Medida del Rendimiento de un Bucle Vectorizado

Se va a realizar un ejemplo del análisis del rendimiento de un procesador vectorial al ejecutar el código obtenido de vectorizar el conocido bucle DAXPY para vectores de ***n*** elementos.

El procesador vectorial consta de:

- Una unidad de suma (6 ciclos de latencia).
- Una unidad de multiplicación (7 ciclos de latencia).
- Una unidad de carga/almacenamiento (12 ciclos de latencia).
- MVL es 64.
- La frecuencia de reloj es 500 MHz.

El fragmento de código vectorial que se genera para realizar las operaciones Y(i) : =a·X(i)+Y(i) es:

```
LV      V1,R1
MULTSV  V2,V1,F0
LV      V3,R2
ADDV    V3,V3,V2
SV      R2,V4
```

y los costes debidos a las instrucciones escalares son T_{base} = 10 ciclos y T_{bucle} = 15 ciclos. Inicialmente, se considera que solo se emite un nuevo convoy cuando las instrucciones del convoy anterior han terminado.

Caso 1:

Sin Encadenamiento de

Resultados entre Unidades

Analizando los riesgos estructurales y de datos existentes entre las cinco instrucciones vectoriales, los convoyes existentes son cuatro:

Convoy 1:	LV	V1,R1
Convoy 2:	MULTSV LV	V2,V1,F0 V3,R2
Convoy 3:	ADDV	V4,V3,V2
Convoy 4:	SV	R2,V4

La secuencia de ejecución de los cuatro convoyes, si se considera que VLR es 64, es la que se muestra en la Figura 3.34.

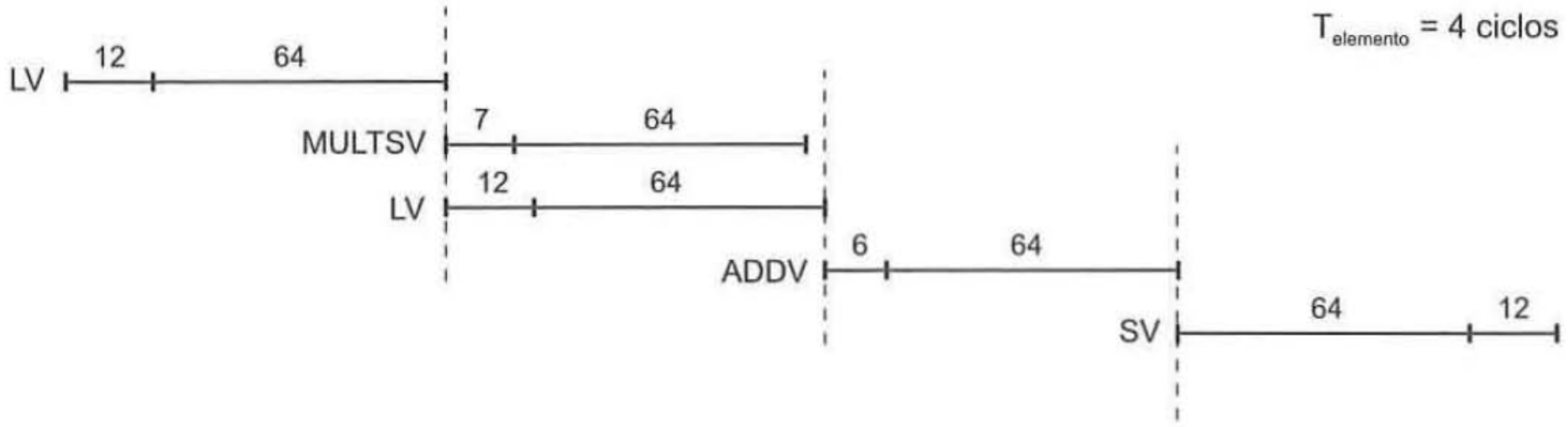

Figura 3.34: Secuencia de ejecución de los cuatro convoyes con VLR = 64.

