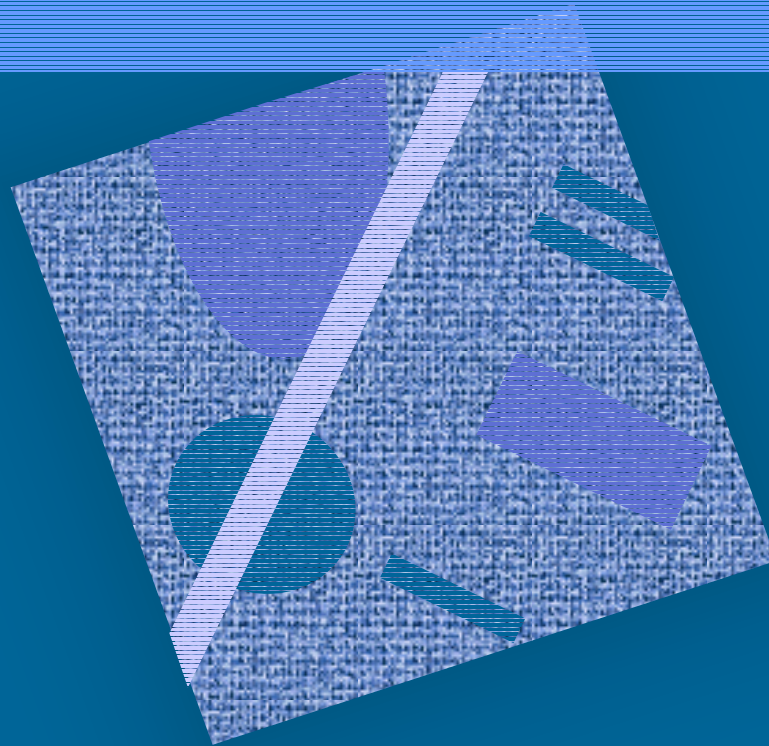


# Luento 5

## Suoritin ja väylä



Suorittimen rakenne

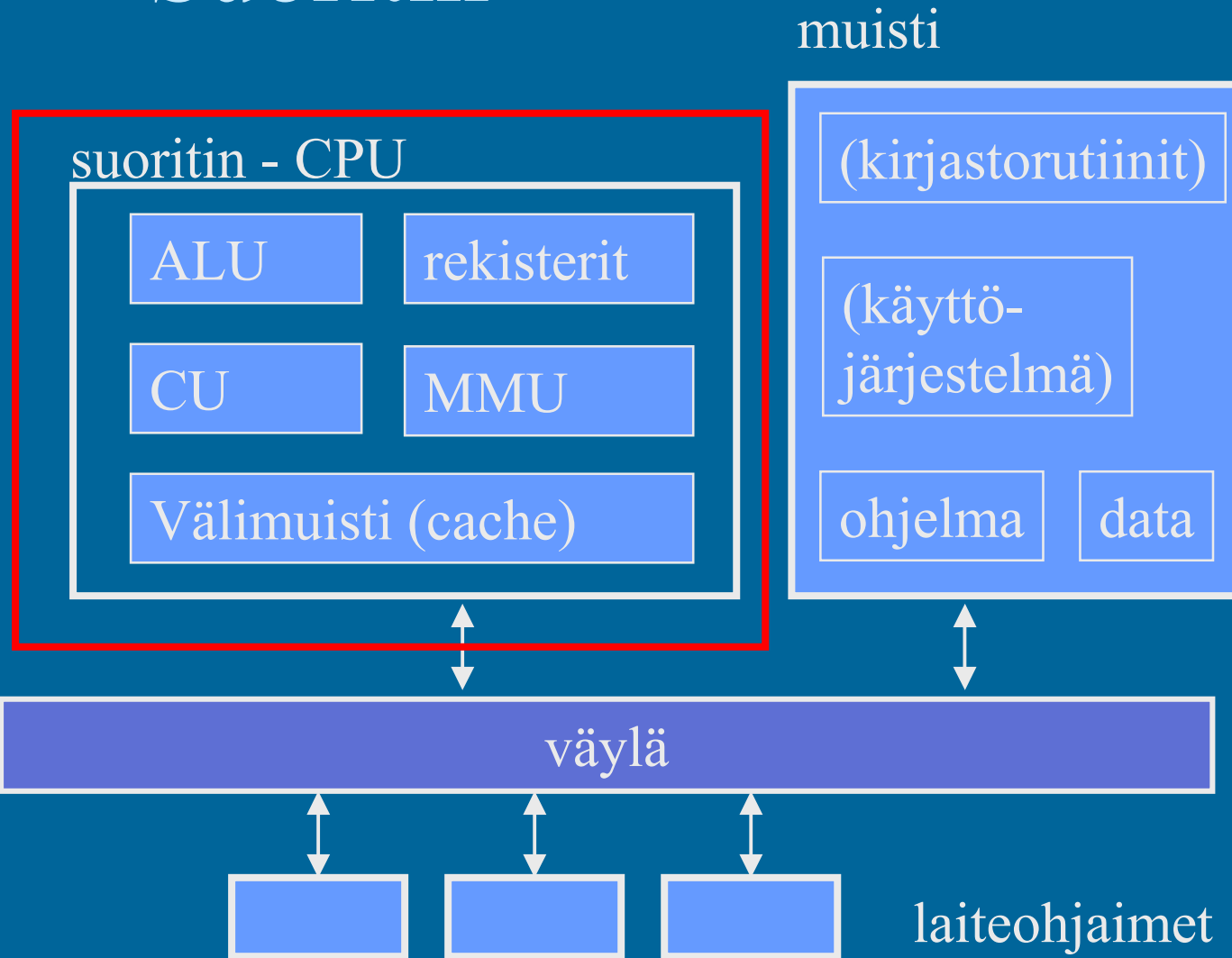
Väylän rakenne

Käskyjen suoritusssykli

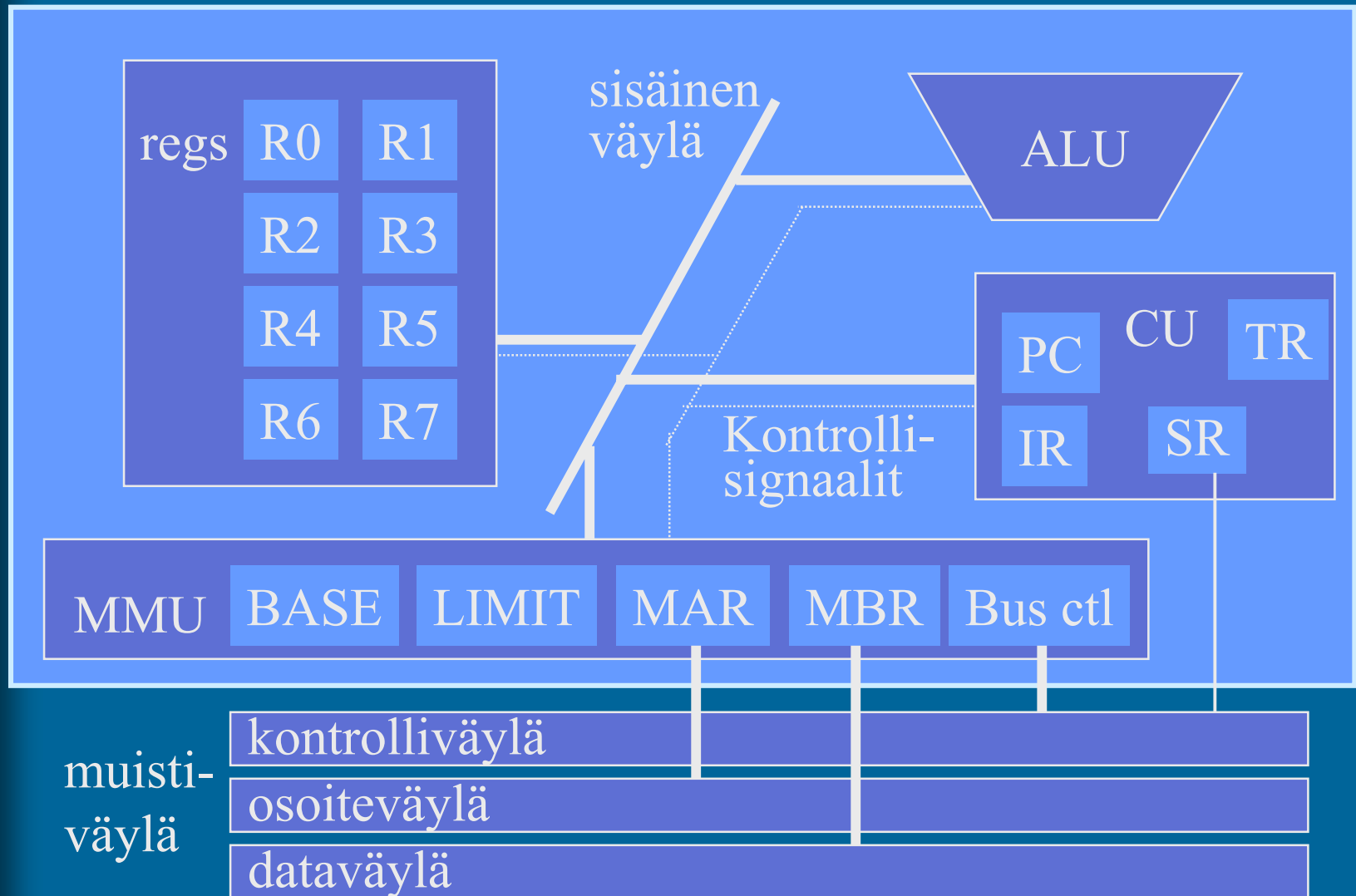
Poikkeukset ja  
keskeytykset

KOKSI:n rakenne

# Suoritin

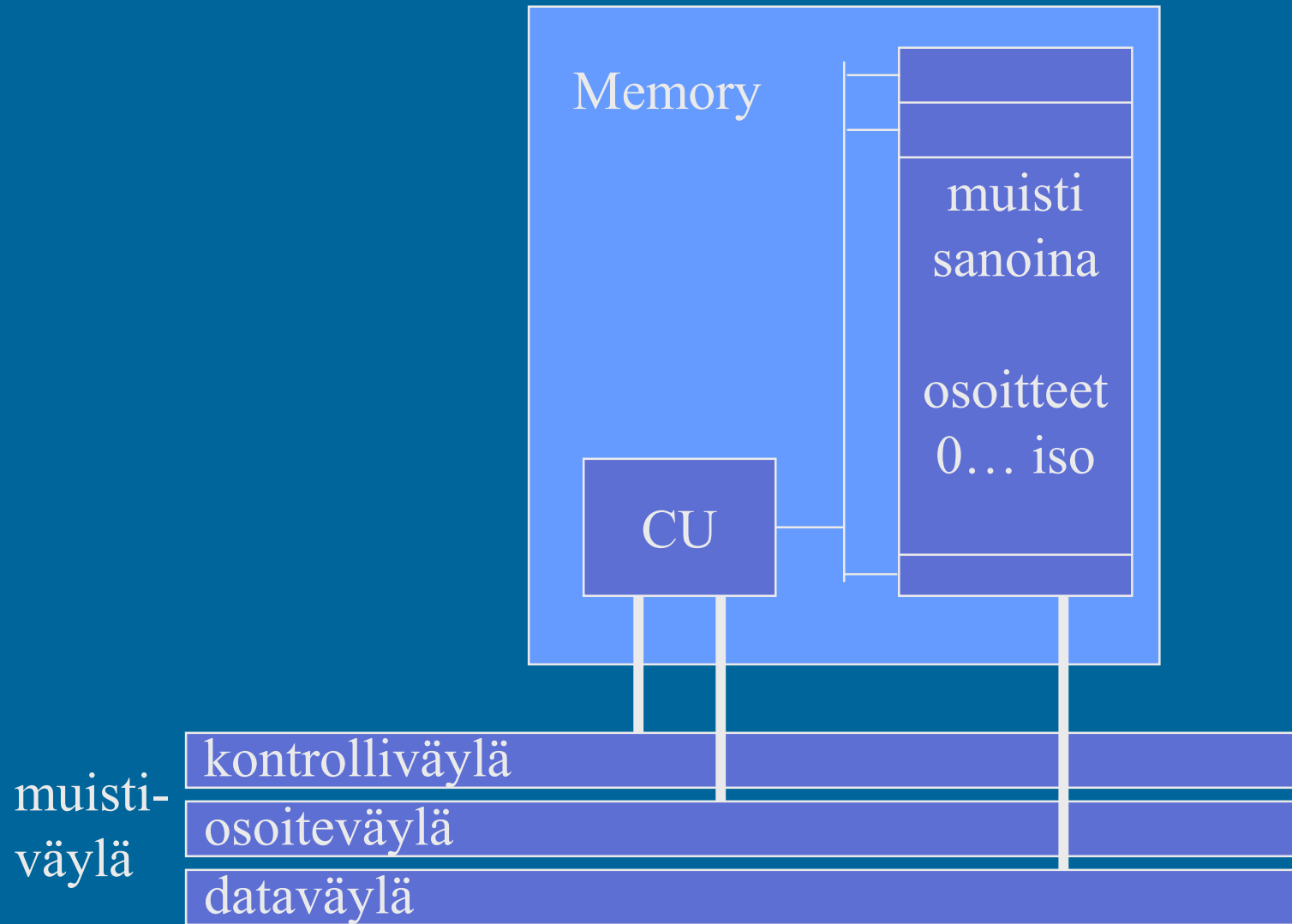


# TTK-91 suorittimen rakenne (1)



ks. Edsac, lastut 286, 486, PIII, PIII Xeon

