Arquitectura de Computadoras para Ingeniería

(Cód. 7526)
1° Cuatrimestre 2016

Dra. Dana K. Urribarri
DCIC - UNS

Memoria Virtual

Memoria Virtual

- Arquitectura 32 bits:
  - Registros de 32 bits
  - Referencia de memoria de 32 bits
  - Espacio direccionable de 2^{32} bytes = 4GB
- Arquitectura 64 bits:
  - Registros de 64 bits
  - Referencias de memoria de 64 bits
  - Espacio direccionable de 2^{64} bytes = 2^{34} GB

Memoria Virtual

- Gran desproporción entre el espacio direccionable y la capacidad de memoria principal.
- Multiprogramación:
  - Varios programas residentes en memoria a la vez.
  - Cuando un programa realiza una operación de I/O, otro programa toma el control de la CPU.

Administración de memoria

- *Overlay*:
  - Programa muy grande para la memoria física
  - El programador dividía al programa en porciones mutuamente excluyentes (*overlays*)
  - El programa controlaba en ejecución qué *overlays* se cargaban o sacaban de la memoria física.
  - Responsabilidad del programador que el programa no acceda a memoria no cargada en memoria física.

Administración de memoria

- Memoria Virtual (MV)
  - Introducido en 1960
  - Administra automáticamente los niveles de la jerarquía representados por memoria principal y secundaria.
  - Permite compartir pequeñas porciones de memoria física entre varios procesos.
  - Permite que algunos programas corran en cualquier posición de memoria física (*relocation*)

Location ≠ locación

Administración de memoria

- Los programas se compilan y enlazan (*linking*) como si pudieran acceder a todo el espacio direccionable.
- Dirección virtual (lógica): Dirección generada por la CPU.
- Dirección física: dirección virtual traducida a una dirección real de memoria física.
- *Memory Management Unit* (MMU): dispositivo que mapea direcciones virtuales a físicas.

Caché vs. Memoria Virtual

Parameter	First-level cache	Virtual memory
Block (page) size	16–128 bytes	4096–65,536 bytes
Hit time	1–3 clock cycles	100–200 clock cycles
Miss penalty	8–200 clock cycles	1,000,000–10,000,000 clock cycles
(access time)	(6–160 clock cycles)	(800,000–8,000,000 clock cycles)
(transfer time)	(2–40 clock cycles)	(200,000–2,000,000 clock cycles)
Miss rate	0.1–10%	0.00001–0.001%
Address mapping	25–45-bit physical address to 14–20-bit cache address	32–64-bit virtual address to 25–45-bit physical address

Caché vs. Memoria Virtual

- Reemplazos
  - Los reemplazos en caché los controla el hardware.
  - Los reemplazos en MV los controla el sistema operativo (SO).

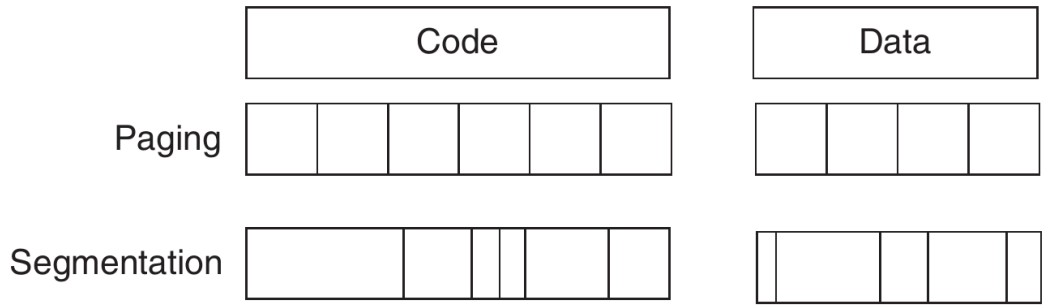
    La penalización por un faltante es mayor, por lo tanto el SO puede tomarse más tiempo para decidir qué reemplazar.
- Tamaño de la dirección
  - El direccionamiento del procesador determina el tamaño de la memoria virtual.
  - El tamaño de la caché es independiente.

Organización de la MV

Se puede dividir en dos categorías de organización en función de cómo se divide la memoria:

- Paginación
  - Bloques de tamaño fijo (páginas)
- Segmentación
  - Bloques de tamaño variable (segmentos)

- La decisión afecta al procesador.



Organización de la MV

Paginación vs. Segmentación

	Page	Segment
Words per address	One	Two (segment and offset)
Programmer visible?	Invisible to application programmer	May be visible to application programmer
Replacing a block	Trivial (all blocks are the same size)	Difficult (must find contiguous, variable-size, unused portion of main memory)
Memory use inefficiency	Internal fragmentation (unused portion of page)	External fragmentation (unused pieces of main memory)
Efficient disk traffic	Yes (adjust page size to balance access time and transfer time)	Not always (small segments may transfer just a few bytes)

Organización de la MV

Estrategias híbridas

- Segmentación con paginado
  - Cada segmento se divide en un número entero de páginas.
  - Busca obtener lo mejor de ambos.
- Paginación con tamaños variables
  - Múltiples tamaños de páginas.
  - El tamaño mayor es un múltiplo potencia de 2 del tamaño menor.

Dónde ubicar un bloque en MP

- La penalización de un faltante en MV es muy alta.
- Se elige menor tasa de fallos sobre simplicidad del algoritmo.
- Estrategia *fully associative*: Los bloques se pueden ubicar en cualquier lugar de la memoria.

Cómo encontrar un bloque en MP

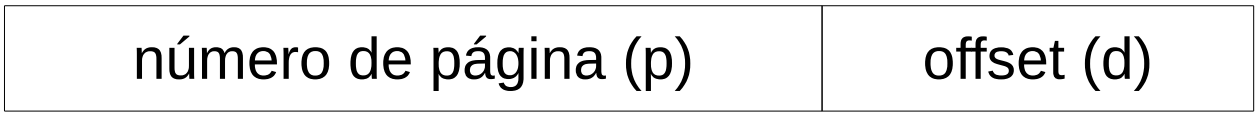
- Estructura de datos indexada por el número de página (paginación) o el número de segmento (segmentación).
- Una tabla que contiene la dirección física del bloque (página o segmento).

- La dirección física del dato buscado:

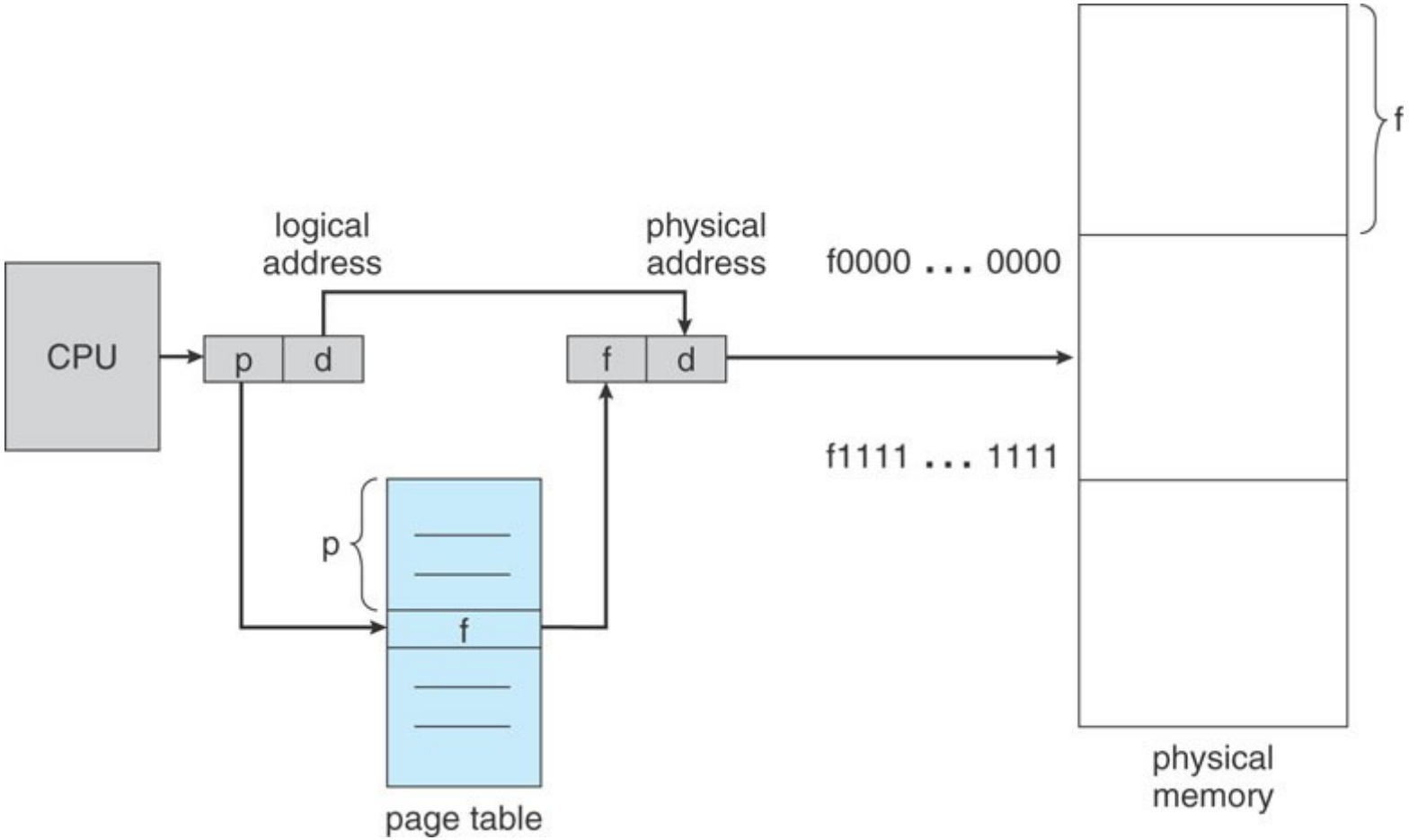
f(dirección física página|segmento , offset)

Paginación

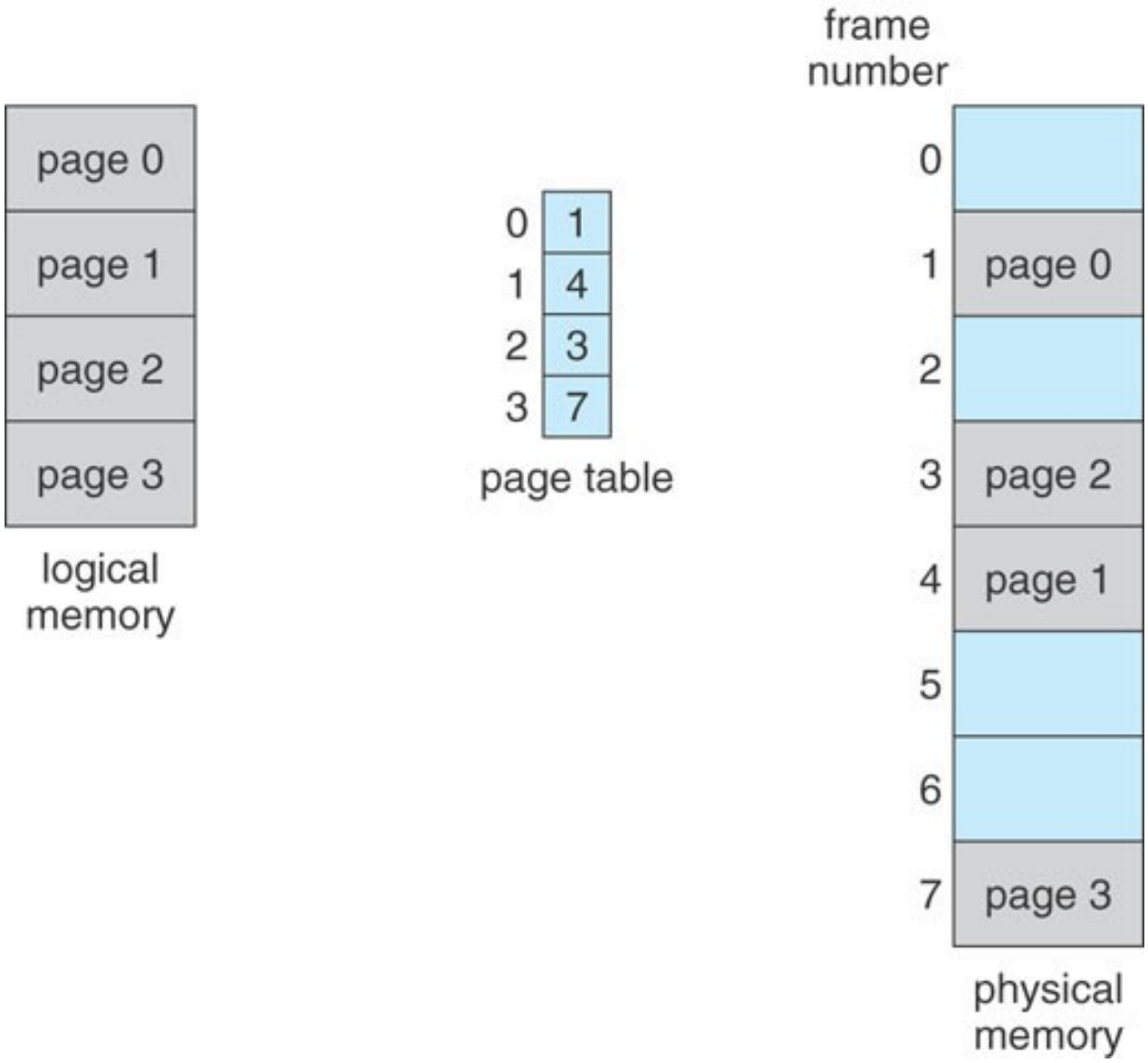
- *Frame*: bloques en los que se divide la memoria principal.
- Página: bloque en los que se divide la memoria virtual.
- Tabla de páginas: tabla indexada por número de página que contiene la dirección base de la página en memoria física.
- Las direcciones generada por el CPU se divide en dos:



Paginación

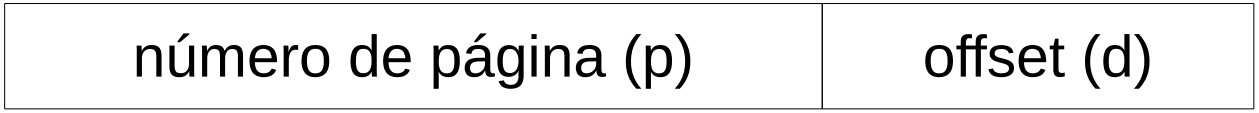

CPU
logical address
p
d
physical address
f
d
f0000 . . . 0000
f1111 . . . 1111
f
p
f
page table
physical memory

Paginación


page 0
page 1
page 2
page 3
logical memory
0 1
1 4
2 3
3 7
page table
frame number
0
1 page 0
2
3 page 2
4 page 1
5
6
7 page 3
physical memory

Paginación

- Las direcciones generada por el CPU se divide en dos:



- El hw determina
  - El tamaño del bloque
  - La longitud de la dirección virtual
- Por lo tanto también determina la cantidad de páginas posibles.

Implementación de la Tabla de páginas

Registros de alta velocidad

- Realizan de manera eficiente la traducción de dirección virtual a física
- Es útil cuando la tabla de páginas es pequeña.
- DEC PDP-11 (1970/80)
  - 16 bits de dirección lógica
  - 8K de página → 13 bits de offset
  - 3 bits de número de página → tabla de páginas de 8 entradas

✗ No es aplicable en computadoras modernas.

