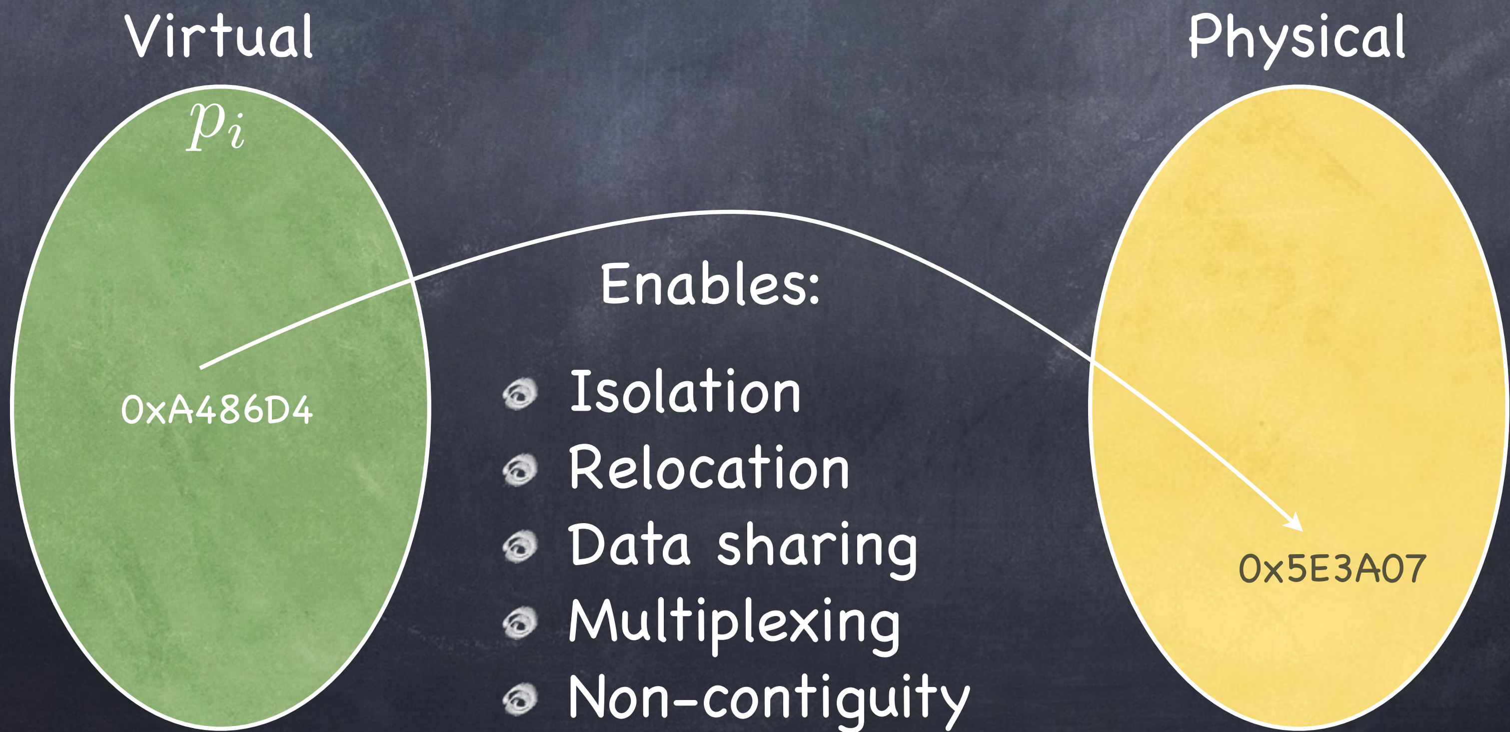
II. Memory Isolation

Step 2: Address Translation

- Implement a function mapping

$\langle pid, virtual\ address \rangle$ into $physical\ address$

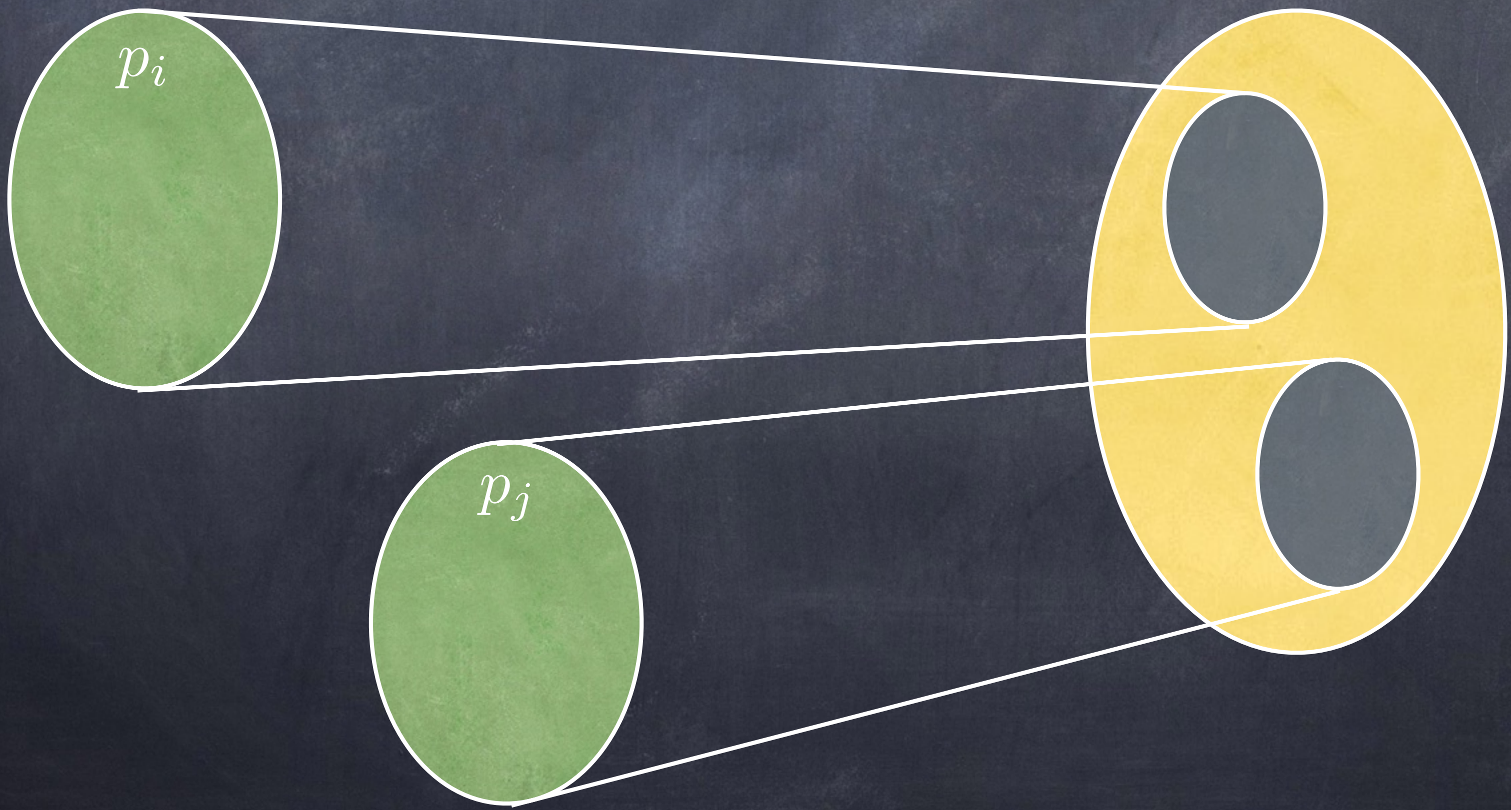


Enables:

- Isolation
- Relocation
- Data sharing
- Multiplexing
- Non-contiguity

Isolation

- At all times, functions used by different processes map to disjoint ranges – aka "Stay in your room!"

