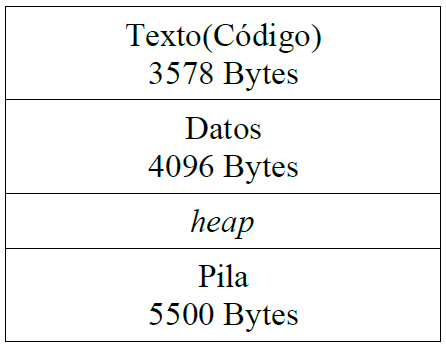
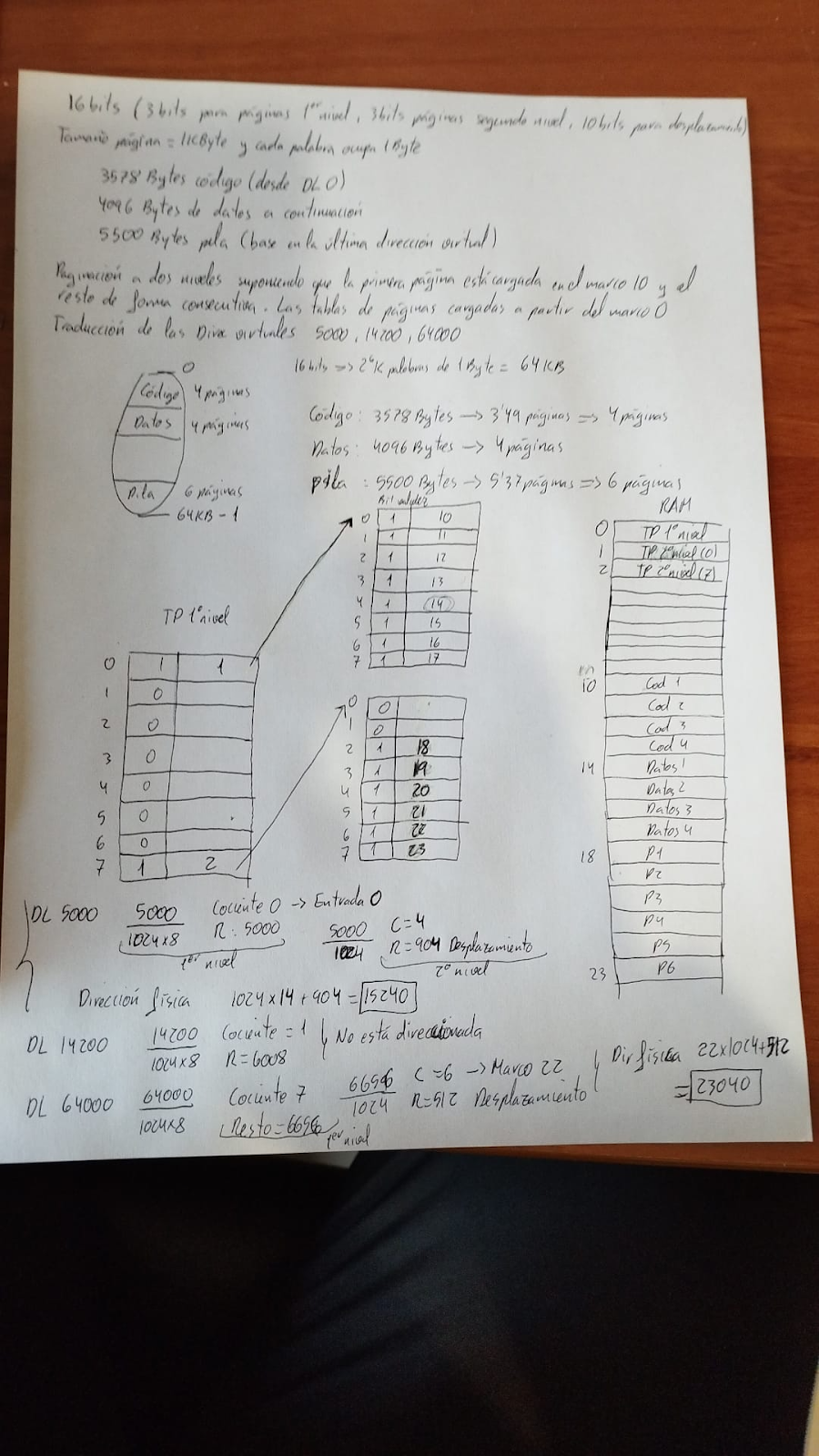
**TEMA 3**

