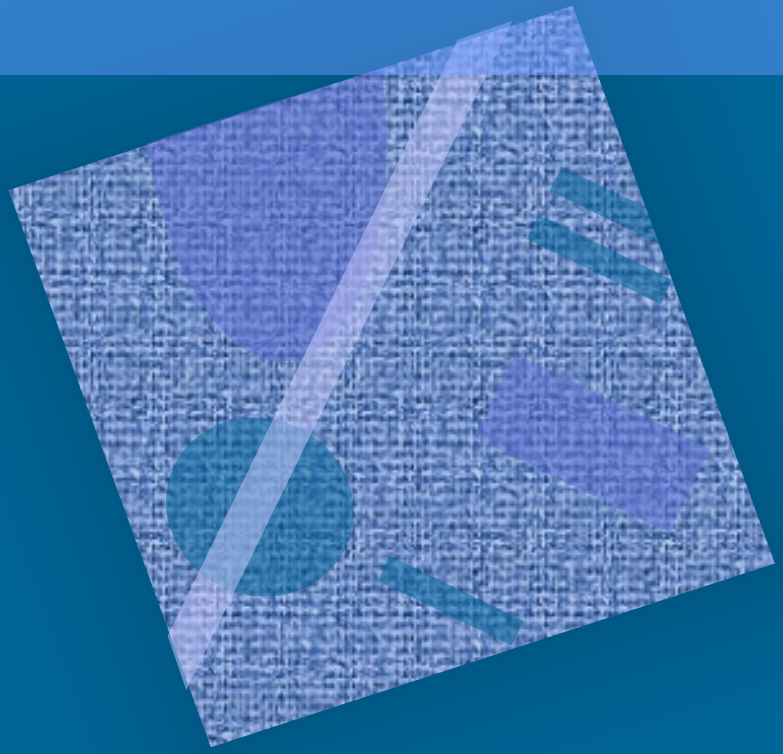
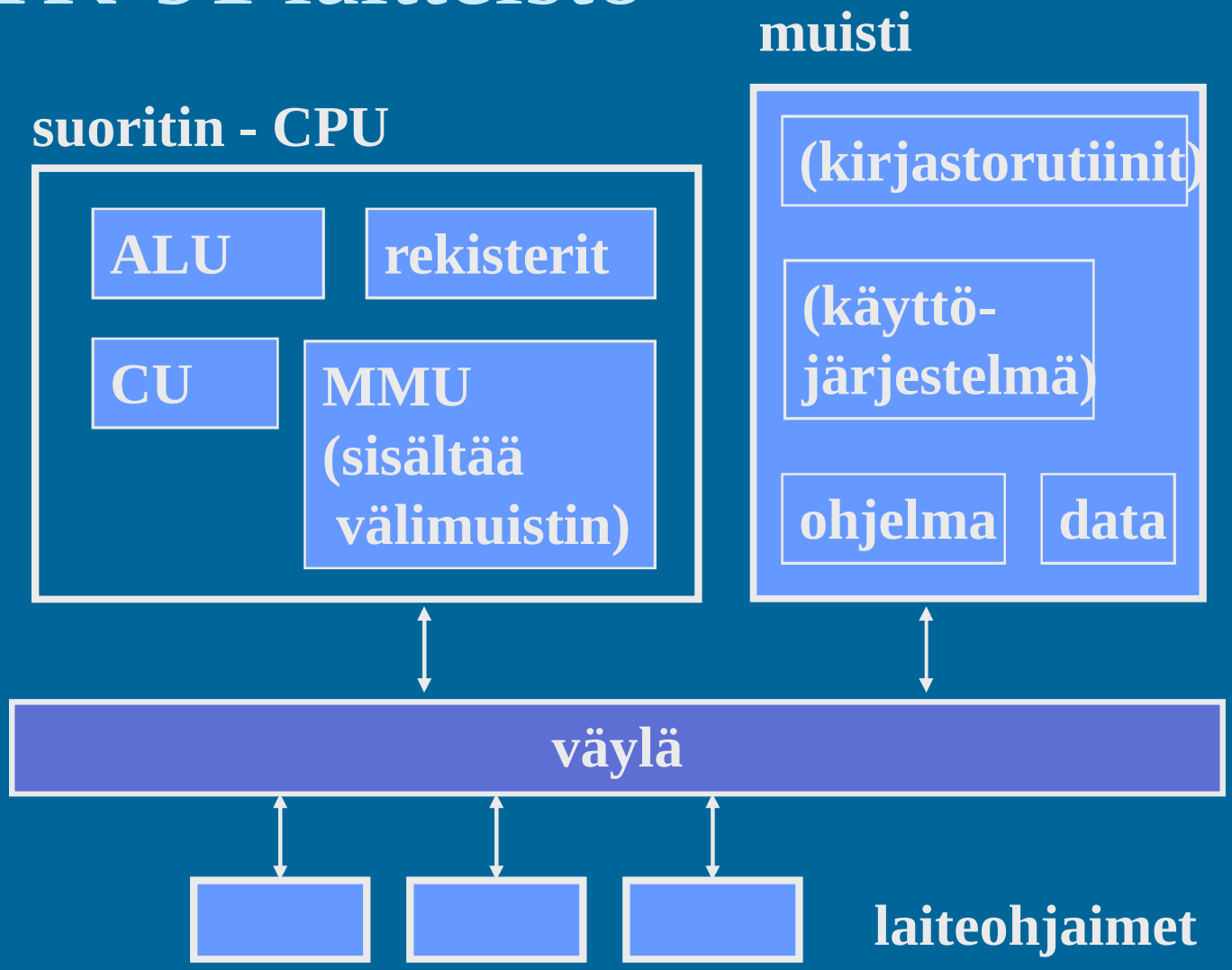
Luento 11

