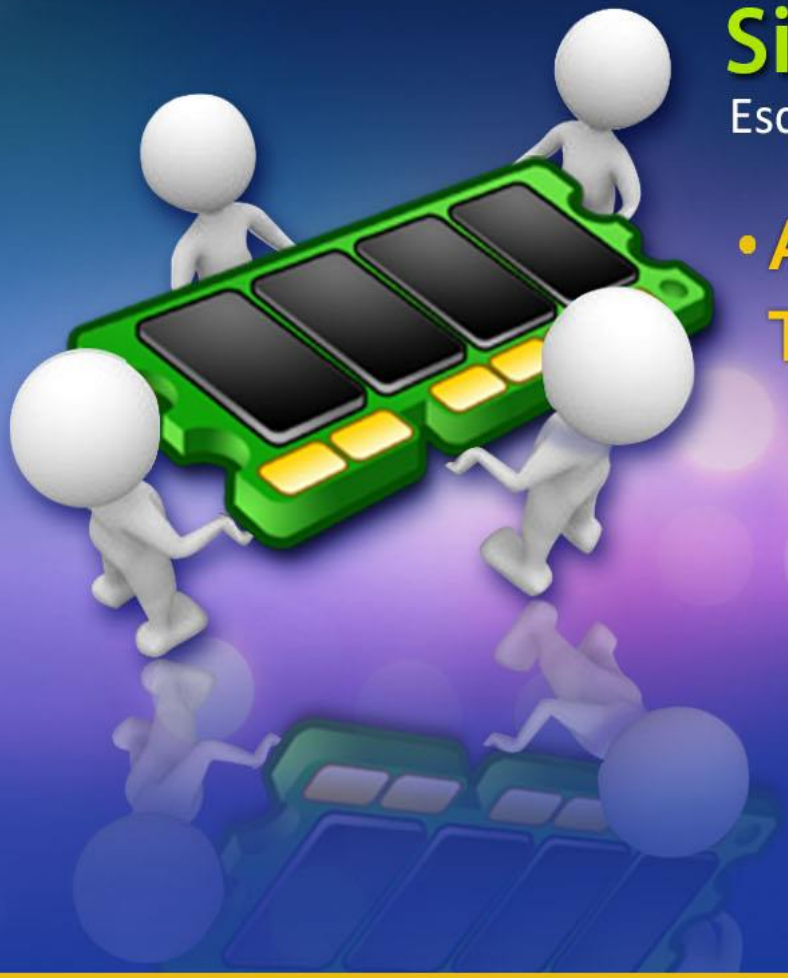


Sistemas Operativos

Escuela de Ingeniería Civil Informática

- Administración de Memoria
  Técnica de Paginamiento

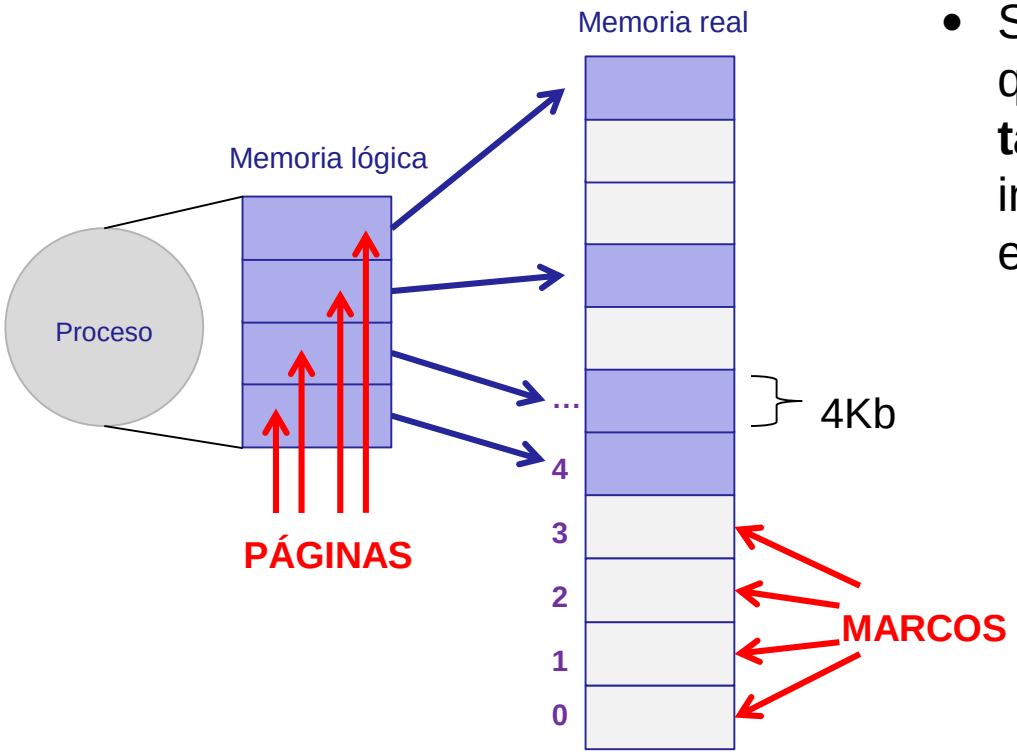
UNIVERSIDAD DEL BÍO-BÍO

EL PAGINAMIENTO

- Es una manera de relocalización dinámica de la memoria.
- Esta técnica consiste en:
  - Dividir la memoria virtual (lógica) en **páginas** de tamaño fijo.
  - Dividir la memoria real (física) en **marcos de página** de tamaño fijo.
