

```
EEEEEEEEEEEEEEEEEE MMM      MMM      UUU      UUU      LLL      AAAAAAAAAA      TTTTTTTTTTTTTTTT
EEEEEEEEEEEEEEEEEE MMM      MMM      UUU      UUU      LLL      AAAAAAAAAA      TTTTTTTTTTTTTTTT
EEEEEEEEEEEEEEEEEE MMM      MMM      UUU      UUU      LLL      AAAAAAAAAA      TTTTTTTTTTTTTTTT
EEE      MMMMMM      MMMMMM      UUU      UUU      LLL      AAA      AAA      TTT
EEE      MMMMMM      MMMMMM      UUU      UUU      LLL      AAA      AAA      TTT
EEE      MMMMMM      MMMMMM      UUU      UUU      LLL      AAA      AAA      TTT
EEE      MMM      MMM      UUU      UUU      LLL      AAA      AAA      TTT
EEE      MMM      MMM      UUU      UUU      LLL      AAA      AAA      TTT
EEEEEEEEEEEEEEEEEE MMM      MMM      UUU      UUU      LLL      AAA      AAA      TTT
EEEEEEEEEEEEEEEEEE MMM      MMM      UUU      UUU      LLL      AAA      AAA      TTT
EEEEEEEEEEEEEEEEEE MMM      MMM      UUU      UUU      LLL      AAA      AAA      TTT
EEE      MMM      MMM      UUU      UUU      LLL      AAA      AAA      TTT
EEE      MMM      MMM      UUU      UUU      LLL      AAA      AAA      TTT
EEE      MMM      MMM      UUU      UUU      LLL      AAA      AAA      TTT
EEE      MMM      MMM      UUU      UUU      LLL      AAA      AAA      TTT
EEEEEEEEEEEEEEEEEE MMM      MMM      UUUUUUUUUUUUUUUUU LLLLLLLLLLLLLLLLLL AAA      AAA      TTT
EEEEEEEEEEEEEEEEEE MMM      MMM      UUUUUUUUUUUUUUUUU LLLLLLLLLLLLLLLLLL AAA      AAA      TTT
EEEEEEEEEEEEEEEEEE MMM      MMM      UUUUUUUUUUUUUUUUU LLLLLLLLLLLLLLLLLL AAA      AAA      TTT
```

[illegible]

(2)	73	Declarations
(3)	110	VAX\$CVTLP - Convert Long to Packed
(4)	322	DECIMAL_RORPRAND
(5)	356	CVTLP_ACCVIO - Reflect an Access Violation
(6)	406	Context-Specific Access Violation Handling for VAX\$CVTLP
(7)	539	CVTLP_RESTART - Unpack and Restart CVTLP Instruction


```
0000 1 .TITLE VAX$CVTLP - VAX-11 Instruction Emulator for CVTLP
0000 2 .IDENT /V04-000/
0000 3
0000 4
0000 5 *****
0000 6
0000 7 *
0000 8 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 9 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 10 * ALL RIGHTS RESERVED.
0000 11 *
0000 12 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 13 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 14 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 15 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 16 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 17 * TRANSFERRED.
0000 18 *
0000 19 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 20 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 21 * CORPORATION.
0000 22 *
0000 23 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 24 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 25 *
0000 26 *****
0000 27
0000 28
0000 29 ++
0000 30 Facility:
0000 31
0000 32 VAX-11 Instruction Emulator
0000 33
0000 34 Abstract:
0000 35 The routine in this module emulates the VAX-11 packed decimal
0000 36 CVTLP instruction. This procedure can be a part of an emulator
0000 37 package or can be called directly after the input parameters
0000 38 have been loaded into the architectural registers.
0000 39
0000 40 The input parameters to this routine are the registers that
0000 41 contain the intermediate instruction state.
0000 42
0000 43 Environment:
0000 44
0000 45 This routine runs at any access mode, at any IPL, and is AST
0000 46 reentrant.
0000 47
0000 48 Author:
0000 49
0000 50 Lawrence J. Kenah
0000 51
0000 52 Creation Date:
0000 53
0000 54 18 October 1983
0000 55
0000 56 Modified by:
0000 57
```


- VAX-11 Instruction Emulator for¹ CVTLP² 16-SEP-1984 01:32:35 VAX/VMS Macro V04-00 Page 2
7-SEP-1984 17:14:10 [EMULAT.SRC]VAXCVTLP.MAR;2 (1)

```

0000 58 : V01-004 LJK0040 Lawrence J. Kenah 24-Jul-1984
0000 59 : Do not use an INCL instruction to modify the contents of
0000 60 : the sign byte of the output string.
0000 61 :
0000 62 : V01-003 LJK0032 Lawrence J. Kenah 5-Jul-1984
0000 63 : Fix restart routine to take into account the fact that restart
0000 64 : codes are based at one when computing restart PC.
0000 65 :
0000 66 : V01-002 LJK0024 Lawrence J. Kenah 22-Feb-1984
0000 67 : Add code to handle access violations. Perform minor cleanup.
0000 68 :
0000 69 : V01-001 LJK0008 Lawrence J. Kenah 18-Oct-1983
0000 70 : The emulation code for CVTLP was moved into a separate module.
0000 71 : --

