BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

Facultad de Ciencias de la Computación

2025

SISTEMAS OPERATIVOS II

Traducción de Direcciones Virtuales a Físicas

Carrera: Ingeniería en Ciencias de la Computación
Sección: 003
Grado: 6to semestre

Integrantes del equipo:

Rodríguez Maldonado José Antonio: 202250718
Santiago Ibáñez José Luis: 202253693
Vergara Mora Jorge Luis: 202256565

Contents

INTRODUCCIÓN

Hoy en día en los sistemas modernos, la función de gestión de memoria es una de las tareas más fundamentales e importantes que nos permite que múltiples procesos puedan ejecutarse de manera segura y consistente. Por esta razón mencionamos que, para lograr esta gestión, los sistemas emplean memoria virtual, que crea la ilusión de que cada proceso dispone de un espacio de direcciones continuo y exclusivo. Pero en realidad, este espacio virtual se mapea a la memoria física disponible mediante tablas de páginas, donde cada página virtual se asocia con un marco físico correspondiente.

Por ello el presente programa tiene como finalidad simular el proceso de traducción de direcciones virtuales a físicas mediante un esquema comúnmente llamada paginación, emulando así el funcionamiento básico de una MMU (Unidad de Gestión de Memoria).
Para lograr esto implementamos primero la carga de datos de configuración inicial que va desde el tamaño de la memoria virtual, memoria física y tamaño de página.
La inicialización de la tabla de páginas con mapeos establecidos desde un archivo externo (txt). Seguido del cálculo de la dirección física a partir de una dirección virtual ingresada por el usuario, mostrando el proceso paso a paso (extracción de número de página, desplazamiento, consulta de la tabla de páginas y ensamblaje de la dirección física). Además de indicar los fallos de página que es cuando una página virtual no se encuentra cargada en memoria mediante un bit que nos indica si está presente o ausente.

De esta manera, el programa se constituye de una simulación que sirve como herramienta de aprendizaje que permite comprender y visualizar de forma práctica cómo se lleva a cabo la traducción de direcciones en un sistema con paginación.

DESARROLLO

1.1 Análisis del Problema

El objetivo fundamental del programa es simular la traducción de direcciones de memoria virtual, que es la que maneja un programa. En una dirección física, que corresponde a la ubicación real en la memoria RAM.

Para lograr esto se emplea un esquema de paginación, con este enfoque dividimos la memoria virtual y física en bloques de tamaño fijo (páginas y marcos).

La traducción se hace a partir de la tabla de páginas. Por cada página virtual, la tabla indica en que marco físico se encuentra almacenada.

Por lo que nuestro simulador debe ser capaz de:

1. Configurar los tamaños de memoria virtual y física, además del tamaño de cada página/marco
2. Inicializar una tabla de páginas, que especifique que páginas virtuales ya se encuentran en qué marcos físicos.
3. Procesar una dirección virtual:
   - Descomponerla en sus componentes: el número de página virtual y el desplazamiento dentro de esa página.
   - Consultar la tabla de páginas con el número de página virtual para encontrar el marco físico correspondiente.
4. Calcula la dirección física: combinar el número de marco físico obtenido de la tabla con el desplazamiento original.

1.2 Diseño de la solución.

Para encapsular la lógica del simulador optamos por un diseño orientado a objetos utilizando una clase llamada TraductorDeDirrecciones.

Componentes Principales:

- **Clase TraductorDeDirecciones:** Es el núcleo del simulador.
- **Atributos:** Almacena todos los parámetros de configuración (tamaños de memoria, tamaño de página), los valores calculados (número de bits para cada campo, máscaras) y, lo más importante, la tabla_de_paginas.
- **Constructor (__init__):** Se encarga de recibir la configuración inicial. A partir de estos datos, calcula y almacena todos los parámetros derivados, como el número total de

páginas, el número de marcos y la cantidad de bits necesarios para el desplazamiento y el número de página. Esto centraliza la configuración en un solo punto, haciendo el sistema más robusto.

- **Método _inicializar_tabla_paginas:** Pobla la estructura de datos de la tabla de páginas. Se diseñó para marcar cada página como "presente" o "no presente" en memoria, lo cual es esencial para simular fallos de página.
- **Método traducir:** Contiene la lógica principal del proceso. Recibe una dirección virtual y orquesta los pasos de descomposición, consulta y construcción de la dirección física final. Su diseño se enfoca en la claridad, mostrando cada cálculo intermedio.
- **Métodos auxiliares (imprimir_tabla_paginas, imprimir_binario):** Funciones de apoyo diseñadas para visualizar el estado del sistema (la tabla de páginas) y formatear los números en binario de manera legible, mejorando la comprensión del proceso.

