MEMORIA VIRTUAL

Definición

Gestión de memoria automática que da al programador la ilusión de que su espacio de direccionamiento no está limitado por el espacio de memoria principal reservado a su programa (espacio físico), sino por el rango de direcciones que permite el sistema (espacio virtual).

Ventajas de la memoria virtual

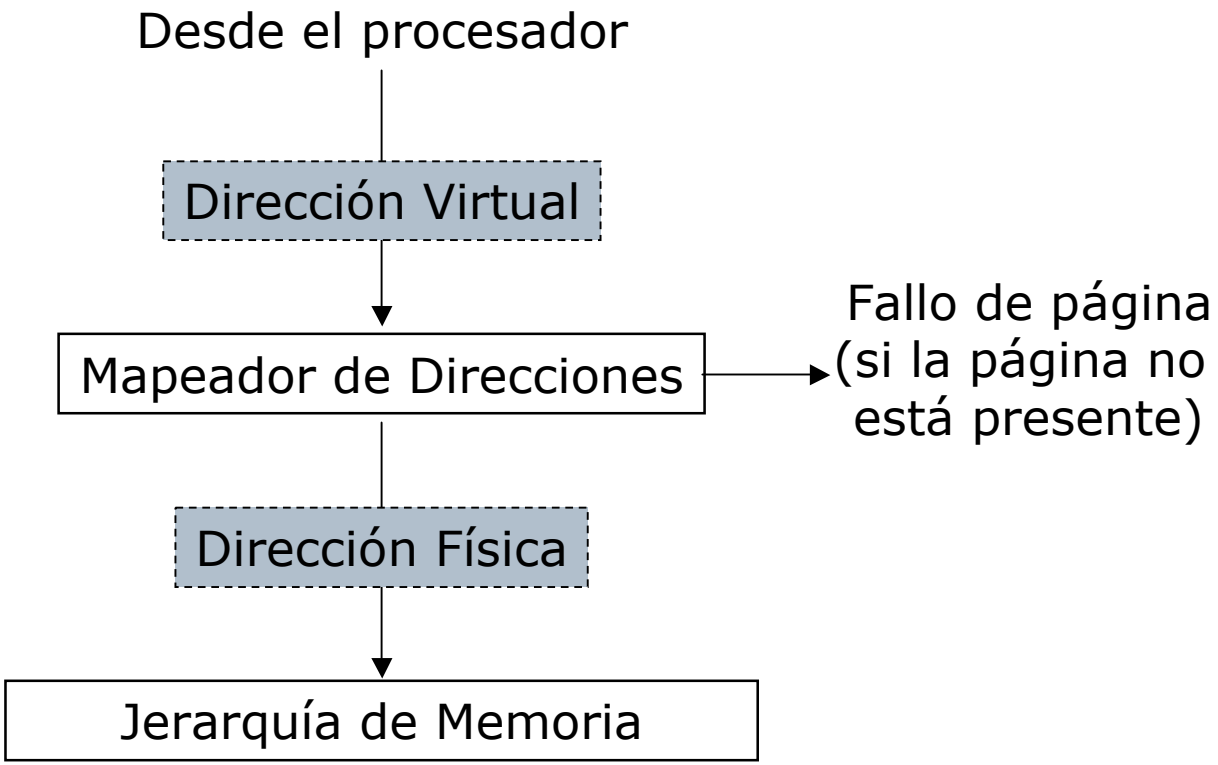
- El espacio virtual puede ser mucho mayor que el espacio físico. (A veces menor, raro).
- Facilita la multiprogramación.
- Mejor aprovechamiento de la memoria principal.
- Facilita la protección de los programas.
- Transparente al programador.

Inconvenientes de la memoria virtual

- Gasto temporal relativamente elevado de la gestión de memoria (traducción de direcciones, reemplazos de bloques reservados, etc.)
- Gasto de procesamiento en la resolución de excepciones.
- Gasto hardware para conseguir una gestión de memoria rápida y eficiente (MMU, Memory Management Unit).

Estructura de la gestión de la memoria virtual

- Traducción hw-sw:



UNIVERSIDADE DA CORUÑA

GRUPO DE ARQUITECTURA DE COMPUTADORES

Estrategias de implementación de la memoria virtual (I)

- MMU interna:
  - MMU en el mismo circuito integrado que el procesador.
  - Se da en casi todos los procesadores actuales
  - Ventajas:
    - Tiempos de acceso reducidos.
    - Alta portabilidad de programas.
    - Compartición de hardware entre el procesador y la MMU.

Estrategias de implementación de la memoria virtual (II)

- MMU externa:
  - MMU en un circuito integrado independiente.
  - Ejemplos:
    - Motorola 68000/10 (68020) + 58451 (68851)
    - Zilog 8001 (8803) + 8010 (8015)
    - Nat. Semic. 16000 + 16082
  - Ventajas:
    - Se ahorra espacio del circuito integrado del procesador para otros recursos (caché, etc.)

Definiciones (I)

- Espacio virtual ≡ espacio nominal ≡ espacio de direcciones:
  - Conjunto de direcciones que puede direccionar un proceso.
- Espacio físico ≡ espacio de memoria ≡ reserva de memoria primaria (PMA):
  - Espacio de memoria principal reservado para el proceso.

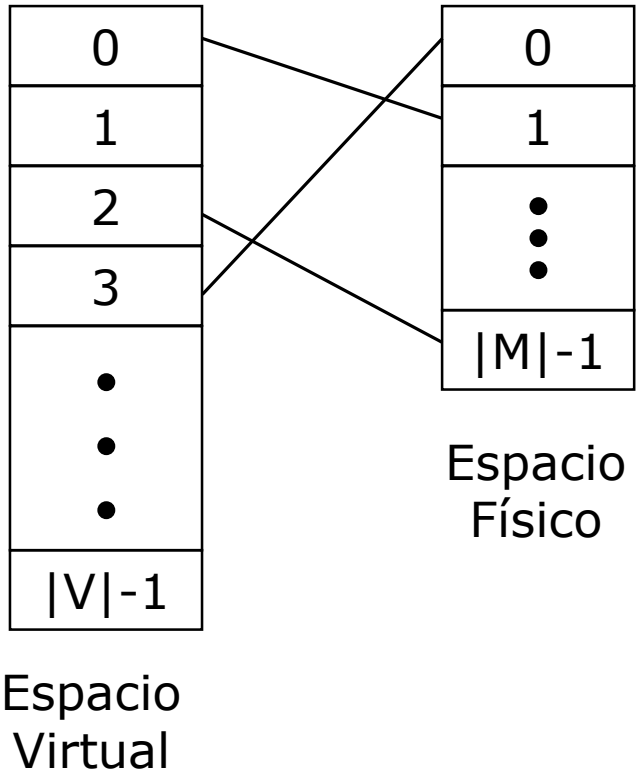
Definiciones (II)

- Traductor de direcciones:
  - V: espacio virtual
  - M: espacio físico

$$f : V \rightarrow M \cup \{\phi\}$$

$x \in V \Rightarrow f(x) = y$, si x está en M en la posición y

$f(x) = \phi$, en otro caso



Definiciones (III)

- Excepción o fallo de direccionamiento:
  - Se produce cuando f(x) = ø.
  - La referencia x debe transferirse de la memoria secundaria a la primaria.
- Reglas para resolver los fallos de direccionamiento:
  - Regla de carga: CUANDO se transfiere x.
  - Regla de ubicación: DONDE se sitúa x en la memoria principal.
  - Regla de reemplazo: QUE referencia virtual situada en la memoria principal debe eliminarse para hacer sitio a x. (sólo si la memoria principal está totalmente ocupada).

UNIVERSIDADE DA CORUÑA

GRUPO DE ARQUITECTURA DE COMPUTADORES

Clasificación de los sistemas de memoria virtual (I)

- Los sistemas de memoria virtual agrupan las referencias virtuales en bloques.
- Una referencia virtual, por tanto, está compuesta de dos campos: número de bloque y desplazamiento dentro del bloque.
- Estos bloques se consideran como las unidades de transferencia de información entre la memoria secundaria y la principal.

UNIVERSIDADE DA CORUÑA

GRUPO DE ARQUITECTURA DE COMPUTADORES

Clasificación de los sistemas de memoria virtual (II)

- El traductor de direcciones, por tanto, sólo debe traducir el campo de bloque, dejando invariante el desplazamiento.
- El tamaño del traductor, de esta manera, es proporcional al número de bloques del espacio virtual (o físico). (Justificación de la definición de bloques).
- Referencia: Nº de bloque + desplazamiento
  - Sólo es necesario traducir el número de bloque.

UNIVERSIDADE DA CORUÑA

GRUPO DE ARQUITECTURA DE COMPUTADORES

Clasificación de los sistemas de memoria virtual (III)

- Tipos de sistemas de memoria virtual según el tamaño de los bloques:
  - Sistemas paginados:
    - Los bloques son todos del mismo tamaño.
    - Los bloques se llaman páginas y una excepción se llama fallo de página.
  - Sistemas segmentados:
    - Los bloques son de tamaño diferente.
    - Los bloques se llaman segmentos y una excepción se llama fallo de segmento.
  - Sistemas segmentados con paginación:
    - Los bloques (segmentos) son de tamaño desigual pero múltiplo de un tamaño unidad (página).

