

Virtual Memory

I

CSCI-0310

Pascal Van Hentenryck



Overview

Motivation

- running programs that exceed the size of main memory
- running several programs on the machine

Outline

- slide 3
 - slide 4
 - slide 5
 - ...
-
- slide 37

Virtual Memory

Computer systems often use main memory as a “cache” for a much larger memory space on a magnetic disk

- addresses in this larger space are called “virtual addresses”
- disk accesses are much, much slower than RAM, so our strategies change

Important properties of disks:

- it takes a long time to initiate a data transfer
- getting more data from the same place doesn't cost too much extra

Note that I am not talking about file systems and persistent storage

- Take CS-167

Virtual Memory

Because of the way disks work

- ❑ any miss is *very* expensive
- ❑ large blocks do not cost much more than small ones

Most VM systems use large blocks; called *pages* of about 4-16 kbytes.

- ❑ spacial locality

Another term, in the VM world, for a miss is a *page fault*.

Because disks are so slow, we can handle page faults in software without a big penalty.

Virtual Memory

Address space

- ❑ This is the set of possible addresses that can be accessed by a program

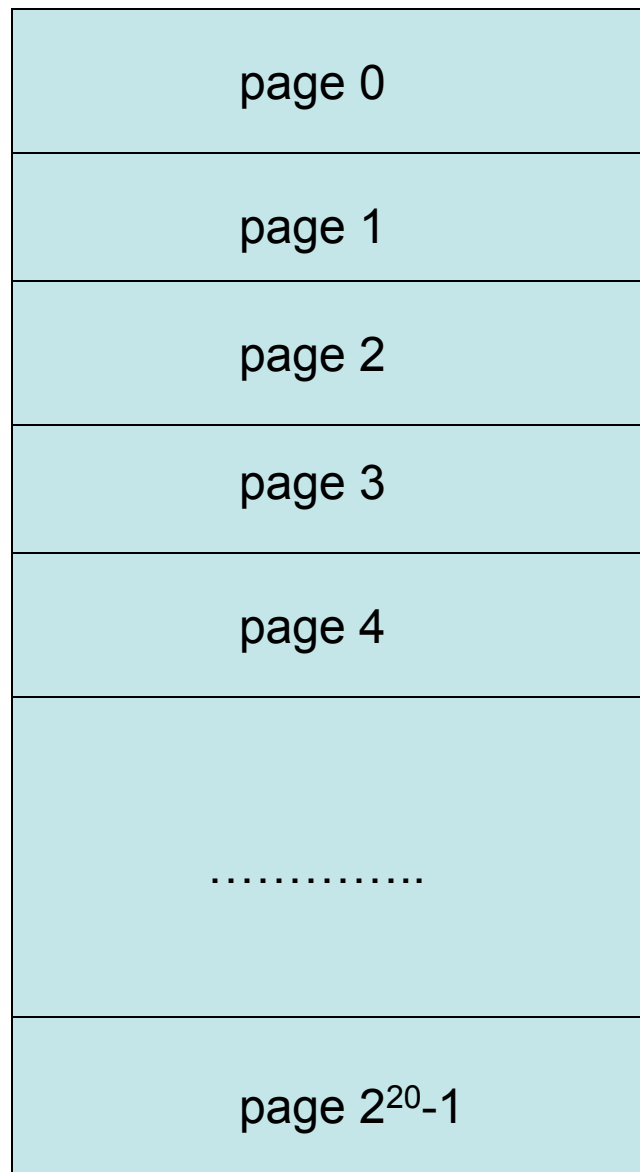
What is the size of the address space on a 64-bit processor?