```

VA Sy ..
CV CV
CV CV
CV CV
CV CV
CV CV
CV CV
CV CV
CV CV
CV CV
CV CV
CV CV
CV CV
CV CV
CV CV
CV CV
CV CV
CV CV
DE DE
EX EX
HA HA
MO MO
PA PA
PC PC
PS PS
PS PS
PS PS
RE RE
TA TA
VA VA
VA VA
VA VA


```
0000 73      .SUBTITLE      Declarations
0000 74
0000 75      ; Include files:
0000 76
0000 77      .NOCROSS
0000 78      .ENABLE      SUPPRESSION      ; No cross reference for these
0000 79                                          ; No symbol table entries either
0000 80      CVTLP_DEF      ; Bit fields in CVTLP registers
0000 81      PACK_DEF      ; Stack usage by exception handler
0000 82      STACK_DEF      ; Stack usage for original exception
0000 83
0000 84      $PSLDEF      ; Define bit fields in PSL
0000 85
0000 86      .DISABLE      SUPPRESSION      ; Turn on symbol table again
0000 87      .CROSS      ; Cross reference is OK now
0000 88
0000 89      ; External declarations
0000 90
0000 91      .DISABLE      GLOBAL
0000 92
0000 93      .EXTERNAL -
0000 94      DECIMAL$BINARY_TO_PACKED_TABLE
0000 95
0000 96      .EXTERNAL -
0000 97      VAX$EXIT_EMULATOR,-
0000 98      VAX$REFLECT_FAULT,-
0000 99      VAX$ROPRAND,-
0000 100      VAX$DECIMAL_OVERFLOW
0000 101
0000 102      ; PSECT Declarations:
0000 103
0000 104      .DEFAULT      DISPLACEMENT , WORD
0000 105
00000000 106      .PSECT _VAX$CODE PIC, USR, CON, REL, LCL, SHR, EXE, RD, NOWRT, LONG
0000 107
0000 108      BEGIN_MARK_POINT      RESTART
```



```
0000 110 .SUBTITLE VAX$CVTLP - Convert Long to Packed
0000 111 ;+
0000 112 : Functional Description:
0000 113 :
0000 114 : The source operand is converted to a packed decimal string and the
0000 115 : destination string operand specified by the destination length and
0000 116 : destination address operands is replaced by the result.
0000 117 :
0000 118 : Input Parameters:
0000 119 :
0000 120 : R0 - src.rl Input longword to be converted
0000 121 : R2 - dstlen.rw Length of output decimal string
0000 122 : R3 - dstaddr.ab Address of output packed decimal string
0000 123 :
0000 124 : Output Parameters:
0000 125 :
0000 126 : R0 = 0
0000 127 : R1 = 0
0000 128 : R2 = 0
0000 129 : R3 = Address of byte containing most significant digit of
0000 130 : the destination string
0000 131 :
0000 132 : Condition Codes:
0000 133 :
0000 134 : N <- destination string LSS 0
0000 135 : Z <- destination string EQL 0
0000 136 : V <- decimal overflow
0000 137 : C <- 0
0000 138 :
0000 139 : Register Usage:
0000 140 :
0000 141 : This routine uses R0 through R5 and R11 as scratch registers. R10
0000 142 : serves its usual function as an access violation routine pointer. The
0000 143 : condition codes are stored in R11 as the routine executes.
0000 144 :
0000 145 : Notes:
0000 146 :
0000 147 : The algorithm used in this routine builds the packed decimal from
0000 148 : least significant digit to most significant digit. The least
0000 149 : significant digit is obtained by dividing the input longword by 10 and
0000 150 : storing the remainder as the least significant digit. The rest of the
0000 151 : result is obtained by taking the quotient from the first step,
0000 152 : repeatedly dividing by 100, and converting the resulting remainder
0000 153 : into a pair of packed decimal digits. This process continues until the
0000 154 : quotient goes to zero.
0000 155 :
0000 156 : No special processing is observed for an input longword of zero. The
0000 157 : correct results for this case drops out of normal processing.
0000 158 :-
0000 159 :
0000 160 .ENABLE LOCAL_BLOCK
0000 161
0000 162 ASSUME CVTLP_B_STATE EQ 7 ; Make sure we test the right FPD bit
0000 163
00142 31 0000 164 2$: BRW VAX$CVTLP_RESTART ; Restart somewhere else
0003 165
0003 166 VAX$CVTLP::
```

```
F9 51 1B E0 0003 167 BBS #<CVTLP_V_FPD+24>,R1,2$ ; Branch if this is a restart
OC30 8F BB 0007 168 PUSHF #^M<R4,R5,R10,R11> ; Save some registers
000B 169 ESTABLISH_HANDLER - ; Store address of access
000B 170 CVTLP_ACCVIO ; violation handler
0010 171
0010 172 ; Get initial settings for condition codes. The initial settings for V and C
0010 173 ; will be zero. The initial setting of N depends on the sign of the source
0010 174 ; operand. The Z-bit starts off set and remains set until a nonzero digit is
