Paging

Glenn Bruns
CSUMB

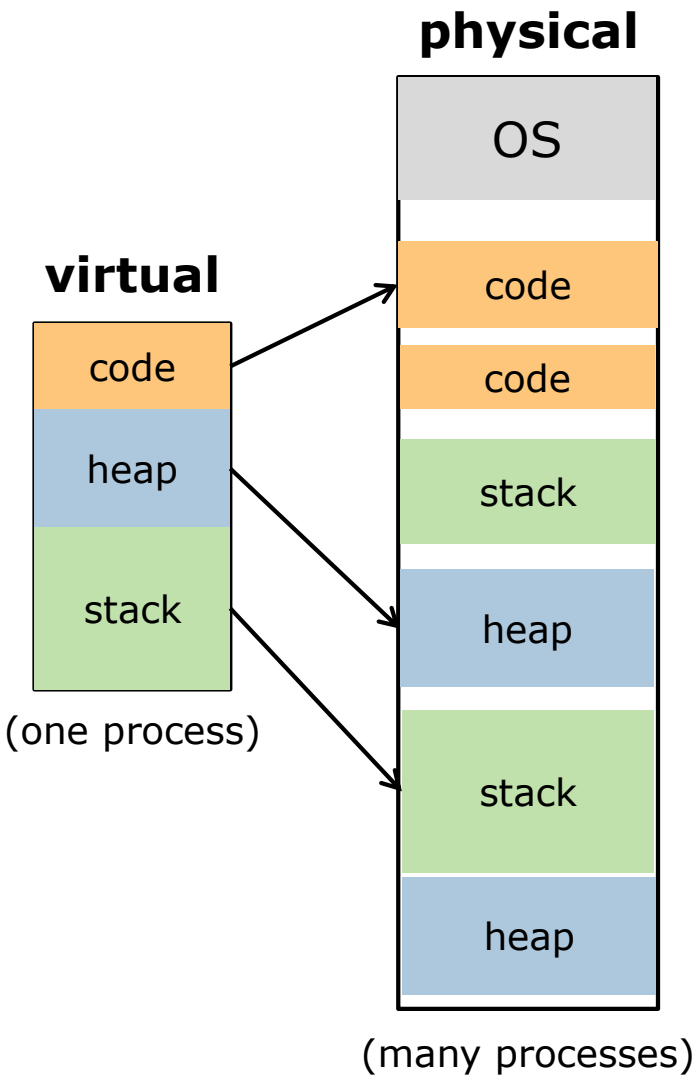
Lecture Objectives

After this lecture, you should be able to:

- Explain the advantages/disadvantages of memory virtualization with paging
- Manually perform address translation with page tables