  - El tamaño debe ser una potencia de 2 (por ejemplo 4Kb, 8Kb…).
  - Una página se otorga completa a un proceso.
  - Las páginas contiguas en el espacio de direcciones virtuales no necesitan estar contiguas en el espacio de direcciones reales.
  - Las otras páginas pueden estar en disco o no asignadas.

EL PAGINAMIENTO

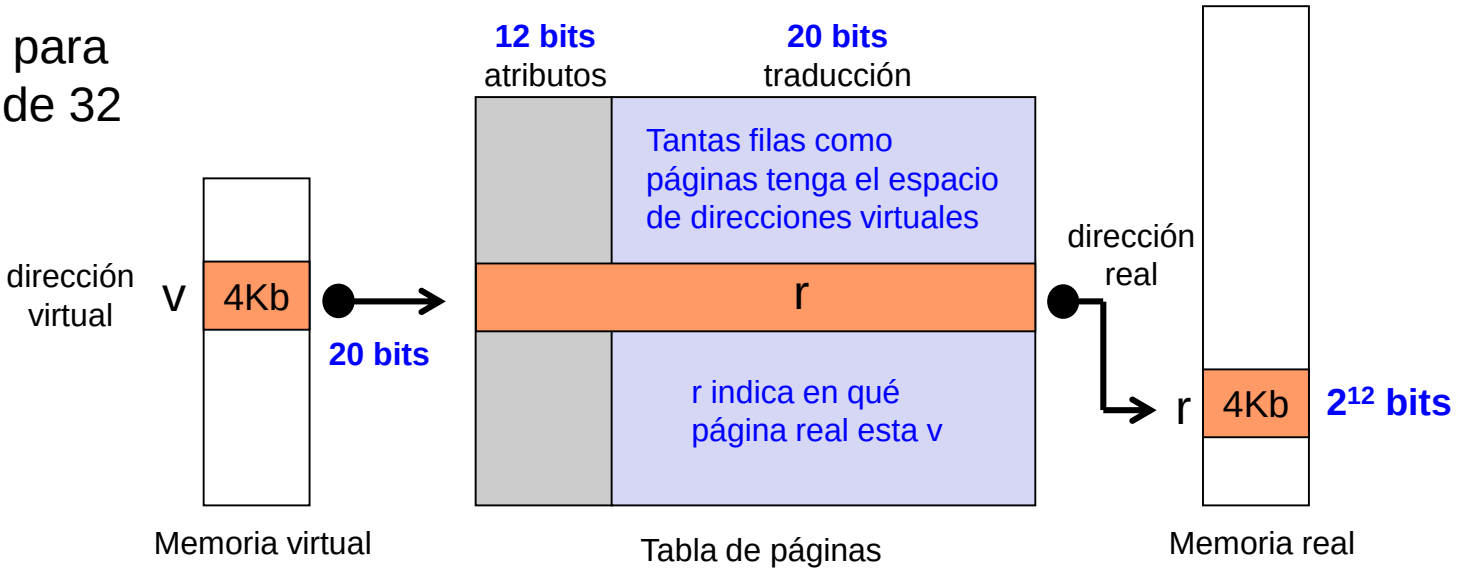
- El siguiente diagrama presenta esta situación:



