18.1 GENERAL

Memory management in TOSS is available in four different options, depending on the hardware installed. These options are specified at SYSGEN time and their hardware requirements are as follows:-

* MMU Paging.
  - UK01 with MMU; 64 - 128kB memory.
  - 6813 with MMU; 64 - 256kB memory.

* Disk Paging.
  - 6805, 10, 12, 13, or UK01; disk or flexible disk.

* Disk and MMU paging.
  - UK01 with MMU; 64 - 128kB; disk or flexible disk.
  - 6813 with MMU; 64 - 256kb; disk or flexible disk.

* Swappable work blocks.
  - 6805, 10, 12, 13, or UK01; disk or flexible disk.

18.2 MEMORY MANAGEMENT UNIT (MMU)

18.2.1 MMU Structure

The Memory Management Unit is a hardware option which allows memory addressing up to a maximum of 256kB.

The MMU consists of a 16-register table, whose contents can be changed by four special instructions, i.e. Table Load, Table Store, Table Load Register, and Table Store Register.

18.2.2 Address Translation

The CPU operates in either System mode or User mode.

In System mode, memory locations are addressed by using the whole contents of a word, 16 bits.

This allows addressing of the first 64kB of memory and MMU is not used. However, the system is able to extend its addressing by using the extended instructions, which make use of MMU address translation.

Address translation is always performed by the MMU while the CPU operates in User mode.

In User mode, the MMU translates the first four bits of a 16 bit Logical Address into a six-bit physical page address (MMU-registers having previously been loaded). The remaining 12 bits of the Logical Address then select a 4kB slot within the selected page.

It is important to note that the MMU addresses the whole of the memory, not just the area above the 64kB low address space. Pages can equally as well be placed at addresses below 64kB as above.

See figure 18.1 which illustrates MMU addressing of memory.

