

Tietokoneen rakenne

Internal Memory, Cache (välimuisti)

From Computer Outstrip Encyclopedia © 1999 The Computer Language Co. Inc.

Stallings: Ch 4, Ch 5

- Key Characteristics
- Locality
- Cache
- Main Memory

Tietokoneen rakenne / 2007 / Liisa Marttinen 11/9/2007 Luento 4 - 1

Key Characteristics of Memories / Storage

Location	Performance
Processor	Access time
Internal (main)	Cycle time
External (secondary)	Transfer rate
Capacity	Physical Type
Word size	Semiconductor
Number of words	Magnetic
Unit of Transfer	Optical
Word	Magneto-Optical
Block	Physical Characteristics
Access Method	Volatile/nonvolatile
Sequential	Erasable/nonerasable
Direct	Organization
Random	
Associative	

(Sta06 Table 4.1)

Tietokoneen rakenne / 2007 / Liisa Marttinen 11/9/2007 Luento 4 - 2

Goals

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

Register access viewpoint

- data access as fast as HW register
- data size as large as memory

cache
HW solution

Memory access viewpoint

- data access as fast as memory
- data size as large as disk

virtual memory
HW help for SW solution

Tietokoneen rakenne / 2007 / Liisa Marttinen 11/9/2007 Luento 4 - 3

Memory Hierarchy

- Most often needed data kept close
- Access to small data sets can be made fast
 - simpler circuits
 - smaller gate delays
- Faster ~ more expensive
- Large can be bigger and cheaper (per B)

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

(Sta06 Fig 4.1)

Tietokoneen rakenne / 2007 / Liisa Marttinen 11/9/2007 Luento 4 - 4

Principle of locality (paikallisuus)

- In any given time period, memory references occur only to a small subset of the whole address space = 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

- Average cost is close to the cost of small data set
- How to determine data for that small set?
- How to keep track of it?

(Sta06 Fig 4.2)

Tietokoneen rakenne / 2007 / Liisa Marttinen 11/9/2007 Luento 4 - 5

Principle of locality

- In any given time period
 - memory references occur only to a small subset of the whole address space
- Temporal locality** (ajallinen) ○ ○
 - it is likely that a data item referenced a short time ago will be referenced again soon
- Spatial locality** (alueellinen) ○ ○
 - it is likely that a data items close to the one referenced a short time ago will be referenced soon

MEM: 345 233 71 8 305 63 91 2

Tietokoneen rakenne / 2007 / Liisa Marttinen 11/9/2007 Luento 4 - 6

Tietokoneen rakenne

Cache

Tietokoneen rakenne / 2007 / Liisa Marttinen 11/9/2007 Luento 4 - 7

Teemu's Cheesecake

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

0.5 sec (register)
1 sec (cache)
10 sec (memory)
12 days (disk)
4 years (tape)

Tietokoneen rakenne / 2007 / Liisa Marttinen 11/9/2007 Luento 4 - 8

Cache Memory (välimuisti)

How to access main memory as fast as registers?

Locality → Use (CPU) cache!

- Keep most probably referenced data in fast cache close to processor, and rest in memory
- Most of data accesses only to cache
 - hit ratio 0.9-0.99
- Cache is much smaller than main memory
- Cache is (much) more expensive (per byte) than memory

Tietokoneen rakenne / 2007 / Liisa Marttinen 11/9/2007 Luento 4 - 9

Cache

Word Transfer, Block Transfer, CPU, Cache, Main Memory, Memory, Memory address, Block (K words), Line Number, Tag, Block, C-1, Block Length (L words), Word Length, (Sta06 Fig 4.3, 4.4)

Tietokoneen rakenne / 2007 / Liisa Marttinen 11/9/2007 Luento 4 - 10

Cache Read

(RA = Real Address)

```

    graph TD
      START([START]) --> Receive[Receive address RA from CPU]
      Receive --> IsBlock{Is block containing RA in cache?}
      IsBlock -- Miss --> AccessMain[Access main memory for block containing RA]
      IsBlock -- Hit --> Fetch[Fetch RA word and deliver to CPU]
      Fetch --> DONE([DONE])
      AccessMain --> Allocate[Allocate cache line for main memory block]
      Allocate --> WriteDirty{Write 'dirty' cache line back to memory?}
      WriteDirty --> Load[Load main memory block into cache line]
      WriteDirty --> Deliver[Deliver RA word to CPU]
      Load --> Fetch
      Deliver --> DONE
  