Estructura de Datos:

- **Tabla de Páginas:** Se implementó como un diccionario de Python. La clave de cada entrada es el número de página virtual (un entero). El valor es otro diccionario que contiene dos datos clave: el marco físico asignado y un bit de presente (1 si está en memoria, 0 si no). Esta estructura es eficiente para búsquedas y flexible para representar la información necesaria.

1.3. Implementación

La implementación se centró en el uso de operaciones a nivel de bits, ya que son la forma natural y eficiente en que un procesador real manipula las direcciones.

Puntos Clave de la Implementación:

1. **Carga del Archivo de Configuración:** Para implementar la configuración externa, se desarrolló una función (cargar_configuracion_desde_archivo) responsable de leer y procesar el archivo de texto. La lógica de distingue entre dos secciones dentro del archivo mediante una palabra clave (MAPEOS:). Esto permite leer primero los parámetros globales y luego los mapeos de página a marco, almacenándolos en estructuras de datos separadas.

2. **Cálculo de Bits:** En el constructor, se usan logaritmos en base 2 para determinar cuántos bits se necesitan para cada parte de la dirección. Por ejemplo, para el desplazamiento:

3. **Descomposición de la Dirección Virtual:** En el método traducir, se extraen el número de página y el desplazamiento mediante operaciones de bits.
   - Número de Página: Se obtiene desplazando los bits de la dirección virtual hacia la derecha. Esto elimina los bits del desplazamiento y deja únicamente los bits correspondientes al número de página.

   - Desplazamiento (Offset): Se aísla aplicando una máscara con la operación AND. La máscara tiene '1' en las posiciones de los bits del desplazamiento y '0' en el resto, lo que permite "recortar" la parte de la dirección que nos interesa.

4. **Construcción de la Dirección Física:** Una vez obtenido el número de marco desde la tabla de páginas, la dirección física se ensambla combinando el marco y el desplazamiento. Esto también se realiza con operaciones de bits:
   - Primero, se desplaza el número de marco hacia la izquierda para posicionarlo en los bits de orden superior de la dirección.
   - Luego, se usa la operación OR para "pegar" los bits del desplazamiento en los bits de orden inferior que quedaron en cero.

5. **Manejo de Errores:** Se implementaron dos validaciones críticas dentro del método traducir:
   - Página Inválida: Se verifica si el número de página extraído existe como clave en la tabla_de_paginas. Si no, la dirección es ilegal.
   - Fallo de Página: Si la página existe, se comprueba su bit de presente. Si es 0, se reporta un fallo de página, ya que la página es válida pero no reside actualmente en un marco físico.

EVIDENCIAS DE REPORTE

A continuación de muestran algunas evidencias de ejecución de funcionalidad del proyecto de traducciones virtuales a físicas, simulando al MMU

```
--- Parámetros del Traductor (cargados desde archivo) ---
Tamaño Memoria Virtual: 512
Tamaño Memoria Física: 256
Tamaño de Página: 32
--------------------
Número total de páginas: 16
Número total de marcos: 8
Bits para dirección virtual: 9
Bits para dirección física: 8
Bits para página virtual: 4
Bits para marco: 3
Bits para desplazamiento: 5
Máscara de desplazamiento: 0000 0000 0001 1111 (0x1F)

✅ Tabla de páginas inicializada desde el archivo.

📘 Tabla de Páginas:
------------------------------------------------
Página    Marco     Presente
------------------------------------------------
0         2         1
1         n/a       0
2         3         1
3         4         1
4         5         1
5         n/a       0
6         7         1
7         0         1
8         n/a       0
9         n/a       0
10        n/a       0
11        n/a       0
12        n/a       0
13        n/a       0
14        n/a       0
15        n/a       0
------------------------------------------------
```

