UNIVERSIDAD DE PANAMÁ
CENTRO REGIONAL UNIVERSITARIO DE VERAGUAS
FACULTAD DE INFORMÁTICA, ELECTRÓNICA Y COMUNICACIÓN

MONOGRAFÍA:

LA MULTIPROGRAMACIÓN COMO HERRAMIENTA PARA IMPLEMENTAR LA MEMORIA VIRTUAL

PRESENTA:

RAÚL ENRIQUE DUTARI DUTARI

2009

TABLA DE CONTENIDOS

1. OBSERVACIONES PRELIMINARES.

A lo largo del tiempo, la cantidad de memoria principal presente en un sistema se ha incrementado notablemente, pero el tamaño y la complejidad de los programas han crecido con mayor velocidad que la memoria. Esto ha obligado a estudiar la manera de ampliar la memoria sin que esto sea excesivamente oneroso para los usuarios, en términos económicos.

Por otro lado, en los sistemas de computadores se acostumbra compartir la memoria principal entre varios usuarios, por lo tanto debe existir un modo de proteger y de compartir la memoria.

Para optimizar el uso de la memoria hay que fijar la organización y la administración de la memoria de manera que se consigan:

- **Máximo número de procesos en la memoria:** Para evitar que la CPU esté inactiva.

- **Protección de la memoria:** Para evitar que al ejecutarse un proceso acceda a zonas de memoria que pertenecen a otro proceso.

- **Compartición de la memoria:** Para permitir que procesos diferentes compartan información cuando sea conveniente.

Esta monografía se enfoca en analizar la manera de optimizar el uso de la memoria del sistema computador para alcanzar el primero de los objetivos.

2. GESTIÓN DE LA MEMORIA REAL.

La memoria real también se denomina memoria principal o memoria primaria. Los programas y los datos deben estar almacenados en la memoria principal para poder ejecutarse, de acuerdo al modelo de Von Newman. Desde el punto de vista lógico es un solo bloque de posiciones contiguas, tal como se aprecia en la siguiente ilustración.

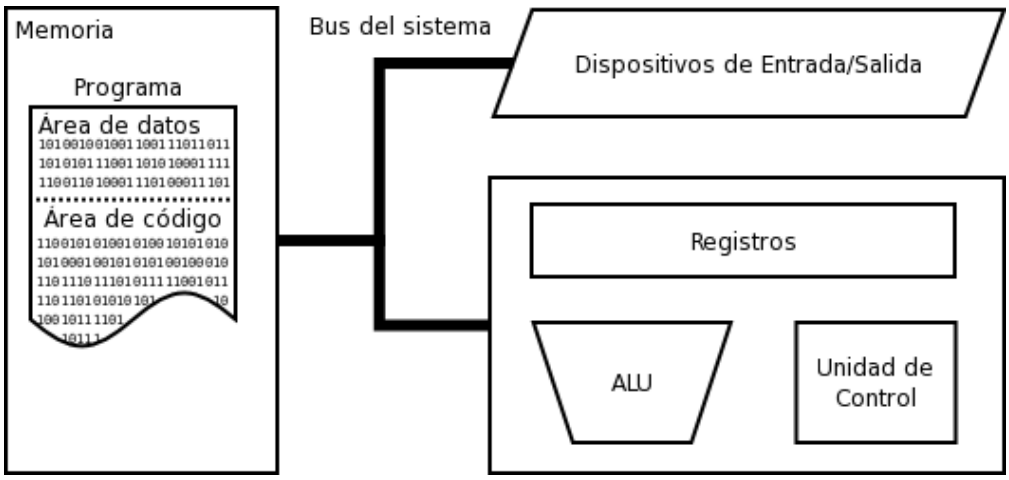

Ilustración 1: Arquitectura de Von Neumann para el computador moderno.

3. GESTIÓN PARA UN USUARIO.

Como la memoria era muy pequeña, surgía el problema de que hacer con los programas excesivamente largos. Este último problema se solucionó dividiendo el programa en segmentos claramente definidos, de manera que solo se cargaban en la memoria los segmentos en ejecución. A cada segmento se le llamaba recubrimiento (overlay). Esta era una tarea que debía realizar el programador de forma manual, cuidadosa, lenta y altamente propensa a los errores humanos, tal como se aprecia en la siguiente ilustración.

