# Boletín de Ejercicios de Administración de la Memoria

## 2º curso del Grado en Informática

28 de octubre de 2021

1. Un sistema operativo corre sobre una máquina que tiene 256 MiB de memoria física, con una dirección física de 30 bits y una dirección virtual de 32 bits. Utiliza memoria virtual paginada con una tabla de traducción de páginas (TTP) de un único nivel. El tamaño de la página virtual es de 4 KiB. Responde a las siguientes cuestiones:

   *a*) Dibuja un esquema de la dirección virtual, dividiéndola en la parte correspondiente a la página virtual y la correspondiente al desplazamiento.

   *b*) ¿Cuántas páginas físicas o marcos de página tiene el sistema? ¿Cuál es el número máximo de páginas físicas que podría tener? ¿Cuánta memoria física suponen esa cantidad de páginas?

   *c*) ¿Cuál es el número máximo de páginas virtuales que admite la tabla de páginas?

   *d*) Si cada entrada de la TTP tiene 4 bytes, ¿cuánto puede llegar a ocupar una TTP en memoria?

   *e*) Si en el sistema corren 100 procesos, ¿cuánta memoria física podrían llegar a consumir las TTPs?

   *f*) Descompón las siguientes direcciones virtuales de memoria en página virtual y desplazamiento: 0x12345678, 0xFFFFFFF0, 0x40404040.

   *g*) Resuelve los apartados anteriores de nuevo suponiendo que el tamaño de nuestra página virtual es de 8 KiB.

2. Un sistema operativo utiliza un sistema de paginación de 1 nivel por demanda y sigue una política local. Un proceso tiene asignados cuatro marcos de página (0-3), inicialmente vacíos, y produce la siguiente serie de referencias a páginas:

   ```
   0 1 0 * 0 1 2 1 3 * 3 15 16 * 16 17 15 16 17 20 16 * 20 21 * 21 0
   1 0 21 * 0 22 * 22 23 24 23 25
   ```

   Un * representa el momento en que se produce una marca de reloj. Calcula el número de fallos de página para los siguientes algoritmos:

   *a*) Algoritmo de maduración con contadores de 2 bits. En caso de empate, se reemplaza la página que se encuentre en el marco más pequeño.

   *b*) LRU implementado mediante una matriz hardware.

3. Tenemos un SO que implementa memoria virtual paginada pura. El hardware nos proporciona direcciones virtuales de 32 bits y direcciones físicas de 24 bits. El tamaño de la página es de 4 KiB. La política de reemplazo utilizada es NRU donde en caso de empate se selecciona el número de marco físico más bajo. El SO asigna 4 marcos a nuestro proceso (del 0 al 3). La lista de referencias a memoria que genera el proceso es la siguiente (indicando con una E que es un acceso a escritura, con una L que es de lectura. Las referencias suceden de izquierda a derecha y de arriba a abajo.

   ```
   00003124 (E), 00004ABC (E), 00007812 (L), 0000F000 (E),
   00000888 (L), TIC DE RELOJ, 00000354 (L), 00003010 (E),
   0000FFFE (E), 00005002 (L).
   ```

*a*) Indica cuál de las opciones muestra la página virtual que se encuentra en cada marco físico (del 0 al 3) junto con los bits de referencia y modificación.

i) 0(r), 5(r), 3(m), F(m)

ii) 3(rm), 5(r), 0(r), F(rm)

iii) 5(r), 4(m), 0(r), F(m)

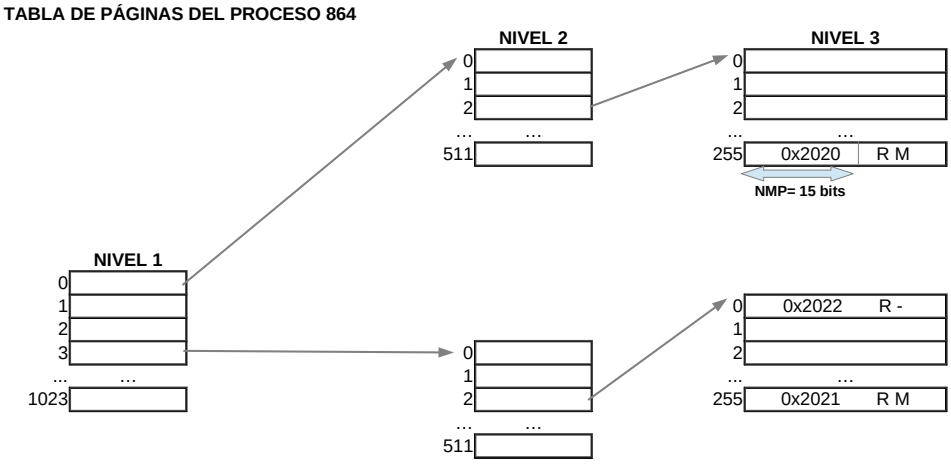
*b*) Indica cuál de las opciones muestra la página virtual que se encuentra en cada marco físico (de 0 al 3) si utilizamos como algoritmo de reemplazo el FIFO de segunda oportunidad.

i) 3(m), 5(r), 0(r), F(m)

ii) 0(r), 3(r), F(r), 5(r)

iii) 0(r), 3(r), 5(r), F(r)

4. Supongamos una arquitectura de procesador que soporta tablas de páginas de 3 niveles como la que se muestra a continuación para el proceso 864[1]. `NMP` hace referencia al Número de Marco de Página en las direcciones físicas, que tiene una longitud de 15 bits, y `R` y `M` a los bits de referencia y modificado, respectivamente (que aparezca la letra correspondiente significa que el bit está activado). Se emplean páginas de tamaño 4 KiB y el SO asigna inicialmente a cada proceso un total de 3 marcos de página. Se emplea una estrategia de asignación dinámica con reemplazo local, en la que el SO otorga 1 marco de página adicional (a partir del marco de página `0x4040`–inclusive) a cada proceso que, habiendo consumido los 3 marcos de página inicialmente asignados, ve cómo 2 accesos consecutivos a memoria producen fallo de página (el marco se otorga para resolver el 2º fallo de página). Para los reemplazos se utiliza la política NRU.




Se pide:

*a*) Calcula el tamaño del espacio de direcciones virtuales de cada proceso y la cantidad máxima de memoria física que podría ser instalada.

