Virtual memory management in a paging environment

by NORMAN WEIZER and G. OPPENHEIMER

Radio Corporation of America
Camden, New Jersey

INTRODUCTION

The Spectra 70/46 Time Sharing Operating System (TSOS) is designed to be a combined time-sharing and multiprogramming system that will support up to 48 conversational users or a combined total of 64 batch and interactive tasks processing simultaneously.

The memory management subsystems of TSOS maintain complete control of main core memory, the drum backing store and the virtual memory facilities of the entire system. The virtual memory management subsystem controls the allocation and release of the backing store space, the organization of the 2 million byte virtual memory and the characteristics (the control-bit settings) of the allocated virtual memory space.

Hardware description

A short description of the relevant Spectra 70/46 Processor features is presented here to provide a background for the discussion of the virtual memory management subsystem. The 70/46 is basically identical to the Spectra 70/45 Mod II Processor with the addition of a flip-flop implemented hardware translation memory. The dynamic translation facilities of the 70/46 are provided by this translation memory and the special functions implemented in the read only memory.

The translation memory (TM) contains 512 half word elements each of which represents a single virtual page. The page size used within TSOS is 4096 bytes, and thus the virtual memory is a linear space of two million bytes.

Each half word element in the translation memory is composed of a set of control bits and a physical page number, shown in detail in the Appendix. The control bits indicate whether the page has been modified, whether it has been accessed, whether it may be modified, whether access is restricted to privileged users and whether the page is in memory. If a referenced page is in memory the physical page number is used in conjunction with the 12-bit displacement field of the virtual address to determine the physical address. If the page is not in memory the hardware generates a paging queue interrupt, and the software, utilizing a hardware special analysis function, determines the page(s) required and causes the page(s) to be brought into main memory.

The 24 bit virtual address format is shown in Figure 1. It represents the address formulated after all address arithmetic has been performed.

The Page and Displacement portions of the virtual address constitute the 18-bit address field and are generated by the 18-bit address arithmetic. If the sum of the least significant 18 bits of the base register, an index register, and a displacement field of an instruction would normally cause a carry into the 19th bit of the address field, this carry is lost and an address wrap around to the lower boundary of a segment takes place, thus providing a modified form of segmentation.

The segment, unused, and D bit fields of an address can be changed in the base registers by using the normal binary addition capabilities of the processor. The D bit is used to obtain direct (untranslated) addressing capability while in the 70/46 or translation addressing mode. Only privileged Control system functions can use this facility.

The nine bit field formed by the segment and page bits of the virtual address forms the index which is used to determine which of the 512 translation memory elements corresponds to the addressed virtual page.

---

* The Spectra 70/46 is also capable of being run in a 70/45 mode. In the 70/45 mode no address translation takes place and the address space is limited to 262K bytes.
The six page bits contained within the indexed TM entry (see Appendix) are concatenated with the 12 low order bits of the virtual address to form the 18 bit physical address actually used by the processor to address memory.

Two memory protection capabilities are provided in the 70/46. The first capability is provided by a set of protection key locations associated with main core memory. These keys are only used in the 70/45 mode of processing, although they are also operational in the 70/46 mode. The second capability is provided by the translation memory implementation and is only available when in the 70/46 mode. A nonprivileged routine in the 70/46 addressing mode, or a privileged function not using the direct (untranslated) addressing capability, cannot address information unless a translation memory element for that task allows translation to that memory location. In this way, unless the entries for two users are simultaneously loaded into translation memory, no user can access the private information of another user. Also, the control bits of the translation memory entries prevent nonprivileged access to unauthorized information and also prevent modification of code which is executable only.

The backing store for the 70/46 is a fixed head drum of either 800 or 1600 tracks. The track capacity of the drum is approximately 5000 bytes. By assigning a single 4096 byte page per track, the 3600 RPM drum can accommodate 60 page transfers per second. The time to transfer a page between core and drum is approximately 13.65 msecs, thus leaving about 3 msecs-free time between the end of one page transfer and the beginning of the next. This free time (gap time) is an upper bound on the amount of processing that may be performed between page transfers if the full drum transfer rate capability is to be realized.

All of the I/O operations, including the paging transfers, use untranslated or direct addresses. This requires that the virtual to physical address conversion must be made before an I/O is initiated. Also, any pages involved in an I/O operation, including those which contain the I/O control information, must remain in core during the duration of the I/O operation.

### Paging algorithm

A demand type paging algorithm is implemented in TSOS. This algorithm limits the number of tasks simultaneously competing for the processor and main memory by using a "working set" concept in the scheduling of tasks.

When a task is made "active" (i.e., is allowed to compete for processor time and main memory) the counter of available main memory pages is decremented to set aside the number of pages it is anticipated the task will require. This number is equal to the number of pages used by the task during its previous activation period.

Rather than fully swapping a program's working set into memory or allocating specific memory pages for the task at activation time, however, only those pages required by the task's first instruction are actually pre-paged into core. During a task's active period, its pages in core are normally considered non-pageable. When a task is deactivated, the counter of available main memory pages is incremented, and all of the task's pages in core are placed on the page-out queues.

When a task is blocked by a paging queue interrupt, pages are chosen from the page-out queues and the appropriate drum transfers are initiated. During the period in which the required pages are being brought into core, other active tasks are placed in control of the processor.

### Memory management design considerations

In general a memory management subsystem for a multi-access system should have the following characteristics:

1. Protection—no user should be able to destroy the data belonging to another user or to the system as a whole;
2. Privacy—without authorization, no user should be able to access the data belonging to another user or the private system data;
3. Shared Code Use—several users should be able to simultaneously use the same physical copy of commonly used routines or programs;
4. Flexibility—the full memory management capabilities provided by the hardware, consistent with the protection and privacy considerations, should be made available to the user programs;
5. Ease of Usage—the memory management facilities should be provided to the user in a manner which allows them to be easily used;
6. Low Overhead—the use of the memory man-

* An exception to this rule occurs if a single task requires more pageable main memory space than is available in the system.
agement facilities should add as little overhead to the system as possible, consistent with the other characteristics;

7. Integrity of Design—the memory management subsystem should not be designed as a unit separated from the remainder of the operating system. It must be designed as an integral part of the overall system but with clearly defined boundaries and interfaces. The clearly defined boundaries and interfaces prevent a great many problems in the implementation and debugging phases of operating system development. (The method used to develop the scheduling and paging algorithms for TSOS is described in Reference 4.)

8. Modularity—the memory management subsystem should be designed as a set of modular routines. There should be simple and sharply defined interfaces between the various routines to simplify implementations and debugging problems.

In the following sections a description of the TSOS Virtual Memory Management Subsystem is provided and an attempt is made to show how all of the above criteria were met within the hardware environment described above.

Virtual memory organization

The two million bytes of virtual memory are divided into two equal units. Each user of the system is permitted to use the first one million bytes for code and data areas related strictly to his own task. The second one million bytes are reserved for Control System functions and shared code.

In terms of the use of the translation memory this means that the first 256 entries are used for private user task information. Each time a new task gains control of the processor the previous task’s translation memory entries are stored in main memory, and the new task’s entries are loaded into the translation memory. During this entire process the upper 256 entries in the translation memory are unchanged.

This organization of the virtual memory, aside from reducing the overhead entailed by the loading and unloading of the translation memory, permits the Control System to be written using virtual addressing and at the same time to have full access to all user areas. Since the task in control of the processor has its entries loaded into the translation memory while it is running, the task’s memory is directly available to the Control System through the translation mechanism. (The converse is not true, in that the Control Program pages are privileged and the pages containing shared code are executable only, preventing user code from accessing Control Program information and from modifying shared code.)

If the virtual memory were not divided as it has been, and the full two million bytes had been made available to each user, the Control System would have had to use direct addressing to a much greater degree and would have required much more code to be resident, or the loading and unloading of the translation memory would have been appreciably greater.

Backing store allocation

Although each user task has a private one million byte virtual memory, the memory is not actually usable until it is dynamically allocated by means of the memory management macros; that is, until a relationship is set up between a page of virtual memory and a page of backing store. In a conventional processor this is analogous to saying that the address space (which is normally equal to the physical memory size), is not usable until the program is loaded into memory. And then only the assigned portion of the total address space (memory) may be referenced.

Within TSOS user pages are allocated when a program is loaded and when additional space is dynamically requested. When a page is allocated, a translation memory entry is initialized for it and a drum track is assigned. This track is associated with the page on a permanent basis, i.e., until the page is released and the translation memory entry is no longer valid.

The relationship between the backing store track and the page of virtual memory is maintained even while the page is in main memory for the following reasons. The number of pages of main memory is small compared to the number of pages on the backing store. Therefore, the marginal gain in drum tracks available to the system through the use of a dynamic assignment system would be small. General utilization of the drum tracks in this manner would also increase the probability of binding the system intolerably should the drum become saturated.

From another viewpoint, the fact that there is only a single page per track means that schemes which reassign drum tracks to core pages that must be written out, so as to optimize drum utilization in a multiple page per track environment, are not applicable in the environment of TSOS.

In summary, until a virtual page is requested and backing store assigned to it, the virtual memory space it represents is not usable. Any attempt to access an unallocated page is detected by a combination of hard-
ware and software and is treated as a program addressing error.

**Virtual memory classification**

To regulate the use of virtual memory, and to simplify its request, particularly within the Control System, virtual memory is divided into six somewhat arbitrary classes. The address assignments for the six classes are shown in Figure 2. The characteristics of each class are described below.

**Class 1 Virtual Memory** is occupied by the resident portion (kernel) of the Control System. All Class 1 pages are privileged and nonpageable. There are no drum images of these pages. At present there are 10 Class 1 pages in TSOS.

**Class 2 Virtual Memory** is occupied by the nonresident portion of the Control System. All Class 2 pages are privileged and pageable and may be marked as executable only, depending upon the nature of the routines occupying them. There is a drum image for each of these pages.

Classes 1 and 2 virtual memory are preallocated at system generation. The boundary between these two classes (CILIM) may be varied from system to system dependent upon installation requirements.

**Class 3 Virtual Memory** is occupied by the dynamically acquired resident portion of the Control System. All Class 3 pages are privileged and nonpageable. There are no drum images of these pages. This memory class is used for task control blocks, terminal I/O buffers and certain system work space. It is also dynamically released when the requirement for resident space lessens.

**Class 4 Virtual Memory** is occupied by the nonresident work space dynamically acquired by the Control Program and by the shared code called by the users of the system. All Class 4 pages are pageable and have drum images. The Class 4 pages used by the Control System are marked privileged, but those used for shared code are marked nonprivileged.

Virtual memory Classes 1 through 4 constitute the system virtual memory. As a group these four classes must be contained within the one-million bytes of address space available to the system. They reside in the upper one-half of the translation memory and are not changed (swapped) in the translation memory as control is passed from user to user.

Virtual memory Classes 5 and 6 constitute the user's virtual memory. Together these two classes are limited to the one million bytes available to the user. They occupy the lower one-half of the translation memory and as control is passed from user to user the Class 5 and Class 6 translation memory entries for each user are swapped out of and into the translation memory. This means that any data stored in a user's Classes 5 and 6 Virtual Memory cannot be accessed using virtual addresses when that user's entries are not loaded into TM. (This, in turn, means that the system must use direct, non-translated, addressing to access user memory for a user that is not in control of the processor.)

**Class 5 Virtual Memory** is occupied by dynamically allocated pageable areas acquired for the specific user by the Control System. These pages may be marked privileged or nonprivileged. They are used for task dependent information such as task dependent virtual memory tables, protected file control blocks, program loader data, data maintained by the interactive debugging language and I/O buffers acquired for the task by the system.

**Class 6 Virtual Memory** is occupied by dynamically allocated pages acquired by the user for his code and work areas. The pages of Class 6 memory are under control of the user task.

The boundary between Class 5 and 6 memory (C6LIM) is completely variable and depends upon the requirements of each individual task. Normally Class 5 memory occupies the 16 pages from page 240 through 255, and Class 6 memory occupies the 240 pages from page 0 through 239. Each memory class is allocated contiguously such that a page of Class 5
memory is never bounded on both sides by pages of Class 6 memory or vice-versa.

Shared code

Nonprivileged shared code offers the potential advantages of savings of main memory and backing store space plus a reduction in the paging rate. However, additional memory management control logic is required to realize these advantages. In systems with true segmentation, the segment is normally the unit which is shared and shared code may be used by attaching the called segment to the virtual memory of the calling task. This degree of generality in a system with a linear address space requires more control logic than the potential advantages warrant.

