Memory hierarchy / Cache

Hung-Wei Tseng
Announcement

- Register your iClicker!
- Reading quizzes due this Thursday
- Homework #4 due next “Tuesday”
Recap: why memory hierarchy?

The access time of DDR3-1600 DRAM is around 50ns, 100x to the cycle time of a 2GHz processor! SRAM is as fast as the processor, but $$$
Recap: memory hierarchy

- **Fastest, Most Expensive**
  - CPU: < 1ns
- **Access time**
  - Cache: < 1ns ~ 20 ns
  - Main Memory: 50-60 ns
  - Secondary Storage: 10,000,000 ns

- **Storage types**
  - CPU
  - Cache
  - Main Memory
  - Secondary Storage
  - Biggest
Recap: locality

- Temporal Locality
  - Referenced item tends to be referenced again soon.
- Spatial Locality
  - Items close by referenced item tends to be referenced soon.
  - example: consecutive instructions, arrays
Recap: cache organization

Block (cacheline): The basic unit of data storage in cache. Contains all data with the same tag and index in their address block / cacheline.

Offset: The position of the requesting word in a cache block.

Tag: the high order address bits stored along with the data to identify the actual address of the cache line.

Hit: The data was found in the cache.
Miss: The data was not found in the cache.
Recap: way-associative cache

Set: cache blocks/lines sharing the same index

- Memory address:
  - 1000 0000 0000 0000 0000
  - 0001 0101 1000

- Valid
- Tag
- Data

- Index
- Offset

- Block / Cacheline

- Hit?
Recap: C = ABS

• C: Capacity
• A: Way-Associativity
  • How many blocks in a set
  • 1 for direct-mapped cache
• B: Block Size (Cacheline)
  • How many bytes in a block
• S: Number of Sets:
  • A set contains blocks sharing the same index
  • 1 for fully associate cache
Recap: Athlon 64

- L1 data (D-L1) cache configuration of Athlon 64
  - Size 64KB, 2-way set associativity, 64B block
  - Assume 32-bit memory address

Which of the following is correct?

A. Tag is 17 bits
B. Index is 8 bits
C. Offset is 7 bits
D. The cache has 1024 sets
E. None of the above

\[
C = \text{ABS} \\
64KB = 2 \times 64 \times S \\
S = 512 \\
\text{offset} = \log(64) = 6 \text{ bits} \\
\text{index} = \log(512) = 9 \text{ bits} \\
\text{tag} = 32 - \log(512) - \log(64) = 17 \text{ bits}
\]
Outline

• How cache works
• Evaluating cache performance
• Cause of misses
How cache works
What happens on a write? (Write Allocate, write back)

- Write hit?
  - Update in-place
  - Set dirty bit (Write-Back Policy)
- Write miss?
  - Select victim block
    - LRU, random, FIFO, ...
    - Write back to lower memory hierarchy if dirty
  - Fetch Data from Lower Memory Hierarchy
    - As a unit of a cache block
    - Miss penalty
What happens on a write? (Write Allocate, write back)

- Write hit?
  - Update in-place
  - Set dirty bit (Write-Back Policy)
- Write miss?
  - Select victim block
    - LRU, random, FIFO, ...
    - Write back to lower memory hierarchy if dirty
  - Fetch Data from Lower Memory Hierarchy
    - As a unit of a cache block
    - Miss penalty
What happens on a write? (Write Allocate, write back)

- Write hit?
  - Update in-place
  - Set dirty bit (Write-Back Policy)
- Write miss?
  - Select victim block
    - LRU, random, FIFO, ...
    - Write back to lower memory hierarchy if dirty
  - Fetch Data from Lower Memory Hierarchy
    - As a unit of a cache block
    - Miss penalty
What happens on a write? (Write Allocate, write back)

- Write hit?
  - Update in-place
  - Set dirty bit (Write-Back Policy)
- Write miss?
  - Select victim block
    - LRU, random, FIFO, ...
    - Write back to lower memory hierarchy if dirty
  - Fetch Data from Lower Memory Hierarchy
    - As a unit of a cache block
    - Miss penalty
What happens on a write? (Write Allocate, write back)

- Write hit?
  - Update in-place
  - Set dirty bit (Write-Back Policy)
- Write miss?
  - Select victim block
    - LRU, random, FIFO, ...
    - Write back to lower memory hierarchy if dirty
  - Fetch Data from Lower Memory Hierarchy
    - As a unit of a cache block
    - Miss penalty
What happens on a write? (Write Allocate, write back)

- Write hit?
  - Update in-place
  - Set dirty bit (Write-Back Policy)
- Write miss?
  - Select victim block
    - LRU, random, FIFO, ...
    - Write back to lower memory hierarchy if dirty
  - Fetch Data from Lower Memory Hierarchy
    - As a unit of a cache block
    - Miss penalty
What happens on a write? (Write Allocate, write back)