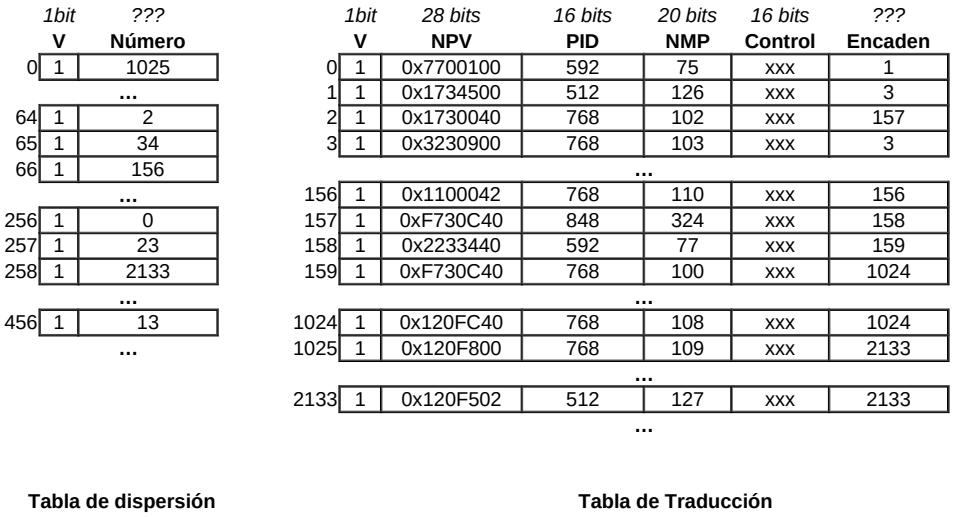
*b*) Dado que el proceso ha realizado únicamente 3 accesos a memoria, indica las páginas virtuales a las que ha accedido y si los accesos han sido de lectura o escritura.

*c*) Muestra cómo queda la tabla de páginas después de los 3 eventos siguientes: `Lectura Dirección virtual 0x000000`$^2$ → `TIC` → `Escritura Dirección virtual 0x020000`$^2$.

[1]En realidad se muestra una versión simplificada, donde solo incluimos los campos de interés en cada entrada y en la que una entrada sin contenido se supone que tiene el bit de validez a 0, mientras que una entrada con contenido tiene el bit de validez a 1.

*d*) Suponiendo que la tabla de páginas se encuentra de forma completa en memoria principal, que cada acceso a memoria principal para obtener una entrada de cualquiera de los niveles requiere un total de 10 ns, y que la MMU dispone de un TLB con un tiempo de acceso de 1 ns y una tasa de aciertos del 95 %, calcula los tiempos mínimo y máximo para detectar un fallo de página y un acierto de página (asume que el procesamiento de la información del TBL y tabla de páginas por parte de la MMU es despreciable).

5. Supón la tabla de páginas invertida mostrada a continuación (únicamente se muestran algunas de las entradas que se están usando en este momento), en la que los tamaños en bits de cada uno de los campos se han ajustado al mínimo imprescindible en cada caso (`NPV` representa el número de página virtual, `NMP` el número de marco de página y `Control` incluye, entre otros, los bits R, M, bits de permiso y bit de visible en modo núcleo, que no son de relevancia para el ejercicio–por eso el valor de este campo aparece como `xxx`; los tamaños en bits que aparecen marcados como `???` habrán de ser calculados):



| | *1bit* V | *???* Número |
|---|---|---|
| 0 | 1 | 1025 |
| ... | | |
| 64 | 1 | 2 |
| 65 | 1 | 34 |
| 66 | 1 | 156 |
| ... | | |
| 256 | 1 | 0 |
| 257 | 1 | 23 |
| 258 | 1 | 2133 |
| ... | | |
| 456 | 1 | 13 |
| ... | | |

**Tabla de dispersión**

| | *1bit* V | *28 bits* NPV | *16 bits* PID | *20 bits* NMP | *16 bits* Control | *???* Encaden |
|---|---|---|---|---|---|---|
| 0 | 1 | 0x7700100 | 592 | 75 | xxx | 1 |
| 1 | 1 | 0x1734500 | 512 | 126 | xxx | 3 |
| 2 | 1 | 0x1730040 | 768 | 102 | xxx | 157 |
| 3 | 1 | 0x3230900 | 768 | 103 | xxx | 3 |
| ... | | | | | | |
| 156 | 1 | 0x1100042 | 768 | 110 | xxx | 156 |
| 157 | 1 | 0xF730C40 | 848 | 324 | xxx | 158 |
| 158 | 1 | 0x2233440 | 592 | 77 | xxx | 159 |
| 159 | 1 | 0xF730C40 | 768 | 100 | xxx | 1024 |
| ... | | | | | | |
| 1024 | 1 | 0x120FC40 | 768 | 108 | xxx | 1024 |
| 1025 | 1 | 0x120F800 | 768 | 109 | xxx | 2133 |
| ... | | | | | | |
| 2133 | 1 | 0x120F502 | 512 | 127 | xxx | 2133 |
| ... | | | | | | |

**Tabla de Traducción**

Se pide:

*a*) Calcula la cantidad total de memoria física que podría instalarse y el tamaño del espacio de direcciones virtuales de los procesos. Supón que el SO emplea páginas de 4 KiB.

*b*) Sabiendo que la función de dispersión empleada devuelve siempre los 10 bits menos significativos del número de página virtual, ¿cuántas entradas se necesitan para la tabla de dispersión? ¿Cuántas entradas se necesitarán para la tabla de traducción?

*c*) Calcula el tamaño (en bits) de los campos `Número` de la tabla de dispersión y `Encaden` de la tabla de traducción. Para este último supón que se utiliza el número de la propia entrada como marca de fin (es decir, para la entrada `3` se usa el mismo `3` como marca de fin en lugar del valor `-1` usado en los ejercicios).

*d*) Supón el acceso a la dirección virtual `0x2233440123` por parte del proceso con PID `768`. Explica si dicho acceso provoca un acierto o fallo de página, e indica las entradas de las tablas de dispersión y traducción que se usarían para determinarlo.

*e*) Suponiendo que la tabla de páginas se encuentra de forma completa en memoria (tanto la tabla de dispersión como la de traducción), que cada acceso a memoria para obtener una entrada tanto para la tabla de traducción como para la tabla de dispersión requiere un total de 10 ns y que el tamaño medio y máximo de las listas formadas con la información de encadenamiento es de 8,75 y 34 entradas de la tabla de traducción respectivamente,

[2] Asume que los bits que faltan a la izquierda hasta completar cada dirección virtual valen siempre 0.

calcula los tiempos mínimo, medio y máximo para detectar un fallo de página (considera únicamente el tiempo de acceso a memoria, es decir asume que el procesamiento de la información de la tabla de páginas por parte del SO es despreciable).