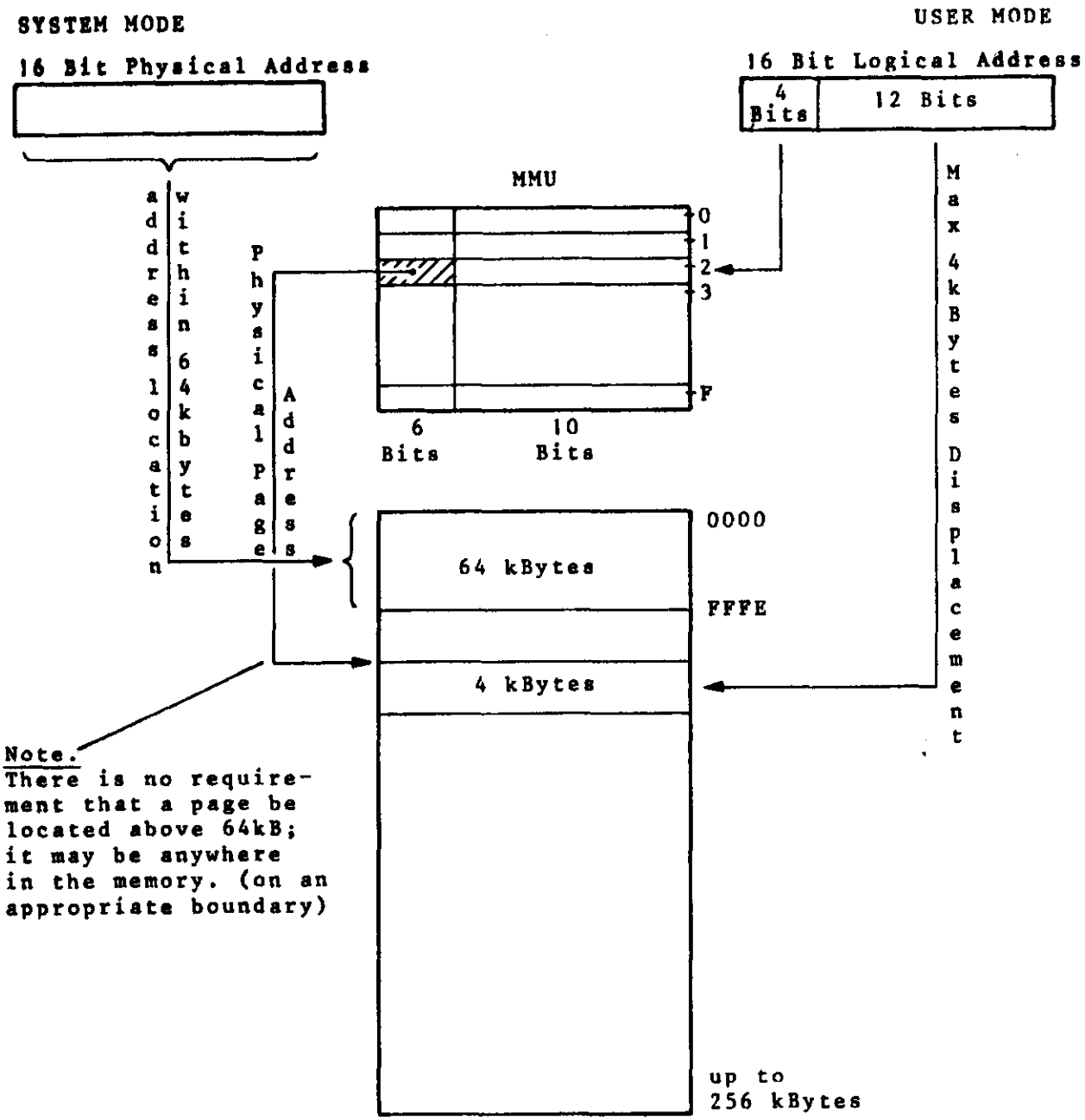


Figure 18.1. MMU Memory Addressing.

18.3 MMU PAGING ONLY

With MMU systems a distinction is made between logical memory and physical memory.

Logical memory is an area addressed in user mode containing data, code pages, etc. These data and code pages are not necessarily contiguous physically.

The transformation of a part of logical memory into physical memory is a function of the MMU.

At any one time 64kB of memory are directly accessible.

The contents of the MMU is called the task window. When CREDIT is used a number of entries in the MMU table are used for data, interpreter, etc. and some entries are left for a code pages. One code page at a time is present in the task window. Which page this is can be found via the segment block address in the task table (TTB:SB in TTAB).

Since all pages are stored in memory when only MMU paging is used, each segment block (SEGBLK) has a corresponding pageblock (PAGBLK). The tables which map these blocks, SEGTAB and PAGTAB, are never changed after program loading. For details of SEGTAB and PAGTAB, see section 18.6.

The code page entry in the MMU table and the segment base in the task control area (T:Axxy for CREDIT) are changed to point to the new code page when a branch is performed from one code page to another. This request is performed without any task switching, which makes it considerably faster than it would be otherwise.

18.4 DISK PAGING, NO MMU (64kB memory)

After allocation of memory for the different tables and work areas, the rest of the memory is divided into code pages. The size of a page is decided at linking time.

The loading of the program into these pages is then controlled by the monitor. Pages are only overwritten between task switches on the same priority level, or on request from the active task. This implies that the number of pages must be at least equal to the number of different task priorities in the system. The loading process is controlled by a special load task running at priority level 49. Application tasks may be running when the load task is executing disk I/O.

The monitor keeps a segment table in memory, SEGTAB, which contains one entry for each segment in the program. Each entry contains information about whether the segment is loaded or not, where it is loaded, PAGTAB, disk address, queue pointers, etc. The new segment address is stored in TTAB and the task control area, T:Axxy.