- Write hit?
  - Update in-place
  - Set dirty bit (Write-Back Policy)
- Write miss?
  - Select victim block
    - LRU, random, FIFO, ...
    - Write back to lower memory hierarchy if dirty
  - Fetch Data from Lower Memory Hierarchy
    - As a unit of a cache block
    - Miss penalty
What happens on a write? (Write Allocate, write back)

- Write hit?
  - Update in-place
  - Set dirty bit (Write-Back Policy)
- Write miss?
  - Select victim block
    - LRU, random, FIFO, ...
  - Write back to lower memory hierarchy if dirty
- Fetch Data from Lower Memory Hierarchy
  - As a unit of a cache block
  - Miss penalty
What happens on a write? (Write Allocate, write back)

- **Write hit?**
  - Update in-place
  - Set dirty bit (Write-Back Policy)

- **Write miss?**
  - Select victim block
  - LRU, random, FIFO, ...
  - Write back to lower memory hierarchy if dirty

- Fetch Data from Lower Memory Hierarchy
  - As a unit of a cache block
  - Miss penalty
What happens on a write? (Write Allocate, write through)

- Write hit?
  - Update in-place
  - Write to lower memory (Write-Through Policy)

- Write miss?
  - Select victim block
    - LRU, random, FIFO, ...
    - Write back if dirty
  - Fetch Data from Lower Memory Hierarchy
    - As a unit of a cache block
    - Miss penalty
What happens on a write? (Write Allocate, write through)

- **Write hit?**
  - Update in-place
  - Write to lower memory (Write-Through Policy)

- **Write miss?**
  - Select victim block
    - LRU, random, FIFO, ...
    - Write back if dirty
  - Fetch Data from Lower Memory Hierarchy
    - As a unit of a cache block
    - Miss penalty
What happens on a write? (Write Allocate, write through)

- Write hit?
  - Update in-place
  - Write to lower memory (Write-Through Policy)

- Write miss?
  - Select victim block
    - LRU, random, FIFO, ...
    - Write back if dirty
  - Fetch Data from Lower Memory Hierarchy
    - As a unit of a cache block
    - Miss penalty
What happens on a write? (Write Allocate, write through)

- Write hit?
  - Update in-place
  - Write to lower memory (Write-Through Policy)

- Write miss?
  - Select victim block
    - LRU, random, FIFO, ...
    - Write back if dirty
  - Fetch Data from Lower Memory Hierarchy
    - As a unit of a cache block
    - Miss penalty
What happens on a write? (Write Allocate, write through)

- Write hit?
  - Update in-place
  - Write to lower memory (Write-Through Policy)
- Write miss?
  - Select victim block
    - LRU, random, FIFO, ...
    - Write back if dirty
  - Fetch Data from Lower Memory Hierarchy
    - As a unit of a cache block
    - Miss penalty
What happens on a write? (Write Allocate, write through)

- Write hit?
  - Update in-place
  - Write to lower memory (Write-Through Policy)

- Write miss?
  - Select victim block
    - LRU, random, FIFO, ...
    - Write back if dirty
  - Fetch Data from Lower Memory Hierarchy
    - As a unit of a cache block
    - Miss penalty

```
CPU

SW: tag index offset

L1 $

L2 $
```
What happens on a write? (Write Allocate, write through)

- Write hit?
  - Update in-place
  - Write to lower memory (Write-Through Policy)

- Write miss?
  - Select victim block
    - LRU, random, FIFO, ...
    - Write back if dirty
  - Fetch Data from Lower Memory Hierarchy
    - As a unit of a cache block
    - Miss penalty
What happens on a write? (Write Allocate, write through)

- **Write hit?**
  - Update in-place
  - Write to lower memory (Write-Through Policy)
- **Write miss?**
  - Select victim block
    - LRU, random, FIFO, ...
    - Write back if dirty
  - Fetch Data from Lower Memory Hierarchy
    - As a unit of a cache block
    - Miss penalty

![Diagram showing CPU, L1$, L2$, SW, tag, index, offset, miss?, fetch (if write allocate), tag, index, 0, L2$]
What happens on a write? (Write Allocate, write through)

- Write hit?
  - Update in-place
  - Write to lower memory (Write-Through Policy)

- Write miss?
  - Select victim block
    - LRU, random, FIFO, ...
    - Write back if dirty
  - Fetch Data from Lower Memory Hierarchy
    - As a unit of a cache block
    - Miss penalty
What happens on a write? (Write Allocate, write through)

- Write hit?
  - Update in-place
  - Write to lower memory (Write-Through Policy)

- Write miss?
  - Select victim block
    - LRU, random, FIFO, ...
    - Write back if dirty
  - Fetch Data from Lower Memory Hierarchy
    - As a unit of a cache block
    - Miss penalty

CPU

L1 $

L2 $

Write Allocate, write through

tag  index  offset

tag  index  0

tag  index  B-1

fetch (if write allocate)

miss?

update in L1

write (if write-through policy)

sw
Write-back v.s. write-through

• How many of the following statements about write-back and write-through policies are correct?
  • Write back can reduce the number of writes to lower-level memory hierarchy
  • The average write response time of write-back is better
  • A read miss may still result in writes if the cache uses write-back
  • The miss penalty of the cache using write-through policy is constant.

