2, #0, R9			IF	ID	EX	ME	WB				
DIV R12, R13, R14				X	X	X	IF	ID	ME	EX	WB

Tarda 11 ciclos.

3.8. Problema 8

¿Cuál es el tiempo de ejecución (en ciclos) de la siguiente secuencia de instrucciones en el cauce segmentado del procesador MIPS (sin cortocircuitos)? Suponga que no se produce el salto. Si el procesador tiene un período de reloj de 2 ns, ¿cuál es el tiempo de ejecución en ns?

```
ADD R1, R4, R7
BEQ R2, #0, R1
SUB R8, R10, R11
MUL R12, R13, R14
```

Solución:

Para ejecutar BEQ necesitamos el resultado de R1 con lo cual BEQ estará en la etapa ID durante 3 ciclos, después SUB tendrá que esperar para saber si hay salto o no, entrando en el cauce en la etapa WB de BEQ resultando, en total, 4 ciclos perdidos más, finalmente MUL se ejecutará sin problemas tardando 1 ciclo más. Siendo el resultado final los 5 ciclos iniciales $+3+4+1=13$ ciclos que además dan un tiempo de ejecución de $13 \cdot 2\text{ns}=26\text{ns}$.

3.9. Problema 9

Para la siguiente secuencia de código ejecutado en el procesador MIPS segmentado sin anticipación, responder a las siguientes cuestiones.

1. ¿Cuál es el tiempo de ejecución (en ciclos) de la siguiente secuencia de instrucciones en el cauce segmentado del procesador MIPS (sin cortocircuitos)?
2. ¿Puede mejorarse el tiempo de ejecución de la secuencia de instrucciones reordenando las instrucciones sin cambiar el resultado del cálculo? Si es así, muestre la secuencia de instrucciones con el menor tiempo de ejecución e indique cuál es ese tiempo.

```
ADD R3, R4, R5
SUB R7, R3, R9
MUL R8, R9, R10
ASH R4, R8, R12
```

Solución:

1. La instrucción SUB tardará 3 ciclos mas en ejecutarse que la instrucción ADD debido a la dependencia que presentan en R3, MUL es independiente con lo que solo tardará 1 ciclo con respecto a SUB pero ASH presenta riesgo en R8 con MUL tardando 3 ciclos más que ésta última, resultando todo en un tiempo final de $5+3+1+3=12$ ciclos

2. Si existe una reordenación que nos lleve a un mejor tiempo, puesto que MUL es independiente de ADD y SUB podemos intercalarla entre ellas para así reducir el tiempo de parada de SUB y de ASH, quedando el código como sigue:

```
ADD R3, R4, R5
MUL R8, R9, R10
SUB R7, R3, R9
ASH R4, R8, R12
```

Con esta configuración la única parada en el cauce es de SUB que tiene que esperar 2 ciclos a que ADD termine, gracias a esto ASH no necesita esperar puesto que cuando empieza su etapa ID, MUL ya ha terminado. El tiempo total es $5+2+1+1=9$ ciclos.

3.10. Problema 10

Calcular el tiempo de ejecución de la secuencia de instrucciones siguiente en el procesador MIPS segmentado sin anticipación. A continuación, encontrar la reordenación de las instrucciones que proporcione el menor tiempo de ejecución posible y calcular dicho tiempo.

```
MUL R10, R11, R12
SUB R8, R10, R15
ADD R13, R14, r0
ASH R15, R2, R3
OR R7, R5, R6
```

Solución:

1. El tiempo de ejecución del código son 11 ciclos, se obtiene aplicando la expresión:

$$N_{\text{ciclos}} = N_{\text{etapas}} + (N_{\text{Inst}} - 1) + \text{CiclosBloqueo} = 5 + (5 - 1) + 1 \cdot 2 = 11.$$
2. La reordenación de código propuesta es la siguiente: MUL ADD OR SUB ASH, con esta combinación logramos reducir las dependencias sin que afecten al funcionamiento normal del programa.

3.11. Problema 11

Si el tiempo empleado por una operación de la ALU en el procesador MIPS fuera de 4 ns en lugar de 2 ns, ¿cómo se vería afectado el incremento de velocidad que se obtiene al segmentar una realización uniciclo?

Solución:

Si consideramos una operación ALU con un tiempo de 4ns. implica que en una implementación MIPS monociclo tendremos un tiempo de ciclo total de 10 ns.

Si consideramos que la ALU es la etapa mas lenta, al realizar una implementación segmentada tendremos un tiempo de ciclo de 4ns. con lo cual el speedup conseguido sería de $10/4\text{ns}$.

3.12. Problema 12

Mostrar los caminos de anticipación que se necesitan para ejecutar las siguientes instrucciones:

```
ADD R2, R3, R4
ADD R4, R5, R6
ADD R5, R3, R4
```

Solución:

Suponiendo una arquitectura MIPS, simplemente bastaría con un adelantamiento de EX/MEM R4 a la entrada de la ALU.

