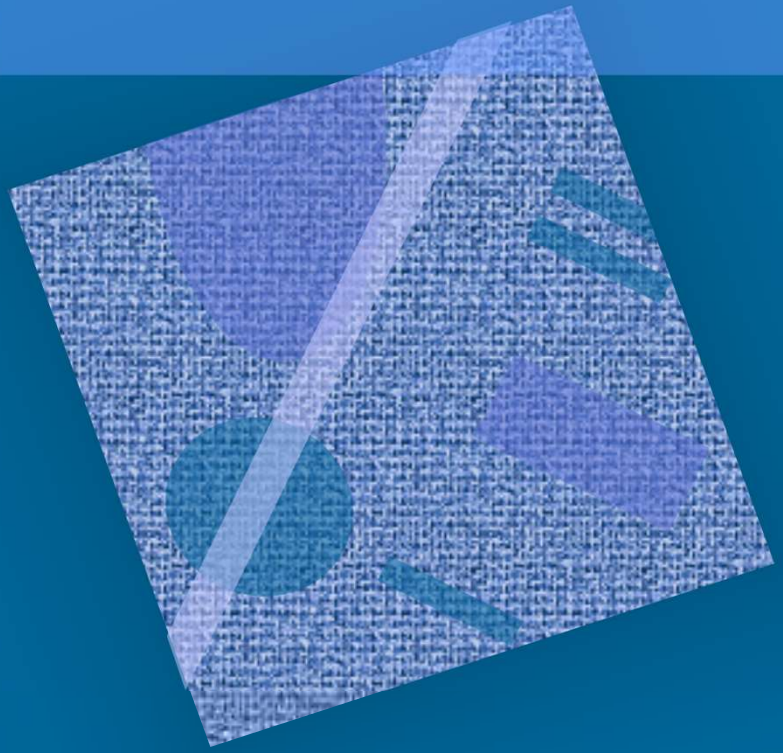
Luvut 7 ja 8
Kertausluento 4

Tiedon muuttumattomuuden tarkistus Järjestelmän sisäinen ja ulkoinen muisti I/O:n toteutus



Pariteetti, Hamming-koodi

Välimuisti, muisti

Muistihierarkia

Tiedostojärjestelmä

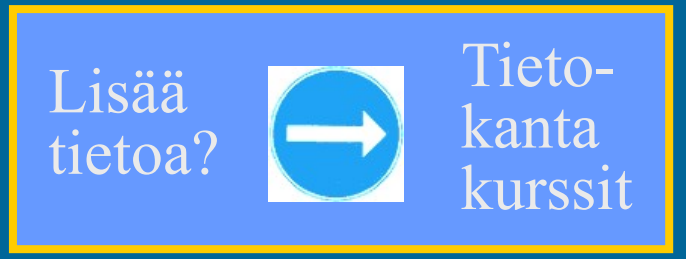
Kiintolevyt, SSD, NVMe

I/O-tyypit

I/O:n toteutus laiteajurin ja laiteohjaimen yhteistyöllä

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 vika)
  - Vian takia tieto muuttuu virheelliseksi (virhe)
  - Korjaamattomasta virheestä voi aiheutua häiriö
- Tietokannan eheys on eri asia!



Vika, virhe, häiriö

- *Rillit huurussa* tv-sarja
  - Penny liukastuu ammeessa?
  - Sheldonin mielestä vika oli liukuestematon puuttuminen, liukastuminen sen aiheuttama virhe ja murtuma virheen aiheuttama häiriö.
- Avaruussukkula
  - Vika yleensä ohjelmoijan koulutuksessa
  - Virhe usein koodissa
    - Luokittelu sen aiheuttaman häiriön perusteella
  - Häiriö voi olla olematon, tai sukkulan tuhoutuminen

Pariteettibitti

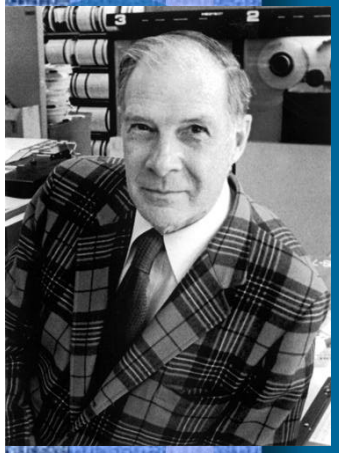
Havaitsee: Kaikki yhden bitin virheet

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

0010 001 0 1000 1101 1111 001 1

pariteettitilantarve: 12.5% 6.25%

Hamming etäisyys (R. Hamming 1950)