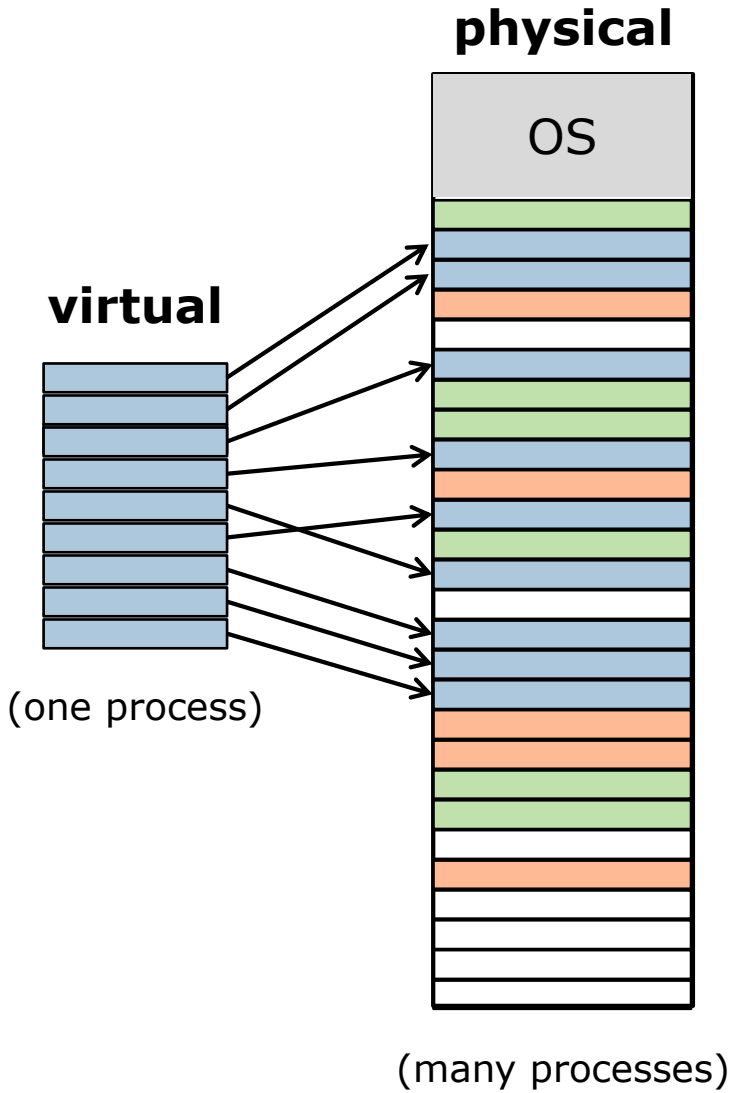
Recap: Segmentation

- Segmentation breaks the virtual address space up into **logical** pieces, of varying sizes
  - For example, one segment for heap
  - Can use base-and-bounds for address translation
- Good news:
  - segments can be given their own permission
  - more efficient use of physical memory space
- Bad news:
  - variable-sized segments leads to **memory fragmentation**



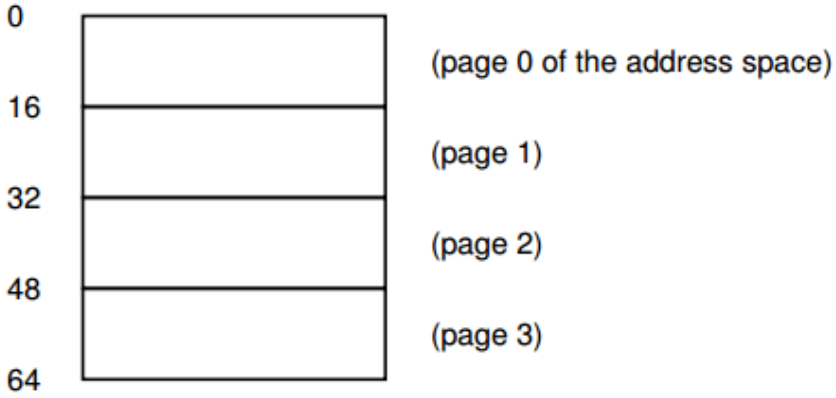
Paging

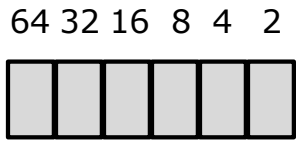
- An alternative to segmentation
- Breaks virtual address space up into pieces of equal size
- A process can have many pages
- Good news:
  - avoids fragmentation
  - simple, flexible
- Bad news:
  - space: big data structures needed for translation
  - speed: translation info won't fit into MMU registers
  - speed: translation not as simple



Example, part 1: virtual address space

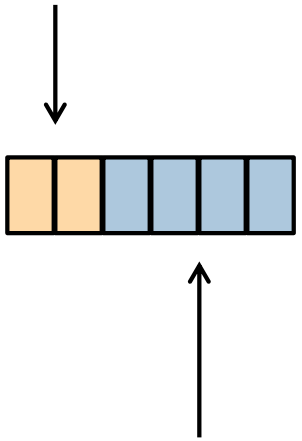
A tiny virtual address space of 64 bytes, **4 pages**