Implementación de la Tabla de páginas

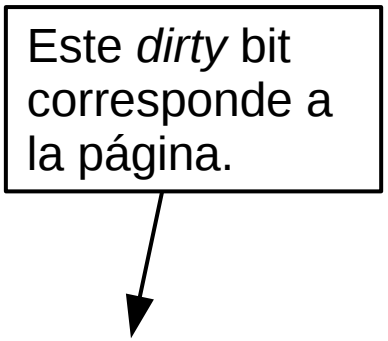
Tabla de páginas en MP

- El hw tiene un registro PTBR (*Page-Table Base Register*) que apunta a la dirección base de la tabla de páginas.
- Cambiar de tabla de páginas requiere solo cambiar el registro.
- ✗ Cada acceso a memoria demanda dos accesos a memoria
  1) Acceder a la tabla de páginas para realizar la traducción
  2) Acceder al byte buscado

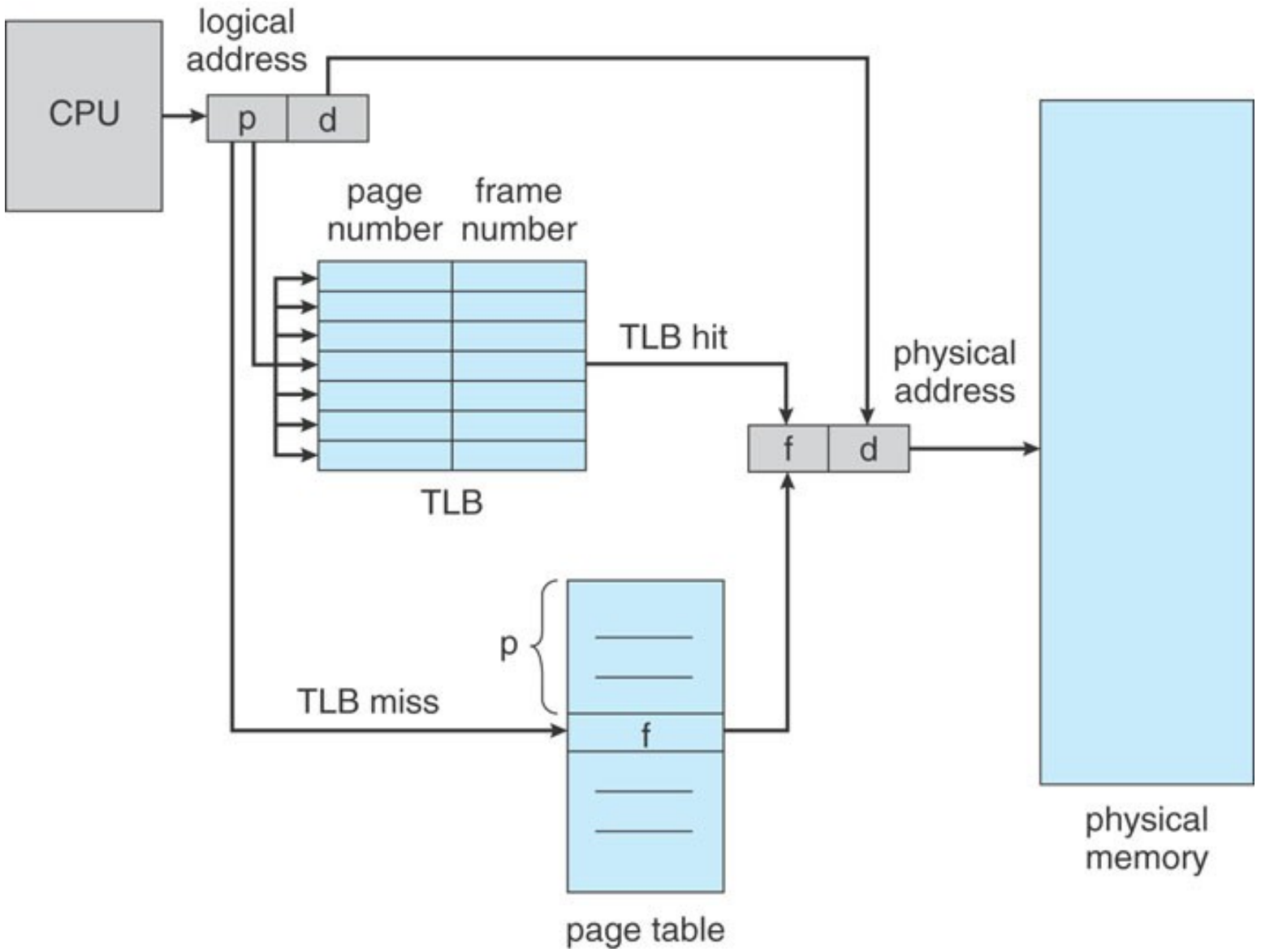
Implementación de la Tabla de páginas

Translation look-aside buffer (TLB)

- Un conjunto de registros de alta velocidad *fully associative*
- Búsqueda rápida pero hw caro. Pocas entradas (entre 8 y 2048).
- Cada entrada en el TLB tiene:
  - TAG (key)
  - Valor
    - Número de *frame* físico
    - Bit de válidos, bits de protección, *use* bit, *dirty* bit



Implementación de la Tabla de páginas



Implementación de la Tabla de páginas

- Cuando el número de página no se encuentra en el TLB hay que hacer una referencia a memoria para acceder a la tabla de páginas y obtener el *frame* correspondientes.
- Además, se actualiza el TLB para futuros accesos.
- Si el TLB está lleno, el SO decide qué entrada reemplazar.
- Cuando se cambia de tabla de páginas hay que limpiar el TLB.

Paginación Multinivel

- Supongamos
  - Dirección lógica de 32 bits
  - Página de 4KB (12 bits de offset)
  - Hay 2^{20} entradas en la tabla de páginas
  - Si cada entrada tiene 4 bytes → 4MB de tabla de páginas
- La tabla de páginas se divide niveles.
- El primer nivel siempre se mantiene en memoria principal.

