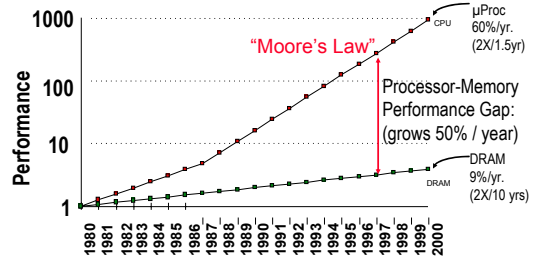


# Lecture 13 Memory Hierarchy

Instructor: Jun Yang

# Motivation

Processor-DRAM Memory Gap (latency)

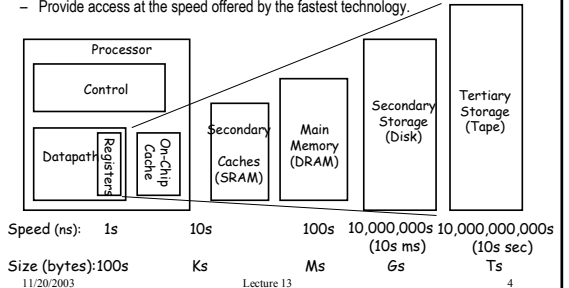


# The Goal: Illusion of Large, Fast, Cheap Memory

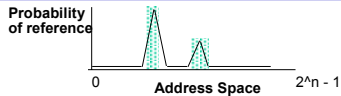
- Fact:
  - Large memories are slow
  - Fast memories are small
- How do we create a memory that is large, cheap and fast (most of the time)?
  - Hierarchy

# Memory Hierarchy of a Modern Computer System

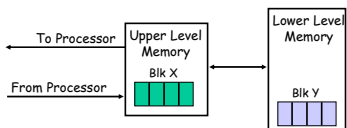
- By taking advantage of the principle of locality:
  - Present the user with as much memory as is available in the cheapest technology.
  - Provide access at the speed offered by the fastest technology.



# Memory Hierarchy: Why Does it Work? Locality!



- **Temporal Locality (Locality in Time):**
  - => Keep most recently accessed data items closer to the processor
- **Spatial Locality (Locality in Space):**
  - => Move blocks consists of contiguous words to the upper levels



# Memory Hierarchy Technology

- **Random Access:**
  - "Random" is good: access time is the same for all locations
  - **DRAM:** Dynamic Random Access Memory
    - ❖ High density, low power, cheap, slow
    - ❖ Dynamic: need to be "refreshed" regularly
  - **SRAM:** Static Random Access Memory
    - ❖ Low density, high power, expensive, fast
    - ❖ Static: content will last "forever"(until lose power)
- **"Not-so-random" Access Technology:**
  - Access time varies from location to location and from time to time
  - Examples: Disk, CDROM
- **Sequential Access Technology:** access time linear in location (e.g., Tape)
- We will concentrate on random access technology
  - The Main Memory: DRAMs + Caches: SRAMs

# Introduction to Caches

- Cache
  - is a small very fast memory (SRAM, expensive)
  - contains copies of the most recently accessed memory locations (data and instructions): **temporal locality**
  - is fully managed by hardware (unlike virtual memory)
  - storage is organized in **blocks (lines)** of contiguous memory locations: **spatial locality**
  - unit of transfer to/from main memory (or L2) is the cache block
- General structure
  - $n$  blocks per cache organized in  $s$  sets
  - $b$  bytes per block
  - total cache size  $n \cdot b$  bytes

11/20/2003

Lecture 13

7

# Caches

- For each block:
  - an address tag: unique identifier
  - state bits:
    - ⚡ (n)valid
    - ⚡ Modified (or dirty)
  - the data:  $b$  bytes
- Basic cache operation
  - every memory access is first presented to the cache
  - **hit**: the word being accessed is in the cache, it is returned to the cpu
  - **miss**: the word is not in the cache,
    - ⚡ a whole block is fetched from memory (L2)
    - ⚡ an "old" block is evicted from the cache (kicked out), which one?
    - ⚡ the new block is stored in the cache
    - ⚡ the requested word is sent to the cpu

11/20/2003

Lecture 13

8

# Caches

- Terminology
  - hit (miss) rate: fraction of references that hit (miss) in the cache
  - hit time: access time of a hit in the cache
  - miss penalty, includes:
    - ⚡ access time to memory (L2)
    - ⚡ replacement time (was the evicted block modified?)
    - ⚡ access to requested word
- Avg. Memory Access Time
  - $t_{avg} = t_{hit} + mr \times t_{miss}$
  - e.g.  $t_{hit} = 1, mr = 0.1, t_{miss} = 100$
  - $t_{avg} = 11$
- Main design issues
  - block identification
    - ⚡ which is the right block
  - block placement
    - ⚡ where to put the new block
  - block replacement
    - ⚡ which block to evict
  - write strategy
    - ⚡ update L1 only, or both L1 & L2
  - allocation strategy
    - ⚡ what to do on a **write miss**?
  - block transfer
    - ⚡ fetch the whole block first, or the requested word?
  - consistency
    - ⚡ what about the DMA?
    - ⚡ see write strategy

11/20/2003

Lecture 13

9

# Block Placement and Identification

- Placement & Identification – determined by the addresses
  - **Direct mapped cache**: one block per set.



