Multilevel Memories

Joel Emer
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology

Based on the material prepared by
Krste Asanovic and Arvind
Performance of high-speed computers is usually limited by memory bandwidth & latency

- **Latency** (time for a single access)
  Memory access time >> Processor cycle time

- **Bandwidth** (number of accesses per unit time)
  if fraction $m$ of instructions access memory,
  ⇒ $1+m$ memory references / instruction
  ⇒ CPI = 1 requires $1+m$ memory refs / cycle
Core Memory

- Core memory was first large scale reliable main memory
  - invented by Forrester in late 40s at MIT for Whirlwind project
- Bits stored as magnetization polarity on small ferrite cores threaded onto 2 dimensional grid of wires
- Coincident current pulses on X and Y wires would write cell and also sense original state (destructive reads)
- Robust, non-volatile storage
- Used on space shuttle computers until recently
- Cores threaded onto wires by hand (25 billion a year at peak production)
- Core access time $\sim 1\mu s$

Image removed due to copyright restrictions.

DEC PDP-8/E Board, 4K words x 12 bits, (1968)

October 3, 2005
Semiconductor Memory, DRAM

- Semiconductor memory began to be competitive in early 1970s
  - Intel formed to exploit market for semiconductor memory

- First commercial DRAM was Intel 1103
  - 1Kbit of storage on single chip
  - charge on a capacitor used to hold value

- Semiconductor memory quickly replaced core in 1970s
One Transistor Dynamic RAM

1-T DRAM Cell

Explicit storage capacitor (FET gate, trench, stack)

TiN top electrode ($V_{REF}$)

$\text{Ta}_2\text{O}_5$ dielectric

Poly word line

$W$ bottom electrode

Access fet

Image removed due to copyright restrictions.

TiN/Ta2O5/W Capacitor

October 3, 2005
Processor-DRAM Gap (latency)

"Moore’s Law"

Processor-Memory Performance Gap: (grows 50% / year)

μProc 60%/year

CPU

DRAM 7%/year

Four-issue superscalar could execute 800 instructions during cache miss!

October 3, 2005
Little’s Law

Throughput \( T \) = Number in Flight \( N \) / Latency \( L \)

Example:

--- Assume infinite bandwidth memory
--- 100 cycles / memory reference
--- 1 + 0.2 memory references / instruction

⇒ Table size = 1.2 \( \times \) 100 = 120 entries

120 independent memory operations in flight!
DRAM Architecture

- Bits stored in 2-dimensional arrays on chip
- Modern chips have around 4 logical banks on each chip
  - each logical bank physically implemented as many smaller arrays

October 3, 2005
DRAM Operation

Three steps in read/write access to a given bank

- **Row access (RAS)**
  - decode row address, enable addressed row (often multiple Kb in row)
  - bitlines share charge with storage cell
  - small change in voltage detected by sense amplifiers which latch whole row of bits
  - sense amplifiers drive bitlines full rail to recharge storage cells

- **Column access (CAS)**
  - decode column address to select small number of sense amplifier latches (4, 8, 16, or 32 bits depending on DRAM package)
  - on read, send latched bits out to chip pins
  - on write, change sense amplifier latches which then charge storage cells to required value
  - can perform multiple column accesses on same row without another row access (burst mode)

- **Precharge**
  - charges bit lines to known value, required before next row access

Each step has a latency of around 20ns in modern DRAMs

Various DRAM standards (DDR, RDRAM) have different ways of encoding the signals for transmission to the DRAM, but all share the same core architecture

October 3, 2005
Multilevel Memory

Strategy: **Hide** latency using small, fast memories called caches.

Caches are a mechanism to hide memory latency based on the empirical observation that the patterns of memory references made by a processor are often highly predictable:

```
...  
loop: ADD r2, r1, r1  100
SUBI r3, r3, #1  104
BNEZ r3, loop  108
...  112
```

What is the pattern of instruction memory addresses?
Typical Memory Reference Patterns

- Instruction fetches
- Stack accesses
- Data accesses

Address

Time

n loop iterations
linear sequence

October 3, 2005
Common Predictable Patterns

Two predictable properties of memory references:

- **Temporal Locality**: If a location is referenced it is likely to be referenced again in the near future.

- **Spatial Locality**: If a location is referenced it is likely that locations near it will be referenced in the near future.
Caches exploit both types of predictability:

- Exploit **temporal locality** by remembering the contents of recently accessed locations.

- Exploit **spatial locality** by fetching blocks of data around recently accessed locations.
Memory Hierarchy

- **size**: Register << SRAM << DRAM  
  - why?
- **latency**: Register << SRAM << DRAM  
  - why?
- **bandwidth**: on-chip >> off-chip  
  - why?