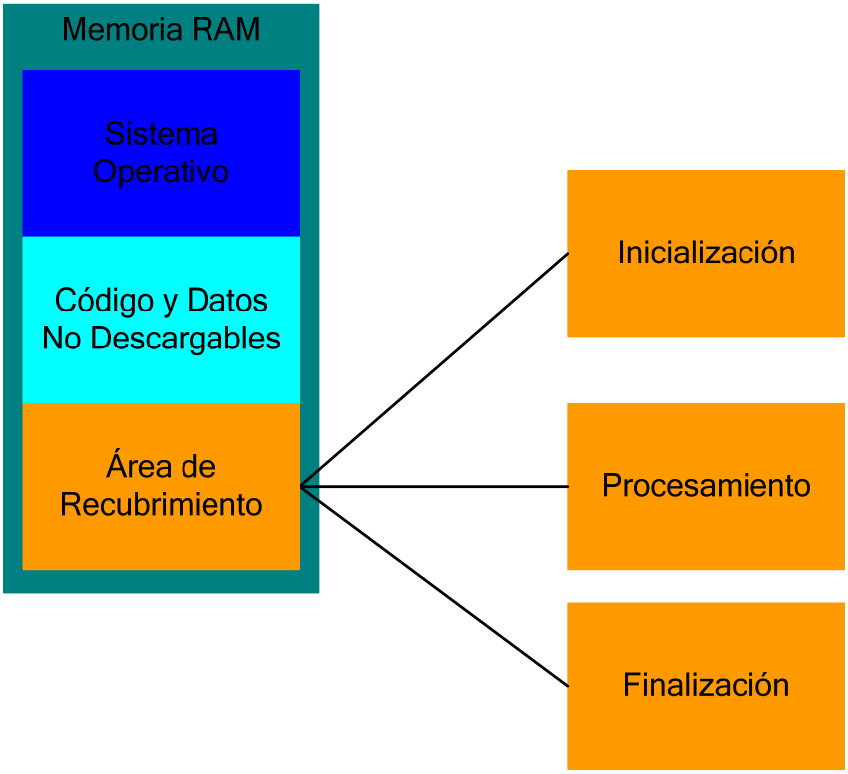

Ilustración 2: Organización de la memoria del computador al utilizar recubrimientos.

Adicionalmente, los primeros sistemas sólo permitían la utilización de la máquina a un único usuario a la vez. En estos sistemas se producía un desaprovechamiento de la CPU en los tiempos en que ella debía esperar a que se produjera una entrada/salida. El problema de la falta de uso de la CPU se solucionó mediante las técnicas de multiprogramación que consisten en que varios usuarios compiten por los recursos del sistema.

4. LA MULTIPROGRAMACIÓN.

Consiste en que: el proceso que está esperando una operación de Entrada/Salida cederá la CPU a otro proceso que esté listo para realizar los cálculos. Para aprovechar al máximo las ventajas de la multiprogramación es

necesario que haya varios procesos en la memoria principal, es decir la memoria se reparte entre varios procesos.

Existen varios tipos de multiprogramación, a saber: partición fija, la partición variable y swapping.

4.1. MULTIPROGRAMACIÓN DE PARTICIÓN FIJA.

La memoria principal (MP) se encuentra dividida en partes de tamaño fijo aunque no del mismo tamaño. Existen dos políticas de carga de los trabajos, carga absoluta y carga reubicable.

4.1.1. CARGA ABSOLUTA.

La memoria principal se divide en regiones de límites fijos (particiones). Un proceso se coloca completo dentro de una partición – de manera que cada partición contiene sólo un proceso - y siempre en la misma partición. Esta gestión la realiza el programador. Esta política puede dar lugar a diferentes problemas:

- **El proceso es mayor que la partición:** Una solución al problema es la técnica del recubrimiento.

- **El proceso es menor que la partición:** Se produce la fragmentación, es decir se produce un uso ineficaz de la memoria porque se desperdicia espacio de la partición.

- **Los procesos ocupan siempre las mismas posiciones de memoria:** por lo tanto si un trabajo está listo para su ejecución y su partición está

ocupada, ese trabajo debe esperar. Una solución al problema es la carga reubicable.

La ilustración que se presenta a continuación, muestra la implementación de la política en mención.