# TTK-91 muistin rakenne



# Käskyjen nouto- ja suoritussykli <sup>(5)</sup>

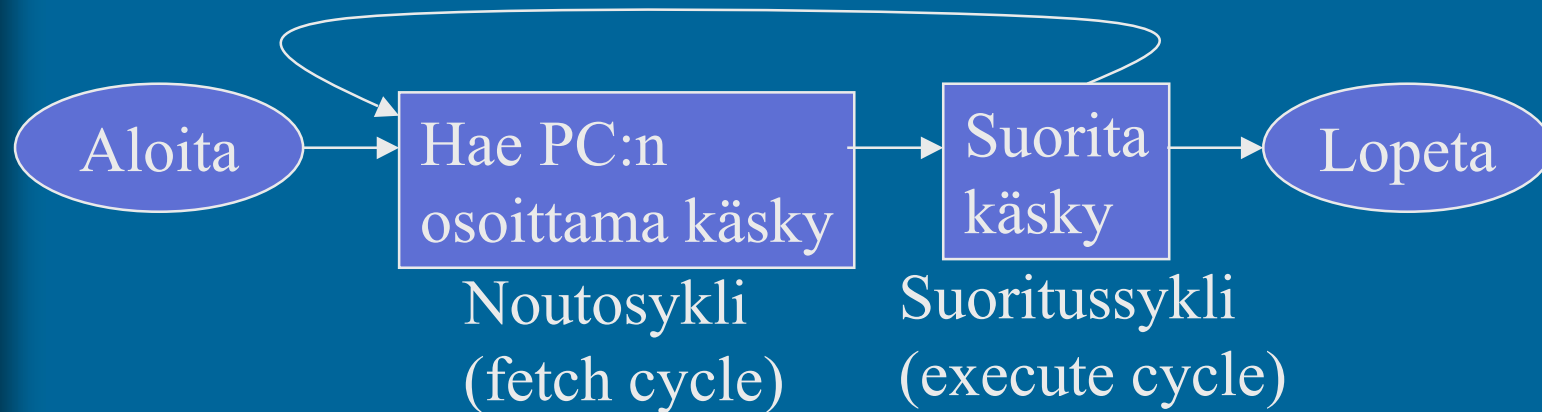
- Hae PC:n osoittama konekäsky muistista
  - lisää samalla PC:n arvoa yhdellä
- Suorita konekäsky
  - jos (ehdollinen) hyppykäsky, niin PC:n arvo voi vielä muuttua

Suoritin ei näe mitään suurempia kokonaisuuksia kuin konekäskyjä!

Suoritin ei tiedä mitään ohjelmista!



# Nouto- ja suoritussykli



- Käskyn suoritus voi muuttaa systeemin tilaa
  - sisäiset ja ulkoiset rekisterit
  - muisti
  - laitteet

# TTK-91 konekäskyn rakenne

- Käskyn esitys bittitasolla on aina:



Rj = käskyn ensimmäinen operandi

Ri = indeksirekisteri (R0 ≡ 0)

M = muistinoutojen määrä toiseen operandiin  
(ennen mahdollista muistiin talletusta)

muistiosoite tai  
(pienekkö) vakio

(addressing  
mode)

00 eli 0 kpl, välitön osoitus (STORE: suora osoitus)

01 eli 1 kpl, suora osoitus (STORE: epäsuora osoit.)