Sistemas paginados

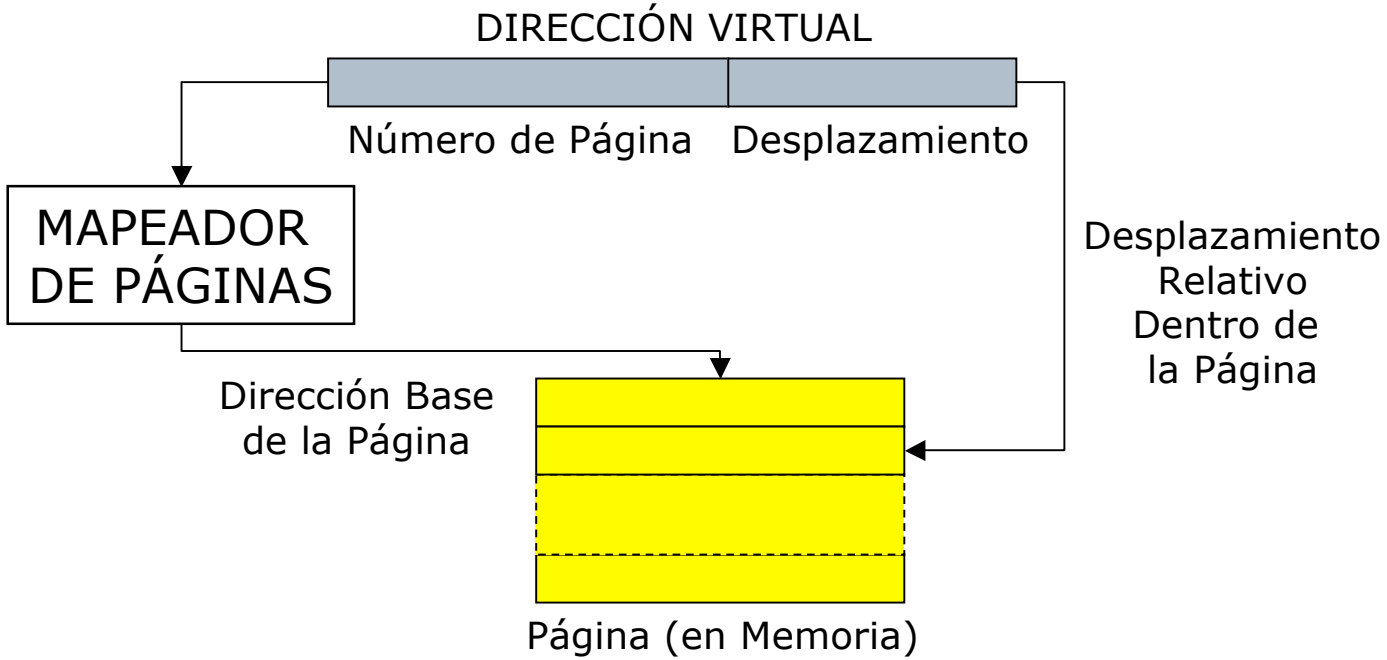
Esquema de memoria virtual más difundido.

Representación

- Programa P:
  - $P = \{p_1, p_2, \ldots, p_N\}$, p_i página virtual.
- Normalmente, tamaño(p_i) = $p = 2^k$.
- Dirección virtual incluída en pi:
  - $a_{ij} = p_i d_j$, $p_i \in P$, $0 \le d_j \le p$
    m bits m-k k bits
- Vector de referencias (referencias a páginas generadas al ejecutarse P):
  - $R = r(1)r(2)\ldots r(n)$, $r(i) = p_j$, $1 \le i \le n$, $1 \le j \le N$

Traducción de direcciones (I)

- Proyección del subespacio virtual asociado a P sobre su espacio físico reservado en la memoria principal (PMA):



Traducción de direcciones (II)

- La traducción sólo afecta al campo de página virtual. El desplazamiento se concatena con el resultado de la traducción.
  - Página virtual ⇨ página física

Esquemas básicos de implementación de la traducción (I)

- Traducción directa:
  - El traductor se implementa mediante una tabla de acceso directo de tamaño |V|/p, llamada tabla de páginas.
    - Tabla indexada por el número de PV
    - Bit de residencia
    - Bit de validez
    - Bit de modificación (siempre postescritura)
    - Reemplazo (ej. Bits de referencia)
    - Protección

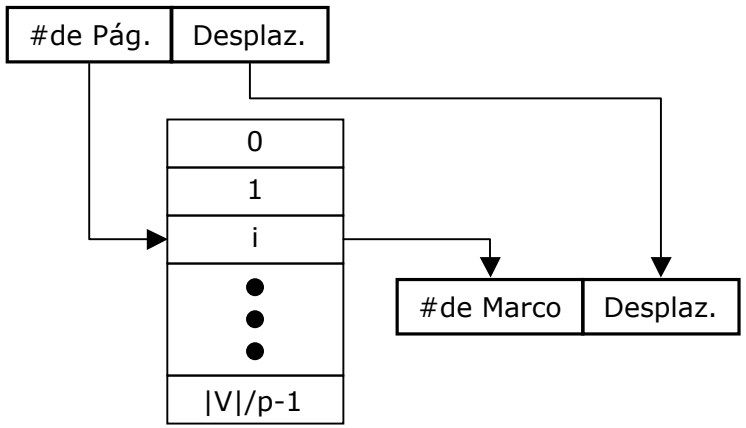
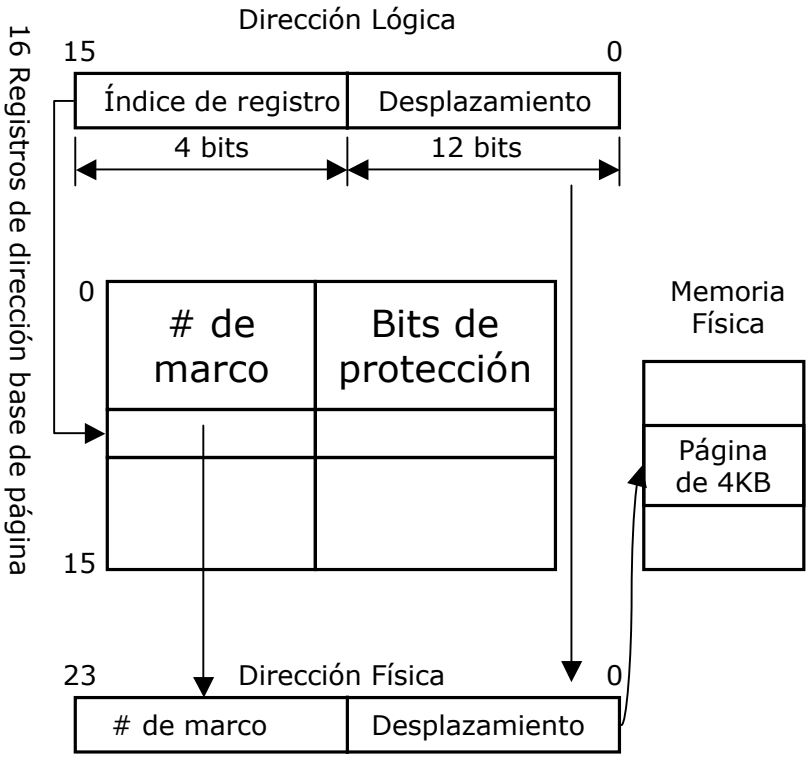


Tabla de Páginas

Esquemas básicos de implementación de la traducción (II)

- Traducción directa (cont.):
  - Almacenamiento de la tabla de páginas:
    - En registros rápidos:
      - Traducción rápida y costosa.
    - En la memoria principal:
      - Traducción más lenta pero menos costosa que la anterior.



Esquemas básicos de implementación de la traducción (II)

- Traducción asociativa:
  - La traducción se implementa mediante una tabla de páginas, pero ésta se almacena en una memoria asociativa. Su tamaño es |M|/p.
  - Tabla de páginas invertida por hash (f(DV) → DF).