**1. Suponga un espacio virtual de direcciones de 16 bits (3 bits para la tabla de páginas de primer nivel, 3 bits para las tablas de páginas de segundo nivel y 10 bits para el desplazamiento). El tamaño de página es 1Kbyte y cada palabra de memoria ocupa 1Byte. El mapa de memoria de un proceso es el siguiente: 3578 Bytes de texto (código que comienza en la dirección virtual 0), 4096 Bytes de datos (a continuación del código) y 5500 Bytes de pila (cuya base está el la última dirección virtual).**

**La Figura muestra las tres regiones en el espacio virtual. Construya las tablas de páginas necesarias para describir este espacio de memoria suponiendo apaginación a dos niveles, y suponiendo que la primera página está cargada en el marco 10 y el resto de forma consecutiva. Las tablas de páginas están cargadas a partir del marco 0. A continuación, realice la traducción de las siguientes direcciones virtuales: 5000, 14200 y 64000 y ponga todos los cálculos utilizados.**



**2. A partir del espacio de direcciones virtuales descrito en el ejercicio anterior:**

**a) Representar las estructuras de datos (mm\_struct y vm\_area) que utilizaría Linux para describir dicho espacio, rellenando aquellos campos de las estructura que sean posibles y suponiendo que el archivo ejecutable de dicho proceso reside en un archivo denominado “./mi\_programa”.**

MMSTRUCT:

-Vm\_area\_struct\* mmap : lista de vm\_areas

-pgd: tabla de páginas de primer nivel.

-start\_code: 0

-end\_code: 3577

VM\_AREA (REGIÓN DE TEXTO): tamaño 3578 bytes

-end: 3577

-start: 0x0000

-indicadores: VM\_READ, VM\_EXEC

-file: “./mi\_programa”

-vm\_next : apunta a la vm\_struct de la región de datos

VM\_AREA (REGIÓN DE DATOS): tamaño 4096 bytes

-end: 7673

-start:3578

-file: “./mi\_programa”

-indicadores: VM\_READ, VM\_WRITE

-vm\_next : apunta a la vm\_struct de la región de pila

VM\_AREA (REGIÓN DE PILA):  tamaño 5500 bytes

-end: 64K-1

-start: 60035

-file: “./mi\_programa”

-indicadores: VM\_READ, VM\_WRITE, VM\_GROWSDOWN (faltaba este se habla de él en las diapositivas y lo he agregado)

-vm\_next : NULLPTR

**b) Qué modificaciones habría que realizar en las estructuras de datos antes citadas, si se realiza una proyección de un archivo de datos de 3500 bytes en la dirección virtual 16384.**

Se crea una nueva vm\_area con inicio en la dirección lógica 16384 y fin en 16384+3500 1.

**3. (a) Indicar los pasos que sigue un sistema operativo general en el tratamiento de una excepción de falta de página. (b) En el caso de Linux, ¿qué papel juegan las vm-areas en el tratamiento de la excepción del falta de página?**

1. Etapas de la rutina de tratamiento de la excepción falta de página:

1. Mira si la referencia es válida, si no lo es aborta al proceso. Consulta las vm-areas y a partir de ahí determina si es válida o no.

2. Si es válida pero no está en memoria.

3. Obtiene un marco vacío.

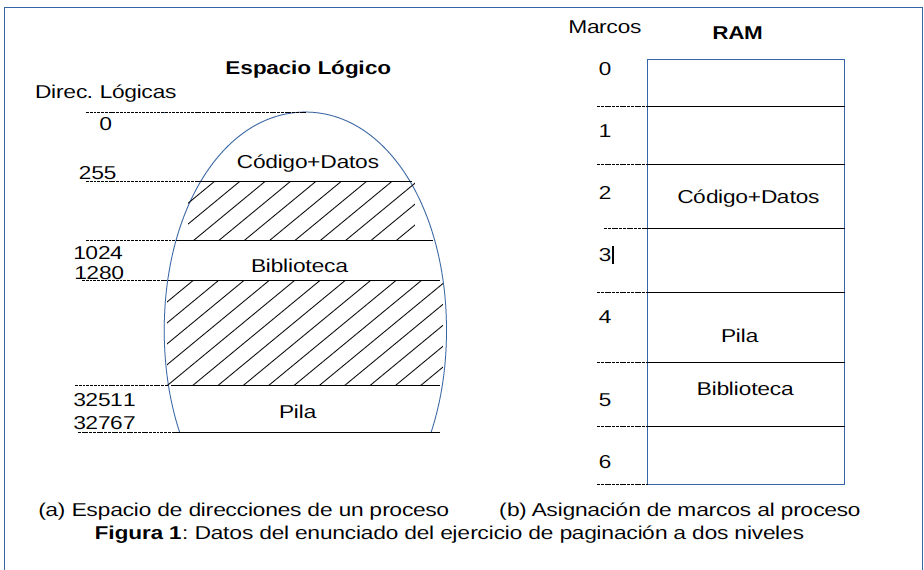
4. Carga la página faltante en el marco libre.