6. Tenemos un computador con direcciones virtuales de 32 bits, direcciones físicas de 20 bits y página de 4 096 bytes. El SO ha asignado a nuestro proceso 4 marcos de memoria (del 0 al 3). Dicho proceso ha producido la siguiente lista de accesos a memoria:

| Orden | Dirección | L/E |
|---|---|---|
| 1 | 0x00000123 | L |
| 2 | 0x00002567 | E |
| 3 | 0xFFFFFC00 | L |
| 4 | 0x02020401 | L |
| 5 | 0x02022F00 | E |
| 6 | 0x020204FF | E |
| Tic | | |
| 7 | 0x00003978 | L |
| 8 | 0x00002667 | L |
| 9 | 0xFFFFFD12 | L |

Trabajamos con paginación pura y usamos una tabla de páginas invertida, con una función hash (de dispersión) *h(x) = x mod 16*, una tabla de dispersión, y una tabla de traducción. En caso de colisión se forma una lista ligada en la tabla de traducción gracias a un puntero al siguiente elemento que será el primer hueco libre en la tabla. Se pide mostrar cómo queda la tabla de páginas invertida (tablas de dispersión y traducción) TRAS CADA UNA DE LAS REFERENCIAS A MEMORIA. Indique también el número de entradas de la tabla. El algoritmo de reemplazo es LRU. Indica junto a cada entrada de la tabla, el índice al que corresponde (0, 1, 2, ...)

7. En un computador con segmentación paginada pura utilizamos un registro de segmento de 8 bits y una tabla de paginación de dos niveles con PT1=8 y PT2=8. Las direcciones virtuales son de 32 bits, las direcciones físicas de 30 bits y el tamaño de página es de 64 KiB. El SO ha asignado a nuestro proceso 4 marcos de memoria (marco0-marco3). La política de reemplazo es NRU. En caso de empate, se selecciona el marco con número más pequeño de los 4 disponibles.

   *a*) Indica para cada referencia a memoria de la tabla que se muestra a continuación:
   - Los valores de Número de Página Virtual (NPV) y su desglose para PT1 y PT2.
   - Si se produce acierto o fallo de página.
   - El contenido de los 4 marcos de página.
   - El valor de los bits de control (R y M) para los 4 marcos de página.

   *b*) Dibuja el estado de la/s tabla/s de páginas y segmentos de este único proceso al finalizar.

   *c*) Viendo la solución del apartado anterior, calcula el ahorro de memoria destinada a la/s tabla/s de páginas frente al caso en el que tuviéramos un sistema de segmentación paginada con paginación de 1 único nivel.

| Orden | Segmento | Dirección | L/E |
|---|---|---|---|
| 1 | 1 | 0x20FF0123 | E |
| 2 | 0 | 0x00002567 | L |
| 3 | 0 | 0x0033FC00 | L |
| 4 | 2 | 0x00000401 | E |
| 5 | 0 | 0x33002F00 | L |
| 6 | 0 | 0x003304FF | E |
| Tic | | | |
| 7 | 0 | 0x33003978 | L |
| 8 | 2 | 0x00FF2667 | E |
| 9 | 1 | 0x20FFFD12 | E |

8. Tenemos un procesador con memoria virtual por paginación pura, con tabla de traducción de páginas (TTP) lineal con 2 bytes por entrada, tamaño de página de 8 KiB, 24 bits de dirección virtual y 20 bits de dirección física. El computador tiene instaladas 256 KiB de RAM y el SO ha asignado 4 marcos físicos al único proceso que está en ejecución. Suponemos que los primeros 10 accesos a memoria han sido a las siguientes direcciones virtuales (en hex. con E=escritura, L=lectura, *=Tic de reloj): `00A120(E), 009344(E), 006220(E), 00BF00(L), 004BA0(L), *, 003ABC(E), 006148(L), 008320(L), 005730(L), *, 00D444(L).`

   Se pide:

   *a*) Enumera 4 campos que puedes encontrar en una entrada de la TTP.

   *b*) Calcula el tamaño en bytes de la TTP

   *c*) Calcula el número de marcos físicos existentes

   *d*) Dada la secuencia de accesos a memoria que se indica en el enunciado, y sabiendo que el SO utiliza LRU como política de reemplazo de páginas, muestra TRAS CADA ACCESO, qué página virtual está en cada uno de los 4 marcos asignados.

   *e*) Si utilizásemos una tabla de traducción de páginas invertida con un tamaño de 4 bytes por entrada, ¿cuánto ocuparía en bytes?

   *f*) ¿Por qué hemos necesitado 4 bytes por entrada? ¿No nos hubieran bastado los 2 bytes de la TTP inicial? Explícalo brevemente

   *g*) Repite el apartado d) pero con política NRU. Recuerda que el asterisco significa la llegada de un tic de reloj.

9. Un computador con segmentación paginada pura y tabla de paginación de dos niveles con PT1=10 y PT2=10, direcciones virtuales de 32 bits, direcciones físicas de 20 bits y página de 4096 bytes. El SO ha asignado a nuestro proceso 4 marcos de memoria (marco0-marco3). Utilizando los datos de la tabla que se muestra, dibuje el estado final de la/s tabla/s de páginas y segmentos de un único proceso. La política de reemplazo es NRU. En caso de empate, se selecciona el marco con número más pequeño de los 4 disponibles.

| Orden | Segmento | Dirección | L/E |
|---|---|---|---|
| 1 | 0 | 0x00000123 | L |
| 2 | 0 | 0x00002567 | E |

| | | | |
|---|---|---|---|
| 3 | 2 | 0xFFFFFC00 | L |
| 4 | 1 | 0x02020401 | L |
| 5 | 1 | 0x02022F00 | E |
| 6 | 1 | 0x020204FF | E |
| Tic | | | |
| 7 | 0 | 0x00003978 | L |
| 8 | 0 | 0x00002667 | L |
| 9 | 0 | 0xFFFFFD12 | L |

10. Tenemos un computador con 8 GiB de RAM, memoria virtual paginada con direcciones virtuales de 64 bits, físicas de 64 bits, tamaño de página de 4 096 bytes y palabras de 64 bits. La memoria virtual usa **reemplazo local**, LRU como algoritmo de reemplazo y una tabla de páginas invertida que tiene: una tabla de traducción, una tabla de dispersión y una función hash (de dispersión) $h(PID, p) = (PID \cdot 13 + p) \bmod 8$, siendo $PID$ el identificador del proceso que hace el acceso y $p$ la página a la que quiere acceder; las colisiones se resuelven mediante una lista ligada en la propia tabla de traducción.

    En este computador, ejecutamos dos procesos con PIDs 768 y 1024 a los que el SO ha asignado 2 marcos de memoria a cada uno (0 y 1 para el primer proceso, y 2 y 3 para el segundo). Dichos procesos han producido la siguiente lista de accesos a memoria en orden cronológico:

| Orden | PID | Dirección | L/E |
|---|---|---|---|
| 1 | 768 | 0x0000000000004A44 | E |
| 2 | 1024 | 0x2560034211077540 | E |
| 3 | 1024 | 0x0000000000004880 | L |
| 4 | 768 | 0x0000000000004010 | L |
| 5 | 1024 | 0xFDA75BBF47FF07F0 | L |
| 6 | 768 | 0x010000000101D721 | E |

    Se pide:

    *a*) Dibujar en qué campos se descomponen las direcciones virtuales y físicas. Dibujar también las tablas de dispersión y traducción de la tabla de páginas invertida en su estado inicial, incluyendo aquellos campos que sean necesarios (JUSTIFICANDO su inclusión). Supondremos que las primeras cuatro entradas de la tabla de traducción se encuentran ya ocupadas (su contenido no es importante). Calcular finalmente el tamaño de cada tabla (el tamaño de cada entrada siempre debe ser múltiplo de palabra). Supondremos que, para guardar el PID de un proceso, son necesarios 32 bits.

    *b*) Mostrar cómo queda la tabla de páginas invertida (tablas de dispersión y traducción) TRAS CADA UNA DE LAS REFERENCIAS A MEMORIA dadas.

# Soluciones a ejercicios seleccionados

- **Ejercicio 1**:

  a) Como la página virtual es de 4 KiB, para saber los bits que necesitamos para el desplazamiento calcularíamos el logaritmo en base 2 de 4 096, lo que nos da 12. Como el tamaño de las páginas siempre será una potencia de 2, el cálculo del logaritmo se hace prácticamente de cabeza. Entonces tenemos una dirección de 32 bits que tiene un desplazamiento de 12 bits, y por tanto los 20 bits restantes corresponden al número de página virtual. El esquema quedaría como sigue:

| Página virtual de 20 bits | Desplaz. de 12 bits |
|---|---|

  b) El número de marcos físicos viene dado por la cantidad de memoria física que tenga. Como tenemos 256 MiB y cada marco físico tiene el mismo tamaño que la página virtual (4 KiB), el resultado sería $\frac{256\times1\,024\times1\,024}{4\,096} = 65\,536$ marcos físicos.

  El número máximo de marcos físicos que puede tener el sistema viene determinado por el tamaño de su dirección física. En este caso es de 30 bits, y por tanto podemos tener un máximo de $\frac{2^{30}}{2^{12}} = 2^{18} = 262\,144$ páginas físicas. Esto supone una memoria física máxima de $2^{30}$ bytes, es decir, 1 GiB.

  c) El número máximo de páginas virtuales viene dado por el tamaño en bits de la parte de la dirección virtual dedicada a contener la página virtual. Como vimos, este tamaño es de 20 bits, y por tanto el número máximo de páginas sería $2^{20} = 1\,048\,576$ páginas virtuales.

  d) A partir del dato anterior y sabiendo que cada página tiene una entrada en la tabla de traducción de páginas, simplemente lo multiplicamos por 4 bytes por entrada, y nos da que una tabla de páginas puede llegar a ocupar 4 MiB de memoria física.

  e) También a partir del apartado anterior, simplemente tenemos que multiplicar por 100 y nos da que tendríamos dedicados 400 MiB a las tablas de página de los procesos. Siempre suponiendo que se utilizaran las últimas entradas de las tablas, ya que si no es así, se podría buscar alguna solución un poco menos costosa.

  f) Descompongamos las direcciones en dos partes: una de 20 bits y otra de 12. Como en hexadecimal cada dígito corresponde a 4 bits, 12 bits serían 3 dígitos y 20 bits, 5 dígitos. De esta manera la descomposición es muy sencilla: `0x12345678 = 0x12345 / 0x678`, `0xFFFFFFF0 = 0xFFFFF / 0xFF0`, `0x40404040 = 0x4040 / 0x040` donde el primer campo es la página virtual y el segundo el desplazamiento.

  g) Si suponemos un tamaño de página de 8 KiB, los resultados serían algo diferentes. Al tener 8 192 bytes, el número de bits necesarios para el desplazamiento serían 13, y por tanto el número de bits que quedan para la página virtual serían $32 - 13 = 19$.

  El número de marcos físicos ahora serían $\frac{256\times1\,024\times1\,024}{8\,192} = 32\,768$ y podríamos tener un máximo de $\frac{2^{30}}{2^{13}} = 2^{17} = 131\,072$ páginas físicas, lo que sigue siendo la misma cantidad de memoria, 1 GiB. La máxima cantidad de memoria física viene determinada por el tamaño de la dirección física, no por el tamaño de la página. Si aumentamos el tamaño de la página, simplemente tendremos menos páginas para la misma cantidad de memoria.

  El número máximo de páginas virtuales ahora serían $2^{19} = 524\,288$. Si las multiplicamos por 4 bytes por entrada, ahora una TTP máxima ocuparía 2 MiB de memoria física. Si

multiplicamos por 100, obtendremos que ahora nos bastan 200 MiB para contener todas las tablas de los procesos.

A la hora de descomponer las direcciones, ahora no es tan sencillo. Tendremos que pasarlas a binario para saber cómo dividirlas, ya que ahora hay que separar por la derecha 13 bits y reagrupar los 19 de la izquierda para que representarlos en hexadecimal:

- `0x12345678 = 0001 0010 0011 0100 0101 0110 0111 1000`
  `= 0001 0010 0011 0100 010 / 1 0110 0111 1000 =`
  `000 1001 0001 1010 0010 / 1 0110 0111 1000 = 0x091A2 / 0x1678`
- `0xFFFFFFF0 = 1111 1111 1111 1111 1111 1111 1111 0000`
  `= 1111 1111 1111 1111 111 / 1 1111 1111 0000 =`
  `111 1111 1111 1111 1111 / 1 1111 1111 0000 = 0x7FFF / 0x1FF0`
- `0x40404040 = 0100 0000 0100 0000 0100 0000 0100 0000`
  `= 0100 0000 0100 0000 010 / 0 0000 0100 0000 =`
  `010 0000 0010 0000 0010 / 0 0000 0100 0000 = 0x20202 / 0x0040`

- **Ejercicio 2**:

  a) Vamos a suponer que los contadores solo se actualizan en las marcas de reloj y que, además, se ponen a cero cada vez que una nueva página ocupa un marco. Vamos a llevar un contador por marco, que entenderemos asociado a la página que se encuentra dicho marco. Los * representan a las marcas de reloj en las cuales se actualizan los contadores y la F los instantes en los cuales se producen fallos de página. En total se producen 19 fallos de página.

