Init\_units\_globals, proc #13?

Something is storing into addr 10 (or is it 5) that is putting a value of 1 into SEMCNT (count of semaphores). Where is that getting done?

* Crapping out in CurProc=9 (RLOCSEG)
* Getting a call to Kernel Proc #12 which doesn’t exist in CSPTABLE?
* 1st call to RLOCSEG coming from procedure labeled as #23 in MyDecoded.txt (actually LOAD). Compiled code accesses it as proc #9.
* In MyDecoded.txt @ E3C1 + a few. Crashes somewhat after the RPU
* Dying in Procedure #32 INIT\_TASKS
* TASKSW: needs to switch the execution to a different process
* SIGNAL switching from proc #33 (TASK\_START) to #27 (INIT\_KERNEL\_GLOBALS)
* Goes from SIGNAL in TASK\_START (#33) to INIT\_KERNEL\_GLOBALS? to WAITER (#49) and loops forever (APPARENTLY waiting for an event to break out)
* SysCom^.PoolInfo getting zapped at offset 102 in INIT\_SYSCOM
* Consider NOPs to bypass the call to WAITER. Remember to remove the NOPs.
* BACK to SIG- being called by an apparent TASK SWITCH triggered by reference to SC\_INIT which does not appear to be in an initialized segment (does this make any sense?). Need to determine exactly which semaphore has been triggered, where it was initialized. It should have something waiting on it to be triggered. It should be the Semaphore in SysCom (I think). Make sure the references are to the same location.
* First call to SIG occurs in procedure #13 (INIT\_SCREENOPS) -- inside call to SC\_INIT? In a Segment fault?
* SC\_Init is repeatedly being called which is causing a segment fault which should (?) be loading a new segment but there is nothing in the queue for the semaphore. There is no call to load a different segment.
* Where is the code to load a non-resident segment? In procedure LOAD.
* Look for pSysSrcs2-- It may contain source code for SCREENOPS. I cannot see any evidence that it is loaded. Should something be loading it?
* Breaks at SIG, FAULTCOM, WAIT
* Task\_Start @ 37: Initializing the semaphore wait queue
* Task\_Start begins, does the SIGNAL and then loops forever. - either SIGNAL should return it to the caller (init\_tasks) or the call to UNITSTATUS should signal or something. Note: VI.2.29 PROCEDURE SIGNAL ( VAR SEM: SEMAPHORE) SIGNALs the semaphore SEM. If no processes are waiting for SEM, the count associated with SEM is simply incremented. If one or more processes are waiting for SEM, then SEM is not incremented, and the process at the head of SEM's queue (the process with the highest priority) is added to the ready queue, where it competes with other ready processes for processor time. See Chapter IX.
* Look at FAULHAN code-- what is it supposed to do?
* MainTask.Local\_Data appears to have a pointer to the\_semaphore in LocalData
* RLOCSEG is called by LOAD. Is it called by anything else?
* StartingIPCofProcNumber - trace through this to see if result makes any sense

It makes sense but is incorrect. This bears further study.

* Save/Restore to Debug window - compare for equality
* A SCXG1 calls WAITER. Where is the SCXG1 located? The call is in INIT\_TASKS which calls TASK\_START. TASK\_START does a SIGNAL which returns control back to INIT\_TASKS (however, ProcCode (at least) is messed up).
* {insert this tib in the task list after maintask}: the\_tib.regs.task\_link:=maintask^.regs.task\_link; // is maintask^.regs.task\_link initialized to anything?
* Need to verify ENQUE- follow the TIB links
* Need to verify RESTORE - Why is ProcCode messed up?
* Is something crazy going on with the assm language returns that would cause the flow control to behave differently?
* Look into source code in psyssrc2 for inspiration
* Is there an 8086 emulator that I could trace through?
* Start task is expecting a return address on the top of the stack that must somehow be used as a queue pointer
* MemDump for queue/dequeue
* Look at all uses of ZF to ensure that they are consistant
* To mount all volumes in the p-system filer: (O(line, m(ount, System4:=SVOL) FOR EXAMPLE
* Start looking at 421.2F DOSBOOT which looks like the code seen when using emu8086 to try and run PSYSTEM.COM. Unfortunately, I cannot find any of the routines that get called in DOSBOOT.
* Assembler source code in P211FA
* PSYSTEM.COM constants start at about 7AA
* DOSBOOT-Assembled constants start at about AA6
* PSYSTEM.COM is copyrighted 1986 by Pecan Software
* DOSBOOT - copyrighted 1984 by Softech
* <http://pascal.hansotten.com/ucsd-p-system/ucsd-files/>
* COMPRESS.CODE is stored on 8086kit: