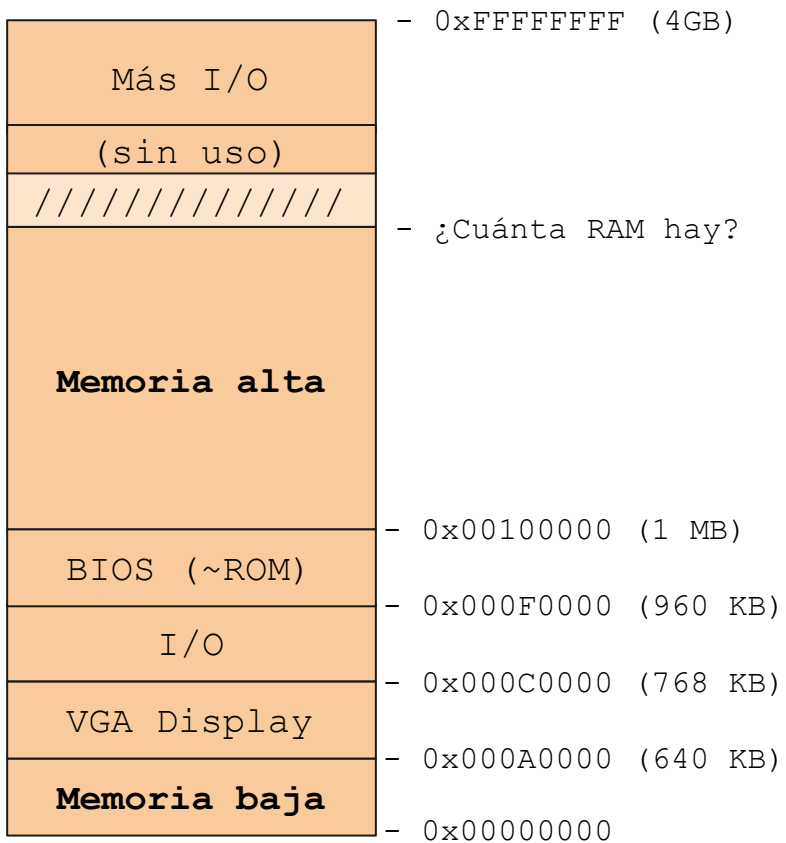
Memoria virtual en JOS

Sistemas Operativos - FIUBA

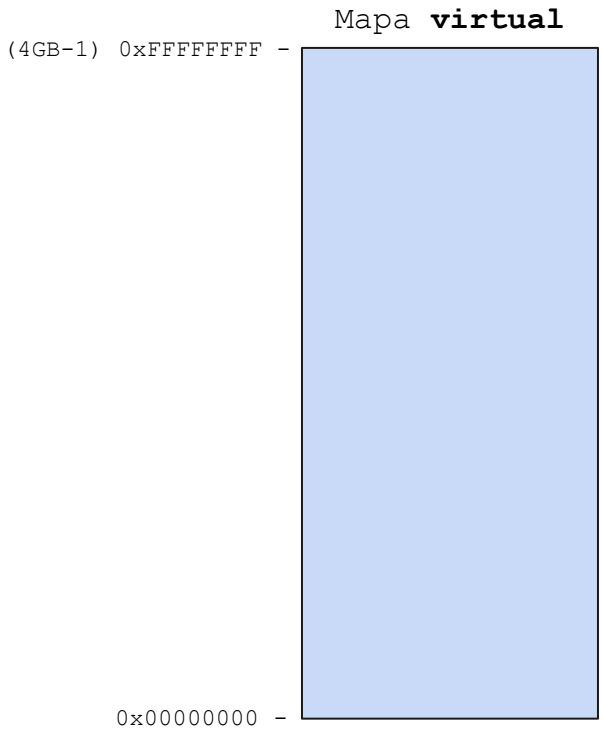
Buscando a JOS...

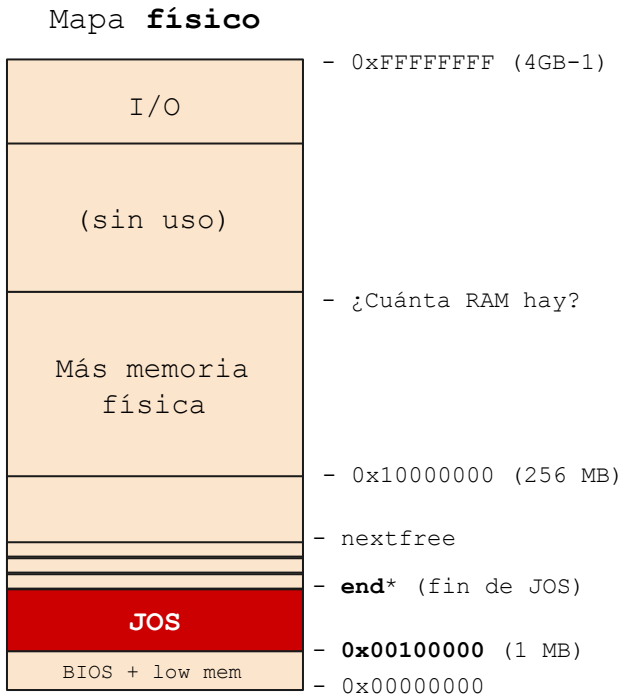
Mapa de memoria física

- Determinado por la cantidad de bits del procesador
- Bloques mapeados a dispositivos
- Bloques mapeados a RAM real

