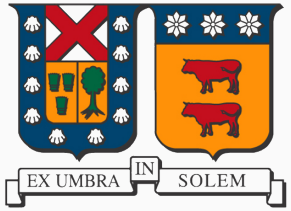


Sistemas Operativos

Memoria Principal

Viktor Andrés Tapia Vásquez

Segundo Semestre 2021

Departamento de Informática, Campus SSJJ.

Índice de Contenidos

Conceptos

Conceptos

- Para ser ejecutados los procesos deben estar en memoria principal.
- Debemos validar el rango de direcciones a las cuales puede acceder.
- Utilizaremos dos registros de apoyo:
  - ✓ **Base:** Almacena la primera dirección.
  - ✓ **Límite:** El tamaño del rango.

Conceptos

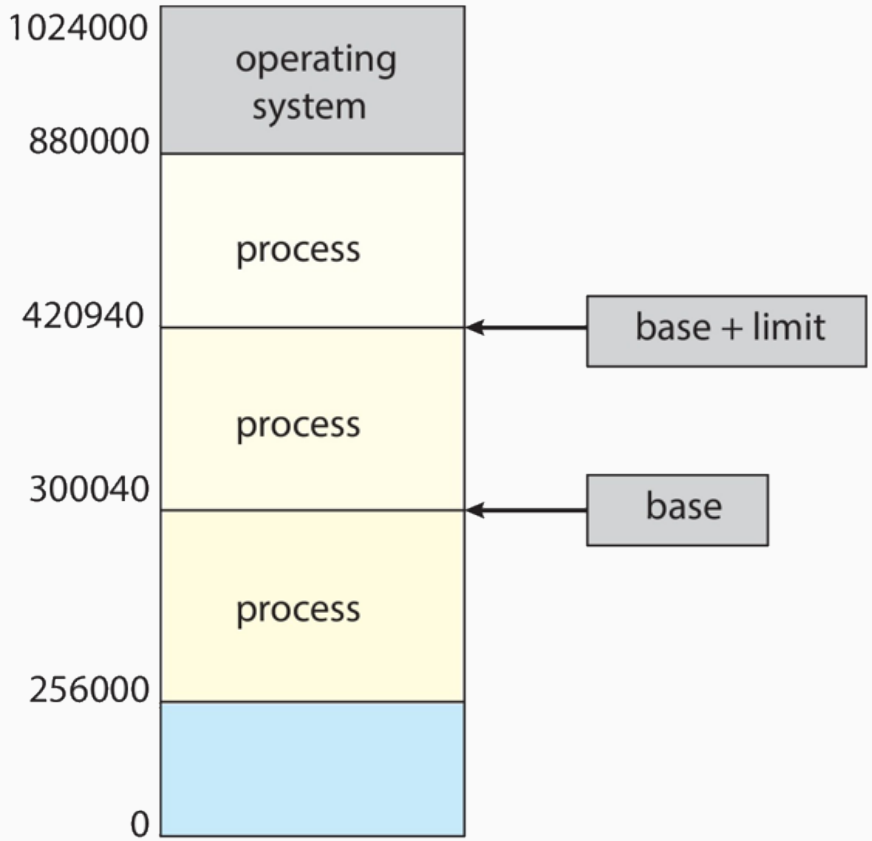


Figure 1: Protección de memoria

Conceptos

- La CPU valida todas las direcciones generadas por un proceso.
- En cada cambio de contexto se actualizan los datos en los registros.
- El intento de acceso a una dirección fuera de rango produce un error.

Conceptos

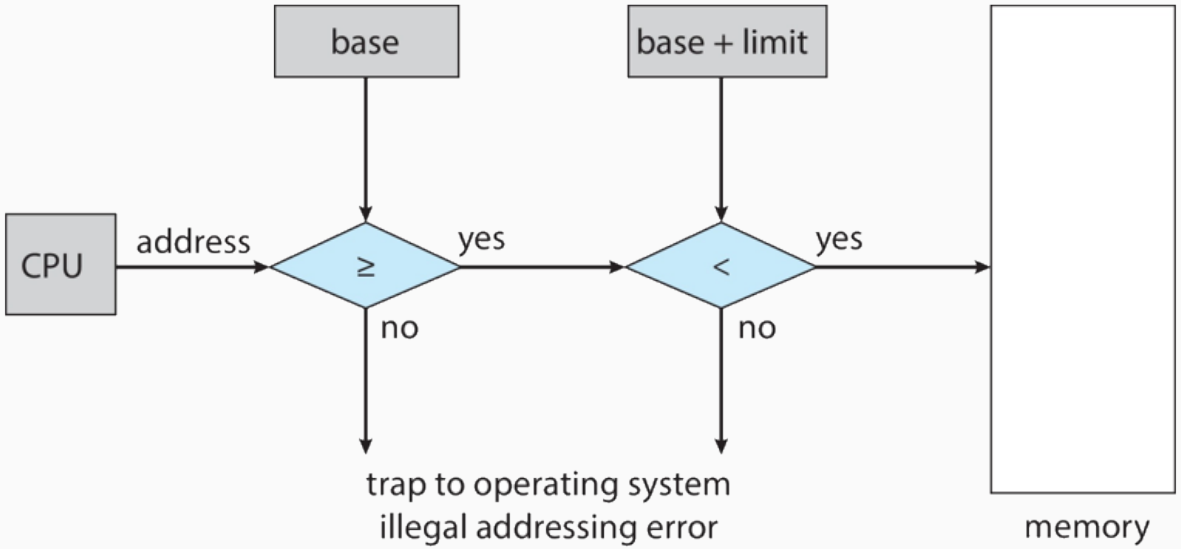


Figure 2: Protección de hardware

Conceptos

- La asignación de direcciones a instrucciones o datos puede ocurrir:
  - ✓ **Compilación:** Se conoce a priori la región.
  - ✓ **Carga:** Si no se conoce la región se crea código re localizable.
  - ✓ **Ejecución:** Necesitamos soporte de hardware.