3.13. Problema 13

Etiquetar cada registro de segmentación del cauce del MIPS con el nombre del valor que está guardado en él. Determinar la longitud en bits de cada campo.

Solución:

1. Registro IF/ID:

- a) PC+4=32 bits
- b) Instrucción=32 bits

Total = 32bits.

2. Registro ID/EX:

- a) PC+4=32 bits
- b) Datos Instrucción= 32+32=64 bits
- c) Extensión de Inmediato=32 bits
- d) 2 posibles registros destino=5+5=10bits

Total=138 bits.

3. Registro EX/MEM

- a) PC por si hay salto=32 bits
- b) Zero flag = 1 bit
- c) Dato 2= 32 bits
- d) Registro destino=5 bits

Total=102 bits.

4. Registro MEM/WB

- a) Datos leídos de memoria= 32 bits
- b) Resultado ALU= 32 bits
- c) Registro de destino= 5 bits

Total=69 bits.

3.14. Problema 14

Considerar la ejecución del siguiente código en el camino de datos segmentado. Al final de la ejecución del 5º ciclo

1. ¿Qué registros se han leído y cuáles se escribirán?
2. Explicar qué hace la unidad de anticipación en el 5º ciclo.
3. Explicar que está haciendo la unidad de detección de riesgos en el 5º ciclo.

```
ADD R1, R2, R3
ADD R4, R5, R6
ADD R7, R8, R9
ADD R10, R11, R12
ADD R113, R14, R15
```

Solución:

1. En el ciclo 5º de ejecución se habrá escrito el registro R1 y se habrán leído los registros R11 y R12.
2. La unidad de adelantamiento examina si hay dependencias para ver si necesita hacer algún adelantamiento. En concreto trabaja con las instrucciones que están en la etapa EX, MEM y WB (las 3 primeras), comprobando si las 2 primeras están intentado escribir algún registro que use la tercera. Las comparaciones que hará son:

- a) R8=R1 ó R9=R1
- b) R8=R4 ó R9=R4

3. La unidad de detección de riesgos comprueba si la instrucción de la ALU (etapa EXE) y si la instrucción en ID lee el mismo registro que el LOAD va a escribir. Si esto es así se debe introducir un bloqueo. Chequeará la señal "MemRead" de la 3ª instrucción y los registros R4 y R12.

3.15. Problema 15

Considere la secuencia de instrucciones empleada para realizar una copia de memoria a memoria:

```
LW R2, 100(R5)
SW R2, 200(R6)
```

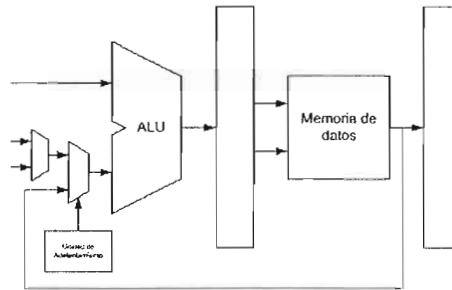

Un hardware de anticipación adicional puede mejorar el rendimiento. Mostrar los cambios necesarios en el camino de datos para permitir que un código como el anterior se ejecute sin bloqueos. Incluir las ecuaciones de anticipación para cada una de las señales de control de cada multiplexor, tanto nuevo como modificado, en este camino de datos.

Solución:

Hay dos soluciones posibles:

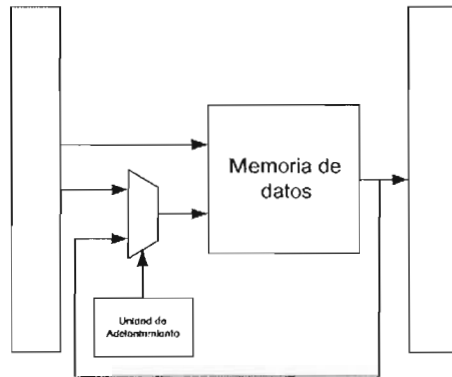
Examinar las instrucciones LW-SW cuando LW está en MEM y SW en EXE. Las condiciones que se deben cumplir es:

```
if(ID/EX.MEMwrite and EX/MEM.Memread and
(ID/EX.RegisterRt=EX/MEM.RegisterRd) and (EX/MEM.RegisterRd!=0))
then MUX=1, else MUX=0.
```



También se puede plantear la misma solución cuando lw está en WB y sw en MEM, la condición es:

```
if(EX/MEM.write and (MEM/WB.memtoReg=1) and MEM/WB.RegWrite and
(EX/MEM.registerRd!=MEM/WB.RegisterRd) and (MEM/WB.registerRd!=0))
then Mux=1, else Mux=0.
```



3.16. Problema 16

En un procesador MIPS se elimina la capacidad para especificar un desplazamiento en las instrucciones de acceso a memoria. ¿Cómo afectará esta modificación en la arquitectura del repertorio de instrucciones en la realización segmentada del procesador? Describir los cambios necesarios en el camino de datos y cómo afectará esto al rendimiento.