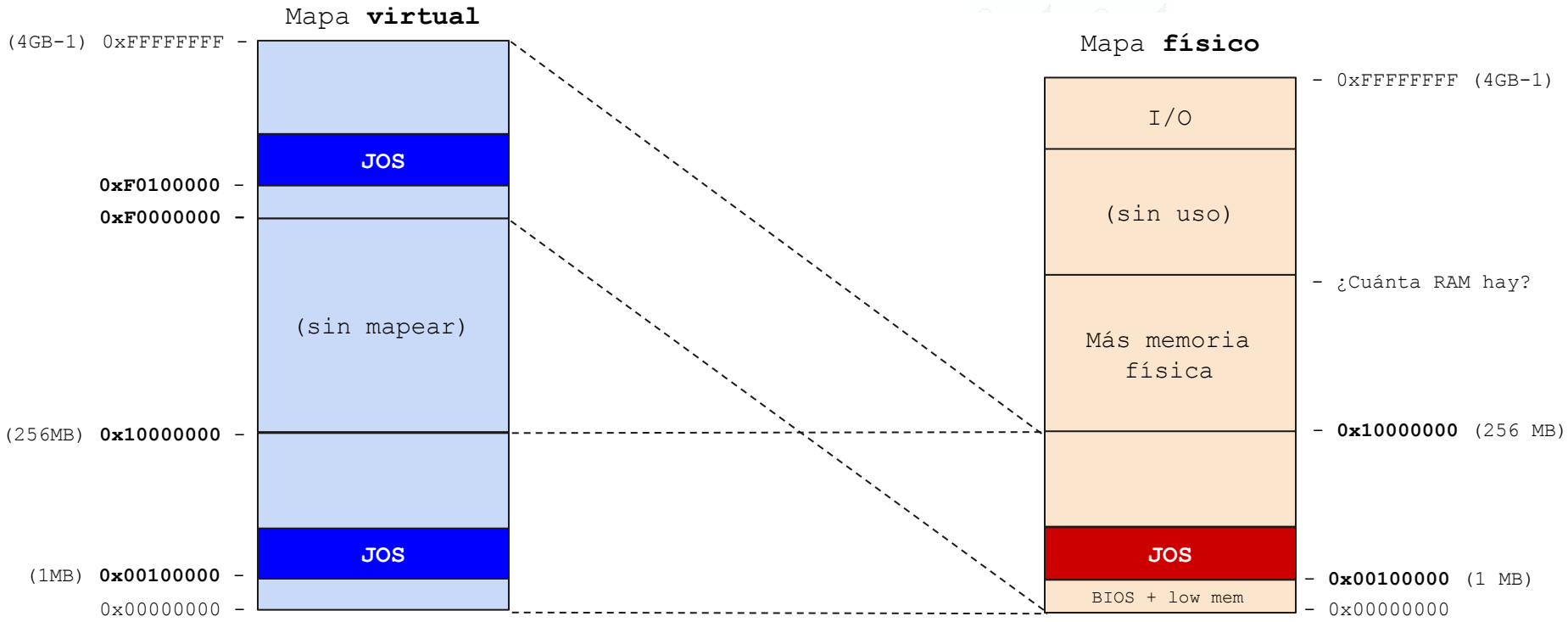


¿Dónde está JOS?

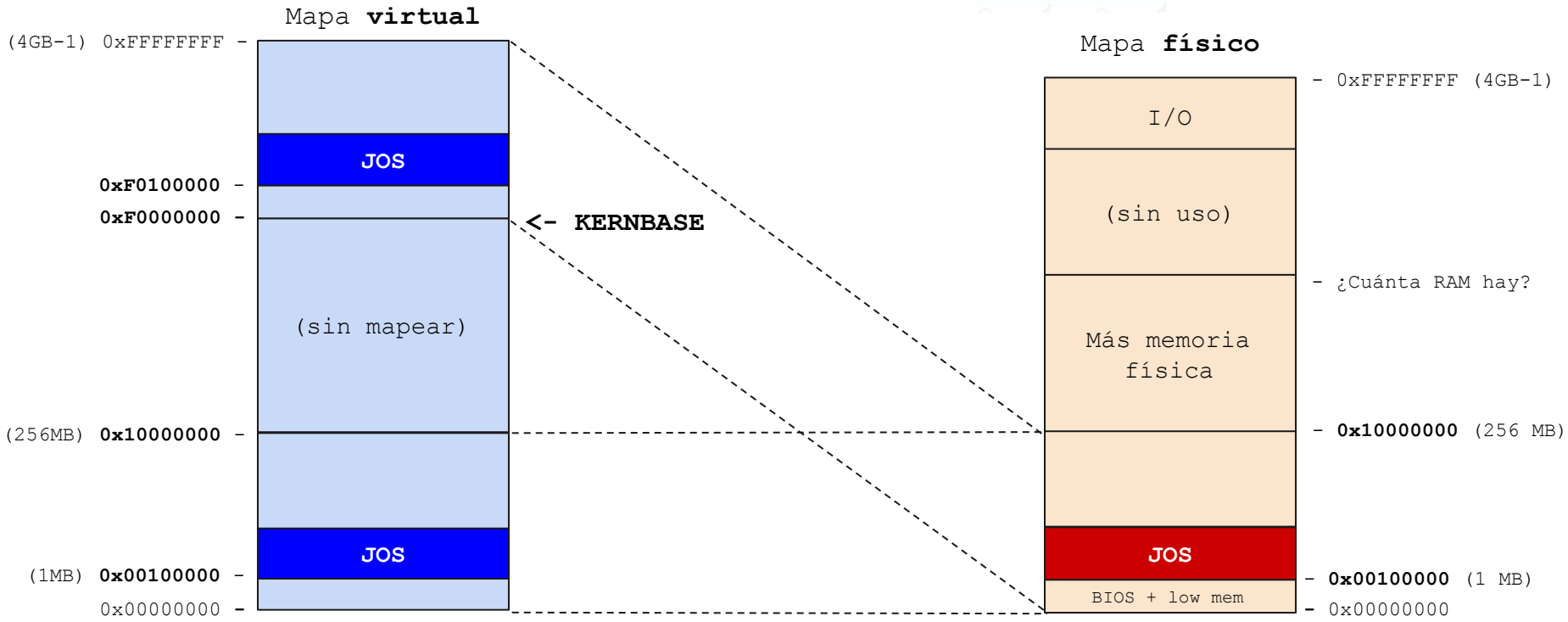




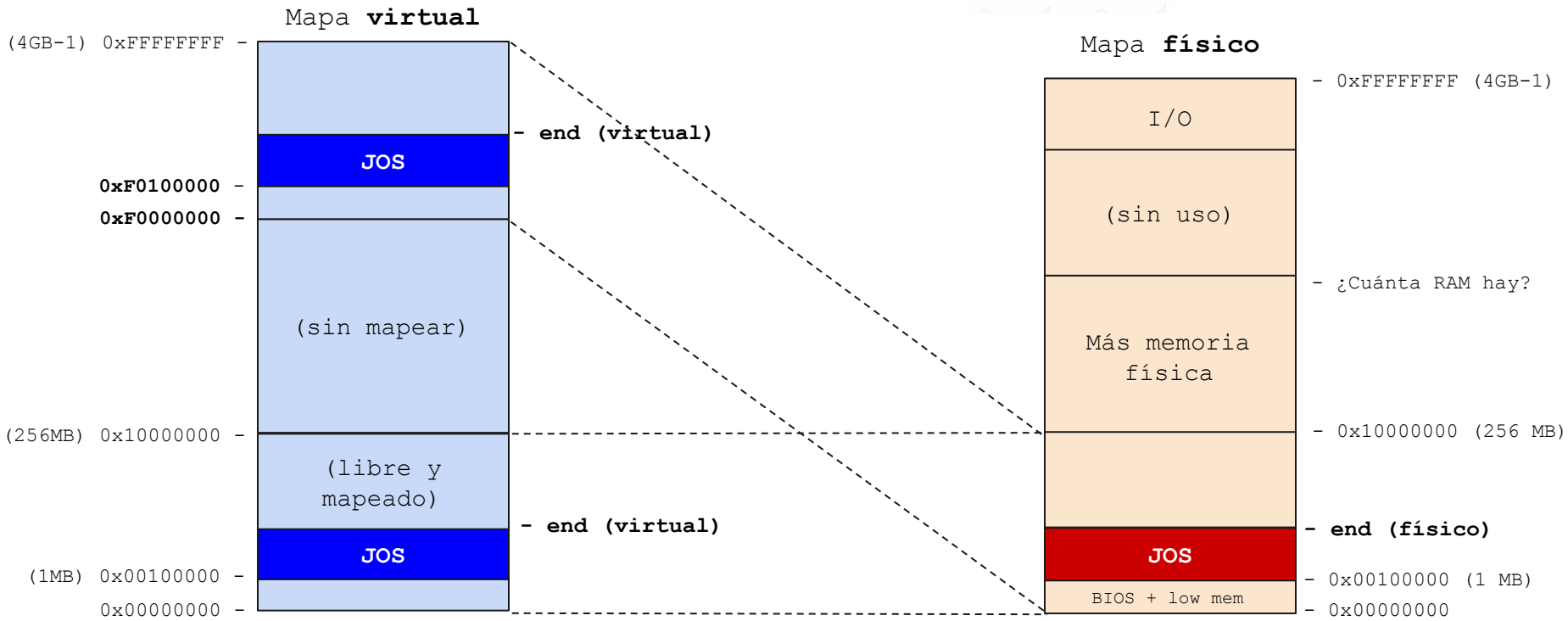
¿Dónde está JOS?