- ❑ The physical memory is in general much smaller than the address space
- ❑ a small portion of the address space is generally in the RAM

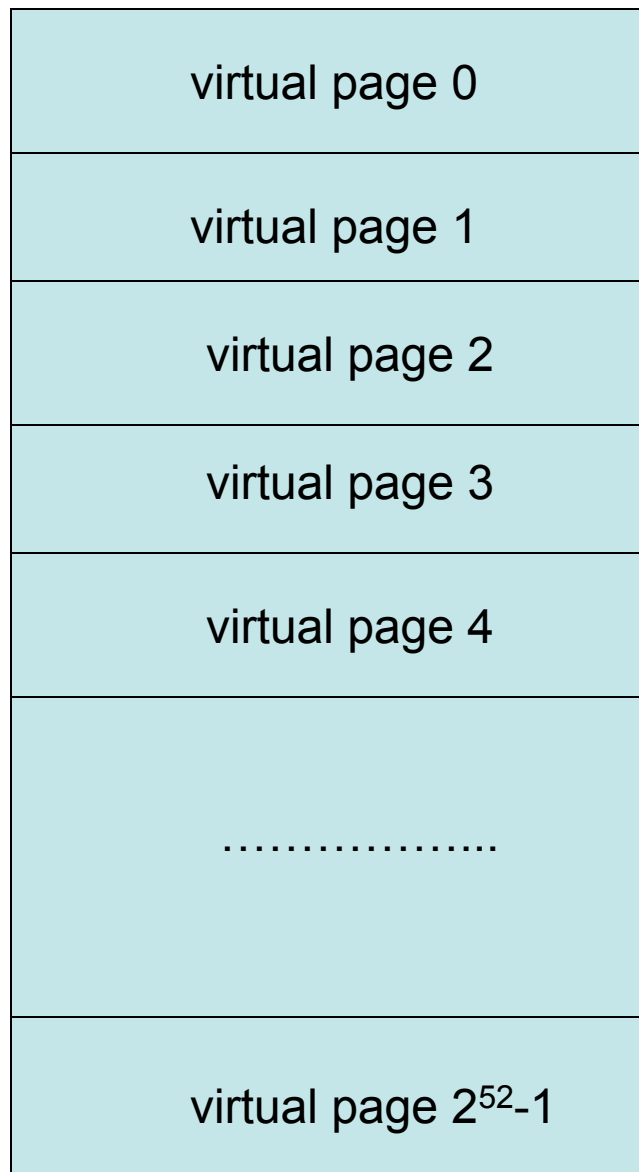
The RAM stores pages of the address space

- ❑ e.g., blocks of 4K

Physical Memory



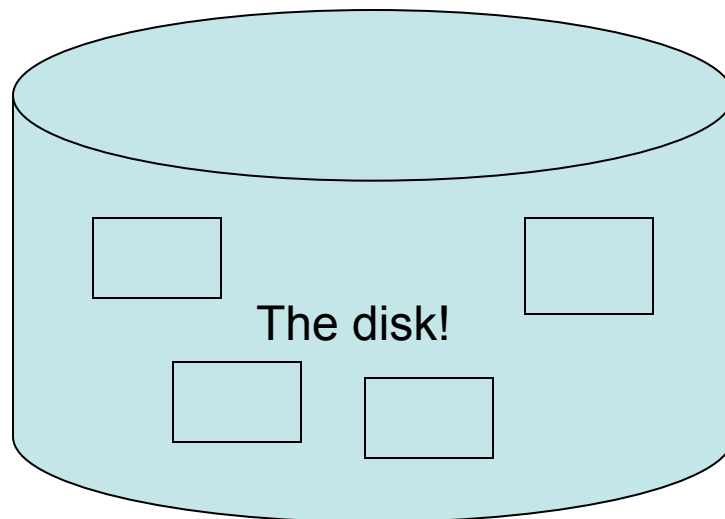
Address Space



Address Space

Where is the address space stored?