Conceptos

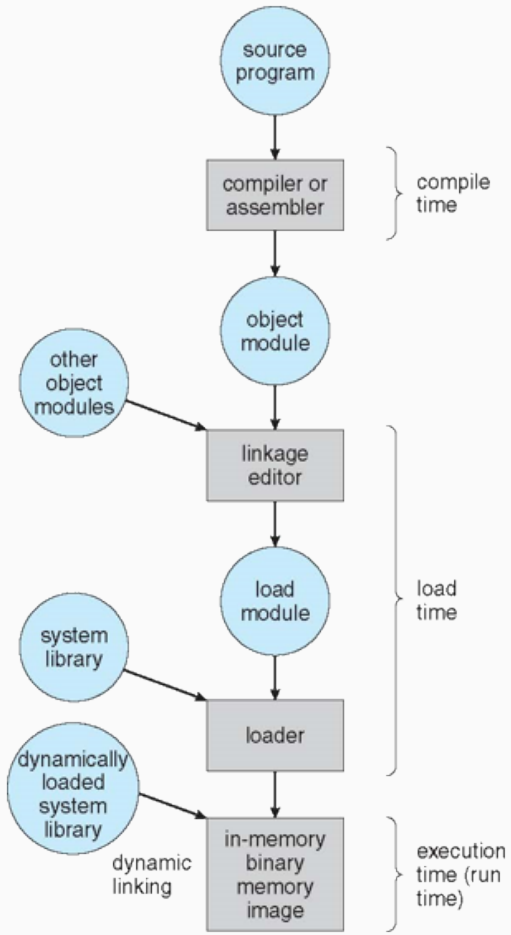


Figure 3: Etapas de la asignación de direcciones

Conceptos

- Dos tipos de direcciones:
  - ✓ **Lógica:** Producida por la CPU.
  - ✓ **Física:** La que identifica la unidad de memoria.
- Son identicas cuando se asignan en tiempo de compilación y carga.
- Difieren cuando se asignan en tiempo de ejecución.
- Será necesario mapear direcciones lógicas a físicas.
- La MMU hace esta pega.

Conceptos

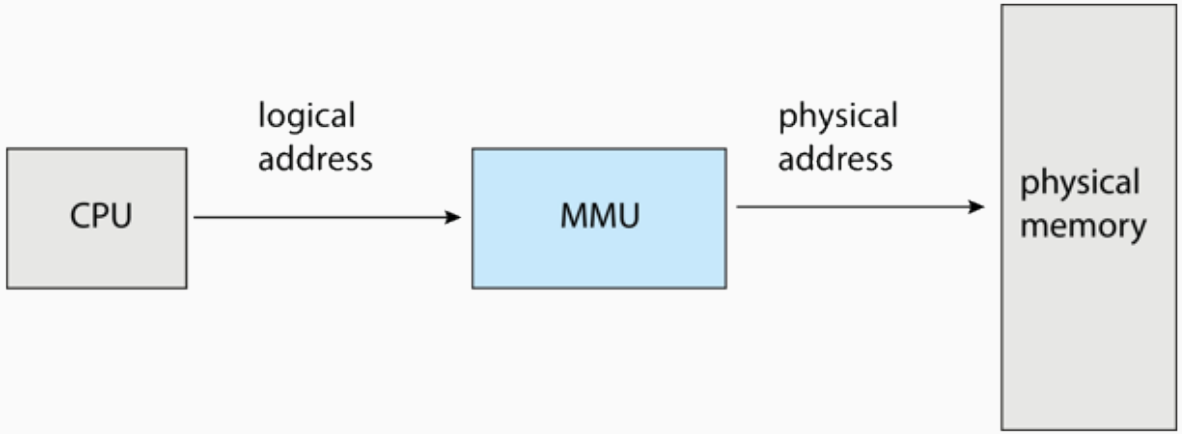


Figure 4: Mapeo de direcciones a través de hardware

Conceptos

- Lo más simple es usar un registro de apoyo.
- El valor de dicho registro se suma a toda dirección generada.
- El proces en ejecución trabaja con direcciones lógicas.
- El registro se conoce como **de Relocalización**.

Conceptos

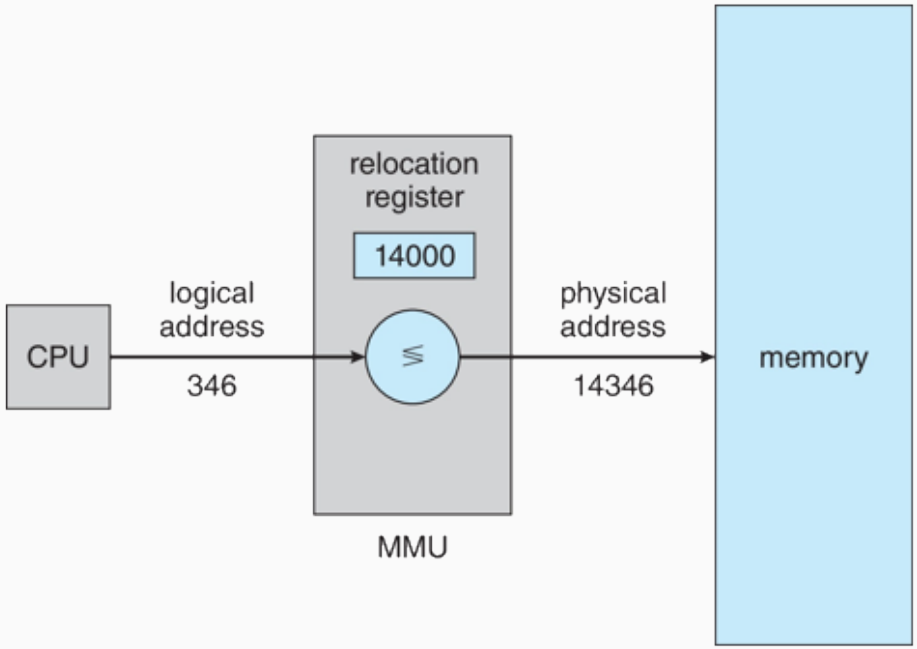


Figure 5: Uso registro relocalización

Conceptos

- Generalmente la memoria se divide en dos partes:
  - ✓ **SO**.
  - ✓ **Procesos de Usuario**.
- Se utilizan los registros de relocalización para proteger los procesos entre si y proteger los accesos indevidos a la región del SO.
- La MMU dinámicamente hace la conversión.