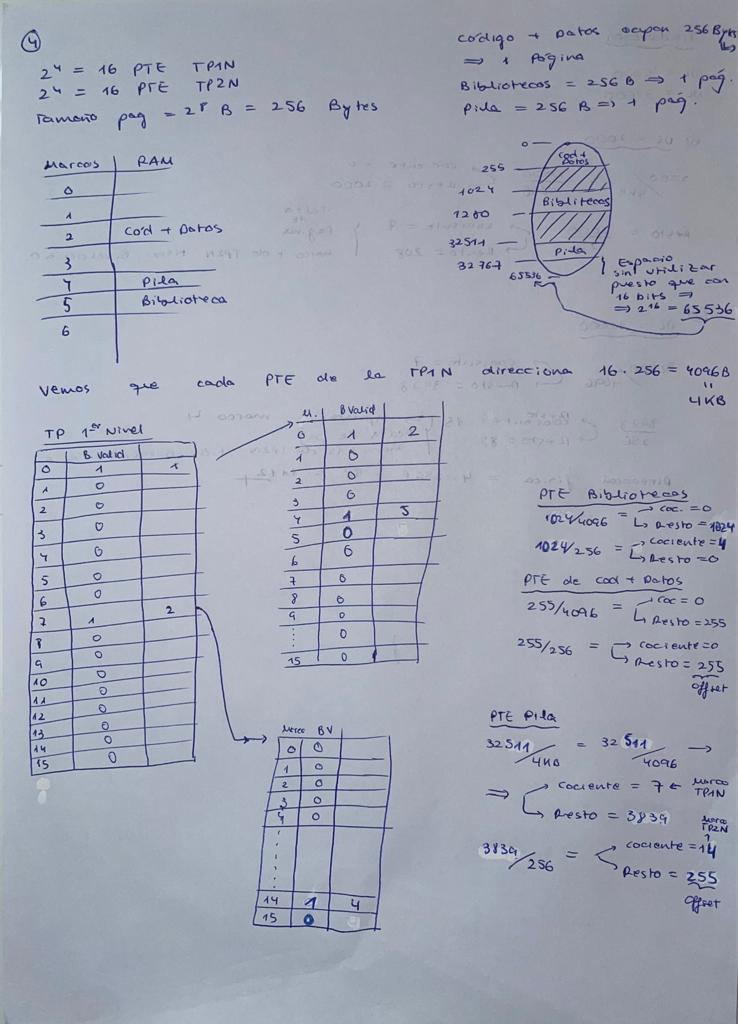
5. Activa el bit de validez.

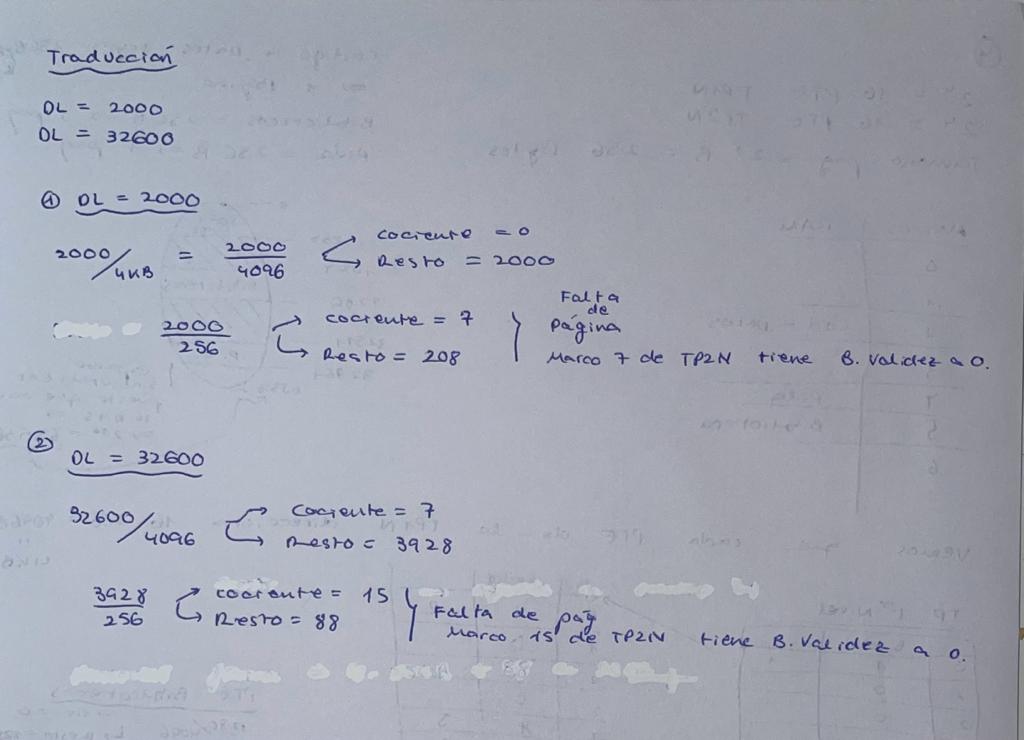
6. Rearranca la instrucción que produjo la falta.

1. Sabemos que en Linux, por cada región de memoria en uso del ED hay un vm-area que la describe, pues bien, si hay una falta de página, se coge la dirección faltante y se comprueba si está en alguna de las vm-area. Si está en alguno de los huecos, se está haciendo una referencia a una dirección inválida. Si no, ya se determina cuál es la vm-area a utilizar. La vm-area mantiene el almacén de respaldo de la región e información para recuperar la página faltante. Localizada esta se busca un marco libre, se carga página en él y se construye la PTE correspondiente, tras lo cual se re-arranca la instrucción faltante.

**4.Suponga un espacio virtual de direcciones de 16 bits en un sistema paginado a doble nivel que direcciona byte (4 bits para la tabla de páginas de primer nivel, 4 bits para la tabla de página de segundo nivel y 8 bits para el desplazamiento u offset). El mapa de memoria sobre el espacio lógico de un proceso y la asignación de memoria se muestra en la Figura 1. Las direcciones de (a) marcan el inicio y fin de la región, en (b) se muestra en qué marcos están cargadas cada una de las páginas del proceso. Construya las tablas de páginas y sus contenidos más relevantes (marco, bit de presencia/validez) para describir el espacio de la Figura 1(a). A continuación, realice la traducción de las siguientes direcciones virtuales: 2000 y 32600 (indicando todos los cálculos utilizados).**



****

****

**5. (a) Dibujar/describir las estructuras de datos que utiliza el kernel de Linux para representar el espacio de usuario del proceso descrito en el ejercicio anterior e indicar los contenidos más relevantes. (b) Supongamos que el proceso realiza una invocación de malloc() para reservar 500B de memoria adicionales para datos dinámicos. Cómo cambiaría la descripción realizada en el Apartado (a).**

1. El kernel utilizaría tres vm-áreas en representación a cada una de las regiones del ejercicio, es decir, un vm-área para la región de código+datos, otro para las bibliotecas y un último para la pila.
2. Dado que estamos añadiendo memoria para datos dinámicos, se expande la vm-área de código+datos, es decir, la dirección de fin será ahora de 255+500 (previamente acababa en la 255).

**6. En un sistema Linux, qué acciones toma el kernel cuando el proceso actual (resultado de la ejecución del programa que reside en el archivo ejecutable mi\_programa) genera una falta de página en la región de código.**

Coge la dirección donde salta la falta y comprueba si es válida, es decir, si está en alguna de las vm-área.

Si está en uno de los huecos, estamos haciendo referencia a una dirección inválida y se aborta el proceso.

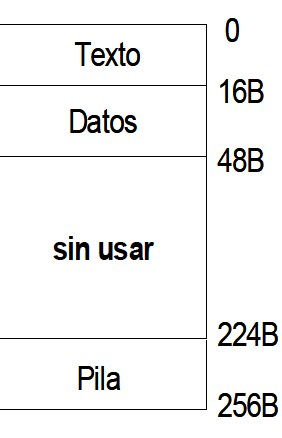
Si no, ya sabe cuál es la vm-área que tiene que utilizar. La vm-área dice dónde está el programa que ha generado la excepción.

