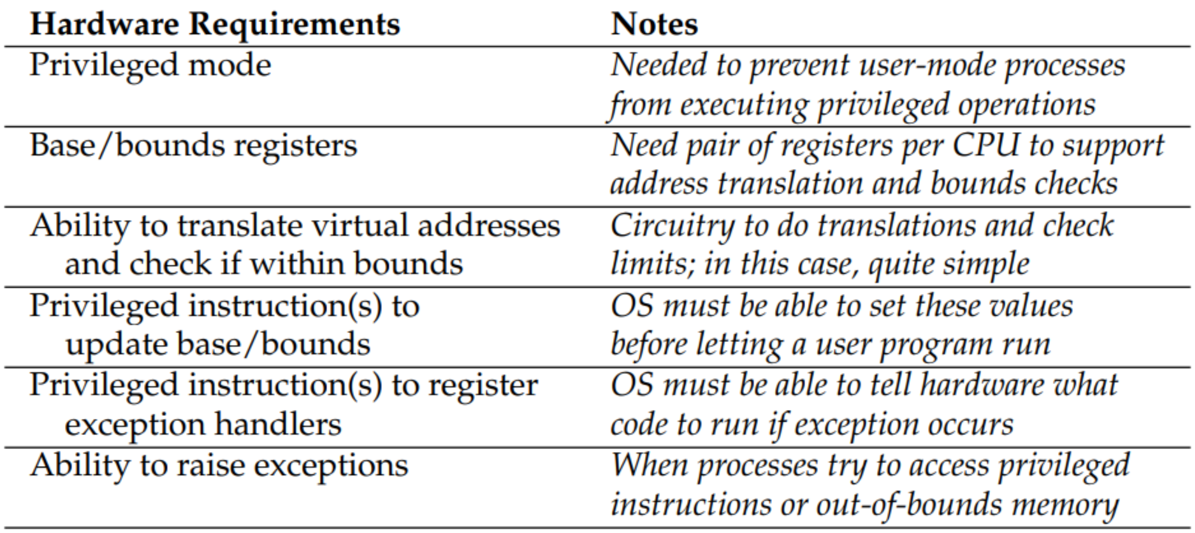
【OS】Day18

【Ch1

【Ch15】Address Translation(2)

15.4 Hardware Support: A Summary



Hardware Requirements	Notes
Privileged mode	*Needed to prevent user-mode processes from executing privileged operations*
Base/bounds registers	*Need pair of registers per CPU to support address translation and bounds checks*
Ability to translate virtual addresses and check if within bounds	*Circuitry to do translations and check limits; in this case, quite simple*
Privileged instruction(s) to update base/bounds	*OS must be able to set these values before letting a user program run*
Privileged instruction(s) to register exception handlers	*OS must be able to tell hardware what code to run if exception occurs*
Ability to raise exceptions	*When processes try to access privileged instructions or out-of-bounds memory*

Figure 15.3: **Dynamic Relocation: Hardware Requirements**

15.5 Operating System Issues

There are a few criticial junctures where the OS must get involved to implement our base-and-bounds version of virtual memory.

First, the OS must take action when a process is created, finding space for its address space in memory.

When a new process is created, the OS will have to search a data structure(often called a free list) to find room for the new address space and then mark it used.

Let’s see an example:

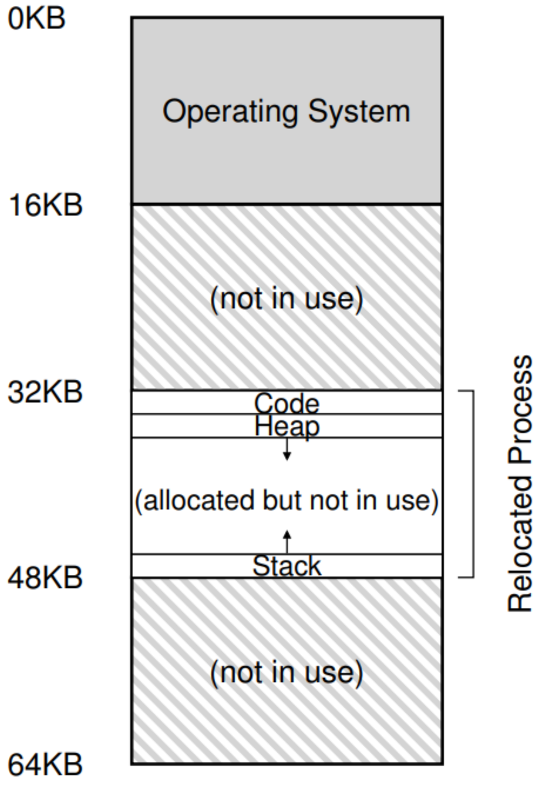


Figure 15.2: **Physical Memory with a Single Relocated Process**

A process is created at 32KB. The other two slots are free(16KB-32KB and 48KB-64KB); thus, the free list should consist of these two entries.

Second, the OS must do some work when a process is terminated.(i.e. when it exits gracefully or is forcefully killed because it misbehaved)

The OS has to reclaim all of its memory for use in other processes or the OS. Upon termination of a process, the OS thus puts its memory back on the free list, and cleans up any associated data structures as need be.

Third, the OS must also perform a few additional steps when a context switch occurs.

There is only one base and boudns register pair on each CPU. Thus, the OS must save and restore the base-and-bounds pair when it switches between processes, in some per-process structure such as the process structurer or process control block(PCB).

Similarly, when the OS resumes a running process, it must set the values of the base and bounds on the CPU to the correct values for this process.

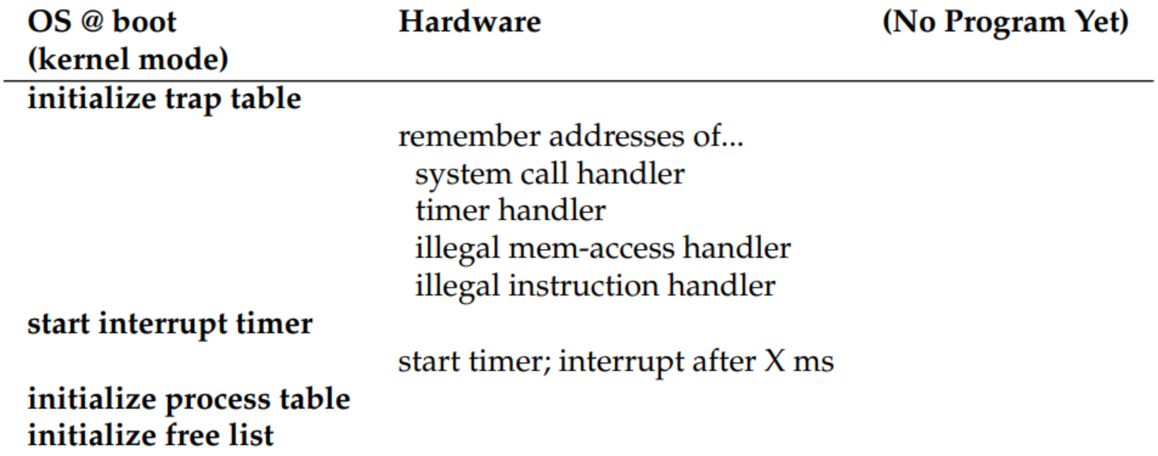
We should note that when a process is stopped(i.e. not running), it is possible for the OS to move an address space from one location in memory to another rather easily.

1. To move a process's address space, the OS first deschedules the process
2. Then, the OS copies the address space from the current location to the new location
3. Finally, the OS updates the saved base register to point to the new location.

When the process is resumed, its (new) base register is restored and it begins running again.

Fourth, the OS must provide exception handlers, or functions to be called. The OS installs these handlers at boot time(via privileged instructions)

For example, if a process tries to access memory outside its bounds, the CPU will raise an exception; the OS must be prepared to take ation when such an exception arises.



OS @ boot **(kernel mode)**	**Hardware**	**(No Program Yet)**
initialize trap table		
	remember addresses of... system call handler timer handler illegal mem-access handler illegal instruction handler	
start interrupt timer		
	start timer; interrupt after X ms	
initialize process table **initialize free list**		

Figure 15.5: **Limited Direct Execution (Dynamic Relocation) @ Boot**