

# ARQUITECTURA DE COMPUTADORES

## RELACIÓN DE EJERCICIOS 1

### Ejercicios

- ① En el código de prueba (benchmark) que ejecuta un procesador no segmentado que funciona a 300 MHz, hay un 20% de instrucciones LOAD que necesitan 4 ciclos, un 10% de instrucciones STORE que necesitan 3 ciclos, un 25% de instrucciones con operaciones de enteros que necesitan 6 ciclos, un 15% de instrucciones con operandos en coma flotante que necesitan 8 ciclos por instrucción, y un 30% de instrucciones de salto que necesitan 3 ciclos.

<u>Tipo i de instrucción</u>	<u>CPI<sub>i</sub></u>	<u>NI<sub>i</sub></u>
LOAD	4	0'20 NI
STORE	3	0'10 NI
ENTEROS	6	0'25 NI
FP	8	0'15 NI
BR	3	0'30 NI
		+ <u>NI</u>

$$T_{\text{sin\_mejoría}} = NI \cdot CPI \cdot T_{\text{ciclo}} = NI \cdot (0'20 \cdot 4 + 0'10 \cdot 3 + 0'25 \cdot 6 + 0'15 \cdot 8 + 0'30 \cdot 3) \cdot T_{\text{ciclo}} = NI \cdot 4'7 \cdot T_{\text{ciclo}}$$

$$F = 300 \text{ MHz} \Rightarrow T_{\text{ciclo}} = \frac{1}{F} = \frac{1}{300 \text{ MHz}} = \frac{1}{3 \cdot 10^8 \text{ Hz}} = 0'0033 \mu\text{s}$$

- a) ¿Cuál es la ganancia que se puede obtener por reducción a 3 ciclos de las instrucciones con enteros?

Para esta mejora tenemos que:

<u>Tipo i de instrucción</u>	<u>CPI<sup>a</sup></u>	<u>NI<sub>i</sub><sup>a</sup> = NI<sub>i</sub></u>
LOAD	4	0'20 NI
STORE	3	0'10 NI
ENTEROS	3	0'25 NI
FP	8	0'15 NI
BL	3	+ <u>0'30 NI</u> <u>NI</u>

$$T_{\text{mejora\_enteros}} = NI \cdot CPI^a \cdot T_{\text{ciclo}} = NI \cdot (0'20 \cdot 4 + 0'10 \cdot 3 + 0'25 \cdot 3 + 0'15 \cdot 8 + 0'30 \cdot 3) \cdot T_{\text{ciclo}} = NI \cdot 3'95 \cdot T_{\text{ciclo}}$$

Por tanto, la ganancia sería:

$$S_{\text{mejora\_enteros}} = \frac{T_{\text{sin\_mejora}}}{T_{\text{mejora\_enteros}}} = \frac{NI \cdot 4'7 \cdot T_{\text{ciclo}}}{NI \cdot 3'95 \cdot T_{\text{ciclo}}} = 1'18987 \approx 1'19 \Rightarrow$$

$$\Rightarrow \boxed{S_{\text{mejora\_enteros}} = 1'19}$$

b) ¿Cuál es la ganancia que se puede obtener por reducción a 3 ciclos de las instrucciones en coma flotante?

Para esta mejora tenemos que:

<u>Tipo i de instrucción</u>	<u>CPI<sup>b</sup></u>	<u>NI<sub>i</sub><sup>b</sup> = NI<sub>i</sub></u>
LOAD	4	0'20 NI
STORE	3	0'10 NI
ENTEROS	6	0'25 NI
FP	3	0'15 NI
BL	3	+ <u>0'30 NI</u> <u>NI</u>

$$T_{\text{mejora\_fp}} = NI^b \cdot CPI^b \cdot T_{\text{ciclo}} = NI \cdot (0'20 \cdot 4 + 0'10 \cdot 3 + 0'25 \cdot 6 + 0'15 \cdot 3 + 0'30 \cdot 3) \cdot T_{\text{ciclo}} = NI \cdot 3'95 \cdot T_{\text{ciclo}}$$

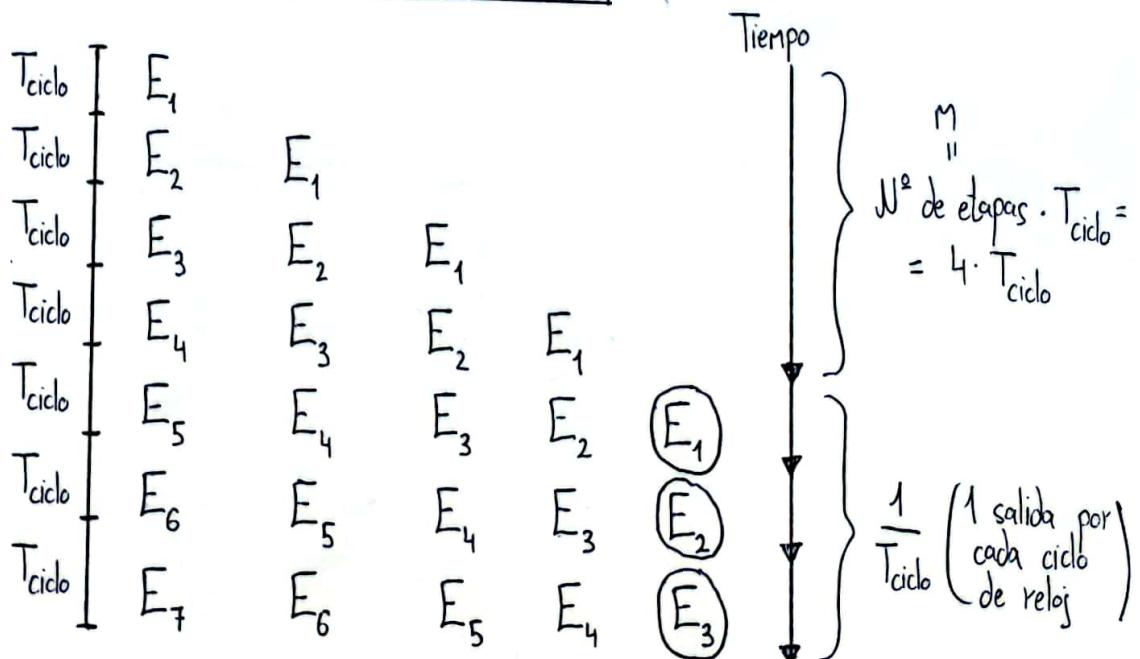
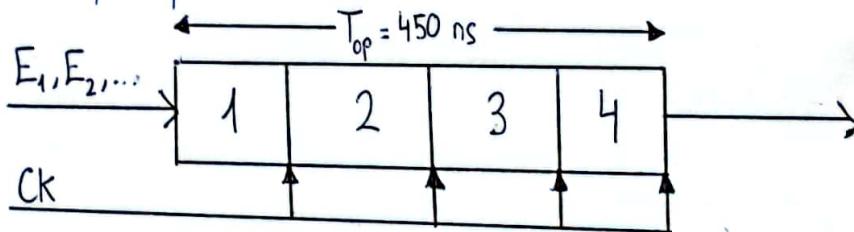
Por tanto, la ganancia sería:

$$S_{\text{mejora-fp}} = \frac{T_{\text{sin\_mejora}}}{T_{\text{mejora-fp}}} = \frac{\cancel{N} \cdot 4'7 \cdot T_{\text{ciclo}}}{\cancel{N} \cdot 3'95 \cdot T_{\text{ciclo}}} = 1'18987 \approx 1'19 \Rightarrow$$

$$\Rightarrow \boxed{S_{\text{mejora-fp}} = 1'19}$$

Nótese que la ganancia conseguida es la misma en ambos casos ( $S_{\text{mejora\_enteros}} = S_{\text{mejora-fp}} = 1'19$ ).