- On a disk which is large and slow



We do not want to use the disk for memory accesses

- it would be way to slow

Virtual Memory

Fundamental functionality

- bringing pages from disk to the RAM
- offloading pages from the RAM to disk
- translating virtual addresses into physical addresses

The disk can be seen as a simple ADT

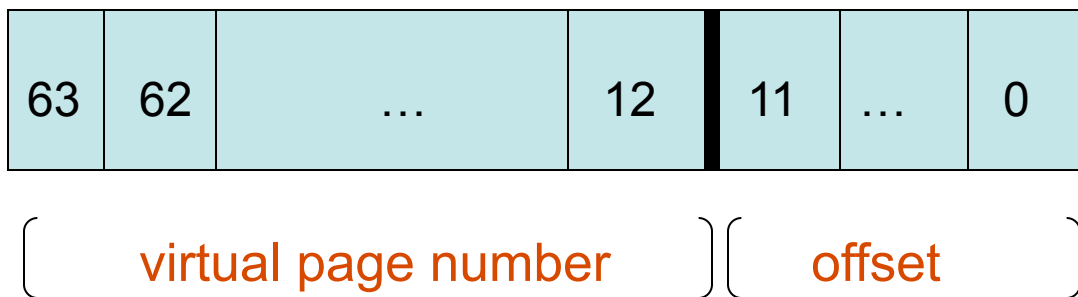
- get a disk page
- write a page to the disk

