PAGE NOT IN CORE TRAP

Judy Simon

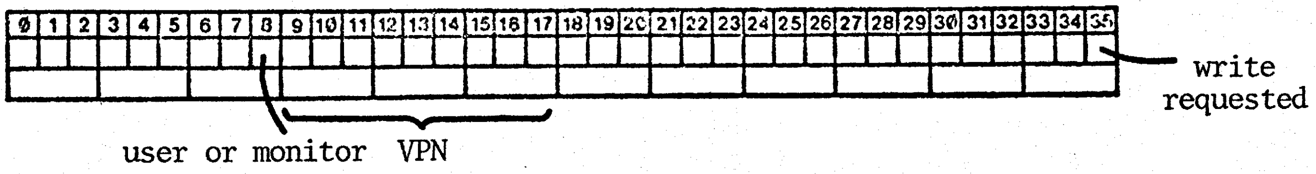
April 8, 1974

Technical Memo TM.74-17

INTRODUCTION

A page failure is first fielded by KISRV. Then PAGEM is called if KISRV cannot build the hardware map using the software maps. Thus KISRV (KIPTM, KIPTU) simulates the BBN pager which attempts to build the hardware map trapping to PAGEM (PGRTRP) for various trap conditions and for page not in core.

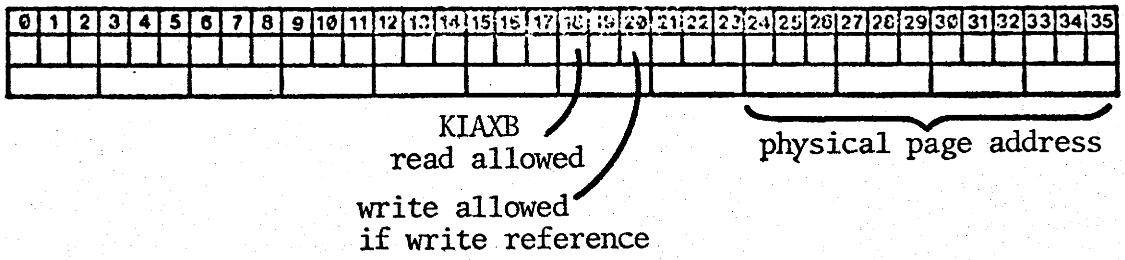
<u>KISRV</u> KIMPFW or KIUPFW



KISRV gets the virtual page number (VPN) of the trap from the Monitor or User Page Fault Word. Using the VPN and whether the trap is from the monitor or user, KISRV determines the map word to modify in KIEPT (monitor hardware page table) or KIUPT (user hardware page table). In the process of map word determination, KISRV may use UPTA (user software page table), PSB (Process Storage Block), or MMAP (Monitor Map).

If the trap is for a map word not set, KISRV then tries to load the hardware map from the software page table (map). The page table can contain three types of pointers: private, shared or indirect. For a private pointer which points to a page in core (indicated in the pointer), KISRV updates CSTØ's age field, sets the map half-word in KIEPT or KIUPT, and returns.

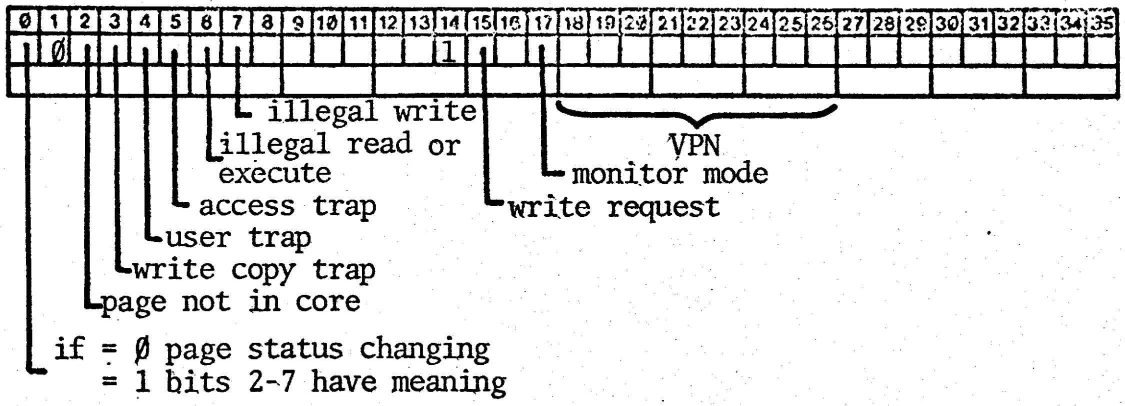
HARDWARE MAP HALF-WORD



A shared pointer (shared file page) is an index into the SPT to a pointer to the shared page. The SPT entry has the "in core" bits and the page address. If the page is in core, the hardware map is loaded as with the private pointer. An indirect pointer allows one fork to access a page in another fork's page table. In this case, the indirect pointer is an index into the SPT + a page number. The SPT entry contains the address of the other fork's page table and the indirect pointer's page number is used as an index into this second page table to get the page's address. If both the second page table and the page are in core (indicated in the SPT entry and page table entry), then KISRV loads the hardware map as with the private pointer.

If, in any of the above cases, the page table (indirect pointer) or the page is not in core, or the trap is for other than a map word not set (such as, an error), then KISRV creates TRAPSØ (Trap Status Word) and control is passed to PAGEM.

