

# Espacio direcciones de un proceso

Yolanda Becerra Fontal  
Juan José Costa Prats

Facultat d'Informàtica de Barcelona (FIB)  
Universitat Politècnica de Catalunya (UPC)  
BarcelonaTech  
2017-2018 QP

# Indice

- Generación de ejecutables y carga
- Espacios de direcciones
- Espacio lógico de un proceso
- Soporte HW: MMU
- Gestión de memoria en ZeOS

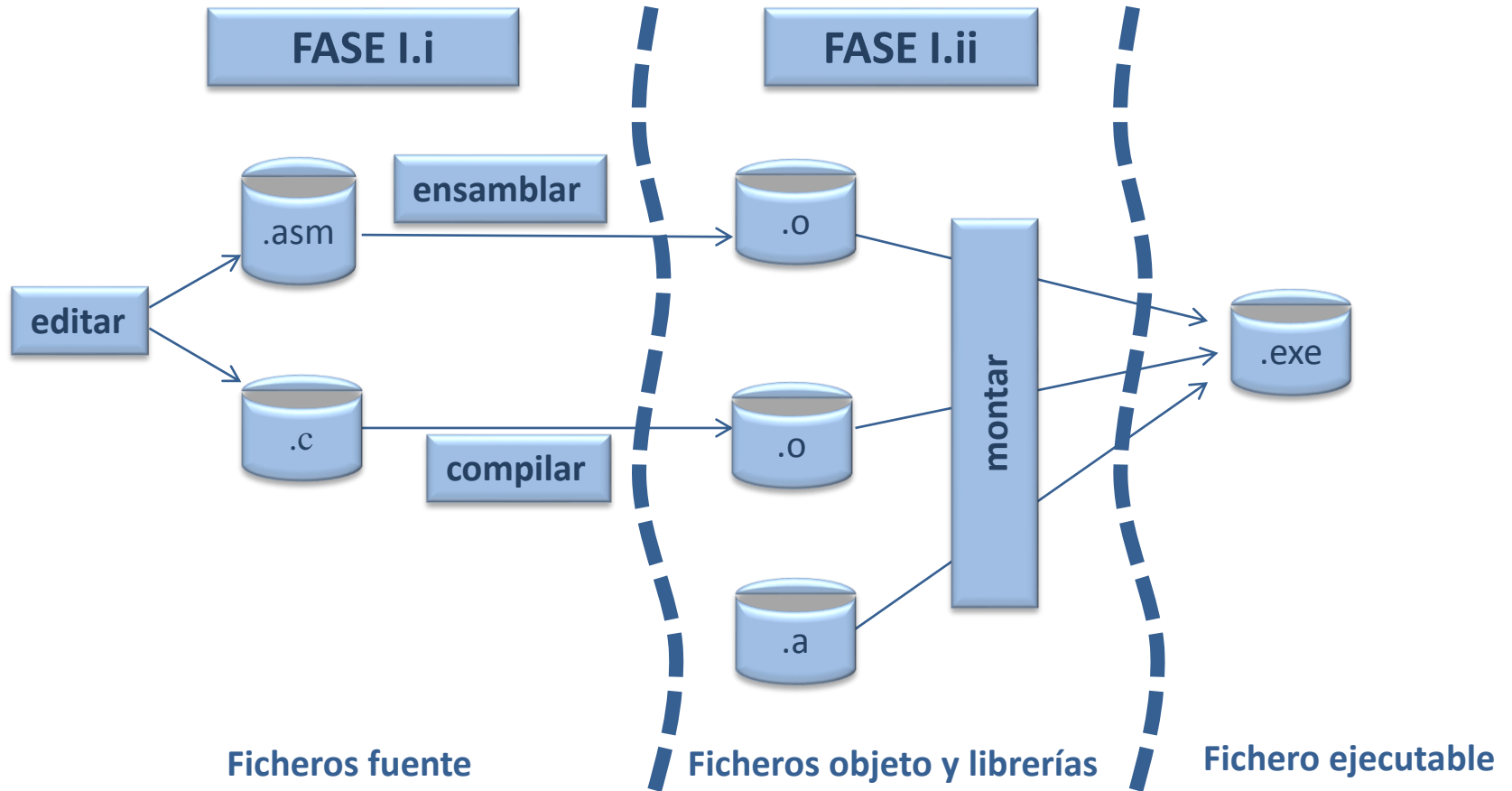
# Indice

- **Generación de ejecutables y carga**
- Espacios de direcciones
- Espacio lógico de un proceso
- Soporte HW: MMU
- Gestión de memoria en ZeOS