- Se facilita la reubicación ya que basta con asignar en una **tabla de correlación** que indica en qué marco se encuentra cada página.

LA TABLA DE PÁGINAS

- Cada proceso posee una tabla de páginas que indica como traducir las direcciones virtuales a direcciones reales.
- En la fila v se indica la posición r en la memoria real de la página virtual nº v.
- Ejemplo para un S.O. de 32 bits.



BITS DE ATRIBUTOS DE LA TABLA DE PÁGINAS

- Los 4 bits de atributos más importantes de la tabla de páginas son los siguientes:
  - **bit V (acceso o validez):** indica si la página se puede leer (o escribir). Para poder leer la página basta con que v sea igual a 1. Si está en 0, cualquier tipo de acceso a esa página va a producir una interrupción segmentation fault que se interpreta de manera distinta dependiendo del segmento: pila, datos, etc.
  - **bit W (escritura):** indica si la página se puede escribir (solo si v=1 y w=1). Si alguno de los dos bits, v o w es 0 → no se puede escribir la página.
  - **bit R (referencia):** el hardware coloca en 1 este bit cada vez que se accesa la página. Este bit es imprescindible para implementar la técnica de paginamiento por demanda. El S.O. cada cierto tiempo coloca este bit en 0.
  - **bit D (dirty):** se coloca en 1 si la página es modificada (escrita). Si la página no ha sido modificada no es necesario actualizarla en disco.

CONSIDERACIONES DEL PAGINAMIENTO

- Con paginación la memoria real de un proceso no es contigua.
- El tamaño de página lo define el hardware:
  - En la arquitectura x86 de 32 bits es de un tamaño de 4 Kbytes.
  - En la arquitectura SPARC es de un tamaño de 8 Kbytes.
- La paginación no tiene fragmentación externa, pero sí interna (dentro de cada página).
  - A mayor tamaño de página mayor fragmentación interna.
  - A menor tamaño de página mayor gasto adicional en tablas de páginas.
- La paginación permite compartir memoria entre distintos procesos:
  - basta que las entradas correspondientes de sus tablas de páginas apunten a los mismos marcos de memoria física.

CONSIDERACIONES DEL PAGINAMIENTO

- Con paginación la memoria está protegida:
  - un proceso solo accede a donde indica su tabla de páginas
  - dicha tabla de páginas solo puede ser modificada por el S.O.

- Requiere el soporte de hardware MMU (p.e un intel 286 no tenía paginación, un i386 sí).

- Además de la dirección del marco de memoria física, en cada entrada de la tabla de páginas hay más bits de información: página de solo lectura, modificada, nivel de privilegio . . .

IMPLEMENTACIÓN DE LA TABLA DE PÁGINAS

- Hay dos maneras de implementar la tabla de páginas:

1. **Registros dedicados:** El microprocesador tendría que ofrecer algunos registros donde almacenar la tabla de páginas del proceso en ejecución.

   - En el cambio de contexto se salva (guarda) la tabla de páginas del proceso que abandona la CPU y se carga la del que entra en CPU:
     - Muy rápido al ejecutar el proceso
     - Muy costoso (hacen falta muchos registros: impracticable)
     - Cambio de contexto lento (salvar los registros).

IMPLEMENTACIÓN DE LA TABLA DE PÁGINAS

2. **En memoria:** La tabla de páginas de un proceso se almacena en memoria principal.

- El microprocesador tiene un registro que indica en que dirección está la tabla de páginas del proceso que se ejecuta en la CPU.

- En el cambio de contexto solo hay que cambiar el puntero a la tabla
  - Lento: Cada vez que el proceso accede a memoria son necesarios dos accesos:
    - uno para acceder a la tabla de páginas para determinar la entrada a utilizar (fila).
    - otro para acceder a la dirección almacenada en la fila, es decir, al dato.

IMPLEMENTACIÓN DE LA TABLA DE PÁGINAS

3. **Implementación actual**

- En la actualidad, la tabla de páginas (al igual que la de segmentos) está en memoria RAM (implementarlas en registros dedicados sería muy costoso).

- Un registro del procesador apunta a la tabla de páginas.

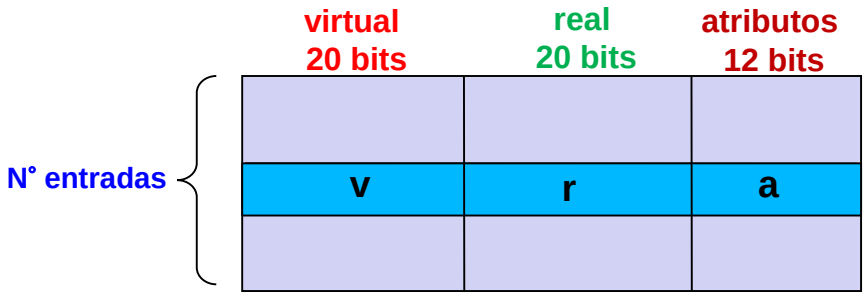
- Dado que se necesitan 2 accesos a memoria, y esto es lento, se usan unas pequeñas memorias caches con las entradas de la tabla de página más recientemente usadas. Esta caché se denomina **TLB** (**Translation Lookaside Buffers**)

LA TLB

- Es una memoria caché administrada por la MMU.
- Contiene traducciones de direcciones (lógicas a físicas) accedidas frecuentemente.
- El acceso es en paralelo, lo que aumenta el rendimiento de la MMU.
- Si no existe la entrada buscada, se debe revisar la tabla de páginas (ubicada en memoria) y tardará varios ciclos más de CPU.
- En algunos sistemas hay dos niveles de TLB (un core i7 tiene un TLB1 de 64 entradas y un TLB2 de 512 entradas)