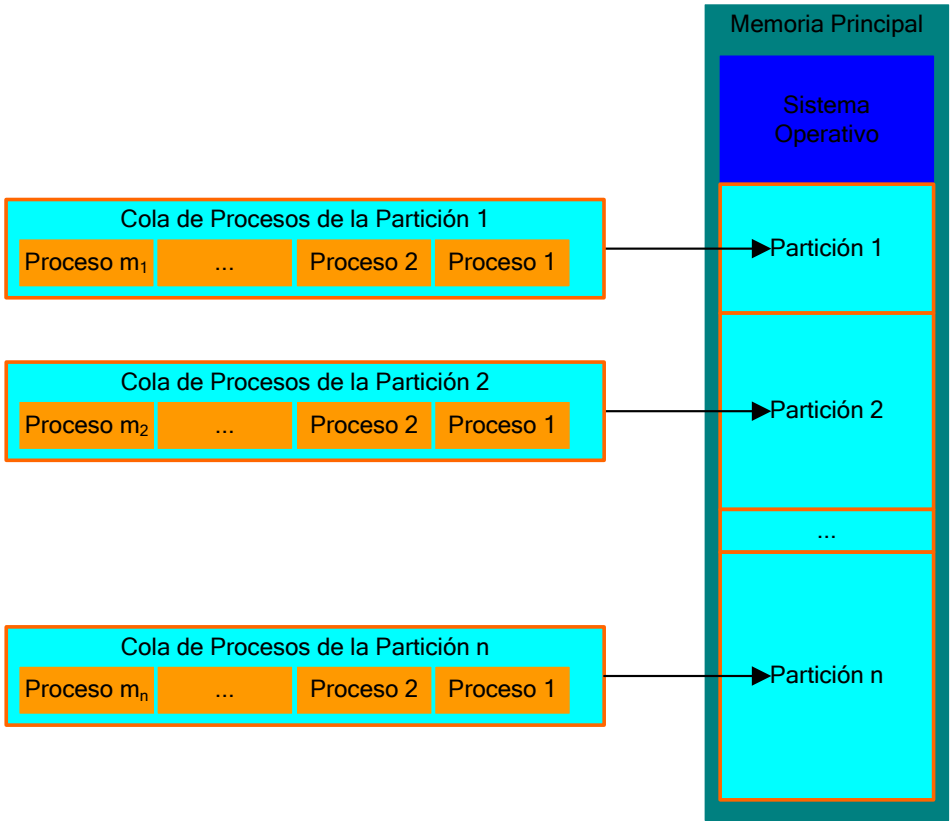

Ilustración 3: Organización los procesos en memoria del computador, según la política de carga absoluta.

4.1.2. CARGA REUBICABLE.

Los procesos se pueden ejecutar en cualquiera de las particiones que se encuentren disponibles en ese instante y en las que pueda acomodarse

completo. Aparece el concepto de dirección lógica y dirección física que obliga a introducir los mecanismos de traducción de direcciones.

La ilustración que se presenta a continuación, muestra la implementación de la política en mención.

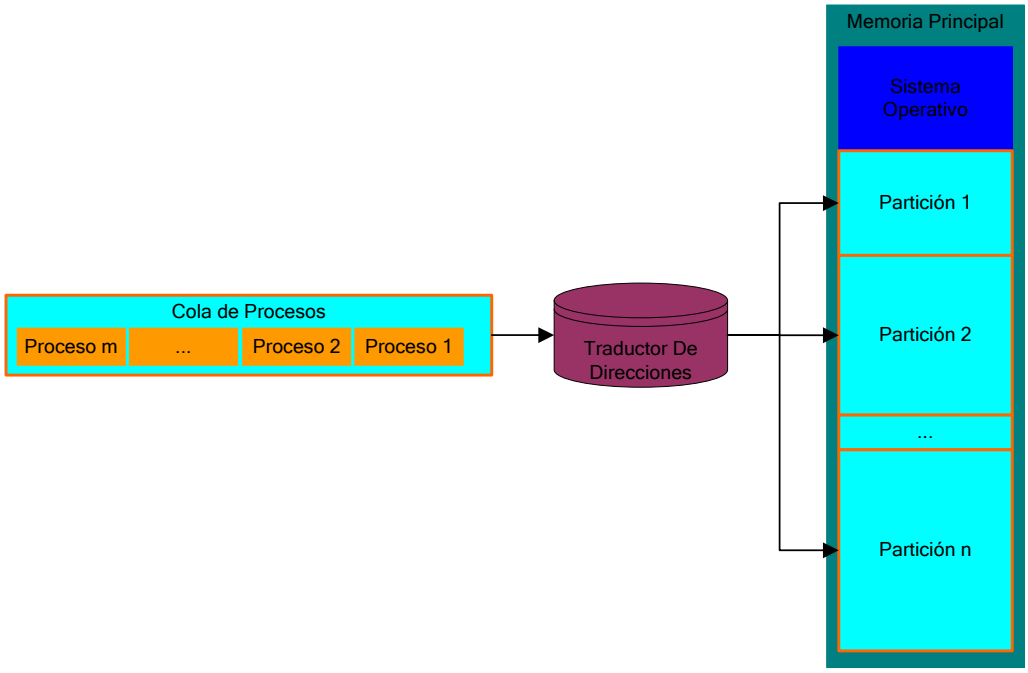

Ilustración 4: Organización los procesos en memoria del computador, según la política de carga reutilizable.

