



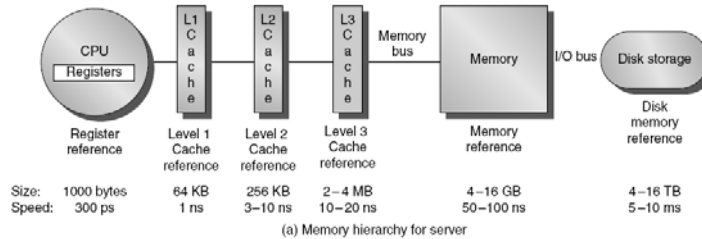
Chapter 2

Memory Hierarchy Design

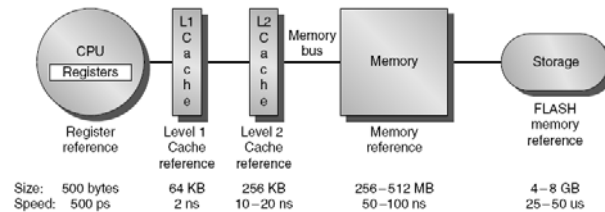
Introduction

- Programmers want unlimited amounts of memory with low latency
- Fast memory technology is more expensive per bit than slower memory
- Solution: organize memory system into a hierarchy
 - Entire addressable memory space available in largest, slowest memory
 - Incrementally smaller and faster memories, each containing a subset of the memory below it, proceed in steps up toward the processor
- Temporal and spatial locality insures that nearly all references can be found in smaller memories
 - Gives the illusion of a large, fast memory being presented to the processor

Memory Hierarchy

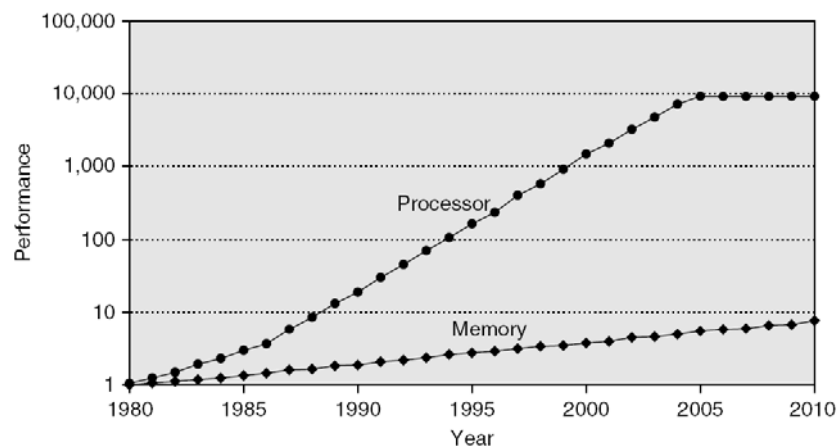


(a) Memory hierarchy for server



(b) Memory hierarchy for a personal mobile device

Memory Performance Gap



Memory Hierarchy Design

- Memory hierarchy design becomes more crucial with recent multi-core processors:
 - Aggregate peak bandwidth grows with # cores:
 - Intel Core i7 can generate two references per core per clock
 - Four cores and 3.2 GHz clock
 - 25.6 billion 64-bit data references/second +
 - 12.8 billion 128-bit instruction references
 - = 409.6 GB/s!
 - DRAM bandwidth is only 6% of this (25 GB/s)
 - Requires:
 - Multi-port, pipelined caches
 - Two levels of cache per core
 - Shared third-level cache on chip

Performance and Power

- High-end microprocessors have >10 MB on-chip cache
 - Consumes large amount of area and power budget

Terminology

- **A Block:** The smallest unit of information transferred between two levels.
- **Hit:** Item is found in some block in the upper level (example: Block X)
- **Miss:** Item needs to be retrieved from a block in the lower level (Block Y)
 - **Miss Rate** = 1 - (Hit Rate)
 - **Miss Penalty:** Time to replace a block in the upper level + Time to deliver the block the processor

Cache operation

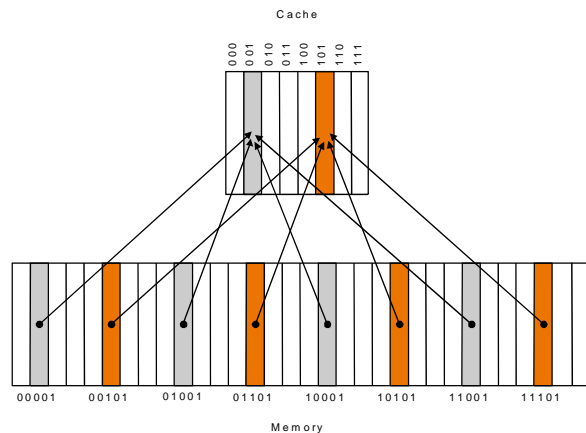
- Questions
 1. Where a block be placed in the cache (**placement**)
 2. How is a block is found if it is in the cache (**identification**)
 3. Which block should be replaced on a miss (**replacement**)
 4. What happens on a write (**write strategy**)