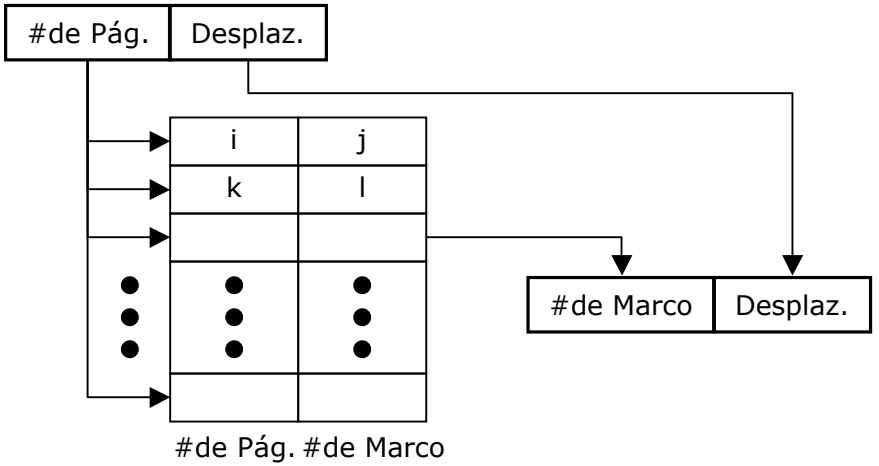


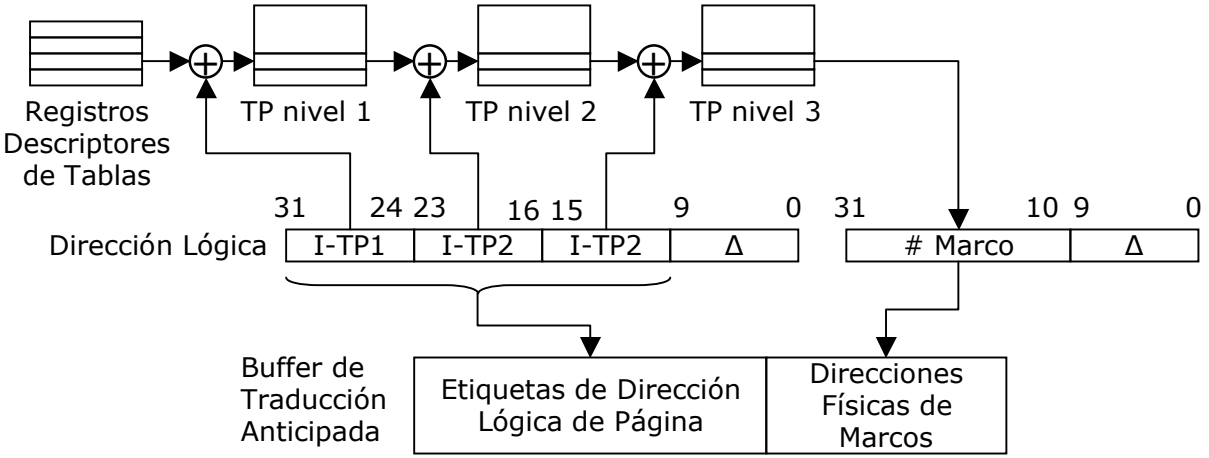
Tabla de Páginas Asociativa (|M|/p entradas)

Esquemas básicos de implementación de la traducción (III)

- Traducción directa en varios niveles:
  - El alto coste de los registros rápidos y de las memorias asociativas así como el gran tamaño de la tabla de páginas de acceso directo restringen el uso de los esquemas de traducción directo y asociativo a sistemas pequeños.
  - La mayoría de los sistemas realizan la traducción mediante un esquema directo en dos niveles, es decir, almacenan la tabla de páginas en el espacio virtual (paginan la tabla de páginas).

Esquemas básicos de implementación de la traducción (IV)

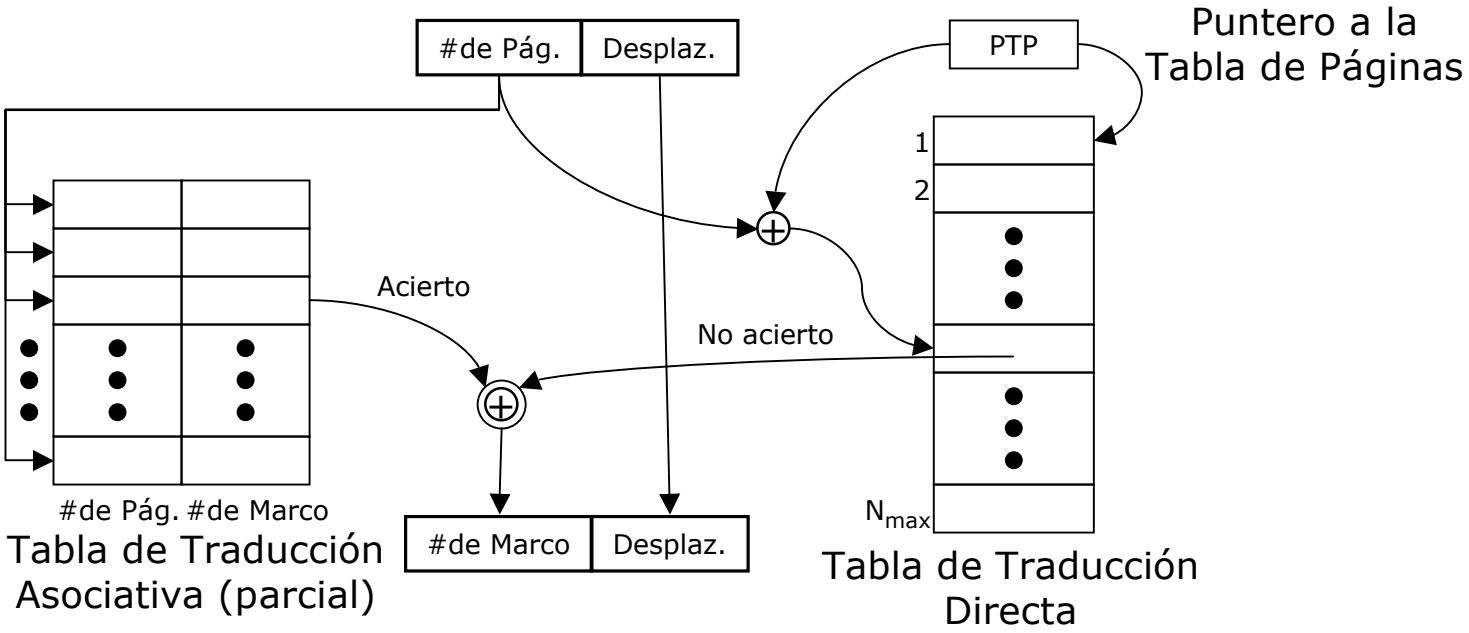
- Traducción directa en varios niveles (cont.):
  - Algunos sistemas amplían la traducción a tres niveles, como el Z80,000.



  - Inconvenientes:
    - La traducción es lenta, pues hay que hacer tres (o cuatro, tres niveles más dato) accesos a la memoria principal.
    - La gestión de los fallos de página es compleja.

Esquemas básicos de implementación de la traducción (V)

- Traducción directa y asociativa combinada:
  - Pretende combinar la ventaja del bajo coste hardware de la traducción directa con la ventaja de la alta velocidad de la traducción asociativa.



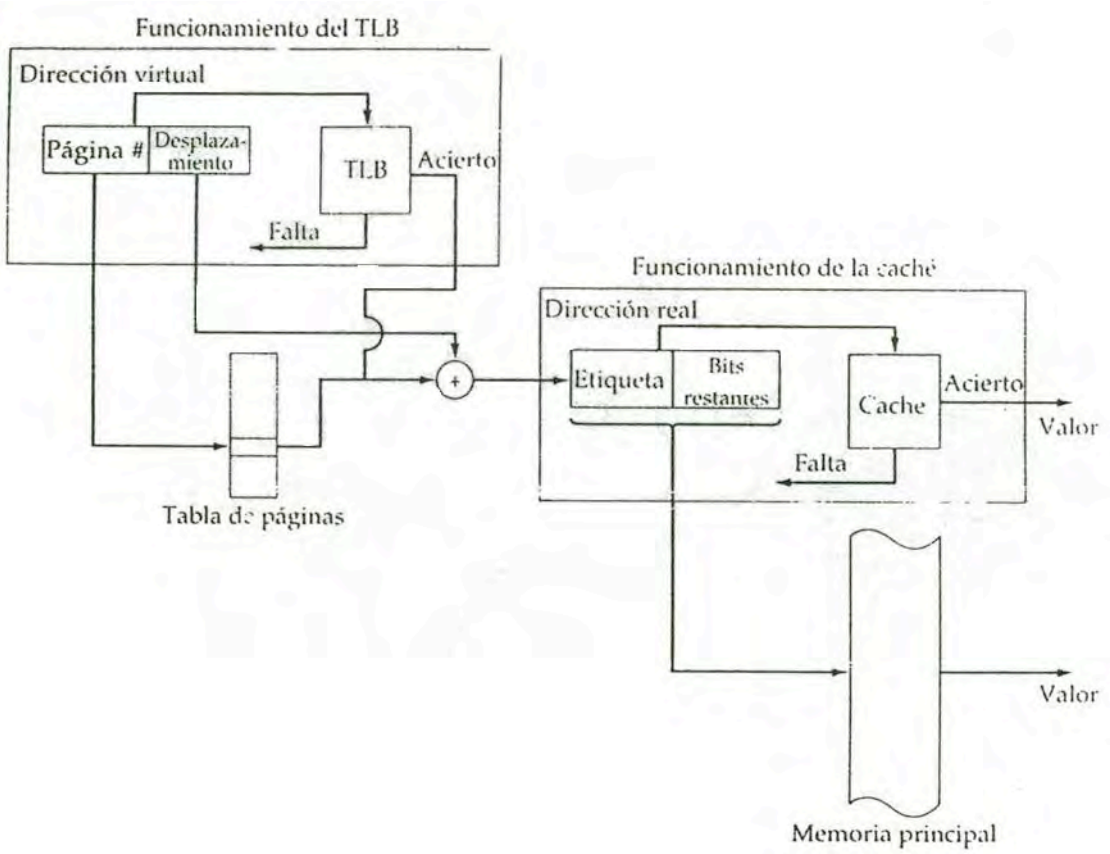
Esquemas básicos de implementación de la traducción (VI)

