Computer Architecture: Caches + Virtual Memory

Berk Sunar and Thomas Eisenbarth

ECE 505

WPI

Memory Hierarchy Design

- Basic Principles of Data Cache Chap. 2
- Six basic optimizations for caches Appendix B
- Ten advanced optimizations for caches

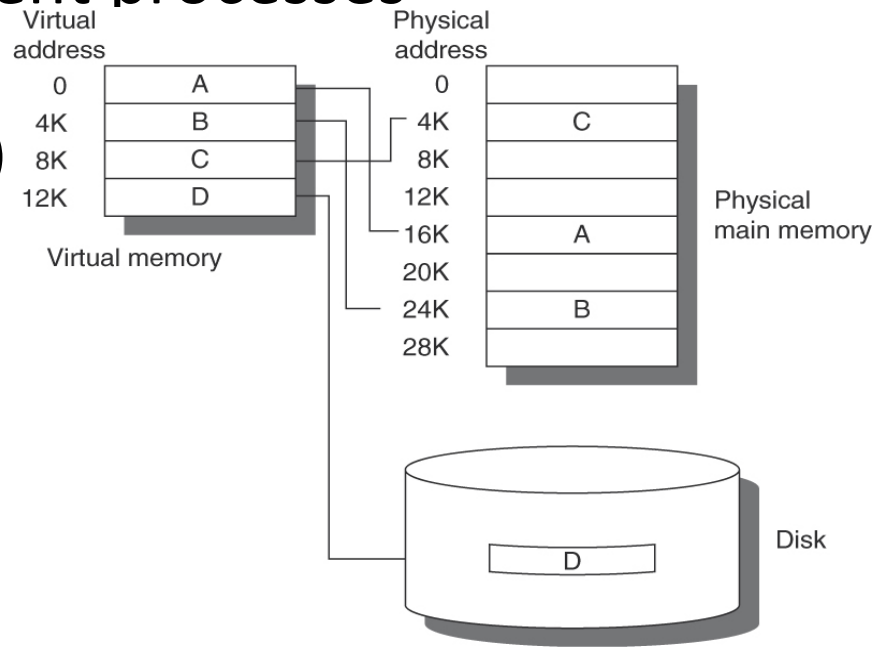
- Virtual Memory

- Virtual Machines

Virtual Memory

Motivation: several processes share same hardware, but need isolation (protection) *or* transparent extension of memory space using disk storage

- *Virtual memory* breaks physical memory into blocks and allocates them for different processes
  - Hides multi-layer memory (very similar to cache concept)
  - Makes program startup faster
  - Programs become location-independent



Virtual Memory vs Caches

- Memory *Pages* or *segments* instead of *blocks*
- *Page fault* or *address fault* instead of *miss*
- Requires *Address Translation* (or *memory mapping)* between virtual and physical addresses
  - Typically handled by a mix of hardware and software
  - *Context switches* between processes handled by *supervisor* (typically part of OS)

Parameter	First-Level Cache	Virtual Memory
Block/Page size	16-128 bytes	4kB – 64kB
Hit Time	1 – 3 cc	100 cc
Miss Penalty	8 – 200cc	1 – 10 million cc
Miss Rate	.1 – 10%	$10\uparrow-5$ $-10\uparrow-3$ %
Address Mapping	25-45 bit phys. Addr. To 14-20 bit cache addr.	32-64 bit virtual Addr. To 25-45 bit physical addr.

Virtual Memory vs Caches:

- Where can a block be placed in Physical Memory:
  - Blocks can be placed anywhere "fully associative"
- Which block is replaced on Memory miss?
  - Miss penalty is extremely high: do something clever
  - OS estimates Least Recently Used (LRU) policy
- What happens on write?
  - Due to slow access time to hard disk space:
    Write back (write through is impractical)
  - Often used with dirty bit to minimize write-backs

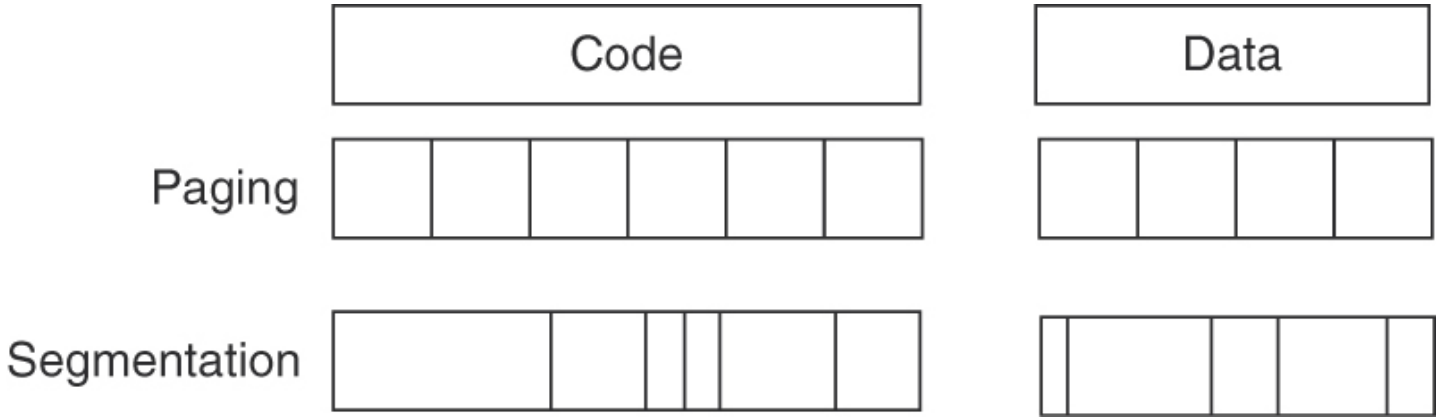
Memory Pages vs Segments

Pages: fixed-size memory blocks, typically 4kB or 8kB

- Simple approach: makes loading and moving pages easy
- Fragmentation due to unused space in pages

