Virtual Memory

Virendra Singh

Associate Professor

Computer Architecture and Dependable Systems Lab

Department of Electrical Engineering

Indian Institute of Technology Bombay

http://www.ee.iitb.ac.in/~viren/

E-mail: viren@ee.iitb.ac.in

CP-226: Computer Architecture

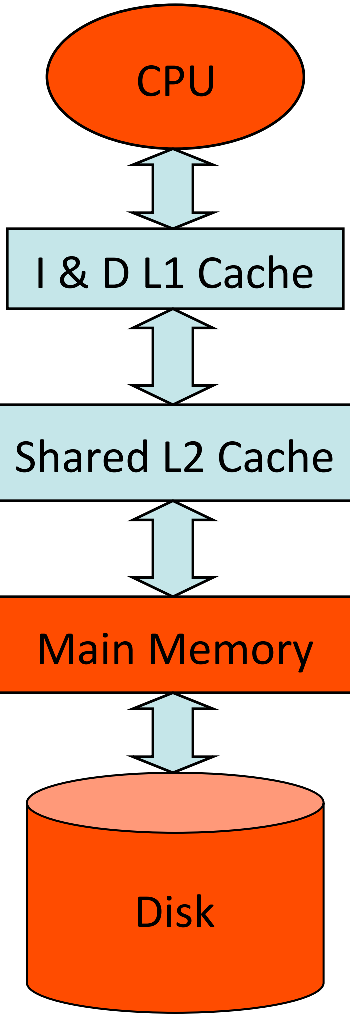
Lecture 21 (12 April 2013)

CADSL

Memory Hierarchy

Temporal Locality

- Keep recently referenced items at higher levels
- Future references satisfied quickly



Spatial Locality

- Bring neighbors of recently referenced to higher levels
- Future references satisfied quickly

Placement

Memory Type	Placement	Comments
Registers	Anywhere; Int, FP, SPR	Compiler/programmer manages
Cache (SRAM)	Fixed in H/W	*Direct-mapped,* *set-associative,* *fully-associative*
DRAM	Anywhere	O/S manages
Disk	Anywhere	O/S manages

Main Memory and Virtual Memory

- Use of virtual memory
  - Main memory becomes another level in the memory hierarchy
  - Enables programs with address space or working set that exceed physically available memory
    - No need for programmer to manage overlays, etc.
    - Sparse use of large address space is OK
  - Allows multiple users or programs to timeshare limited amount of physical memory space and address space

- Bottom line: efficient use of expensive resource, and ease of programming

Virtual Memory

- Enables
  - Use more memory than system has
  - Program can think it is the only one running
    - Don't have to manage address space usage across programs
    - E.g. think it always starts at address 0x0
  - Memory protection
    - Each program has private VA space: no-one else can clobber
  - Better performance
    - Start running a large program before all of it has been loaded from disk

Virtual Memory – Placement

- Main memory managed in larger blocks
  - *Page size* typically 4K – 16K
- Fully flexible placement; fully associative
  - ➢ Operating system manages placement
  - ➢ Indirection through *page table*
  - ➢ Maintain mapping between:
    - Virtual address (seen by programmer)
    - Physical address (seen by main memory)

Virtual Memory – Placement

- Fully associative implies expensive lookup?
  - In caches, yes: check multiple tags in parallel
- In virtual memory, expensive lookup is avoided by using a level of indirection
  - Lookup table or hash table
  - Called a *page table*

Virtual Memory – Identification

Virtual Address	Physical Address	Dirty bit
0x20004000	0x2000	Y/N

- Similar to cache tag array
  - Page table entry contains VA, PA, dirty bit
- Virtual address:
  - Matches programmer view; based on register values
  - Can be the same for multiple programs sharing same system, without conflicts
- Physical address:
  - Invisible to programmer, managed by O/S
  - Created/deleted on demand basis, can change

Virtual Memory – Replacement

- Similar to caches:
  - FIFO
  - LRU; overhead too high
    - Approximated with reference bit checks
    - Clock algorithm
  - Random
- O/S decides, manages

Virtual Memory – Write Policy

- Write back
  - Disks are too slow to write through

- Page table maintains dirty bit
  - Hardware must set dirty bit on first write
  - O/S checks dirty bit on eviction
  - Dirty pages written to backing store
    - Disk write, 10+ ms

Virtual Memory Implementation

- Caches have fixed policies, hardware FSM for control, pipeline stall

- VM has very different miss penalties
  - Remember disks are 10+ ms!

- Hence engineered differently

Thank You