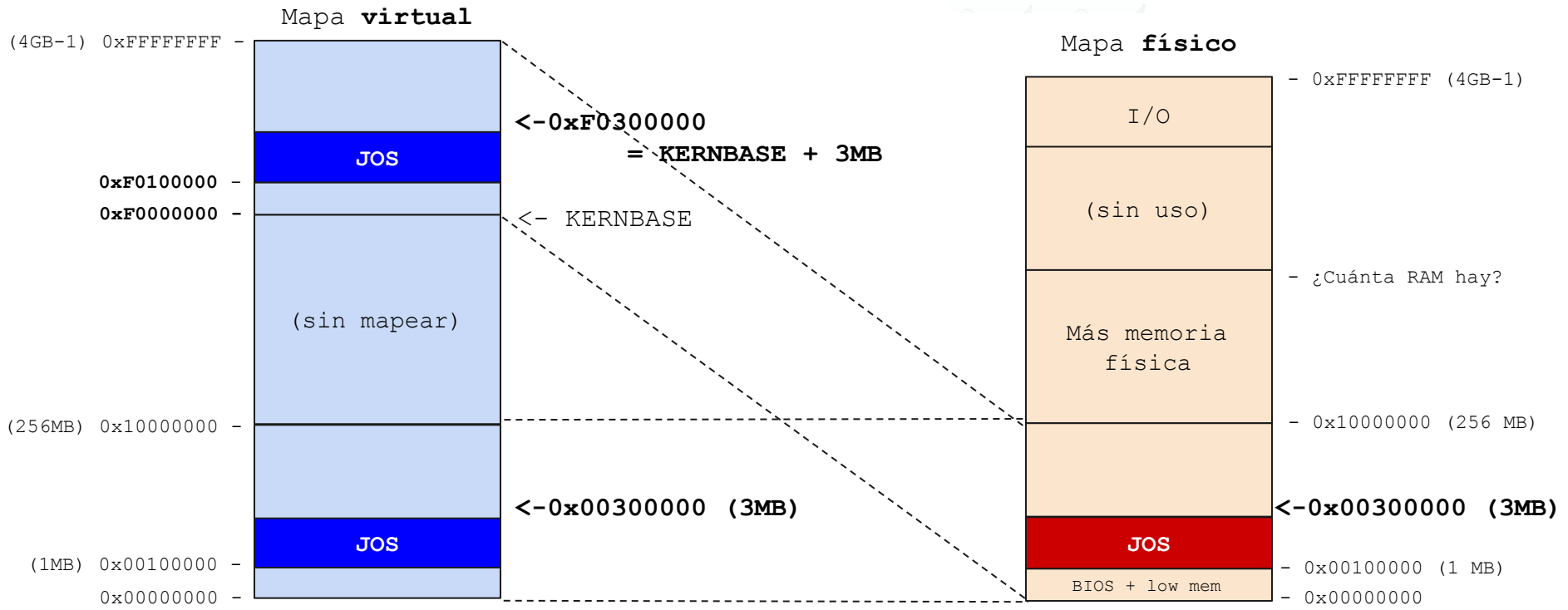
¿Dónde está JOS?



¿Dónde está JOS?

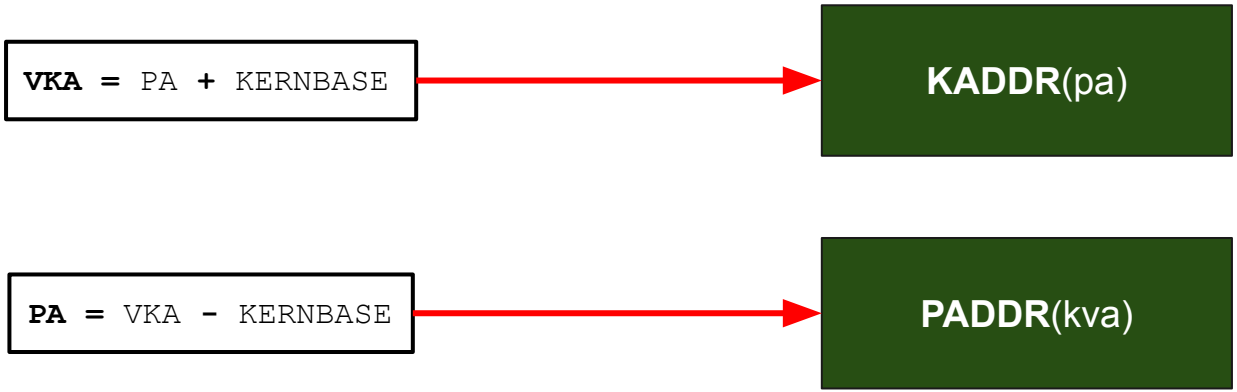


¿Dónde está JOS?



¿Dónde está JOS?