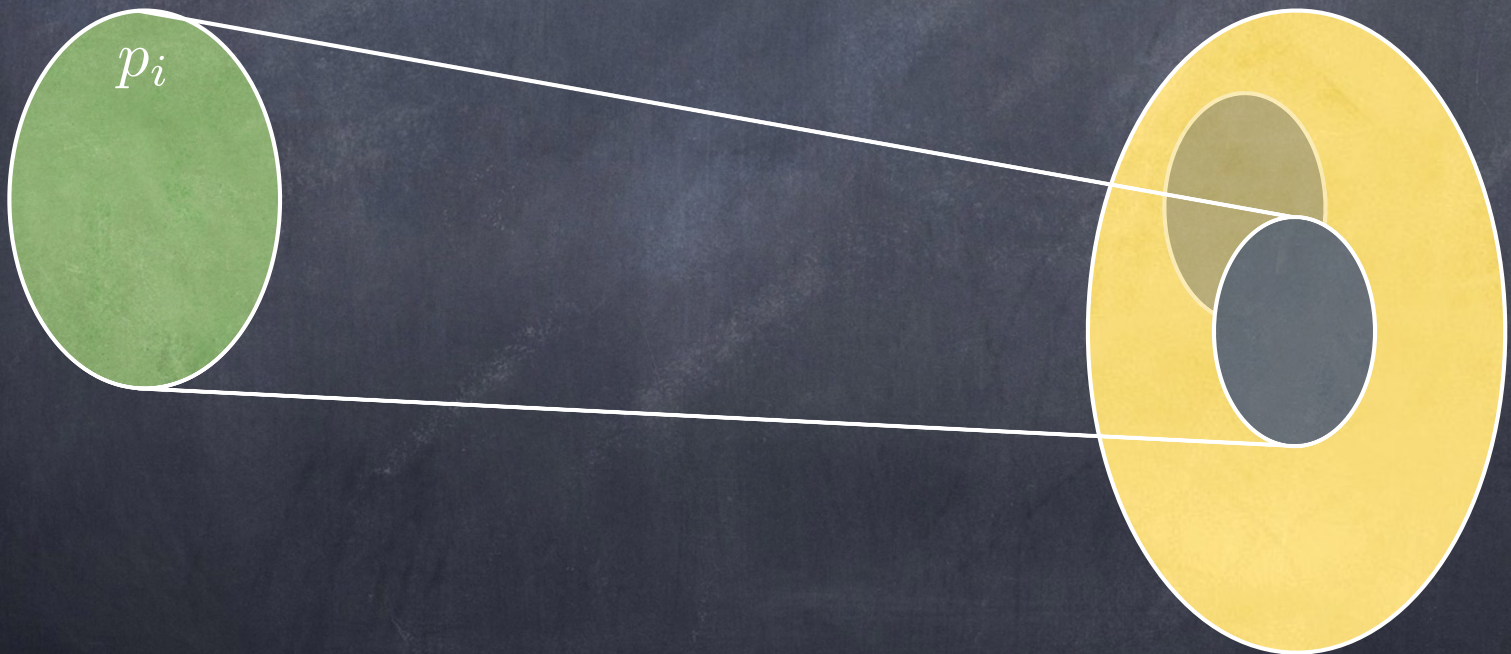


Relocation

- The range of the function used by a process can change over time

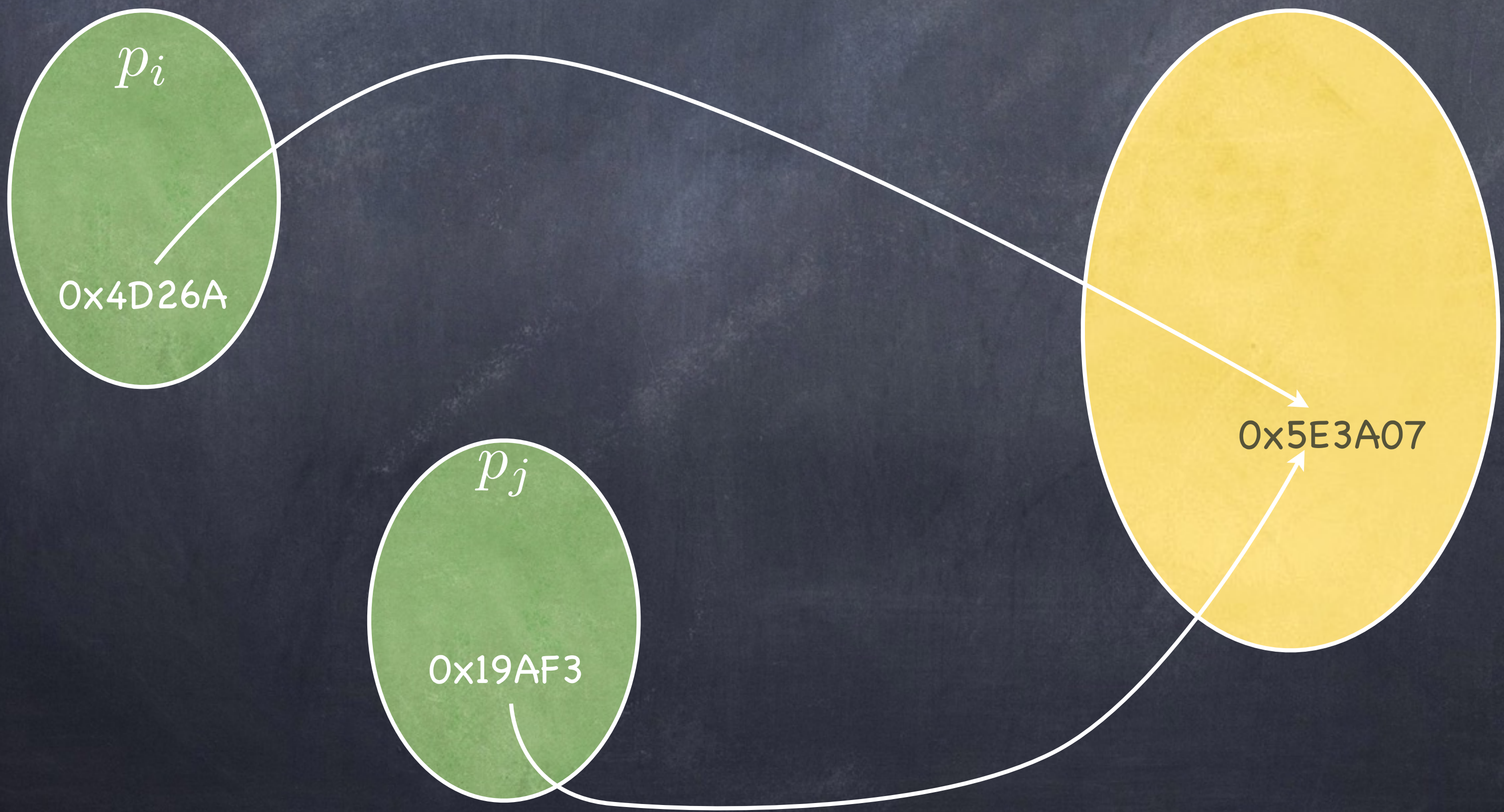
Relocation

- The range of the function used by a process can change over time — "Move to a new room!"