Dado que hay cuatro convoyes, $T_{elemento}$ es 4 ciclos y el $T_{arranque}$ total es igual a la suma de los tiempos de arranque visibles de los cuatro convoyes. Esto es:

$$T_{arranque} = 2 \cdot T_{arranque}\ LV + T_{arranque}\ ADDV + T_{arranque}\ SV$$

$$T_{arranque} = (2 \cdot 12 + 6 + 12)\ ciclos = 42\ ciclos$$

Sustituyendo los valores conocidos de $T_{arranque}$ y $T_{elemento}$ en la expresión que determina el tiempo de ejecución de un bucle vectorizado para vectores de longitud ***n*** se tiene

$$T_n = T_{base} + \left\lceil \frac{n}{64} \right\rceil \cdot \left(T_{bucle} + T_{arranque} \right) + n \cdot T_{elemento}$$

$$T_n = 10 + \left\lceil \frac{n}{64} \right\rceil \cdot \left(15 + 42 \right) + 4n$$

que para el caso particular de n = 1000

$$T_{1000} = 10 + \left\lceil \frac{1000}{64} \right\rceil \cdot (15 + 42) + 4 \cdot 1000 =$$

$$T_{1000} = 10 + 16 \cdot 57 + 4000 = 4922 \ ciclos$$

El rendimiento expresado en FLOP/ciclo es:

$$R_\infty = \lim_{n\to\infty}\left(\frac{2n}{T_n}\right)$$

$$R_\infty = \lim_{n\to\infty}\left(\frac{2n}{10+\left\lceil\frac{n}{64}\right\rceil\cdot(15+42)+4n}\right)$$

Para simplificar los cálculos, la expresión $\left\lceil \frac{n}{64} \right\rceil$ se puede reemplazar por una cota superior dada por (n/64 + 1). Sustituyendo esta cota en R_∞ y teniendo en cuenta que el número de operaciones vectoriales que se realizan en el bucle DAXPY son dos, una multiplicación y una suma, se tiene

$$R_\infty = \lim_{n\to\infty}\left(\frac{2n}{10+\left(\frac{n}{64}+1\right)\cdot 57+4n}\right) =$$

$$R_\infty = \lim_{n\to\infty}\left(\frac{2n}{67+4,89\cdot n}\right) = 0,409\ FLOP/ciclo$$

Para expresar R_∞ en FLOPS ha que multiplicar el valor anterior por la frecuencia del procesador. Se tiene así

$$R_\infty = 0{,}409 \text{ FLOP/ciclo} \cdot 500 \text{ MHz}$$

$$R_\infty = 204{,}5 \text{ MFLOPS}$$

Caso 2:

Con Encadenamiento de

Resultados entre Unidades

Dado que ahora es posible encadenar los resultados de las unidades, la organización del código vectorial en convoyes quedaría de la siguiente forma:

Convoy 1:	LV MULTSV	V1,R1 V2,V1,F0
Convoy 2:	LV ADDV	V3,R2 V4,V3,V2
Convoy 3:	SV	R2,V4

El tercer convoy se mantiene debido a que no se permite que en un mismo convoy haya instrucciones que presenten riesgos estructurales. Por esa razón, en el segundo convoy no pueden coexistir dos instrucciones con acceso a la única unidad de

carga/almacenamiento disponible. La secuencia de ejecución de los tres convoyes se muestra en la Figura 3.35.

