Ejercicio 1. En 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. **(a)** ¿Cuál es la ganancia que se puede obtener por reducción en el tiempo de las instrucciones con enteros a 3 ciclos? y **(b)** ¿cuál es la ganancia que se puede obtener por reducción en el tiempo de las instrucciones en coma flotante a 3 ciclos?

Solución

Datos del ejercicio:

Tipo i de instr.	CPI_i (c/i)	NI_i	CPI_i^a	CPI_i^b
LOAD	4	0.20*NI	4	4
STORE	3	0.10*NI	3	3
ENTEROS	6	0.25*NI	3	6
FP	8	0.15*NI	8	3
BR	3	0.30*NI	3	3
TOTAL		NI		

El tiempo total de procesamiento antes de la mejora es igual a:

$$T_{sin\_mejora} = NI \times CPI \times T_{ciclo} = (NI \times 0.2 \times 4 + NI \times 0.1 \times 3 + NI \times 0.25 \times 6 + NI \times 0.15 \times 8 + NI \times 0.3 \times 3) \times T_{ciclo} =$$
$$NI \times (0.2 \times 4 + 0.1 \times 3 + 0.25 \times 6 + 0.15 \times 8 + 0.3 \times 3) \times T_{ciclo} =$$
$$NI \times (0.8 + 0.3 + 1.5 + 1.2 + 0.9) \times T_{ciclo} = NI \times 4.7 \times T_{ciclo}$$

donde NI es el número de instruciones y T_{ciclo}=1/frecuencia

(a) En el caso de mejoras en la ejecución de enteros, el tiempo sería:

$$T_{mejora\_ent} = NI \times CPI \times T_{ciclo} = NI \times (0.2 \times 4 + 0.1 \times 3 + 0.25 \times \mathbf{3} + 0.15 \times 8 + 0.3 \times 3) \times T_{ciclo} =$$
$$NI \times (0.8 + 0.3 + 0.75 + 1.2 + 0.9) \times T_{ciclo} = NI \times 3.95 \times T_{ciclo}$$

y por tanto la ganancia sería:

$$S_{mejora\_ent} = T_{sin\_mejora} / T_{mejora\_ent} = NI \times 4.7 \times T_{ciclo} \; / \; NI \times 3.95 \times T_{ciclo} = 4{,}7 / 3{,}95 = 1{,}18987$$

$$S_{mejora\_ent} = 1.19$$

(b) En el caso de mejoras en operaciones de coma flotante, el tiempo sería:

$$T_{mejora\_fp} = NI \times CPI \times T_{ciclo} = NI \times (0.2 \times 4 + 0.1 \times 3 + 0.25 \times 6 + 0.15 \times 3 + 0.3 \times 3) \times T_{ciclo} =$$
$$NI \times (0.8 + 0.3 + 1.5 + 0.45 + 0.9) \times T_{ciclo} = NI \times 3.95 \times T_{ciclo}$$

y por tanto la ganancia sería:

$$S_{mejora\_fp} = T_{sin\_mejora} / T_{mejora\_fp} = NI \times 4.7 \times T_{ciclo} \; / \; NI \times 3.95 \times T_{ciclo} = 4{,}7/3{,}95 = 1{,}18987$$

$$S_{mejora\_fp} = 1.19$$

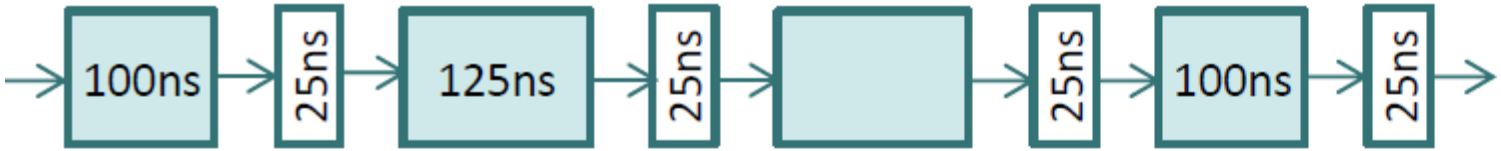
En ambos casos la ganancia conseguida es la misma.