0010 175 ; stored in the output string. Note that the final Z-bit may be set for
0010 176 ; nonzero input if the output string is not large enough. (The V-bit is set
0010 177 ; in this case.) In this case, the saved DV bit will determine whether to
0010 178 ; reflect an exception or merely report the result to the caller.
0010 179
5B 04 00 5B DC 0010 180 MOVPSL R11 ; Get DV bit from PSL on input
04 04 FO 0012 181 INSV #PSL$M_Z,#0,#4,R11 ; Start with Z-bit set, others clear
51 52 FF 8F 78 0017 182 ROPRAND_CHECK R2 ; Insure that R2 LEQU 31
53 51 CO 0022 183 ASHL #-1,R2,R1 ; Convert digit count to byte count
63 0C 90 0027 184 ADDL R1,R3 ; Get address of sign byte
50 08 18 002A 185 MARK_POINT CVTLP_1 , RESTART
50 08 D5 002A 186 MOVB #12,(R3) ; Assume that sign is PLUS
08 18 002D 187 TSTL R0 ; Check sign of source operand
0031 188 BGEQ 10$ ; Start getting digits if not negative
0031 189
0031 190 ; Source operand is minus. We remember that by setting the saved N-bit but work
0031 191 ; with the absolute value of the input operand from this point on.
0031 192
0031 193 MARK_POINT CVTLP_2 , RESTART
50 63 96 0031 194 INCB (R3) ; Convert '+' to '-' (12 -> 13)
5B 50 CE 0033 195 MNEGL R0,R0 ; Normalize source operand
5B 08 88 0036 196 BISB #PSL$M_N,R11 ; Set N-bit in saved PSW
0039 197
0039 198 ;+
0039 199 ; The first (least significant) digit is obtained by dividing the source
0039 200 ; longword by ten and storing the remainder in the high order nibble of the
0039 201 ; sign byte. Note that at this point, the upper four bits of the sign byte
0039 202 ; contain zero.
0039 203 ;-
0039 204
51 D4 0039 205 10$: CLRL R1 ; Prepare R1 for input to EDIV
54 52 D0 003B 206 MOVL R2,R4 ; Special exit if zero source length
64 13 003E 207 BEQL 90$ ; Only overflow check remains
55 50 50 0A 7B 0040 208 EDIV #10,R0,R0,R5 ; R5 gets remainder, first digit
55 55 04 78 0045 209 ASHL #4,R5,R5 ; Shift digit to high nibble position
5B 04 8A 004B 210 BEQL 20$ ; Leave Z-bit alone if digit is zero
63 55 80 004E 211 BICB #PSL$M_Z,R11 ; Turn off Z-bit if nonzero
54 54 FF 8F 78 004E 212 MARK_POINT CVTLP_3 , RESTART
38 13 004E 213 ADDB R5,(R3) ; Merge this digit with low nibble
50 D5 0051 214 20$: DECL R4 ; One less output digit
04 12 0053 215 BEQL 90$ ; No more room in output string
73 94 0055 216 ASHL #-1,R4,R4 ; Number of complete bytes remaining
1D 11 005A 217 BEQL 80$ ; Check for last digit if none
0060 218 TSTL R0 ; Is source exhausted?
0062 219 BNEQ 30$ ; Go get next digits if not
0064 220 MARK_POINT CVTLP_4 , RESTART
0064 221 CLRB -(R3) ; Store a pair of zeros
0064 222 BRB 50$ ; Fill rest of output with zeros
0064 223
```



```
0064 224 :+
0064 225 : The following loop obtains two digits at a time from the source longword. It
0064 226 : accomplishes this by dividing the current value of R0 by 100 and converting
0064 227 : the remainder to a pair of decimal digits using the table that converts
0064 228 : binary numbers in the range from 0 to 99 to their packed decimal equivalents.
0064 229 : Note that this technique may cause nonzero to be stored in the upper nibble
0064 230 : of the most significant byte of an even length string. This condition will
0064 231 : be tested for at the end of the loop.
0064 232 :-
0064 233 :
55 50 50 00000064 8F 7B 0064 234 30$: EDIV #100,R0,R0,R5 ; R5 gets remainder, next digit
73 0000'CF45 90 006D 235 MARK_POINT CVTLP_5, RESTART
0073 236 MOVB- DECIMAL$BINARY_T0_PACKED TABLE[R5],-(R3)
0073 237 ; Store converted remainder
5B 03 13 0073 238 BEQL 40$ ; Leave Z-bit alone if digit is zero
0075 239 BICB #PSL$M_Z,R11 ; Turn off Z-bit if nonzero
50 04 8A 0078 240 40$: TSTL R0 ; Is source exhausted?
007A 241 BEQL 50$ ; Exit loop is no more source
E5 54 F5 007C 242 SOBGTR R4,30$ ; Check for end of loop
007F 243 ;
13 11 007F 244 BRB 80$ ; Check for remaining digit
0081 245 ;
0081 246 : The following code executes if the source longword is exhausted. If there
0081 247 : are any remaining digits in the destination string, they must be filled
0081 248 : with zeros. Note that one more byte is cleared if the original input length
0081 249 : was odd. This includes the most significant digit and the unused nibble.
0081 250 ;
02 52 E8 0081 251 50$: BLBS R2,65$ ; One less byte to zero if odd input length
