

Course Name: Operating System

Lecturer: Dr. Sahar A. AL-Talib

----- O. S. Protection -----

• Hardware Protection

A properly designed operating system must ensure that an incorrect (or malicious) program cannot cause other programs to execute incorrectly. Many programming errors are detected by the hardware. These errors are normally handled by the operating system. If a user program fails in some way- such as :

- Execute an illegal instruction, or
- Access memory that is not in the user's address space.

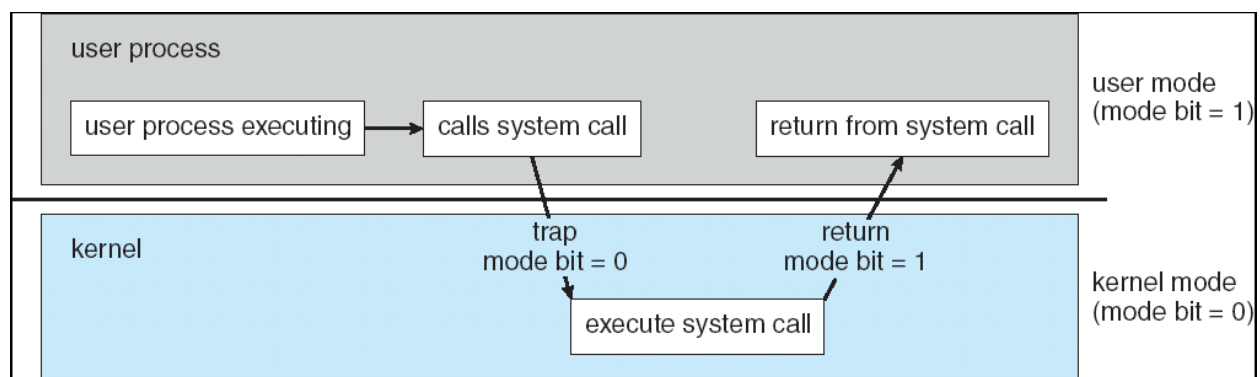
Then the HW will trap to the O.S.

Whenever a program error occurs, the O.S. must abnormally terminate the program. An appropriate error message is given, and the memory of the program is dumped.

• Dual Mode Operation

To protect the O.S., two modes of operation are needed: user mode and monitor mode. A bit, called the mode bit is added to the HW of the computer to indicate the current mode: monitor (0) or user (1).

At system boot time, the HW starts in monitor mode. The O.S. is then loaded, and starts user process in user mode. Whenever a trap or interrupt occurs, the HW switches from user mode to monitor mode. Microsoft Windows NT, and IBM OS/2 use this feature.



- **I/O Protection**

To prevent a user from performing illegal I/O, define all I/O instructions to be privileged instructions. Thus, users cannot issue I/O instructions directly; they must do it through the operating system.

- **Memory Protection**

- Must provide memory protection at least for the interrupt vector and the interrupt service routines.
- To provide memory protection; add two registers that determine the range of legal addresses a program may address.
- Base Register - holds smallest legal physical memory address.
- Limit register - contains the size of the range.
- Memory outside the defined range is protected.
- When executing in monitor mode, the OS has unrestricted access to both monitor and users' memory.

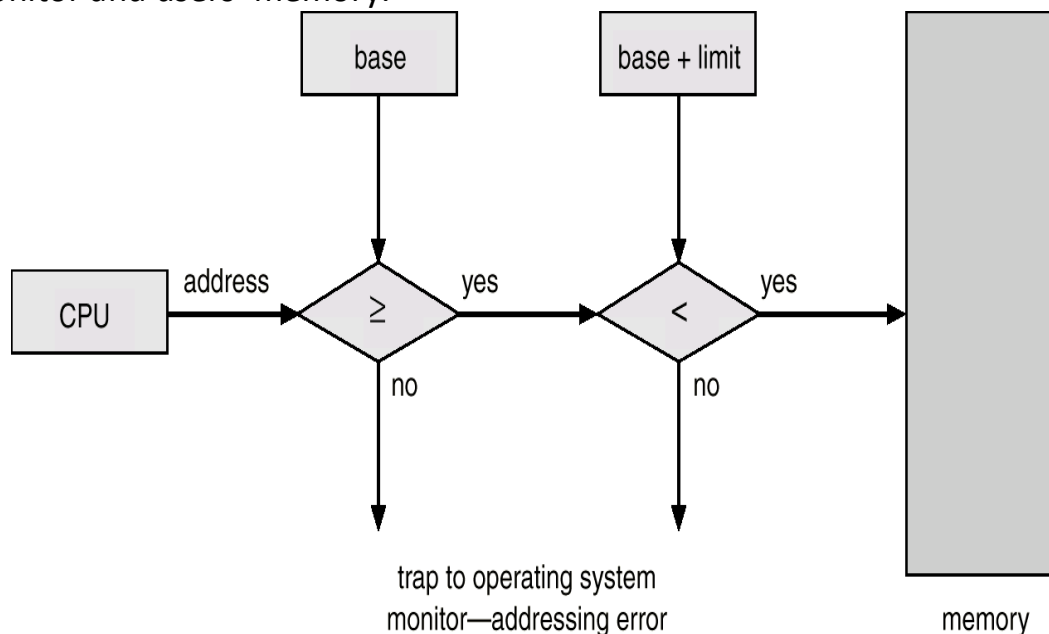


Figure: Hardware address protection with base and limit registers

- **CPU protection**

A clock prevents programs from using all the CPU time. This clock causes an interrupt that causes the operating system to gain control from a user program.