

Memory Hierarchy and Cache

Ch 4.1-3

Memory Hierarchy
Main Memory
Cache
Implementation

21.9.2000 Copyright Teemu Kerola 2000 1

Teemu's Cheesecake

Register, on-chip cache, memory, disk, and tape speeds relative to times locating cheese for the cheese cake you are baking...

21.9.2000 Copyright Teemu Kerola 2000 2

Goal ⁽⁴⁾

- I want my memory lightning fast
- I want my memory to be gigantic in size
- Register access viewpoint:
 - data access as fast as from HW register
 - data size as large as memory
- Memory access viewpoint:
 - data access as fast as from memory
 - data size as large as disk

cache
HW solution
virtual memory
HW help for SW solution

21.9.2000 Copyright Teemu Kerola 2000

Memory Hierarchy ⁽⁵⁾

- Most often needed data is kept close
- Access to small data sets can be made fast
 - simpler circuits
- Faster is more expensive
- Large can be bigger and cheaper

Memory Hierarchy

up: smaller, faster, more expensive, more frequent access

down: bigger, slower, less expensive, less frequent access

Fig. 4.1

21.9.2000 Copyright Teemu Kerola 2000 4

Locality ⁽⁵⁾

(paikallisuus)

Temporal locality
data referenced again soon (ajallinen paikallisuus)

Spatial locality
nearby data referenced soon (alueellinen paikallisuus)

- The reason why memory hierarchies work

Prob (small data set) = 99%	Cost (small data set) = 2 μs
Prob (the rest) = 1%	Cost (the rest) = 20 μs

Aver cost 99% * 2 μs + 1% * 20 μs = 2.2 μs

Close to cost of small data set

21.9.2000 Copyright Teemu Kerola 2000 5

Memory

- Random access semiconductor memory
 - give address & control, read/write data
- ROM, PROMS
 - system startup memory, BIOS (Basic Input/Output System)
 - load and execute OS at boot
 - also random access
- RAM
 - “normal” memory accessible by CPU

Table 4.2

21.9.2000 Copyright Teemu Kerola 2000 6

RAM

- Dynamic RAM, DRAM E.g., \$1 / MB (year 2000)?
 - simpler, slower, denser, bigger (?)
 - main memory? E.g., 60 ns access
 - periodic refreshing required
- Static RAM, SRAM E.g., \$100 / MB (year 2000)?
 - more complex, faster, smaller
 - cache? E.g., 5 ns access?
 - no periodic refreshing needed
 - data remains until power is lost

21.9.2000 Copyright Teemu Kerola 2000 7

DRAM Access

- 16 Mb DRAM Fig. 4.4 Fig. 4.5 (b)
 - 4 bit data items
 - 4M data elements, 2K * 2K square
 - Address 22 bits
 - row access select (RAS)
 - column access select (CAS)
 - interleaved on 11 address pins
- Simultaneous access to many 16Mb memory chips to access larger data items
 - Access 32 bit words in parallel? Need 8 chips.

21.9.2000 Copyright Teemu Kerola 2000 8

SDRAM (Synchronous DRAM)

- 16 bits in parallel
 - access 4 SDRAMs in parallel
- Faster than plain DRAM
- Current main memory technology (year 2000)

E.g., \$1 / MB (year 2000)

21.9.2000 Copyright Teemu Kerola 2000 9

RDRAM (RambusDRAM)

- New technology, works with fast memory bus
 - expensive E.g., \$2 / MB (year 2000)?
- Faster transfer rate than with SDRAM E.g., 1.6 GB/sec vs. 200 MB/sec (?)
- Faster access than SDRAM E.g., 38 ns vs. 44 ns
- Fast internal Rambus channel (800 MHz)
- Rambus memory controller connects to bus
- Speed slows down with many memory modules
 - serially connected on Rambus channel
 - not good for servers with 1 GB memory (for now!)
- 5% of memory chips now, 30% next year (2001)?