  Se podrían haber hecho otras suposiciones que hubieran acercado este algoritmo más al LRU, por ejemplo, que cuando una página entra nueva, su contador no tenga el valor 00, sino el 10, para evitar que sea expulsada inmediatamente si se pide otra página nueva a continuación.

| | r | 0F |
|---|---|---|
| 0 | 1 | 00 |
| | | |
| | | |
| | | |

| | r | 1F |
|---|---|---|
| 0 | 1 | 00 |
| 1 | 1 | 00 |
| | | |
| | | |

| | r | 0 |
|---|---|---|
| 0 | 1 | 00 |
| 1 | 1 | 00 |
| | | |
| | | |

| | r | * |
|---|---|---|
| 0 | 0 | 10 |
| 1 | 0 | 10 |
| | | |
| | | |

| | r | 0 |
|---|---|---|
| 0 | 1 | 10 |
| 1 | 0 | 10 |
| | | |
| | | |

| | r | 1 |
|---|---|---|
| 0 | 1 | 10 |
| 1 | 1 | 10 |
| | | |
| | | |

| | r | 2F |
|---|---|---|
| 0 | 1 | 10 |
| 1 | 1 | 10 |
| 2 | 1 | 00 |
| | | |

| | r | 1 |
|---|---|---|
| 0 | 1 | 10 |
| 1 | 1 | 10 |
| 2 | 1 | 00 |
| | | |

| | r | 3F |
|---|---|---|
| 0 | 1 | 10 |
| 1 | 1 | 10 |
| 2 | 1 | 00 |
| 3 | 1 | 00 |

| | r | * |
|---|---|---|
| 0 | 0 | 11 |
| 1 | 0 | 11 |
| 2 | 0 | 10 |
| 3 | 0 | 10 |

| | r | 3 |
|---|---|---|
| 0 | 0 | 11 |
| 1 | 0 | 11 |
| 2 | 0 | 10 |
| 3 | 1 | 10 |

| | r | 15F |
|---|---|---|
| 0 | 0 | 11 |
| 1 | 0 | 11 |
| 15 | 1 | 00 |
| 3 | 1 | 10 |

| | r | 16F |
|---|---|---|
| 0 | 0 | 11 |
| 1 | 0 | 11 |
| 16 | 1 | 00 |
| 3 | 1 | 10 |

| | r | * |
|---|---|---|
| 0 | 0 | 01 |
| 1 | 0 | 01 |
| 16 | 0 | 10 |
| 3 | 0 | 11 |

| | r | 16 |
|---|---|---|
| 0 | 0 | 01 |
| 1 | 0 | 01 |
| 16 | 1 | 10 |
| 3 | 0 | 11 |

| | r | 17F |
|---|---|---|
| 17 | 1 | 00 |
| 1 | 0 | 01 |
| 16 | 1 | 10 |
| 3 | 0 | 11 |

| | r | 15F |
|---|---|---|
| 15 | 1 | 00 |
| 1 | 0 | 01 |
| 16 | 1 | 10 |
| 3 | 0 | 11 |

| | r | 16 |
|---|---|---|
| 15 | 1 | 00 |
| 1 | 0 | 01 |
| 16 | 1 | 10 |
| 3 | 0 | 11 |

| | r | 17F |
|---|---|---|
| 17 | 1 | 00 |
| 1 | 0 | 01 |
| 16 | 1 | 10 |
| 3 | 0 | 11 |

| | r | 20F |
|---|---|---|
| 20 | 1 | 00 |
| 1 | 0 | 01 |
| 16 | 1 | 10 |
| 3 | 0 | 11 |

| | r | 16 |
|---|---|---|
| 20 | 1 | 00 |
| 1 | 0 | 01 |
| 16 | 1 | 10 |
| 3 | 0 | 11 |

| | r | * |
|---|---|---|
| 20 | 0 | 10 |
| 1 | 0 | 00 |
| 16 | 0 | 11 |
| 3 | 0 | 01 |

| | r | 20 |
|---|---|---|
| 20 | 1 | 10 |
| 1 | 0 | 00 |
| 16 | 0 | 11 |
| 3 | 0 | 01 |

| | r | 21F |
|---|---|---|
| 20 | 1 | 10 |
| 21 | 1 | 00 |
| 16 | 0 | 11 |
| 3 | 0 | 01 |

| | r | * |
|---|---|---|
| 20 | 0 | 11 |
| 21 | 0 | 10 |
| 16 | 0 | 01 |
| 3 | 0 | 00 |

| | r | 21 |
|---|---|---|
| 20 | 0 | 11 |
| 21 | 1 | 10 |
| 16 | 0 | 01 |
| 3 | 0 | 00 |

| | r | 0F |
|---|---|---|
| 20 | 0 | 11 |
| 21 | 1 | 10 |
| 16 | 0 | 01 |
| 0 | 1 | 00 |

| | r | 1F |
|---|---|---|
| 20 | 0 | 11 |
| 21 | 1 | 10 |
| 16 | 0 | 01 |
| 1 | 1 | 00 |

| | r | 0F |
|---|---|---|
| 20 | 0 | 11 |
| 21 | 1 | 10 |
| 16 | 0 | 01 |
| 0 | 1 | 00 |

| | r | 21 |
|---|---|---|
| 20 | 0 | 11 |
| 21 | 1 | 10 |
| 16 | 0 | 01 |
| 0 | 1 | 00 |

| | r | * |
|---|---|---|
| 20 | 0 | 01 |
| 21 | 0 | 11 |
| 16 | 0 | 00 |
| 0 | 0 | 10 |

| | r | 0 |
|---|---|---|
| 20 | 0 | 01 |
| 21 | 0 | 11 |
| 16 | 0 | 00 |
| 0 | 1 | 10 |

| | r | 22F |
|---|---|---|
| 20 | 0 | 01 |
| 21 | 0 | 11 |
| 22 | 1 | 00 |
| 0 | 1 | 10 |