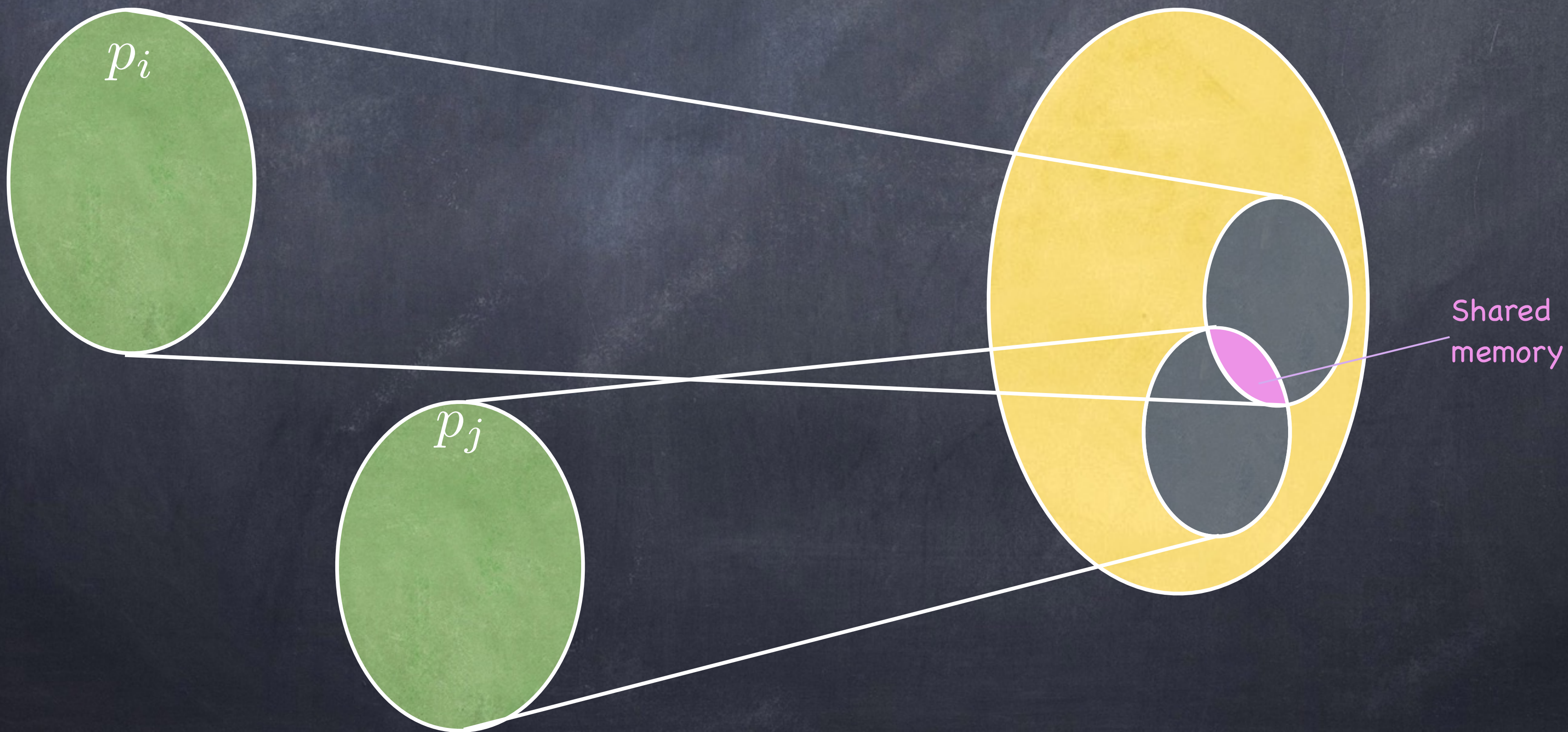
Data Sharing

- Map different virtual addresses of distinct processes to the same physical address – ("Share the kitchen")



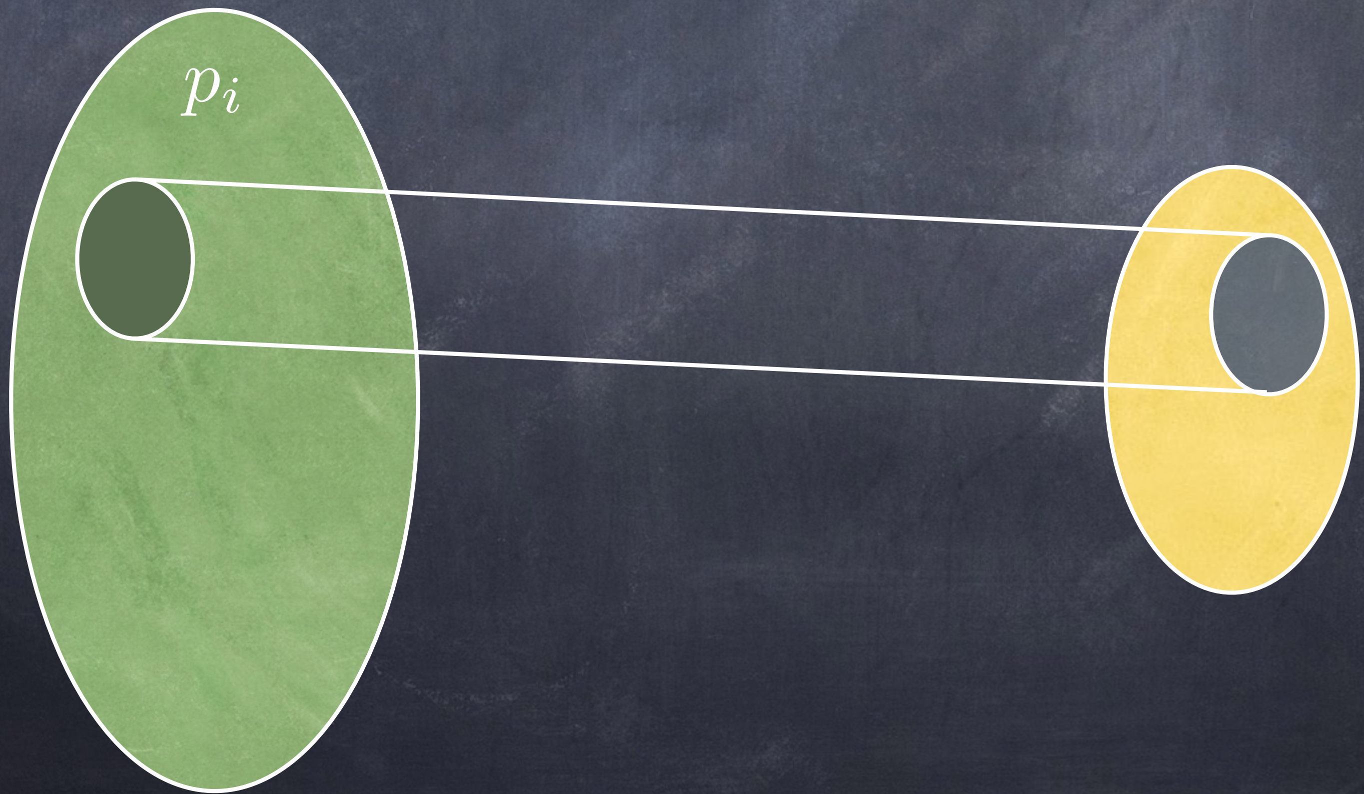
Data Sharing

- Map different virtual addresses of distinct processes to the same physical address – ("Share the kitchen")



