Paginación: MMU Básica

Francisco Giordano

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

22 de octubre de 2015

La clase pasada: Paginación kernel

1. Armaron un directorio de páginas con *identity mapping* para el rango `0x00000000` – `0x003FFFFF` (kernel y área libre).
2. Activaron paginación.
3. Escribieron las rutinas `mmu_mapear_pagina` y `mmu_unmapear_pagina`.

```
call mmu_inicializar_dir_kernel
mov cr3, eax

mov eax, cr0
or  eax, 0x80000000
mov cr0, eax
```

Hoy... Preparando paginación para tareas perro

- Hoy van a implementar una pequeña

Memory **M**anagement **U**nit.

- ¿Qué tiene que poder hacer?
  - Inicializar sus estructuras internas
  - Inicializar el mapa de memoria de una tarea
  - Mapear páginas físicas en páginas virtuales
  - Desmapear páginas

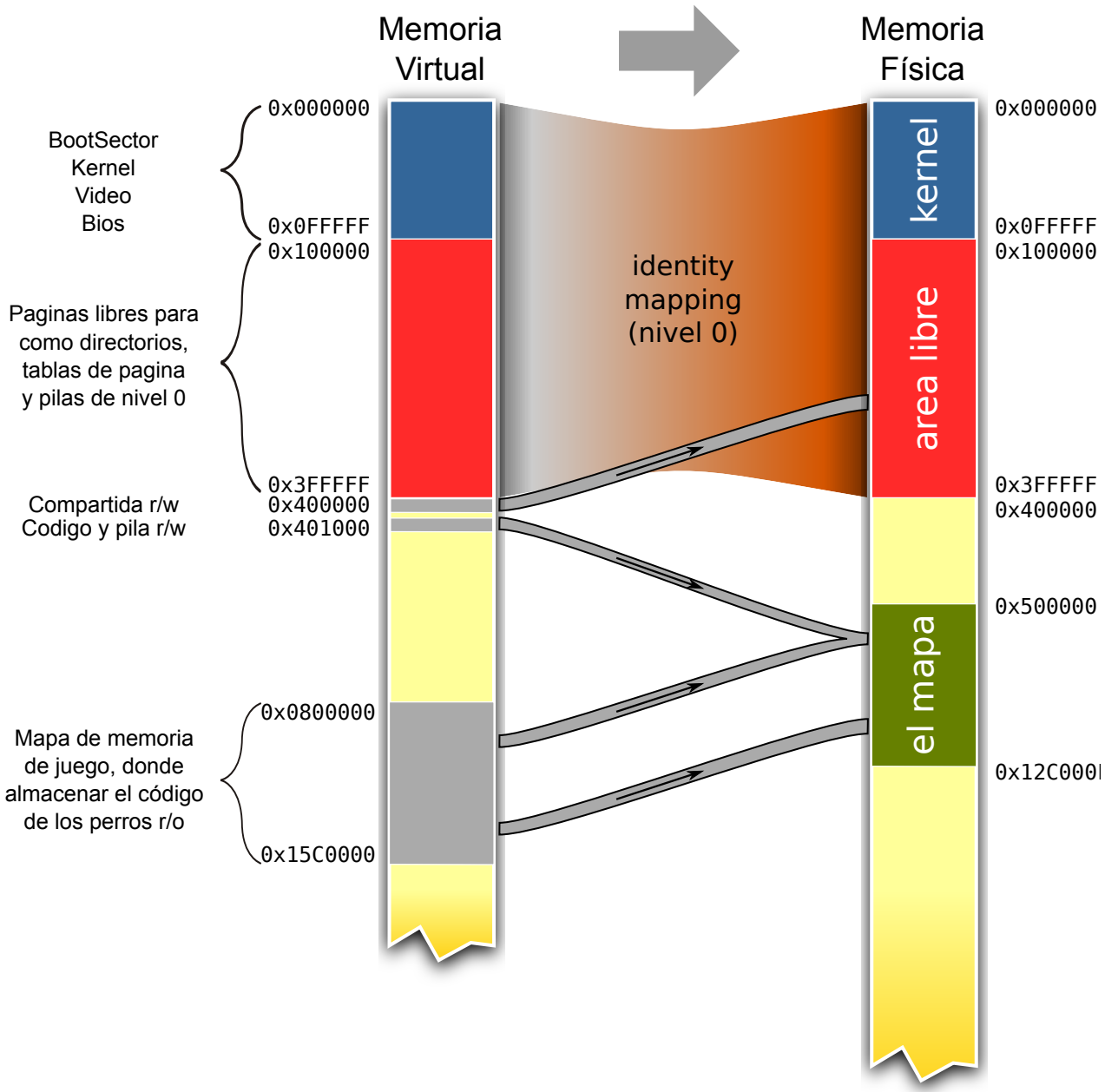
Hoy... Preparando paginación para tareas perro

- Hoy van a implementar una pequeña

Memory **M**anagement **U**nit.

- ¿Qué tiene que poder hacer?
  - Inicializar sus estructuras internas
  - Inicializar el mapa de memoria de una tarea
  - **Mapear páginas físicas en páginas virtuales**
  - **Desmapear páginas**

Mapa de memoria de una tarea



¿Cómo construimos el mapa de memoria de una tarea?

- Necesitamos un lugar libre donde colocar un *directorio* y *tablas* de página. Para esto es el área libre del kernel.

¿Cómo construimos el mapa de memoria de una tarea?

- Necesitamos un lugar libre donde colocar un *directorio* y *tablas* de página. Para esto es el área libre del kernel.

Una vez que tenemos un directorio necesitamos las tablas para mapear:

1. Identity mapping
2. Código de tarea
3. Página compartida

¿Cómo construimos el mapa de memoria de una tarea?

- Necesitamos un lugar libre donde colocar un *directorio* y *tablas* de página. Para esto es el área libre del kernel.

Una vez que tenemos un directorio necesitamos las tablas para mapear:

1. Identity mapping
2. Código de tarea
3. Página compartida

- Creamos una tabla y la vinculamos con el directorio.
- Hacemos *identity mapping* sobre las direcciones `0x00000000` a `0x003FFFFF`.

¿Cómo construimos el mapa de memoria de una tarea?

- Necesitamos un lugar libre donde colocar un *directorio* y *tablas* de página. Para esto es el área libre del kernel.

Una vez que tenemos un directorio necesitamos las tablas para mapear:

1. Identity mapping
2. Código de tarea
3. Página compartida

- Creamos una tabla y la vinculamos con el directorio.
- Hacemos *identity mapping* sobre las direcciones `0x00000000` a `0x003FFFFF`.

- Luego, necesitamos *copiar* y *mapear* la página de **código**.
  - Esto es en la dirección virtual **`0x400000`**.
- ¿A qué dirección física la mapeamos?