# Generación de ejecutables y carga

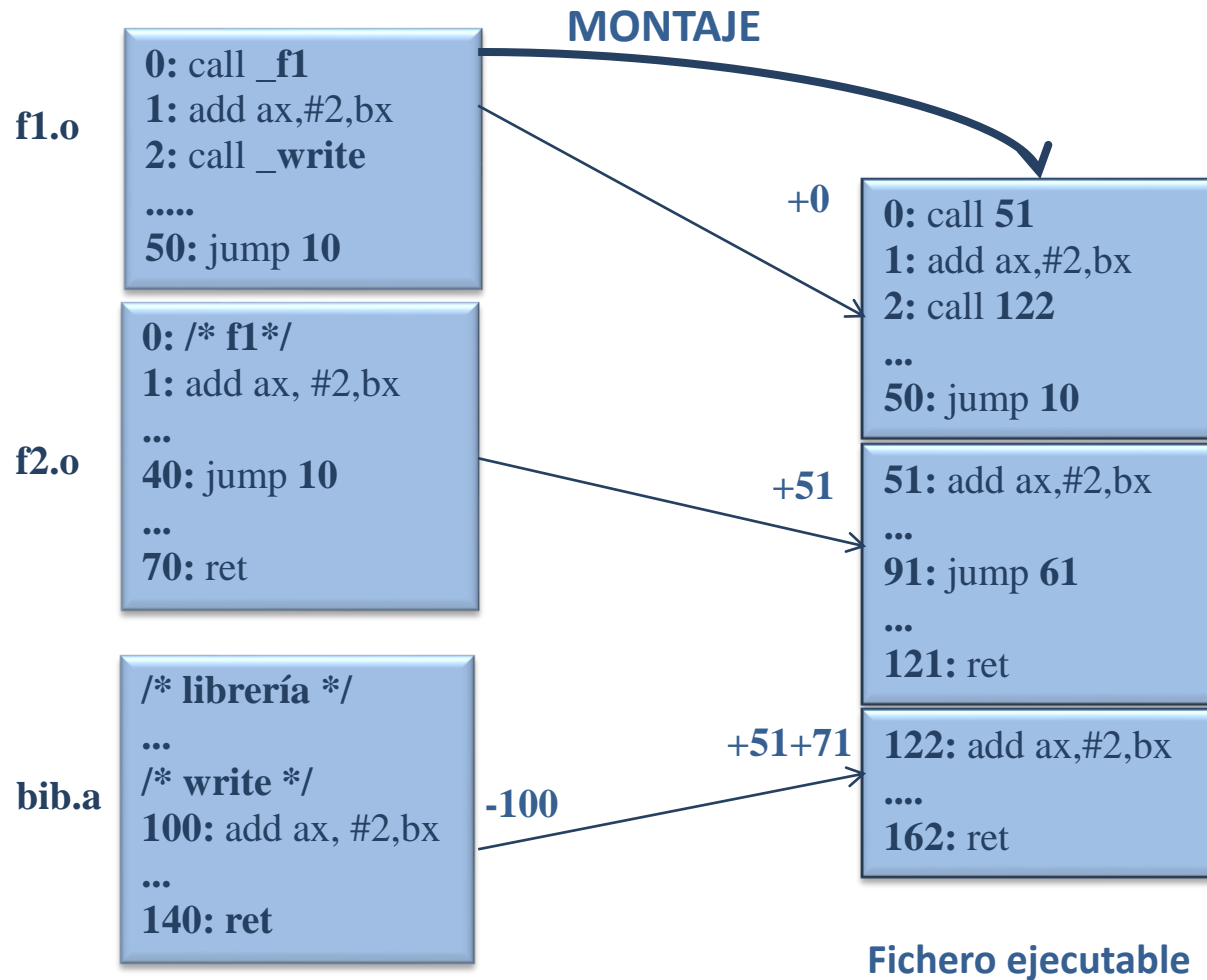
- Evolución de los programas
  - Lenguaje alto nivel → Lenguaje máquina → Ejecución
- Fase I
  - i. Compilación: Traducción de lenguaje alto nivel a código objeto
  - ii. Montaje: Creación de un fichero ejecutable a partir de 1 o varios ficheros objeto y librerías
- Fase II
  - i. Carga/Ejecución: Carga un fichero ejecutable en memoria física y da control a la primera instrucción del programa