Esta política tiene como principales inconvenientes: la fragmentación interna y que el número de procesos en la memoria es fijo, es decir habrá tantos procesos en la memoria como particiones. Esto puede provocar la parada del procesador si todos los trabajos están en la cola de bloqueados. Sus ventajas son: ser sencillo de implementar y producir poca sobrecarga de sistema operativo.

El principal problema de la multiprogramación de partición fija, es que en ambas políticas aparece el problema de la fragmentación. En realidad este problema se

da en todos los sistemas, independientemente de la organización de memoria seleccionada. En la partición fija ocurre cuando los trabajos no llenan completamente la partición o cuando la partición es demasiado pequeña para los trabajos y se queda vacía.

4.2. MULTIPROGRAMACIÓN DE PARTICIÓN VARIABLE.

Aparece como una solución a la los problemas que presenta la multiprogramación de partición fija. Ahora las particiones se crean dinámicamente de manera que los procesos ocupan todo el espacio de memoria que necesitan. Este sistema tiene la ventaja de que inicialmente optimizan el uso de la memoria.

Los problemas aparecen cuando van finalizando los procesos iniciales, porque aparecen espacios libres en la memoria. La tendencia es que cada vez haya más espacios libres y más pequeños. Estos espacios son externos a las particiones en uso, por eso al problema se le llama fragmentación externa.

Entre las ventajas de la multiprogramación con particiones variables se encuentra que desaparece la fragmentación interna y que por lo tanto se hace un uso más eficiente de la memoria.

Por otro lado, entre las desventajas se puede citar la ineficiencia del procesador debido a la necesidad de compactar la memoria para evitar una fragmentación externa tan elevada que los procesos de cierta entidad no se pudieran ubicar en la memoria.

La ilustración que se presenta a continuación, muestra la implementación de la política en mención, así como el problema de fragmentación antes señalado.