Functionalities

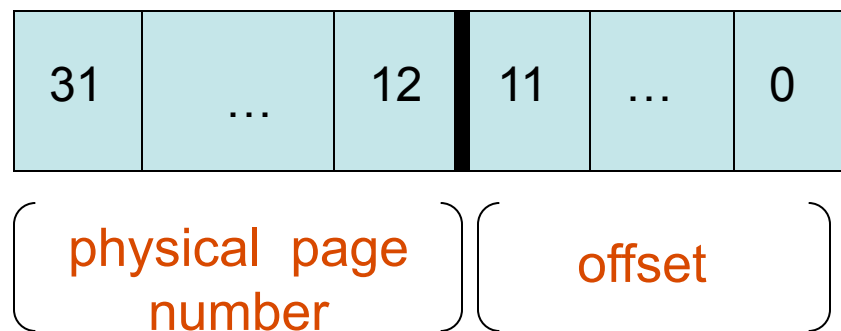
- demand paging

Address Translation

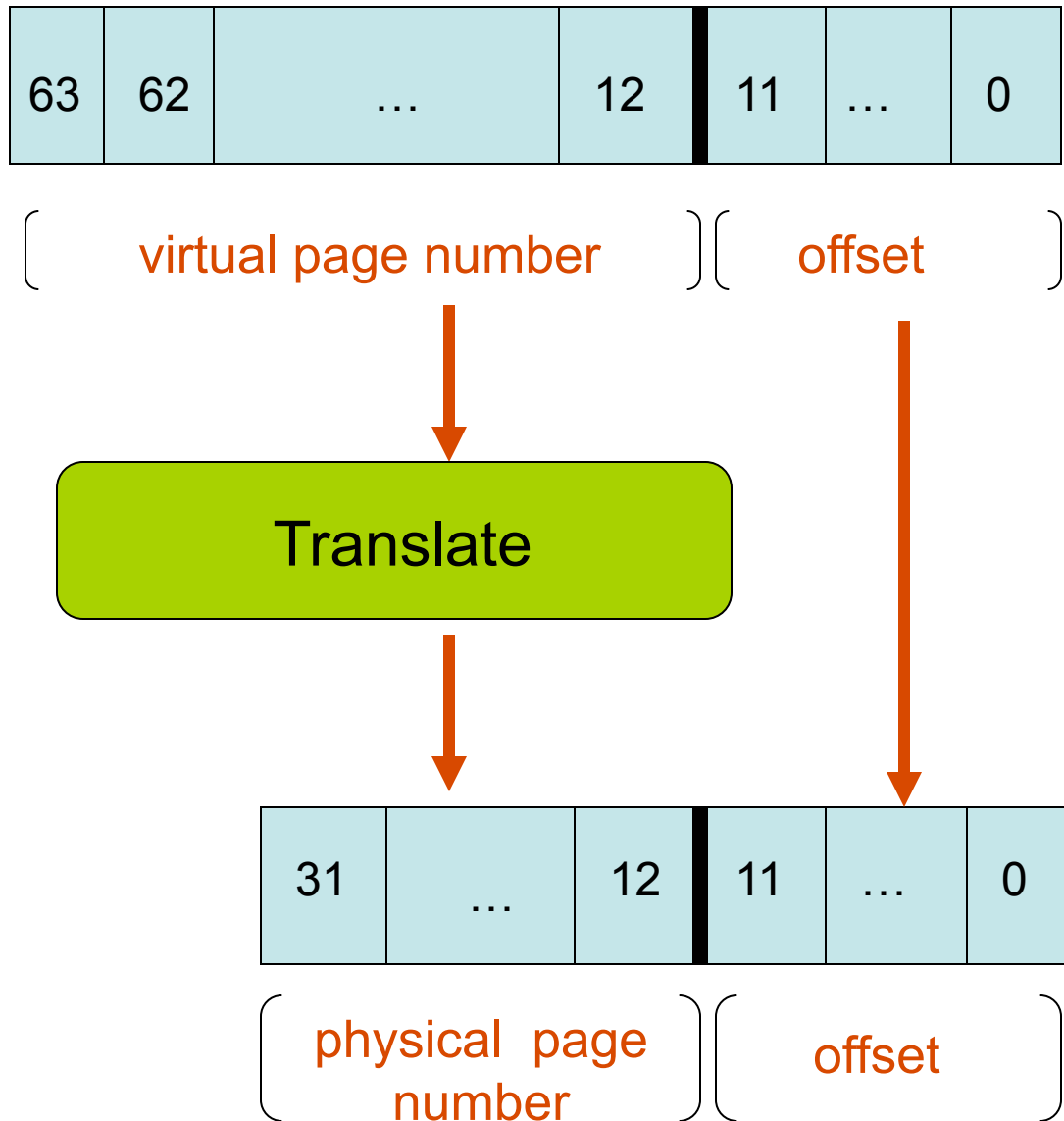
Virtual addresses (64-bit)



Physical address (32-bit)



