Revista internacional de métodos numéricos para cálculo y diseño en ingeniería. Vol. 3, 4, 429-443 (1987)

SIMVIR: UN SIMULADOR DE MEMORIA VIRTUAL PARA MICROORDENADORES

MANUEL DOBLARE

E.T.S. Ingenieros Industriales. Universidad de Zaragoza

RESUMEN

Se presenta en este artículo la descripción, fundamentos y forma de utilización de un conjunto de subrutinas, escritas en FORTRAN 77, cuya misión es la de simular el efecto de memoria virtual de grandes ordenadores en aquellos que no la posean, esencialmente microordenadores. El objetivo fundamental es olvidar las limitaciones de memoria principal que se presentan en programas de gran tamaño, mediante una gestión automática y transparente al usuario de la memoria auxiliar, a través de ficheros de acceso directo.

SUMMARY

In this paper the description, basis and user manual of a virtual memory emulator is presented. It is compound by a set of subroutines, written in FORTRAN 77, which allow to simulate the effect of a virtual memory package in those computers that do not include it, especially microcomputers. The main objective is to forget the limitation of "in core" memory, which appears in medium to large size programas, by an automatic and user independent management of the auxiliary memory, through direct access files.

INTRODUCCION

La rápida extensión de los microordenadores en el mercado industrial ha hecho necesaria la adaptación de muchas de las técnicas inicialmente desarrolladas para grandes ordenadores a este nuevo ente informático. La velocísima caida de los precios de este hardware y con ello del segundo de C.P.U. ha hecho que el tiempo de ejecución no sea ya, al menos en muchas de las aplicaciones, una de las limitaciones principales en la utilización de estos equipos, incluso en programas de grandes proporciones.

No ocurre igual con el apartado de memoria, si bien es cierto que las disponibilidades en este sentido se han ido incrementando progresivamente, todavía existe una enorme variedad de equipos muy extendidos que poseen serias limitaciones a este respecto, ya sea por el hardware inicial ofertado o lo que es aún más importante por el sistema operativo que se incluye. Este efecto se ve agravado, lógicamente, cuando se pretende resolver un problema de gran tamaño, o bien utilizar un programa inicialmente desarrollado para grandes ordenadores con el mínimo de cambios posibles.

Recibido: Abril 1987

 ISSN 0213-1315

La solución es clara, la utilización de memoria auxiliar. Sin embargo, en este caso la programación se hace más engorrosa, se pierde portabilidad y sobre todo es necesario pensar en cada programa específico la mejor forma de resolver el problema de memoria planteado.

En grandes ordenadores, el problema suele estar resuelto mediante la integración de técnicas de gestión automática de la memoria auxiliar a través de sistemas de memoria virtual que permiten trabajar en la práctica como si se dispusiera de una memoria central ilimitada, naturalmente con el consiguiente incremento del tiempo de ejecución.

En este artículo se presenta un conjunto de rutinas escritas en FORTRAN 77 que simulan el efecto de memoria virtual en microordenadores. Este sistema posee las ventajas siguientes:

i) Un tratamiento general y estandarizado de los conjuntos que no quepan en memoria principal y sea necesario por tanto su inclusión, al menos parcialmente, en fichero.

ii) Una mayor portabilidad, es decir un paso casi inmediato de programas realizados en un ordenador a otro que disponga asimismo de memoria virtual. Esto es especialmente importante en el paso de programas desde microordenadores con limitaciones de memoria a macroordenadores con memoria suficiente, ya sea como consecuencia de una memoria central suficientemente grande o por incluir tratamiento con memoria virtual. Esto es de especial utilidad en los procesos, cada día más comunes, de elaborar un programa en microordenador para posteriormente resolver problemas de entidad en ordenadores de mayores prestaciones, con el consiguiente ahorro económico y de disponibilidad durante el período de desarrollo del programa.

iii) Una programación mucho más simple y estandarizada.

iv) Finalmente también es posible este sistema como subrutinas de almacenamiento de conjuntos en memoria auxiliar, en la forma tradicional, pero con una programación y sistematización mayor.

La desventaja esencial se encuentra en el incremento de tiempo que supone su utilización. Sin embargo, no nos ha preocupado sustancialmente este aspecto por varias razones. En primer lugar cada día más los ordenadores son más rápidos en sus accesos a fichero, sobre todo por la aparición de sistemas de "hard disk" con una velocidad mucho mayor y prácticamente generalizados a cualquier microordenador. En segundo lugar también es cierto que en el caso de microordenadores el tiempo de ordenador es mucho más barato que el tiempo de programación, que sin duda se reduce drásticamente mediante la utilización de este sistema.