TRAPSØ


Ø 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
Ø
1
illegal write
illegal read or execute
access trap
user trap
write copy trap
page not in core
if = Ø page status changing
= 1 bits 2-7 have meaning
VPN
monitor mode
write request

PAGEM (PGRTRP)

PAGEM was originally intended to handle page traps directly from the hardware. Since some of the hardware work was taken over by KISRV and KISRV does not completely simulate the hardware, PAGEM has code which is not used. Specifically, some of the bit combinations possible in the TRAPSØ are never set by KISRV and therefore, some of the branches in PAGEM based on these bits are never taken.

PAGEM uses the TRAPSØ bits 1-7 to determine which routine to execute.

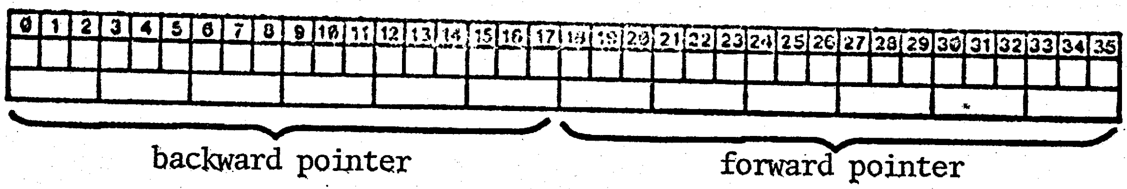
BIT	ROUTINE	
1=0	TRPØ	page in transition
1=1 and 2=1	NIC	page not in core
3=1	WCPY	write copy
4=1	UTRP	user trap
5=1	NPG	access (page not in existence)
6=1	ILRD	illegal read or execute
7=1	ILWR	illegal write

NIC Routine (See TM.74-16 for formats of entries in tables.)

The "not in core" trap can be caused either by the page not being in core or by both the page and the page table for the page not being in core. PAGEM first checks to see that the process has not overgrown its reserve. (If it has, some pages must be removed from the process. See below.) It then analyzes the trap address to determine the page (or page table) not in core. As in KISRV, first the trap address is used to find the pointer in the software map where the trap occured. If the pointer is "private", the pointer contains the current address of the missing page. For a shared pointer, the SPT contains the current address. In the case of the indirect pointer, the SPT contains the address of the page table which may not be in core or which contains the pointer to the page which is not in core.

Once the pointer for the missing page is located, in general it will contain the drum or disk location where the page can be found. (If it does not contain an address, it is a newly assigned page. If it is intended for a file, it is given a disk address in the same general area as the rest of the file.) Before the page (or page table) is brought in, the replaceable queue (free list) of core pages is checked to make sure there is a page available. If there is, a page is removed from the queue.

REPLACEABLE QUEUE (RPLQ)



If the page table is missing, it is brought in first. Special measures are taken to make sure that until the page table is in core, any indirect references through it are taken care of. The page is brought in next. (In the case of a newly created page, the page is, of course, not brought in, but it is zeroed.)

When the page (or page table) is brought in, the Core Status Tables for that core page are updated. CSTØ is given an age for the page, CST1 gets the storage address of the page (disk or drum), CST2 gets the address of the owning page table, and CST3 receives the owning fork number. These four tables describe the ownership of a particular core page. (If CST1 had had the storage address of an old page, this address would have been saved in the old page's page table or SPT slot.) The page table or SPT slot for the new page is given the new core address.

Finally, when the page (and its page table) is in core and all of the tables are set, the trap is "undone" and the process is resumed via SCHED.

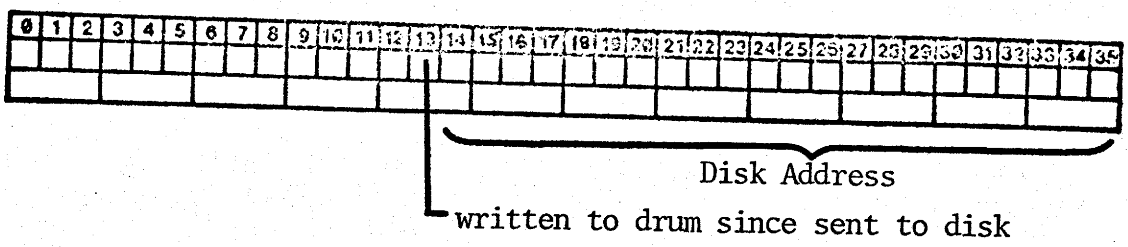
Fork Overgrown its Reserve

When the number of pages needed in a fork equals the reserve size of the fork (working set limit), the reserve is increased until the maximum number of pages allowed is reached or until there is not enough room in core for the Balance Set. When one of the last two conditions occurs, some of the fork's pages must be removed.

The routine which removes pages scans for the oldest pages (least recently used) by setting an arbitrary cut-off age and then scanning CST1 for all pages belonging to this fork which are older than the cut-off age.

The process of removing a page from core is complicated only by the desire to put file pages back on the disk, unshared (not in use) pages on the disk, and all other pages, including index blocks, on the drum. Pages which have not been modified are not written out. If a page is goint to drum, the Drum Status Table (DST) entry for this page (which contains the disk address for the page) is marked to indicate there is a new copy on the drum. If the storage address in CST1 for this page (on its way to the drum) had been a disk address, a drum address would be assigned, placed in CST1 and the disk address stored in the DST.

DRUM STATUS TABLE (indexed by a function of drum address)



Pages headed for disk which have a disk address in CST1, have no special procedures. For pages going to disk which have a drum address, the disk address is gotten from the DST, the DST entry is released, and the disk address is put in CST1. After the page is removed, control is returned to the routine which scans core for another "old" page.