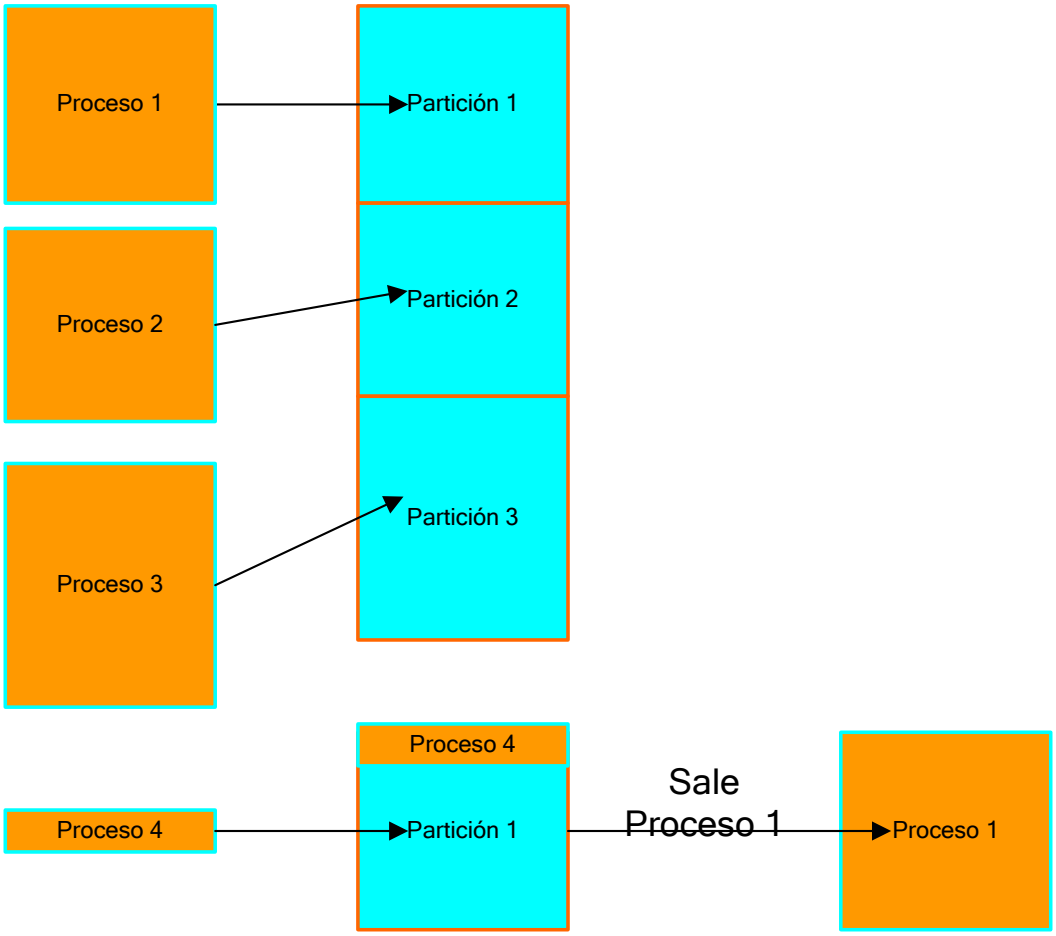

Ilustración 5: Organización los procesos en memoria del computador, según el modelo de particiones variables.

4.3. SOLUCIONES A LA FRAGMENTACIÓN: AGRUPACIÓN Y COMPACTACIÓN.

Se han diseñado dos políticas que minimizan el impacto que provocan los problemas de fragmentación en la memoria principal, a saber se conocen como:

- **Agrupación:** se agrupan espacios de memoria contiguos para así disponer de espacios de memoria libres más grandes, tal como se aprecia en la siguiente ilustración.

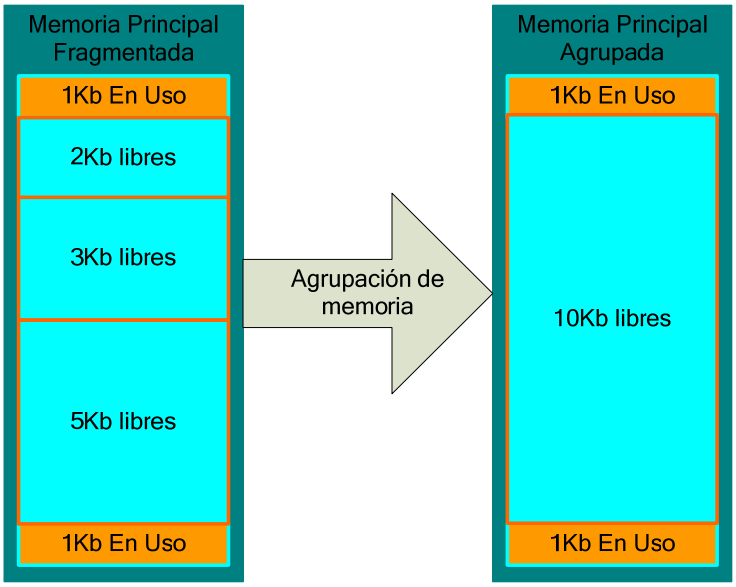

Ilustración 6: Agrupación de memoria fragmentada.

- **Compactación:** se trasladan todos los espacios libres a uno de los extremos del almacenamiento principal, tal como se aprecia en la siguiente ilustración.

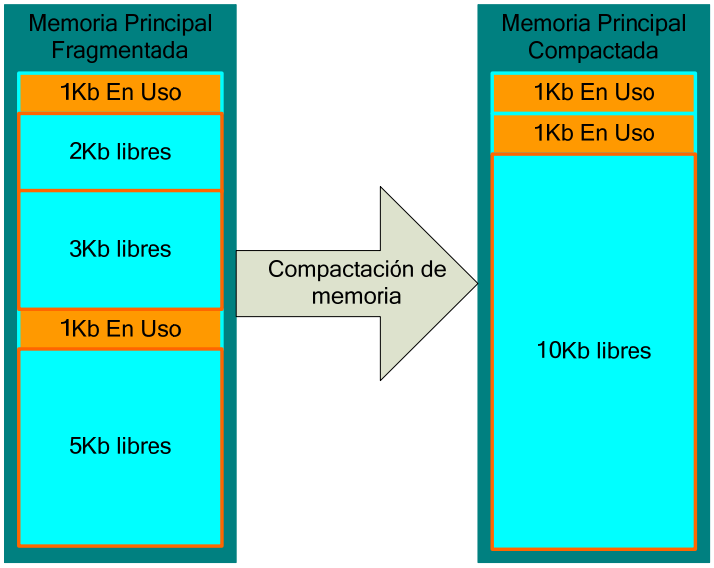

Ilustración 7: Compactación de memoria fragmentada.