- Traducción directa y asociativa combinada (cont.):
  - La memoria asociativa o TLB (translation-lookaside buffer) almacena los pares [página virtual, página física] más recientemente referenciados junto con los bits de gestión que se requieran
  - Su éxito está justificado por el principio de localidad.
  - Tamaño típicos de la TLB: 32 a 256 entradas

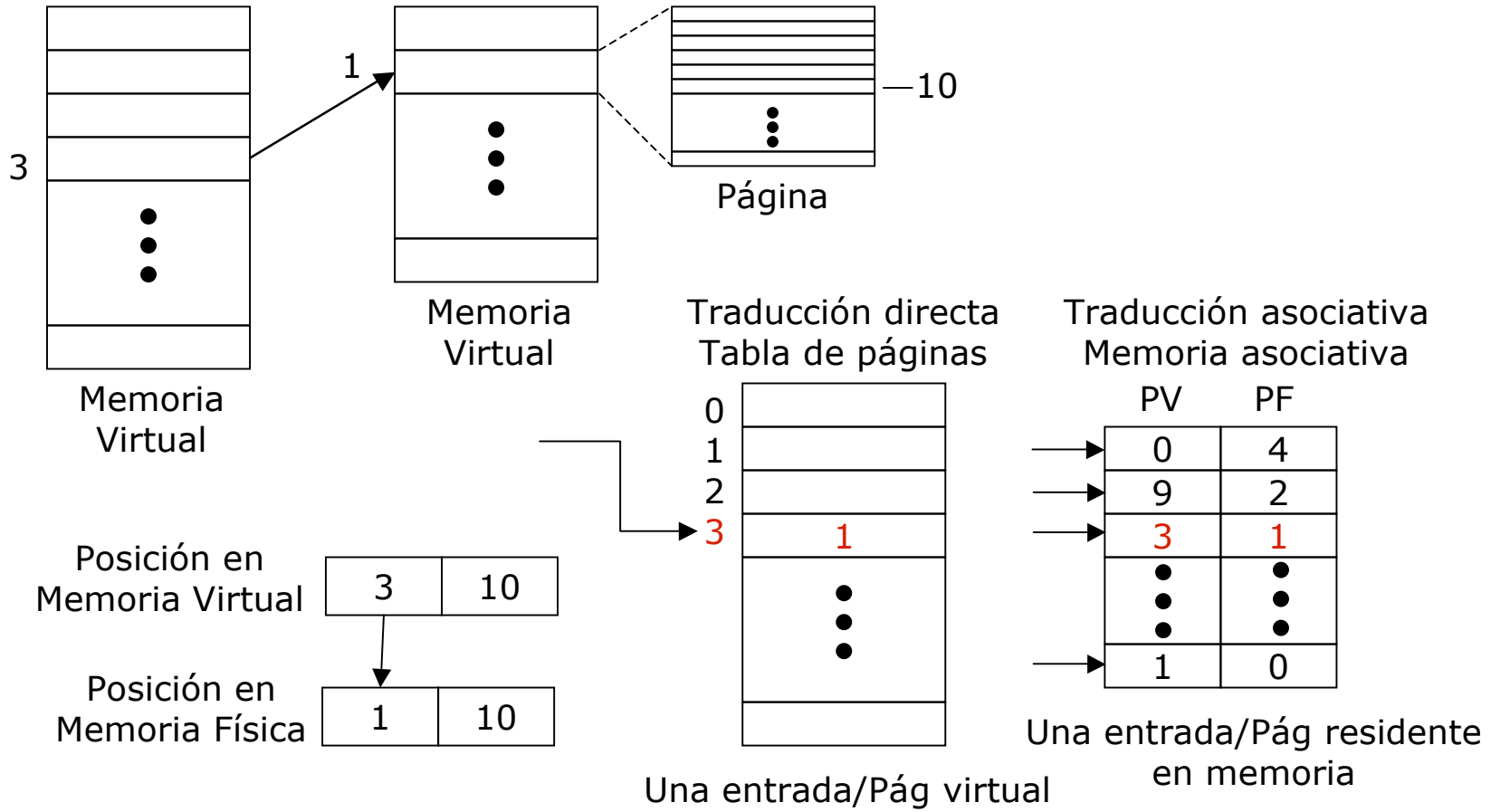
Esquemas básicos de implementación de la traducción (VII)

Processor	Address space: Real	Address space: Virtual	Address translation scheme	Mapping levels	Associative cache
Intel 80286	16M	1G	Segmentation	1	Four segment descr. registers
Intel 432	16M	1T	Segmentation	2	Assoc. cache TLB
Intel 80386	4G	64T	Paging and segmentation	2	Assoc. cache TLB
MC68000	16M	4G	User selectable	1	32 Content-addr. registers
MC68010 + 58451 MMU	16M	4G	User selectable	1	32 Content-addr. registers
MC68020 + 68851 MMU	4G	?	User selectable	1	32 Content-addr. registers
Z8001 + Z8010 MMU	16M	4G	Segmentation	1	64 Segm. content-addr. registers
Z8003 + Z8015 MMU	8M	4G	Paging (page size = 2K)	1	64 Page descr. registers
Z800	16M	4G	Paging (page size = 4K)	1	16 Addr.-accessible registers
Z80,000	16M	4G	Paging (page size = 1K)	3	Assoc. cache TLB
NS16032 NS32032 + 16082 MMU	16M	4G	Paging (page size = 512K)	2	32 Content-addr. caches
NCR/32 + NCR32101	16M	4G	Paging (page size = 1K)	1	16 Associative memories
WE32100 + WE32101 MMU	4G	?	Paging and segmentation	1	64-Entry page descr. table 32-Entry segm. descr. table

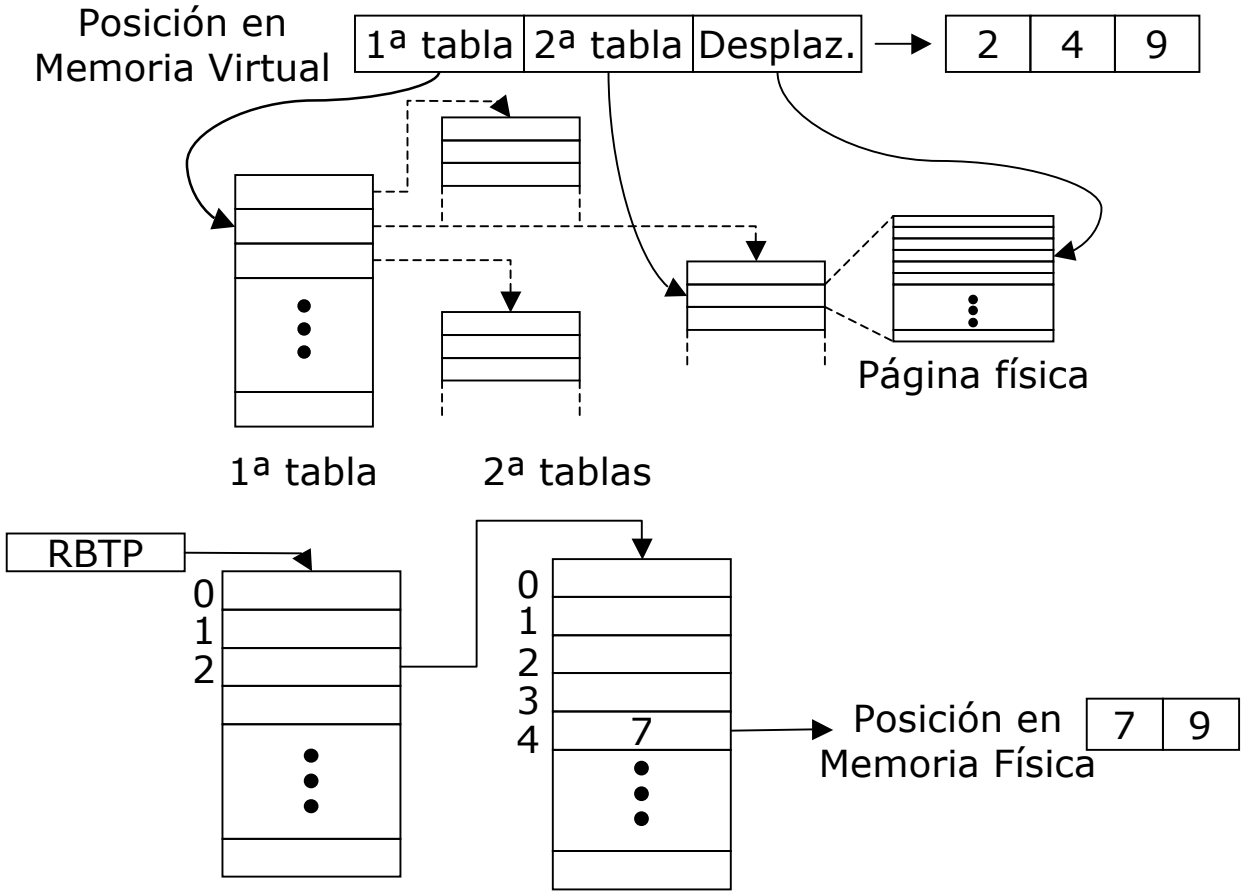
Buffer de traducción rápida y funcionamiento de la caché



Ejemplos de aplicación de las técnicas de traducción



Ejemplos de aplicación de las técnicas de traducción



Tamaño de las páginas (I)

- Compromiso entre los factores:
  - Eficiencia del dispositivo de memoria secundaria: el más influyente.
  - Espacio ocupado por la tablas de páginas.
  - Tamaño medio de las estructuras lógicas de los programas.
  - Otros (menos importantes).
- Influencia del tamaño de página:
  - Fragmentación interna:
    - Espacio de memoria física desperdiciado al final de la última página física.
    - Grave si el tamaño de página es grande.
  - Fragmentación de la tabla:
    - Espacio de memoria física desperdiciado en almacenar las tablas de páginas.
    - Grave si el tamaño de página es pequeño.
- Estudios indican que la mayoría de los bloques lógicos ocupan menos de 1000 palabras.