10 eli 2 kpl, epäsuora osoitus (STORE: epäkelpo arvo)

( 11 eli 3 kpl, epäkelpo arvo → poikkeustilanne )

# Nouto- ja suoritussykli tarkemmin <sup>(5)</sup>

- Noutovaihe
  - muistista MBR:n kautta IR:ään
  - Lisää 1 PC:hen
- Käselyn purku ja muistiosoitteen (EA) lasku
  - OPER, Rj, M, Ri, ADDR
  - $TR \leftarrow (Ri) + ADDR$  (pelkkä ADDR, jos  $Ri=R0$ )
- Operandin nouto
  - muistista MBR:n kautta TR:ään (0-2 krt ?)
- ALU operaatio
  - tulos rekisteriin R0-R7 tai TR:ään (STORE, PUSH)
- Muistiin talletus
  - muistiin MBR:n kautta

ks. TTK-91  
suorittimen  
rakennekuva

Ei kaikilla käskyillä

Ei kaikilla käskyillä

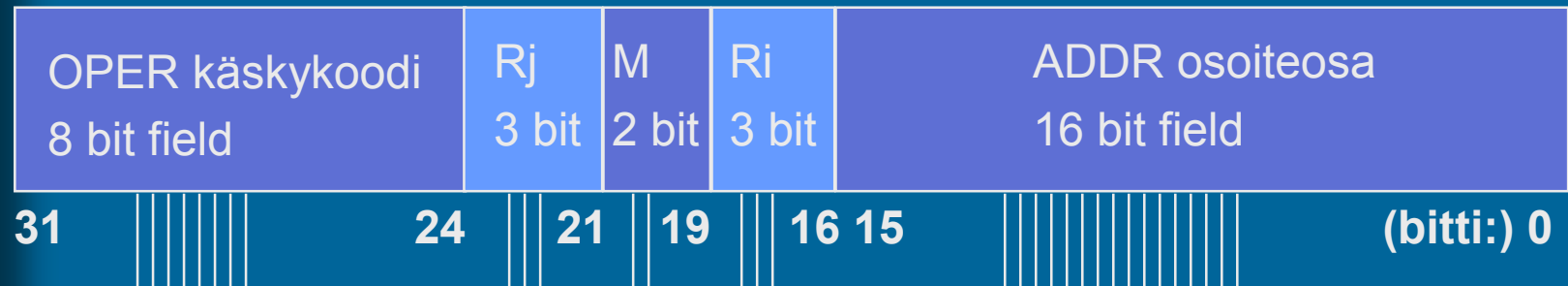


# Käskyn noutovaihe (4)

ks. TTK-91 suorittimen rakennekuva

- Vie PC:n arvo MAR:iin
- Aseta muistin lukusignaali kontrolliväylälle asentoon ”lue”
- Odota, kunnes muistipiiri toimittaa väylän kautta uuden arvon MBR:ään
- Siirrä konekäsky MBR:stä IR:ään

# Käskyn purku ja tehollisen muisti-osoitteen (EA) laskemisvaihe



- Purku automaattisesti langoitettuna IR:stä
- Muistiosoitteen lasku, tulos TR:ään
  - jos  $R_i=0$ , niin  $TR \leftarrow ADDR$
  - muutoin  $TR \leftarrow (R_i)+ADDR$ 
    - ALU suorittaa laskutoimituksen
  - Effective Address (EA) on nyt TR:ssä

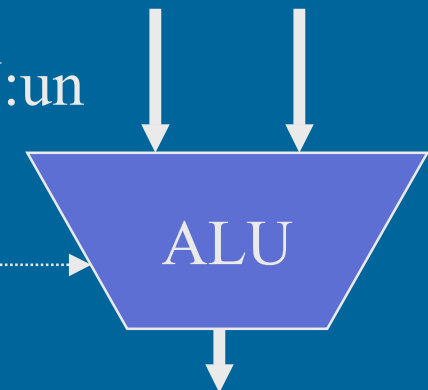
# Operandin luku vaihe <sup>(4)</sup>

ks. TTK-91 suorittimen rakennekuva

- Vie muistiosoite MAR:iin
- Aseta muistin lukusignaali kontrolliväylälle asentoon "lue"
- Odota kunnes muistipiiri toimittaa väylän kautta uuden arvon MBR:ään
- Siirrä sana MBR:stä TR:ään
  - (tai suoraan johonkin laiterekisteriin R0-R7)

# ALU operaatio -vaihe <sup>(10)</sup>

- Lähtötilanne ks. TTK-91 suorittimen rakennekuva
  - käsky haettu ja purettu osiin IR:ssä
  - 1. operandi rekisterissä (R0, ..., R7)
  - 2. operandi TR:ssä
- Käskyn suoritus ALU:ssa
  - vie operandit sisäistä väylää pitkin ALU:un
  - anna ALU:lle sopiva ohjaussignaali
    - add, mul, copyLeft, comp, ...
  - odota, että tulos valmis
  - talleta tulos rekisteriin, MBR:ään, PC:hen ja/tai SR:ään



Tässä tapahtuu tietokoneen tekemä työ,  
kaikki muu on hallintoa

# Tuloksen muistiin kirjoitus -vaihe (5)

ks. TTK-91 suorittimen rakennekuva

- Vie muistiosoite MAR:iin
- Vie kirjoitettava sana MBR:ään
- Aseta kirjoitussignaalit kontrolliväylälle asentoon ”kirjoita muistiin”
- Odota kunnes sana siirretään muistiin väylää pitkin ja väylän kontrollisignaalit kertovat muistiinkirjoittamisen tapahtuneen

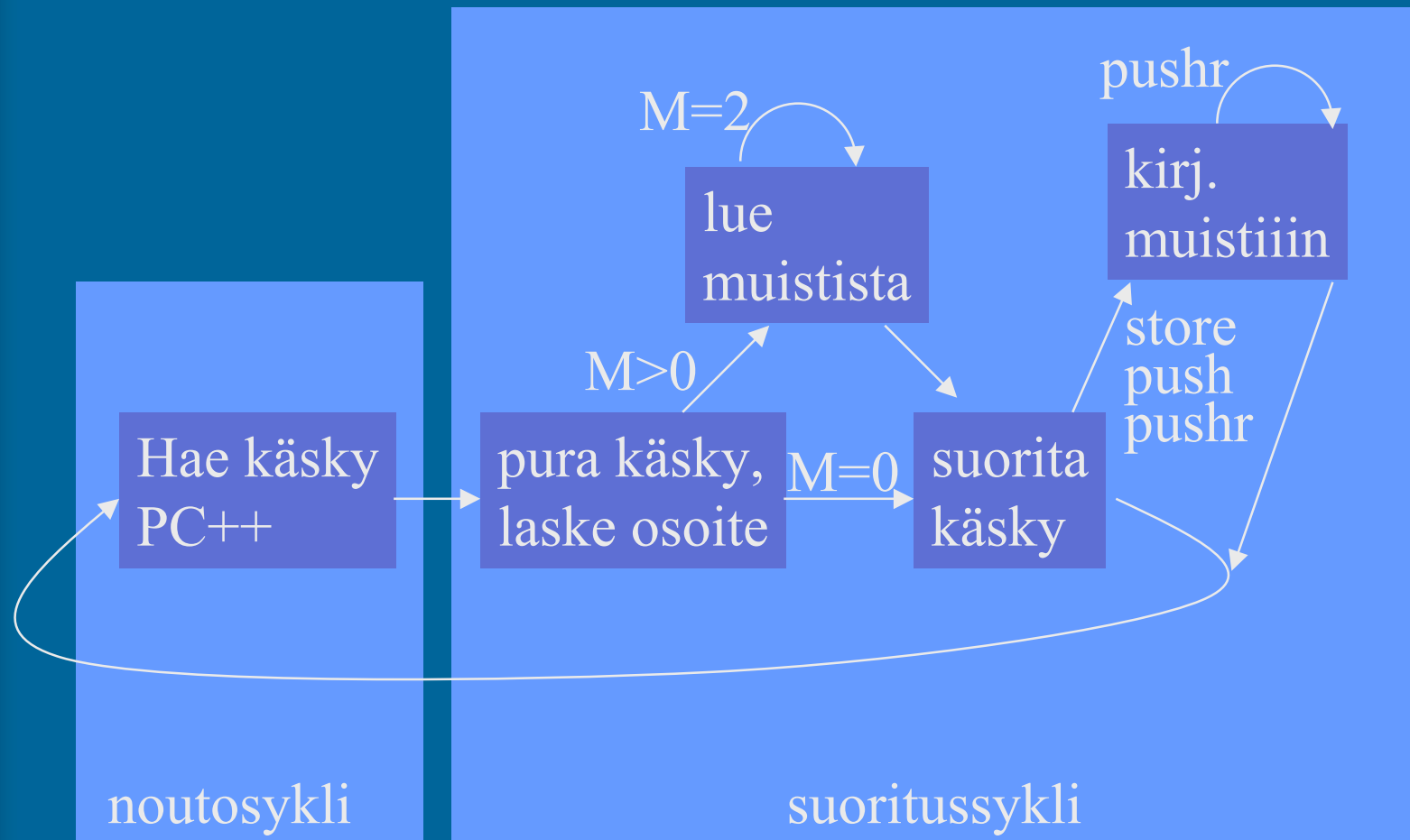
Lisää  
tietoa?



Tietokoneen  
rakenne-  
kurssi



# TTK-91 Nouto- ja suoritussykli vähän tarkemmin <sup>(1)</sup>



pushr, popr erikoistapauksia: aika monimutkaisia

# MMU:n toiminta (2)

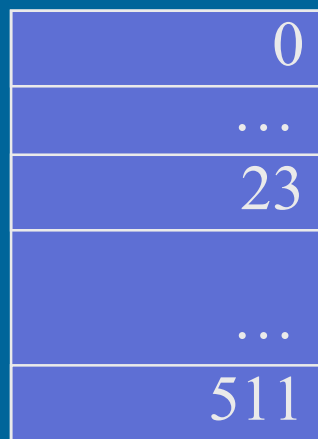
ks. TTK-91 suorittimen rakennekuva

- Ohjelman käyttämät muistiosoitteet (VA) ovat näennäisiä, välillä  $0 \dots \text{LIMIT}-1$ 
  - ne eivät ole samoja osoitteita kuin keskusmuisti käyttää
- MAR:iin menevä arvo VA ei käytetä suoraan, vaan se tarkistetaan ja muokataan ensin
  - Tarkista, onko  $VA \in [0, \text{LIMIT}-1]$ .  
Jos ei ole, niin aseta SR:n bitti M päälle ja lopeta käskyn suoritus
  - Lisää VA:han BASE ja laita tämä arvo (PA) MAR:iin

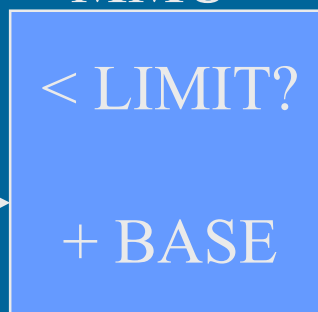
VA = virtual address, PA = physical address = BASE+VA

# TTK-91 virtuaalimuisti

Virtuaalinen osoiteavaruus



MMU

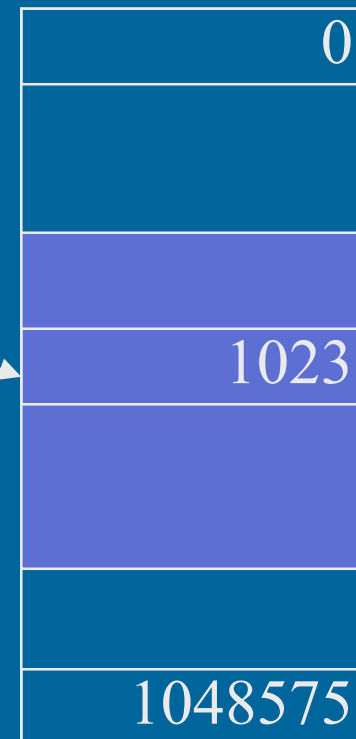


BASE: 1000

LIMIT: 512

ohjelman käyttämät osoitteet

Fyysinen osoiteavaruus



muistipiirien käyttämät osoitteet

# Virtuaalimuistin osoitteenmuunnos menetelmiä (4)

- Kanta- ja rajarekisteriin perustuva
  - base ja limit rekisterit (esim. ttk-91, 8086, ...)
- Sivuttava
  - sivutaulut
  - virtuaaliavaruus jaettu saman kokoisiin sivuihin
- Segmentoiva
  - virtuaaliavaruus jaettu ohjelman mukaan erillisiin eri kokoisiin segmentteihin
    - koodi segmentti, data segmentti, ...

Lisää  
tietoa?



käyttö-  
järjestelmä  
kurssit

Lisää  
tietoa?



käyttö-  
järjestelmä  
kurssit

# Keskeytystilanteet <sup>(3)</sup>

- Mikä tahansa tilanne, jonka käsittely vaatii poikkeuksen käskyjen normaaliin suorituserjestykseen
- Rakkaalla lapsella on monta nimeä:
  - poikkeus, keskeytys, virhetilanne, trappi, ...
  - exception, interrupt, fault, trap, failure, ...
  - SCV, KJ-kutsu, ...
- Jatkossa yleisnimi keskeytys tarkoittaa kaikkia näitä eri tapauksia tai tyyppejä



# Keskeytysten käsittely (4)

- Jokainen mahdollinen keskeytystyyppi on ennalta tunnettu
- Jokaiselle keskeytystyypille on oma käyttöjärjestelmän tuntema keskeytyskäsitteilyrutiini `interrupt handler`
- Jokaisen käskyn suorituksen jälkeen tarkistetaan keskeytysten olemassaolo SR:stä ja haaraudutaan keskeytyskäsitteilyrutiiniin tarvittaessa
  - joskus keskeytykset on estetty (ttk-91:ssä SR:n bitti D)
  - paluu käsitteilyrutiiniin ”return-from-interrupt” käskyllä
- ”Yllättävä aliohjelmakutsu”

# Keskeytystyyppejä (3)

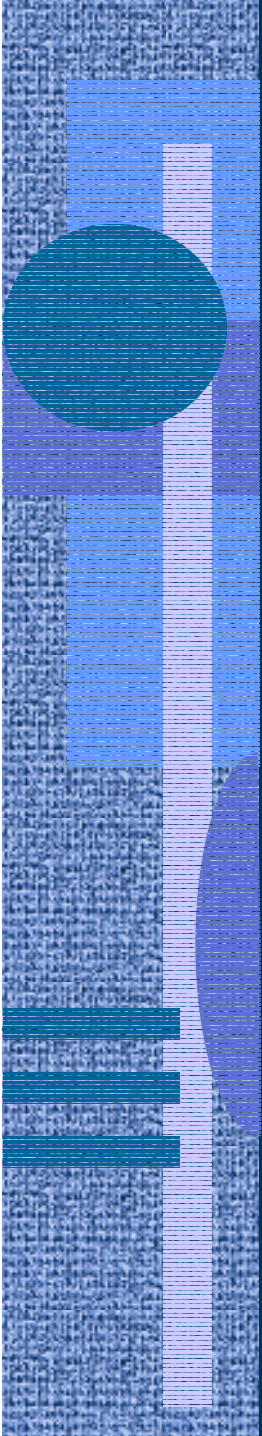
- Käskyn aiheuttamat virhetilanteet
- Käskyn aiheuttamat muut poikkeustilanteet
  - kyseessä ei siis ole virhetilanne
- Ulkoapäin (muualta kuin CPU:lta) tulleet

# Käskyn aiheuttamat virhetilanteet (5)

- Virheellinen käskyn tai datan osoite
- Tuntematon käsky (opcode)
- Nollalla jako
- Kokonaisluvun tai liukuluvun yli/alivuoto
- Käytetty osoite ei ole muistissa (MMU)

# Käskyn aiheuttamat muut poikkeustilanteet <sup>(3)</sup>

- SVC käsky
- I/O konekäsky
- Trace keskeytys



# Ulkoapäin (muualta kuin suorittimelta) tulleet keskeytykset <sup>(3)</sup>

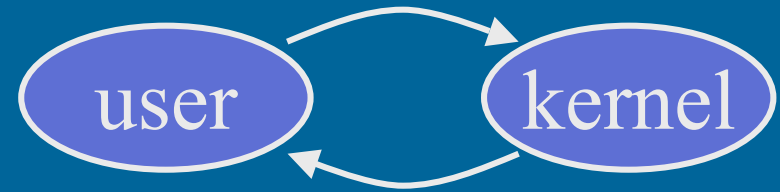
- Kellolaitekeskeytys (esim. joka 10 ms)
- Laitekeskeytys (esim. levy I/O valmis)
- Laitteistovirhe (esim. virhe väylän tiedonsiirrossa)