Address Translation



Address Translation

Function

- ❑ translate the virtual page number into a physical page number

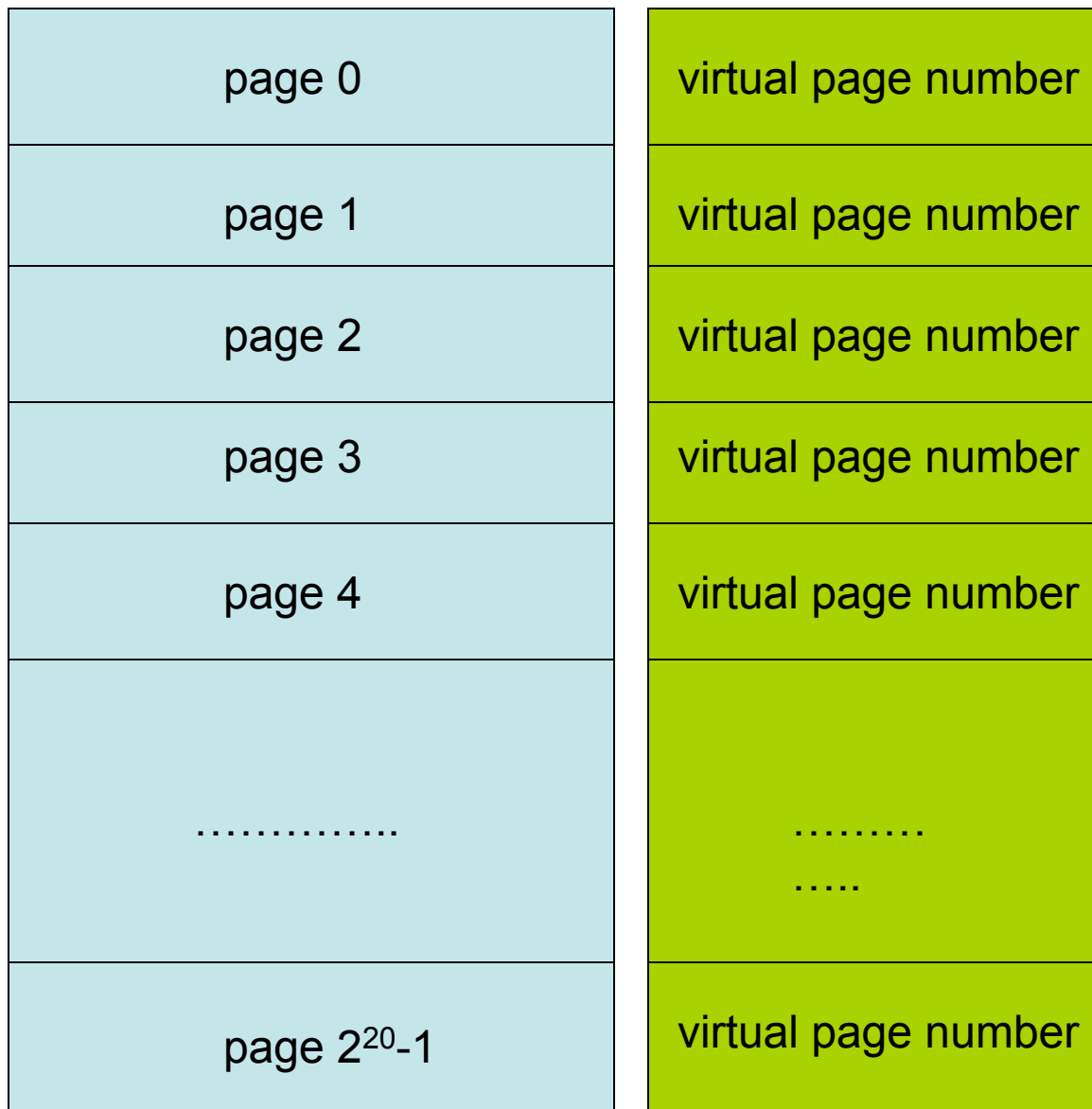
What do we need?

- ❑ If the page is already in memory, simply find out where it is.
- ❑ Otherwise, get the page from the disk and load it in memory

Note that

- ❑ a virtual page is either in memory or on the disk
- ❑ we will associate information with each physical page

Physical Memory



Paging in

We have time to think!