Ejercicio 2. Un circuito que implementaba una operación en T_{op}=450 ns. se ha segmentado mediante un cauce lineal con cuatro etapas de duracción T1=100 ns., T2=125 ns., T3=125 ns., y T4=100 ns. respectivamente, separadas por un registro de acoplo que introduce un retardo de 25 ns. **(a)** ¿Cuál es la máxima ganancia de velocidad posible? ¿Cuál es la productividad máxima del cauce? **(b)** ¿A partir de qué número de operaciones ejecutadas se consigue una productividad igual al 90% de la productividad máxima?

Solución

Datos del ejercicio:

Sin segmentar



Con segmentación

(a) La ganancia en velocidad para n entradas se obtiene dividiendo en tiempo que tardan en ejecutarse n entradas en el circuito sin segmentar entre el tiempo que supone la ejecución en el circuito segmentado:

$$S(n) = \frac{T^{ss}(n)}{T^{cs}(n)}$$

El tiempo de ejecución de n entradas en el circuito sin segmentar es:

$$T^{ss}(n) = T_{op} \times n = 450 \text{ ns} \times n$$

El circuito se ha segmentado en cuatro etapas separadas por un registro de acoplo con un retardo de d=25 ns. El tiempo de ciclo del cauce, t, se obtiene a partir de la expresión:

$$t = \max\{T1, T2, T3, T4\} + d = \max\{100, 125, 125, 100\} + 25 \text{ ns} = 150 \text{ ns}$$

La ganancia en prestaciones conseguida al segmentar sería:

$$S(n) = \frac{T^{ss}(n)}{T^{cs}(n)} = \frac{450\ ns \times n}{4 \times 150\ ns + (n-1) \times 150\ ns} = \frac{3 \times n}{3+n}$$

El valor máximo de la ganancia de velocidad se obtiene aplicando el límite cuando n tiende a infinito:

$$S(n) = \lim_{n\to\infty} \frac{3 \times n}{3+n} = \mathbf{3}$$

La productividad del cauce es:

$$P(n) = \frac{n}{T^{cs}(n)} = \frac{n}{4 \times 150\ ns + (n-1) \times 150\ ns} = \frac{n}{(3+n) \times 150\ ns}$$

La productividad máxima es:

$$P(n) = \lim_{n\to\infty} \frac{n}{(3+n) \times 150\ ns} = \frac{1}{150\ ns} = \mathbf{6.67\ Mop./s}$$

(b) El valor de n para el que se consigue una productividad igual al 90% de la productividad máxima es:

$$P(n) = \frac{n}{(3+n) \times 150\ ns} = 0.9 \times \frac{1}{150\ ns} \Rightarrow \frac{n}{3+n} = 0.9 \Rightarrow n = 2.7 + 0.9 \times n \Rightarrow n = \frac{2.7}{0.1} = \mathbf{27\ op.}$$

Con 27 operaciones se alcanza el 90% de la productividad máxima. La productividad aumentará conforme se incremente n.

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

•ALT1: Una instrucción COMPARE actualiza un código de condición y es seguida por una instrucción BRANCH que comprueba esa condición.

•ALT2: Una sola instrucción incluye la funcionalidad de las instrucciones COMPARE y BRANCH.

Hay que tener en cuenta que para ALT1, en el conjunto de programas de prueba, el 20% de las instrucciones son instrucciones BRANCH asociadas a salto condicional; que las instrucciones BRANCH en ALT1 y COMPARE+BRANCH en ALT2 necesitan 4 ciclos mientras que todas las demás necesitan sólo 3; y que el ciclo de reloj de la ALT1 es un 25% menor que el de la ALT2, dado que en éste caso la mayor funcionalidad de la instrucción COMPARE+BRANCH ocasiona una mayor complejidad en el procesador.

