# Inter-Office Memorandum

| | | | |
|---|---|---|---|
| To | Debugger Planners | Date | June 15, 1978 |
| From | Barbara Koalkin | Location | Palo Alto |
| Subject | Issues - VM to DA log file | Organization | SDD/SD/DE |

XEROX

Filed on: [Iris]<Koalkin>XD>PilotDebugger6-15.bravo

The following list is a set of issues related to working out the interface for the VM to DA log file. These issues have been discussed in the June 14th meeting of Pilot Implementors (Johnsson, Koalkin, Lauer, McJones, Redell, Sandman, and Wick).

1. We need a mapping from virtual memory to physical disk address.
   Can Pilot provide this?

Pilot has agreed to provide a log of changes to its virtual memory to physical disk address (e.g. drive, head, track, sector) map for use by the debugger. Pilot must go through several levels of mapping in order to provide this: from virtual memory to file id and file page number, to virtual disk address, to seek address (through the exception table and Pilot disk driver), to physical disk address. The debugger will use this log to build a complete map of virtual memory to disk addresses.

2. We propose having part of the file label (i.e., 2 words) stored along with the address, for use for (minimal) checking purposes. This means that at least 4 words per entry will be stored in the log.
   - (a) Is this acceptable?
   - (b) Where is the space allocated for the log?
     (Alto file system or fixed set of pages on Pilot disk)?
   - (c) How does the debugger find the log?
   - (d) How about Pilot maintaining an up-to-date copy of the entire map?

(a) Quite a lot of time was spent trying to nail down the details of what a log entry should look like. We agreed to create a DEFINITIONS file containing all of this interface information (with the fine details to be worked out at a later meeting). It seems like our simple 2 word entry has grown into a 6 word variant record containing the virtual page number and page count as common fields (2 words), a variant tag to indicate what type of disk we are talking about (Alto, Pilot, Shugart, 9730, Trident on D1, T80, T300..), followed by the required information on a per disk basis (4 words for variant field). However, this new proposal allows run-encoding which will cut down considerably on the number of entries to the log.

This is partly because of the difficulty involved in working out an easy (elegant) way for Pilot to access Alto files to get the disk address. It was decided to pass this difficulty (and slowness) off to the debugger, instead of Pilot, since the debugger already has access to all of the Alto file machinery (this was proposed by McJones).

Another reason for adopting this scheme for Alto files is that since Pilot has no caching, Pilot startup would be slow due to swapping lots of code (this was pointed out by Wick).

Johnsson proposed that at startup the image file could be scanned to get all the code for the initial map entries; this would take about 2 seconds for large image files (McJones and Redell did not seem to like this idea). Since the image file is not moved, the disk addresses remain valid. However, Johnsson pointed out that this strategy does not work on check files.

Therefore, in order to be able to go through the Alto file machinery, an Alto entry must contain a file pointer and file page number (this allows run-encoding within Alto files, although we felt most mapping to Alto files would happen during start-up); a Pilot file will include a disk address and 3 words of check (file page number and 2 words of the file ID were suggested). The Pilot start-up time problem (mentioned above) will go away with the chosen scheme of logging since the slowness was only if Pilot had to log actual disk addressses for VM pages mapped to Alto files.

With each entry consisting of 6 words, there can be approximately 40 entries per page (~~40 spaces).

There was lots of discussion on what the check should consist of. We compared the Alto file label and the Pilot label to see if we could find an interesting word or two from each label, that was in the same position, so that we could check that word(s). This was too difficult; it seems that several variants are required, rather than simply a hash function on the labels or changing the format of the Pilot label to agree with the Alto.

(b) We agreed that the log would reside on the Pilot disk; and the debugger's map would reside on the Alto disk. If space gets extremely tight on the Alto disk, this file could be moved to the Pilot disk (it was pointed out that we are probably already over the size of an Alto disk anyway).

(c) Pilot will need to initialize the storage and the pointers in a section of memory that never gets re-mapped; the debugger will find the log by being passed a pointer to it as part of the 18 word **InLoad/OutLoad** message. By starting out with a virtual address and physical disk address as a "distinguished log entry", the entire table can get initialized from that (we have to watch out for how deep the recursion can get !!).