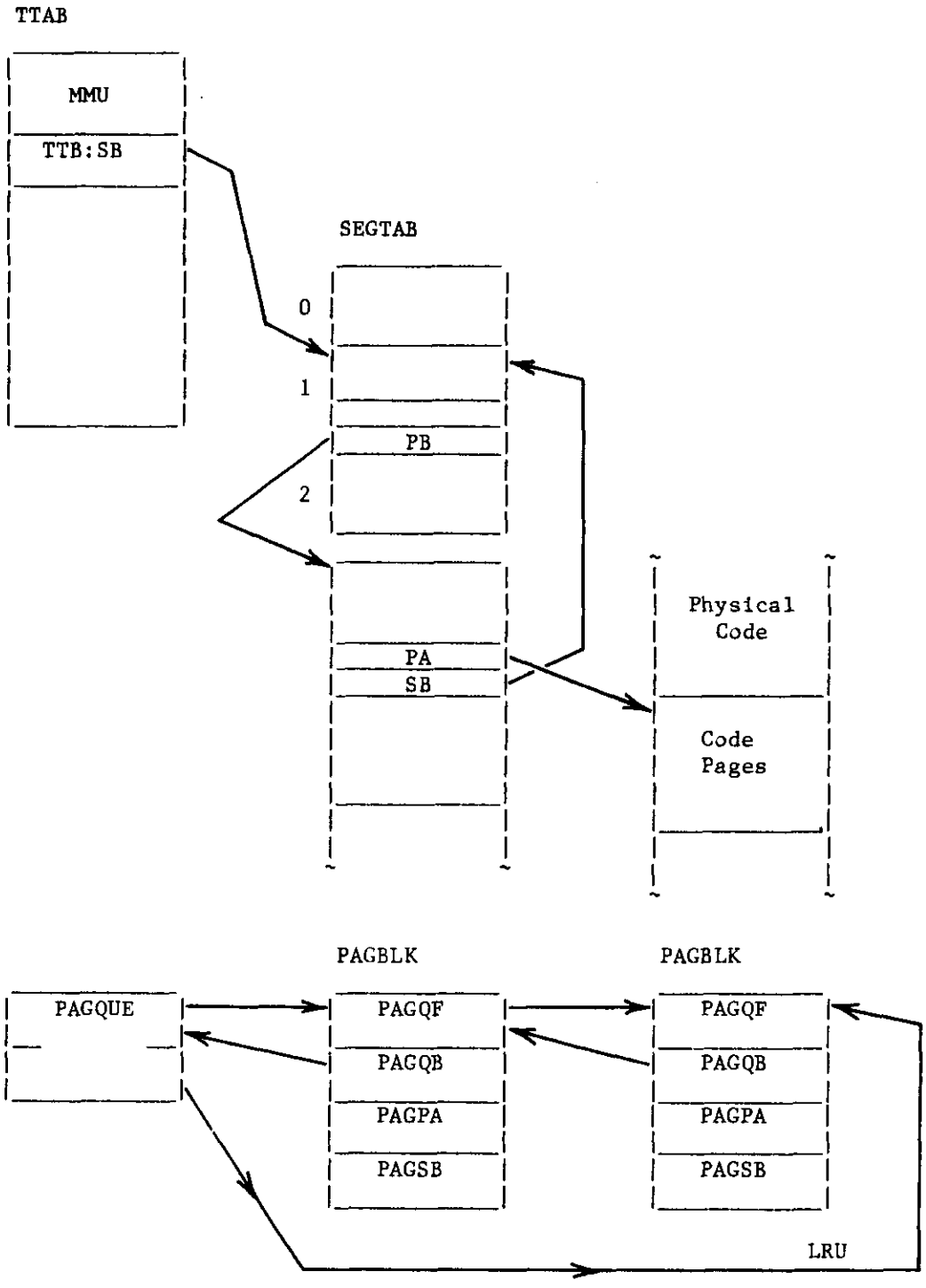


Figure 18.2. Page Queue and Associated Control Blocks.

A page currently being used by a task active in the CPU is called active. All other pages are called inactive, and are placed in a page queue (PAGQUE). When a new segment has to be loaded, space is allocated by taking out one page from this queue using a Least Recently Used (LRU) basis (figure 18.2).

If the requested segment is already present in memory and inactive, then the corresponding page is taken out of the page queue and set active.

The page queue is normally updated each time task switching occurs, which is much more often than segment loading occurs, so that the system can react dynamically to changes in workload conditions.

If the new segment is in the page queue no task switching is performed.

If the new segment has to be loaded from disk, then task switching is necessary. See figure 18.3.