Paginación Multinivel

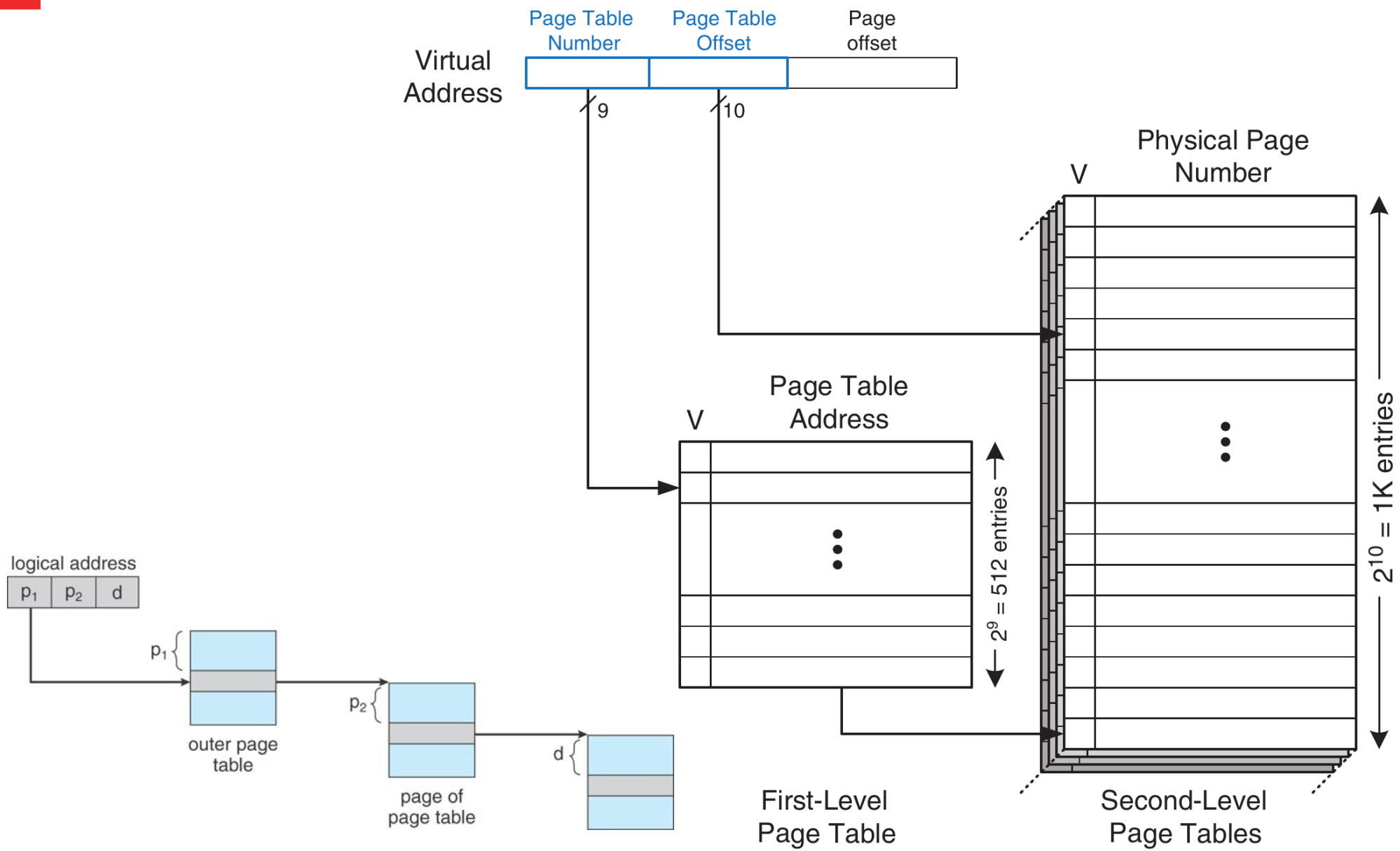
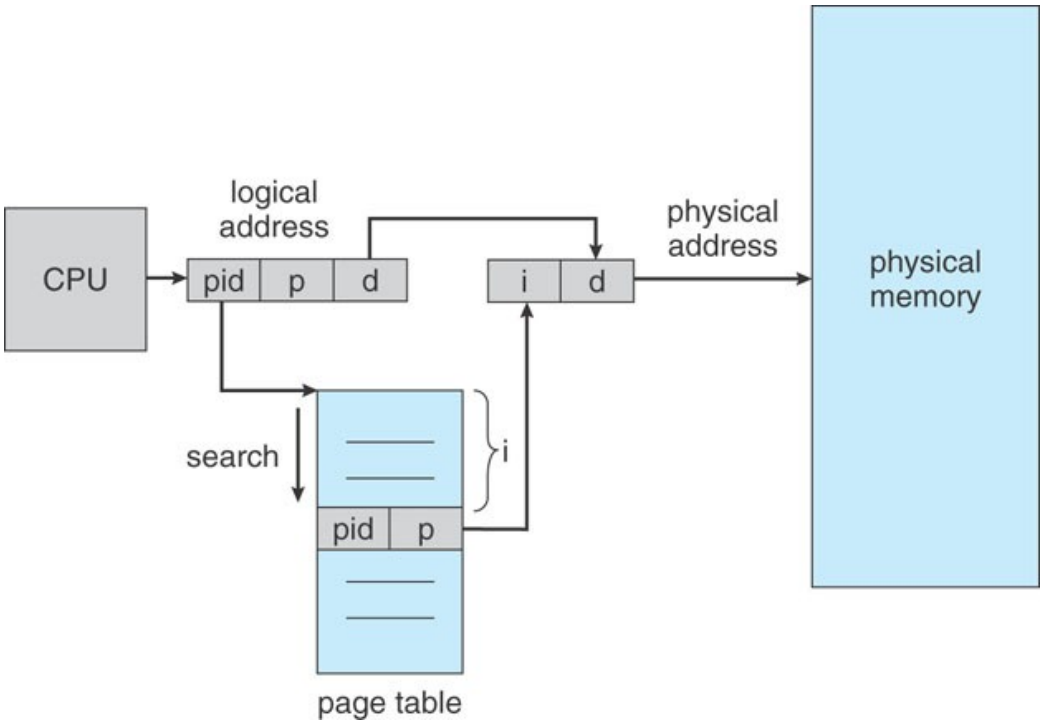


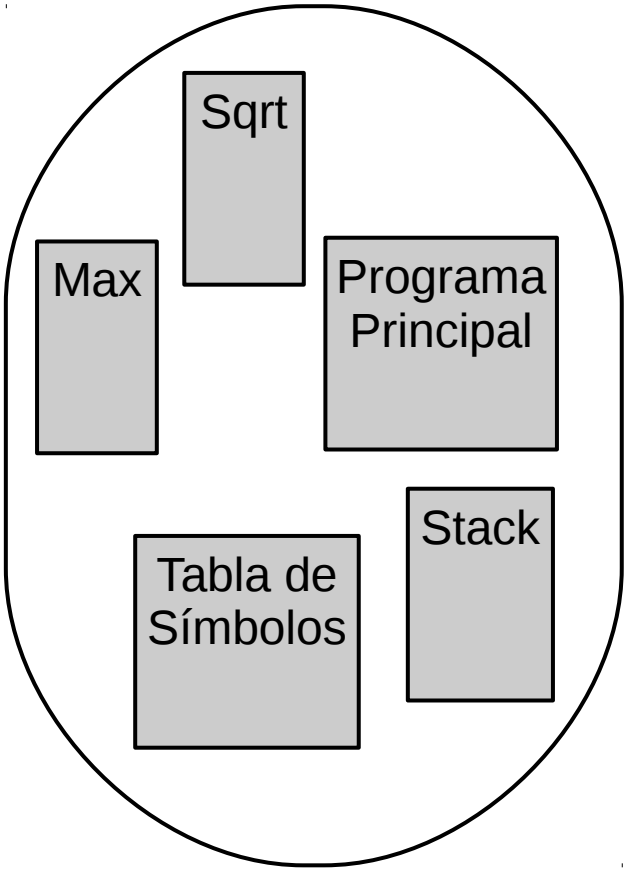
Tabla de páginas invertida

- La cantidad de *frames* es mucho menor que la cantidad de páginas virtuales.
- Tienen una entrada por cada *frame* en memoria principal.
- Reduce el tamaño de la tabla de páginas.
- HP/Intel IA-64 ofrece las dos opciones. Deja al programador del SO la elección de que mecanismo usar.



Segmentación

- La vista del usuario de la memoria no es complemente lineal.
- Varios módulos con un propósito propio que interactúan entre sí.
- No hay un orden definido entre los módulos.
- Los módulos son de tamaño variable.