```

(Sta06 Fig 4.5)

Tietokoneen rakenne / 2007 / Liisa Marttinen 11/9/2007 Luento 4 - 11

Cache Organization

Processor, Cache, Address buffer, Data buffer, System Bus, Address, Control, Data

(Sta06 Fig 4.6)

Tietokoneen rakenne / 2007 / Liisa Marttinen 11/9/2007 Luento 4 - 12

Cache Design

Cache Size	Write Policy
Mapping Function	Write through
Direct	Write back
Associative	Write once
Set Associative	Line Size
Replacement Algorithm	Number of caches
Least recently used (LRU)	Single or two level
First in first out (FIFO)	Unified or split
Least frequently used (LFU)	
Random	

Cache Size & Line Size

- Many blocks help for temporal locality
- Large blocks help for spatial locality
- Larger cache is slower
- Multi-level cache

Typical sizes:
L1: 8 KB - 64 KB
L2: 256KB - 8 MB

(Sta06 Table 4.2)

Tietokoneen rakenne / 2007 / Liisa Marttinen 11/9/2007 Luento 4 - 13

Mapping

- Which block contains the memory location?
- Is the block in cache?
- Where is it located?

Solutions

- direct mapping (suora kuvaus)
- fully associative mapping (täysin assosiatiivinen)
- set associative mapping (joukko assosiatiivinen)

Cache simulation tools:
<http://www.ecs.umass.edu/ece/koren/architecture/Cache/frame0.htm>

Tietokoneen rakenne / 2007 / Liisa Marttinen 11/9/2007 Luento 4 - 14

Direct Mapping (4)

- Each block has only one possible location (line) in cache
 - determined by index field bits
- Several blocks may map into same cache line
 - identified with tag field bits

Block number (in memory): 0x2480, 0x6480, 0xA480

34 bit address (byte address): tag (21), index (8), offset (5)

Cache line size ~ Block size = $2^5 = 32$ B

Unique bits that are different for each block. Stored into cache line. ~2 million different!

Fixed location in cache
~ fixed cache size = $2^8 = 256$ blocks = 8 KB

(Sta06 Fig 4.7, PaHe98 Fig 7.10)

Tietokoneen rakenne / 2007 / Liisa Marttinen 11/9/2007 Luento 4 - 15

Main memory block (a' 32 B) groups

Cache lines

Tag Data

0 1 2 3 4 ... 255

34 b address = memory size ~ 17 GB

Tietokoneen rakenne / 2007 / Liisa Marttinen 11/9/2007 Luento 4 - 16

Direct Mapping Example (5)

Word = 4B (here)

Block size = $2^3 = 8$ bytes = 64 bits

Cache line size

ReadW L2, 0xA4

8 bit address (byte address): 10 100 100

tag	index	offset	block, 64b
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:			
111:			

No match

compare

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

Tietokoneen rakenne / 2007 / Liisa Marttinen 11/9/2007 Luento 4 - 17

Direct Mapping Example 2 (5)

ReadW L2, 0xB4

8 bit address (byte address): 10 110 100

tag	index	offset	block 64
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:	10	00 11 22 33 44 55 66 77	
111:			

start with 4th byte

compare

Match

Tietokoneen rakenne / 2007 / Liisa Marttinen 11/9/2007 Luento 4 - 18

Fully Associative Mapping (6)

- Each block can be in any cache line
 - tag must be complete block number

Alpha AXP uses 34 bit memory addresses

34 bit address (byte address)

Block number (in memory)

tag 29

Offset from the beginning of the block (in bytes)

offset 5

Block size = $2^5 = 32 B$

Unique bits that are different for each block

Each block can be anywhere Cache size can be any number of blocks

Sta06 Fig 4.9

Tietokoneen rakenne / 2007 / Liisa Marttinen 11/9/2007 Luento 4 - 19

Fully Associative Example (4)

ReadW I2, 0xB4

tag 5

offset 3

tag 5

block 64

000: 1101 12 34 56 78 9A 01 23 45

001: 1011 87 00 32 89 65 A1 B2 00

010: 0001 87 54 00 89 65 A1 B2 00

011: 1010 54 A7 00 91 23 66 32 11

100: 0011 77 55 55 66 66 22 44 22

101: 1010 65 43 21 98 76 65 43 32

110: 1011 00 11 22 33 44 55 66 77

111: 1001 87 54 32 89 65 A1 B2 00

Parallel!

