INTERNAL CORRESPONDENCE

To: LIST
FROM: Greg Blanck SC4-59 x6-2352
DATE: 1/8/86
SUBJECT: Returning to real mode on the 80386
LIST:

This memo describes the procedure necessary to return the 80386 from operation in protected mode, back to operating in real mode.

Details:

Their are several functions that the routine needs to perform, namely:

- o Resetting internal segment descriptors to look like real mode
- o Re-Entering real mode
- o Jumping to the real mode code to be executed

The first task requires resetting the segment descriptors to appear like real mode. It is useful to understand why this is necessary; While operating in REAL mode, the 80386 uses exactly the same memory management functions as in protected mode. However, when the part resets into real mode the values loaded into the descriptors make the segments *appear* as if they were 8086 style segments. In real mode, when a segment register is loaded, only the *base* field is changed, in particular the value placed into the base is *selector*16*. Since only the base is changed, it is necessary to set the access rights while still in protected mode. This is done by loading a descriptor with the appropriate attributes. The value loaded into each of the SS, DS, ES, FS, GS descriptors is:

Limit=64K (FFFFH)
Byte Granular (G=0)
Expand Up (E=0)
Writeable (W=1)
Present (P=1)
Base=Don't Care

The base is a don't care since it will be set again after re-entering real mode. These attributes make segments act exactly as in real mode. Two additional segment must be reset- the IDT (Interrupt Descriptor Table), and the CS (Code Segment). Since the LIDT instruction also works in real mode, it will be reset there. The CS is a special case: since it is not possible to make the CS writeable while in protected mode, an architectural feature reloads real mode attributes into the CS descriptor during real mode far jumps.

Re-entering real mode is quite easy to do, since the PE (protection enabled) bit in CR0 (MSW) is not sticky (as it was on the 286). The same rules apply as when switching into protected mode; the internal instruction FIFO must be cleared by performing a jump instruction. The transition becomes more difficult it PAGING has been enabled- this will be discussed later.

Finally, once in real mode, the IDT must be loaded with an LIDT instruction, and the CS access rights get set when a far jump is executed (the same jump used to get to the code which is to be executed). Also note, that for system integrity, the possibility of interrupts while changing modes must be removed. Normal INTR interrupts can be inhibited by a CLI instruction. Non-Maskable interrupts however can not be disabled. To prevent these, either external circuitry can be used to mask out NMI's, or the protected mode Interrupt Descriptor Table which is used in during the mode transition must contain a REAL mode style vector at offset 8 in the IDT.

The code

The following code fragment performs the functions described above. It was written for the ASM/P3 assembler, and may require some changes for DSO's ASM/386.

The code assumes the following: Use16, Level 0, CS Limit=64K. The limit of 64K is necessary to reset the CS limit back to the real mode value. It is the only way to get the limit back to its required value. A complete program which starts at power-up reset, switches into protected mode, and then back to real mode is attached to this memo. The following is a code fragment that does the switch back to real mode.

```
Start segment
            MOV         AX, RealModeSel
            MOV         DS, AX
            MOV         ES, AX
            MOV         SS, AX
            MOV         FS, AX
            MOV         GS, AX
            MOV         EAX, CR0
            AND         EAX, 07FFFFFFEH
            CLI
            MOV         CR0, EAX
            JMP         FlushQueueReal
FlushQueueReal:
            ; Now in real mode again
            Assume ds: idtloadsegment
            MOV         AX, idtloadsegment
            MOV         DS, AX
            LIDT        idt
            STI
            ; Load up segments as needed by target code
            MOV         AX, 1000H; use 1000H as an example
            MOV         DS, AX
            MOV         ES, AX
            MOV         SS, AX
            MOV         FS, AX
            MOV         GS, AX; All Segments now REAL
                        ; mode, start at 10000H
            JMP         far ptr RealModeReturn; Back to targed program.

            org         10000H
end segment
```

Paging considerations

If paging was enabled, special care must be taken when re-entering real mode, and in general when paging is turned off. Prior to turning off paging, the 80386 must be running with an *identity* page map (i.e. linear=physical address). This is required so that when paging is turned off, execution will continue at the same point. After turning off paging, CR3 should be

```
                           ;*****************************************
                           ; Program to test returning to real mode
                           ;    1-7-86 Greg Blanck
                           ;*****************************************
                           NullSel        EQU     0
                           dataseg_SEL    EQU     8
                           maincode       EQU     16
                           RealModeSel    EQU     24
00000000 ----              _Tables             SEGMENT
00000002 --------          MainTable              DW        ?
                                          DD      ?
                           _Tables             ENDS
00000010 78563412          _dataseg_SEG           SEGMENT
                           Data1      DD   12345678H
                           _dataseg_SEGEND:
                           _dataseg_SEG ENDS
00000020 0000000000000000  ldtloadsegment segment
                           ldt     dw 0, 0, 0, 0
                           ldtloadsegment ends
                           ReEntry  SEGMENT  use16
                           RealModeReturn:
                                 ; Try writing to CS, self modify at address modloc (100H)
                                 ; There was a halt there before, place a NOP there.
00000030 2EC606000190            MOV   CS: byte ptr [100H], 90H
00000036 E9----                  JMP   ModLoc
                                 org 100H
                           ModLoc:
00000130 F4                      HLT        ; Should get turned into a NOP
                                 ; That's it!
00000131 90                      NOP
00000132 90                      NOP
00000133 F4                      HLT
                           ReEntry  ENDS
                           _maincode_SEG          SEGMENT   use16
                                 ; We are in protected mode, load up a few segments
                                 ; then return to real at address RealModeReturn
00000140 B80800                  MOV      AX, dataseg_SEL
00000143 8ED8                    MOV      DS, AX
00000145 8EC0                    MOV      ES, AX
00000147 8ED0                    MOV      SS, AX
00000149 8EE0                    MOV      FS, AX
0000014B 8EE8                    MOV      GS, AX
                                 ; Now reset descriptors to real mode
0000014D B81800                  MOV      AX, RealModeSel
00000150 8ED8                    MOV      DS, AX
00000152 8EC0                    MOV      ES, AX
00000154 8ED0                    MOV      SS, AX
00000156 8EE0                    MOV      FS, AX
00000158 8EE8                    MOV      GS, AX
0000015A 660F20C0                MOV      EAX, CR0
0000015E 6625FEFFFF7F            AND      EAX, 07FFFFFFEH      ; No PG no PE
0000015F FA                      CLI
00000165 660F22C0                MOV      CR0, EAX
00000169 E9----                  JMP      FlushQueueReal
                           FlushQueueReal:
                                 ; Now we're in real mode again.
                                 assume ds: ldtloadsegment
0000016C B80200                  MOV      AX, ldtloadsegment
0000016F 8ED8                    MOV      DS, AX
00000171 0F011E0000              LIDT     ldt
```