On a data access:
- *hit* (data ∈ fast memory) ⇒ low latency access
- *miss* (data ∉ fast memory) ⇒ long latency access (DRAM)

*Fast mem. effective only if bandwidth requirement at B << A*
Management of Memory Hierarchy

- Small/fast storage, e.g., registers
  - Address usually specified in instruction
  - Generally implemented directly as a register file
    • but hardware might do things behind software’s back, e.g., stack management, register renaming

- Large/slower storage, e.g., memory
  - Address usually computed from values in register
  - Generally implemented as a cache hierarchy
    • hardware decides what is kept in fast memory
    • but software may provide “hints”, e.g., don’t cache or prefetch
A Typical Memory Hierarchy c.2003

Split instruction & data primary caches (on-chip SRAM)

CPU

RF

L1 Instruction Cache

L1 Data Cache

Unified L2 Cache

Multiported register file (part of CPU)

Large unified secondary cache (on-chip SRAM)

Multiple interleaved memory banks (DRAM)

Memory

Memory

Memory

Memory

October 3, 2005
Workstation Memory System (Apple PowerMac G5, 2003)

Image removed due to copyright restrictions.
To view image, visit http://www.apple.com/powermac/pciexpress.html

- Dual 2GHz processors, each with 64KB I-cache, 32KB D-cache, and 512KB L2 unified cache
- 1GB/s @ 1GHz, 2x32-bit bus, 16GB/s
- North Bridge Chip
- Up to 8GB DRAM, 400MHz, 128-bit bus, 6.4GB/s
- AGP Graphics Card, 533MHz, 32-bit bus, 2
- PCI-X Expansion, 133MHz, 64-bit bus, 1 GB/s

October 3, 2005
Five-minute break to stretch your legs
Inside a Cache

100
Data Byte

304
Data Byte

6848

Address

Tag

Line

Data Block

Address

Copy of main memory location 100

Copy of main memory location 101

Processor

CACHE

Main Memory

Data

October 3, 2005
Cache Algorithm (Read)

Look at Processor Address, search cache tags to find match. Then either

- Found in cache
  - a.k.a. HIT
  - Return copy of data from cache

- Not in cache
  - a.k.a. MISS
  - Read block of data from Main Memory
  - Wait ...
  - Return data to processor and update cache

Q: Which line do we replace?
Placement Policy

<table>
<thead>
<tr>
<th>Block Number</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2 3 4 5 6 7 8 9</td>
<td>1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 3 3</td>
</tr>
</tbody>
</table>

Set Number

Cache

- **Fully Associative**
- **(2-way) Set Associative**
- **Direct Mapped**

block 12 can be placed anywhere

anywhere in set 0 (12 mod 4)

only into block 4 (12 mod 8)
Direct-Mapped Cache
Direct Map Address Selection

higher-order vs. lower-order address bits

Index | Tag | Block Offset

TagV = Block + Offset

2^k lines

Data Block

HIT

Data Word or Byte

October 3, 2005
2-Way Set-Associative Cache

Block Offset

b

t

k

V Tag

Data Block

V Tag

Data Block

Data Word or Byte

HIT

October 3, 2005
Fully Associative Cache

Diagram showing the structure of a fully associative cache with inputs for Tag, Block Offset, and Output for HIT and Data Word or Byte.
Replacement Policy

In an associative cache, which block from a set should be evicted when the set becomes full?

• Random

• Least Recently Used (LRU)
  • LRU cache state must be updated on every access
  • true implementation only feasible for small sets (2-way)
  • pseudo-LRU binary tree often used for 4-8 way

• First In, First Out (FIFO) a.k.a. Round-Robin
  • used in highly associative caches

• Not Least Recently Used (NLRU)
  • FIFO with exception for most recently used block

This is a second-order effect. Why?
Block Size and Spatial Locality

Block is the unit of transfer between the cache and memory.

- **Tag**
- **Word0**
- **Word1**
- **Word2**
- **Word3**

4 word block, \( b = 2 \)

Split CPU address:

- **block address**
- **offset\(_b\)**

32-b bits

\[ 2^b = \text{block size a.k.a line size (in bytes)} \]

Larger block size has distinct hardware advantages:

- less tag overhead
- exploit fast burst transfers from DRAM
- exploit fast burst transfers over wide busses

*What are the disadvantages of increasing block size?*

October 3, 2005
Average Cache Read Latency

\( \alpha \) is HIT RATIO: Fraction of references in cache
\( 1 - \alpha \) is MISS RATIO: Remaining references

Average access time for serial search:

\[
t_c + (1 - \alpha) t_m
\]

Average access time for parallel search:

\[
\alpha t_c + (1 - \alpha) t_m
\]

\( t_c \) is smallest for which type of cache?
Improving Cache Performance

Average memory access time = Hit time + Miss rate x Miss penalty

To improve performance:

• reduce the miss rate (e.g., larger cache)
• reduce the miss penalty (e.g., L2 cache)
• reduce the hit time

What is the simplest design strategy?
Write Performance

- Tag
- Index
- Block Offset

V Tag

Data

2^k lines

HIT

Data Word or Byte

October 3, 2005
Write Policy

• Cache hit:
  – write through: write both cache & memory
    • generally higher traffic but simplifies cache coherence
  – write back: write cache only
    (memory is written only when the entry is evicted)
    • a dirty bit per block can further reduce the traffic

• Cache miss:
  – no write allocate: only write to main memory
  – write allocate (aka fetch on write): fetch into cache

• Common combinations:
  – write through and no write allocate
  – write back with write allocate
Thank you!
Backup
DRAM Packaging

- Clock and control signals
- Address lines multiplexed row/column address
- Data bus (4b, 8b, 16b, 32b)

- DIMM (Dual Inline Memory Module) contains multiple chips with clock/control/address signals connected in parallel (sometimes need buffers to drive signals to all chips)
- Data pins work together to return wide word (e.g., 64-bit data bus using 16x4-bit parts)
Double-Data Rate (DDR2) DRAM

Figure removed for copyright reasons.

Source: Micron 256Mb DDR2 SDRAM datasheet - Bank Read Mode on pg. 44 of Micron Synchronous DRAM Specification.