# Etapas



Generación de ejecutables y carga

# Fase de montaje

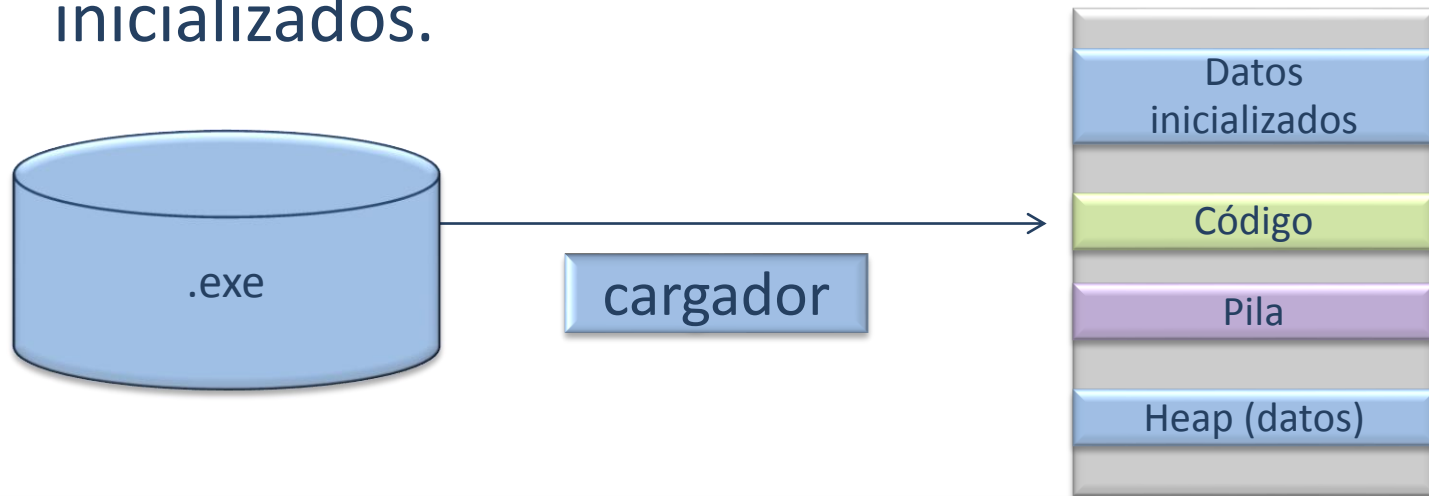


# Compilar - Montar

- El fichero ejecutable contiene además una cabecera dónde indica qué es código, datos inicializados, memoria necesaria para pila y datos no inicializados

# Cargador

- El programa cargador lee un ejecutable y:
  - Carga en memoria todo su código y sus datos
    - Aunque puede cargar sólo una parte
  - Reserva espacio en memoria para pila y datos no inicializados.





# Indice

- Generación de ejecutables y carga
- **Espacios de direcciones**
- Espacio lógico de un proceso
- Soporte HW: MMU
- Gestión de memoria en ZeOS

# Espacios de direcciones

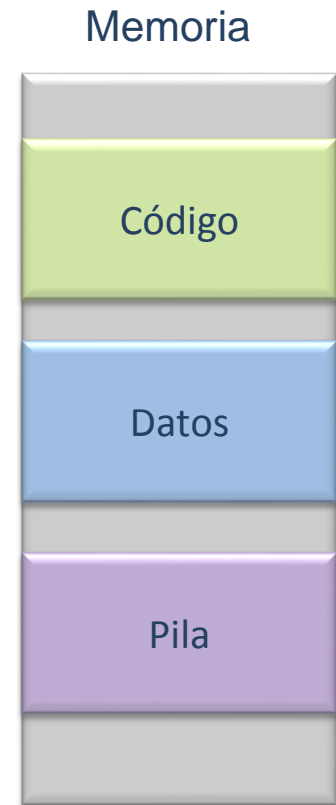
- Espacio de direcciones lógico del procesador
  - Rango de direcciones que puede acceder un procesador
  - Depende del bus de @
- Espacio de direcciones lógico del proceso
  - Espacio que ocupa un proceso en ejecución
  - Las direcciones que lanza un procesador cuando quiere acceder a datos/código/pila del proceso
  - RELATIVAS
- Espacio de direcciones físico del proceso
  - Direcciones de memoria física asociadas a las direcciones lógicas

# Indice

- Generación de ejecutables y carga
- Espacios de direcciones
- **Espacio lógico de un proceso**
- Soporte HW: MMU
- Gestión de memoria en ZeOS

# Espacio lógico de un proceso

- Regiones
  - Código
  - Datos (inicializados o no)
  - Heap: memoria dinámica
  - Pila



# Espacio lógico de un proceso

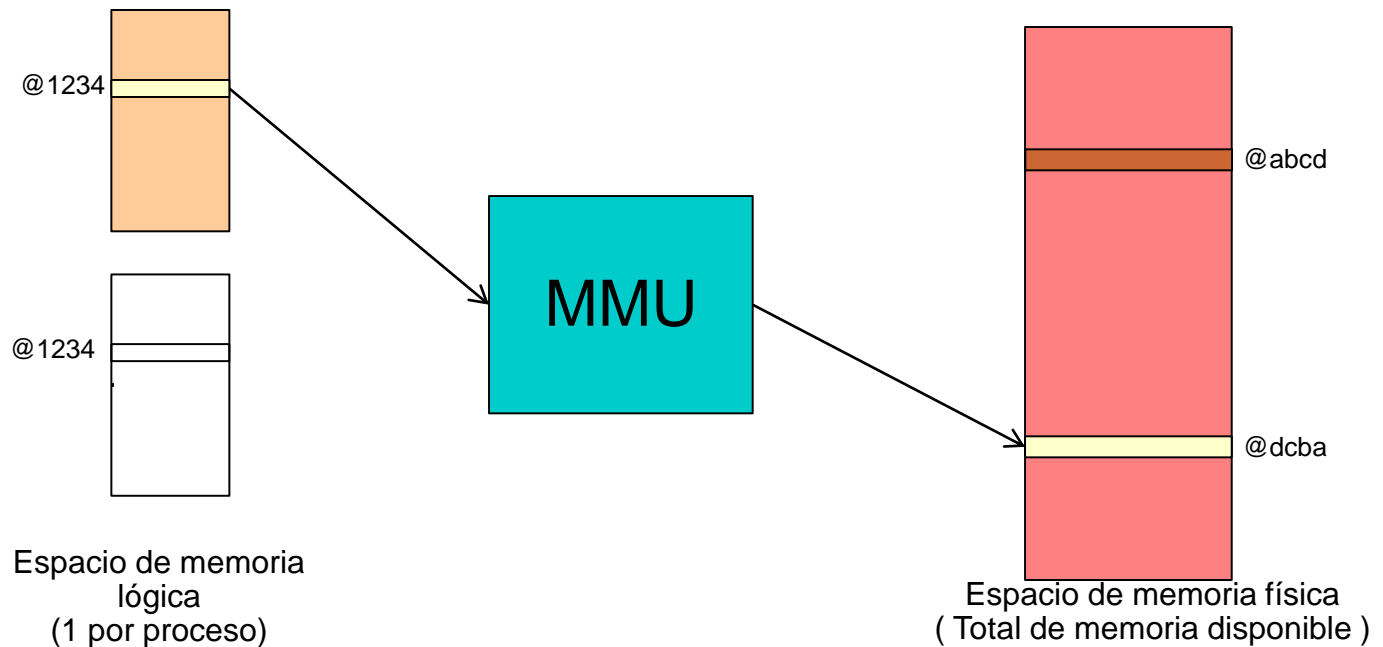
- Las direcciones que genera un procesador cuando está ejecutando un proceso son lógicas
  - Relativas a una dirección 0, igual para todos los procesos
  - Pero los datos se guardan en posiciones físicas de memoria
- Hace falta **traducir** de direcciones lógicas a físicas
  - MMU: *memory management unit*. El hardware necesario para producir esta traducción
- Puede haber más cosas
  - *Swap* o memoria virtual