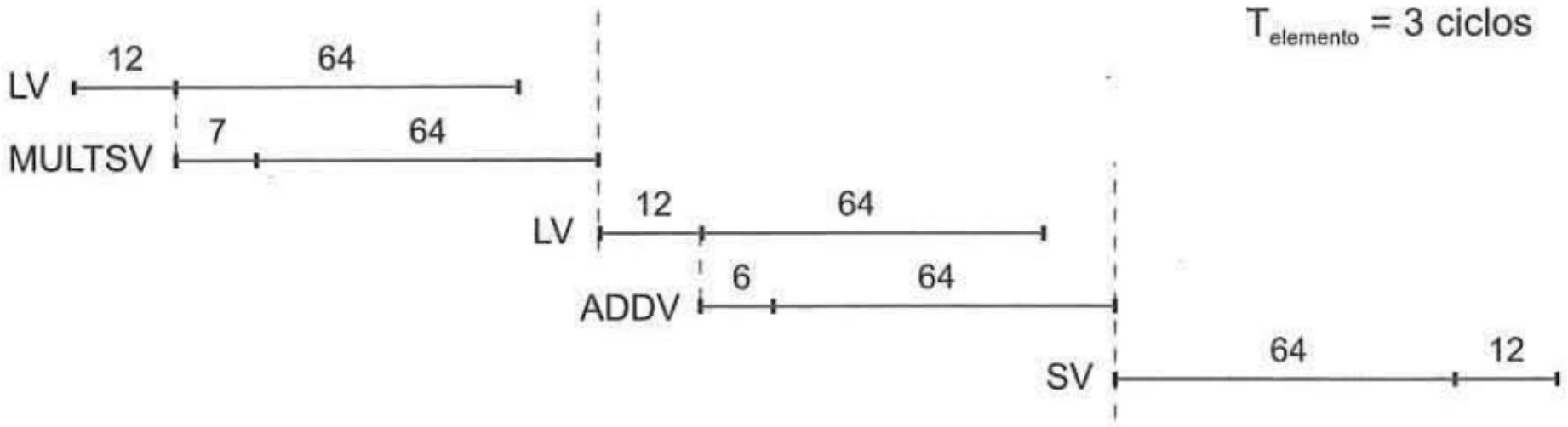

Figura 3.35: Secuencia de ejecución de los tres convoyes con VLR = 64

El $T_{elemento}$ ha pasado a ser de 3 ciclos dado que ahora se tienen 3 convoyes. El $T_{arranque}$ total se obtiene de sumar los tiempos de arranque visibles de las unidades funcionales. Si se analiza la Figura 3.35 se tiene

$$T_{arranque} = 2 \cdot T_{arranque}\ LV + T_{arranque}\ MULTSV + T_{arranque}$$

$$ADDV + + T_{arranque}\ SV$$

$$T_{arranque} = (2 \cdot 12 + 7 + 6 + 12)\ ciclos = 49\ ciclos$$

Con estos valores la expresión del tiempo total de ejecución queda

$$T_n = T_{base} + \left\lceil \frac{n}{64} \right\rceil \cdot \left(T_{bucle} + T_{arranque} \right) + n \cdot T_{elemento}$$

$$T_n = 10 + \left\lceil \frac{n}{64} \right\rceil \cdot \left(15 + 49 \right) + 3n$$

que para el caso particular de n = 1000

$$T_{1000} = 10 + \left\lceil \frac{1000}{64} \right\rceil \cdot (15 + 49) + 3 \cdot 1000 =$$

$$T_{1000} = 10 + 16 \cdot 64 + 3000 = 4034 \; ciclos$$

Con respecto al caso 1, el permitir encadenamiento de resultados entre unidades funcionales ha reducido el tiempo de ejecución un 18%, pasando de 4922 a 4034 ciclos. En lo que respecta al rendimiento expresado en FLOP/ciclo

$$R_{\infty} = \lim_{n\to\infty}\left(\frac{2n}{T_n}\right) =$$

$$\lim_{n\to\infty}\left(\frac{2n}{10+\left\lceil\frac{n}{64}\right\rceil\cdot(15+49)+3n}\right) = \lim_{n\to\infty}\left(\frac{2n}{10+\left(\frac{n}{64}+1\right)\cdot 64+3n}\right) =$$

$$= \lim_{n\to\infty}\left(\frac{2n}{74+4\cdot n}\right) = 0,5\ FLOP\ /\ ciclo$$

Para expresar R_∞ en FLOPS ha que multiplicar el valor anterior por la frecuencia del procesador. Se tiene así

$$R_\infty = 0{,}5 \text{ FLOP/ciclo} \cdot 500 \text{ MHz}$$

$$R_\infty = 250 \text{ MFLOPS}$$

Claramente se aprecia la mejora en el rendimiento del procesador gracias al encadenamiento de los resultados entre las unidades funcionales.

Caso 3:

Con Encadenamiento y Dos Unidades de Carga/Almacenamiento

Si se permite el encadenamiento y el número de unidades de carga/almacenamiento se duplica se obtiene la siguiente secuencia de convoyes:

Convoy 1:	LV MULTSV LV ADDV	V1,R1 V2,V1,F0 V3,R2 V4,V3,V2
Convoy 2:	SV	R2,V4

Ahora el primer y el segundo convoy del caso 2 se pueden unir gracias a que la existencia de una segunda unidad de acceso a memoria permite colocar dos instrucciones de carga en el mismo convoy sin que existan riesgos estructurales.

La Figura 3.36 muestra la secuencia de ejecución.