Esta librería consta de 11 subrutinas y 3 funciones, de las cuales tan solo 6 subrutinas y 1 función son utilizadas en un programa general, siendo el resto de ellas subrutinas de segundo nivel a las que accede tan solo el programador que mantiene la librería. Las funciones que realizan las subrutinas y funciones accesibles (a lo largo de este texto se sigue esta denominación para distinguirlas) son las siguientes:

a) Iniciación del área de memoria que controla el proceso.

b) Finalización del proceso, cerrando los ficheros con la situación actual de los conjuntos.

c) Inclusión de un nuevo conjunto para ser manejado por SIMVIR.

d) Eliminación de ese conjunto del sistema.

e) Puesta a cero de un conjunto completo.

f) Transferencia de un conjunto completo a otro.

t) Localización de un elemento cualquiera de dicho conjunto.

A continuación pasamos a describir los fundamentos del conjunto, y en los epígrafes siguientes se incluye una descripción breve de cada una de las subrutinas y algunos ejemplos de aplicación.

La biblioteca SIMVIR está escrita en FORTRAN 77 e implementada en microordenadores con sistema operativo MS-DOS y FORTRAN de Microsoft standard. En cualquier caso es conveniente indicar que esta biblioteca hace uso extensivo del conjunto de subrutinas de ayuda general, incluidos en la biblioteca LIBGEN[1]. El conjunto de subrutinas objeto ocupa un total de 27 Kbytes.

BASES DE LA BIBLIOTECA LIBVIR

La idea fundamental de la memoria virtual es establecer una gestión automática de la memoria auxiliar, de forma que el programador no conozca en ningún momento qué parte de cada conjunto se encuentra actualmente en memoria central o auxiliar, manejando dicho conjunto como si todo él estuviese en memoria central. Ello se consigue generalmente mediante el establecimiento de un proceso de paginación.

Denominaremos **página** a un subconjunto de elementos de un conjunto, generalmente seguidos en la ordenación del lenguaje de programación utilizado, es decir por columnas en FORTRAN, de forma tal que en cada momento se encuentra en memoria central una (o varias dependiendo de la memoria disponible) página de cada conjunto, que se va utilizando mientras que los elementos necesarios se encuentren en ella. Cuando ello no ocurre se identifica la página del conjunto donde se encuentra el nuevo elemento a utilizar y se cambia la página en memoria por esta nueva, imprimiendo en fichero el estado actual de aquella que ha salido de la misma.

La idea global de SIMVIR es la misma, es decir establecer un conjunto de subrutinas que gestionen de forma automática el proceso de paginación. Sin embargo, de lo anterior surgen una serie de preguntas a responder.

1. **¿Cuántas páginas se mantendrán en memoria para un conjunto?** Es obvio que mientras menos páginas existan mayores podrán ser éstas y por tanto el número de accesos a fichero será menor. Sin embargo, depende en gran medida de la forma de utilización de cada conjunto. Efectivamente, si se tiene un conjunto de utilización quasisecuencial es obvio que la inclusión de una sola página garantiza el menor número de accesos a fichero. En cambio si simultáneamente se está utilizando también de forma secuencial dos zonas distintas del conjunto es también fácil comprender que la inclusión de dos páginas disminuirá el número de accesos a fichero, objetivo fundamental del sistema. Es por ello que **se ha optado por un sistema de número de páginas variables para cada conjunto** que se definen en el momento de "crearlo", es decir de incluirlo en el proceso. Realmente, más que páginas podríamos denominarlas "**áreas buffer**", ya que en cada una de ellas pueden incluirse distintas páginas como se indica en el párrafo siguiente.

Finalmente tan solo indicar que este número está limitado en SIMVIR a 5, número que en la práctica no se suele nunca alcanzar, manteniendose entre 1 y 3.

2. **¿Cuál debe ser el tamaño de cada página?** La respuesta más simple e inmediata es decir: un tamaño fijo para todas ellas igual al área de memoria central disponible dividido por el número de páginas máximo que se gestionen simultáneamente a lo largo del programa. Sin embargo esta respuesta si bien es la más cómoda de programar no es la mejor. Efectivamente, si se utiliza este proceso aparecen las siguientes desventajas:

a) Existirán momentos en los que una parte de la memoria central esté libre y por tanto desaprovechada, con la disminución innecesaria del numero de páginas residentes simultáneamente en memoria o del tamaño de las mismas, y con ello el incremento del número de accesos a fichero.

b) Dependiendo del tamaño de la fila o columna de cada conjunto, existirá parte de la página desaprovechada por no incluir un trozo de fila o columna, o bien será necesario incluir una gestión mucho más compleja que incluya la posibilidad de incluir trozos de fila o columna.

La respuesta pues a esta pregunta parece inicialmente obvia, es decir el tamaño de cada página ha de ser distinto para cada conjunto, e igual a un número determinado fijo de filas o columnas. Se ha optado en definitiva por la posibilidad de que **el usuario defina el tamaño de la página en función del número de filas o columnas que quiera incluir en la misma.** Por supuesto si este número es tan grande que no cabe en memoria se da un mensaje de error. Esta respuesta nos lleva inmediatamente a una conclusión; **cada conjunto debe almacenarse en un fichero distinto,** ya que es conveniente hacer coincidir el tamaño de página con el del registro del fichero (además de otra serie de consideraciones menores que avalan el utilizar ficheros distintos, fundamentalmente la flexibilidad de su uso posterior), y también conduce a una nueva pregunta:

3. **¿Cuál es el número de páginas que se debe incluir en cada área buffer?** De nuevo podría responderse que un número igual para cada conjunto. Sin embargo, si existen conjuntos utilizándose simultáneamente con tamaños muy distintos de páginas, puede ocurrir que quede un trozo de memoria disponible para incluir una o varias páginas más del menor y sin embargo no poderse incluir ninguna del mayor desaprovechandose de nuevo parte de la memoria. Se ha optado pues por un número de páginas diferentes para cada conjunto, de forma que se ocupe totalmente la memoria disponible (al menos hasta que no quepa ninguna página adicional de los conjuntos gestionados en ese instante).

