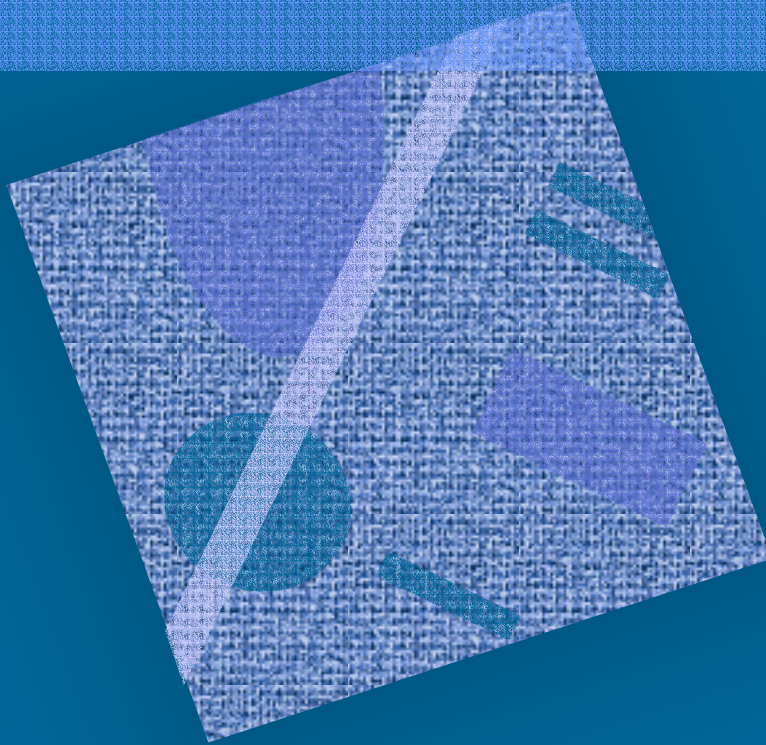


Luento 12

Yhteenveto



Keskeiset asiat
Mitä hyötyä tästä on?
Mitä seuraavaksi?
Kurssit?
Asiat?

Tavoitteet (4)

- Ymmärtää tietokonejärjestelmän keskeiset piirteet sillä suoritettavan ohjelman näkökulmasta
- Miten tietokonejärjestelmä suorittaa sille annettua ohjelmaa?
- Minkälaista koodia suoritin ymmärtää?
- Mikä on käyttöjärjestelmän rooli?

Mitä hyötyä tästä on? (2)

- Ohjelman suoritusnopeus perustuu suorittimen (CPU) suorittamiin konekäskyihin, ei pelkästään ohjelman korkean tason esitysmuotoon
- Ylemmän tason asioiden ymmärtäminen on helpompaa (mahdollista), kun ymmärtää alemman tason asiat

Keskeisiä asioita

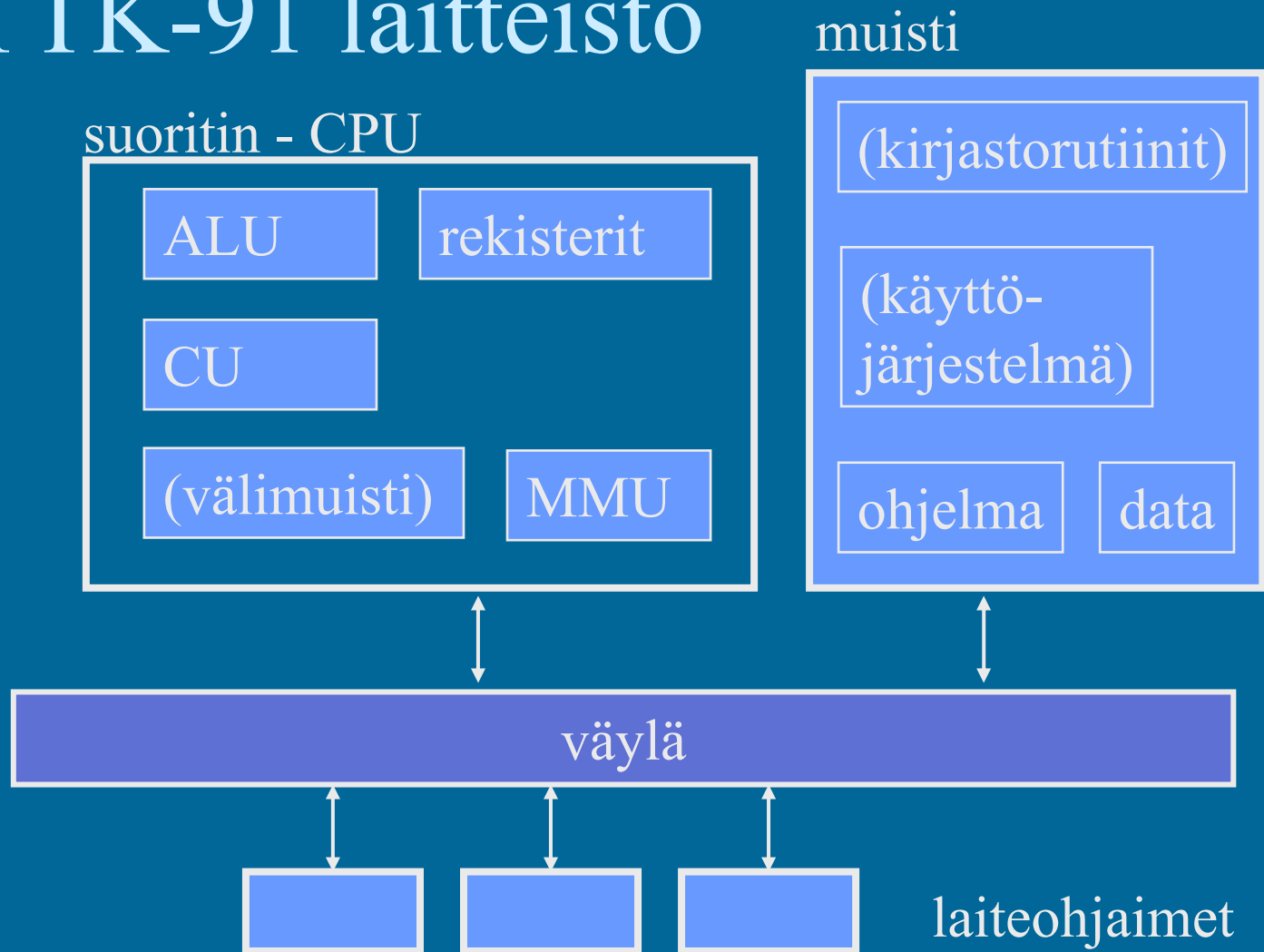
- Järjestelmä kokonaisuudessaan, nopeuserot
- Esimerkkikone ja sen käyttö
- Konekielinen ohjelmointi
- Suoritin, rekisterit, väylät, muisti
 - konekäskyjen suoritussykli, keskeytykset
- Aktivointitietuepino, aliohjelmien toteutus
- Tiedon esitysmuodot (ohjelma vs. laitteisto)
- Prosessi ja sen toteutus (PCB)
- I/O laitteet
 - laiteajurit, laitekeskeytykset, levymuisti
- Ohjelmien suoritus järjestelmässä
 - käänös, linkitys, lataus, tulkinta, emulointi, simulointi
- *Esimerkkejä keskeisistä asioista seuraavilla kalvoilla*

Nopeuserot: Teemun juustokakku

Rekisterien, välimuistin, muistin, levymuistin ja magneettinauhan nopeudet suhteutettuna juuston haku-aikaan juustokakkuja tehdessä?