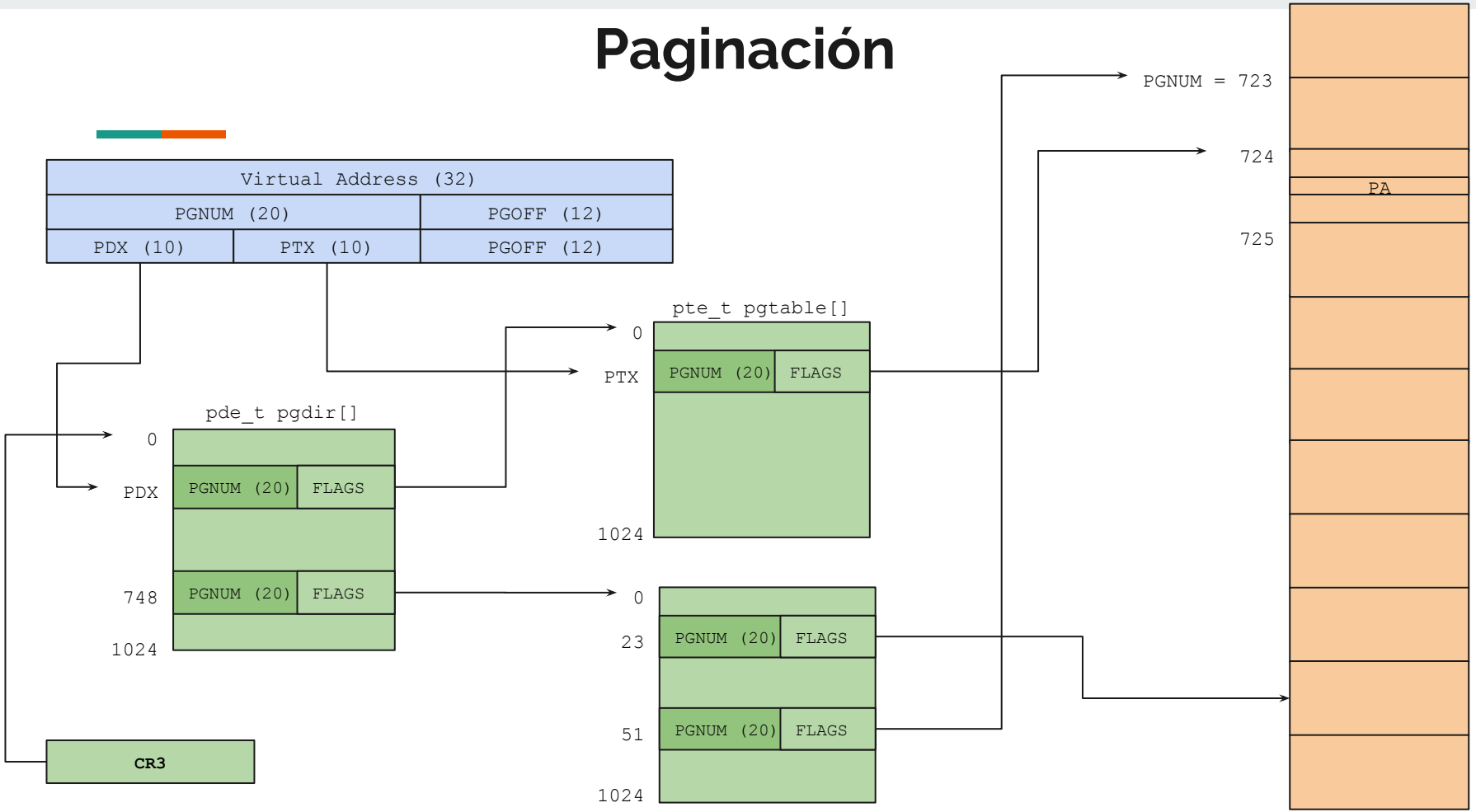
VKA = "Virtual Kernel Address"
PA = "Physical Address"



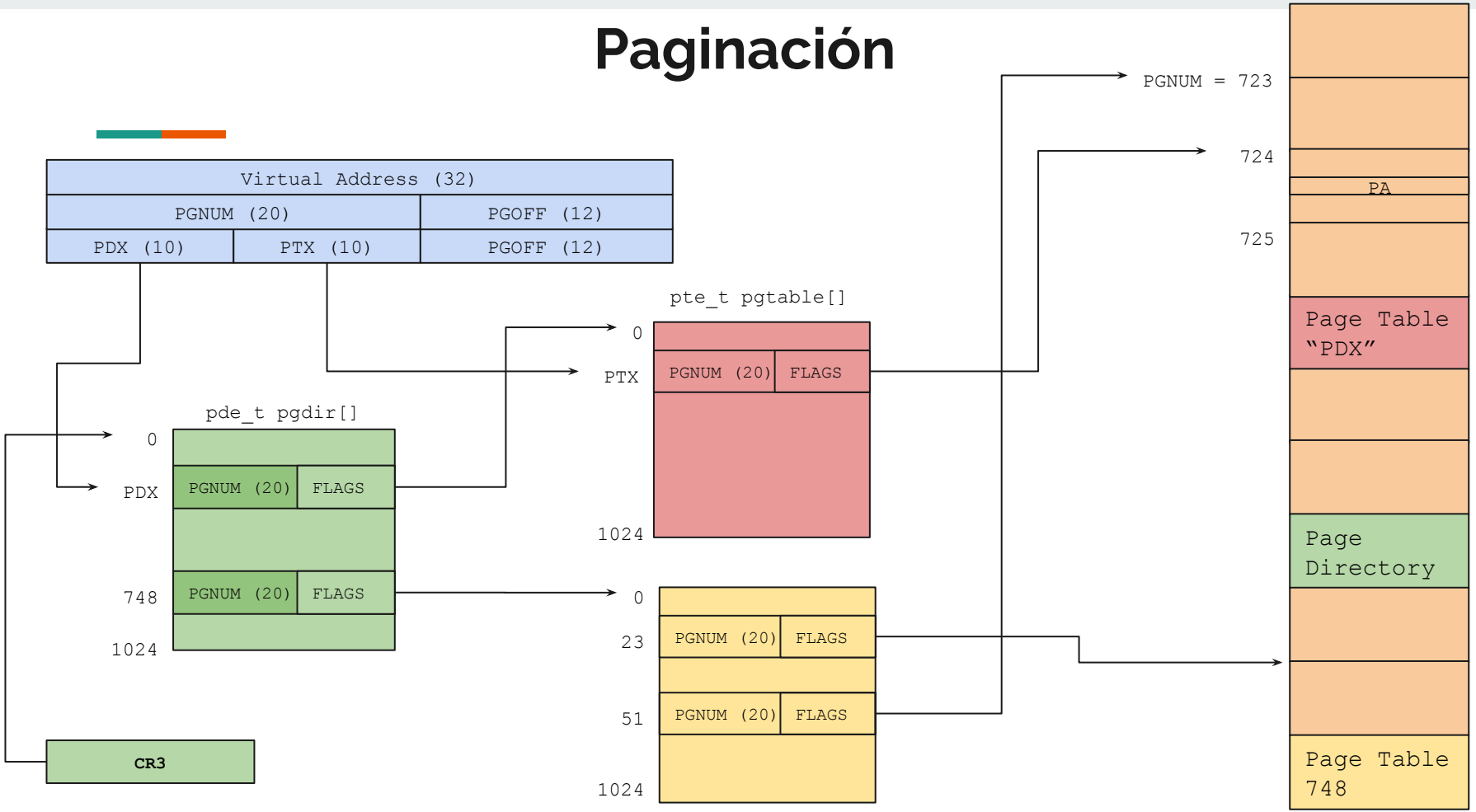
Y para qué sirven **pa2page()**, **page2pa()**, y **page2kva()** ?? (están en kern/pmap.h)

Paginado y *Page-Directory*

Paginación

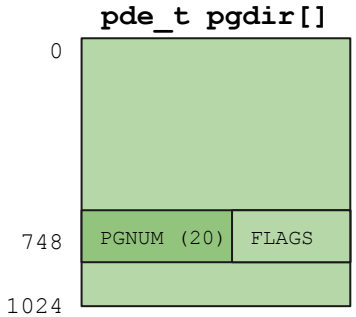

Virtual Address (32)
PGNUM (20)
PGOFF (12)
PDX (10)
PTX (10)
PGOFF (12)
pde_t pgdir[]
0
PDX
PGNUM (20)
FLAGS
748
PGNUM (20)
FLAGS
1024
CR3
pte_t pgtable[]
0
PTX
PGNUM (20)
FLAGS
1024
0
23
PGNUM (20)
FLAGS
51
PGNUM (20)
FLAGS
1024
PGNUM = 723
724
PA
725

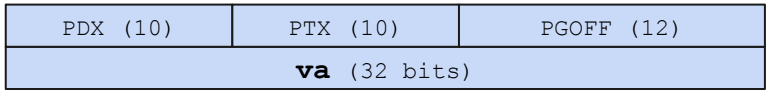
Paginación


Virtual Address (32)
PGNUM (20)
PGOFF (12)
PDX (10)
PTX (10)
PGOFF (12)
pde_t pgdir[]
0
PDX
PGNUM (20)
FLAGS
748
PGNUM (20)
FLAGS
1024
CR3
pte_t pgtable[]
0
PTX
PGNUM (20)
FLAGS
1024
0
23
PGNUM (20)
FLAGS
51
PGNUM (20)
FLAGS
1024
PGNUM = 723
724
PA
725
Page Table "PDX"
Page Directory
Page Table 748

Navegando el *Page-Directory*

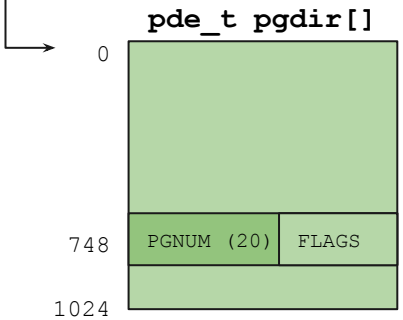
pte_t ***pgdir_walk**(pde_t ***pgdir**, const void ***va,** int **create**)

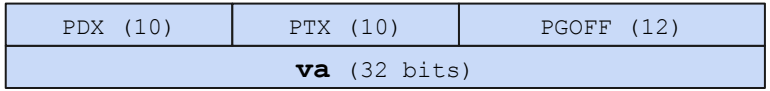




Navegando el *Page-Directory*

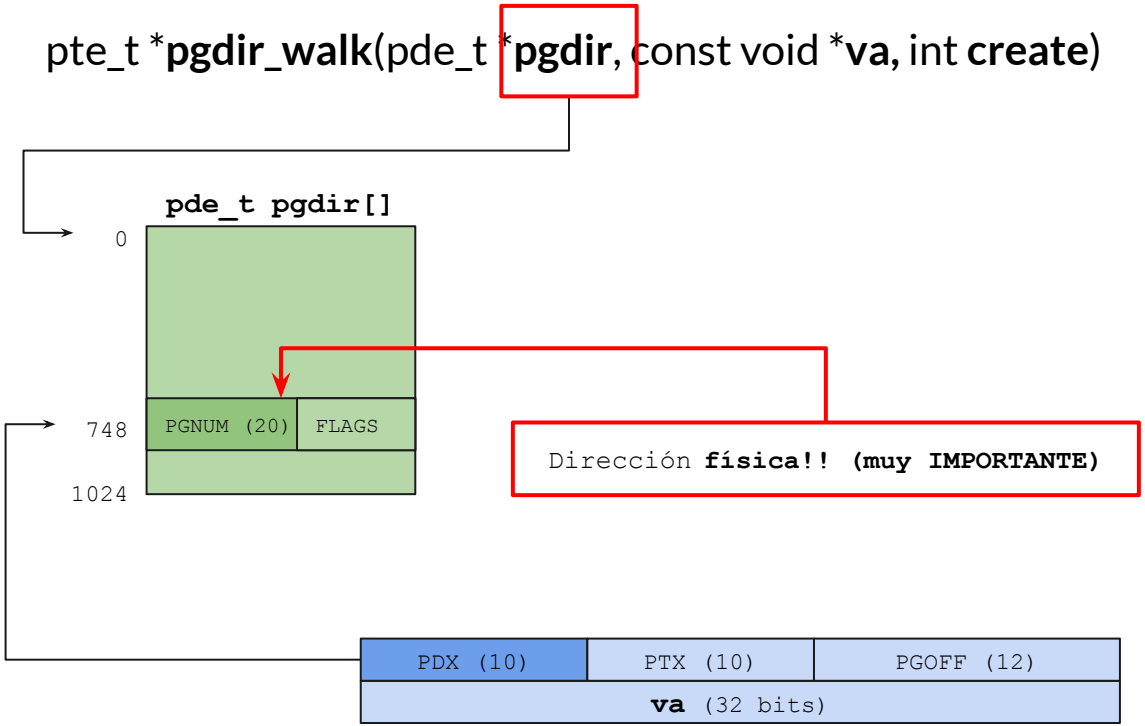
pte_t ***pgdir_walk**(pde_t ***pgdir**, const void ***va,** int **create**)





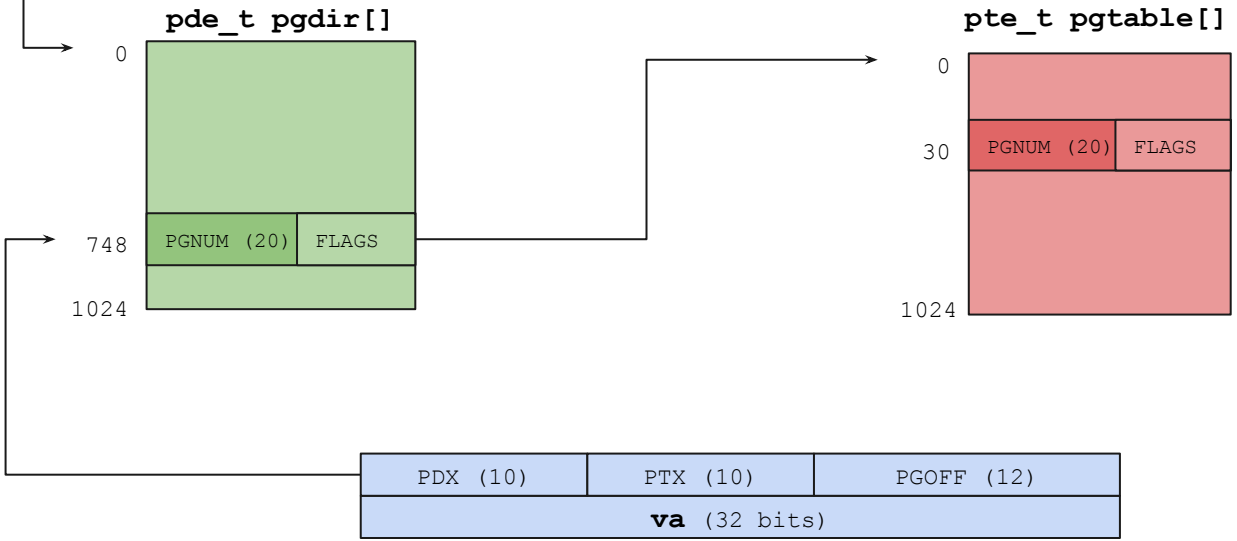
Navegando el *Page-Directory*

pte_t ***pgdir_walk**(pde_t ***pgdir**, const void ***va,** int **create**)



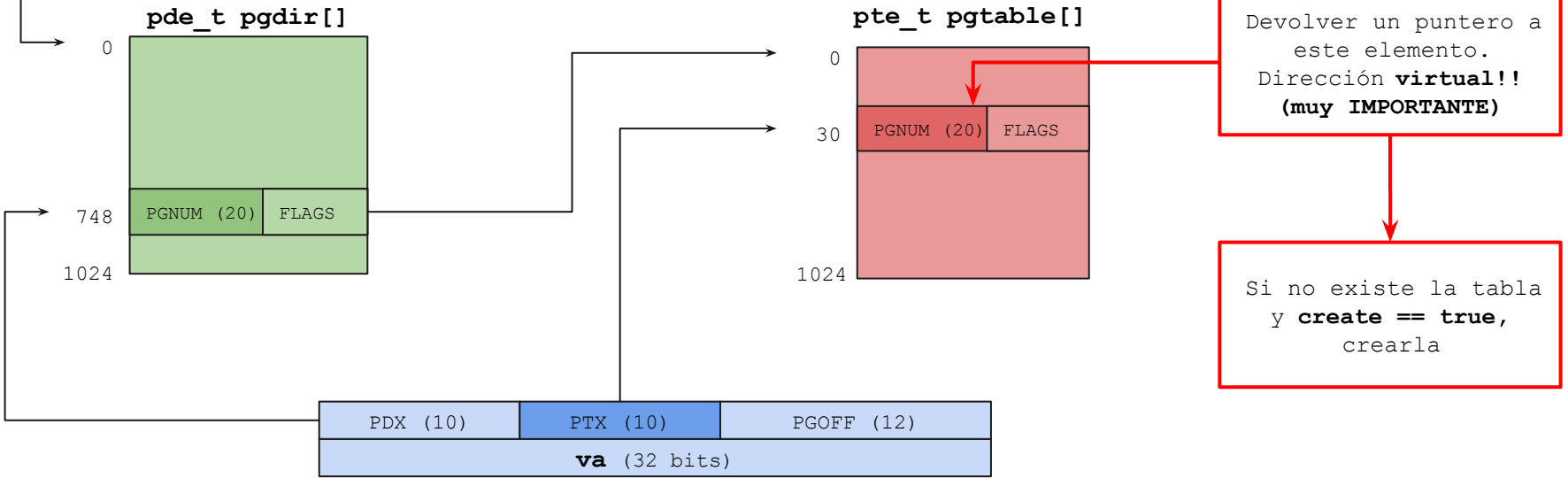
Navegando el *Page-Directory*

pte_t ***pgdir_walk**(pde_t ***pgdir**, const void ***va,** int **create**)



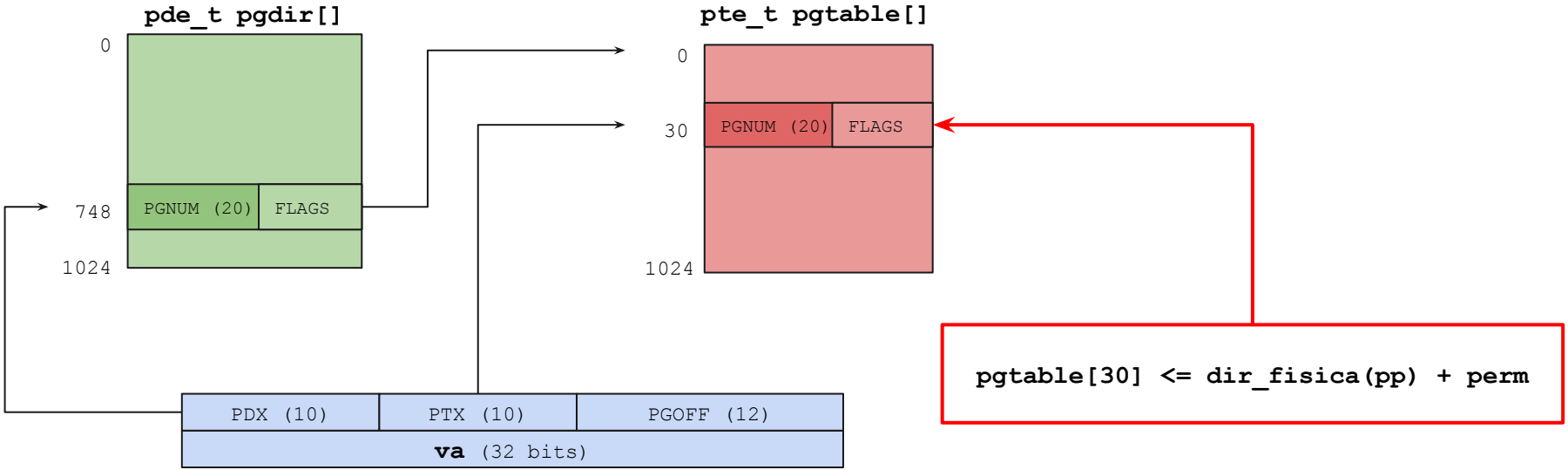
Navegando el *Page-Directory*

pte_t ***pgdir_walk**(pde_t ***pgdir**, const void ***va,** int **create**)



Modificando el *Page-Directory*

int **page_insert**(pde_t ***pgdir**, struct PageInfo ***pp**, void ***va**, int **perm**)



Modificando el *Page-Directory*

int **page_insert**(pde_t ***pgdir**, struct PageInfo ***pp**, void ***va**, int **perm**)

- Leer la documentación de la función en el código
- Qué sucede si ya había una página mapeada en esa misma dirección?
- La parte de invalidar la TLB se puede hacer como paso de la función **page_remove()**
- **pgdir_walk** puede ser útil (pensar qué valor de **create** se le debe pasar)

Modificando el *Page-Directory*

struct PageInfo ***page_lookup**(pde_t ***pgdir**, void ***va,** pte_t ****pte_store**)

- Leer la documentación de la función en el código
- Cuáles son los casos en los cuáles no hay nada mapeado para **va**?
- **pgdir_walk** puede ser útil (pensar qué valor de **create** se le debe pasar)

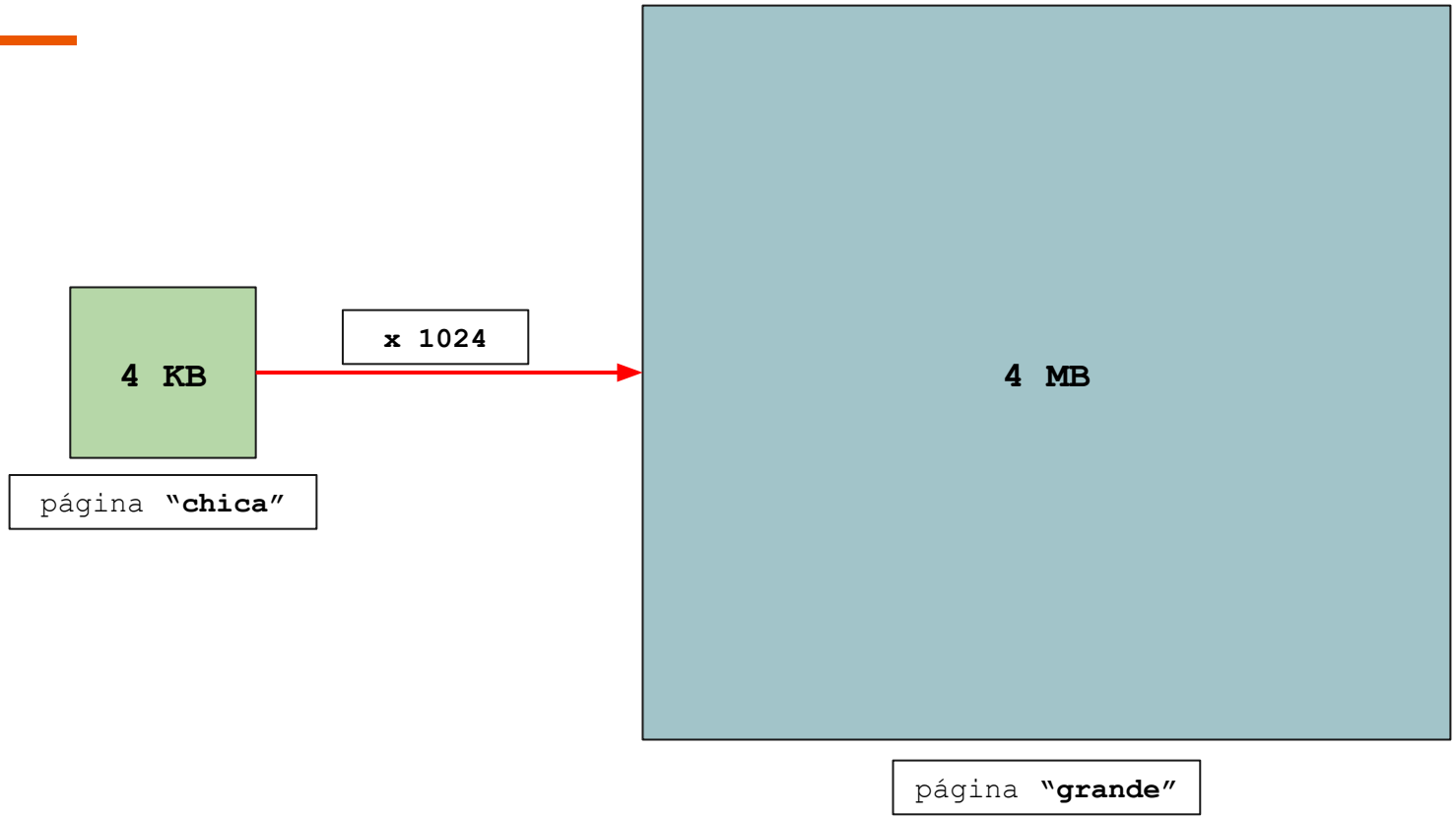
Modificando el *Page-Directory*

void **page_remove**(pde_t ***pgdir**, void ***va**)

