**Tutorial 8**

En este tutorial se habilito la linea A20 del Address Bus, para asi poder pasar de el limite de 1MB que tenian los sistemas computacionales anteriores. Esto nos permite acceder en teoria hasta 4GB de RAM. Para poder realizar esta tarea es necesario habilitarlo programando directamente un controlador del sistema y se puede realizar de varias formas. En los primeros sistemas que pudieron acceder a mas de 1MB de memoria esto se hacia directamente programando un bit en el controlador del teclado Intel 8042. Existen 4 metodos : Utilizando un puerto de sistema generalmente mapeado en la direccion 0x92, utilizando la BIOS y unos interruptores, y utilizando el controlador de teclado. De cualquier forma en que se realice es muy probable que funcione, sin embargo, existen diferencias entre los sistemas asi que no todas estas formas estan totalmente soportadas o son formas estandarizadas de realizarlo en la vida real.

Usando el puerto de sistema de esta forma:

ov al, 2 ; set bit 2 (enable a20)

out 0x92, al

Se puede habilitar la linea A20 para acceder a mas de 1MB de RAM. Sin embargo, no todos los sitemas soportan usar el puerto 0x92 porque puede ser que la BIOS mapea este puerto de sistema a otro numero, por lo cual este metodo no es tan confiable.

Usando interruptores de esta forma:

mov ax, 0x2401 (Funcion 2401)

int 0x15

Se puede habilitar la linea A20 para acceder a mas de 1MB de RAM. Para efectos de practica se puede utilizar este metodo y muchas BIOS la soportan. Sin embargo, en nuestro caso, hay unas versiones de BOCHS que no aceptan el uso de este interruptor y no nos funcionaria.

El metodo mas fiable y utilizado es el de usar el controlador de teclado para poder habilitar la linea A20 del Address Bus. El mapeado de puertos es el siguiente:

|  |  |  |
| --- | --- | --- |
| Port Mapping | | |
| Port | Read/Write | Descripton |
| 0x60 | Read | Read Input Buffer |
| 0x60 | Write | Write Output Buffer |
| 0x64 | Read | Read Status Register |
| 0x64 | Write | Send Command to controller |

Los comandos que acepta son :

|  |  |
| --- | --- |
| Keyboard Controller Commands | |
| Keyboard Command | Descripton |
| 0x20 | Read Keyboard Controller Command Byte |
| 0x60 | Write Keyboard Controller Command Byte |
| 0xAA | Self Test |
| 0xAB | Interface Test |
| 0xAD | Disable Keyboard |
| 0xAE | Enable Keyboard |
| 0xC0 | Read Input Port |
| 0xD0 | Read Output Port |
| 0xD1 | Write Output Port |
| 0xDD | Enable A20 Address Line |
| 0xDF | Disable A20 Address Line |
| 0xE0 | Read Test Inputs |
| 0xFE | System Reset |
| Mouse Command | Descripton |
| 0xA7 | Disable Mouse Port |
| 0xA8 | Enable Mouse Port |
| 0xA9 | Test Mouse Port |
| 0xD4 | Write to mouse |

Si nos fijamos muy bien, el comando 0xDD habilita la linea A20 que ocupamos para acceder a mas de 1MB de memoria. Para habilitar la linea A20 utilizamos el siguiente codigo:

; send read output port command

mov al,0xD0

out 0x64,al

call wait\_output

; read input buffer and store on stack. This is the data read from the output port

in al,0x60

push eax

call wait\_input

; send write output port command

mov al,0xD1

out 0x64,al

call wait\_input

; pop the output port data from stack and set bit 1 (A20) to enable

pop eax

or al,2 // 2 = 10 binary s

out 0x60,al // write the data to the output port. This is done through the output buffer