With a linear address space it seems preferable to allocate some of the address space for shared code and to take this space out of the system's area of virtual memory. This procedure eliminates the need for any overhead producing special actions when a task using shared code gains or loses control of the processor. It also permits the same algorithm to be used for paging the Control System and the shared code, simplifying the design and implementation of the paging subsystem and thus reducing system overhead.

The major disadvantage of this approach is that the (virtual memory) space for the shared code must be allocated for every user, whether or not he uses the shared code. However, it is felt that the low overhead, ease and flexibility of use, and ease of implementation more than make up for the loss of some address space.

In TSOS the system administrator determines for his specific installation what major routines will be considered eligible for sharing and makes this determination known to the system by means of a special command. He may choose only RCA supplied software such as the File Editor, and the Interactive Fortran compiler, or some user designed programs; or any combination of the two. Upon the first call for one of these shared routines the loader allocates the amount of memory needed to load this routine. This memory is allocated as nonprivileged, execute only, Class 4 virtual memory. Upon succeeding calls for the same routine, the loader establishes links between the shared routine and the calling task without the need for reloading the shared program in any form.

During execution each user of the shared routine uses the same physical (and virtual) copy of the routine as all other users.

Macro calls

The acquisition and release of virtual memory and the control of the characteristics of allocated virtual memory are the major services performed for users and other Control System functions by the virtual memory management subsystem. These services are requested by means of macros which generate standardized linkages and parameter lists. These linkages may be either Supervisor Call instructions (SVCs) or standardized branching conventions, both of which provide clean interfaces, an invaluable aid in the debugging phases of complex system development.

The macros are named REQM (request memory), RELM (release memory), and CSTAT (change memory status). There are two forms of each macro, one which may be used by nonprivileged and privileged (Control System) routines, and the second which is restricted to privileged routines only.

The nonprivileged forms of the REQM and RELM macros permit the user to request and release Class 6 memory in multiples of one page, with a maximum of 64 pages per call. If the address spaces and backing store space is available, the requested memory will be allocated in the first unallocated area (lowest available area in the address space) large enough to satisfy the request; or if the user so specifies, the memory will be allocated starting at a specific address.

The nonprivileged form of the CSTAT macro allows the user to change the status of any page in Class 6 memory to read-only or read-write. The CSTAT macro also provides the mechanism for users to request that specific Class 6 pages be made pageable or nonpageable.*

The privileged forms of the virtual memory macros allow Control System routines to operate on any page in Classes 2, 3, 4, 5 and 6 virtual memory. The option of the CSTAT macro which changes a page's status to read-only or read-write is available for all memory classes. The option to make pages pageable or nonpageable is available only for memory Classes 2, 4, 5 and 6. This option of the CSTAT macro is the most heavily used as it permits the Control System to lock into (unlock from) main memory pages which are (were) required to be resident for I/O operations.**

The privileged forms of the REQM and RELM macros permit Control System functions to request and release Classes 3, 4 and 5 virtual memory. Classes

* Provision exists within the system, in certain well defined situations to permit users to use this option of the CSTAT macro. The limit of the number of pages that a user may make nonpageable is established based upon system-wide parameters and conditions set at task initialization.

** All I/O is done with nontranslated addressing and thus commands must contain physical addresses and buffers must not be moved until the I/O operations complete.
Partial page allocation

Many Control System functions require different sized areas of memory during their execution. This memory may be required specifically for a single task or it may contain system wide information. Memory space which need not always be resident and which is required for a single task is acquired as Class 5 memory; system wide information which is pageable is stored in Class 4 memory and user dependent or system dependent information which is nonpageable is stored in Class 3 memory.

To conserve address space, better utilize main memory and reduce the paging rate for Control System pages, Classes 3, 4 and 5 memory are allocatable in partial page units. The units of allocation are 8n bytes where 2 ≤ n ≤ 509.

Any request for larger size areas are allocated in full page increments. Any size area may be requested, but during the allocation process the size allocated is rounded up to the next larger standard size. This standardization, making all allocations multiples of a single quantum size, eases both the allocation and garbage collection processes employed.