De cualquier forma, por simplicidad, y sobre todo por no disponerse de criterios que prueben una mejor alternativa, **se ha optado por incluir el mismo número de páginas para cada área buffer de un conjunto determinado.**

Todo este proceso de obtención del número de páginas por área buffer de cada conjunto, se realiza automáticamente sin intervención del usuario y siempre, como se ha indicado, tratando de ocupar el máximo espacio posible de memoria disponible. Es de resaltar que la modificación de este número tan solo se realiza cuando se incluye o elimina un conjunto, cosa que no es muy frecuente a lo largo de un programa, ya que el número de conjuntos que se gestionan mediante este sistema suele ser pequeño. Es por ello que no se considera importante en la evaluación del tiempo global de gestión el invertido en este proceso de modificación del número de páginas por área buffer, permitiendo, por el contrario, una ocupación mayor de la memoria central y con ello una disminución de accesos a fichero, que sí es importante a la hora de considerar tiempos.

4. **¿Qué área buffer debe salir de memoria central cuando ha de incluirse en ella una página que previamente no se encontraba?** La respuesta en este caso es también variada, ya que depende de la forma de utilización de cada conjunto, el hecho de que sea mejor una forma de actuación u otra. En general son dos las formas principales de actuación:

a) Sale el área buffer que más tiempo lleva en memoria, es decir el proceso habitualmente denominado **"FIFO"** (First In First Out).

b) Sale el área buffer que menos veces se ha utilizado durante el proceso, lo que podríamos denominar **"LUFO"** (Less Used First Out).

Como se ha indicado depende de la forma de utilización de un conjunto el hecho de que sea mejor una u otra forma. En nuestro caso se ha optado por definir para cada conjunto la forma en que se pretende utilizar, a la hora de crearlo, mediante un parámetro adecuado que se almacena en el área de control adscrita a dicho conjunto.

5. **¿En caso de caber en un momento determinado una página más de varios conjuntos en qué orden se van rellenando?** La respuesta inmediata podría ser: en el orden de cración. Sin embargo no siempre tiene que ser esta la mejor solución, ya que existen conjuntos con dimensiones de página parecidas que, sin embargo, se utilizan mucho más unos que otros. Sería conveniente entonces establecer una **ordenación de prioridades** entre los conjuntos que se gestionarán con este sistema. Ello se realiza en el momento de la creación del conjunto.

Todas estas respuestas se han incluido en SIMVIR, dotando al conjunto de una gran flexibilidad y generalidad, de forma tal que cada conjunto puede en todo momento almacenarse y gestionarse en la forma más idónea según sea su utilización.

ALMACENAMIENTO Y GESTION DE CONJUNTOS MEDIANTE SIMVIR

Todo el proceso se ha gestionado mediante un área de memoria, incluida en un COMMON que denominamos /CIAB/, donde se almacena un único conjunto IAB, entero, de dimensión MEMOB (variable almacenada en el COMMON de constantes generales) igual a la máxima dimensión permitida por las restricciones de memoria central impuestas. En dicha área /CIAB/ se incluye tanto la zona de control del proceso como la reservada para el conjunto de áreas buffer ocupadas por cada conjunto gestionado en cada instante.

El área de control, que se encuentra en las primeras palabras del conjunto /IAB/, está a su vez dividida en dos partes, una primera en la que se incluyen los parámetros de control globales de todo el proceso, y otra zona donde se van incluyendo progresivamente las área de control particulares de cada conjunto gestionado y no eliminado definitivamente. A continuación se van incluyendo las distintas áreas buffer de cada conjunto según el orden de creación del mismo.

El área de control general está formada por 8 palabras enteras (para aumentar la portabilidad se ha utilizado un múltiplo de 4, de forma que sea posible almacenar conjuntos doble precisión. El mismo esquema redondeando a múltiplos de 4 se sigue en los tamaños del resto de conjuntos almacenados en /CIAB/), que se ocupan como sigue

IAB(1) = Tamaño del área de control global (8 en este caso)

IAB(2) = Tamaño del área de control particular de cada conjunto almacenado (40 en este caso)

IAB(3) = Tamaño máximo del área /CIAB/ que denominaremos MEMOB

IAB(4) = Número de ficheros abiertos desde el comienzo del programa

IAB(5) = Número de conjuntos actualmente gestionados en /CIAB/ incluyendo los no activos, es decir los borrados pero no definitivamente.

