* Must be running PSYSTEM.EXE
* XI=UTILITY:U2 (to dismount/mount stuff)
* FILER
* Prefix 421.2F:
* Assemble DOSBOOT
* Link 421.2F:DOSBOOT.CODE, 421.2F:MSDOS.MISC → DOSBOOT.CODE
* 8086KIT:COMPRESS DOSBOOT.CODE → DOSBOOT.COM (100h)
* Use DOSFILER.CODE to transfer to DOSBOOT.COM → C:\PSYS\DOSBOOT.COM

TO MAKE PME (UTILITY:I=MAKEPME)

* Link: PME, RSP, FP4, DOSBOOT, TERTBOOT,CONFIGBIOS → TEMP2.CODE. Compress TEMP.CODE → SYSTEM.PME.86
* SYSTEM.PME.86 gets loaded over original DOSBOOT.COM
* What address does the PME get loaded to? (A: 0) Is it relocatable? (NO)

TO MAKE DOSBOOT.COM (UTILITY:I=MAKEBOOT)

* Link: DOSBOOT, MSDOS.MISC → TEMP.CODE. Compress TEMP.CODE → DOSBOOT.COM.

BUILDING

* Prefix to UTILITY:
* XI=L2 (LINK to DOSBOOT.CODE)
* XI=C2 (COMPRESS to SYSTEM.PME.86)

Assembled listing is PME-debug.txt on HPLaptop

DEBUGGING - Display standard registers (^F7)

DEBUGGING (yellow is the macro ^F8)

* Execute TD.BAT in the \psys folder
* Set a breakpoint at 0571 Run until the break.
* Step until boot has been moved to upper memory
* Set a new breakpoint at 0CB0 (call to DB\_DIR in DB\_START). Run until it is reached.
* Follow the jump
* Set a break at 0AC1 [JMPL (X\_PME)] which starts the tertiary boot. Run to the break. Step once to get to TERTBOOT.

DEBUGGING (yellow is the macro ^F9)

* I am debugging DOSBOOT.COM
* Execute TD.BAT in the \psys folder
* Use this as a watch point: IP-3AFE+01CE, or this (IP-3b71+025E) after the “DEBUG” jump?
* Change default folder to psys
* Set a breakpoint at 0571 Run until the break.
* Step until boot has been moved to upper memory
* Set a new breakpoint at 0CB0 (call to DB\_DIR in DB\_START). Run until it is reached.
* Step to 0CBC (remember to completely kill the CMD window if trouble occurs)
* Follow the jump
* Location 0A3F is in DOSBOOT (AT STEP 10)
* Set a break at 0AC1 [JMPL (X\_PME)] which starts the tertiary boot. Run to the break. Step once to get to TERTBOOT.
* The TertBoot seems to be located at 3A30 - first instruction is [JMP 3ACE]
* Set watch IP-3ACE+009E - Use “TERTBOOT-Assembled”
* Set a break at 3E95 (which is 0465 in the TERTBOOT listing). Run to the break.
* Set a break at CS:3EA3 (which is 0473 in TERTBOOT listing). Step once to get to FETCH. The PME-Assembled listing now corresponds to actual offsets (for a while)
* Set the data area to GOTO DS:SI - this should follow the code being interpreted
* Set a breakpoint in SIGNAL at cs:17D7 (17D5 in PME-Assembled)
* Set a breakpoint in SIG AT CS:1793 (1791 IN PME\_Assembled)
* When SIGNAL is reached, it shows a SemaPhore located at D952
* Put a break at 1757 (which is 1755 in PME-Assembled.text)

NOTES

* The following files look like they might contain the start of the tertiary boot: SYSTEM.PME.86, SYSTEM.CONFIG, 421.2F:AUTOBOOT.TEXT
* See PME\_Assembled:
  + IPCSAV @ 24
  + MP @ 2E
  + CURPROC @ 32
  + SEGB @ 2A
  + CPOFFSET @ 42
* REPLACE ac 97 d1 e7 2e ff 25 (LODSB; XCHG X,DI;SHL DI,1;JMP CS:(DI)-->

WITH:EA EA 06 02 C1 5B 90 90 ?? IS THIS THE CORRECT INSTRUCTION? PSYSTEM.COM PSYSTEMX.VOL

* UNITLIST may not be properly initialized? It appears to be initialized in INIT\_ENVIRONMENT around offset 393.
* DS:0 usually points to the start of the segment
* To display ASCII on laptop use: DS:SI,s or ES:DI,s for example

Host EVEC @ 3CCC

Vect\_Length=28

Map[1]=0598, [2]=06ac, [3]=0930, [4]=084a, [5]=07c0, [6]=076e,

* Delphi & Laptop are same in BYTECMP until around fDbgCnt 16331 (possibly after init UNIT\_LIST). Laptop is comparing SPDEVICE and REALOPS. Delphi is comparing ???????? and REALOPS.
* Delphi thinks that ped\_total\_evec\_words is 230
* [3] Currently broken in INIT\_ENVIRONMENT near “if host\_sib^.segname='KERNEL '”. Takes different branches at “FJP 454”. Delphi is comparing “KERNEL “ to “HEAPOPS “ but Laptop is comparing ‘KERNEL “ to “KERNEL “. Can I build DOSBOOT with debug printing? Could the discrepancy be related to not copying the boot directory to the stack?
* MSDOS.MISC loaded immediately following DOSBOOT.
* The directory BUFFER is located at DS:0D30
* [1] SYSTEM.PME.86 gets loaded from 0400 to 4400 (1024 to 17,408). SYSTEM.PME.86 starts with 303a 0000 9c0c 0000 e600 2600 (which is probably not code). 303a=BOOT; 0000=unused; 9C0C=NATRET; 0000=CS of interp; E600=IORSLT; etc.
* Reads SYSTEM.CONFIG (starting block = $0028 (40d), Number of blocks = $001B (27D))
* DOSBOOT includes its own tertiary boot (DB\_TERT?) which pushes BOOTPARMS onto the stack before branching to the address contained in the first word of the PME.
* Boot directory is pushed onto the stack. Is this what I do for p-System? What use is made of the stuff pushed onto the stack?
* First word of the PME gets patched to $3A30 (by loading SS:[0]) then takes FAR jump to $3A30 which has a jump to $3ACE (BOOT: JMP STEP1). I.e., the data segment info is getting patched into the code segment.
* Pop the BOOTPARM parameters into BIOSTB (which is located at 00EA). Source code is in BIOS.DATA.TEXT which is INCLUDed by CONFIGBIOS.TEXT.
* PME gets loaded at END\_PERM ($0400).
* My version of SYSTEM.PME.86 is one block bigger and the 1st JMP is incorrect.
* [2] RSP appears to be located at $2b3d-$2b25->18. Or $2b2f-2b17->18
* None of the variants of real size gives a size of 32 blocks to SYSTEM.PME.86 (FP0->29 blocks, FP2->31 blocks, FP4->33 blocks)
* Do I have the version of SYSTEM.PME.86 that has been fully patched or not? Yes. I think so. Appears to be OK up until somewhere between fDbgCnt in 800..900. See item [3].
* There is a discrepancy between the listing of RSPLIST (2).TXT and the actual executable code in the IOR function (address ~$2AFA). The listing also seems to be offset by 18 bytes.
* My TERTBOOT appears to load 58 bytes earlier than theirs
* Module load addresses (approximate)
* 4/2/2019- HPLaptop does not seem to be loading the directory of PSYSTEMX.VOL?

|  |  |  |  |
| --- | --- | --- | --- |
| MODULE | LOAD ADDRESS | END ADDRESS | LENGTH |
| PME | $0400 | $2674 | $2274 (8,820) |
| RSP | $2A9A | $2F6C | $04D2 (1,238) |
| FP4 | $2270 | $2AA1 | $0831 (2,102) |
| CONFIGBIOS | $2FB1 | $3A14 | $0A63 (2,664) |
| TERTBOOT | $3A30 | $3F74 | $0544 (1,693) |
| (UNKNOWN) | $3F74 | $7C00 | $3C8C (15,500) |
| DOSBOOT | $7C00 | $88C0 | $0CC0 (3,264) |

NEWSEGMENT

* BP = 1
* A Break at CS:0FAC is at start of NEWSEGMENT
* After reaching CS:101D (which is the start of BLDFRM)...
* [Ss:0032] is CURPROC- open a watch on this. Also open a watch on BP. At the start of BLDFRM, this is the procedure being called.
* [SS:003A] is EVECp
* ss:04B0 is the actual EVEC- position the data window to that address
* 003E is EREC (EREC appears to have 28 entries on the laptop)
* 0040 is OLDEREC
* 067C is new environment (ERECp)
* Get a data breakpoint on ds:04b0 (the EVEC) being changed in procedure #10 with SI=0CF4A which is a REPNZ MOVSW located at cs:194C which is in the MOVELEFT procedure. The current p-System procedure is #10 (INIT\_ENVIRONMENT). There are three calls to MOVELEFT in INIT\_ENVIRONMENT. The call to MOVELEFT is followed by a FJP call. The most likely call to MOVELEFT is the one that occurs at offset 343 in INIT\_ENVIRONMENT. Calls MYMOVE wanting to move 460 bytes (seems crazy). Src address is CE48 (psys). Destination address 03CC (psys). 03CC may be the EVEC but the map[] values all seem to be small numbers which do not agree with the MAP[] numbers above. *Try to figure out where ped\_total\_evec\_words comes from. Seems like it ought to be 28 words?*

Procedures called

Put a break at cs:101D (start of BLDFRM)

1->2, 2-26, 26->6, 2->10, 10->6 (EVEC set here in INIT\_ENVIRONMENT), 10->12

In BLDFRM, DI temporarily(?) has the start address of the procedure (PROCCODE) .To set a conditional breakpoint: [word SS:0032h] eq 012D (for example)

SPECIAL LOCATIONS

(see PROCEDURE NUMBERS SPREADSHEET)

BREAKPOINTS

|  |  |
| --- | --- |
| NEWSEGMENT | CS:0FAC |
| Entering BLDFRM | CS:101D |
| Exiting BLDFRM | CS:108c |
| Calling Opcode Procedure | CS:020CH |
| [WORD SS:0032] EQ 10D | Break when procedure number is 10D |
| [word ss:0e6+32d],2 | Break when SysCom.PoolInfo.PoolOutside is changed |

4/16/2019

* GOTOXY called and successfully loaded- not relocated
* Loading KERNEL (LOAD <- INIT\_INTERNAL\_POOL <- OS\_INIT <- USERPROG)
* Kernel is 2184 bytes, Base 1650, Seg\_Addr 251, Data\_Size 406
* Putting a break when entering procedure #24 (SC\_INIT?) WHICH DOES NOT CORRESPOND TO ANY OF THE PROCEDURES IN SCREENOPS.CODE? ALSO, IT DOES NOT SEEM TO AGREE WITH WHAT HPLAPTOP IS DOING!
* The crash is occurring at fDbgCnt 20211 in proc #24 Ofs 27
* On the HP, the first instruction in proc #24 is DS:DI=0 D9 03 81 0A which appears to be Proc #24 (!) in ScreenOps. This, in turn, appears to be SC\_INIT:

*PROCEDURE sc\_init;*

*BEGIN*

*sc\_ansi\_crt:=false;*

*END {sc\_init};*

4/17/2019

* On HP, the code is located at DS:029A. Appears to have been loaded when exiting proc #10, fDbgCnt = 16140. The actual address to watch is: 65BB:029A
* The SCREENOPS segment seems to have been loaded when exiting procedure #64 which returns to procedure #3 at offset 175 (FLIPSEG). Already changed when *entering* proc #64 (fDbgCnt = 19983)?
* Changed in proc #3 with offset 3. Possibly in CALLIO from READSEG? fDbgCnt ~19977. Do I need to break on each call to READSEG?
* Could the problem possibly be related to SYSCOM being located at address 0?
* Try putting in the code @ $0221 (mov bp,[cs:0032]; call 5bc1:108f; jmp 210) to better track where the code is executing

4/18/2019

* The code AT 65BB:029A is loaded during the third call to READSEG.
* SCXGx is missing a SHR DI,1- This seemed significant but wasn’t because HP is doing a SHL DI,1 before indexed jump, but I am not.
* There seems to be a discrepancy in SCXG2 in what is executed in NEWENV. Delphi has exited NEWENV back to NEWSEGMENT but HP is still in NEWENV? On HP NEWSEGMENT falls into NEWENV rather than calling it.
* Call to POOLBASE was preceded by pushing DS so that ES could be copied to DS. Why? Neither DS or ES seems to be used in POOLBASE?
  + First call to NEWSEGMENT occurs @ fDbgCnt 18202
  + Second call to NEWSEGMENT occurs @ fDbgCnt 18313
  + Next is the first call of SCXG2
  + Third call to NEWSEGMENT occurs @ fDbgCnt 20187
  + Then comes the crash at fDbgCnt 20211
  + The search string “0 D9 03 81 0A” never seems to have gotten loaded?
  + On the HP, it seems to have been loaded by fDbgCnt 20119
* SEGFAULT never gets called on Delphi?
* In FAULTHAN, Delphi goes to proc #33 (TASK\_START), OFS: 104 (following the call to SIGNAL) but HP goes to Proc #48,6. Both end up in TASK\_START (but HP has weird offset). Actually appears to be the call to WAIT @ 105. Back to INIT\_TASKS (HP now gets PROCCODE corrected). Unknown proc in FAULTHAN calls another UNKNOWN proc which call proc #6 (which is NOT FILE\_PRESENT) @ fDbgCnt = 18459. Getting as far as the call to SCIP1 IN PROC #34 (INIT\_UNITS\_GLOBALS) which appears to be calling INIT\_SCREENOPS (#13). There is also a TASK\_SWITCH that is occurring (at least on HP).
* After the call to NEWSEGMENT in proc #13, HP is going to SEGBACK. Delphi is NOT!
* SEGBACK raised in proc #13 offset 0. The task switch (if it is occurring) is not forcing the segment to be loaded.

4/19/2019

* The SCREENOPS segment does seem to have gotten loaded at $687A
* The HP never executes Proc #60
* The call stack for Proc #60 is wonky-- probably because I don’t have procedure names defined.
* To compile KERNEL.TEXT I need original versions of PED\_DEFS,

4/20/2019

* I need a way to see the SEGMENT names and procedure names
* When entering READSEG the third time (presumably for SCREENOPS), the call stack is wacky: 3:FLIPSEG <- 50: ???? <- 27: INIT\_KERNEL\_GLOBALS
* Procedure number 3 is not FLIPSEG but some procedure in SCREENOPS
* Maybe procedure #27 is NOT INIT\_KERNEL\_GLOBALS, but some procedure in either SCREENOPS.CODE or KERNEL.CODE (INIT\_SYSCOM?)
* Unimplemented OPCODES: LDM, SRS, UNI, CXG. Exiting procedure #52 (POOL\_ADJUST) to procedure #58 (FINISH\_ERROR) seems to give an absurd value to IPC offset.

4/22/2019

* See the file “F:\NDAS-I\d7\Projects\pSystem\UEDIT.PRJ” for a list of files opened in UEDIT
* First call to LDM occurs in Delphi @ P#60 [PROMPT\_FOR\_VOLUME], O#43 but occurs at P#13, O#9 on HP
* My decoded procedure #60 (prompt\_for\_volume) does not exactly match the version that I decoded from SYSTEM.PASCAL (although they are very similar).
* The call stack is: PROMPT\_FOR\_VOLUME <- NEED\_VOLUME <- CHECKSEG <- FLIPSEG <- POOL\_SEG <- FAULTHAN <- INIT\_KERNEL\_GLOBALS
* Appears to be planning to for volume PSYX to be mounted
* HP NEVER calls procedure #60 [PROMPT\_FOR\_VOLUME]
* Opcode SRS called substantially different fDbgCnt and appears to be working on different source code location
* UNI is untested

4/23/2019

* The call SCXG1 (IN screenops?) is going into an infinite loop.
* In Delphi, the call (Proc #13, Ofs:0) is [SCXG1 24] . HP is also doing [SCXG1 24]. HP is doing a task switch. Both get to a task switch at fDbgCnt = 18200.
* Neither Delphi OR HP seems to have a correct PROCCODE after the call to RESTORE. Both say that the next OpCode is $60 (96) which is an SLLA, Both (Delphi/HP) have the instruction sequence (SLLA1, SCXG1, LDL ⇒ 60 70 31 9c 9c 9c) which I cannot locate in the code. Because after SLLA1, SCXG2 switches to FAULT\_HANDLER.
* [Delphi] Is the call to WAIT leaving ProcCode unresolved?

4/24/2019

* Following the call to 2ND call to RESTORE, control is jumping to proc #27, weird offset, leading to the above instruction sequence (SLLA1, SCXG1). This is in INIT\_TASKS. Switches to FAULT\_HANDLER. Going into a infinite loop somewhere near the SIGNAL.
* Around fDbgCnt 18,600, there is a call to MOVESEG
* First call to SCXG2 occurs at 20187. Delphi shows fDbgCnt of 20187 (HP shows 19831) but both show Proc #13, Ofs: 0 which has the call to SCXG2 (SC\_INIT?) in ScreenOps. Delphi enters Proc#13 Ofs#0 with fDbgCnt of 20187, HP enters P#13 O#0 with fDbgCnt of 19789. Both show call to SCXG2. Delphi gets caught in a loop. HP invokes the fault handler.
* With a break set on Proc#48, O#0, Delphi breaks at fDbgCnt 18314, HP at 18272. This is where the segment fault occurs.