Segments: variable-size memory blocks

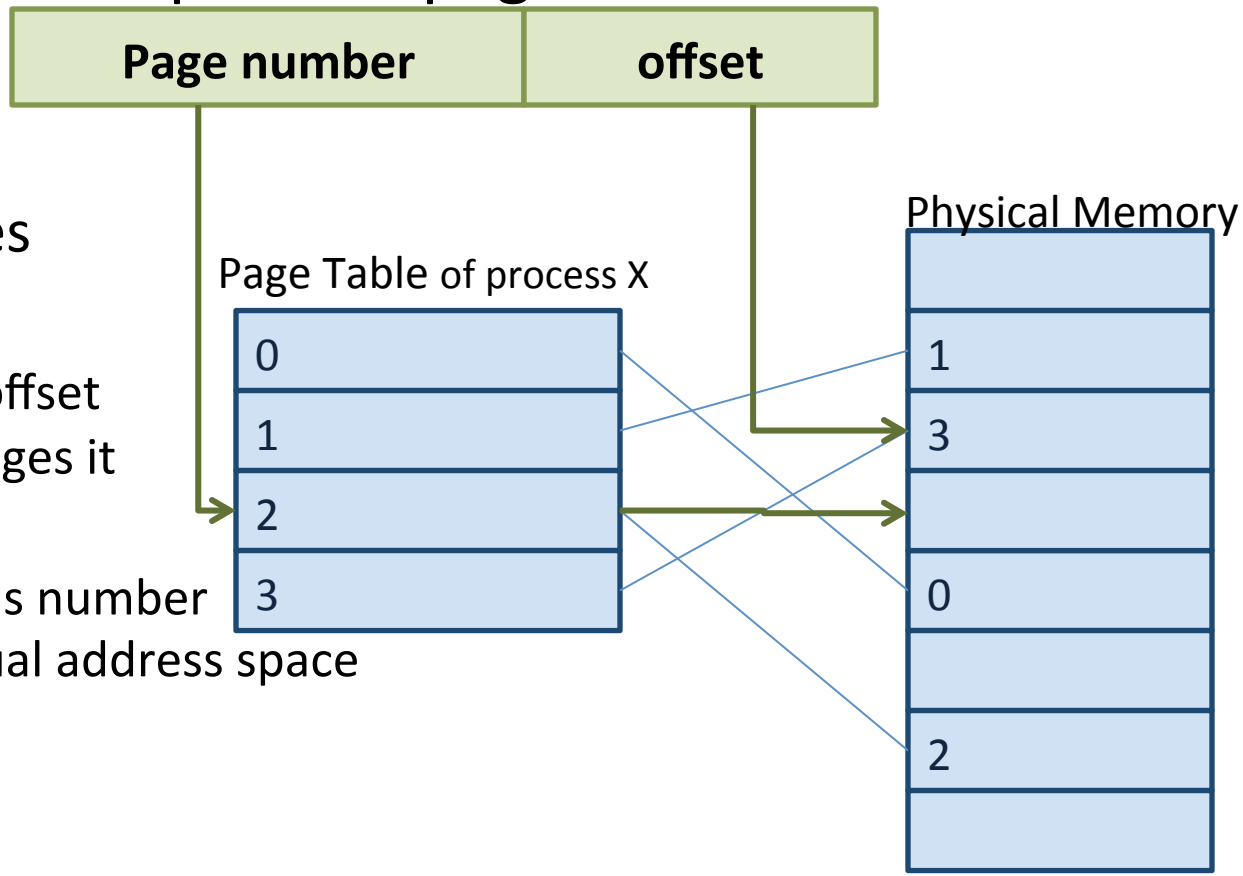
- Fragmentation due to unused space in physical memory
- Hybrid approach: paged segments (segment is multiple of page)



Page Table

CPUs virtual address split into page number and offset:

virtual address =



- Page Table stores physical address
  - For segments, offset is added. For pages it is concatenated
  - Page table size is number of pages in virtual address space

Page Table Size Example

Assume 32-bit virtual addresses, 4 KB pages, 4 byte page table entries (PTEs)

- $2\uparrow32 / 2\uparrow12 = 2\uparrow20$ PTEs i.e., 4 MB page table per user per process
- 4 GB of virtual address space

- Larger pages?
  - \+ Reduces TLB size, more memory can be mapped efficiently
  - – Internal fragmentation (Not all memory in page is used, roughly 1.5x page size wasted per process)
  - – Larger page fault penalty (more time to read from disk)

- How about 64-bit virtual address space?
  - – Even 1MB pages would require $2\uparrow44$ 8-byte PTEs (35 TB!)