Solución:

1. La dirección de memoria en este caso viene de la propia instrucción y nunca de la ALU. Esto significa que el acceso a memoria se puede hacer en la propia etapa EX, ya que no se necesita esperar a la salida de la ALU. Por lo tanto, se puede eliminar la etapa MEM completamente.
2. El número de instrucciones de un programa se incrementará ya que se necesitarán instrucciones adicionales de suma para calcular las direcciones de memoria.
3. El CPI disminuirá ya que no habrá bloqueos en el procesador con un LOAD seguido de una instrucción que use el dato obtenido. Los adelantamientos pueden resolver este problema como si se tratara de una instrucción ALU.

3.17. Problema 17

Se piensa añadir una instrucción ADDM que permita que una instrucción aritmética acceda directamente a memoria. Explicar qué problemas habría en el cauce del MIPS para añadir esta instrucción.

Solución:

La instrucción ADDM necesita 6 etapas:

1. Búsqueda.
2. Decodificación.
3. Cálculo de la dirección de memoria del operando.
4. Lectura de memoria.
5. Suma.
6. Escritura en registros.

Esto implica una etapa extra para todas las instrucciones, incrementando la latencia de una instrucción, pero no el rendimiento de la CPU (1 instrucción resuelta por ciclo). Hay más probabilidad de riesgos y penalizaciones mayores para esos riesgos que disminuirán el rendimiento. Mayor coste en las nuevas etapas, unidad de detección y unidad de adelantamiento.

3.18. Problema 18

El siguiente fragmento de código se ejecuta en un MIPS con segmentación:

```
SUB R1,R2,R3
ADD R4,R5,R6
SUB R5,R4,R8
ADD R7,R2,R3
ADD R9,R7,R3
LW R1,10(R6)
ADD R3,R1,R4
SUB R6,R7,R8
```

Suponiendo que un dato se puede escribir en un banco de registros y leer su nuevo valor en el mismo ciclo:

1. Calcular el número de ciclos necesarios para ejecutar el código si no existe posibilidad de adelantar operandos ni de reordenar el código.
2. Calcular el número de ciclos necesarios si existe adelantamiento de operandos.
3. Si se permite reordenar el código para que el número de ciclos sea mínimo, ¿cuántos ciclos son necesarios en este caso?

Solución:

Apartado 1:

Instrucción	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
SUB R1,R2,R3	IF	ID	EX	ME	WB													
ADD R4,R5,R6		IF	ID	EX	ME	WB												
SUB R5,R4,R8			IF	ID	EX	ME	WB											
ADD R7,R2,R3				IF	ID	EX	ME	WB										
ADD R9,R7,R3					IF	ID	EX	ME	WB									
LW R1,10(R6)						IF	ID	EX	ME	WB								
ADD R3,R1,R4							IF	ID	EX	ME	WB							
SUB R6,R7,R8								IF	ID	EX	ME	WB						

Podemos calcular el número de ciclos que tarda el código en ejecutarse de varias formas:

1. Contando los ciclos sobre el diagrama: 18 ciclos
2. Calculándolo a partir de los datos de los que disponemos: 4 ciclos de llenado + 8 instrucciones + 3 dependencias RAW*2 ciclos de parada cada una = 18 ciclos
3. También podemos sumar el número de ciclos de parada contando el número de ciclos en que la etapa de WB no se ejecuta: 4 ciclos de llenado + 8 instrucciones + 6 ciclos en que la etapa de WB no se ejecuta = 18 ciclos

Apartado 2:

Instrucción	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SUB R1,R2,R3	IF	ID	EX	ME	WB										
ADD R4,R5,R6		IF	ID	EX	ME	WB									
SUB R5,R4,R8			IF	ID	EX	ME	WB								
ADD R7,R2,R3				IF	ID	EX	ME	WB							
ADD R9,R7,R3					IF	ID	EX	ME	WB						
LW R1,10(R6)						IF	ID	EX	ME	WB					
ADD R3,R1,R4							IF	ID	EX	ME	WB				
SUB R6,R7,R8								IF	ID	EX	ME	WB			

En este caso podemos calcular el tiempo de ejecución de la misma forma:

1. Contando los ciclos sobre el diagrama: 13 ciclos
2. Calculándolo a partir de los datos de los que disponemos: 4 ciclos de llenado + 8 instrucciones + 1 dependencia RAW*1 ciclos de parada cada una = 13 ciclos
3. También podemos sumar el número de ciclos de parada contando el número de ciclos en que la etapa de WB no se ejecuta: 4 ciclos de llenado + 8 instrucciones + 1 ciclo en que la etapa de WB no se ejecuta = 13 ciclos

Apartado 3:

Podemos intercambiar las dos últimas instrucciones. De esta forma la dependencia entre el LW y el ADD desaparece y el código tarda 12 ciclos en ejecutarse.