0084 252 ;
0084 253 MARK_POINT CVTLP_6, RESTART
73 94 0084 254 60$: CLRB -(R3) ; Set a pair of digits to zero
FB 54 F5 0086 255 65$: SOBGTR R4,60$ ; Any more digits to zero?
0089 256 ;
0089 257 : The following code is the exit path for this routine. Note that all code
0089 258 : paths that arrive here do so with R0 containing zero. R1 and R2, however,
0089 259 : must be cleared on exit.
0089 260 ;
51 7C 0089 261 70$: CLRQ R1 ; Conform to architecture
OF B9 008B 262 BICPSW #<PSL$M_N!PSL$M_Z!PSL$M_V!PSL$M_C> ; Clear condition codes
5B B8 008D 263 BISPSW R11 ; Set appropriate condition codes
OC30 8F BA 008F 264 POPR #^M<R4,R5,R10,R11> ; Restore registers, preserving PSW
0093 265 RSB
0094 266 ;
0094 267 :+
0094 268 : The following code executes when there is no more room in the destination
0094 269 : string. We first test for the parity of the output length and, if even,
0094 270 : determine whether a nonzero digit was stored in the upper nibble of the
0094 271 : most significant byte. Such a nonzero store causes an overflow condition.
0094 272 :
0094 273 : If the source operand is not yet exhausted, then decimal overflow occurs.
0094 274 : If decimal overflow exceptions are enabled, an exception is signalled.
0094 275 : Otherwise, the V-bit in the PSW is set and a normal exit is issued. Note
0094 276 : that negative zero is only an issue for this instruction when overflow
0094 277 : occurs. In the no overflow case, the entire converted longword is stored in
0094 278 : the output string and there is only one form of binary zero.
0094 279 :-
0094 280
```

```
55 50 50 0D 52 E8 0094 281 80$: BLBS R2,90$ ; No last digit if odd output length
50 0A 7B 0097 282 EDIV #10,R0,R0,R5 ; Get next input digit
009C 283 MARK_POINT CVTLP_7 , RESTART
73 55 90 009C 284 MOV B R5,-(R3) ; Store in last output byte
03 03 13 009F 285 BEQL 90$ ; Leave Z-bit alone if zero
5B 04 8A 00A1 286 BICB #PSL$M_Z,R11
00A4 287
50 D5 00A4 288 90$: TSTL R0 ; Is source also all used up?
E1 13 00A6 289 BEQL 70$ ; Yes, continue with exit processing
00A8 290
00A8 291 ; An overflow has occurred. If the Z-bit is still set, then the N-bit is cleared.
00A8 292 ; Note that, because all negative zero situations occur simultaneously with
00A8 293 ; overflow, the output sign is left as minus.
00A8 294
03 5B 50 D4 00A8 295 100$: CLRL R0 ; R0 must be zero on exit
5B 02 E1 00AA 296 BBC #PSL$V_Z,R11,110$ ; Z-bit and N-bit cannot both be set
5B 08 8A 00AE 297 BICB #PSL$M_N,R11 ; Clear N-bit if Z-bit still set
5B 02 88 00B1 298 110$: BISB #PSL$M_V,R11 ; Set V-bit in saved PSW
00B4 299
00B4 300 ; If the V-bit is set and decimal traps are enabled (DV-bit is set), then
00B4 301 ; a decimal overflow trap is generated. Note that the DV-bit can be set in
00B4 302 ; the current PSL or, if this routine was entered as the result of an emulated
00B4 303 ; instruction exception, in the saved PSL on the stack.
00B4 304
10 5B 07 E0 00B4 305 BBS #PSL$V_DV,R11,120$ ; Report exception if current DV-bit set
54 0000 CF 9E 00B8 306 MOVAB VAX$EXIT_EMULATOR,R4 ; Set up R4 for PIC address comparison
10 AE 54 D1 00BD 307 CMPL R4,<4*4>(SP) ; Is return PC EQLU VAX$EXIT_EMULATOR ?
C6 12 00C1 308 BNEQU 70$ ; No. Simply return V-bit set
C1 40 AE 07 E1 00C3 309 BBC #PSL$V_DV,<<4*<4+1>>+EXCEPTION_PSL>(SP),70$ ; Only return V-bit if DV-bit is clear
00C8 310
00C8 311
00C8 312 ; Restore the saved registers and transfer control to DECIMAL_OVERFLOW
00C8 313
51 7C 00C8 314 120$: CLRQ R1 ; Conform to architecture
0F B9 00CA 315 BICPSW #<PSL$M_N!PSL$M_Z!PSL$M_V!PSL$M_C> ; Clear condition codes
5B B8 00CC 316 BISPSW R11 ; Set appropriate condition codes
0C30 8F BA 00CE 317 POPR #^M<R4,R5,R10,R11> ; Restore registers, preserving PSW
FF2B 31 00D2 318 BRW VAX$DECIMAL_OVERFLOW ; Report overflow exception
00D5 319
00D5 320 .DISABLE LOCAL_BLOCK
```



```
00D5 322 .SUBTITLE DECIMAL_ROPRAND
00D5 323 :-
00D5 324 : Functional Description:
00D5 325 :
00D5 326 : This routine receives control when a digit count larger than 31
00D5 327 : is detected. The exception is architecturally defined as an
00D5 328 : abort so there is no need to store intermediate state. The digit
00D5 329 : count is made after registers are saved. These registers must be