Each page allocated is treated as a separate unit so that no partial page allocation crosses a page boundary. This serves two purposes. First it eases the record keeping involved by limiting the number of areas considered in a single operation. Second it prevents dynamically acquired I/O buffers from being allocated across page boundaries.

The latter is significant in that otherwise it would be necessary to page contiguous virtual pages into contiguous main memory pages, and this would vastly complicate the paging and physical memory management subsystems.

To manage the partial page allocation two linked lists are maintained in each subdivided page. One list, termed the main list, links all of the areas on the page in address order. The second list, termed the free list, links all of the unallocated areas in area size order, with the smallest area at the head of the list. The links of both lists are eight bytes long. The entries in the links include a free bit, which is used to indicate unallocated areas, a size field, forward and backward link fields and a two byte integrity field used by software to check that the link was not destroyed by some other software routine.

In addition to the memory links, partial page tables are also maintained by the system to manage partial page allocation. Two of the tables are maintained in Class 3 memory to control the Class 3 and Class 4 memory partial page allocations. There is also a corresponding partial page table in each user's Class 5 memory which is used to control the Class 5 partial page allocations for that user. The entries in each table are identical. They consist of the virtual page number of the page to which they correspond and the size of the largest free area on the page. There is one entry for each page which is subdivided for partial page allocation.

The placement of the memory links on the same page as the partial page areas presents the possibility of malfunctioning system components destroying the links. However, rather than proving to be a hindrance, this link placement proved to be a great aid in system debugging. This is due to the fact that the memory management routines will often be the first system function to find the destroyed link. This, in turn, helps to avoid the problem that some other system function will malfunction, because it uses an adjacent area which was also destroyed, allowing many bugs of the type which would only occur at widely scattered intervals to be more easily tracked down.

Memory management tables

A relatively complex table structure is required to support the memory management functions of TOS. These tables support the physical memory management and paging subsystems along with the virtual memory management subsystem. They are used primarily to maintain allocation status information for the major memory resources—the core pages, the drum pages, the system virtual address space and user virtual address space.

The allocation status information for drum pages and for system virtual memory pages is maintained in bit-per-page maps called the Paging Drum Memory Map and the System Virtual Memory Map. These tables are used when the request memory (REQM) macro code must find an unallocated drum page during the allocation of a page of pageable virtual memory and when it is necessary to determine the address of free pages during the allocation of system virtual memory. These tables are also used during the corresponding RELM (release memory) processing.

The core status data are maintained in two tables called the Physical Memory Map and the Physical Page Allocation Table. Each entry of the Physical Memory Map indicates whether the page is free or allocated, the memory class data for non pageable pages and certain reservation information. The Phys-
The main portion contains the entry for pageable pages, the I/O count (the number of I/O operations in process or scheduled into this virtual page), link space for the page out queues, and the address of the Virtual Page Table entry for pageable pages.

The System Virtual Page Table is a two part table. The main portion contains the core image of the entries loaded into the translation memory for the system virtual memory. However, when the pages represented by these entries are not in core, the cylinder portion of the backing store address is maintained in these entries. The secondary part of the table is used to store the drum track portion of the backing store address.

The above described tables are maintained in Class 1 virtual memory. They are system wide tables. In addition, there are four private tables maintained for each user. They are the Block Address Table and the associated User Virtual Page Table which are maintained in Class 3 Virtual Memory, and the User Virtual Memory Map and the Class 5 Partial Page Table.

The Block Address Table entries for each user are maintained within the Task Control Block (TCB). The TCB contains the master information about each task in the system. The Block Address Table entries of a task are used within a special function to cause the User Virtual Page Table entries to be loaded into the translation memory when the task is to be given control of the processor, and conversely when these entries are to be stored in core when control of the processor is removed from the task. The space used to store these entries is maintained in System Virtual Memory to guarantee their accessibility by the Control System at any time. Otherwise, they would be accessible only when the user was in control of the processor.

The User Virtual Memory Map parallels the System Virtual Memory Map and is allocated in the user’s Class 5 memory. The Class 5 partial page table is also allocated in the user’s Class 5 memory. It is used to control the partial page allocation of the user’s Class 5 memory.

**SUMMARY**

The salient hardware features of the system described are: a linear address space of 512 pages of 4096 bytes each; a main memory of 64 pages; a single level page per track backing store of 800 or 1600 pages; and the use of a 512 entry translation memory to effect the virtual memory of the system.