We need 6 bits to address 64 bytes

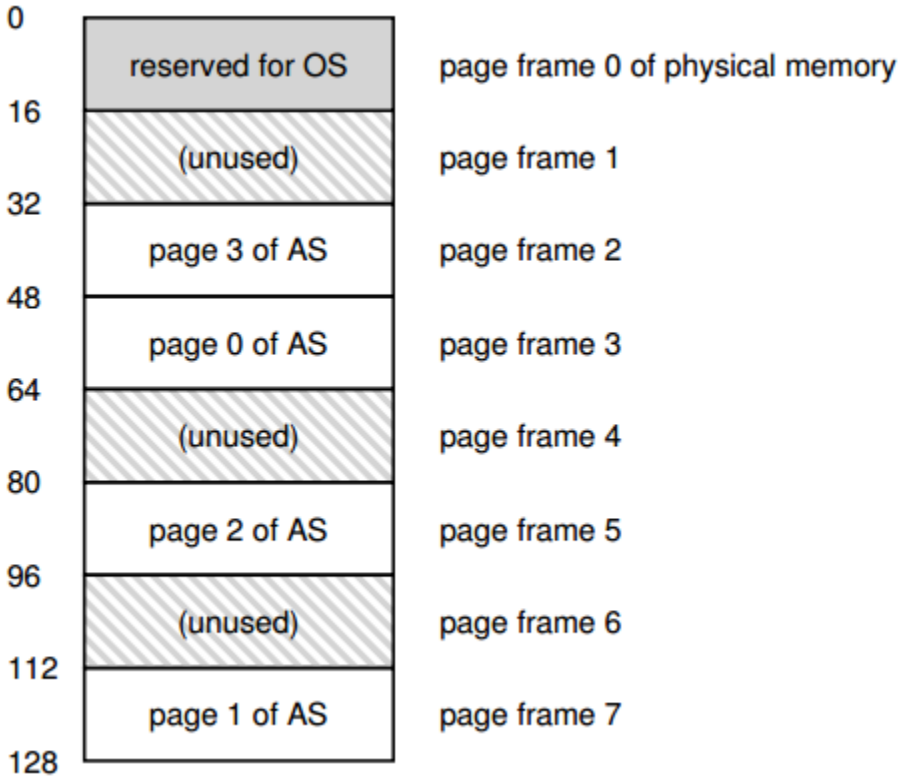
2 bits to address one of four pages



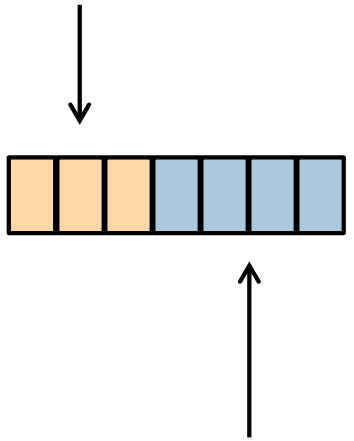
4 bits to address one of 16 bytes within page