Match

Tietokoneen rakenne / 2007 / Liisa Marttinen 11/9/2007 Luento 4 - 20

Fully Associative Mapping

- Lots of circuits
 - tag fields are long - wasted space?
 - each cache line tag must be compared parallelly with the memory address tag
 - lots of wires, comparison circuits
 - large surface area on chip
- Final comparison "or" has large gate delay
 - did any of these 64 comparisons match?
 - $\log_2(64) = 6$ levels of binary OR-gates
 - how about 262144 comparisons?
 - 18 levels?

Can use it only for small caches

Tietokoneen rakenne / 2007 / Liisa Marttinen 11/9/2007 Luento 4 - 21

Set Associative Mapping

- With set size $k=2$, each cache entry contain 2 blocks
 - Use set (set index) field to find the cache entry
 - Use tag to determine if the block belongs to the set
 - Use offset to find the proper byte in the block

34 bit address (byte address)

tag 22

set 7

offset 5

Block size = $2^5 = 32 B$

Unique bits that are different for each block, stored with block

Nr of sets = $v = 2^7 = 128$ blocks = 4 KB

Total cache size = $k*v = 2*4 KB = 8 KB$ (without tag bits!)

Sta06 Fig 4.11

PaHe98 Fig 7.19

Tietokoneen rakenne / 2007 / Liisa Marttinen 11/9/2007 Luento 4 - 22

2-way Set Associative Cache

- $k=2$ g Two blocks in each set (= in one cache entry)
- 4 sets g 2 bits for set index
- 2 words in a block = 8 Bytes g 3 bits for byte offset
- 3 bits for tag

3 2 3

tag set offset

8 bit address (byte address)

set tag block tag block

00: 110 12 34 56 78 9A 01 23 45 011 77 55 55 66 66 22 44 22

01: 110 87 00 32 89 65 A1 B2 00 101 65 43 21 98 76 65 43 32

10: 100 87 54 00 89 65 A1 B2 00 101 00 11 22 33 44 55 66 77

11: 101 54 A7 00 91 23 66 32 11 111 00 11 22 33 44 55 66 77

3 64 3 64

Tietokoneen rakenne / 2007 / Liisa Marttinen 11/9/2007 Luento 4 - 23

2-way Set Assoc. Cache Example (5)

ReadW I2, 0xB4

tag set offset

101 10 100

set tag block tag block

00: 110 12 34 56 78 9A 01 23 45 110 77 55 55 66 66 22 44 22

01: 110 87 00 32 89 65 A1 B2 00 101 65 43 21 98 76 65 43 32

10: 100 87 54 00 89 65 A1 B2 00 101 00 11 22 33 44 55 66 77

11: 101 54 A7 00 91 23 66 32 11 111 00 11 22 33 44 55 66 77

3 64 3 64

Parallel!

Match

Tietokoneen rakenne / 2007 / Liisa Marttinen 11/9/2007 Luento 4 - 24

Set Associative Mapping

- n Set associative cache with set size $k=2$
= 2-way cache (common)
- n Degree of associativity = nbr of blocks in a set = v
 - u Large degree of associativity?
 - § More data items in one set
 - § Less "collisions" within set
 - § Final comparison (matching tags?) gate delay?
 - u Maximum (nr of cache lines)
 - fully associative mapping Whole cache is one set!
 - direct mapping Each cache line is a set!

Tietokoneen rakenne / 2007 / Liisa Marttinen 11/9/2007 Luento 4 - 25

Cache Replacement Algorithm

- n Which cache block to replace to make room for new block from memory?
- n Direct mapping: trivial
- n First-In-First-Out (FIFO)?
- n Least-Recently-Used (LRU)?
- n Least-Frequently-Used (LFU)?
- n Random?
- n Which one is best / possible?
 - u Chip area?
 - u Fast? Easy to implement?