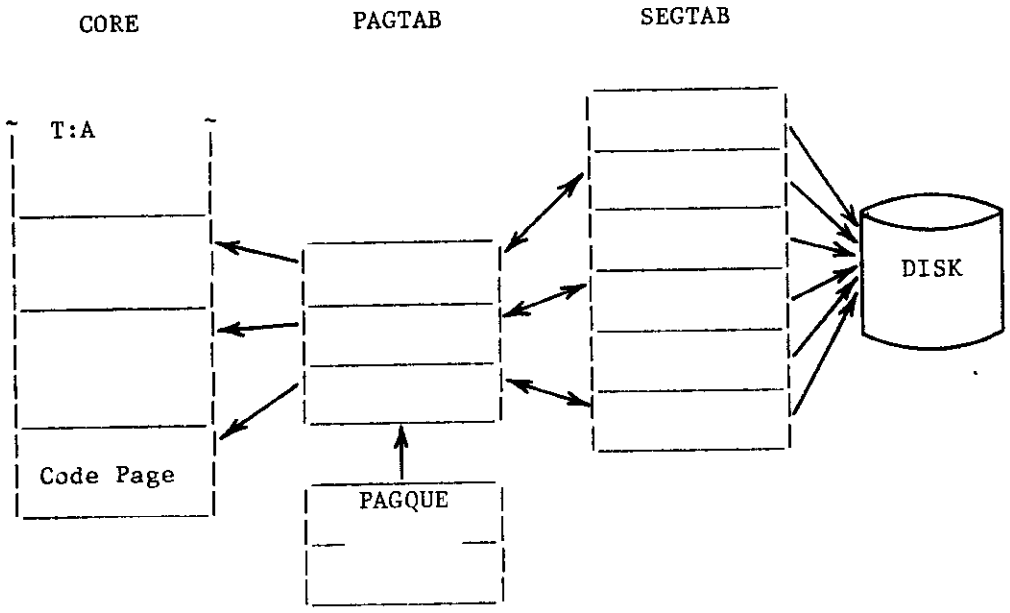


Figure 18.3. Segment Loading from Disk.

18.5 DISK AND MMU PAGING TOGETHER

When both disk paging and extended memory are in use, both previously described methods are combined.

The new segment can either be in extended memory or on disk. If it is in extended memory the loading function is the same as when only MMU was used, and if it is on disk it will be the same as if only disk paging was used.

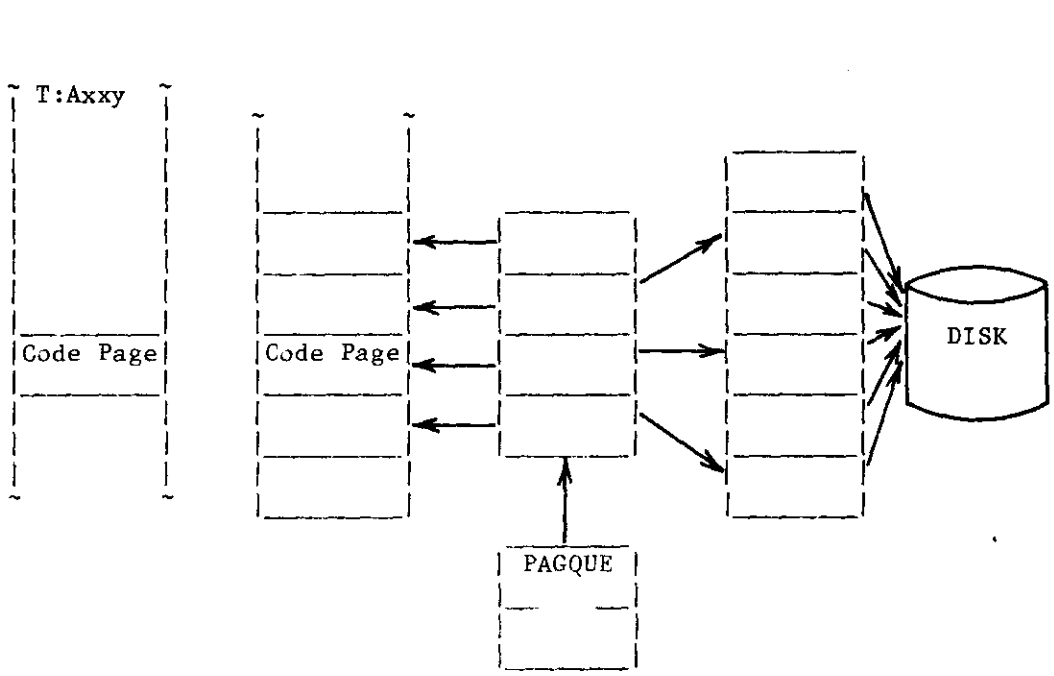


Figure 18.4. Combined Disk and MMU Paging.