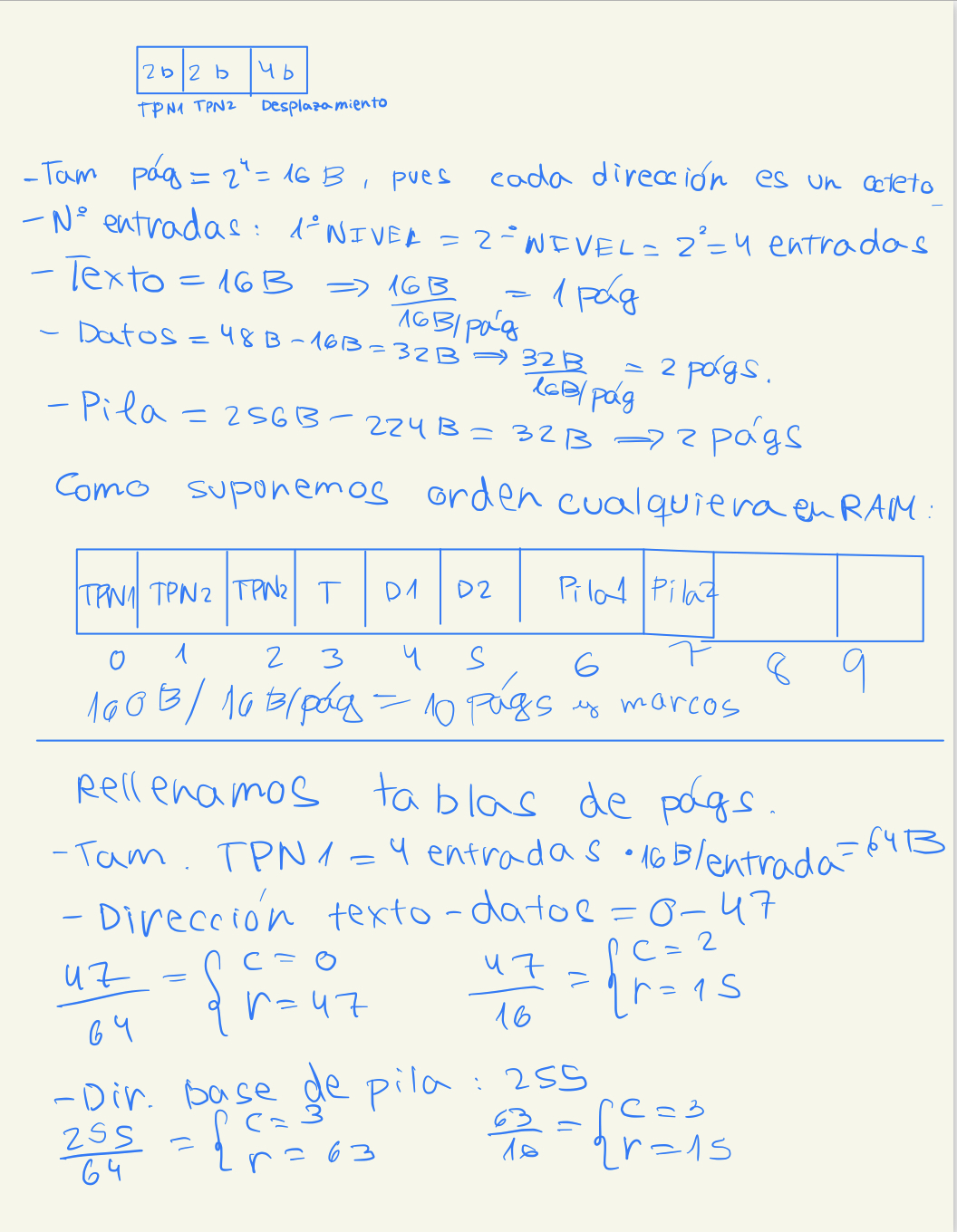
* Si la página no está en memoria (bit de validez a 0), se obtiene un marco vacío y se carga la página faltante en dicho marco libre (se modifica la tabla de páginas). Si no hay ningún marco libre, se emplea el algoritmo LRU para asignarle un marco. Para ello se busca la página que lleva más tiempo sin ser referenciada, se saca de memoria llevándola a disco y se carga la página faltante en el marco que ha quedado libre. Se activa el bit de validez y, finalmente, se rearranca la instrucción donde se produjo la falta.
* Si la página sí que está en memoria (bit de validez a 1), pero la traducción no es válida, se reasigna la página.