21.9.2000 Copyright Teemu Kerola 2000 10

21.9.2000 Copyright Teemu Kerola 2000 11

Cache Memory (välimuisti)

- Problem: how can I make my (main) memory as fast as my registers?
- Answer: (processor) cache
 - keep most probably referenced data in fast cache close to processor, and rest of it in memory
 - much smaller than main memory
 - (much) more expensive (per byte) than memory
 - most of data accesses to cache 90% 99%?

Fig. 4.13 Fig. 4.16

21.9.2000 Copyright Teemu Kerola 2000 12

Cache Operation (5)

- Data is in cache? **Hit** Fig. 4.15
- Data is only in memory? **Miss**
- Read it to cache
- CPU waits until data available

Many blocks help for temporal locality
many different data items in cache Fig. 4.14

Large blocks help for spatial locality
lots of "nearby" data available

Fixed cache size?
Select "many" or "large"?

21.9.2000 Copyright Teemu Kerola 2000 13

Cache Features

- Size
- Mapping function (kuvausfunktio)
- Replacement algorithm (poistoalgoritmi)
- Write policy (kirjoituspolitiikka)
- Line size (block size)? (rivin tai lohkon koko)
- Number of caches?

21.9.2000 Copyright Teemu Kerola 2000 14

Cache Size

- Bigger is better in general
- Bigger may be slower
 - lots of gates, cumulative gate delay?
- Too big might be too slow!
 - Help: 2- or 3-level caches

1KW (4 KB), 512KW (2 MB)?

21.9.2000 Copyright Teemu Kerola 2000 15

Mapping: Memory Address (3)

- Alpha AXP issues 34 bit memory addresses
 - At cache hit block offset is controlling a multiplexer to select right word

34 bit address (byte address)

block address

block offset

29 bits 5 bits

max physical address space = $2^{34} = 16\text{GB}$

Number of possible blocks in physical address space = $2^{29} = 500\text{M blocks}$

Cache line size = block size = $2^5 = 32$ bytes = 4 words

21.9.2000 Copyright Teemu Kerola 2000 16

Direct Mapping (6) (suora kuvaus)

- Every block has only one possible location (cache line number) in cache
 - determined by index field bits

34 bit address (byte address)

Block address (in memory)

block

tag index offset

Unique bits that are different for each block, stored in each cache line

Addr in cache, cache size = $2^8 = 256$ blocks = 8 KB

Cache line size = block size = $2^5 = 32$ bytes

21 bits 8 bits 5 bits

Fig. 4.17 (s = 29, r = 8, w = 5)
Fig. 7.10 from [PaHe98]

21.9.2000 Copyright Teemu Kerola 2000 17

Direct Mapping Example (5)

Word = 4 bytes (here)

ReadW I2, 0xA4
0xA4 = 1010 0100

8 bit address (byte address)