# Indice

- Generación de ejecutables y carga
- Espacios de direcciones
- Espacio lógico de un proceso
- **Soporte HW: MMU**
- Gestión de memoria en ZeOS

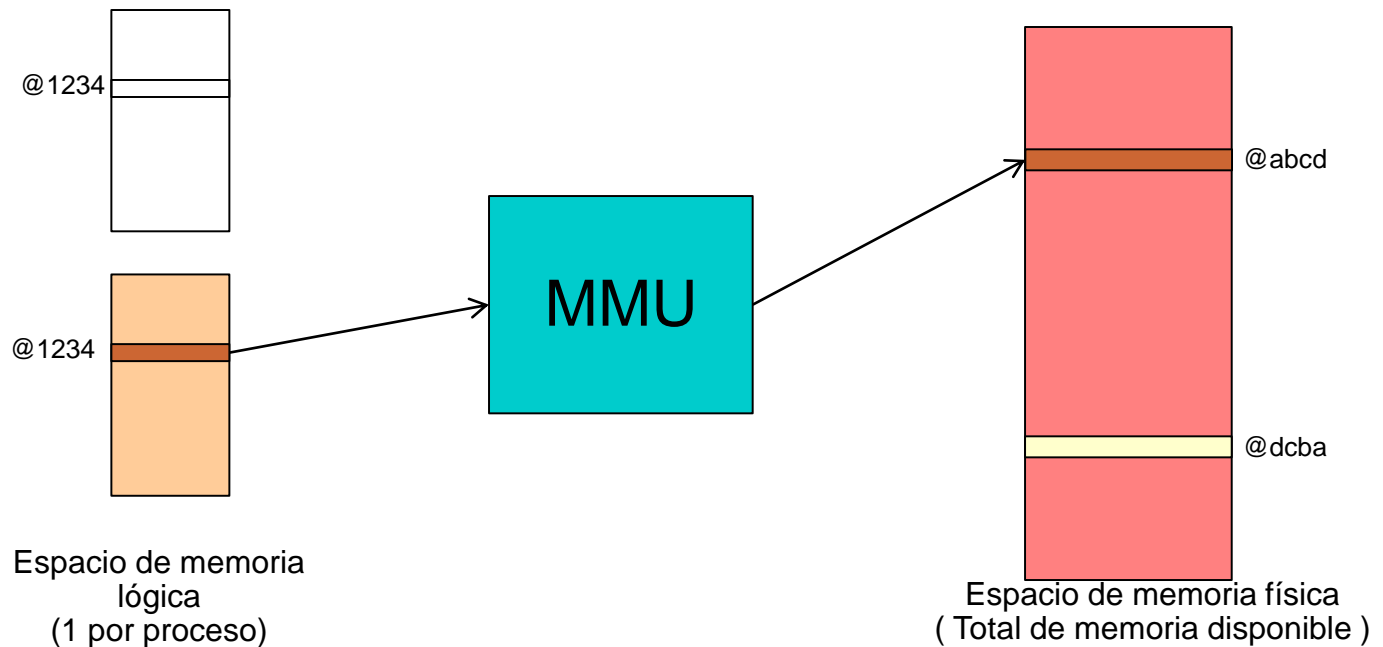
# Soporte HW: MMU

- MMU: *memory management unit*
  - Unidad encargada de traducir las @lógicas a @físicas