| | r | * |
|---|---|---|
| 20 | 0 | 00 |
| 21 | 0 | 01 |
| 22 | 0 | 10 |
| 0 | 0 | 11 |

| | r | 22 |
|---|---|---|
| 20 | 0 | 00 |
| 21 | 0 | 01 |
| 22 | 1 | 10 |
| 0 | 0 | 11 |

| | r | 23F |
|---|---|---|
| 23 | 1 | 00 |
| 21 | 0 | 01 |
| 22 | 1 | 10 |
| 0 | 0 | 11 |

| | r | 24F |
|---|---|---|
| 24 | 1 | 00 |
| 21 | 0 | 01 |
| 22 | 1 | 10 |
| 0 | 0 | 11 |

| | r | 23F |
|---|---|---|
| 23 | 1 | 00 |
| 21 | 0 | 01 |
| 22 | 1 | 10 |
| 0 | 0 | 11 |

| | r | 25F |
|---|---|---|
| 25 | 1 | 00 |
| 21 | 0 | 01 |
| 22 | 1 | 10 |
| 0 | 0 | 11 |

b) Encima de cada tabla aparecen las referencias a páginas (con un * hay fallo de página) y delante de cada fila aparece un * si corresponde con el marco cuya fila y columna hay que actualizar (en el caso de un fallo de página, esto nos dice en qué marco estamos situando la página). En total se producen 15 fallos de página y, al final, los cuatro marcos de página se encuentran ocupados por las páginas 24, 25, 23 y 22, respectivamente.

| 0F | | | | |
|---|---|---|---|---|
| * | 0 | 1 | 1 | 1 |
| | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 |

| 1F | | | | |
|---|---|---|---|---|
| | 0 | 0 | 1 | 1 |
| * | 1 | 0 | 1 | 1 |
| | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 |

| 0 | | | | |
|---|---|---|---|---|
| | 0 | 1 | 1 | 1 |
| | 0 | 0 | 1 | 1 |
| | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 |

| 1 | | | | |
|---|---|---|---|---|
| | 0 | 0 | 1 | 1 |
| | 1 | 0 | 1 | 1 |
| | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 |

| 2F | | | | |
|---|---|---|---|---|
| | 0 | 0 | 0 | 1 |
| | 1 | 0 | 0 | 1 |
| * | 1 | 1 | 0 | 1 |
| | 0 | 0 | 0 | 0 |

| 1 | | | | |
|---|---|---|---|---|
| | 0 | 0 | 0 | 1 |
| | 1 | 0 | 1 | 1 |
| | 1 | 0 | 0 | 1 |
| | 0 | 0 | 0 | 0 |

| 3F | | | | |
|---|---|---|---|---|
| | 0 | 0 | 0 | 0 |
| | 1 | 0 | 1 | 0 |
| | 1 | 0 | 0 | 0 |
| * | 1 | 1 | 1 | 0 |

| 15F | | | | |
|---|---|---|---|---|
| * | 0 | 1 | 1 | 1 |
| | 0 | 0 | 1 | 0 |
| | 0 | 0 | 0 | 0 |
| | 0 | 1 | 1 | 0 |

| 16F | | | | |
|---|---|---|---|---|
| | 0 | 1 | 0 | 1 |
| | 0 | 0 | 0 | 0 |
| * | 1 | 1 | 0 | 1 |
| | 0 | 1 | 0 | 0 |

| 17F | | | | |
|---|---|---|---|---|
| | 0 | 0 | 0 | 1 |
| * | 1 | 0 | 1 | 1 |
| | 0 | 0 | 0 | 1 |
| | 0 | 0 | 0 | 0 |

| 15 | | | | |
|---|---|---|---|---|
| * | 0 | 1 | 1 | 1 |
| | 0 | 0 | 1 | 1 |
| | 0 | 0 | 0 | 1 |
| | 0 | 0 | 0 | 0 |

| 16 | | | | |
|---|---|---|---|---|
| | 0 | 1 | 0 | 1 |
| | 0 | 0 | 0 | 1 |
| * | 1 | 1 | 0 | 1 |
| | 0 | 0 | 0 | 0 |

| 17 | | | | |
|---|---|---|---|---|
| | 0 | 0 | 0 | 1 |
| * | 1 | 0 | 1 | 1 |
| | 1 | 0 | 0 | 1 |
| | 0 | 0 | 0 | 0 |

| 20F | | | | |
|---|---|---|---|---|
| | 0 | 0 | 0 | 0 |
| | 1 | 0 | 1 | 0 |
| | 1 | 0 | 0 | 0 |
| * | 1 | 1 | 1 | 0 |

| 16 | | | | |
|---|---|---|---|---|
| | 0 | 0 | 0 | 0 |
| | 1 | 0 | 1 | 1 |
| * | 1 | 1 | 0 | 1 |
| | 1 | 1 | 0 | 0 |

| 20 | | | | |
|---|---|---|---|---|
| | 0 | 0 | 0 | 0 |
| | 1 | 0 | 0 | 0 |
| | 1 | 1 | 0 | 0 |
| * | 1 | 1 | 1 | 0 |

| 21F | | | | |
|---|---|---|---|---|
| * | 0 | 1 | 1 | 1 |
| | 0 | 0 | 0 | 0 |
| | 0 | 1 | 0 | 0 |
| | 0 | 1 | 1 | 0 |

| 0F | | | | |
|---|---|---|---|---|
| | 0 | 0 | 1 | 1 |
| * | 1 | 0 | 1 | 1 |
| | 0 | 0 | 0 | 0 |
| | 0 | 0 | 1 | 0 |

| 1F | | | | |
|---|---|---|---|---|
| | 0 | 0 | 0 | 1 |
| | 1 | 0 | 0 | 1 |
| * | 1 | 1 | 0 | 1 |
| | 0 | 0 | 0 | 0 |

| 0 | | | | |
|---|---|---|---|---|
| | 0 | 0 | 0 | 1 |
| * | 1 | 0 | 1 | 1 |
| | 1 | 0 | 0 | 1 |
| | 0 | 0 | 0 | 0 |

| 21 | | | | |
|---|---|---|---|---|
| * | 0 | 1 | 1 | 1 |
| | 0 | 0 | 1 | 1 |
| | 0 | 0 | 0 | 1 |
| | 0 | 0 | 0 | 0 |

| 0 | | | | |
|---|---|---|---|---|
| | 0 | 0 | 1 | 1 |
| * | 1 | 0 | 1 | 1 |
| | 0 | 0 | 0 | 1 |
| | 0 | 0 | 0 | 0 |

| 22F | | | | |
|---|---|---|---|---|
| | 0 | 0 | 1 | 0 |
| | 1 | 0 | 1 | 0 |
| | 0 | 0 | 0 | 0 |
| * | 1 | 1 | 1 | 0 |

| 23F | | | | |
|---|---|---|---|---|
| | 0 | 0 | 0 | 0 |
| | 1 | 0 | 0 | 0 |
| * | 1 | 1 | 0 | 1 |
| | 1 | 1 | 0 | 0 |

| 24F | | | | |
|---|---|---|---|---|
| * | 0 | 1 | 1 | 1 |
| | 0 | 0 | 0 | 0 |
| | 0 | 1 | 0 | 1 |
| | 0 | 1 | 0 | 0 |

| 23 | | | | |
|---|---|---|---|---|
| | 0 | 1 | 0 | 1 |
| | 0 | 0 | 0 | 0 |
| * | 1 | 1 | 0 | 1 |
| | 0 | 1 | 0 | 0 |

| 25F | | | | |
|---|---|---|---|---|
| | 0 | 0 | 0 | 1 |
| * | 1 | 0 | 1 | 1 |
| | 1 | 0 | 0 | 1 |
| | 0 | 0 | 0 | 0 |

- **Ejercicio 3**:

  La solución de cada apartado se muestra a continuación:

  a) La secuencia de accesos (donde se indica el número de página en hexadecimal, seguido de una letra minúscula l/e indicando si el acceso es de lectura o escritura) sería:

  3e, 4e, 7l, Fe, 0l, tic de reloj, 0l, 3e, Fe, 5l

  Los 4 primeros accesos dejan los marcos físicos y los bits de referencia como siguen:

  3(rm), 4(rm), 7(r), F(rm)

  El acceso 0l provoca un fallo de página. Aplicando NRU los marcos y bits de referencia quedan

  3(rm), 4(rm), 0(r), F(rm)

  Tras el tic de reloj los marcos y bits quedan

  3(m), 4(m), 0(), F(m)

  Los accesos 0l, 3e y Fe (aciertos) dejan los marcos y bits como sigue

  3(rm), 4(m), 0(r), F(rm)

  Finalmente el acceso 5l provoca un fallo de página. Aplicando NRU nos queda

  3(rm), 5(r), 0(r), F(rm)

  b) La secuencia de accesos (donde se indica el número de página seguido de una letra minúscula l/e indicando si el acceso es de lectura o escritura), sería:

  3e, 4e, 7l, Fe, 0l, tic de reloj, 0l, 3e, Fe, 5l

  Los 4 primeros accesos dejan los marcos físicos y el bit r como sigue:

  3(r), 4(r), 7(r), F(r)

  El acceso 0l provoca un fallo de página. Aplicando «FIFO segunda oportunidad» los marcos y bits r quedan

0(r), 4(), 7(), F()

El tic de reloj haría que el sistema desactivase los bits r

0(), 4(), 7(), F()

El acceso 0l (acierto) haría

0(r), 4(), 7(), F()

El acceso 3e provoca un fallo de página y se aplica «FIFO segunda oportunidad»

0(r), 3(r), 7(), F()

El acceso Fe (acierto) deja los marcos y bits r como sigue

0(r), 3(r), 7(), F(r)

Finalmente el acceso 5l provoca un fallo de página y el sistema queda (aplicando FIFO segunda oportunidad)

0(r), 3(r), 5(r), F(r)

- **Ejercicio 4**:

  Veamos cómo se resuelve cada apartado:

  a) Para calcular el tamaño del espacio de direcciones virtuales de cada proceso debemos calcular en primer lugar el tamaño de la dirección virtual. Para ello debemos tener en cuenta el tamaño de las tablas de cada nivel, lo que determinará el número de bits de la dirección virtual que necesitaremos para indexarlas. Para los niveles uno, dos y tres necesitaremos 10, 9 y 8 bits respectivamente, por lo que el tamaño del Número de Página Virtual (NPV) será de 27 bits. Además, el tamaño de página es de 4 KiB, lo que implica que el desplazamiento de byte dentro de página de la dirección virtual es de 12 bits. Se requieren, por lo tanto, un total de 39 bits para las direcciones virtuales, con lo que el **espacio de direcciones virtuales** es de $2^{39}$ bytes o lo que es lo mismo **512 GiB**.

     Para las direcciones físicas, sabemos que disponemos de 15 bits para codificar el Número de Marco de Página, con lo que el tamaño de la dirección física será de 27 bits (15 + 12 del desplazamiento de byte), y el **tamaño máximo de la memoria física será de 128 MiB**.

  b) Para calcular las direcciones virtuales a las que ha accedido el proceso, debemos fijarnos en las entradas de tercer nivel que están ocupadas (las traducciones disponibles). Para saber si el acceso fue de lectura o escritura, bastará con ver si el bit M en la entrada de tercer nivel correspondiente está activado o no.

     Comenzando por la traducción de más arriba, vemos que se alcanza desde la entrada 0 de la tabla de primer nivel ($00\,0000\,0000_{)2}$), la entrada 2 de la tabla de segundo nivel ($0\,0000\,0010_{)2}$) y la entrada 255 de la tabla de tercer nivel ($1111\,1111_{)2}$), con lo que la página virtual sería $0000000000\,000000010\,11111111_{)2}$ = **0x02FF**. Esta página virtual ha sido escrita.

     La traducción de en medio se corresponde con la página virtual $0000000011\,000000010\,00000000_{)2}$ = **0x060200**. Esta página virtual solamente ha sido leída.