# Keskeytyskäsitteijä

- Osa käyttöjärjestelmää
- Ennen keskeytyskäsitteijän aloittamista asetetaan suoritin ja MMU käyttöjärjestelmätilaan (supervisor state)
  - SR:n bitti P on päällä => etuoikeutettu tila eli käyttöjärjestelmä tila
  - käyttöjärjestelmätilassa saa viitata mihin tahansa kohtaan muistia (MMU: BASE=0, LIMIT="hyvin iso")
  - käyttöjärjestelmätilassa saa käyttää kaikkia konekäskyjä
- Käsitteijästä paluun yhteydessä MMU:n tila ja suorittimen tila asetetaan ennalleen

# Suorittimen tilat <sup>(6)</sup>



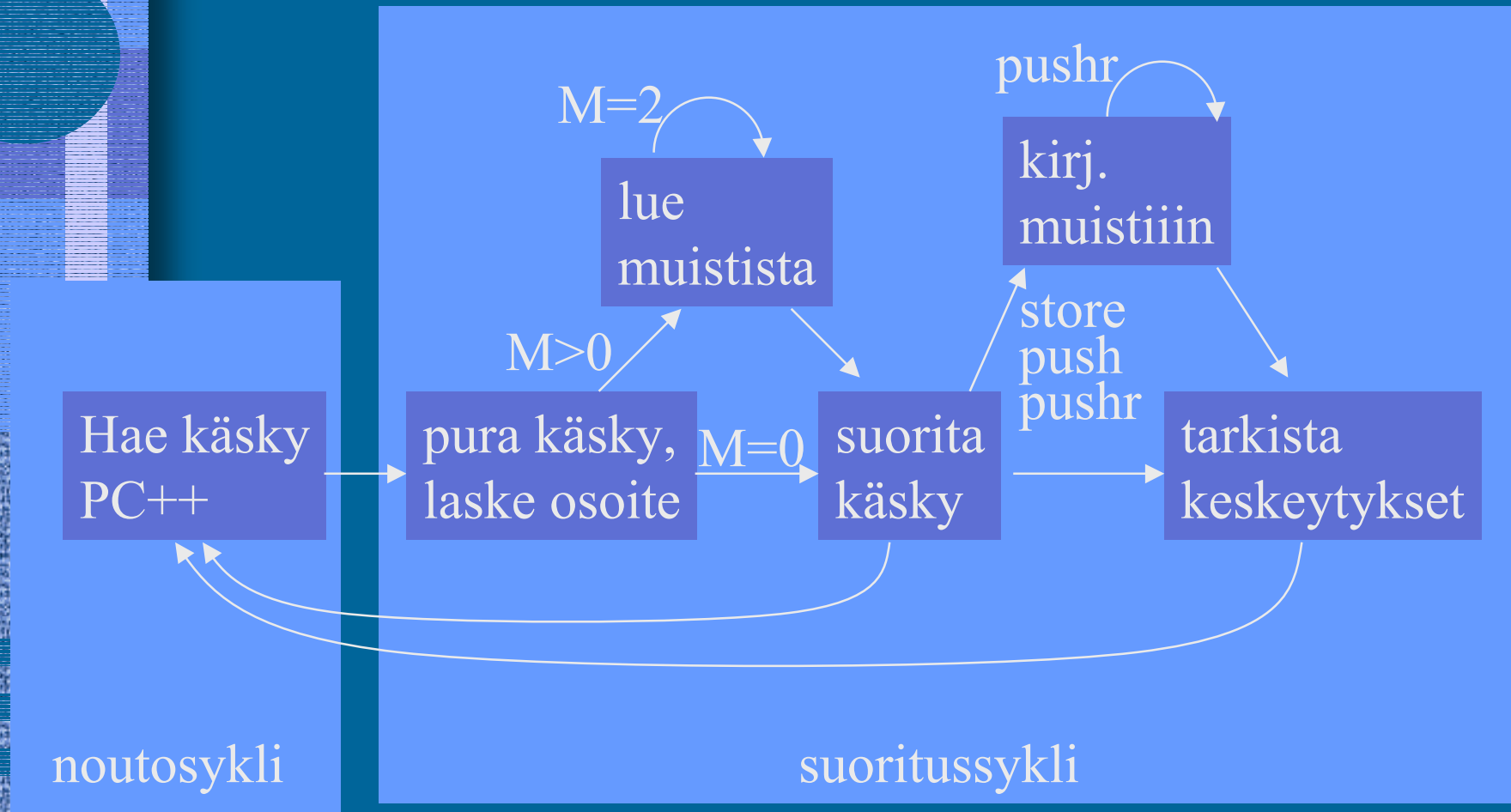
- Käyttäjätila (user mode, normal mode)
  - voi käyttää vain tavallisia käskyjä
  - voi viitata vain käyttäjän omaan muistiavaruuteen (MMU valvoo)
- Etuoikeutettu tila tai (KJ:n) ytimen tila (kernel mode, privileged mode)
  - voi käyttää kaikkia konekäskyjä, myös etuoikeutettuja (esim. clear\_cache,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