4/25/2019

To DO:

1. Trace through seg fault (seems to work identically)
2. Look for fDbgCnt diffs-- break at SCXG2 and then do modulo on fDbgCnt
3. Stop in loop and re-do with breakpoint

Notes:

* Could my not using Segment registers be causing the loss (or overflow) of address information? Should the HP code inside the .IF EXTMEM be skipped?
* Following the RESTORE, IPC claims to be in procedure INIT\_KERNEL\_GLOBALS (proc #27) but it is actually in (#32?) INIT\_TASKS (which gives an Ofs of #8 which makes sense)
* Putting a break at SCIP (SCPI on HP) for procedure #34 breaks on both Delphi and HP at fDbgCnt of 18770 (Ofs #111: FILE\_PRESENT), (Ofs #122: UNIT\_PRESENT) and (Ofs:603: SC\_INIT).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Proc | Actor | fDbgCnt | Ofs | OpCode |
| 34 | Delphi | 18770 | 111 | SCIP1 |
| 34 | HP | 18770 | 111 | SCIP1 |
| 34 | Delphi | 19627 | 122 | SCIP1 |
| 34 | HP | 19627 | 122 | SCIP1 |
| 34 | Delphi | 20060 | 141 | STO |
| 34 | HP | 19703 | 141 | STO |
| 34 | Delphi | 20186 | 603 | SCIP1 |
| 34 | HP | 19829 | 603 | SCIP1 |

* It appears that the UNIT\_LIST is different between Delphi & HP which causes UNIT\_PRESENT to behave differently?
* QUESTION: does the procedure being called @ Proc #34, Ofs:603 get called the same on Delphi as on the HP? Is this really a call to INIT\_SCREENOPS or something else? It appears to be a call to PTR\_SUB rather than INIT\_SCREENOPS?
* At procedure #6 (unsure of which segment), I am getting a sloppy error message: “Need segment. Need segment SCREE ,, Put volume Put volume [PSYS PSYSX] in drive #4 then press <space>”
* The above error message is actually coming from procedure #60 (PROMPT\_FOR\_VOLUME) which was called from procedure #59 [NEED\_volume] which was called by procedure #64 [CHECK\_SEG].
* Debugmessage: Untested SRS
* DebugMessage: Untested UNI

4/26/2019

* Call to GOTOXY (#2) from procedure ERASE\_ERROR\_LINE (#56) seems to erroneously be calling SC\_INIT (which has the same procedure number).
* 1st Call to READSEG comes from proc #9 (LOAD), offs #4. fDbgCnt same on Delphi and HP. Reading Unit #4 on Delphi and #12 on HP. Both want 64 words in segment. Both show a starting block number of $105 (261.) which is in SYSTEM.PASCAL at an offset of 261-247=14 which appears to be GOTOXY. This should be at the start of block 14:

0: 3E00 0000 474F 544F 5859 2020 0100 3A00 0400 6710 >---GOTOXY --:---g-

10: 6516 e-

Delphi says to load segment at 0 as does HP. After call to POOLBASE, Delphi still has address of 0 but HP has address of $5F54 (because HP has SS=$5F54). ES set to 0 (Delphi) $5F54 (HP). Delphi shows buffer address to be 11472 ($2CD0). HP also has $2CD0. Both have successfully loaded the segment at the desired address. HP returns to STFRSLT. DELPHI DOES NOT! Possibly fallen into from CALLIO? The following calls should exit to STFRSLT:

UWAIT should exit GETU to RSTFETCH

UCLEAR, UCLR and SYSCLR should call CALLIO and exit to RSTFETCH

UIO should exit RSTFETCH

SYSIO should exit to BX

SYSRBOOT should exit SYSIO to whoever called SYSRBOOT

* The call to RLOCSEG in procedure LOAD (of GOTOXY) is weird. Only moving two bytes and then seems to be weird SI after return?

4/27/2019

* Following the call to READSEG, both systems advance to SCIP1 (in proc #9:LOAD) and descend into procedure #3:FLIPSEG. Both exit FLIPSEG without doing much of anything. Both call RLOCSEG at ofs 14 in 9:LOAD. In RLOCSEG, the address from “add offset in pool” is very different. Delphi seems to be referring to the just loaded GOTOXY. The function POOLBASE behaves differently because of the “IF EXTMEM”. Even with POOLBASE no-opped, getting different results

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Block # Delphi | Block # HP | Nr Bytes Delphi | Nr Bytes HP | Addr Delphi | Addr HP |
| GOTOXY | 261 | 261 | 128 | 128 | $2CD0 | SS:$2CD0 |
| KERNEL | 251 | 251 | 4368 | 4368 | $1730 | SS:$1730 |

* What effect does my not using segment registers actually have?
* In 56:ERASE\_ERROR\_LINE the call to CXG ?? at offset 3, is not a call to PUTPOOL but a call to GOTOXY.
* Current call stack:

'Call stack @Addr $0000 [ 0]: 58: FINISH\_ERROR <- 60: PROMPT\_FOR\_VOLUME <- 59: NEED\_VOLUME <- 64: CHECKSEG <- 3: FLIPSEG <- 50: POOLSEG <- 48: FAULTHAN <- 27: INIT\_KERNEL\_GLOBALS'

* When entering procedure #58, DS, ES and SEGB point to $1730 which refers to KERNEL. Can I use this to determine which procname to use

04/29/2019

* P #60, Ofs #122 is writing 'Need segment ' but it does not yet show up in the window.
* The constant pool: Constant Pool: Globals.LowMem.SEGB + Globals.LowMem.CPOFFSET
* Something in Constant Pool: Globals.LowMem.SEGB + Globals.LowMem.CPOFFSET + something\_offset within constant pool
* Check UNI (once again) to verify that it is actually working. SRS, UNI and ADJ all seem to be working. I think that STM is working.
* HP never goes to UCLEAR
* CHECK\_SEG compares (nothing, ‘SCREENOPx” (multiple times))
* I could try to enable some of the debug code in KERNEL.

4/30/2019:

* Occurring in ADJ @ FillChar(Bytes[ES+DI], AX, 0);
* I inadvertently removed SYSTEM.STARTUP from PSYSTEMX.VOL (by using an untested “Rename file” in FILER.EXE
* Range check error: fDbgCnt=28126, CurProc=2:OSINIT, Offset = 62753, DS=$0, SI=$FF6e. Occurs in procedure ADJ at line 6420 (FillChar(Bytes[ES+DI], AX, 0);)

5/3/2019

* Occurring in ADJ @ Fillchar(Bytes[ES+DI], AX, 0)
* AX = 0, DI = $FF5E, ES = $7040
* Call stack: ' 2: OS\_Init <- 51: COMMAND <- 1: USERPROG <- 2: OS\_Init'

5/20/2019:

* Occurring in ADJ @ Fillchar(Bytes[ES+DI], AX, 0)
* Last segment loaded was GETCMD

5/21/2019

* EREC.ENV\_SIB^.SEG\_NAME

5/22/2019:

* DebugWindow won’t print to file- everyline appears to be empty
* Code in READSEG appears to understand the EREC and SIB and correctly display both. Similar (?) code in ProcName() does not work.

5/23/2019:

* fScreenBuf was never getting set in pSystemWindow

5/24/2019:

* TIB.ENV
* Globals.Lowmem.CurTask^ → TIB
* Getting the segment name (step 3 is not right):

1. Move(Bytes^[CurTask], TIB, SizeOf(TTib));
2. Move(Bytes^[TIB.Regs.Env], EREC, SizeOf(EREC));
3. Move(Bytes^[EREC.env\_sib], SIB, SizeOf(TSib));

This code is currently in READSEG

5/25/2019:

* No progress- trying to determine current segment name. Not clear how to get current EREC. Should relate to the env field in the MSCW.

5/28/2019:

* Changed the code in ProcName() to default to the Kernel segment

6/3/2019:

* Seems like something has already mangled memory by the time that I get to sets DIF “setup” call to MemDump
* Something is changing SP to $FFFE (65534) and I cannot determine where it is happening
* In “SetValue”, i=4994?
* At fDbgCnt=5000, aCaption is crap and everything else is crap
* Problem is occurring after fDbgCnt = 4800 and 4905. Close to fDbgCnt = 4872. fDbgCnt may vary some (fDbgCnt = 4875)
* Be sure to do a VisDiff before replacing Interp7

6/26/2019:

* Remember that the offsets on HP Laptop are probably wrong! Trying to debug will probably crash unceremoniously!
* Why is debugging running so slowly? Because {$DEFINE TEMPORARY} was calling MemDump thousands of time.

6/28/2019:

* In DOSBOOT, patched jumps are like “JMP 5BC1:0210” but after loading the needed registers are (today) DS=ES=SS=CS=5B81. These registers need to be pointing to 5BC1.
* trying:

|  |  |  |  |
| --- | --- | --- | --- |
| ip | New code | New code (hex) | Old code |
| 100 | MOV AX,5BC1H | B8 C1 5B | E9 AA 0B |
| 103 | PUSH AX | 50 |  |
| 104 | POP DS | 1F |  |
| 105 | PUSH AX | 50 |  |
| 106 | POP ES | 07 |  |
| 107 | PUSH AX | 50 |  |
| 108 | POP SS | 17 |  |
| 109 | JMP 5BC1H:0CADH | EA AD 0C C1 5B |  |

The code at 0CADH:

CALL 04EC

CALL 0616

CALL 0712

CALL 0746

JMP 0A3F

The above does not work - presumably because the CS register does not correctly point to where the code actually is.

* Check ADJ Proc #2, Offset #23, fDbgCnt = 28126 (This is probably not actually OS\_INIT). Appears to be requesting an impossible set having 0 bytes? Cannot determine actual source code corresponding to the sequence SRO; LAE; SLDC0; ADJ. Appears to be requesting and adjustment of set size to 0.

6/29/2019:

* Remember that there was some startup file that I deleted on HPLAPTOP that changed the startup flow.
* Source code for above item appears to be procedure #2 in GETCMD at an offset of about 23. Source code is SYSSEGS.TEXT. Volume appears to be FCONVB.
* SEGNAMES[] array can be used to calculate the segment number more cleanly.
* ERECp is NOT being used in PROCNAME() call.
* The current version of PROCNAME() will probably give incorrect results when trying to display the call stack and probably other information.
* Why are the first two things on the call stack always USERPROG.USERPROG?

7/1/2019:

* Try to examine the source code
* Look into DS register being saved and restored. Can it give me a better handle on how to properly display the call stack?

7/2/2019:

* Maybe the segment registers DS, ES, CS could be set on HPLAPTOP BEFORE dosboot.com gets loaded. Perhaps in a macro. This does not work. After opening DOSBOOT.COM, all of the segment registers get restored to their previous values. Maybe it DID work. After doing it once, then exiting TD and then reloading and doing an F9, the CD register has somehow magically gotten set to $5BC1 which is what I wanted it to be.? No idea how this happened.
* Does the HPLAPTOP also show a call stack that includes 1:USERPROG.USERPROG twice? See to comparable call stacks in file “Procedure Numbers” tab “RPU Call Stack”
* Cannot find any real source code for SEGMENT PROCEDURE GETCMD. See file SEGNAMES.TXT for decoded versions of GETCMD.
* I don’t trust the names that are showing up in the call stack. Need to verify. Perhaps I can set Delphi breakpoints when entering various KNOWN procedures. These seem to be OK now.

7/3/2019:

* Finishing the CallStack when the dynamic or stack link points to the MAINMSCW
* Need to try and determine what the actual procedure is when the crash occurs. Appears to be in GETCMD.PROC #2. Maybe GETCMD.EXECERROR?? Procedure appears to be about 923 bytes long. See SEGNAMES.TXT. Appears to have several calls to MOVESEG and several pointer operations. Do the pointer operations correspond to the PTR\_GEQ (etc) source code that I have? Appears to be PROC #2 in GETCMD which is the exact right length and procedure number. Source code also matches but I don’t know what the code is doing. It seems to involve calls to CUPOPS, MSGOPS, ASSOCIAT, SCREENOP, GOTOXY, PASCALIO,

7/5/2019:

* Renaming SYSTEM.STARTUP to STARTUP.CODE crashes at fDbgCnt = 29095 rather than @ 28126, however, the call stack looks the same: FJP SLDO2; SLDC0; ADJ. Could procedure #2 actually be INIT\_CUPOPS? Seems to be too long, however- INIT\_CUPOPS is a very short procedure.
* On HPLAPTOP, volume PSYSTEMX.VOL will not boot unless run from the debugger.
* Can I just NOP out the call to INIT\_CUPOPS?
* Currently getting as far enough to display the welcome message:

Welcome to the Power System (™). Regents l Copyright 1987 Pecan Software

Version [IV.2.2 R1.1]

System Date is 5-Jul

* This is where the crash is occurring:

' @Addr $7A9C [31388]: 31 A5 01 9B 01 66 00 C7 08 8E 08 00 D9 01 63 00 D9 01 6E 9A 01 01 E7 1D 01 06 00 CA 90 11 31 00 B0 31 01 B0 A0 D4 47 31 00 B1 85 5A A1 85 5B A1 D4 02'

* Could Procedure #2 be GETCMD.EXECERROR? EXECERROR is only about 173 bytes long. Too short!
* There are many indications pointing to ADJ being where a number of crashes have occurred!
  + The requested set size is 8 bytes
  + The actual set size is 0
* Something flakey going on with byte counts in ADJ. Compare to what happens on HPLAPTOP, Do they:
  + Move the same number of bytes?
  + Do they zero the same number of bytes?
  + Are the addresses the same (or at least similar)?

7/6/2019:

* Segment register in the FILLCHAR in ADJ does not seem to be correct. Should have been using SS rather than ES.

10/22/2019:

* KEY\_FILL seems to be getting called repeatedly- put a break into CXG- near fDbgCnt = 29335, Ofs = 14
* “Local number for target segment”

10/23/2019:

* Really need a more reliable way of listing the current segment name (particularly in ProcName()). DS = ES = SEGB (+4) to get segment name.
* Debugger.pas?
* What gets pushed onto the stack in a call to BldFrm? Must include a pointer to SEGb somehow- perhaps indirect?
* Look up Environment Rec & Environment Vec
* Step through WELCOME\_MESSAGE

10/24/2019:

* Crashing at fDbgCnt 94868. No new opcodes since
* Watch List: MemDump(SP, 'W', 'Stack') is giving “Delphi exception ERangeError at $B78F905”. SP is $FFD8 but HIMEM is $FFFE
* GETCMD.EXECERROR is getting called

10/25/2019:

* Makes it to the call to S\_GETCMD
* But never makes it past there
* See GETCMD1.TXT, SYSTEM-LISTING.TXT

10/26/2019:

* Check out PTR\_SUB to see if it is actually working
* Makes it to the call to S\_GETCMD which invokes the faulthandler. Faulthandler skips over the call to moverite at offset 42. Fault is a segment fault $80. Fault handler calls POOL\_SEG. (Possibly) POOL\_SEG follows a different route starting around offset 189? GetCmd.EXEC\_ERROR gets called when trying to call PTR\_GEQ at offset 5. CUPOPS.PROC #07 GETTING CALLED from GETCMD.EXEC\_ERROR (proc #2). Multiple calls to INIT\_CUPOPS?
* The call to FILLCHAR from CUPOPS.PROC #7 appears to be giving a rangecheck error on the line “FILLCHAR(Bytes[ES+DI], CX, AL);”. ES = 0. DI = 65490. CX = 6; AL = 0. But Bytes[ES+DI] does not give an error.
* Why is the date not the same as on HPLAPTOP?
* Why does the HPLAPTOP exit when not running under the debugger?
* “CUPOPS” - concurrent user ops

10/28/2019:

* Seems to be trying to call PTR\_SUB in EXEC\_ERROR but ends up calling something in CUPOPS?
* Goes into the death spiral at around fDbgCnt 29351
* MyDecoded.txt INIT\_SYSTEM does not agree at all to the code executing in PSYSTEMY.VOL. Much of this is due to changes where I patched SYSTEM.PME.86 and other changes that I updated the OS with. Still have the odd fact that the patched code works under TD.EXE but not without TD.EXE.
* Neither of the two versions of EXEC\_ERROR (or EXECERROR) seems to agree with my decoded code.
* Trying to boot using PSYSTEM.VOL (rather than PSYSTEMX.VOL) crashes almost immediately in USERPROG.INIT\_EXTERNAL\_POOL. Is this because I have disabled external pool in PSYSTEMX.VOL? NO. Even after disabling extended memory, the same crash occurs in INIT\_EXTERNAL\_POOL. Why is INIT\_EXTERNAL\_POOL even getting called?
* When using PSYSTEMY.VOL (psystem.vol external pool turned off), it seems to go off the rails and end up in what Delphi thinks is INIT\_EXTERNAL POOL. It doesn’t match anything that I can find.
* GETCMD.EXEC\_ERROR is called from KERNEL.COMMAND
  + (print spooler should not be enabled)
  + syscom^.spool\_avail IS FALSE
  + Delphi getting a segment fault when trying to call s\_getcmd but HP doesn’t? Check this out!
  + Try to put a breakpoint into NEWENV on both systems

10/30/2019:

* In FaultHandler
  + 142: if nofit or pooloverflow then
  + 255: syscom^.miscinfo.nobreak:=save\_nobreak;
  + 262: JPL 2
  + 5: WAIT
  + 6: SLDO1
  + (magic task switch)
* Off to KEY\_FILL *forever*
* In routine WAIT, Delphi is loading SSDSVAL as 0. However, HP has a value of $5F54 (@ location CS:[0318h]. It has this value IMMEDIATELY after bootstrap runs-- even before executing ANY p-Codes. This appears to be the SS, DS value for which (0) may, in fact, be the correct value on Delphi.
* After the task switch, HP ProcCode offset is wrong. No real surprise there.
* Both Delphi and HP seem to have gotten back to USERPROG.TASK\_START after the task switch.
* On the HP, I am putting a break into WAIT @ $1757

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Wait Call # | Proc # | Proc Name | Delphi | HP | Delphi fDbgCnt | HP fDbgCnt |
| 1 | 33 | TASK\_START | 105 | 105 | 18783 | 18783 |
| 2 | 48 | FAULTHAN | 5 | 5 | 18902 | 18902 |
| 3 | 33 | TASK\_START | 105 | 105 | 18905 | 18905 |
| 4 | 48 | FAULTHAN | 5 | 5 | 19244 | 19244 |
| FAULTCOM entered |  |  |  |  |  |  |
| 5 | 48 | FAULTHAN | 5 | 5 | 20701 | 20701 |

* FAULTCOM:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 39 | INIT\_CUPOPS | 0 | 8? | 28023 | 20828 |

* Both Delphi and HP seem to be executing the same p-Codes in INIT\_CUPOPS. But Delphi has fDbgCnt of 28026 and HP has fDbgCnt of 27984 (42 opcodes off).
* FAULTCOM entries:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 33 | INIT\_SCREENOPS | 105 |  | 20414 | 20414 |
| 2 | 39 | INIT\_CUPOPS | 8 | 8 | 28028 | 28028 |
| 3 | 51 | COMMAND | 12 | 12 | 28803 | 28803 |
| 4 | 20 | GetCmd.PTR\_GEQ | 13 | 13 | 29267 | 29267 |
|  | 48 | FAULTHAN | 6 | 56 | 29269 | 29269 |

* Incidentally, the system date now seems to be correct?
* Both Delphi & HP seem to be pointing at the same p-Code sequence but PROCCODE on HP, seems to be incorrect? After the call to the fault handler, PROCCODE is not getting set properly.
* After this, MEMLOCK\_SEG call on both with same dFbgCnt (29302). Crash is occurring about 50 opcodes from here (29351).