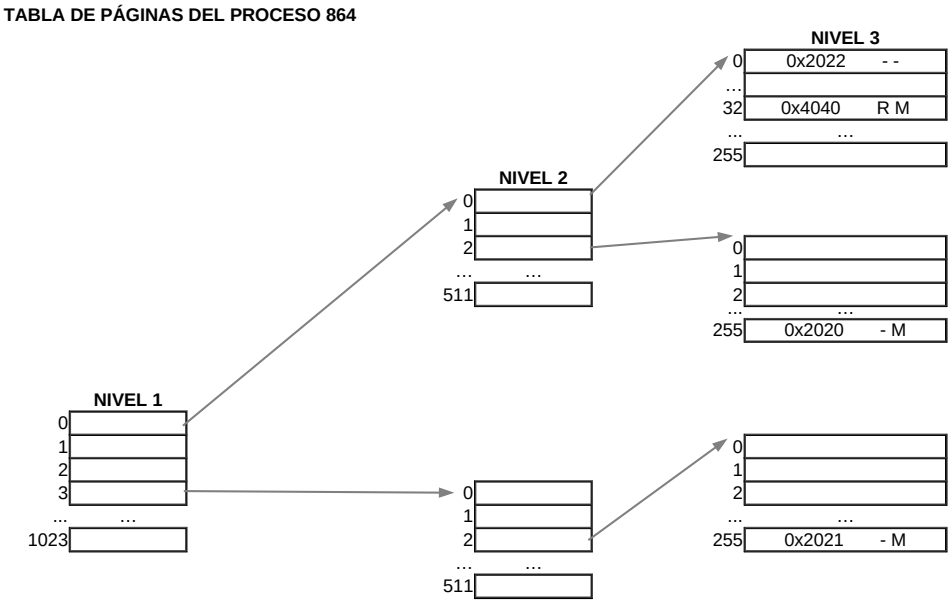
     La traducción de más abajo sería para la página virtual $0000000011\,000000010\,11111111_{)2}$ = **0x0602FF**. Esta página virtual ha sido escrita.

c) El primer acceso a memoria es a la dirección virtual 0x00, con lo que el número de página virtual es la ristra de bits todos a 0. Eso hace que se utilice la primera entrada de cada uno de los 3 niveles de tablas. El fallo se detectaría al acceder a la tabla de segundo nivel, pues su entrada 0 no tiene vinculada la tabla de tercer nivel, e implicaría la creación de una nueva tabla de páginas de tercer nivel. Dado que no se han producido 2 fallos de página consecutivos aún con los 3 marcos de páginas ocupados (de hecho este es el primero), habrá que aplicar el algoritmo de reemplazo NRU, que hará que se reemplace la página 0x060200 y su marco de página (0x2022) sea asignado a la página virtual 0. Dado que el acceso es de lectura, se activará solamente el bit `R` (no el `M`). La traducción se almacenará en la primera entrada (entrada 0) de la nueva tabla de páginas de tercer nivel creada.

El TIC provocará que todos los bits `R` se pongan a 0.

El segundo acceso a memoria es a la página virtual 0x020, que tampoco está en memoria, y por lo tanto, ocasiona un fallo de página. En este momento el SO otorgará un nuevo marco de página al proceso (concretamente el 0x4040) y la traducción se almacenará en la entrada 32 de la tabla de páginas de tercer nivel creada como consecuencia del acceso a memoria anterior. Dado que el acceso es de escritura, se activarán los bits `R` y `M`.

De esta forma, la tabla de páginas quedaría como sigue:




d) Para detectar un acierto de página (AP) debemos encontrar la traducción correspondiente. Esta puede estar en el TLB, y por lo tanto se detectaría el acierto en **1 ns** (este sería el **tiempo mínimo para detectar un AP**), o requeriría acceder a la tabla de páginas (a los 3 niveles), con lo que se requerían **31 ns** (acceso al TLB + acceso a los 3 niveles de la TP) y este sería **el tiempo máximo para detectar un AP**.

Para detectar un fallo de página (FP) se accedería primeramente al TLB (1 ns), y al no encontrar la traducción ahí, se continuaría después por la tabla de primer nivel (10 ns). Si ahí no se encuentra correspondencia con una tabla de segundo nivel es porque hay un fallo de página, y por lo tanto, esta situación es la que determina el **tiempo mínimo para detectar un FP** que será de **11 ns**. El **tiempo máximo para detectar un FP** se da cuando en la tabla de tercer nivel no se encuentra la traducción buscada, y será, por lo tanto, de **31 ns**.

- **Ejercicio 5**:

La solución de cada apartado se muestra a continuación:

a) En una tabla de páginas invertida, el número de entradas de la tabla de traducción es igual al número de marcos de página existentes. Vemos que se emplean 20 bits para el NMP, con lo que el tamaño de las direcciones físicas es de 32 bits (20 bits para el NMP + 12 bits para el desplazamiento de byte dentro de la página). Esto implica que el tamaño máximo de la memoria física en este sistema es de 4 GiB.

   Para el NPV vemos que se utilizan 28 bits, con lo que aplicando lo anterior, vemos que el tamaño de las direcciones virtuales es de 40 bits, y por lo tanto, el tamaño de la memoria virtual de cada proceso es de 1 TiB.

b) Dado que la función de dispersión devuelve 10 bits, la tabla de dispersión necesitará 1024 entradas. En cuanto al número de entradas de la tabla de traducción, este es igual al número de marcos de página existentes. Vemos que hay un total de $2^{20}$ marcos de página (se usan 20 bits para codificar el número de marco de página), con lo que este será el número de entradas requerido para la tabla de traducción (1 048 576 entradas).

c) Tanto en el campo `Número` de la tabla de dispersión como en el campo `Encaden` de la tabla de traducción hay que almacenar un número de entrada de la tabla de traducción. Se requerirán, por lo tanto, 20 bits en cada caso.

d) El acceso a la dirección `0x2233440123` por parte del proceso con PID `768` provoca un fallo de página. Para detectarlo, se accede primero a la entrada 64 de la tabla de dispersión (a partir del valor de los 10 bits menos significativos del NPV `0x2233440`). Esta entrada apunta a la entrada 2 de la tabla de traducción, que se usaría para comparar el PID y NPV que almacena con los buscados. Dado que no hay coincidencia se seguiría buscando a partir de la entrada a la cual apunta, es decir, la 157. A partir de la 157, iríamos a la 158 (aquí se encontraría coincidencia con el NPV pero no con el PID), de ahí a la 159, y de ahí a la 1024, en la que dado que la información de encadenamiento no nos lleva a una nueva entrada (vale 1024) se detectaría el fallo de página.

e) Podríamos detectar un fallo de página si el bit de validez en la entrada correspondiente de la tabla de dispersión vale 0 (un acceso a memoria), lo que daría lugar al caso con menor número de accesos a memoria (1) y un coste total de 10 ns. Este sería el tiempo mínimo requerido para detectar un fallo.

   El extremo contrario lo tendríamos cuando se detectase el fallo de página después de recorrer la lista de mayor tamaño (34 entradas de la tabla de traducción). En este caso el número de accesos a memoria que se requeriría sería 1 (tabla de dispersión) + 34 (tabla de traducción), o lo que es lo mismo, el tiempo máximo requerido para detectar un fallo sería de 350 ns.

   A partir de lo anterior, el tiempo medio para detectar un fallo de página sería $(1+8,75) \times 10 = 97,5$ ns.