- ② Un circuito que implementaba una operación en un tiempo de  $T_{\text{op}} = 450 \text{ ns}$  se ha segmentado mediante un cauce lineal con cuatro etapas de duración  $T_1 = 100 \text{ ns}$ ,  $T_2 = 125 \text{ ns}$ ,  $T_3 = 125 \text{ ns}$  y  $T_4 = 100 \text{ ns}$ , respectivamente, separadas por un registro de acople que introduce un retardo de 25 ns.



$$T_{\text{ciclo}} = \max \{T_1, T_2, T_3, T_4\} + 25 \text{ ns} = 125 \text{ ns} + 25 \text{ ns} = 150 \text{ ns}$$

a) ¿Cuál es la máxima ganancia de velocidad posible? ¿Cuál es la productividad máxima del cauce?

La máxima ganancia de velocidad posible para n entradas viene dada por:

$$S(n) = \frac{T_{\text{sin-segmentación}}(n)}{T_{\text{con-segmentación}}(n)} = \frac{n \cdot T_{\text{op}}}{m \cdot T_{\text{ciclo}} + (n-1) \cdot T_{\text{ciclo}}} = \frac{n \cdot 450 \text{ ns}}{4 \cdot 150 \text{ ns} + (n-1) \cdot 150 \text{ ns}} = \\ = \frac{n \cdot 450 \text{ ns}}{150 \text{ ns} (4 + n - 1)} = \frac{3n}{n + 3}$$

$$S_{\max} = \lim_{n \rightarrow \infty} S(n) = \lim_{n \rightarrow \infty} \frac{3n}{n+3} = 3 \Rightarrow \boxed{S_{\max} = 3}$$

Por otro lado, la productividad máxima del cauce es:

$$P(n) = \frac{n}{T_{\text{con-segmentación}}(n)} = \frac{n}{m \cdot T_{\text{ciclo}} + (n-1) \cdot T_{\text{ciclo}}} = \frac{n}{4 \cdot 150 \text{ ns} + (n-1) \cdot 150 \text{ ns}} = \\ = \frac{n}{150n \text{ ns} + 450 \text{ ns}}$$

$$P_{\max} = \lim_{n \rightarrow \infty} P(n) = \lim_{n \rightarrow \infty} \frac{n}{150n \text{ ns} + 450 \text{ ns}} = \frac{1}{150 \text{ ns}} \Rightarrow \\ \Rightarrow \boxed{P_{\max} = \frac{1}{T_{\text{ciclo}}} = \frac{1}{150 \text{ ns}} = 6.67 \text{ Mop./s}}$$

b) ¿A partir de qué número de operaciones ejecutadas se consigue una productividad igual al 90% de la productividad máxima?

El valor de n para el cual se consigue una productividad igual al 90% de la productividad máxima es:

$$P(n) = 0'9 P_{\max} \Rightarrow \frac{n}{150n \text{ ns} + 450 \text{ ns}} = \frac{0'9}{150 \text{ ns}} \Rightarrow \frac{n}{n + 3} = 0'9 \Rightarrow$$

$$\Rightarrow n = 0'9(n+3) \Rightarrow n = 0'9n + 2'7 \Rightarrow 0'1n = 2'7 \Rightarrow$$

$$\Rightarrow n = \frac{2'7}{0'1} \Rightarrow \boxed{n = 27 \text{ operaciones}}$$

③ En un procesador sin segmentación de cauce, determine cuál de estas dos alternativas para realizar un salto condicional es mejor:

- )  $\text{ALT}_1$ : Una instrucción COMPARE actualiza un código de condición y es seguida por una instrucción BRANCH que comprueba esa condición. Se usan dos instrucciones.
- )  $\text{ALT}_2$ : Una sola instrucción incluye la funcionalidad de las instrucciones COMPARE y BRANCH. Se usa una única instrucción.

Hay que tener en cuenta que hay un 20% de instrucciones BRANCH para  $\text{ALT}_1$ , en el conjunto de programas de prueba; que las instrucciones BRANCH en  $\text{ALT}_1$  y COMPARE + BRANCH en  $\text{ALT}_2$  necesitan 4 ciclos mientras que todas las demás necesitan solo 3; y que el ciclo de reloj de la  $\text{ALT}_1$  es un 25% menor que el de la  $\text{ALT}_2$ , dado que en este caso la mayor funcionalidad de la instrucción COMPARE + BRANCH ocasiona una mayor complejidad en el procesador.

Para  $\text{ALT}_1$  tenemos que:

<u>Tipo i de instrucción</u>	<u>CPI<sup>1</sup><sub>i</sub></u>	<u>NI<sup>1</sup><sub>i</sub></u>	<u>Comentarios</u>
BRANCH	4	$0'2NI^1$	
Resto de instrucciones	3	$+ \frac{0'8NI^1}{NI^1}$	Incluye instrucciones COMPARE

$$T_{\text{ciclo}}^1 = T_{\text{ciclo}}^2 - 0'25 T_{\text{ciclo}}^2 = 0'75 T_{\text{ciclo}}^2$$

Por otro lado, para  $ALT_2$  tenemos que:

<u>Tipo i de instrucción</u>	<u><math>CPI_i^2</math></u>	<u><math>NI_i^2</math></u>	<u>Comentarios</u>
COMPARE + BRANCH	4	$0'2NI^1$	
Resto de instrucciones	3	$+ \frac{0'6NI^1}{0'8NI^1}$	$ALT_2$ no tendrá aparte las instrucciones COMPARE, que son un 20% en $ALT_1$

Calculemos el tiempo de CPU para ambas alternativas:

- Para  $ALT_1$  se tiene que:

$$\begin{aligned} T_{CPU}^1 &= NI^1 \cdot CPI^1 \cdot T_{ciclo}^1 = NI^1 \cdot (0'2 \cdot 4 + 0'8 \cdot 3) \cdot T_{ciclo}^1 = \\ &= NI^1 \cdot 3'2 \cdot 0'75 \cdot T_{ciclo}^2 = NI^1 \cdot 2'4 \cdot T_{ciclo}^2 \end{aligned}$$

- Para  $ALT_2$  se tiene que:

$$\begin{aligned} T_{CPU}^2 &= NI^2 \cdot CPI^2 \cdot T_{ciclo}^2 = NI^2 \cdot \frac{NI^1 \cdot (0'2 \cdot 4 + 0'6 \cdot 3)}{NI^2} \cdot T_{ciclo}^2 = \\ &= NI^1 \cdot 2'6 \cdot T_{ciclo}^2 \end{aligned}$$

Tenemos que:

$$\frac{T_{CPU}^2}{T_{CPU}^1} = \frac{NI^1 \cdot 2'6 \cdot T_{ciclo}^2}{NI^1 \cdot 2'4 \cdot T_{ciclo}^2} = 1'0833 \Rightarrow \boxed{T_{CPU}^2 = 1'0833 T_{CPU}^1}$$

Podemos observar que  $T_{CPU}^1 < T_{CPU}^2$ , luego,  $ALT_2$  no mejora a  $ALT_1$ . Es decir, aunque un repertorio de instrucciones más complejas puede reducir el número de instrucciones de los programas, reduciendo el número de instrucciones a ejecutar, esto no tiene que suponer una mejora de las prestaciones.

④ ¿Qué ocurriría en el problema anterior si el ciclo de reloj fuese únicamente un 10% mayor para la  $ALT_2$ ?