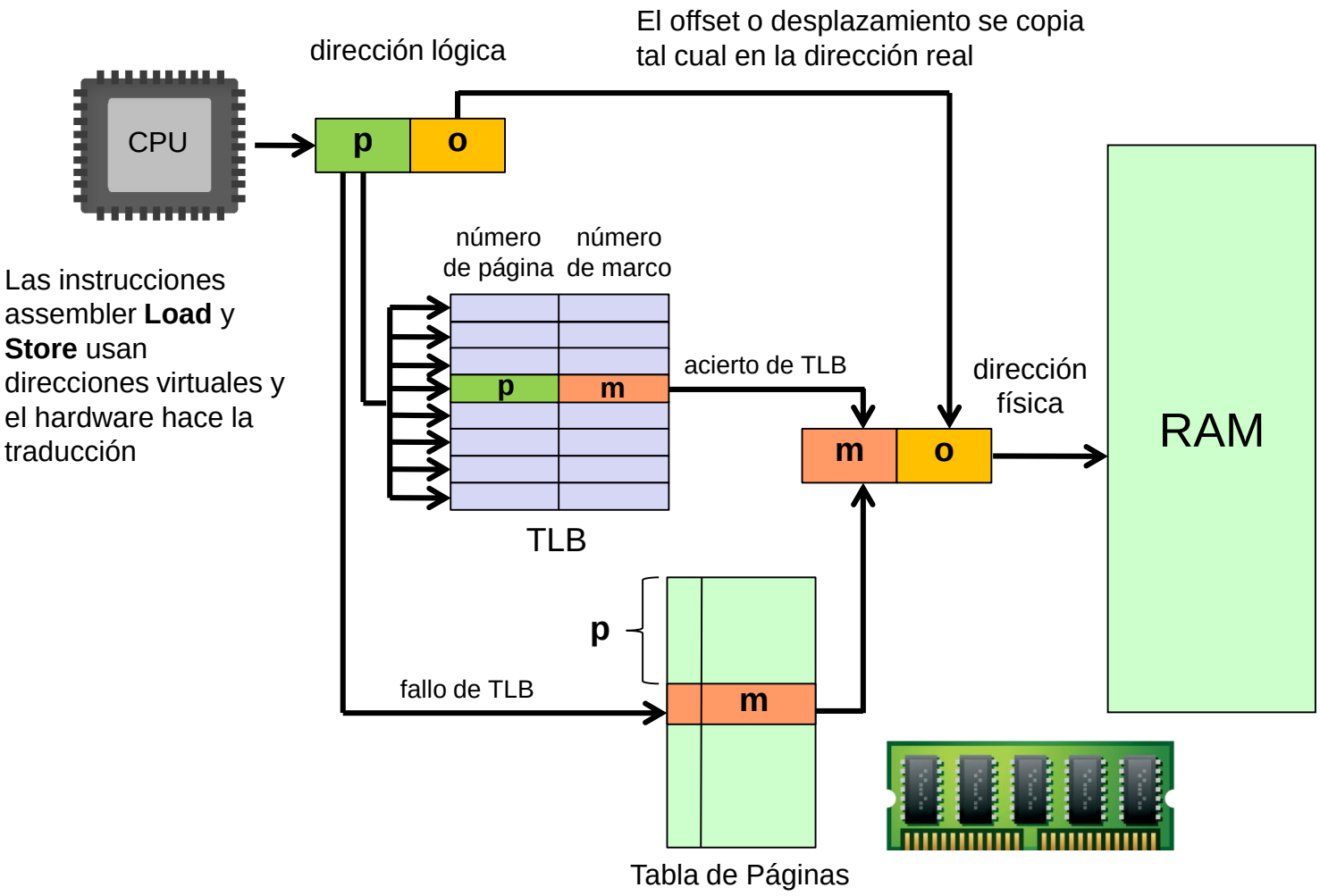
TRADUCCIÓN DE DIRECCIONES

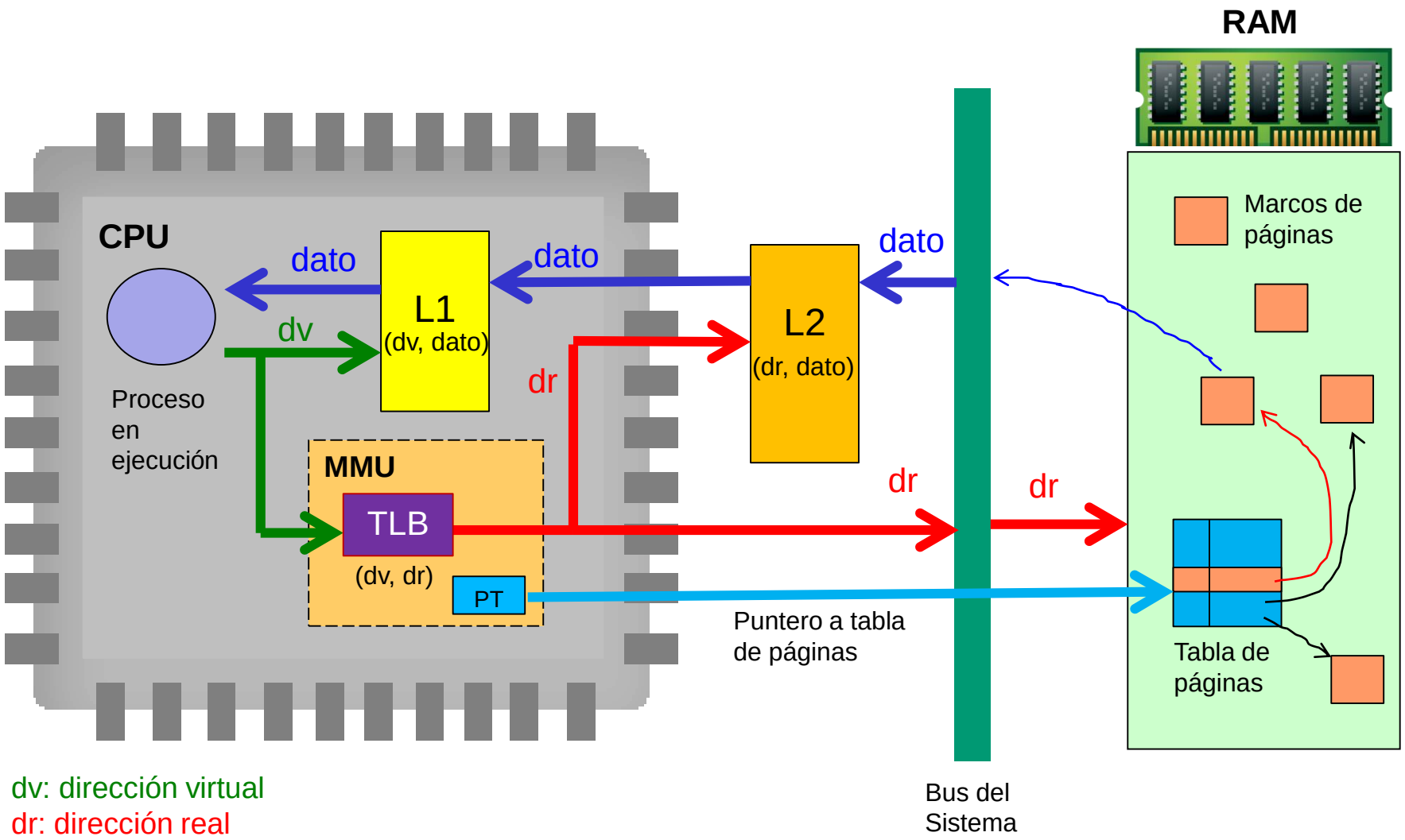
- Las entradas almacenadas son del estilo (**v, r, a**), esta caché se denomina TLB (Translation Lookaside Buffer).



- La TLB no almacena datos, sino solamente traducción de direcciones.
- La TLB se invalida durante un cambio de contexto.

TRADUCCIÓN DE DIRECCIONES





dv: dirección virtual
dr: dirección real
dato: dato que se quiere accesar en memoria RAM

EL COSTO DE UN CAMBIO DE CONTEXTO

- El cambio de contexto consiste en el traspaso del procesador de un proceso a otro.
- El procesador posee un registro que indica la dirección en memoria RAM de la tabla de páginas para el proceso en ejecución.
- En un cambio de contexto se cambia este registro por el puntero a la tabla del proceso entrante.
- Típicamente es necesario invalidar el caché primario L1 y TLB (porque se subindica con direcciones virtuales). **ESTO ES LO COSTOSO!!!**
- El costo adicional del “cambio de contexto” está dado porque hay que repoblar los cachés e ir directamente a la memoria.

OTROS PROBLEMAS

- En los sistemas con gran cantidad de memoria la tabla de páginas de cada proceso tiene que ser lo suficientemente grande para comprender todo el espacio de direcciones del proceso:

- Por ejemplo, en un sistema de 32 bits, cada entrada ocupa 4 bytes, cada tabla de páginas ocupa 4Mb.