Espacio virtual

Segmentación

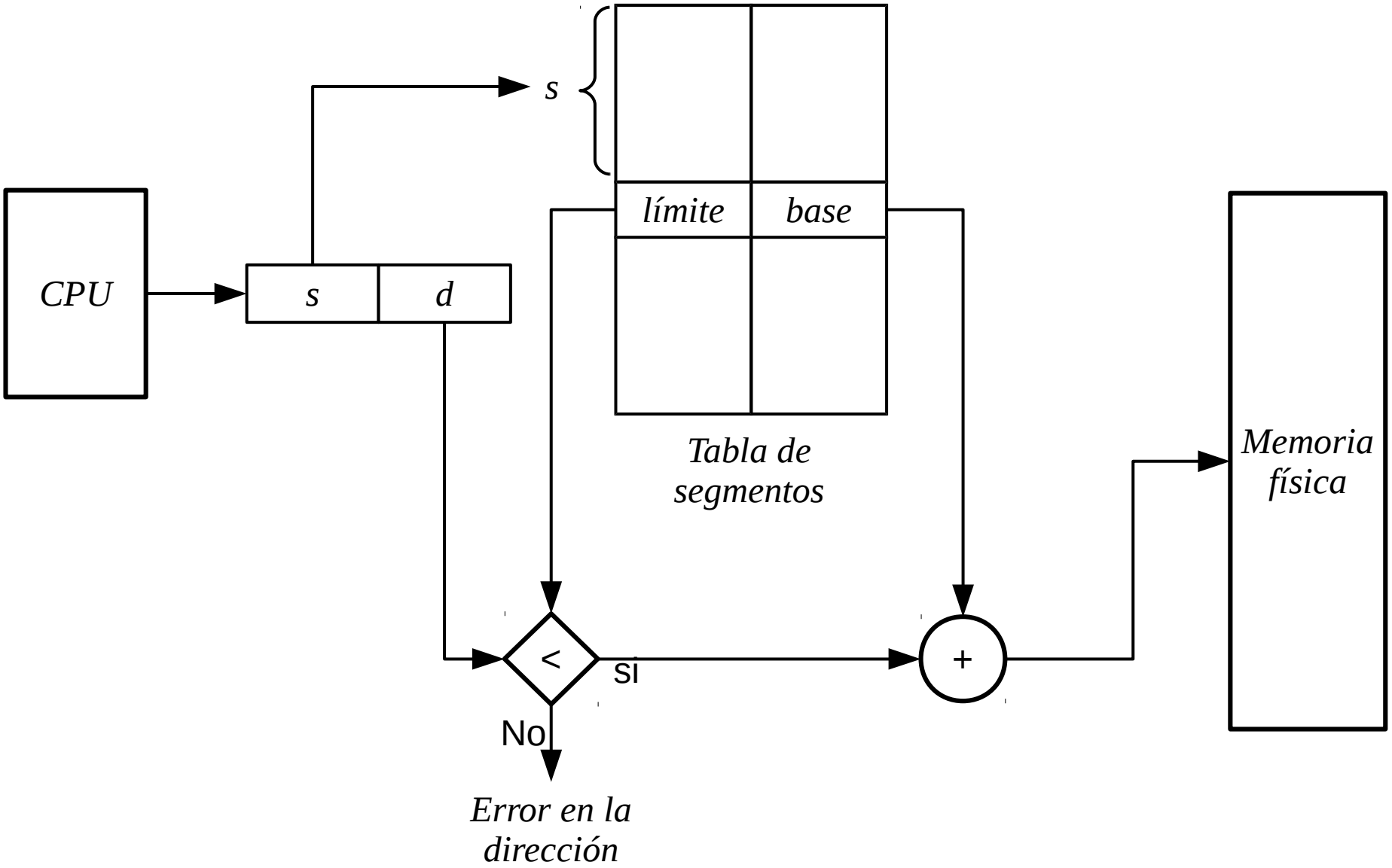
- Segmentación refleja el punto de vista del usuario en la administración de la memoria.
- El espacio direccionable se divide en un conjunto segmentos.
- Cada segmento tiene un nombre y una longitud.
  - Consiste de una secuencia lineal de direcciones, de 0 a un máximo.
- Una dirección especifica el nombre del segmento (número) y el offset dentro del segmento.

número de segmento (s)	offset (d)

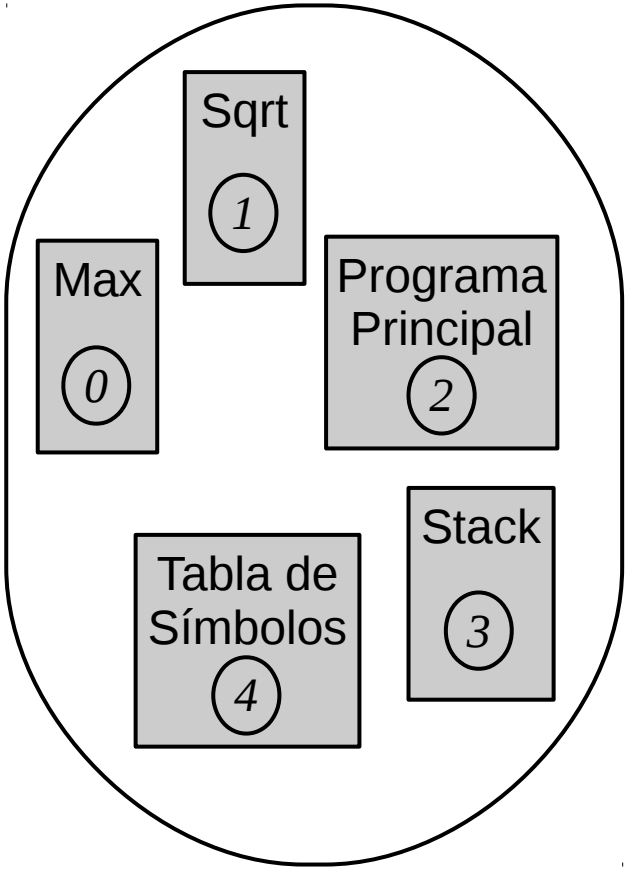
Segmentación

- Normalmente, el compilador crea los segmentos automáticamente.
- Un compilador de C puede crear segmentos para:
  1) Código
  2) Variables globales
  3) El heap
  4) La pila
  5) Standard C library
- El *loader* se asigna números a los segmentos.

Segmentación



Segmentación



Espacio virtual

límite	base
1000	1400
400	6300
400	4300
1100	3200
1000	4700

Tabla de segmentos

Implementación de la Tabla de Segmentos

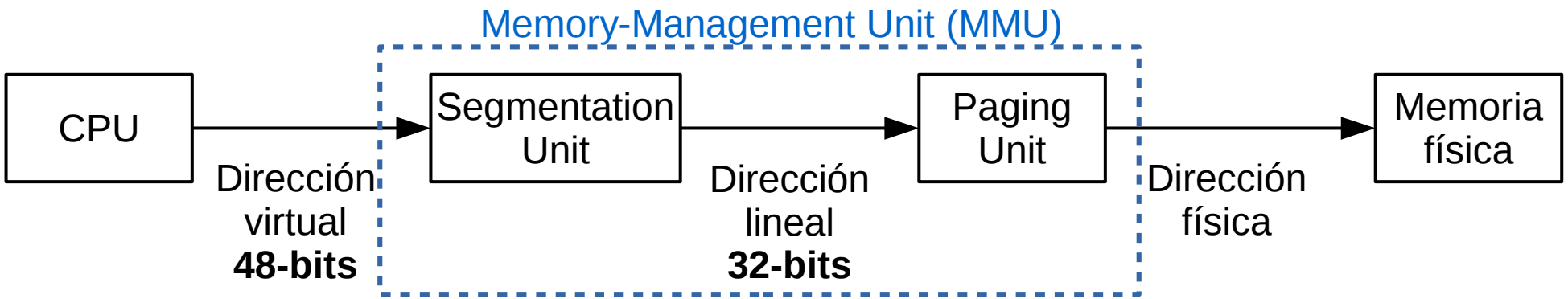
- Registros
  - ✔ Rápido
  - ✗ Muy grande
- Memoria
  - Requiere un registro STBR (*segment-table base register*) con la dirección base del segmento y otro registro STLR (*segment-table length register*) para la longitud.
  - ✗ El mapeo de dirección lógica a dirección física requiere dos accesos a memoria.
- Tabla de segmentos en memoria + TLB

Segmentación con Paginación

- Los segmentos se dividen en páginas de tamaño fijo.
- Algunas de las páginas del segmento están en MP y otras en memoria secundaria.
- Al paginar los segmento (que son un espacio lineal), cada segmento necesita su propia tabla de páginas.