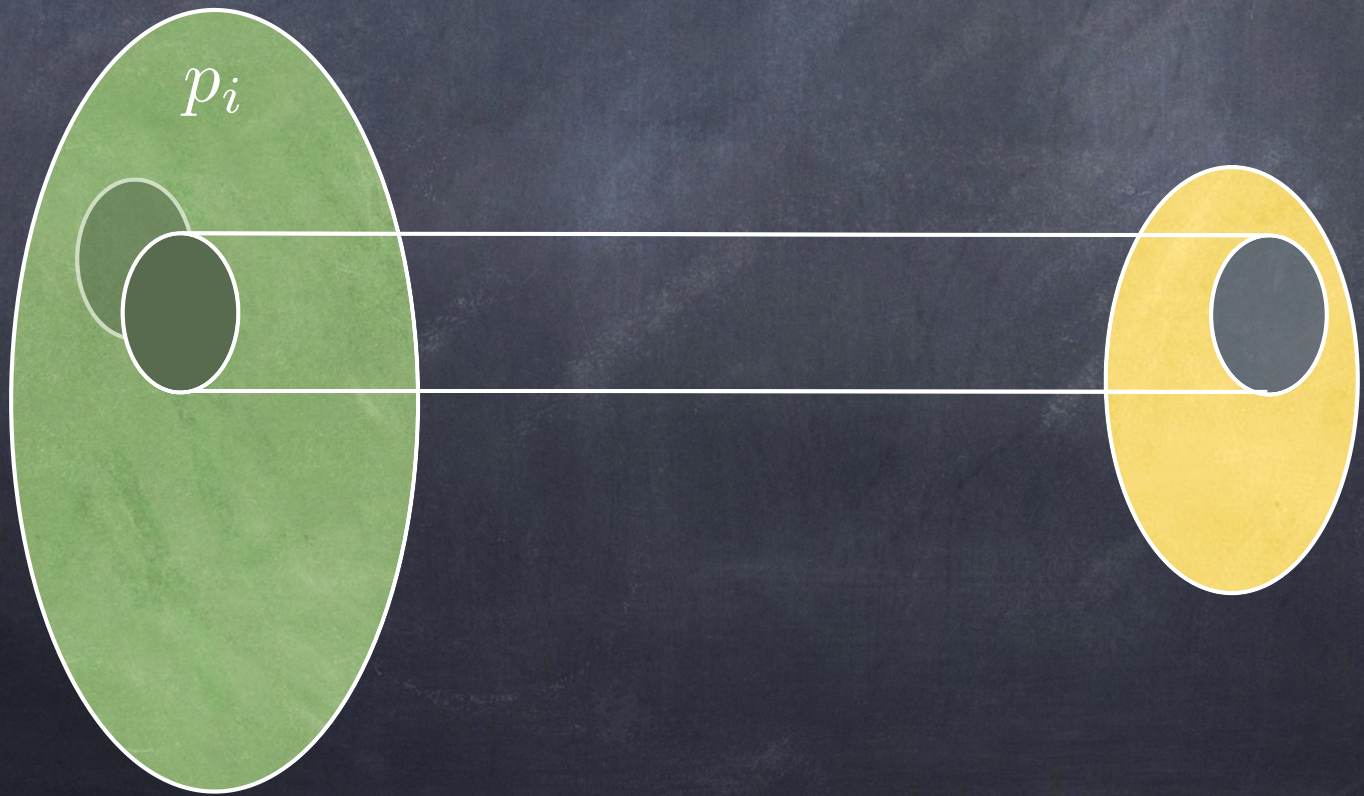
Multiplexing

- Create illusion of almost infinite memory by changing domain (set of virtual addresses) that maps to a given range of physical addresses — ever lived in a studio?



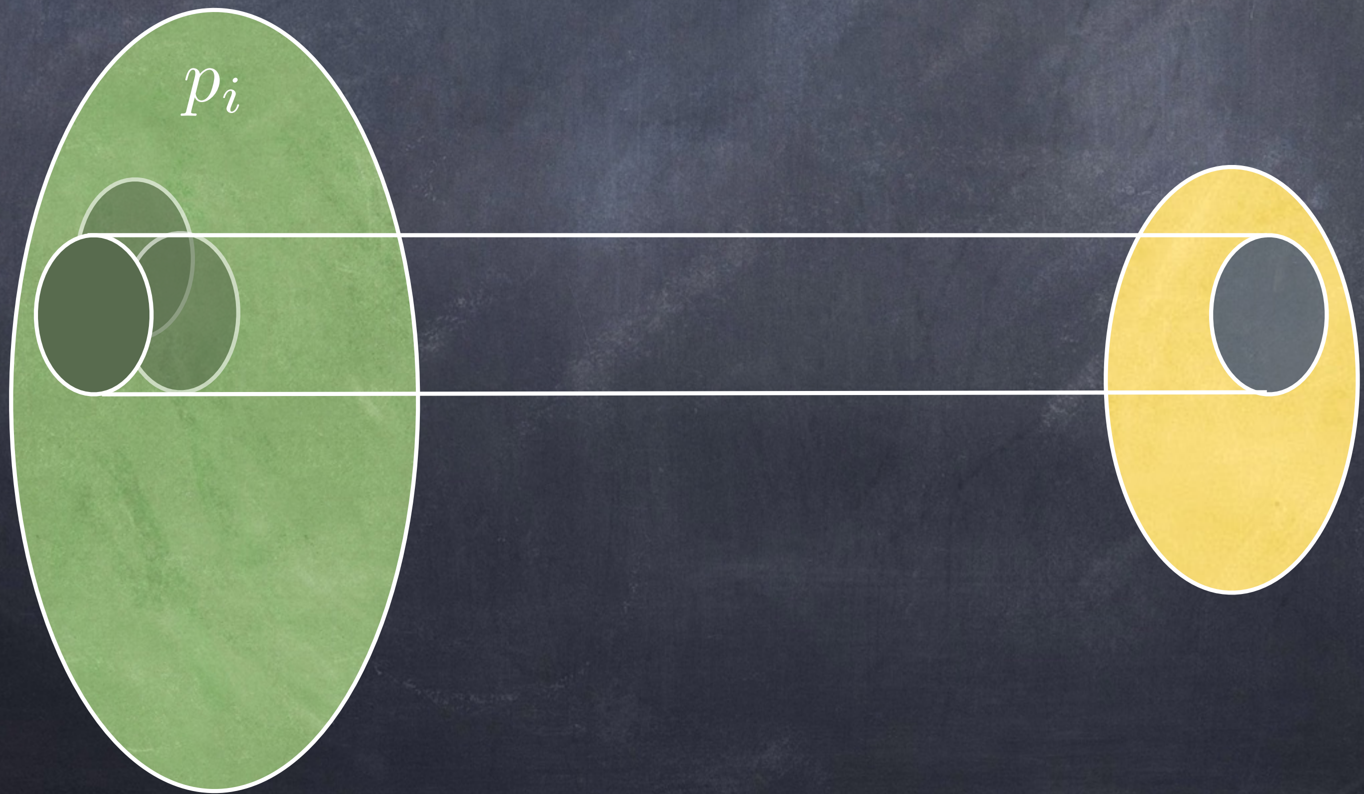
Multiplexing

- The domain (set of virtual addresses) that map to a given range of physical addresses can change over time