IAB(6) = Número de conjuntos actualmente activos

IAB(7) = Area desocupada en /CIAB/ en cada instante en palabras enteras

IAB(8) = No usada

En cuando al área de control de cada conjunto está formada por 40 palabras enteras (también múltiplo de 4) distribuidas en la forma siguiente:

1 a 4 – Números enteros que corresponden a los ASCII adscritos a los 4 caracteres que componen el nombre del conjunto (cada conjunto está definido por un nombre de 4 caracteres).

5 y 6 – Número de filas y de columnas del conjunto (solo se admiten conjuntos bidimensionales).

7 – Número ASCII que corresponde a la letra que define el tipo de conjunto (E=ENTERO; R=REAL; C=CARACTER; D=DOBLE PRECISION)

8 – Número ASCII que corresponde a la letra que define si se almacena el conjunto por filas o columnas (F=FILAS; C=COLUMNAS)

9 – Número de buffers del conjunto, NBUF

10 – Número de registros (páginas) por área buffer

11 – Tamaño de registro, igual al tamaño de página

12 – Número total de registros del conjunto

13 – Número del fichero donde se almacena el conjunto

14 – Prioridad relativa del conjunto. Mientras más baja mayor prioridad

15 – Número de filas/columnas por página

16 – Número ASCII que corresponde a la letra que define la forma de entrar o salir las áreas buffer (F=FIFO; L=LUFO)

17 a 16 + NBUF – Posición de comienzo en /CIAB/ de cada buffer

17 + NBUF a 16 + 2*NBUF – Número del registro inicial amacenado en cada buffer en cada momento.

17 + 2*NBUF a 16 + 3*NBUF – Número de la primera fila/columna almacenada en cada buffer en cada momento.

17 + 3*NBUF a 16 +4*NBUF – Número de veces que se ha utilizado cada área buffer desde que se incluyó en /CIAB/ en caso de conjunto LUFO o número que indica la posición en que se incluyó dicha área buffer en /CIAB/ en caso FIFO.

Resto – No ocupadas

El proceso que se sigue cuando se crea un conjunto nuevo es el siguiente

1) Se detecta si ese conjunto ha sido previamente creado. Si es así se chequea si las características de creación coinciden con las nuevas propuestas. En caso afirmativo se

chequea si el conjunto está activo o no. Si lo está se devuelve el control y continúa el proceso sin más cambios. Si no lo está se vuelve activo y se pasa al punto 2). Si las características no coinciden con las previamente establecidas se modifica provisionalmente el nombre del conjunto, siguiendo el proceso hasta el final como si fuera un conjunto nuevo, borrando al final definitivamente el conjunto con las características iniciales. Si el conjunto no está creado o se está en algunas de las situaciones explicadas se pasa al punto 2).

2) Se imprime el estado actual de todos los conjuntos en sus ficheros correspondientes.

3) Se determina el tamaño de registro del conjunto nuevo y con ello el número de páginas que caben en cada área buffer de cada conjunto en la nueva situación, de acuerdo asimismo con las prioridades establecidas.

4) Se modifica el área de control de acuerdo con las nuevas instrucciones.

5) Se pone a cero el fichero donde se incluye el nuevo conjunto (en caso de que se cree por vez primera) o se transfiere desde el fichero en el cual estaba previamente almacenado a un nuevo fichero con las nuevas características definidas (en caso de que estuviera creado y se hayan modificado sus características).

6) Se vuelve a rellenar cada una de las áreas buffer de acuerdo con las instrucciones dadas a partir de la información de control almacenada.

Este proceso se realiza esencialmente con una subrutina denominada CREARF que utiliza otro conjunto de ellas de segundo nivel tal como se indica en el apartado siguiente.

El proceso de eliminación de un conjunto es en todo análogo al anterior, salvo que hay dos formas de eliminar un conjunto: provisionalmente y definitivamente. En la primera forma se deja inactivo pero el fichero sigue disponible y el área de control rellena. La única diferencia es que el número de páginas por área buffer es nulo, es decir no se almacena en memoria ninguna parte del conjunto. Esta forma de eliminar es muy útil cuando se pretende volver a utilizar el conjunto en un momento posterior del programa, sin necesidad de ocupar memoria durante el tiempo que no está siendo utilizado. En la segunda forma se cierra el fichero y se elimina el conjunto del área de control.

La eliminación se realiza con la subrutina BORRAF y el conjunto de subrutinas auxiliares que ella utiliza, esencialmente las mismas que CREARF.

Finalmente la forma de utilizar un conjunto es en todo análoga a como se realiza para un conjunto almacenado en memoria principal. Se chequea automáticamente si el elemento buscado del conjunto se encuentra en memoria o no. Si la respuesta es sí se obtiene la posición de IAB en la que se encuentra y en caso contrario se elige el área buffer que sale, de acuerdo con la definición inicial del conjunto, y se incluye aquella página en la que se encuentra dicho elemento leyendola del fichero correspondiente para, a continuación, incluir las siguientes páginas hasta completar el número asignado a cada área buffer. Finalmente se vuelve a detectar la posición en IAB del elemento, ahora sí en memoria.