Datos del ejercicio:

$$T^1_{ciclo} = T^2_{ciclo} - 0{,}25 \times T^2_{ciclo} = 0{,}75 \times T^2_{ciclo}$$

ALT1

Tipo i de instr.	ciclos	NI_i	Proporción (%)	Comentarios
BRANCH en ALT1	4	$0{,}2 \times NI^1$	20	
RESTO en ALT1	3	$0{,}8 \times NI^1$	80	Incluye instrucciones COMPARE
TOTAL		**NI^1**	**100**	**NI^1 : nº instr. en ALT1**

Datos del ejercicio:

$$T^1_{ciclo} = T^2_{ciclo} - 0{,}25 \times T^2_{ciclo} = 0{,}75 \times T^2_{ciclo}$$

ALT1

Tipo i de instr.	ciclos	NI_i	Proporción (%)	Comentarios
BRANCH en ALT1	4	$0{,}2 \times NI^1$	20	
RESTO en ALT1	3	$0{,}8 \times NI^1$	80	Incluye instrucciones COMPARE
TOTAL		**NI^1**	**100**	**NI^1 : nº instr. en ALT1**

ALT2

Tipo i de instr.	ciclos	NI_i	Proporción (%)	Comentarios
COMPARE+BRANCH en ALT2	4	$0{,}2 \times NI^1$	$20 \times 100/80 = 25$	ALT2 no tendrá aparte las instrucciones COMPARE, que son un 20% en ALT1
RESTO en ALT2	3	$0{,}6 \times NI^1$	$(80-20) \times 100/80 = 75$	ALT2 no tendrá aparte las instrucciones COMPARE, que son un 20% en ALT1
TOTAL		**$NI^2 = 0{,}8 \times NI^1$**	**100**	Se reducen el nº de instr. en ALT2, **NI^2,** al no tener instr. COMPARE aparte

- Para la alternativa primera, *ALT1*, se tiene:

$$T^1_{CPU} = NI^1 \times CPI^1 \times T^1_{ciclo} = NI^1 \times (0.2 \times 4 + 0.8 \times 3) \times T^1_{ciclo} = NI^1 \times 3.2 \times 0.75 \times T^2_{ciclo} = NI^1 \times 2.4 \times T^2_{ciclo}$$

(CPI para ALT1 es $CPI^1 = 0.2 \times 4 + 0.8 \times 3 = 3.2$)

- Para la alternativa segunda, *ALT2*, se tiene:

$$T^2_{CPU} = NI^2 \times CPI^2 \times T^2_{ciclo} = NI^1 \times (0.2 \times 4 + 0.6 \times 3) \times T^2_{ciclo} = NI^1 \times 2.6 \times T^2_{ciclo}$$

(CPI para ALT2 es $CPI^2 = \frac{0.2 \times NI^1 \times 4 + 0.6 \times NI^1 \times 3}{NI^2} = (0.2 \times 4 + 0.6 \times 3) \times \frac{NI^1}{NI^2} = \frac{0.8 + 1.8}{0.8} = \frac{2.6}{0.8} = 3.25$)

Ejercicio 4. ¿Qué ocurriría en el problema anterior si el ciclo de reloj fuese únicamente un 10% mayor para la ALT2?

Solución

En este caso $T^2_{CICLO} = 1.10 * T^1_{CICLO}$; por tanto:

$$T^1_{CPU} = NI^1 \times CPI^1 \times T^1_{ciclo} = NI^1 \times (0.2 \times 4 + 0.8 \times 3) \times T^1_{ciclo} = NI^1 \times 3.2 \times T^1_{ciclo}$$

$$T^2_{CPU} = NI^2 \times CPI^2 \times T^2_{ciclo} = NI^1 \times (0.2 \times 4 + 0.6 \times 3) \times 1.1 \times T^1_{ciclo} = NI^1 \times 2.86 \times T^1_{ciclo}$$

Ejercicio 5. Considere un procesador no segmentado con una arquitectura de tipo LOAD/STORE en la que las operaciones sólo 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% BRANCHs (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. Tengan 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 BRANCHs, pero no afectan al ciclo de reloj.