Tietokoneen rakenne / 2007 / Liisa Marttinen 11/9/2007 Luento 4 - 26

Cache Write Policy - memory writes?

- n **Write through** (läpikirjoittava)
 - u Each write goes always to cache and memory
 - u Each write is a cache miss!
- n **Write back** (lopuksi/takaisin kirjoittava)
 - u Each write goes only to cache
 - u Write cache block back to memory only when it is replaced in cache A bit set
 - u Memory may have stale (old) data
 - o cache coherence problem (yhdenmukaisuus, yhtäpitävyys)
- § **Write once** ("vain kerran kirjoittava?")
 - § Write-invalidate Snoopy-cache coherence protocol for multiprocessors
 - § Write invalidates data in other caches
 - § Write to memory at replacement time, or when some other cache needs it (has read/write miss)

Coherence problems:

- More users of the same data: memory valid? cache valid?
- multiple processors with own caches

Tietokoneen rakenne / 2007 / Liisa Marttinen 11/9/2007 Luento 4 - 27

Cache Line Size

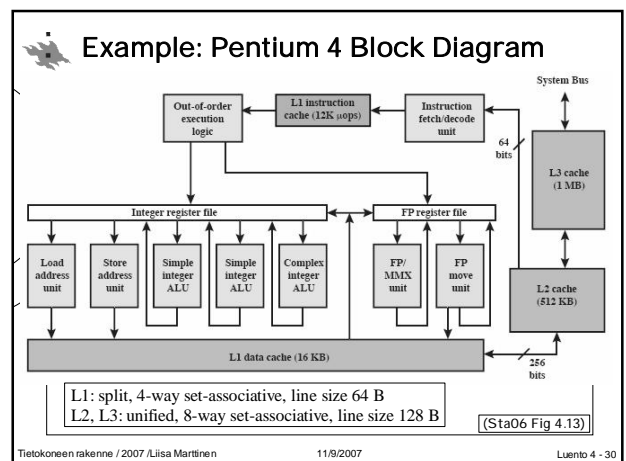
- n How big cache line?
- n Optimise for temporal or spatial locality?
 - u bigger cache line
 - better for spatial locality
 - u more cache lines
 - better for temporal locality
- n Best size varies with program or program phase?
- n Best size different with code and data?
- n 2-8 words?
 - u word = 1 float??

Tietokoneen rakenne / 2007 / Liisa Marttinen 11/9/2007 Luento 4 - 28

Types and Number of Caches

- n Same cache for **data** and **code**, or not?
 - u Data references and code references behave differently
- n **Unified vs. split cache** (yhdistetty/erilliset)
 - u split cache: can optimise structure separately for data and code Trend towards split caches: Pentium, Power PC... (instruction pipelining)
- n One cache too large for best results
- n Multiple levels of caches
 - u L1 on same chip as CPU
 - u L2 on same package or chip as CPU
 - § older systems: same board
 - u L3 on same board as CPU

Tietokoneen rakenne / 2007 / Liisa Marttinen 11/9/2007 Luento 4 - 29



Tietokoneen rakenne

Main Memory

Tietokoneen rakenne / 2007 / Liisa Marttinen 11/9/2007 Luento 4 - 31

Main Memory Types

Memory Type	Category	Erasure	Write Mechanism	Volatility
Random-access memory (RAM)	Read-write memory	Electrically, byte-level	Electrically	Volatile
Read-only memory (ROM)	Read-only memory	Not possible	Masks	Nonvolatile
Programmable ROM (PROM)			Electrically	
Erasable PROM (EPROM)	Read-mostly memory	UV light, chip-level	Electrically	Nonvolatile
Electrically Erasable PROM (EEPROM)		Electrically, byte-level		
Flash memory		Electrically, block-level		

(Sta06 Table 5.1)

- Random access semiconductor memory
 - Direct access to each memory cell
 - Access time same for all cells

Tietokoneen rakenne / 2007 / Liisa Marttinen 11/9/2007 Luento 4 - 32

RAM

- Dynamic RAM, DRAM
 - Periodic refreshing required
 - Refresh required after read
 - Simpler, slower, denser, bigger (bytes per chip)
 - Access time ~ 60 ns
 - Main memory? (early systems)
- Static RAM, SRAM
 - No periodic refreshing needed
 - Data remains until power is lost
 - More complex (more chip area/byte), faster, smaller
 - Access time ~ 2-5 ns
 - Level 2 cache?