Este proceso se realiza con la función IWW que devuelve la posición en /CIAB/ de un determinado elemento de un conjunto, realizando toda la gestión anterior.

A continuación se pasa a describir cada una de las subrutinas y la forma de utilizarlas.

DESCRIPCION DE LAS SUBRUTINAS

Son un total de 11 subrutinas y 3 funciones las que componen el conjunto de esta librería. En lo que sigue se describe brevemente las funciones de cada una de ellas y los parámetros que utilizan.

SUBRUTINA IABIN: Es la subrutina de inicialización del proceso. En ella se inicializa el área de control general. En el caso de que el sistema se encuentre dentro de un segmento de un programa segmentado, en el que se han obtenido y almacenado una serie de resultados previos, recupera la información almacenada hasta este instante y continúa a partir del estado en que se encontraba. Para ello se transfiere como parámetro un nombre de 4 letras FICHO que indica el nombre del fichero en el que se encontraba almacenada el área total /CIAB/ abriendose los ficheros pertinentes. Si se comienza desde el principio (FICHO = 'NULO'), esta subrutina define los parámetros básicos del área de control general, que son los 8 indicados previamente.

SUBRUTINA IABFIN: Es la subrutina de finalización del proceso. Cierra los ficheros en los que se han almacenado los conjuntos, previa actualización de los mismos con el estado último de cada uno de ellos. En caso de que se pretenda continuar posteriormente con la situación actual se imprime en un determinado fichero FICHO la situación última del área /CIAB/ que posteriormente será rescatada si se requiere en IABIN.

SUBRUTINA CREARF: Incluye un nuevo conjunto en el área de trabajo /CIAB/ con las directrices indicadas previamente. Rellena el área de control particular del conjunto, actualiza el área de control global y asimismo el número de páginas que después de la inclusión de este nuevo conjunto, y de acuerdo con las prioridades y tamaños establecidos, es posible incluir en cada área buffer de cada conjunto activo.

SUBRUTINA BORRAF: Elimina un conjunto del área de /CIAB/ manteniendo su área de control o no, de acuerdo con un parámetro de la subrutina. En el primer caso el conjunto se mantiene inactivo pero en cualquier momento puede recuperarse, teniendo hasta ese momento un número de páginas nulo por área buffer. En el segundo caso el conjunto se borra definitivamente no pudiendo volver a utilizarse. Es similar a CREARF en su proceso y utiliza las mismas subrutinas.

SUBRUTINA NRBUF: Es una subrutina de segundo nivel que es utilizada por CREARF y BORRAF. Su función es la de determinar el número de páginas que caben en cada área buffer de cada conjunto, de acuerdo con las prioridades establecidas y con el tamaño de cada página de cada conjunto.

SUBRUTINA PRIOR: Es de tercer nivel, siendo llamada por NRBUF. Es esencialmente un proceso de ordenación de los conjuntos, según la prioridad de cada uno de ellos, incluyendo el nuevo conjunto a crear e inutilizando los conjuntos eliminados o inactivos.

SUBRUTINA NRBUF1: También de tercer nivel y llamada por NRBUF. Calcula el número de páginas por área buffer una vez ordenados los conjuntos según sus prioridades.

SUBRUTINA REINF: Actualiza los elementos del área de control de cada conjunto que se refieren a las áreas buffer (posiciones 17 en adelante), una vez definido el número de páginas a incluir en cada una de ellas (posición 11 que también se actualiza). Es una subrutina de segundo nivel utilizada por CREARF y BORRAF.

SUBRUTINA LEIMF: Lee o imprime en cada momento de los ficheros pertinentes las páginas (registros) que en cada momento se indican en el área de control de cada conjunto. Es la subrutina de actualización del estado de los ficheros o del área de almacenamiento de /CIAB/ una vez actualizada el área de control. Se llama en CREARF y BORRAF.

FUNCION LOCAF: Localiza la primera palabra en /CIAB/ del área de control de un determinado cojunto. Chequea por tanto si dicho conjunto está creado o no. El proceso consiste tan solo en ir chequeando el nombre de cada conjunto almacenado con el propuesto.

SUBRUTINA CEROSF: Pone a cero un conjunto completo, tanto sus áreas buffer como el fichero completo en el que se incluye. Es pues una subrutina de inicialización de conjuntos, siendo llamada tan solo en CREARF.

SUBRUTINA TRANSF: Transfiere el contenido de un conjunto a otro, ambos previamente creados. Se transfiere elemento a elemento sin preocuparse de la forma en que se encuentren almacenados en fichero, tan solo el hecho de que el elemento (I,J) del conjunto al que se transfiere recibirá el elemento (I,J) del conjunto desde el que se transfiere. Se transfieren tantas filas y columnas como posea este último.