The facilities controlled by the virtual memory management subsystem described include the organization of the virtual memory, the subdivision of the virtual memory into classes, the management of the shared code within the system, and the allocation of backing store and of partial pages.

Within the context of the hardware structure, the major aspects and advantages of the described software system are summarized below.

The partitioning of the virtual memory to concurrently accommodate the system and a single user reduces translation memory swapping overhead and provides the system code with full accessibility to all user code, while still permitting the system code to be written using virtual addresses.

The division of the virtual memory into classes structures the use of the virtual memory, regulates its use and simplifies the request and release procedures, especially within the Control System.

The incorporation of sharable code within the system virtual memory affords its direct accessibility to all users, permits a single page table to be maintained for the code, and allows the same paging algorithm to be applied to shared code as is used for system code; but it requires all users to give up the same amount of virtual memory for the shared code, whether or not they use the shared code.

The allocation of backing store only when a page of pageable virtual memory is allocated enables more users to be run concurrently with a given level of backing store than if the backing store was allocated for the entire user virtual memory, regardless of the user’s intent to utilize his entire virtual memory.

The maintenance of a relationship between a virtual memory page and a track on the backing store, even when the page is in memory, is justified based upon the real probability that the page may not be modified and therefore will not have to be written out—if the backing store association is maintained while it is in memory; and the added consideration that the ratio of drum tracks to memory tracks is such that the marginal gain in drum tracks available to the system from reassigning pages in memory is extremely small. The drum characteristic of a single page per track is also a factor in this regard.

The provision for partial page allocation for other Control System functions, while it increases the calls on the virtual memory subsystem, provides for better utilization of memory and easier development of reentrant code.

**REFERENCES**

1. RCA Spectra 70/46 Processor Reference Manual
2. RCA Spectra 70/33 45 55 Processor Reference Manual
3. B RANDELL C J KUEHNER

*Dynamic storage allocation systems*
APPENDIX

The translation memory

The Translation Memory is 512 half-words in size. Each entry in Translation Memory has the format shown in Figure 3.

The meaning of each of the control bits and the physical page number in the translation memory entry is given below:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>Parity bit (invisible to the software).</td>
</tr>
<tr>
<td>W</td>
<td>Written Into Bit: indicates when set, that the page addressed in memory by this translation halfword has been written into. This bit is automatically set by hardware and reset by software.</td>
</tr>
<tr>
<td>G</td>
<td>Accessed Bit: indicates, when set, that the page addressed in memory by this translation halfword has been accessed (read, or written into). This bit is automatically set and reset by hardware.</td>
</tr>
<tr>
<td>U</td>
<td>Utilization Bit: indicates, when set, that the addressed translation word can be utilized. This bit indicates, when reset, that the addressed translation word cannot be utilized (i.e., this virtual page is not in core) and a Paging Queue Program Interrupt occurs. This bit is set and reset by software.</td>
</tr>
<tr>
<td>S</td>
<td>State Bit: Indicates when set, that the addressed translation word is nonprivileged. When this bit is reset, it indicates that the address page is privileged and can only be accessed by a program operating in the privileged mode (i.e., a portion of the system software). When this bit is reset and a nonprivileged program attempts to access this page, a Paging Error Program Interrupt occurs. This bit is set and reset by software.</td>
</tr>
<tr>
<td>E</td>
<td>Executable Bit: indicates when set, that the page addressed in memory by this translation word can be read as an operand or instruction but cannot be written into. When this bit is reset, all forms of access are allowed for this page. If a program attempts to write into a page with this bit set in the translation word, a Paging Error Interrupt occurs. This bit is set and reset by software.</td>
</tr>
</tbody>
</table>

$M$ and $H$ bits are used when the 2048 byte virtual page mode is used. Under TSOS only the 4096 byte virtual page mode is used.

Physical Page Number: when the $U$ bit is set, these six bits contain the six most significant bits of the actual physical address of the page represented by this T.M. entry. The full physical address is obtained by concatenating these six bits with the low order 12 bits of the virtual address. When the $U$ bit is reset no meaningful information is contained in this field.

![Figure 3 — Format of a translation memory entry](From the collection of the Computer History Museum (www.computerhistory.org))