Dado que hay dos convoyes, el $T_{elemento}$ es de 2 ciclos y del análisis de la Figura 3.36 se obtiene que el tiempo de arranque total de las unidades funcionales es

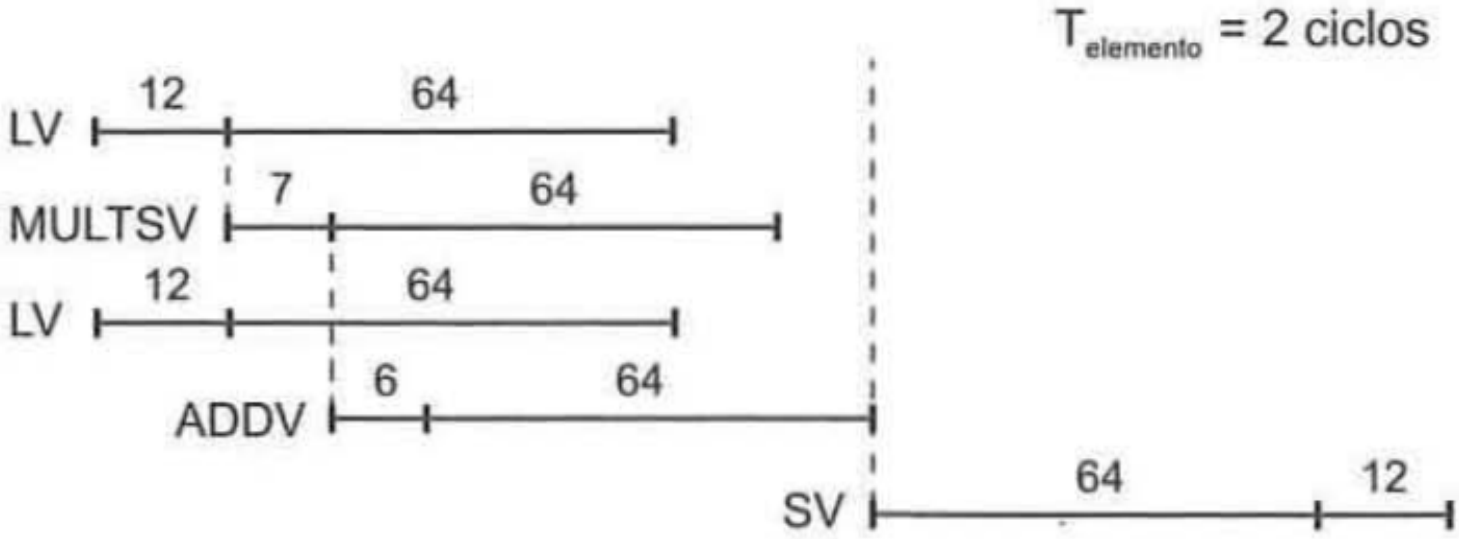

Figura 3.36: Secuencia de ejecución de los dos convoyes con VLR = 64.

El tiempo de arranque total de las unidades funcionales es:

$$T_{arranque} = T_{arranque}\ \text{LV} + T_{arranque}\ \text{MULTSV} + T_{arranque}\ \text{ADDV}$$

$$+ + T_{arranque}\ \text{SV}$$

$$T_{arranque} = (12 + 7 + 6 + 12)\ \text{ciclos} = 37\ \text{ciclos}$$

Se tiene ahora:

$$T_n = T_{base} + \left\lceil \frac{n}{64} \right\rceil \cdot \left(T_{bucle} + T_{arranque} \right) + n \cdot T_{elemento}$$

$$T_n = 10 + \left\lceil \frac{n}{64} \right\rceil \cdot \left(15 + 37 \right) + 2n$$

que para el caso particular de n = 1000

$$T_{1000} = 10 + \left\lceil \frac{1000}{64} \right\rceil \cdot (15 + 37) + 2 \cdot 1000 =$$

$$T_{1000} = 10 + 16 \cdot 52 + 2000 = 2842 \; ciclos$$

Con respecto al caso 1 la mejora es del 73 %, ya que el total de ciclos consumidos para procesar el bucle DAXPY con 1000 elementos se ha reducido de 4922 a 2842. El rendimiento en FLOP/ciclo es

$$R_{\infty} = \lim_{n \to \infty} \left(\frac{2n}{T_n} \right) =$$

$$\lim_{n \to \infty} \left(\frac{2n}{10 + \left\lceil \frac{n}{64} \right\rceil \cdot (15 + 37) + 2n} \right) = \lim_{n \to \infty} \left(\frac{2n}{10 + \left(\frac{n}{64} + 1 \right) \cdot 52 + 2n} \right) =$$

$$= \lim_{n \to \infty} \left(\frac{2n}{62 + 2{,}8125 \cdot n} \right) = 0{,}711 \; FLOP / ciclo$$

Para expresar R_∞ en FLOPS ha que multiplicar el valor anterior por la frecuencia del procesador. Se tiene así

$$R_\infty = 0{,}711 \text{ FLOP/ciclo} \cdot 500 \text{ MHz}$$

$$R_\infty = 355{,}55 \text{ MFLOPS}$$

Caso 4:

Con Encadenamiento,

Dos Unidades de Carga/Almacenamiento y Solapamiento entre Convoyes dentro de la misma iteración

El último caso contempla que se puedan solapar convoyes en una misma iteración del bucle permitiendo así que una unidad funcional pueda ser utilizada por otra instrucción antes de que abandone la unidad la instrucción actual: Además, se considerará que las instrucciones escalares se pueden ejecutar en paralelo a las instrucciones vectoriales, por lo que T_{bucle} desaparece al quedar oculto tras los tiempos de ejecución de las instrucciones vectoriales.

En el bucle DAXPY, el número de convoyes continúa siendo dos. Sin embargo, es posible solapar en una unidad de carga/almacenamiento una de las dos instrucciones LV del primer convoy y la instrucción SV del segundo convoy. De esta forma, la ejecución de los dos convoyes se superpone dando lugar a un $T_{elemento}$ inferior al número de convoyes. En el diagrama de la Figura 3.37 se aprecia esta situación.