A. 0
B. 1
C. 2
D. 3
E. 4
Write-back v.s. write-through

• How many of the following statements about write-back and write-through policies are correct?
  • Write back can reduce the number of writes to lower-level memory hierarchy
  • The average write response time of write-back is better
  • A read miss may still result in writes if the cache uses write-back
  • The miss penalty of the cache using write-through policy is constant.

A. 0
B. 1
C. 2
D. 3
E. 4
What happens on a write? (No-Write Allocate, write through)

- **Write hit?**
  - Update in-place
  - Write to lower memory (Write-Through only)
    - write penalty (can be eliminated if there is a buffer)

- **Write miss?**
  - Write to the first lower memory hierarchy has the data
    - Penalty
What happens on a write? (No-Write Allocate, write through)

- Write hit?
  - Update in-place
  - Write to lower memory (Write-Through only)
    - write penalty (can be eliminated if there is a buffer)

- Write miss?
  - Write to the first lower memory hierarchy has the data
    - Penalty
What happens on a write? (No-Write Allocate, write through)

- **Write hit?**
  - Update in-place
  - Write to lower memory (Write-Through only)
    - write penalty (can be eliminated if there is a buffer)

- **Write miss?**
  - Write to the first lower memory hierarchy has the data
    - Penalty
What happens on a write? (No-Write Allocate, write through)

- Write hit?
  - Update in-place
  - Write to lower memory (Write-Through only)
    - write penalty (can be eliminated if there is a buffer)

- Write miss?
  - Write to the first lower memory hierarchy has the data
    - Penalty
What happens on a write? (No-Write Allocate, write through)

- Write hit?
  - Update in-place
  - Write to lower memory (Write-Through only)
    - write penalty (can be eliminated if there is a buffer)

- Write miss?
  - Write to the first lower memory hierarchy has the data
  - Penalty
What happens on a write? (No-Write Allocate, write through)

- Write hit?
  - Update in-place
  - Write to lower memory (Write-Through only)
    - write penalty (can be eliminated if there is a buffer)

- Write miss?
  - Write to the first lower memory hierarchy has the data
    - Penalty
What happens on a write? (No-Write Allocate, write through)

- Write hit?
  - Update in-place
  - Write to lower memory (Write-Through only)
    - write penalty (can be eliminated if there is a buffer)

- Write miss?
  - Write to the first lower memory hierarchy has the data
  - Penalty
What happens on a write? (No-Write Allocate, write through)

- **Write hit?**
  - Update in-place
  - Write to lower memory (Write-Through only)
    - write penalty (can be eliminated if there is a buffer)

- **Write miss?**
  - Write to the first lower memory hierarchy has the data
    - Penalty
What happens on a write? (No-Write Allocate, write back)

CPU

- Write hit?
  - Update in-place
- Write miss?
  - Write to the first lower memory hierarchy has the data
    - Penalty

L1 $

L2 $
What happens on a write? (No-Write Allocate, write back)

- **Write hit?**
  - Update in-place
- **Write miss?**
  - Write to the first lower memory hierarchy has the data
    - Penalty
What happens on a write? (No-Write Allocate, write back)

- Write hit?
  - Update in-place
- Write miss?
  - Write to the first lower memory hierarchy has the data
    - Penalty
What happens on a write? (No-Write Allocate, write back)

- Write hit?
  - Update in-place
- Write miss?
  - Write to the first lower memory hierarchy has the data
  - Penalty
What happens on a write? (No-Write Allocate, write back)

- Write hit?
  - Update in-place
- Write miss?
  - Write to the first lower memory hierarchy has the data
  - Penalty
What happens on a write? (No-Write Allocate, write back)

- Write hit?
  - Update in-place
- Write miss?
  - Write to the first lower memory hierarchy has the data
    - Penalty
What happens on a write? (No-Write Allocate, write back)

- Write hit?
  - Update in-place
- Write miss?
  - Write to the first lower memory hierarchy has the data
    - Penalty
What happens on a read?