Ambas soluciones presentan efectos colaterales que se pueden considerar como desventajas, ya que:

- Consumen recursos del sistema
- Se deben detener todos los procesos mientras se compactan o se agrupan los bloques de memoria libre
- Si el tamaño de los procesos es pequeño, las operaciones de compactación o agrupamiento se deben realizar con demasiada frecuencia.

4.4. ESTRATEGIAS DE UBICACIÓN DE PROCESOS EN LA MULTIPROGRAMACIÓN.

Las estrategias de ubicación son las políticas que se siguen para decidir la ubicación de memoria donde se colocará un proceso, a nivel de los sistemas operativos. Esencialmente, son tres:

- **Utilizar el primer espacio disponible:** En primer lugar, ubica la memoria del sistema operativo y a continuación la memoria que requiere el proceso en el primer espacio disponible en el que pueda colocarse completa. El sistema operativo necesita una lista, no necesariamente ordenada de espacios libres para implementar esta estrategia. Su principal ventaja es la rapidez de decisión.
- **El mejor ajuste:** Coloca el trabajo en el espacio que se ajuste mejor, con lo que se evita el desperdicio de memoria. Para implementarla, el sistema operativo debe disponer de una lista de espacios libres en la memoria ordenada ascendentemente, por tamaño.

- **El peor ajuste:** El proceso se coloca en el espacio libre más grande, con lo que se consigue que el espacio sobrante sea también amplio. Para implementarla el sistema operativo necesita de una lista de espacios libres en la memoria ordenada descendentemente, por tamaño.

4.5. PAGINACIÓN Y SEGMENTACIÓN SIMPLES.

Estas dos estrategias de administración de memoria, no se han aplicado en la organización de la memoria real nunca, pero se puntualizarán, al menos, por ser la base de las estrategias de memoria virtual que serán consideradas en las siguientes secciones.

4.5.1. PAGINACIÓN SIMPLE.

Se caracteriza por:

- La memoria se divide en marcos de igual tamaño.
- Los procesos se dividen en páginas de igual tamaño que los marcos.
- Se carga todo el proceso en memoria, pero las paginas se sitúan en marcos no necesariamente contiguos con lo que se soluciona la fragmentación externa.
- La fragmentación interna en la última página del proceso es mínima.
- La dirección lógica se compone de nº de página y desplazamiento.
- Implementa una tabla de páginas, como mecanismo de correspondencia entra la página del proceso y el marco de la memoria.

- Es una estrategia trasparente para los programadores.

4.5.2. SEGMENTACIÓN SIMPLE.

Se caracteriza por:

- El Programa se divide en segmentos de tamaño variable que se cargan dinámicamente en memoria.
- Los segmentos no tienen que estar contiguos en memoria.
- La dirección lógica de una palabra viene dada por nº de segmento más un desplazamiento.
- Por lo tanto se requiere una tabla de segmentos que haga la equivalencia entre el segmento lógico y el segmento que este ocupa en la memoria.
- No tiene fragmentación interna y la fragmentación externa es pequeña.
- Visible comodidad para organizar programas y datos.

4.6. MULTIPROGRAMACIÓN CON INTERCAMBIO (SWAPPING).

En los esquemas vistos hasta el momento, los programas del usuario permanecen en la memoria principal hasta su terminación. Con la estrategia denominada swapping esto ya no sucede.

Ahora un proceso se ejecuta hasta que no puede continuar. En este punto, dicho programa “cede” el almacenamiento y la CPU al siguiente trabajo.

Un proceso que espera una operación de entrada / salida (E/S) puede enviarse a la memoria secundaria. Un proceso se puede intercambiar varias veces entre la memoria secundaria y la principal, hasta su terminación. En este caso el intercambio es de procesos completos, no de páginas.

La combinación de este método con la paginación simple da lugar a lo que se conoce como la memoria virtual, que es básica en todas las implementaciones de sistemas operativos modernos.

La ilustración que se presenta a continuación, muestra a grandes rasgos, la implementación de la política en mención.