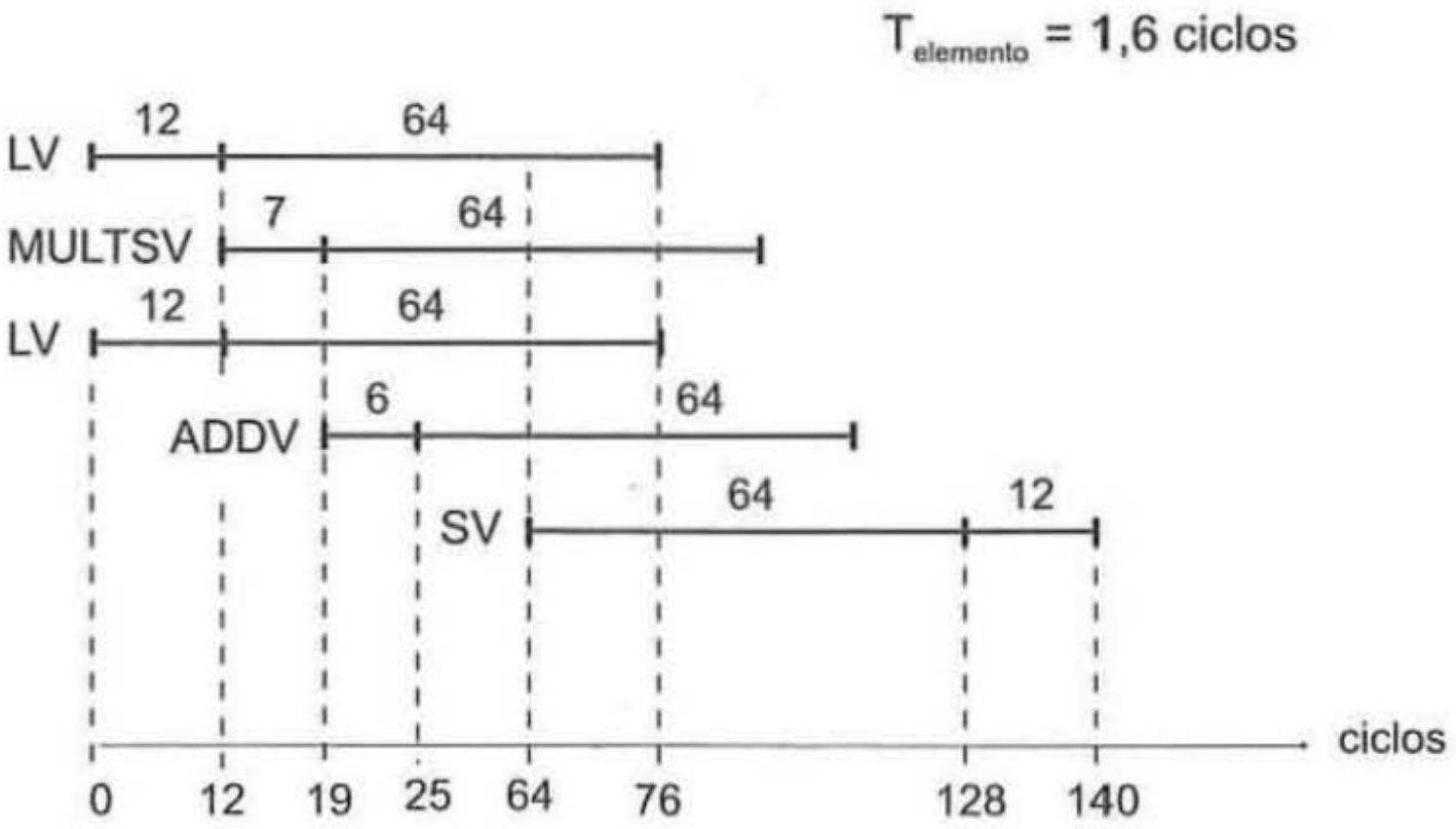

Figura 3.37: Secuencia de ejecución de los dos convoyes con VLR = 64.

Para calcular el $T_{elemento}$ en situaciones con solapamiento, hay que recordar que para un fragmento de código vectorial se tiene que

$$T_{elemento} = (T_n - T_{arranque}) / \boldsymbol{n}$$

Dado que para un valor de VL de 64, el tiempo de ejecución de las instrucciones vectoriales en este caso es de 140 ciclos (Figura 3.37) y los tiempos de arranque son

$$T_{arranque} = T_{arranque}\ \text{LV} + T_{arranque}\ \text{MULTSV} + T_{arranque}\ \text{ADDV}$$

$$+ \ \ + T_{arranque}\ \text{SV}$$

$$T_{arranque} = (12 + 7 + 6 + 12) \text{ ciclos} = 37 \text{ ciclos}$$

Se tiene que:

$$T_{elemento} = (T_{64} - T_{arranque}) / \mathbf{64}$$

$$T_{elemento} = (140 - 37) / \mathbf{64}$$

$$T_{elemento} = 1{,}6 \text{ ciclos}$$

Como T_{bucle} es cero dado su solapamiento con el código vectorial, el tiempo de ejecución del bucle vectorizado para ***n*** elementos es

$$T_n = T_{base} + \left\lceil \frac{n}{64} \right\rceil \cdot \left(T_{arranque} \right) + n \cdot T_{elemento}$$

$$T_n = 10 + \left\lceil \frac{n}{64} \right\rceil \cdot 37 + 1,6 \cdot n$$

y particularizando para un vector de 1000 elementos

$$T_{1000} = 10 + \left\lceil \frac{1000}{64} \right\rceil \cdot 37 + 1,6 \cdot 1000 =$$

$$T_{1000} = 10 + 16 \cdot 37 + 1600 = 2202 \; ciclos$$

La mejora que se obtiene con respecto al caso 1 es del 123 % al reducirse el total de ciclos consumidos de 4922 a 2202. El rendimiento en FLOP/ciclo es

$$R_\infty = \lim_{n\to\infty}\left(\frac{2n}{T_n}\right) =$$

$$\lim_{n\to\infty}\left(\frac{2n}{10+\left\lceil\frac{n}{64}\right\rceil\cdot(37)+1,6\cdot n}\right) = \lim_{n\to\infty}\left(\frac{2n}{10+\left(\frac{n}{64}+1\right)\cdot 37+1,6\cdot n}\right) =$$

$$= \lim_{n\to\infty}\left(\frac{2n}{47+2,18\cdot n}\right) = 0,918\ FLOP/ciclo$$

Para expresar R_∞ en FLOPS ha que multiplicar el valor anterior por la frecuencia del procesador.

$$R_\infty = 0{,}918 \text{ FLOP/ciclo} \cdot 500 \text{ MHz}$$

$$R_\infty = 459 \text{ MFLOPS}$$