Paginación - MMU

Organización del Computador II

Natalia Pesaresi

[1]Departamento de Computación
Facultad de Ciencias Exactas y Naturales

29/05/2018

Qué hicimos hasta ahora?

- Armamos un esquema de paginación con identity mapping.
  - Rango: 0x00000000 a 0x003FFFFF (Primeros 4Mb)

Qué hicimos hasta ahora?

- Armamos un esquema de paginación con identity mapping.
  - Rango: 0x00000000 a 0x003FFFFF (Primeros 4Mb)
- Cómo podemos comprobarlo?
  - info tab

Qué vamos a hacer?

Implementar una limitada Memory Management Unit

1. Funciones para obtener una página de memoria física libre para utilizar.
2. Funciones para mappear y desmappear páginas.
3. Funciones para construir los equemas de paginación de las tareas.
   - Cada tarea tiene su page directory.

Obtención de páginas físicas libres

- Cómo podemos administrar qué **páginas físicas** tenemos **libres** de manera sencilla?

Obtención de páginas físicas libres

- Cómo podemos administrar qué **páginas físicas** tenemos **libres** de manera sencilla?
  - Vamos a tener un entero que guarde la dirección de una página física libre.

Obtención de páginas físicas libres

- Cómo podemos administrar qué **páginas físicas** tenemos **libres** de manera sencilla?
  - Vamos a tener un entero que guarde la dirección de una página física libre.
  - Cada vez que querramos una nueva página física libre, la obtenemos desde nuestro entero y luego incrementamos el mismo.

Obtención de páginas físicas libres

- Cómo podemos administrar qué **páginas físicas** tenemos **libres** de manera sencilla?
  - Vamos a tener un entero que guarde la dirección de una página física libre.
  - Cada vez que querramos una nueva página física libre, la obtenemos desde nuestro entero y luego incrementamos el mismo.
  - Necesitamos las funciones para pedir las páginas y un entero global.

Implementación

```
unsigned int proxima_pagina_libre;

void mmu_inicializar() {
    proxima_pagina_libre = INICIO_PAGINAS_LIBRES;
}

unsigned int mmu_proxima_pagina_fisica_libre() {
    unsigned int pagina_libre = proxima_pagina_libre;
    proxima_pagina_libre += PAGE_SIZE;
    return pagina_libre;
}
```

Mappeo de páginas

- Tenemos que hacer una función que **mappee** páginas de una dirección **virtual** a otra **fisica**.
  Necesitamos:

Mappeo de páginas

- Tenemos que hacer una función que **mappee** páginas de una dirección **virtual** a otra **fisica**.
  Necesitamos:
  - La **dirección virtual**.

Mappeo de páginas

- Tenemos que hacer una función que **mappee** páginas de una dirección **virtual** a otra **fisica**.
  Necesitamos:
  - La **dirección virtual**.
  - La **dirección** en donde se encuentra el **page directory** donde vamos a hacer el mappeo.

Mappeo de páginas

- Tenemos que hacer una función que **mappee** páginas de una dirección **virtual** a otra **fisica**.
  Necesitamos:
  - La **dirección virtual**.
  - La **dirección** en donde se encuentra el **page directory** donde vamos a hacer el mappeo.
  - La **dirección física**.

Mappeo de páginas

- Tenemos que hacer una función que **mappee** páginas de una dirección **virtual** a otra **fisica**.
  Necesitamos:
  - La **dirección virtual**.
  - La **dirección** en donde se encuentra el **page directory** donde vamos a hacer el mappeo.
  - La **dirección física**.
- Ejemplo:

```
void mmu_mappear_pagina(unsigned int virtual,
                        unsigned int dir_pd,
                        unsigned int fisica)
```

Procedimiento para mappear una página

1. Descomponemos la **dirección virtual** en **índice del page directory** y en **índice del page table**.

Procedimiento para mappear una página

1. Descomponemos la **dirección virtual** en **índice del page directory** y en **índice del page table**.
2. Utilizar la dirección del **page directory** y el **índice del page directory** para encontrar el **page directory entry** asociado al índice.

Procedimiento para mappear una página

1. Descomponemos la **dirección virtual** en **índice del page directory** y en **índice del page table**.
2. Utilizar la dirección del **page directory** y el **índice del page directory** para encontrar el **page directory entry** asociado al índice.
   - Hay que chequar que la pagle table a la que referencia el **pde** exista.
   - Si no existe, hay que crearla y settear correctamente (bits de propiedades) la **pde** para que pueda ser accedida posteriormente.

Procedimiento para mappear una página

1. Descomponemos la **dirección virtual** en **índice del page directory** y en **índice del page table**.
2. Utilizar la dirección del **page directory** y el **índice del page directory** para encontrar el **page directory entry** asociado al índice.
   - Hay que chequar que la pagle table a la que referencia el **pde** exista.
   - Si no existe, hay que crearla y settear correctamente (bits de propiedades) la **pde** para que pueda ser accedida posteriormente.
3. Utilizar el **índice del page table** para obtener la **page table entry** correspondiente.

