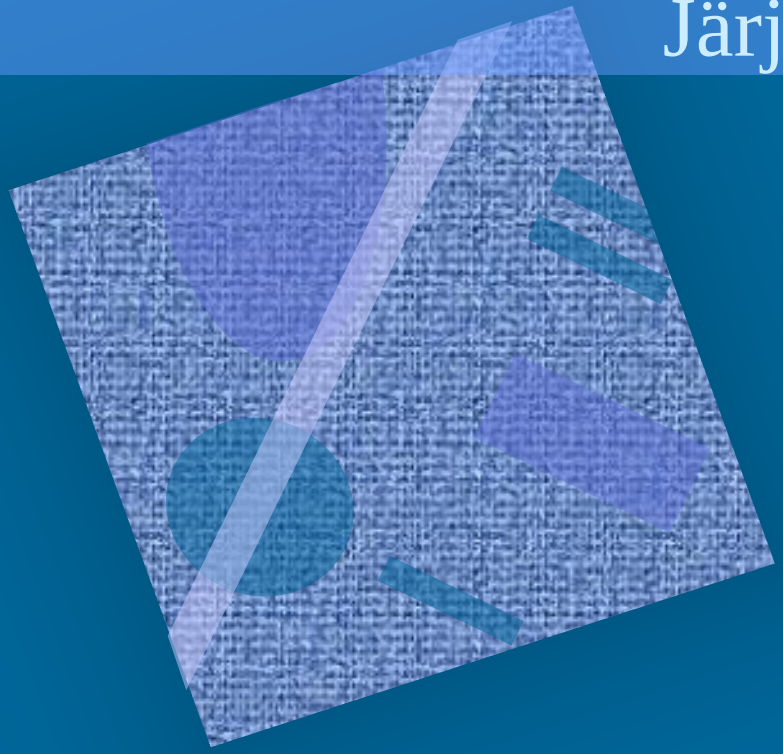
Luentokerta 4

# Tiedon esitysmuodot
# Tiedon muuttumattomuuden tarkistus
# Järjestelmän sisäinen muisti



Lukujärjestelmät

Kokonaisluvut, liukuluvut

Merkit, merkkijonot

Ohjelman esitysmuoto

Rakenteellinen tieto

Pariteetti, Hamming-koodi

Välimuisti, muisti

# Tiedon tyypit

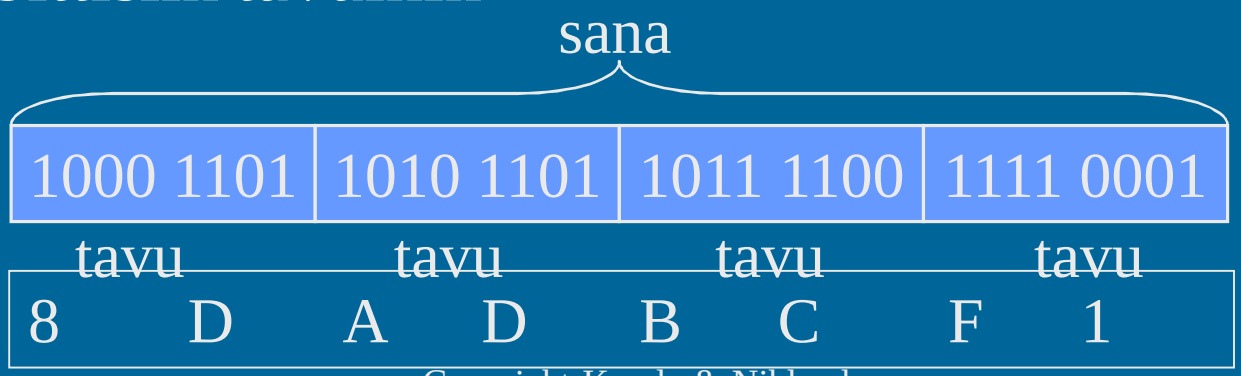
- Kommunikointi ihmisen kanssa
  - kuva, ääni, merkit, …
- Laitteiston sisäinen talletus
  - kuvaformaatit, ääniformaatit, pakkausstandardit, ...
  - kokonaisluvut, liukuluvut, merkit, merkistöt
  - ohjelmat
- Suorittimen omana lajinaan ymmärtämät tyypit
  - on olemassa konekäskyjä tälle tietotyypille
  - kokonaisluvut
  - liukuluvut (useimmat suorittimet nykyään)
  - totuusarvot (jotkut suorittimet)
  - merkit (jotkut suorittimet)
  - konekäskyt

# Tiedon esitys

- Kysymys: miten esittää eri tyyppisiä tietoja?
- Vastaus: koodataan ne biteiksi
  - kaikki tieto on koneessa bitteinä
- Kaikelle käsitellylle tiedolle on omat koodausmenetelmänsä
  - kaikkia koodausmenetelmiä ei ole standardoitu
  - samalle tietotyypille voi olla useita koodausmenetelmiä
    - kokonaisluvut, liukuluvut, merkit, merkkijonot, kuvat, ...
  - ongelma: ymmärtävätkö koneet toisiaan?
    - tiedon esitysmuotoa voidaan joutua muuttamaan, kun tietoa siirretään koneelta toiselle

# Tiedon esitys laitteistossa

- Kaikki tieto koneessa on binääribitteinä (0 tai 1)
  - binäärijärjestelmän numerot: 0, 1
  - helppo toteuttaa piireillä
  - helppo suunnitella logiikkaa Boolen algebran avulla
- Muisti jaettu tasapituisiin sanoihin (word)
  - sana = word = 32 bittiä (16 bittiä, 64 bittiä, …)
- Yleensä sana on jaettu tasapituisiin 8-bittisiin tavuihin (byte)



# Suorittimen ymmärtämä tieto

- Kaikki tieto koneessa on koodattuna biteiksi
- Muistissa voidaan esittää kaikki tieto millä tahansa sovitulla esitystavalla (koodauksella)
- Suoritin osaa tehdä operaatioita joillakin esitystavoilla koodatuille tiedoille
  - kokonaisluvut ja liukuluvut (lähes aina)
  - totuusarvot, merkit ja merkkijonot (joskus)
  - kuvat ja äänet (ei yleensä ellei erikoistunut suoritin)
  - hajut (ei vielä)
- Muiden tietojen käsittely tapahtuu ohjelmallisesti
  - esim. merkkejä (merkkien esitysmuotoa) voidaan käsitellään kokonaislukuoperaatioilla ja aliohjelmilla

TTK-91: kokonaisluvut

# Numeromuunnokset

- Binääriluvuista kymmenjärjestelmään

  10110011 ⇨ 179

- Kymmenjärjestelmästä binääriluvuiksi

  179 ⇨ 10110011

- Binääriluvusta heksadesimaaliluvuksi

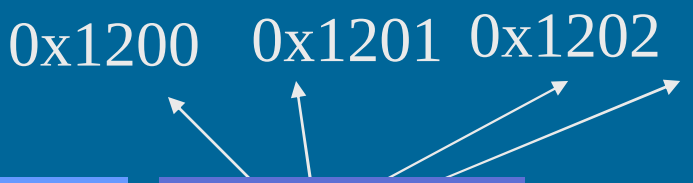
  10110011 ⇨ B3

- Heksadesimaaliluvusta binääriluvuksi

  B3 ⇨ 10110011

# Big vs. Little Endian

- Miten monitavuiset arvot talletetaan?

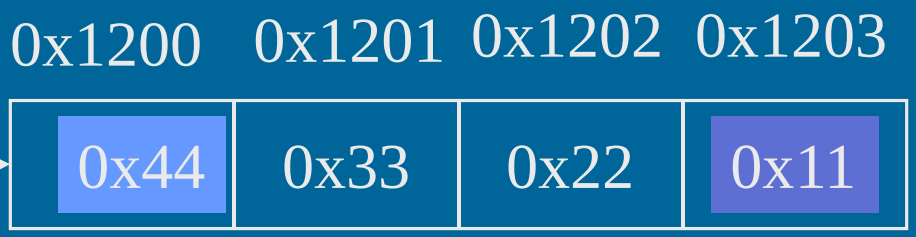




Sanan osoite

talleta 0x11223344 ??

tavuosoitteet

Big-Endian: eniten merkitsevä tavu pienimpään osoitteeseen


Little-Endian: vähiten merkitsevä tavu pienimpään osoitteeseen

# Negatiiviset luvut

arvo talletus

+57 = 0011 1001

- Etumerkkibitti erikseen

sign bit = MSB = most significant bit

luku -57 = 1011 1001 talletus-muoto

- Yhden komplementtiesitys

-57 = 1100 0110

"sign" bit

+1

- Kahden komplementtiesitys

-57 = 1100 0111

"sign" bit

- Vakiolisäys
  - Esim lisää 127 (=$2^8$ –1)
    - yleensä: $2^{\text{bittilkm}}$ -1
  - Talleta etumerkittömänä

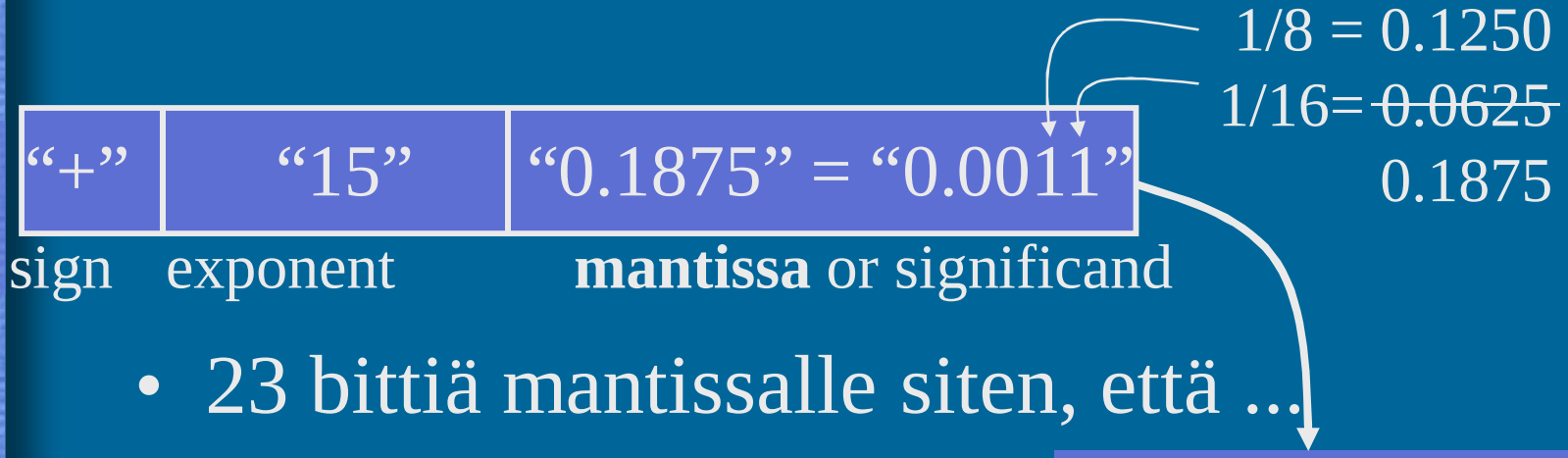
-57 = 0100 0110

-57 + 127 =70

+57 = 1011 1000

+57+127=184

# IEEE 32-bit FP Standard




- 23 bittiä mantissalle siten, että ...

1) Binääripiste (.) on heti ensimmäisen bitin jälkeen

2) Mantissa on normalisoitu: vasemmanpuolimmainen bitti on 1

3)Vasemmanpuolimmaista (eniten merkitsevä) bittiä (1) ei talleteta (implied bit, piilobitti)

| mantissa | eksponentti |
|---|---|
| 0.0011 | “15” |
| 1.1000 | “12” |
| **1000** | “12” |

24 bitin mantissa!

Miksi käytetään piilobittiä?

# UCS ja Unicode

- UCS - Universal Character Set
- Samat merkistöt, eri standardit
- 2 tavua eli 16 bittiä per merkki
  - 65536 merkkiä koko maailmassa käytössä oleville n. 200000 symbolille
- Kontrollimerkit
  - 0x0000-001F and 0x0080-009F
  - 0x007F = DELETE,  0x0020 = SPACE
- UCS:ssä myös 8-bittiset koodi ”rivit”
  - eri alueille tai tarkoituksiin (zone) omat 8-bittiset koodinsa

# Merkkijonot

- Yleensä peräkkäin talletettu joukko tavuja
- Lisäksi tarvitsee jollain tavalla koodata merkkijonon pituus
  - laitetaan loppuun erikoismerkki, tai
    - C-kieli: ’\0’ = 0x00
  - toteutetaan tietueena


  - ei omia konekäskyjä, manipulointi aliohjelmilla
    - kokonaisluku- ja bittimanipulointikäskyt
    - joissakin koneissa ”strcopy” ja ”strcmp” konekäskyt

# Konekäskyjen esitysmuoto

- Konekohtainen, jokaisella omansa
- Käskyt ovat 1 tai useamman tavun mittaisia
  - SPARC, kaikki käskyt: 1 sana eli 4 tavua
  - ARM, kaikki käskyt: 1 sana eli 4 tavua
  - Pentium II: 1-16 tavua, paljon variaatioita
- Käskyillä on yksi tai useampi muoto, kussakin tietty määrä erilaisia kenttiä
  - opcode, Ri, Rj, Rk, osoitusmoodi
  - pitkä tai lyhyt vakio

TTK-91, kaikki käskyt: 1 sana, 1 muoto

# Taulukkojen esitysmuoto

- Peräkkäisrakenteena, kuten esimerkit aikaisemmin
- Riveittäin tai sarakkeittain
- Ei omia konekäskyjä, manipulointi aliohjelmilla tai toistorakenteilla
  Poikkeus: vektorisuorittimet, joilla
  - vektorirekisterit (esim. 64 liukulukua) tavallisten rekistereiden lisäksi
  - omia konekäskyjä vektorioperaatioita varten
- Indeksoitu tiedonosoitusmoodi tukee 1-ulotteisten taulukoiden käyttöä

# Tietueiden esitysmuoto

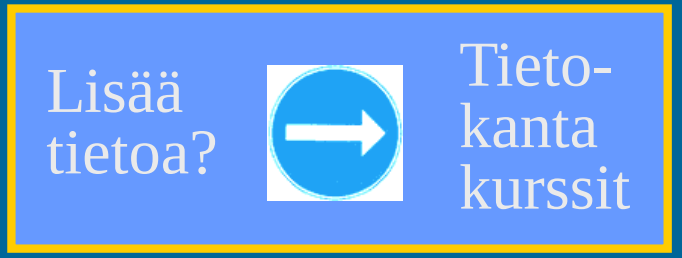
- Peräkkäisrakenteena
- Osoite on jonkin osoitinmuuttujan arvo
- Ei omia konekäskyjä, manipulointi aliohjelmilla tai kääntäjän generoimien vakiolisäysten avulla
- Indeksoitu tiedonosoitusmoodi tukee tietueiden käyttöä

# Olioiden esitysmuoto

- Kuten tietueet, yleensä varattu keosta (heap)
- Useat olion kentistä sisältävät vuorostaan osoitteen keosta suoritusaikana varattuun toiseen olioon
- Metodit ovat aliohjelmien osoitteita
- Ei omia konekäskyjä, manipulointi aliohjelmilla

# Tiedon tarkistus

- Tiedon oikeellisuutta ei voi tarkistaa yleisessä tapauksessa
- Laitteistovirheitä voidaan havaita ja joskus automaattisesti korjata
  - bitti voi muuttua muistissa tai tiedon siirrossa
    - muistipiirissä voi olla vika (staattinen vika)
    - sopiva alkeishiukkanen voi muuttaa bitin tiedonsiirron aikana (transientti virhe)
  - korjaamattomasta virheestä voi aiheutua häiriö
- Tietokannan eheys on eri asia!



# Tiedon muuttumattomuus

- Perusidea: otetaan mukaan ylimääräisiä bittejä, joiden avulla virheitä voidaan havaita ja ehkä myös korjata
- Järjestelmä suorittaa tarkistukset automaattisesti joko laitteistotasolla tai ohjelmiston avulla

# Bittitason tarkistukset

- Muistipiirit, levyt, väylät, tiedonsiirrot

120464-121C
(merkkejä, ei bittejä)

- Monenko bitin muuttuminen havaitaan?

Hetu: 1

- Monenko bitin muuttuminen voidaan automaattisesti korjata?

Hetu: 0

- Havaitsemiseen ja/tai korjaamiseen tarvitaan enemmän (ylimääräisiä) bittejä
  - lisämuistitilan tai levytilan tarve?
  - lisäpiuhojen tarve väylällä?

Hetu: +10%

- Tarkistukset/korjaukset laitteisto- vai ohjelmistotasolla?

Hetu: ohjelmistotasolla

# Pariteettibitti

- Yksi ylimääräinen bitti per tietoalkio
  - sana, tavu, tietoliikennepaketti
- Parillinen (pariton) pariteetti: 1-bittien lukumäärä on aina parillinen (pariton)
- Havaitsee: 1 bitti
- Korjaa: 0 bittiä
- Esimerkki (parillinen pariteetti)

0010 001 0

1000 1101 1111 001 1

# Hamming etäisyys

- Montako bittiä jossain koodijärjestelmässä (esim ISO Latin-1) esitetyllä koodilla (esim. ´A’ = 0x41 = 0100 0001) täytyy muuttua, että se muuttuu johonkin toiseen (mihin tahansa) lailliseen koodiin.

´A’ = 0x41 = 0100 0001 → (2 bittiä) ´B’ = 0x42 = 0100 0010 → (1 bittiä) ´C’ = 0x43 = 0100 0011

- ISO Latin-1:n Hamming etäisyys: 1
- Pariteettibitin kanssa Hamming etäisyys: 2
  - mikä todennäköisyys 2 bitin (vs. 1 bitin) virheeseen?
  - riittävän pieni?

$(\text{Prob}\{\ \text{”yhden bitin virhe”}\ \})^2$

# Virheen korjaava Hamming koodi

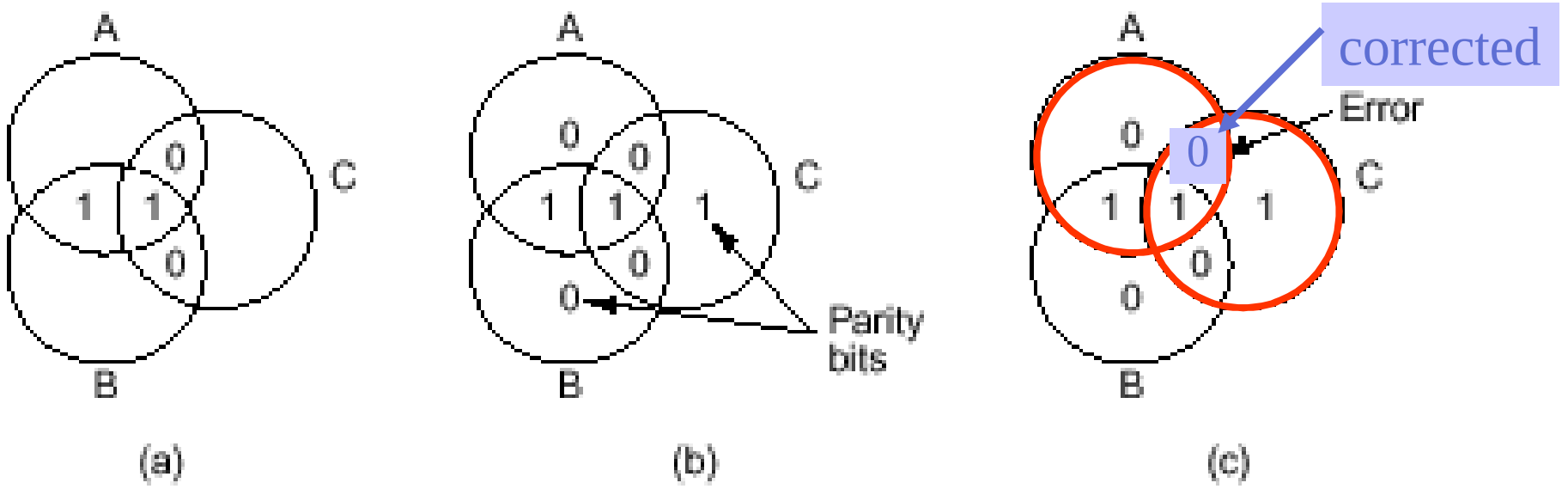



**Figure 2-14.** (a) Encoding of 1100. (b) Even parity added. (c) Error in *AC*.

[Tane99]

(c) Joukot A ja C havaitsevat virheen ja siten paikallistavat virheellisen bitin

(a) Kukin databitti (4 kpl) kuuluu erilaisiin pariteettijoukkoihin (3 kpl)

Täsmälleen 1 databitti identifioituu kerrallaan!

(b) Tarvitaan 3 ”ylimääräistä” bittiä!

entä jos virhe pariteettibitissä?

# Virheen korjaava Hamming koodi

Miten Hamming koodi havaitsee 2 bitin virheen?

| | | | |
|---|---|---|---|
| Data: | 100 1100 | 110 1100 | (parillinen pariteetti) |
| Bitti nro: | 765 4321 | 765 4321 | |

1 = 001
2 = 010
3 = 011
4 = 100
5 = 101
6 = 110
7 = 111

Pariteettibitti 1 tarkistaa bittejä 1, 3, 5, 7

Pariteettibitti 2 tarkistaa bittejä 2, 3, 6, 7

Pariteettibitti 4 tarkistaa bittejä 4, 5, 6, 7

Tapahtuu virhe: bitti 6 muuttuu (flips)

Pariteettibitti 2 tarkistaa bittejä 2, 3, 6, 7: VIRHE

Pariteettibitti 4 tarkistaa bittejä 4, 5, 6, 7: VIRHE

2+4 = 6 ⇒ korjaa bitti nro 6

# CRC - Cyclic Redundancy Code

- Tiedonsiirrossa käytetty tarkistusmenetelmä
- Tarkistussumma (16 bittiä) isolle tietojoukolle
  - laske CRC = f (viesti) % $2^{16}$ (ota 16 viimeistä bittiä)
  - lähetä viesti ja CRC
  - vastaanota viesti ja CRC
  - laske CRC ja tarkista, oliko se sama kuin viestissä
  - jos pielessä, niin pyydä uudelleenlähetystä

CRC-CCITT CRCs detect:
All single- and double-bit errors
All errors of an odd number of bits
All error bursts of 16 bits or less
In summary, 99.998% of all errors

# Virheiden tarkistusmenetelmien käyttöalueet

- Mitä lähempänä suoritinta, sitä tärkeämpää tiedon oikeellisuus on
- Sisäinen väylä, muistiväylä
  - virheet lennossa korjaava Hamming koodi
- Paikallisverkko
  - uudelleenlähetyksen vaativa CRC
  - kun tulee virheitä, niin niitä tulee yleensä paljon
    - Hamming koodi ei riitä kuitenkaan
    - pariteettibitti päästää läpi (esim.) 2 virheen paketit

# Laitteiden monistaminen

- Monta muistipiiriä tai levyä, samat tiedot monistettu
- Monta suoritinta, samat käskyjen suoritukset monistettu
- Monta laitteistoa, samat ohjelmat monistettu
  - äänestysmenettely: enemmistö voittaa
  - monimutkainen, hidas?
  - virheelliseksi havaittu laitteisto suljetaan pois häiriköimästä automaattisesti?
- Eri tai saman tyyppiset laitteistot, samankaltaiset ohjelmat
  - samat speksit, samat syötteet, eri ohjelmoijat

"Four of the five computers (IBM AP-101) on the Columbia ran identical software and compared results with each other before giving the go-ahead to take a specific action. The fifth computer (also IBM AP-101) ran a different version of the software and was used only if the others failed."

**http:**//www.hq.nasa.gov/office/pao/History/computers/contents.html

# Välimuisti (cache)

- Ongelma: keskusmuisti on aika kaukana suorittimesta

rekisterin viittausaika: X
muistin viittausaika: 10X

- Ratkaisu: välimuisti lähelle suoritinta
  - pidetään siellä kopioita viime aikoina viitatuista keskusmuistin alueista

välimuistin viittausaika: 2X

- Jokainen muistiviite on nyt seuraavanlainen
  - jos data ei ole välimuistissa, niin hae se sinne
    - suoritin odottaa tällä aikaa, laitteistototeutus!
  - tee viittaus dataan (käskyyn) välimuistissa
  - (talleta muutettu tieto keskusmuistiin)

# Muistin toteutus

- Eri teknologioita eri tasoisiin muisteihin
- RAM - Random-Access Semiconductor Memory
  - anna osoite ja lue/kirjoita signaali
  - mistä vaan voi lukea/kirjoittaa samassa ajassa
  - virta pois $\Rightarrow$ tiedot häviävät

(volatile memory)

Huom: kaikki nykyiset muistit ovat ”random access”

# RAM:n kaksi eri teknologiaa

- DRAM: dynaaminen RAM, halvempi, hitaampi, tietoja pitää virkistää vähän väliä (esim. joka 2 ms)
  - tavallinen keskusmuisti (1975-..) useimmissa koneissa
  - toteutettu kondensaattoreilla, jotka ”vuotavat” …
- SRAM: staattinen RAM, kalliimpi (~10-20x), nopeampi (~10x), ei vaadi tietojen virkistämistä
  - välimuisti useimmissa koneissa
  - muisti superkoneissa (esim. Cray C-90)
  - toteutettu samanlaisilla logiikkaporteilla (gate) kuin prosessorikin

# Nykyaikaisen keskusmuistin termejä

- SDRAM (synchronous dynamic random access memory)
  - Kello tahdittaa siirrot väylälle
  - Sisäinen puskuri, useita muistioperaatioita jonossa
- DDR (double data rate) SDRAM
  - Yhden kellopulssin aikana kaksi datasiirtoa, sekä nousevalla että laskevalla pulssin osalla

Kello, väylä - Lisää tietoa?

Tietokoneen rakenne

# ROM teknologia

- ROM - Read-Only Memory
  - tieto säilyy virran katkettua (non-volatile)
  - Voi käytönaikana vain lukea, ei voi kirjoittaa
    - esim. järjestelmän alustustiedot (BIOS)
  - kirjoitus lastun valmistusaikana, Mask-ROM
    - ei enää käytössä
  - huono puoli: kerran väärin, aina väärin (ehkä…)
  - päivitys: laita valmistajalta saatu uusi lastu paikalleen
  - tietoa voi lukea mistä vain samassa ajassa (random access)
  - yleensä hitaampi kuin RAM (~10x)

# Kirjoitettavia ROM-muisteja

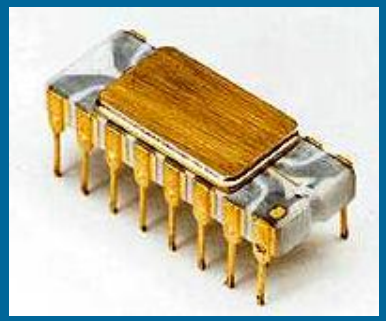
- PROM - Programmable ROM
  - kerran kirjoitettava
  - tiedon päivitys: ”polta” tiedot tyhjään PROM:iin
- EPROM - Erasable PROM
  - tietoja ei voi päivittää sana kerrallaan
  - vanhat tiedot voidaan (kaikki!) poistaa 20 min. UV-säteilyllä, jonka jälkeen päivitetyt tiedot voidaan ladata
- EEPROM - Electronically Erasable PROM
  - tietojen pyyhkiminen tavukohtaisesti elektronisesti
- FLASH EEPROM memory
  - tietojen pyyhkiminen nopeasti kerralla elektronisesti
  - normaalijännitteellä, kaikki tai lohko kerrallaan
  - nopeampi kuin EEPROM

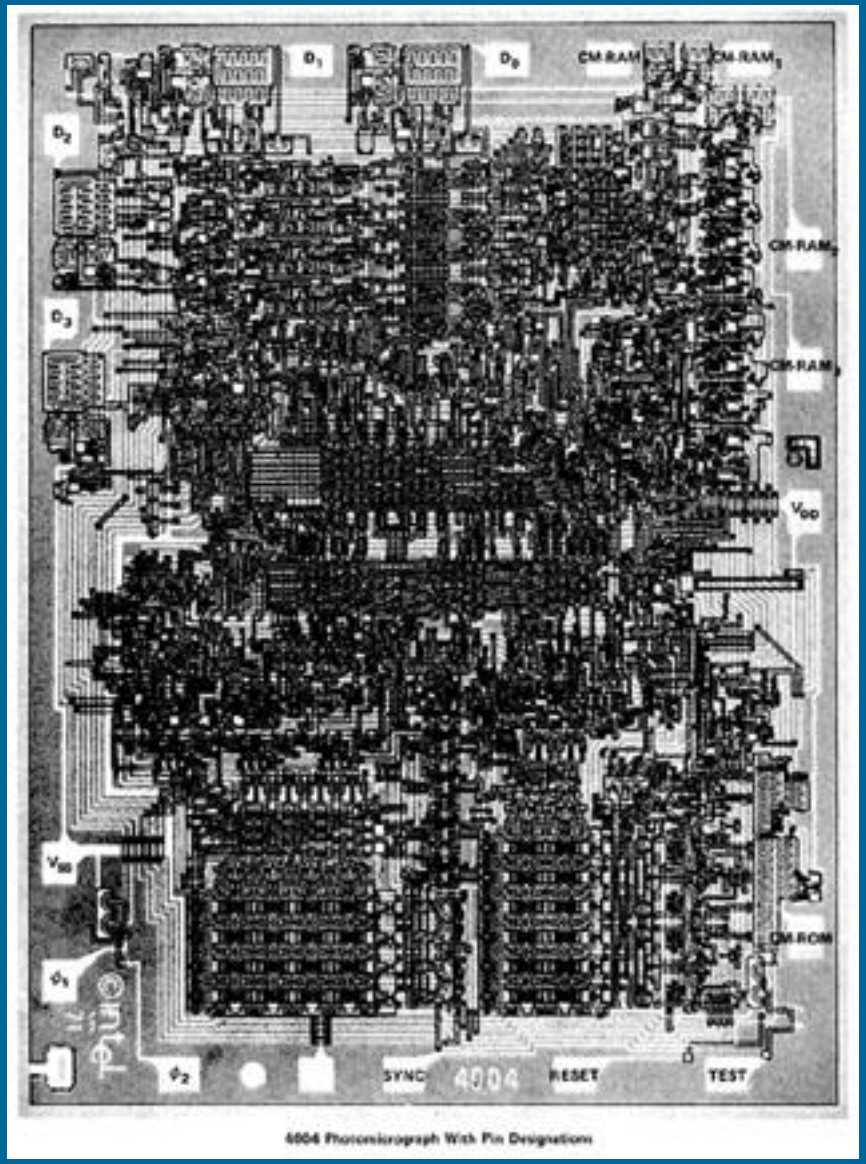
read-mostly memory

# -- Luennon 4 loppu --

## Intel 4004, 1971

- Faggin, Hoff, Mazor
- Ens. suoritin lastulla 3x4 mm, $200
- 2300 transistoria
- 4 bitin sana
- Laskinta varten
- Sama laskentateho kuin Eniacilla (18000 tyhjiöputkea)






4004 Photomicrograph With Pin Designations