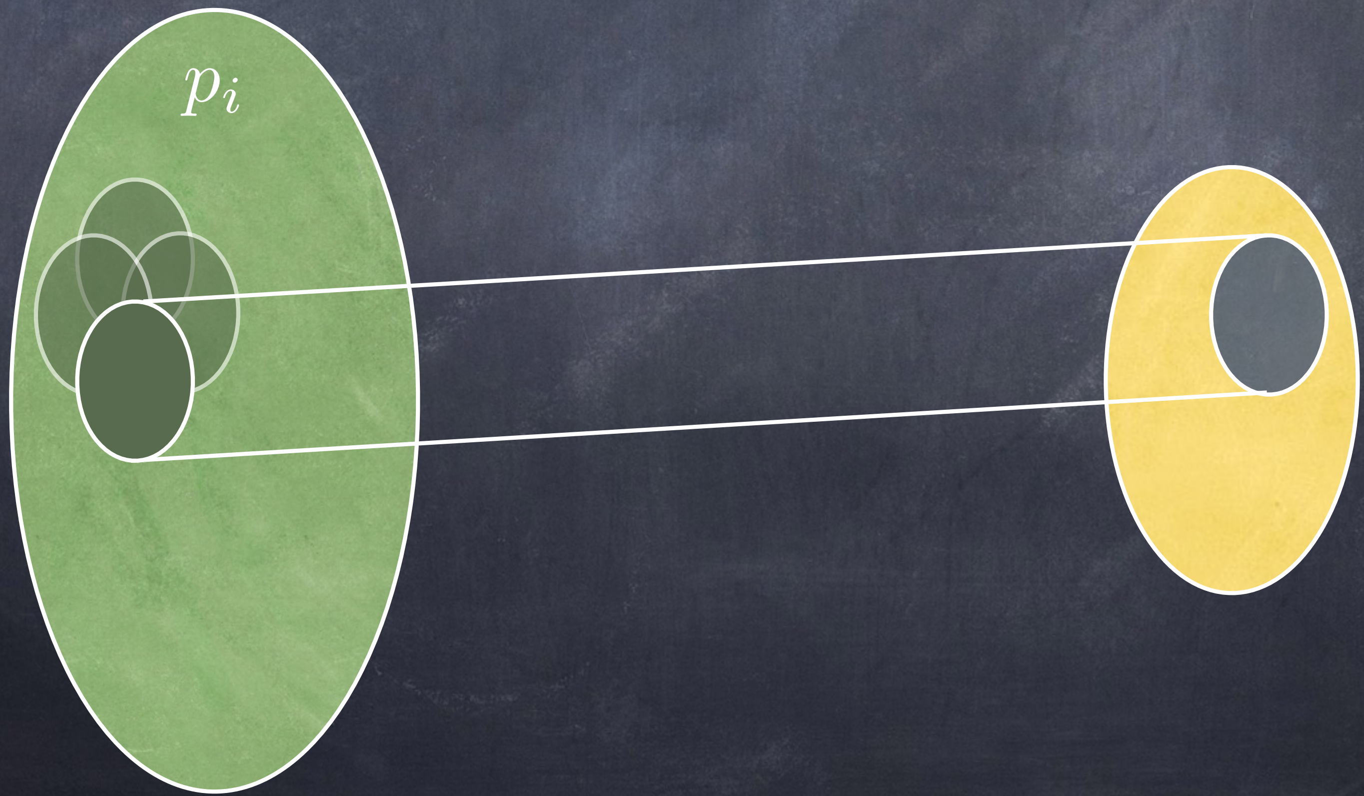
Multiplexing

- The domain (set of virtual addresses) that map to a given range of physical addresses can change over time



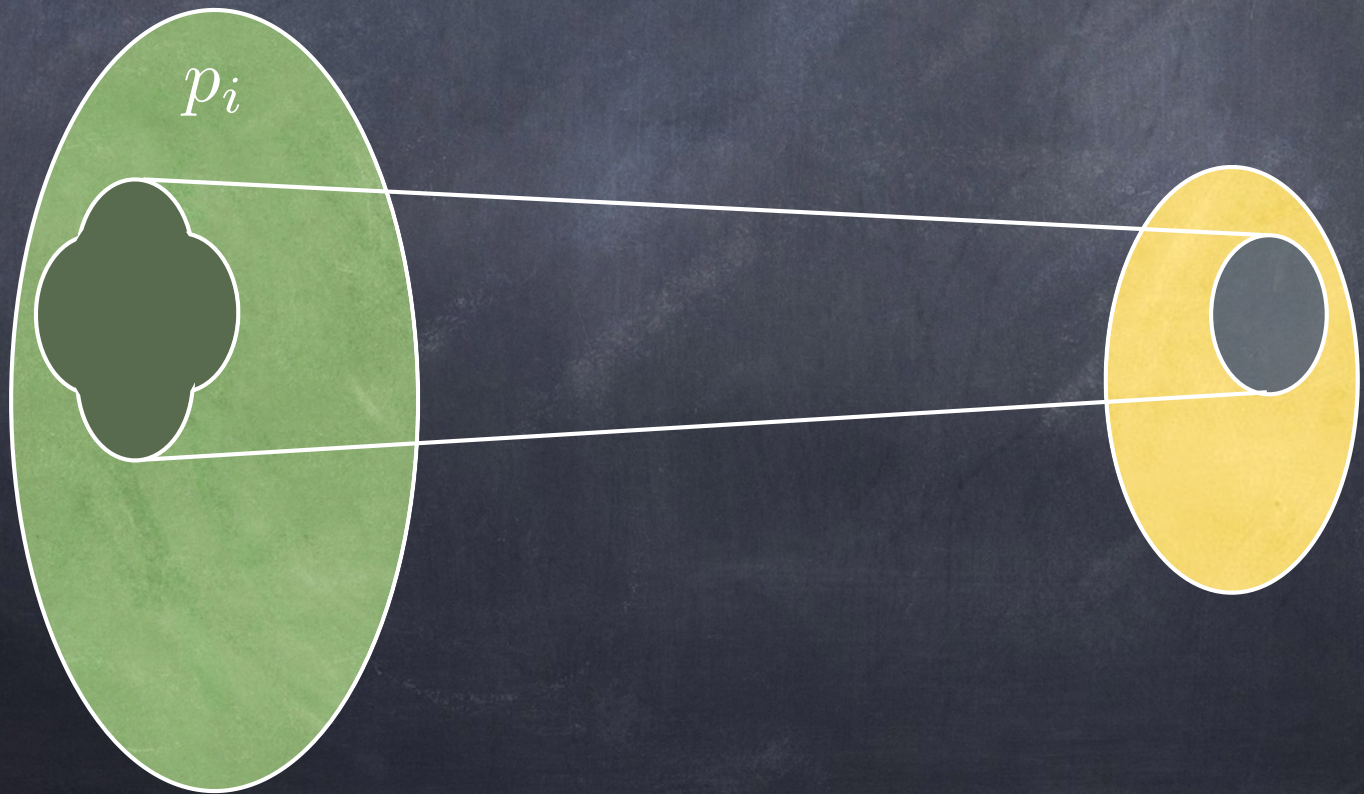
Multiplexing

- The domain (set of virtual addresses) that map to a given range of physical addresses can change over time



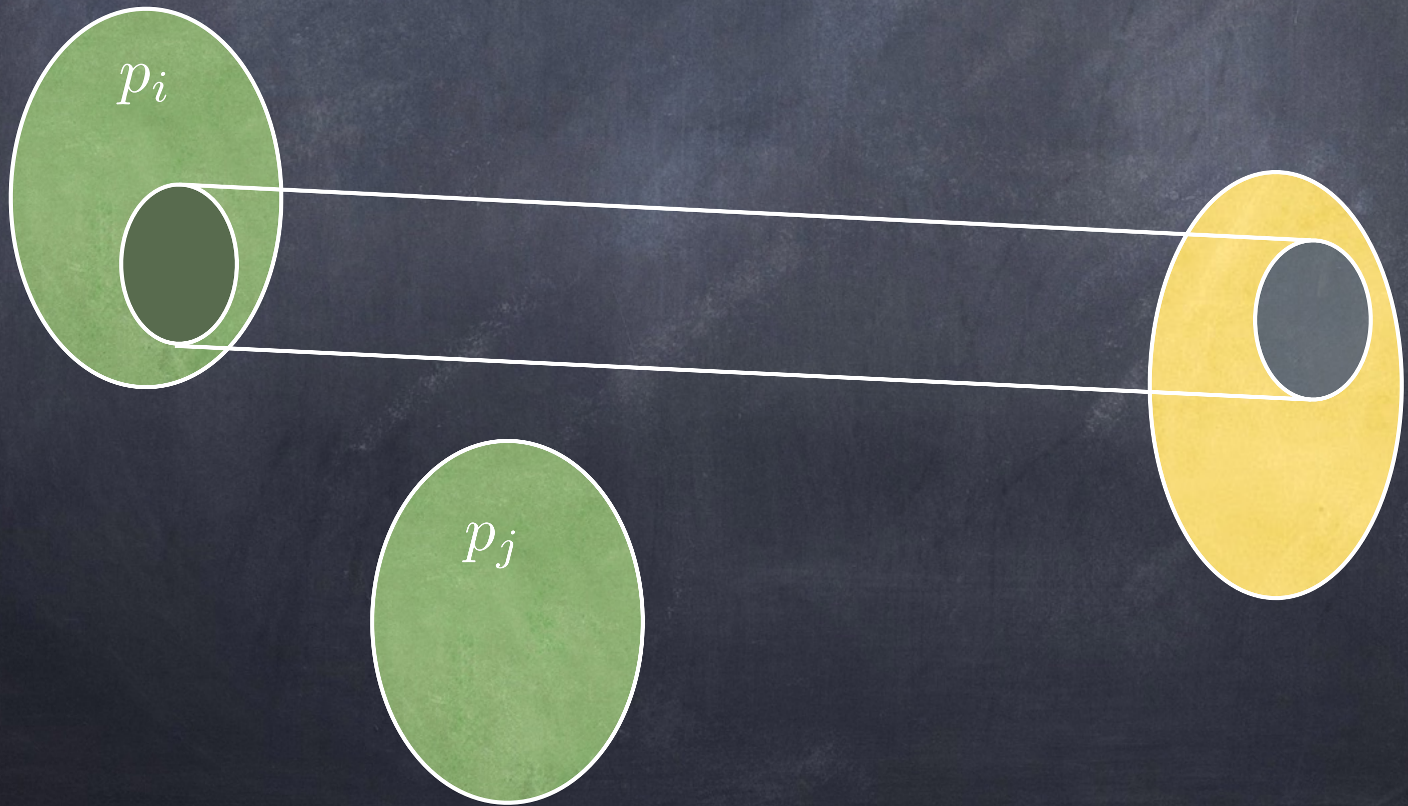
Multiplexing

- The domain (set of virtual addresses) that map to a given range of physical addresses can change over time



