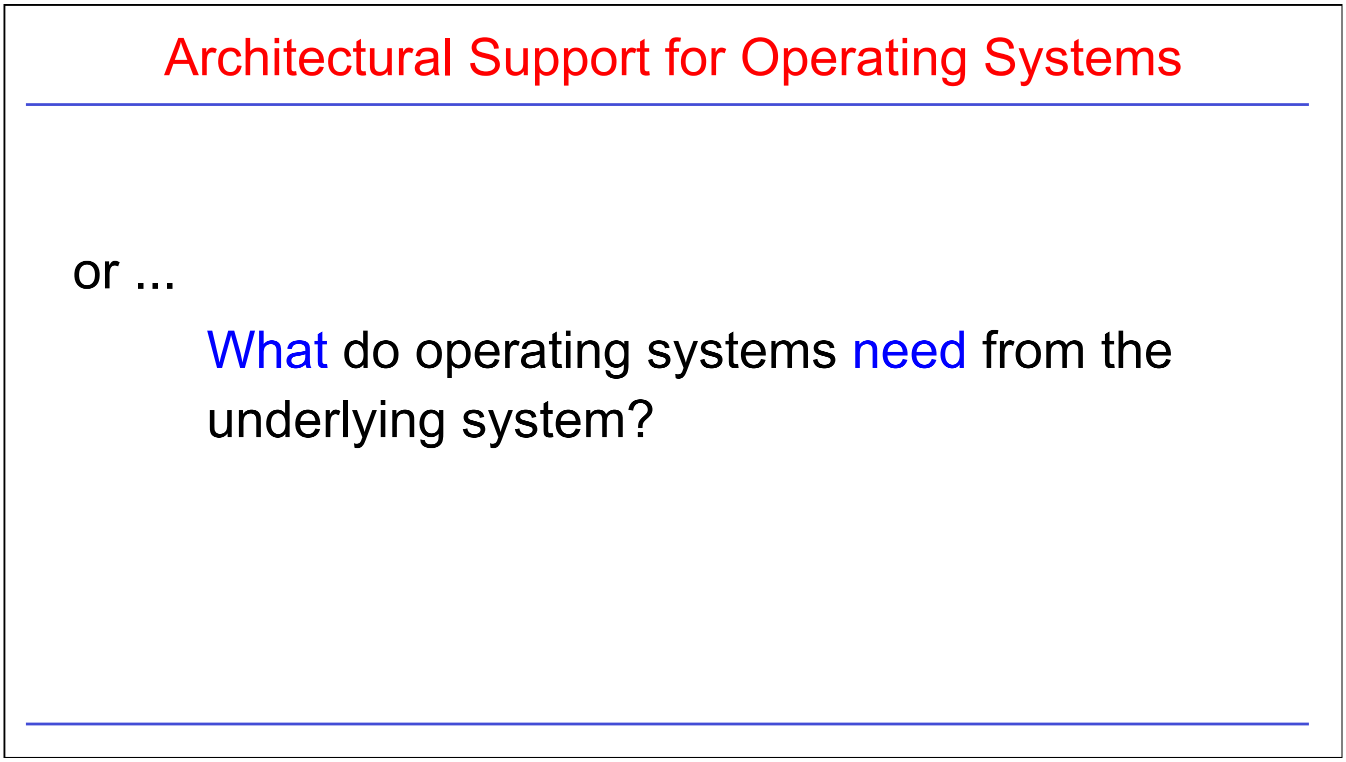
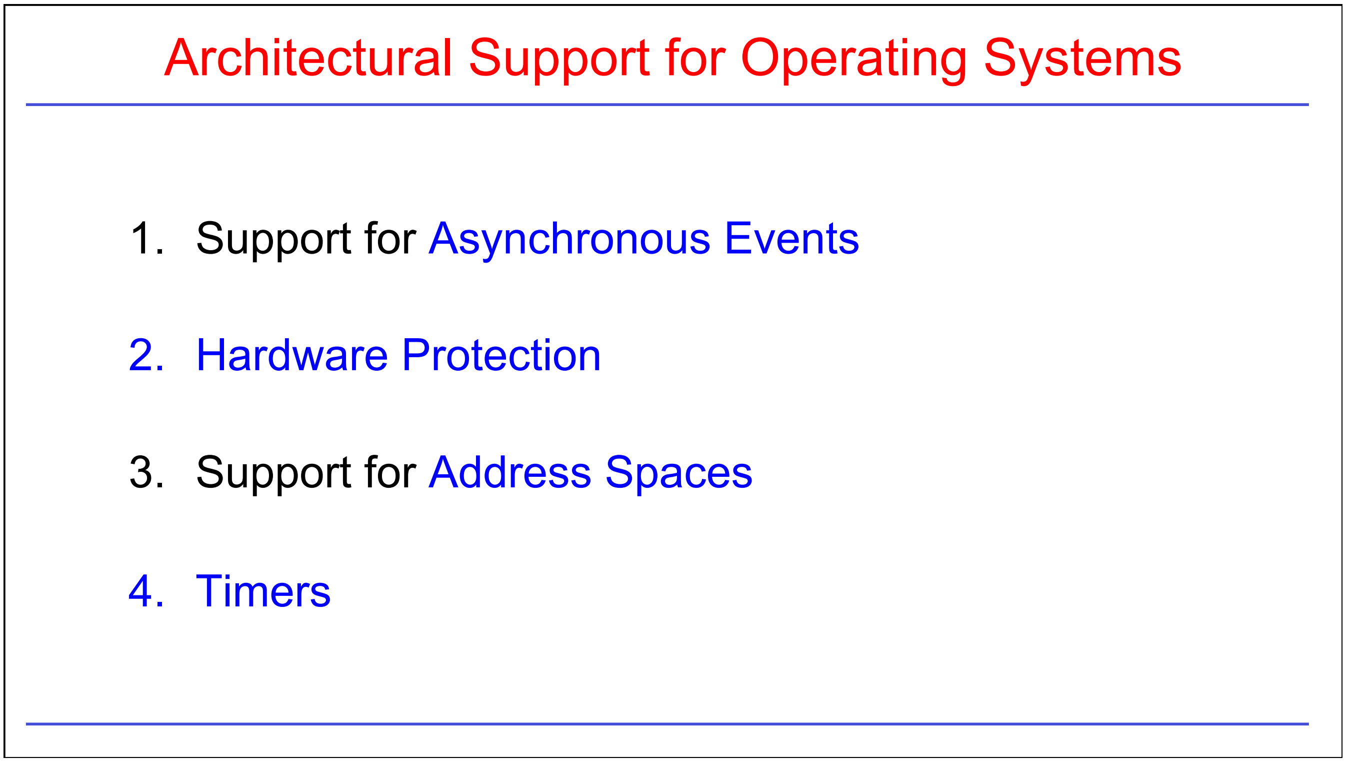


Architectural Support for Operating Systems

or ...

What do operating systems need from the underlying system?



Architectural Support for Operating Systems

1. Support for Asynchronous Events
2. Hardware Protection
3. Support for Address Spaces
4. Timers

1. Support for Asynchronous Events

Observation: Operating systems handle many **Asynchronous Events**

- events from devices
- user input
- timer events

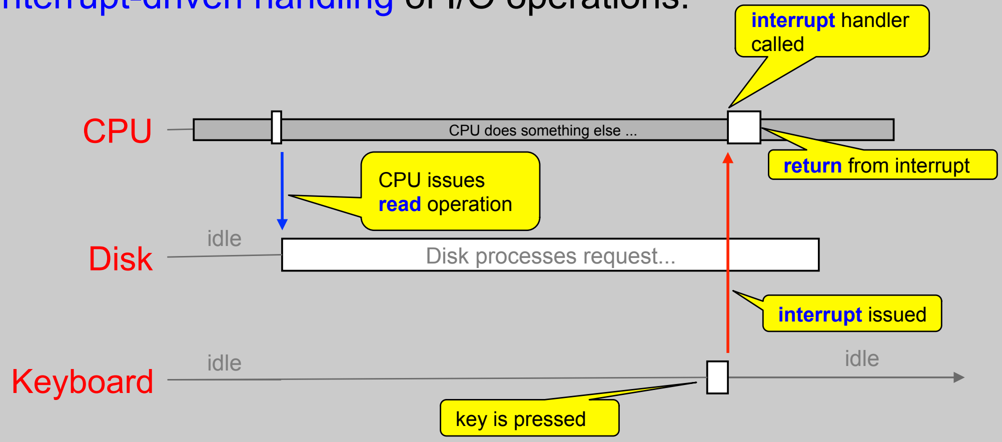
How to handle asynchronous events?

- **Polling**
- **Interrupt-Driven**

Observation: Most operating systems (not all!) handle asynchronous events using interrupts.

Most modern OS’s are Interrupt-Driven

Interrupt-driven handling of I/O operations:



interrupt handler called

CPU

CPU does something else ...

return from interrupt

CPU issues **read** operation

idle

Disk

Disk processes request...

interrupt issued

idle

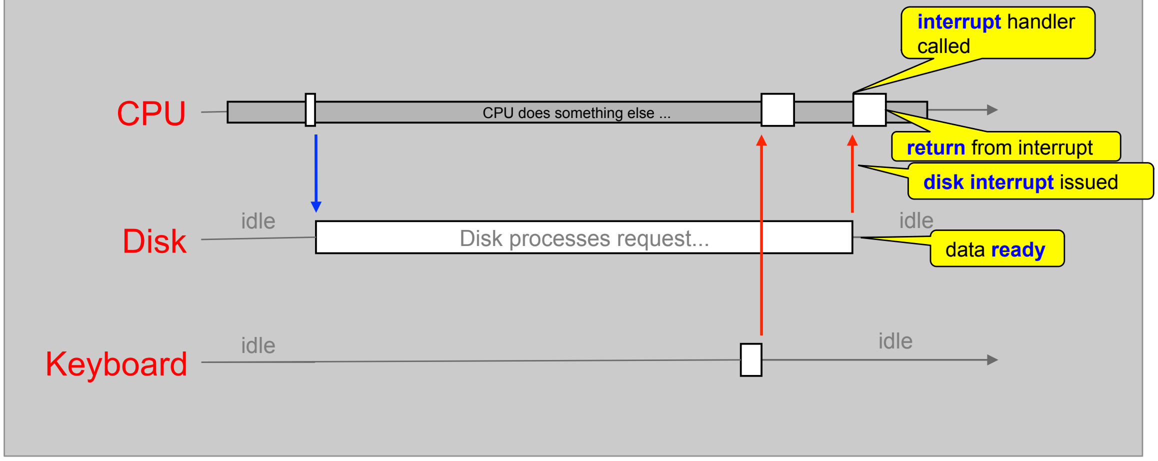
Keyboard

idle

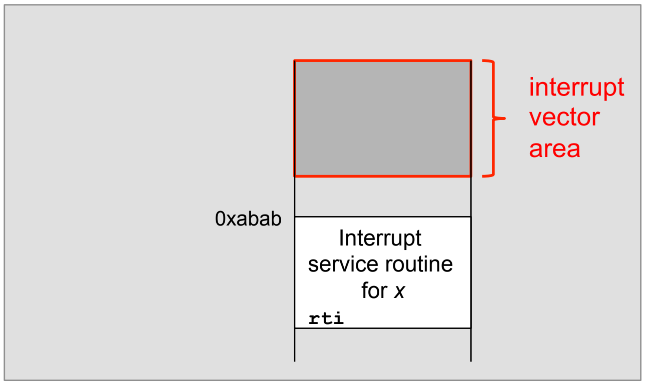
key is pressed

Most modern OS’ s are Interrupt-Driven

Interrupt-driven handling of I/O operations:



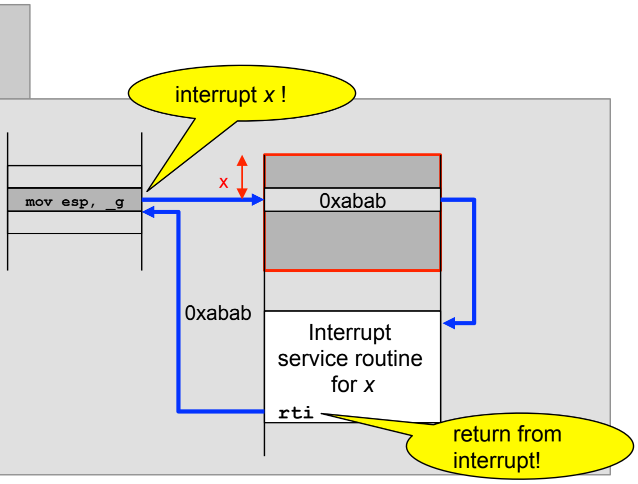
Interrupts



Interrupts

When an interrupt occurs, the CPU ...

1. stops
2. saves state
3. changes into supervisor mode
4. branches to predefined location.

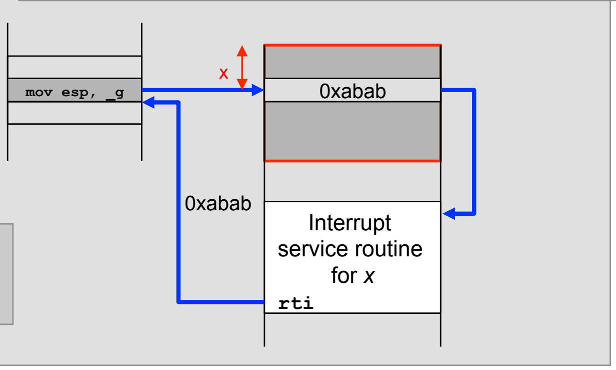


Interrupts

When an interrupt occurs, the CPU ...

1. stops
2. saves state
3. changes into supervisor mode
4. branches to predefined location.

Appropriate interrupt service routine is found through the interrupt vector.



Return-from-interrupt (**`rti`**) automatically restores state.

Interrupts/Exceptions can be invoked by *asynchronous events* (I/O devices, timers, various errors) or can be *software-generated* (system calls).

2. Hardware Protection

Recall: Originally, user owned the machine; there was no monitor.

- Therefore, protection was not necessary.

Starting with resident monitor, user programs start sharing resources, either sequentially, or concurrently.

- One program can adversely affect the execution of others.

Benign (bug) *vs.* malicious (virus)

How?!

1. `halt` and other instructions
2. access/modify data on devices
3. modify data or code in other programs or monitor itself
4. refuse to relinquish processor

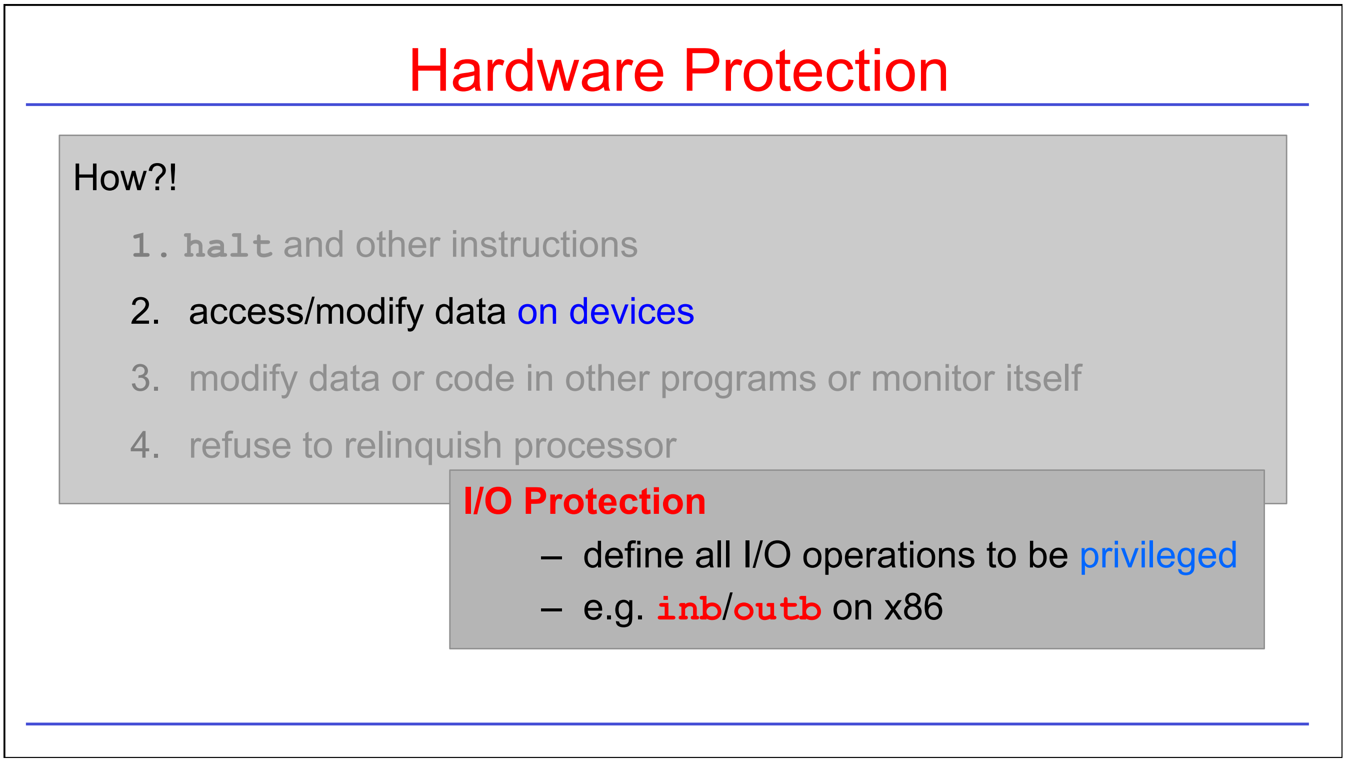
Hardware Protection

How?!

1. `halt` and other instructions
2. access/modify data on devices
3. modify data or code in other programs or monitor itself
4. refuse to relinquish processor