FUNCION CHEQUE: Chequea si un conjunto previamente creado posee las mismas características que otro que se pretende volver a crear con el mismo nombre. Se utiliza en CREARF, y tiene utilidad en los casos que se pretende mantener un determinado conjunto pero alterando sus características. Para ello se detecta primero mediante esta función este hecho y a continuación se realiza la transferencia pertinente eliminando el conjunto con la primera situación.

FUNCION IWW: Realiza la labor de búsqueda de un elemento cualquiera de un conjunto. Para ello localiza en /CIAB/ la posición correspondiente a un elemento (I,J) dado, de un determinado conjunto. Si dicho elemento no se encuentra en ninguna de las páginas incluidas en ese momento en /CIAB/, detecta en qué registro del fichero donde se encuentra almacenado dicho conjunto se encuentra el elemento. Se alteran las áreas buffers, de forma que se incluya la nueva página donde se encuentra dicho elemento, imprimiendo previamente en el fichero la situación actual del área buffer que sale de memoria. Finalmente se actualiza el área de control particular incluyendo la situación de las nuevas áreas buffer y el número de accesos a cada una de ellas si el proceso es LUFO.

EJEMPLOS DE UTILIZACION DE SIMVIR

Supongamos que se pretende utilizar mediante este procedimiento tres conjuntos simultáneamente que responderán a los nombres de CON1, CON2 y CON3. El primer proceso será la creación de los mismos, para lo cual previamente ha debido definirse en el programa principal el COMMON /CIAB/ a la dimensión de la que se pueda disponer. Supongamos que esta dimensión es de 5000 palabras enteras, es decir MEMOB=5000. Se tiene entonces en el programa principal y en todas las subrutinas que vayan a hacer uso de estos conjuntos la siguiente definición.

```
COMMON /CIAB/ IAB(5000)
COMMON /CONST/ LECP, LECF, IMPP, IMPF, NBYTE, NPER, NPEDP,
              MEMO, MEMOB, MEMOC
```

De la anterior definición de variables del COMMON /CONST/ tan solo nos interesa en este momento MEMOB, aunque por necesidades de la librería de subrutinas generales LIBGEN[1] que se utiliza en SIMVIR se ha de incluir el resto de variables en el orden indicado. MEMOB será necesario definirla con el valor de 5000 en el programa principal.

A continuación se crean los conjuntos anteriores con las siguientes características:

a) **CON1**: Conjunto de 10 filas y 5 columnas, entero, con prioridad 0 entre los conjuntos a gestionar en /CIAB/, con 3 áreas buffer, 2 columnas en cada registro o página y con entrada-salida de áreas buffer tipo FIFO.

b) **CON2**: Conjunto de 100 filas y 200 columnas, real, con prioridad 1, con 1 área buffer, 10 filas en cada registro y LUFO.

c) **CON3**: Primero se crea con las características siguientes: Conjunto de 50 filas y 1 columna, carácter, con prioridad 2, con 2 áreas buffer, 5 filas por registro, y "FIFO. Posteriormente se modifica de forma que se incluyan 3 áreas buffers con 2 filas por registro y prioridad 0.

A continuación se explica la forma de realizar algunas operaciones típicas y como va quedando el área /CIAB/.

Lo primero será inicializar el área /CIAB/, que supondremos corresponde a un programa nuevo que la va a inicializar por primera vez. La forma de hacerlo es la siguiente:

CALL IABIN('NULO')

Con ello se inicializa el área /CIAB/ de forma que se rellena el área de control general quedando las 7 primeras palabras de IAB en la forma

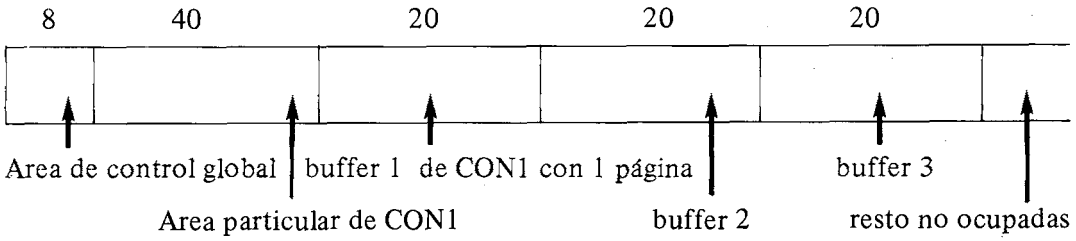
8	40	5000	0	0	0	4992		no ocupadas

A continuación crearemos los tres conjuntos

CALL CREARF('CON1', 10, 5, 'E', 0, 3, 2, 'C', 'F', ILOC1)

nombre, filas, colum, tipo, prior, buffers, pag, alm-col, FIFO

Con la anterior se rellena el área de control particular de este conjunto tal como se indicó previamente y como cabe todo él en /CIAB/ se inicializa a 0 ocupándose 60 palabras de /CIAB/ además de las 40 del área de control particular y las 8 del área de control global. Habrá pues 3 áreas buffers con 1 página cada una de 2 columnas (20 elementos) cada una de ellas, de forma que en la primera área buffer estarán las columnas 1 y 2, en la segunda la 3 y 4 y en la tercera la 5 y 10 elementos desocupados. El aspecto de IAB es el siguiente:



('CON2', 100, 200, 'R', 1, 1, 10, 'F', 'L', ILOC2)

Se rellena el área de control de este nuevo conjunto de forma que ahora el área de control ocupará 8+ 40+40=88 elementos. En este caso no cabe el conjunto completo, así que será necesario proceder a la determinación de cuantas páginas caben por cada buffer. El proceso de determinación es el siguiente, sabiendo que la prioridad de este último conjunto es más elevada que la del primero y por tanto se considera posteriormente. Se tendrá

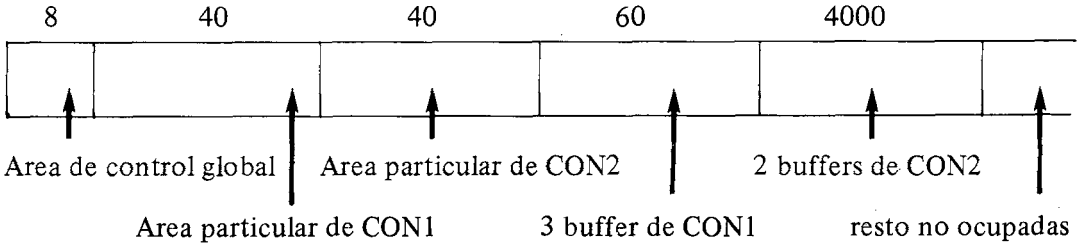
1 página para cada buffer de CON1=60 elementos

1 página por cada buffer de CON2=2000 elementos = 200 elem/fila x 10 filas/pág x 1 área buffer

1 página por cada buffer de CON2=2000 elementos

No cabe una nueva página de CON2 por lo que en definitiva hay 4148 palabras ocupadas y 1 página en cada buffer de CON1 y 2 de CON2.

La situación de IAB es ahora



('CON3', 50, 1, 'C', 2, 2, 5, 'F', 'F', ILOC3)

En este caso el área de control estará formada por 88+40=128 elementos, no cabiendo por supuesto los conjuntos completos. La prioridad de este último conjunto es la más alta por lo que sus áreas buffer se rellenarán las últimas. El proceso de llenado pues es el siguiente:

1 página para cada buffer de CON1=60 elementos

1 página por cada buffer de CON2=2000 elementos

1 página por cada buffer de CON3=10 elementos = 1 elem/fila x 5 filas/pag x 2 buffers

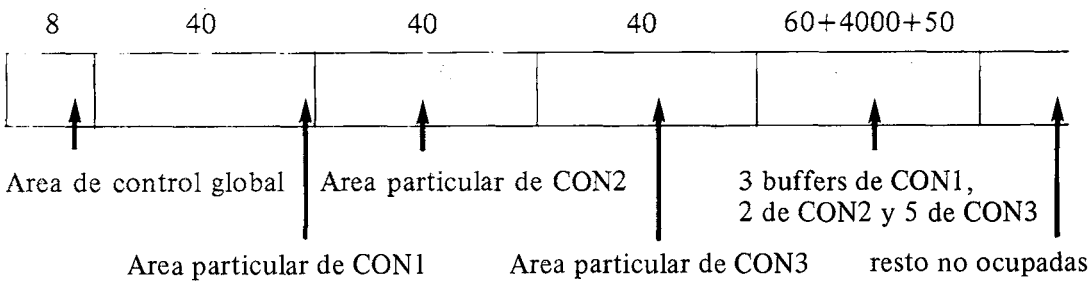
1 página por cada buffer de CON2=2000 elementos

1 página por cada buffer de CON3=10 elementos

No cabe una nueva página de CON2

1 + 1 + 1 páginas por cada buffer de CON3 = 30 elementos

Con ello los conjuntos CON1 y CON3 se encuentran en memoria completos mientras que CON2 tan solo parcialmente. El número de elementos ocupados es de 4238 y la situación de IAB la siguiente:



En este momento todos los conjuntos están a cero, pudiendo utilizarse normalmente. Supongamos que pretendemos poner a 1 la fila 3 de CON1. El proceso una vez creado sería el siguiente.

```
      DO 10 J1 = 1,5
          IAB(IWW('CON1',3,J1))=1
10    CONTINUE
```

es decir totalmente análogo al procedimiento normal de puesta a 1 de un conjunto en memoria principal con la salvedad de incluir la función IWW como detección de la posición del elemento I, J de CON1.

Supongamos que pretendemos transferir la 3ª fila de CON1 a los 5 primeros elementos de la 4ª columna de CON2. Como CON2 es real habrá previamente que definir un conjunto RIAB real que ocupe asimismo /CIAB/. El proceso sería el siguiente:

```
      COMMON /CIAB/ IAB(5000)
      REAL RIAB(1)
      EQUIVALENCE (IAB, RIAB)
      DO 10 J1 = 1,5
          RIAB(IWW('CON2',J1,4)) = IAB(IWW('CON1',3,J1))
10    CONTINUE
```

es decir de nuevo idéntico a un tratamiento en memoria principal. Para poner de nuevo a cero todo el conjunto CON2 se realiza mediante

```
CALL CEROSF('CON2')
```

y para transferir todo el conjunto CON1 a los elementos correspondientes de CON2 mediante

```
CALL TRANSF('CON1', 'CON2')
```

Finalmente el cambio de condiciones de CON3 a las definidas posteriormente se realiza sin más que volver a crearlo de nuevo con estas nuevas características.