Tamaño de las páginas (II)

- El tamaño de página práctico se sitúa entre 128 y 1024 palabras. (normalmente entre 512 y 8192 bytes)
- Ubicación (totalmente asociativo).

Tamaños de páginas para diversos sistemas.

Sistema	Tamaño de página (palabras)	
IBM 360/67	1024	(32 bits)
IBM 370/168	512 o 1024	(32 bits)
MULTICS	1024	(36 bits)
RCA Spectra 70/46	1024	(32 bits)
Xerox Sigma 7	512	(32 bits)
DEC PDP-10,20	512	(36 bits)
CDC Star-100	4192	(8 bits)
DEC VAX-11/780	128	(32 bits)
DG Eclipse	2048	(32 bits)
MMU NS 16082	512	(8 bits)
MMU NCR 32101	1024	(8 bits)
MMU Z8015	2048	(8 bits)
Z800	4096	(8 bits)
Z80,000	1024	(8 bits)

Algoritmo de carga (I)

- ¿Cuándo debe transferirse una página virtual desde la memoria secundaria a la primaria?
- Soluciones:
  - Carga por demanda:
    - La página virtual se carga en la memoria principal cuando es referenciada y produce un fallo de página.
    - Método simple y no sobrecarga el canal de paginación.
    - Usual en los microprocesadores de 16 y 32 bits.
  - Precarga:
    - Como consecuencia de un fallo de página, se carga en la memoria principal la página fallada junto con otra/s adicional/es.
    - Método predictivo: se precargan aquellas páginas que producirían fallos en un futuro próximo.
    - Número de páginas precargadas:
      - Según la organización de la información en la memoria secundaria.
      - Según el tipo de interface entre la memoria principal y la secundaria.
      - Según el tamaño de la PMA y si es de tamaño constante. (Influye en la sobrecarga del canal de paginación, reemplazo).
    - Normalmente, se precarga una sola página.

Algoritmo de carga (II)

- Ejemplo de un algoritmo de predicción satisfactorio:
  1) A p_i se le asocia PRED[i] (inicialmente, PRED[i]=p_i+1).
  2) LAST=k si p_k produjo el último fallo de página (inicialmente 0).
  3) Si pi produce un fallo de página, entonces:
     - Se carga p_i por demanda.
     - Se precarga PRED[i] si no es residente.
     - Si no hubo precarga en el fallo anterior, o la página precargada no fue referenciada, entonces PRED[LAST]=p_i.
  - Algoritmo de predicción operativo cuando el tamaño de página es pequeño (muchas páginas).

- Ejemplo de sistema con un algoritmo de precarga:
  - Sistema operativo VAX/VMS de DEC VAX-11/780.
  - El conjunto de páginas precargadas puede elegirse de forma manual o automática.

- Seguimiento dinámico del comportamiento del programa.

Algoritmo de reemplazo

- Controlado por el Sistema Operativo
  - Aproximaciones teóricas: FIFO, LRU, …
  - Aproximaciones prácticas
    - LFU (Least Frequentlu Used)
    - NUR (Not Used Recently)
    - etc.

Sistemas segmentados

Inconveniente fundamental de los sistemas paginados

- Las páginas son de tamaño fijo y arbitrario: no tienen relación con la estructura lógica del programa.
- Puede haber datos en una página no relacionados con los demás. Ineficiencia por el principio de localidad.

Sistema segmentado

- Sistema de memoria virtual que considera bloques de tamaño desigual, definidos en función de la estructura lógica del código y datos del programa. (procedimientos, funciones, arrays, matrices).
- Cada bloque se denomina segmento.
- La mayoría de la características (conceptos) de los sistemas segmentados son similares a las de los sistemas paginados.

Traducción de direcciones (I)

- Direcciones virtuales:

  (s_i,d_j) s_i número de segmento; d_j desplazamiento

- Tabla de segmentos:
  - Contiene información para traducir el número de segmento a una dirección física de comienzo del segmento.
  - Sus entradas contienen información similar a la contenida en las entradas de las tablas de páginas, junto con la longitud del segmento.

- Fallo de segmento:
  - Referencia a un segmento no residente en la memoria principal.

- Registro base de la tabla de segmentos.

- La tabla de segmentos es un segmento más (para el S.O.).

Traducción de direcciones (II)

- Tipos de traducción:
  - Los mismos que en los sistemas paginados (directa, asociativa o híbrida).
  - Diferencia: el desplazamiento se suma (no se concatena).
  - Ejemplo: directa.

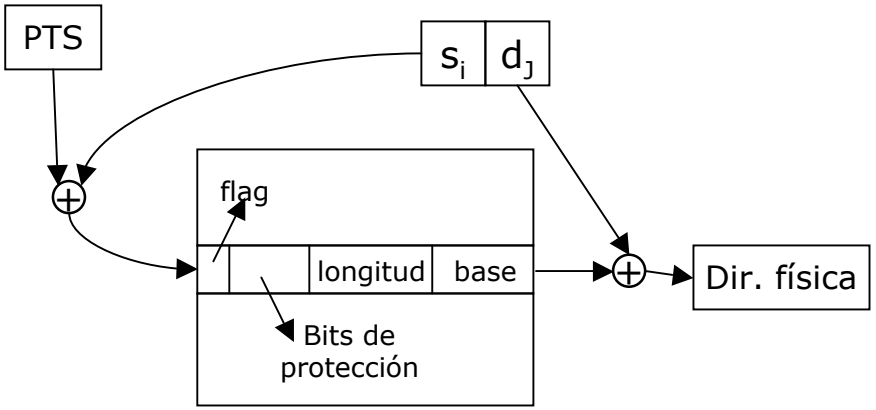


Tabla de Segmentos

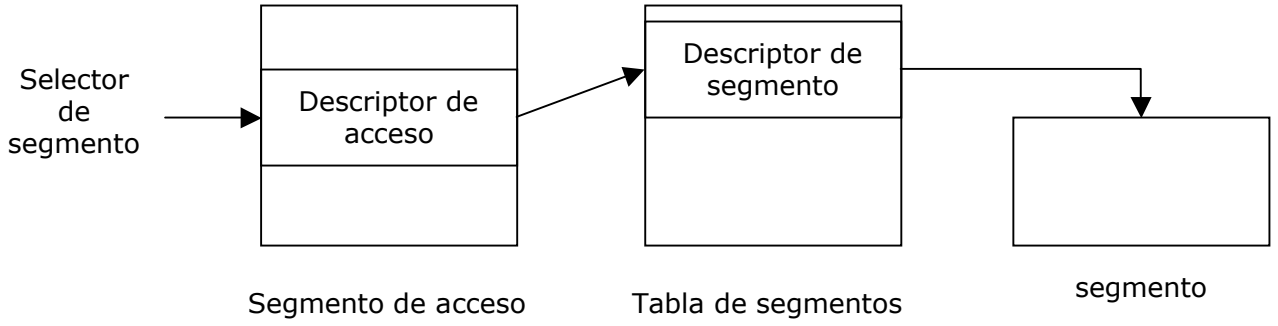
-PTS (Puntero a la Tabla de Segmentos): contiene la dirección física de comienzo de la tabla de segmentos.

- Se chequea d_j contra la longitud para comprobar que la dirección virtual se encuentra dentro del segmento referenciado.

- Mayor facilidad de protección (entidad lógica).

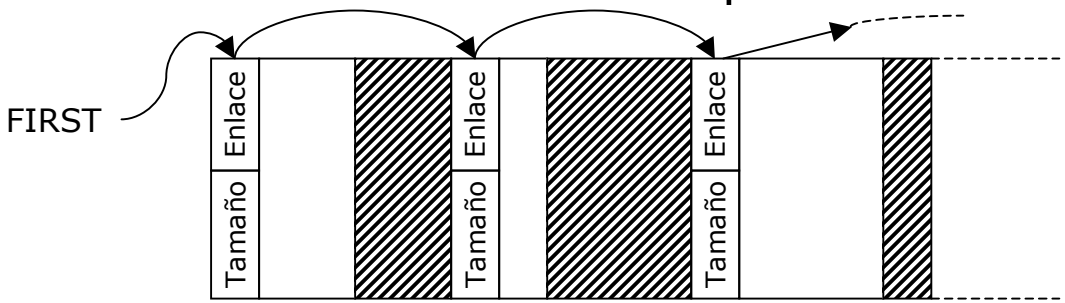
Traducción de direcciones (III)

- Ejemplo: Intel 432 con traducción en dos niveles
  - Se almacenan los derechos de acceso independientemente de la tabla de segmentos
  - Permite que varios módulos compartan segmentos con diferentes derechos de acceso



Fallos de segmento

- Es necesario determinar si hay espacio de memoria principal para ubicar el segmento virtual fallado (este problema no existe en los sistemas paginados).
- Solución:
  - Estructurar la memoria principal. Se definen dos listas encadenadas similares:
    - Lista de segmentos reservados.
    - Lista de segmentos libres (LAVS), normalmente en orden ascendente de su posición.

