Tietokonejärjestelmän rakenne

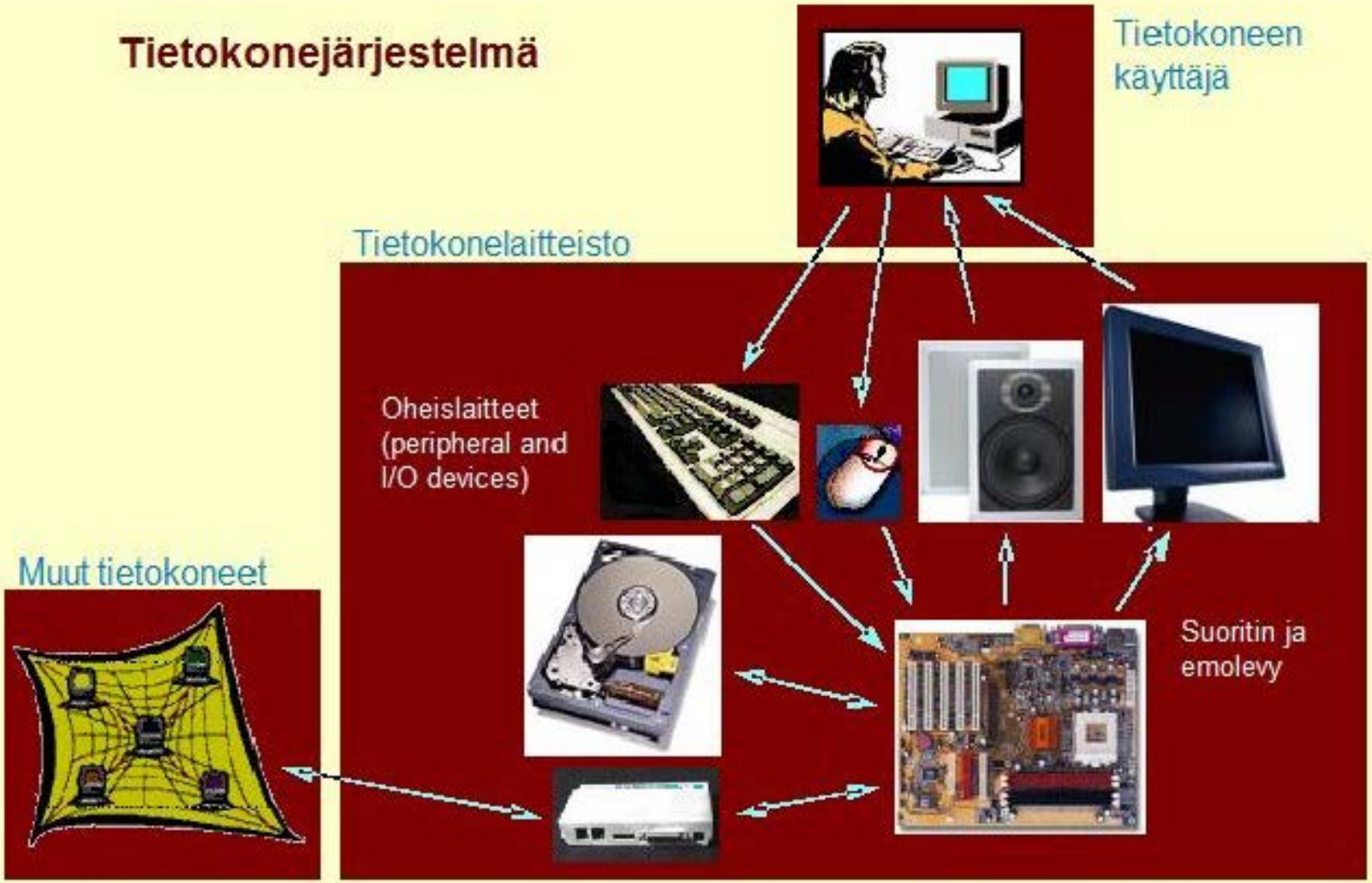
Johdanto

Ohjelman sijainti ja esitysmuoto

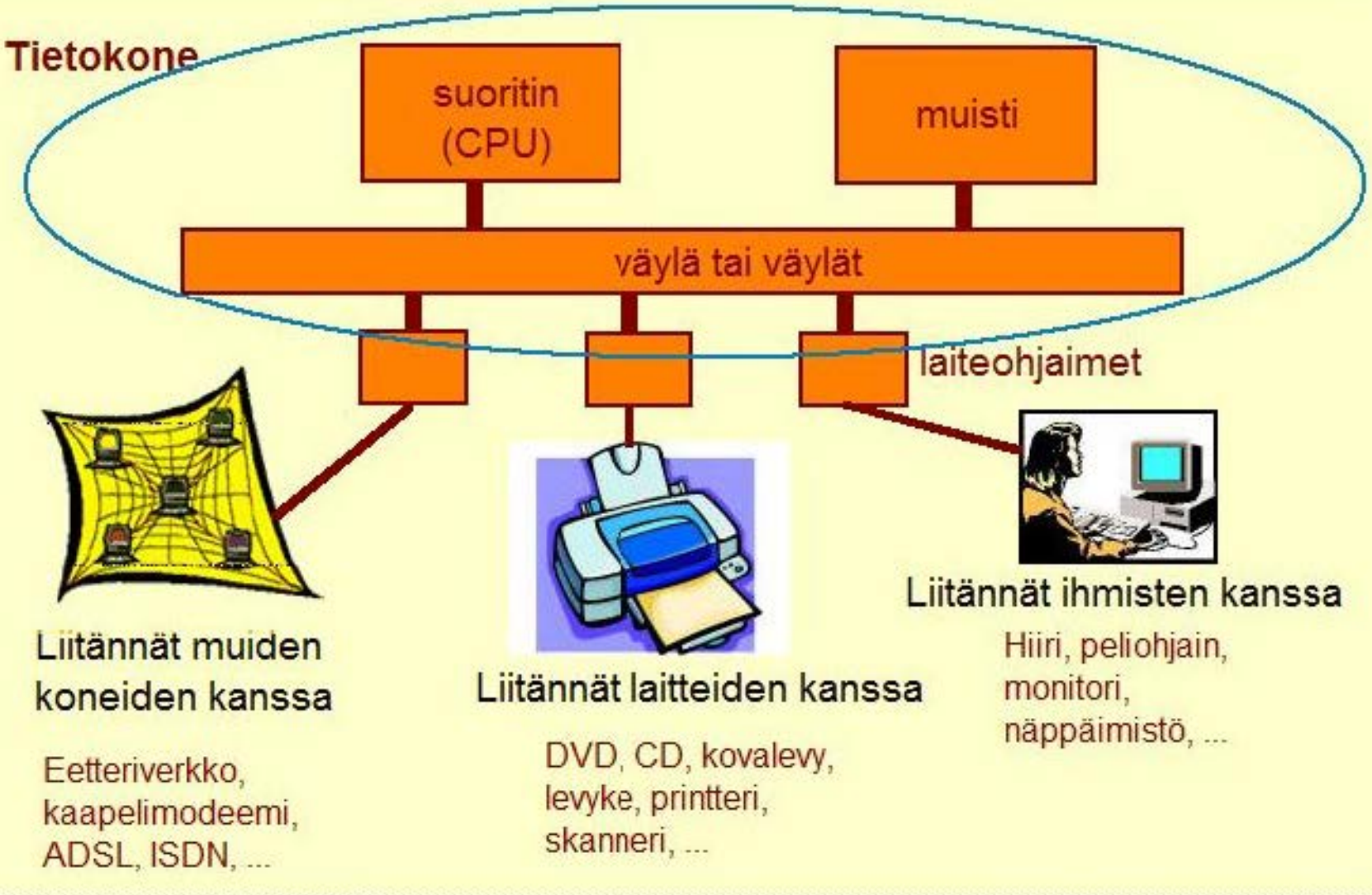
Laitteiston nopeus

Tässä luennossa annetaan yleiskuva tietokonejärjestelmästä. Johdannossa kuvataan tietokonejärjestelmän erilaisia laitteita ja miten ne on liitetty suorittimeen. Seuraavaksi esittelemme lyhyesti, kuinka ja miksi tietokoneohjelman sijainti ja esitysmuoto vaihtelevat ohjelman käsittelyn eri vaiheissa. Lopuksi käsittelemme tietokonelaitteiston eri komponenttien valtavia nopeuseroja ja niistä aiheutuvia ongelmia.

Tietokonejärjestelmä

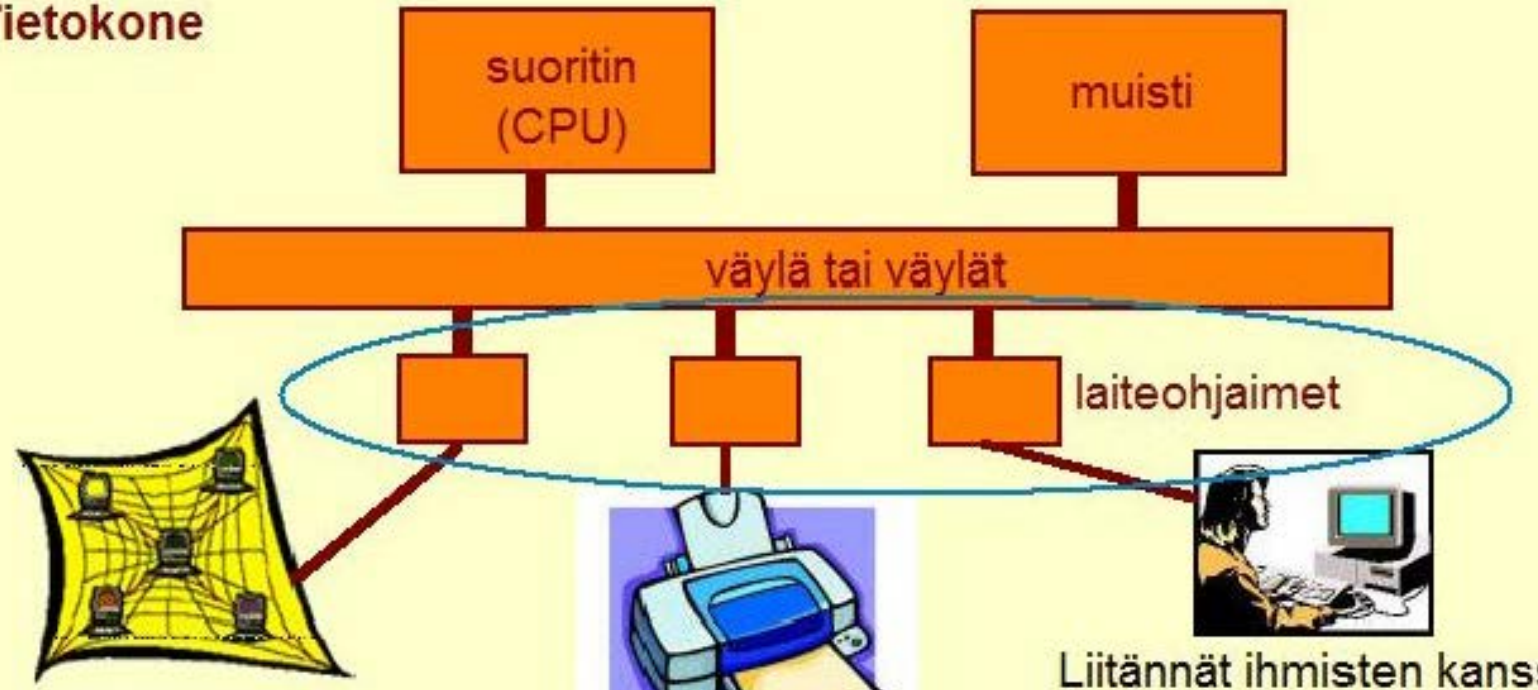


Tietokonelaitteisto koostuu suorittimesta ja siihen liitetyistä erilaisista oheislaitteista. Useimmat oheislaitteet ovat ihmiselle sopivia tiedon syöttö- tai tulostuslaitteita, kuten esimerkiksi näppäimistö ja monitori. Osa laitteista on ainoastaan järjestelmän sisäistä käyttöä varten, ja ne tukevat yleensä tiedon pitkäaikaista talletusta. On myös laitteita, joiden avulla tämä tietokonejärjestelmä voidaan liittää muihin tietokoneisiin ja/tai verkkoon eli Internettiin.



Tietokonelaitteiston tärkeimmät osat ovat suoritin ja muisti, jotka on liitetty toisiinsa väylällä. Ohjelma ja data ovat suoritusaikana muistissa, josta ne haetaan väylän avulla suorittimelle käyttöä varten. Väylä on johdinkimppu, joka varhaisissa koneissa oli toteutettu paksuhkojen sähköjohtojen avulla. Nykyään väylä voi olla toteutettuna silikonilastulla olevilla sähköä johtavilla kerroksilla, emolevyllä painettuina piireinä tai jopa I/O-laitteiden kytkentään tarkoitetuilla paksuhkoilla kaapeleilla.

Tietokone



Liitännät muiden koneiden kanssa

Eetteriverkko, kaapelimodeemi, ADSL, ISDN, ...

Liitännät laitteiden kanssa

DVD, CD, kovalevy, levyke, printteri, skanneri, ...

Liitännät ihmisten kanssa

Hiiri, peliohjain, monitori, näppäimistö, ...

Kaikki laitteet liitetään järjestelmään periaatteessa samalla tavalla. Jokaiselle laitetyypille on väylää varten oma sovittimensa eli laiteohjain. Varsinainen laite liitetään sitten tähän laiteohjaimeen. Kuhunkin laiteohjaimeen liittyy sen käytön toteuttava ohjelma eli laiteajuri, joka on osa käyttöjärjestelmää. Laiteajurin toiminta käsitellään myöhemmin tarkemmin.

Tietokoneohjelman sijainti ja esitysmuoto käyttäjän näkökulmasta

Jossain tietokonelaitteistossa

Jossain muodossa

Helppo suorittaa

- napauta ikonia hiirellä
- anna ohjelman nimi ja parametrit tekstuaaliselle käyttöliittymälle

```
C:\> DIR
melkinpaasi ~: ls
```

- sijoita levy CD-asemaan

```
autoexec.bat
```


Tietokoneohjelman sijainti ja esitysmuoto tietokonelaitteistossa ei ole vakio. Käyttäjän kannalta tällä ei ole merkitystä. Hänelle on ihan samantekevää, missä päin laitteistoa tai missä muodossa ohjelma on talletettuna, kunhan vain sitä on helppo käyttää eli että ohjelma on helppo käynnistää. Ikkunointiin perustuvissa järjestelmissä ohjelman kuvaketta (ikonia) voi napauttaa hiirellä. Tekstuaalisissa käyttöliittymissä ohjelman nimi annetaan näppäimistöllä. Lisäksi yleensä on käytettävissä erilaisia liki automaattisia ohjelman käynnistystapoja. Windows käyttöjärjestelmässä esimerkiksi autoexec.bat niminen tiedosto voi käynnistyä automaattisesti kun levy asetetaan CD tai DVD-asemaan.

Tietokoneohjelman sijainti ja esitysmuoto pitkäaikaisessa talletuksessa

Jollain laitteella, jossa tieto säilyy ilman sähkövirtaa

kovalevy, levyke, magneettinauha, CD, DVD, Flash-muisti, ...

Jollain kielellä kuvattuna

ohjelmointikielet: Java, Fortran, C, Pascal, LISP, Prolog, ...

```
(T (cons (car expression)
(substitute pattern replacement (cdr
expression)) )) ) )
```

tietokannan kuvauskielet: SQL, SQL*Forms, ...

suorittimen konekielet: x86, MIPS, PA-RISC, ...

Pakattuna ehkä jollain tavoin

zip, tar, gz, BinHex4, Stuffit, Zoo, uuencode...

Tietokonejärjestelmän kannalta tilanne on monimutkaisempi. Pitkäaikaista talletusta varten ohjelma täytyy taltioida sellaiselle medialle, jossa tieto säilyy ilman sähkövirtaa (non-volatile memory). Tyypillisiä talletusmedioita ovat tällöin median magneettisuuteen perustuvat ratkaisut, kuten esimerkiksi levykkeet, kovalevyt ja magneettinauhat. Median optisia ominaisuuksia hyödyntäviä ratkaisuja taas ovat CD- ja DVD-levyt. Flash-muisti taas erikoistapaus elektronisesti toteutetuista pysyväismuisteista ja se on jo oikeastaan korvannut levykkeet helposti paikasta toiseen siirrettävänä muistina. Flash-muisteja käytetään myös yleisesti kannettavien laitteiden pysyväismuistina.

Tietokoneohjelman sijainti ja esitysmuoto pitkäaikaisessa talletuksessa

Jollain laitteella, jossa tieto säilyy ilman sähkövirtaa

kovalevy, levyke, magneettinauha, CD, DVD, Flash-muisti, ...

Jollain kielellä kuvattuna

ohjelmointikielet: Java, Fortran, C, Pascal, LISP, Prolog, ...

tietokannan kuvauskielet: SQL, SQL*Forms, ...

suorittimen konekielet: x86, MIPS, PA-RISC, ...

```
(T (cons (car expression)
(substitute pattern replacement (cdr
expression)) )) ) )
```

Pakattuna ehkä jollain tavoin

zip, tar, gz, BinHex4, Stuffit, Zoo, uuencode...

Pitkäaikaisesti talletettu ohjelma voi olla oikeastaan missä tahansa esitysmuodossa. Se voi olla esimerkiksi esitettynä jollakin korkean tason ohjelmointikielellä, kuten Javalla tai C:llä. Etuna ja tekijänoikeuksien haltijan näkökulmasta haittana tässä on ohjelman helppo muutettavuus lähdekoodin tasolla. Yhtä hyvin ohjelma voi olla kuvattuna jollakin tietokannan kuvauskielellä, esimerkiksi SQL:llä. Tai se voi olla kuvattuna jonkin suorittimen ymmärtämällä konekielellä. Ohjelma on tällöin helppo suorittaa, mutta sitä on vaikea muokata, mikä taas on oikein hyvä asia ohjelman tekijänoikeuksien omistajan kannalta.

Tietokoneohjelman sijainti ja esitysmuoto pitkäaikaisessa talletuksessa

Jollain laitteella, jossa tieto säilyy ilman sähkövirtaa

kovalevy, levyke, magneettinauha, CD, DVD, Flash-muisti, ...

Jollain kielellä kuvattuna

ohjelmointikielet: Java, Fortran, C, Pascal, LISP, Prolog, ...

tietokannan kuvauskielet: SQL, SQL*Forms, ...

suorittimen konekielet: x86, MIPS, PA-RISC, ...

```
(T (cons (car expression)
(substitute pattern replacement (cdr
expression)) )) ) )
```

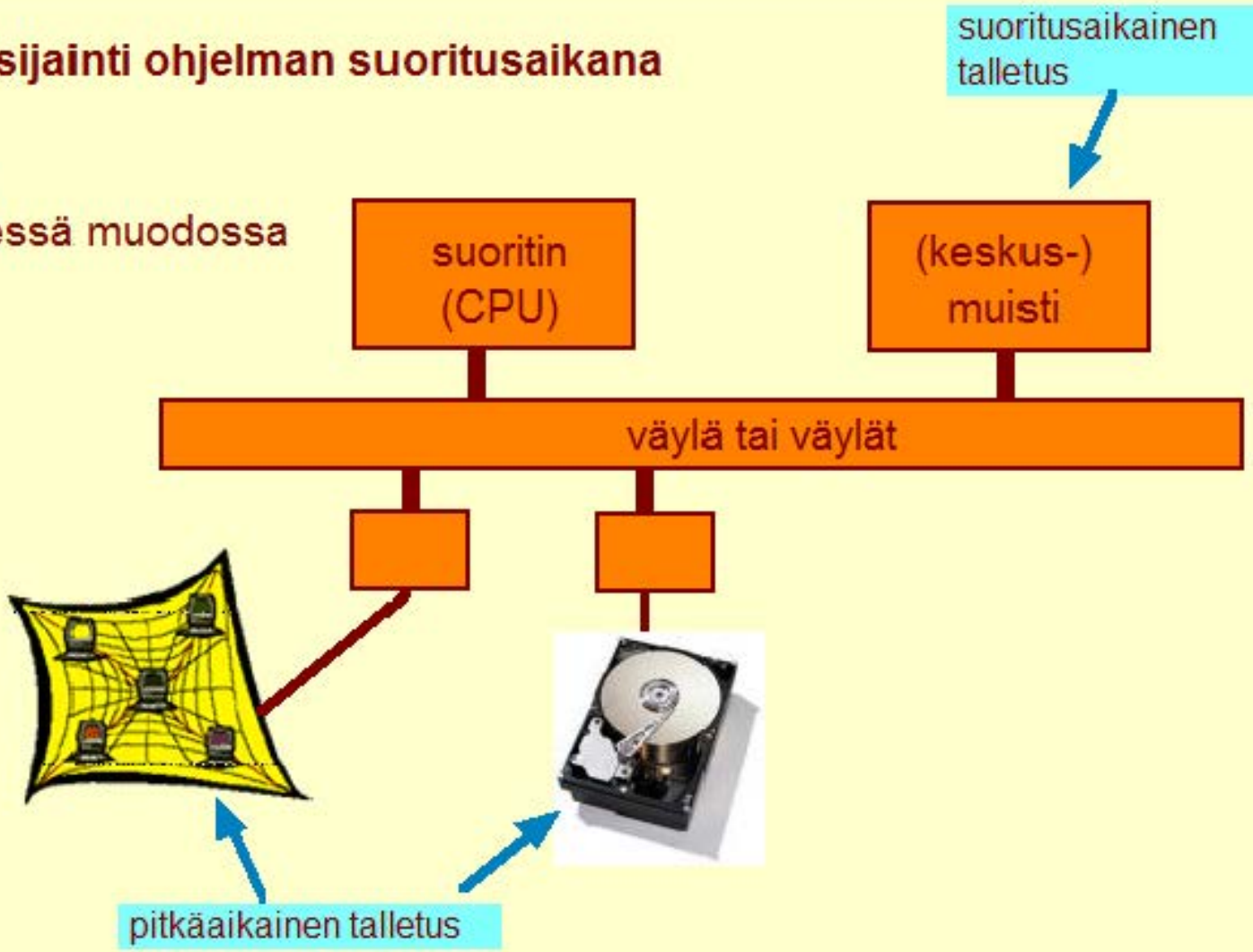
Pakattuna ehkä jollain tavoin

zip, tar, gz, BinHex4, Stuffit, Zoo, uuencode...

Pitkäaikaista talletusta (ja verkon yli tapahtuvaa tiedonsiirtoa) varten ohjelma on usein pakattu, mikä tarkoittaa, että ohjelman esitysmuoto on tiivistetty jollakin sopivalla algoritmilla. Ohjelmisto voidaan nyt tallettaa jopa 50-90% pienempään tilaan. Tällä on huomattava merkitys sekä levytilan että verkon käytön optimoinnissa. On mukavaa, jos levityksessä oleva ohjelma mahtuu yhdelle DVD:lle neljän levyn asemesta tai jos ohjelmiston siirtoon verkon yli kuluu vain 10 miruuttia 40 minuutin asemesta. Huonona puolena pakkauksessa on, että pakattu ohjelma pitää purkaa ohjelmallisesti ennen käyttöä ja myös tähän kuluu aikaa.

Ohjelman sijainti ohjelman suoritusaikana

Muistissa,
konekielisessä muodossa



Ohjelman suoritusaikana sen sijainti ja esitysmuoto on tarkkaan säädetty. Suoritusnopeusvaatimusten vuoksi ohjelma täytyy olla talletettuna järjestelmän keskusmuistiin, joka yleensä on ns. 'haihtuvaa' muistia (volatile memory), jossa tieto ei säily ilman sähkövirtaa. Lisäksi ohjelman tulee olla nimenomaan tämän järjestelmän konekielisessä muodossa, koska järjestelmän suoritin ymmärtää vain sen oman konekielen käskyjä. Meillä on siis selvä ongelma: pitkäaikaista talletusta varten ohjelma pitää tallettaa (hitaaseen) pysyväismuistiin missä tahansa esitysmuodossa, mutta suoritusta varten sen tulee olla nopeassa (haihtuvassa) keskusmuistissa.

Konekieli

Suorittimen konekielen käskykanta (konekäskyjen joukko)

Tietokoneen käskykanta-arkkitehtuuri
(ISA, Instruction Set Architecture)

Konekäsky = 10-numeroinen kokonaisluku (esim.)

```
2234563212
5437658756
```

Symbolinen konekieli

Käsky jaettu kiinteän mittaisiin kenttiin
Joidenkin kenttien arvot koodattu symboleilla
Helpompi ihmisten lukea ja kirjoittaa

```
LOAD R1, Summa
ADD  R1, =543
```


Suorittimen konekieli on sen konekäskyjen joukko. Konekäskyt ovat hyvin yksinkertaisia. Tyypillinen konekäsky laskee kaksi suorittimen rekisterin arvoa yhteen tai hakee muistista yhden luvun johonkin rekisteriin. Konekäskyt muodostavat käyttöliittymän suorittimelle, koska suoritin ymmärtää vain tätä hyvin rajoittunutta käskyjen joukkoa. Kullakin suorittimella on siis oma käskykantansa, eivätkä ne ymmärrä toisten suorittimien konekäskyjä. Jopa saman valmistajan eri suorittimilla on usein toisistaan poikkeavat käskykannat.

Konekieli

Suorittimen konekielen käskykanta (konekäskyjen joukko)

Tietokoneen käskykanta-arkkitehtuuri
(ISA, Instruction Set Architecture)

Konekäsky = 10-numeroinen kokonaisluku (esim.)

```
2234563212
5437658756
```

Symbolinen konekieli

Käsky jaettu kiinteän mittaisiin kenttiin
Joidenkin kenttien arvot koodattu symboleilla
Helpompi ihmisten lukea ja kirjoittaa

```
LOAD R1, Summa
ADD  R1, =543
```


Yhden suorittimen konekäskyt voivat olla kooltaan vaihtelevan mittaisia, tai sitten ne voivat olla kaikki saman pituisia, esimerkiksi 32 tai 64 bittiä. Esimerkkikoneen kaikki konekäskyt ovat 32 bittisiä, jolloin konekäskyjen muistista nouto on helppoa. Konekäskyt voidaan esittää kokonaislukuina, esimerkiksi 10-numeroisina etumerkittöminä kokonaislukuina. Ohjelma voidaan tällöin esittää näiden 10-numeroisten kokonaislukujen joukkona, mikä on koneelle selkeätä, mutta ihmiselle aika epäkäytännöllistä. Konekäsky on sisäisesti jaettu yleensä kiinteän mittaisiin kenttiin, joissa ilmaistaan esimerkiksi mikä operaatio on kyseessä ja mitä rekistereitä käskyssä käytetään.

Konekieli

Suorittimen konekielen käskykanta (konekäskyjen joukko)

Tietokoneen käskykanta-arkkitehtuuri
(ISA, Instruction Set Architecture)

Konekäsky = 10-numeroinen kokonaisluku (esim.)

```
2234563212
5437658756
```

Symbolinen konekieli

Käsky jaettu kiinteän mittaisiin kenttiin
Joidenkin kenttien arvot koodattu symboleilla
Helpompi ihmisten lukea ja kirjoittaa

```
LOAD R1, Summa
ADD  R1, =543
```


Symbolinen konekieli on konekielen laajennus, missä usealle konekäskyn kentälle on annettu symbolinen nimi, jolloin käskyjä on huomattavasti helpompi ihmisen lukea ja kirjoittaa. Esimerkiksi jokaiselle 8-bittiselle operaatiokoodille voidaan antaa sitä kuvaava symbolinen nimi, kuten ADD, DIV tai MUL. Suorittimen rekistereistä käytetään niiden nimiä niiden tyypin ja numeron asemesta. Rekistereiden nimistä ilmenee tällöin myös niiden tyyppi, jolloin esimerkiksi I4 olisi kokonaislukurekisteri ja F7 liukulukurekisteri. Rekisteri R4 voisi olla vaikkapa yleisrekisteri. Myös muuttujien osoitteina voidaan käyttää symboleja. On huomattavasti helpompaa puhua muuttujasta 'Summa' kuin muuttujasta osoitteessa x98A56D32.

Symbolinen konekieli

Yleinen esitystapa konekielisille ohjelmille ("luettavaa konekieltä")

Helppo muuttaa konekieleksi
- suora vastaavuus konekieleen
- usein samaistetaan puhekielessä konekieleen

```
129543876
439874387
544399765
```

konekieli (koodi)

```
LOAD   R2, Summa        ; R2 <- Memory(Summa)
ADD    R2, =5           ; R2 <- R2 + 5
JUMP   Loop             ; PC <- Loop     (hyppykäsky)
```

symbolinen konekieli (; kommentti)

Konekielen tason ohjelmat esitetään siis ihmiselle sopivassa symbolisessa konekielisessä muodossa. Esitystavat ovat kuitenkin niin lähellä toisiaan, että ne usein puhekielessä samaistetaan. Symbolien käytön lisäksi symbolisessa konekielessä on muitakin ohjelmointia ja ohjelmien lukemista helpottavia piirteitä, kuten esimerkiksi erilaiset muistiinviittausmenetelmät ja kommenttien käyttömahdollisuus. Kokenut ohjelmoija voi tuottaa koodia symbolista konekieltä käyttäen yhtä tehokkaasti kuin korkean tason ohjelmointikielen avulla, vaikka näin ei useimmiten enää tehdäkään.

Ohjelma vs. konekieli

Ongelma: ohjelma on talletettu korkean tason ohjelmointikielellä (esim. C tai Java) järjestelmän pitkäaikaismuistiin (esim. kovalevy), mutta suoritusta varten sen tulee olla suorittimen konekielellä laitteiston (keskus)muistissa.

Ratkaisu: esitysmuodon muutokset

käännös	**ohjelmointikieli => konekieli**
linkitys	**paketoidaan kirjasto-ohjelmat mukaan**
lataus	**konekielinen, linkitetty ohjelma sijoitetaan (keskus)muistiin suoritettavaksi**

Meillä on tässä nyt selkeä ongelma. Pitkäaikaista talletusta varten ohjelma voi olla missä tahansa muodossa esimerkiksi levymuistissa, mutta suoritusta varten sen tulisi olla normaalissa muistissa eli keskusmuistissa tämän suorittimen konekielisessä muodossa. Meillä täytyy siis olla jonkinlainen menetelmä muuntaa ohjelman esitystapa sen yleisestä talletukseen sopivasta muodosta sen suoritukseen sopivaan muotoon. Tavanomaisesti esitystavan muutosta ei tehdä yhdellä kertaa vaan useammassa vaiheessa. Tällä tavoin esitystavan muunnosprosessi on helpompi hallita ja toteuttaa.

Ohjelma vs. konekieli

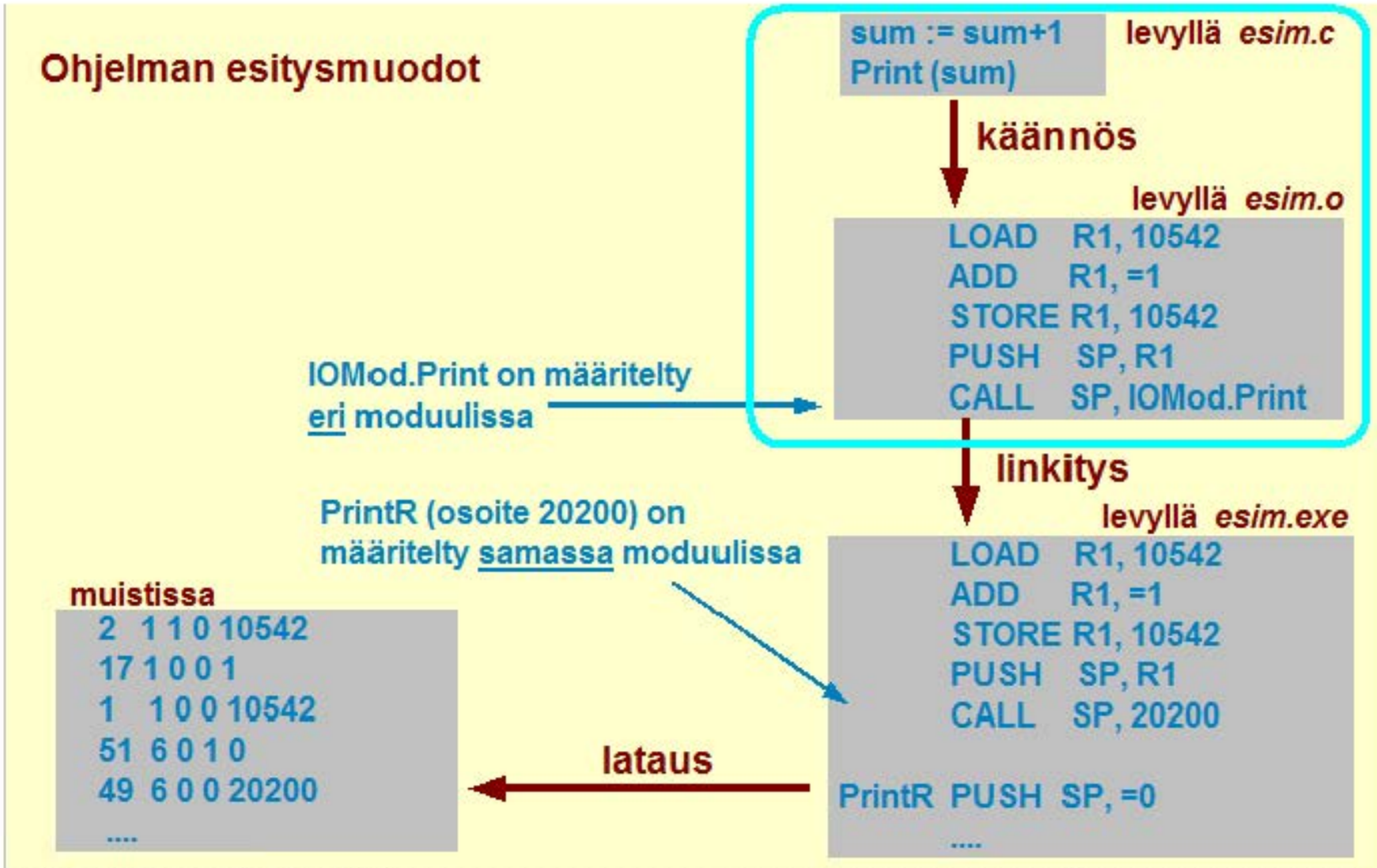
Ongelma: ohjelma on talletettu korkean tason ohjelmointikielellä (esim. C tai Java) järjestelmän pitkäaikaismuistiin (esim. kovalevy), mutta suoritusta varten sen tulee olla suorittimen konekielellä laitteiston (keskus)muistissa.

Ratkaisu: esitysmuodon muutokset

käännös	**ohjelmointikieli => konekieli**
linkitys	**paketoidaan kirjasto-ohjelmat mukaan**
lataus	**konekielinen, linkitetty ohjelma sijoitetaan (keskus)muistiin suoritettavaksi**

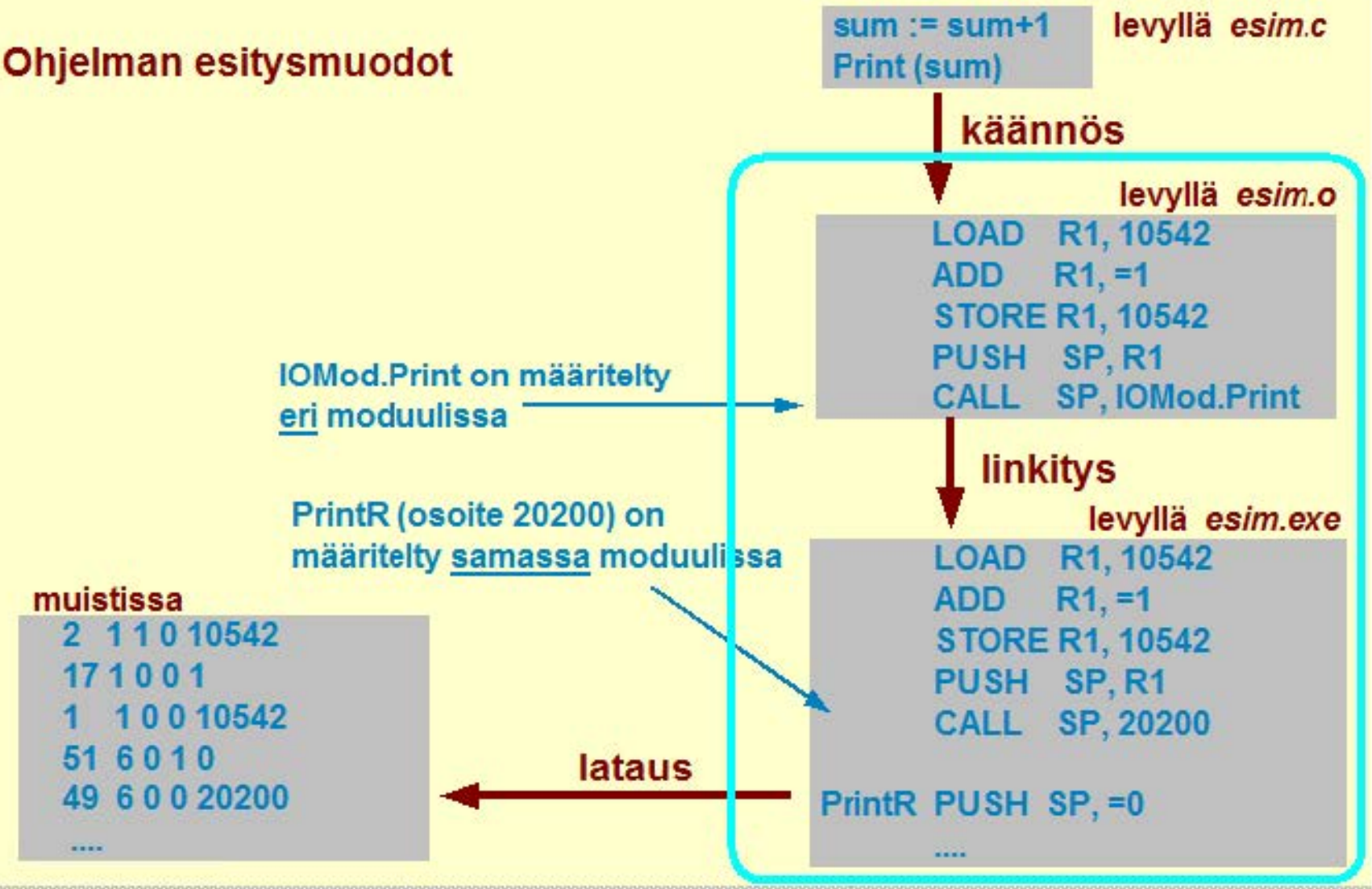
Ensimmäisessä vaiheessa korkean tason ohjelmointikielellä kirjoitettu ohjelma käännetään konekieliseen muotoon. Konekielinen ohjelmatiedosto on vielä vajavainen, koska siinä on yleensä viittauksia muihin ohjelmanosiin, ohjelmointikielen omiin kirjasto-ohjelmiin tai käyttöjärjestelmään kuuluviin kirjasto-ohjelmiin. Linkityksessä nämä muissa tiedostoissa sijaitsevat apuohjelmat paketoidaan uuden ohjelman kanssa yhtenäiseksi konekieliseksi tiedostoksi, joka talletetaan levylle. Vasta latauksessa ohjelmalle tehdään viime hetken muunnokset ja se kopioidaan keskusmuistiin suoritusta varten. Näitä kaikkia muunnosvaiheita käsitellään tarkemmin myöhemmällä luennolla.

Ohjelman esitysmuodot



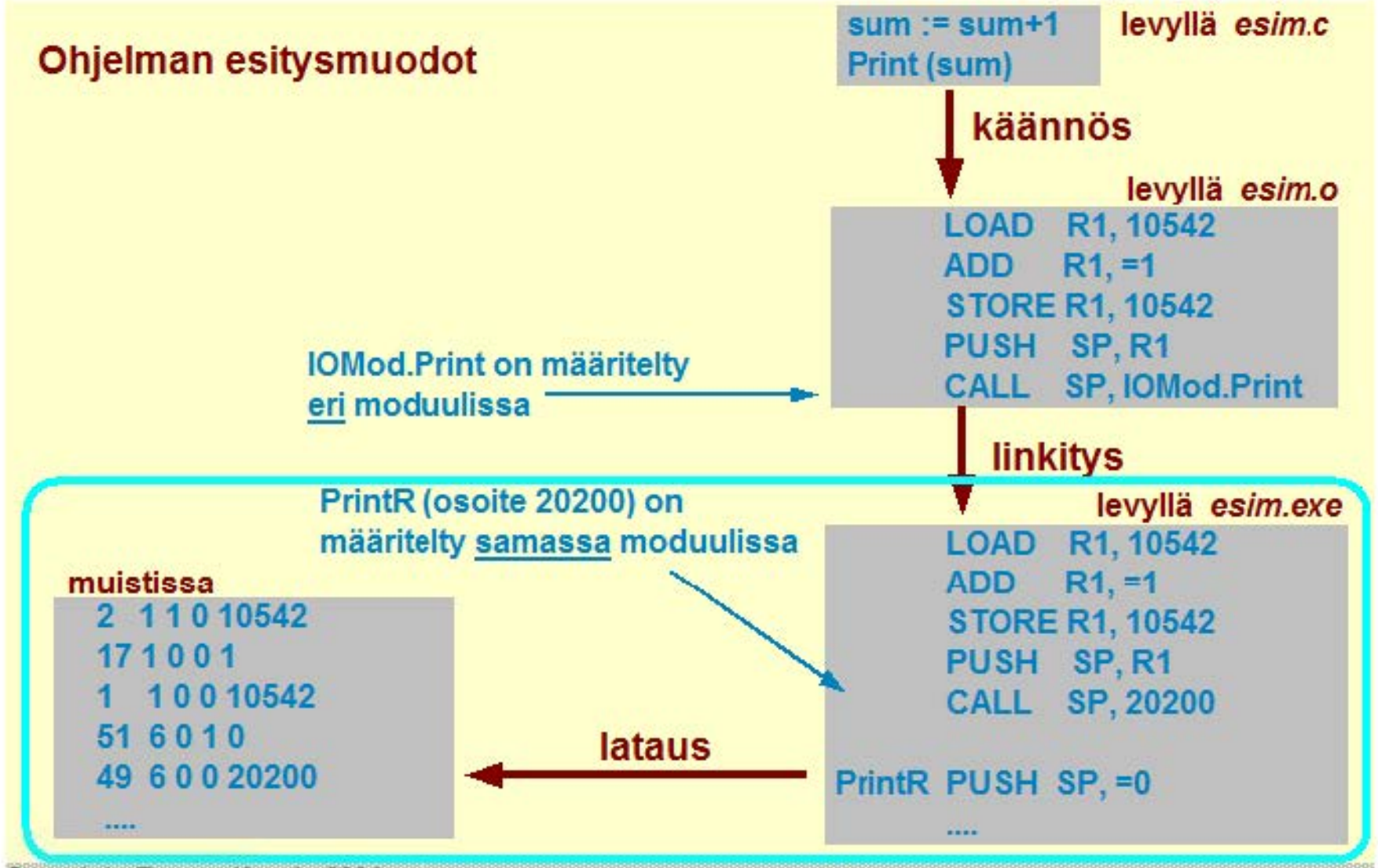
Tarkastellaan ohjelman esitysmuotoja esimerkin avulla. Meillä on geneerisellä korkean tason ohjelmointikielellä kirjoitettu ohjelmanosa, jossa muuttujan arvoa ensin kasvatetaan yhdellä ja sitten se tulostetaan. Käännöksen jälkeen sama ohjelmanosa on (symbolisessa) konekielisessä muodossa, mutta tulostusrutiinin kutsukohta on vielä auki. Tulostusrutiini sijaitseen jossakin toisessa moduulissa ja tähän ohjelmanosaan on merkitty ainoastaan tuon toisen moduulin ja kyseisen tulostusrutiinin nimi siellä. Käännetyn ohjelmanosan sisältämä moduuli talletetaan tavallisena tiedostona levylle.

Ohjelman esitysmuodot



Linkitysvaiheen jälkeen tuo IO-mocuuli on yhdistetty meidän moduuliimme ja tulostusrutiinin kutsu on nyt tavanomainen aliohjelman kutsu tässä samassa moduulissa. Sekä käännöksen että linkityksen jälkeen ohjelman symbolit on oikeastaan jo korvattu numeroarvoillaan, mutta esimerkissä vakioarvoiset symbolit (kuten esim. ADD ja R1) näytetään luettavuuden vuoksi vielä symboleina. Symboleita pidetään tallessa erityisessä suurehkossa symbolitaulussa, jossa on listattura kunkin symbolin numeerinen arvo ja sen käyttökohdat koodissa. Linkitetty koko ohjelman sisältämä moduuli talletetaan tavallisena tiedostona levylle.

Ohjelman esitysmuodot



Latauksen yhteydessä koodiin voicaan tehdä vielä joitakin viime hetken muunnoksia. Sekä itse ohjelma että kaikki sen tarvitsemat data-alueet kopioidaan muistiin käyttöjärjestelmän viittaamiin tietorakenteisiin ja ohjelmasta luodaan kayttojarjestelman tunnistama suorituskelpoinen prosessi.Tassa vaiheessa ohjelman esitysmuoto on puhtaasti numeerinen ja jokainen ohjelman konekäsky or esitetty sitä vastaavana lukuna. Symbolitaulua ei enää tarvita, ellei sitä sitten pidetä mukana ihmisläheisempien virheilmoitusten vuoksi. Symbolitaulu voidaan liittää mukaan ohjelman kehitystyön aikana ja jättää sitten pois valmiista ohjelmasta.

Ohjelman esitysmuodot, C ohjelma -esimerkki

C-kieli esim1.c

```
main ()
{
    int x;
    x = 5;
    printf ("x: %d\n", x);
}
```


Todellisuudessa sekä käännettyjen että linkitettyjen ohjelmien esitysmuodot ovat aika suuria. Tässä esimerkissä meillä on hyvin lyhyt C-kielinen ohjelma esim1.c. Jos C-kieli ei ole sinulle ennestään tuttu, älä huolestu. Ohjelmassa ensin määritellään kokonaislukuarvoinen muuttuja x, sitten sille annetaan arvoksi luku 5 ja lopuksi sen arvo tulostetaan C-kielen kirjastorutiinilla printf.

Ohjelman esitysmuodot, C ohjelma -esimerkki

C-kieli esim1.c

```
main ()
{
    int x;
    x = 5;
    printf ("x: %d\n", x);
}
```

cc -S esim1.c

symbolinen konekieli esim1.s

```
 .file "esim1.c"
 .section .rodata
.LC0:
 .string "x: %d\n"
 .text
.globl main
 .type main, @function
main:
 pushl %ebp
 movl %esp, %ebp
 subl $8, %esp
 andl $-16, %esp
```


Tämä ohjelma (esim1.c) käännetään symboliselle konekielelle Unix komennolla 'cc -S esim1.c'. Symbolinen konekielinen esitys (esim1.s) tietenkin vaihtelee kohdelaitteiston mukaan. Tässä esimerkissä kohdelaitteisto on varustettu Intelin i686 suorittimella, joten symbolinen konekieli on tuon suorittimen konekieltä. Jälleen, tarkoitus on vain näyttää todellinen esimerkki, eikä tarkastella kyseistä konekielistä koodia mitenkään yksityiskohtaisesti. Älä siis yritäkään selvittää, mitä kcodi tarkalleen tekee.

Ohjelman esitysmuodot, C ohjelma -esimerkki

C-kieli esim1.c

```
main ()
{
    int x;
    x = 5;
    printf ("x: %d\n", x);
}
```

cc -S esim1.c

symbolinen konekieli esim1.s

```
 .file "esim1.c"
 .section .rodata
.LC0:
 .string "x: %d\n"
 .text
.globl main
 .type main, @function
main:
 pushl %ebp
 movl %esp, %ebp
 subl $8, %esp
 andl $-16, %esp
```

cc esim1.s

ladattava moduuli esim1

```
0000000  1179403647        65793            0            0
0000020      196610            1    134513344           52
0000040        1928            0      2097204      2621447
0000060     1638428            6           52    134512692
0000100   134512692          224          224            5
0000120           4            3          276    134512916
0000140   134512916           19           19            4
0000160           1            1            0    134512640
0000200   134512640         1148         1148            5
0000220        4096            1         1148    134517884
```

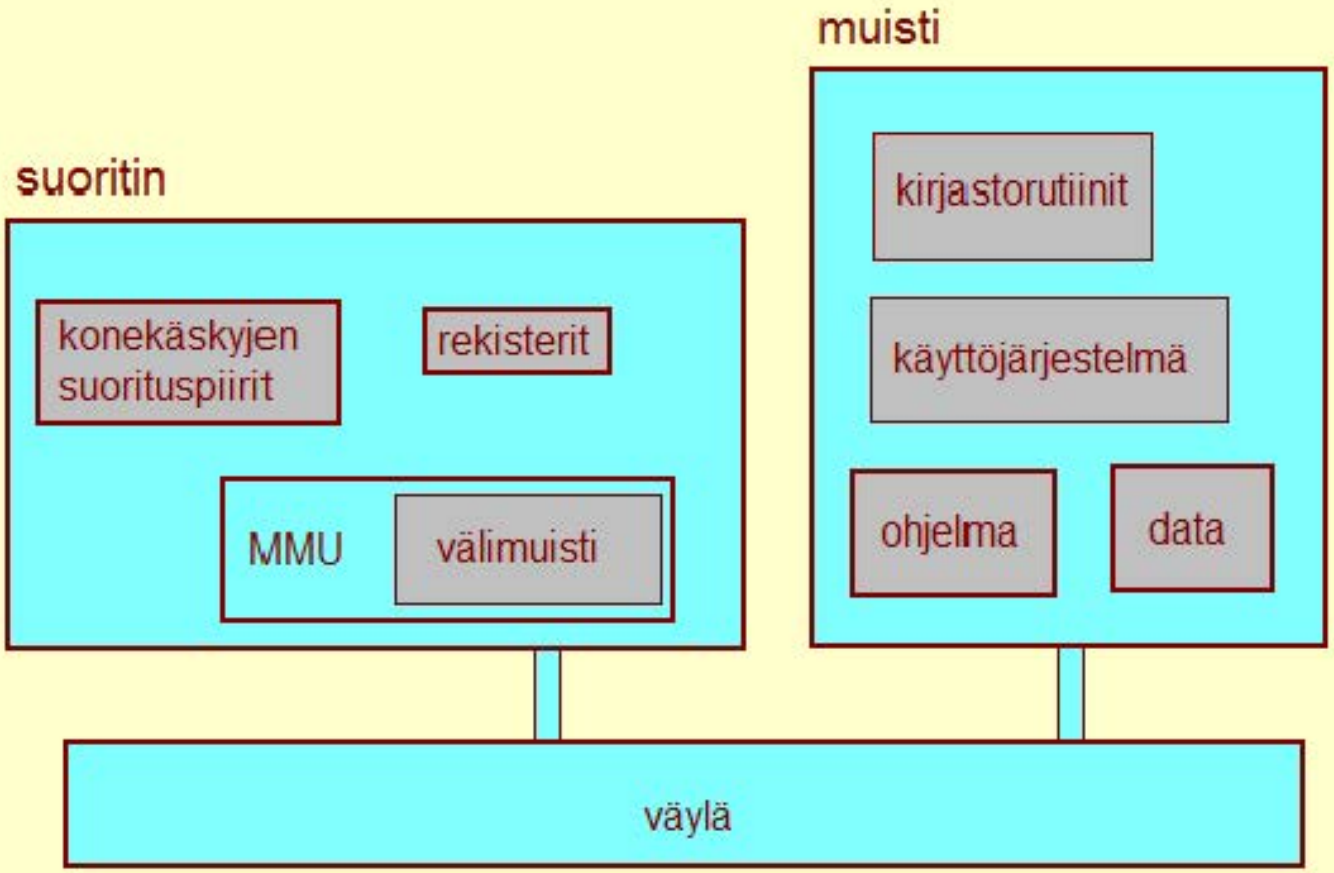
esim1

ohjelman tuloste

x: 5

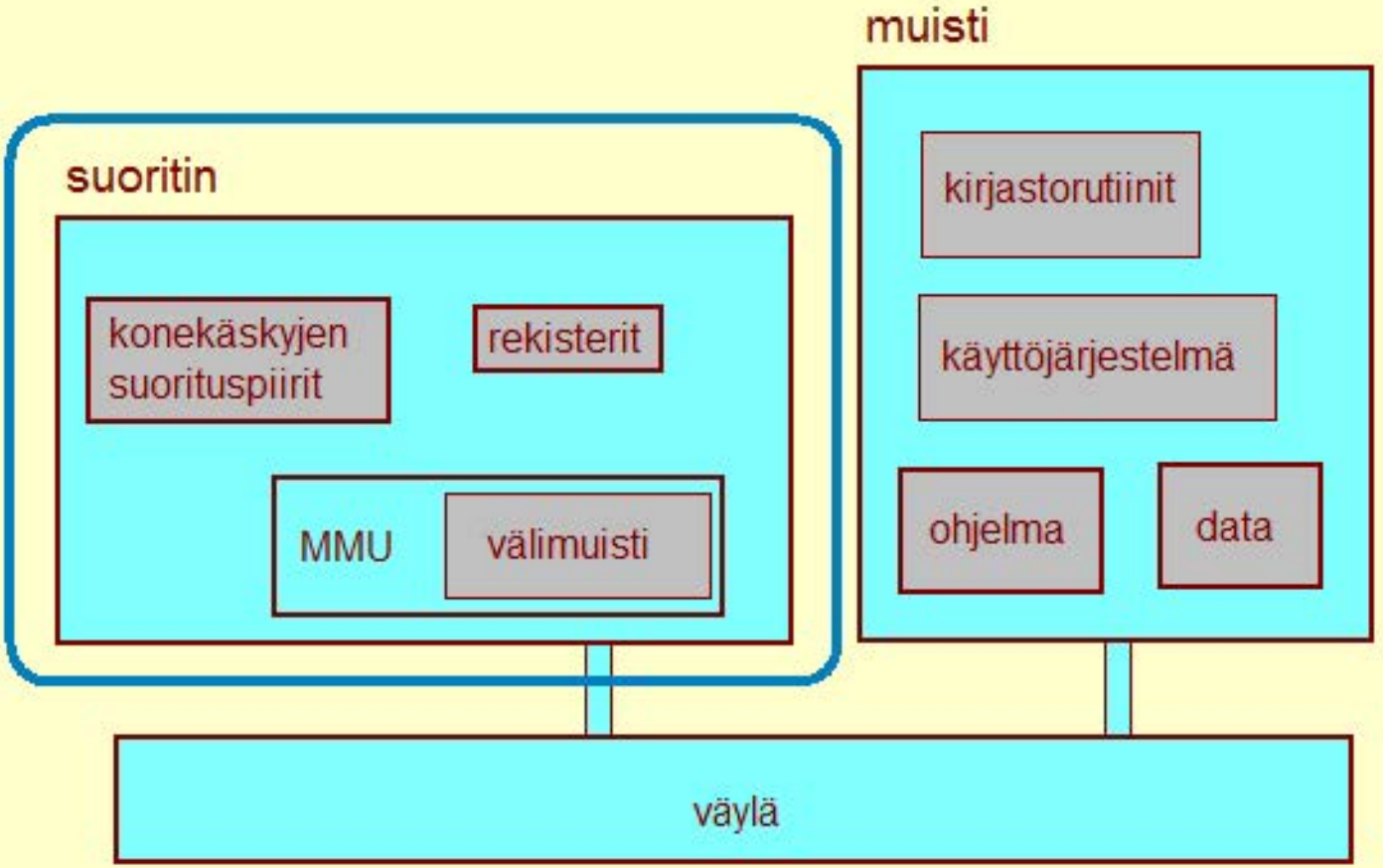
Yksittäinen käännetty moduuli (esim1.s) voidaan linkittää vakiokirjastojen kanssa yksinkertaisella Unix-komennolla 'cc -o esim1 esim1.s', joka ennen linkitystä kääntää symbolisen konekielisen esityksen (esim1.s) puhtaaseen konekieliseen muotoon. Ladattava Unix-moduuli (esim1) sisältää 4735 kappaletta 32-bittisiä lukuja. Ladattavan moduulin esityksessä on vasemmassa reunassa moduulin osoitteet oktaaliesityksessä. Latauksen jälkeen muistissa on tietenkin ohjelmakoodin lisäksi paljon muuta käyttöjärjestelmätietoa tästä ohjelman suorituskerrasta. Ohjelmaa esim1 suoritettaessa se sitten tietenkin vain tulostaa muuttujan x uuden arvon 5.

Suorittimen ja muistin sisältö



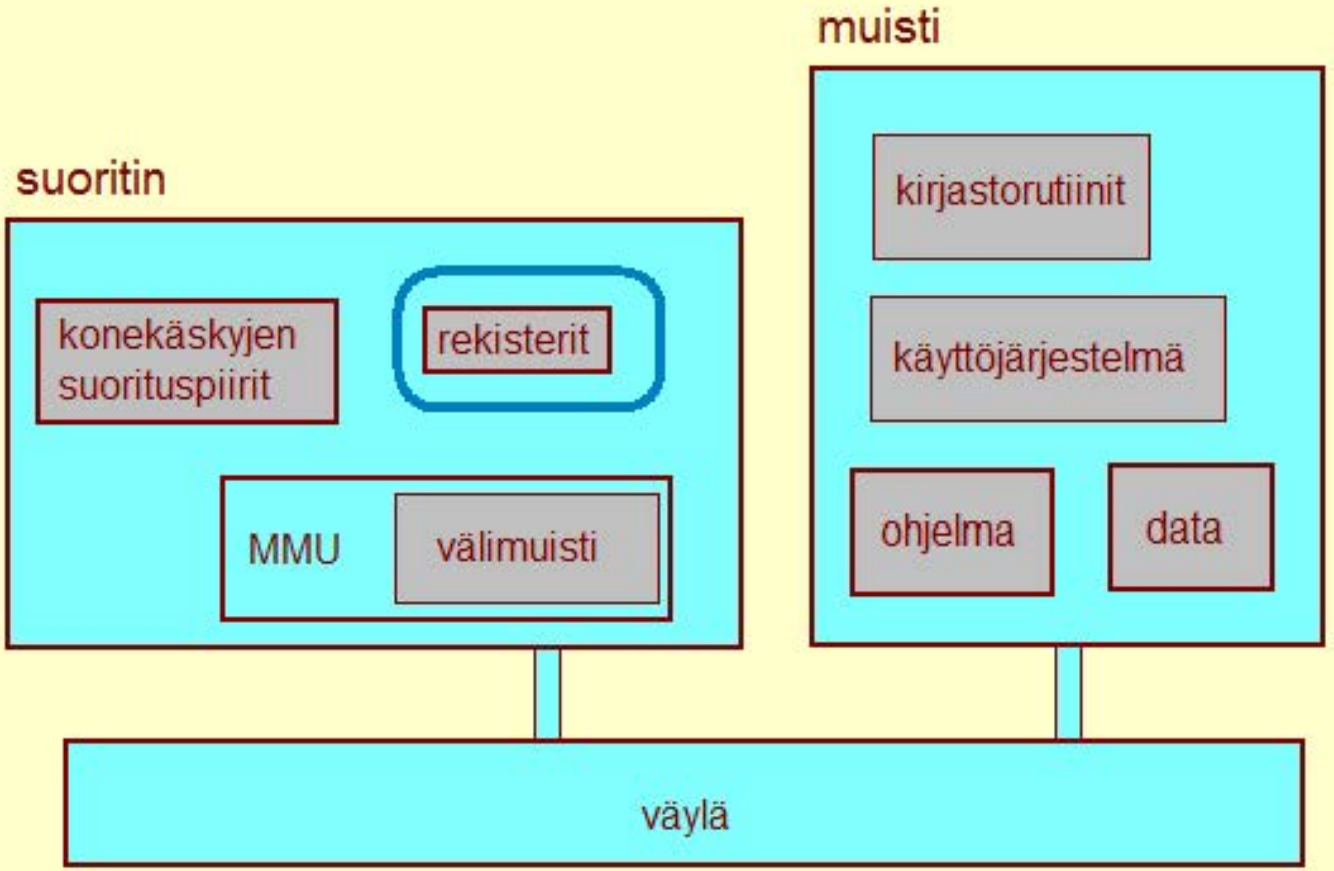
Peruslaitteisto ohjelmien suorituksen kannalta koostuu suorittavasta yksiköstä eli suorittimesta ja muistista, jossa kaikenlaista tietoa - sekä ohjelmia että dataa - pidetään tallessa ohjelmia suoritettaessa. Tietoa siirretään väylää pitkin, johon myös kaikki oheislaitteet on kytketty. Unohdamme nyt tässä vaiheessa oheislaitteet ja keskitymme vain suorittimeen ja muistiin.

Suorittimen ja muistin sisältö



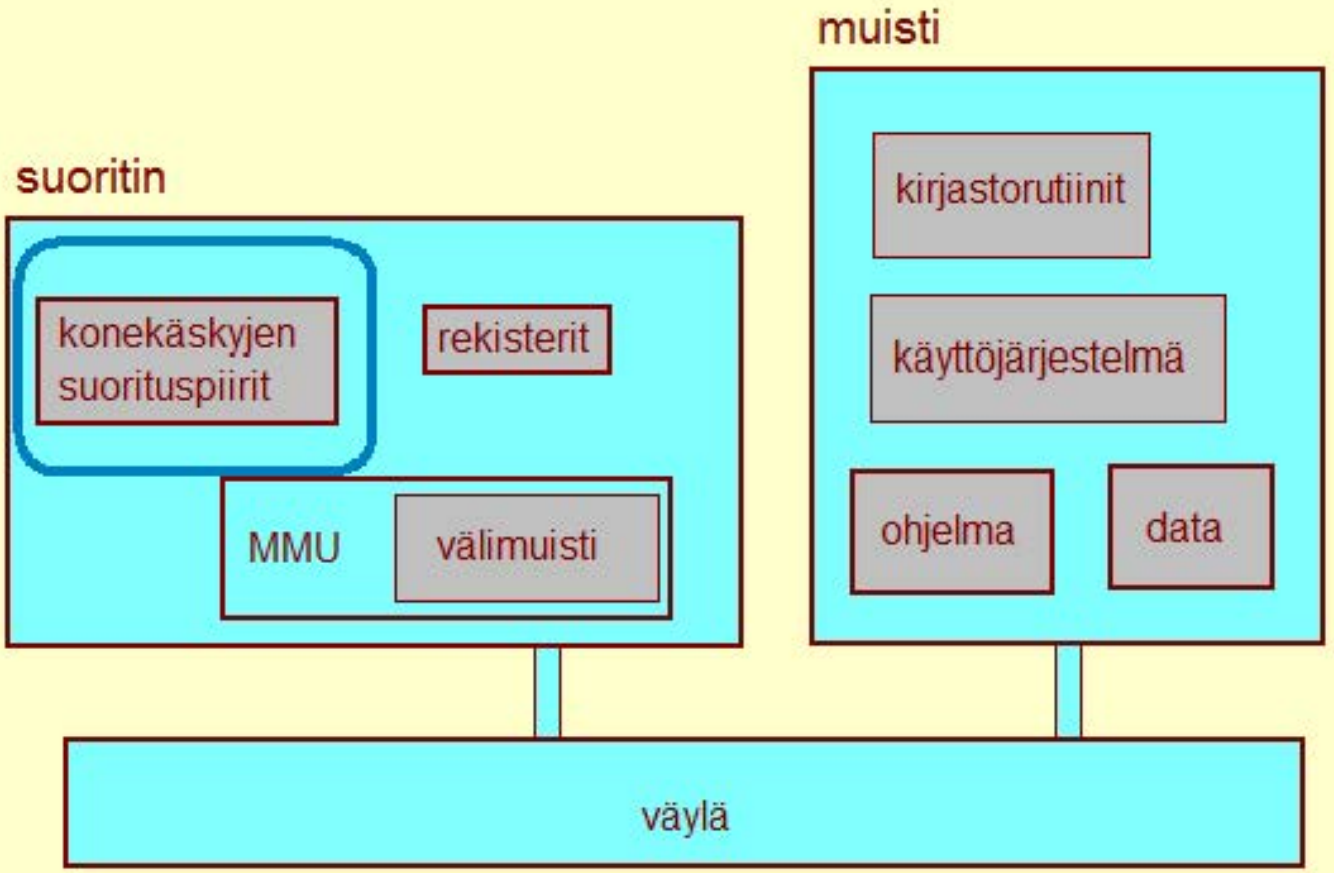
Suorittimella on fyysisesti erillisiä komponentteja, jotka on yhdistetty suorittimen sisällä johtimilla toisiinsa. Tarkastelemme tässä konekäskyjen suorituspiirejä, suorittimen rekistereitä, ja muistinhallintayksikön (MMU) välimuistia. Muitakin komponentteja on, mutta tutustumme niihin myöhemmin.

Suorittimen ja muistin sisältö



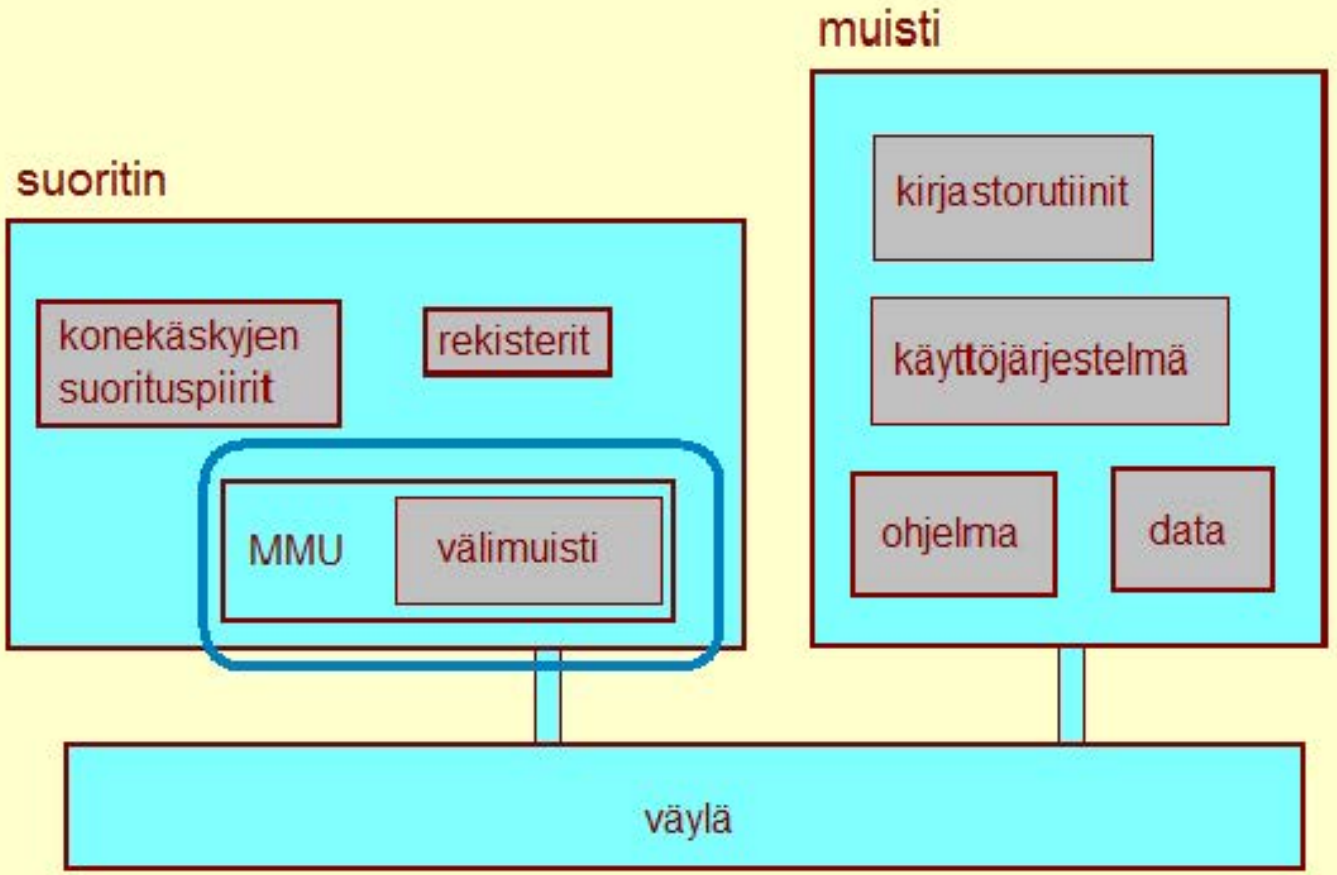
Laiterekisterit muodostavat pienen, nopean muistialueen suorittimen sisällä. Kaikki tietokoneen laskenta tapahtuu suorittimella ja kaikki laskenta tapahtuu näiden rekistereiden sisältämien lukujen välillä. Muistissa olevia lukuja ei siis suoraan manipuloida, vaan kaikki data tulee ladata muistista rekistereihin laskentaa varten. Rekistereitä on vain muutama kappale, esimerkiksi 32 tai 128. Esimerkkikoneessa on 8 konekäskyissä osoitettavaa rekisteriä ja muutama muu niiden lisäksi. Vertailun vuoksi kannattaa pitää mielessä, että muistin koko on tyypillisesti ainakin miljoona kertaa noin iso. Rekistereiden lukumäärää taas ei voi kasvattaa hidastamatta niiden käyttöä.

Suorittimen ja muistin sisältö



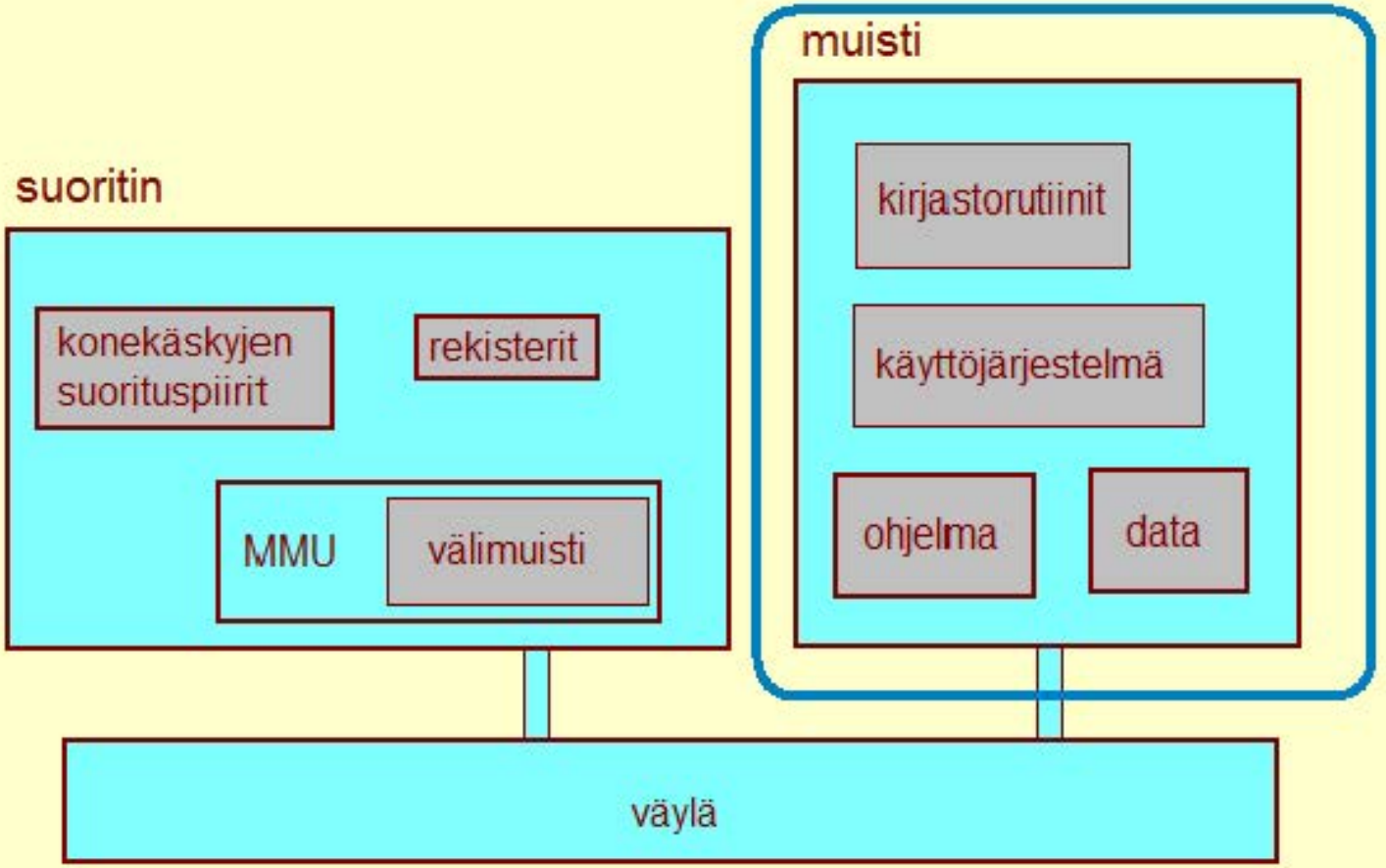
Konekäskyjen suorituspiirit tekevät varsinaisen työn. Kaikki laskutoimitukset tapahtuvat loppujen lopuksi siten, että suorituspiirit hakevat operandit rekistereistä, suorittavat itse laskennan, ja tallettavat tuloksen johonkin rekisteriin. Jotkut konekäskyt operoivat muistin kanssa, jolloin ne pyytävät muistinhallintayksikköä (MMU) siirtämään tietoa muistin ja rekistereiden välillä. Tiedon haku muistista kestää tyypillisesti 10-20 kertaa niin kauan kuin rekisteristä, joten muistista hakuihin liittyy yleensä suorittimen odottelua.

Suorittimen ja muistin sisältö



Rekisterit ovat siis 10-20 kertaa nopeampia kuin keskusmuisti, mutta niiden lukumäärä on vain ehkä miljoonasosa muistipaikkojen lukumäärästä. Tästä seuraa, että hyvin usein käsiteltävä tieto pitää ensin hakea muistista rekisteriin tai rekisterissä oleva tieto pitää tallettaa muistiin. Välimuisti on rekisterijoukkoa isompi muistinhallintayksikön toteuttama muistialue, mutta sen koko on selvästi keskusmuistia pienempi. Sen nopeus on lähes rekistereiden luokkaa ja siellä pidetään automaattisesti viime aikoina viitattujen muistialueiden kopioita. Tällä tavoin aika usein (esim. 99%) viitattu muistipaikka löytyykin nopeasta välimuistista eikä hitaammasta keskusmuistista. Esimerkkikoneessa ei ole välimuistia.

Suorittimen ja muistin sisältö



Keskusmuisti on fyysisesti yksi yhtenäinen moduuli ja sen kaikki muistialueet ovat samanarvoisia ja voivat sisältää sekä ohjelmia että dataa. Ohjelman suorituksen aikana siellä on koko ohjelmakoodi ja kaikki ohjelman tarvitsema data. Sen lisäksi muistissa on käyttöjärjestelmä, joka sekin koostuu useasta ohjelmasta ja aika suuresta määrästä hallintodataa. Muistissa on myös erilaisia kirjastorutiineja tämän ja muiden ohjelmien käyttöön. Todellisessa tietokoneessa muistissa on samaan aikaan myös monien muiden ohjelmien koodi- ja data-alueet, mutta esimerkkikoneessamme keskitymme vain suorituksessa olevan ohjelman omiin koodi- ja data-alueeseen.

Laitteiston nopeus

Järjestelmän eri komponenteilla on hyvin suuret nopeuserot

Laiterekisterit kaikkein nopeimmat	Esim. 1 ns
Välimuisti lähes yhtä nopea	Esim. 2 ns
Muisti aika hidasta (rekistereihin verrattuna)	Esim. 10 ns
Laitteet hyvin kaukana	Esim. 10-30 ms
Muut tietokoneet hyvin kaukana	Esim. 10-30 ms
Eräät laitteet todella hyvin kaukana (magneettinauha)	Esim. 1-100 s
Ihminen yleensä todella hyvin kaukana suorittimesta eli ihmisen käyttämät laitteet ovat hyvin hitaita	Esim. 0.5-3 s

näppäimistö, hiiri

Laitteiston osat ovat yllättävän eri nopeuksisia ja tämä aiheuttaa huomattavia ongelmia laitteistojen ja niitä hyödyntävien ohjelmistojen suunnittelussa. Laiterekisterit on yleensä suunniteltu niin pieniksi, että niiden nopeus täsmää laskentaa suorittavien piirien kanssa yhteen. Välimuisti on lähes yhtä nopeaa, mutta ei ihan. Muisti on jo vähän hidasta suorittimelle, minkä vuoksi ohjelmien konekielisessä toteutuksessa joudutaan kiinnittämään paljon huomiota siihen, että juuri nyt laskennassa käytettävä data olisi valmiina jossakin rekisterissä.

Laitteiston nopeus

Järjestelmän eri komponenteilla on hyvin suuret nopeuserot

Laiterekisterit kaikkein nopeimmat	Esim. 1 ns
Välimuisti lähes yhtä nopea	Esim. 2 ns
Muisti aika hidasta (rekistereihin verrattuna)	Esim. 10 ns
Laitteet hyvin kaukana	Esim. 10-30 ms
Muut tietokoneet hyvin kaukana	Esim. 10-30 ms
Eräät laitteet todella hyvin kaukana (magneettinauha)	Esim. 1-100 s
Ihminen yleensä todella hyvin kaukana suorittimesta eli ihmisen käyttämät laitteet ovat hyvin hitaita	Esim. 0.5-3 s

näppäimistö, hiiri

Kaikki oheislaitteet ovat jo ihan liian kaukana suorittimen näkökulmasta. Yleensä suoritin ei voi odottaa dataa oheislaitteelta, vaan kyseisen ohjelman suoritus keskeytetään oheislaitteen odottelun ajaksi. Sama pätee muihin tietokoneisiin verkon takana, esimerkkinä vaikkapa oma tiedostopalvelin tai Internetin jokin palvelinkone. Ihmisen käyttämät laitteet ovat yleensä niitä ihan hitaimpia, koska ihmisen reagointiaikaskaala on sekuntiluokkaa. Tosin joillakin graafisilla käyttöliittymillä ihmisenkin antama syöte voi tulla hyvin nopeasti.

Teemun juustokakku

Rekistereiden, välimuistin, muistin, levymuisti, magneettinauhan ja ihmisen nopeudet suhteutettuina suorittavan laitteiston nopeuteen verrattuna juuston hakuaikaan juustokakkua tehdessä

Missä tieto?

Rekisterissä? Välimuistissa? Muistissa? Levyllä? Ihminen antaa?

Miten kauan kestää tiedon haku?

Missä juusto?

Miten kauan kestää juuston haku?

Juustokakkuesimerkki kuvaa käytännönläheisesti laitteiston eri komponenttien suhteellisia nopeuksia antamalla vastaavan esimerkin ihmiselle paremmin sopivassa aikaskaalassa. Kuvittele, että olet keittiössäsi tekemässä juustokakkua ja tarvitse sopivan määrän pehmeätä kermajuustoa kakkuasi varten. Missä juusto on? Älä pahastu, vaikka tämä esimerkki voi tuntua vähän lapselliselta. Sorry. Esimerkki kuitenkin heijastaa konkreettisesti laitteiston nopeuseroja ja auttaa sinua hahmottamaan niiden merkityksen tietokonelaitteistossa. Ihmisen aikaskaalassahan kaikki tietokoneen komponentit ovat hyvin nopeita.

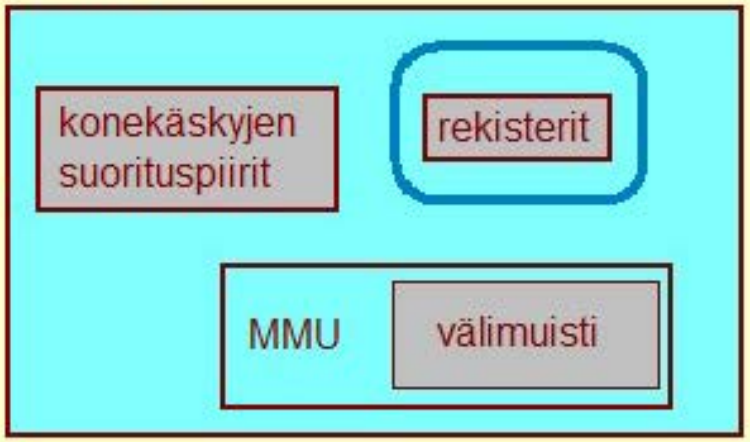
Teemun juustokakku

Rekistereiden, välimuistin, muistin, levymuisti, magneettinauhan ja ihmisen nopeudet suhteutettuina suorittavan laitteiston nopeuteen verrattuna juuston hakuaikaan juustokakkua tehdessä

Missä juusto?
Kokin kädessä

1 s

Missä tieto?
Suorittimen rekisterissä



1 ns

Parhaimmassa tapauksessa juusto on jo kokin kädessä. Siitä sen voi käden käänteessä, yhdessä sekunnissa ottaa käyttöön, ilman että kakun tekeminen hidastuu lainkaan. Tämä vastaa tietokoneessa tilannetta, jossa laskennassa käytettävä tieto löytyy suorittimen rekisteristä. Todellisuudessa tiedon siirtoon rekisteristä laskentapiireille voisi mennä aikaa vaikkapa 1 ns. Nanosekunnin lyhyyttä kuvaa hyvin se, että valo kulkee tyhjiössä yhden nanosekunnin aikana noin 30 cm.

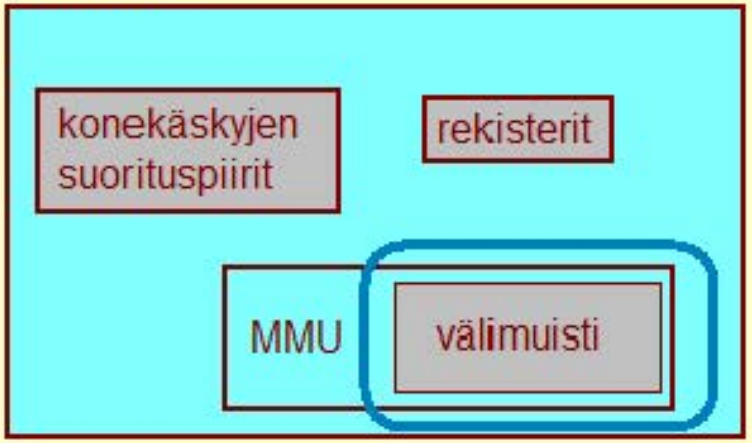
Teemun juustokakku

Rekistereiden, välimuistin, muistin, levymuisti, magneettinauhan ja ihmisen nopeudet suhteutettuina suorittavan laitteiston nopeuteen verrattuna juuston hakuaikaan juustokakkua tehdessä

Missä tieto?
Välimuistissa

Missä juusto?
Pöydällä

suoritin



2 s

2 ns

Seuraavaksi paras tilanne juuston löytymiseen nopeasti on, että se on valmiina käytettäväksi pöydällä. Ei sen käyttämiseen nytkään kulu paljon aikaa, ehkä vain 2 sekuntia, jos kokki on valmiina pöydän vieressä. Tietokoneessa tämä vastaa tilannetta, jossa viitattu tieto haetaan muistista, mutta se löytyykin huomattavsti muistia lähempänä olevasta välimuistista. Välimuisti on toteutettu samalla lastulla kuin suoritinkin, joten sen käyttö on hyvin nopeata eikä tiedon viittaamisen tarvita väylää lainkaan. Aikaa voisi kulua vaikkapa 2 ns.

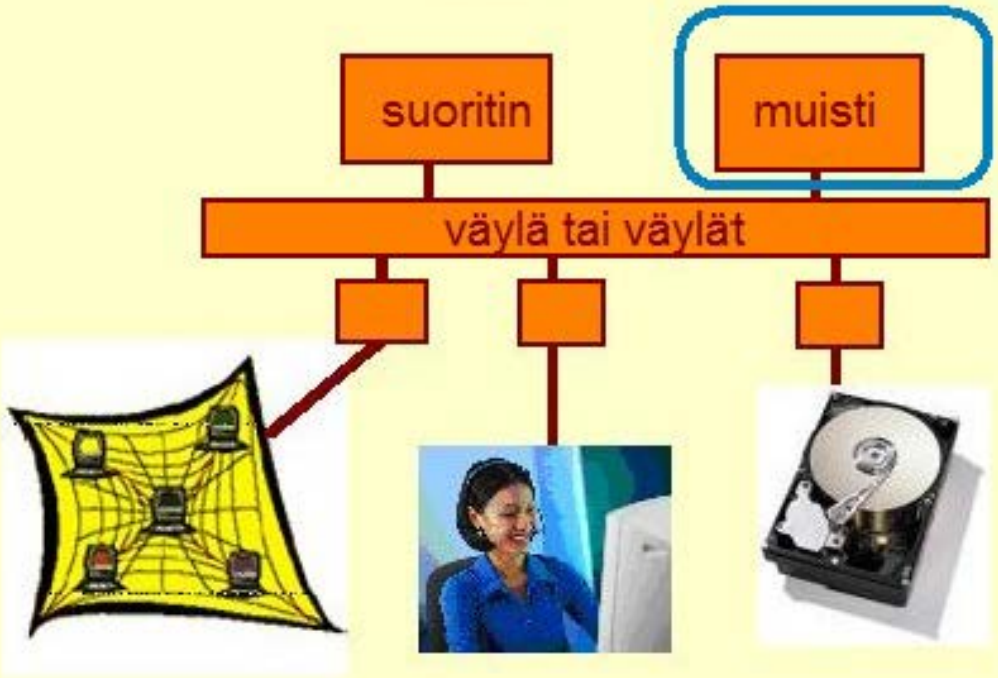
Teemun juustokakku

Rekistereiden, välimuistin, muistin, levymuisti, magneettinauhan ja ihmisen nopeudet suhteutettuina suorittavan laitteiston nopeuteen verrattuna juuston hakuaikaan juustokakkua tehdessä

Missä tieto?
Muistissa
10 ns

Missä juusto?
Jääkaapissa

10 s



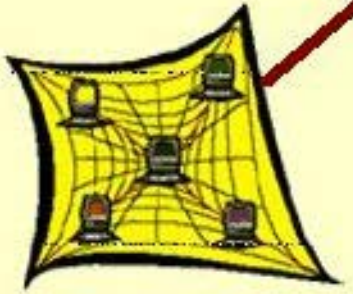
Jos juusto haetaan jääkaapista, niin tähän kuluu jo 10 sekuntia. Pitää ensin kävellä jääkaapille, avata ovi, ottaa juusto esille ja tuoda se takaisin pyödän ääreen. Tämä nyt kuitenkin pitää tehdä, joten ei auta muuta kuin ryhtyä hommiin. Tietokoneessa tämä vastaa muistiviitteen suoritusta. Väylä varataan tiedon siirtoa varten, tieto etsitään muistipiiristä ja kopioidaan väylän kautta suorittimen rekisteriin. Tähän voisi kulua vaikkapa 10 ns, minkä aikana yksinkertainen suoritin vain odottelee. Aika (10 ns) on kuitenkin liian lyhyt siihen, että odotusaikana tekisi jotain muuta hyödyllistä. Fiksumpi suoritin osaisi hyödyntää odotusaikaa paremmin, mutta siitä myöhemmin lisää!

Teemun juustokakku

Rekistereiden, välimuistin, muistin, levymuisti, magneettinauhan ja ihmisen nopeudet suhteutettuina suorittavan laitteiston nopeuteen verrattuna juuston hakuaikaan juustokakkua tehdessä

Missä juusto?
Kuussa

Missä tieto?
Levymuistissa



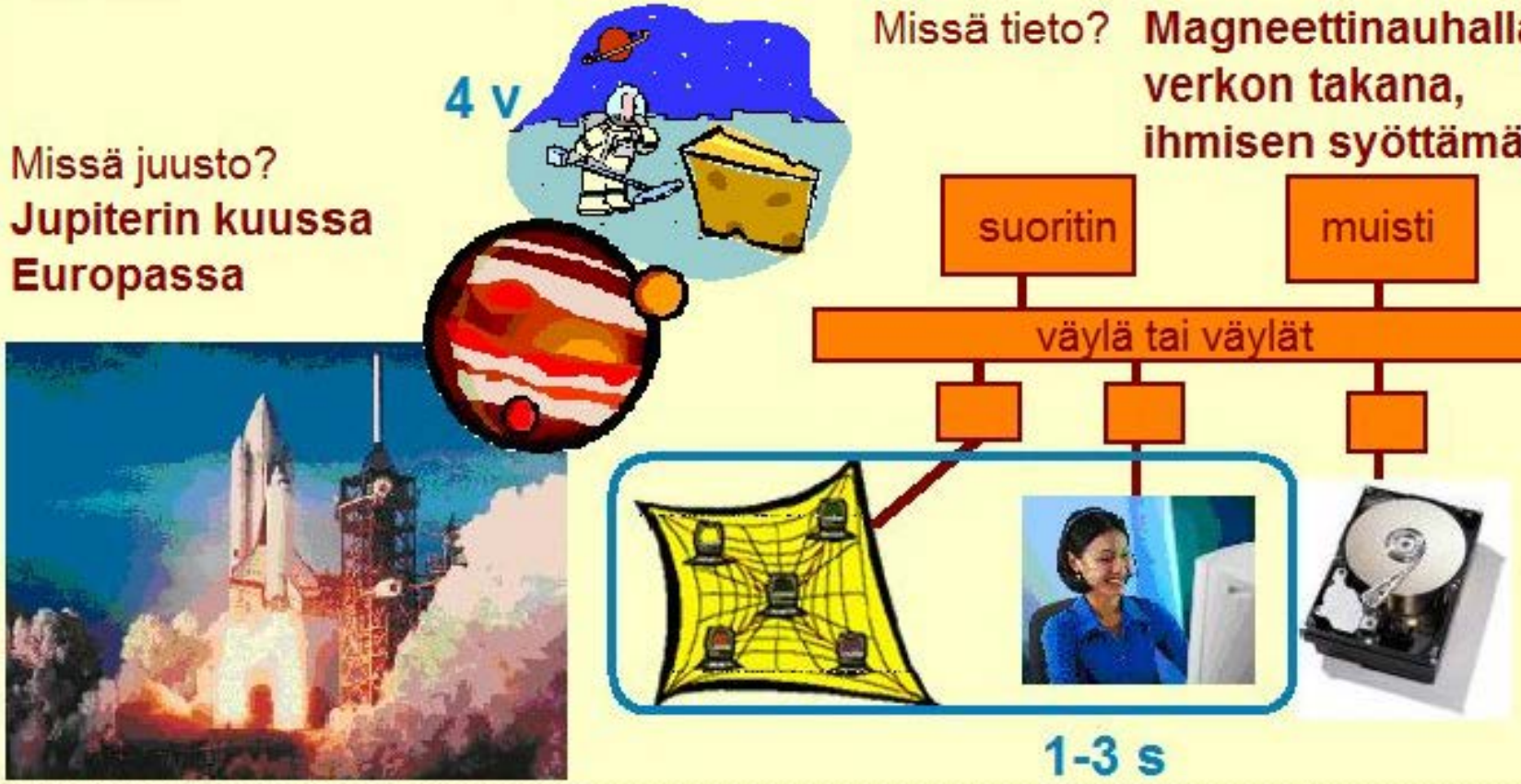
12 päivää

10-30 ms

Oletetaan nyt, että juusto pitää hakea Kuusta. Varaamme siis ensimmäisen lennon Helsingistä Cape Canaveral'iin, josta kunnostettu Saturnus-raketti lähettää juustonhakijat kuumatkalle. Aikaa voisi realistisesti kulua 12 päivää. Lienee selvää, että juustokakun tekeminen kyllä keskeytyy kuumatkan ajaksi, koska muut ainekset pilaantuisivat sinä aikana pöydällä. Kokki voisi paistaa vaikkapa pihvejä odotellessaan. Tämä vastaa hyvin levymuistin nopeutta, koska levyn saantiaika on suuruusluokkaa 10 ms eli 10 miljoonaa ns. Suorittimen ei kannata vain odottaa tiedon hakua levyltä, koska sinä aikana voisi suorittaa miljoonia konekäskyjä. Odottelun aikana tyypillisesti suoritetaan muita ohjelmia tai käyttöjärjestelmän hallintoa.

Teemun juustokakku

Rekistereiden, välimuistin, muistin, levymuisti, magneettinauhan ja ihmisen nopeudet suhteutettuina suorittavan laitteiston nopeuteen verrattuna juuston hakuaikaan juustokakkua tehdessä



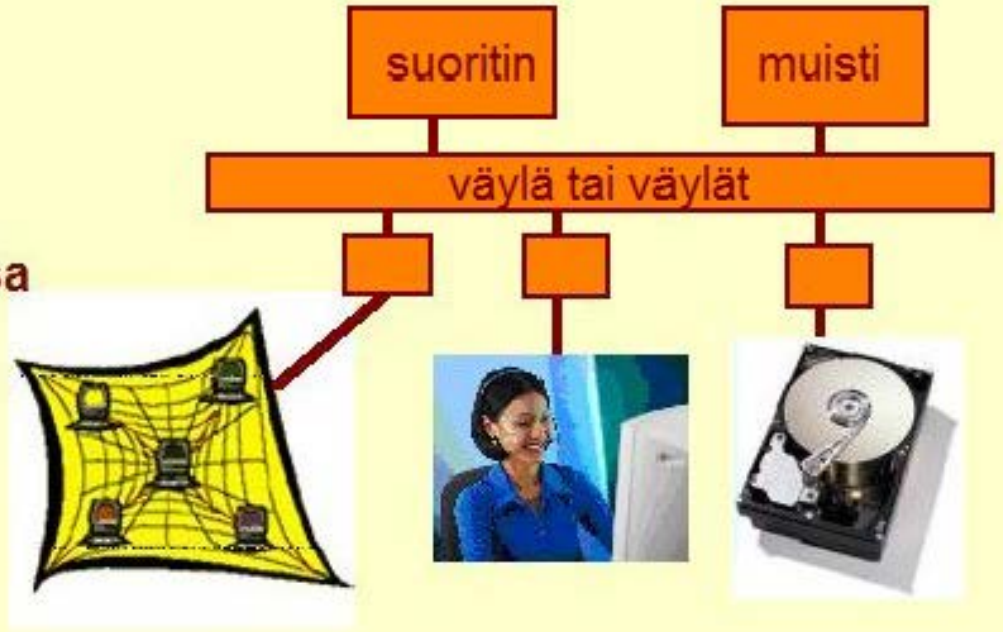
Jos Kuusta ei saa kunnon kermajuustoa, niin ehkäpä Jupiterin kuu Europa olisi parempi paikka. Nykyteknologialla sinne ei vielä päästä, mutta hyvä aika-arvio olisi 4 vuotta, koska Europa on niin kaukana. Juustokakun tekijällä olisi nyt todellisia ongelmia kakun tekemisensä kanssa, kun ei edes tiedetä, kuinka paljon kakkua neljän vuoden päästä tarvitaan. Tietokoneessa tämä on kuitenkin ihan tavallinen ongelma ja vastaa hyvin tiedon lukua verkosta tai ihmisen antamasta syötteestä.

Teemun juustokakku

Rekistereiden, välimuistin, muistin, levymuisti, magneettinauhan ja ihmisen nopeudet suhteutettuina suorittavan laitteiston nopeuteen verrattuna juuston hakuaikaan juustokakkua tehdessä

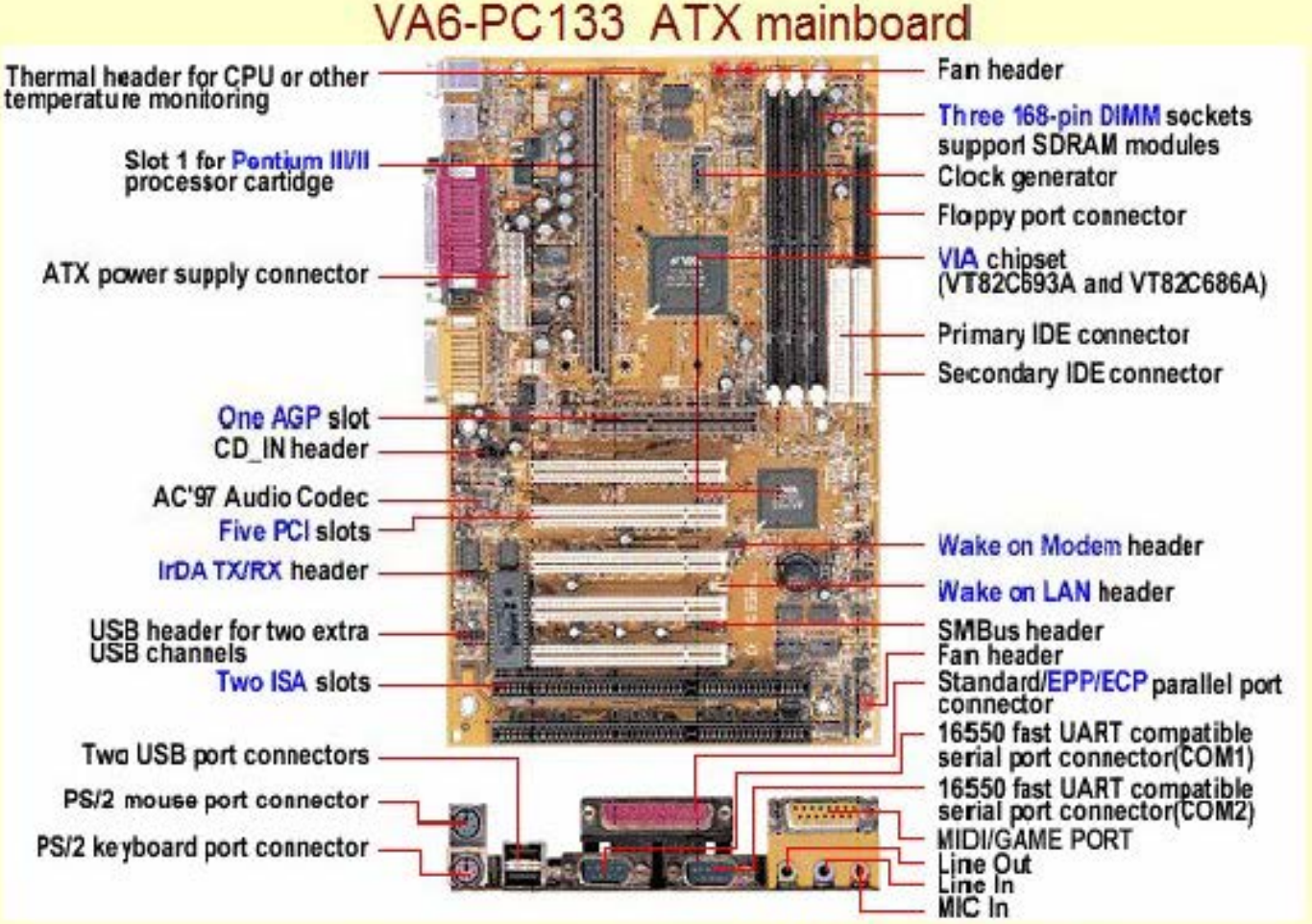
Missä juusto?

Kädessä
Pöydällä
Jääkaapissa
Kuussa
Jupiterin kuussa Europassa



Juustokakkuesimerkki heijastaa hyvin monia tietokonejärjestelmien ongelma-alueita. Esimerkiksi, kaikkien ohjelmien tulee toimia hyvin riippumatta niiden suoritusnopeudesta. Välimuisti on kehitetty ratkaisuksi nimenomaan rekisteriviittauksien ja muistiviittauksien suureen nopeuseroon. Muistin ja levyn välistä nopeuseroa ei ole (vieläkään) ratkaistu niiden väliin sijoitetuilla fyysisillä laitteilla, mutta muuten nopeuseron tuomia haittoja on minimoitu puskuroimalla ja virtuaalimuistiratkaisuilla. Suorittimen luppoaikaa minimoidaan suorittamalla jotain toista ohjelmaa levylukua odotellessa ja uusimmissa suorittimissa jopa muistilukua odotellessa. Nämä asiat kyllä kuuluvat jo myöhempiin kursseihin.

Yhteenveto



http://www.abit.nl/english/product/

Kävimme juuri läpi tietokonejärjestelmän rakenteen yleispiirteet. Näytimme, kuinka erilaiset oheislaitteet liitetään järjestelmään ja miten oheislaitteet voidaan luokitella karkeasti kolmeen luokkaan. Kävimme myös läpi laitteiston suuria sisäisiä nopeuseroja juustokakkuesimerkin nojalla.