Memory Hierarchy and Cache
Ch 4.1-3

Memory Hierarchy

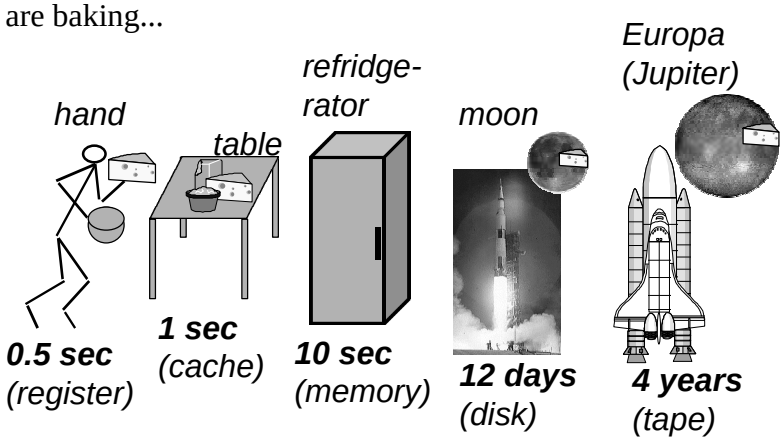
Main Memory

Cache

Implementation

Teemu's Cheesecake

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



Goal (4)

- I want my memory lightning fast
- I want my memory to be gigantic in size

- Register access viewpoint: cache
  - data access as fast as from HW register
  - data size as large as memory HW solution
- Memory access viewpoint virtual memory
  - data access as fast as from memory
  - data size as large as disk HW help for SW solution

21.9.2000 Copyright Teemu Kerola 2000 3

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 Fig. 4.1

up:	smaller, faster, more expensive, more frequent access
down:	bigger, slower, less expensive, less frequent access

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%
Prob (the rest) = 1%

Cost (small data set) = 2 µs
Cost (the rest) = 20 µs

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

Close to cost of small data set

Memory

- Random access semiconductor memory
  - give address & control, read/write data
- ROM, PROMS

Table 4.2

  - system startup memory,
    BIOS (Basic Input/Output System)
    - load and execute OS at boot
  - also random access
- RAM
  - "normal" memory accessible by CPU

RAM

- Dynamic RAM, DRAM
  - simpler, slower, denser, bigger (?)
  - main memory?
  - periodic refreshing required
- Static RAM, SRAM
  - more complex, faster, smaller
  - cache?
  - no periodic refreshing needed
  - data remains until power is lost

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

E.g., 60 ns access

E.g., $100 / MB (year 2000)?

E.g., 5 ns access?

DRAM Access

- 16 Mb DRAM
  - 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.

Fig. 4.4

Fig. 4.5 (b)

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)

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)?

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

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)
  - how to find data in cache?
- Replacement algorithm (poistoalgoritmi)
  - which block to remove to make room for a new block?
- Write policy (kirjoituspolitiikka)
  - how to handle writes?
- Line size (block size)? (rivin tai lohkon koko)
- Number of caches?

21.9.2000 Copyright Teemu Kerola 2000 14

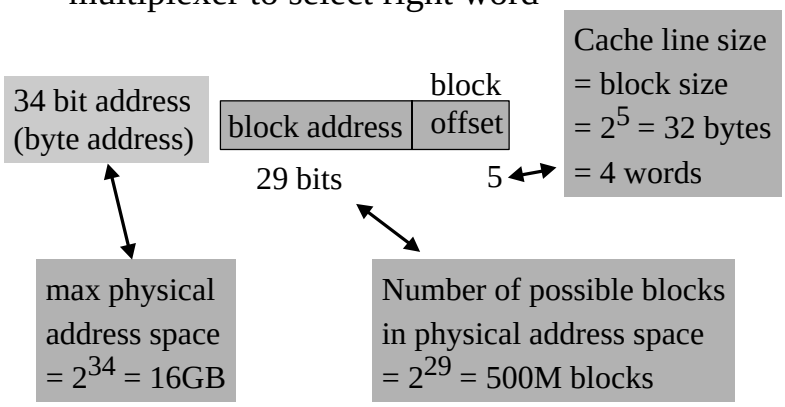
Cache Size

- 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)?