10/31/2019:

* Break when entering FAULTHAN is: [word ss:0032h] eq 48D
* Break at offset 53 in FAULTHAN IS : (si-[word ss:002ch]-1) eq 53D
* I think that my previous idea about paths diverging at the XJP were incorrect
* A POOL FAULT IS occurring. POOL\_ADJUST has been entered.
* Final entry to FAULTHAN occurs at fDbgCnt 29093

FAULTCOM ENTRIES:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Proc # | Proc Name | Delphi | HP | Delphi fDbgCnt | HP fDbgCnt |
| 1 | 13 | INIT\_SCREENOPS | 0 | 0 | 20414 | 20414 |
| 2 | 39 | INIT\_CUPOPS | 9 | 9 | 28028 | 28028 |
|  | 51 | COMMAND |  | 12 | 28785 | 28785 |
| 3 | 51 | COMMAND | 13 | 13 | 28803 | 28803 |
| 4--> | 20 | GETCMD.PROC #20 | 13 | 13 | 29267  29249  Note 1 | 29267\*  29249 |
| 5 | 20 | GETCMD.PROC #20 | 13 | 30 | 29351 | 29635 |

* At offset 32 in FAULTHAN (fDbgCnt = 29291), the two stacks do not agree.
* At point 4 (above, fDbgCnt=29249) the two stacks DO agree
* (0007 0000 0ED2 FF5E FF70 FF70 0B0A 095C).
* After FAULTCOM (@ fDbgCnt 29270), the stacks do not agree-- maybe because of the task switch?

11/1/2019:

* Note 1: Today both systems ran to 29249 on step 4. Yesterday they ran to 29267-- 18 fewer instructions? Both stacks agree on the first 8 values on the stack.
  + At the task switch point, both stacks agree
  + After the SAVEREG & RES\_JT in TASKSW, the stacks agree. Delphi SP = $FF2A, HP SP = $FF2A.
  + After call to RESTORE, the stacks do not agree!
  + HP has a return address on the stack
  + Both return from the call to RESTORE to TASKSW
  + However, Delphi is returning to KERNEL.FAULTHAN @ offset 6, BUT, HP appears to return to FAULTHAN at offset 56, BUT because PROCCODE did not get updated following the RESTORE, the actual offset is ALSO 6.
  + At the SIG call (SP=FF2A: 0007 0000 0ed2 ff5e ff70 0b0a 095c) on both
  + After the call to SAVEREG, both stacks are the same
  + After the call to RES\_JT, both stacks are the same
  + After the call to XQUIET, both stacks are the same
  + At the point where SP gets reset in RESTORE, the two stacks agree.
  + Delphi changes SP from $FF2A to $2C86
  + HP change SP from $FF2A to $2C86
  + However, the restored stacks do NOT agree:
    - Delphi: SP=$2C86: 0170 0000 12E6 DA42 001B 6670 0001
    - HP: SP=$2C86: 0170 0170 12E6 DA42 001B 6670 0001
    - Delphi: returns to TASKSW
    - HP: returns to FAULTCOM:
    - Delphi: RESTORE is called from within TASKSW (line 8106)
    - HP: RESTORE is called from within TASKSW (@$1804)
    - Delphi: RESTORE returns to line following the call to RESTORE (Line 8106)
    - HP: RESTORE returns to where is should in TASKSW
    - At KERNEL.POOLSEG offset 149 the stacks do agree. HOWEVER, the mis-matched parts of the stack have already been popped.
    - Both make it to KERNEL.LOADSEG (ofs 0, fDbgCnt = 29427 which is beyond my previously specified crash site which was 29351).
    - Somehow, the call to READSEG @ LOADSEG offset 12, didn’t actually seem to call anything on either system? How is that even possible?
    - Once again, the call to GETPOOL @ OFFSET 15 in CHECKSEG seems to have gotten ignored?
    - I have not decoded #53:FLIPSET in KERNEL
    - The reason that I thought READSEG and GETPOOL were skipped is because they are both INTERNAL procedures which do not process any p-Codes.
    - READSEG exits back to POOLSEG (fDbgCnt = 29504) and everything seems OK. Same Opcode (150), Procnum (50), offset (326) and fDbgCnt (29504).
    - Delphi ends with “No procedure” exception at fDbgCnt 30093, curproc=#21:SC\_PROMPT, OFFSET=22, call to SCXG1. Stack: (SP=$Fd6E: 0000 0001 FD9A 003A 0000 008f 0000 0BE0…)

11/2/2019:

* History appears to end in SCREENOP.SC\_PROMPT with calls APPEND but MyDecoded.txt does not agree with history.

|  |  |  |
| --- | --- | --- |
| Caller | Callee | at |
| #64:CHECKSEG | GETPOOL | 15 |
| #3:loadseg | #53:flipseg | 179 |
| #53 | #15 | 112 |
| #3:loadseg | rlocseg | 182 |
| #48:faulthan | memlock\_seg | 140 |
| #20:key\_fill | NONSENSE |  |

* Date showing at boot appears to be the date of last boot
* SC\_ERAS\_EOS calls other things but it does not appear to have been called. Everything it calls appears to be BAD\_BASE (=0).
* Things appear to go wonkey around fDbgCnt = 22212
* Trying to use CallsLogFile.txt
* Pcode calling native code exception
* Look for references to “BAD\_BASE”
* Memory leak related to {$IfDef LOGCALLS}
* frmDecodeWindow being freed without ever being created? Perhaps this is because of the use of MemDump(xxx, ‘O’, …) in the debugger.

11/4/2019:

* Stacks do not match following the FAULTCOM at 29249 when the PROCCALL is reached.
* Stacks do not match when exiting RESTORE
* BAD\_BASE already showing when I reach restore at 29250
* The MouseTrap is sprung when popping a word off of the stack: Proc #20 (GetCmd.Proc #20), IPC = 13, fDbgCnt = 29249. This occurring in SEGBACK during an exception PRIOR to calling FAULTCOM and prior to calling TaskSwitch.
* Delphi: SP=$2C86: 0170 0000 12E6 DA40 001B 6670 0001.
  + Where do each of the values get used?
  + $170 is popped at proc #65, offset 22, fDbgCnt = 29296. This supposedly contains the Static Link and is discarded in RPU when exiting MEMLOCK\_SEG which was called by FAULTHAN
* On the HP, break when (TOS eq $170) and (TOS+2 = $170) ⇒ actually seems to be true almost immediately → before executing ANY opcodes on BOTH systems. SP = $DA32
* The $DA40 on the stack appears to be Globals.LowMem.ERECp
* Watch for references to SizeOf(EVec) or SizeOf(ERec)
* In the initialization code, I changed a couple of uses of SizeOf(TErec) and SizeOf(TEvec) to correspond to the hard- coded constants previously used.
* The stacks still differ at the ProcCode call following FAULTCOM:

D: Stack @Addr $2C84 [11396]: 00E6 0170 0000 12E6 DA42 001B 6670 0001

H: Stack @Addr $2C84 [11396]: 00E6 0170 0170 12E6 DA42 001B 6670 0001

* What is the Delphi 0000 used for? Try setting a breakpoint at fDbgCnt = 29296 (as indicated above). Alternatively, use the mousetrap to look for the pattern. The pattern is first set in KERNEL.FAULTHAN, offset = 8, fDbgCnt = 18946 when storing something to syscom^.fault\_sem.message
* The 0 location is getting set to 55630 [$D944] in BldFrm trying to save mscw.static (PUSH SAVESTAT)

11/5/2019:

* The 0000 on the stack is getting set in BldFrm when pushing SAVESTAT.
  + Coming from “Inner” following XCHG(fAX.w, fBP.w), i.e. it is coming from AX which should contain the procedure number
  + TOS should contain the static link which comes from Globals.LowMem.Base (which is 0).
  + Globals.LowMem.Base is set to 0 via:
    - DX := WordAt[SS+BP+ENVDATA]; //^GlobalDataSegment
    - CurProc = 13 (INIT\_SCREENOPS)
    - OffSet = 781 (GoToXY)
    - fDbgCnt = 22350
  + HP is doing the same thing?
* There does not seem to ever be any attempt to fetch the 0 at $2c88
* $D944 put on BldFrm when building the frame for FAULTHAN (pushing the dynamic link- never popped off). Appears that FAULTHAN never returns thru RPU.
* Appears to be trying to exit #48 FAULTHAN at fDbgCnt 29296 but already has a BAD\_BASE (temporary?).
* $0170 appears to be the MAINMSCW. Both the initial static link and dynamic link of the MSCW are set to this (i.e. pointing at itself). INIT\_ENVIRONMENT also seems to be saving this.
* Alternatively: the 0000 is getting set in TASK\_START AT 118 when removing the userprog mscw's from mscw chain
* The first indication of BAD\_BASE occurs in SCREENOP.SC\_ERAS\_EOS (fDbgCnt = 20703).
* Initially, all of the BAD\_BASE are related to SCREENOP. Can I break when SCREENOP gets loaded? Later calls in segment CUPOPS, GOTOXY, GETCMD, STRINGOP and SDSTD also reference BAD\_BASE.
* Put breaks at proc #13 (SC\_ERAS\_EOS) on both machines. Look at data segment register (DS) and see how things differ.
  + Get a segment fault when it tries to call SC\_ERAS\_EOS
  + Can I put a break into READSEG to see what/where SCREENOP gets loaded?
  + Not Extended (is true) whatever that is
  + Returned from READSEG with IORSLT = 0
  + In ReadSeg, Delphi pushes AX three times. HP only pushes AX twice? But both are pushing WordAt[BP+SIBBASE] which is $6670. The net result is the same.
  + Both say that the number of words in the segment is $08F0 (2288). Both say that the starting block is $0107 (263). Both plan to load the segment at $1712. IORSLT appears to be 0 on both.
  + Delphi loaded SCREENOP @ $6670. HP appears to have loaded it at ES:$3990. (ES is $5F54).
  + I should put a break into READSEG (or SYSRBOOT) with fDbgCnt >= 20582. I don’t understand why the locations are different. Both were supposedly loading to $1712. It must have something to do with segment registers.

11/6/2019:

* Delphi: The offset in the pool (AX) is $6670. The number of words in the segment is $0478 ($08F0 bytes = 2288). The starting block is 263. This is block 16 in SYSTEM.PASCAL.
* DS is always 0 in CALLIO.
* In READSEG, the instruction “BP := WordAt[SS+BP+SIBPOOL]” sets BP but it never gets used? Because it is not using the pool (?)
* On the HP, the first occurrence of a non-zero BP occurs in proc #52 (POOL\_ADJUST) (fDbgCnt = 19184). The call is coming from MOVESEG. This is also true on Delphi.
* Non-zero BP also occurs in KERNEL.LOADSEG call to READSEG.
* Following the call to RLOCSEG in KERNEL.LOADSEG, the decoded instructions do not seem to exactly match mydecoded?
* After the call to RLOCSEG, before it gets back to LOADSEG. READSEG: @183. This is near fDbgCnt 28274. The instruction following the call to RLOCSEG has an offset of 183 (which does not make any sense). The instruction claims to be SLDC4 (which executes- undoubtable throwing the stack off by 1. Something is not restoring SI correctly. Appears to be executing 19 p-Codes in RLOCSEG.
* Even after “correcting” the previous problem, it still crashes
* Now crashing slightly differently from before: Delphi ends with “No procedure” exception at fDbgCnt 30086, curproc=#21:SCREENOP.SC\_PROMPT, OFFSET=23, DS=6670, si=$6A; call to SCXG1. Stack: (SP=$Fd6E: 0000 0001 FD9A 003A 0000 008f 0000 0BE0…)
* in the interpreter, the procedure (CSPTABLE[DI].ProcCall) is JUMPED to-- not called, so, to prevent an obsolete SI from being restored, I am trying to save it again
* Need to check all of the CSP procs to see how each handle saving/restoring the IPC. Quick look seems to indicate that all (EXCEPT RLOCSEG) were calling SAVEIPC
* Maybe try to stop both systems at SCREENOP.SC\_PROMPT?
* Run CXG RLOCSEG on HP. How does SI get properly set?

11/7/2019:

* Looks like the HP also returns to the wrong instruction following the call to RLOCSEG!
* BldFrm has a SAVESTAT which points to an all zero MSCW, zero EREC and zero SIB. The value of SAVESTAT is 3040 ($BE0).
* 3040 is being pushed onto the stack from SAVESTAT in USERPROG.SC\_INIT in BldFrm.
  + #13:USERPROG.INIT\_SCREENOPS is calling #24:SC\_INIT.
  + Static link coming from GLOBALS.LOWMEM.BASE. GLOBALS.LOWMEM.BASE is getting set to 3040 in NewEnv.
  + Coming from WordAt[SS+BP+ENVDATA] (which is supposed to be ^GlobalDataSegment. DS <> ES?
  + Base is set to 3040 ($BE0) which appears to be all zeros on both machines. EVECp is 1314 ($0522).
  + SIBp is $079A on both.
  + “Base of Segment” is 0 on both.
  + HP has jumped to SEGBACK indicating a segment fault.
  + Delphi also raises ESEGBACK.
  + Both enter SEGBACK code.
  + Both restore SI to $07D6.
  + Both push an ERECp of $0790.
  + Both restore BP (environment) to $DA42.
  + Both enter NEWENV WITH bp SET TO $da42.
  + Delphi SOMEHOW exits NEWENV and ends up back in INNER? Maybe the exception is unraveling?
* Here is the sequence:
  + Executing in NEWENV
  + The base of the segment is 0000
  + This forces and exit from NEWENV with IsSegFault set to TRUE
  + NEWSEGMENT is exited (IsSegFault is still true)
  + Control returns to INNER in CXGIMMED
  + Since IsSegFault is TRUE, the ESEGBACK exception is raised
  + INNER handles the exception by calling SEGBACK
  + SEGBACK restores the previous environment and then calls NEWENV again.
  + NEWENV is called for the second time. This time, the base of the segment is not zero (55920 = $DA70)
  + Control returns from NEWENV back to SEGBACK.
  + On Delphi

SEGBACK calls SEGFAULT

SEGFAULT calls FAULTCOM

In the assembly code,

it simply jumps to SEGFAULT

and then jumps to FAULTCOM.

* Both call SIG (which should cause a task switch)
* Both say that there a 0 (?) semaphores waiting
* Both have a semaphore at $02A2
* Both ENQUE $0154
* Both set up a task switch by calling SIG
* Both call the TASKSW procedure
* But somehow, magically, the HP is in RESTORE called from TASKSW but Delphi has returned to BLDFRM called from INNER called from CXGIMMED called from SCXG???? Maybe I inadvertently pressed to F9 key rather than the F8 key
* SAVESTAT gets a big change in BLDFRM (Proc #27, offset 134, fDbgCnt 18765)
* SAVESTAT gets changed to 3040 around Proc #21 fDbgCnt = 30142 but the area pointed to is still all zeroes.
* How did I get into SC\_ERAS\_EOS? It never seems to have been called.
* Likewise for QUIET

11/8/2019:

* INIT\_UNITS\_GLOBALS calls INIT\_SCREENOPS
* Looks like INIT\_SCREENOPS is trying to call SC\_INIT when it gets a segment fault. The fault takes control into FAULTHAN
* FAULTHAN loads the missing segment
* FAULTHAN tries to flip the segment
* FAULTHAN relocates the segment
* FAULTHAN locks some (?) segment
* FAULTHAN eventually hits the WAIT which transfers control back to INIT\_SCREENOPS
* INIT\_SCREENOPS appears to be retrying the call to SC\_INIT which is when things go South
* Is SAVEREG using the TIBPROC correctly? Yes. I think so. TIBPROC = 18 and TIBIPC = 14
* Maybe I’m saving the SI code and/or procnum at some point when it is not supposed to be saved?
* Following the RESTORE (near 18782) CURPROC says procedure #27 INIT\_KERNEL\_GLOBALS (on both). However, Delphi actually appears to be in #32:INIT\_TASKS. HP, on the other hand, somehow gets to WAITER and then enters TASK\_START.
* Following the RESTORE, it seems like the HP is jumping directly to the SLLA rather than going through the jump table? This needs to be verified. I don’t think that this is true.
* Bring back the debug window
* Use meaningful names
* Do not use temporary variable names (like BP, etc)

11/9/2019:

* Trace thru RESTORE on both machines again carefully looking for things that point to something that looks different.
* Find out where the SI that is going to be used was saved.

11/11/2019:

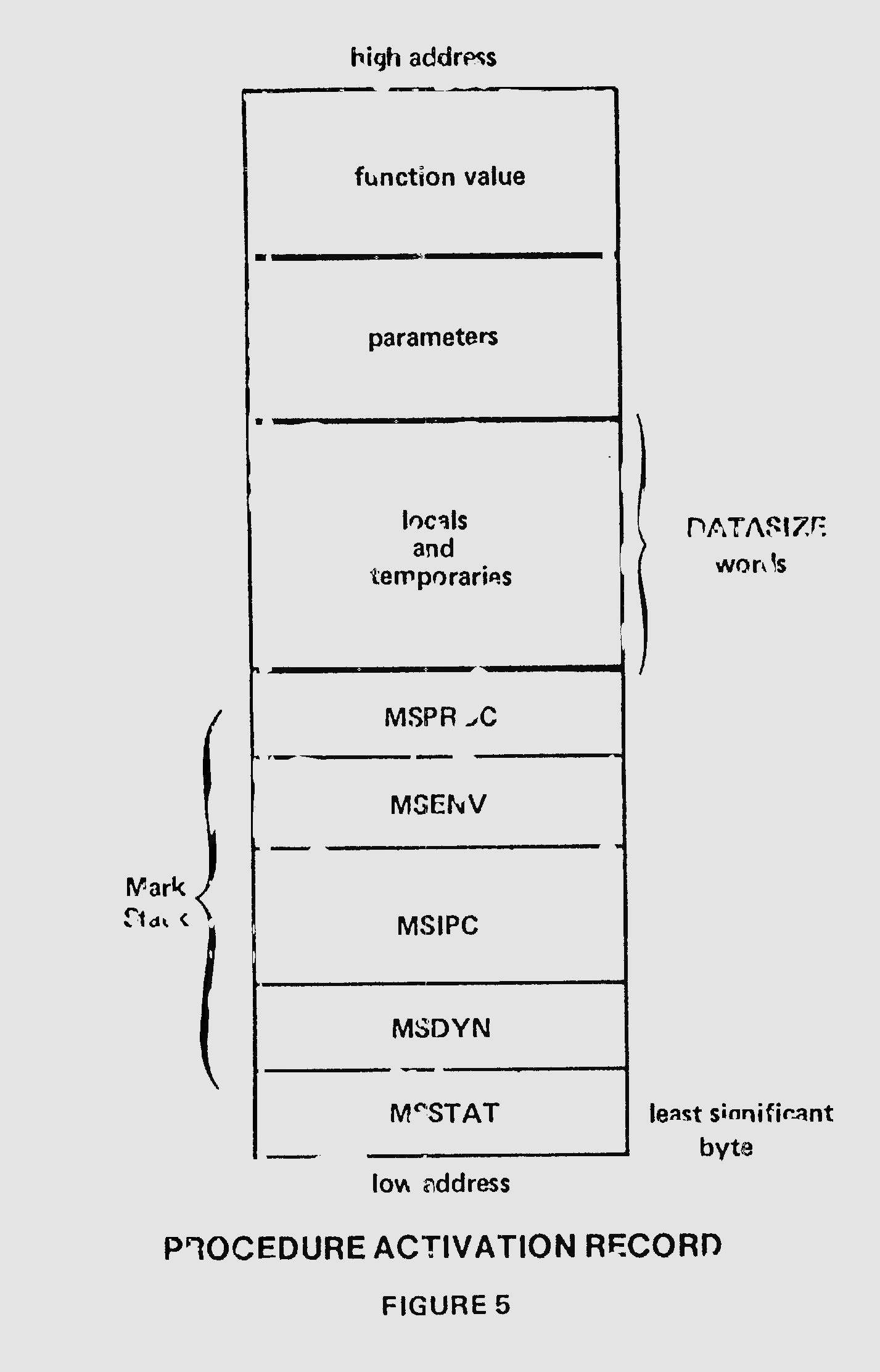
* MPPLUS in restore

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Addr | Static Link | Dynamic Link | MS IPC | MS Env | MS Proc | Local Data & parameters |
| Delphi | $D906 | 0002 | 0100 | 001E | 00F7 | 0000 | 0007 0005 0019 0000 067C 04B0 CF14 |
| HP | $D906 | 0002 | 0100 | 001E | 00F7 | 2524 | 0007 0005 0019 0000 067C 04B0 CF14 |

* BASEPLUS in restore
  + Delphi: 0000 00E6 03FE 03AE 035E 0000 0000
  + HP: 0001 00E6 03FE 03AE 035E 0000 0000
* MPPLUS in restore

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Addr | Static Link | Dynamic Link | MS IPC | MS Env | MS Proc |
| Delphi | $D94C | 0002 | 0000 | 0154 | 0000 | 0000 |
| HP | $D94C | 0002 | 0000 | 0154 | 0000 | 0000 |

* Activation Record:



* After the restore both systems appear to think that they are in procedure #27 (INIT\_KERNEL\_GLOBALS). Both execute a SLLA1
* Both appear to have a “virgin” stack located at $297E (which probably came from the TIB @ 0286).
* In NEWENV, is the failure to PUSH DS, POP DS surrounding the call to POOLBASE significant?
* Unknown. Still crashing at fDbgCnt = 30104.
* Why is DS 0 when entering NEWENV? (Only on Delphi?)
* In WAIT, DS is getting set to SSDSVAL, which has a 0 in it? fDbgCnt = 18781
* First entry to NEWENV with DS = 0 occurs at fDbgCnt = 18784, CurProc = 27, offset is weird. Occurring at the SCXG1. DS got set to 0 at the WAIT command when setting to SSDSVAL.
* Would it be possible to leave DS alone in WAIT and just refer to SS? It should not matter because DS is getting PUSHed and POPed in WAIT ) (unless ENQUE is doing something).
* Look at SSDSVAL usage on HP. Why must it be stored in the code segment?
* Why is DS 0 when entering NEWENV? (Only on Delphi?) Compare to HP!
* DS gets set to 0 in INNER in CXGIMMED called from SCXG via DS := SS
* The code in INNER seems to be referring to different data segments. But, even though in apparently different locations, the contents seem to be the same:
  + ' @Addr $DA70 [55920]: ;SEGMENT @:DA70;DictOffs:1B7E;RelocOffs:0000;SegName:USERPROG;Sex:0001;Const1:1854;CONST1:F2C4;NrReal:0000;RealSize:0004;RealSubpOffs:0002;NrProc:0028;ConstPoolSize:730'
  + ' @Addr $DA70 [55920]: 0DBF 0000 5355 5245 5250 474F 0001 0C2A 0004 2001 5700 014A 000E 7A2E 6879 7A2E 6978 0C84 2021 06E7 7002 2B18 B101 19D5 2E01'
* The DS on Delphi is 0000 (taken from the SS)
* The DS on HP is $5F54
* Track backwards on Delphi each time that DS gets set to 0

11/12/2019:

* DS:= SS at fDbgCnt = 18784 sets DS to 0
* ------------------ FINAL EXCEPTION
* Range check error: DbgCnt=20110, CurProc = 13, Offset = -1, DS=$DA70, SI=$7D6
* Dies on this line with a range check error:
  + BP := WordAt[DS+DI+SIBPOOL];
  + DS = $DA70
  + DI = $DA4C
* Now I am getting infinite series of segment faults when calling SC\_INIT
* Compare what happens on HP for each call to NEWENV
* Never allow DS to get set to 0? Continue with this! Back out 11/12/2019 if only if absolutely necessary!

11/13/2019:

* I appear to have had MP stored in two different locations: Globals.LowMem.MP and as a Global variable
* Currently hung in a loop in FAULTHAN: SLDC1; SLLA1;SLDC1; UJP, etc.
* Latest code changed was in SIG, ENQUE, DEQUE
* Since MP can be directly addressed by p-Code, my register version of MP should refer to the version in low memory.
* Check LPR/SPR for direct access TIB registers. The offsets do not seem to agree with TTib.
  + LPR Used: 2, 4, 5, 8, -1
  + Change BX to ALWAYS be the local activation record
  + Is there someplace where I am calling SAVEIPC that is not being called in the assembler?

11/14/2019:

* // Name: ParmDscr
* // uses: BP, AX, DI; may seg\_fault! please IpcSave before calling
* Currently waiting for input in USERPROG.BOOT\_ERROR (CurProc=4, ofs=22, fDbgCnt=893)
* Called from INIT\_SYSCOM @ ofs = 38
* Following call to #6 UserProg.FILE\_PRESENT
* Probably related to a failed CSP
* Or to a failed call to EQSTR
* Most likely CSP is setting up the destination incorrectly
* In turn, probably caused by ParmDscr setting up destination incorrectly
* Dst is getting returned as 0 (which may be incorrect)
* Currently infinite loop in Proc #33: UserProg.Init\_Kernel\_Globals around offset #105 following a WAIT command (fDbgCnt=18814). Repeated calls to SEGBACK

11/15/2019:

* Possible problems:
  + ParmDscr (see above)
  + MOV
  + Is there someplace where I am calling SAVEIPC that is not being called in the assembler?
  + BACKUP\_INTERP7 is not working- should be using the version on NDAS\ndas-i\temp. Only between 9am and 3pm?
  + Need a function to compare with most recent versions (did I have this on Gamer?)
* Most recent change was to XEnable-- seemed to be working

11/16/2019:

* The various fault handlers may be depending on SAVEIPC having been previously called!
* Procedures that need to have dependance on DS removed include:
  + BldFrm
  + SEGBACK
  + STKBACK
  + NEWENV
  + STKCHK
  + SEGFAULT
  + FAULTCOM
  + CPL
  + GPG
  + CXL
  + CPIIMMED
  + CXI
  + XCFP
  + ParmDscr
  + UNI
  + RESTORE
  + BLDFRM
  + SIG
  + WAIT
  + DEQUE
  + ENQUE
  + EVENT
  + USR\_BREAK
  + MOVERITE
  + MOVELEFT
  + USR\_BREAK
  + NEWSEGMENT
* Look for DS ← 0
* Look for “DS := SS”
* Remember to re-enable BACKUPS

11/18/2019:

* What are the parameters to NEWENV?
* CXGIMMED contains a call to SaveIPC which does not exist in the assembler code
* HP never calls FLIPSEG
* In procedure USERPROG.FLIPSEG at ofs 20, Delphi calls the low-level proc FLIPSEG, but HP branches to Ofs 304
* Appears to be returning a byte sex of 0 (which cannot be right)
* The SIB that is referred to appears to be incorrect. The address of the SIB appears to be coming from the EREC.
* The EREC that is passed to FLIPSEG does not appear to be valid.
* FLIPSEG is called by #9: USERPROG.LOAD
* #9: USERPROG.LOAD has a parameter the\_erec:erecp which is passed to IT
* #9: USERPROG.LOAD is called by #25: INIT\_INTERNAL\_POOL @ an offset of 108. It is passing GOTOXY\_EREC as a parameter. The value of GOTOXY\_EREC is $0762. $0762 Looks like it might be a valid EREC (it points to a valid looking EVEC and to a valid SIB for “GoToXY”. I.e.,
* INIT\_INTERNAL\_POOL() // no parameters
  + USERPROG.LOAD( the\_erec:erecp; new\_residency:integer )
  + USERPROG.LOAD( GOTOXY\_EREC {$0762}, new\_residency:integer )
    - USERPROG.FLIPSEG(segerec:erecp {$D958}) // i.e., this is NOT GOTOXY\_EREC
* Question: How come GOTOXY\_EREC did not make it to FLIPSEG?
* Most likely problem source is BLDFRM or NEWENV.
* Re-enable backups

11/19/2019:

* When entering the procedure FLIPSEG, the 1st instruction (SLDL 15) loads $D958 on Delphi, but loads $0762 on HP (which is GOTOXY\_EREC).
* The stack looks fine just prior to to SCIP1 which calls #9 USERPROG.FLIPSEG
* fDbgCnt = 18784 is the FIRST time that NEWENV has been called? Also the first time that NEWSEGMENT has been called?
* In NEWENV, following the call to POOLBASE, popping into DS, gives it a different value - try a break at 0FEE
* When entering NEWSEGMENT, DS and ES are different in Delphi, but not in HP
* NEWSEGMENT is called by INNER
* INNER used to set DS to SS (but not any more)
* Restored the DS := SS but then call to USTATUS returns to the wrong place
* Removing the call to SAVIPC in CXGIMMED then leads to infinite loop in WAITER
* Restoring the call didn’t fix the infinite loop
* I put back in all of the places that saved and restored SAVDS-- still in a WAITER loop
* Renamed current working version interp.pas → interp7\_21.pas
* Replaced it with the backup interp7\_1.pas (1:43PM). This also gets hung in WAITER .
* Replaced it with the 1:27 version. This also gets an “Operator 83 is not assigned error” which occurs in WAITER (ofs 14)
* Replaced it with an earlier version which still loops in WAITER.
* The code in WAITER says:
  + SCXG1 USTATUS // {causes polling to occur}
* Precisely what polling is supposed to occur? Does that mean that if any interrupts have occurred (key press, for example) they would get handled?
* Reverted to the 12:30 pm version. 12:30 version crashes with “Operator 83 is not assigned error” in WAITER which is occurring following the call to USTATUS.
* The call to SavIPC is putting the IPC back to where it is going to cause FETCH to get the wrong instruction.
* I need to look at WAITER on the HP and see what it does following the call to USTATUS.

11/20/2019:

1. In WAITER, Delphi advances from ofs 3 to ofs 3. HP, however, goes into procedure #33:TASK\_START. Instruction appears to be SLDL9
2. Delphi sees instruction at ofs 2 as SIGNAL. So does HP.
3. They have very different addresses for the semaphore pushed onto the stack. Stuff below the TOS looks the same. Delphi’s does not look like a real semaphore.
4. 1ST instruction in WAITER is LDL 31. Where did the 31 come from? This does not seem to correspond to source code?
5. The code generated for a process seems to assume that the semaphore has been pushed onto the stack and that it will be just above (?) the local parameters.
6. SIGNAL is entered with very different values of DS, but the segment info looks identical. HP appears to have the same segment info in two different locations (unless they are really the same location?)
7. // NOTE: NEWENV, BLDFRM, CHGSIB, SETSTAMP assume DS = SS, ES = SEGB !!!!
8. Also NEWSEGMENT?
9. CHGSIB: Does the above assumption work backwards, also? Think about the ramifications in CHGSIB.
10. [Re: 1 above] Following most recent changes, Delphi also seems to be going to proc #33 USERPROG.TASK\_START
11. I need to break on Globals.LowMem.Segb getting set to 0
12. Breaking in BLDFRM because the SEGBASE, coming from MSCW→ erec→ sib→ seg\_base IS ZERO. INIT\_SCREENOPS is calling SC\_INIT. fDbgCnt = 20700.
13. The MSCW is all 0. The EREC is all 0. The SIB is all 0. The SEG BASE is, of course, returned as 0. This is INIT\_SCREENOPS attempting to call SC\_INIT. Why is the MSCW all 0?
14. I saved a (mostly) working version in F:\NDAS-I\Temp\Working Version
15. Getting as far as:

' @Addr $FED6 [65238]: \_Command: E(dit, R(un, F(ile, C(omp, L(ink, X(ecut

Calls procedure SCAN

Unimplemented: DbgCnt=30321, CurProc = 30, Offset = 48, DS=$6670, SI=$2F1

Appears to be a UWrite

11/21/2019:

* Today I made it as far as the p-system prompt !
* UnTested: UNI AH < AL:
  + DbgCnt=38314,
  + CurProc = 9:
  + FILEOPS.PROC #9,
  + Offset = 22,
  + LastOpCode = UNI,
  + DS=$CC80,
  + SI=$2C1
* FINAL STACK:
  + @Addr $F940 [63808]:
  + 9: FILEOPS.PROC #9 <-
  + 1: SEGFOPEN.PROC #1 <-
  + 2: FILEOPS.PROC #2 <-
  + 11: ASSOCIAT. <-
  + 2: ASSOCIAT. <-
  + 8: GETCMD.PROC #08 <-
  + 4: GETCMD.PROC #04 <-
  + 2: GETCMD.EXEC\_ERROR <-
  + 51: KERNEL.COMMAND <-
  + 1: KERNEL.Initialization <-
  + 2: rŠ‡„.PROC #2
* Need to fix MOVSW (could pass a loop count)
* I still have places that are setting DS (DUPR, for example)
* Latest sources backed up to F:\NDAS-I\Temp\Working Version
* Remember to re-enable backup

11/22/2019:

* Need source to SEGFOPEN
* The call to UNI in FILEOPS.PROC #9 at offset #23 is wacky-- impossible set!
* Recent instructions include:
  + DUP1 {226} used 41 times
  + SRS {188} used 16 times
  + UNI {219} used 14 times
  + The call to UNI is generating an impossible set. fDbgCnt = 38314
  + UNI is using the alternate branch { $3 } in UNI.
  + Completely re-do the UNI (2nd branch) code-- postpone optimization!

11/23/2019:

* Trying to FLIPSEG at fDbgCnt 40294. Proc #53: KERNEL.FLIPSEG, OFS: 19- appears to be the wrong byte sex.
* Being called from CHECKSEG. Segment just loaded was LOCK
* KERNEL.FLIPSEG looks identical to USERPROG.FLIPSEG. Is this a case where the segment number is “local”?
* On the first call to FLIPSEG, di=7. Flipping constant pool ptr-- 2 words with an offset of 7 in the segment
* Unimplemented NATNFO
* Untested XJP for wrong byte sex
* Current upper limit fDbgCnt = 43766, in CXI
* Currently in CXI. SAVEPAR is getting 257 which is to big to fit into CL, a byte sized register
* Operator 70 is not assigned @ CurProc 116, Offset 0; fDbgCnt = 44742. Occurs shortly after the call to CXI. Segment may be named FILEOPS. Proc #116. There is a segment with that name but it only has about 29 procedures.

11/25/2019:

* CXI - ending up with different value for ES following call to CHGSIB

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Where | ES | DS | SI | MP | AX | SavePar | Segment |
| Delphi | @NEWSEGMENT | $BA90 | $BA90 | $05C0 | $FB00 |  | $0101 | DS: ASSOCIATE |
| HP | @NEWSEGMENT | $6AFD | $5F54 | $05C0 | $FB00 |  | $0101 | ES: ASSOCIATE (same as above) |
| Delphi | After ESEGBACK | $BA90 | $5BA0 | $05C0 | $FB00 |  | $0101 | DS:ASSOCIATE |
|  |  | $650E | $5F54 | $05C0 | $FB00 |  | $0101 | ES: ASSOCBAS |
| Delphi | @CHGSIB | 3F30 | 6347 | 08B6 | E9CC | F896 |  |  |
| hp |  | 6347 | 5F54 | 08B6 | E9CC | F896 |  |  |
| Delphi | Proc 1, Ofs 0- LLA |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

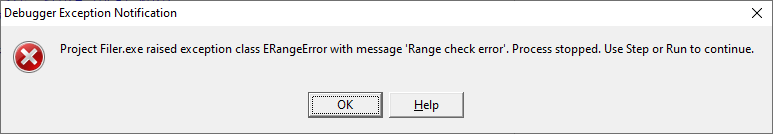
* Delphi Returns to CurProc=1, Offset=0
* HP does the same
* Operator 137 not assigned. fDbgCnt = 50800. This is function LOD.

11/27/2019:

* GetWordAt passed an ODD address: 11847. fDbgCnt = 54606. ASSOCIAT.PROC #10
* Crashes after exiting OS\_INIT, in RPU-- probably because of a negative procedure number
* The CspTable is not being initialized for fDebugOpsTable because the proc parameter is NIL
* GetWordAt odd address crashes at loopcount i = $c
* Need an expression for map[i] → segment info. This may have offsets from something?
* New MemDump() functions are not displaying the correct addresses
  + ' @Addr $2D12 [11538]: TEvec: Vect\_Length:30, Map[]=[ 1]=2CE4, [ 2]=08D2, [ 3]=08A4, [ 4]=0790, [ 5]=095C, [ 6]=092E, [ 7]=0848, [ 8]=0A14, [ 9]=098A, [10]=4D50, [11]=2E45, [12]=07BE, [13]=081A, [14]=0706, [15]=27B4, [16]=0028, [17]=0043, [18]=06AA, [19]=530D, [20]=5359, [21]=4554, [22]=2E4D, [23]=4F43, [24]=464E, [25]=4749, [26]=0FF4, [27]=0200, [28]=06D8, [29]=0043, [30]=0000, '
  + ' Env\_Data: $0170 [ 368]; Env\_Vect: $04B0 [ 1200]; Env\_Sib: $0686 [ 1670]'
  + ' TSib: Seg\_Pool:0000,Seg\_Base:1730,seg\_refs:3,timestamp:264,seg\_pieces:0,residency:1,seg\_name:KERNEL ,Seg\_Leng:0888,Seg\_Addr:00FB,Vol\_Info:04A6,Data\_Size:406,Next\_SIB:0000,Prev\_SIB:0000'
  + ';SEGMENT @:1730;DictOffs:110C;RelocOffs:0000;SegName:KERNEL ;Sex:0001;Const1:0F3E;CONST1:266E;NrReal:0000;RealSize:0004;RealSubpOffs:0002;NrProc:0042;ConstPoolSize:330'

11/29/2019:

* @cur\_erecp^.env\_vect^ = $FD58
* Loop Limit (Vect\_Length) @FD5A = 30
* Current Value of i @$FD54
* Vector #2 is $2CE4 which should be the address of the EREC but I am getting the address $2252? Wasn’t copying the entire vector?
* The Seg\_Base of the SIB points to segment info being located at $0000 (which is wrong, of course)
* Looks weird at map[11] and crashes around map[12]
* Does a 0000 Seg\_Base have any significance? I.e., perhaps segment no loaded?
* Reminders:
  + Procedures calling which were not called
  + BAD\_BASE
  + Memory not returned
  + Recent volumes
* The interpreter is being destroyed. It is freeing the drivers.
  + The driver is freed
  + Which is destroying the volume
  + Which belongs to the Filer
* The following information needs to be copied to the duplicate volume
  + fVolStartBlockInParent
  + DOSFileName
  + pSystemName- is never used in TVolume.Create? Removed.
* Working:
  + H(alt (maybe)
  + Prompts for R(un
  + U(ser restart says “User restart not possible” -- this is the same on HP
* Not working
  + I(nitialize -closes everything
  + X(ecute
  + R(un
  + C(ompile
  + A(ssemble
  + S(et - generates a range check error
  + Trying to execute Hello.code gives a range check error in KERNEL.S\_EXIT. Last opcode is CAP.
  + I’m getting various IO errors when trying to boot from other than #4
  + When trying to boot from PSYSTEM.VOL:



Occurs at fDbgCnt 44165, Proc #11, Offset 5, ASSOCIAT.program\_associate. Called by ASSOCIAT.S\_ASSOCIAT

11/30/2019:

* How many bytes does STL require?
* ASSOCIAT.BUILD\_OS\_UNIT\_LIST called VARNEW (I Think), but name is show up in debugger as PROC #3?
* Filer Search function should have an option to ignore ‘\_’
* Why does S\_ASSOCIATE appear to have a MODI op at the end? It dosen’t make any sense!
* {$IfDef PHITS} and {$IfDef TEMPORARY} do not seem to be generating any code
* Crashing on the “LLA 559” in PROGRAM\_ASSOCIATE
* I need do decode PROGRAM\_ASSOCIATE on the HP and compare the code
* The address passed to GETADDR in LLA is $FBF0 which is the same as MPPLUS
* Need to find the crash that is occurring in ASSOCIAT.program\_associate on the “LLA 559” instruction
* The crash does not occur when running pSystemx.vol (presumably because AX+Offset will fit into a word)
* Latest crash is occurring in ParmDscr(), CurProc = 17 [KERNEL.S\_EXIT], Offset 309, fDbgCnt = 46026 when trying to execute HELLO.CODE
* Still need to figure out procs that have not been called, calling other procs and BAD\_BASE
* I could also look at duplicate characters on the terminal

12/2/2019:

* Error says: “The procedure 13: SCREENOP.SC\_ERAS\_EOS (which has not been called) is calling 24: bad\_base.PROC #24”
* But the call is actually coming from 13: USERPROG.INIT\_SCREENOPS
* Which (Delphi) is coming from BldFrm, which is using Globals.LowMem.Segb
* Current crash occurring in ParmDscr which is called by CAP. fDbgCnt = 46026
* The instruction “10. yes, extra increment of ipc” never gets called? Can I duplicate this on the HP?
* HP never gets a segment fault in PARMDSCR
* The segment fault in PARMDSCR seems to simply exit everything
* When the ESEGFAULT is declared in PARMDSCR,
  + Src = 0
  + SrcO = 0
  + Dst = $9830
  + DstO = $FCEE
  + But this gives a range check error in CAP because Dst+DstO = $1951E
  + Dst (& DstO?) never getting set?
  + I could use decoded #17 KERNEL.S\_EXIT ofs 310
  + Backspace not working correctly to erase previous character
  + Does CAP really need to skip an extra byte here?
  + Current crash is in NEWENV, KERNEL.INITIALIZATION, fDbgCnt = 51116. This is a “range check error” in NEWENV. The call stack looks wonky: “' 1: KERNEL.Initialization <- 2: ° ¡Ô4‡$.PROC #2'”. This follows a bunch of RPU’s. Trying to exit KERNEL.Initialization into garbage. Strange call to Proc #15 @ fDbgCnt = 51113.
  + KERNEL.S\_EXIT <-- GETCMD.PROC #8 <-- GETCMD.PROC #15 ← GETCMD.EXEC\_ERROR <- KERNEL.COMMAND ← KERNEL.Initialization

12/3/2019:

* Is CAP still a suspect?
* Current crash is in NEWENV, KERNEL.INITIALIZATION, fDbgCnt = 51116
* By fDbgCnt = 51000, the procedure numbers are all mucked up
* fDbgCnt for proc number corruption is inconsistent-- possibly around 50163-- getting a procedure in the call stack BELOW 1: Kernel.Initialization. By 51000 there are NEGATIVE procedure numbers.
* “BAD\_BASE” first shows up at fDbgCnt 48342. The SEGNAME is garbage. This occurs is proc #17, ofs 414 on or near a STP instruction.
  + 48071- call stack already mucked by here
  + 40038 - call stack mucked up exiting
  + 40838 - call stack mucked up (maybe)
  + 28100 - KERNEL.FAULTHAN
  + ~40800 - bad\_base.PROC #27 show up in FAULTHAN as the root procedure. Perhaps FAULTHAN is doing something fishy with DS or its own private stack? Once it shows up, it doesn’t go away. Its below KERNEL.Initialization
  + 40800-40810 - “bad\_base” starts showing up- in FAULTHAN -- using its own private call stack?
  + 40806 - “bad base” show up- #3 KERNEL.LOADSEG -- following the READSEG opcode - SYSRBOOT is messing up something-
  + CALLIO was (HP) pushing DX & SI. I don’t know where they were getting popped
  + SYSRBOOT is still messing something up
  + Stack before SYSRBOOT:
    - 'Stack @Addr $2C32 [11314]: 08A4 0000 2C4E 0170 2C58 08B0 067C 0032 0001 D170 0001 0170 2C58 0825 067C 0032 08AE 0170 08A4 0170 2C86 0A59 067C 0030 D170'
    - Messed up at the call to Driver.Dispatcher- reading 8 blocks starting at block 314 - remainder is 384- reading from SYSTEM.PASCAL. Trying to load FILEOPS. Is the READ overflowing its buffer, or is there a bad buffer address? Should load 4480 bytes. FILEOPS is segment #14 in SYSTEM.PASCAL.

12/4/2019:

* $DA32 @ Kernel.Initialization
* $FFF4 @ Userprog.OsInit
* InTHE FAULT HANDLER this is the stack: '64: KERNEL.CHECK\_SEG <- @2C38: 3: KERNEL.LOADSEG <- @2C58:50: KERNEL.POOLSEG <- @2C86:48: KERNEL.FAULTHAN <- @ 0:27: USERPROG.INIT\_KERNEL\_GLOBALS'
* Kernel.Initialization later shows as : @$FFEA
* By fDbgCnt 41140, I have bad base
* The MSCW USERPROG.INIT\_KERNEL\_GLOBALS appears to be located at $0027 which is an ODD address?
* Bad base by 40653
* The CALL TO sysio is NOT messing up the mscw

Just before SYSIO:

'MSCW @ $2C86 [11398]: StatLink:0170, DynLink:0000, MSIPC:12E6, MSENV:DA42, Proc#:27, LocalData[0]:6670'

'EREC @ $DA42 [55874]: $0170 [ 368]; Env\_Vect: $04B0 [ 1200]; Env\_Sib: $DA4C [55884]'

'SIB @ $DA4C [55884]: Seg\_Pool:0000,Seg\_Base:DA70,seg\_refs:1,timestamp:53,seg\_pieces:0,residency:0, seg\_name:'#0#0#0#0#0#0#0#0',Seg\_Leng:0DC0,Seg\_Addr:0143,Vol\_Info:0000,Data\_Size:0,Next\_SIB:0000,Prev\_SIB:0000'

'Segment @ $DA70 [55920]: ;DictOffs:1B7E;RelocOffs:0000;SegName:USERPROG;Sex:0001;Const1:1854;CONST1:F2C4;NrReal:0000;RealSize:0004;RealSubpOffs:0002;NrProc:0028;ConstPoolSize:730'

After: The EREC @ $DA42 has been smashed

The SIB @ $DA4C has been smashed

The SegBase@ $DA70 has been smashed

The buffer address for loading is $CF10

The byte count is $1180

This means that the data loaded will extend from $CF10 to $E090 which WILL overwrite items listed above

* The call is coming from #3:KERNEL.LOADSEG ofs 13
* Is SEG\_LENG the number of words, or the number of bytes? Appears to be number of words as documented.
* I appear to have references to two different ERECs:

1st: Just before SYSIO

2nd:

1st: 'EREC @ $DA42 [55874]: $0170 [ 368]; Env\_Vect: $04B0 [ 1200]; Env\_Sib: $DA4C [55884]'

2nd: 'EREC @ $08A4 [ 2212]: $0C6C [ 3180]; Env\_Vect: $058C [ 1420]; Env\_Sib: $08AE [ 2222]'

They refer to the following SIBs:

1st: 'SIB @ $DA4C [55884]: Seg\_Pool:0000,Seg\_Base:DA70,seg\_refs:1,timestamp:53,seg\_pieces:0,residency:0, seg\_name:'#0#0#0#0#0#0#0#0',Seg\_Leng:0DC0,Seg\_Addr:0143,Vol\_Info:0000,Data\_Size:0,Next\_SIB:0000,Prev\_SIB:0000'

2nd: 'SIB @ $08AE [ 2222]: Seg\_Pool:1712,Seg\_Base:D170,seg\_refs:0,timestamp:160,seg\_pieces:0,residency:0, seg\_name:FILEOPS ,Seg\_Leng:08C0,Seg\_Addr:013A,Vol\_Info:04A6,Data\_Size:1,Next\_SIB:06B4,Prev\_SIB:076C'

Segment Info:

1st: 'Segment @ $DA70 [55920]: ;DictOffs:1B7E;RelocOffs:0000;SegName:USERPROG;Sex:0001;Const1:1854;CONST1:F2C4;NrReal:0000;RealSize:0004;RealSubpOffs:0002;NrProc:0028;ConstPoolSize:730'

2nd: 'Segment @ $D170 [53616]: Invalid Segment Info: Range check error'

BREAK @ 36314 (I think). The segment info looks OK after the call to SYSIO

12/5/2019:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| fDbgCnt | Segment | Start Blk Nr | Low Address | High Address | Byte Sex | Size | Note |
| 18537 | GOTOXY | 261 | 11472 | 11599 | 1 | 127 |  |
| 18603 | KERNEL | 251 | 5936 | 10303 | 1 | 4367 |  |
| 20531 | SCREENOP | 263 | 26224 | 28511 | 1 | 2287 |  |
| 28115 | CUPOPS | 262 | 28512 | 28735 | 1 | 223 |  |
| 28870 | GETCMD | 268 | 28736 | 33663 | 1 | 4927 |  |
| 29333 | STRINGOP | 293 | 33664 | 34143 |  | 479 |  |
| 29700 | SDSTD | 382 | 34144 | 46127 |  | 11983 |  |
| 30486 | PASCALIO | 289 | 46128 | 47759 |  | 1631 |  |
| 34555 | OSUTIL | 341 | 47760 | 49023 |  | 1263 |  |
| 35448 | ASSOCIAT | 300 | 49024 | 51295 |  | 2271 |  |
| 35994 | EXTRAHEA | 296 | 51296 | 53007 |  | 1711 |  |
| 36361 | HEAPOPS | 294 | 53008 | 53615 |  | 607 |  |
| 40653 | FILEOPS | 314 | 53616 | 58095 | 1 | 4479 | Note 1 |
| 44312 | SEGFOPEN |  |  |  |  | 0 |  |
| 45658 | LOCK | 344 | 59120 | 60575 |  | 1455 |  |
| 47900 | EXTRAIO | 248 | 60576 | 61695 |  | 1119 |  |
| 48437 | ASSOCBAS | 308 | 23456 | 26095 |  | 2639 |  |
| 50358 | SEGFCLOS | 339 | 54288 | 55279 |  | 991 |  |

* Note 1: The segment appears to load successfully. However, the call stack is wacked with a “bad\_base”. The referenced MSCW is at $2c86 = 11,398. The environment (EREC) pointed to in the CallStack is 55874, which is in the middle of the area that is loaded to in the SYSRBOOT call which loads FILEOPS. The referenced area appears to be for USERPROG which starts at $DA70 (55920) and extends to $E830 (59440).
* The SIB does not appear to (ever?) actually contain the segment name when I need it. When does it get set?
* Change ProcName() TO the SegName in the SIB via the EREC, rather than the SegName in the SegBase. Maybe someday.
* Currently debugging in NEWENV. Register usage is being changed. {$IfDef InProgress}

12/6/2019:

* Console driver not handling backspace correctly
* Turn POOLBASE into a function
* Still getting anonymous procedure names (e.g., “Proc #14”)
* Currently crashing in NEWENV on this instruction: [AX := SegBase + BP] where SEGBASE = $eca0, bp = $1712 SegBase + BP = $103b2. fDbgCnt = 44784 (or 44828)
* Compare step by step with HP? ^F7 to get to break point on HP

12/7/2019:

* Final crash at 51008, in NewEnv
* At 51009, ES is pointing to invalid Segment Base
* SibPtr is also bad (=0).
* The ErecAddr is also bad. This came from BP which was passed as a parameter.
* Everything seems to be OK until gDbgCnt = 51008
* Is occurring between 51003 and 51008. This is bunch of RPU calls (unraveling from an error?).

51003: KERNEL.S\_EXIT 415 RPU

51004: GETCMD.ASSOCIATE 8 RPU

51005: Seg #4, Proc #15 281 RPU ⇐ THIS SEEMS ODD

51006: GETCMD.EXEC\_ERROR 923 RPU

51007: KERNEL.COMMAND 124 RPU

51008: KERNEL.Initialization 31 RPU

* At least once, the crash moved back to 50964 but the final OpCode executed looks the same (bunch of RPUs).
* Most likely this is the same problem that I noticed a few days ago in which SYSRBOOT is overwriting stuff.

12/9/2019:

* The call to SYSIO which craps everything up comes from #3:KERNEL.LOADSEG, O#13.
* Pushed in READSEG
  + PUSH(UnitNr);
  + PUSH(SegBase); // 10. offset in pool (58096) [$E2F0]
  + PUSH(0); // 12. Byte offset (0)
  + PUSH(NrBytes); // 15. number of bytes in segment (1024)
  + PUSH(StartBlockNr); // 16. starting block # of segment (337)
* Pushed in SYSRBOOT
  + PUSH(AX); // This is the control word
* SYSIO is loading the Seg\_Base, which comes from the EREC->SIB.SEG\_BASE which is passed into READSEG
* KERNEL.LOADSEG pushes the segerec:erecp which was passed to it as a parameter when it was called by KERNEL.FAULTHAN
* Something altered where the breakpoints are occurring? I thought it was MAXHIST but probably not.
* Procedure KERNEL.POOLSEG called LOADSEG (This does not agree with the call stack which shows FAULTHAN as the caller.) But #50:KERNEL.POOLSEG is calling READSEG @ ofs: 324. It is passing env\_rec as the parameter.
* The env\_rec is, in turn, passed as a parameter to POOLSEG,
* (at p #50, Ofs:143) #50:KERNEL.POOLSEG has an address for the EREC of $790 and $79A for the SIB
* (at p #50, Ofs:324) #50:KERNEL.POOLSEG has an address for the EREC of $790 and $79A for the SIB (I.e., no change)
* The MainTask (TIB) is located $DA42
* Its SIB is located at $DA4C
* It SEGBASE is located at $DA70. This is the segment USERPROG.
* PoolBase is a TWO word quantity. Which part(s) are being used? Word [1] = $2CD0. Where does this get set?
* The 2nd word of the pool descriptor gets set in INIT\_INTERNAL\_POOL @ ofs: 70. Pool info is located @ $1712.
* Start stepping through INIT\_INTERNAL\_POOL and find out why 2nd word of the pool descriptor gets set. Actually a TPoolDescriptor is not the same as a TPoolInfo. Apparently it is minoffset & maxoffset that are getting set,

12/10/2019:

* Maybe I should try to find out what is happening just prior to BAD\_BASE appearing

12/11/2019:

* I may be able to use the START\_MSCW field in the TIB when displaying the call stack to know when to quit
* Exiting:
  + SEGFCLOS.EXIT\_F\_CLOSE
  + SEGFCLOS.seg\_f\_close
  + FILEOPS.F\_CLOSE {this appears to be part of the AFS=Advanced File System}
  + A FIB has things packed (somewhat). Need to adjust the definition for Delphi

12/12/2019:

Priorities:

1. Find out how I got to SEGFCLOS.EXIT\_F\_CLOSE
   1. The call to EXEC\_ERROR is coming from 1.KERNEL.INITIALIZATION, O#51. There may be a task switch occurring from FAULTHAN following a call to MEMLOCK\_SEG (in FAULTHAN).
      1. #64:KERNEL.CHECK\_SEG returns to #3:KERNEL.LOADSEG
      2. #3:KERNEL.LOADSEG calls #53:KERNEL.FLIPSEG
      3. #53:KERNEL.FLIPSEG returns to #3:KERNEL.LOADSEG
      4. #3:KERNEL.LOADSEG returns to #50:KERNEL.POOLSEG
      5. #50:KERNEL.POOLSEG returns to the #48:KERNEL.FAULTHAN
      6. #38:KERNEL.FAULTHAN calls #65:KERNEL.MEMLOCK\_SEG
      7. #65:KERNEL.MEMLOCK\_SEG returns to #48:KERNEL.FAULTHAN
      8. In #48:KERNEL.FAULTHAN, the task switches #51:KERNEL.COMMAND
      9. #51:KERNEL.COMMAND ( @ O#13) calls #2:GETCMD.S\_GETCMD
   2. In #51:KERNEL.COMMAND, the attempt to call S\_GETCMD causes a segment fault.
2. Exit the program cleanly
   1. GETCMD.LEAVE\_GETCMD calls s\_exit TO EXIT all procedures in progress (not just GETCMD)
      1. This exits back to GETCMD.PROC #15
      2. This exits back to GETCMD.S\_GETCMD
      3. This exits back to KERNEL.COMMAND
      4. This exits back to KERNEL.INITIALIZATION at the final RPU
      5. Which then crashes
   2. Watch MAINMSCW for corruption.
   3. Check Readseg to see if it is still overwriting stuff
   4. See which proc numbers are being negated

12/13/2019:

* GETCMD.LEAVE\_GETCMD is getting called by #15:GETCMD.EXECUTE because #2:ASSOCIATE.S\_ASSOCIATE has returned true. #2:ASSOCIATE.S\_ASSOCIATE was called by #15:GETCMD.EXECUTE.
* #2:associate.s\_associate calls #15:fileops.fetchdir
* The call to LEAVE\_GETCMD passes a state parameter whose value appears to be 5 (COMPONLY) (which seems odd).
* KERNEL.S\_EXIT is entered with the following parameters: 0, 2, 2
* What does S\_ASSOCIATE actually do? Is it supposed to execute the program? Decode program\_associate and s\_associate.

12/14/2019:

* Today I want to find out if s\_exit is working properly
* TfrmDecodeWindow.OpsCXG is not properly handling internal calls (where segment number is 1)
* The call to LPR is in #15:GETCMD.EXECUTE, O#253 is suspicious. Supposedly loading a procedure number? Is OK. Bad documentation. Assuming that BP points to the current TIB? Yes. SAVEREG does that.
* S\_associate succeeds
* S\_Exit (parameters passed in)
  + Ex\_proc\_num = 2
  + Ex\_seg\_num = 2
* At this point the call stack starts with
  + @FFEA=2: bad\_base.PROC #2'

But it should start with:

* + @FFEA=2: USERPROG.OS\_Init'
  + I.e., high memory has been clobbered.
* In s\_exit, ex\_proc\_num has somehow changed from 2 to 1? No. 1st call to S\_EXIT is from SEGFCLOS.EXIT\_F\_CLOSE.
* The local variable (ex\_proc\_num) is located at 64764 in the S\_EXIT which is leaving GETCMD. EX\_PROC\_NUM = 2.
* I’m still ignoring the elephant in the room. Why is high memory getting clobbered when FILEOPS gets loaded? High memory has already been clobbered by the time S\_EXIT gets called to leave GETCMD. Would still be nice to trace through S\_EXIT to see which procedures are getting marked for exit.

12/16/2019:

* Appears to exit the first call to S\_EXIT without crashing
* ffrmPSysWindow is not created until TCharacterDriver gets created
* TCharacterDriver is not created until InitUnitTable is created
* frmPsysWindow must be created before InitUnitTable is called

12/17/2019:

Memory Leaks

* Continue Moving debugger code to DebuggerUnit.
* Move MemDumps to a new unit
* The pSysWindow is not getting freed

12/18/2019:

* Add the ability to display the debugger and show some information when a breakpoint is reached
* Hopefully, be able to display a decoded procedure in the p-Code window-- possible from p-Code saved to a database.

12/19/2019:

* AddProcInfo
* UpdateProcInfo
* SegmentTable
* Clean up references to TPsystemInterpreter-- perhaps just have a single global field!
* Where does CspOps table get initialized?
* CspTable[118] getting a range check exception. CspTable only has 52 entries
* The same breakpoint is getting added multiple times

12/20/2019:

* Could not compile used unit ‘pCodeDebugger’

12/21/2019:

* Currently getting a range check error in RPU. DbgCnt=40393, CurProc=2:Kernel.EXEC\_ERROR; O:99; LastOpCode=RPU
* Display: registers, stack, call stack

12/23/2019:

* Where is the “{$IfDef debugging}” coming from?
* Allow “Run to” breakpoints to be set in procedures that are not current.
* After hitting a break point, it will continue to hit the same breakpoint
* When doing a straight boot (from the Filer.Run menu), it is trying to use debugger info that has not been initialized, e.g. fpCodesProcTable
* “Step over” and “Step into” should be able to handle breakpoints.
* After HALTing, the p-Code monitor is not being released.

12/24/2019:

* Is the database “pcode” memo getting updated unnecessarily?
* USERPROG.FILE\_PRESENT converts

DB21 ( 9): SLDL1 // while (entry<=syscom^.gdirp^[0].dnumfiles) and (not found) do

Into

9:.dnumfiles) and (not found) do

* Also, I need warning about p-Code not being saved after changes are made to it

12/25/2019:

* “Step Into” does not seem to be working
* Add the ability to TAB within the p-Codes memo
* Display Current Line number & Current Column
* Start backing up pCodeDebugger frequently

12/27/2019:

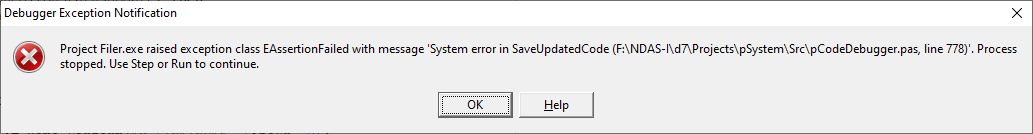
* Not tracing through the source code
* Modified needs more testing
* Source Code Step: Fetch until LineNr changes or procedure number changes
* IPC should be a global property in the debugger

12/28/2019:

* Toggle breakpoint not working
* Is S\_EXIT really supposed to be exiting the entire program? -- Or just …
* S\_EXIT source code got stored with USERPROG.Initialization. Same problem with other source code.
* F9 seems to get stuck in CHAR\_DEV\_GET @ Ofs 271- This also seems to lead to strange stuff on the p-System Screen. “Xxxxxxxxxxxxxxxxxxxxxxxxx”. Also hangs the system and cannot close any of the windows. Appears to be hung in TfrmPSysWindow.ReadKey- This may be occurring following an “Invalid Type Cast” error.
* Skip the locate prior to saving the code
* ‘Delete’ key does not work
* Separate memo field for procedure call definition?
* Change font on p-System registers to “Courier New”
* Add date/time added to table
* Need a SELECT ALL menu item for the memos
* Still occasionally trying to overwrite something
* If I say “No” to “Do you want to save?” it asks several times more.
* Need a “Find” command for the memos
* Crashes in the final call to RPU
* Why doesn’t USERPROG.LOAD display the p-Code? Sometimes? Maybe because of the mode (Run, Step Into, Step Over, etc)
* Breakpoint editor doesn't always remember the procedure name
* Add ^F2 reset
* Ability to saved changed breakpoints prior to exiting the program
* Putting a breakpint just after SYSRBOOT in READSEG and doing MemDump(SegBase, 'b') to see the segments as they get loaded
* KERNEL.FAULTHAN is calling KERNEL.LOADSEG
* Calling SMALLCOM

1/1/2020:

* When PROGRAM\_ASSOCATE got called, it didn’t load the raw p-Code?
* In PROGRAM\_ASSOCIATE, the call to (something at line 8), does not load the referenced code
* Does not step into DUMP\_ENVIRONMENT
* Something is still mis-positioning the file cursor:



* When code returns to a previous loaded procedure, does the gLast (stuff) get up-dated?
* Are things getting overwritten with newly created stuff (p-Codes)?

1/2/2020:

* Currently the Debugger belongs to the Interpreter
* Spurious CR or LF?
* Trying to get “Run to Here” to work.

1/3/2020:

* F8 does not exit on an RPU
* Multiple indications that the original source of the crash is occurring in HEAPOPS
* Unknown procedure “Liason”-- boolean that is true is there is a SPDEVICE:
  + I.e. LIAISON:=UNIT\_PRESENT( 'SPDEVICE' );
* Need to clear Memo fields if data cannot be loaded
* Look for the Pascal Beautifier
* “Find” command does not work in p-Code window
* Add a Tab to display the history list
* 332: SCXG1 GETPOOL // THIS IS WHERE THE CALL STACK GOES TO CRAP IN S\_EXIT
* When SINGLE\_STEP is called, it has already executed the instruction that OPCODE references.

1/4/2020:

* Add the ability to indent a section of code.
* When generating p-Code, automatically put a blank line after “store” and “jump”
* ^S for “Save Updated”
* Appears to be displaying the Fault Handler when it thinks that it is in “scantitle”
* The procedure with the dubious p-Code is “f\_open”, O#56
* Add F3 to “Find Again”

1/6/2020:

* After cancelling out from “Does not have a name” prompt, does not clear the memo field(s)
* There is a segment procedure called SEGFOPEN with these procedures:
  + LOCKFILUNLDIR
  + FOPEN\_ERR
  + SETTYPE
  + OPEN\_OLD\_FILE
  + OPEN\_NEW\_FILE
  + DIRECTOPEN
  + DOSOFTBUF
* It gets called by F\_OPEN

1/7/2020:

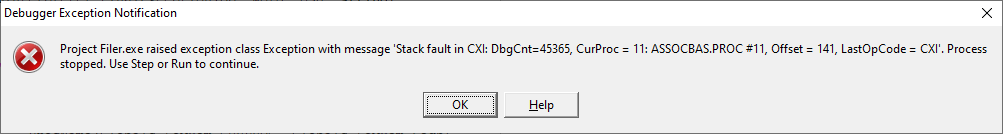
* SIBs are linked together in a code pool
* FAULTHAN seems to end up in SEGFOPEN at gDbgCnt = 41270 which shortly after FILEOPS gets “loaded”
* STRINGOP.SDELETE could not be loaded
* Getting to a SYSTEM HALT @ gDbgCnt = 41270 which appears to be happening in SEGFOPEN.SEGFOPEN
* Add ability to save only procedures/p-code >= aDate
* Refresh on SYSTEM HALT

1/8/2020:

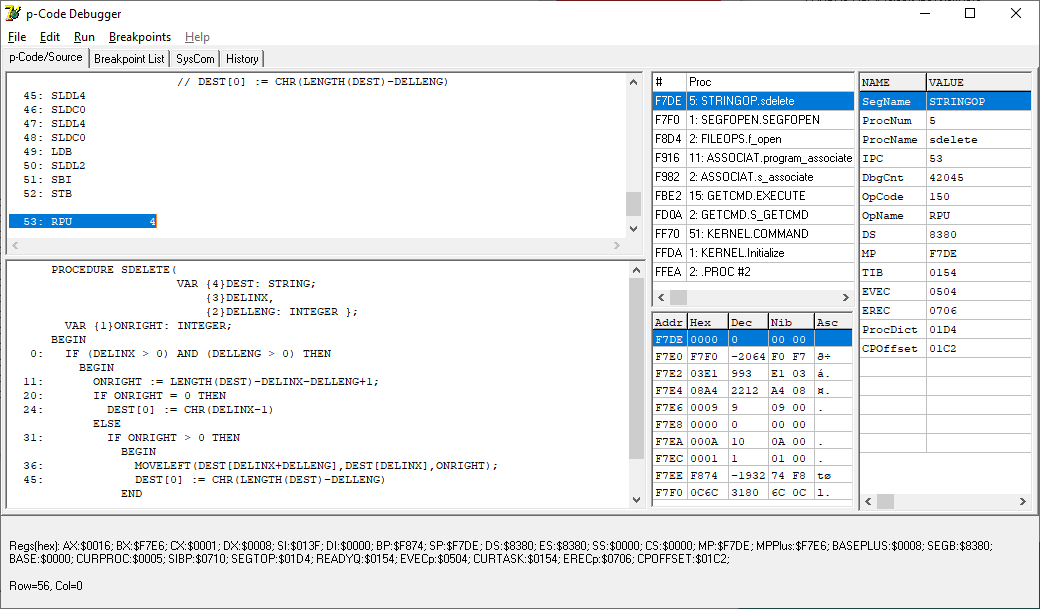
* I have a dozen different procedures in the DB all listed as “EXEC\_ERROR” in different segments.
* I also have a bunch of segments that appear to contain the PTR\_ ops such as PTR\_ADD
* This also appears to be true for a number of procedures but I don’t think that these were recent additions to the DB
* Remember which was the active memo after entering a new procedure
* SC\_Printable\_Chars seems to include almost everything:
  + 'Set(char) @ $F7CC [63436]: [#12..#15,#20,' '..'~',#128..#255]'
* I lost a bunch of changes that I made to the SCANTITLE p-Code
* Why is LDCN being used prior to a call to POS(?)
* SYSTEM HALT at DbgCnt 43539 or at 41270
* Seems like I am magically ending up at O:62 in SCANTITLE
* How do I end up in the middle of SCANTITLE (from SEGFOPEN)?
* SEGFOPEN does call SCANTITLE but I’m ending up in the middle. This appears to be happening when I am stepping through SEGFOPEN (p-Code). Take a look at the history. The history shows that I just exited UPCASE.
* SCANTITLE gets called from SEGFOPEN @ o:34. SCANTITLE calls UPCASE @ O:59
* I do not seem to be getting a “backup (1).pas” copy made. The dates are not getting updated. The final message does not indicate WHAT got updated.
* Is Single Step returning to the wrong caller? O:34 in SEGFOPEN.
  + SegIdx0 = sn\_Seg\_f\_Open, CurProc0 = 1, Was\_RPU = false
  + Its going to continue stepping through things until it encounters an RPU (which will occur) when exiting UPCASE.
  + When saving, give a reminder as to WHAT (p-Code, or .pas) is being saved. Don’t bother to save unless the date is later.
* Remove records which have no p-Code or source code.

1/9/2020:

* There were a lot of mis-saved records which belonged to Kernel but which were saved with other segment names
* USERPROG has two FLIPSEGs in the ProcNames[] list and KERNEL has one
* A lot of stuff listed as being in the KERNEL segment seems to duplicate stuff in the interpreter?



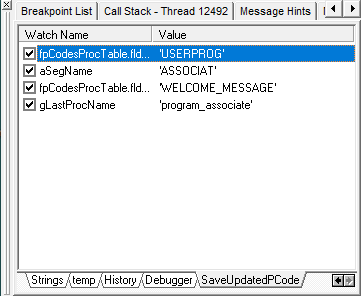
* Crashing with a STACK fault when exiting S\_DELETE:



* SDELETE should be exiting to SEGFOPEN but we seem to have exited to FILEOPS. This might be pCodeDebugger not handling F8 (to exit SDELETE) properly.
* The blowout is occurring in a call to CXI.
* The only obvious call to CXI occurs in ASSOCIAT.PROGRAM\_ASSOCIAT @ about O:150 or at O:138. There are two CXI calls. These are the first times that CXI gets called.
* Need a SAVE command for breakpoints.

1/10/2020:

* When stepping, it does not remember which window was being used.
* Apparently my hopes about a problem with CXI being the problem were not correct. Apparently caused by Delphi stepping through the wrong source code file. Or maybe not. Hitting CXI in a call to BlockRead?. Coming from an unknown segment? Maybe the load of FILEOPS has already blown everything.
* F9 is taking a break in Source code every time a procedure is returned from
* Getting as far as ASSOCIATE.PROGRAM\_ASSOCIATE o:55 (maybe further)
* Following a ^S (Save), I am getting an assertion failure in SaveUpdatedPCode. It appears to have moved to the last record in the file:



* I’m getting s SYSTEM HALT that does not actually seem to be the real SYSTEM HALT-- can keep on executing after it.
* Does not break at the call to HALT SYSTEM @253 in FAULTHAN
* System\_Halt := (Globals.LowMem.CURPROC = 1) and (fOp = OPCODE\_RPU);
* Needs to include proper segment in its determination
* The static CallStack is suspicious

1/11/2020:

* There is a VOLSEARCH in GETCMD
* There is a LVOLSEARCH in FILEOPS
* Ended at UPDATEXY in PASCALIO
* No calls to UREAD / UnitRead?

1/13/2020:

* STRINGOPS and CUPOPS have the same EREC

1/14/2020:

* Need a way to keep a watch on the “current” EREC & SIB. I.e., base on the value of ERECp, or SIBp or TIBp, etc.

1/15/2020:

* Where does the segment name get put into the SIB?
* Look for a segments being marked as not “Locked”

1/16/2020:

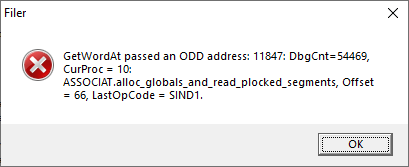
* Doing a “Program Reset” causes memory leaks
* Once again, it is clear that the call to READSEG from LOADSEG is the one that trashes memory
* The EVEC points back to an EREC?
* SI is saved as the IPC

1/17/2020:

* Getting an Odd Address error almost immediately in ParmDscr (from CSP from Fetch)
* Init\_Units\_Globals changes the MAINMSCW- source is completely commented out
* Not remembering where p-Code or sources are saved
* Save Updated not enabled and/or lblStatus not updated
* Getting an Assertion error when trying to save updated source code - SOMEHOW GOT ADVANCED TO WELCOME\_MESSAGE

1/18/2020:

* TIBIoResult = 9?, Proc #39-- Possibly in R\_READSEG
* When an Inspector is first created, it does not display its value-- NEED TO CALL RefreshAll or some such
* Is not stepping into READSEG
* When MemDump trys to display a SIb @ address $0710, it is showing the address as $0016 (but the SIB itself looks OK). This appears to be to be the SIB for STRINGOPS.
* SibAddr references should be local
* Only use PrefixInfo for the pCodeDebugger watch list
* Keeps forgetting about S\_GETCMD? I seem to have 9 different records for S\_GETCMD
* Changing the Residency in MemLockSeg does not reflect in the watch list
* The TIB has a field called START\_mscw. Can this be used when doing a callstack?
* No ScrollBars on the Watches grid (or any other?)



ASSOCIAT.alloc\_globals\_and\_read\_plocked\_segments

1/20/2020:

* Fault\_Message @ $F8 - Fault\_Type does not make sense
* The fault is referencing the SIB to change residency BUT the SIB does not seem to be valid
* Find out when the fault TIB gets changed
* I’m only getting ONE break into the faulthandler? Perhaps my changes for dbMemChanged have broken the breakpoints.
* Closing the debugger is NOT closing the Inspectors
* I’m getting a breakpoint when entering init\_screenops. Perhaps the fault\_type has been cleared.
* Closing the debugger ought to be sending a message to its creator to indicate that it has been freed. Likewise for the inspectors.
* Residency still does not get updated when I would expect it to be

1/21/2020:

* It appears tht ‘residency’ @ $DA56 is going to get changed. This does not seem to coincide with any of my known SIBs? Appears to be trying to process CUPOPS.
* frmDecodeRange does not exist when needed in DecodeWindow
* Ptr\_sub not making sense. MaxPool =$2c66, MinPool=$2c64 but PTR\_SUB gives $5622??
* Pool\_adjust getting called with req\_seg\_length = 0
* THIS IS THE ONLY QUESTION: IS PTR\_SUB WORKING OR NOT? PTR\_SUB gets called with space on the stack for 4 parameters and a function result.
* CallStackDblClick

1/22/2020:

* FAULTHAN seems to have a value of $DA4C for the SIB but I don’t see it in the fault message. I can get there by following FAULT\_EREC^.ENV\_SIB (This uses a hard-wired address $DA4C-- taken from the executing p-code rather then from the Fault\_EREC).
  + $F8 is the address for syscom^.fault\_sem.message. This agrees with my value.
  + $0154 is the address is p-Code using for faulttib^.regs. This is TIB. Using the env field from this TIB give the address of the EREC $da42. (This is a different address than the one from FAULT\_EREC.)
  + Using the ENV\_SIB field from that record, gives an address of $DA4C which agrees with the p-Code.
* OK. I now have the SIB showing to be located at $DA4C.
* OK. MemLock\_SEG seems to be working
* None of the SIB addresses that I have saved seem to match the segments where the faults are occurring.
* Globals.Lowmem.BASE becomes the static link for the next frame
* Ptr\_sub says that it needs 0 words in the activation record. The two parameters have been pushed onto the stack.
* What is SEG\_ADDR in a SIB used for?