Cache Organization: Placement

- 1 Direct mapped cache: A block can be placed in only one location (cache block frame), given by the mapping function:
$$\text{index} = (\text{Block address}) \text{ MOD } (\text{Number of blocks in cache})$$
- 2 Fully associative cache: A block can be placed anywhere in cache. (no mapping function).
- 3 Set associative cache: A block can be placed in a restricted set of places, or cache block frames. A set is a group of block frames in the cache. A block is first mapped onto the set and then it can be placed anywhere within the set. The set in this case is chosen by:
$$\text{index} = (\text{Block address}) \text{ MOD } (\text{Number of sets in cache})$$

If there are n blocks in a set the cache placement is called n -way set-associative.

Direct Mapped Cache



Direct Mapped Cache

1K = 1024 Blocks
 Each block = one word

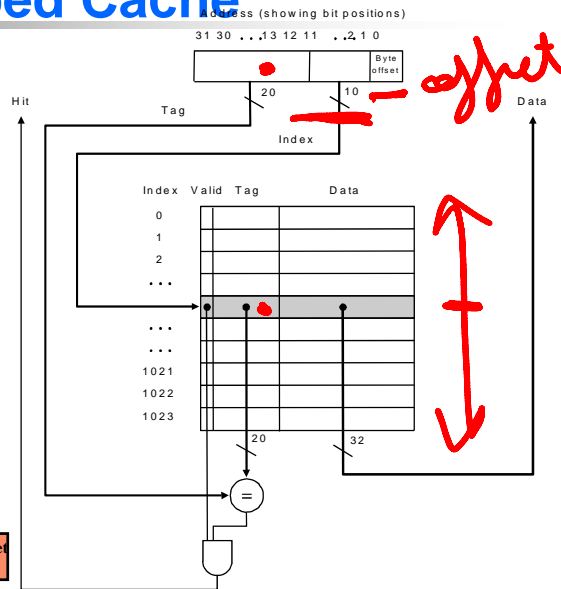
Can cache up to
 2^{32} bytes = 4 GB
 of memory

Mapping function:

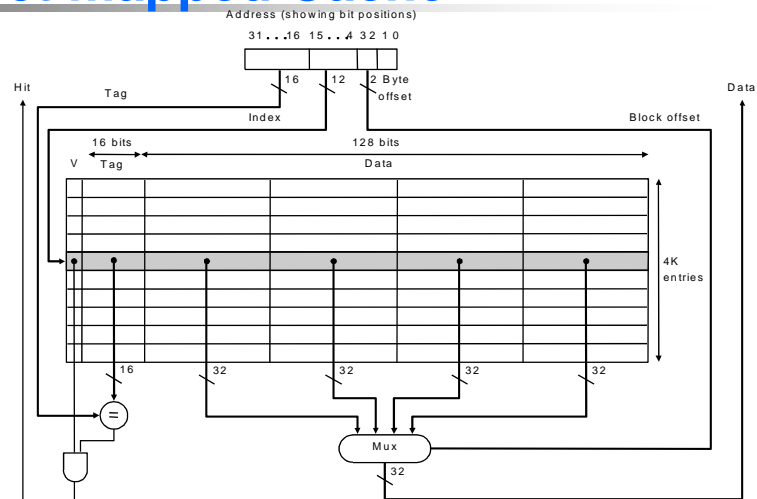
Cache Block frame number =
 (Block address) MOD (1024)