3.19. Problema 19

En un computador MIPS64 ejecutamos la siguiente secuencia de código.:

```
SUB R1,R2,R3
ADD R4,R5,R6
OR R5,R1,R4
SW R5,10(R6)
LW R1,20(R8)
AND R3,R8,R5
ADD R3,R1,R7
```

Considerando que el banco de registros permite leer los datos en el flanco de bajada del reloj y se escriben en el de subida:

1. Mostrar el diagrama temporal de ejecución del código cuando no se permite adelantamiento de operandos.
2. Mostrar este mismo diagrama si se permite adelantamiento.

El salto se repite 100 veces con lo cual tenemos 100 iteraciones. Ahora bien la primera y la última son diferentes ya que no se ejecuta el mismo número de instrucciones.

Para calcular el número de ciclos que se tarda en ejecutar el código distinguiremos 3 casos:

1. Primera iteración: 4 ciclos de llenado + 102 instrucciones antes del bucle + 106 instrucciones en el bucle + 3 dependencias RAW * 2 ciclos de parada + 1 ciclo de parada por el salto
2. 98 iteraciones: 106 instrucciones en el bucle + 3 dependencias RAW * 2 ciclos de parada + 1 parada por el salto
3. Última iteración: 106 instrucciones en el bucle + 1 instrucción después del bucle + 3 dependencias RAW * 2 ciclos de parada + 1 ciclo de parada por el salto

La suma de todos esos ciclos da un total de 11407 ciclos.

Apartado 2:

En este apartado rellenamos el hueco del salto retardado con la instrucción destino del salto y modificamos el destino del salto a 1004.

Instrucción	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OR R4,R8,R9															
100 ins															
ADD R5,R6,R7															
OR R4,R12,R6															
1 ins															
99 ins															
LW R10,20(R4)	IF	ID	EX	ME	WB										
ADD R8,R9,R10		IF	ID	ID	ID	EX	ME	WB							
SUB R6,R8,R1			IF	IF	IF	ID	ID	ID	EX	ME	WB				
OR R1,R3,R5						IF	IF	IF	ID	EX	ME	WB			
BEQ R1,R2,1000									IF	ID	ID	ID	EX	ME	WB
[OR R4,R12,R6]										IF	IF	IF	ID	EX	ME
ADD R1,R5,R6													IF	ID	EX

En este caso podemos calcular el tiempo de ejecución de la misma forma:

1. Primera iteración: 4 ciclos de llenado + 103 instrucciones antes del bucle + 106 instrucciones en el bucle + 3 dependencias RAW * 2 ciclos de parada
2. 98 iteraciones: 106 instrucciones en el bucle + 3 dependencias RAW * 2 ciclos de parada
3. Última iteración: 106 instrucciones en el bucle + 1 instrucción después del bucle + 3 dependencias RAW * 2 ciclos de parada

La suma de todos esos ciclos da un total de 11308 ciclos.

Apartado 3:

En este apartado nos permiten utilizar adelantamiento de operandos. Además al igual que en el apartado anterior rellenamos el hueco del salto retardado con la instrucción destino del salto y modificamos el destino del salto a 1004.

Instrucción	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OR R4,R8,R9															
100 ins															
ADD R5,R6,R7															
OR R4,R12,R6															
1 ins															
99 ins															
LW R10,20(R4)	IF	ID	EX	ME	WB										
ADD R8,R9,R10		IF	ID	ID	EX	ME	WB								
SUB R6,R8,R1			IF	IF	ID	EX	ME	WB							
OR R1,R3,R5					IF	ID	EX	ME	WB						
BEQ R1,R2,1000						IF	ID	ID	EX	ME	WB				
[OR R4,R12,R6]							IF	IF	ID	EX	ME	WB			
ADD R1,R5,R6									IF	ID	EX	ME	WB		

En este caso también podemos calcular el tiempo de ejecución de la misma forma:

1. Primera iteración: 4 ciclos de llenado + 103 instrucciones antes del bucle + 106 instrucciones en el bucle + 2 dependencias RAW * 1 ciclos de parada
2. 98 iteraciones: 106 instrucciones en el bucle + 2 dependencias RAW * 1 ciclos de parada
3. Última iteración: 106 instrucciones en el bucle + 1 instrucción después del bucle + 2 dependencias RAW * 1 ciclos de parada

La suma de todos esos ciclos da un total de 11104 ciclos.