Example, part 2: physical address space

A tiny physical address space of 128 bytes, 8 **page frames**

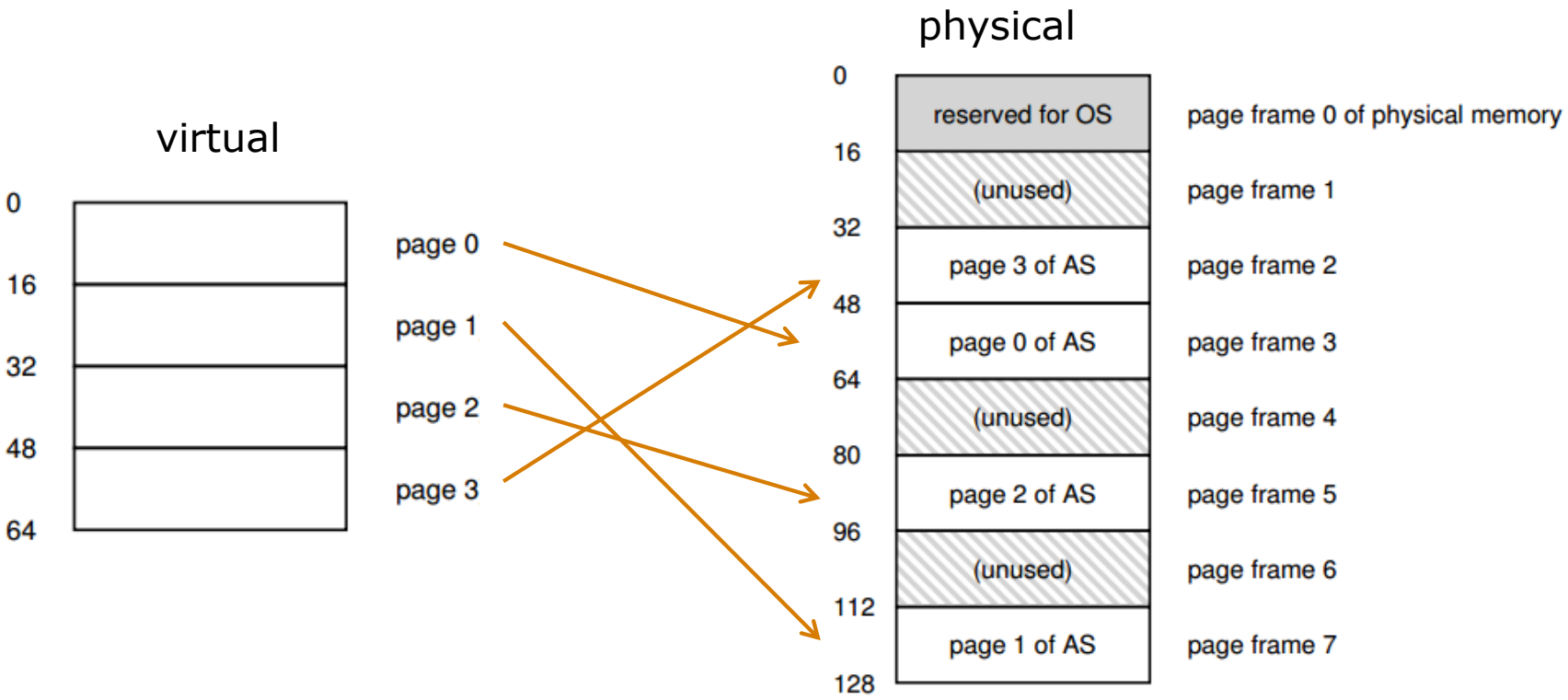


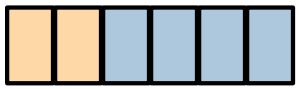
3 bits to address one of 8 page frames



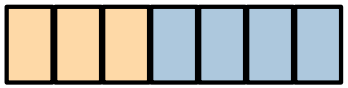
4 bits to address one of 16 bytes within page frame

Example: Page mapping





virtual address

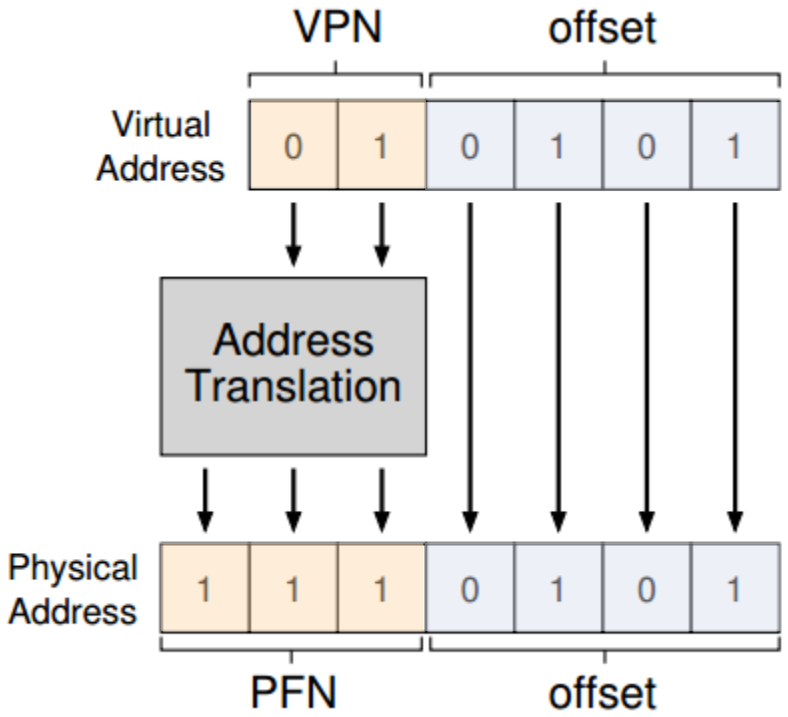


physical address

Example: Address translation

page table

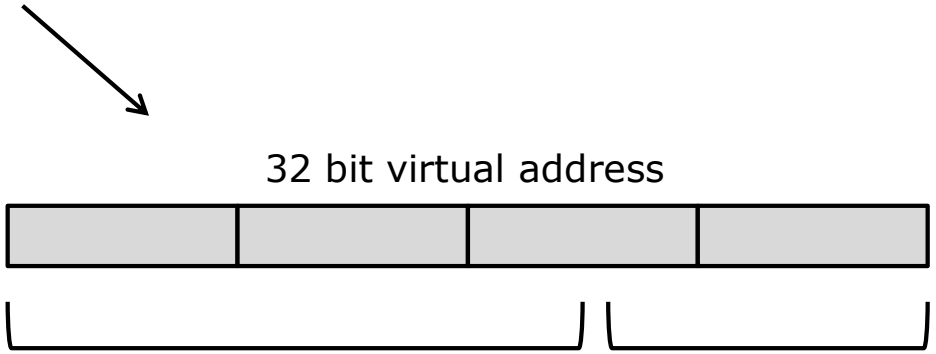
virtual page number	page frame number
00	011
01	111
10	101
11	010



Notice terminology: VPN and PFN

Another example

Virtual address space:
- 32 bits
- 4 KB pages



remaining 20 bits for page number

12 bits for offset

(memory aid: 10 bits = 1024)

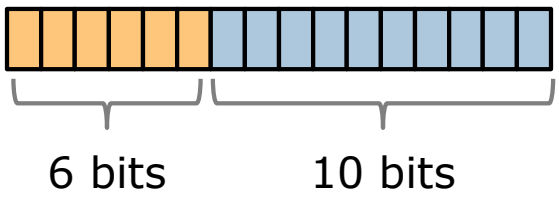
Exercise

Virtual address space:
- 16 bits
- 64 pages

What does a virtual address look like?

- how many bits for the virtual page numbers?
- how many bits for the offset?

For 64 pages we need 6 bits, so address has form:



A more realistic page table

page table

virtual page number	page frame number	valid	protection	present	dirty	reference
00	011	1	11	1	0	1
01	111	1	01	1	1	0
10	101	0	11	1	0	1
11	010	1	00	0	1	1

valid: is the translation valid?

protection: read/write/execute permissions

present: is page in physical memory or on disk?

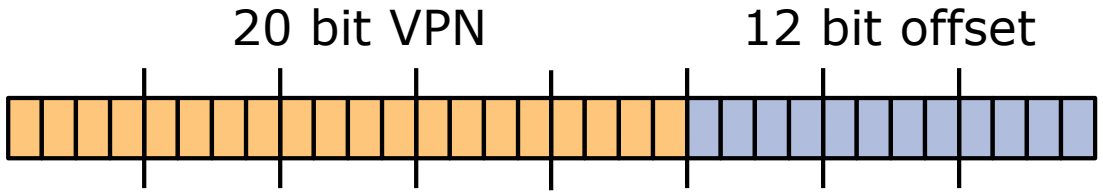
dirty: has page been modified since being brought into memory

reference: has page been accessed?

A problem with paging

Assume 32 bit virtual addresses, with 4 KB pages.

How many pages do we have?



(12 bits needed to address a page of 4 KB)

How big will the page table need to be?

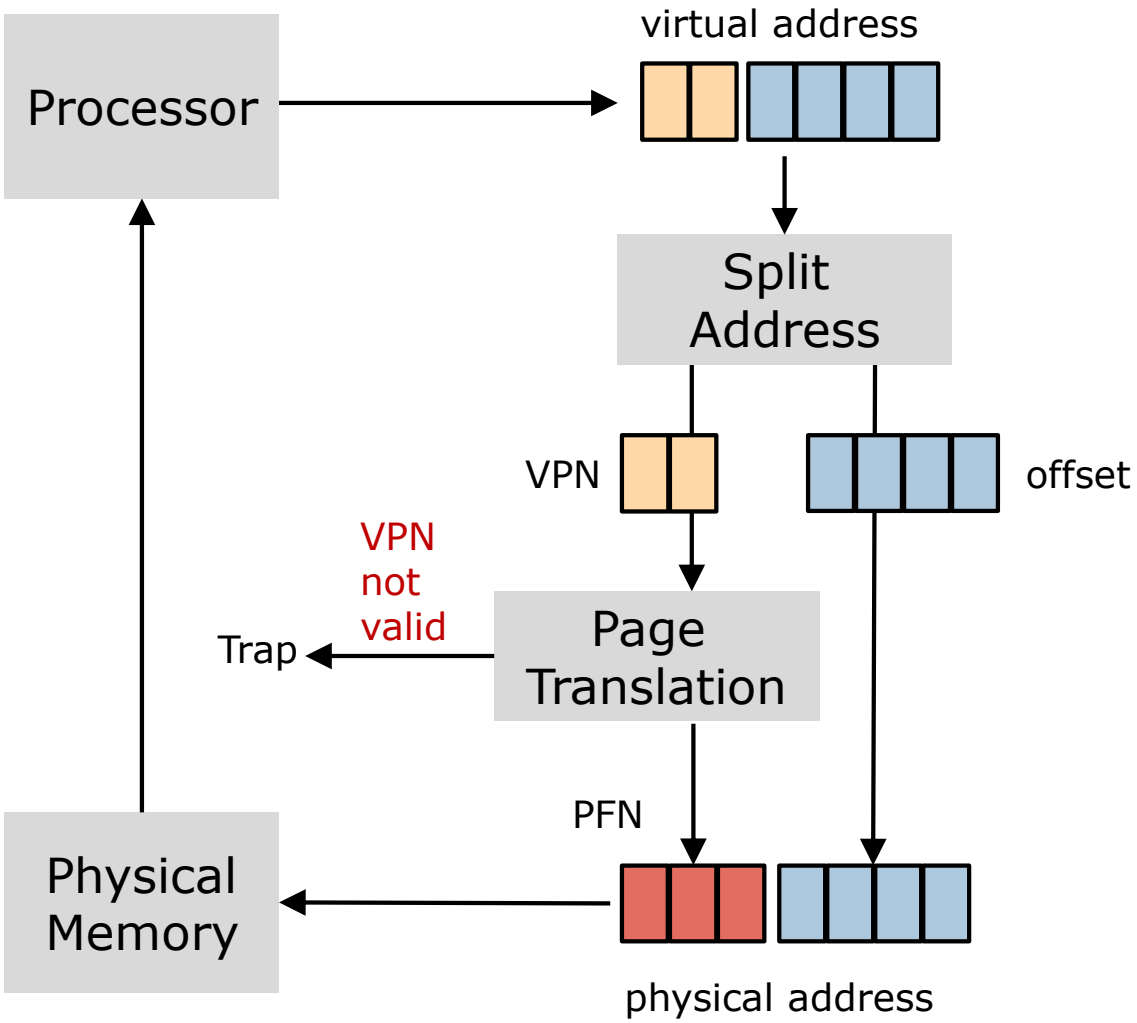
There are 2^{20} possible virtual page numbers.

(2^{20} is about 1 million)

If each row of page table is 4 bytes, page table will be 4 MB!

(what about 64 bit virtual addresses with 64 KB pages?)

Is paging fast?



1. get the VPN part of the virtual address
2. figure out where page table is located
3. use the VPN to index into the page table
4. if VPN not valid, trap
5. if VPN valid, get the PFN and combine with offset

Things to remember

- virtual page size = physical page frame size
- each process has its own page table
- with paging (unlike segmentation), the virtual address space is simple and linear

Summary

In paging, the virtual address space is split up into equal-sized pages.

A virtual address has two parts: virtual page number, and offset.

In address translation, a virtual page number is mapped to a page frame number.

Benefits of paging: flexible, eliminates memory fragmentation

Drawbacks: space (to store page table) and time (to map VPNs to PFNs)