Traducción de Direcciones virtuales a físicas

```
--- Listo para traducir ---
Ingrese una dirección virtual en HEXADECIMAL (ej. 0x1A2F) o 'q' para salir: b9

--- Traduciendo Dirección Virtual: 0xB9 ---
DV en binario (9 bits): 0101 1100 1

1. Extracción de Componentes:
   Número de Página = 185 >> 5  = 5
   Desplazamiento   = 185 & 31 = 25 (bin: 1100 1, hex: 0x19)

2. Consulta a la Tabla de Páginas:
   -> Entrada para página 5: {'marco': 0, 'presente': 1}
   -> ✅ La página está presente en memoria.

3. Cálculo de la Dirección Física:
   Fórmula: (marco << bits_desplazamiento) | desplazamiento
   Cálculo: (0 << 5) | 25 = 25
   Bits del marco: 3, Bits de dirección física: 8

--- Resultado ---
Dirección Física: binario = 0001 1001
                  hexadecimal = 0x19
-----------------
```

```
--- Traduciendo Dirección Virtual: 0xC0 ---
DV en binario (9 bits): 0110 0000 0

1. Extracción de Componentes:
   Número de Página = 192 >> 5  = 6
   Desplazamiento   = 192 & 31 = 0 (bin: 0000 0, hex: 0x0)

2. Consulta a la Tabla de Páginas:
   -> Entrada para página 6: {'marco': 7, 'presente': 1}
   -> ✅ La página está presente en memoria.

3. Cálculo de la Dirección Física:
   Fórmula: (marco << bits_desplazamiento) | desplazamiento
   Cálculo: (7 << 5) | 0 = 224
   Bits del marco: 3, Bits de dirección física: 8

--- Resultado ---
Dirección Física: binario = 1110 0000
                  hexadecimal = 0xE0
-----------------
```

```
Ingrese una dirección virtual en HEXADECIMAL (ej. 0x1A2F) o 'q' para salir: 4f

--- Traduciendo Dirección Virtual: 0x4F ---
DV en binario (9 bits): 0010 0111 1

1. Extracción de Componentes:
   Número de Página = 79 >> 5  = 2
   Desplazamiento   = 79 & 31 = 15 (bin: 0111 1, hex: 0xF)

2. Consulta a la Tabla de Páginas:
   -> Entrada para página 2: {'marco': 3, 'presente': 1}
   -> ✅ La página está presente en memoria.

3. Cálculo de la Dirección Física:
   Fórmula: (marco << bits_desplazamiento) | desplazamiento
   Cálculo: (3 << 5) | 15 = 111
   Bits del marco: 3, Bits de dirección física: 8

--- Resultado ---
Dirección Física: binario = 0110 1111
                  hexadecimal = 0x6F
-----------------
```

Casos de fallos de página

```
-----------------
Ingrese una dirección virtual en HEXADECIMAL (ej. 0x1A2F) o 'q' para salir: E4

--- Traduciendo Dirección Virtual: 0xE4 ---
DV en binario (9 bits): 0111 0010 0

1. Extracción de Componentes:
   Número de Página = 228 >> 5  = 7
   Desplazamiento   = 228 & 31 = 4 (bin: 0010 0, hex: 0x4)

2. Consulta a la Tabla de Páginas:
   -> Entrada para página 7: {'marco': 0, 'presente': 0}
   ❌ FALLO DE PÁGINA: La página 7 no está cargada en memoria.
```

```
Ingrese una dirección virtual en HEXADECIMAL (ej. 0x1A2F) o 'q' para salir: E4

--- Traduciendo Dirección Virtual: 0xE4 ---
DV en binario (9 bits): 0111 0010 0

1. Extracción de Componentes:
   Número de Página = 228 >> 5  = 7
   Desplazamiento   = 228 & 31 = 4 (bin: 0010 0, hex: 0x4)

2. Consulta a la Tabla de Páginas:
   -> Entrada para página 7: {'marco': 0, 'presente': 0}
   ❌ FALLO DE PÁGINA: La página 7 no está cargada en memoria.
Ingrese una dirección virtual en HEXADECIMAL (ej. 0x1A2F) o 'q' para salir:
```

Detección de Errores

```
Error durante la ejecución: Página 18 inválida (0..15)
PS C:\Users\rodriguezmldo\Documents\Anaconda\sistemasOperativos\MMU> & C:/Users/rodriguezmldo/anaconda3
zmldo/Documents/Anaconda/sistemasOperativos/MMU/main.py

Error durante la ejecución: El tamano de memoria virtual debe ser divisible por el tamano de pagina
PS C:\Users\rodriguezmldo\Documents\Anaconda\sistemasOperativos\MMU>
 main*  ⊗ 0 ⚠ 0
```

```
Ingrese una dirección virtual en HEXADECIMAL (ej. 0x1A2F) o 'q' para salir: y64
❌ Entrada inválida. Use formato hexadecimal (ej: 0x1A2F o 1A2F).
Ingrese una dirección virtual en HEXADECIMAL (ej. 0x1A2F) o 'q' para salir: h8
❌ Entrada inválida. Use formato hexadecimal (ej: 0x1A2F o 1A2F).
Ingrese una dirección virtual en HEXADECIMAL (ej. 0x1A2F) o 'q' para salir: 0x901

--- Traduciendo Dirección Virtual: 0x901 ---
DV en binario (9 bits): 1000 0000 1

1. Extracción de Componentes:
   Número de Página = 2305 >> 5  = 72
   Desplazamiento   = 2305 & 31 = 1 (bin: 0000 1, hex: 0x1)

2. Consulta a la Tabla de Páginas:
   ❌ Error: La página 72 es inválida para este espacio de direcciones.
Ingrese una dirección virtual en HEXADECIMAL (ej. 0x1A2F) o 'q' para salir:
```

ALCANCE Y LIMITACIONES DEL PROGRAMA

Alcances

1. **Simulación de la lógica de traducción de direcciones:** El programa permite comprender como funciona la traducción de direcciones virtuales a física bajo un esquema de paginación:
    - Divide correctamente la memoria virtual
    - Reemplaza el número de página por el marco físico correspondiente
    - Reconstruye la dirección física con los bits del marco y el desplazamiento
2. **Procesos detallados:** Este simulador no solo calcula la dirección física, sino que también muestra paso a paso:
    - Extracción de componente (número de página y desplazamiento)
    - Consulta a la tabla de páginas
    - Cálculo de la dirección física con representaciones en binario, hexadecimal y decimal
3. **Manejo de fallos de página:** El simulador es capaz de detectar fallos de página cuando el bit “Presente” está en 0.
4. **Flexibilidad de configuración: se** permite simular diferentes escenarios y comprender el impacto de tamaño de página en la división de bits.

Limitaciones

1. **No consideramos protecciones de memoria:** en la MMU real de manejan permisos como lo son de lectura, escritura y ejecución, solo contenemos marcos y bits presente, suficiente para las traducciones de direcciones.
2. **Sin reemplazos de páginas:** ocurriendo un fallo de página, la MMU invoca el sistema operativo para traer la página desde disco y en de ser necesario aplicar algoritmos de reemplazo, en nuestro caso solo indicamos error al acceso de página.
3. **Limitación de traducciones:** Al no incluir el manejo de reemplazos, permiso y jerarquías nos limitamos a traducciones estáticas con una tabla de páginas plana.

CONCLUSIÓN

Este simulador ha demostrado exitosamente el mecanismo de traducción de direcciones de memoria virtual a física mediante paginación, permitiendo visualizar de manera práctica cómo el sistema:

- Extrae el número de página y el desplazamiento a partir de una dirección virtual.
- Consulta la tabla de páginas para localizar el marco físico correspondiente.
- El simulador detecta fallos de página con el bit "Presente" en la tabla de páginas.
- Calcula la dirección física final combinando el marco con el desplazamiento.

La implementación reforzó nuestra comprensión de conceptos fundamentales como la división de memoria en páginas y marcos, la relación entre direcciones virtuales y físicas, y el papel crítico de la tabla de páginas en la traducción.

Este proyecto constituye una base sólida para explorar conceptos avanzados como paginación multinivel, segmentación y algoritmos de reemplazo de páginas (FIFO, LRU), que gestionan los fallos de página cuando no hay marcos disponibles.

Por lo tanto, el traductor no solo cumple con simular una MMU básica, sino que funciona como una herramienta didáctica que nos permite comprender el funcionamiento interno de los sistemas de memoria virtual, consolidando el aprendizaje teórico mediante la experimentación práctica.

BIBLIOGRAFÍA

[1] Tanenbaum, A. S., & Bos, H. (2015). *Modern Operating Systems* (4.ª ed.). Pearson Education. ISBN: 978-0133591620. Recuperado de https://csc-knu.github.io/sys-prog/books/Andrew%20S.%20Tanenbaum%20-%20Modern%20Operating%20Systems.pdf CSC KNU

[2] IBM. *Paging space and virtual memory*. Documentación IBM AIX. Recuperado el 25 de septiembre de 2025, de https://www.ibm.com/docs/es/aix/7.2.0?topic=management-paging-space-virtual-memory

[3] Elvira, J. L. (2017, 22 de marzo). *Sistemas Operativos, Memoria virtual 2 paginación por demanda* [Video]. YouTube. Recuperado el 25 de septiembre de 2025, de https://www.youtube.com/watch?v=JYyyoeHEAXU

[4] Reyes Reyes, J. A. (2023). *Simulador interactivo para la enseñanza-aprendizaje de algoritmos de reemplazo de páginas en los sistemas operativos* (Tesis de Licenciatura). Benemérita Universidad Autónoma de Puebla. Recuperado de https://repositorioinstitucional.buap.mx/bitstreams/8d5307b4-e495-43ae-923d-e8cf7cf8949c/download

[5] Zivanov, S. (2023, 3 de agosto). *Paging in Operating Systems: What it Is & How it Works*. PhoenixNAP Knowledge Base. Recuperado de https://phoenixnap.com/kb/paging phoenixNAP | Global IT Services