Procedimiento para mappear una página

1. Descomponemos la **dirección virtual** en **índice del page directory** y en **índice del page table**.
2. Utilizar la dirección del **page directory** y el **índice del page directory** para encontrar el **page directory entry** asociado al índice.
   - Hay que chequar que la pagle table a la que referencia el **pde** exista.
   - Si no existe, hay que crearla y settear correctamente (bits de propiedades) la **pde** para que pueda ser accedida posteriormente.
3. Utilizar el **índice del page table** para obtener la **page table entry** correspondiente.
4. Completar la **pte** para que referencie a la **dirección física**.

Procedimiento para mappear una página

1. Descomponemos la **dirección virtual** en **índice del page directory** y en **índice del page table**.
2. Utilizar la dirección del **page directory** y el **índice del page directory** para encontrar el **page directory entry** asociado al índice.
   - Hay que chequar que la pagle table a la que referencia el **pde** exista.
   - Si no existe, hay que crearla y settear correctamente (bits de propiedades) la **pde** para que pueda ser accedida posteriormente.
3. Utilizar el **índice del page table** para obtener la **page table entry** correspondiente.
4. Completar la **pte** para que referencie a la **dirección física**.
5. Ejecutar **tlbflush()** para invalidar la cache de traducciones.

Desmappeo de páginas

- Tenemos que hacer una función que **desmappee** una dirección **virtual**.
  Necesitamos:

Desmappeo de páginas

- Tenemos que hacer una función que **desmappee** una dirección **virtual**.
  Necesitamos:
  - La **dirección virtual**.

Desmappeo de páginas

- Tenemos que hacer una función que **desmappee** una dirección **virtual**.
  Necesitamos:
  - La **dirección virtual**.
  - La **dirección** en donde se encuentra el **page directory** donde vamos a hacer el desmappeo.

Desmappeo de páginas

- Tenemos que hacer una función que **desmappee** una dirección **virtual**.
  Necesitamos:
  - La **dirección virtual**.
  - La **dirección** en donde se encuentra el **page directory** donde vamos a hacer el desmappeo.

Desmappeo de páginas

- Tenemos que hacer una función que **desmappee** una dirección **virtual**.
  Necesitamos:
  - La **dirección virtual**.
  - La **dirección** en donde se encuentra el **page directory** donde vamos a hacer el desmappeo.
- Ejemplo:

```
void mmu_desmappear_pagina(unsigned int virtual,
                           unsigned int dir_pd)
```

Procedimiento para desmappear una página

1. Descomponemos la **dirección virtual** en **índice del page directory** y en **índice del page table**.

Procedimiento para desmappear una página

1. Descomponemos la **dirección virtual** en **índice del page directory** y en **índice del page table**.
2. Utilizar la dirección del **page directory** y el **índice del page directory** para encontrar el **page directory entry** asociado al índice.

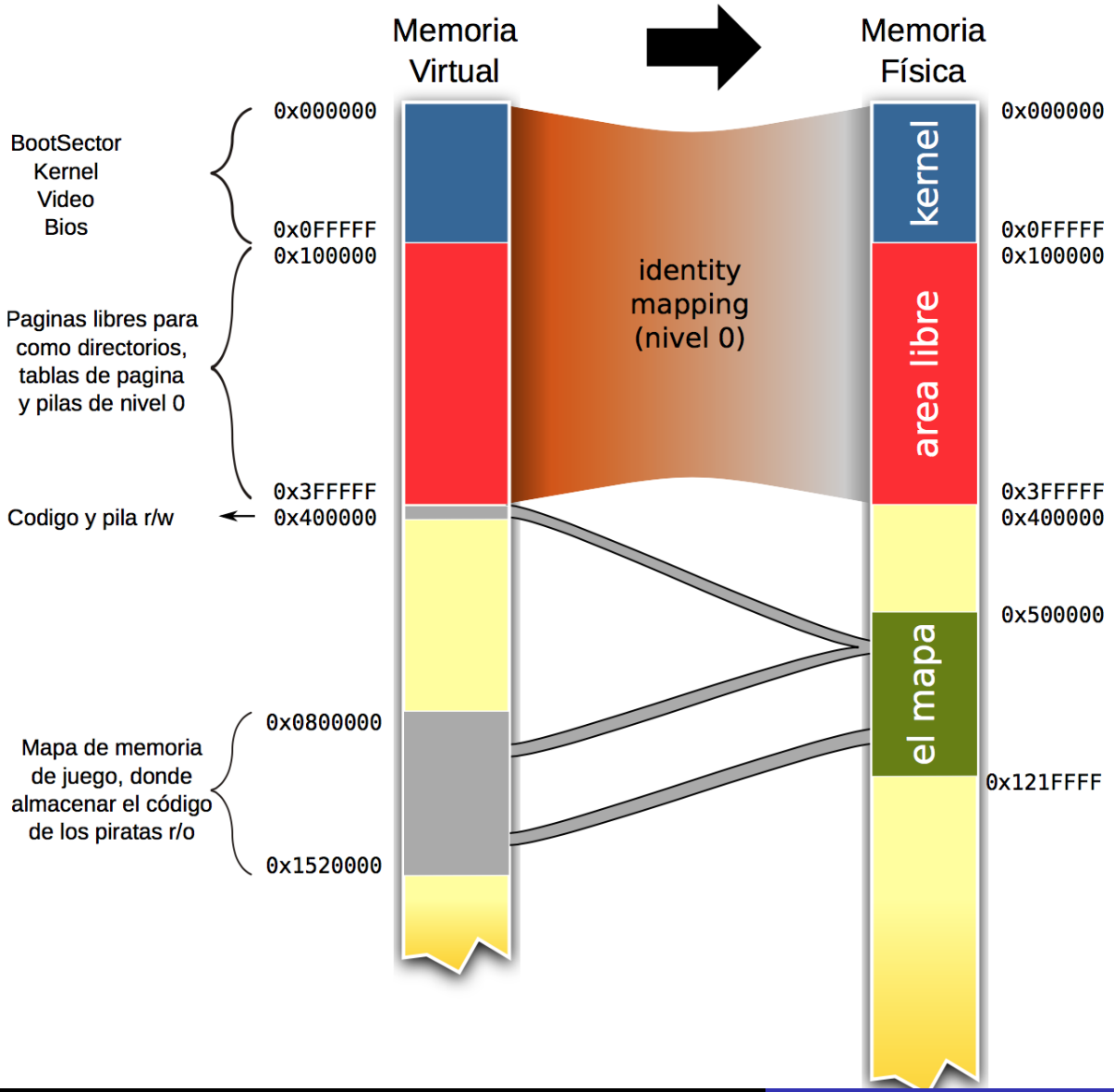
Procedimiento para desmappear una página

1. Descomponemos la **dirección virtual** en **índice del page directory** y en **índice del page table**.
2. Utilizar la dirección del **page directory** y el **índice del page directory** para encontrar el **page directory entry** asociado al índice.
3. Utilizar el **índice del page table** para obtener la **page table entry** correspondiente.

Procedimiento para desmappear una página

1. Descomponemos la **dirección virtual** en **índice del page directory** y en **índice del page table**.
2. Utilizar la dirección del **page directory** y el **índice del page directory** para encontrar el **page directory entry** asociado al índice.
3. Utilizar el **índice del page table** para obtener la **page table entry** correspondiente.
4. Establecer el bit de presente en 0.

Esquema de paginación de las tareas



Construcción de esquema de paginación de las tareas

1. Necesitamos pedir una **página física libre** para el directorio de páginas.

Construcción de esquema de paginación de las tareas

1. Necesitamos pedir una **página física libre** para el directorio de páginas.
2. Tenemos que armar un esquema de **indentity mapping** de los primeros 4Mb.
   - Los **pde** y **pte** deben estar setteados como supervisor.

Construcción de esquema de paginación de las tareas

1. Necesitamos pedir una **página física libre** para el directorio de páginas.
2. Tenemos que armar un esquema de **indentity mapping** de los primeros 4Mb.
   - Los **pde** y **pte** deben estar setteados como supervisor.
3. Copiar el código de la tarea a su posición en el mapa.

Construcción de esquema de paginación de las tareas

1. Necesitamos pedir una **página física libre** para el directorio de páginas.
2. Tenemos que armar un esquema de **indentity mapping** de los primeros 4Mb.
   - Los **pde** y **pte** deben estar setteados como supervisor.
3. Copiar el código de la tarea a su posición en el mapa.
4. Mappear la dirección virtual del código de la tarea a su posición física en el mapa donde fue copiada.

Construcción de esquema de paginación de las tareas

1. Necesitamos pedir una **página física libre** para el directorio de páginas.
2. Tenemos que armar un esquema de **indentity mapping** de los primeros 4Mb.
   - Los **pde** y **pte** deben estar setteados como supervisor.
3. Copiar el código de la tarea a su posición en el mapa.
4. Mappear la dirección virtual del código de la tarea a su posición física en el mapa donde fue copiada.
5. Mappear las demás páginas que pertenezcan a la tarea según el enunciado.
   - Los **pde** y **pte** (código y páginas de la tarea) deben estar setteados como user.

Alguna recomendaciones útiles

- Pueden modificar las funciones anteriores parametrizando distintos atributos a settear.
- Pueden comprobar que el mappeo es correcto setteando el **CR3** actual con el **page directory** que crearon.
- Utilicen **info tab** para ver si el mappeo es correcto.
- Macros útiles:

```
#define PDE_INDEX(virtual) virtual >> 22
#define PTE_INDEX(virtual) ???
#define PG READ WRITE      ???
#define PG USER            ???
#define PG PRESENT         0x00000001
```