- Existen dos soluciones:
  - Tabla de páginas multinivel
  - Tabla de páginas invertida

EL POTENCIAL DEL PAGINAMIENTO

- No hay fragmentación externa → No requiere compactación (Si hay fragmentación interna)
- Implementación eficiente de la instrucción fork.
- Permite implementar protección entre procesos por medio de espacios de direcciones virtuales.
- Extensión automática de la pila.
- Extensión explícita del segmento de datos.
- Swapping: llevar los procesos completos a disco
- Demand Paging!! (Paginamiento en demanda).

PAGINAMIENTO EN DEMANDA

- La **Memoria Virtual** permite usar más páginas de las que caben en la memoria real a cambio de realizar **swapping** con el disco duro, esto se conoce como "**paginación por demanda**".

- Las páginas residentes en disco se marcan con el bit v=0, **no es válida** (no está en memoria RAM).

- Cuando el proceso intenta acceder a una página que no es válida, se produce un **fallo de página** (**page-fault**).

- Entonces el kernel la recupera transparentemente desde disco y pone el bit v=1.

- Típicamente un proceso concentra sus accesos en el 20% a 50% de sus páginas por largos períodos.

PAGINAMIENTO EN DEMANDA

- **Procedimiento:**
  - Las páginas se van cargando en memoria conforme se necesitan.
  - Cuando la memoria RAM está llena y un proceso demanda una nueva página:
    - El hardware detecta el problema y genera una interrupción, atendida por el S.O.
    - Se mueve una página desde memoria real a disco (swap-out).
    - Se carga la página demandada en el marco libre (swap-in).
    - Esto se conoce como **reemplazo de página**

PAGINAMIENTO EN DEMANDA

REEMPLAZO DE PÁGINAS

- Toda la dificultad del paginamiento en demanda está en decidir qué páginas conviene llevar a disco cuando la memoria se hace escasa.

- La página seleccionada es aquella que debe causar un page-fault lo más tarde posible.

- Se puede deducir que el sistema de paginamiento ideal debe llevar a disco aquella página que no será usada por el período de tiempo más largo posible.

ESTRATEGIAS DE REEMPLAZO DE PÁGINAS

- En términos prácticos el kernel no puede predecir por cuanto tiempo una página permanecerá sin ser accesada.
- Es necesario recurrir a estrategias que se aproximen al caso ideal.
- Estas estrategias se denominan **estrategias de reemplazo de páginas**.
- La componente de código del kernel que se encarga de esta labor se define como **scheduler de páginas**.

ESTRATEGIA IDEAL

- Consiste en llevar a disco aquella página que no será usada por el período de tiempo más largo.

# de página virtual	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
a	7			2																
b		0			*		*													
c			1			3														

marcos

Total de page-faults (PF) =

- Esta estrategia es óptima, pero desde luego, no se puede implementar “hay que adivinar”. Sólo sirve para comparar cifras.

ESTRATEGIA FIFO

- Entre las páginas virtuales residentes, se elige como la página a reemplazar la que lleva más tiempo en memoria.

# de página virtual	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
a	7			2																
b		0			*	3														
c			1				0													

Total de page-faults (PF) =

- Problema: muy ineficiente, muchos page-faults

ESTRATEGIA LEAST RECENTLY USED (LRU)

- Consiste en llevar a disco la página que ha permanecido más tiempo sin ser accesada.
- Se ha observado estadísticamente que mientras más tiempo permanece una página sin ser accesada, menos probable es que se accese en el futuro inmediato.
- Si bien, es posible implementar esta estrategia, sería necesario conocer en qué instante fue accesada cada página.
- Se podría modificar el hardware para que coloque esta información en la tabla de páginas en cada acceso, pero en la práctica ningún procesador lo hace.
- Por lo demás, si esta información estuviese disponible, de todas formas, habría que recorrer todas la páginas en cada page-fault, lo que sería demasiado costoso.