00D5 330 : restored before reporting the exception.
00D5 331 :
00D5 332 : Input Parameters:
00D5 333 :
00D5 334 : 00(SP) - Saved R4
00D5 335 : 04(SP) - Saved R5
00D5 336 : 08(SP) - Saved R10
00D5 337 : 12(SP) - Saved R11
00D5 338 : 16(SP) - Return PC from VAX$CVTLP routine
00D5 339 :
00D5 340 : Output Parameters:
00D5 341 :
00D5 342 : 00(SP) - Offset in packed register array to delta PC byte
00D5 343 : 04(SP) - Return PC from VAX$CVTLP routine
00D5 344 :
00D5 345 : Implicit Output:
00D5 346 :
00D5 347 : This routine passes control to VAX$ROPRAND where further
00D5 348 : exception processing takes place.
00D5 349 :-
00D5 350
00D5 351 DECIMAL_ROPRAND:
0C30 8F BA 00D5 352 POPR #M<R4,R5,R10,R11> ; Restore registers
00D5 353 PUSHL #CVTLP_B_DELTA_PC ; Store offset to delta PC byte
FF22' 31 00DB 354 BRW VAX$ROPRAND ; Pass control along
```



```
00DE 356 .SUBTITLE CVTLP_ACCVIO - Reflect an Access Violation
00DE 357 ;+
00DE 358 ; Functional Description:
00DE 359 ;
00DE 360 ; This routine receives control when an access violation occurs while
00DE 361 ; executing within the VAX$CVTLP emulator routine.
00DE 362 ;
00DE 363 ; The routine header for ASHP_ACCVIO in module VAX$ASHP contains a
00DE 364 ; detailed description of access violation handling for the decimal
00DE 365 ; string instructions. This routine differs from most decimal
00DE 366 ; instruction emulation routines in that it preserves intermediate
00DE 367 ; results if an access violation occurs. This is accomplished by
00DE 368 ; storing the number of the exception point, as well as intermediate
00DE 369 ; arithmetic results, in the registers R0 through R3.
00DE 370 ;
00DE 371 ; Input Parameters:
00DE 372 ;
00DE 373 ; See routine ASHP_ACCVIO in module VAX$ASHP
00DE 374 ;
00DE 375 ; Output Parameters:
00DE 376 ;
00DE 377 ; See routine ASHP_ACCVIO in module VAX$ASHP
00DE 378 ; -
00DE 379 ;
00DE 380 CVTLP_ACCVIO:
00DE 381 CLRL R2 ; Initialize the counter
FF1C 52 D4 00E0 382 PUSHAB MODULE_BASE ; Store base address of this module
51 8E C2 00E4 383 SUBL2 (SP)+,R1 ; Get PC relative to this base
00E7 384
0000'CF42 51 B1 00E7 385 10$: CMPW R1,PC_TABLE_BASE[R2] ; Is this the right PC?
07 13 00ED 386 BEQL 30$ ; Exit loop if true
F4 52 07 F2 00EF 387 AOBLS #TABLE_SIZE,R2,10$ ; Do the entire table
00F3 388
00F3 389 ; If we drop through the dispatching based on PC, then the exception is not
00F3 390 ; one that we want to back up. We simply reflect the exception to the user.
00F3 391
0F BA 00F3 392 20$: POPR #*M<R0,R1,R2,R3> ; Restore saved registers
05 05 00F5 393 RSB ; Return to exception dispatcher
00F6 394
00F6 395 ; The exception PC matched one of the entries in our PC table. R2 contains
00F6 396 ; the index into both the PC table and the handler table. R1 has served
00F6 397 ; its purpose and can be used as a scratch register.
00F6 398
51 0000'CF42 3C 00F6 399 30$: MOVZWL HANDLER_TABLE_BASE[R2],R1 ; Get the offset to the handler
FEFF CF41 17 00FC 400 JMP MODULE_BASE[R1] ; Pass control to the handler
0101 401
0101 402 ; In all of the instruction-specific routines, the state of the stack
0101 403 ; will be shown as it was when the exception occurred. All offsets will
0101 404 ; be pictured relative to R0.
```



```
0101 406 .SUBTITLE Context-Specific Access Violation Handling for VAX$CVTLP
0101 407 :+
0101 408 : Functional Description:
0101 409 :
0101 410 : The intermediate state of the instruction is packed into registers R0
0101 411 : through R3 and control is passed to VAX$REFLECT_FAULT that will, in
0101 412 : turn, reflect the access violation back to the user. The intermediate
0101 413 : state reflects the point at which the routine was executing when the
0101 414 : access violation occurred.
0101 415 :
0101 416 : Input Parameters:
0101 417 :
0101 418 : R0 - Address of top of stack when access violation occurred
0101 419 :
0101 420 : 00(R0) - Saved R4 on entry to VAX$CVTLP
0101 421 : 04(R0) - Saved R5
0101 422 : 08(R0) - Saved R10
0101 423 : 12(R0) - Saved R11
0101 424 : 16(R0) - Return PC from VAX$CVTLP routine
0101 425 :
0101 426 : 00(SP) - Saved R0 (restored by VAX$HANDLER)
0101 427 : 04(SP) - Saved R1
0101 428 : 08(SP) - Saved R2
0101 429 : 12(SP) - Saved R3
0101 430 :
0101 431 : Output Parameters:
0101 432 :
0101 433 : R0 - Address of return PC from VAX$CVTLP
0101 434 : R1 - Byte offset to delta-PC in saved register array
0101 435 : (PACK_V_FPD and PACK_M_ACCVIO set to identify exception)
0101 436 :
0101 437 : See list of input parameters for CVTLP_RESTART for a description of the
0101 438 : contents of the packed register array.
0101 439 :
0101 440 : Implicit Output:
0101 441 :
0101 442 : R4, R5, R10, and R11 are restored to the values that they had
0101 443 : when VAX$CVTLP was entered.
0101 444 : -
0101 445 :
0101 446 .ENABLE LOCAL_BLOCK
0101 447 :
0101 448 :+
0101 449 : CVTLP_1 or CVTLP_2
0101 450 :
0101 451 : An access violation occurred while storing the initial sign in the output
0101 452 : string. R1, R4, and R5 contain junk at this point.
0101 453 :
0101 454 : R0 - Input source longword
0101 455 : R2 - Digit count of destination string
0101 456 : R3 - Address of sign byte in destination string
0101 457 : R11 - Current PSW (with Z-bit set and all others clear)
0101 458 :
0101 459 : R1 - Not important
0101 460 : R4 - Scratch but saved anyway
0101 461 : R5 - Scratch but saved anyway
0101 462 : -
```



```
0101 463
0101 464 CVTLP_1:
90 0101 465      MOVB      #<CVTLP_1 RESTART!-      ; Store code that locates exception PC
07 AE 09 0102 466      CVTLP_M_FPD>,-
22 11 0102 467      CVTLP_B_STATÉ(SP)
0105 468      BRB      10$      ; Join common code
0107 469
0107 470 CVTLP_2:
90 0107 471      MOVB      #<CVTLP_2 RESTART!-      ; Store code that locates exception PC
0108 472      CVTLP_M_FPD>,-
07 AE 0A 0108 473      CVTLP_B_STATÉ(SP)
1C 11 0108 474      BRB      10$      ; Join common code
010D 475
010D 476 ;+
010D 477 ; CVTLP_3 through CVTLP_7
010D 478 ;
010D 479 ; An access violation occurred while storing a digit or digit pair in the
010D 480 ; output string.
010D 481 ;
010D 482 ; R0 - Input source longword (updated)
010D 483 ; R1 - Zero (so that R0/R1 can be used as input quadword to EDIV)
010D 484 ; R2 - Digit count of destination string
010D 485 ; R3 - Address of current byte in destination string
010D 486 ; R4 - Updated digit or byte count
010D 487 ; R5 - Most recent remainder from EDIV
010D 488 ; R11 - Current PSW (condition codes reflect results so far)
010D 489 ;-
010D 490
010D 491 CVTLP_3:
90 010D 492      MOVB      #<CVTLP_3 RESTART!-      ; Store code that locates exception PC
07 AE 0B 010E 493      CVTLP_M_FPD>,-
16 11 010E 494      CVTLP_B_STATÉ(SP)
0111 495      BRB      10$      ; Join common code
0113 496
0113 497 CVTLP_4:
90 0113 498      MOVB      #<CVTLP_4 RESTART!-      ; Store code that locates exception PC
0114 499      CVTLP_M_FPD>,-
07 AE 0C 0114 500      CVTLP_B_STATÉ(SP)
10 11 0117 501      BRB      10$      ; Join common code
0119 502
0119 503 CVTLP_5:
90 0119 504      MOVB      #<CVTLP_5 RESTART!-      ; Store code that locates exception PC
011A 505      CVTLP_M_FPD>,-
07 AE 0D 011A 506      CVTLP_B_STATÉ(SP)
OA 11 011D 507      BRB      10$      ; Join common code
011F 508
011F 509 CVTLP_6:
90 011F 510      MOVB      #<CVTLP_6 RESTART!-      ; Store code that locates exception PC
0120 511      CVTLP_M_FPD>,-
07 AE 0E 0120 512      CVTLP_B_STATÉ(SP)
04 11 0123 513      BRB      10$      ; Join common code
0125 514
0125 515 CVTLP_7:
90 0125 516      MOVB      #<CVTLP_7 RESTART!-      ; Store code that locates exception PC
0126 517      CVTLP_M_FPD>,-
07 AE 0F 0126 518      CVTLP_B_STATÉ(SP)
0129 519
```



```
04 AE 54 90 0129 520 10$: MOV B R4,CVTLP_B_SAVED_R4(SP) ; Store current digit/byte count
05 AE 55 90 012D 521      MOV B R5,CVTLP_B_SAVED_R5(SP) ; Store latest EDIV remainder
06 AE 5B 90 0131 522      MOV B R11,CVTLP_B_SAVED_PSW(SP) ; Store current condition codes
                    0135 523
                    0135 524 ; At this point, all intermediate state has been preserved in the register
                    0135 525 ; array on the stack. We now restore the registers that were saved on entry
                    0135 526 ; to VAX$CVTLP and pass control to VAX$REFLECT_FAULT where further exception
                    0135 527 ; dispatching takes place.
                    0135 528
                    54 80 7D 0135 529      MOVQ (R0)+,R4 ; Restore R4 and R5