Conceptos

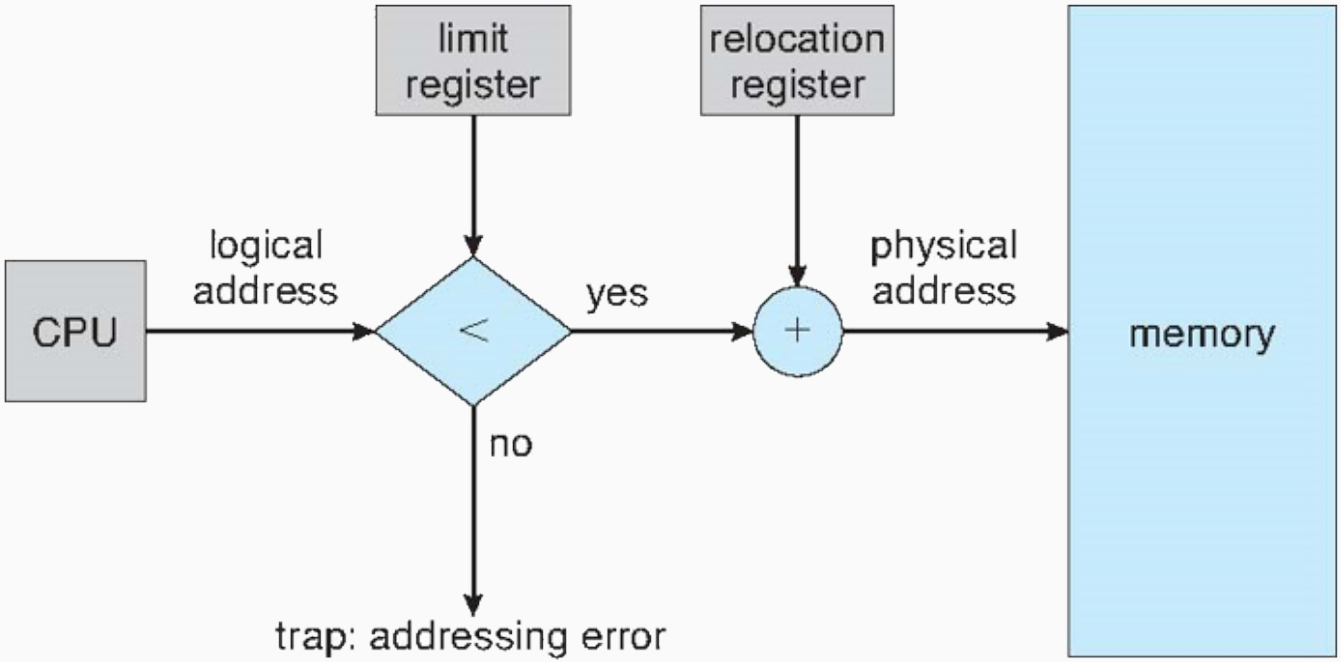


Figure 6: Soporte de hardware para protección

Técnicas de Gestión

Técnicas de Gestión

- Revisaremos las siguientes:
  - ✓ Asignación por bloques fijos.
  - ✓ Asignación por bloques dinámicos.
  - ✓ Segmentación.
  - ✓ Paginación.

Técnicas de Gestión - Asignación por bloques fijos

- Es la forma más sencilla de asignar la memoria.
- Se divide en bloques del mismo tamaño.
- Cada bloque puede albergar solo un proceso.
- Problema: Grado de multiprogramación dependiente.
- El SO mantiene una tabla con las particiones libres.
- En resumen, es poco eficiente.

Técnicas de Gestión - Asignación por bloques dinámicos

- Inicialmente toda la memoria está disponible.
- Se considera un único agujero.
- Cuando llega un proceso:
  - ✓ Se busca una partición lo suficientemente grande.
  - ✓ Se asigna la memoria justa.
- El SO mantiene información:.
  - ✓ Particiones utilizadas.
  - ✓ Particiones disponibles.

Técnicas de Gestión - Asignación por bloques dinámicos

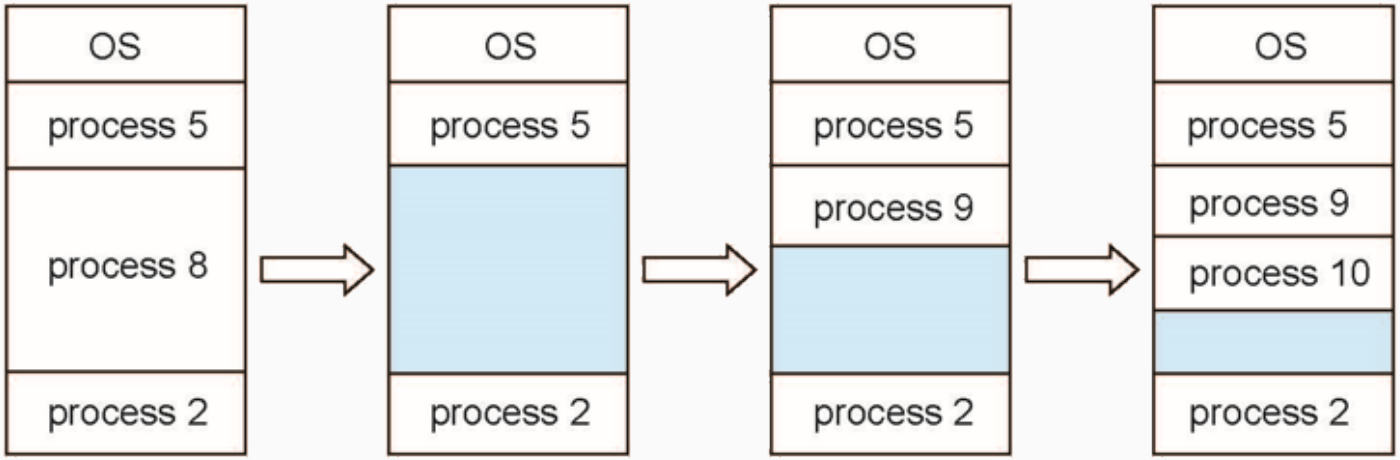


Figure 7: Asignación de bloques dinámicos

Técnicas de Gestión - Asignación por bloques dinámicos

- Podemos tener muchos agujeros libres y dispersos por la memoria.
- Ante la llegada de un proceso ¿qué agujero asignamos?
  - ✓ **Primer ajuste:** Primer bloque con espacio suficiente.
  - ✓ **Mejor ajuste:** El mejor bloque disponible.
  - ✓ **Peor ajuste:** Se asigna la peor opción.

Técnicas de Gestión - Fragmentación

- Ambas técnicas de asignación sufren fragmentación.
- Dos tipos:
  - ✓ **Interna:** Espacio asignado mayor al tamaño del proceso.
  - ✓ **Externa:** El espacio disponible alcanza pero no está contiguo.
- La técnica de **compactación** reduce la externa.
- Regla del 50%: Aún optimizando, dado N bloques asignados, se pierde la mitad por fragmentación.