Dual-mode operation

- *user mode* vs. *supervisor mode*
- e.g. `halt` instruction is privileged



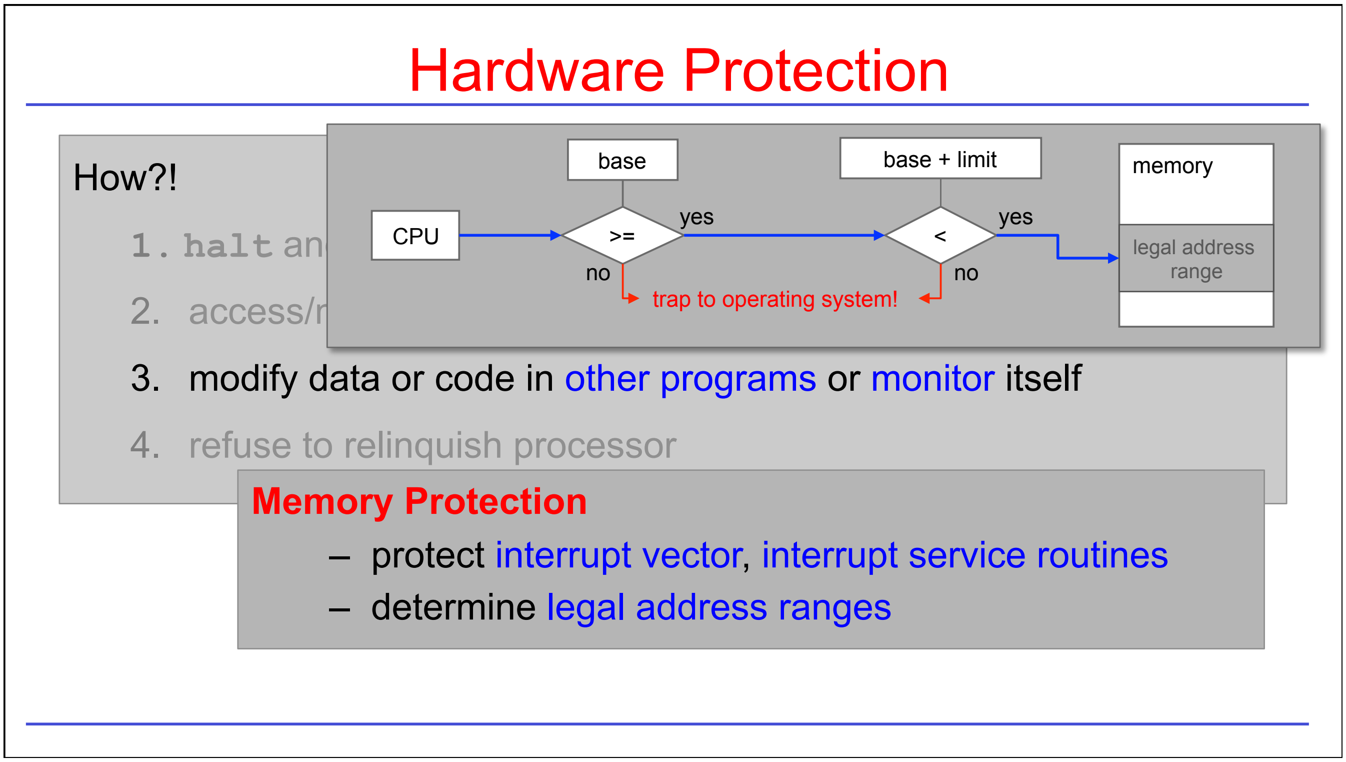
Hardware Protection

How?!

1. `halt` and other instructions
2. access/modify data on devices
3. modify data or code in other programs or monitor itself
4. refuse to relinquish processor

I/O Protection

- – define all I/O operations to be privileged
- – e.g. `inb`/`outb` on x86



Hardware Protection

How?!

1. `halt` an
2. access/r
3. modify data or code in other programs or monitor itself
4. refuse to relinquish processor

Memory Protection

- – protect interrupt vector, interrupt service routines
- – determine legal address ranges

Hardware Protection

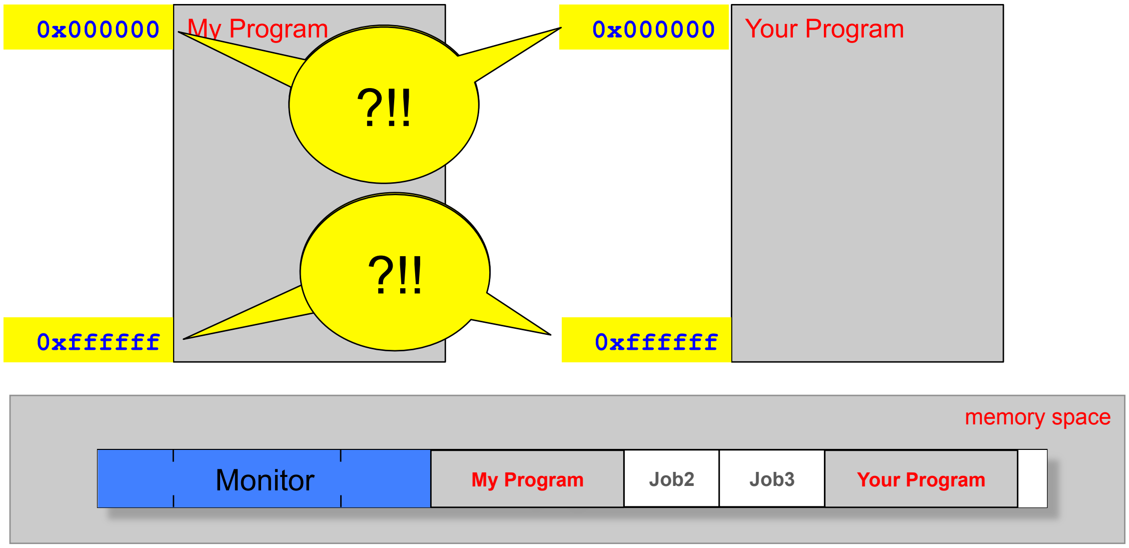
How?!