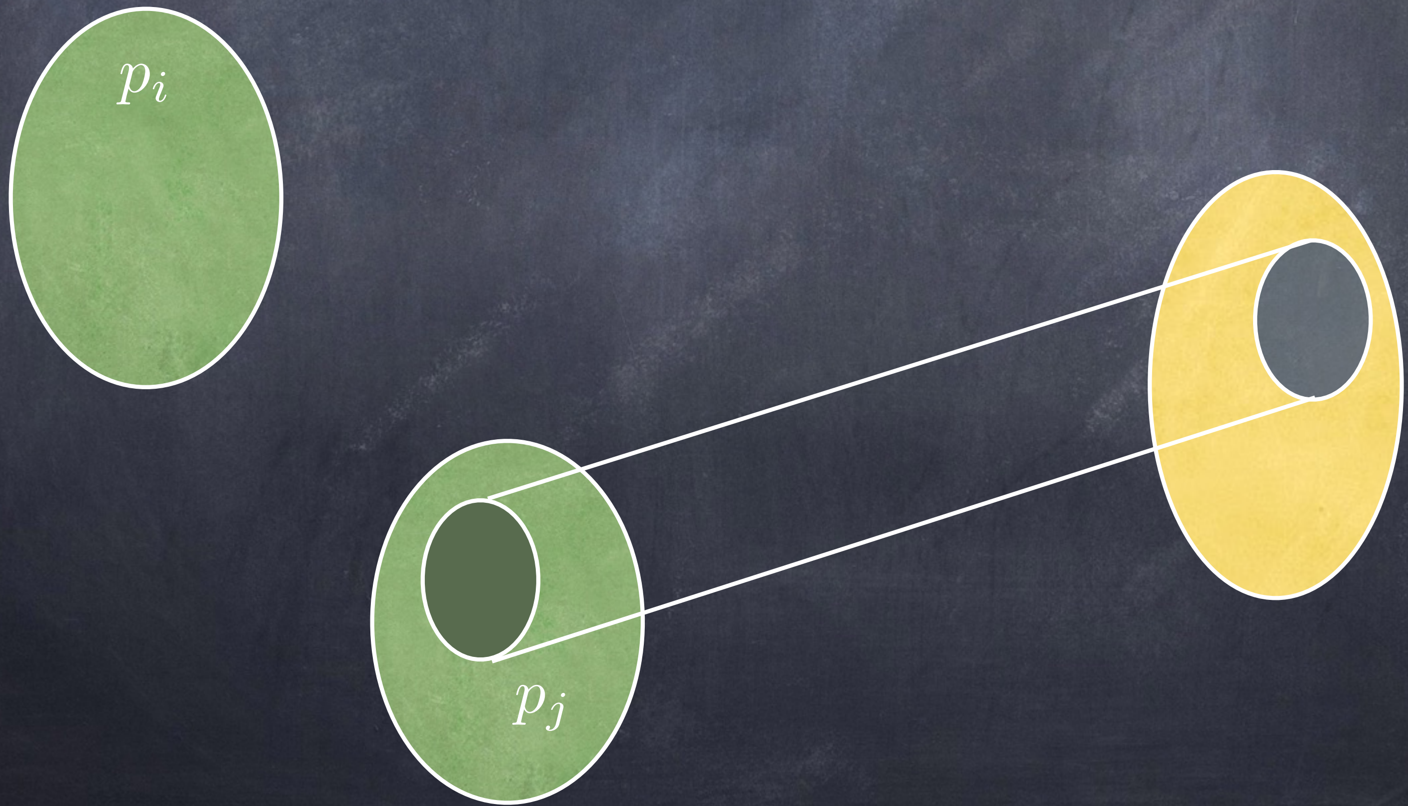
More Multiplexing

- At different times, different processes can map part of their virtual address space into the same physical memory – (change tenants)



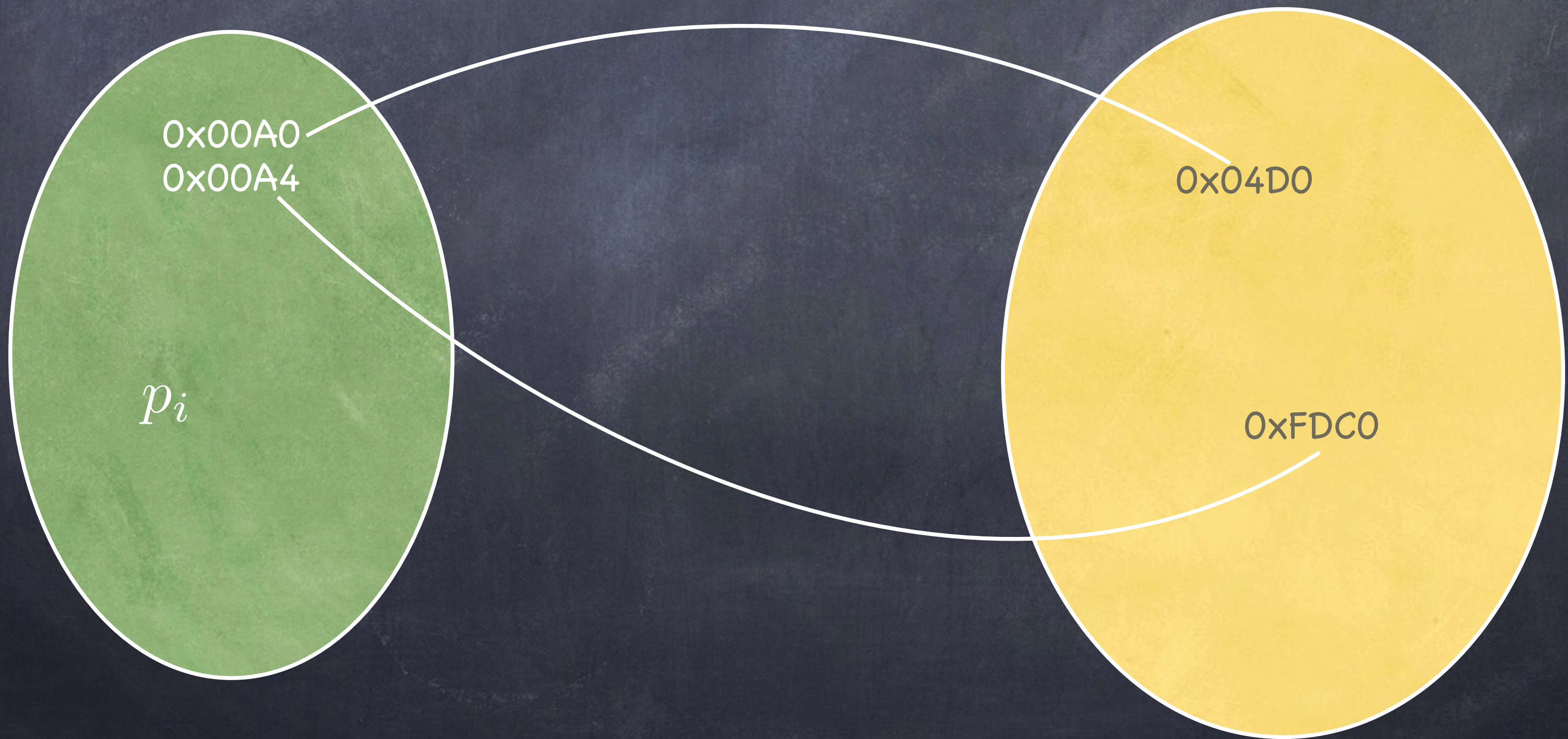
More Multiplexing

- At different times, different processes can map part of their virtual address space into the same physical memory — (change tenants)