18.6 MONITOR TABLES USED BY MEMORY MANAGEMENT

Two tables are central to the memory management system. They are:-

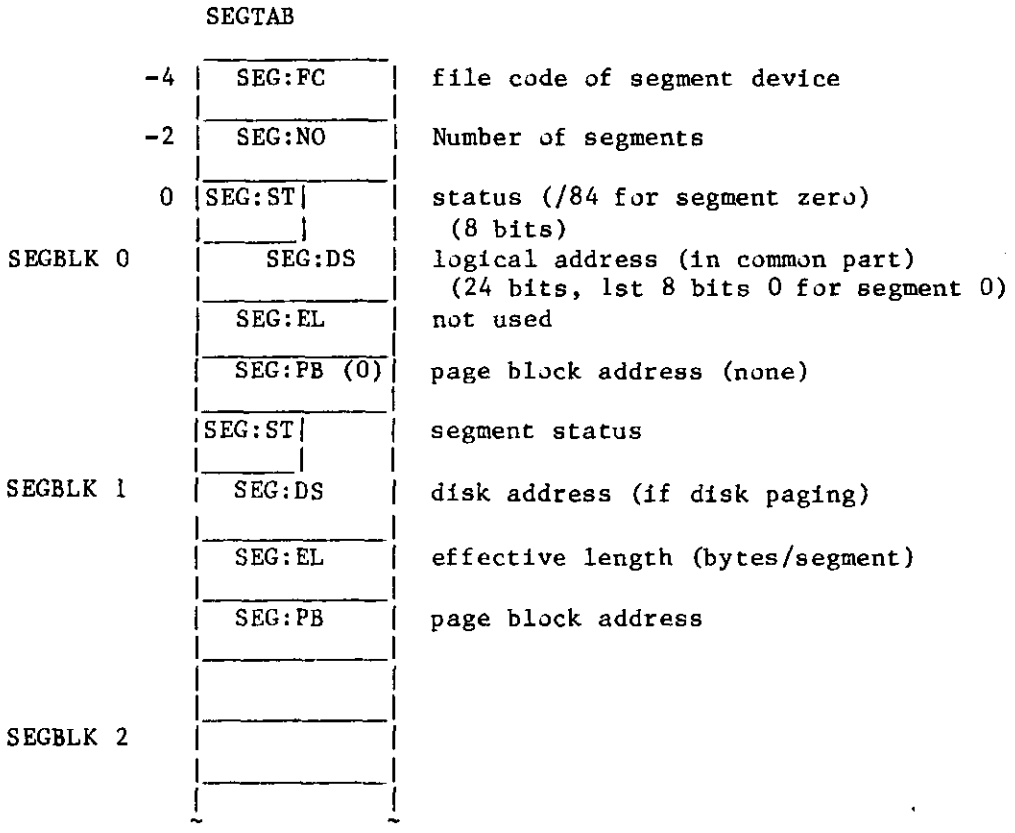
* SEGTAB, the Segment Block Table, which contains segment blocks (SEGBLK's). Each SEGBLK describes, for example, the status and location of a segment.

* PAGTAB, the Page block table, which contains page blocks (PAGBLK's). Each PAGBLK contains, for example, the location of a page in memory (physical address) and a pointer to the corresponding segment block.

Each SEGBLK is related to one segment; SEGBLK 0 belongs to Segment 0, SEGBLK 1 belongs to Segment 1, etc. Each PAGBLK is related to one page in memory; the first one is Page 1, the details of which are defined in PAGBLK 1.

18.6.1 Segment Block Table (SEGTAB)

Each segment is described in a segment block, SEGBLK. All segment blocks are contained in the table SEGTAB which is built by the system loading program, SYSLOD, at configuration time.