Esimerkkikone: TTK-91 laitteisto



Konekielinen ohjelmointi

```
for (int i=20; i < 50; ++i)  
    T[i] = 0;
```

I DC 0

...

```
LOAD R1, =20  
STORE R1, I
```

Loop

```
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
```

Käskyjen nouto- ja suoritusyksi



Suorittimen tilat

user

kernel

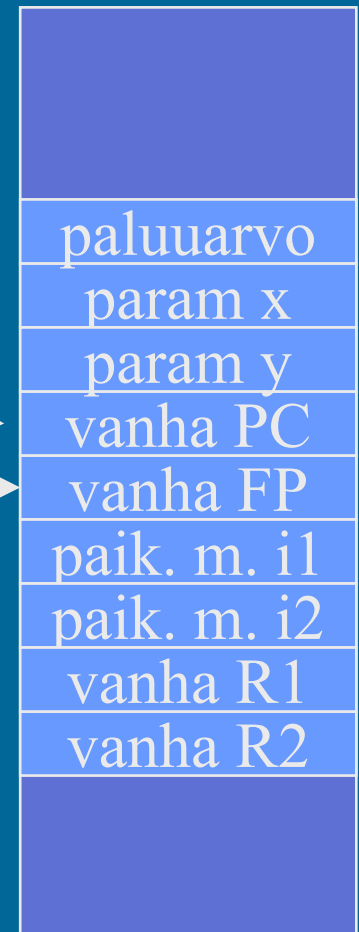
- Käyttäjätila (user mode, normal mode)
 - voi käyttää vain tavallisia käskyjä
 - voi viitata vain käyttäjän omaan muistiavaruuteen (MMU valvoo)
- Etuoikeutettu tila tai (KJ:n) ytimen tila (kernel mode, privileged mode)
 - voi käyttää kaikkia konekäskyjä, myös etuoikeutettuja (esim. `clear_cache`, `iret`)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

Aktivointitietue

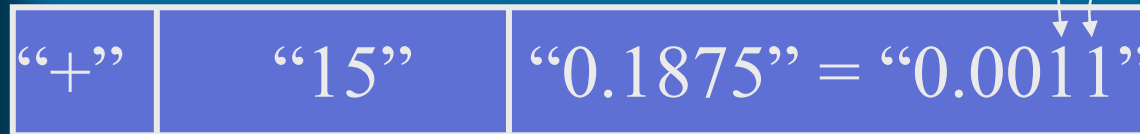
(activation record,
activation frame)

```
int funcA (int x,y);
```

- 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



Tiedon esitysmuodot



sign exponent mantissa or significand

$$\begin{array}{r} 1/8 = 0.1250 \\ 1/16 = 0.0625 \\ \hline 0.1875 \end{array}$$

- 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)

mantissa eksponentti

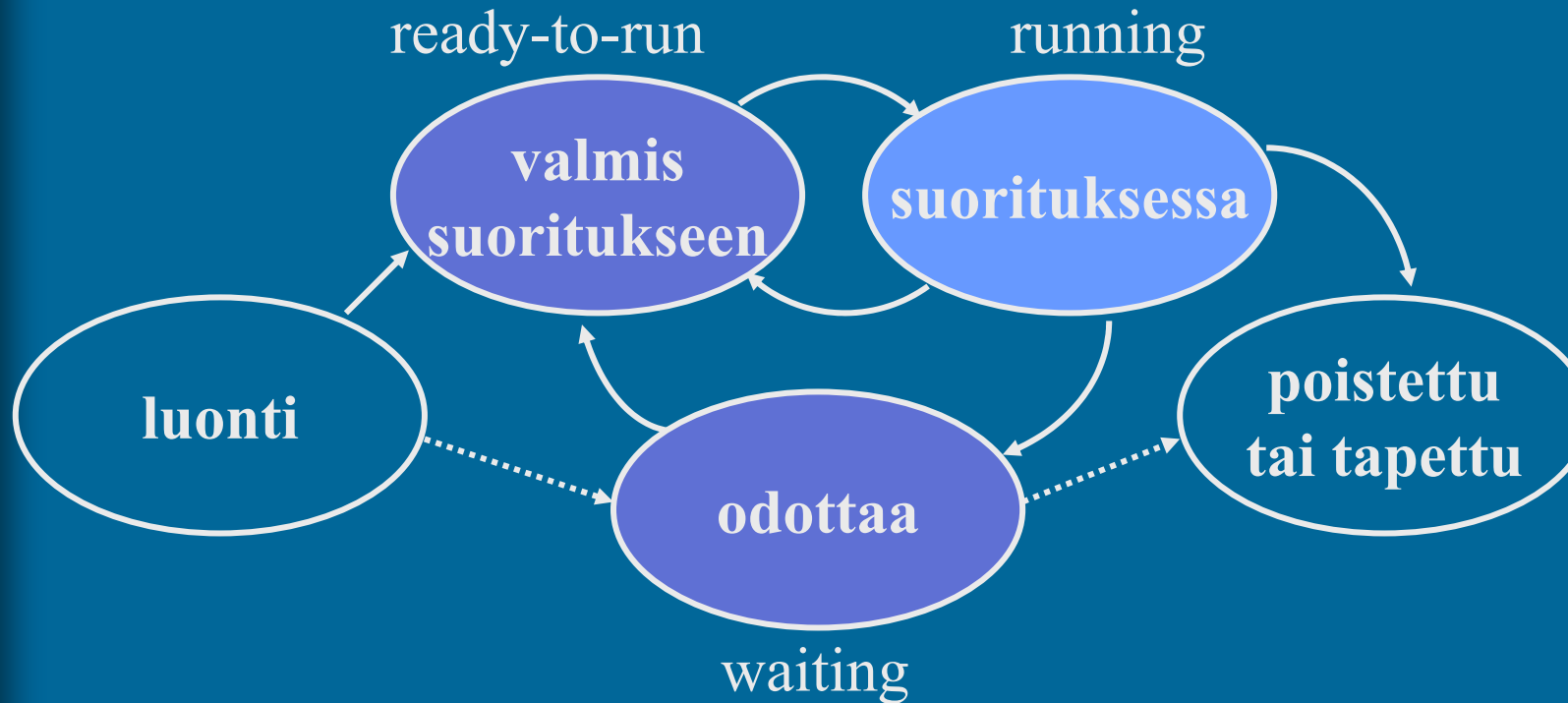
0.0011 “15”

1.1000 “12”

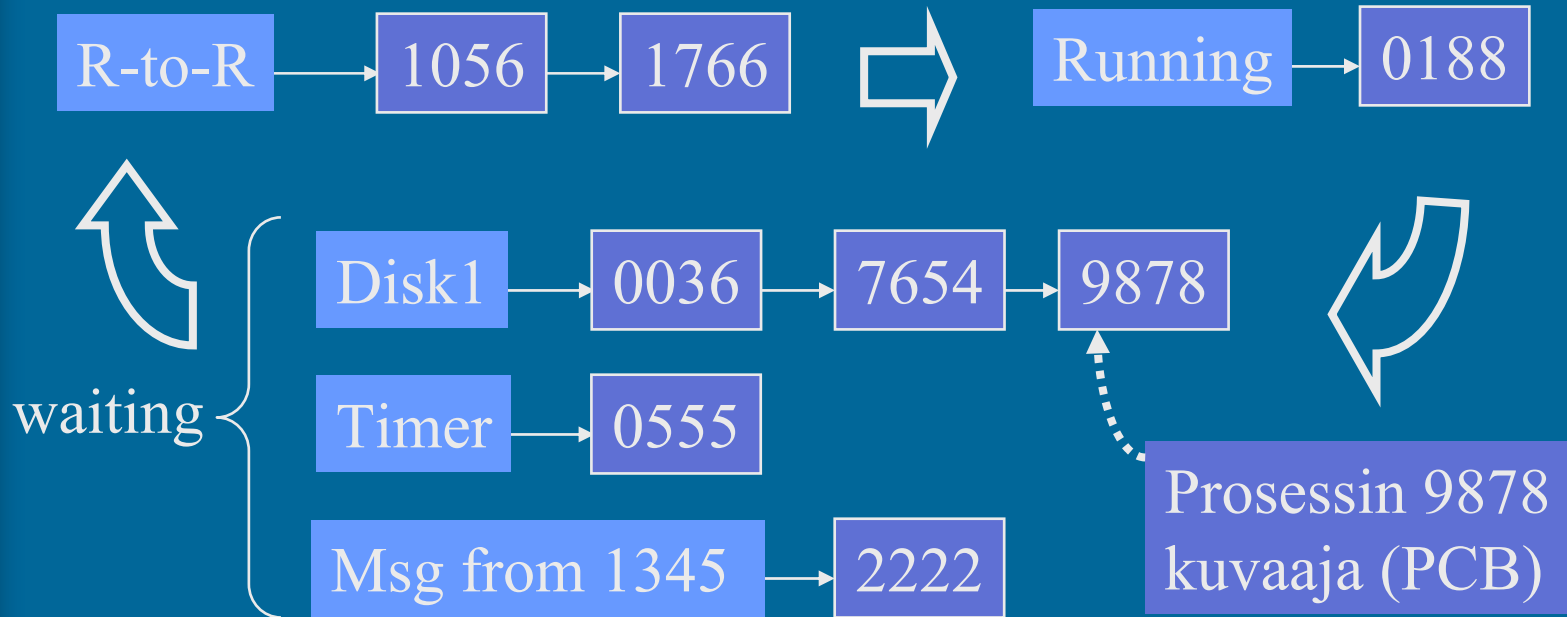
1000 “12”

24 bitin mantissa!

Prosessin tilat ja elinkaari



Prosessit jonoissa ja PCB

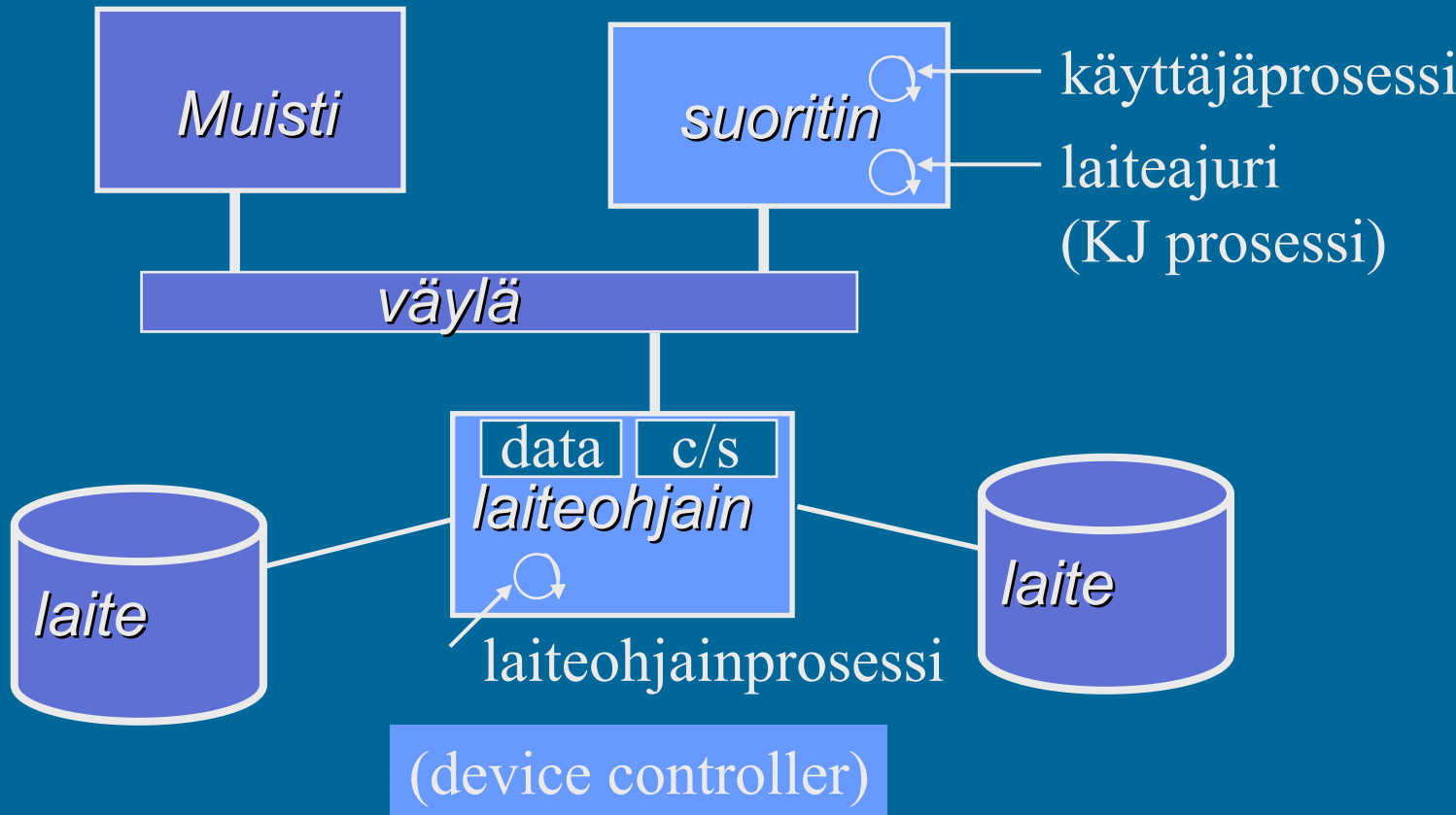


Vuoronanto:

valitse seuraava prosessi Ready-to-Run -jonosta ja
siirrä se suoritukseen CPU:lle

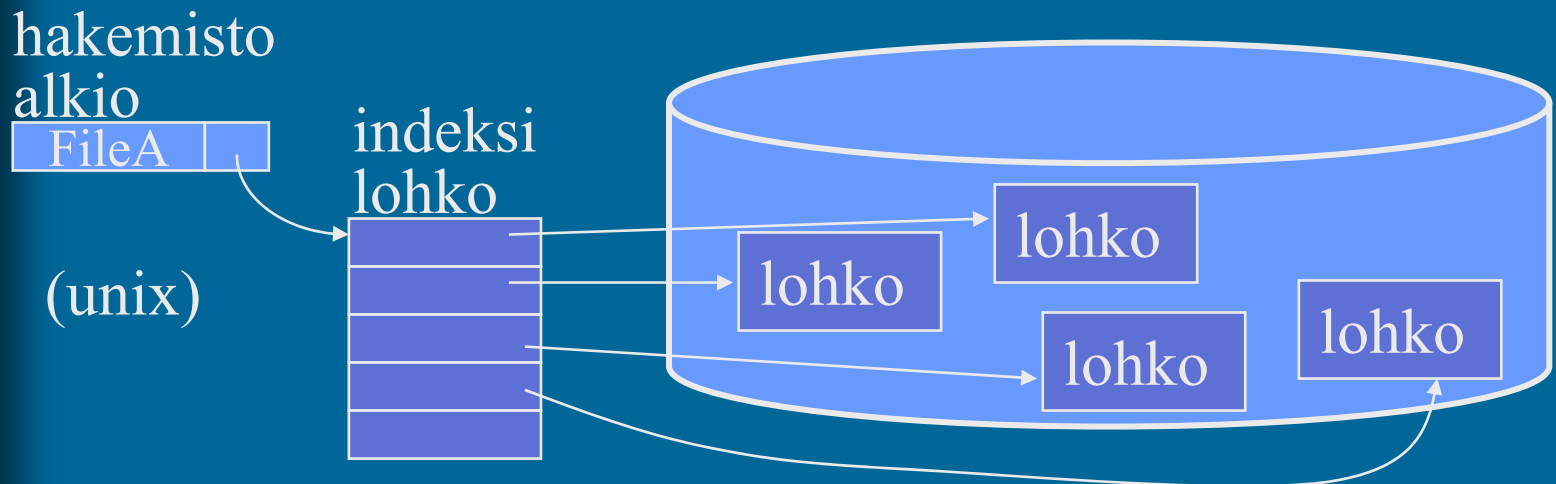
(kopioi tämän prosessin suorittimen tila suorittimelle)

Laiteohjain ja laiteajuri



Levyjen käyttö

- Tiedosto koostuu useista lohkoista
 - lohko per sektori
- Levyn hakemistossa on tieto kunkin tiedoston käyttämistä lohkoista
 - luetaan lohkot annetussa järjestyksessä



Lausekielestä suoritukseen

Käännös
lausekie-
lestä

Linkitys
muiden
ja kirjasto-
moduulien
kanssa

Lataus
muistiin
prosessia
varten

Käännösyksikkö

myprog.c

Lausekielinen ohjelma tai moduuli
osoitteet: symbolit

Objektimoduuli

myprog.obj

Käännetty ohjelma (konekielellä)
osoitteet: lineaariset (per moduuli)

Ajomoduuli

myprog.exe

osoitteet: lineaariset,
osa puutteellisia (?)

Prosessi

Suorituskelpoinen ohjelma
osoitteet: lineaariset (virt. avar.?)

prog.c

prog.

math.l

prog

Tulkinta ja emulointi

`k = i+j;` Java ohjelma

Java
käännös

`iload i`
`iload j`
`iadd`
`istore k`

Java
byte-koodi

Java virtuaalikone

data

data

load

Java
tulkki

Pentium II
suoritin

(• esimerkiksi)

`iload i`
`iload j`
`iadd`
`istore k`

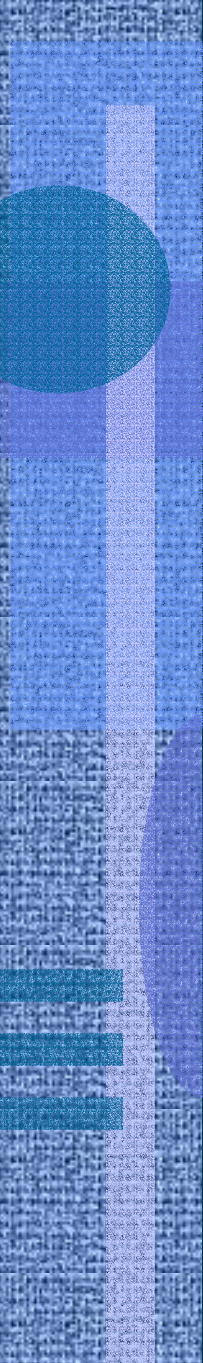
Java
suoritin

natiiviympäristö

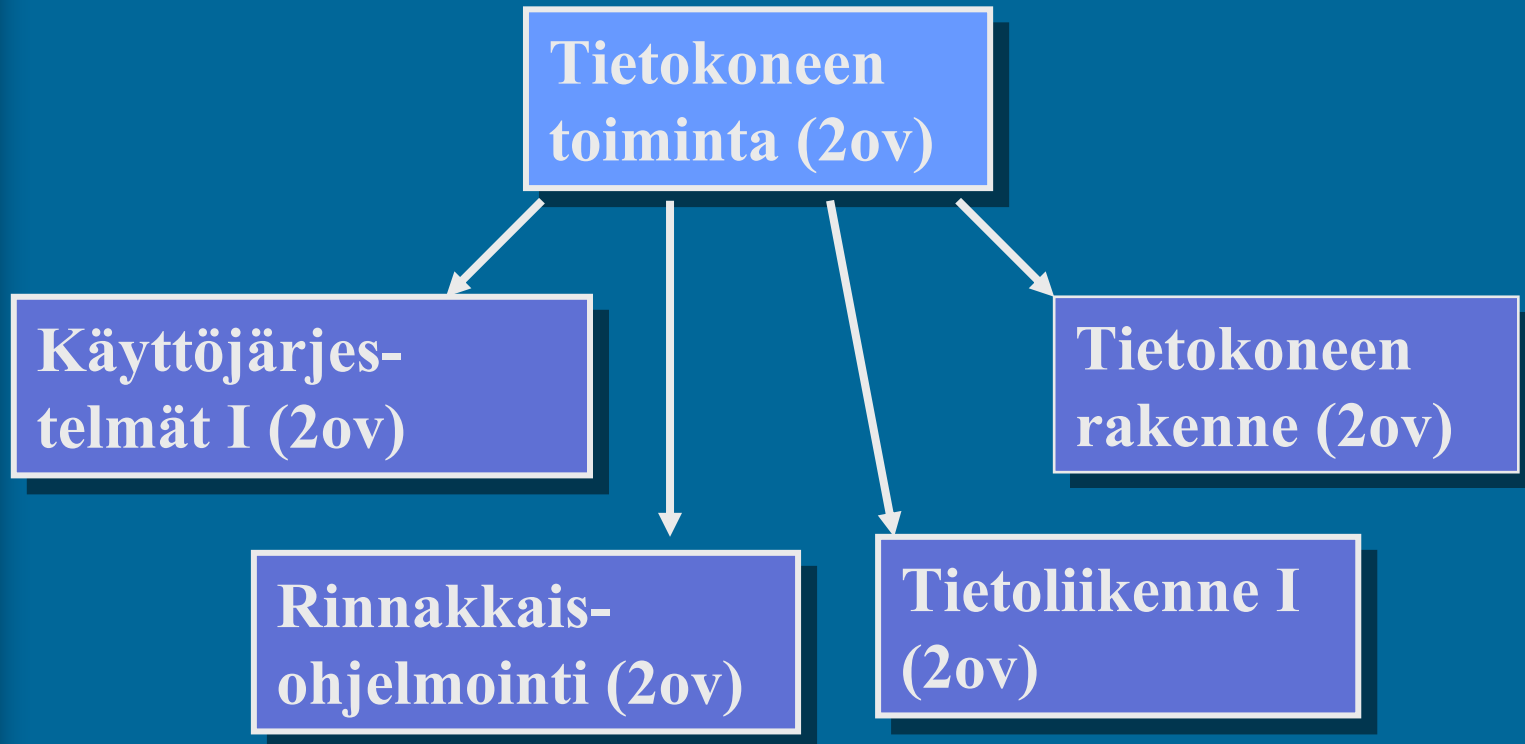
(JIT)
käännös

dl lataus-
moduuli

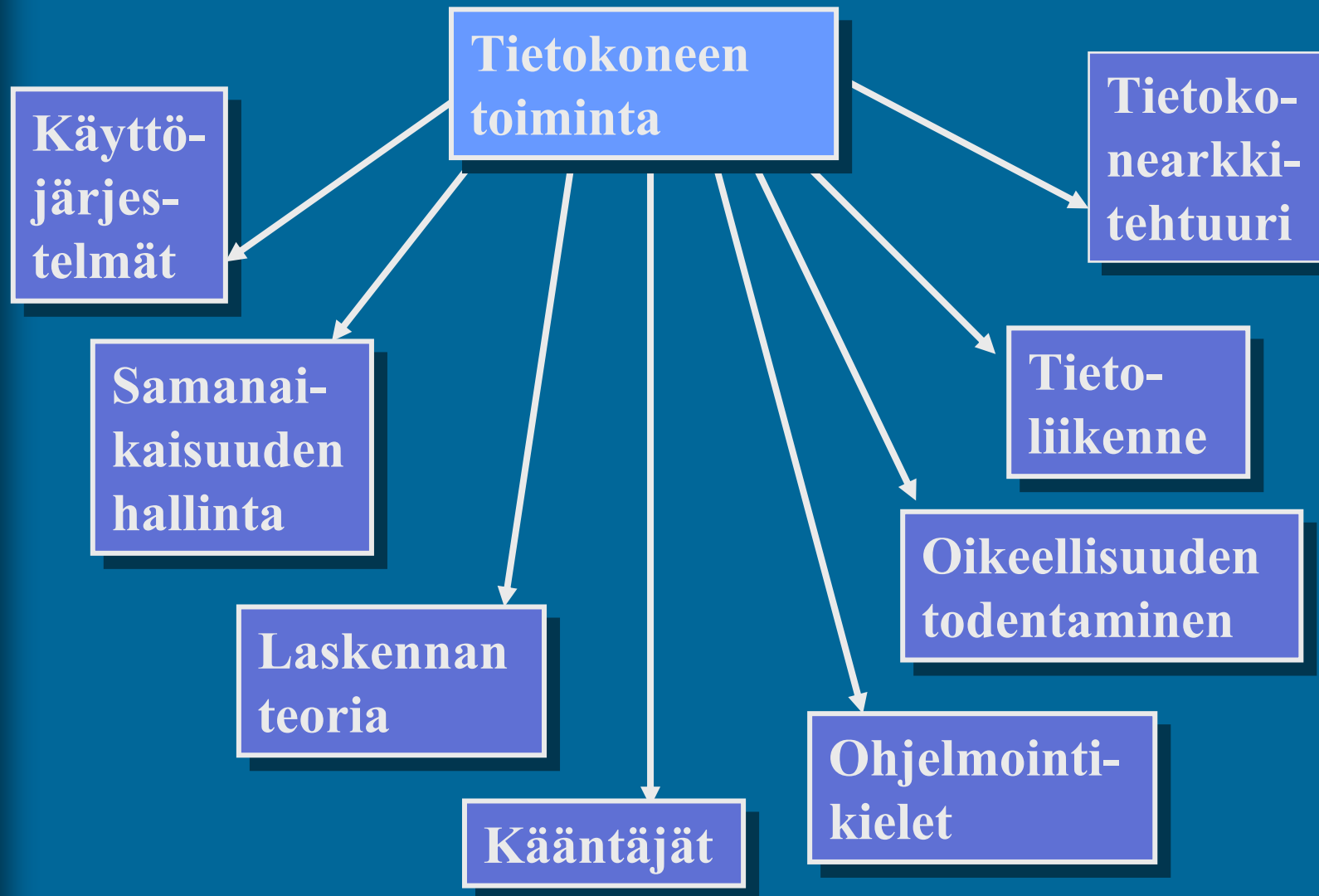
Pentium II
suoritin



Kurssien välisiä suhteita



Asioiden välisiä suhteita (8)



Tietokoneen rakenne, 2 ov

- Yksi taso alaspäin TITOsta
- Sopiva 2. vuoden opiskelijalle
- Useissa yliopistoissa yhdistetty TITOon
- ”Miten kellopulssi saa suorittimen suorittamaan konekäskyjä?”
- ”Miten suorittimen aritmetiikka on toteutettu?”
- Usea käsky on todellisuudessa suorituksessa samanaikaisesti
 - Miten tämä toteutetaan, mitä ongelmia siitä seuraa ja miten noita ongelmia ratkotaan?
- Jatkoa syventävällä tasolla
 - Tietokonearkkitehtuurit, 4 ov

TiKRa

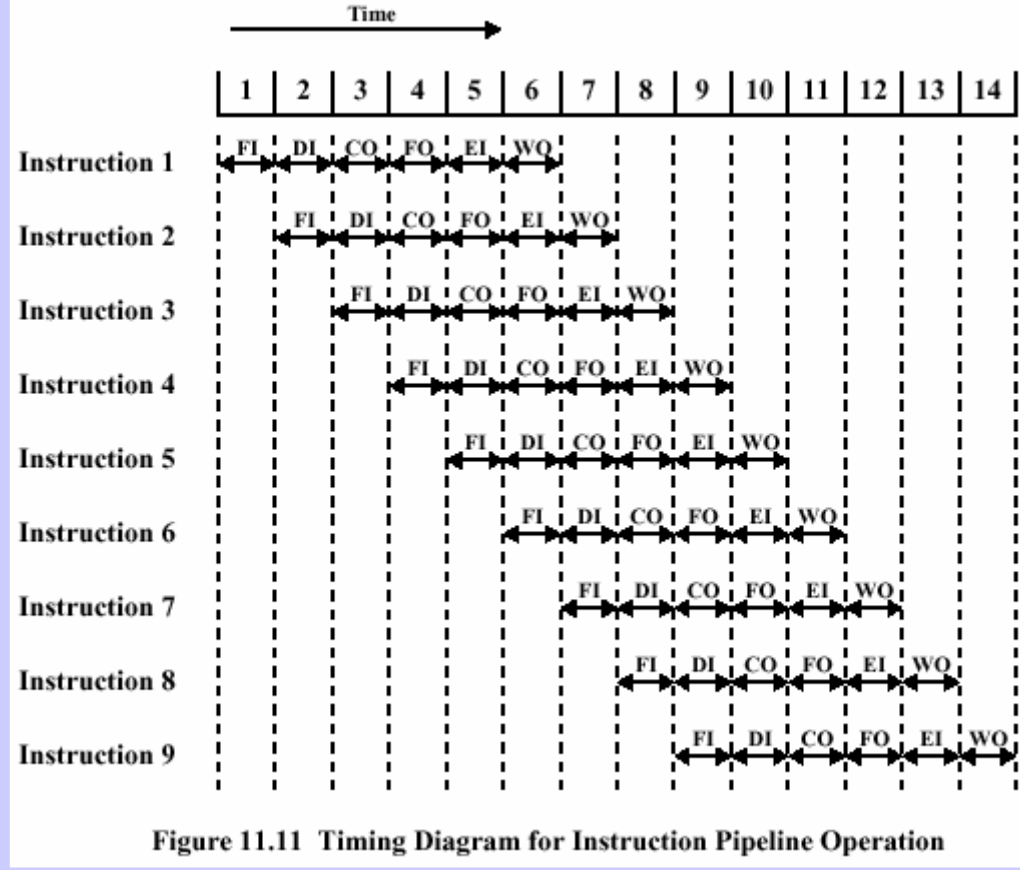


Figure 11.11 Timing Diagram for Instruction Pipeline Operation

[Stal99]

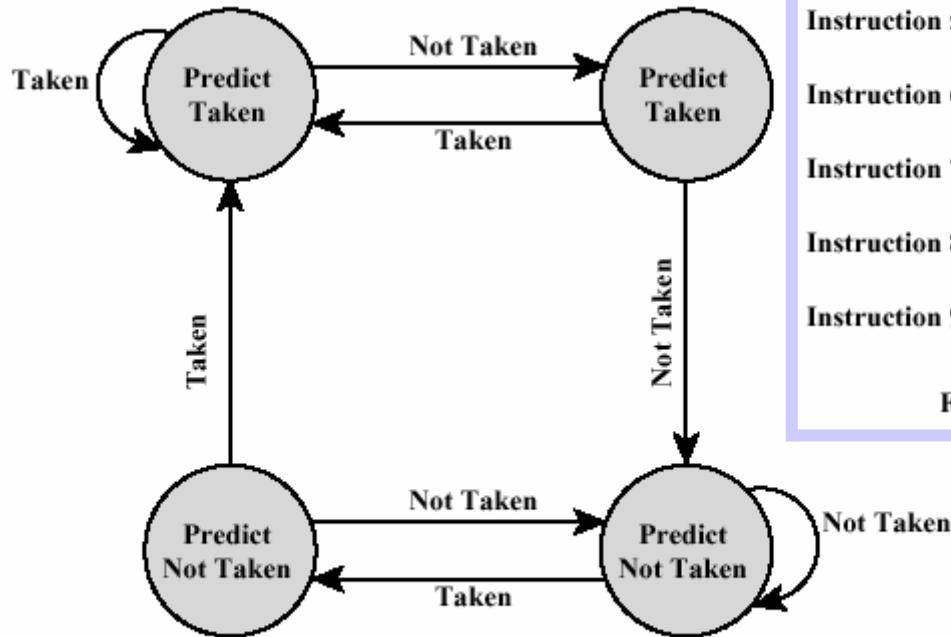


Figure 11.16 Branch Prediction State Diagram

Käyttöjärjestelmät I, 2 ov

- Sopiva 2. vuoden opiskelijalle
- Käyttöjärjestelmän rooli yhden prosessin valvojana
- Täsmentää ja jatkaa TITOn käyttöjärjestelmien piirteiden esittelyä
- Samanaikaiset prosessit resurssien käyttäjinä
- Systemin resurssien jakelu
- Prosessien vuoronanto (skedulointi)
- Jatkoa perustasolla ja syventävällä tasolla
 - Käyttöjärjestelmät II, 2 ov
 - Käyttöjärjestelmämetodiikka, 3 ov

KJ ...

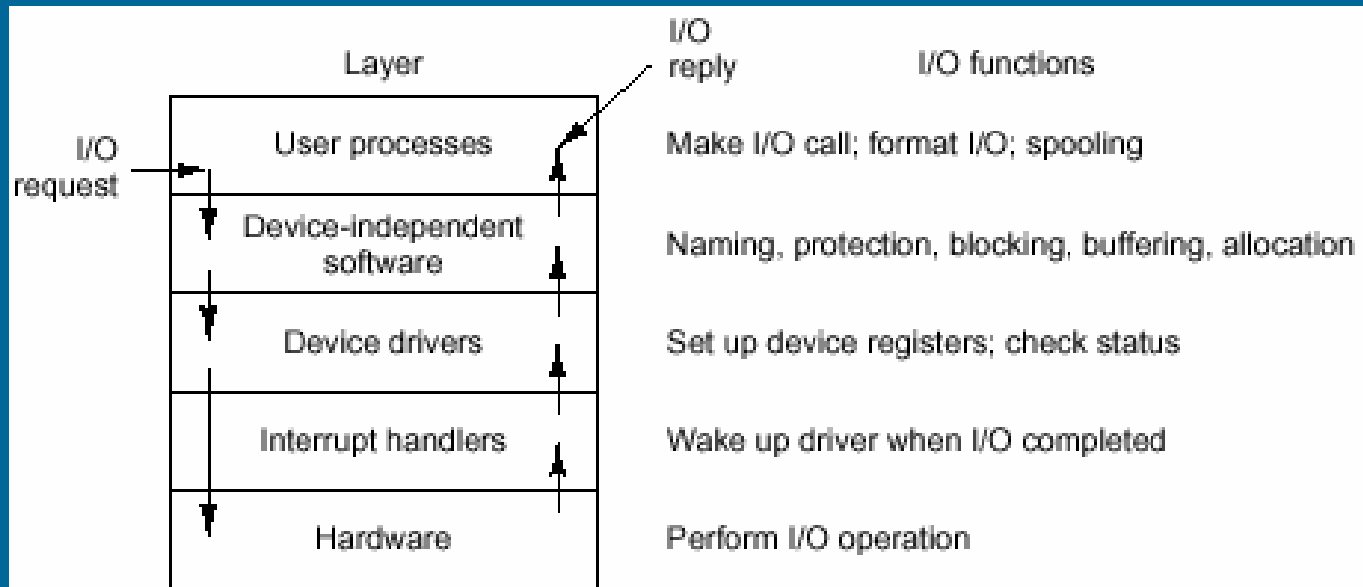


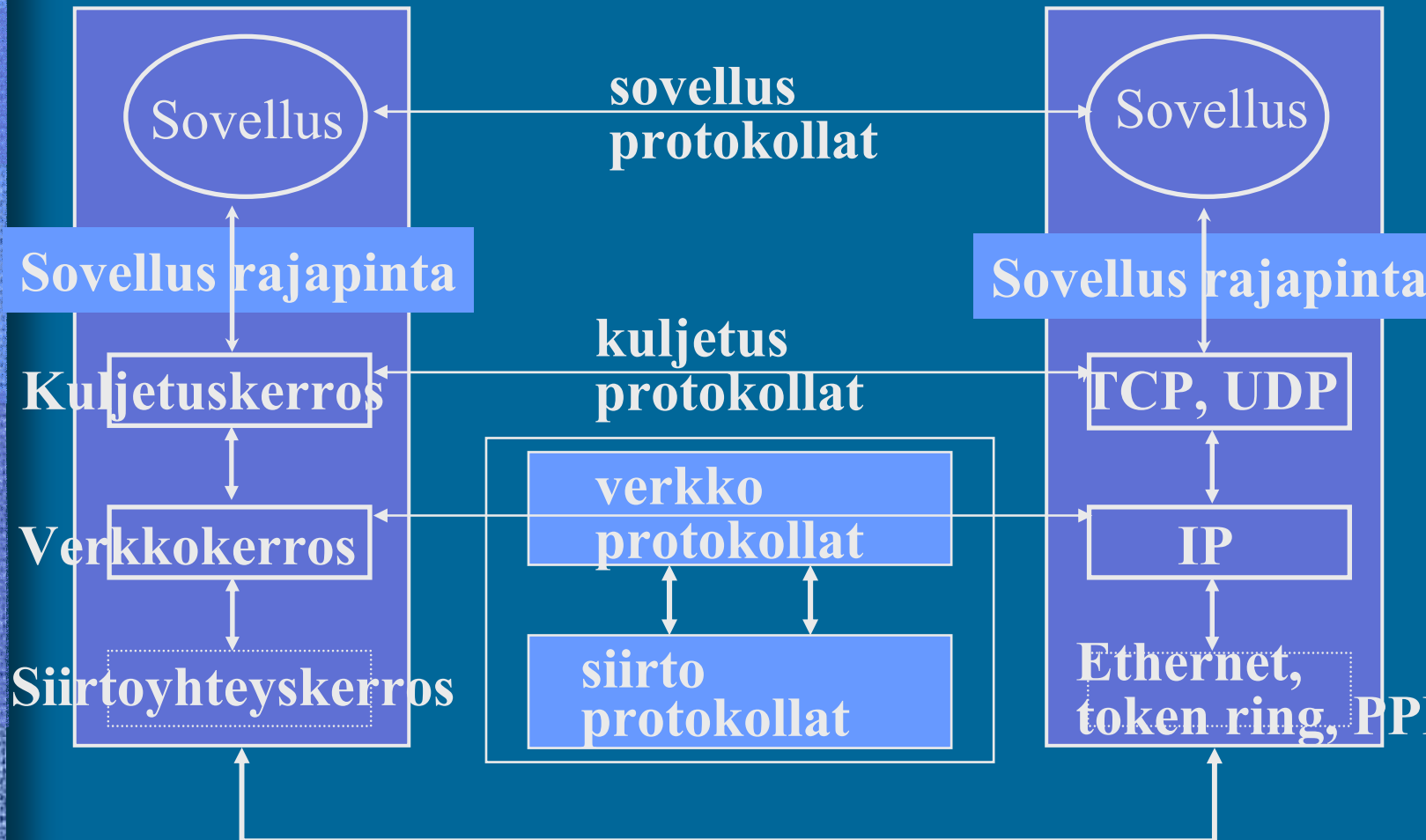
Figure 3-6. Layers of the I/O system and the main functions of each layer.

Tietoliikenne I, 2 ov

- Sopiva 2. vuoden opiskelijalle
- Tietokoneverkkojen peruspalvelut käyttäjälle ja sovelluksille
- Verkkojen tiedonsiirron perusvälineistö
- Verkkoarkkitehtuurin kerrosrakenne ja kunkin tason palvelut
- Jatkoa perustasolla ja syventävällä tasolla
 - Tietoliikenne II, 2 ov
 - Tietoliikennejärjestelmät, 3 ov

Tietoliikenne ...

TCP/IP -kerrosmalli



Rinnakkaisohjelmointi, 2 ov

- Sopiva: 2. vuoden opiskelijoille
- Samanaikaisuuden aiheuttamat ongelmat
 - järjestelmä kaatuu ... miksi niin kävi?
- Samanaikaisuuden aiheuttamat vaatimukset systeemille
- Prosessien synkronointi eri tapauksissa
 - odottamalla vai prosessia vaihtamalla?
- Prosessien kommunikointi eri tavoin
 - yhteinen muistialue? viestit?
 - verkon ylitse?
- Jatkoa syventävällä tasolla
 - Hajautetut järjestelmät, 3 ov

RIO: synkronointiongelman ratkaisu Test-and-Set -käskyllä

tämän
käskyn
osoite

- TAS R_i, L
(ttk-91:n
laajennus)

$R_i := mem[L]$

if $R_i == 1$ then

{ $R_i := 0, mem[L] := R_i, jump *+2$ }

- Kriittinen
vaihe

LOOP: TAS R_1, L # $L: 1$ (vapaa) 0 (varattu)
 JUMP LOOP # odota kunnes lukko auki

...

kriittinen vaihe: yksi prosessi kerrallaan

...

LOAD $R_1, =1$ # avaa lukko L

STORE R_1, L

- Toimiiko, jos tulee keskeytys pahassa kohtaa?
 - Mikä on “paha kohta”?

Ohjelmointikielten periaatteet, 4 ov

- Lähtötiedot: OLPM, TiKi, ohjelmointilabrat
- Sopiva: 3. vuoden opiskelijat
- Ohjelmointikielten määrittelyn välineistö
- Erilaiset ohjelmointiparadigmat esimerkkikielten avulla
 - proseduraaliset kielet C, Pascal
 - oliokielet Smalltalk
 - funktionaaliset kielet Scheme, ML
 - logiikkaohjelmointikielet Prolog
- Jatkoa syvemmällä tasolla:
 - ??

Ohjelmointikielten kääntäjät, 5 ov

- Lähtötiedot: OLPM, ohjelmointilabrat
- Sopiva: 3. vuoden opiskelijat
- Ohjelmointikielten kääntäjien tyypit
 - rekursiivisesti etenevä jäsentelijä
- Kääntäjän osat
 - selaaja
 - jäsentelijä
 - semantiikan analyysi
 - koodin generointi
- Jatkoa syvemmällä tasolla:
 - ??

lex

yacc

Spesifioinnin ja verifioinnin perusteet, 2 ov

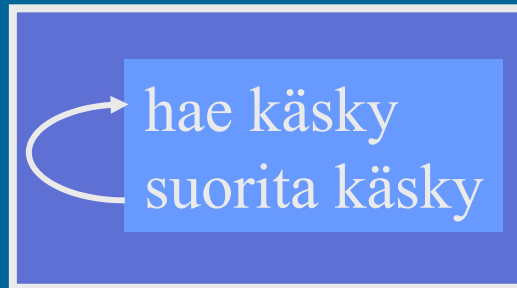
- Lähtötiedot: hajautuksen ja samanaikaisuuden problematiikka
- Sopiva: 3. vuoden opiskelijalle
- Mallinnetaan prosesseja siirtymäsystemeillä
 - askel: konekäsky? metodi? tapahtuma? ohjelma?
- Automaattisen verifioinnin periaatteet
- Yksinkertaisia protokollien verifiointi
- Jatkoa syventävällä tasolla
 - Ohjelmien semantiikka, 3 ov
 - Automaattinen verifiointi, 3 ov

Ohjelmoinnin ja laskennan perusmallit (OLPM), 2 ov

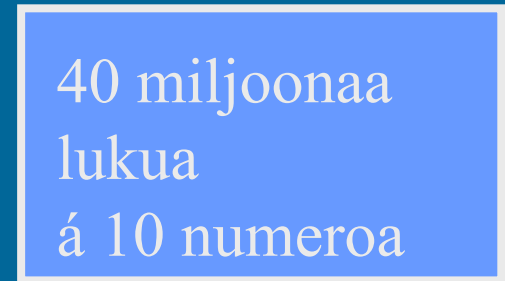
- Lähtötiedot: matematiikkaa
 - appro tai disk. mat., ... + tira?
- Sopiva: 1. vuoden (2. vuoden?) opiskelijalle, joka on opiskellut jo matematiikkaa
- Laskennalliset ongelmat, niiden luokittelu
- Äärelliset automaatit ja säännölliset kielet
- Kieliopit
- Turingin kone
- Jatkoa syventävällä tasolla
 - Laskennan teoria, 3 ov

Laskennan teorian perusta (1)

suoritin - CPU

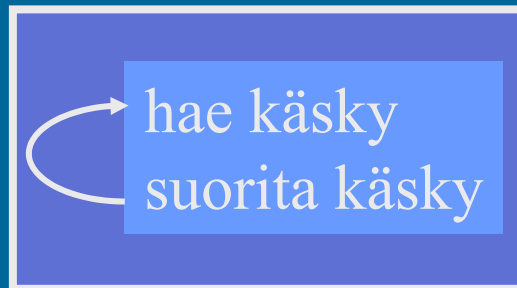


muisti

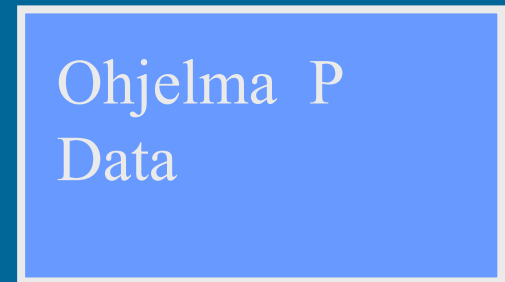


väylä

suoritin - CPU

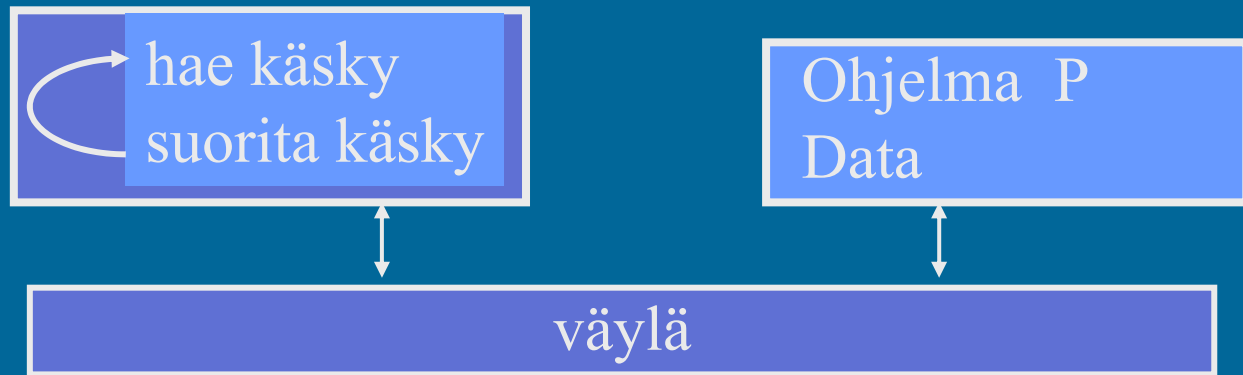


muisti



väylä

Laskennan teoriaa ... (4)



Muistin sisältö
ennen P:n suoritusta:

X = hyvin iso kokonaisluku
(200M numeroa?)

Muistin sisältö P:n
suorituksen jälkeen:

Y = joku toinen hyvin iso luku

P on kokonaislukuarvoinen funktio $P:\mathbb{U} \rightarrow \mathbb{U}$

Ohjelman P esitysmuoto muistissa: Iso kokonaisluku, $P \in \mathbb{U}$

Laskennan teoriaa ... (4)

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



- Todistettuja lauseita ohjelmien ominaisuuksista
 - pätevät kaikille tietokoneille
 - nyt ja tulevaisuudessa

Laskennan teoriasta ja algoritmianalyysistä todistettuja lauseita ⁽³⁾

- Valitaanpa mikä tahansa aikaraja, 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 12
ja
koko kurssin
loppu
--



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

21.1.2003



Teemu Kerola, Copyright 2003