1/23/2020:

* Getting a break at 46230, in #11:ASSOCIAT.PROGRAM\_ASSOCIATE, O#138. The call stack may be corrupted.
* Getting some problem in PROGRAM\_ASSOCIATE-- maybe at the call to F\_INIT at IPC = 8. This causes a segment fault. This is going to load something in at addr $D170 (53616). The “something” is FILEOPS (surprise, surprise!).
* HEAPBASE is $D962. There are 2034 bytes from $D170 to $D962 (i.e., not enough to load FILEOPS). It is possible that the initial USERPROG is loaded there.
  + The initial segment dictionary gets loaded to 62974 ($F5FE)
  + USERPROG claims its length to be 3520 words (7040 bytes). 1st block of USERPROG is at 247 (SYSTEM.PASCAL).
  + USERBASE is set to 55920 ($DA70).
  + USERPROG gets loaded to 55920 ($DA70). It extends to 55920 + 7040 → 62960 ($F5F0).
  + Global SIB (Globals.Lowmem.SIBp) is at $DA4C .
  + Segment dictionary (BX) is located at 62974 ($F5FE).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Address | Len | High Addr | Contents |  |
| 62974 | 512 | 63486 | Initial segment dictionary |  |
| 55920 | 3520 | 62960 | UserProg |  |
| 55858 | 8 | 55866 | MSCW |  |
|  |  |  |  |  |
|  |  |  |  |  |

1/24/2020:

* Is the pool outside on the HP?
* I may be able to get the datasize for a procedure from the segment dictionary field data\_size:
  + SegmentDiscionary.Seg\_Family[ProcNo].Data\_Size
* Continue tracing through FaultHandler to see

1/25/2020:

* Could FreeSpaceInfo and SocketPoolInfo be reversed?
* Working on INIT\_INTERNAL\_POOL
* BX is the MP?
* In a TIB, SP\_Low is the upper boundary for an internal pool
* HEAP → CODE POOL → STACK
* MemDump($1712, 'p','PoolDescInfo', 0): PoolBase:$0:$0, PoolSize:$0, MinOffset:$2CD0, MaxOffset$2CD0, Resolution:$10, PoolHead:$0, PermSIB:$0, Extended:false, NextPool:$1712, MustCompact:true

1/27/2020:

* Try to align INIT\_INTERNAL\_POOL p-Code & Source Code

1/28/2020:

* How often is FILLCHAR getting called?
* Need ability to paste a “variable list” into local variables
* Any change should update the grid
* Only update the memo when necessary
* The local variables come first. THEN the parameters.
* Switching to a different procedure is going to change SEGNAME/PROCNAME
* Cannot paste to source memo?
* Exception while processing exception
* ANYSEGMENT/ANYPROC
* Can’t adjust LocalParams column width
* Copy/paste does not work with docs delphi
* Going into an infinite loop of BadExpression calls
* Need cut, copy, paste for mmoVariables

1/29/2020:

* Add “Alias” WatchTypes
* SendMessage to add LocalVars window
* Pointer size for FIB

1/30/2020:

* When doing an “INSPECT” on a local variable, it is using the parameter address rather than what it is pointing to
* Still getting memory loss occasionally
* The inspectors are not getting updated
* p-Code for UNIT\_PRESENT had a lot of crap in front of the offsets
* UNIT\_PRESENT is not displaying type ALPHA correctly

1/31/2020:

* MemDump for heap\_info & task\_info
* Does an Inspector get properly added and deleted from the list?
* MaxOffset = MinOffset?
* My definition for erec does not seem to correspond to the p-System definition: link\_count & next\_erec switched?
* Code to look into:
  + Init\_heap- make a map of memory usage
  + Init\_environment- need to cleanup source code & p-code
  + Start from the READSEG crash and work backwords

2/1/2020:

* Gets as far as gDbgCnt (past the load of FILEOPS) before crashing -- possibly starting in “alloc\_globals\_and\_read\_plocked\_segments”. Appears to be calling without returning? Maybe this is because S\_EXIT is somehow skipping over the RPU?
* The parameters to S\_EXIT are not displayed properly
* Locals allocations:
  + Local Variables
  + Unnamed Variables
  + Parameters (allocated in reverse order!)
  + result
* Important values:
  + codepool^.minoffset - lowest address of codepool
  + Heapinfo.heaptop - highest address in heap
* Getting the error “Range Check Error” at the line “CPOFFSET := …” in NEWENV, gDbgCnt = 51008 on an RPU instruction when exiting J\_NEW

49865: HEAPOPS.J\_NEW 197 RPU

49886: EXTRAHEA.VarNew 152 RPU

* Are the procedure numbers displaying as negative when executing S\_EXIT?

2/3/2020:

* MIght be nice to have a complete map of memory
* Why is the first MSCW changing when the faulthandler is entered?
* I am getting a MemChanged break on every step in FAULTHAN
* POOLSEG seems to be the segment that I need to understand.
* POOLSEG
  + The parameter (env\_rec : e\_rec\_ptr) does not seem to be actually pointing to a valid EREC- OK once the ENV\_REC.E\_REC\_PTR is located at offset 18.
  + NEXT\_SIB & PREV\_SIB SEEM to be mis-ordered
  + MIN\_ADDR seems to be located at 2 correctly
  + MAX\_ADDR OK
  + DX refers to the Global variables
  + When I create in INSPECTOR window, I should set the COMMENT field to the “variable\_name”
  + (CODEPOOL appears to be at offset 113 in the Kernel Globals)
    - I think this is what MINOFFSET & MAXOFFSET are referring to
    - NEEDED\_SPACE should be located at 12. Its not.
    - The parameter ENV\_REC should be at ofs 18
    - The SPACE\_NEEDED for SCREENOP (1144) agrees with the previous values that I have seen. (Word count = 2288, bytecount = 1144)?
    - Is LOADSEG getting called with an invalid EREC?
    - LOADSEG is called by POOLSEG @ 234
    - LOADSEG appears to have 10 words of local variables preceding seg\_erec: erec\_p
  + In FAULT\_MESSAGE, do I have MESSAGE\_SEM & REAL\_SEM in the wrong order? NO.

2/4/2020:

* Trys to access SCREENOP. Fault. Gets loaded at $6670
* Residency has an RefdAddr address of 1. The VAR parameter “residency” has an address of 5697 which (used as an indirect”), leads to an ODD address. The actual address of “residency” is $DA56.
* I think that maybe the RPU parameter give the offset to the procedure parameters?
* Global parameters is offset from a segment’s data area. This is NOT RELATIVE TO SYSCOM OR TO ABSOLUTE ADDRESSES.
* Need to fix the address offsets for POOLSEG
* Might be able to use the SIB to get a data length (and thus calculate offsets for parameters).
* KEEP A WATCH on the EREC, SIB and BASE for FILEOPS.
* The SIB for FILEOPS gets created by GET\_PSEUDO\_SIB at DbgCnt = 8111.
* Need a definition for a PED\_PSEUDO\_SIB and for erecsib\_array. Maybe these should be put into the DB?

2/5/2020:

* Seg\_Base contains the current memory address of the code segment. If the code segment is not in memory, Seg\_Base contains NIL.
* Possibly save Inspector info to the Settings file using the “Inspector Name” (MemDump(...)) as the key.
* The address of the SIB is $AD6 (but I calculate it to $AD0)?
* None of my SIBs have a value for ?SegBase, SegRefs, TimeStamp, SegPieces, Residency (at least in GET\_PSEUDO\_SIB).
* Why does FILEOPS appear to have a DATASIZE of 1?
* Host\_Erec.link\_count did not get updated
* FileOps.NextSIB & PrevSIB do not seem to be linked correctly?
* Posting in the p-Code debugger could overwrite the wrong record. Need an ASSERT.
* KERNEL.POOLSEG source code offsets are incorrect. Might do better to re-compile

2/6/2020:

* Why amI getting a MemChanged breakpoint when first entering FAULTHAN and when entering POOLSEG? The data looks identical.
* I appear to be getting a phoney MemChanged breakpoint everytime that I enter POOLSEG.
* Only one codepool. MinOffset:$65F0, MaxOffset:$8560
* Getting a MemChanged break even though Brk has not been changed? This may be doing it:
  + ((CURPROC = TgtProc) and (IPC = TgtIPC)) or

2/7/2020:

* POOLSEG does not get called again until after eXecuting “HELLO”
* FILEOPS claims to need only ONE word for data. Is this for the segment or for a procedure.
* We may be attempting to call F\_CLOSE in FILEOPS.
* Procedure 50 (POOLSEG) calling procedure 3 (F\_CLOSE)?
* POOLSEG is called by FAULTHAN @ O: 59 (which is a segment fault). The segment should be locked but does not appear to be?
* FIND command does not work.
* Form.MonitorNum

2/8/2020:

* Lowest memory available: MainTask (TTib) @$0154 SP\_LOW
* POOL\_ADJUST: p-Code does not match source code
* POOL\_ADJUST gets called seven times (DbgCnt: 18951, 20398, 27981, 28737, 29200, 29567, 30353) before memory gets trashed
* TAKE A LOOK at DUMP SEGS
* Where does FILEOPS originally get loaded?
* It is loading into the stack area. It should not be doing that.

2/11/2020

* MaxAddr all seem to be too high- break on MaxAddr change
* Are the words of “poolbase” properly ordered
* MinOffset comes from heap\_info.heap\_top (to resolution boundary)
* Is it possible that when I create the 1st MSCW, I am not setting the upper bound for the task?
  + MainMscwp = $170 (368)
  + This is nowhere near the high memory address that I am seeing for the first procedure
  + Could it be related to DS or ES?

2/12/2020:

* The codepool is located at $1712 on both machines.
* MainTask.Regs is $0154
* In INIT\_INTERNAL\_POOL, both look identical: PoolDescInfo @ $1712 [ 5906]: PoolBase:$0:$0, PoolSize:$0, MinOffset:$2CD0, MaxOffset:$2D50, Resolution:$10, PoolHead:$0, PermSIB:$0, Extended:false, NextPool:$1712, MustCompact:true
* In INIT\_INTERNAL\_POOL, the MainTask values are the same.
* Exiting POOL\_ADJUST, CODEPOOLs are the same.
* MaxAddr is $FDC6 but the original MSCW is at $DA32. I.e., they overlap. $FDC6 comes from the current stack (SP). It does not seem to be coming from SP\_LOW of the main task.
* Why do calls to SERIAL\_FIB\_INIT (and other things) seem to be changing the MAINTASK TIB?
* Getting a “Range Check Error” @ DbgCnt 33451 in an LLA instruction, Current Procedure 11:Associate.program\_associate, called by 2:associat.s\_associat, 2:getcmd.s\_getcmd, 51:kernel.command, 1:kernel.kernel9
* NATNFO instruction not implemented

2/13/2020:

* Getting both a SEGMENT fault and a STACK fault in CXI when trying to load something but the MCSW has already been trashed
* MainTask will get changed every time that the LPR op is executed
* Does MinOffset get properly set?
* Remember positions of Inspectors based on WatchType
* Would be nice to highlight variables that have been changed each time an update occurs
* The LOADSEG which loads FILEOPS gets called at DbgCnt 40970 (called by POOLSEG). POOLSEG was called by FAULTHAN while processing fault seg.

2/14/2020:

* In OSUTIL call, MaxOffset is BF80, but READ\_LOC $BA90. MAXOFFSET did not get bumped at line 236? BX not valid? Erroneous assumptions.
* Most of the MaxAddr values, exceed the original MSCW addr of $DA32 (Why this location? This implies a stack of nearly 10,000 bytes?)
* Going to load 2,240 bytes at $D170. I.e., $D170 thru $DA30.
* Both the MainTask TIB and first MSCW seem to be located at $DA32- NO? The MainTask TIB is at 368 ($170)
* Still consider the possibility that my disk read is assuming a byte count parameter rather than a word count. I really don’t think that this is true.
* Where does the codepool set MaxOffset? It is getting set in POOLSEG (at least) from the MainTask SP.

2/15/2020:

* Where does HEAPTOP get set. It gets set when entering OS\_INIT.
* SP gets set to $FFF4 in OS\_INIT, IPC:91 and then enters the KERNEL. Trying to display local variables in KERNEL.KERNEL gives a range check error. This occurs in DisplayLocalParameters on this line “ParamAddr := Offset + BaseRegister”. This occurs because the BASEREGISTER value is $FFF2 which, when added to 60 gives an overflow.

2/17/2020:

* Is it possible that I should be using the registers from CURTASK? No. I don’t think so. LPR saves the registers to the current task TIB.
* When MOVELEFT is called, what are the four (4) things on the stack?
* 74: moveleft( kernel\_erec^.env\_data.c^[0], new\_mp^, sizeof(mscw) );
* {note that the mscw just copied should have the static and dynamic links
* pointing to itself so this new mscw will already be pointing to it}
* The above statement does not appear to be true. Both static and dynamic links are 0?
* Something is smashing the MSCW at FFEA.
* The final range check error occurs when trying to display the call stack in the debugger
* By the time that I get into ADD\_LIBRARY, the call stack already looks fishy. This is because USERPROG is being referenced by the MSCW but it is no longer relevent.
* Add the ability to look at an MSCW directly from the call stack

2/18/2020:

* The MSCW changes because the procedure number changed. The procedure number changed because of the SPR call (which updated the saved registers).
* The MSCW at FFEA does not seem to get changed. It does not seem to be a valid MSCW.
* Getting an “Out of Memory” error at around DbgCnt 52083

2/19/2020:

* KERNEL.KERNEL static link is to $0170. Dynamic link is to $FFEA. Why wouldn’t these be the same?
* The MSCW at $FFEA is referring to USERPROG segment which is no longer in use.
* Still getting “Out of Memory” at DbgCnt 52083. I don’t know why. The calls history doesn’t seem to show anything useful happening.
* EVENTQIB seems like it is located in an odd location. I am also dubious that it ever gets used.
* When I run without the debugger, I get a “Range Check Error” @ multi millions of instructions

2/20/2020:

* Removing ALL of the watches and breakpoints did not change the “Out of Memory” error at all.
* The OOM call is occurring in LogProcCall (from BldFrm, from CPG). Trying to log a call to KERNEL.MEMLOCK\_SEG from KERNEL.FAULTHAN.
* Running without LOGCALLS generates millions of instructions but does not get the OOM.
* Still getting an OOM at DbgCnt 52083 even with strings changed to short strings.
* This is more likely to be memory getting corrupted.
* Maybe the LOAD address should be the OFFSET within the pool rather than an absolute address! This is true, but for me, the codepool is always located at 0. Should it be?
* FAULTHAN calls POOLSEG.
* LoadSeg is called by FAULTHAN.
* POOLSEG only getting called once?
* The DataSize for all of these SIBs is small.
* The SegBase for FILEOPS is $D170. The SegLeng is 2240 which means that it will extend to $E2F0. The EREC is located at $DA42. The EREC will get overwritten by the LOADSEG.
* Watch carefully the MSCW located at $2C86 when calling LOADSEG.
* Would be nice if double-clicking on a “Call Stack” item correctly positioned the IPC.
* POOL\_ADJUST sets the MAXPOOL to $D8F4. This exceeds the loading location for FILEOPS which is $D170.
* MainTask.Regs.SP = $D944. STACKSLOP is 40 words or 80 bytes. This gives a value for MAXPOOL of $D8F4. FILEOPS is going to load beyond this. The “requested stack” (req\_stack) was 0. The test (at line 25 in POOL\_ADJUST) seems to be comparing the space needed to the stack pointer address (without adding in something(?)). This all seems very fishy.

2/21/2020:

* In the call stack, INIT\_KERNEL\_GLOBALS shows a very weird IPC value (1138) which is beyond the end of the procedure.
* The “Requested Segment Length” (REQ\_SEG\_LENGTH) appears to be 0?
* Being called from the fault handler (FAULTHAN) via a POOL FAULT { Pool\_Adjust(0, 0, p) }.
* This sets both REQ\_STACK and REQ\_SEG\_LENGTH to 0 { maybe trying to unload? }

2/24/2020:

* Is it KERNEL.PUT\_POOL\_BYTES or USERPROG.INIT\_INTERNAL\_POOL getting called?
* Why isn’t OS\_INIT is the call stack? Because OS\_INIT cleans out the call stack when it calls the Kernel.
* When USERPROG.INIT\_INTERNAL\_POOL is called, the message handler logs it as KERNEL.PUT\_POOL\_BYTES.
* SegNameIdx will be wrong for dbMemChanged.

2/25/2020:

* Does FIND\_SEGS\_IN\_POOL ever get called? No. only for external pool.
* Does HP use an external pool? No. It does not.
* Does poolsize (pool\_size) ever get changed? No. I don’t think so.
* Still have a mystery as to why gotoxy gets moved by 2 bytes.
* Max\_addr is based on the MainTask.SP ($F97C)
* MaxOffset (in the codepool) is set to $E2F0 in POOLSEG at line 203. It was $D170 when I loaded HEAPOPS.
* What is getting overwritten from $D170 → $E2F0? Note above indicates that it is an EREC getting overwritten. I do have a chain of erecs: $0876→ $0848→ $081A→ $07ec→ $07be→ $0790→ $0762→ $0734→ $0706→ …→ $67C. There is no indication of
* LOOK FOR NOTE ABOVE ABOUT DUPLICATE Erecs? I do not see an EREC located at $DA42. It is possible that there IS an EREC located at $DA42. It appears to be connected to the USERPROG base info.
* (from above) I appear to have references to two different ERECs:

1st: Just before SYSIO

2nd:

1st: 'EREC @ $DA42 [55874]: $0170 [ 368]; Env\_Vect: $04B0 [ 1200]; Env\_Sib: $DA4C [55884]'

2nd: 'EREC @ $08A4 [ 2212]: $0C6C [ 3180]; Env\_Vect: $058C [ 1420]; Env\_Sib: $08AE [ 2222]'

* Would be nice to be able to duplicate an Inspector. Popup?

2/26/2020:

* There does appear to be an EREC for USERPROG located at $DA42. This IS the EREC created during the boot process.
* The corresponding SIB for USERPROG is located at $DA4C.
* USERPROG segment base is located at $DA70.
* Should ROOTTASKP be a global variable? Yes. It is.
* Start\_MSCW, in the RootTask TIB, does not seem to have been set. Fixed. Does not seem to matter.
* The MAINMSCW seems to be incomplete. Trace through HP code to see where MAINMSCW gets changed.
* The STATLINK & DYNLINK are already set (to point to MAINMSCW) when “Step 1” is reached.
* I do not see where MAINMSCW is actually defined. It is dynamically allocated on the stack. Assholes! It is named MAIN\_MSCW!
* SSDSVAL appears to be obsolete and should go AWAY!
* The USERPROG MSCW is located at $DA32.
* SLDO saying that SysCom is loaded at 0? Fixed.
* Don’t forget to restore the original version of INTERP7.PAS. Fixed.
* After changes to the bootstrap process, I am back to “Out of Memory” ar 52083.
* INIT\_KERNEL\_GLOBALS has this note:
* Dying in OS\_INIT after MP gets set to $FFF4. This might be my code trying to display an infinite call stack.
* FILEOPS is still smashing memory when it loads. The base address is $D170.
* USERPROG segment base is located at $D170. It is going to load as far as $E2F0 (which will overwrite the USERPROG info that I am using to get segment names). Is USERPROG still in use or not? I think so because USERPROG.INIT\_KERNEL\_GLOBALS is still in the call stack.
* I might be able to use SegNames[] and ProcNames[] as an alternative way to display the call stack. Probably not since I would have to be able to know the segment name to get the SegNameIdx.?