tag

index

offset

2 3 3

10 100 100

No match

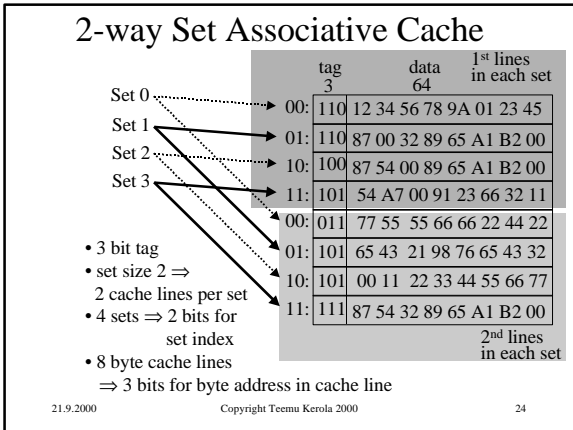
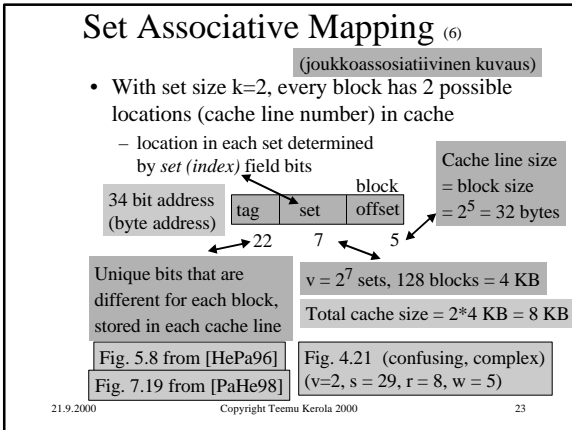
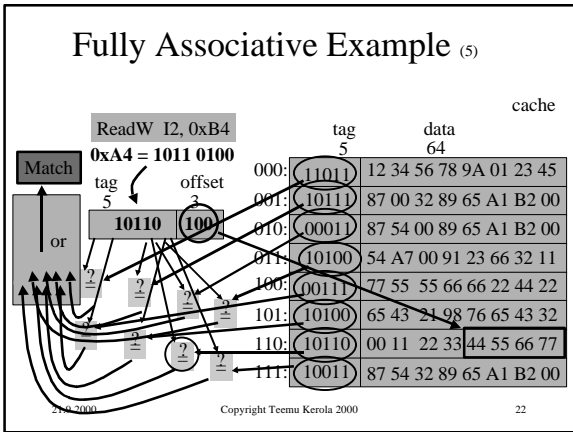
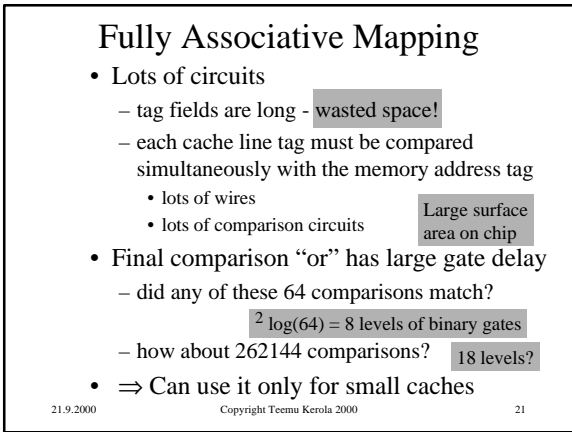
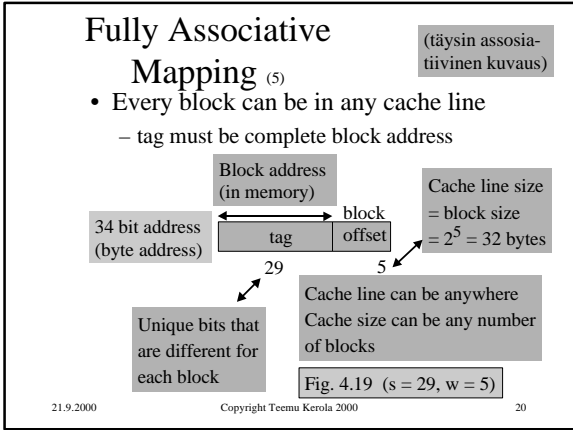
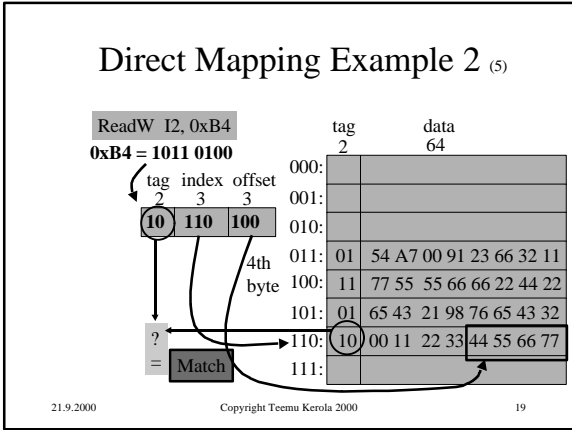
? =