Técnicas de Gestión - Ejemplo

Utilizando particionamiento dinámico determine cómo se gestiona la memoria si se dispone de 1MB y el SO ocupa los primeros 124KB.

Proceso	Llegada	Duración	KB
A	0	15	100
B	3	5	500
C	6	10	200
D	10	7	50
E	12	2	460
F	15	10	300
G	17	3	250

Técnicas de Gestión - Segmentación

- Soluciona muchas de las limitaciones anteriores.
- Utiliza un arreglo de registros base y límite por proceso.
- Cada entrada controla un segmento del espacio virtual.
- Cada segmento se almacena en direcciones contiguas.
- Diferentes segmentos pueden estar en diferentes ubicaciones.

Técnicas de Gestión - Segmentación

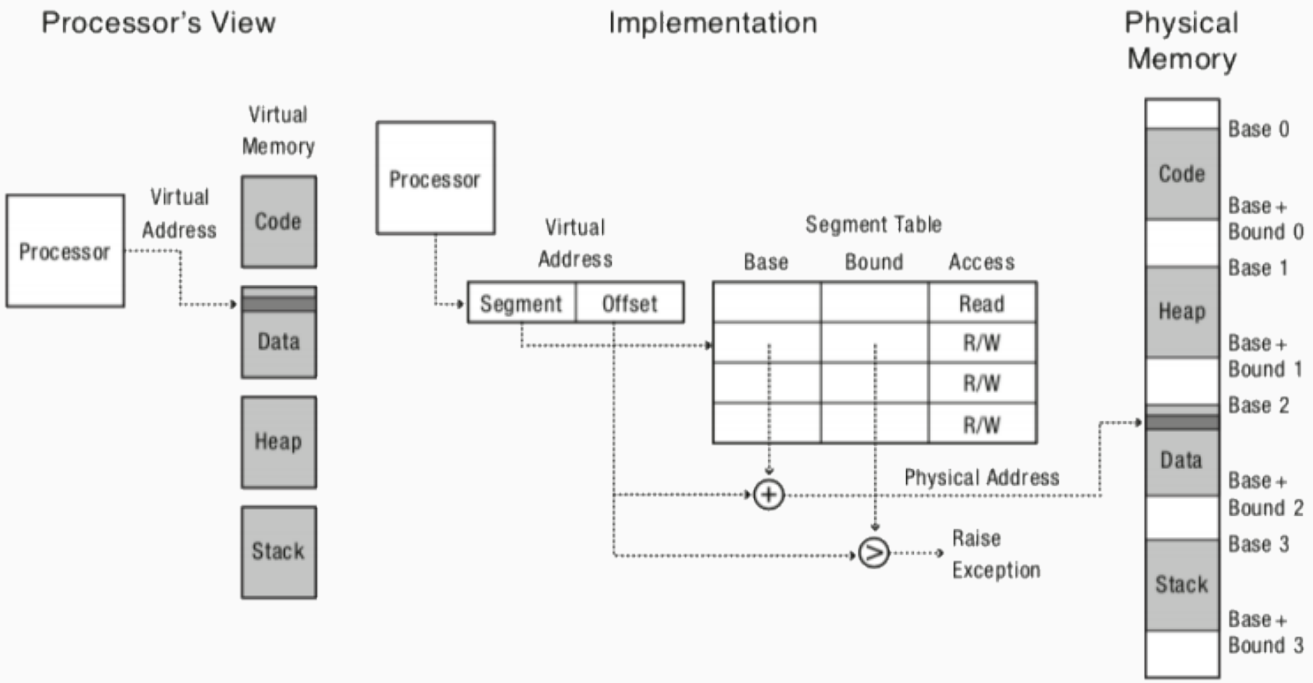


Figure 8: Implementación segmentación

Técnicas de Gestión - Segmentación

- Cada segmento puede tener distintos permisos.
- Entre segmentos pueden quedar agujeros libres.
- Si un segmento referencia un agujero se produce una excepción.
- Este esquema es simple y poderoso.
- El SO puede compartir segmentos entre procesos.

Técnicas de Gestión - Segmentación

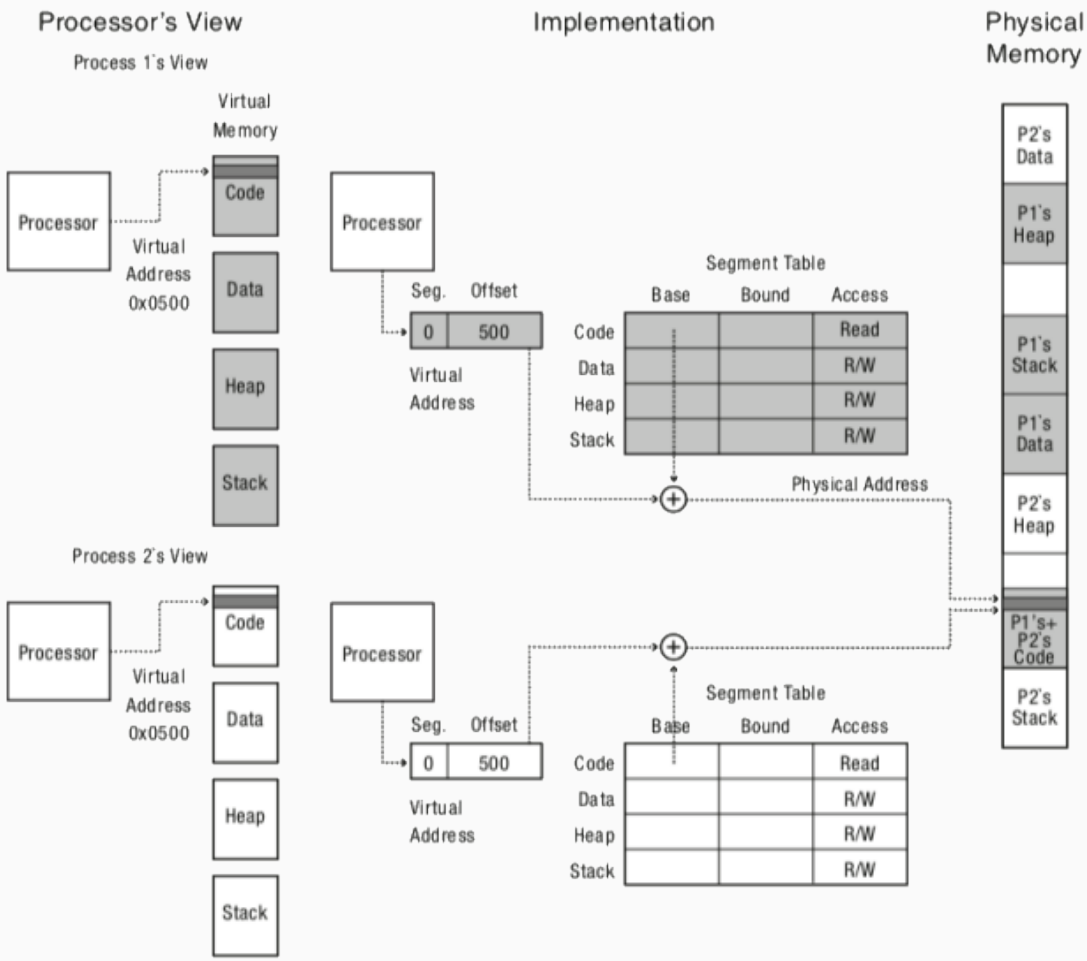


Figure 9: Procesos compartiendo segmentos

Técnicas de Gestión - Segmentación

- En Unix, un fork copia completamente un proceso.
- Utilizando segmentación es más eficiente:
  - ✓ Se copia la tabla de segmentos al hijo.
  - ✓ Se marcan segmentos del padre e hijo como read only.
  - ✓ Se activa el proceso hijo.
  - ✓ Al intentar escribir se produce un trap. El kernel hace la pega.

Técnicas de Gestión - Segmentación

- Problema: Existe overhead con segmentos:
  - ✓ Tamaño variable.
  - ✓ Crecen dinámicamente.
- Se produce fragmentación externa.

Técnicas de Gestión - Ejemplo

Considere direcciones lógicas de 14 bits que usan 2 bits de segmento y 12 de offset. La tabla de segmentos:

Segmento	Base	Largo
0	0x300	0xAAA
1	0x100	0x50
2	0x200	0x500
3	0x2	0x200

Determine las direcciónes físicas para: 0x1AE, 0x1001, 0x29C4, 0x3190.

Técnicas de Gestión - Ejemplo

0,0x1AE	0	0	0	0	0	1	1	0	1	0	1	1	1	0
1,0x1	0	1	0	0	0	0	0	0	0	0	0	0	0	1
2, 0x9C4	1	0	1	0	0	1	1	1	0	0	0	1	0	0
3, 0x190	1	1	0	0	0	1	1	0	0	1	0	0	0	0

#s	Dirección Lógica	Offset	Largo	Dirección Física	Error
0	0x1AE	0x1AE	0xAAA	0x300+0x1AE= 0x4AE	No
1	0x1001	0x001	0x50	0x100+0x001=0x101	No
2	0x29C4	0x9C4	0x500	0x200+0x9C4=0xAC4	Si
3	0x3190	0x190	0x200	0x2+0x190=0x192	No

Figure 10: Solución

Técnicas de Gestión - Paginación

- Esta técnica permite que el espacio de direcciones no sea contiguo.
- Siempre asigna memoria a un proceso cuando hay disponibilidad.
- Dos instrucciones consecutivas pueden estar en regiones diferentes.
- Espacio físico dividido en bloques de tamaño fijo llamados **frames**.
- Espacio lógico dividido en bloques de tamaño fijo llamados **páginas**.

Técnicas de Gestión - Ejemplo

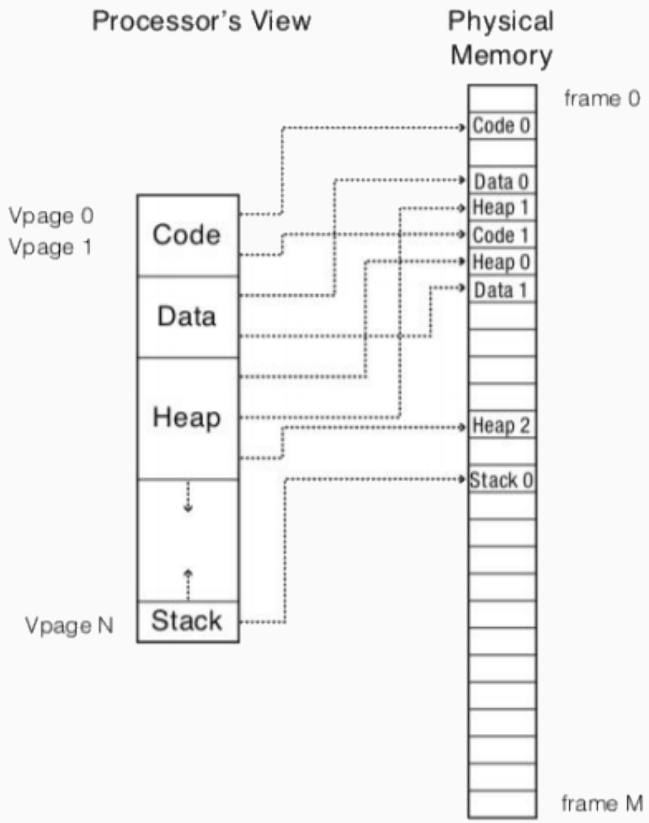


Figure 11: Paginación

Técnicas de Gestión - Paginación

- El tamaño de los bloques es potencia de 2. Entre 512B y 8KB.
- Es necesario llevar un registro de todos los frames libres.
- Un proceso de n páginas requiere n frames libres para ejecutarse.
- Para ejecutarse se deben cargar todas las páginas.
- Se elimina fragmentación externa.
- Se produce fragmentación interna.

Técnicas de Gestión - Paginación

- Se define una estructura llamada **tabla de páginas**.
- Permite transformar direcciones lógicas en físicas.
- La dirección generada por la CPU se divide en:
  - ✓ **Número de página (p):** Índice.
  - ✓ **Offset (d):** Junto con la base, define la dirección.
- Dirección: $p + d$. Con $p = m - n$ y $d = n$.
- Espacio lógico: 2^m y tamaño de página 2^n.

Técnicas de Gestión - Paginación

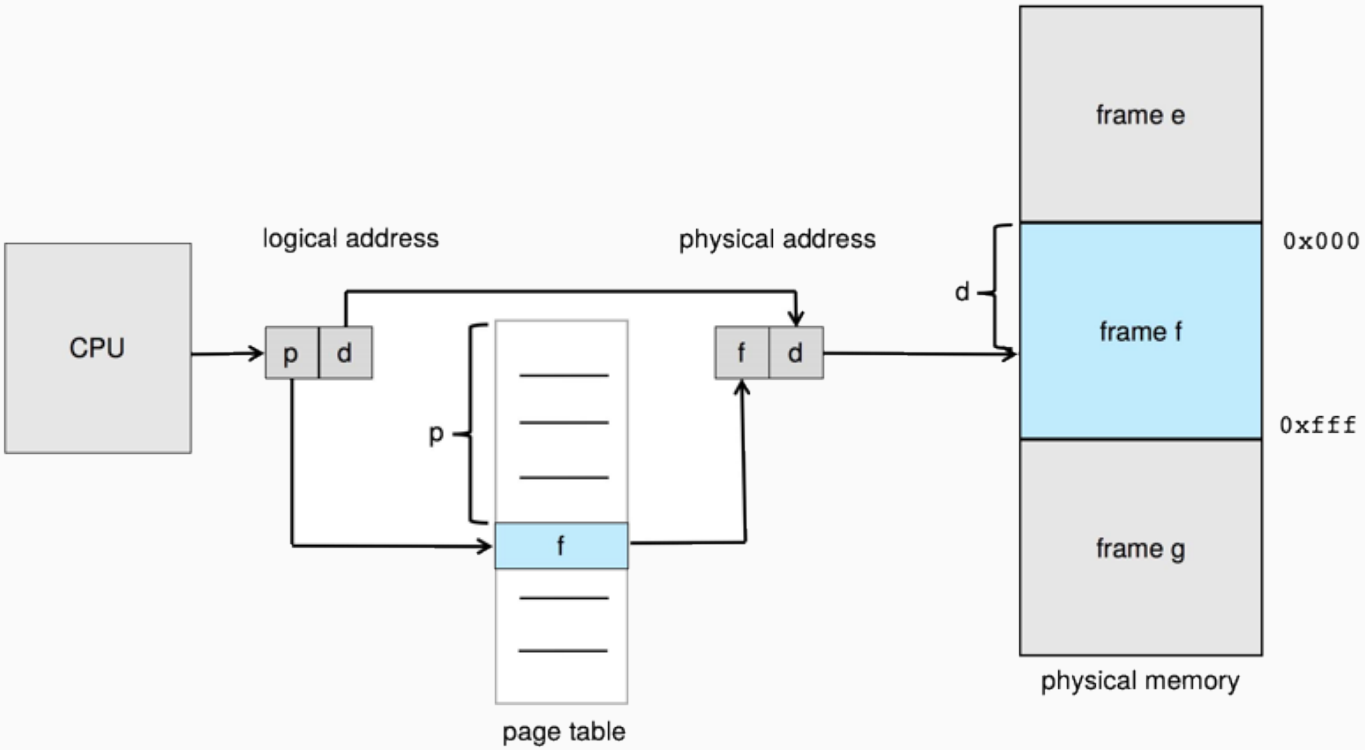


Figure 12: Paginación

Técnicas de Gestión - Paginación

- Se requiere soporte de hardware.
- Usualmente se asigna una tabla de páginas por proceso.
- Se guarda un puntero a la tabla en la PCB.
- Cuando la tabla es grande se mantiene en memoria:
  - ✓ **PTBR:** Registro base que contiene una dirección de la tabla.
  - ✓ **PLTR:** Ancho de la tabla.
- Problema: Se requieren dos accesos a memoria.

Técnicas de Gestión - Paginación

- ¿Se puede ejecutar un programa antes que esté en memoria?
- Inicialmente todas las páginas se marcan inválidas.
- Cuando se hace referencia por primera vez:
  - ✓ Trap al kernel y carga la página.
  - ✓ Se reanuda la ejecución.
- Las páginas restantes se cargan en background.

Técnicas de Gestión - Paginación

- Problema: Tablas de páginas pueden ser muy grandes.
- Overhead si el espacio virtual está muy disperso.
- Muchas entradas a la tabla estarán inválidas.
- Se puede reducir el tamaño de la tabla agrandando el frame.
- Esto produce fragmentación interna.

Técnicas de Gestión - Ejemplo I

Considere un SO que administra la memoria por paginación. Dispone de 256KB. Las direcciones virtuales y físicas son de 16 bits. Un extracto de la tabla de páginas:

Páginas	Marco
0x01	0xDB
0x02	0xFC
0x03	0x5A
0x04	-
0x05	0x7A

La lista de marcos libres es: 0xDA, 0xFB y 0xAC. Si el sistema tiene 1024 marcos determine la dirección física de: 0x01AA, 0x05FB, 0x047B.

Técnicas de Gestión - Ejemplo II

Considere un SO que administra la memoria por paginación. Dispone de 256KB. El tamaño de página es 4KB y las direcciones de 32 bits.

- Determine si las direcciones 0xABC10008 y 0xABC100AA pertenecen al mismo marco.
- Lo mismo para 0xABC1FA00 y 0xABC2FA08.
- ¿Cuántos marcos tiene el sistema?
- Si las páginas fuesen de 1KB. La misma primera pregunta.
- Determine el tamaño máximo de la tabla de páginas.

Gestión Eficiente

Gestión Eficiente

- Todas las técnicas anteriores sufren uno que otro problema.
- Además, la búsqueda en un arreglo no es eficiente.
- Sistemas basados en árboles tienen un mejor comportamiento.
- Nivel inferior siempre será paginación.

Gestión Eficiente

- Revisaremos:
  - ✓ Segmentación + Paginación.
  - ✓ Paginación de varios niveles.
  - ✓ Uso de TLB.

Gestión Eficiente

- ¿Por qué paginación en el nivel inferior?
  - ✓ **Asignación de memoria:** Utilización de bit maps para gestionar espacio libre de memoria.
  - ✓ **Transferencia de disco:** Si la página es múltiplo del sector de disco se simplifica la transferencia.
  - ✓ **Búsqueda:** Uso de la TLB como caché.
  - ✓ **Granularidad en protección y memoria compartida.**

Gestión Eficiente - Segmentación Paginada

- Es un árbol de dos niveles.
- Cada entrada de la tabla de segmentos apunta a una página.
- El tamaño de los segmentos es múltiplo de la página.
- La tabla de segmentos se puede almacenar en registros especiales.
- La tabla de página de cada segmento es almacenada en memoría.