# Soporte HW: MMU

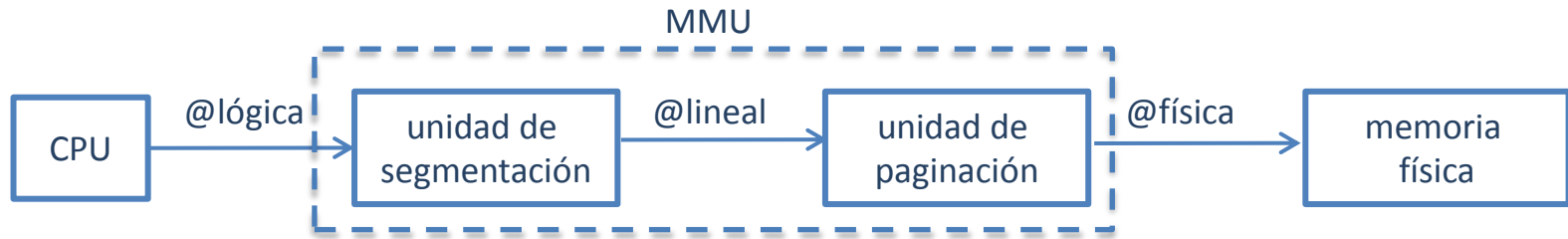
- MMU: *memory management unit*
  - Unidad encargada de traducir las @lógicas a @físicas





# Segmentación paginada

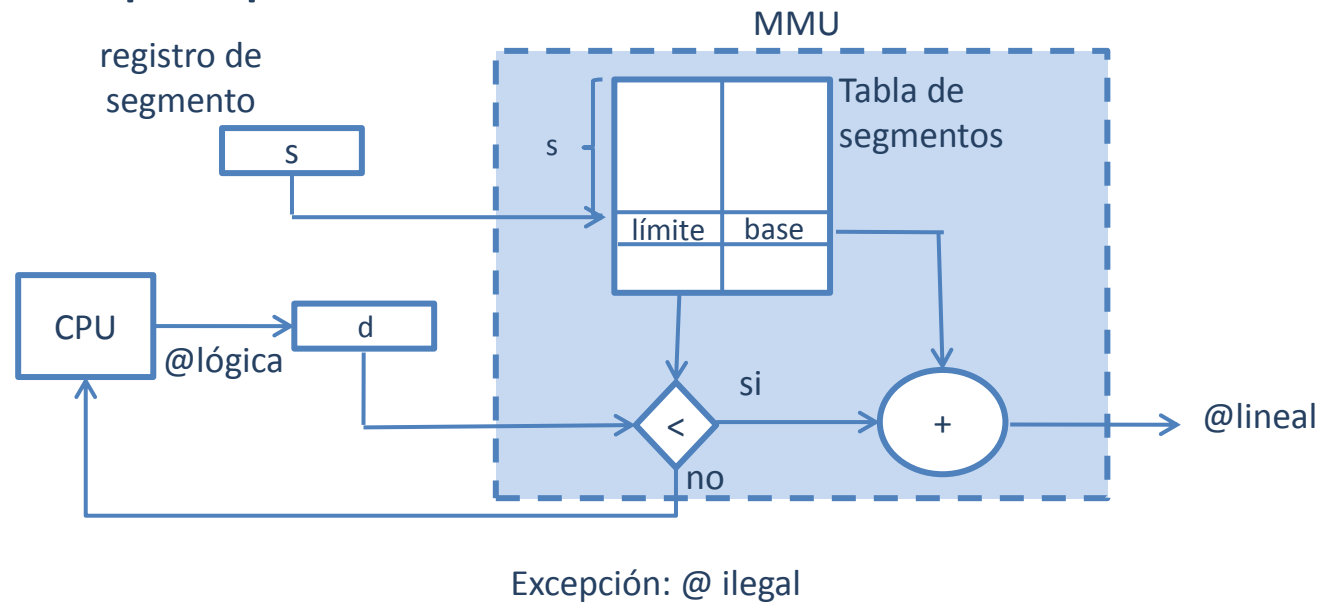
- Intel pentium: MMU usa segmentación paginada



- Espacio lógico del proceso dividido en segmentos
- Segmentos divididos en páginas
  - Tamaño de segmento múltiplo del tamaño de página
  - Unidad de trabajo del SO es la página

# Segmentación

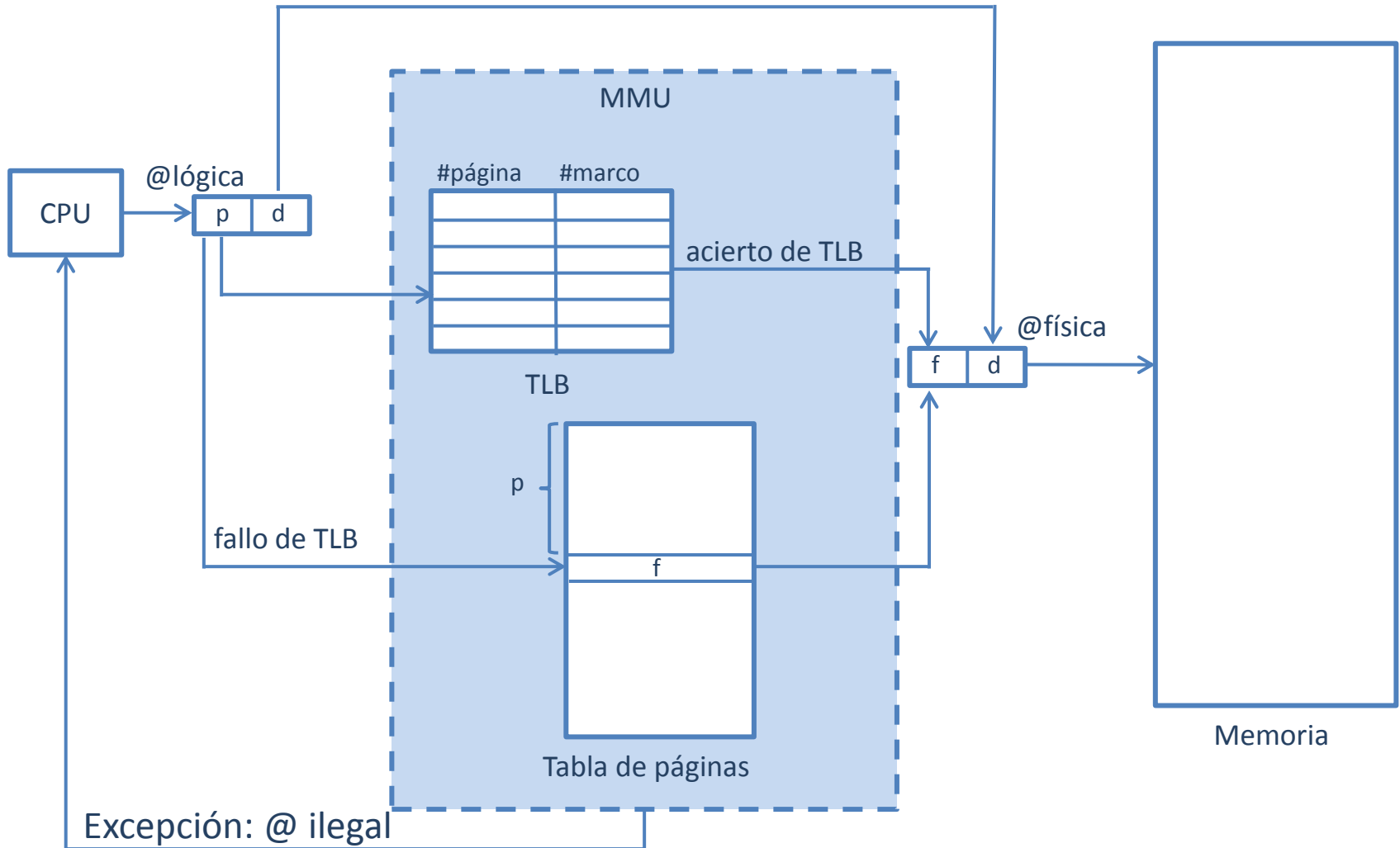
- Tabla de segmentos
  - Para cada segmento: @ base y tamaño
  - Una tabla por proceso



# Paginación

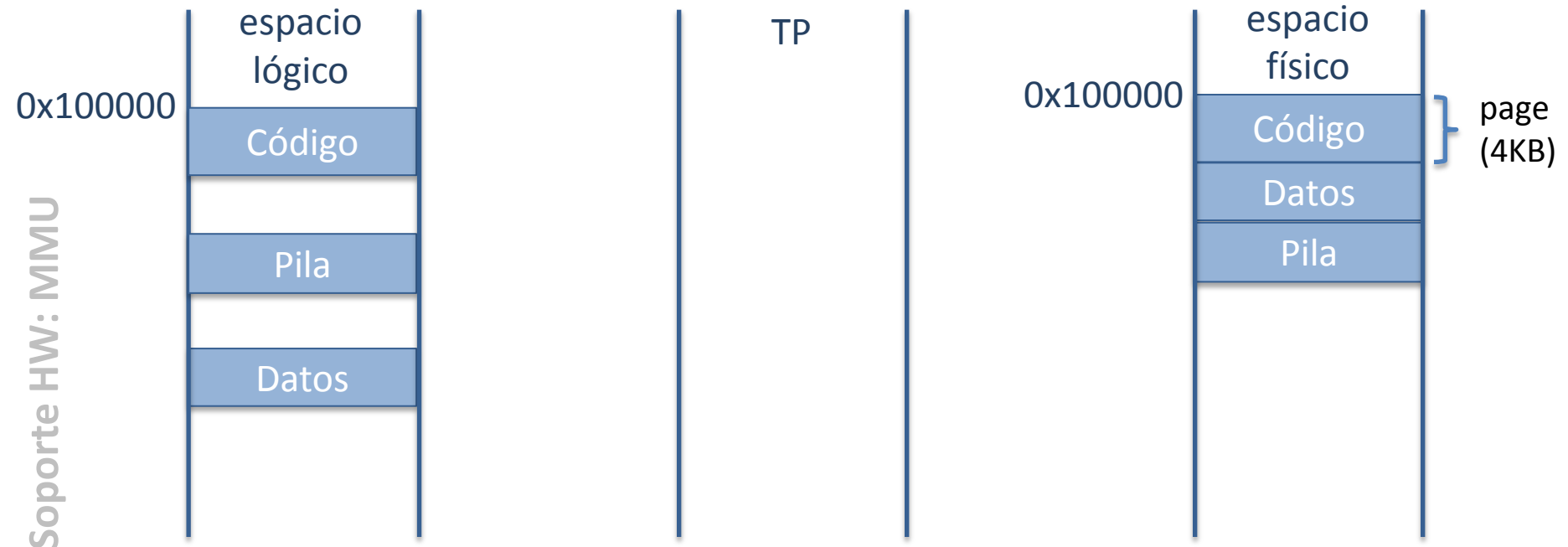
- **Tabla de páginas**
  - Para mantener información a nivel de página: validez, permisos de acceso, marco asociado, etc....
  - Una entrada para cada página
  - Una tabla por proceso
- Suele guardarse en memoria y SO debe conocer la @ base de la tabla de cada proceso (por ejemplo, guardándola en el PCB)
- Procesadores actuales también disponen de TLB (*Translation Lookaside Buffer*)
  - Memoria asociativa (cache) de acceso **más rápido** en la que se almacena la información de traducción para las páginas *activas*
  - Hay que actualizar/invalidar la TLB cuando hay un cambio en la MMU
    - Gestión HW del TLB/Gestión Software (SO) del TLB
    - Muy dependiente de la arquitectura

# Paginación



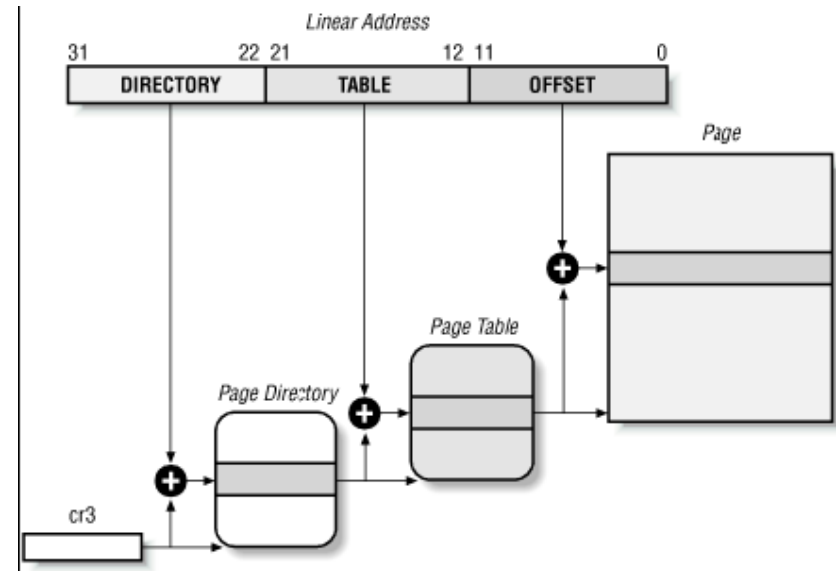
# Ejemplo de traducción

Dado un sistema de memoria donde el espacio lógico de un proceso y su correspondiente espacio físico aparece tal y como se indica en la siguiente figura, representa el contenido de la tabla de páginas.



# Paginación

- Espacio necesario para las tablas de páginas?
- Tabla de páginas multinivel
  - Ahorrar memoria necesaria para las tablas
  - Sólo traducción para páginas en uso
- Intel Pentium: 2 niveles



# Indice

- Generación de ejecutables y carga
- Espacios de direcciones
- Espacio lógico de un proceso
- Soporte HW: MMU
- **Gestión de memoria en ZeOS**

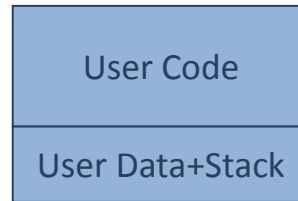
# Gestión de memoria en ZeOS

- No hay cargador
  - Ejecutable de usuario se carga en tiempo de boot
- En el código base todos los procesos tienen:
  - Mismo espacio lógico de direcciones
  - Misma cantidad de memoria física
- Todos los procesos comparten memoria física del código
- No se explota la segmentación
  - Sólo se utiliza para implementar protección
  - Todos los segmentos: misma @base y mismo tamaño