En este caso,  $T_{ciclo}^2 = 1'10 T_{ciclo}^1$ , luego:

$$T_{CPU}^1 = NI^1 \cdot CPI^1 \cdot T_{ciclo}^1 = NI^1 \cdot (0'2 \cdot 4 + 0'8 \cdot 3) \cdot T_{ciclo}^1 = NI^1 \cdot 3'2 \cdot T_{ciclo}^1$$

$$T_{CPU}^2 = NI^2 \cdot CPI^2 \cdot T_{ciclo}^2 = NI^2 \cdot \frac{NI^1(0'2 \cdot 4 + 0'6 \cdot 3)}{NI^2} \cdot 1'10 \cdot T_{ciclo}^1 = NI^1 \cdot 2'86 \cdot T_{ciclo}^1$$

$$\frac{T_{CPU}^2}{T_{CPU}^1} = \frac{NI^1 \cdot 2'86 \cdot T_{ciclo}^1}{NI^1 \cdot 3'2 \cdot T_{ciclo}^1} = 0'89375 \Rightarrow \boxed{T_{CPU}^2 = 0'89375 T_{CPU}^1}$$

Ahora,  $T_{CPU}^2 < T_{CPU}^1$  y, por lo tanto, la segunda alternativa sí que es mejor que la primera. Es decir, el mismo planteamiento que antes no mejoraba la situación de partida, ahora sí lo consigue. Solo ha sido necesario que el aumento del ciclo de reloj que se produce al hacer un diseño más complejo del procesador sea algo menor.

⑤ Considere un procesador no segmentado con una arquitectura de tipo LOAD / STORE en la que las operaciones solo utilizan como operandos registros de la CPU. Para un conjunto de programas representativos de su actividad se tiene que el 43% de las instrucciones son operaciones con la ALU (3 CPI), el 21% LOADs (4 CPI), el 12% STOREs (4 CPI) y el 24% BRANCHes (4 CPI).

Se ha podido comprobar que un 25% de las operaciones con la ALU utilizan operandos en registros que no se vuelven a utilizar. Compruebe si mejorarían las prestaciones si, para sustituir ese 25% de operaciones, se añaden instrucciones con

un dato en un registro y otro en memoria. Tenga en cuenta en la comprobación que para estas nuevas instrucciones el valor de CPI es 4 y que añadirlas ocasiona un incremento de un ciclo en el CPI de los BRANCHes, pero no afectan al ciclo de reloj.

Para la alternativa 1 tenemos que:

$$T_{\text{ciclo}}^1 = T_{\text{ciclo}}^2 = T_{\text{ciclo}}$$

<u>Tipo i de instrucción</u>	<u>CPI<sup>1</sup><sub>i</sub></u>	<u>NI<sup>1</sup><sub>i</sub></u>	<u>Comentarios</u>
ALU	3	0'43NI <sup>1</sup>	
LOAD	4	0'21NI <sup>1</sup>	
STORE	4	0'12NI <sup>1</sup>	
BRANCH	4	+ 0'24NI <sup>1</sup>	
		NI <sup>1</sup>	El 25% de las instrucciones de la ALU que usan operandos en registros no se vuelve a utilizar

Por otro lado, para la alternativa 2 tenemos que:

<u>Tipo i de instrucción</u>	<u>CPI<sup>2</sup><sub>i</sub></u>	<u>NI<sup>2</sup><sub>i</sub></u>	<u>Comentarios</u>
ALU r, r	3	0'75 · 0'43NI <sup>1</sup> = = 0'3225NI <sup>1</sup>	
ALU r, M	4	0'25 · 0'43NI <sup>1</sup> = = 0'1075NI <sup>1</sup>	
LOAD	4	0'21NI <sup>1</sup> - 0'25 · 0'43NI <sup>1</sup> = = 0'1025NI <sup>1</sup>	El 25% del 43% desaparece al usarse ese mismo número de operaciones con la ALU que acceden a memoria
STORE	4	0'12NI <sup>1</sup>	
BRANCH	5	+ 0'24NI <sup>1</sup>	
		0'8925NI <sup>1</sup>	

Calculemos el tiempo de CPU para ambas alternativas:

$$T_{\text{CPU}}^1 = NI^1 \cdot (0'43 \cdot 3 + 0'21 \cdot 4 + 0'12 \cdot 4 + 0'24 \cdot 4) \cdot T_{\text{ciclo}} = NI^1 \cdot 3'57 \cdot T_{\text{ciclo}}$$

$$T_{\text{CPU}}^2 = NI^1 \cdot (0'3225 \cdot 3 + 0'1075 \cdot 4 + 0'1025 \cdot 4 + 0'12 \cdot 4 + 0'24 \cdot 5) \cdot T_{\text{ciclo}} = NI^1 \cdot 3'4875 \cdot T_{\text{ciclo}}$$

Tenemos que:

$$\frac{T_{\text{CPU}}^2}{T_{\text{CPU}}^1} = \frac{NI^1 \cdot 3'4875 \cdot T_{\text{ciclo}}}{NI^1 \cdot 3'57 \cdot T_{\text{ciclo}}} = 0'97689 \Rightarrow \left\{ \begin{array}{l} T_{\text{CPU}}^2 = 0'97689 T_{\text{CPU}}^1 \end{array} \right.$$

Podemos observar que, en este caso,  $T_{CPU}^2 < T_{CPU}^1$ , y, por tanto, se mejoran las prestaciones. Si, por ejemplo, el porcentaje de instrucciones sustituidas fuese un 20% en lugar de un 25%, en ese caso, tendríamos que  $T_{CPU}^2 = NI^1 \cdot 3'59 \cdot T_{ciclo}$ , luego,  $T_{CPU}^2 = 1'0056 T_{CPU}^1$ . Ahora, en cambio, la segunda opción no mejora la primera.

Por tanto, queda clara la importancia que tiene el proceso de definición de conjuntos de benchmarks para evaluar las prestaciones de los computadores, y las dificultades que pueden surgir para que los fabricantes se pongan de acuerdo en aceptar un conjunto de benchmarks estándar.

- ⑥ Se ha diseñado un compilador para la máquina LOAD/STORE del problema anterior. Ese compilador puede reducir en un 50% el número de operaciones con la ALU, pero no reduce el número de LOADs, STOREs y BRANCHes. Suponiendo que la frecuencia de reloj es de 50 MHz, ¿cuál es el número de MIPS y el tiempo de ejecución que se consigue con el código optimizado? Compárelos con los correspondientes del código no optimizado.

Para la alternativa 1 tenemos que:

<u>Tipo i de instrucción</u>	<u>CPI<sup>i</sup></u>	<u>NI<sup>i</sup></u>
ALU	3	0'43 NI <sup>i</sup>
LOAD	4	0'21 NI <sup>i</sup>
STORE	4	0'12 NI <sup>i</sup>
BRANCH	4	+ $\frac{0'24 NI^i}{NI^i}$

$$T_{CPU}^1 = NI^1 \cdot CPI^1 \cdot T_{ciclo} = NI^1 \cdot (0'43 \cdot 3 + 0'21 \cdot 4 + 0'12 \cdot 4 + 0'24 \cdot 4) \cdot T_{ciclo} = NI^1 \cdot 3'57 \cdot T_{ciclo}$$

$$MIPS^1 = \frac{NI^1}{T_{CPU}^1 \cdot 10^6} = \frac{NI^1}{NI^1 \cdot 3'57 \frac{\text{ciclos}}{\text{instrucción}} \cdot T_{ciclo} \cdot 10^6} = \frac{1}{T_{ciclo}} \cdot \frac{1}{3'57 \frac{\text{ciclos}}{\text{instrucción}} \cdot 10^6} =$$

$$= \frac{F}{3'57 \frac{\text{ciclos}}{\text{instrucción}} \cdot 10^6} = \frac{50 \cdot 10^8 \frac{\text{Hz}}{\text{segundo}}}{3'57 \frac{\text{ciclos}}{\text{instrucción}} \cdot 10^8} = 14'005 \text{ MIPS} \Rightarrow \boxed{\text{MIPS}^1 = 14'005 \text{ MIPS}}$$

Por otro lado, para la alternativa 2 tenemos que:

<u>Tipo i de instrucción</u>	<u>CPI<sub>i</sub><sup>2</sup></u>	<u>NI<sub>i</sub><sup>2</sup></u>	<u>Comentarios</u>
ALU r,r	3	$0'5 \cdot 0'43 NI^1 = 0'215 NI^1$	Se reducen las instrucciones que usan la ALU en un 50%
LOAD	4	$0'21 NI^1$	
STORE	4	$0'12 NI^1$	
BRANCH	4	$0'24 NI^1$	
		$+ \frac{0'24 NI^1}{0'785 NI^1}$	

$$T_{\text{CPU}}^2 = NI^2 \cdot CPI^2 \cdot T_{\text{ciclo}} = NI^2 \cdot \frac{NI^1 (0'215 \cdot 3 + 0'21 \cdot 4 + 0'12 \cdot 4 + 0'24 \cdot 4)}{NI^2} \cdot T_{\text{ciclo}} = NI^1 \cdot 2'925 \cdot T_{\text{ciclo}} = \frac{NI^2}{0'785} \cdot 2'925 \cdot T_{\text{ciclo}} = NI^2 \cdot 3'726 \cdot T_{\text{ciclo}}$$

$$\begin{aligned} \text{MIPS}^2 &= \frac{NI^2}{T_{\text{CPU}}^2 \cdot 10^6} = \frac{NI^2}{NI^2 \cdot 3'726 \frac{\text{ciclos}}{\text{instrucción}} \cdot T_{\text{ciclo}} \cdot 10^6} = \frac{1}{T_{\text{ciclo}}} \cdot \frac{1}{3'726 \frac{\text{ciclos}}{\text{instrucción}} \cdot 10^6} = \\ &= \frac{F}{3'726 \frac{\text{ciclos}}{\text{instrucción}} \cdot 10^6} = \frac{50 \cdot 10^8 \frac{\text{Hz}}{\text{segundo}}}{3'726 \frac{\text{ciclos}}{\text{instrucción}} \cdot 10^8} = 13'419 \text{ MIPS} \Rightarrow \\ &\Rightarrow \boxed{\text{MIPS}^2 = 13'419 \text{ MIPS}} \end{aligned}$$

Como se puede observar, se consigue una reducción del tiempo de ejecución ( $T_{\text{CPU}}^2 < T_{\text{CPU}}^1$ )

pero, sin embargo, el número de MIPS para la segunda opción es menor. Se tiene aquí un ejemplo en el que los MIPS dan una información inversamente proporcional a las prestaciones. La razón de esta situación, en este caso, es que se ha reducido el número de las instrucciones que tenían un menor valor de CPI. Así, se incrementan las proporciones de las instrucciones más lentas en el segundo caso (por eso crece el valor de CPI) y, claramente, el valor de los MIPS se reduce. No obstante, hay que tener en cuenta que, aunque las instrucciones que se ejecutan son más lentas, hay que ejecutar un menor número de instrucciones y, por ello, el tiempo de ejecución se reduce.

⑦ En un programa que se ejecutan en un procesador no segmentado que funciona a 100 MHz, hay un 20% de instrucciones LOAD que necesitan 4 ciclos, un 15% de instrucciones STORE que necesitan 3 ciclos, un 40% de instrucciones con operaciones en la ALU que necesitan 6 ciclos, y un 25% de instrucciones de salto que necesitan 3 ciclos.

- a) Si en las instrucciones que usan la ALU el tiempo en la ALU supone 4 ciclos, determine cuál es la máxima ganancia que se puede obtener si se mejora el diseño de la ALU de forma que se reduce su tiempo de ejecución a la mitad de ciclos.

Para la alternativa 1 tenemos que:

<u>Tipo i de instrucción</u>	<u>CPI<sup>1</sup><sub>i</sub></u>	<u>NI<sup>1</sup><sub>i</sub></u>
ALU	6	0'4NI <sup>1</sup>
LOAD	4	0'2NI <sup>1</sup>
STORE	3	0'15NI <sup>1</sup>
BRANCH	3	0'25NI <sup>1</sup>
		+ <u><math>\frac{0'25NI^1}{NI^1}</math></u>

$$T_{CPU}^1 = NI^1 \cdot CPI^1 \cdot T_{ciclo} = NI^1 \cdot (0'4 \cdot 6 + 0'2 \cdot 4 + 0'15 \cdot 3 + 0'25 \cdot 3) \cdot T_{ciclo} = NI^1 \cdot 4'4 \cdot T_{ciclo}$$

Por otro lado, para la alternativa 2 tenemos que:

<u>Tipo i de instrucción</u>	<u>CPI<sup>2</sup><sub>i</sub></u>	<u>NI<sup>2</sup><sub>i</sub></u>
ALU	2+2=4	0'4NI <sup>1</sup>
LOAD	4	0'2NI <sup>1</sup>
STORE	3	0'15NI <sup>1</sup>
BRANCH	3	0'25NI <sup>1</sup>
		+ <u><math>\frac{0'25NI^1}{NI^1}</math></u>

$$T_{CPU}^2 = NI^2 \cdot CPI \cdot T_{ciclo} = NI^2 \cdot \frac{NI^1 \cdot (0'4 \cdot 4 + 0'2 \cdot 4 + 0'15 \cdot 3 + 0'25 \cdot 3)}{NI^2} \cdot T_{ciclo} = NI^1 \cdot 3'6 \cdot T_{ciclo}$$

De esta forma, la ganancia sería:

$$S = \frac{T_{CPU}^1}{T_{CPU}^2} = \frac{NIT \cdot 4'4 \cdot T_{ciclo}}{NIT \cdot 3'6 \cdot T_{ciclo}} = 1'22 \Rightarrow \boxed{S = 1'22}$$

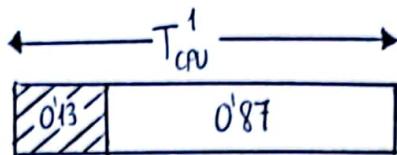
- b) ¿Con qué porcentaje de instrucciones con operaciones en la ALU se podría haber obtenido en los cálculos del apartado a) una ganancia mayor que 2? Razona su respuesta.

La ganancia en prestaciones conseguida en una instrucción de la ALU es de  $\frac{6 \text{ ciclos}}{4 \text{ ciclos}} = 1'5$ . Aunque todas las instrucciones fuesen instrucciones de la ALU, la ganancia nunca podría llegar a 2, llegaría a 1'5, que es lo que han mejorado las instrucciones de la ALU. Supongamos que todas las instrucciones son de la ALU, entonces tendríamos que:

$$S = \frac{T_{CPU}^1}{T_{CPU}^2} = \frac{4 \text{ ciclos} \cdot NIT}{6 \text{ ciclos} \cdot NIT} = 1'5$$

- ⑧ Suponga que en los programas que constituyen la carga de trabajo habitual de un procesador las instrucciones de coma flotante consumen un promedio del 13% del tiempo del procesador.

- a) Ha aparecido en el mercado una nueva versión del procesador en la que la única mejora con respecto a la versión anterior es una nueva unidad de coma flotante que permite reducir el tiempo de las instrucciones de coma flotante a tres cuartas partes del tiempo que consumían antes. ¿Cuál es la máxima ganancia de velocidad que puede esperarse en los programas si se utiliza la nueva versión del procesador?



•) Resolución 1:

$$T_{CPU}^2 = \frac{3}{4} \cdot 0'13 \cdot T_{CPU}^1 + 0'87 \cdot T_{CPU}^1 = 0'9675 \cdot T_{CPU}^1$$

$$S \leq \frac{T_{CPU}^1}{T_{CPU}^2} = \frac{T_{CPU}^1}{0'9675 \cdot T_{CPU}^1} = \frac{1}{0'9675} \approx 1'0336 \Rightarrow \\ \Rightarrow \boxed{S \leq 1'0336}$$

•) Resolución 2 (usando la ley de Amdahl):

Como el tiempo de las instrucciones de coma flotante se reduce tres cuartas partes, la mejora de velocidad es  $p = \frac{4}{3}$  (la inversa de la reducción del tiempo). Como el porcentaje de instrucciones de coma flotante es el 13%, la fracción a la que se puede aplicar la mejora es  $(1-f) = 0'13 \Rightarrow f = 0'87$ . Sustituyendo estos valores en la expresión de la ley de Amdahl, se tiene que:

$$S \leq \frac{P}{1 + f \cdot (p-1)} = \frac{\frac{4}{3}}{1 + 0'87 \left( \frac{4}{3} - 1 \right)} \approx 1'0336 \Rightarrow \boxed{S \leq 1'0336}$$

- b) ¿Cuál es la máxima ganancia de velocidad con respecto a la versión inicial del procesador que, en promedio, puede esperarse en los programas debido a mejoras en la velocidad de las operaciones en coma flotante?
- ) Resolución 1:

La ganancia será mejor cuanto menor sea el tiempo que supongan las operaciones FP. La mayor ganancia se conseguiría haciendo ese tiempo despreciable, es decir, si la mejora de las unidades FP se hace muy grande, infinita. En este caso:

$$T_{CPU}^2 = 0 + 0'87 \cdot T_{CPU}^1 = 0'87 \cdot T_{CPU}^1$$

$$S_{\max} \leq \frac{T_{CPU}^1}{T_{CPU}^2} = \frac{T_{CPU}^1}{0'87 \cdot T_{CPU}^1} = \frac{1}{0'87} \approx 1'149 \Rightarrow$$

$\Rightarrow S_{\max} \approx 1'149$

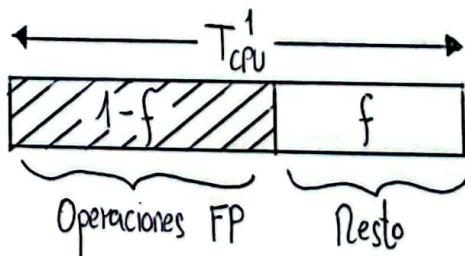
•) Resolución 2 (usando la ley de Amdahl):

Para determinar el valor de la ganancia máxima que se pide, se calcula el límite cuando la mejora del recurso (instrucciones en coma flotante) tiende a infinito:

$$S_{\max} \leq \lim_{P \rightarrow \infty} \frac{P}{1 + f \cdot (P-1)} = \frac{1}{f} = \frac{1}{0'87} \approx 1'149 \Rightarrow$$

$\Rightarrow S_{\max} \approx 1'149$

c) ¿Cuál debería ser el porcentaje de tiempo de cálculo con datos en coma flotante en los programas para esperar una ganancia máxima de 4?



•) Resolución 1:

Si se cambia la fracción de código con FP y tenemos en cuenta que para obtener la mayor ganancia se debe hacer el tiempo de ejecución de las FP igual a 0, entonces:

$$T_{CPU}^2 = 0 + f \cdot T_{CPU}^1 = f \cdot T_{CPU}^1$$

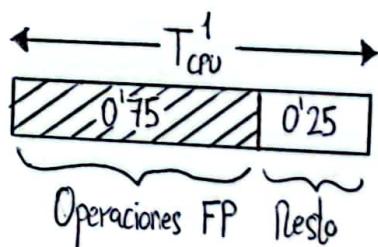
$$S_{\max} \leq \frac{T_{CPU}^1}{T_{CPU}^2} = \frac{T_{CPU}^1}{f \cdot T_{CPU}^1} = \frac{1}{f} = 4 \Rightarrow f = \frac{1}{4} = 0'25 \Rightarrow \left\{ \begin{array}{l} 1-f = 0'75 \\ \text{(Las operaciones FP deberían ser un 75%)} \end{array} \right.$$

•) Resolución 2 (usando la ley de Amdahl):

Para que la ganancia máxima (es decir, cuando la mejora,  $p$ , tiende a infinito) sea igual a 4, la fracción del tiempo inicial que no se reduce con la mejora,  $f$ , se obtendría a partir de:

$$\begin{aligned} S_{\max} &\leq \lim_{p \rightarrow \infty} \frac{p}{1 + f \cdot (p-1)} = \frac{1}{f} = 4 \Rightarrow f = \frac{1}{4} = 0'25 \Rightarrow \\ &\Rightarrow \{1-f=0'75\} \end{aligned}$$

d) ¿Cuánto debería reducirse el tiempo de las operaciones en coma flotante con respecto a la situación inicial para que la ganancia máxima sea 2 suponiendo que en la versión inicial el porcentaje de tiempo de cálculo con coma flotante es el obtenido en c)?



•) Resolución 1:

$$T_{CPU}^2 = 0'75 \cdot \frac{1}{p} \cdot T_{CPU}^1 + 0'25 \cdot T_{CPU}^1$$

$$\begin{aligned} S_{\max} &\leq \frac{T_{CPU}^1}{T_{CPU}^2} = \frac{T_{CPU}^1}{0'75 \cdot \frac{1}{p} \cdot T_{CPU}^1 + 0'25 \cdot T_{CPU}^1} = \frac{1}{0'75 \cdot \frac{1}{p} + 0'25} = 2 \Rightarrow \\ &\Rightarrow 1 = \frac{1'5}{p} + 0'5 \Rightarrow p = \frac{1'5}{0'5} \Rightarrow \{p=3\} \end{aligned}$$

El tiempo se debería reducir a  $\frac{1}{3}$  del tiempo original, es decir, las instrucciones de coma flotante deberían hacerse tres veces más rápidas.

•) Resolución 2 (usando la ley de Amdahl):

$$S_{\max} \leq \frac{p}{1 + f \cdot (p-1)} = \frac{1}{f + \frac{1-f}{p}} = \frac{1}{0.25 + \frac{0.75}{p}} = 2 \Rightarrow$$

$$\Rightarrow 1 = 0.5 + \frac{1.5}{p} \Rightarrow p = \frac{1.5}{0.5} \Rightarrow \boxed{p=3}$$

Es decir, las instrucciones de coma flotante deberían hacerse tres veces más rápidas.

- ⑨ Suponga que, en el código siguiente,  $a[]$  es un array de números de 32 bits en coma flotante y  $b$  un número de 32 bits en coma flotante y que debería ejecutarse en menos de 0.5 segundos para  $N=10^9$ :

for ( $i=0$ ;  $i < N$ ;  $i++$ )  $a[i+2] = (a[i+2] + a[i+1] + a[i]) \cdot b$ ;

- a) ¿Cuántos GFLOPS se necesitan para poder ejecutar el código en menos de 0.5 segundos?

Puesto que hay tres operaciones en coma flotante (dos sumas y un producto, con igual coste para el producto y la suma), el número de operaciones en coma flotante tras realizar  $N$  iteraciones es  $N_{fp} = 3N = 3 \cdot 10^9$  operaciones. Por tanto:

$$\text{GFLOPS} = \frac{N_{fp}}{T_{CPU} \cdot 10^9} = \frac{3 \cdot 10^9}{0.5 \cdot 10^9} = 6 \Rightarrow \boxed{\text{GFLOPS} = 6 \text{ GFLOPS}}$$

- b) Suponiendo que este código en ensamblador tiene 7N instrucciones y que se ha ejecutado en un procesador de 32 bits a 2 GHz. ¿Cuál es el número medio de instrucciones que el procesador tiene que completar por ciclo para poder ejecutar el código en menos de 0.5 segundos?

Si el programa tiene  $NI = 7 \cdot 10^9$  instrucciones, tenemos que:

$$T_{CPU} < 0'5 \text{ s} \Rightarrow NI \cdot CPI \cdot T_{ciclo} < 0'5 \text{ s} \Rightarrow$$

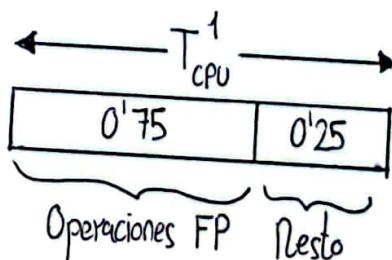
$$\Rightarrow \frac{NI}{\frac{1}{CPI} \cdot \frac{1}{T_{ciclo}}} < 0'5 \text{ s} \Rightarrow$$

$$\Rightarrow \frac{NI}{IPC \cdot F} < 0'5 \text{ s} \Rightarrow$$

$$\Rightarrow IPC > \frac{NI}{0'5 \cdot F} = \frac{7 \cdot 10^9 \text{ instrucciones}}{0'5 \cdot 2 \cdot 10^9 \frac{\text{ciclos}}{\text{segundo}}} = 7 \frac{\text{instrucciones}}{\text{ciclo}} \Rightarrow$$

$$\Rightarrow \left\{ \begin{array}{l} IPC > 7 \frac{\text{instrucciones}}{\text{ciclo}} \end{array} \right.$$

- c) Estimando que el programa pasa el 75% de su tiempo de ejecución realizando operaciones en coma flotante, ¿cuánto disminuiría como mucho el tiempo de ejecución si se redujese un 75% los tiempos de las unidades de coma flotante?



- Resolución 1:

$$T_{CPU}^2 = 0'25 \cdot 0'75 \cdot T_{CPU}^1 + 0'25 \cdot T_{CPU}^1 = 0'4375 \cdot T_{CPU}^1$$

$$S \leq \frac{T_{CPU}^1}{T_{CPU}^2} = \frac{T_{CPU}^1}{0'4375 \cdot T_{CPU}^1} = \frac{1}{0'4375} = 2'286$$

$$\frac{T_{CPU}^1 - T_{CPU}^2}{T_{CPU}^1} = 1 - \frac{T_{CPU}^2}{T_{CPU}^1} = 1 - \frac{1}{S} = 1 - 0'4375 = 0'5625 \Rightarrow \left\{ \begin{array}{l} \text{El tiempo} \\ \text{se reduce} \\ \text{un 56'25\%} \end{array} \right.$$

## •) Resolución 2 (usando la ley de Amdahl):

En este caso,  $f = 0'25$  es la fracción de tiempo en el que no se puede aprovechar la mejora; el incremento de velocidad es  $p = \frac{1}{0'25} = 4$  (si el tiempo de las operaciones en coma flotante era  $t$ , el tiempo de las operaciones en coma flotante con la mejora es  $0'25t$ , dado que se ha reducido un 75%). La mejora de velocidad que se observaría se puede obtener aplicando la ley de Amdahl:

$$S \leq \frac{p}{1 + f \cdot (p-1)} = \frac{4}{1 + 0'25 \cdot (4-1)} = 2'286$$

Como  $S = \frac{T_{CPU}^1}{T_{CPU}^2} \leq 2'286 \Rightarrow \frac{T_{CPU}^1}{2'286} \leq T_{CPU}^2 \Rightarrow 0'4375 \cdot T_{CPU}^1 \leq T_{CPU}^2$ , es decir, el tiempo con la mejora podría pasar a ser el 43'75% del tiempo sin la mejora y, por tanto, se reduce el tiempo en un 56'75%.

## Cuestiones

- ① Indique cómo se podría aprovechar un computador MISD para acelerar la determinación de si  $n$  números son primos o no. Considere que se conocen los  $M$  números primos entre 1 y el máximo valor que puede tener un número de entrada.

Pendiente.

- ② Indique cuál es la diferencia fundamental entre una arquitectura CC-NUMA y una arquitectura SMP.

Ambos, CC-NUMA y SMP, son multiprocesadores, es decir, comparten el espacio de direcciones físicas. También en ambos se mantiene coherencia entre cachés de distintos procesadores (CC = Cache-Coherence).

En un SMP (Symmetric Multiprocessor), la memoria se encuentra centralizada en el sistema a igual distancia (latencia) de todos los procesadores, mientras que, en un CC-NUMA (Cache-Coherence Non Uniform Memory Access), cada procesador tiene físicamente cerca un conjunto de sus direcciones de memoria porque los módulos de memoria están físicamente distribuidos entre los procesadores y, por tanto, los módulos no están a igual distancia (latencia) de todos los procesadores.

La memoria centralizada del SMP hace que el tiempo de acceso a una dirección de memoria en un SMP sea igual para todos los procesadores y el tiempo de acceso a memoria de un procesador sea el mismo independientemente de la dirección a la que se esté accediendo. Por estos motivos se denomina SMP (Symmetric Multiprocessor) o UMA (Uniform Memory Access).

Por otro lado, los módulos de memoria físicamente distribuidos entre los procesadores de un CC-NUMA hacen que el tiempo de acceso a memoria dependa de si el procesador accede a una dirección de memoria que está en la memoria física que se encuentra en el nodo de dicho procesador (cercana, por tanto, al procesador que accede) o en la memoria física de otro nodo. Por este motivo, el acceso no es uniforme o simétrico y recibe el nombre de NUMA (Non Uniform Memory Access).

③ ¿Cuándo diría que un computador es un multiprocesador y cuándo que es un multicomputador?

Será un multiprocesador si todos los procesadores comparten el mismo espacio de direcciones físico y será un multicomputador si cada procesador tiene su espacio de direcciones físico propio.

④ ¿Un CC-NUMA escala más que un SMP? ¿Por qué?

Sí, un CC-NUMA escala más que un SMP, porque al añadir procesadores al cálculo, el incremento en el tiempo de acceso a memoria medio aumenta menos en un CC-NUMA que en un SMP si el sistema operativo, el compilador y/o el programador se esfuerzan en ubicar en la memoria cercana a un procesador la carga de trabajo que se va a procesar. La menor latencia media se debe a un menor número de conflictos en la red de interconexión en el acceso a datos y a la menor distancia con el módulo de memoria físico al que se accede. Al aumentar menos la latencia se puede conseguir un tiempo de respuesta mejor en un CC-NUMA que en un SMP al ejecutar un código paralelo o múltiples códigos secuenciales a la vez.

⑤ Indique qué niveles de parallelismo implícito en una aplicación puede aprovechar un PC con un procesador de 4 cores, teniendo en cuenta que cada core tiene unidades funcionales SIMD (también llamadas unidades multimedia) y una microarquitectura segmentada y superescalar. Plázone su respuesta.

- ) Parallelismo a nivel de operación. Al ser una arquitectura con parallelismo a nivel de instrucción (ejecuta instrucciones en paralelo) por tener una arquitectura segmentada y superescalar, puede ejecutar operaciones en paralelo, luego, puede aprovechar el parallelismo a nivel de operación.
- ) Parallelismo a nivel de bucle (parallelismo de datos). Las operaciones realizadas en las iteraciones del bucle sobre vectores y matrices se podrían implementar en las unidades SIMD de los núcleos, lo que reduciría el número de iteraciones de los bucles. Además, las iteraciones de los bucles, si son independientes, se podrían repartir entre los 4 núcleos.
- ) Parallelismo a nivel de función. Las funciones independientes se podrían asignar a núcleos distintos para que se puedan ejecutar a la vez.
- ) Parallelismo a nivel de programa. Los programas del mismo o distinto usuario se pueden asignar a distintos núcleos para así ejecutarlos al mismo tiempo.

⑥ Si le dicen que un ordenador es de 20 MIPS, ¿puede estar seguro que ejecutará cualquier programa de 20 000 instrucciones en un microsegundo?

Los MIPS se definen como el número de instrucciones que puede ejecutar un procesador (en millones) divididos por el tiempo que tardan en ejecutarse.

$$\text{MIPS} = \frac{\text{Número de instrucciones}}{\text{Tiempo (en segundos)} \cdot 10^6}$$

Los MIPS suelen utilizarse como medida de las prestaciones de un procesador, pero realmente solo permiten estimar la velocidad pico del procesador, que solo permitiría comparar procesadores con el mismo repertorio de instrucciones en cuanto a sus velocidades pico. Esto se debe a que:

- ) Esta medida no tiene en cuenta las características del repertorio de instrucciones de procesador. Se da el mismo valor a una instrucción que realice una operación compleja que a una instrucción sencilla.
- ) Pueden tenerse valores de MIPS inversamente proporcionales a la velocidad del procesador. Si un procesador tiene un repertorio de instrucciones de tipo CISC que permite codificar un algoritmo con, por ejemplo, 1000 instrucciones, y otro con un repertorio de tipo RISC lo codifica con 2000, si el primero tarda 1 microsegundo y el segundo 1'5 microsegundos, tendremos que el primer procesador tiene 1000 MIPS y el segundo 1333 MIPS. Por tanto, si nos fijamos en los MIPS, el segundo procesador será mejor que el primero, aún precisando un 50 % más de tiempo que el primero, para resolver el mismo problema.

De ahí que incluso se haya dicho que MIPS significa "Meaningless Information of Processor Speed" (información sin sentido de la velocidad del procesador).

En relación a la pregunta que se plantea, la respuesta puede tener en cuenta dos aspectos:

- 1) El número de instrucciones que constituyen un programa (número estático de instrucciones) puede ser distinto del número de instrucciones que ejecuta el procesador finalmente (número dinámico de instrucciones), ya que puede haber instrucciones de salto, bucle, etc. que hacen que ciertas instrucciones del código se ejecuten más de una vez, y otras no se ejecuten nunca. Si la pregunta, al hacer referencia al número de instrucciones, se refiere al número estático, la respuesta es que no se puede estar seguro puesto que quizás al final no se ejecuten 20 000 millones de instrucciones.
- 2) Si el repertorio de instrucciones contiene instrucciones que tardan en ejecutarse tiempos diferentes, para que el programa tarde el mismo tiempo es preciso que se ejecute el mismo número de instrucciones y del mismo tipo (en cuanto a tiempo de ejecución de cada una). En este caso, la respuesta también es que no se puede estar seguro.

⑦ ¿Aceptaría financiar/embarcarse en un proyecto en el que se plantease el diseño e implementación de un computador de propósito general con arquitectura MISD? (Justifique su respuesta).

El tipo de procesamiento que realiza un procesador MISD puede implementarse en un procesador MIMD con la sincronización correspondiente entre los procesadores del computador MIMD para que los datos vayan pasando adecuadamente de un procesador a otro (definiendo un flujo único de datos que pasan de procesador a procesador). Por esta razón, un computador MISD no tiene mucho sentido como computador de propósito general. Sin embargo, puede ser útil un diseño MISD para un dispositivo de propósito específico que solo tiene que ejecutar una aplicación que implica un procesamiento en el que los datos tienen que pasar por distintas etapas en las que sufren ciertas transformaciones que pueden programarse (en cada uno de los procesadores). La especificidad del diseño puede permitir generar diseños muy eficientes desde el punto de vista del consumo, de la complejidad hardware, etc.

⑧ Deduzca la expresión que se usa para representar la ley de Amdahl suponiendo que se mejora un recurso del procesador, que hay una probabilidad  $f$  de no utilizar dicho recurso y que la mejora supone un incremento en un factor  $p$  de la velocidad de procesamiento del recurso.

El tiempo necesario para ejecutar un programa en el procesador sin la mejora es igual a:

$$T_{\text{sin\_mejora}} = f \cdot T_{\text{sin\_mejora}} + (1-f) \cdot T_{\text{sin\_mejora}}$$

donde  $f \cdot T_{\text{sin\_mejora}}$  es el tiempo que no podría reducirse al aplicar la mejora y  $(1-f) \cdot T_{\text{sin\_mejora}}$  es el tiempo que podría reducirse como máximo en un factor igual a  $p$  al aplicar la mejora. Por tanto, el tiempo al aplicar la mejora sería:

$$T_{\text{con\_mejora}} \geq f \cdot T_{\text{sin\_mejora}} + \frac{(1-f) \cdot T_{\text{sin\_mejora}}}{p}$$

El signo " $>$ " se usa porque al aplicar la mejora es posible añadir algún tiempo de sobrecarga extra. Según esto, la ganancia de velocidad que se podría conseguir sería:

$$S = \frac{T_{\text{sin\_mejora}}}{T_{\text{con\_mejora}}} \leq \frac{T_{\text{sin\_mejora}}}{f \cdot T_{\text{sin\_mejora}} + \frac{(1-f) \cdot T_{\text{sin\_mejora}}}{p}} = \frac{1}{f + \frac{1-f}{p}} = \frac{p}{1+f(p-1)}$$

Y así llegamos a la expresión que se utiliza para describir la ley de Amdahl:

$$\left\{ S \leq \frac{p}{1+f \cdot (p-1)} \right.$$

⑨ ¿Es cierto que si se mejora una parte de un sistema (por ejemplo, un recurso de un procesador) se observa experimentalmente que, al aumentar el factor de mejora, llega un momento en que se satura el incremento de velocidad que se consigue? (Justifique la respuesta).

Para responder a esta pregunta recurrimos a la ley de Amdahl, que marca un límite superior a la mejora de velocidad. Como se tiene que  $S \leq \frac{P}{1+f(p-1)}$ , donde  $p$  es el factor de mejora y  $f$  la fracción de tiempo sin la mejora en el que dicha mejora no puede aplicarse, se puede calcular la derivada de  $\frac{P}{1+f(p-1)}$  con respecto a  $p$  y ver qué pasa cuando cambia  $p$ . De esta forma, tenemos que:

$$\frac{d}{dp} \left( \frac{P}{1+f(p-1)} \right) = \frac{1+f(p-1)-pf}{(1+f(p-1))^2} = \frac{1-f}{(1+f(p-1))^2}$$

y, como puede verse, a medida que aumenta  $p$ , el denominador se va haciendo mayor y, como  $(1-f)$  se mantiene fijo, la derivada tiende a cero. Eso significa que la pendiente de la curva que describe la variación  $\frac{P}{1+f(p-1)}$  con respecto a  $p$  va haciéndose igual a cero y, por lo tanto, no aumenta la ganancia de velocidad: se satura o, lo que es lo mismo, llega a una asintota que estará situada en  $\frac{P}{f}$  (que es al valor al que tiende  $\frac{P}{1+f(p-1)}$  cuando  $p$  tiende a infinito).

- ⑩ ¿Es cierto que la cota para el incremento de velocidad que establece la ley de Amdahl crece a medida que aumenta el valor del factor de mejora aplicado al recurso o parte del sistema que se mejora? (Justifique la respuesta).

La respuesta a esta pregunta se deduce de la misma expresión de la derivada de la cota que establece la ley de Amdahl, calculada al responder la cuestión 9:

$$\frac{d}{dp} \left( \frac{P}{1+f(p-1)} \right) = \frac{1+f(p-1)-pf}{(1+f(p-1))^2} = \frac{1-f}{(1+f(p-1))^2}$$

Dado que  $(1-f)$  es positivo (solo en el peor de los casos, si no se pudiera aplicar el factor de ganancia a ninguna parte del programa, tendríamos que  $(1-f)=0$  y no se obtendría ninguna ganancia de velocidad), y, puesto que  $(1+f(p-1))^2$  siempre es positivo, la derivada es positiva. Eso significa que  $\underline{1+f(p-1)}$  crece al crecer  $p$ . Lo que ocurre es que, tal y como se ha visto en la cuestión 9, este crecimiento es cada vez menor (tiende a cero).

- 11** ¿Qué podría ser mejor suponiendo velocidades pico, un procesador superescalar capaz de emitir cuatro instrucciones por ciclo, o un procesador vectorial cuyo repertorio permite codificar 8 operaciones por instrucción y emite una instrucción por ciclo? (Justifique su respuesta).

Considérese una aplicación que se codifica con  $N$  instrucciones de un repertorio de instrucciones escalar (cada instrucción codifica una operación). La cota inferior para el tiempo que un procesador superescalar podría tardar en ejecutar ese programa sería:

$$T_{\text{superescalar}} = \frac{N \cdot T_{\text{ciclo\_superescalar}}}{4}$$

Un procesador vectorial ejecutaría un número de instrucciones menor puesto que su repertorio de instrucciones permitiría codificar hasta ocho operaciones iguales (sobre datos diferentes) en una instrucción. Si suponemos que la aplicación permite que todas las operaciones que hay que ejecutar se puedan "empaquetar" en instrucciones vectoriales, el procesador vectorial tendría que ejecutar  $\frac{N}{8}$  instrucciones. Por tanto, considerando también que el procesador vectorial consigue terminar instrucciones según su velocidad pico (como supusimos en el caso del superescalar), la cota inferior para el tiempo que tarda la ejecución del programa en el procesador vectorial sería:

$$T_{\text{vectorial}} = \frac{N}{8} \cdot T_{\text{ciclo\_vectorial}}$$

Por tanto:

$$\frac{T_{\text{superescalar}}}{T_{\text{vectorial}}} = \frac{\frac{N \cdot T_{\text{ciclo\_superescalar}}}{4}}{\frac{N}{8} \cdot T_{\text{ciclo\_vectorial}}} = \frac{2 \cdot T_{\text{ciclo\_superescalar}}}{T_{\text{ciclo\_vectorial}}}$$

Según esto, si los dos procesadores funcionan a la misma frecuencia, el procesador vectorial podría ser mejor (siempre y cuando las hipótesis que se están considerando de que están procesando instrucciones según su máximo rendimiento se puedan considerar suficientemente aproximada a la realidad).

No obstante, hay que tener en cuenta que usualmente, los procesadores superescalares han sido los que más rápidamente han incorporado las mejoras tecnológicas y, normalmente, los procesadores superescalares funcionan a frecuencias más altas que los procesadores vectoriales contemporáneos. En el ejemplo, si la frecuencia del superescalar fuera más del doble de la del vectorial, sería mejor utilizar un procesador superescalar.

Además, hay que tener en cuenta que para que los procesadores vectoriales sean eficientes (se puedan admitir las hipótesis en cuanto a codificación de las operaciones y funcionamiento según la velocidad pico) es necesario que el programa tenga un alto grado de paralelismo de datos (son procesadores más específicos que los superescalares). Por eso, para dar una respuesta más precisa habría que conocer más detalles de la carga de trabajo que se piensa ejecutar en los procesadores.

- ⑫ En la Lección 2 de AC se han presentado diferentes criterios de clasificación de computadores y en el Seminario 0 de prácticas se ha presentado alcgrid. Clasifique alcgrid, sus nodos, sus encapsulados y sus núcleos dentro de la clasificación de Flynn y dentro de la clasificación que usa como criterio el sistema de memoria. Plázone su respuesta.

Nos vamos a centrar en los nodos de cómputo de alcgrid (alcgrid1 - alcgrid4).

Los nodos de cómputo de alcgrid, alcgrid1 - alcgrid4, están conectados entre sí mediante un conmutador Ethernet. Esto supone que, por hardware, un nodo no puede acceder a la memoria que se encuentra en otros nodos. El software del sistema de comunicación ofrece funciones para copiar datos de la memoria de un nodo a la memoria de otro nodo. Por tanto, en el nivel de empaquetamiento/conexión de sistema, alcgrid es una arquitectura MIMD (clasificación de Flynn) y, en particular, un multicomputador o arquitectura NORMA (clasificación del sistema de memoria).

Los dos nodos de atcgrid tienen una placa con dos sockets (encapsulados) que comparten memoria, aunque cada socket tiene acceso mediante un enlace (conexión punto-a-punto dedicada exclusivamente a la comunicación entre esos puntos) a un conjunto de direcciones de memoria del sistema de memoria compartido. Por tanto, en el nivel de placa, atcgrid es una arquitectura MIMD (clasificación de Flynn) y, en particular, un multiprocesador CC-NUMA (clasificación del sistema de memoria). Es NUMA por tener el espacio de memoria compartido físicamente distribuido entre los sockets (encapsulados), lo que hace que un núcleo tarde menos en acceder a la memoria que está directamente conectada con un enlace al encapsulado en el que se encuentra que a la memoria conectada al otro encapsulado de la placa. Además, el hardware mantiene coherencia entre las cachés de último nivel de los encapsulados, de ahí que aparezca CC (Cache-Coherent) en el nombre.

Los encapsulados de atcgrid1-atcgrid4 constan de un único dado de silicio con múltiples núcleos que comparten el último nivel de caché. Por tanto, en el nivel de encapsulado, atcgrid es una arquitectura MIMD (clasificación de Flynn) y, en particular, un multiprocesador UMA o SMP en un chip (clasificación del sistema de memoria), también llamado en la bibliografía científica CMP (Chip MultiProcessor). Es UMA porque la latencia de acceso al último nivel de memoria caché del encapsulado es igual para todos los núcleos.

Los núcleos de atcgrid tiene repertorio de instrucciones vectoriales porque tiene unidades funcionales multimedia que permiten ejecutar en paralelo múltiples operaciones escalares (suma, and, multiplicación, etc.). Las unidades funcionales multimedia implementan una arquitectura SIMD (la memoria en la que están los datos con los que se opera es un registro). Además, cada núcleo de atcgrid ejecuta dos flujos de control en paralelo (dos threads del SO en paralelo) debido a la implementación de multithread simultáneo (HyperThreading). Desde el punto de vista del SO, un núcleo de atcgrid son dos procesadores o núcleos independientes (es como si el núcleo tuviera dos núcleos lógicos). Por tanto, desde el punto de vista del SO, un núcleo tiene dos procesadores de una arquitectura MIMD, aunque en realidad físicamente los dos flujos de control de un núcleo comparten una unidad de control y una unidad de procesamiento.

(13) En la Lección 1 de AC se han presentado diferentes criterios de clasificación del paralelismo implícito en una aplicación y en el Seminario 0 de prácticas se ha presentado alcgrid. ¿Qué tipos de paralelismo aprovecha alcgrid? Razona su respuesta.

Nos vamos a centrar en los nodos de cómputo de alcgrid (alcgrid1 - alcgrid4).

•) Paralelismo a nivel de operación:

- Al ser una arquitectura con paralelismo a nivel de instrucción, por tener una arquitectura segmentada y superescalar, puede ejecutar operaciones en paralelo, luego, puede aprovechar el paralelismo a nivel de operación.

•) Paralelismo a nivel de bucle (paralelismo de datos):

- Las operaciones realizadas en las iteraciones del bucle sobre vectores y matrices se podrían implementar en las unidades SIMD de los núcleos de alcgrid, lo que reduciría el número de iteraciones de los bucles del código máquina.
- Las iteraciones de los bucles se podrían repartir entre threads del SO. Los núcleos de alcgrid pueden ejecutar dos threads del SO en paralelo debido a la implementación de multithread simultánea (HyperThreading). Los threads de un mismo código también se pueden repartir entre los múltiples núcleos físicos de un encapsulado de un nodo de cómputo de alcgrid y entre los núcleos de los dos encapsulados de una placa alcgrid. En alcgrid1, alcgrid2 o alcgrid3 se puede repartir trabajo entre 12 núcleos físicos y 24 lógicos y, en alcgrid4, entre 32 núcleos físicos y 64 lógicos.
- Si las iteraciones de los bucles se reparten además entre procesos del SO, se podrían aprovechar los núcleos de todos los nodos de alcgrid. En total se podrían ejecutar en paralelo en los nodos de cómputo  $24 \cdot 3 + 64$  flujos de instrucciones repartidos entre un mínimo de 4 procesos (uno por nodo).
- Las operaciones realizadas en las iteraciones del bucle sobre vectores y matrices se podrían ejecutar en paralelo por los núcleos de procesamiento de la GPU del nodo alcgrid4.

•) Paralelismo a nivel de función (parallelismo de tareas):

- Las funciones independientes se podrían ejecutar en paralelo usando los dos threads de los núcleos, todos los núcleos de los dos encapsulados e incluso, si el parallelismo se hace explícito a nivel de proceso, todos los nodos.

•) Paralelismo a nivel de programa:

- Los programas del mismo o distinto usuario se pueden asignar a distintos núcleos de nodos de cómputo de algoritmo, estén o no en el mismo nodo, para así ejecutarlos al mismo tiempo. El parallelismo a nivel de programa solo se puede hacer explícito a nivel de proceso del SO.