Gestión Eficiente - Segmentación Paginada

- Ejemplo implementación. Considere:
  - ✓ Dirección virtual de 32 bits.
  - ✓ Páginas de 4KB.
  - ✓ Dirección virtual:
  - ✓ 10 bits para el número de segmento.
  - ✓ 10 bits para el número de página.
  - ✓ Offset de página de 12 bits.
- Si cada entrada de la tabla de página tiene 4B, la tabla de página de cada segmento cabe en un marco de memoria.

Gestión Eficiente - Segmentación Paginada

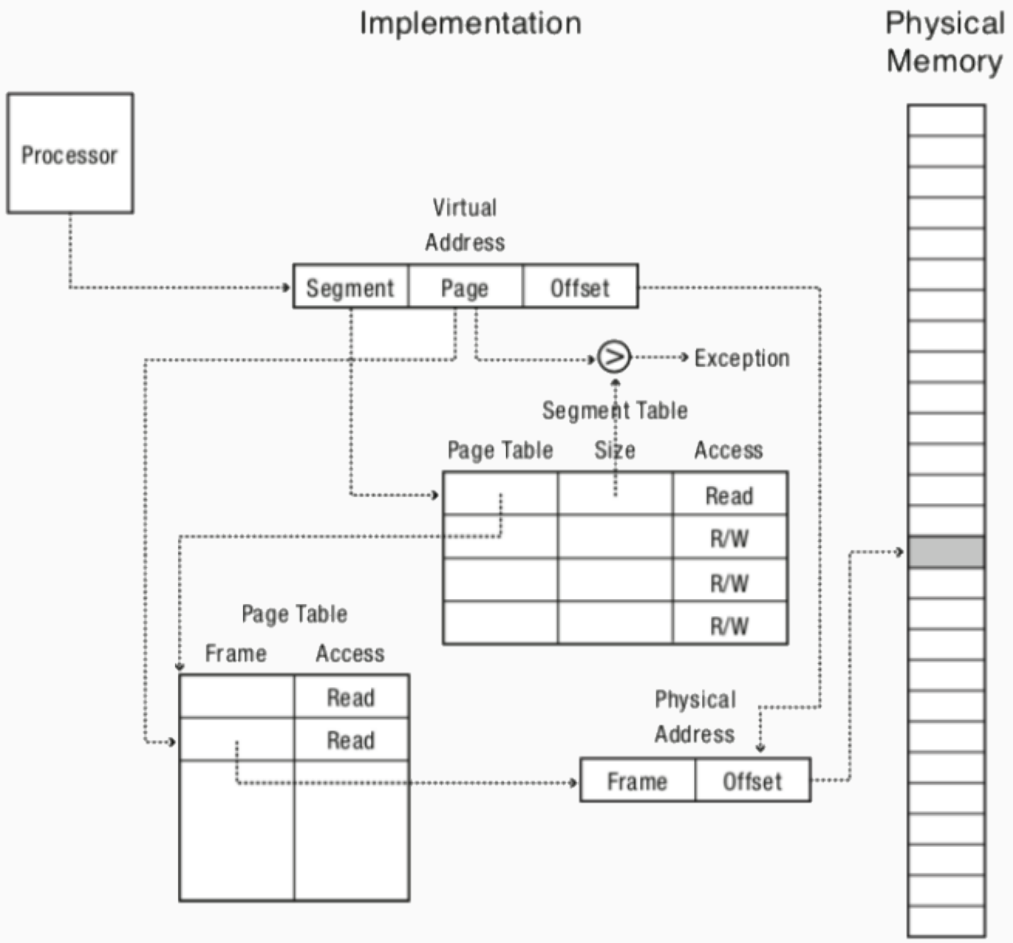


Figure 13: Segmentación Paginada

Gestión Eficiente - Paginación de Varios Niveles

- Cada nivel de la tabla de página se diseña para entrar en un marco.
- En cada nivel:
  - ✓ Distintos permisos.
  - ✓ Compartir páginas.

Gestión Eficiente - Paginación de Varios Niveles

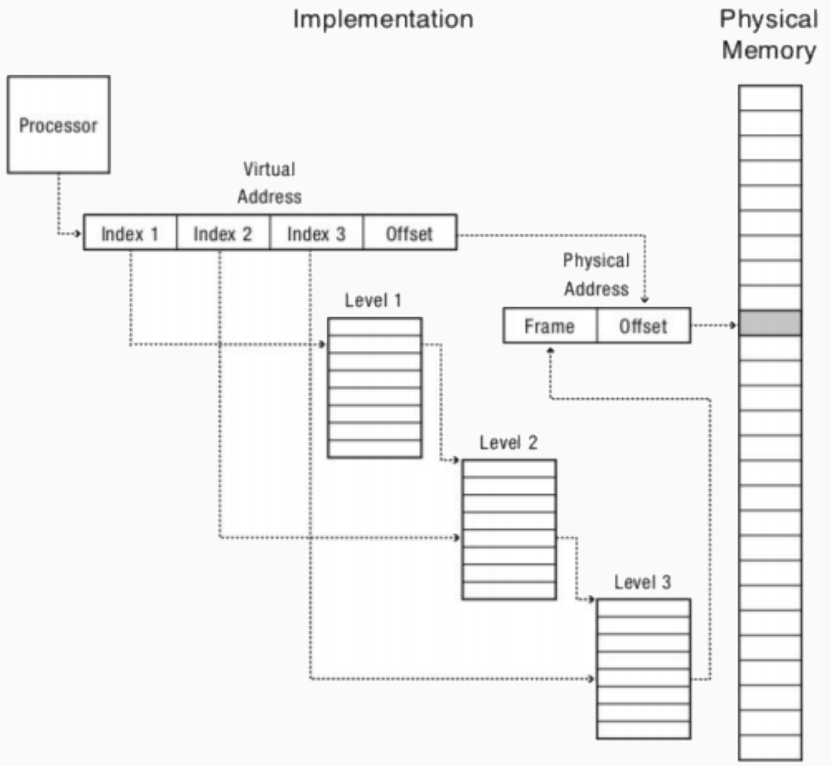


Figure 14: Paginación de Varios Niveles

Gestión Eficiente - Uso de TLB

- Podemos mejorar estas técnicas usando cachés.
- La TLB es una tabla de hardware que guarda el resultado anterior.
- Cada entrada mapea una página virtual en una página física.