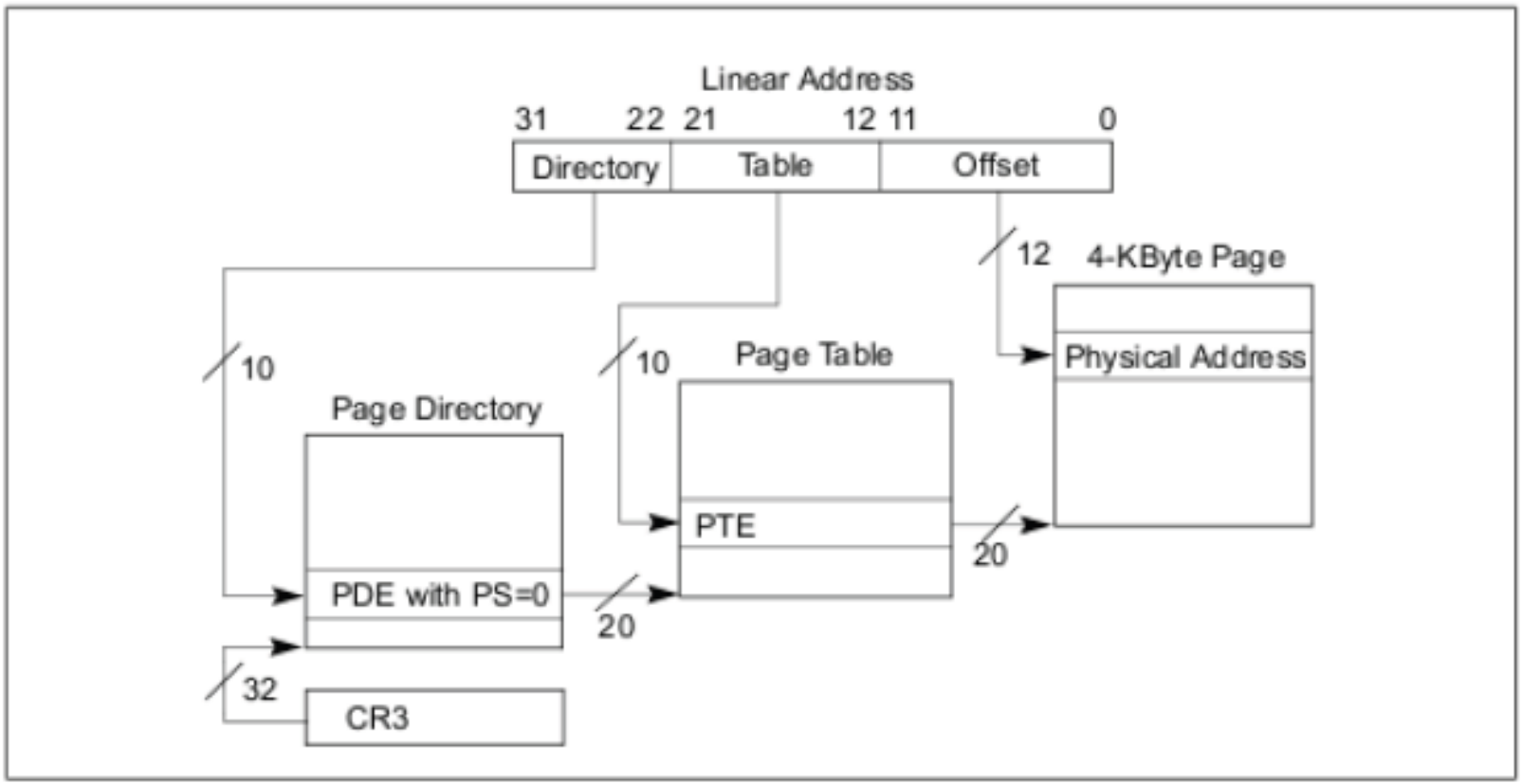
Mapear Páginas

Para llevar a cabo esto último, usaremos la función:

- ```
  void mmu_mapear_pagina(
          unsigned int virtual,
          unsigned int cr3,
          unsigned int fisica,
          unsigned int attrs)
  ```

Se encarga de *mapear* direcciones *virtuales* a direcciones *físicas* en un mapa de memoria dado (a través de un directorio de páginas).
  ```

Directorios y tablas...



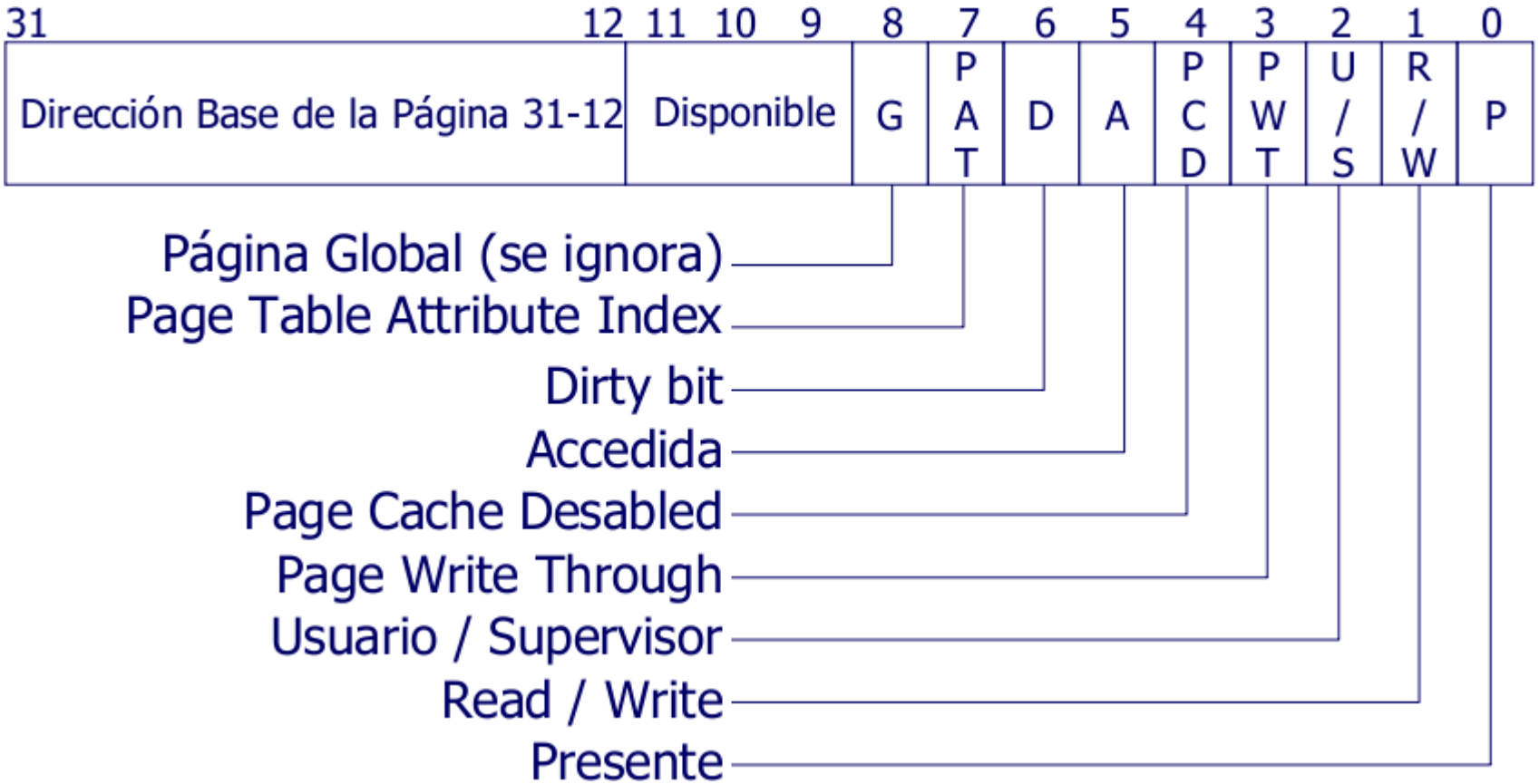
Mapear

1. Tomamos la dirección virtual y la descomponemos en sus partes:
   - *Índice en el directorio*,
   - *Índice en en la tabla* y
   - Desplazamiento dentro de la página (no la vamos a necesitar).
2. Obtenemos la *PDE* correspondiente.
3. Obtenemos la *PTE* correspondiente.
4. Completamos la *PTE* según corresponda.

PDE


31
12 11 10 9 8 7 6 5 4 3 2 1 0
Dirección Base de la Página 31-12
Disponible
G
PS
0
A
PCD
PWT
U/S
R/W
P
Página Global (se ignora)
Tamaño de Página (0=4Kbytes)
Reservado (poner en 0)
Accedida
Page Cache Desabled
Page Write Through
Usuario / Supervisor
Read / Write
Presente

PTE


31
12
11
10
9
8
7
6
5
4
3
2
1
0
Dirección Base de la Página 31-12
Disponible
G
PAT
D
A
PCD
PWT
U/S
R/W
P
Página Global (se ignora)
Page Table Attribute Index
Dirty bit
Accedida
Page Cache Desabled
Page Write Through
Usuario / Supervisor
Read / Write
Presente

Finalmente...

- Deben ejecutar la función `tlbflush()` para invalidar la cache de traducción de direcciones (en `mmu_mapear_pagina`).

Ejercicio 4

Implementar en `mmu.c` las funciones:

- `inicializar_mmu`
- `mmu_inicializar_memoria_perro`
  (no olvidarse de copiar el código de las tareas...)

Ejercicio 4

Además, deben probar que las funciones que implementaron hacen lo que se supone que hacen. Para eso, en `kernel.asm`:

- Llaman a `inicializar_mmu` y luego
- a `mmu_inicializar_memoria_perro`
- Cambian el *cr3* actual por el que retorna la función
- y cambian el color del fondo del primer caracter de la pantalla.

Algunas macros de C útiles...

```
#define   PDE_INDEX(virtual)   ???
#define   PTE_INDEX(virtual)   ???
#define   ALIGN(dir)           ???
#define   PG_PRESENT           ???
#define   PG_READ_WRITE        ???
#define   PG_USER              ???
```

¿Preguntas?