# ZeOS: espacio lógico de direcciones

## – User mode



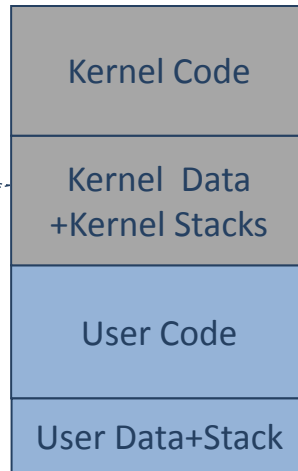
L\_USER\_START

$DATA\_START=L\_USER\_START+(NUM\_PAG\_CODE*PAGE\_SIZE)$

$DATA\_END=DATA\_START+(NUM\_PAG\_DATA*PAGE\_SIZE)$

## – Kernel mode

Incluye task table



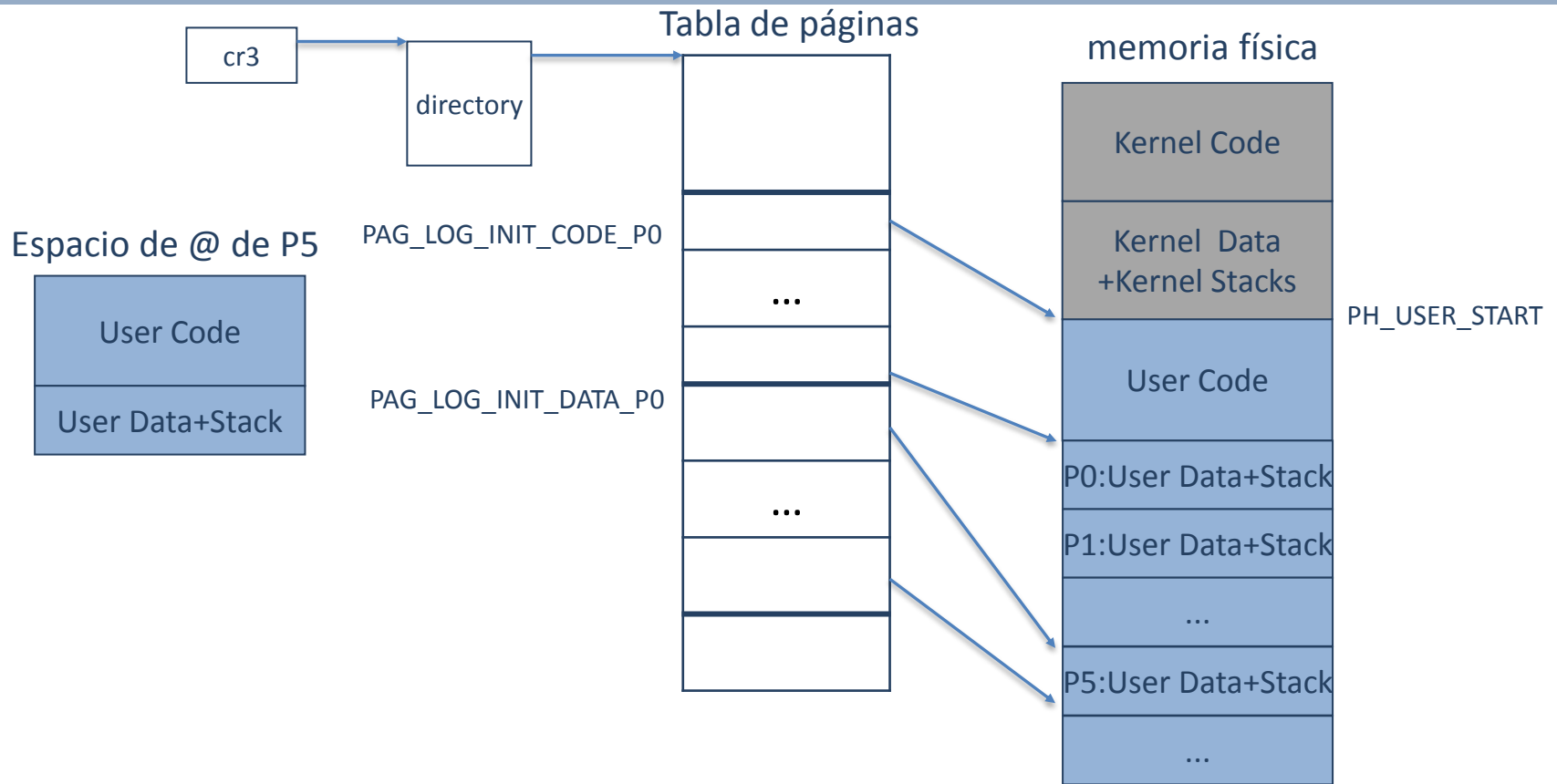
KERNEL\_START

L\_USER\_START

$DATA\_START=L\_USER\_START+(NUM\_PAG\_CODE*PAGE\_SIZE)$

$DATA\_END=DATA\_START+(NUM\_PAG\_DATA*PAGE\_SIZE)$

# ZeOS: memoria física



# Atributos por página: entrada TP

```
typedef union
{
    unsigned int entry;
    struct {
        unsigned int present    : 1; presente
        unsigned int rw        : 1; permisos
        unsigned int user      : 1; user/supervisor
        unsigned int write_t   : 1; write through/write_back (linux write back)
        unsigned int cache_d   : 1; caching is enable (linux sets this)
        unsigned int accessed  : 1; (reset by OS)
        unsigned int dirty     : 1; (set by OS)
        unsigned int ps_pat    : 1; (page_size: normal or big)
        unsigned int global    : 1; (if set, tlb is not flushed after modifying CR3)
        unsigned int avail     : 3; (not in use)
        unsigned int pbase_addr : 20;
    } bits;
} page_table_entry;
```

# References

- [1] Understanding Linux Kernel 3rd ed. Chapter 2 Memory addressing.
- [2] Intel<sup>®</sup> 64 and IA-32 architectures software developer's manual volume 3A: System programming guide. Chapter 3 and 4.