ESTRATEGIA LEAST RECENTLY USED (LRU)

- Ejemplo:

# de página virtual	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
a	7			2				4												
b		0			*		*													
c			1			3														

Total de page-faults (PF) =

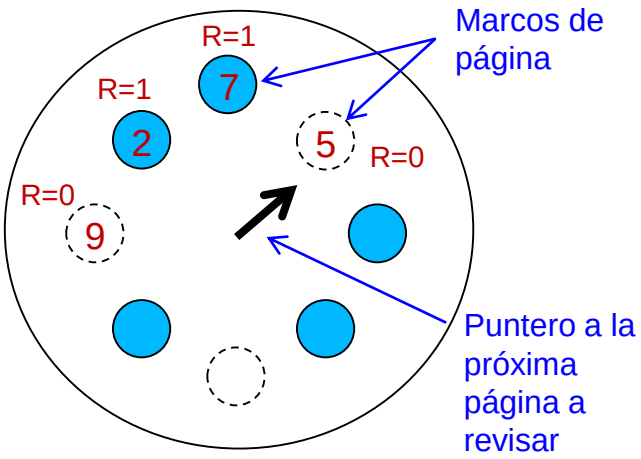
- Problema: caro de implementar.

ESTRATEGIA DEL RELOJ

- Es una aproximación de LRU.
- El fundamento es que no es necesario escoger exactamente la página LRU, sino que es suficiente elegir una página que lleva bastante tiempo sin ser accesada.
- La estrategia usa el bit R de la tabla de página, el hardware coloca en 1 este bit cuando la página es accesada.
- El scheduler de páginas coloca en 0 este bit y lo consulta cuando estima que ha transcurrido el tiempo suficiente como para que aquella página sea reemplazada sino ha sido accesada, es decir si el bit R continúa en 0.

ESTRATEGIA DEL RELOJ

- Esta estrategia ordena las páginas reales circularmente como si fueran los minutos de un reloj.
- Un puntero apunta en cada instante una de las páginas.
- Inicialmente todas las páginas parten con el bit R en 0.
- Cuando una página es accesada su bit R pasa a 1.



ESTRATEGIA DEL RELOJ

- Cuando ocurre un page-fault se ejecuta el siguiente procedimiento:

```
Mientras (el bit R de la página apuntada sea 1) {
   - Colocar el bit R en 0.
   - Avanzar el puntero en una página.
   - Si (la página apuntada tiene su bit R en 0) {
        - Elegir esa página para ser reemplazada,
          es decir que esa página virtual se lleva a disco,
          lo que libera un marco. Ese marco real se usa
          para cargar la página que causó el page-fault.
        - Avanzar el puntero en una página.
        - break
     }
  }
```

- La idea es que la página elegida para ser reemplazada no ha sido accesada por una vuelta completa del reloj.

- La estrategia considera que este tiempo es suficiente como para suponer que la página no será accesada en el futuro inmediato.

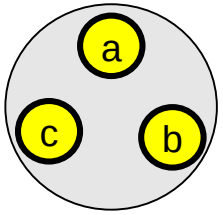
ESTRATEGIA DEL RELOJ

- El tiempo que demora el puntero en dar una vuelta depende de la tasa de page-faults:
  - A mayor tasa, menor será el tiempo en dar la vuelta, entonces el scheduler reemplazará páginas que llevan menos tiempo sin ser accesadas.
  - Al contrario, si la localidad de los accesos a memoria es buena, la tasa de page-faults será baja y el tiempo de una revolución del puntero aumentará. Entonces el scheduler tendrá que reemplazar páginas que llevan más tiempo sin ser accesadas.

- Esta estrategia es fácil de implementar eficientemente con el hardware disponible y funciona muy bien con un solo proceso.

- En caso de varios procesos, la estrategia se comporta razonablemente con carga moderada, pero se comporta desastrosamente cuando la carga es alta.

ESTRATEGIA DEL RELOJ



El círculo alrededor del n° de marco indica que se cargo (R=1) una página producto de un page-fault.

- Ejemplo: