Universidad del País Vasco
Facultad de Informática

Departamento de Arquitectura y Tecnología de Computadores

Arquitectura de Computadores I

Sistema de memoria 5 (solución): Paginación + Bancos + Tiempo de acceso total

Las características del sistema de memoria de un computador son las siguientes:

- La unidad de direccionamiento es el byte y las palabras son de 8 bytes.
- **Memoria virtual:** memoria paginada de 1 MB. Las páginas son de 1 kB y se utiliza un TLB para la traducción de direcciones. El tiempo de acceso al TLB es de 1 ciclo en caso de acierto y 20 ciclos, en caso de fallo. Al principio el TLB está vacío.
- **Memoria principal** de 16 kB, formada por 2 bancos con 4 módulos entrelazados cada uno. El tiempo de acceso es de 10 ciclos (1 desde el buffer de entrelazado).

Se pide:

a) Los esquemas de las direcciones lógicas y físicas, indicando cuántos bits ocupa cada campo. ¿Cuántas páginas puede tener como máximo un programa? ¿Cuántas páginas tiene la memoria principal?

b) En cuanto a memoria principal, ¿cuál es el tamaño de un banco en bytes? ¿Cuál es el tamaño de un módulo en bytes?

c) En este computador se ejecuta el siguiente programa:

```
for (i=0;i<200;i++)
      B[0]=B[0]+A[i]*A[i+1];
```

```
        movi r1,#0
        movi r2,#199
        load r5,B[r1]
bucle:  load r3,A[r1]
        load r4,A[r1+8]
        mul r4,r3,r4
        add r5,r4,r5
        addi r1,r1,#8
        subi r2,r2,#1
        bge r2,bucle
        store r5,B
```

El programa está almacenado a partir de la dirección lógica 0. El vector A comienza en la dirección lógica 8192 y el vector B, en la dirección lógica 10000. Tanto las instrucciones como los datos ocupan una palabra. Calcula el tiempo necesario para acceder a memoria (tiempo de traducción y tiempo de acceso) para las direcciones lógicas generadas en la primera pasada del bucle. Para ello explica claramente con una tabla los pasos empleados para calcular los tiempos de acceso. El contenido de la tabla de páginas es el siguiente:

Página lógica:	0	14	8	9	....
Página física:	2	6	10	11	....

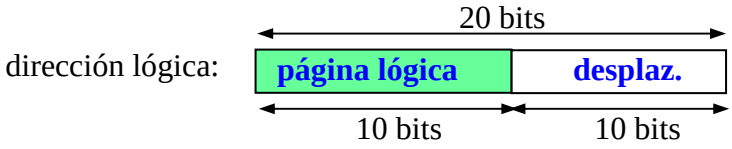
d) Calcula el tiempo de acceso al sistema de memoria (tiempo de traducción + tiempo de acceso) para las direcciones generadas en la ejecución de todo el programa.

Solución

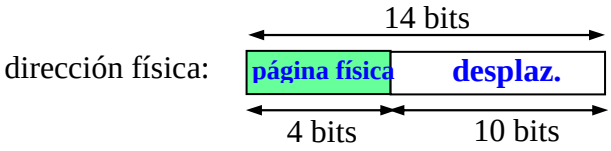
En primera lugar, hay que analizar en profundidad las características del sistema de memoria que proporciona el enunciado para inferir toda la información necesaria. Así, tenemos que tener en cuenta que el sistema de memoria utiliza **palabras de 8 bytes** y que la **unidad de direccionamiento es el byte**. Analicemos ahora lo que se nos pide en cada apartado.

(a) Los esquemas de las direcciones lógicas y físicas, indicando cuántos bits ocupa cada campo. ¿Cuántas páginas puede tener a lo sumo un programa? ¿Cuántas páginas tiene la memoria principal?

Las características de la **memoria virtual** son: es de 1 MB (2^{20}) y está paginada, con páginas de 1 kB. Se pueden extraer muchas conclusiones a partir de esos datos relativas a la estructura de las **direcciones lógicas**. Por un lado, se necesitan 20 bits para expresar las direcciones al byte, ya que 1 MB = $1 \times 2^{20} = 2^{20}$ bytes. Por otro lado, como la memoria virtual está paginada, la dirección lógica tiene dos campos: los bits que indican la **página lógica** y los que indican el **desplazamiento** del byte dentro de la página. Dado que las páginas lógicas son de 1 kB, se necesitan 10 bits para expresar un desplazamiento ($1024 = 2^{10}$), con lo que para indicar una página lógica son necesarios 20 – 10 = 10 bits (y, por ello, un **programa** puede tener 2^{10} = **1024 páginas**). El esquema de las direcciones lógicas queda como sigue:



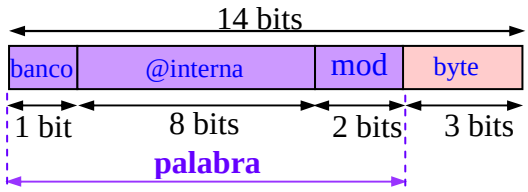
Con respecto a la **memoria principal**, sabemos que es de 16 kB, con lo que las **direcciones físicas** se expresan en 14 bits ($16 \text{ kB} = 2^4 \times 2^{10} = 2^{14}$). Por ello, en la traducción para obtener la dirección física correspondiente a la dirección lógica, se convierten 20 bits en 14. Dado que el desplazamiento no cambia, el campo para expresar la página física es de 4 bits (14 – 10), por lo que la **memoria principal** tiene 2^4 = **16 páginas físicas**.



(b) En cuanto a memoria principal, ¿cuál es el tamaño de un banco en bytes? ¿Cuál es el tamaño de un módulo en bytes?

Teniendo en cuenta la información sobre la estructura de memoria principal, obtenemos que los 3 bits de menos peso de los 14 de la dirección física indican cuál es el byte y los otros 11, la palabra. Dentro del campo de la palabra, debemos distinguir tres campos: dado que la memoria la componen dos bancos consecutivos, el bit de más peso nos indica el **banco**; ya que en cada banco hay 4 módulos entrelazados, necesitamos el campo que indique en qué módulo está la palabra (**mod**, de 2 bits, porque $2^2 = 4$, siendo 4 el número de módulos); además, se debe indicar la dirección concreta en donde está la palabra dentro del módulo (**@interna**, de 10 – 2 = 8 bits). Ésta es la estructura de la dirección física:

dirección física:



Por tanto, sabemos que la memoria principal está compuesta por dos bancos. De ahí que el tamaño de cada banco sea la mitad de la memoria total, es decir, 8 kB.

$$\frac{16\text{ kB}}{2\text{ bancos}} = \frac{2^{14}\text{ bytes}}{2\text{ bancos}} = 2^{13}\text{ bytes/banco} = 8\text{ kB/banco}$$

Del mismo modo, ya que hay 4 módulos en cada banco, cada módulo es de 2 kB:

$$\frac{8\text{ kB/banco}}{4\text{ módulos/banco}} = 2\text{ kB/módulo}$$

(c) Para ejecutar el programa que nos dan, el procesador genera las siguientes direcciones lógicas en la primera pasada del bucle:

Instrucción / dato	Dirección lógica
`movi r1,#0`	0
`movi r2,#199`	8
`load r5,B[r1]`	16 / 10000
`bucle: load r3,A[r1]`	24 / 8192
`load r4,A[r1+8]`	32 / 8200
`mul r4,r3,r4`	40
`add r5,r4,r5`	48
`addi r1,r1,#8`	56
`subi r2,r2,#1`	64
`bge r2,bucle`	72

Hay que tener en cuenta que la última instrucción del programa (`store r5, B`) no se ejecuta dentro del bucle nunca, ya que sólo se ejecuta una vez al final del bucle.

Recordemos las ecuaciones a utilizar para calcular los valores a rellenar en la tabla. Por un lado, para calcular la página lógica y el desplazamiento conocida la dirección lógica:

página lógica = @lógica `div` tamaño página en bytes = @lógica `div` 1024

desplazamiento = @lógica `mod` tamaño página en bytes = @lógica `mod` 1024

Buscaremos en la tabla de páginas la página física correspondiente a cada página lógica, y, con esto y el desplazamiento, calcularemos así la dirección física:

@física = página física × tamaño página en bytes + desplazamiento = $p.f. \times 1024 + d$

Después calcularemos la palabra correspondiente a la dirección física:

palabra = @física `div` tamaño palabra en bytes = @física `div` 8

Además, dado que son bancos consecutivos, calculamos así los campos de la palabra:

banco = palabra `div` tamaño banco en palabras = palabra `div` 2^{10} = palabra `div` 1024
resto = palabra `mod` tamaño banco en palabras = palabra `mod` 2^{10} = palabra `mod` 1024

En el resto de la división anterior todavía hay que distinguir dos campos, por lo que del resultado anterior se extraen el módulo y la dirección interna:

@interna = resto `div` número módulos en banco = resto `div` 4
mod = resto `mod` número módulos en banco = resto `mod` 4

Estas últimas cuatro ecuaciones se pueden generalizar del modo siguiente, para no tener que calcular el resto de la división que nos ha salido al obtener el banco:

banco = (palabra `div` número módulos en banco) `div` tamaño módulo en palabras =
= (palabra `div` 2^2) `div` 2^8 = (palabra `div` 4) `div` 256
@interna = (palabra `div` número módulos en banco) `mod` tamaño módulo en palabras =
= (palabra `div` 2^2) `mod` 2^8 = (palabra `div` 4) `mod` 256
mod = palabra `mod` número módulos en banco = palabra `mod` 2^2 = palabra `mod` 4

Considerando todas estas ecuaciones, ésta es la tabla que se obtiene:

@*l*	*p.l.*	*d*	$t_{trad.}$ (1)	*p.f.*	*d*	@*f*	palabra	banco	resto	@interna	mod	t_{acceso} (2)
0	0	0	20	2	0	2048	256	0	256	64	0	$10^{(a)}$
8	0	8	1	2	8	2056	257	0	257	64	1	$1^{(a)}$
16	0	16	1	2	16	2064	258	0	258	64	2	$1^{(a)}$
10000	9	784	20	11	784	12048	1506	1	482	120	2	$10^{(b)}$
24	0	24	1	2	24	2072	259	0	259	64	3	$1^{(c)}$
8192	8	0	20	10	0	10240	1280	1	256	64	0	$10^{(d)}$
32	0	32	1	2	32	2080	260	0	260	65	0	$10^{(e)}$
8200	8	8	1	10	8	10248	1281	1	257	64	1	$1^{(f)}$
40	0	40	1	2	40	2088	261	0	261	65	1	$1^{(g)}$
48	0	48	1	2	48	2096	262	0	262	65	2	$1^{(g)}$
56	0	56	1	2	56	2104	263	0	263	65	3	$1^{(g)}$
64	0	64	1	2	64	2112	264	0	264	66	0	$10^{(h)}$
72	0	72	1	2	72	2120	265	0	265	66	1	$1^{(h)}$

Aclaraciones:

(1) El tiempo de traducción es de 20 ciclos la primera vez que se accede a una página lógica, ya que no hay referencias a dicha página en el TLB y el resultado de la búsqueda será un fallo; por ello, el sistema debe buscar en la tabla de páginas el número de la página lógica que buscamos y escribir en el TLB el número de la página física que le corresponde. De ahí en adelante, en las siguientes referencias a esa página lógica se necesitará 1 ciclo para la traducción, ya que está su referencia en el TLB, y se producirá un acierto como resultado de la búsqueda. Como se aprecia en la tabla, se produce fallo (tiempo de traducción de 20 ciclos) en tres casos, cuando aparecen las páginas lógicas 0, 9 y 8 por primera vez. El resto de referencias son aciertos y su traducción necesita un único ciclo.

(2) El tiempo de acceso es de 1 ciclo cuando se ha accedido a una palabra con la misma dirección interna y están en los buffers de entrelazado las palabras con la misma dirección interna. En caso contrario, el tiempo de acceso será de 10 ciclos porque hay que leer de memoria principal.

(a) Para acceder a la primera instrucción del programa se necesitan 10 ciclos, ya que hay que leer de memoria principal por primera vez. Pero debido a que hay cuatro módulos entrelazados, además de leer la primera instrucción, se cargan las tres siguientes en los buffers de entrelazado, ya que están en la misma dirección interna (dirección interna 64 del banco 0, en los módulos 0, 1, 2 y 3); por eso para acceder a cada una de las siguientes tres instrucciones (la segunda, la tercera y la cuarta) se necesita un ciclo, porque se consiguen de los buffers de entrelazado.

(b) Tras la tercera instrucción se accede al elemento B[0]. Como se aprecia en la tabla, al elemento B[0] le corresponde la dirección interna 120 del banco 1. Debido a que es la primera dirección que se accede en el banco 1, se necesitan 10 ciclos para acceder al elemento B[0].

(c) Aunque en el acceso anterior se han cargado los buffers de entrelazado las direcciones internas 120 del banco 1, no hay cambios en los contenidos de los buffers de entrelazado del banco 0: siguen estando cargados ahí los contenidos de la dirección interna 64 del banco 0. Por ello, para acceder a la cuarta instrucción, que es el siguiente acceso al del elemento B[0], sólo se necesita un ciclo, porque todavía está en uno de los buffers de entrelazado (la cuarta instrucción está en el módulo 3 de la dirección interna 64 del banco 0).

(d) Tras la cuarta instrucción viene la dirección del elemento A[0]. Como se aprecia en la tabla, tanto la cuarta instrucción como el elemento A[0] están en la dirección interna 64, pero aunque en los buffers de entrelazado del banco 0 todavía están los contenidos de la dirección interna 64, hemos puesto que se necesitan 10 ciclos para acceder al elemento A[0], ya que está en el banco 1 y en este caso los buffers de entrelazado guardan los contenidos de la dirección interna 120 correspondiente al elemento B[0] accedido anteriormente. Por ello, al rellenar la tabla hay que analizar con cuidado si la anterior dirección interna corresponde al mismo banco o no.

(e) Para acceder a la quinta instrucción se necesitan 10 ciclos, ya que hay que volver a leer de memoria principal: pese a estar en el banco 0, su dirección interna difiere de las cuatro anteriores. Por ello, las palabras de los buffers de entrelazado no son utilizables por estar en la dirección interna 64 y pertenecer la quinta instrucción a la dirección interna 65.

(f) Se necesita 1 ciclo para acceder al elemento A[1], ya que todavía figuran en los buffers de entrelazado del banco 1 las direcciones internas 64.

(g) Se necesita 1 ciclo para acceder a cada una de las siguientes instrucciones, porque se han cargado en los buffers de entrelazado del banco 0 a la vez que la quinta instrucción, puesto que todas están en la dirección interna 65.

(h) Finalmente, en el caso de la 9ª instrucción (ya que está en la dirección interna 66 del banco 0), hay que leer de nuevo de memoria principal y se necesitan 10 ciclos. Sin embargo, para las dos últimas instrucciones sólo se necesita 1 ciclo por cada una de ellas, al haber sido cargadas en los buffers de entrelazado a la vez que la 9ª (aunque en la tabla aparece hasta la última instrucción del bucle, aquí también contemplamos la última instrucción del programa).

En resumen, para traducir las direcciones generadas en la primera pasada del bucle se necesitan 70 ciclos (3 fallos × 20 ciclos + 10 aciertos × 1 ciclo), y para acceder a la información de memoria principal, 58 ciclos más (5 lecturas de memoria × 10 ciclos + 8 lecturas de buffers de entrelazado × 1 ciclo). Así, en total se necesitan **128 ciclos** para traducir todas las direcciones de la primera pasada del bucle y acceder a memoria principal.

(d) En el apartado anterior hemos calculado el tiempo necesario en la primera pasada. En éste, en cambio, calcularemos el tiempo necesario para traducir todas las direcciones lógicas que genera el procesador al ejecutar el programa y acceder a memoria principal.

Rellenar toda la tabla sería demasiado largo. Por ello, analizaremos el comportamiento del programa y extraeremos conclusiones generales para calcular el tiempo necesario. Calcularemos primero el tiempo necesario para traducir las direcciones lógicas, y después el tiempo para acceder a memoria principal.

Para calcular el tiempo necesario para traducir las direcciones lógicas es suficiente con generalizar la ecuación que hemos escrito en el apartado anterior:

$$t_{traducción}\,(\text{en ciclos}) = n \text{ fallos} \times 20 \text{ ciclos} + m \text{ aciertos} \times 1 \text{ ciclo}$$

Es decir, debemos saber cuántos fallos ocurren en total en el TLB y cuántos aciertos. Como ya se ha dicho, los fallos ocurren la primera vez que se accede a una página lógica (supondremos que una vez cargadas las páginas lógicas en memoria seguirán ahí hasta que acabe el programa, es decir, que en memoria principal hay suficiente espacio para cargar todas las páginas de este programa, y que el sistema no las reemplazará por otras por falta de espacio).