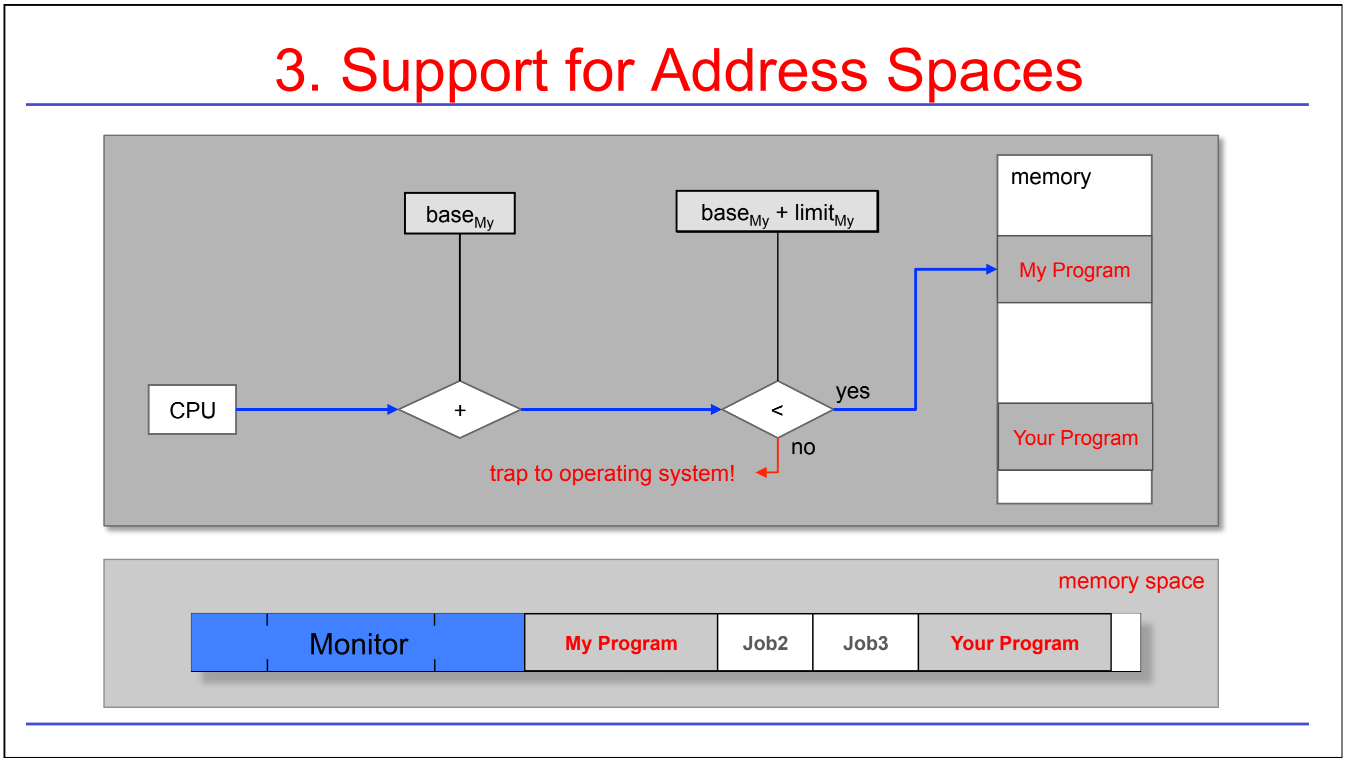
1. `halt` and other instructions
2. access/modify data on devices
3. modify data or code in other programs or monitor itself
4. refuse to relinquish processor