Tietokoneen rakenne / 2007 / Liisa Marttinen 11/9/2007 Luento 4 - 33

DRAM Access, 16 Mb DRAM (4M x 4)

(Sta06 Fig 5.3)

Tietokoneen rakenne / 2007 / Liisa Marttinen 11/9/2007 Luento 4 - 34

256-KB DRAM Memory Organization

- Simultaneous access to 256K 8-bit word memory chip to access larger data items
- Access 64-bit words in parallel? Need 8 chips.

(Sta06 Fig 5.5)

Tietokoneen rakenne / 2007 / Liisa Marttinen 11/9/2007 Luento 4 - 35

SDRAM (Synchronous DRAM)

- CPU clock synchronizes also the bus
 - Runs on higher clock speeds than ordinary DRAM
 - CPU knows how long it takes to make a reference, can do other work while waiting
- 16 bits in parallel
 - Access 4 DRAMs (4 bits each) in parallel
 - Access time ~ 18 ns, transfer rate ~ 1.3 GB/s
- DDR SDRAM, double data rate
 - Current main memory technology
 - Supports transfers both on rising and falling edge of the clock cycle
 - Consumes less power
 - Access time ~ 12 ns, transfer rate ~ 3.2 GB/s

Tietokoneen rakenne / 2007 / Liisa Marttinen 11/9/2007 Luento 4 - 36

Rambus DRAM (RDRAM)

(Sta06 Fig 5.14)

- n Works with fast Rambus memory bus (800Mbps)
 - u Controller + RDRAM modules
 - u Access time - 12 ns, transfer rate ~ 4.8 GB/s
- n Speed slows down with many memory modules
 - u Serially connected on Rambus channel
 - u Not good for servers with 1 GB memory (for now!)

STI Cell processor

Tietokoneen rakenne / 2007 / Liisa Marttinen 11/9/2007 Luento 4 - 37

Flash memory

- n Based on transistors that are separated by a thin oxide layer
 - u Flash cell is analog, not digital storage: uses different charge levels to store 2 (or more) bits in each cell
- n Non-volatile, data remains with power off
 - u Electrical erasing in blocks = "flash"
 - u Slow to write
 - u Access time ~ 50 ns
- n Used as a solid state storage
 - u No moving parts
 - u FlashBIOS in PC's, USB-memory
 - u In phones, digital cameras, hand-held devices,....

Tietokoneen rakenne / 2007 / Liisa Marttinen 11/9/2007 Luento 4 - 38

MRAM

- n Magnetoresistive Random Access Memory (MRAM)
 - u Data stored with magnetic fields on two plates
 - u Magnetic field directions determine bit value
- n Non-volatile, data remains with power off
 - u Fast to read/write
 - u No upper limit for write counts (Flash has upper limit)
 - u Access time comparable to DRAM
 - u Almost as fast as SRAM
- n Future open
 - u Small market share now
 - u Expensive now (2006: \$25 4Mbit)
 - u Still under development
 - u May replace flash in a few years
 - u May replace SRAM later on
 - u May replace DRAM and become "universal memory"

MRAM write operation

<http://www.research.ibm.com/journal/rd/501/maffitt.html>

Tietokoneen rakenne / 2007 / Liisa Marttinen 11/9/2007 Luento 4 - 39

Kertauskysymyksiä

- n Muistihierarkia ja paikallisuus?
- n Millä tavoin paikallisuutta huomioidaan välimuistiratkaisussa?
- n Assosiativisen ja joukkoassosiativisen kuvauksen erot?
- n Miksi käskyille oma välimuisti ja datalle oma?

Tietokoneen rakenne / 2007 / Liisa Marttinen 11/9/2007 Luento 4 - 40