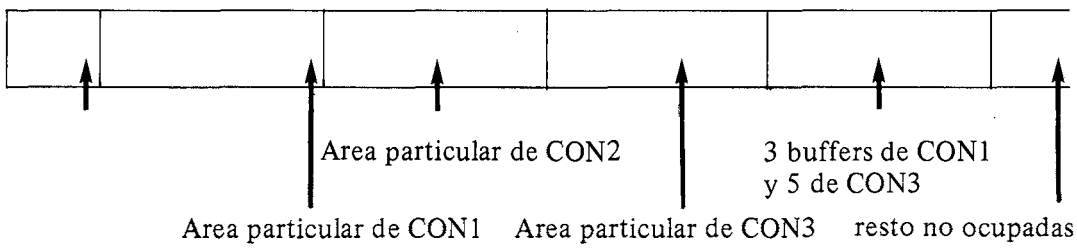
```
CALL CREARF('CON3', 50, 1, 'C', 0, 3, 2, 'F', 'F', ILOC3)
```

En este caso se reordena todo IAB para incluir las nuevas características de CON3 reelaborando el proceso de obtención del número de páginas por buffer de cada conjunto.

El borrrado del conjunto CON2 dejandolo inactivo pero manteniendo su área de control sería

```
CALL BORRAF('CON2',0)
```

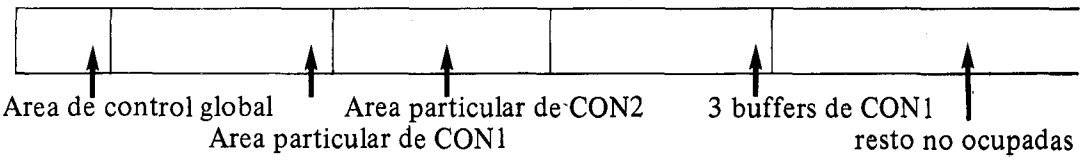
El conjunto IAB queda ahora como



y si se elimina CON3 definitivamente

```
CALL BORRAF('CON3', 1)
```

Con lo que IAB es ahora



La finalización del proceso almacenando toda el área de control /CIAB/ en un fichero previamente abierto denominado, por ejemplo FIC1, sería

```
CALL IABFIN('FIC1')
```

Es de resaltar que el paso de tratar un conjunto mediante este sistema a tratarlo en memoria principal (por ejemplo mediante el paso de un programa de un ordenador a otro con mayor memoria) sería inmedato sin más que modificar la creación de los mismos utilizando las subrutinas CREAR y BORRA[1] en lugar de CREARF y BORRAF, eliminando el COMMON /CIAB/, haciendo un EQUIVALENCE entre el conjunto IAB y el IA del COMMON /CIA/[1] y finalmente modificando la función IWW como

```
FUNCTION IWW('CONJ', I, J)
COMMON /CIA/ IA(1)
ILOC=LOCA('CONJ', 0)
IWW=ILOC+IA(ILOC+5)*J+I
RETURN
END
```

CONCLUSIONES

La utilización del esquema anterior ha demostrado una alta flexibilidad pudiendo tratarse en la misma forma conjuntos de muy diversas características a lo largo de un programa aprovechando al máximo las disponibilidades de memoria en cada instante.

Con él se reduce al máximo el número de accesos a fichero y con ello el tiempo de ejecución a costa de complicar el proceso de creación de un conjunto por lo que este proceso es más efectivo cuanto menor es el número de conjuntos a manejar por este sistema frente al número de veces que se accede a cada uno de ellos.

El tiempo de ejecución depende naturalmente de la velocidad de acceso a fichero del microordenador utilizado frente a la velocidad de cálculo en memoria principal. Para los casos utilizados y el microordenador base (un HP150A con hard disk) el incremento de tiempo ha sido apreciable pero perfectamente asumible.

La programación se simplifica y sobre todo la portabilidad aumenta considerablemente de forma que en el caso considerado el paso de un programa del microordenador base a un VAX 11/780 con memoria virtual en el que se encontraba previamente implementada la biblioteca LIBGEN es inmediato con la modificación de IWW reseñada.

REFERENCIAS

1. M. Doblaré, J. Anza. "LIBGEN. Una biblioteca de subrutinas de ayuda para la programación en FORTRAN77". Monografía AMMCTE-1/86. Depto. Ing. Mec. E.T.S.I.I. Univ. Zaragoza. Enero, (1986).
2. A.F. Martín. "Un sistema de gestión dinámica de la memoria dinámica". Proc. 1 Cong. Iberoamericano de Métodos Comp. en Ing. Madrid Julio (1985).