- 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

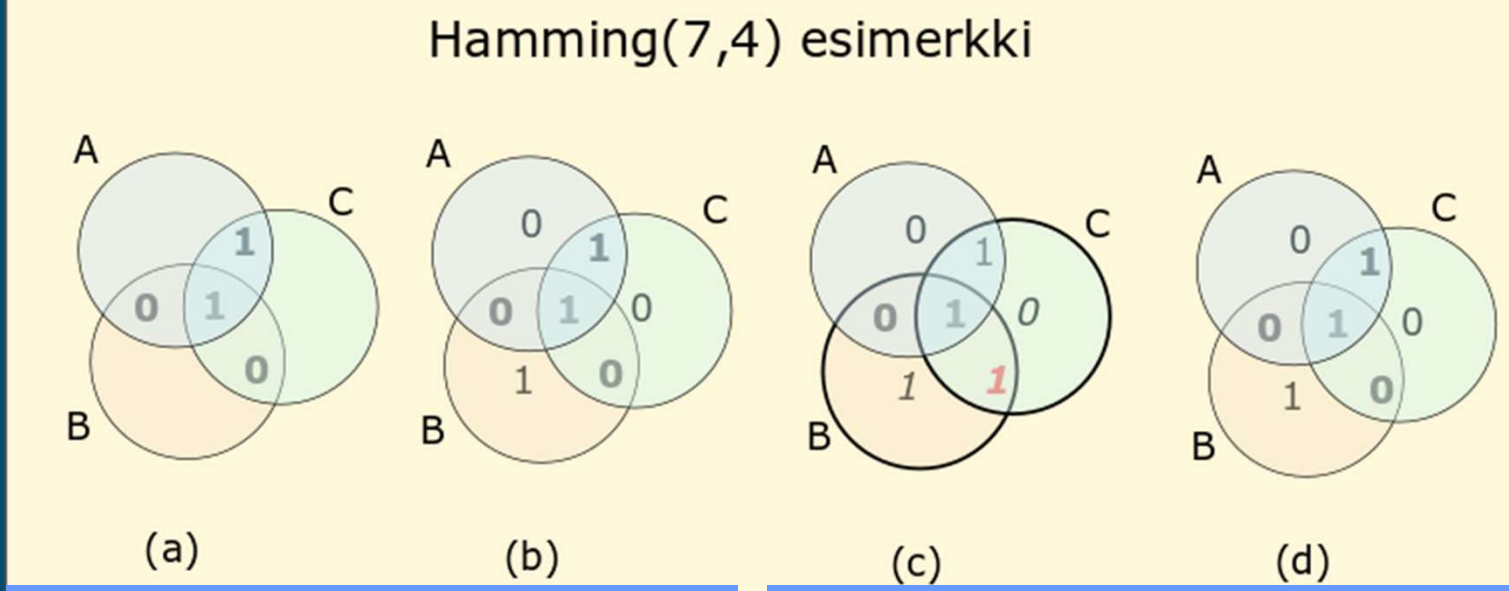
1 bittiä

´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?

$$\text{Prob}\{\text{"2 bitin virhe"}\} = (\text{Prob}\{\text{"1 bitin virhe"}\})^2$$

Virheen korjaavan Hamming koodin toimintaperiaate-esimerkki



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

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

(b) Tarvitaan 3 ”ylimääräistä” bittiä!

(d) Virhe korjataan, tarkistetaan ja raportoidaan

Virheen korjaava Hamming koodi

	oikein		virheellinen
Data:	100 1100	⇒	110 1100
Bitti nro:	765 4321		765 4321

(parillinen pariteetti)

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

Keskustele

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 (laitteistototeutus)
  - virheet lennossa korjaava Hamming-koodi
  - ylim. johtimet pariteettibiteille
  - laitteistopiirit pariteettibittien tarkistamiseen (aika!?)
- Paikallisverkko (ohjelmistototeutus)
  - uudelleenlähetyksen vaativa CRC
  - kun tulee virheitä, niin niitä tulee yleensä paljon
    - Hamming koodi ei riitä kuitenkaan
    - pariteettibitti päästää läpi (esim.) 2 virheen paketit