                    5A 80 7D 0138 530      MOVQ (R0)+,R10 ; ... and R10 and R11
51 0000030B 8F D0 0138 531      MOV L #<CVTLP_B_DELTA_PC!- ; Indicate offset for delta PC
                    0142 533      PACK_M_FPD!- ; FPD bit should be set
                    0142 534      PACK_M_ACCVIO>,R1 ; This is an access violation
                    FE8B' 31 0142 535      BRW VAX$REFLECT_FAULT ; Continue exception handling
                    0145 536
                    0145 537      .DISABLE LOCAL_BLOCK
```



```
0145 539 .SUBTITLE CVTLP_RESTART - Unpack and Restart CVTLP Instruction
0145 540 :+
0145 541 : Functional Description:
0145 542 :
0145 543 : This routine receives control when a CVTLP instruction is restarted.
0145 544 : The instruction state (stack and general registers) is restored to the
0145 545 : state that it was in when the instruction (routine) was interrupted and
0145 546 : control is passed to the PC at which the exception occurred.
0145 547 :
0145 548 : Input Parameters:
0145 549 :
0145 550 : 31 23 15 07 00
0145 551 : +-----+-----+-----+-----+
0145 552 : | | | | | src | | | | | : R0
0145 553 : +-----+-----+-----+-----+
0145 554 : | state | saved_PSW | saved_R5 | saved_R4 | : R1
0145 555 : +-----+-----+-----+-----+
0145 556 : | delta-PC | XXXX | | dstlen | : R2
0145 557 : +-----+-----+-----+-----+
0145 558 : | | | | | dstaddr | | | | | : R3
0145 559 : +-----+-----+-----+-----+
0145 560 :
0145 561 : Depending on where the exception occurred, some of these parameters
0145 562 : may not be relevant. They are nevertheless stored as if they were
0145 563 : valid to make this restart code as simple as possible.
0145 564 :
0145 565 : R0 - Updated source longword
0145 566 : R1<07:00> - Latest digit or byte count (loaded into R4)
0145 567 : R1<15:08> - Most recent remainder from EDIV (loaded into R5)
0145 568 : R1<23:16> - Saved condition codes (loaded into R11)
0145 569 : R1<26:24> - Restart code (identifies point where routine will resume)
0145 570 : R1<27> - Internal FPD flag
0145 571 : R2<15:00> - Initial value of "dstlen"
0145 572 : R2<23:16> - spare
0145 573 : R2<31:24> - Size of instruction in instruction stream
0145 574 : R3 - Address of current byte in destination string
0145 575 :
0145 576 : 00(SP) - Return PC from VAX$CVTLP routine
0145 577 :
0145 578 : Output Parameters:
0145 579 :
0145 580 : R0 - Updated source longword (unchanged from input)
0145 581 : R1 - scratch
0145 582 : R2 - Initial value of "dstlen"
0145 583 : R3 - Address of current byte in output string (unchanged from input)
0145 584 : R4 - Latest digit or byte count
0145 585 : R5 - Most recent remainder from EDIV
0145 586 : R10 - Address of CVTLP_ACCVIO, this module's "condition handler"
0145 587 : R11 - Condition codes
0145 588 :
0145 589 : 00(SP) - Saved R4
0145 590 : 04(SP) - Saved R5
0145 591 : 08(SP) - Saved R10
0145 592 : 12(SP) - Saved R11
0145 593 : 16(SP) - Return PC from VAX$CVTLP routine
0145 594 :
0145 595 : Implicit Output:
```



```
0145 596 :  
0145 597 : Control is passed to the instruction that was executing when the  
0145 598 : access violation occurred.  
0145 599 :-  
0145 600  
0145 601 VAX$CVTLP_RESTART::  
0145 602 PUSH  #^M<R0,R1,R4,R5,R10,R11> ; Save some registers  
0149 603 ESTABLISH HANDLER CVTLP_ACCVIO ; Reload R10 with handler address  
014D 604 EXTZV #CVTLP_V_STATE,-  
014F 605 #CVTLP_S_STATE,-  
0150 606 CVTLP_B_STATE(SP),R1 ; Put restart code into R1  
0153 607 MOVZBL CVTLP_B_SAVED_R4(SP),R4 ; Restore digit/byte count  
0157 608 MOVZBL CVTLP_B_SAVED_R5(SP),R5 ; Restore latest EDIV remainder  
015B 609 MOVZBL CVTLP_B_SAVED_PSW(SP),R11 ; Restore condition codes  
015F 610 MOVZBL R2,R2 ; Clear out R2<31:8>  
0162 611 ADDL #8,SP ; Discard saved R0 and R1  
51 FFFE'CF41 3C 0165 612 MOVZWL RESTART_PC_TABLE_BASE-2[R1],R1 ; Convert code to PC offset  
016B 613  
016B 614 ; In order to get back to the restart point with R1 containing zero, we cannot  
016B 615 ; use R1 to transfer control as we did in other routines like VAX$CVTLP.  
016B 616  
FE90 CF41 9F 016B 617 PUSHAB MODULE_BASE[R1] ; Store "return PC"  
51 D4 0170 618 CLRL R1 ; Restart with R1 set to zero  
05 0172 619 RSB ; Get back to work  
0173 620  
0173 621 END_MARK_POINT CVTLP_M_STATE  
0173 622  
0173 623 .END
```


VAX\$CVTLP
Symbol table

- VAX-11 Instruction Emulator for CVTLP

16-SEP-1984 01:32:35
7-SEP-1984 17:14:10

VAX/VMS Macro V04-00
[EMULAT.SRC]VAXCVTLP.MAR;2

Page 15
(7)

...PC...	= 0000009C		
...RESTART_PC...	= 0000009C		
...ROPRAND...	= 0000001C	R	02
CVTLP_1	= 00000101	R	02
CVTLP_1_RESTART	= 00000001		
CVTLP_2	= 00000107	R	02
CVTLP_2_RESTART	= 00000002		
CVTLP_3	= 0000010D	R	02
CVTLP_3_RESTART	= 00000003		
CVTLP_4	= 00000113	R	02
CVTLP_4_RESTART	= 00000004		
CVTLP_5	= 00000119	R	02
CVTLP_5_RESTART	= 00000005		
CVTLP_6	= 0000011F	R	02
CVTLP_6_RESTART	= 00000006		
CVTLP_7	= 00000125	R	02
CVTLP_7_RESTART	= 00000007		
CVTLP_ACCVIO	= 000000DE	R	02
CVTLP_B_DELTA_PC	= 0000000B		
CVTLP_B_SAVED_PSW	= 00000006		
CVTLP_B_SAVED_R4	= 00000004		
CVTLP_B_SAVED_R5	= 00000005		
CVTLP_B_STATE	= 00000007		
CVTLP_M_FPD	= 00000008		
CVTLP_M_STATE	= 00000007		
CVTLP_S_STATE	= 00000003		
CVTLP_V_FPD	= 00000003		
CVTLP_V_STATE	= 00000000		
DECIMAL\$BINARY_TO_PACKED_TABLE	*****	X	00
DECIMAL_ROPRAND	= 000000D5	R	02
EXCEPTION_PSL	= 0000002C		
HANDLER_TABLE_BASE	= 00000000	R	04
MODULE_BASE	= 00000000	R	02
MODULE_END	= 00000173	R	02
PACK_M_ACCVIO	= 00000200		
PACK_M_FPD	= 00000100		
PC_TABLE_BASE	= 00000000	R	03
PSL\$M_C	= 00000001		
PSL\$M_N	= 00000008		
PSL\$M_V	= 00000002		
PSL\$M_Z	= 00000004		
PSL\$V_DV	= 00000007		
PSL\$V_Z	= 00000002		
RESTART_PC_TABLE_BASE	= 00000000	R	05
RESTART_TABLE_SIZE	= 00000007		
TABLE_SIZE	= 00000007		
VAX\$CVTLP	= 00000003	RG	02
VAX\$CVTLP_RESTART	= 00000145	RG	02
VAX\$DECIMAL_OVERFLOW	*****	X	00
VAX\$EXIT_EMULATOR	*****	X	00
VAX\$REFLECT_FAULT	*****	X	00
VAX\$ROPRAND	*****	X	00

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
VAX\$CODE	00000173 (371.)	02 (2.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG
PC TABLE	0000000E (14.)	03 (3.)	PIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC BYTE
HANDLER_TABLE	0000000E (14.)	04 (4.)	PIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC BYTE
RESTART_PC_TABLE	0000000E (14.)	05 (5.)	PIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC BYTE

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	11	00:00:00.05	00:00:01.82
Command processing	73	00:00:00.53	00:00:05.30
Pass 1	123	00:00:02.91	00:00:13.17
Symbol table sort	0	00:00:00.11	00:00:00.11
Pass 2	113	00:00:01.28	00:00:04.49
Symbol table output	7	00:00:00.06	00:00:00.06
Psect synopsis output	2	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	329	00:00:04.97	00:00:24.99

The working set limit was 1050 pages.
14994 bytes (30 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 121 non-local and 19 local symbols.
623 source lines were read in Pass 1, producing 20 object records in Pass 2.
19 pages of virtual memory were used to define 17 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
-\$255\$DUA28:[EMULAT.OBJ]VAXMACROS.MLB;1	9
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2	5
TOTALS (all libraries)	14

278 GETS were required to define 14 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:VAXCVTLP/OBJ=OBJ\$:VAXCVTLP MSRC\$:VAXCVTLP/UPDATE=(ENH\$:VAXCVTLP)+LIB\$:VAXMACROS/LIB

0144 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