Por ello, lo que hay que hacer es calcular cuántas páginas lógicas ocupa el programa (instrucciones y datos). Analicémoslos uno a uno. Como hemos apreciado en la tabla, todas las instrucciones están en la página lógica 0. Así, sólo ocurre un fallo debido a las direcciones de las instrucciones.

En cuanto al vector A, vemos en la tabla que sus primeros elementos (A[0] y A[1] aparecen en dicha tabla) están en la página lógica 8. Si analizamos el comportamiento del bucle, veremos que en la última pasada (cuando i=199) se accede a los elementos A[199] y A[200]. Por ello, el vector A tiene al menos 201 elementos, los que están entre A[0] y A[200]. Ya que cada elemento ocupa 8 bytes, el vector A ocupará en total 1608 bytes. Como en cada página caben 1024 bytes, se obtiene que el vector A ocupa 2 páginas. En cualquier caso, hay que verificar si eso es cierto o no, ya que puede ocurrir que ocupe 3 páginas si el elemento A[0] no está al principio de una página. No es éste el caso, ya que hemos calculado que el elemento A[0] está en la página lógica 8 con

desplazamiento 0, pero siempre hay que verificarlo. Para ello, calcularemos la dirección lógica del último elemento, y de ahí obtendremos en qué dirección lógica está. Para calcular la dirección lógica del último elemento tendremos en cuenta la dirección inicial del vector, cuántos elementos tiene y cuántas posiciones ocupa cada elemento, de este modo:

$$@A[200] = 8192 + 200 \times 8 = 9792$$

De esta forma obtendremos la página lógica y el desplazamiento que corresponden a esa dirección:

$$@\text{lógica} = 9792 \rightarrow \text{página lógica} = 9792 \texttt{ div } 2^{10} = 3068 \texttt{ div } 1024 = 9$$

$$\text{desplazamiento} = 9792 \texttt{ mod } 2^{10} = 9792 \texttt{ mod } 1024 = 576$$

Es decir, las páginas lógicas que ocupa el vector A son éstas: 8, 9.

En cuanto al vector B, el único elemento que se accede es B[0] y, como hemos calculado en la tabla, está en la página lógica 9 con desplazamiento 784.

Resumiendo, entre las instrucciones y los datos están en las páginas lógicas 0, 8 y 9. Esto es, hay 3 páginas lógicas, 3 fallos.

Para calcular el número de aciertos, debemos calcular el número de accesos total. Para ello, debemos analizar el programa. Fuera del bucle hay cuatro instrucciones, tres delante del bucle y una detrás del bucle, accediéndose a ellas una vez, al comienzo y al final del programa. Además, tanto al principio como al final se accede al elemento B[0].

Dentro del bucle hay 7 instrucciones y se accede a 2 elementos de A en cada pasada. Esto es, hay 9 accesos en cada pasada; dado que el bucle se repite 200 veces, el número de accesos total es éste:

$$\text{número de accesos} = (3 + 1) + (7 + 2) \times 200 + (1 + 1) = 1806$$

De estos accesos, 3 son fallos, por lo que (1806 – 3) = 1803 serán aciertos. Así, el tiempo de traducción total será el siguiente:

$$t_{traducción} = 3 \text{ fallos} \times 20 \text{ ciclos} + 1806 \text{ aciertos} \times 1 \text{ ciclo} = 1866 \text{ ciclos}$$

En cuanto al tiempo de acceso, en cambio, nos basta con analizar la tabla, ya que ahí se refleja el comportamiento de los accesos. Se ve en la tabla que los datos están en el banco 1 y las instrucciones en el banco 0. Por ello, pese a que las direcciones de datos e instrucciones figuran intercaladas, unas no influyen en el tiempo de acceso a las otras, puesto que los accesos se hacen a bancos diferentes. Analicemos todos los casos posibles.

Para empezar, para acceder a las 3 instrucciones y el dato —elemento B[0]—, el número de ciclos necesario es el que se indica en la tabla:

$$\text{Antes del bucle: } 10 + 1 + 1 + 10 = 22 \text{ ciclos}$$

Dentro del bucle, en cambio, calcularemos de manera independiente los tiempos de acceso a instrucciones y datos, porque ya hemos comentado que no se influyen mutuamente por estar en bancos diferentes.

