Enabling the A20 Line

Okay, this isn't exactly a tutorial. What it is a *very* well commented example of enabling A20 in assembly.

```
;;
;; enableA20.s (adapted from Visopsys OS-loader)
;;
;; Copyright (c) 2000, J. Andrew McLaughlin
;; You're free to use this code in any manner you like, as long as this
;; notice is included (and you give credit where it is due), and as long
;; as you understand and accept that it comes with NO WARRANTY OF ANY KIND.
;; Contact me at jamesamc@yahoo.com about any bugs or problems.
;;

enableA20:
        ;; This subroutine will enable the A20 address line in the keyboard
        ;; controller.  Takes no arguments.  Returns 0 in EAX on success,
        ;; -1 on failure.  Written for use in 16-bit code, see lines marked
        ;; with 32-BIT for use in 32-bit code.

        pusha

        ;; Make sure interrupts are disabled
        cli

        ;; Keep a counter so that we can make up to 5 attempts to turn
        ;; on A20 if necessary
        mov CX, 5

        .startAttempt1:
        ;; Wait for the controller to be ready for a command
        .commandWait1:
        xor AX, AX
        in AL, 64h
        bt AX, 1
        jc .commandWait1

        ;; Tell the controller we want to read the current status.
        ;; Send the command D0h: read output port.
        mov AL, 0D0h
        out 64h, AL

        ;; Wait for the controller to be ready with a byte of data
        .dataWait1:
        xor AX, AX
        in AL, 64h
        bt AX, 0
        jnc .dataWait1

        ;; Read the current port status from port 60h
        xor AX, AX
        in AL, 60h

        ;; Save the current value of (E)AX
        push AX                 ; 16-BIT
        ;; push EAX             ; 32-BIT

        ;; Wait for the controller to be ready for a command
        .commandWait2:
        in AL, 64h
        bt AX, 1
        jc .commandWait2

        ;; Tell the controller we want to write the status byte again
        mov AL, 0D1h
        out 64h, AL
```

```
;; Wait for the controller to be ready for the data
.commandWait3:
xor AX, AX
in AL, 64h
bt AX, 1
jc .commandWait3

;; Write the new value to port 60h.  Remember we saved the old
;; value on the stack
pop AX                  ; 16-BIT
;; pop EAX              ; 32-BIT

;; Turn on the A20 enable bit
or AL, 00000010b
out 60h, AL

;; Finally, we will attempt to read back the A20 status
;; to ensure it was enabled.

;; Wait for the controller to be ready for a command
.commandWait4:
xor AX, AX
in AL, 64h
bt AX, 1
jc .commandWait4

;; Send the command D0h: read output port.
mov AL, 0D0h
out 64h, AL

;; Wait for the controller to be ready with a byte of data
.dataWait2:
xor AX, AX
in AL, 64h
bt AX, 0
jnc .dataWait2

;; Read the current port status from port 60h
xor AX, AX
in AL, 60h

;; Is A20 enabled?
bt AX, 1

;; Check the result.  If carry is on, A20 is on.
jc .success

;; Should we retry the operation?  If the counter value in ECX
;; has not reached zero, we will retry
loop .startAttempt1


;; Well, our initial attempt to set A20 has failed.  Now we will
;; try a backup method (which is supposedly not supported on many
;; chipsets, but which seems to be the only method that works on
;; other chipsets).


;; Keep a counter so that we can make up to 5 attempts to turn
;; on A20 if necessary
mov CX, 5

.startAttempt2:
;; Wait for the keyboard to be ready for another command
.commandWait6:
xor AX, AX
in AL, 64h
bt AX, 1
```

```
jc .commandWait6

;; Tell the controller we want to turn on A20
mov AL, 0DFh
out 64h, AL

;; Again, we will attempt to read back the A20 status
;; to ensure it was enabled.

;; Wait for the controller to be ready for a command
.commandWait7:
xor AX, AX
in AL, 64h
bt AX, 1
jc .commandWait7

;; Send the command D0h: read output port.
mov AL, 0D0h
out 64h, AL

;; Wait for the controller to be ready with a byte of data
.dataWait3:
xor AX, AX
in AL, 64h
bt AX, 0
jnc .dataWait3

;; Read the current port status from port 60h
xor AX, AX
in AL, 60h

;; Is A20 enabled?
bt AX, 1

;; Check the result.  If carry is on, A20 is on, but we might warn
;; that we had to use this alternate method
jc .warn

;; Should we retry the operation?  If the counter value in ECX
;; has not reached zero, we will retry
loop .startAttempt2


;; OK, we weren't able to set the A20 address line.  Do you want
;; to put an error message here?
jmp .fail


.warn:
;; Here you may or may not want to print a warning message about
;; the fact that we had to use the nonstandard alternate enabling
;; method

.success:
sti
popa
xor EAX, EAX
ret

.fail:
sti
popa
mov EAX, -1
ret
```

Download enableA20.s