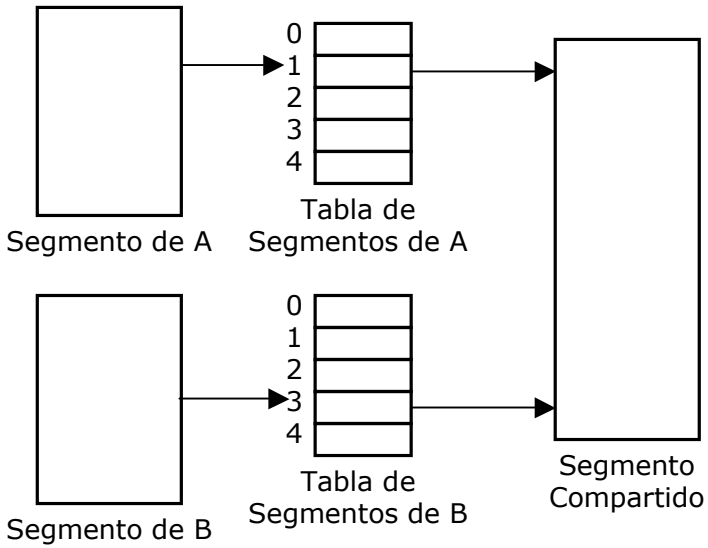


Espacio virtual segmentado versus paginado

- Un sistema segmentado es un espacio bidimensional: los segmentos son bloques de direcciones virtuales contiguas, pero los segmentos no son contiguos entre si.
- Se produce una excepción al intentar acceder a una posición fuera del segmento.

Compartición de los segmentos

- Posibilidades de compartición de segmentos:
  - Un segmento compartido es referenciado por dos procesos mediante el mismo número.
  - Un segmento compartido es referenciado por dos procesos mediante números diferentes:
    - Esquema más versátil.
    - Implementación: Permitiendo una tabla de segmentos privada a cada proceso.
    - Puede necesitarse que el segmento compartido genere números de segmento dependientes del contexto.



Algoritmos de carga y ubicación

- Proceso de carga: similar a los sistemas paginados (precarga), (¿cuándo?)
- Proceso de ubicación: más complejo que en los sistemas paginados. (¿dónde?).

Algoritmos de ubicación

- Todos basados en la lista de LAVS:
  - First-Fit.
  - Best-Fit.
  - Worst-Fit.
  - Binary-Buddy.

Algoritmo First-Fit (I)

- Hace uso de la lista LAVS.
- El segmento solicitado no residente se coloca en el primer segmento libre de la memoria principal donde quepa. (búsqueda en orden ascendente de direcciones físicas).
- Si el procesador solicita el segmento S no residente de tamaño L, se realizan las siguientes acciones.
  1) Sea Q←FIRST (puntero inicial de LAVS).
  2) Si TAMAÑO(Q)>L, S se coloca a partir de la dirección física Q+TAMAÑO(Q)-L. Se actualiza TAMAÑO(Q) a TAMAÑO(Q)-L.
  3) Si TAMAÑO(Q)=L, S se coloca a partir de la dirección física Q. Se borra el segmento libre de LAVS.
  4) Si TAMAÑO(Q)<L, se asigna Q←SIGUIENTE(Q) y se va a 2.

Algoritmo First-Fit (II)

- Para evitar la formación de segmentos libres demasiado pequeños, se cambian:
  2) Si TAMAÑO(Q)>L+C …
  3) Si L≤TAMAÑO(Q)≤L+C …
- Ejemplo:
  - Un sistema tiene la siguiente lista de segmentos libres:
    LAVS=(0,128)→(384,1024)→(1920,512).
  - Se solicitan los segmentos A (tamaño 64) y B (tamaño 256).
  - Evolución de la lista LAVS:
    LAVS=(0,64)→(384,1024)→(1920,512); A comienza en la dirección 64.
    LAVS=(0,64)→(384,768)→(1920,512); B comienza en la dirección 1152.

Algoritmo Best-Fit

- Hace uso de la lista LAVS.
- El segmento solicitado no residente se coloca en el segmento libre de la memoria principal más pequeño donde quepa.
- El procedimiento de ubicación es el mismo que el del algoritmo First-Fit, pero previamente, se ordena la lista LAVS según el orden creciente de los tamaños de los segmentos libres.
- Necesidad de reordenación.
- Ejemplo:
  - LAVS = (0,128)→(384,1024)→(1920,512).
  - Se solicitan los segmentos A (tamaño 64) y B (tamaño 256).
  - Se reordena la lista LAVS:
    LAVS = (0,128)→(1920,512)→(384,1024).
  - Evolución de la lista LAVS:
    LAVS=(0,64)→(1920,512)→(384,1024); A comienza en la dirección 64.
    LAVS=(0,64)→(1920,256)→(384,1024); B comienza en la dirección 2176.

Algoritmo Worst-Fit

- Hace uso de la lista LAVS.
- El segmento solicitado no residente se coloca en el mayor de los segmentos libres de la memoria principal.
- El procedimiento de ubicación es similar al del algoritmo Best-Fit, salvo que la lista LAVS se ordena en orden decreciente de los tamaños de los segmentos libres.
- Ejemplo:
  - LAVS = (0,128)→(384,1024)→(1920,512).
  - Se solicitan los segmentos A (tamaño 64) y B (tamaño 256).
  - Se reordena la lista LAVS:
    LAVS = (384,1024)→(1920,512)→(0,128)
  - Evolución de la lista LAVS:
    LAVS=(384,960)→(1920,512)→(0,128); A comienza en la dirección 1344.
    LAVS=(384,704)→(1920,512)→(0,128); B comienza en la dirección 1088.

Algoritmo Binary-Buddy (I)

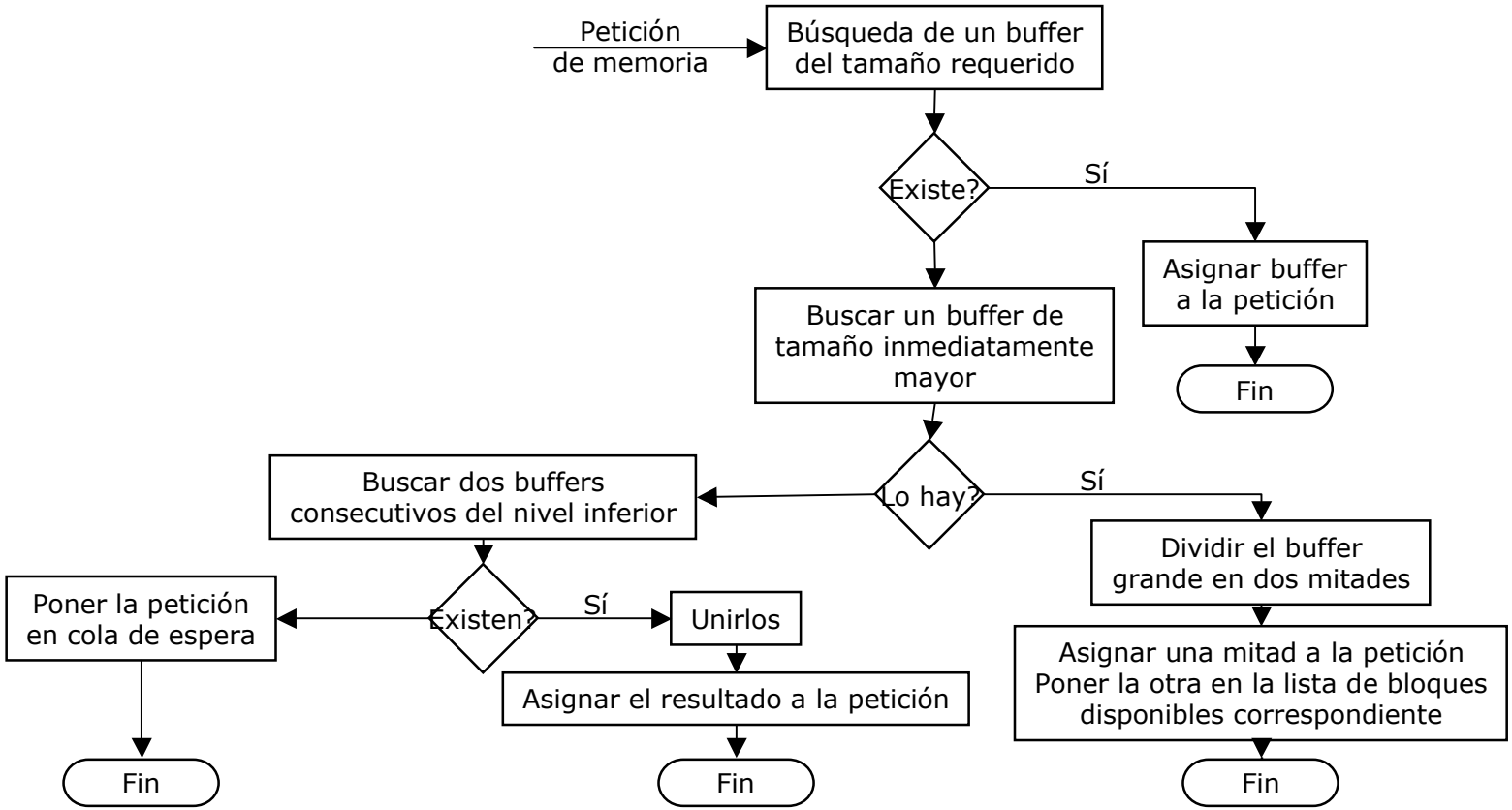
- Tamaño de segmento y LAVS potencia entera de 2.
- Hace uso de la lista LAVS.
- Procedimiento de ubicación:
  - Se divide la lista LAVS en *n* listas (2^n es el tamaño máximo de los segmentos).
  - La i-ésima lista encadena los segmentos libres de tamaño 2^i (buddies).
  - Las n listas pueden reestructurarse de dos formas:

    a) Un segmento libre de la lista (i+1)-ésima se divide en dos segmentos iguales y se encadenan a la lista i-ésima.

    b) Proceso inverso al a).
  - Desarrollo de un algoritmo de ubicación basado en las n listas anteriores y sus dos mecanismos de reestructuración.