SEG:ST Contains the status of the segment. The bits have the following meanings when set:-

Bit 0 : The segment is in memory.
Bit 1 : Loading of this segment is in progress; set and reset by the load task.
Bit 2 : Segment locked, e.g. by the CREDIT debugger; it cannot be overwritten in memory.
Bit 3 : Used to indicate that a task on a higher priority has interrupted this task. The page will not go into the page queue until the lower priority task has returned to it. After that a LKM will cause it to be queued. In this way the page is prevented from moving to different areas of memory.
Bit 4 : Core resident. The segment can not be overwritten in memory.
Bit 5 : Common segment (segment zero).
Bit 6 : Not used.
Bit 7 : Disk I/O error (seek, CRC, or not operable).

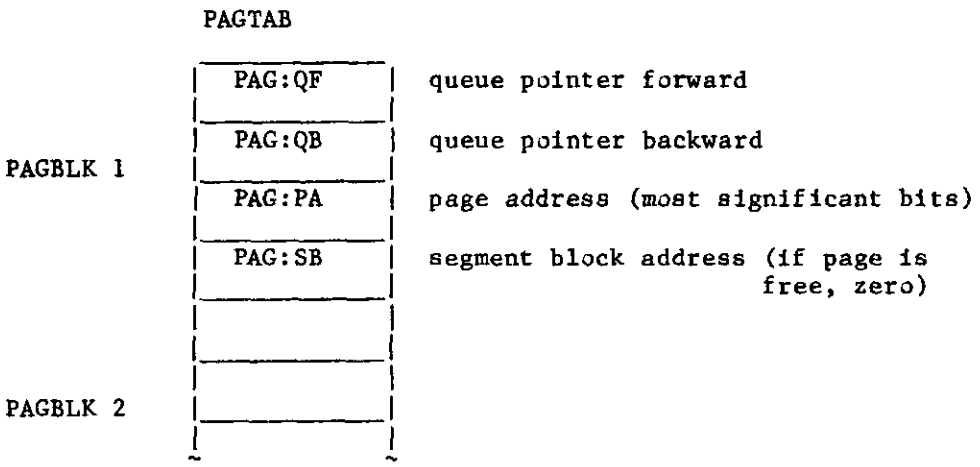
SEG:DS Bits 0 - 7 : Contain zeroes for segment zero;
For other segments bits 1 - 23 contain the disk sector address at which the segment starts (bit 0 is always 0); all zeroes if disk paging is not used. Contains the logical address for segment zero in a disk paging system.

SEG:EL Effective length in bytes of the segment.

SEG:PB Pointer to corresponding page block.

18.6.2 Page Block Table (PAGTAB)

Each page in memory is described in a page block, PAGBLK. All page blocks are contained in the page block table PAGTAB which is built at configuration time by the system loading program, SYSLOD.



PAG:QF Pointer forwards in the page queue, PAGQUE. If the page is not in PAGQUE, this word contains zero.

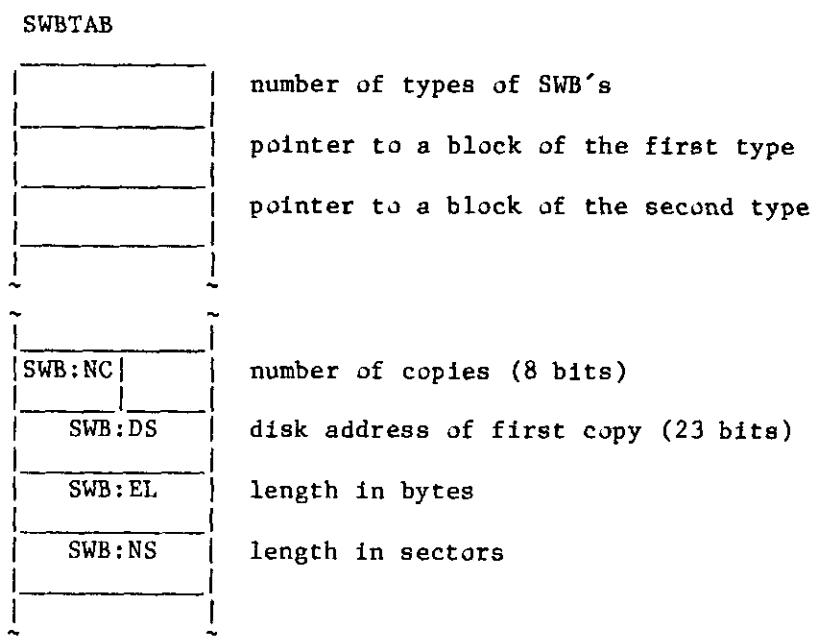
PAG:QB Pointer backwards in the page queue, PAGQUE.