Gestión Eficiente - Uso de TLB

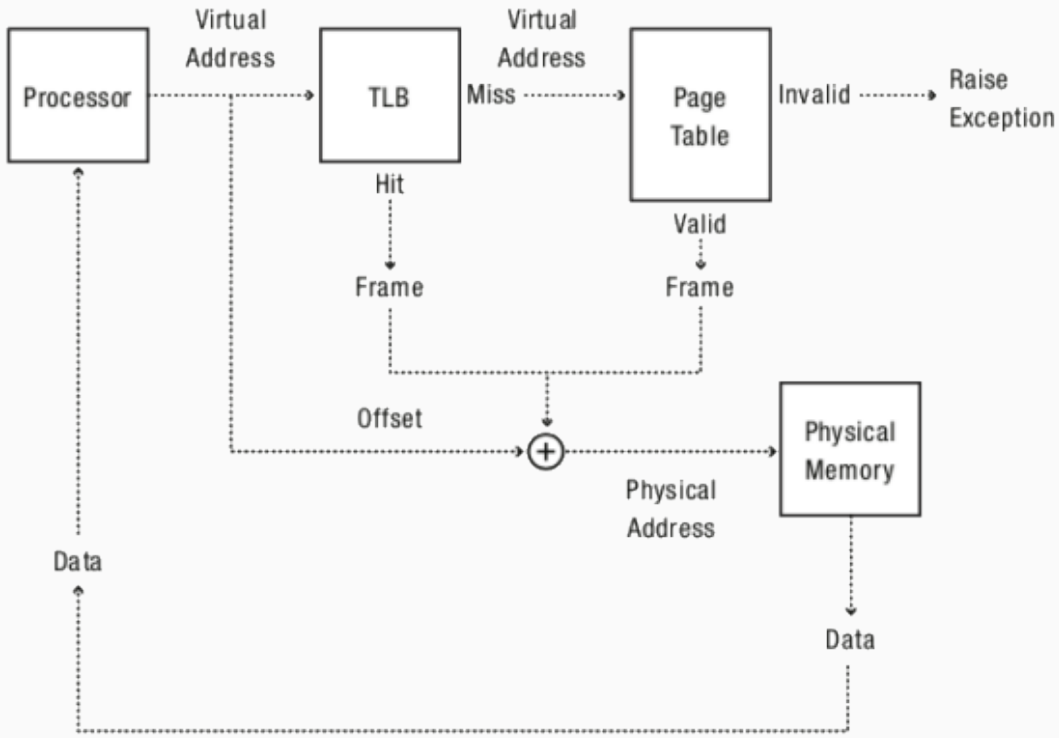


Figure 15: Uso de TLB

Gestión Eficiente - Uso de TLB

- TLB hace búsquedas en paralelo.
- Si hay un calce, utiliza la entrada encontrado.
- Si no encuentra, hace la conversión completa.
- Se implementa en memoria estática cerca del procesador.

Gestión Eficiente - Uso de TLB

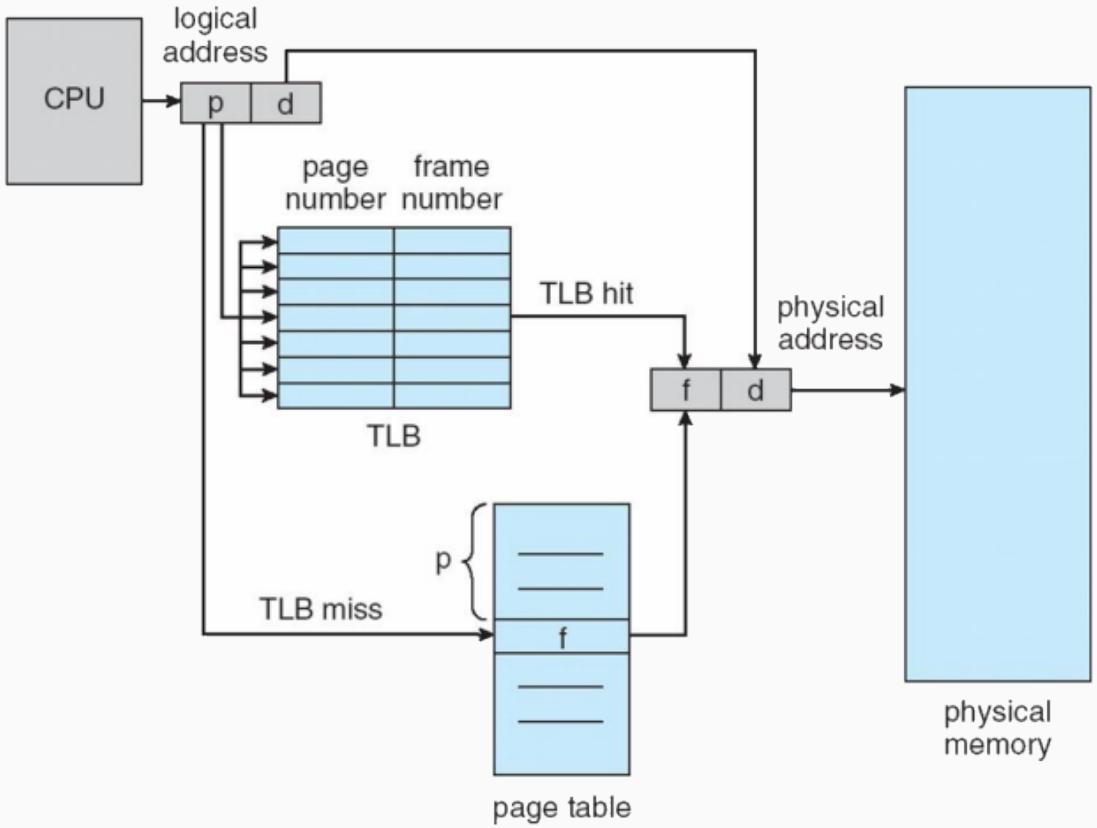
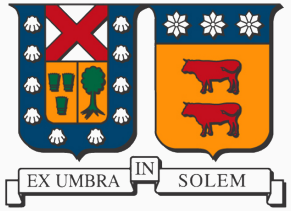


Figure 16: Uso de TLB



Sistemas Operativos

Memoria Principal

Viktor Andrés Tapia Vásquez

Segundo Semestre 2021

Departamento de Informática, Campus SSJJ.