Kertaus ja yhteenveto

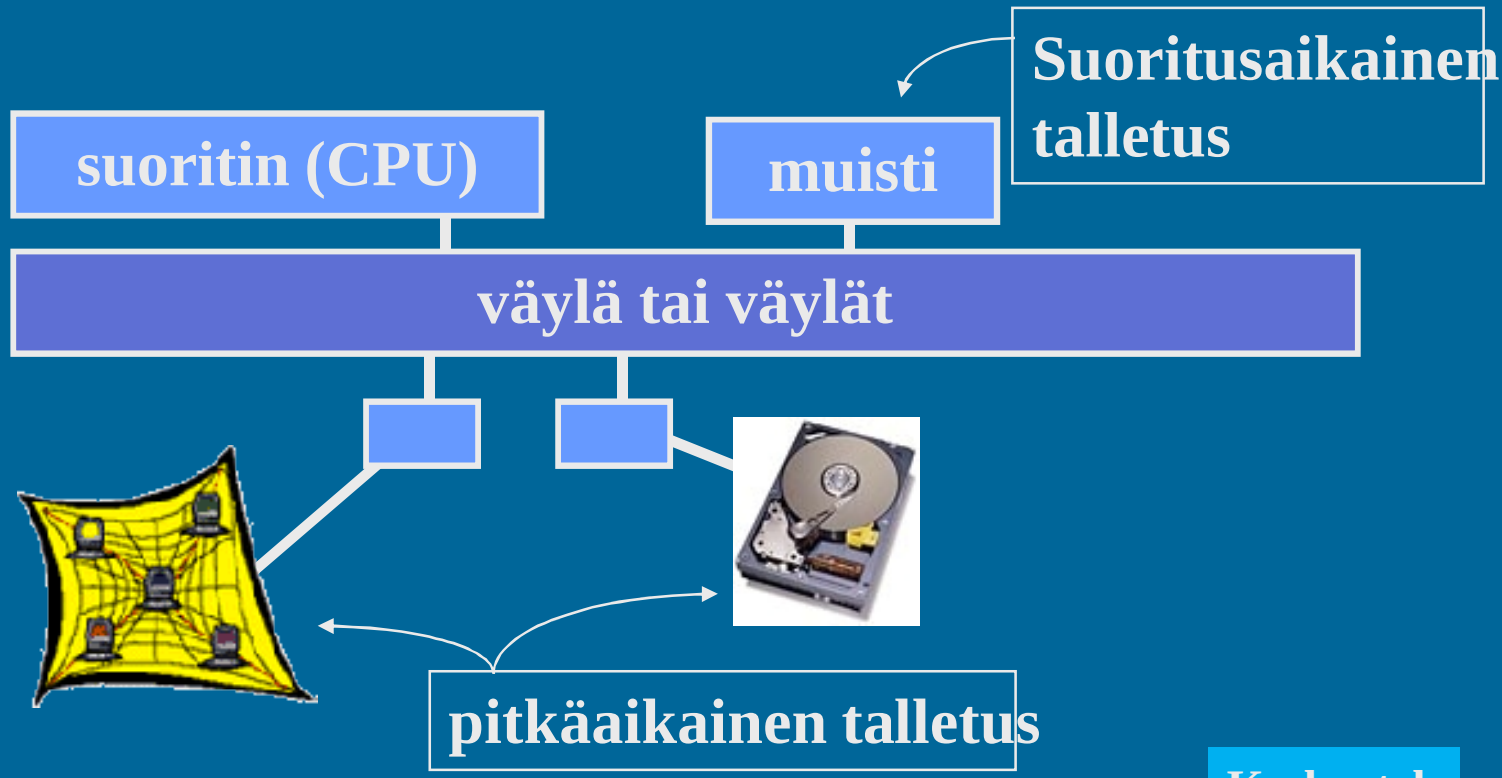


TTK-91 laitteisto



Tietokoneohjelman sijainti

- Suoritusaikana muistissa
- Muuna aikana esim. levyllä, verkossa, tms.

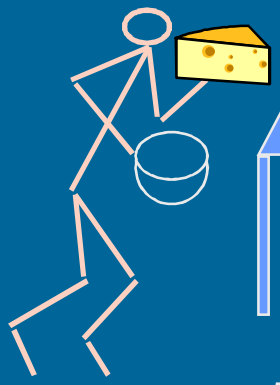


Keskustele

Nopeuserot: Teemun juustokakku

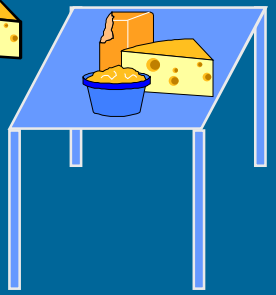
Rekisterien, välimuistin, muistin, levymuistin ja magneettinauhan nopeudet suhteutettuna juuston hakuaikaan juustokakkua tehdessä?

käsi



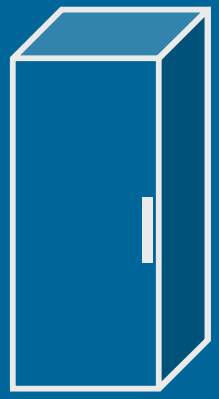
0.5 sek (rekisteri)

pöytä



1 sek (väli-muisti)

jääkaappi

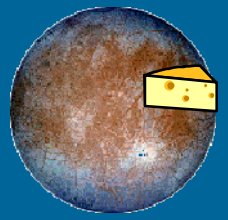


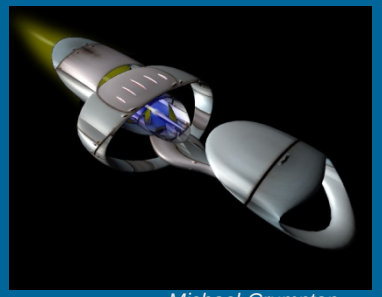
10 sek (muisti)

kuu

12 pv (levy)

Europa (Jupiter)





Michael Crumpton

4 v (verkko,ihminen)

2008:	0.5 ns?	10 ns?	4 ms? (50 pv?)	1 s? (n 65 v?)

Ohjelman esitysmuoto: symbolinen konekieli

- Usein symbolisella konekielellä
  - käsky jaettu osiin (kenttiin)
  - joidenkin kenttien arvot kuvattu symboleilla
  - helpompi ihmisten lukea ja kirjoittaa

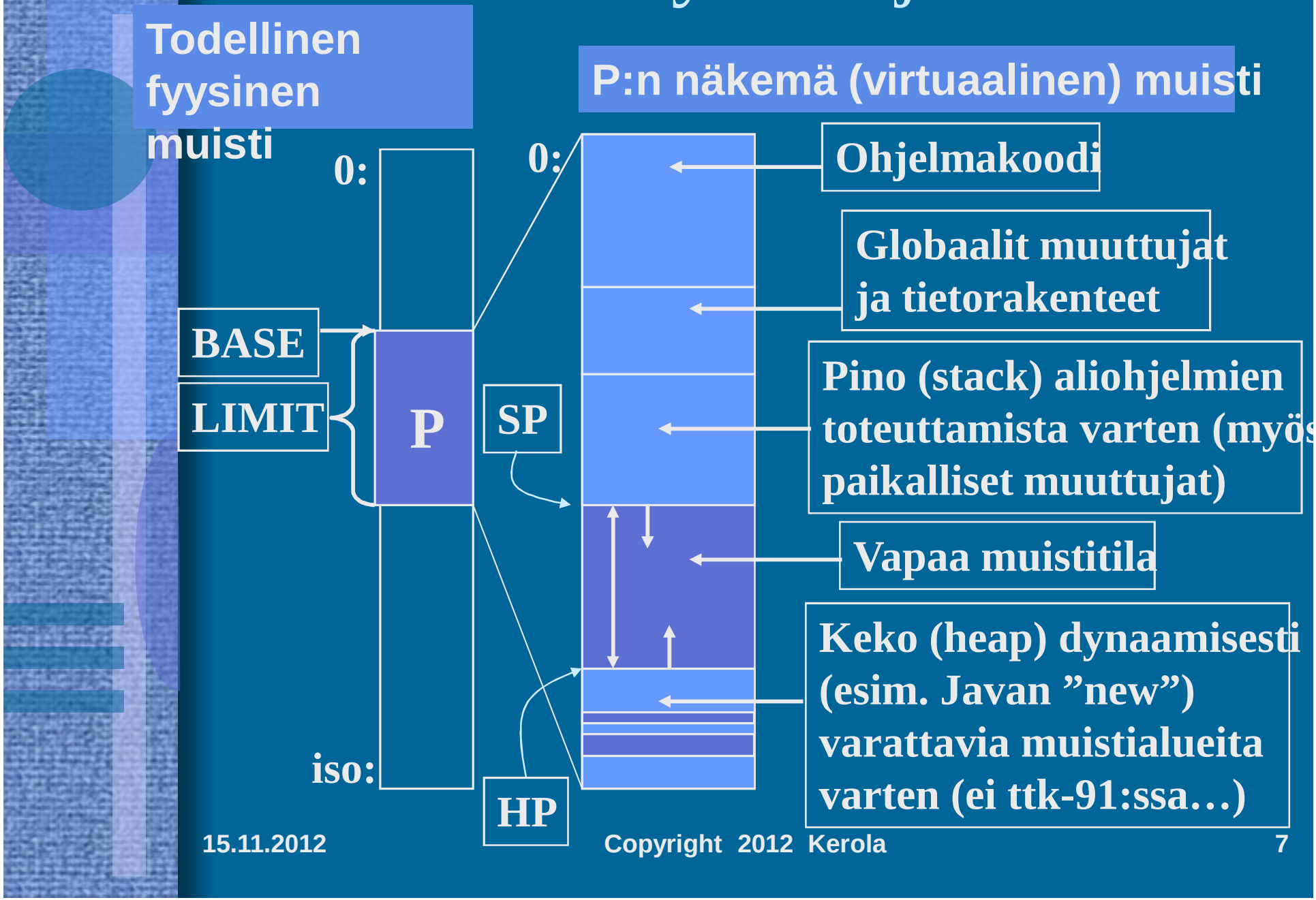
Symb. konekieli	Konekielinen käsky
LOAD R2, =100	0000 0010 000 00 010 0000 0000 0110 0100
LOAD R1, 100	0000 0010 001 01 000 0000 0000 0110 0100
DIV R1, R2	0001 0100 001 00 010 0000 0000 0000 0000
JZER 6	0010 0010 000 00 000 0000 0000 0000 0110
STORE R1, 228	0000 0001 001 00 000 0000 0000 1110 0100
NOP	0000 0000 000 00 000 0000 0000 0000 0000

Miten tietoon viitataan eri paikoissa?

Tiedon sijainti suoritusaikana

- Rekisteri (nopein)
  - kääntäjä päättää milloin muuttujan arvo on rekisterissä
- Välimuisti (nopea)
  - laitteisto hoitaa automaattisesti joillekin muistialueille
- Muisti (hidas)
  - kääntäjä/lataaja valitsee sijaintipaikan
    - globaali data ohjelman latauksen yhteydessä
    - vakiot konekäskyssä
  - ohjelma sijoittaa suoritusaikana
    - aliohjelmien paikalliset muuttujat, parametrit
  - käyttöjärjestelmä sijoittaa suoritusaikana
    - dynaaminen data keossa suorituksen aikana
- Levy, levypalvelin (liian hidas, ei mahdollista)
  - vaatii käyttöjärjestelmän varusohjelmien apua

Muistitilan käyttö ohjelmalle P



Konekielinen ohjelmointi

```
for (int i=
    T[
```

muuttujat, vakiot
taulukot (2D), tietueet

muistissa, rekisterissä?

valinta, loopit
aliohjelmat, SVC:t
parametrit,
paikalliset muuttujat

```
I       DC      0
        ...
        LOAD  R1, =20
        STORE R1, I

        LOAD  R2, =0
        LOAD  R1, I
        STORE R2, T(R1)

        LOAD  R1, I
        ADD   R1, =1
        STORE R1, I

        LOAD  R3, I
        COMP  R3, =50
        JLES  Loop
```

Aliohjelmat, funktiot

- Parametrien ja paluuarvon tyypit
- Parametrien ja paluuarvon välitys
- Paikallisten muuttujien käyttö
- Rekistereiden talletus ja arvon palautus
- Kutsun ja paluun toteutus
- Aktivointitietue, AT-pino

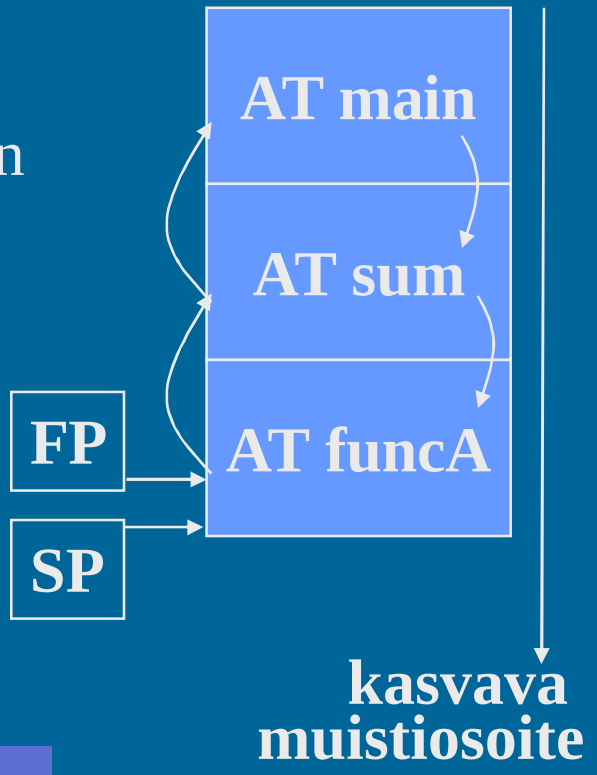
Aktivointitietue (Aktivointitietuepino)

Parametrien tyyppi?
arvo-, viite- ja nimiparametrit

- Aliohjelman toteutusmuoto (ttk-91)
  - funktion paluuarvo (tai kaikki paluuarvot)
  - kaikkien (sisäänmeno- ja ulostulo-) parametrien arvot
  - paluuosoite
  - kutsukohdan aktivointitietue
  - kaikki paikalliset muuttujat ja tietorakenteet
  - aliohjelman ajaksi talletettujen rekistereiden alkuperäiset arvot

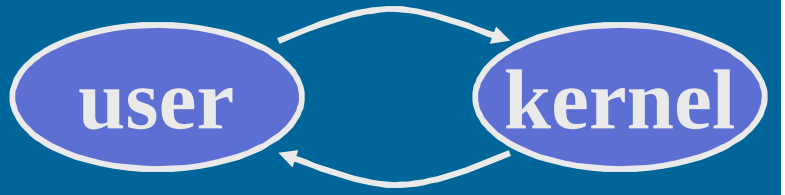
Aktivointitietue pino muistissa

- Aktivointitietueet (AT) varataan ja vapautetaan dynaamisesti (suoritusaikana) pinosta (muistista)
  - SP (=R6) osoittaa pinon pinnalle
- Aktivointitietuepino
  - FP (R7) osoittaa voimassa olevan AT:n sovittuun kohtaan (ttk-91: vanhan FP:n osoite)
- Pinossa olevaa AT:tä rakennetaan ja puretaan käskyillä:
  - PUSH, POP, PUSHR, POPR
  - CALL, EXIT (SVC, IRET)



Käskyjen nouto- ja suoritussykli

Suorittimen suoritustilat



- Käyttäjätila **(user mode, normal mode)**
  - voi käyttää vain tavallisia käskyjä
  - voi viitata vain käyttäjän omaan muistiavaruuteen (MMU valvoo)
- Etuoikeutettu tila tai (KJ:n) ytimen tila **(kernel mode, privileged mode)**
  - voi käyttää kaikkia konekäskyjä, myös etuoikeutettuja (esim. clear_cache, iret)
  - voi viitata kaikkialle muistiin, myös käyttöjärjestelmän ytimeen (kernel)
    - voi käyttää (myös) suoria muistiosoitteita (PA, physical address)

Miten ja milloin tila vaihtuu

Tiedon esitysmuodot

1/8 = 0.1250
1/16= 0.0625
0.1875

“+”	“15”	“0.1875” = “0.0011”
sign	exponent	mantissa or significand

kokonaisluvut
liukuluvut
merkit
merkkijonot
(kuvat)
(äänet)

ei-standardoitu tieto?

suorittimen ymmärtämä tieto?

että …

mantissa	eksponentti
0.0011	“15”
1.1000	“12”
1000	“12”

bitin mantissa!

Keskeytyskäsittelijä

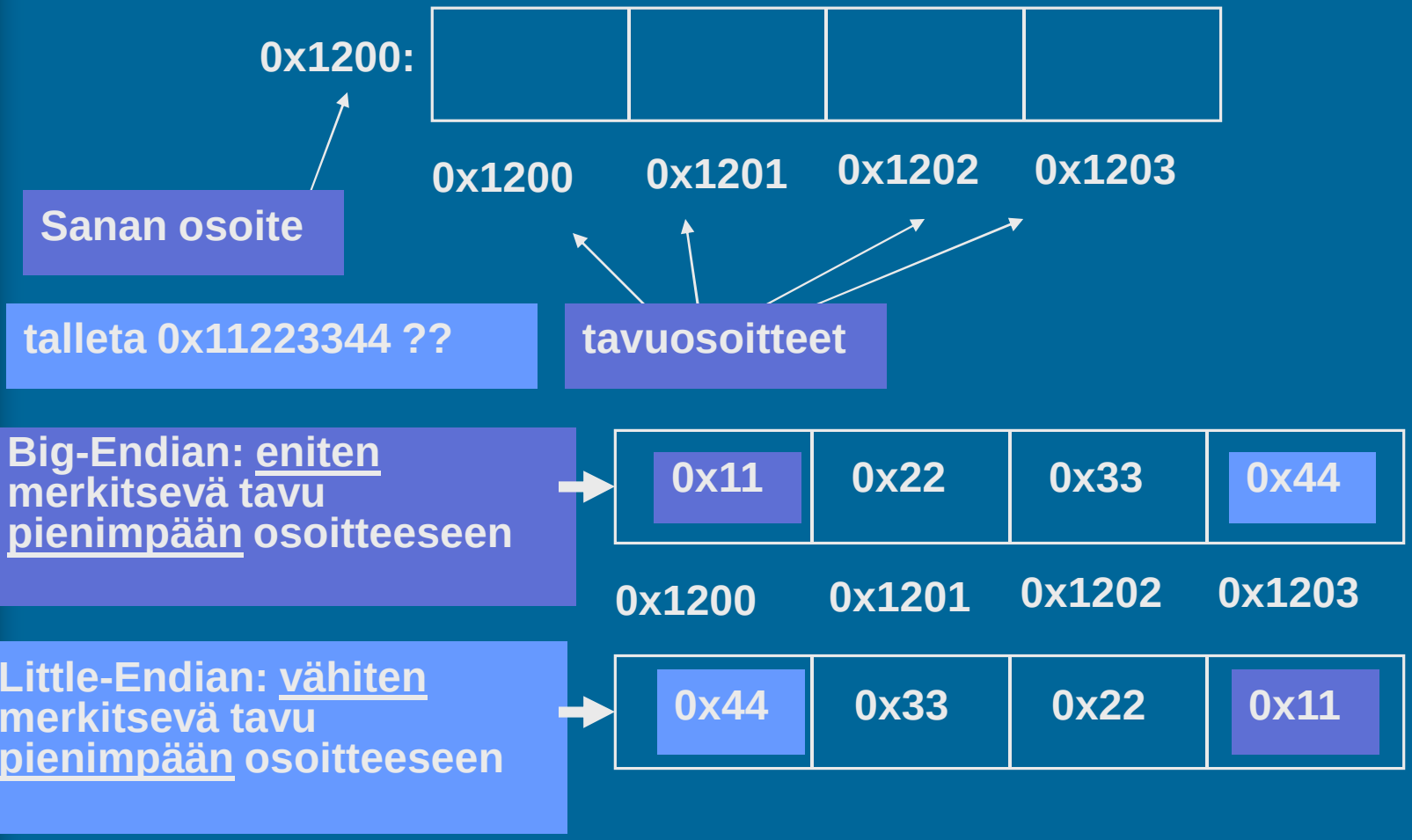
- Osa käyttöjärjestelmää
- Ennen keskeytyskäsittelijään hyppäämistä asetetaan suoritin ja MMU etuoikeutettuun käyttöjärjestelmätilaan **(supervisor state)**
  - SR:n bitti P on päällä => etuoikeutettu tila eli käyttöjärjestelmä tila
    (P = Priviledged)
  - käyttöjärjestelmätilassa saa viitata mihin tahansa kohtaan muistia (MMU: BASE=0, LIMIT=”hyvin iso”)
  - käyttöjärjestelmätilassa saa käyttää kaikkia konekäskyjä (esim. IRET tai ClearCache)
- Käsittelijästä paluun yhteydessä MMU:n tila ja suorittimen tila (bitti P) asetetaan ennalleen

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

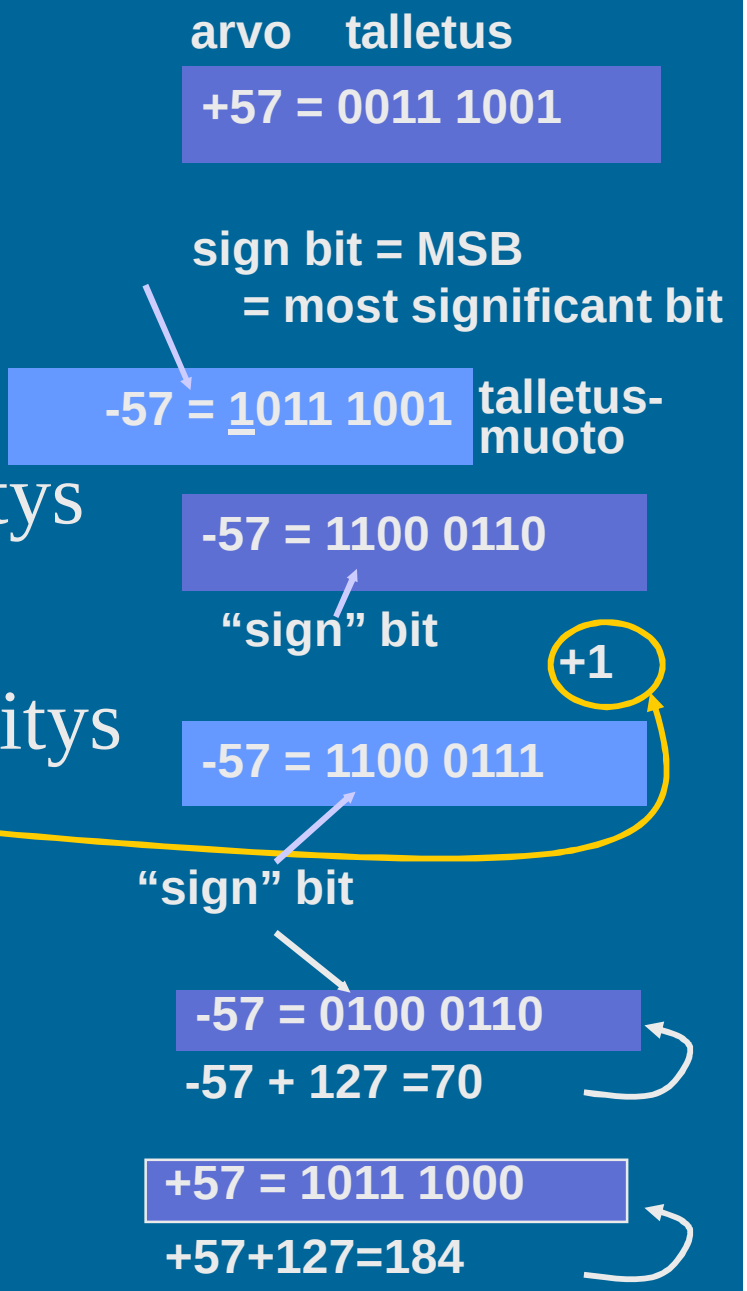
Big vs. Little Endian

- Miten monitavuiset arvot talletetaan?



Negatiiviset kokonaisluvut

- Etumerkkibitti erikseen
- Yhden komplementtiesitys
- Kahden komplementtiesitys
- Vakiolisäys
  - Esim. lisää 127 (=2^7 –1)
    - yleensä: $2^{\text{bittilkm-1}}$ -1
  - Talleta etumerkittömänä



IEEE 32-bit FP Standard

1/8 = 0.1250
1/16= 0.0625
0.1875

"+"	"15"	"0.1875" = "0.0011"
sign	exponent	mantissa or significand

- 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)

24 bitin mantissa!

mantissa	eksponentti
0.0011	"15"
1.1000	"12"
1000	"12"

Miksi käytetään piilobittiä?

Tiedon muuttumattomuus

- Virheitä tapahtuu
- Otetaan mukaan ylimääräisiä bittejä, joiden avulla virheitä voidaan havaita ja ehkä myös korjata
- Järjestelmä suorittaa tarkistukset automaattisesti joko laitteistotasolla tai ohjelmiston avulla

Virheen korjaava Hamming koodi

	oikein		virheellinen	
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

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

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 (esim. Cray C-90)
  - toteutettu samanlaisilla logiikkaporteilla (gate) kuin prosessorikin

Muisti-hier-arkia

- Flash?
- SSD?
- Levypal-velin?
- Pilvi?

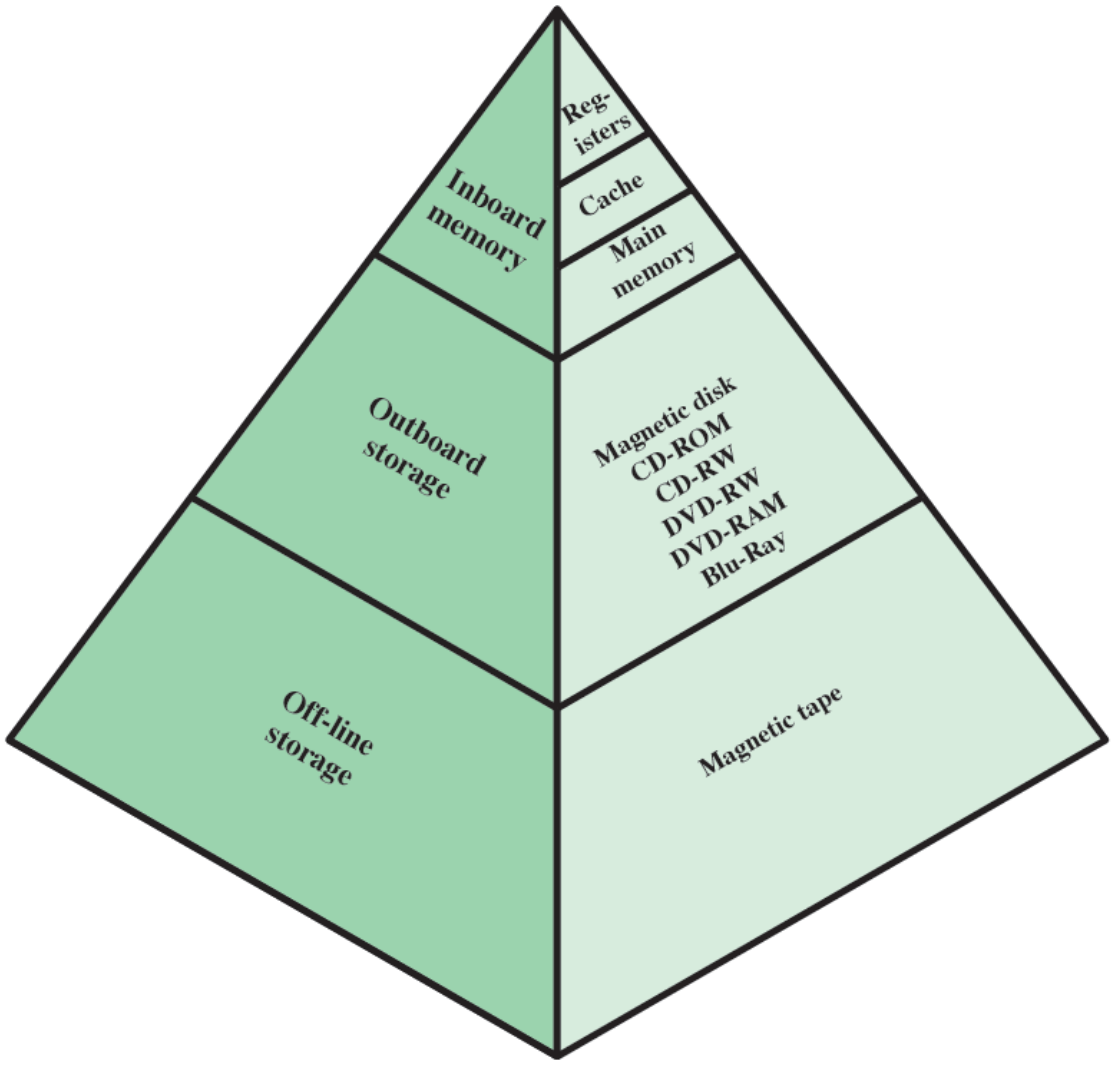
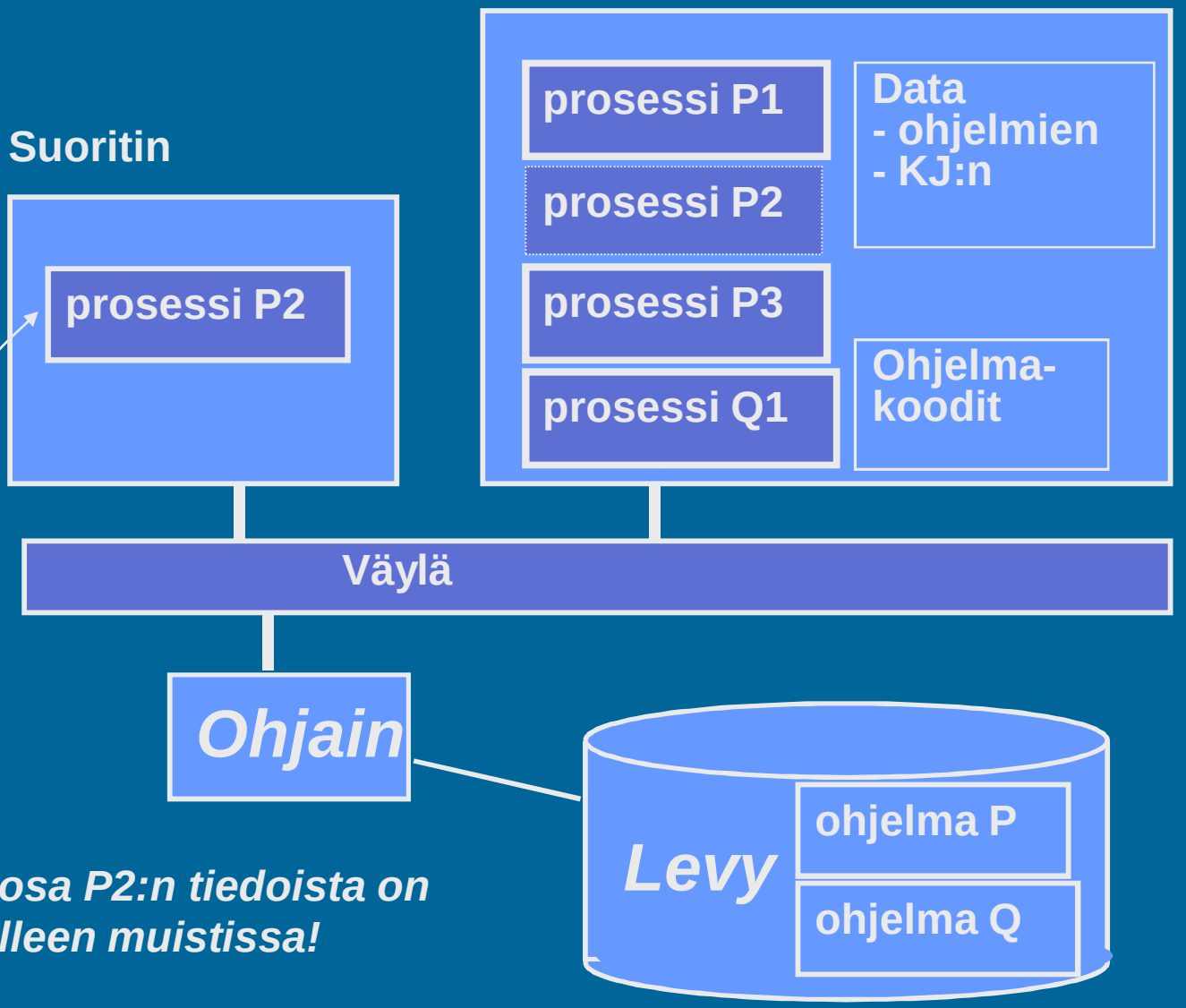
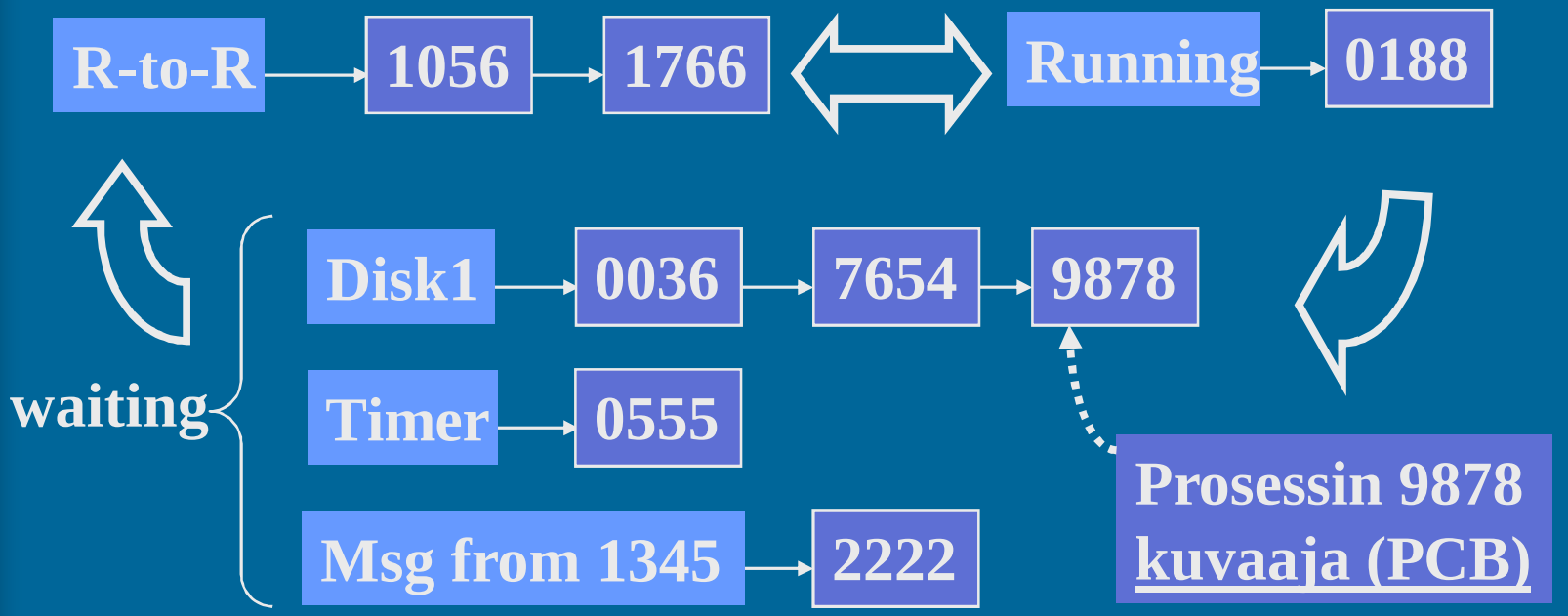


Figure 4.1 The Memory Hierarchy

Prosessi



Prosessit jonoissa ja PCB



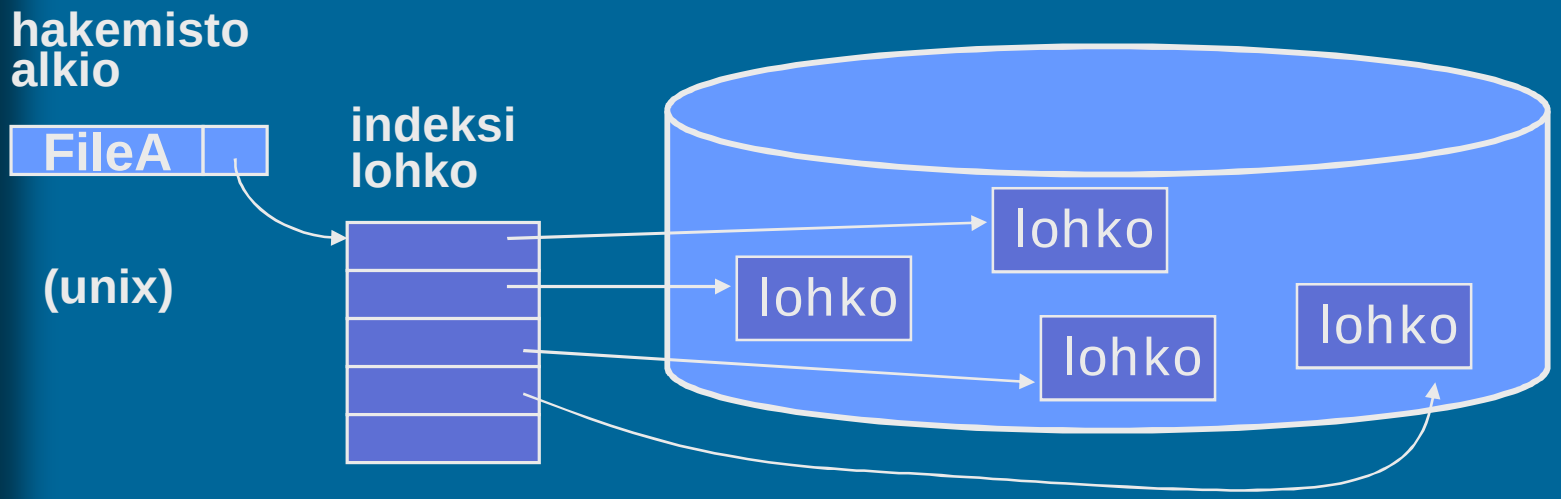
Vuoronanto:
valitse seuraava prosessi Ready-to-Run -jonosta ja
siirrä se suoritukseen CPU:lle
(kopioi tämän prosessin suorittimen tila suorittimelle)

Käyttöjärjestelmän rakenne

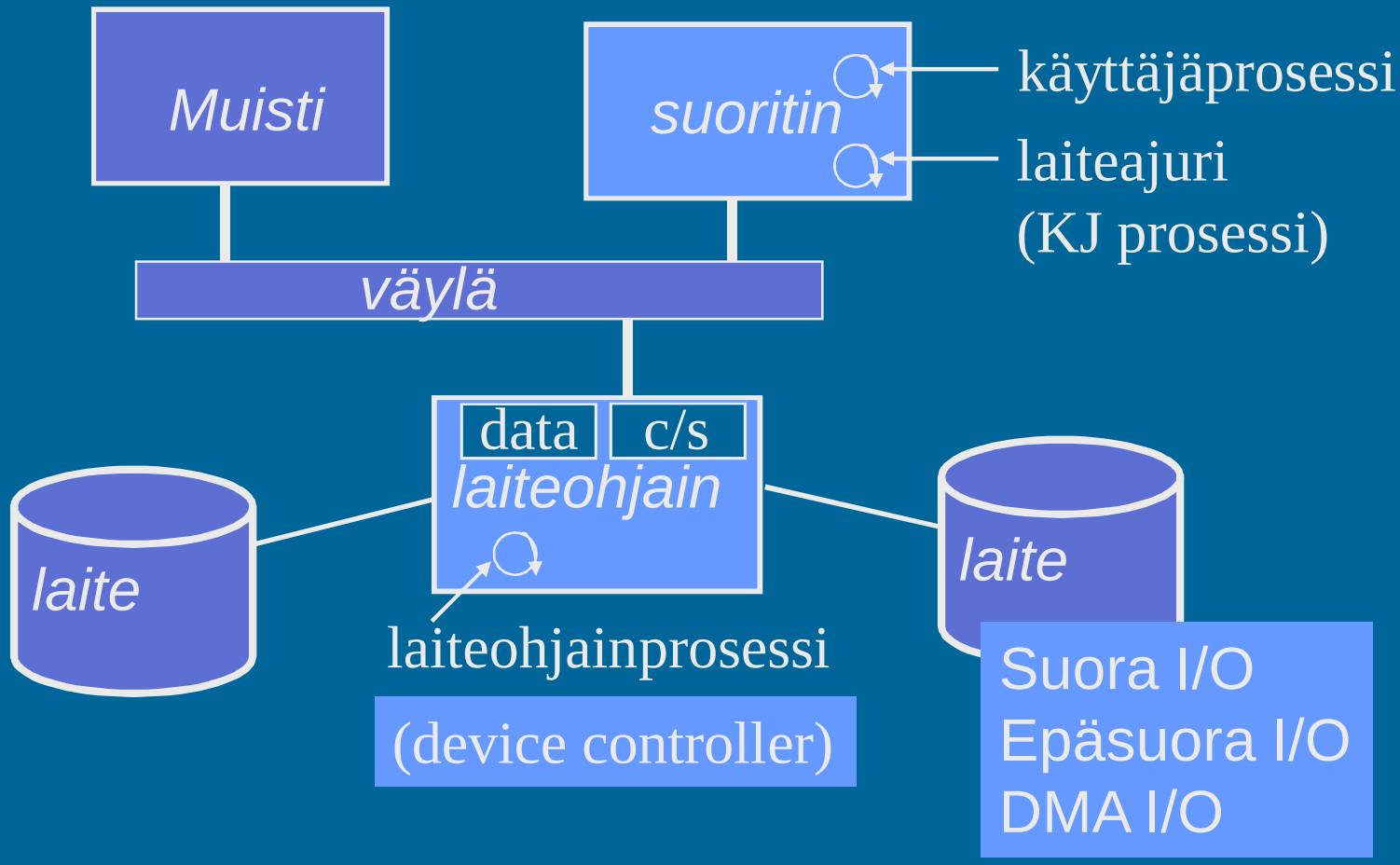
- Prosessien hallinta
- Muistin hallinta
- Tiedostojen ja laitteiden hallinta
- Verkon hallinta

Tiedoston talletus levylle

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

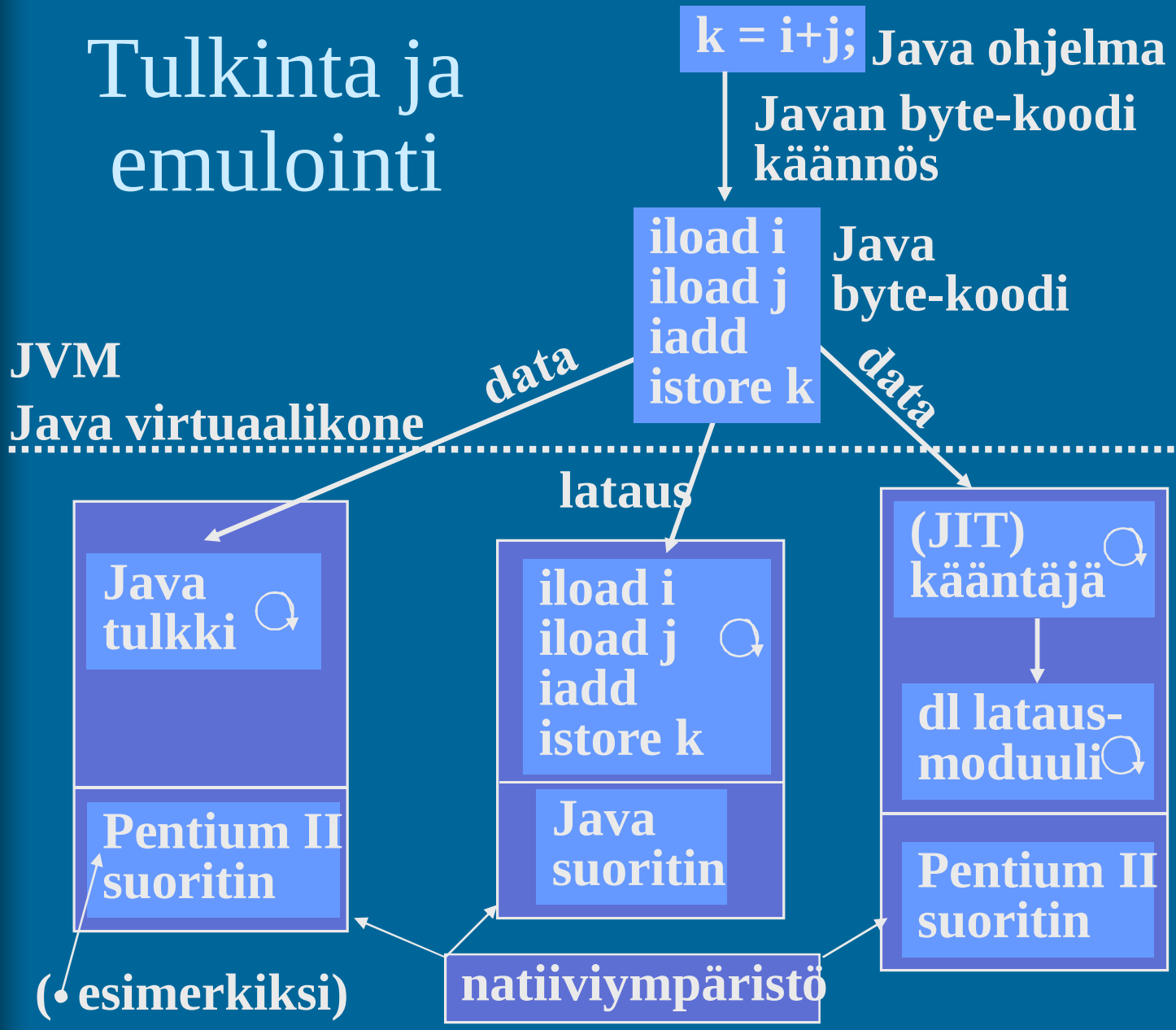


I/O:n toteutus, laiteohjain ja laiteajuri

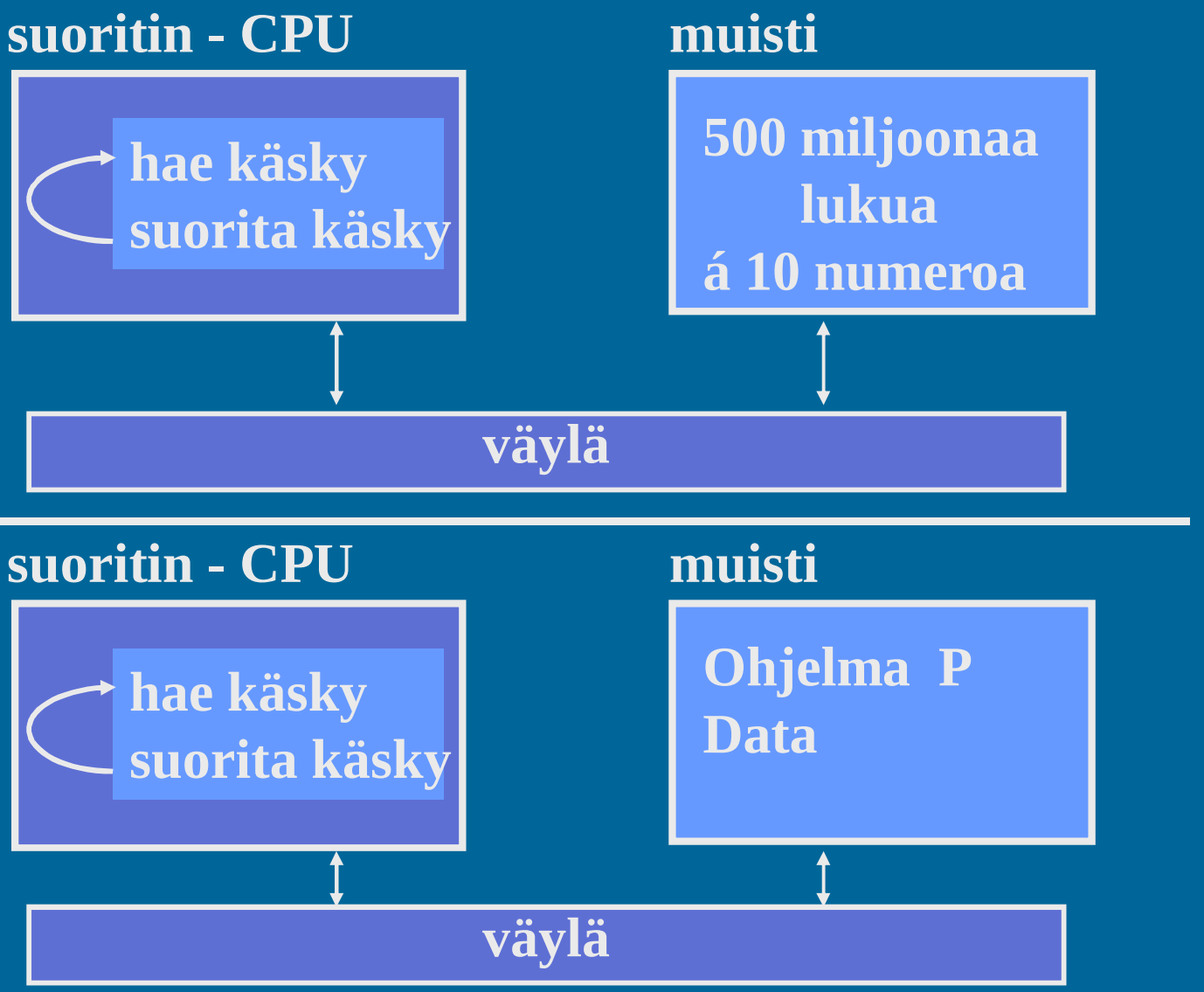


Lausekielestä suoritukseen

Tulkinta ja emulointi



Laskennan teorian perusta (2)



Laskennan teoriaa … (5)

Muistin sisältö ennen P:n suoritusta:

X = hyvin iso kokonaisluku (500M numeroa?)

Muistin sisältö P:n suorituksen jälkeen:

Y = joku toinen hyvin iso luku

P on kokonaislukuarvoinen funktio P: N → N

Ohjelman P es.muoto muistissa: Iso kokonaisluku P∈N

Laskennan teoriaa ...

- Mielivaltaisten ohjelmien ominaisuuksia voi päätellä kokonaislukujen ja niiden välisten funktioiden ominaisuuksista

- **Todistettuja lauseita ohjelmien ominaisuuksista**
  - **pätevät kaikille tietokoneille**
  - **pätevät aina: nyt ja tulevaisuudessa**

Laskennan teoriasta ja algoritmianalyysistä todistettuja lauseita (4)

- Valitaanpa mikä tahansa aikaraja tai muistin koko, niin aina on olemassa sellainen ongelma, että
  - (1) siihen on olemassa ratkaisu ja
  - (2) kaikki ongelman ratkaisevat ohjelmat vievät enemmän aikaa tai muistitilaa kuin ennalta annettu raja
- On olemassa sellaisia ongelmia, että niitä ei voi ratkaista millään tietokoneella
- On olemassa suuri joukko tunnettuja vaikeita ongelmia, joista ei vielä tiedetä, kuinka vaikeita ne oikeastaan ovat

$$P \stackrel{?}{=} NP$$

--

Luennon 11 ja koko kurssin loppu

--

http://lue.kurssikokeeseen.edu/ajoissa.html

http://www.retroweb.com/apollo_retrospective.html