PAG:PA Physical page address in memory; 16 most significant bits.

PAG:SB Pointer to the corresponding segment block. If the page is free, this word contains zero.

18.6.3 Swappable Work Block Table (SWBTAB)

When swappable workblocks are used, the system loading program (SYSLOD) builds tables which have the following layout:-



The swappable workblock type is defined in the application by the 'SWB' declaration.

SWB:NC Number of copies of this type of swappable workblock.

SWB:DS Disk sector address of the first copy of the swappable workblock of this type.

SWB:EL Length of one copy of the swappable workblock of this type in bytes.

SWB:NS Number of sectors occupied by one copy of the swappable workblock of this type.

18.7 I/O HANDLING IN MMU SYSTEMS

The monitor resides in an address space comprising the first 64kB of physical memory. Due to this hardware limitation and the demand for fast response to interrupts, most devices have their own I/O buffer placed in the system area.

At activation or completion of an I/O request, a transfer is made between the user and the device buffer. In most cases this is handled by the I/O initialization and terminating modules (TIO and TENDIO).

The user interface is the same whether the MMU option is used or not, but the device buffers are present only if MMU is implemented.

I/O requests are of several different types, defined by the order and the driver type. When the system itself performs an I/O request, the interface is exactly as in the non-MMU case.

18.7.1 Normal Output Request

This request is used when writing character-by-character, for example on the following devices:-

* GP General Printer.
* TP Teller Printer.
* DY Display.
* TC Cassette.
* LP Line Printer (programmed channel or IOP).
* FD Flexible Disk (programmed channel or IOP).

Each DWT has its own buffer and ECB, and TIO transfers the contents of the user ECB and buffer to the DWT. TENDIO performs the update of the user ECB.

18.7.2 Normal IOP Output Request

This request is used when writing in block mode, for example on:-

* DU Disk.
* MT Magnetic Tape.

TIO assembles the full 18-bit buffer address and the driver transfers it to the IOP. Each DWT has its own ECB to which TIO transfers the user ECB. TENDIO performs the update of the user ECB.

18.7.3 Normal Input Request

This request is used when reading character-wise, for example on:-

* KB Keyboard.
* TC Cassette.
* CR Card Reader.
* FD Floppy Disk (programmed channel or IOP).

Each DWT has its own buffer and ECB, and TIO transfers the contents of the user ECB to the DWT. TENDIO transfers the data from the DWT buffer to the user and updates the user ECB.

18.7.4 Normal IOP Input Request

This request is used when reading in block mode, for example on:-

* DU Disk.
* MT Magnetic Tape.

TIO assembles the full 18-bit buffer address and the driver transfers it to the IOP. Each DWT has its own ECB to which TIO transfers the user ECB. TENDIO performs the update of the user ECB.

18.7.5 Data Communications Input Request

This request is used when reading on:-

* DC Data Communications.

Each DWT has its own ECB to which TIO transfers the user ECB. The driver has a set of buffers and a special monitor routine, DC:MIN, transfers the data from the driver buffer to the user buffer.

18.7.6 Data Communications Output Request

This type of request is used when writing on:-

* DC Data Communications.

The driver keeps one write buffer and, before writing (after poll), a special monitor routine (DC:MOT) is called to transfer the data. Each DWT has its own ECB to which TIO transfers the user ECB. TENDIO performs the update of the user ECB.

18.7.7 Control Requests

This type of request is used when performing a control request (DSCx in CREDIT) on a device. No buffer is needed. Each DWT has its own ECB to which TIO transfers the user ECB. TENDIO performs the update of the user ECB.