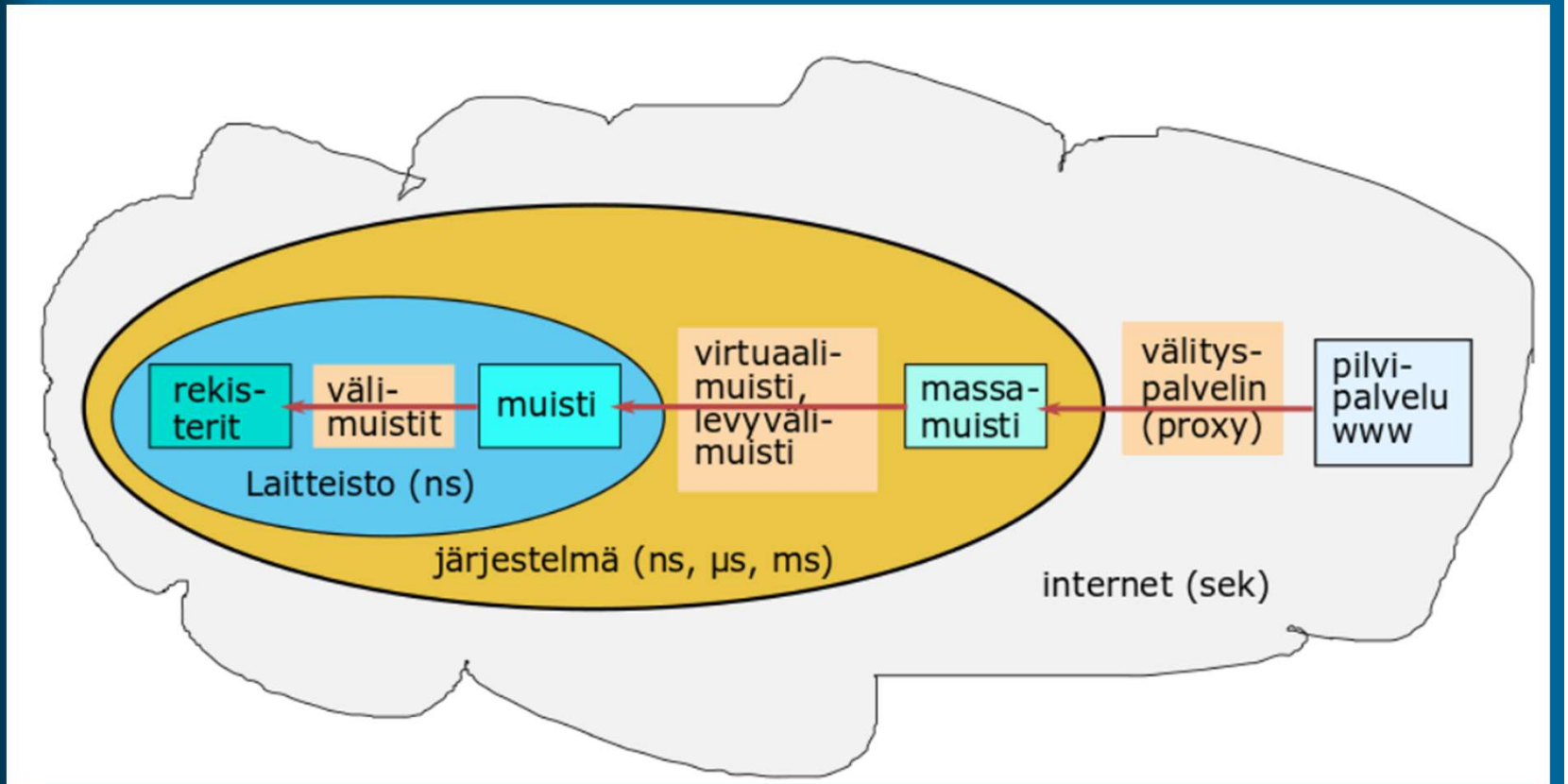
Laitteiden monistaminen

- 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 space shuttle 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

Muistihierarkia



Välimuisti (cache)

- Ongelma: keskusmuisti on aika kaukana suorittimesta

rekisterin viittausaika: X
muistin viittausaika: 200X
(karkealla tasolla)

- 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, ehkä)

Keskustele

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 (~10-50x), vie tilaa enemmän, ei vaadi tietojen virkistämistä
  - välimuisti useimmissa koneissa
  - muisti superkoneissa
  - toteutettu samanlaisilla logiikkaporteilla (gate) kuin prosessorikin

ROM-muisti

- Jotain pysyväismuistia on tarvittu aina järjestelmän käynnistyslohkon toteutukseen
  - Historiaa: ROM, PROM, EPROM, EEPROM, FLASH EEPROM
- Flash muisti
  - Järjestelmässä oleva pienehkö (?) muisti järjestelmän käynnistyslohkolle (BIOS)
    - Voi sisältää myös sulautetun (esim. esineiden internet) järjestelmän kaiken ohjelmiston
  - Firmware-päivitys
  - Suojattu hyvin, jotta virukset eivät pääsisi helpolla käsiksi?
    - Käynnistyslohkovirukset
    - Fyysinen kytkin emolevyllä?