En cuanto a las instrucciones, debemos distinguir la primera pasada del resto, porque el tiempo de acceso la primera vez es distinto del resto. De hecho, en la primera pasada del bucle, como se aprecia en la tabla, se necesita 1 ciclo para acceder a esa instrucción porque está en el buffer de entrelazado: al ser la palabra 259, está en la dirección interna 64 del banco 0, y al leer la primera instrucción (palabra 256) se han cargado en los buffers de entrelazado las palabras 256, 257, 258 y 259, de la dirección interna 64 del banco 0, en los módulos 0, 1, 2 y 3. Pero a partir de la segunda pasada las cosas cambian, debido a que antes de la primera instrucción del bucle se ejecuta la última del bucle, en concreto la 265, que está en la dirección interna 66, en el módulo 1. Así, en los buffers de entrelazado están las palabras 264, 265, 266 y 267. Por ello, cuando se vuelve a necesitar la palabra 259 (esto es, la primera del bucle), ya no está en el buffer de entrelazado y se necesitan 10 ciclos porque, al estar en otra dirección interna, hay que leerla de nuevo de memoria principal.

Por tanto, para calcular el tiempo de acceso de las instrucciones distinguimos la primera pasada del bucle del resto.

Tiempo de acceso a las instrucciones:

$$\text{Primera pasada del bucle: } 2 \times 10 + 5 \times 1 = 25 \text{ ciclos}$$

$$\text{Siguientes pasadas del bucle: } 3 \times 10 + 4 \times 1 = 34 \text{ ciclos}$$

Como el bucle se repite 200 veces en total, a excepción de la primera pasada quedan otras 199. Por ello, el tiempo de acceso de las instrucciones dentro del bucle es:

$$\text{Instrucciones dentro del bucle: } 25 + 34 \times 199 = 6791 \text{ ciclos}$$

En cuanto a los datos, dentro del bucle se accede a los elementos `A[i]` y `A[i+1]`. Como todos los elementos del vector A están en el banco 1, aunque el procesador genere sus direcciones entre las direcciones de las instrucciones, éstas últimas no cambian los valores de los buffers de entrelazado del banco 1, al estar las instrucciones en el banco 0. Por ejemplo, en la primera pasada del bucle (`i=0`) sólo se accede a los elementos `A[0]` y `A[1]` (palabras 1280 y 1281, respectivamente), pero al haber 4 módulos entrelazados, a la vez que esos elementos, se cargan otros dos en los buffers de entrelazado; en concreto, en la primera pasada del bucle se han leído a la vez los elementos `A[0]`, `A[1]`, `A[2]` y `A[3]`, los cuatro de la dirección interna 64 del banco 1, en los módulos 0, 1, 2 y 3. De esta forma, cuando en la segunda pasada del bucle (`i=1`) se debe acceder a los elementos `A[1]` y `A[2]`, palabras 1281 y 1282, éstas ya figuran en el buffer de entrelazado y se necesita 1 ciclo para acceder a cada uno de ellos. En la tercera pasada del bucle (`i=2`) se debe acceder a los elementos `A[2]` y `A[3]`, palabras 1282 y 1283, que ya figuran en el buffer de entrelazado y se necesita otra vez 1 ciclo por cada uno de ellos. En la cuarta pasada del bucle (`i=3`) se debe acceder a los elementos `A[3]` y `A[4]`, palabras 1283 y 1284: la primera figura en el buffer de entrelazado y se necesita 1 ciclo para acceder a ella, pero debido a que la segunda está en otra dirección interna (dirección interna 65 del banco 1), de nuevo hay que acceder a memoria principal y se necesitan 10 ciclos para ello. Podemos reflejar en un esquema la sucesión de direcciones para acceder al vector A, a la espera de extraer de ahí un comportamiento repetitivo.

Pasada del bucle	Contenido del buffer de entrelazado **(elementos a acceder)** módulo M0	M1	M2	M3	Número de ciclos
i=0	**A[0]**	**A[1]**	A[2]	A[3]	10 + 1
i=1	A[0]	**A[1]**	**A[2]**	A[3]	1 + 1
i=2	A[0]	A[1]	**A[2]**	**A[3]**	1 + 1
i=3	A[0] **A[4]**	A[1] A[5]	A[2] A[6]	**A[3]** A[7]	1 + **10**
i=4	**A[4]**	**A[5]**	A[6]	A[7]	1 + 1
i=5	A[4]	**A[5]**	**A[6]**	A[7]	1 + 1
i=6	A[4]	A[5]	**A[6]**	**A[7]**	1 + 1
i=7	A[4] **A[8]**	A[5] A[9]	A[6] A[10]	**A[7]** A[11]	1 + **10**
i=8	**A[8]**	**A[9]**	A[10]	A[11]	1 + 1
i=9	A[8]	**A[9]**	**A[10]**	A[11]	1 + 1
i=10	A[8]	A[9]	**A[10]**	**A[11]**	1 + 1
i=11	A[8] **A[12]**	A[9] A[13]	A[10] A[14]	**A[11]** A[15]	1 + **10**
i=12	**A[12]**	**A[13]**	A[14]	A[15]	1 + 1

Como se puede ver, a partir de la quinta pasada (`i=4`) podemos agrupar las pasadas del bucle de cuatro en cuatro: en la última de las cuatro se necesitan (1 + 10) ciclos para acceder a los elementos, ya que están en una dirección interna diferente; en las otras tres, en cambio, sólo se necesitan (1 + 1) ciclos, dado que los elementos ya están en los buffers de entrelazado. Las cuatro pasadas iniciales del bucle son una excepción. En la primera se necesitan (10 + 1) ciclos, ya que es la primera vez que se accede a los elementos del vector A; el resto de las veces, en cambio, sucede como los otros casos: en la segunda y tercera pasada se necesitan (1 + 1) ciclos y en la cuarta, (1 + 10) ciclos.

Esto es, si agrupamos el número de pasadas del bucle (200) de cuatro en cuatro (dividimos entre cuatro, 50), entonces calculamos cuántas veces se repite la sucesión anterior de accesos y podremos calcular el número de ciclos total que son necesarios: en concreto, en 49 grupos se necesitan $[(1 + 1) \times 3 + (1 + 10)] = 17$ ciclos para acceder a los elementos de A, y en el primero $[(10 + 1) + (1 + 1) \times 2 + (1 + 10)] = 26$ ciclos.

Resumiendo, el número de ciclos que se necesitan para acceder a los elementos del vector A dentro del bucle será el siguiente:

Vector A dentro del bucle: $26 + 17 \times 49 = 859$ ciclos

Finalmente, para acceder a la instrucción tras el bucle (`store`) y su dato asociado —elemento B[0]—, se necesitan (1 + 10) = 11 ciclos. De hecho, la última instrucción está en la dirección lógica 80; aunque no está en la tabla, podemos calcular todos los valores

de esta instrucción y podemos ver que la última instrucción también está en la página lógica 0 con desplazamiento 80. Así, estará en la página física 2, en la dirección física 2128 (es decir, la palabra 266) que se corresponde con una dirección del banco 0, dirección interna 66, módulo 2. Por ello, esta instrucción se lee y se carga en los buffers de entrelazado a la vez que las dos anteriores, y cuando quiera accederse a ella se necesitará 1 ciclo, ya que se lee desde el buffer de entrelazado. En cuanto al dato, en cambio, se necesitan 10 ciclos ya que en la última pasada dentro del bucle se ha accedido al elemento `A[200]` del vector A (dirección lógica 9792, dirección física 11840, palabra 1480, banco 1, dirección interna 114 y módulo 0) y en la última instrucción se necesita acceder al elemento B[0] (dirección lógica 10000, dirección física 12048, palabra 1506, banco 1, dirección interna 120 y módulo 2). Dado que esos dos elementos no están en la misma dirección interna, no son válidos los valores de los buffers de entrelazado y hace falta escribir directamente el valor de B[0] en memoria principal:

Tras el bucle: 1 + 10 = 11 ciclos

Por ello, para acceder a memoria principal, en total se necesitan 7683 ciclos:

$$t_{acceso} = 22 \text{ ciclos} + 6791 \text{ ciclos} + 859 \text{ ciclos} + 11 \text{ ciclos} = 7683 \text{ ciclos}$$

Sumando el tiempo de traducción y el tiempo de acceso a memoria principal, tenemos en definitiva el siguiente tiempo:

1866 ciclos + 7683 ciclos = 9549 ciclos

Es decir, para traducir y acceder a memoria principal en todas las referencias que se generan al ejecutar el programa entero se necesitan 9549 ciclos.