Mapping: Memory Address (3)

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



Direct Mapping (6)

(suora kuvaus)

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

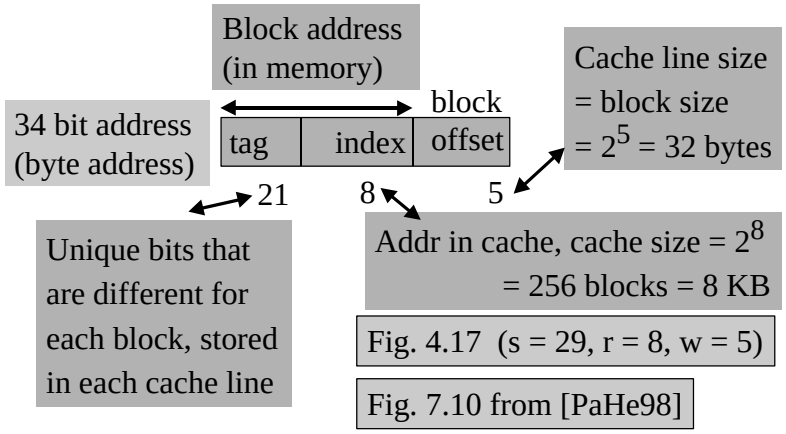
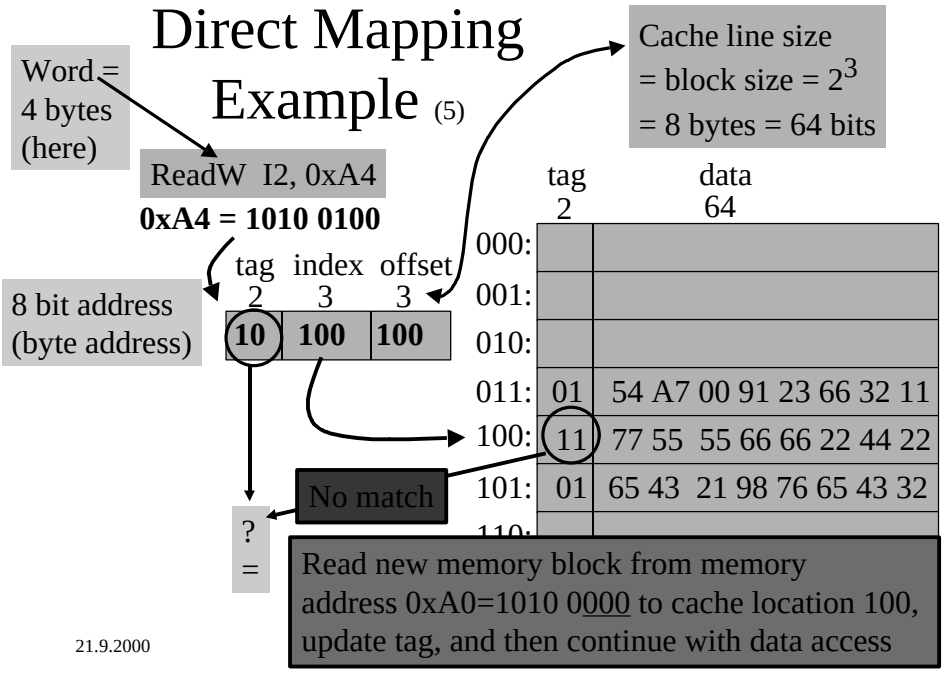


Fig. 4.17 (s = 29, r = 8, w = 5)

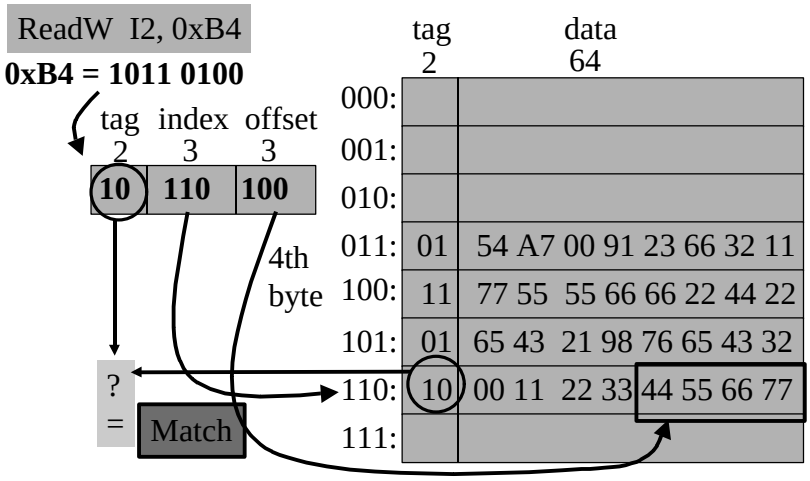
Fig. 7.10 from [PaHe98]

Direct Mapping Example (5)



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

Direct Mapping Example 2 (5)



Fully Associative Mapping (5)

(täysin assosiatiivinen kuvaus)

- Every block can be in any cache line
  - tag must be complete block address

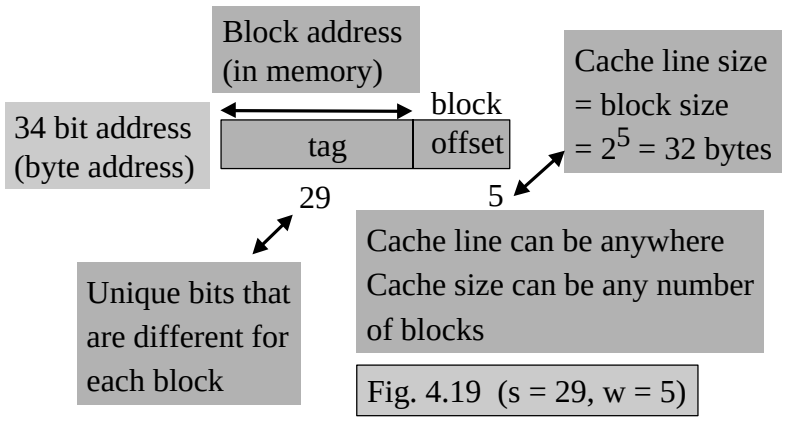


Fig. 4.19 (s = 29, w = 5)

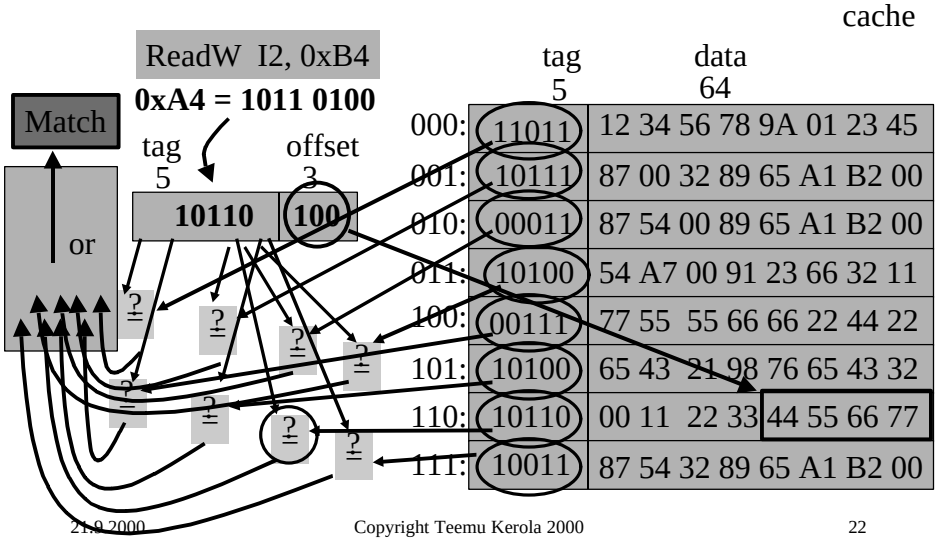
Fully Associative Mapping

- Lots of circuits
  - tag fields are long - wasted space!
  - each cache line tag must be compared simultaneously with the memory address tag
    - lots of wires
    - lots of comparison circuits

    Large surface area on chip
- Final comparison "or" has large gate delay
  - did any of these 64 comparisons match?

    $^2\log(64) = 8$ levels of binary gates
  - how about 262144 comparisons? 18 levels?
- ⇒ Can use it only for small caches

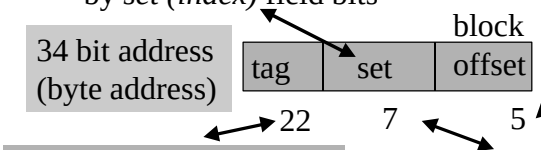
Fully Associative Example (5)



Set Associative Mapping (6)

(joukkoassosiatiivinen kuvaus)

- With set size k=2, every block has 2 possible locations (cache line number) in cache
  - location in each set determined by *set (index)* field bits



34 bit address (byte address)