Intel Pentium

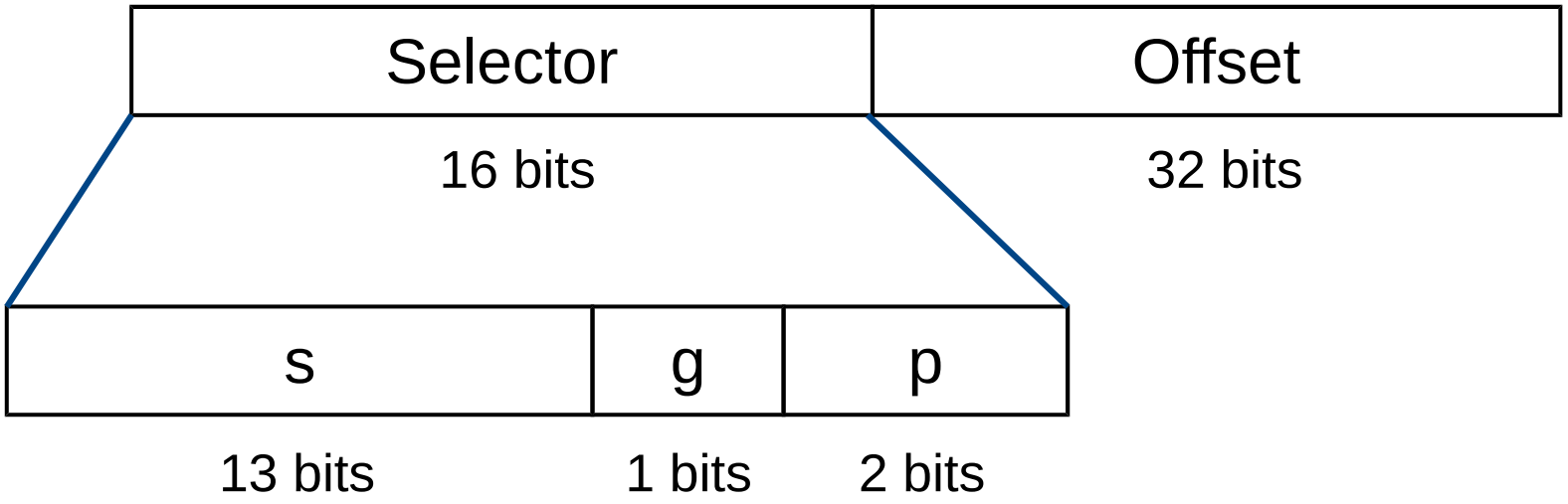
- Soporta segmentación pura y segmentación con paginado.



Segmentación

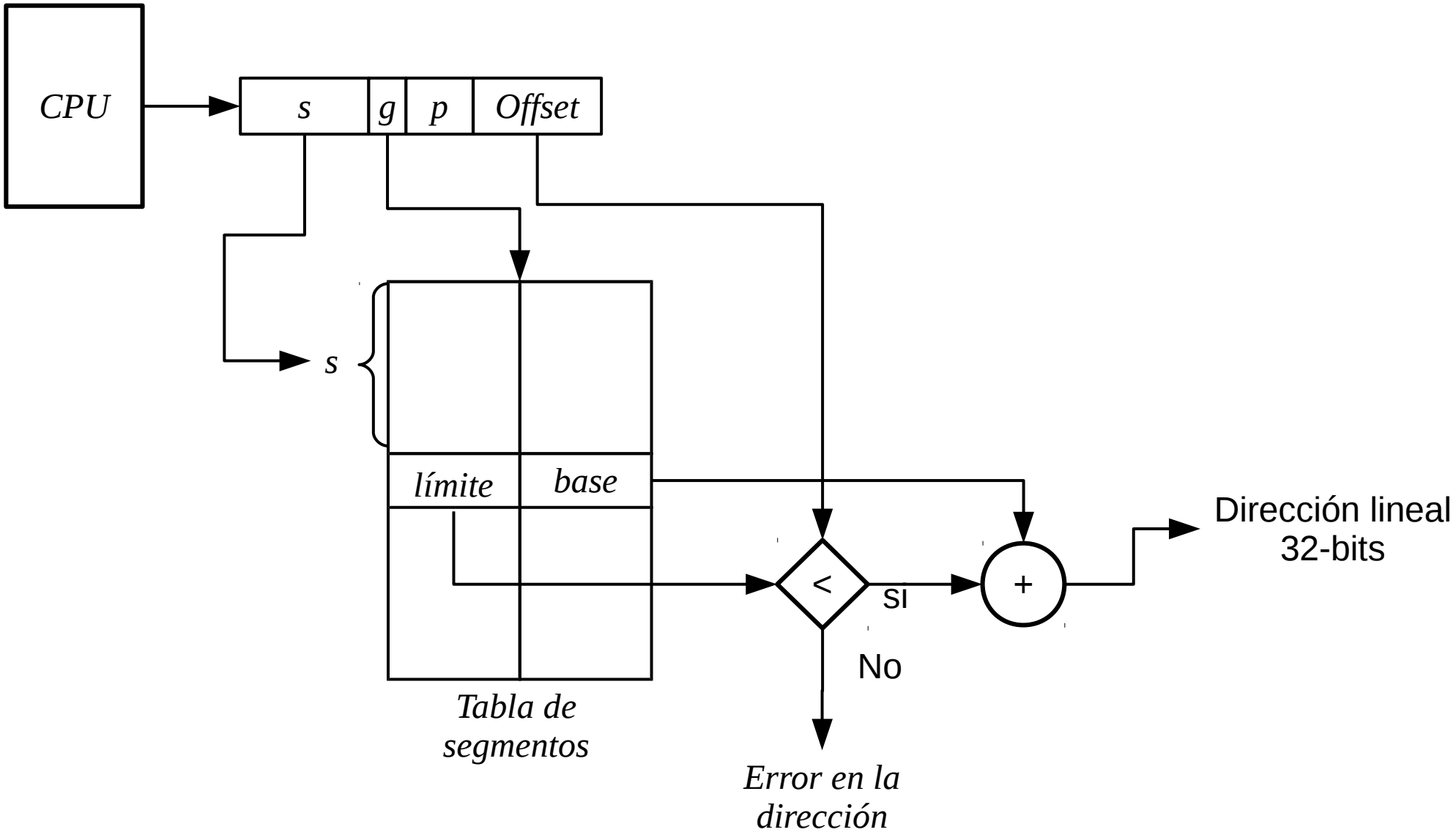
- Permite segmentos de hasta 4GB
- Cada proceso puede tener hasta 16K segmentos.
- El espacio virtual está dividido en 2 (8K segmentos privados al proceso y 8K segmentos compartidos)
- Local descriptor table (LDT) tiene información de los segmentos privados.
- Global descriptos table (GDT) tiene información de los segmentos globales.
- El descriptor del segmento (8 bytes) incluye dirección base y límite.

Dirección lógica



- s: número de segmento
- g: si el segmento está en LDT o en GDT
- p: bits de protección
- Offset indica la posición del byte dentro del segmento.

Segmentación

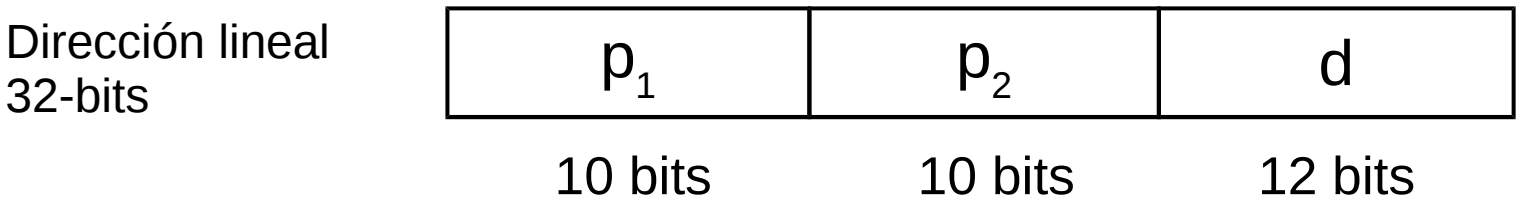


Segmentación

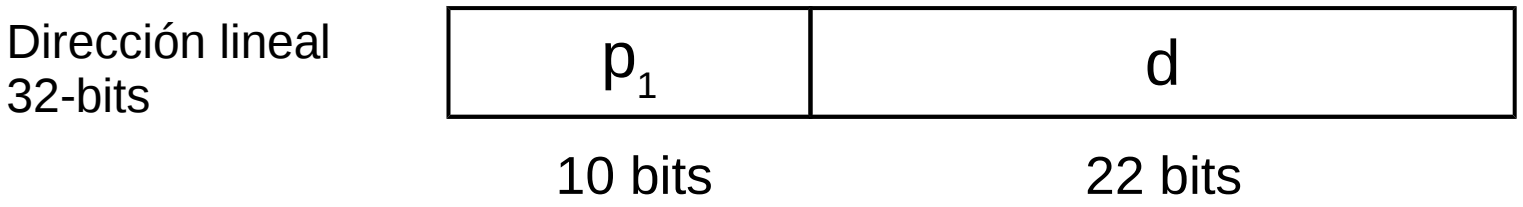
- El procesador tiene una pequeña caché con
  - 6 registros para números de segmentos.
  - Una tabla de 6 entradas de 8 bytes para almacenar descriptores de la LDT o GDT