2/27/2020:

* The call to load FILEOPS occurs at DbgCnt = 41101.
* The OOM exception occurs at DbgCnt 52083 (10000+ later).
* The final call to POOLSEG shows mucked up memory for the EREC.
* SPR “in case something vital changed”-- but then overwrites it from the EREC?
* With the “revised” version of “Restore”. I end up in INNER at the raise ESEGBACK.Create() line with the XPS-8930 tying up the CPU. Appears to be caught in an exception loop. NEWSEGMENT, NEWENV
* Enters RESTORE (1ST time) with DbgCnt of 15560 on Studio, but 16112 on XPS? This appears to be related to using a different version of PSYSTEMy.vol.

2/28/2020:

* Reference to WordAt[SSDSVAL+DI] in SIG when DI hasn’t been given a value? No. DI is the semaphore being signaled.
* Gets to the breakpoint after a LONG pause. Sem count shows as (either) 32768 OR -32768.
* Getting hung on a WAIT in the FAULTHANDLER.
* Old version makes it to the command line. New version gets lost.
* Gets lost at the call to DISPLAY\_TITLE? Fault Handler.
* Procedure 20:GetCmd.Display\_ProgName, Offset 13, 14 repeatedly.
* CXG is getting repeated Segment faults
* Old version never reaches Display\_Progname? REstarting Delphi fixed this.
* Old version is not breaking at breakpoints? Maybe there was a flaw in the debugger(). I have replaced with current version.
* On line 14 of GETCND.DISPLAY\_PROGNAME. The new version is getting caught in a loop which repeatedly updates semaphore wait count at $F4. The old version gets past it in three steps.
* Is the call to SEGBACK being handled differently?
* At DbgCnt, I am getting a range check error in SRS at line 7432. Could this be the change that I made in NEWENV?
* This is a result of the change to NEWENV. Is occurring in INIT\_DEVICES, o:34
* im getting very different values for how many byte MYFILLCHAR is supposed to fill:
* 'Regs(hex): AX:$0000; BX:$D93A; CX:$0600; DX:$0178; SI:$158F; DI:$0015; BP:$0001; SP:$D92E; DS:$DA70; ES:$DA70; SS:$0000; CS:$0000; MP:$D932; MPPlus:$D93A; BASEPLUS:$0178; SEGB:$DA70; BASE:$0170; CURPROC:$0023; SIBP:$DA4C; SEGTOP:$1B7E; READYQ:$0154; EVECp:$04B0; CURTASK:$0154; ERECp:$DA42; CPOFFSET:$1854; '
* 'Regs(hex): AX:$0000; BX:$D93A; CX:$0018; DX:$0BE8; SI:$158F; DI:$0015; BP:$0001; SP:$D92E; DS:$DA70; ES:$DA70; SS:$0000; CS:$0000; MP:$D932; MPPlus:$D93A; BASEPLUS:$0178; SEGB:$DA70; BASE:$0170; CURPROC:$0023; SIBP:$DA4C; SEGTOP:$1B7E; READYQ:$0154; EVECp:$04B0; CURTASK:$0154; ERECp:$DA42; CPOFFSET:$1854; '
* MAXUNIT is 0?
* New version going to fill 24 bytes starting at address 0? New version thinks that there are 0 units, old version thinks that there are 63. Was caused by my revisions to NEWENV not setting BX. I should have considered that.
* Currently, I am getting caught by something incrementing the semaphore count when display\_progname tries to call display\_title.

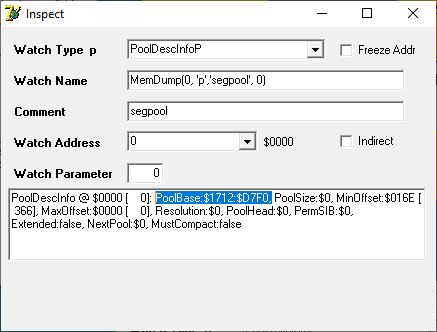
2/29/2020:

* The (2nd) call to WAIT in FAULTHANDLER, goes awry in the new version (following the call to display\_progname(?)).
* New version exits DELINK with DI = 14. Old version exits with 244.
* New version enters WAIT 14 on stack. Ole version enters WAIT with 244 on stack.
* Both have the correct semaphore address in FAULTCOM (244).
* The TASK\_LINK differs in DEQUE. Does not seem to set it to 0. Restore was not setting DX.
* The OOM exception occurs at DbgCnt 52083 (Old version).
* The OOM exception occurs at DbgCnt 52371 (New version).
* Both seem to be occurring in the fault handler, at line 49. At the call to MEMLOCK\_SEG.
* Although the DbgCnt differ by about 300 instructions, the call stacks look very similar-- a lot segment faults.
* The registers look identical.
* ENQUE returning different results. This is the call at line 8021 in WAIT. Old way returns AX 340, New returns 55600. DbgCnt = 18689. Executing WAIT instruction.

3/2/2020:

* (@18689) Delink getting entered with different value for DI
* Looping in WAITER: IPC: 3, 4, 5, 6, 8. Tight loop doing a USTATUS. Old version never goes there.
* TASK\_START hits SIG
* Starts looping around 18692 in #27:INIT\_KERNEL\_GLOBALS, ipc:1121
* Getting a different output from DEQUE (BX=340, TibAddr=55600). 1st pass thru is the same: 0, 646 on both.
* Something has altered the WAIT\_Q before the second pass
* It is the WAIT on the “magic” TIB that causes the problem
* Why is the Sem\_Count getting set to 646?
* The old parameters to ENQUE are AX & BX. Old delink returned result to BP.

3/3/2020:

* Get a ERangeError on both systems @ DbgCnt 30559. PASCALIO.f\_writestring. Has not yet displayed the system prompt. SI-ProcCode-1 = -1? In SegInfo ← TPsystemInterpreter.MemDump ← TfrmPCodeDebugger.MemDump ← DisplayWatches ← UpdateDebuggerDisplay ← Run3Click. SegAdr = 46208 ($B480). ('SCXG5', $7FD5DCA0, 3). Just the debugger with a watch set on a bad segment base.
* When did gdirp get cleared?
* Range check error when exiting varnew
* Loading HELLO.CODE
* Alloc\_Globals\_and\_read\_plocked\_segments is only trying to clear 5+5 words? Where does n\_global\_words get set?
* The procedure names that I have for UNIT ASSOCIAT do not agree with what I am seeing in the source code at all.
* Goes into “never-never” land in Alloc\_Globals\_and\_read\_plocked\_segments
* When entering ptr\_add (called from GETCMD.LEAVE\_GETCMD), the pointer passed in is
* I seem to have gotten lost in a series of Faults. FAULTHAN → MEMLOCK\_SEG → KERNEL.KERNEL → KERNEL.KERNEL. It seems like KERNEL.KERNEL exits to KERNEL.KERNEL? Infinite series of faults referencing KERNEL?
* 
* Why does the POOLBASE display this way? Pooldescinfo @ 0?

3/4/2020:

1. In POOLSEG, SEGPOOL is zero but is supposed to contain the address of the CODEPOOL.
2. Odd that in SYSCOM, FAULT\_EREC has the same value as FAULT\_WORDS? FAULT\_WORDS is intentionally set to trash.
3. Watch the LCL\_SEGPOOL SIB. When does low memory get blown?
4. Low memory gets blown at DbgCnt 51351, KERNEL.POOLSEG, O:305. The EREC which was passed in from INIT\_KERNEL\_GLOBALS, seems to be nonsense. Called from FAULTHAN at line 57. The FAULT\_EREC is NIL. Looks like it is doing LEAVE\_GETCMD.
5. I show FAULT\_EREC to look like this:
6. 250: 20: fault\_e\_rec = $DA42 [55874]EREC @ $DA42 [55874]: Env\_Data: $0CD7 [ 3287]; Env\_Vect: $E82F [59439]; Env\_Sib: $0000 [ 0]; link\_count: -30099; next\_rec: $0002 [ 2]
7. Looks very hinky. Address is 250.
8. I’m looking at an EREC at $DA42. Was $DA42 ever a valid EREC? No. I don’t think so.
9. The first stack fault occurs in PROGRAM\_ASSOCIATE at DbgCnt 45357. It wants 2254 words.
10. I have two DIFFERENT stack fault procs: STKBACK & SSTKBACK. However, the exception handling code, will ONLY call SSTKBACK -- never STKBACK ?
11. Still overwriting low memory at “segpool := codepool;” in KERNEL.POOLSEG (line 305).
12. SSTKBACK @ DbgCnt = 45313 in old. Then goes to OOM.
13. SSTKBACK @ DbgCnt = 45357 in new. Then hits low memory change at 50775.

3/5/2020:

* LCL\_SEGPOOL := LCL\_ENVREC^.ENV\_SIB @ poolseg, line 144
* Loads …, OSUTIL, ASSOCIAT, EXTRAHEA, HEAPOPS, FILEOPS (DbgCnt = 40601), SEGFOPEN, LOCK, EXTRAIO, ASSOCBAS
* Get an SSTKBACK fault at DbgCnt: 45357. First one. Occurring in ASSOCIAT. PROGRAM\_ASSOCIATE.
  + The *offending* EREC is @ 0BB2 and the *offending* SIB is @ 0BBC. Both look plausible.
  + The *restored* EREC is @ 09E6 and the *restored* SIB is @ 09F0.
  + Everything looks plausible.
  + Gets restored to EREC @ 0BB2 and SIB @ 0BBC.
  + This the FIRST stack fault.
  + At 50680, I have LCL\_ENVREC pointing to a bad EREC @ $DA42. (see above). This is ~5000 instructions after the stack fault.
  + POOL\_SEG is called from the FAULTHAN (only on a segment fault).
  + This EREC was passed in as a parameter to POOLSEG (probably called from FAULTHAN).
  + How did I get into POOLSEG w/o the FAULTHAN being entered?
  + Does segment USERPROG also have a FAULTHAN?
  + The first fault (a segment fault: SEGBACK/SEGFAULT/FaultCom) occurs at DbgCnt 35188.
  + The second fault (a segment fault: SEGBACK/SEGFAULT/FaultCom) occurs at DbgCnt 35734.
  + Third fault is @ 36101. (Still no entry into the faulthandler?).
  + Fourth fault @ 40393.
  + Fifth fault @ 40880.
  + @ 42226.
  + @ 44488.
* STKFAULT does not seem to be updating the fault message (fault\_words or fault\_type).
* My stack fault handler (Fault\_Message) is located at $F8.
* Why doesn’t the assignment (lcl\_segpool := {kernel.kernel}codepool;) seem to change anything?
* Crazy EREC occurs at 51189.
* Next: figure out why my assignments in STKFAULT don’t seem to change anything. Was not looking in the right format. Needs to use “FaultMessage” format.
* The stack fault occurs after 46796. The fault message looks like nonsense when STKFAULT is entered.
* The fault\_message is changed by SIGNAL (@ DbgCnt = 47279). Changed these:
  + fault\_erec = $0B28 [ 2856], fault\_words = 0, fault\_type = Segment Fault
* This code:

305: LLA 17 // ^procdict

307: LLA 18 // ^p\_desc

309: CAP 1

* In S\_EXIT, seems to be crapping up the fault message EREC ($F8). CAP is the most likely suspect. DbgCnt < 48014.
* Next: is the fault handler looking at the same information that I am?

Notes:

* Would be nice to highlight the changed row in the Global/Local variables.
* Why did I stop displaying “Seg\_Pool” when I display a SIB? Was changed on Feb 6, 2020. I don’t know but I put it back.
* Ability to highlight lines in Global/Local parameters that have changed
* Would be nice to remember the splitter locations.
* Would be nice to include the DbgCnt in the messages.
* Not displaying integer datatypes in “Memory Changed” breakpoint message.
* “Delete All” breakpoints (or any others) should give a warning message before doing so.

3/6/2020:

* In S\_EXIT, seems to be crapping up the fault message EREC ($F8). CAP is the most likely suspect. DbgCnt < 51118. This appears to be the first time that the “parameter descriptor” has a Parm\_Addr of 0.
* FaultMessage @ $00F8 [ 248]: fault\_tib = $0154 [ 340], fault\_erec = $DA42 [55874], fault\_words = 0, fault\_type = Segment Fault
* TIB @ $0154 [ 340]: wait\_q: 286,prior:80,flags:0,sp\_low:D7F0, sp\_upr:FFFE,sp:FCC2,mp:FCC4,task\_link: 2A2,ipc: CF3, env: 67C,ProcNum:17,TIBIoResult: 0,hang\_p: 0,m\_depend: 0,task\_stuff: 3, Start\_Mscw: 170
* EREC @ $DA42 [55874]: Env\_Data: $0CD7 [ 3287]; Env\_Vect: $E82F [59439]; Env\_Sib: $0000 [ 0]; link\_count: -30099; next\_rec: $0002 [ 2]
* The bad EREC is $DA42
* The change appears to be occurring BEFORE the call to CAP (on an LLA instruction?).
* The next instruction executed (SLDO1) is in the FAULTHAN @ O:6.
* Breaking on the change at FaultMessage.EREC = $DA42, occurs in my debugger BEFORE Delphi sees it (DbgCnt = 50612). Does seem like the last op was a “CAP”.
* The location where “WordAt[$FA] = $DA42” is NOT consistent: DbgCnt: 51188, 51476, 51700
* The p\_desc appears to contain the value $DA42 (which is NOT a valid EREC).
* Question: where/how does p\_desc get filled in? Is this taken directly from the segment dictionary?
* The addr\_of\_ERec comes directly from the MSCW
* The MSCW record on the call stack (located at $FFEA) does have an MSENV value of $DA42 (which is the value that later corrupts the FAULT message.
* The bad EREC address seems to be set upon entry into the kernel (following call from OS\_INIT).
* This is occurring *directly* when OS\_INIT calls the KERNEL.
* It is occurring in the CHGSIB call in CXGIMMED.
* $DA42 is coming from Globals.Lowmem.OLDEREC
* Globals.Lowmem.OLDEREC = $DA42 is getting set in a call to NEWSEGMENT @ DbgCnt = 18692.

'Dynamic: @D944=27: USERPROG.INIT\_KERNEL\_GLOBALS <- @D944=2: USERPROG.OS\_Init <- @D958=1: USERPROG.USERPROG'

* Getting set from Globals.LowMem.ERECp.: SCXG → Inner → NEWSEGMENT → ENVSAVE → NEWENV
* Look closely at NEWSEGMENT, ENVSAVE, NEWENV. The logic does not seem to be consistent with PME-debug.txt

3/7/2020:

Chronology

1. @ DbgCnt 18692 ('Dynamic: @D944=27: USERPROG.INIT\_KERNEL\_GLOBALS <- @D944=2: USERPROG.OS\_Init <- @D958=1: USERPROG.USERPROG')
   1. Globals.LowMem.ERECp gets set to $DA42 in step 13 of the bootstrap. This is initial value of the EREC located at $DA42:

'EREC @ $DA42 [55874]: Env\_Data: $0170 [ 368]; Env\_Vect: $DA3C [55868]; Env\_Sib: $DA4C [55884]; link\_count: 0; next\_rec: $0000 [ 0]'

* 1. This is the initial EVEC:

'EVEC @ $DA3C [55868]: Vect\_Length:2, Map[]=[ 1]=0000, [ 2]=DA42, '

* 1. This is the initial SIB:

'SIB @ $DA4C [55884]: Seg\_Pool:0000, SegBase:DA70, SegRefs:1, TimeStamp: 0, SegPieces:0, Residency:0, SegName: , SegLeng:3520, SegAddr: 323, VolInfo:0000, DataSize: 0, NextSIB:0000, PrevSIB:0000, MType:m\_pseudo'

* 1. The ENV field of to ROOTTASK gets set to $DA42 in step 14 of the bootstrap. It is set from Globals.LowMem.ERECp.
  2. RESTORE sets BP to $DA42 from the local variables (coming from the ENV field of the RootTask TIB- offset is 16 bytes from the TIB which is located at 340)
  3. OldErec gets set to $DA42 from ERECp in ENVSAVE
  4. SCXG calls INNER
  5. Inner calls NEWSEGMENT
  6. NEWSEGMENT calls ENVSAVE
  7. ENVSAVE is setting Globals.Lowmem.OLDEREC = $DA42.

1. Fault message gets a bad EREC from somewhere. Perhaps it is actually a pointer to something else.
2. FAULTHAN gets a bad EREC from fault message.
3. KERNEL.POOLSEG gets passed a bad EREC pointer from FAULTHAN
4. Low memory gets overwritten by “segpool := codepool;” in KERNEL.POOLSEG (line 305).
5. It is a call to LOADSEG @ DbgCnt = ~40626 which is smashing the EREC. This is, as before, the attempt to load FILEOPS starting @ $D170 thru $E2F0. Same as it has been for a long time.
6. SIB.Seg\_Pool is set to $1712. Is that right? Where does SEG\_POOL get set or used.
7. Getting a break at DbgCnt 20485 which is (I thought) where the Seg\_Pool (non-zero) first gets set? But it is already set in (at least) the SCREENOP SIB.
8. SCREENOP SIB Seg\_Pool has gotten set in before call to LOADSEG.
9. LOADSEG is called by FAULTHAN @ 57 (which calls POOLSEG).
10. SCREENOP SIB Seg\_Pool is getting set via the line “seg\_pool := codepool;” in POOL\_SEG.
11. WHAT SHOULD I BE USING SEG\_POOL FOR? It must be used for something!
12. The changes to POOLBASE seem to be leading nowhere. Getting lost again.

3/9/2020:

* Each of the SIBs (in POOLSEG, SCREENOPS-->OSUTIL) has a SEG\_POOL address of $1712
* Do the two systems load FILEOPS into different locations?
* Turbo Debugger Macros:
  + ^F9: Boot
  + ^F8: Display CURPROC, IPC, DbgCnt
  + ^F7: Set a break at entry to READSEG
  + ^F6: Position “Dump” window to the EREC by the local parameter referenced by $2C56
* Could the address passed to the IO routine include the pool offset?
* Prior to loading FILEOPS, both system list SEGPOOL @ $1712 AND SegBase @ $D170
* HP also overwrites the EREC @ $DA42.

|  |  |  |  |
| --- | --- | --- | --- |
|  | ERECAddr | SIBAddr |  |
| Delphi | $8A4 | $8AE |  |
| HP | $8A4 (BP) | $8AE (BP) |  |
| SegUnit | 4 | 12 |  |