Page-based Virtual Memory

VASpace

PA Space

Base

Offset

Fragmentation

Paging - break address up into same-sized chunks (4 KB)
Virtual page numbers (VPN)

Physical page numbers (PPN)

Physical page frames (PPF)

PFN

Fragmentation problem (with segmentation)

(not to scale)
How to translate from VA to PA

Virtual address → MMU → Physical address

Page table
- MMU looks up translation in page table
- One page table per process

VPN vP pPN

Present bit

VPN entries: Page table entry (PTE)
- Includes:
  - Data on disk
  - PPN mapping
  - Read permission

Valid entry?
- Write
- Execute

Present bit → is this on disk or in memory? Physical
Memory as large cache for disk
Virtual memory makes it possible

Page faults → OS look up & page is valid (possibly on disk)
→ OS will allocate a physical page → possibly moving a page to disk
→ OS will copy requested page into memory
→ OS updates the page table(s)

Swapping

Page faults → caused by "page misses"

Address MMU looks up VPN in the page table → in memory (valid + "page hit" present)
→ not in memory page miss
→ page fault
Where is the page table?
→ Stored in memory in hardware

![Diagram of address translation]

MMU has a pointer to the page table → physical address (CR3)
Each process has unique page table and unique PT pointer

Multi-level Page Table
Page table
4KB pages
2^2
VPN

CPU
0xDA010
VPN

MMU
0x40040010

VA space

PA space

VP PPN
F 1 5
E 1 7
P 1 8
C 1 4
A 1 3
B 1 4
S 9 0
X 7 1
4G Pages stack
Glob data
head

0x8A14 -> 0x6A14
4
VPN
Offset

Invalid entry -> Page fault
or not present

Frontend