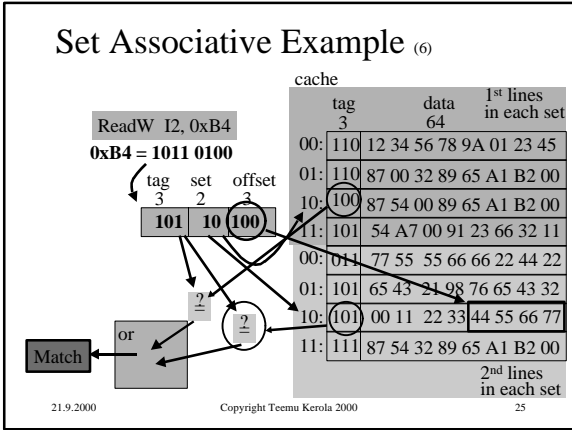
Read new memory block from memory address 0xA0=1010 0000 to cache location 100, update tag, and then continue with data access.

tag	data
000:	
001:	
010:	
011:	01 54 A7 00 91 23 66 32 11
100:	11 77 55 55 66 66 22 44 22
101:	01 65 43 21 98 76 65 43 32
110:	

Cache line size = block size = $2^3 = 8$ bytes = 64 bits

21.9.2000





Set Associative Mapping

- Set associative cache with set size 2 = 2-way cache
- Degree of associativity v? Usually 2
 - v large? Fig. 7.16 from [PaHe98]
 - More data items (v) in one set
 - less "collisions"
 - final comparison (matching tags?) gate delay?
 - v maximum (nr of cache lines) ⇒ fully associative mapping
 - v minimum (1) ⇒ direct mapping

21.9.2000 Copyright Teemu Kerola 2000 26

Replacement Algorithm

- Which cache line to remove to make room for new block from memory?
- Direct mapping case trivial
- First-In-First-Out (FIFO)
- Least-Frequently-Used (LFU)
- Random
- Which one is best?
 - Chip area?
 - Fast? Easy to implement?

21.9.2000 Copyright Teemu Kerola 2000 27

Write Policy

- How to handle writes to memory?
- Write through (läpikirjoittava)
 - each write goes always to memory
 - each write is a cache miss!
- Write back (lopuksi kirjoittava takaisin kirjoittava?)
 - write cache block to memory only when it is replaced in cache
 - memory may have stale (old) data
 - cache coherence problem (välimuistin yhteneväisyysongelma)

21.9.2000 Copyright Teemu Kerola 2000 28

Line size

- How big cache line?
- Optimise for temporal or spatial locality?
- Data references and code references behave in a different way
- Best size varies with program or program phase
- 2-8 words?
 - word = 1 float??

21.9.2000 Copyright Teemu Kerola 2000 29

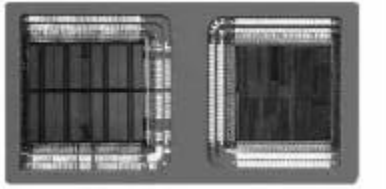
Number of Caches (3)

- One cache too large for best results
- Unified vs. split cache (yhdistetty, erilliset)
 - same cache for data and code, or not?
 - split cache: can optimise structure separately for data and code
- Multiple levels of caches
 - L1 - same chip as CPU
 - L2 - same package or chip as CPU
 - L3 - same board as CPU

Fig. 4.23

21.9.2000 Copyright Teemu Kerola 2000 30

-- End of Ch. 4.3: Cache Memory --



<http://www.intel.com/procs/servers/feature/cache/unique.htm>

“The Pentium® Pro processor's unique multi-cavity chip package brings L2 cache memory closer to the CPU, delivering higher performance for business-critical computing needs.”

21.9.2000

Copyright Teemu Kerola 2000

31