Timers

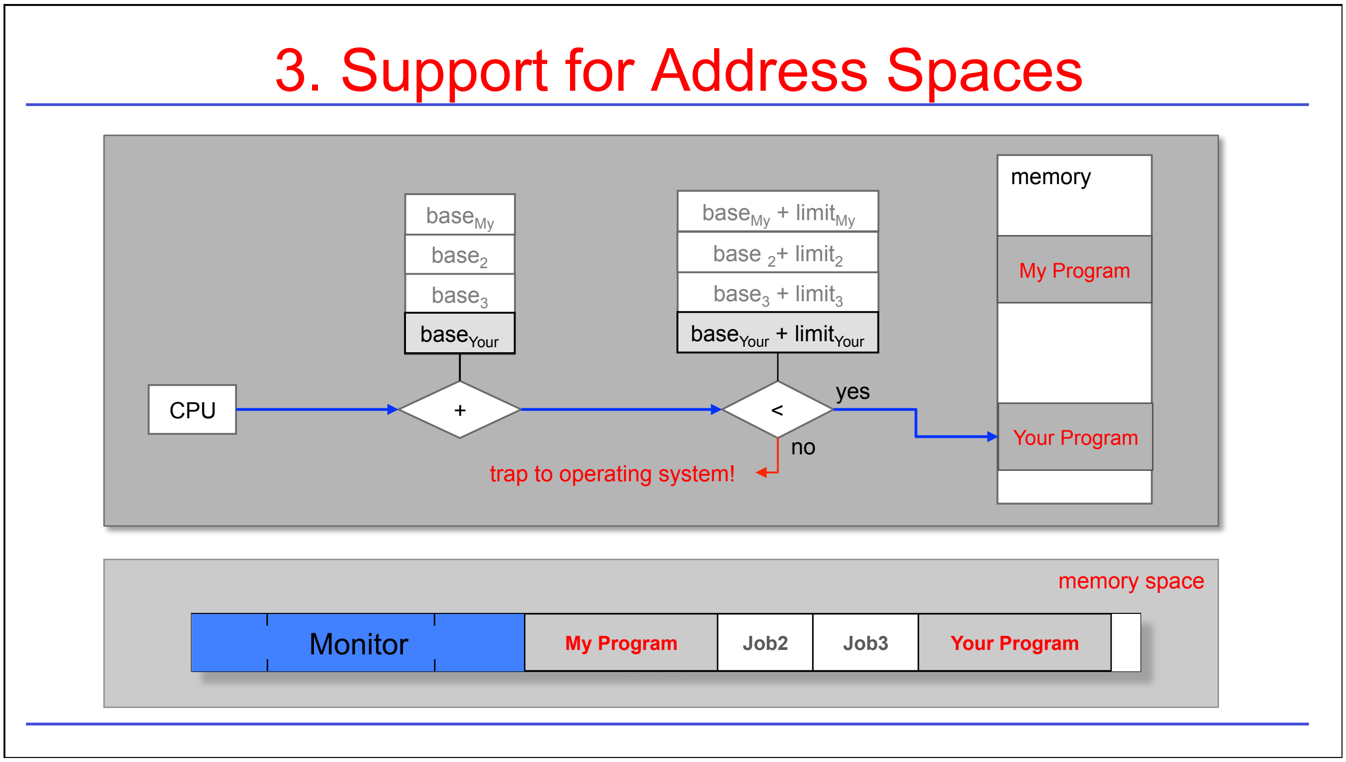
- Timers can be set, and a interrupt occurs when the timer expires.
- OS acquires control over the CPU again.

3. Support for Address Spaces





3. Support for Address Spaces



3. Support for Address Spaces

memory
base_{My}
base_{2}
base_{3}
base_{Your}
$\text{base}_{My} + \text{limit}_{My}$
$\text{base}_{2} + \text{limit}_{2}$
$\text{base}_{3} + \text{limit}_{3}$
$\text{base}_{Your} + \text{limit}_{Your}$
My Program
CPU
+
<
yes
no
Your Program
trap to operating system!
memory space
Monitor
My Program
Job2
Job3
Your Program

4. Timers

Recall: Timers can be set, and an interrupt occurs when the timer expires.

(And OS acquires control over the CPU.)

Other uses of timers

- time sharing
- time-of-day clock

Summary: What does the Operating System need?

- Support for Asynchronous Events: Interrupts
- Hardware Protection: dual/mode operation, I/O protection, memory protection, timers
- Support for Address Spaces: per-process relocation
- Timers
- Atomic Operations? DMA? Task-State Segments?