Algoritmo Binary-Buddy (II)

- Ejemplo de algoritmo de ubicación:



Algoritmo Binary-Buddy (III)

- Ejemplo:
  - LAVS=(0,128)→(384,1024)→(1920,512).
  - Se solicitan los segmentos A (tamaño 64) y B (tamaño 256).
  - Se reordena la lista LAVS en 3 listas:
    - Lista-7 = (0,128)
    - Lista-9 = (1920,512)
    - Lista-10=(384,1024)
  - Evolución de las listas:
    - Se divide la lista-7 en dos *buddies*:
      - Lista-6=(0,64)→(64,64); A comienza en la dirección 0.
    - Se divide la lista-9 en dos *buddies*:
      - Lista-8=(1920,256)→(2176,256); B comienza en la dirección 1920.
  - Lista finales:
    - Lista-6=(64,4)
    - Lista-8=(2176,256)
    - Lista-10=(384,1024)

UNIVERSIDADE DA CORUÑA

GRUPO DE ARQUITECTURA DE COMPUTADORES

Comparación de los algoritmos de ubicación

- Algoritmo Best-Fit:
  - Minimiza el tamaño del segmento libre sobrante tras la ubicación.
- Algoritmo Worst-Fit:
  - Basado en la idea de que, tras la ubicación, el segmento libre sobrante es suficientemente grande como para ser útil.
- Algoritmos más eficientes:
  - First-Fit y Binary Buddy
- Inconveniente asociado a la ubicación:
  - Fragmentación externa:
    - Pérdida de espacio de memoria física debido a la producción de segmentos libres sobrantes de tamaño muy pequeño (inútiles).
  - Puede llegar a ser un gran problema.

Compactación o reemplazo (I)

- Cuando se solicita un segmento no residente, pueden darse dos situaciones:
  - Se encuentra un segmento libre de tamaño suficiente: ubicación.
  - No se encuentra tal segmento libre:
    - Soluciones:
      - Compactación.
      - Reemplazo.
- Compactación (o garbage-collection):
  - Se redisponen los segmentos libres de forma contigua en una zona de la memoria física (se crea un único segmento libre de gran tamaño).
  - Problemas:
    - Hay un gran consumo de tiempo.
    - Tiene asociada una actualización de las tablas de segmentos (o listas de segmentos residentes, si la hubiere).

Compactación o reemplazo (II)

- Reemplazo:
  - Similar al reemplazo de los sistemas paginados.
  - Única diferencia:
    - Debe tenerse en cuenta el tamaño del segmento solicitado.
  - Problemas:
    - Uso poco eficiente de la memoria física.
    - El segmento reemplazado puede necesitarse en un futuro próximo.
- La elección entre compactación y reemplazo no es simple:
  - Depende de cada situación particular.
- Tamaño de los segmentos:
  - Los segmentos pueden ser desmesuradamente grandes.
  - Soluciones:
    - Limitar el tamaño de los segmentos (1 Kbyte para B5500; 64 Kbytes para i80286 y Z8000).
    - Paginar los segmentos: segmentación con paginación.

Sistemas segmentados con paginación (I)

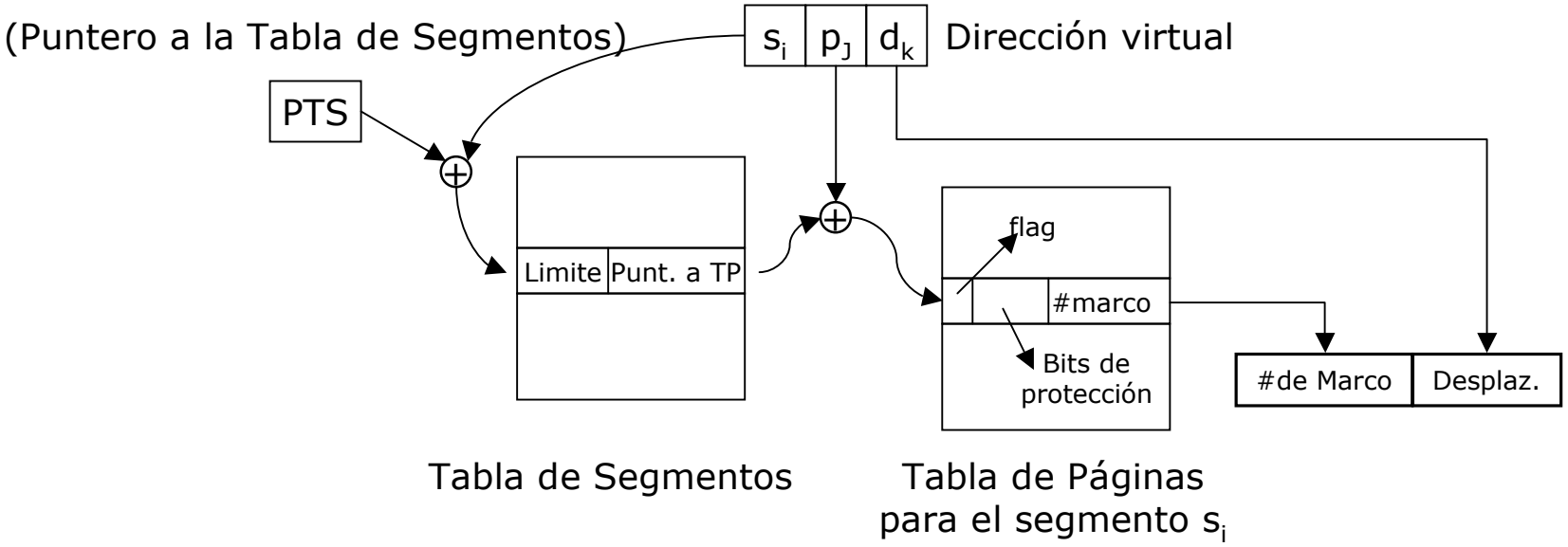
- Hoy en día pocos sistemas utilizan segmentación pura debido a los problemas para efectuar los reemplazos
- Enfoque híbrido: segmentos paginados
  - Simplifica el reemplazo
    - No se precisa que la memoria de un segmento sea contigua
    - No es preciso cargar los segmentos enteros en memoria

Sistemas segmentados con paginación (II)

- Combinación de segmentación y paginación: (los segmentos se dividen en un número entero de páginas)
  - Segmentación lineal (domina paginación).
  - Segmentación nominal (domina segmentación).
- Dirección virtual: (s_i, p_j, d_k) espacio 3-dimensional.
  - s_i es el número de segmento virtual.
  - p_j es el número de página virtual.
  - d_k es el desplazamiento dentro de la página.

Sistemas segmentados con paginación (III)

- - PTS apunta a la tabla de segmentos del proceso en curso.
  - Limite es el campo que indica el tamaño de cada segmento.
  - Los bits de protección están a nivel de páginas (no necesario en segmentación nominal).
  - Flag indica la residencia en PMA de cada página.



Sistemas segmentados con paginación (IV)

- DOS traducciones: elevado coste temporal.
  - Soluciones:
    - Uso de TLB
    - Uso de registros rápidos
- El fallo de página se resuelve de la misma forma que en los sistemas puramente paginados.
- PROBLEMA: fallo en el acceso a la tabla de páginas ⇨ lentifica ⇨ 2 fallos.