- Read hit
  - hit time
- Read miss?
  - Select victim block
    - LRU, random, FIFO, ...
    - Write back if dirty
  - Fetch Data from Lower Memory Hierarchy
    - As a unit of a cache block
      - Data with the same “block address” will be fetch
    - Miss penalty
What happens on a read?

- Read hit
- Read miss?
  - Select victim block
    - LRU, random, FIFO, ...
    - Write back if dirty
  - Fetch Data from Lower Memory Hierarchy
    - As a unit of a cache block
      - Data with the same “block address” will be fetch
    - Miss penalty
What happens on a read?

- Read hit
- hit time

- Read miss?
  - Select victim block
    - LRU, random, FIFO, ...
    - Write back if dirty
  - Fetch Data from Lower Memory Hierarchy
    - As a unit of a cache block
      - Data with the same “block address” will be fetch
    - Miss penalty
What happens on a read?

- Read hit
- Read miss?
  - Select victim block
    - LRU, random, FIFO, ...
    - Write back if dirty
  - Fetch Data from Lower Memory Hierarchy
    - As a unit of a cache block
      - Data with the same “block address” will be fetch
    - Miss penalty
What happens on a read?

**CPU**

- **L1 $**
  - *miss?*
  - *fetch*
    - *tag*  
    - *index*  
    - *offset*  
  
- *lw*
  - *tag*  
  - *index*  
  - *offset*  

**L2 $**

- *write-back (if dirty)*

**Read hit**
- *hit time*

**Read miss?**
- *Select victim block*
  - **LRU, random, FIFO, ...**
  - **Write back if dirty**
- **Fetch Data from Lower Memory Hierarchy**
  - As a unit of a cache block
    - Data with the same “block address” will be fetch
  - **Miss penalty**
What happens on a read?

• Read hit
• hit time

• Read miss?
• Select victim block
  • LRU, random, FIFO, ...
  • Write back if dirty

• Fetch Data from Lower Memory Hierarchy
  • As a unit of a cache block
    • Data with the same “block address” will be fetch
  • Miss penalty
Evaluating cache performance
How to evaluate cache performance

- If the load/store instruction hits in L1 cache where the hit time is usually the same as a CPU cycle
  - The CPI of this instruction is the base CPI
- If the load/store instruction misses in L1, we need to access L2
  - The CPI of this instruction needs to include the cycles of accessing L2
- If the load/store instruction misses in both L1 and L2, we need to go to lower memory hierarchy (L3 or DRAM)
  - The CPI of this instruction needs to include the cycles of accessing L2, L3, DRAM
How to evaluate cache performance

- CPI\text{Average} : the average CPI of a memory instruction
  \[ \text{CPI}_{\text{Average}} = \text{CPI}_{\text{base}} + \text{miss\_rate}_{L1} \times \text{miss\_penalty}_{L1} \]
  \[ \text{miss\_penalty}_{L1} = \text{CPI}_{\text{accessing\_L2}} + \text{miss\_rate}_{L2} \times \text{miss\_penalty}_{L2} \]
  \[ \text{miss\_penalty}_{L2} = \text{CPI}_{\text{accessing\_L3}} + \text{miss\_rate}_{L3} \times \text{miss\_penalty}_{L3} \]
  \[ \text{miss\_penalty}_{L3} = \text{CPI}_{\text{accessing\_DRAM}} + \text{miss\_rate}_{\text{DRAM}} \times \text{miss\_penalty}_{\text{DRAM}} \]

- If the problem (like those in your textbook) is asking for average memory access time, transform the CPI values into/from time by multiplying with CPU cycle time!
Average memory access time

- Average Memory Access Time (AMAT) = Hit Time + Miss rate * Miss penalty
- Miss penalty = AMAT of the lower memory hierarchy
- AMAT = hit\_time_{L1} + miss\_rate_{L1} * AMAT_{L2}
  - AMAT_{L2} = hit\_time_{L2} + miss\_rate_{L2} * AMAT_{DRAM}
Cache & Performance

- 5-stage MIPS processor.
  - Application: 80% ALU, 20% L/S
  - L1 I-cache miss rate: 5%, hit time: 1 cycle
  - L1 D-cache miss rate: 10%, hit time: 1 cycle
  - L2 U-Cache miss rate: 20%, hit time: 10 cycles
  - Main memory hit time: 100 cycles
  - What’s the average CPI?

A. 0.75  
B. 1.35  
C. 1.75  
D. 1.80  
E. none of the above

\[
\text{CPI}_{\text{Average}} = \text{CPI}_{\text{base}} + \text{miss\_rate} \times \text{miss\_penalty} \\
= 1 + (5\% \times (10 + 20\% \times (100))) + 20\% \times (10\% \times (10 + 20\% \times (100))) \\
= 3.1
\]