Replacement policies

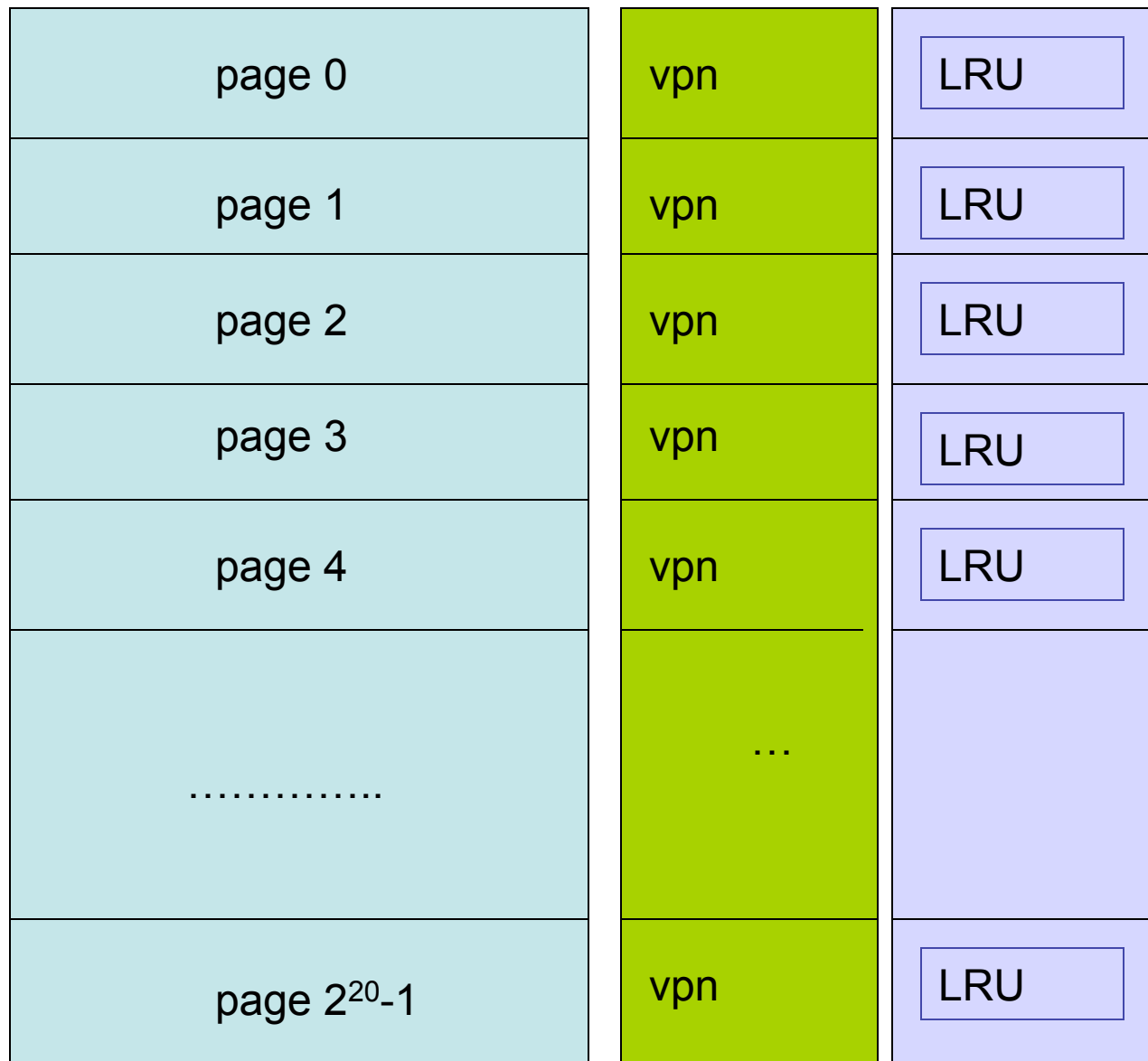
- least recently used (LRU)
- least frequently used (LFU)
- random

Most systems used LRU.

How to implement LRU?

- LRU data: when was the page last accessed
- The LRU info. needs not be accurate. It may just be one bit that is reset with some frequency.

Physical Memory



Writing

When a program changes one word or byte, it is too expensive to write through to disk.