- **Set-associative mapping**:  $n/s$  blocks per set.



- **Fully associative mapping**: one set per cache ( $s = n$ ).



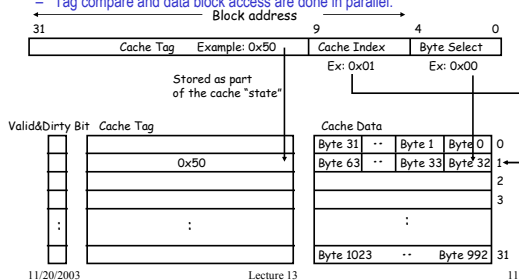
11/20/2003

Lecture 13

10

# Example: 1 KB Direct Mapped Cache with 32 B Blocks

- For a  $2^N$  byte cache:
  - The uppermost  $(32 - N)$  bits are always the Cache Tag
  - The lowest  $M$  bits are the Byte Select (Block Size =  $2^M$ )
  - Tag compare and data block access are done in parallel.



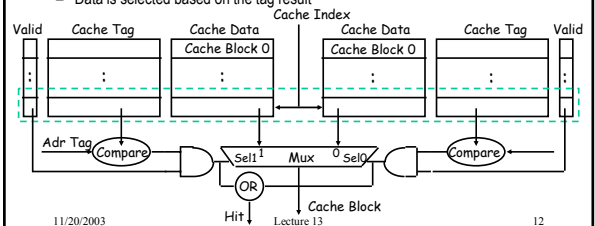
11/20/2003

Lecture 13

11

# Set Associative Cache

- **N-way set associative**:  $N$  entries for each Cache Index
  - $N$  direct mapped caches operates in parallel
- Example: Two-way set associative cache
  - Cache Index selects a "set" from the cache
  - The two tags in the set are compared to the input in parallel
  - Data is selected based on the tag result



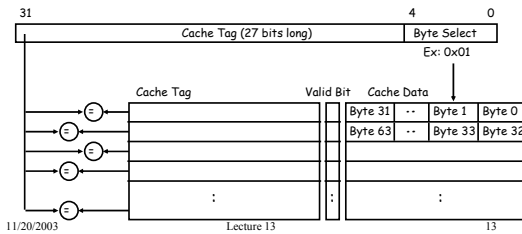
11/20/2003

Lecture 13

12

## Example: Fully Associative

- Fully Associative Cache
  - Forget about the Cache Index
  - Compare the Cache Tags of all cache entries in parallel
  - Example: Block Size = 32 B blocks, we need N 27-bit comparators
- By definition: Conflict Miss = 0 for a fully associative cache



11/20/2003

Lecture 13

13

## Block Replacement

- Block replacement
  - not an issue with direct mapped
  - replacement strategy is more important in small caches than in large ones (see Fig. 5.6).
  - replacement policies:
    - ❖ LRU: has been unused for the longest time. good temporal locality, complex state
    - ❖ pseudo-random: randomly selected
    - ❖ FIFO: oldest block. Approximation to LRU.

11/20/2003

Lecture 13

14

## Cache Organization – Example

- Instruction & Data caches:
  - Because instructions have much lower miss rates (see Fig 5.8): split the cache (at L1 level) into instruction and data. Otherwise, unified.
  - Main advantages: no interference, data misses do not stall instruction fetch, optimize for different access patterns etc.

### • DEC Alpha 21264 Cache

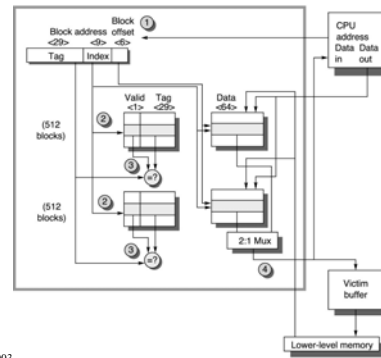
- 64K I & D caches (L1), 2-way set associative;
- b = 64 bytes;
- write-back; write allocate on write miss.
- Organization shown in Fig. 5.7: data RAM is 64 bytes wide (no multiplexer), 3 bits of offset are used (with index) to select a 64 bit word in the block.
- Read & write hits: 3 cycles, pipelined.
- Victim buffer: 8 entries.
- Fig. 5.43 (pp. 483) for the entire hierarchy – read in your textbook!

11/20/2003

Lecture 13

15

## Alpha 21264 Cache Organization



11/20/2003

Lecture 13

16

## Cache Design Tradeoffs & Characteristics

### • Design Parameters

- cache size, block size, associativity

### • Cache Size

- bigger: exploits temporal locality better
- but is slower: slower cycle time

### • Associativity

- lower: shorter access time – faster
- Hit time in d-m is shorter than in s-a.
- higher: slower but higher miss rate
- Rule 1: miss rate of 8-way s-a  $\cong$  fully s-a.
- Rule 2: miss rate of d-m of size N  $\cong$  that of a 2-way s-a of size N/2.

### • Operation

- Writes are slower than reads

### • Block Size

- spatial locality within a block
- too small: too many transfers of data from memory
- too large: wasted bandwidth, unused transferred data
- large block size reduces compulsory misses, taking advantage of spatial locality;
- not always good: increased miss penalty
- tradeoff: latency, bandwidth and block size.

11/20/2003

Lecture 13

17