Alternativa 1

I_i^1	CPI_i^1	NI_i^1	Comentarios
LOAD	4	$0{,}21 \times NI^1$	
STORE	4	$0{,}12 \times NI^1$	
ALU	3	$0{,}43 \times NI^1$	25 % inst. que usan opernados en registros que no se vuelven a utilizar
BR	4	$0{,}24 \times NI^1$	
TOTAL		NI^1	

Alternativa 1

I_i^1	CPI_i^1	NI_i^1	Com
LOAD	4	$0,21 \times NI^1$	
STORE	4	$0,12 \times NI^1$	
ALU	3	$0,43 \times NI^1$	25 % inst. que usan opernados en registros que no se vuelven a utilizar
BR	4	$0,24 \times NI^1$	
TOTAL		NI^1	

Alternativa 2

I_i^2	CPI_i^2	NI_i^2	**Comentarios**
LOAD	4	$(0,21-0,25 \times 0,43) \times NI^1 = 0,1025 \times NI^1$	El 25 % de 43% desaparece al usarse ese mismo número de operaciones con la ALU que acceden a memoria
STORE	4	$0,12 \times NI^1$	
ALU r-r	3	$0,75 \times 0,43 \times NI^1 = 0,3225 \times NI^1$	
ALU r-m	4	$0,25 \times 0,43 \times NI^1 = 0,1075 \times NI^1$	25% de las operaciones con la ALU; es decir el 25% del 43%
BR	5	$0,24 \times NI^1$	
TOTAL		$\mathbf{NI^2 = 0,8925 \times NI^1}$	**Es decir, NI^2 es igual a $0,8925 \times NI^1$**

$$T^1_{CPU} = CPI^1 \times NI^1 \times T_{ciclo} = (0.43 \times 3 + 0.21 \times 4 + 0.12 \times 4 + 0.24 \times 4) \times NI^1 \times T_{ciclo} = (1.29 + 2.28) \times NI^1 \times T_{ciclo} = 3.57 \times NI^1 \times T_{ciclo}$$

(CPI para ALT1 es $CPI^1 = 0.43 \times 3 + 0.21 \times 4 + 0.12 \times 4 + 0.24 \times 4 = 1.29 + 0.57 \times 4 = 1.29 + 2.28 = 3.57$)

$$T^2_{CPU} = CPI^2 \times NI^2 \times T_{ciclo} = [0.3225 \times 3 + (0.12 + 0{,}1025 + 0{,}1075) \times 4 + 5 \times 0{,}24] \times NI^1 \times T_{ciclo} = (0{,}9675 + 0{,}33 \times 4 + 0{,}24 \times 5) \times NI^1 \times T_{ciclo} = 3.4875 \times NI^1 \times T_{ciclo}$$

Como se puede ver, en este caso, $T^1_{CPU} > T^2_{CPU}$, y por lo tanto se mejoran las prestaciones, pero si el porcentaje de instrucciones sustituidas fuese un 20% en lugar de un 25%, en ese caso, se puede ver que

$$T^2_{CPU} = CPI^2 \times NI^2 \times T^2_{ciclo} = 3.59 \times NI^2 \times T_{ciclo}$$

Ahora, en cambio, la segunda opción no mejora la primera. Como conclusión, se puede indicar que una determinada decisión de diseño puede suponer una mejora en el rendimiento del computador correspondiente según sean las características de las distribuciones de instrucciones en los programas que constituyen la carga de trabajo característica del computador.

Ejercicio 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 BRANCHs. 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.

Solución

Datos del ejercicio:

Alternativa 1

I_i^1	CPI_i^1	NI_i^1/NI^1	Comentarios
LOAD	4	0,21	
STORE	4	0,12	
ALU	3	0,43	
BR	4	0,24	
TOTAL		**1**	

Alternativa 2

I_i^2	CPI_i^2	NI_i^2/NI^1	Comentarios
LOAD	4	0,21	
STORE	4	0,12	
ALU r-r	3	0,5 * 0,43 = 0,215	Se reducen las instrucciones que usan la ALU en un 50%
BR	4	0,24	
TOTAL		**0,785**	**Es decir, $NI^2 = 0{,}785 * NI^1$**

En la situación inicial del problema anterior se tenía que

$$T^1_{CPU} = CPI * NI^1 * T_{ciclo} = 3.57 * NI^1 * T_{ciclo}$$

y, por lo tanto

$$MIPS^1 = \frac{NI^1}{T^1_{CPU} * 10^6} = \frac{NI^1}{CPI * NI^1 * Tciclo * 10^6} = \frac{f(hz)}{CPI * 10^6} = \frac{50 * 10^6\ c/s}{3.57\ c/i\ * 10^6} = \mathbf{14.005\ MIPS}$$