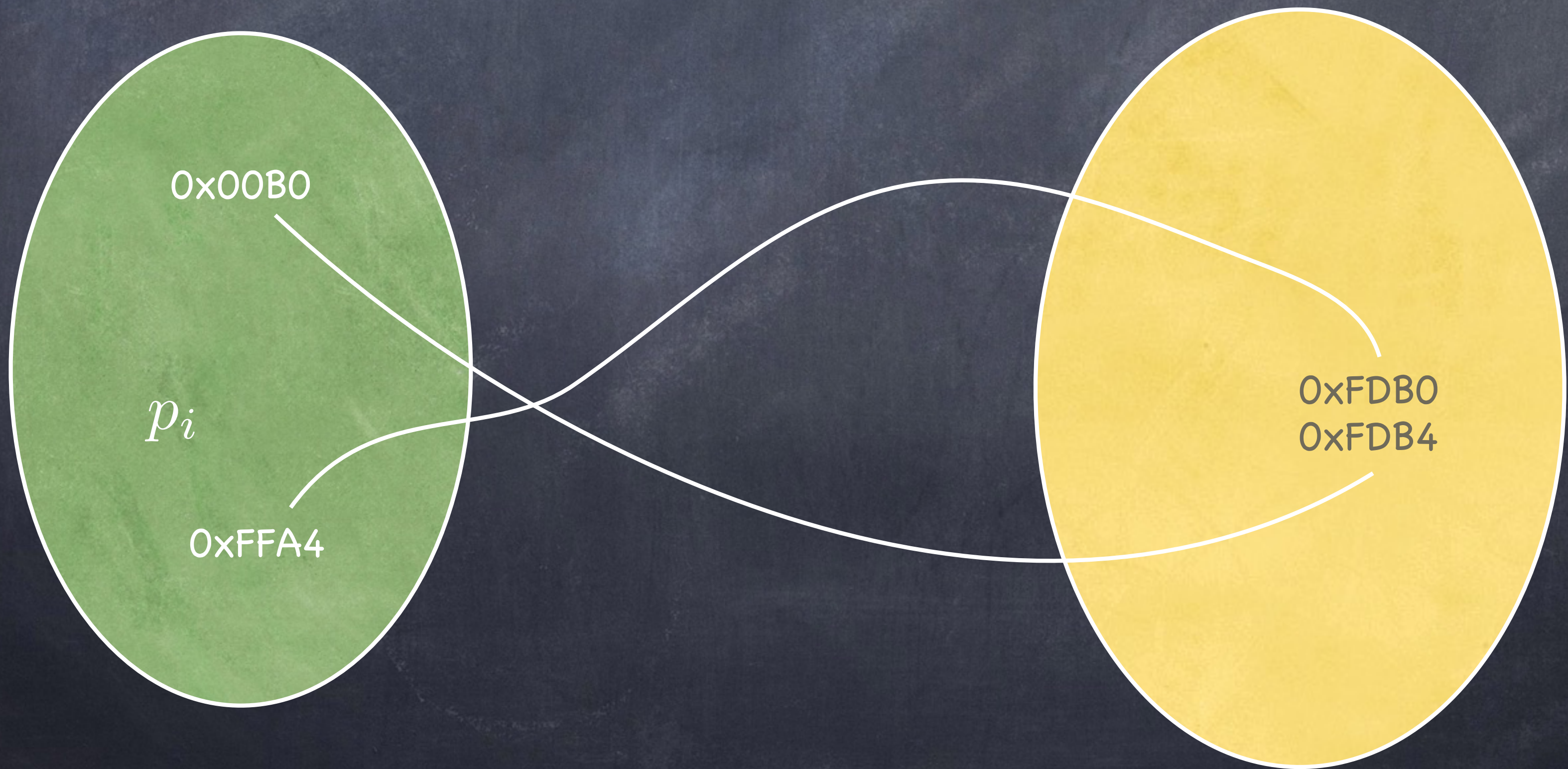
(Non) Contiguity

- Contiguous virtual addresses can be mapped to non-contiguous physical addresses…



(Non) Contiguity

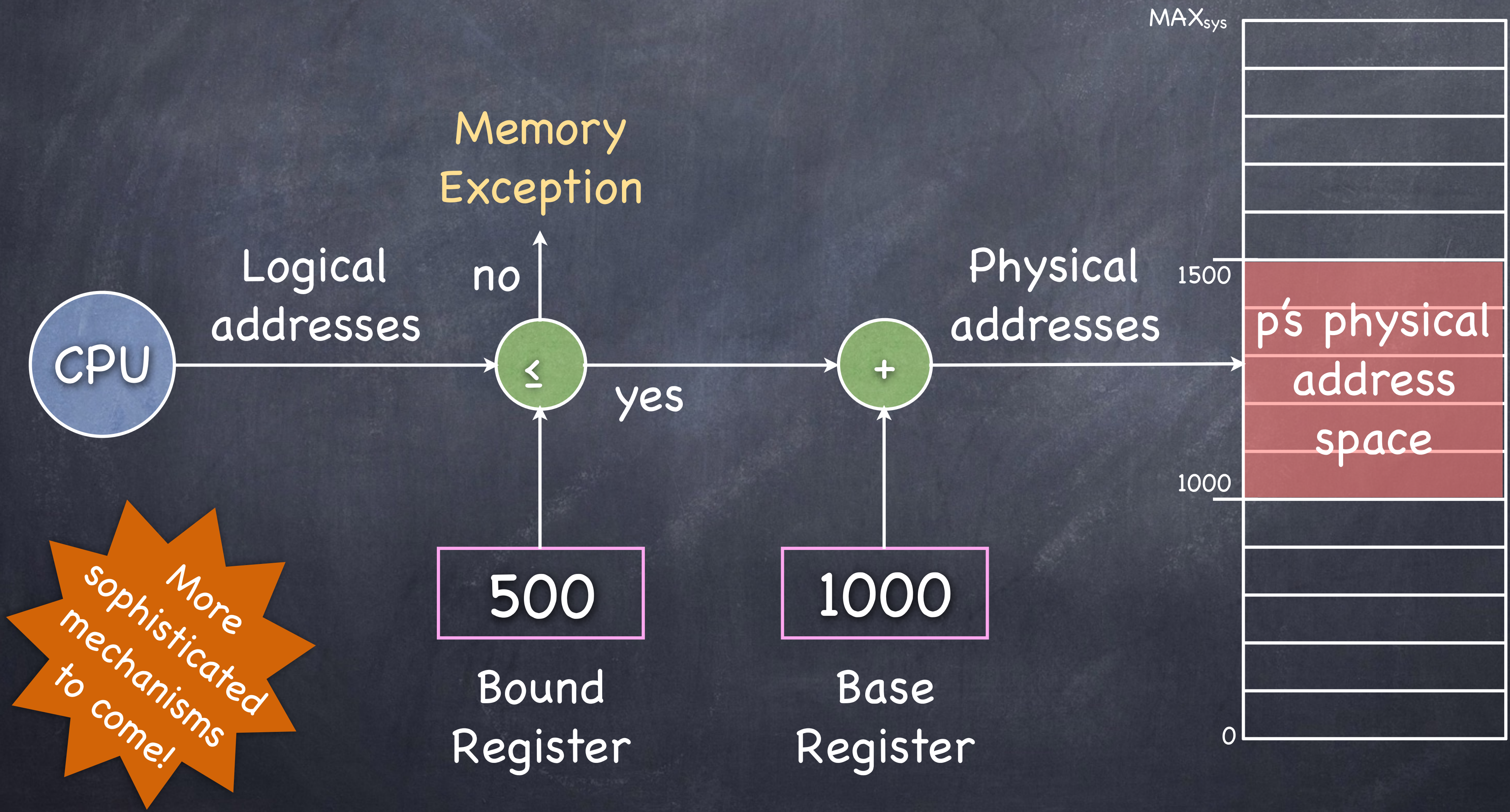
- ...and non-contiguous virtual addresses can be mapped to contiguous physical addresses



A simple mapping mechanism: Base & Bound

Hardware
to the rescue!