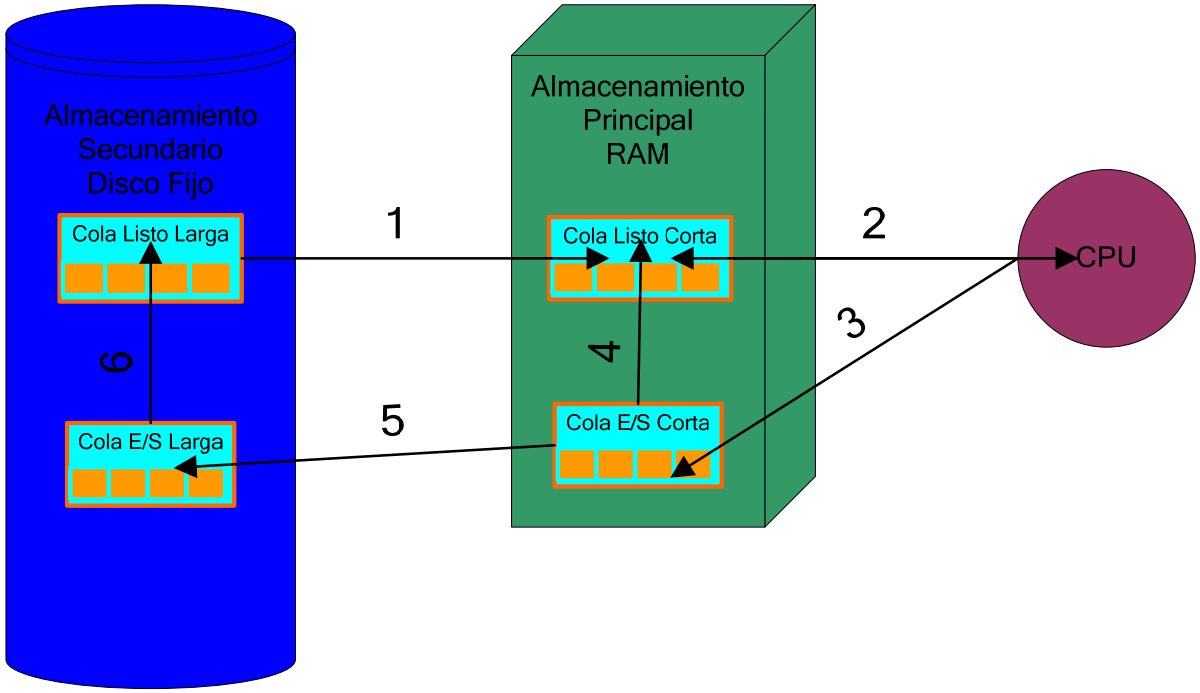

Ilustración 8: Organización los procesos en memoria del computador, según el modelo de swaping.

El sistema operativo implementa el swapping mediante 4 colas:

- **La cola listo larga:** Esta cola se almacena en el disco duro. Cuando el procesador necesita un nuevo proceso lo toma de esta cola.

- **La cola listo corta:** que se almacena en la memoria principal y en ella se encuentran los procesos listos.

- **La cola de E/S corta:** en la que se encuentran los procesos que esperan una operación de E/S.

- **La cola de E/S larga:** en donde se encuentran los procesos que han pasado demasiado tiempo en la cola de E/S corta.

En general, el swaping se realiza como se describe a continuación.

Se van tomando procesos de la cola listo larga, que se introducen posteriormente en la cola listo corta (1), y son eventualmente ejecutados en el CPI (2). Estos procesos se intercambian entre la cola de listos corta y el CPU hasta que finalizan, o se bloquean en una solicitud de E/S, pasando a la cola de E/S corta (3). En caso que la solicitud de E/S se resuelva rápidamente, el proceso pasa nuevamente a la cola de listos corta (4), y si eventualmente, se tarda demasiado tiempo esperando, entonces se envía a la cola de E/S larga (5), para que al resolverse su solicitud pase a la cola listo larga (6), completando el ciclo de vida de los procesos.

5. COMENTARIOS FINALES.

El objetivo de la memoria virtual es que el programador disponga de un nivel de memoria principal mayor que la realmente tiene el sistema. Para conseguirlo se

han creado una serie de mecanismos y políticas nuevas, ya que: El procesador genera direcciones virtuales con un tamaño superior a las direcciones reales de acceso a la memoria física. Por lo tanto, las direcciones virtuales deben convertirse en direcciones reales mientras el programa está en ejecución. Esta función se realiza mediante mecanismos de traducción dinámica de direcciones (DAT) que deben ser tan rápido como sea posible para evitar la degradación del rendimiento del sistema. Esta traducción de las direcciones permite implementar la reubicación que facilita la carga de los programas.