SSD ja NVMe

Solid State Disk

Non-Volatile Memory Express (NVMe, NVM Express)

- SSD
  - Flash-teknologia
  - Yleensä: Käyttöjärjestelmä (KJ) näkee kovalevynä
- NVMe
  - KJ näkee Flash-muistina (ei kovalevynä)
  - Nopeampi, osaa hyödyntää Flash'in samanaikaisuutta
- Lohkot ja sivut
  - Tiedostot (esim.) 4 KB sivuina (vrt. kovalevy)
  - Luku ja kirjoitus (esim.) 512 KB lohkoina
    - Koko lohko pitää tyhjentää ennen kirjoitusta
    - Kirjoitus ehkä uuteen lohkoon
    - Kullakin lohkolla raja kirjoituksille (esim. 100000)
      - Ylimääräisiä lohkoja piirillä

Tiedostojärjestelmä

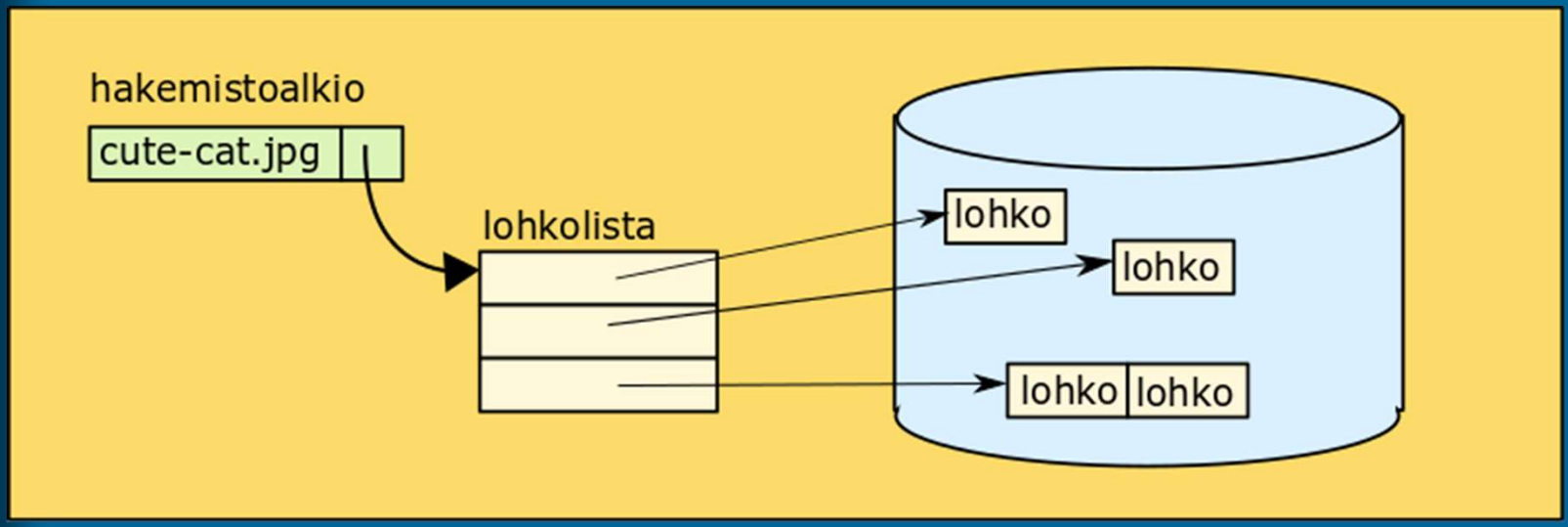
read
write
execute

- KJ:n osa, hallitsee kaikkia tiedostoja
- Valvoo oikeuksia tiedostoa avattaessa
- Muuntaa tiedostonimet fyysisiksi osoitteiksi
- Ylläpitää (vain KJ:n käytössä olevia) taulukoita, joista näkee mitä kohtaa mistäkin tiedostosta kukin prosessi on käsittelemässä
- Tiedostojärjestelmä lukee ja kirjoittaa tiedostoja suurina kerralla käsiteltävinä lohkoina (0.5-8 KB?)
  - käyttäjätason prosessit käsittelevät tiedostoja tavuittain, eikä niiden tarvitse tietää tiedoston todellista fyysistä rakennetta (KJ:n laiteajuri huolehtii siitä)

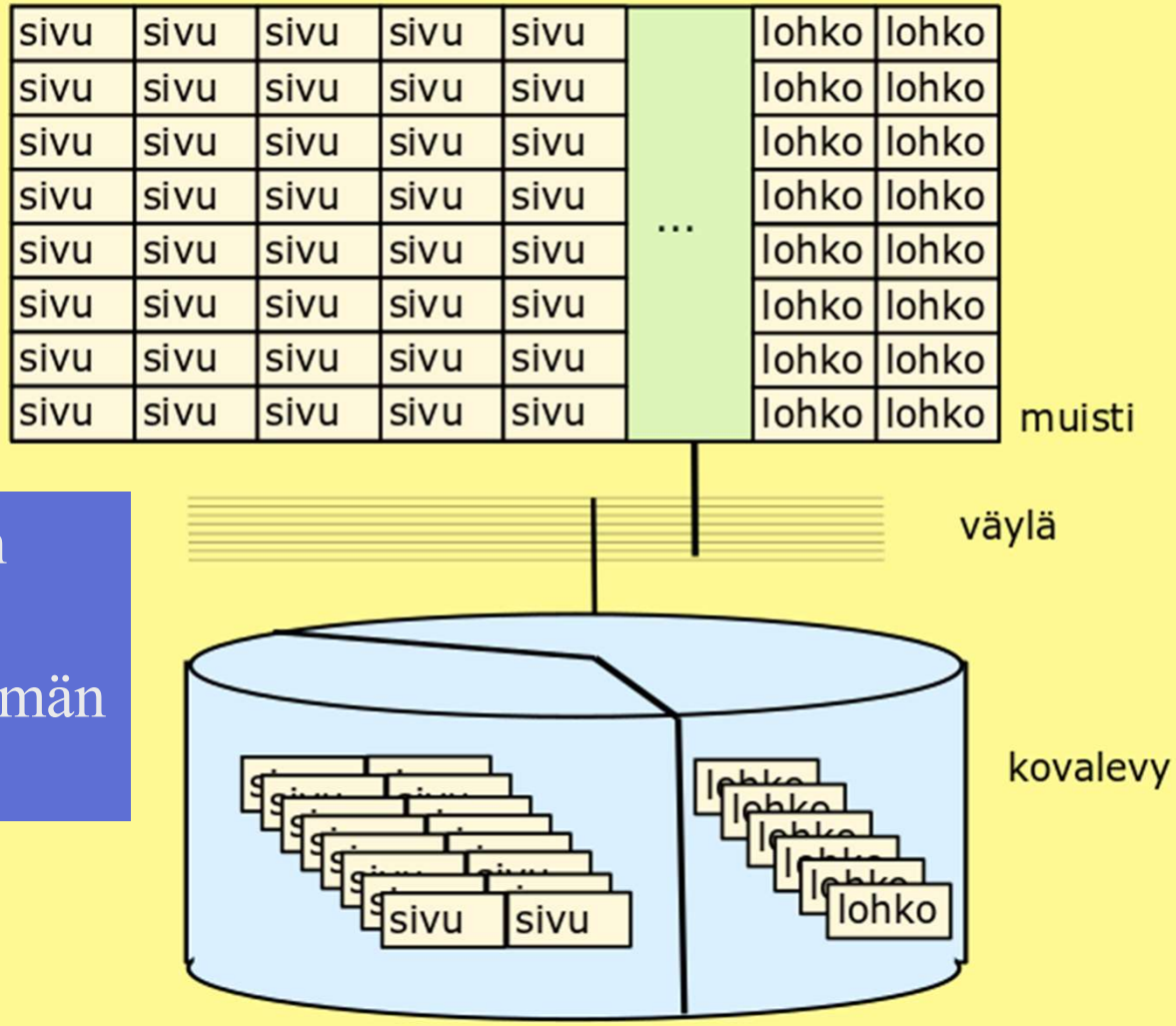
Tiedoston talletus levylle

- Tiedosto koostuu useista lohkoista
  - lohko = 1 tai usea levyn sektori
- Levyn hakemisto
  - tiedoston lohkot
  - luetaan lohkot annetussa järjestyksessä

Pienin alue, jota voi lukea/kirjoittaa levyllä



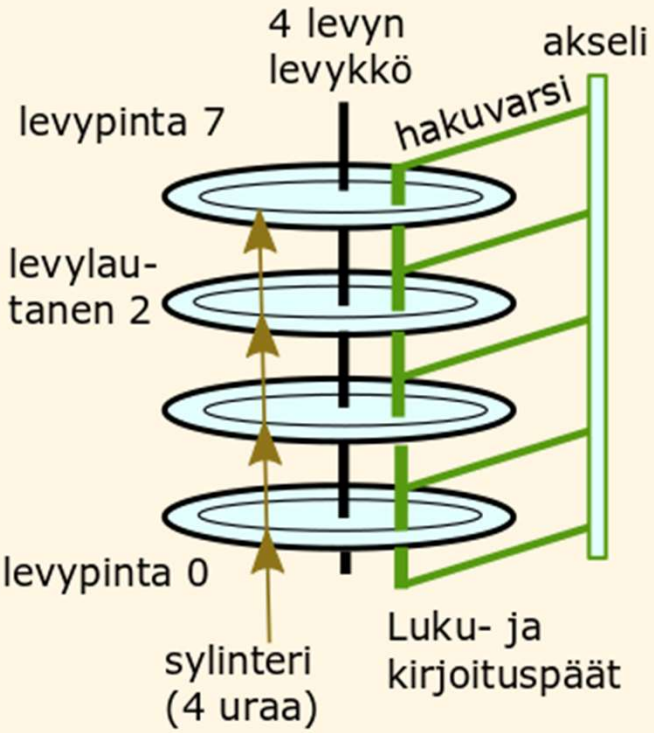
Levyn käyttö

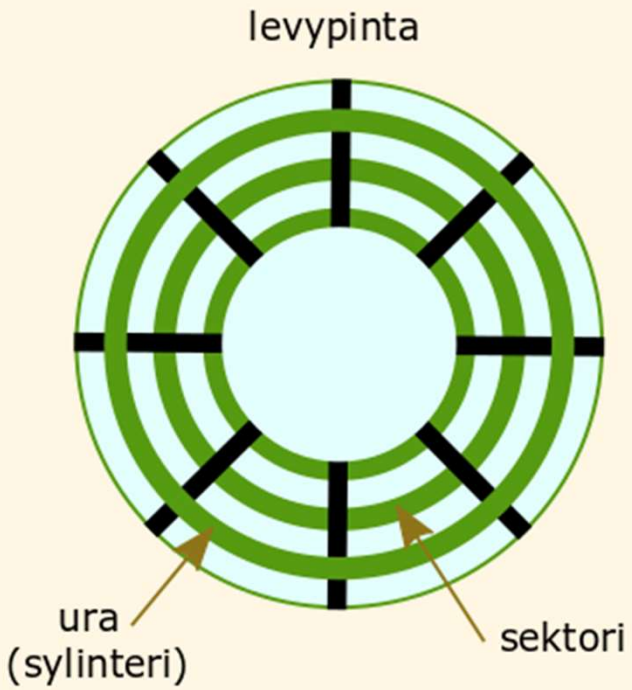


Virtuaalimuistin sivut vs. tiedostojärjestelmän lohkot

Levymuistin saantiaika

- Tiedon (lohkon) osoite: levypinta + ura + sektori
- Laiteajuri etsii KJ-taulukoista loogisen osoitteen perusteella

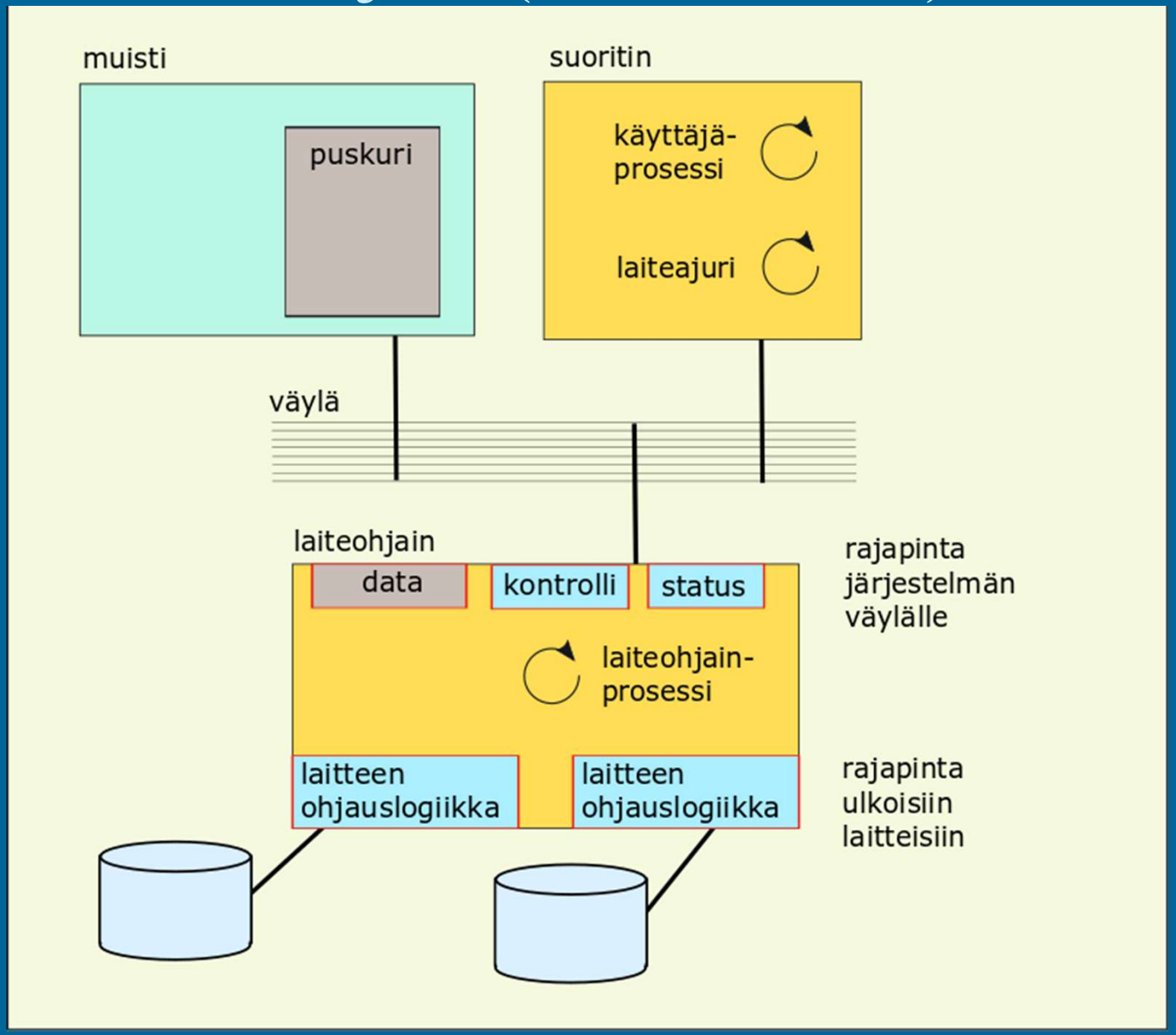




- Saantiaika
  - Hakuaika + pyör.viive + tiedon siirto

Keskustele

Laiteohjain (I/O moduuli)



I/O:n toteutus konekäskyillä (i): Laiteohjaimen rekistereihin (muistiin) viittaaminen I/O konekäskyillä

- I/O-operaation tunnistaa operaatiokoodista
  - I/O laitteille on omat konekäskyt
  - I/O-käskyillä oma osoiteavaruus, eivät viittaa keskusmuistiin
- Käskyssä annetaan laiteohjaimen identifikaatio ja laiterekisterin nro (oma I/O osoiteavaruus)
- Vaikea laajentaa käyttöä uusiin laitteisiin, joilla ”laiterekisterit” voivat olla hyvinkin erilaisia
- Suorittimen konekäskyjä ei voi muuttaa

x86: IN, OUT
INS, OUTS

Ttk-91:
IN, OUT

I/O:n toteutus konekäskyillä (ii): Laiteohjaimen rekistereihin (muistiin) viittaaminen muistin luku/kirjoitus käskyillä

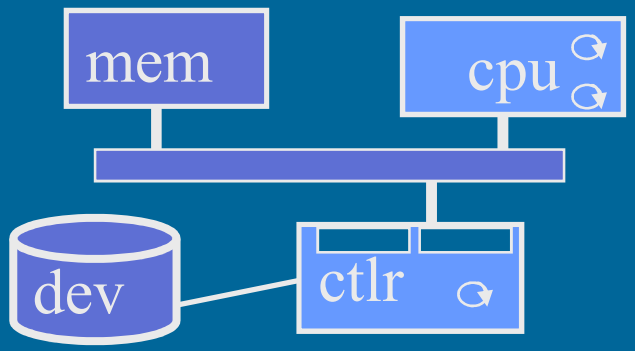
- I/O-operaation tunnistaa viitatusta muistiosoitteesta
- Muistiinkuvattu I/O, memory-mapped I/O
- Laiteajuri lukee/kirjoittaa laiteohjaimella olevia rekistereitä (data, status/kontrolli) tavallisilla muistin luku/kirjoitus käskyillä
  - ei tarvita erillisiä I/O-konekäskyjä!

```
load  R1,=DiskRd
store R1, DiskCtr
```

  - laiteohjaimella olevat ”laiterekisterit” ovat samanlaista viitattavaa muistia kuin ”normaali muisti”
  - muistiosoitteen ensimmäiset bitit (ei siis käskykoodi) valitsevat, mille laitteelle (vai tavallisen muistiin) viittaus kohdistuu
    - Osa muistiavaruudesta on varattu I/O-laitteille!

```
DiskCtr EQU  0x80000001
```

I/O-tyypit



- Suora I/O — Programmed I/O, Direct I/O
  - Laiteajuri koko ajan aktiivinen
  - Laiteajuri odottaa busy-wait loopissa laiteohjainta
- Epäsuora I/O — Indirect I/O, Interrupt driven I/O
  - Laiteohjain osaa tehdä I/O-laitekeskeytyksiä
  - Laiteajuri odottaa odotustilassa laiteohjainta
- DMA I/O — Direct Memory Access I/O
  - Laiteohjain osaa myös viitata keskusmuistiin
    - Tieto kulkee väylän läpi vain yhden kerran
  - Laiteohjaimelle annettavat tehtävät isompia
  - Laiteajuri odottaa odotustilassa laiteohjainta

Keskustele

Esimerkki: kirjoittimen laiteajuri ttk-91 koneelle

- Laitteella voi tulostaa kokonaislukuja yksi kerrallaan
- Muistiinkuvattu I/O, suora I/O
- Laiteportti
  - kontrollirekisteri — muistipaikka 1048576 = 0x80000
  - tilarekisteri — muistipaikka 1048577 = 0x80001
  - datarekisteri — muistipaikka 1048578 = 0x80002
- Laiteajuri Print toimii etuoikeutetussa tilassa
  - Voi viitata portin rekistereihin
- Kutsu:

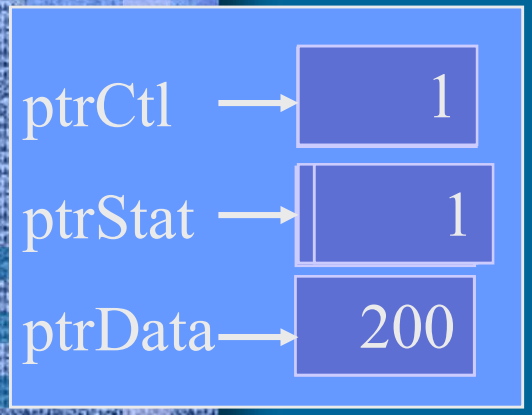
```
PUSH  SP, =0        ; space for return value
PUSH  SP, X         ; parameter to print
SVC   SP, =Print    ; returns Success/Failure
POP   SP, R1
JNZER R1, TakeCareOfTrouble
```

Esim: laiteajurin toteutus (12)

Solution with no timeout

```
ptrCtr    DC  1048576 ; control register address
ptrStat   DC  1048577 ; status register address
ptrData   DC  1048578 ; data register address
retVal    EQU  -3
parData   EQU  -2
```

Oleta: SVC:n ja IRET:n toteutus samalla tavalla kuin CALL ja EXIT




```
Print   PUSHR  SP              ;save regs
        LOAD   R1, parData(FP)
        STORE  R1, @ptrData ; data to print
        LOAD   R1, =0
        STORE  R1, @ptrStat  ; init (clear) state register
        LOAD   R1, =1
        STORE  R1, @ptrCtr  ; give command to print

Wait    LOAD   R1, @PtrStat ; check state register
        JNZER  R1, Done
        JUMP   Wait             ; wait until I/O done

Done    LOAD   R1, =0           ; return "Success"
        STORE  R1, retVal(FP)
        POPR   SP               ; recover regs
        IRET   SP, =1
```

See: driver.k91

http://www.cs.helsinki.fi/group/nodes/kurssit/tito/esimerkit/driver.k91

Keskustele