reloaded to clear out the TLB. The sequence:

```
MOV     EAX, CR0
AND     EAX, 7FFFFFFFH
MOV     CR0, EAX
XOR     EAX, EAX
MOV     CR3, EAX
.
.
```

will turn off paging and clear out the TLB. This should be executed PRIOR to re-entering REAL mode.

```
00000172 FB                        STI
                                   ; Load up segment bases with desired values
00000177 B80010                    MOV     AX, 1000H     ; Put all bases at 1000H
0000017A 8ED8                      MOV     DS, AX
0000017C 8EC0                      MOV     ES, AX
0000017E 8ED0                      MOV     SS, AX
00000180 8EE0                      MOV     FS, AX
00000182 8EE8                      MOV     GS, AX
00000184 EA00000300          Loc1: JMP     far ptr realmodereturn
                             org  10000H
                             _maincode_SEGEND:
                             _maincode_SEG           ENDS
                             Code Segment use16   ;0 at 9000H
                             Startup:
                                   ; Enter protected mode-- jump of to maincode
                                   assume ds: _tables
00009000 B80000                    MOV     AX, _tables
00009003 8ED8                      MOV     DS, AX
00009005 0F01160000                LGDT    MainTable
0000900A 66B801000000              MOV     EAX, 1
00009010 660F22C0                  MOV     CR0, EAX ; Turn on protection
00009014 E9----                    JMP     FlushQueue
                             FlushQueue:
00009017 EA00001000                JMP     0, Maincode
                             Code ENDS
                             _MainTable_GDT                  Segment      RO
00009020 00000000            _NullSel_DESCR     DD   0
00009024 00000000                                    DD        0
00009028 03                  _dataseg_SEL_DESCR             DB      ((  ( offset _dataseg_SEGEND - 1 )   ) _AND 0FFH)
00009029 00                                     DB     ((  ( offset _dataseg_SEGEND - 1 )   ) _AND 0FF00H ) _SHR 8
0000902A 10                                     DB     ( _linaddr _dataseg_SEG _AND 00FFH)
0000902B 00                                     DB     ( _linaddr _dataseg_SEG _AND 0FF00H) _SHR 8
0000902C 0C                                     DB     ( _linaddr _dataseg_SEG _AND 00FF0000H ) _SHR 16
0000902D 92                                     DB    146 ;Data Descriptor
0000902E 00                                     DB    0 _OR ((( (offset _dataseg_SEGEND - 1 )  ) _AND 0F0000H) _SHR 16 )
0000902F 00                                     DB     ( _linaddr _dataseg_SEG _AND 0FF000000H ) _SHR 24
00009030 FF                  _maincode_DESCR                 DB      ((  ( offset _maincode_SEGEND - 1 )   ) _AND 0FFH)
00009031 FF                                     DB     ((  ( offset _maincode_SEGEND - 1 )   ) _AND 0FF00H ) _SHR 8
00009032 40                                     DB     ( _linaddr _maincode_SEG _AND 00FFH)
00009033 01                                     DB     ( _linaddr _maincode_SEG _AND 0FF00H) _SHR 8
00009034 00                                     DB     ( _linaddr _maincode_SEG _AND 00FF0000H ) _SHR 16
00009035 98                                     DB    152 ;Code Descriptor
00009036 00                                     DB    0 _OR ((( (offset _maincode_SEGEND - 1 )  ) _AND 0F0000H) _SHR 16 )
00009037 00                                     DB     ( _linaddr _maincode_SEG _AND 0FF000000H ) _SHR 24
00009038 FFFF0000            _RealModeSel_DESCR   DD        0FFFFH
0000903C 00920000                                 DD        9200H
                             _MainTable_GDT_END:
                             _MainTable_GDT                 ENDS
00000000 1F00                _DummyFixup  Segment           ;0 at _linaddr MainTable
00000002 20900000            _MainTable_FIX                DW _linaddr _MainTable_GDT_END -1 - _linaddr _MainTable_GDT
                                          DD        _linaddr _MainTable_GDT
                             _DummyFixup  ENDS
                             RESET      Segment ;0 at 0FFFF0000H
                                          org 0FFF0H
FFFFFFF0 6631C0                           XOR   EAX, EAX
FFFFFFF3 6631D2                           XOR   EDX, EDX
FFFFFFF6 EA00000009                       JMP   far ptr startup
** WARNING (sem) ** <m-addr> 'STARTUP' may be inaccessable with current ASSUMEs.
                             RESET      ENDS
                                                END
```