Paginación

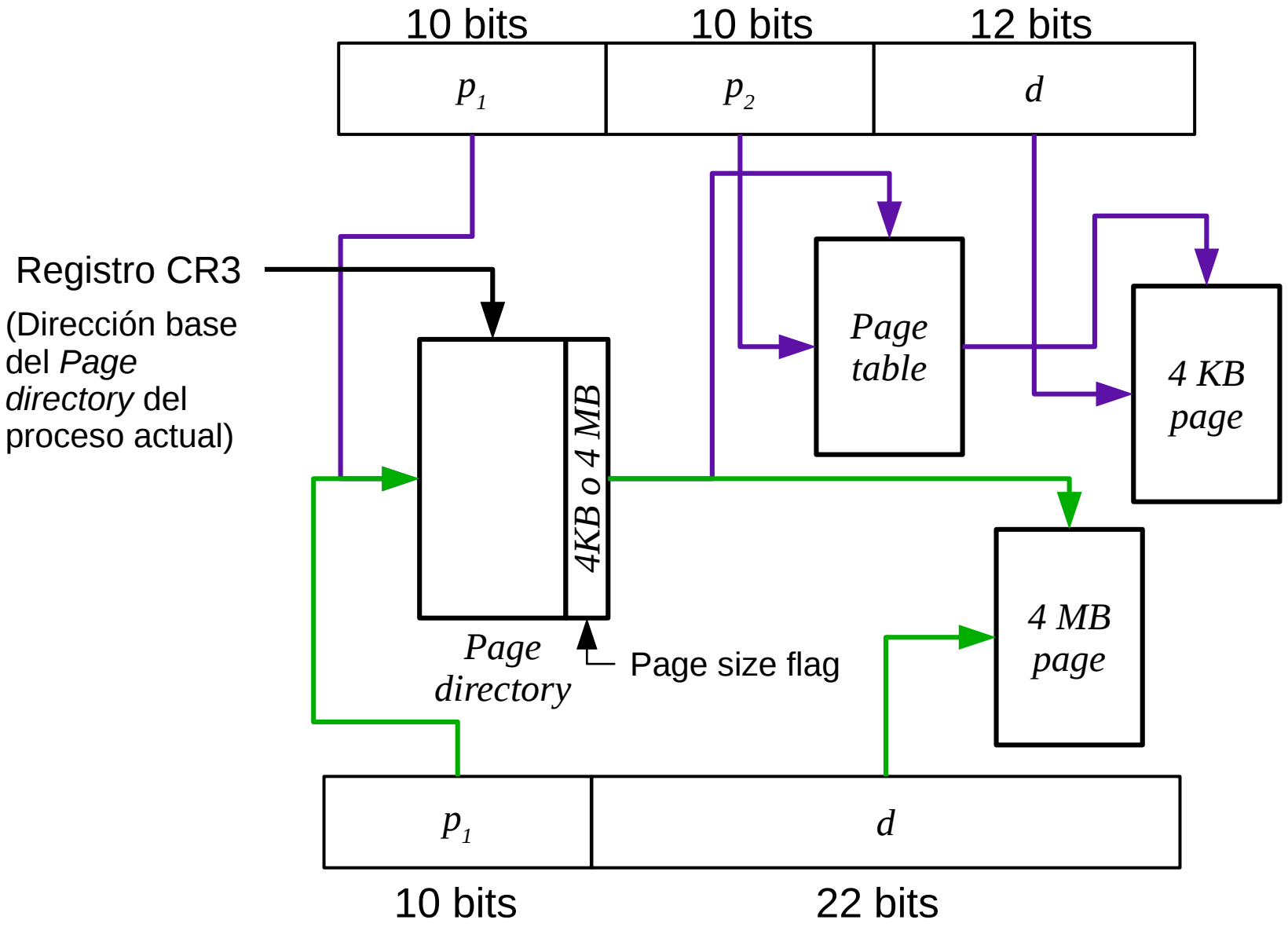
- Permite páginas de 4KB o de 4MB
- Para las páginas de 4KB usa 2 niveles.



- Para las páginas de 4MB usa un único nivel



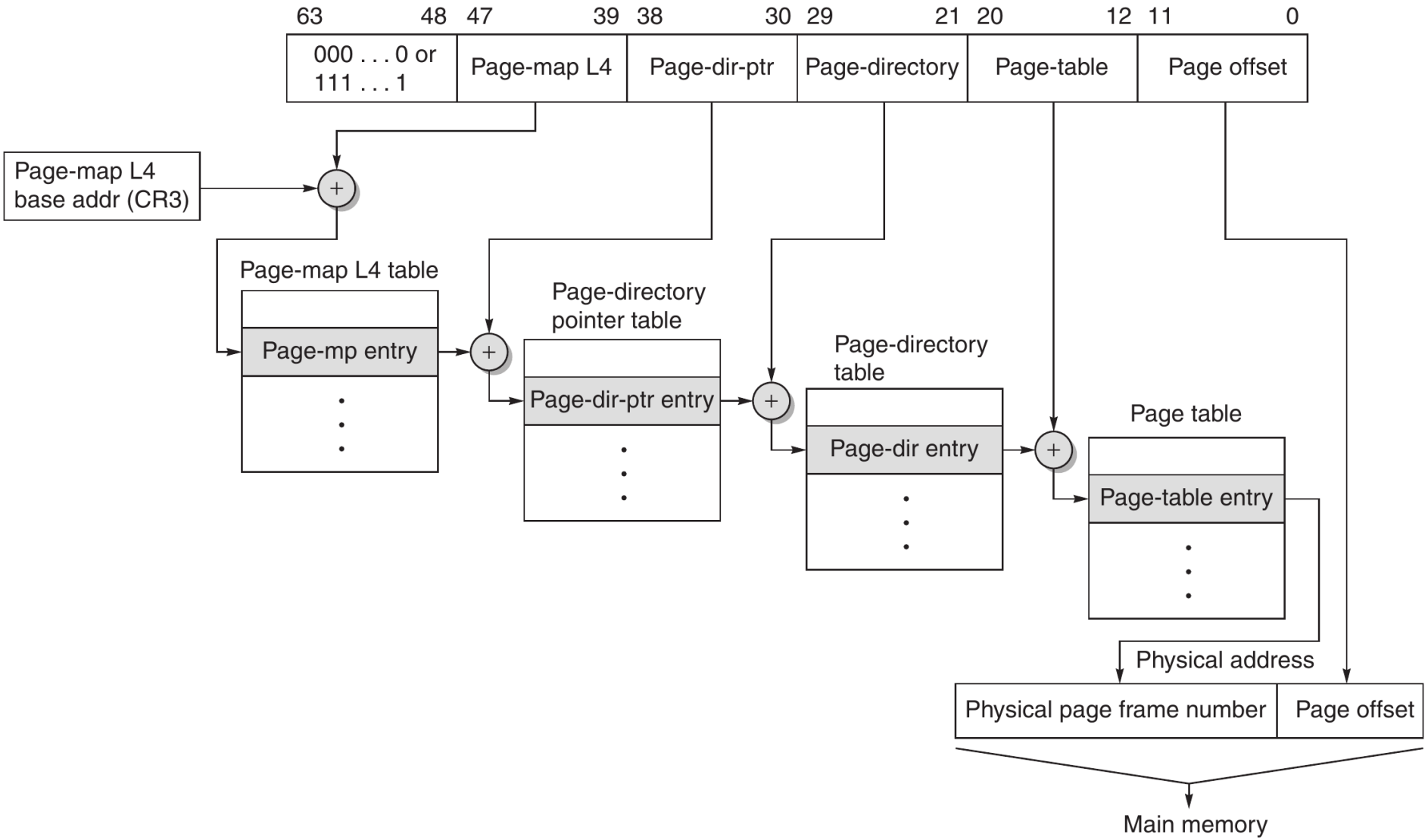
Paginación


10 bits
10 bits
12 bits
p_1
p_2
d
Registro CR3
(Dirección base del *Page directory* del proceso actual)
Page table
4 KB page
4KB o 4 MB
Page directory
Page size flag
4 MB page
p_1
d
10 bits
22 bits

AMD Opteron

- Los 64-bits del espacio virtual de AMD64 se mapea a una dirección física de 52 bits.
- Como el espacio virtual es extremadamente grande, puede reducirse en la implementación.
- AMD Opteron
  - La dirección virtual es de 48-bits y la dirección física de 40-bits.
  - Los 16 bits restantes en la dirección virtual son extensión de signo.
- AMD64 deja de lado la segmentación (soporta modo compatibilidad) y prefiere el modelo lineal de paginación.

AMD Opteron

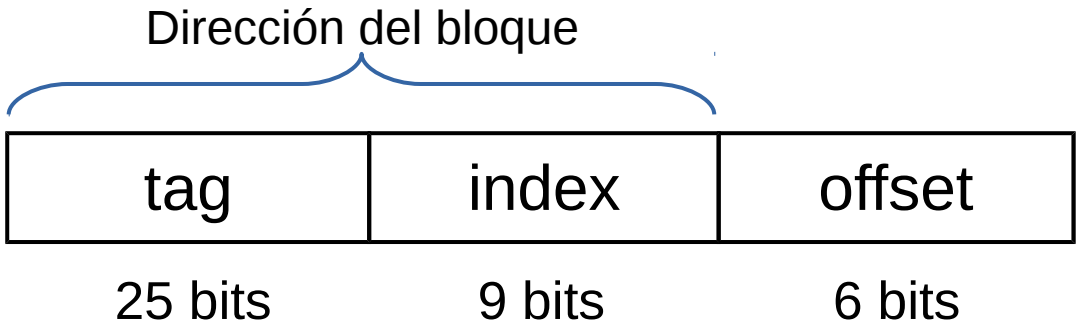

63
48
47
39
38
30
29
21
20
12
11
0
000 . . . 0 or
111 . . . 1
Page-map L4
Page-dir-ptr
Page-directory
Page-table
Page offset
Page-map L4
base addr (CR3)
Page-map L4 table
Page-mp entry
Page-directory
pointer table
Page-dir-ptr entry
Page-directory
table
Page-dir entry
Page table
Page-table entry
Physical address
Physical page frame number
Page offset
Main memory

AMD Opteron Cache

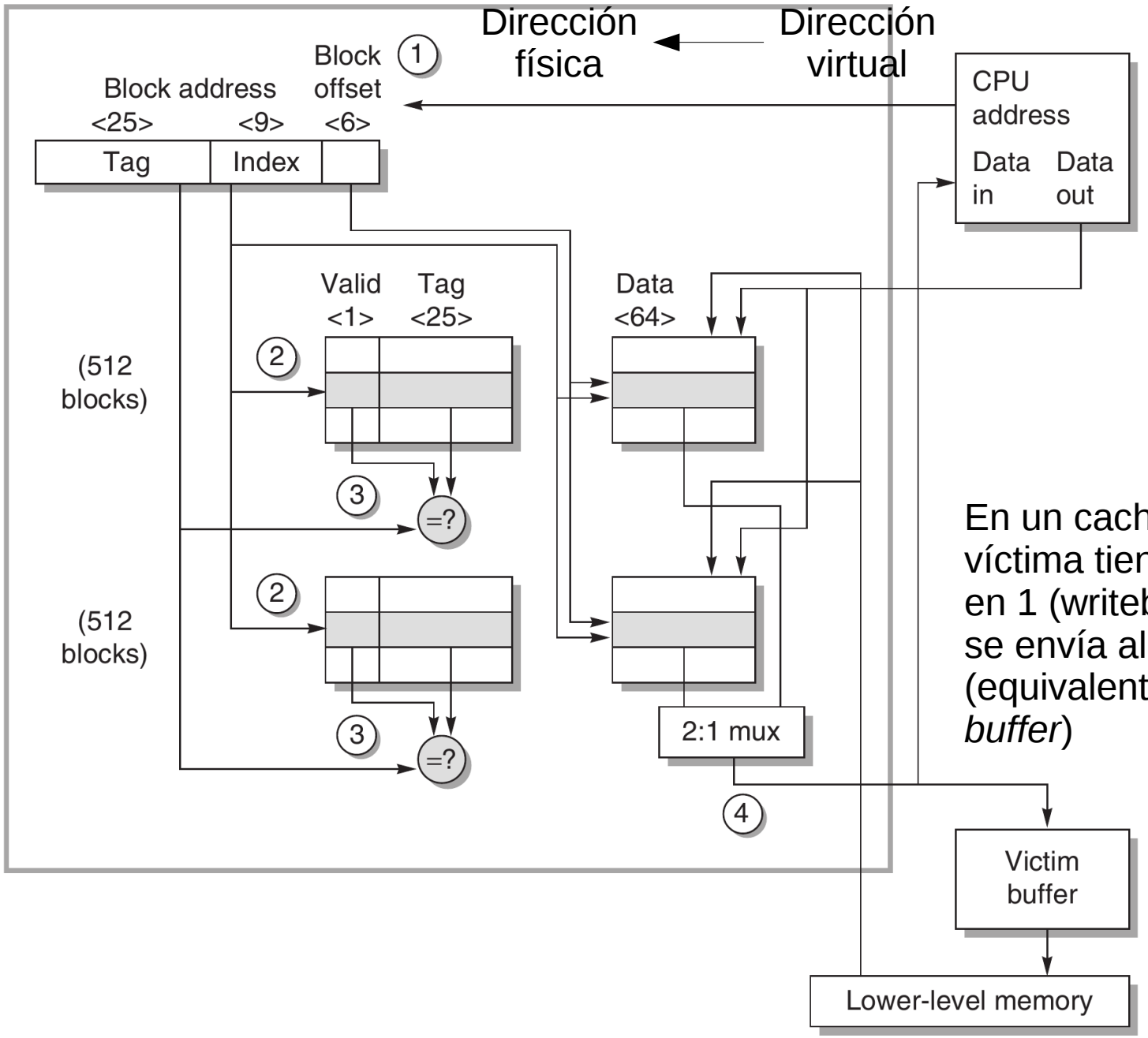
- La caché contiene 64KB de dato en bloques de 64 bytes en una cache
  - 2-way set associative
  - Reemplazo LRU
  - Write-back
  - Write-allocate

Tiene caches separadas para datos e instrucciones.

- La dirección física de 40 bits se divide en 3 campos

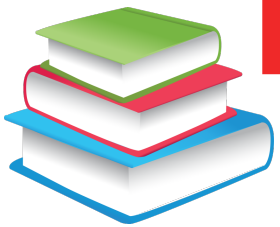


AMD Opteron Cache



En un caché miss, si la víctima tiene el dirty bit en 1 (writeback cache) se envía al *victim buffer* (equivalente al *write buffer*)

Bibliografía



- *Capítulo 5 y Apéndice B.* David A. Patterson & John L. Hennessy. *Computer Organization and Design. The Hardware/Software Interface.* Elsevier Inc. 2014, 5ta Ed.
- *Capítulo 8 y 9.* Abraham Silberschatz & Peter Baer Galvin. *Operating System Concepts*. Addison Wesley 1998. 5ta Ed. (en adelante).

Suplementaria

- AMD64 Technology. *AMD64 Architecture Programmer's Manual Volume 2: System Programming*. Revision 3.23, Mayo 2013. developer.amd.com/wordpress/media/2012/10/24593_APM_v21.pdf