i.e. index field or
 10 low bit of block address

Block Address = 30 bits	Block offset = 2 bits
Tag = 20 bits	Index = 10 bits

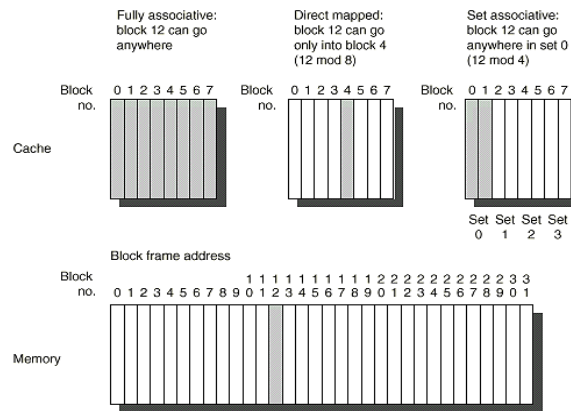


Direct mapped Cache



Block Address = 28 bits	Block offset = 4 bits
Tag = 16 bits	Index = 12 bits

Cache Organization

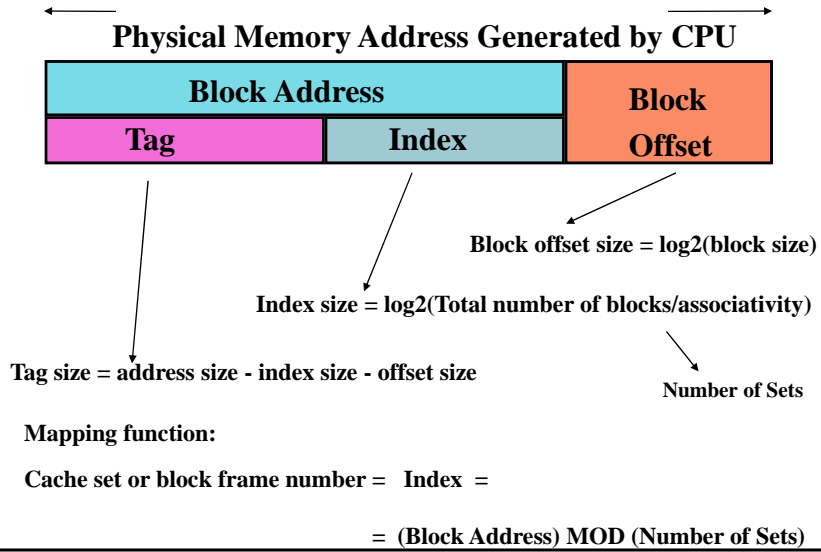


Cache Organization

- Each block frame in cache has an address tag.
- The tags of every cache block that might contain the required data are checked in parallel.
- A valid bit is added to the tag to indicate whether this entry contains a valid address.
- The address from the CPU to cache is divided into:
 - A block address, further divided into:
 - An index field to choose a block set in cache.
 - (no index field when fully associative).
 - A tag field to search and match addresses in the selected set.
 - A block offset to select the data from the block.



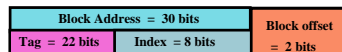
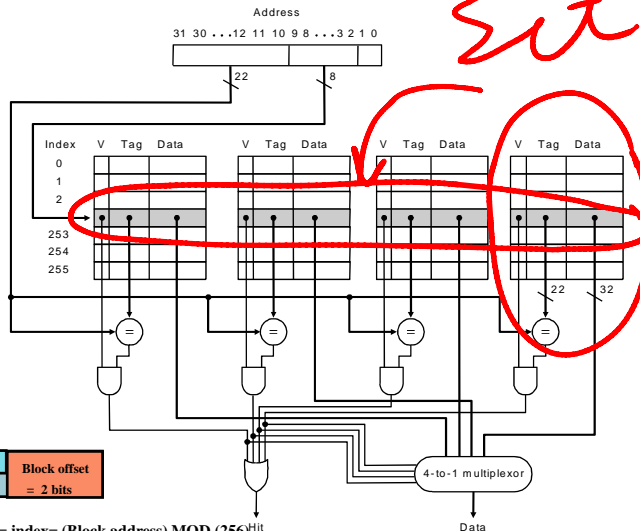
Cache Organization



Set Associative: 4K 4Way

1024 block frames
 Each block = one word
 4-way set associative
 1024 / 4 = 256 sets

Can cache up to
 2^{32} bytes = 4 GB
 of memory



Mapping Function: Cache Set Number = $\text{index} = (\text{Block address}) \text{ MOD } (256)^{\text{Hit}}$

Miss Rate

■ Associativity:	2-way		4-way		8-way	
	LRU	Random	LRU	Random	LRU	Random
■ Size						
■ 16 KB	5.18%	5.69%	4.67%	5.29%	4.39%	4.96%
■ 64 KB	1.88%	2.01%	1.54%	1.66%	1.39%	1.53%
■ 256 KB	1.15%	1.17%	1.13%	1.13%	1.12%	1.12%

Cache Performance

- CPUtime = Instruction count x CPI x Clock cycle time
- CPLeexecution = CPI with ideal memory
- CPI = CPLeexecution + Mem Stall cycles per instruction
- Mem Stall cycles per instruction =
Mem accesses per instruction x Miss rate x Miss penalty
- CPUtime = Instruction Count x (CPLeexecution +
Mem Stall cycles per instruction) x Clock cycle time
- CPUtime = IC x (CPLeexecution + Mem accesses per instruction x
Miss rate x Miss penalty) x Clock cycle time
- Misses per instruction = Memory accesses per instruction x Miss rate
- CPUtime = IC x (CPLeexecution + Misses per instruction x Miss penalty) x
Clock cycle time

Cache Performance

- Assuming the following execution and cache parameters:
 - Cache miss penalty = 50 cycles
 - Normal instruction execution CPI ignoring memory stalls = 2.0 cycles
 - Miss rate = 2%
 - Average memory references/instruction = 1.33
- CPU time = IC x [CPI execution + Memory accesses/instruction x Miss penalty] x Clock cycle time
- CPU time with cache = IC x (2.0 + (1.33 x 2% x 50)) x clock cycle time
- = IC x 3.33 x Clock cycle time
- *Lower CPI execution increases the impact of cache miss clock cycles*

Cache Performance

- Suppose a CPU executes at Clock Rate = 200 MHz (5 ns per cycle) with a single level of cache.
- CPI_{execution} = 1.1
- Instruction mix: 50% arith/logic, 30% load/store, 20% control
- Assume a cache miss rate of 1.5% and a miss penalty of 50 cycles.
- $CPI = CPI_{\text{execution}} + \text{mem stalls per instruction}$
- Mem Stalls per instruction = Mem accesses per instruction x Miss rate x Miss penalty
- Mem accesses per instruction = 1 + .3 = 1.3
- Mem Stalls per instruction = 1.3 x .015 x 50 = 0.975
- $CPI = 1.1 + .975 = 2.075$
- The ideal memory CPU with no misses is 2.075/1.1 = 1.88 times faster

Cache Performance

- Suppose for the previous example we double the clock rate to 400 MHz, how much faster is this machine, assuming similar miss rate, instruction mix?
- Since memory speed is not changed, the miss penalty takes more CPU cycles:
 - Miss penalty = $50 \times 2 = 100$ cycles.
 - $CPI = 1.1 + 1.3 \times .015 \times 100 = 1.1 + 1.95 = 3.05$
 - Speedup = $(CPI_{old} \times C_{old}) / (CPI_{new} \times C_{new})$
 - $= 2.075 \times 2 / 3.05 = 1.36$
- The new machine is only 1.36 times faster rather than 2 times faster due to the increased effect of cache misses.
- CPUs with higher clock rate, have more cycles per cache miss and more memory impact on CPI.*

Cache Performance

- Suppose a CPU uses separate level one (L1) caches for instructions and data (Harvard memory architecture) with different miss rates for instruction and data access:
 - $CPI_{execution} = 1.1$
 - Instruction mix: 50% arith/logic, 30% load/store, 20% control
 - Assume a cache miss rate of 0.5% for instruction fetch and a cache data miss rate of 6%.
 - A cache hit incurs no stall cycles while a cache miss incurs 200 stall cycles for both memory reads and writes. Find the resulting CPI using this cache? How much faster is the CPU with ideal memory?
$$CPI = CPI_{execution} + \text{mem stalls per instruction}$$


$$\text{Mem Stall cycles per instruction} = \text{Instruction Fetch Miss rate} \times \text{Miss Penalty} + \text{Data Memory Accesses Per Instruction} \times \text{Data Miss Rate} \times \text{Miss Penalty}$$

$$\text{Mem Stall cycles per instruction} = 1 \times 0.5/100 \times 200 + 0.3 \times 6/100 \times 200 = 1 + 3.6 = 4.6$$

$$CPI = CPI_{execution} + \text{mem stalls per instruction} = 1.1 + 4.6 = 5.7$$

The CPU with ideal cache (no misses) is $5.7/1.1 = 5.18$ times faster
 With no cache the CPI would have been $= 1.1 + 1.3 \times 200 = 261.1$

Cache Performance



Size	Instruction cache	Data cache	Unified cache
1 KB	3.06%	24.61%	13.34%
2 KB	2.26%	20.57%	9.78%
4 KB	1.78%	15.94%	7.24%
8 KB	1.10%	10.19%	4.57%
16 KB	0.64%	6.47%	2.87%
32 KB	0.39%	4.82%	1.99%
64 KB	0.15%	3.77%	1.35%
128 KB	0.02%	2.88%	0.95%