- we must use a write back strategy

How to implement a write-back strategy

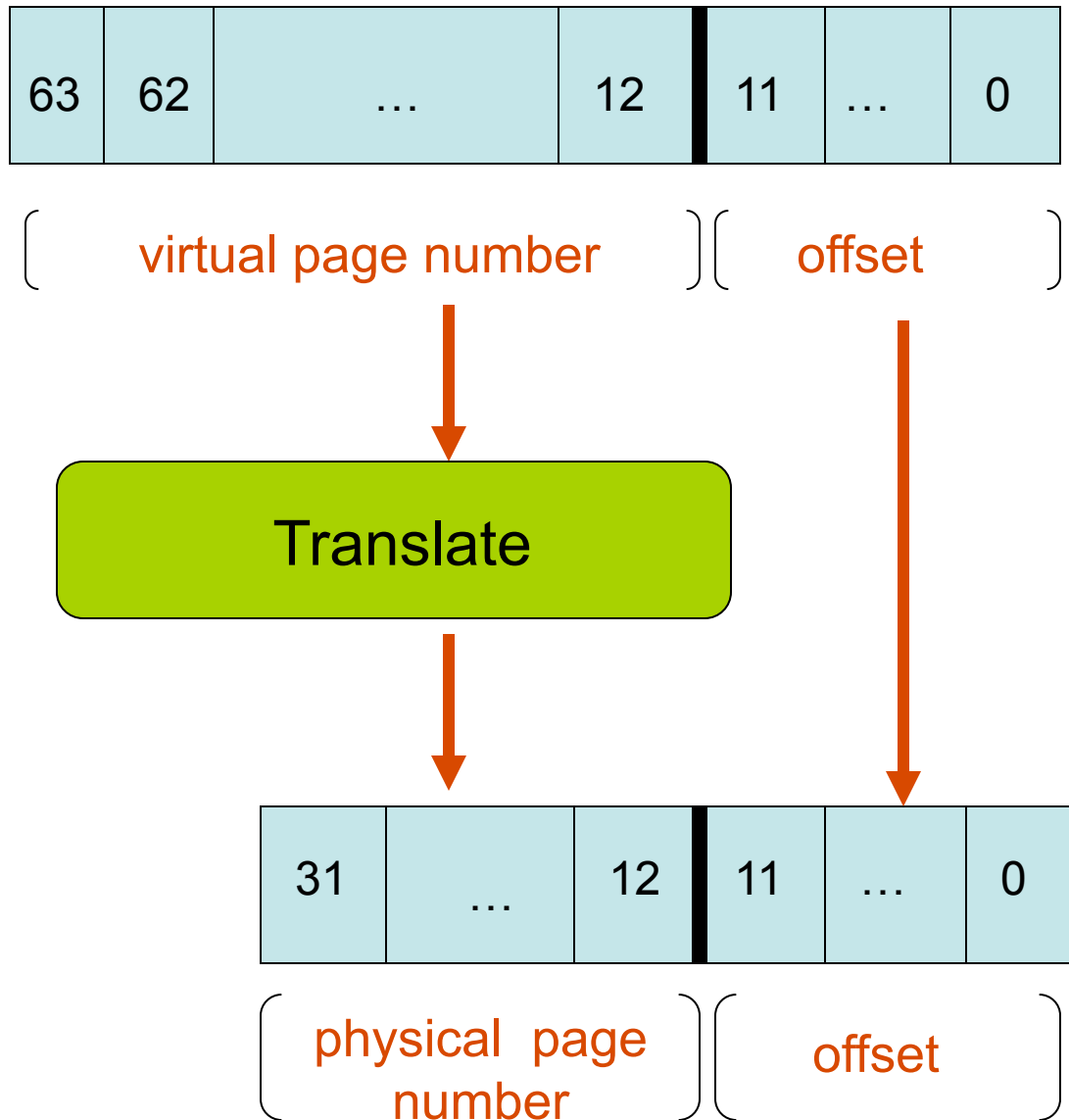
- When we write to a location in physical memory, we mark the page as dirty
- When you swap out a dirty page to make room for a new one, write the dirty page back to disk

We need to add a dirty bit

Physical Memory

page 0 (4K)	vpn	LRU	Dirty
page 1 (4K)	vpn	LRU	Dirty
page 2 (4K)	vpn	LRU	Dirty
page 3 (4K)	vpn	LRU	Dirty
page 4 (4K)	vpn	LRU	Dirty
.....	...		
page $2^{20}-1$ (4K)	vpn	LRU	Dirty

Address Translation



Address Translation

How do we do the translation?

- find out whether a virtual page is in memory or not
- linear in the number of physical pages

How many physical pages?

- 2^{20}
- Should we care?