Paginación vs Segmentación

	Página	Segmento
Palabras por dirección	Una	Dos (segmento y desplazamiento)
¿Visible al programador?	Invisible a la aplicación del programador	Puede ser visible a la aplicación del programador
Reemplazo de un bloque	Trivial (todos los bloques tienen el mismo tamaño)	Difícil (debe encontrar una parte no utilizada de memoria principal de tamaño variable y contigua)
Uso ineficiente de memoria	Fragmentación interna (porción inutilizada de página)	Fragmentación externa (partes no usadas de memoria principal)
Tráfico de disco eficiente	Sí (ajusta tamaño de página para equilibrar tiempo de acceso y tiempo de transferencia)	No siempre (pequeños segmentos pueden transferir sólo unos pocos bytes)

Parámetros comparativos de los niveles de la jerarquía de memoria

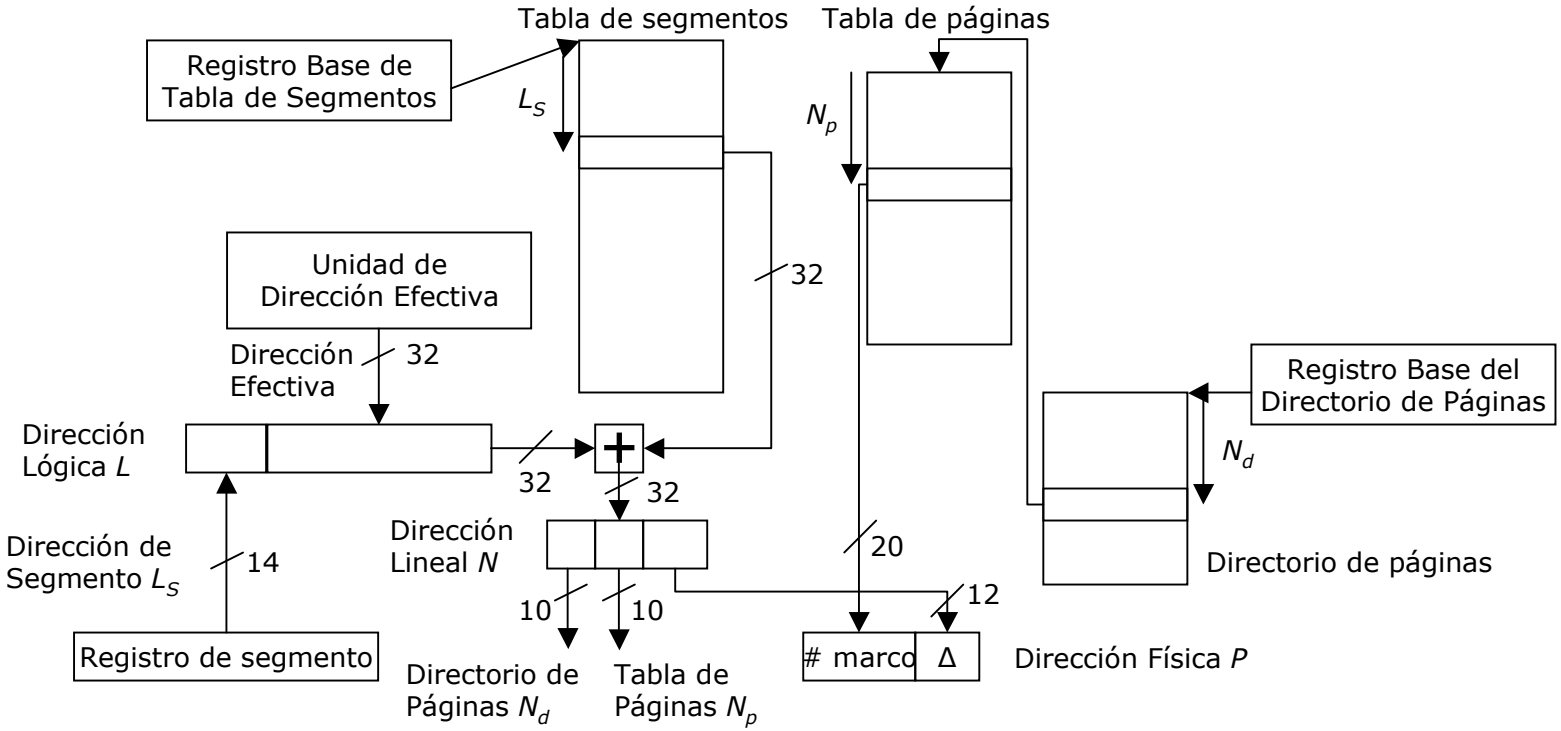
	TLB	Cache 1er nivel	Cache 2º nivel	Memoria virtual
Tamaño de bloque	4-8 bytes (1 PTE)	4-32 bytes	32-256 bytes	512-16,384 bytes
Tiempo de acierto	1 ciclo de reloj	1-2 ciclos de reloj	6-15 ciclos de reloj	10-100 ciclos de reloj
Penalización de fallo	10-30 ciclos de reloj	8-66 ciclos de reloj	30-200 ciclos de reloj	700,000-6,000,000
Tasa de fallos (local)	0.1-2%	0.5-20%	15-30%	0.00001-0.001%
Tamaño	32-8192 bytes (8-1024 PTEs)	1-128 KB	256KB-16MB	16-8192 MB
Nivel inferior	Cache de 1er nivel	Cache de 2º nivel	Memoria Principal (DRAM)	Discos
Q1: ubicación de bloques	Asociativa completamente o por conjuntos	Correspondencia directa o asociativa por conjuntos	Correspondencia directa o asociativa por conjuntos	Totalmente asociativa
Q2: identificación de bloques	Etiqueta/bloque	Etiqueta/bloque	Etiqueta/bloque	Tabla
Q3: reemplazo de bloques	Pseudo-LRU	Pseudo-LRU	Pseudo-LRU	≈ LRU
Q4: estrategia de escritura	Postescritura	Escritura directa o postescritura	Postescritura	Postescritura

Ejemplo: Intel 80386 y 80486 (I)

- Incluyen hardware para la gestión de memoria tanto PAGINADA como SEGMENTADA.
- MMU interna.
- Espacio físico: 4 Gbytes (32 bits)
- Espacio virtual: 64 Tbytes (46 bits)
- Segmentos desde 1 byte a 4 Gbytes.
- Páginas de 4 Kbytes.
- El programador puede activar cualquiera de estas posibilidades:
  - Direccionamiento directo (no usa Memoria Virtual).
  - Paginación pura. (2 niveles)
  - Segmentación pura.
  - Segmenteación con paginación.

Ejemplo: Intel 80386 y 80486 (II)

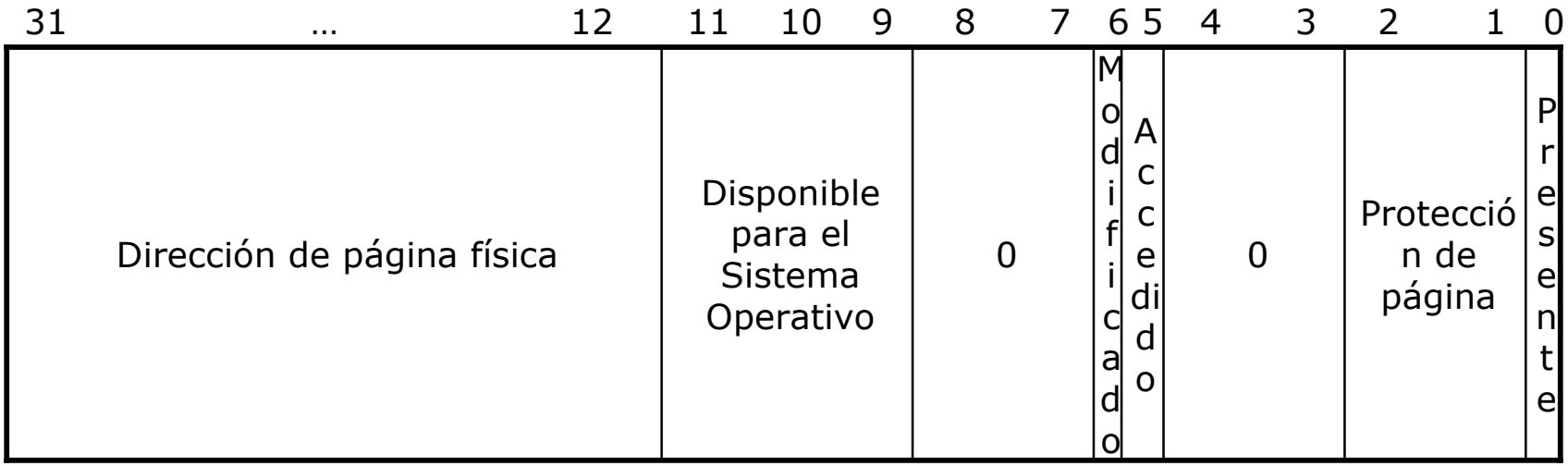
- Traducción de direcciones:



Ejemplo: Intel 80386 y 80486 (III)

- Traducción de direcciones:
  - Sin la unidad de segmentación: L=N
  - Sin la unidad de paginación (en dos niveles): N=P
  - Ambas unidades poseen una pequeña memoria cache.
  - Un proceso activo tiene asociados varios segmentos:
    - Código ejecutable
    - Stack
    - Conjuntos de datos
  - Existen 6 registros base de estos segmentos L_s en la figura.

Formato de la tabla de páginas



- Bit 0: Bit de presencia en memoria física.
- Bits 1-2: Bits de protección de lectura, escritura o supervisor.
- Bit 5: Bit de accedido o de referencia.
- Bit 6: Bit de modificación.
- Bits 9-11: Privados del sistema operativo.
- Bits 3,4,7,8: Reservados para posibles expansiones.
- Bits 3-4: Control de memoria cache. (sólo en 80486)
- Bits 12-31: Dirección base de la página física.