(d) It was suggested that Pilot consider maintaining an up-to-date copy of the entire map, however, McJones reminded us that it is just supposed to be a dribble file (to cut down on the amount of resident storage and/or the disk activity required by Pilot).

3. We need to get a better idea of the frequency of log entries so that we can determine how often it will be necessary to update it.
   - (a) How big is the log file?
     We propose that some heuristics be used for collapsing the map (since many modes of paging activity have high locality of reference). How much work is involved in this? Are there other optimizations that can be made?
   - (b) What to do when the file gets filled up - swap to the debugger? increase the size of the file (up to some maximum)?
     Who cleans it up (ie. is it sufficient to reset the log file each you enter the debugger or does it need to get done more than that)?
   - (c) The debugger needs to get access to the log file buffer so it can get *all* of the changes, not just the ones that have already been written in the file. How is this done?

(a) Much time was spent discussing the size of the log file and where it should be stored. We tried to figure out an estimate of how big the file might get; an upper bound of 16K map entries at 4 words per entry (64,000 words = 256 pages) was proposed at first (by Johnsson). We all agreed sometime later, that this upper bound was clearly too high and McJones thought that even 40-50 pages would be an extravagant size (due to run encoding and other heuristics to keep down the size). It looks like we will start off at about this size and do further optimizations at some later time if necessary. McJones will worry about swap units and what kind of Pilot file it should be (a volume file was suggested).

There was a suggestion to have the debugger's map run from page 0 to the highest virtual page ever seen since it is believed that it will take us quite some time to fill up all of virtual memory. This strategy has the disadvantage that it forces breakpoints and other such large memory allocations to remain towards the low end of memory (ie., McJones suggested that breakpoints could be put in the second 64K).

(b) When the log file gets filled up, Pilot will swap to the debugger (presumably in worry mode so as not to cause any more entries to the log), have the debugger update its copy of the map, and let Pilot go on. This update activity will also occur each time you enter the debugger.

(c) Redell proposed using a ring buffer strategy, with Pilot reserving two cells, for a reader and writer, pointing to this log; the debugger would also maintain these pointers. With this help, Pilot can swap to the debugger whenever its writer is reader-1. By knowing how much of the file the debugger has already updated, Pilot can use some heuristics (like collapsing all references to the same page within the one page buffer it is currently working in).

Further proposals from McJones regarding the format of the log:

The log file will be contiguous in physical address space (the base address must be determinable at disk initialization time, in order to miss bad spots). Then the base address of the log file can be passed to the debugger in the **OutlLoad** message (as well as length, reader, writer).

The ring buffer pointers should be thought of as (page number, entry) pointers into the log file; the debugger must also find the virtual (and if is swapped in) real memory adresses of page containing the "writer" pointer. Two possibilities are:

(1) Pilot maps a big space to the entire log file and creates uniform 1-page swap units; the debugger accesses this file through the standard virtual memory interface with the assistance of a fixed point log entry (passed through the **Inload/Outload** message).

(2) Pilot gives the debugger a distinguished virtual pointer that contains the "writer" (this way Pilot can have a single 1-page buffer space it "windows" along the log); in this case, the debugger cannot use the standard operations for accessing the buffer however (check if in core, ...)

These various proposals will have to be carefully examined from both the Pilot and debugger viewpoints to see what will work out best.

4. Presumably, when you are not debugging, you don't want to make log entries.
How do you turn the log file on and off?

When you turn the log on, you must go through the initialization code to get the debugger's map set up. When the map is off, the debugger will complain if it has to do anything interesting. The on/off mechanism will be worked out at some later time.

There was also some discussion abou the possibility of microcode swapping. Although there will be no microcode swapping included in this proposal, provisions will be made for adding this later (upward compatibility).

It was decided that McJones and Johnsson would meet at some later time to work out the specifics of this interface (with assistance from Redell and Koalkin).