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

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

v = 2^7 sets, 128 blocks = 4 KB

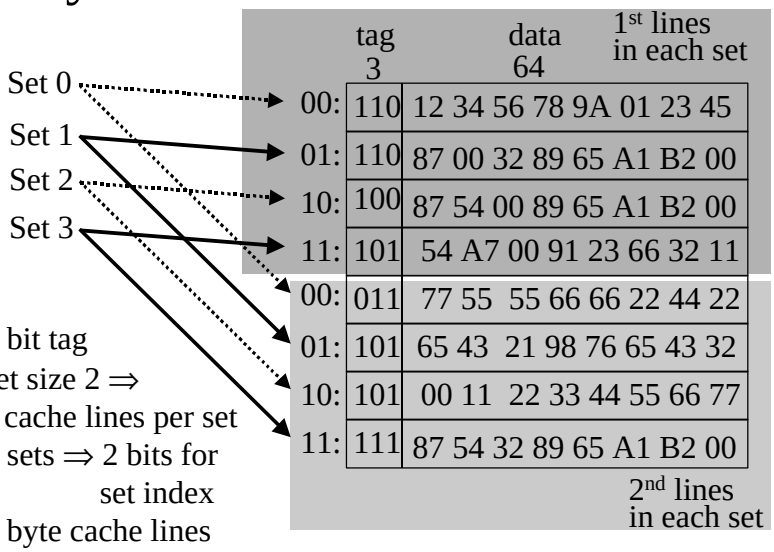
Total cache size = 2*4 KB = 8 KB

Fig. 5.8 from [HePa96]

Fig. 7.19 from [PaHe98]

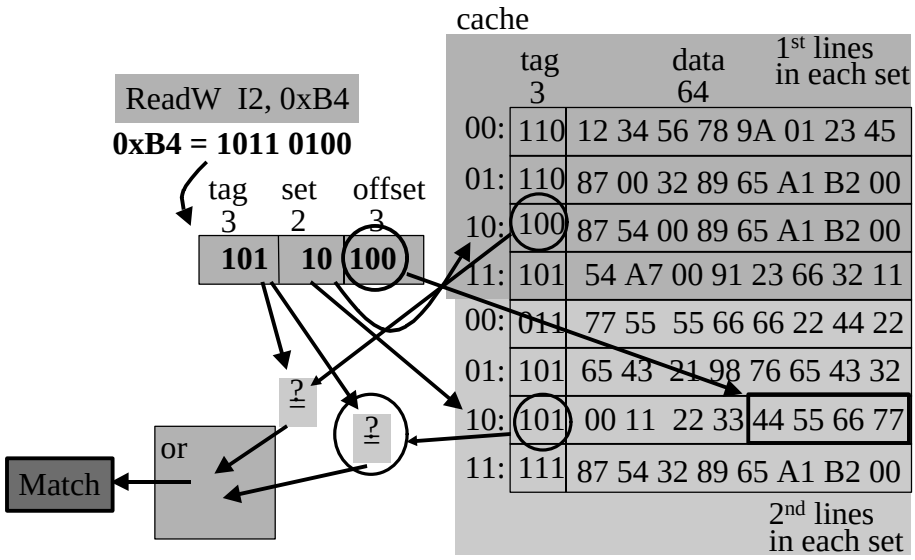
Fig. 4.21 (confusing, complex) (v=2, s = 29, r = 8, w = 5)

2-way Set Associative Cache



- 3 bit tag
- set size 2 ⇒ 2 cache lines per set
- 4 sets ⇒ 2 bits for set index
- 8 byte cache lines
  ⇒ 3 bits for byte address in cache line

Set Associative Example (6)



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

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?

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)

Line size

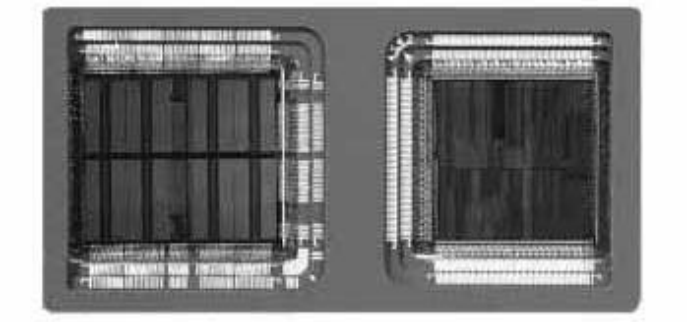
- How big cache line?
- Optimise for temporal or spatial locality?
- Data references and code references behave in a different way
- Best size varies with program or program phase
- 2-8 words?
  - word = 1 float??

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

-- 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."