Write Policy

- 1 Write Through:** Data is written to both the cache block and to a block of main memory.
 - The lower level always has the most updated data; an important feature for I/O and multiprocessing.
 - Easier to implement than write back.
 - A write buffer is often used to reduce CPU write stall while data is written to memory.
- 2 Write back:** Data is written or updated only to the cache block. The modified or dirty cache block is written to main memory when it's being replaced from cache.
 - Writes occur at the speed of cache
 - A status bit called a dirty or modified bit, is used to indicate whether the block was modified while in cache; if not the block is not written back to main memory when replaced.
 - Uses less memory bandwidth than write through.

Write Policy

Write Allocate:

The cache block is loaded on a write miss followed by write hit actions.

No-Write Allocate:

The block is modified in the lower level (lower cache level, or main memory) and not loaded into cache.

Example

- Which has a lower miss rate 16KB cache for both instruction or data, or a combined 32KB cache? (0.64%, 6.47%, 1.99%).
- Assume hit=1cycle and miss =50 cycles. 75% of memory references are instruction fetch. *reads*
- Miss rate of split cache= $0.75*0.64\%+0.25*6.47\%=2.1\%$
- Slightly worse than 1.99% for combined cache. But, what about average memory access time?
- Split cache: $75\%(1+0.64\%*50)+25\%(1+6.47\%*50) = 2.05$ cycles.
- Combined cache: $75\%(1+1.99\%*50)+25\%(1+1+1.99\%*50) = 2.24$

Extra cycle for load/store

Example

- A CPU with $CPI_{\text{execution}} = 1.1$ Mem accesses per instruction = 1.3
- Uses a unified L1 Write Through, No Write Allocate, with:
 - No write buffer,
 - Perfect Write buffer
 - A realistic write buffer that eliminates 85% of write stalls
- Instruction mix: 50% arith/logic, 15% load, 15% store, 20% control
- Assume a cache miss rate of 1.5% and a miss penalty of 50 cycles.
 $CPI = CPI_{\text{execution}} + \text{mem stalls per instruction}$
% reads = $1.15/1.3 = 88.5\%$ % writes = $.15/1.3 = 11.5\%$

Example

- A CPU with $CPI_{\text{execution}} = 1.1$ uses a unified L1 with write back, with write allocate, and the probability a cache block is dirty = 10%
- Instruction mix: 50% arith/logic, 15% load, 15% store, 20% control
- Assume a cache miss rate of 1.5% and a miss penalty of 50 cycles.

1.3 mem acc / inst

50+50

$$\frac{1.5}{.015} (1.3 \times 50 \times 0.9 + 1.3 \times 0.1 \times 100)$$

Example

- CPU with $CPI_{\text{execution}} = 1.1$ running at clock rate = 500 MHz
- 1.3 memory accesses per instruction.
- L_1 cache operates at 500 MHz with a miss rate of 5%
- L_2 cache operates at 250 MHz with local miss rate 40%, ($T_2 = 2$ cycles)
- Memory access penalty, $M = 100$ cycles. Find CPI.

$$1.3 \left(\frac{5}{100} \times 0.6 \times 2 + \frac{5}{100} \times 0.4 \times 100 \right)$$

Example

- CPU with $CPI_{\text{execution}} = 1.1$ running at clock rate = 500 MHz
- 1.3 memory accesses per instruction.
- For L_1 :
 - Cache operates at 500 MHz with a miss rate of $1 - H_1 = 5\%$
 - Write through to L_2 with perfect write buffer with write allocate
- For L_2 :
 - Cache operates at 250 MHz with local miss rate $1 - H_2 = 40\%$, ($T_2 = 2$ cycles)
 - Write back to main memory with write allocate
 - Probability a cache block is dirty = 10%
- Memory access penalty, $M = 100$ cycles. Find CPI.

$$0.05 \left(0.6 \times 2 + 0.4 \times 0.9 \times 100 + 0.4 \times 0.1 \times 700 \right)$$

Example

- CPU with $CPI_{\text{execution}} = 1.1$ running at clock rate = 500 MHz
- 1.3 memory accesses per instruction.
- L_1 cache operates at 500 MHz with a miss rate of 5%
- L_2 cache operates at 250 MHz with a local miss rate 40%, ($T_2 = 2$ cycles)
- L_3 cache operates at 100 MHz with a local miss rate 50%, ($T_3 = 5$ cycles)
- Memory access penalty, $M = 100$ cycles. Find CPI.

HW