A simple mapping mechanism: Base & Bound



On Base & Limit

- Contiguous Allocation: contiguous virtual addresses are mapped to contiguous physical addresses
- Isolation is easy, but sharing is hard
  - Say I have many copies of Safari open... 
    - I may want them to share the same code, or even the same global variables
- And there is more...
  - Hard to relocate
    - Addresses are absolute and may be stored in registers or on the stack (a return address)

Supporting Dual-Mode Operation

- Privileged Instructions
- Memory Isolation
- Timer* Interrupts

Questions?

Supporting Dual-Mode Operation

- Privileged Instructions
- Memory Isolation
- Timer* Interrupts

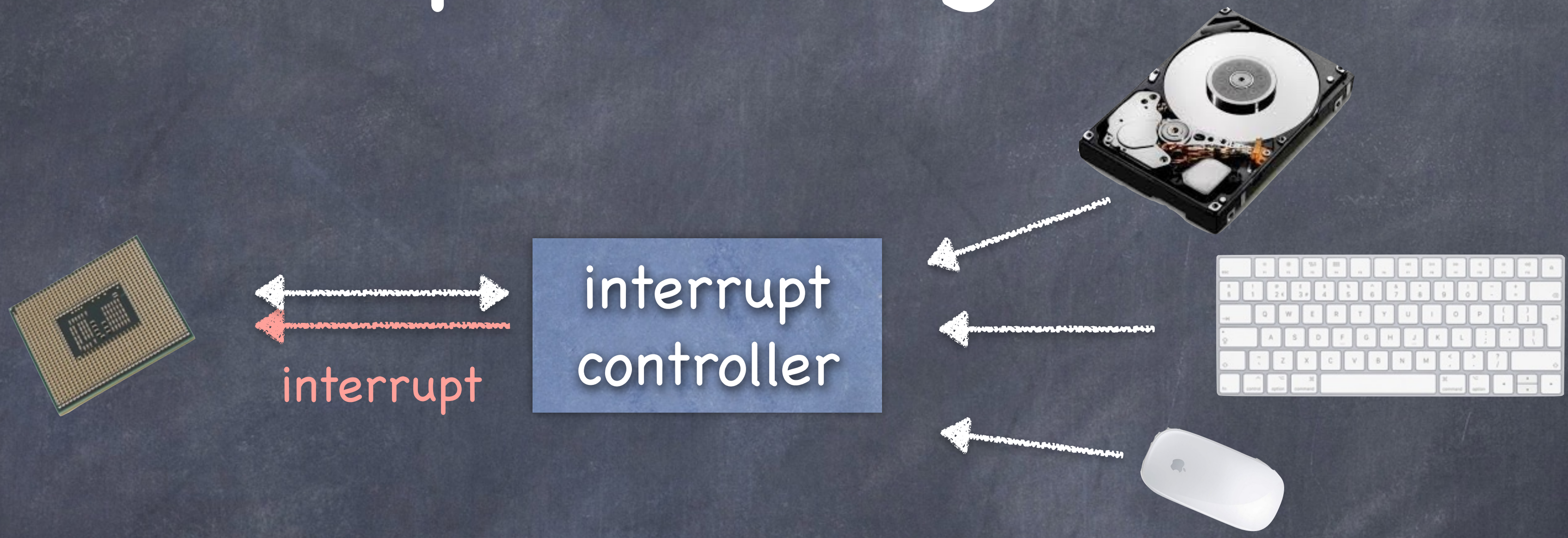
Giving control back to the

Hardware
to the rescue!

III. Timer Interrupts

- Hardware timer
  - can be set to expire after specified delay (time or instructions)
  - when it does, control is passed back to the kernel
- Other interrupts (e.g., due to I/O completion) also give back control to kernel

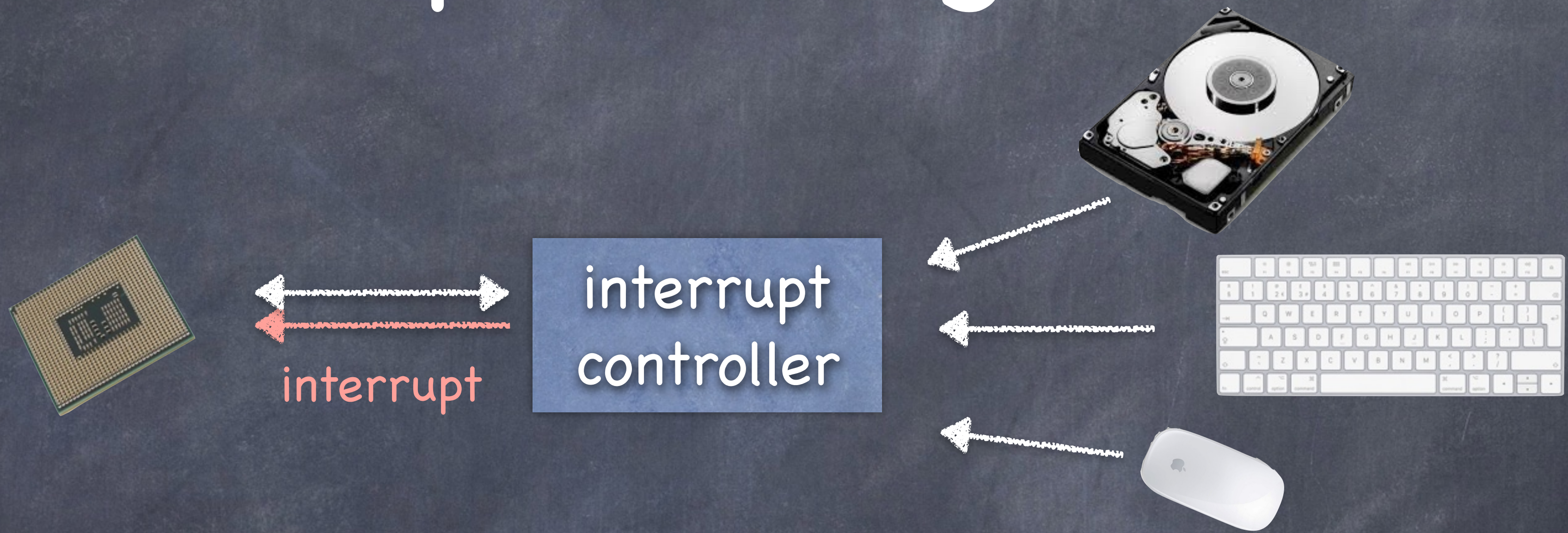
Interrupt Management



Interrupt controllers implements interrupt priorities:

- Interrupts include descriptor of interrupting device
- Priority selector circuit examines all interrupting devices, reports highest level to the CPU
- Controller can also buffer interrupts coming from different devices

Interrupt Management



Maskable interrupts

- can be turned off by the CPU for critical processing

Nonmaskable interrupts

- indicate serious errors (power out warning, unrecoverable memory error, etc.

Types of Interrupts

Exceptions

- process missteps (e.g. division by zero)
- attempt to perform a privileged instruction
  - sometime on purpose! (breakpoints)
- synchronous/non-maskable

System calls

- user program requests OS service
- synchronous/non-maskable

Interrupts

- HW device requires OS service
  - timer, I/O device, interprocessor
- asynchronous/maskable

Interrupt Handling

- Two objectives
  - handle the interrupt and remove the cause
  - restore what was running before the interrupt
    - kernel may modify saved state on purpose
- Two "actors" in handling the interrupt
  - the hardware goes first
  - the kernel code takes control by running the interrupt handler