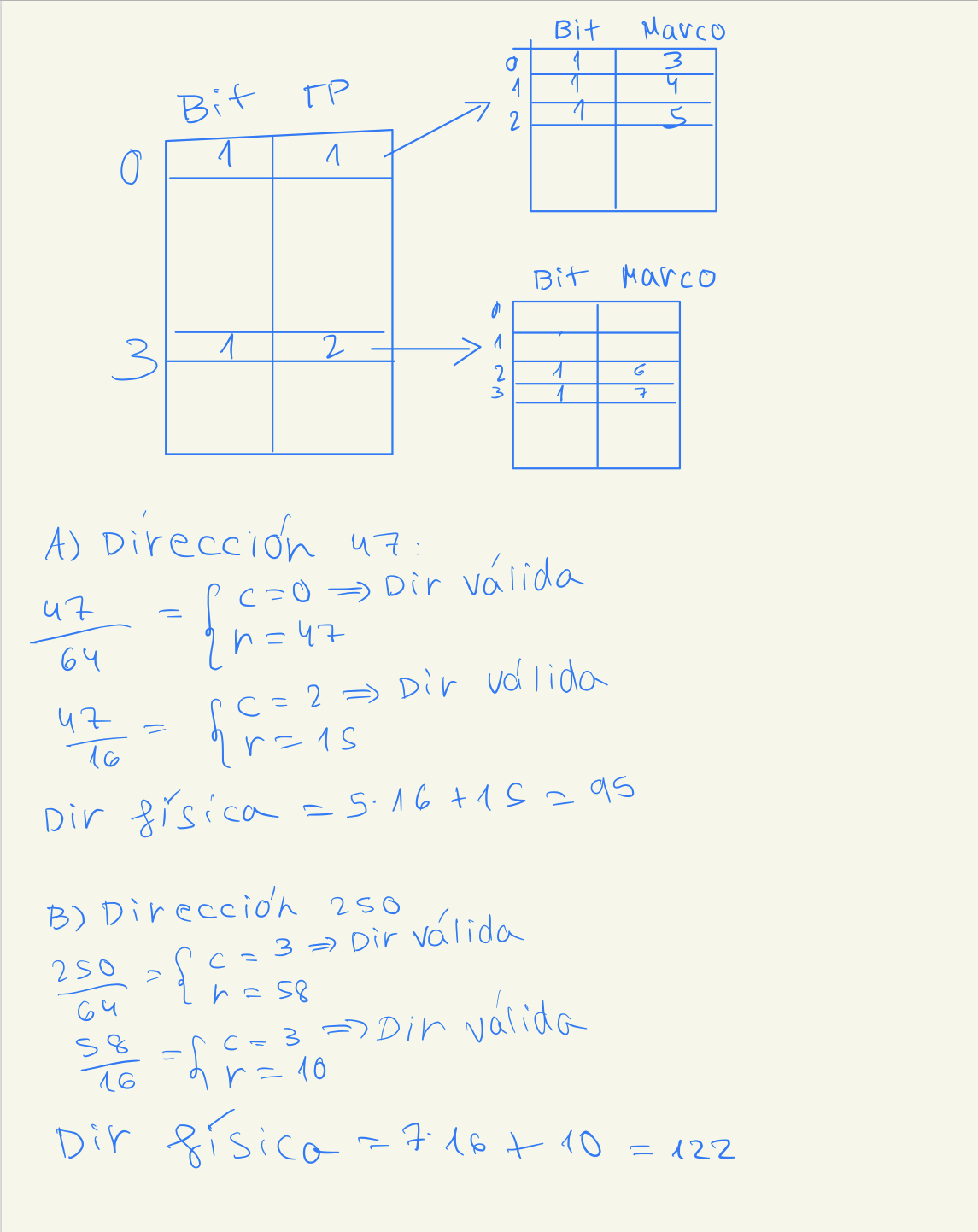
En este caso no hay que comprobar si la dirección es válida porque el enunciado indica que es de una región concreta.

Para localizar la página a partir de la PTE se usa la función nopage de la vmarea.

**7. Suponed un sistema con una memoria principal de 160 Bytes y que utiliza paginación a dos niveles. Las direcciones son de 8 bits con la siguiente estructura: 2 bits en la tabla de páginas de primer nivel, 2 bits en la tabla de páginas de segundo nivel y 4 bits para el desplazamiento. El espacio de direccionamiento virtual de un proceso tiene la estructura del dibujo. Representad gráficamente las tablas de páginas de primer y segundo nivel y sus contenidos, suponiendo que todas las páginas válidas de este proceso están cargadas en memoria principal en el orden que queráis (todos los marcos de página de memoria principal están libres). Dada esa asignación traduce la dirección virtual 47 y la dirección virtual 250.**



****

****