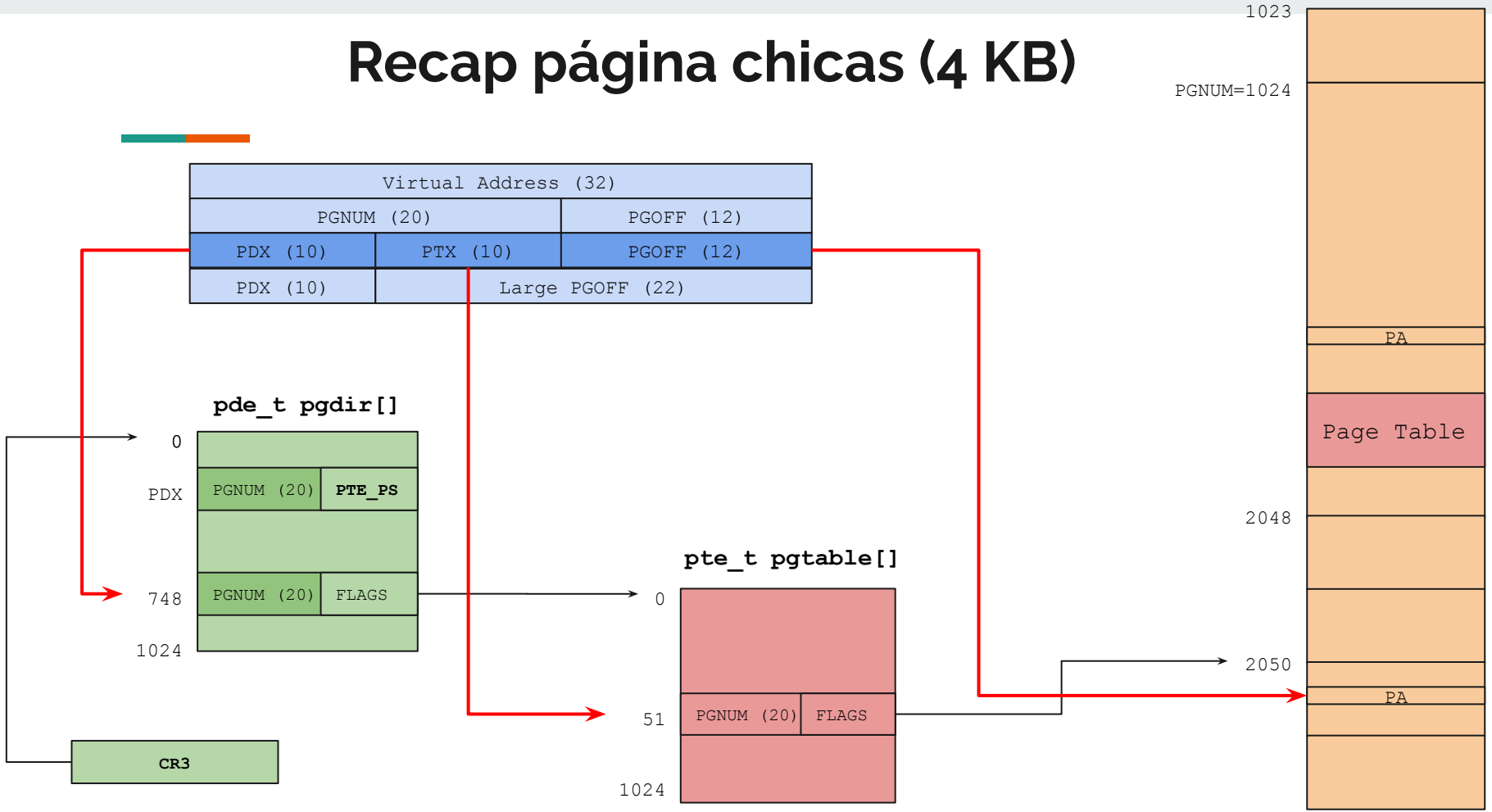
- Leer la documentación de la función en el código
- **page_lookup** puede ser útil
- Cómo hacer para ***setear*** a *cero* el contenido del mapeo anterior? Si es que existía dicho mapeo (pista: tercer argumento del ***page_lookup***)

Páginas “Grandes”

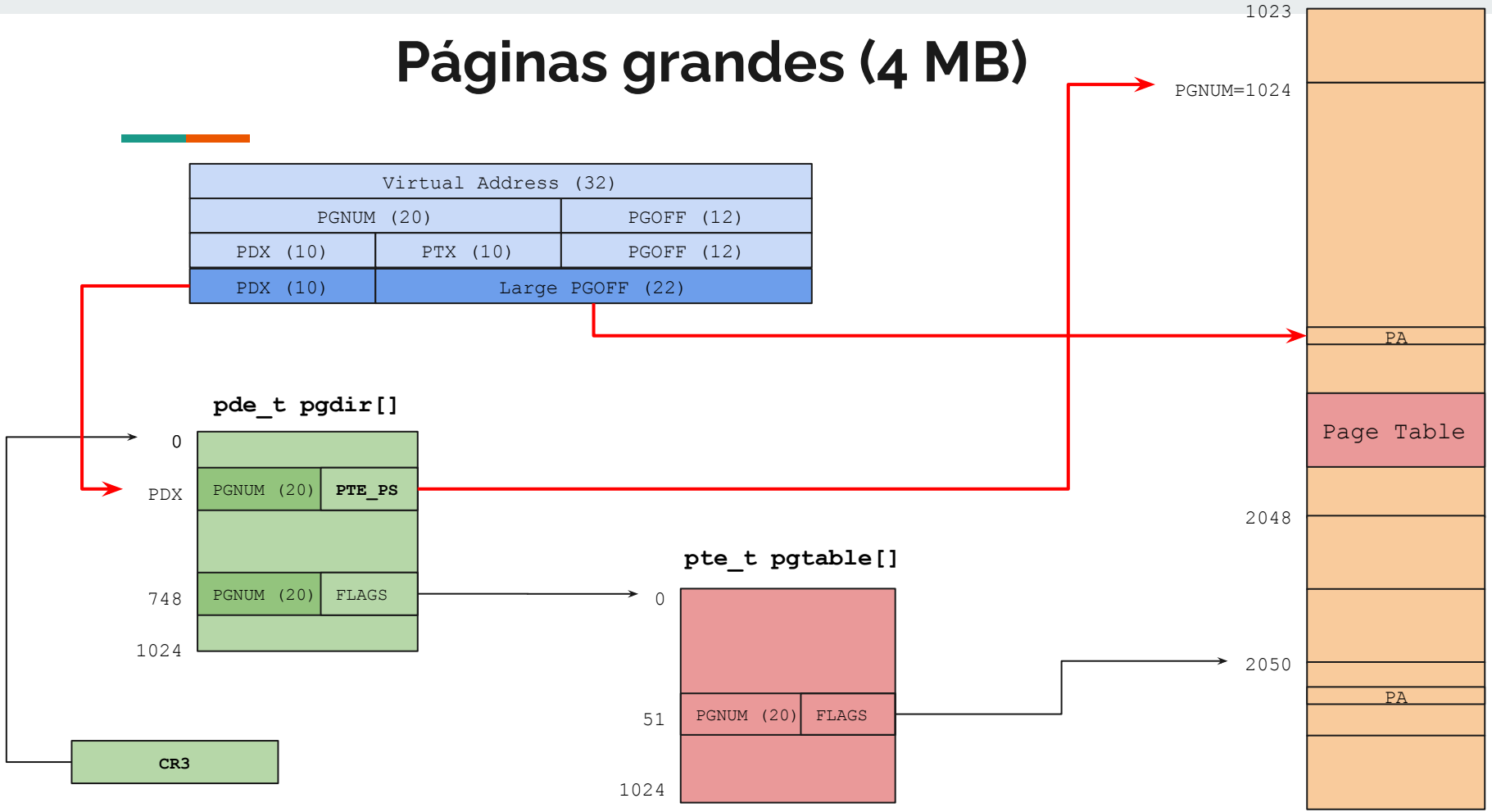
(a.k.a - large pages)


4 KB
x 1024
4 MB
página "chica"
página "grande"

Recap página chicas (4 KB)



Páginas grandes (4 MB)



Páginas grandes (4 MB)

- Análogamente a las páginas *normales*, las direcciones de las páginas *grandes* tienen sus primeros bits en cero para alinearse a dicho tamaño. Cuántos son?

- Activación mediante un registro de control: **cr4**

- Para ***boot_map_region*** tener en cuenta que: solamente se pueden mapear espacios de memoria que ocupen 4 MB o más