Este mecanismo adolece del siguiente problema, si esta traducción se hiciera palabra a palabra, la información de mapa sería tan voluminosa que requeriría tanto o más almacenamiento real que los propios procesos. Por eso, el programa se subdivide en bloques y la correspondencia con la memoria principal se realiza por bloques. Cuanto mayor sea el tamaño de los bloques, menos información de mapeo se necesitará, pero menos procesos activos entraran en la memoria.

Por otro lado, como el espacio de direcciones virtuales es mayor que el espacio de direcciones real, parte del proceso debe estar almacenado en memoria principal y parte debe estar en la memoria secundaria. Luego para ejecutar un proceso íntegramente debe aparecer un flujo de bloques entre la memoria principal y la secundaria. Esto también lleva a pensar que la dirección virtual unas veces se traduce a una dirección de memoria principal (cuando el bloque está en memoria principal) y otras a una dirección de memoria secundaria.

El intercambio de bloques entre la memoria principal y la secundaria plantea problemas de gestión que se solucionan mediante las estrategias de ubicación, reemplazo y búsqueda.

Todo lo antes mencionado, conduce a que los mecanismos de implementación de la memoria virtual son un tema que está en constante proceso de investigación científica, con el objeto de mejorar las prestaciones de los computadores.

6. CONCLUSIONES.

- Para un aprovechamiento eficiente del procesador y de los servicios de E/S es conveniente mantener tantos procesos en memoria principal como sea posible. Además, conviene liberar a los programadores de las limitaciones de tamaño en el desarrollo de programas.

- La forma de abordar ambos problemas es por medio de la memoria virtual. Con memoria virtual, todas las referencias a direcciones son referencias lógicas que se traducen a direcciones reales durante la ejecución. Esto permite a los procesos situarse en cualquier posición de memoria principal y cambiar de ubicación a lo largo del tiempo.

- Los dos enfoques básicos de memoria virtual son la paginación y la segmentación. Con paginación, cada proceso se divide en páginas de tamaño fijo y relativamente pequeño. La segmentación permite el uso de fragmentos de tamaño variable. También es posible combinar segmentación y paginación en un único esquema de gestión de memoria.

- Los esquemas de gestión de memoria virtual exigen soporte tanto de hardware como de software. El soporte de hardware lo proporciona el procesador. Este soporte incluye la traducción dinámica de direcciones virtuales a direcciones físicas y la generación de interrupciones cuando una página o segmento referenciado no están en memoria principal.

Estas interrupciones activan el software de gestión de memoria del sistema operativo.

7. REFERENCIAS BIBLIOGRÁFICAS.

[HEPA93] **HENNESSY, John; PATTERSON, David.** *Arquitectura de Computadoras, Un enfoque Cuantitativo.* Primera Edición, McGraw-Hill, España, 1993.

[MUEL01] **MUELLER, Scott.** *Manual de actualización y reparación de PC's.* Doceava edición. Pearson, México, 2001.

[STAL06] **STALLINGS, William.** *Organización y Arquitectura de Computadoras. Principios de Estructura y de Funcionamiento.* Séptima Edición, Pearson Prentice-Hall, España, 2006.

[STAL98] **STALLINGS, William.** *Sistemas Operativos.* Segunda Edición, Pearson Prentice-Hall, España, 1998.

[TANEA92] **TANENBAUM, Andrew S.** *Organización De Computadoras: Un Enfoque Estructurado.* Tercera Edición, Prentice-Hall, México, 1992.

[TANEB92] **TANENBAUM, Andrew S.** *Sistemas Operativos Modernos.* Prentice-Hall, México, 1992.

[WIKIA09] **FUNDACIÓN WIKIMEDIA.** *Arquitectura de von Neumann.* Wikimedia Foundation, Inc. Fecha de Actualización: 2009-mayo-13. Fecha de Consulta: 2009-agosto-01. Disponible en: http://es.wikipedia.org/wiki/Arquitectura_de_von_Neumann.

[WIKIB09] **FUNDACIÓN WIKIMEDIA.** *Memoria virtual.* Wikimedia Foundation, Inc. Fecha de Actualización: 2009-mayo-13. Fecha de Consulta: 2009-julio-07. Disponible en: http://es.wikipedia.org/wiki/Memoria_virtual.