El tiempo de CPU para la alternativa 2 es:

$$T^2_{CPU} = CPI^2 * NI^2 * T_{ciclo} = [\ 0.215*3 + (0.21+0.12+0.24)*4\ ] * NI^1 * T_{ciclo} = (0.645+0.57*4) * NI^1 * T_{ciclo} = (0.645+2.28) * NI^1 * T_{ciclo} = 2.925 * NI^1 * T_{ciclo}$$

$$T^2_{CPU} = 2.925 * \frac{NI^1}{NI^2} * NI^2 * T_{ciclo} = 2.925 * \frac{NI^1}{NI^2} * NI^2 * T_{ciclo} = 3.726 * NI^2 * T_{ciclo}$$

$$MIPS^2 = \frac{NI^2}{T^2_{CPU} * 10^6} = \frac{NI^2}{CPI * NI^2 * Tciclo * 10^6} = \frac{f(hz)}{CPI * 10^6} = \frac{50 * 10^6\ c/s}{3.726\ c/i\ * 10^6} = \mathbf{13.42\ MIPS}$$

Como se puede ver, se consigue una reducción de tiempo de ejecución (T^2_{CPU} es menor que T^1_{CPU}), 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 claro, 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 número menor de instrucciones, y al final, el tiempo se reduce.

Ejercicio 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. **(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? Razone su respuesta.

Datos del ejercicio:

Alternativa 1

I_i^1	CPI_i^1	NI_i/NI	Comentarios
LOAD	4	0,2	
STORE	3	0,15	
ALU	6	0,4	4 ciclos en la ALU
BR	3	0,25	
TOTAL		**1**	

Alternativa 2

I_i^2	CPI_i^2	NI_i/NI	Comentarios
LOAD	4	0,2	
STORE	3	0,15	
ALU	2+2=4	0,4	Se reducen en 2 ciclos lo que consumen en la ALU
BR	3	0,25	
TOTAL		**1**	

$$T^1_{CPU} = NI*(0.2*4 + 0.15*3 + 0.4*6 + 0.25*3)*T_c = NI * 4.4 * T_c$$

$$T^2_{CPU} = NI*(0.2*4 + 0.15*3 + 0.4*4 + 0.25*3)*T_c = NI * 3.6 * T_c$$

De esta forma, la ganancia de velocidad sería:

$$S = T^1_{CPU} / T^2_{CPU} = 4.4/3.6 = \mathbf{1.22}$$

(b) La ganancia en prestaciones conseguida en una instrucción de la ALU es de 6 ciclos/4 ciclos=1.5. Aunque todas las intrucciones 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:

$$T^2_{CPU} = 4 \text{ ciclos} * NI$$

$$T^1_{CPU} = 6 \text{ ciclos} * NI$$

$$S = T^1_{CPU} / T^2_{CPU} = 1.5$$

Ejercicio 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.

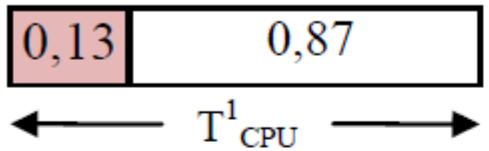
(a) Ha aparecido en el mercado una nueva versión del procesador en la que la única mejora con respecto a la versión anterior es una nueva unidad de coma flotante que permite reducir el tiempo de las instrucciones de coma flotante a tres cuartas partes del tiempo que consumían antes. ¿Cuál es la máxima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

(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?

(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 en lugar de la obtenida en el apartado (b)?

(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 porcetaje de tiempo de cálculo con como flotante es el obtenido en **(c)**?

Datos del ejercicio:



(a) Resolución 1:

$$T^2_{CPU} = 0.13 \times \frac{3}{4} \times T^1_{CPU} + 0.87 \times T^1_{CPU} = 0.0975 \times T^1_{CPU} + 0.87 \times T^1_{CPU} = 0.9675 \times T^1_{CPU}$$

$$S \leq \frac{T^1_{CPU}}{T^2_{CPU}} = \frac{T^1_{CPU}}{0.9675 \times T^1_{CPU}} \cong \mathbf{1.0336}$$

(a) Resolución 2 (se usa la expresión que caracteriza la ley de Amdahl):

Como el tiempo de las instrucciones de coma flotante se reduce tres cuartas partes, la mejora de velocidad es *p=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, y por tanto, f =0.87

Sustituyendo estos valores en la expresión de la ley de Amdahl se tiene:

$$S \leq \frac{p}{1 + f \times (p - 1)} = \frac{4/3}{1 + 0.87 \times (4/3 - 1)} = \frac{4/3}{1 + 0.87 \times 1/3} = \frac{4}{3 + 0.87} \cong \mathbf{1.0336}$$

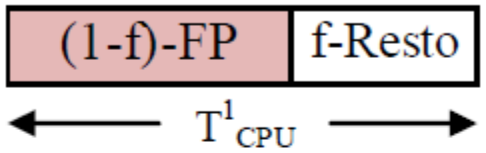
(b) Resolución 1: La ganancia será mejor cuanto menor sea el tiempo que supongan las operaciones FP. La mayor ganancia se conseguiría haciendo ese tiempo despreciable, es decir, si la mejora de las unidades FP se hace muy muy grande, infinito. En este caso:

$$T^2_{CPU} = 0 + 0.87 \times T^1_{CPU} = 0.87 \times T^1_{CPU} \qquad S \leq \frac{T^1_{CPU}}{T^2_{CPU}} = \frac{T^1_{CPU}}{0.87 \times T^1_{CPU}} = \frac{1}{0.87} \cong \mathbf{1.149}$$

(b) Resolución 2 (se usa la expresión que caracteriza la ley de Amdahl): Para determinar el valor de la ganancia máxima que se pide, se obtiene el límite cuando la mejora del recurso (instrucciones en coma flotante) tiende a infinito:

$$S_{max(f=cte)} \leq \lim_{p \to \infty} \frac{p}{1 + f \times (p - 1)} = \frac{1}{f} = \frac{1}{0.87} \cong \mathbf{1.149}$$

(c) Resolución 1:



Si se cambia la fracción de código con FP (la notamos por (1-f)) y tenemos en cuenta que para obtener la mayor ganancia se debe hacer el tiempo de ejecución de las FP igual a 0, entonces:

$$T^2_{CPU} = 0 + f \times T^1_{CPU} \qquad S \leq \frac{T^1_{CPU}}{T^2_{CPU}} = \frac{T^1_{CPU}}{f \times T^1_{CPU}} = \frac{1}{f} = 4 \Rightarrow f = 0.25 \text{ y } (1-f) = \mathbf{0.75}$$

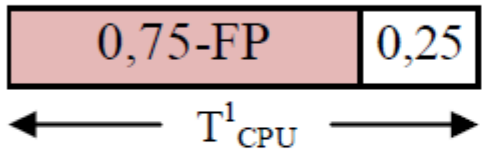
f debería ser al menos 0,25 y, por tanto, 1-f=0,75; es decir,las operaciones FP deberían ser un 75%.

(c) Resolución 2 (se usa la expresión que caracteriza 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 se obtendría a partir de:

$$S_{max(f=cte)} \leq \frac{1}{f} = 4$$

Por lo que *f=1/4=0.25*, y la fracción de tiempo que tendría que poder reducirse con la mejora tendría que ser *(1-f)=1-0.25=**0.75***.

(d) Resolución 1:



$$T^2_{CPU} = 0.75 \times \frac{1}{p} \times T^1_{CPU} + 0.25 \times T^1_{CPU}$$

$$S \leq \frac{T^1_{CPU}}{T^2_{CPU}} = \frac{T^1_{CPU}}{0.75 \times \frac{1}{p} \times T^1_{CPU} + 0.25 \times T^1_{CPU}} = \frac{1}{0.75 \times \frac{1}{p} + 0.25} = 2 \Rightarrow p = \frac{1.5}{0.5} = 3$$

El tiempo se debería reducir a 1/3 del tiempo original. Es decir, las instrucciones de coma flotante deberían hacerse tres veces más rápidas.

(d) Resolución 2 (se usa la expresión que caracteriza la ley de Amdahl): Si, al utilizar el valor de *f* anterior se deseara obtener una mejora de velocidad de 2 en los programas, la mejora de velocidad en las instrucciones de coma flotante (es decir *p*) se puede obtener a partir de la ley de Amdahl:

$$S \leq 2 = \frac{p}{1 + f \times (p - 1)} = \frac{1}{f + \frac{f - 1}{p}} = \frac{1}{25 + \frac{0.75}{p}} \Rightarrow p = \frac{1.5}{0.5} = \mathbf{3}$$

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

Ejercicio 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])*b;
```

(a) ¿Cuántos GFLOPS se necesitan para poder ejecutar el código en menos de 0,5 segundos?

(b) Suponiendo que este código en ensamblador tiene `7N` instrucciones y que se ha ejecutado en un procesador de 32 bits a 2 GHz. ¿Cual es el número medio de instrucciones que el procesador tiene que poder completar por ciclo para poder ejecutar el código en menos de 0,5 segundos?

(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 redujesen un 75% los tiempos de las unidades de coma flotante?

(a) Puesto que hay tres operaciones en coma flotante por iteración (dos sumas y un producto, con igual coste para el producto y la suma) el número de operaciones en coma flotante a realizar tras las N iteraciones es $N_{flot}=3\times N = 3\times 10^9$.

Por lo tanto, GFLOPS = $N_{flot}/(t\times 10^9)$ donde t es el tiempo en segundos. Así:

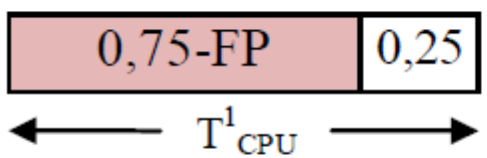
$$\text{GFLOPS} = N_{flot}/(t\times 10^9) = 3\times 10^9/0.5\times 10^9 = 6 \text{ GFLOPS}$$

(b) Si el programa tiene NI = 7N = 7 10^9 instrucciones, teniendo en cuenta que

$$T = NI\times CPI\times Tciclo = NI\times CPI/f = (7\ 10^9 \text{ instrucciones}) \times CPI\ /\ (2\times 10^9 \text{ ciclos/s}) < 0.5 \text{ s}$$

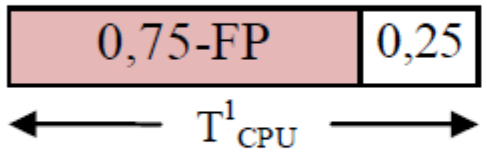
Al despejar CPI < 0.5/3.5, y el número de instrucciones por ciclo IPC = 1/CPI > 7

(c) <u>Resolución 1:</u>



$$T^2_{CPU} = 0.25 \times T^1_{CPU} + 0.25 \times 0.75 \times T^1_{CPU} = 0.25 \times T^1_{CPU} \times (1 + 0.75)$$

(c) Resolución 1:



$$T^2_{CPU} = 0.25 \times T^1_{CPU} + 0.25 \times 0.75 \times T^1_{CPU} = 0.25 \times T^1_{CPU} \times (1 + 0.75)$$

$$S \leq \frac{T^1_{CPU}}{T^1_{CPU}} - \frac{T^2_{CPU}}{T^1_{CPU}} = 1 - 0.25 \times 1.75 = 1 - 0.4375 = 0.5625 \Rightarrow 56.25\%$$

Por tanto, el tiempo se reduce un 56.25%

Resolución 2: Para resolver la última cuestión se puede recurrir a 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=1/0.25 (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, sustituyendo convenientemente:

$$S \leq p\,/(1+f(p-1)) = 4\,/\,(1+0.25\times3) = 2.286$$

Como $S = T_{sin\_mejora}/T_{con\_mejora} \leq 2.286$ se tiene que $T_{sin\_mejora}/2.286 = 0.4375*T_{sin\_mejora} \leq T_{con\_mejora}$. 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%.