# Suorittimen tilan muuttaminen (6)



- Käyttäjätila → etuoikeutettu tila
  - keskeytys tai suora KJ:n palvelupyyntö (SVC käsky)
  - keskeytyskäsittelijä tarkistaa onko (oliko) oikeutta tilan vaihtoon (interrupt handler)
- Etuoikeutettu tila → käyttäjätila
  - etuoikeutettu konekäsky “return from interrupt handler” esim. IRET (Pentium II)
  - palauttaa kontrollin keskeytyneeseen kohtaan ja suorittimen tilan keskeytystä edeltäneeseen tilaan

# TTK-91 Nouto- ja suoritussykli vielä vähän tarkemmin



# Väylät (5)

- Tiedon siirtoa varten laitteistossa
- Yksi kirjoittaja kerrallaan
- Toteutettu johdinkimppuina
- Eri tasoilla



– suorittimen sisällä ”sisäinen väylä” (internal bus)

– muistiväylä suorittimen ja muistin välillä (memory bus)

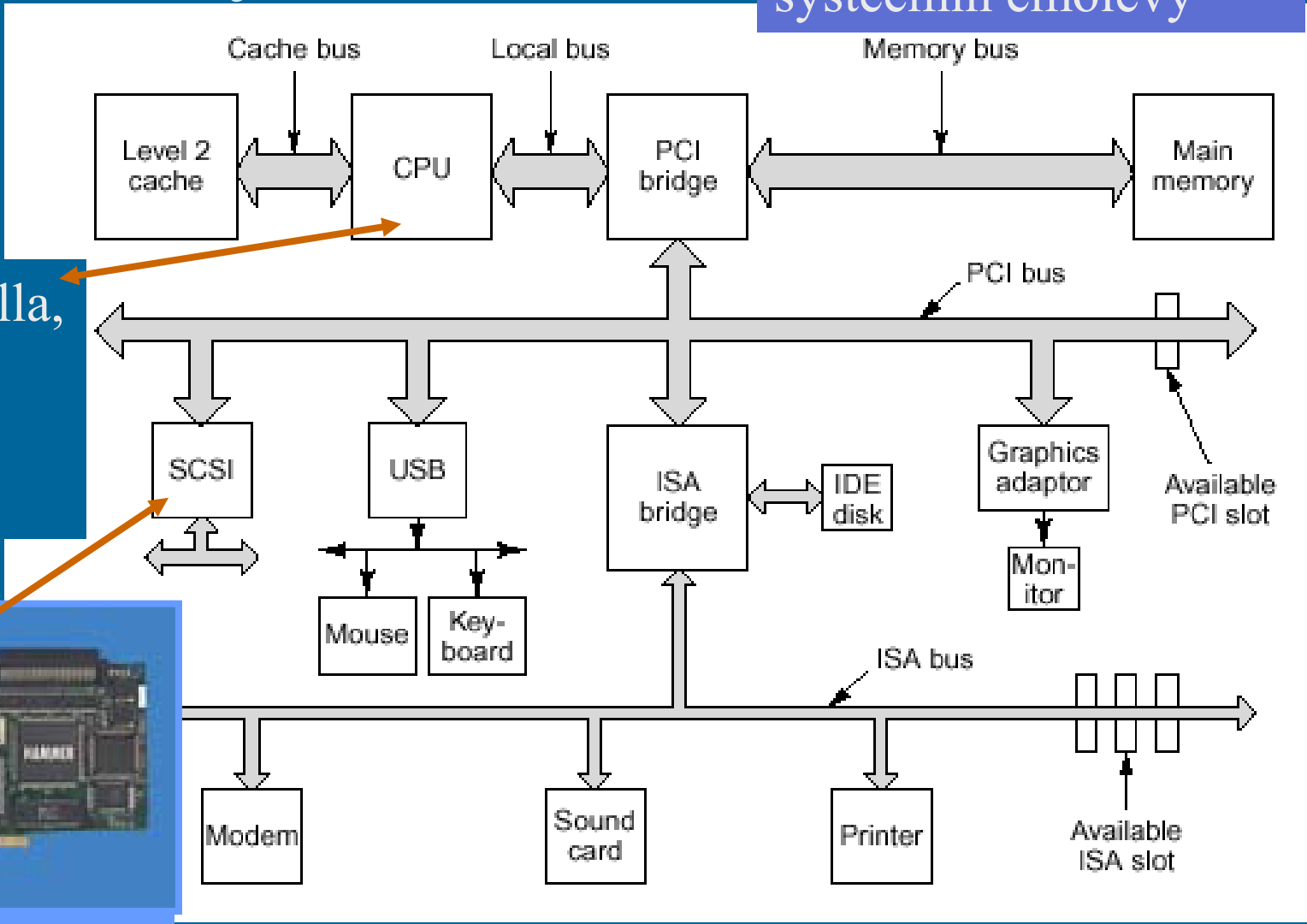
– I/O-väylä muistiväylän ja I/O-laitteiden välillä (I/O bus)

- Useita eri tapoja yhdistellä edellä olevia

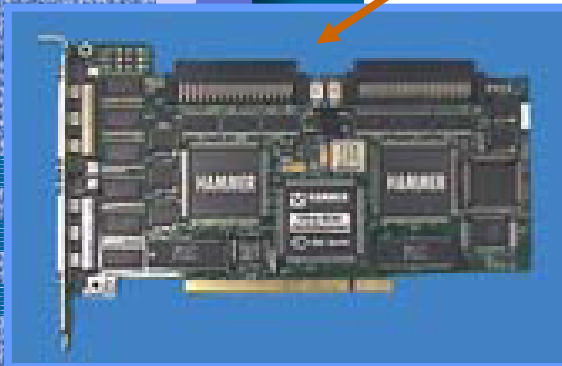


# Väylähierarkia

Tyypillinen Pentium II systeemin emolevy



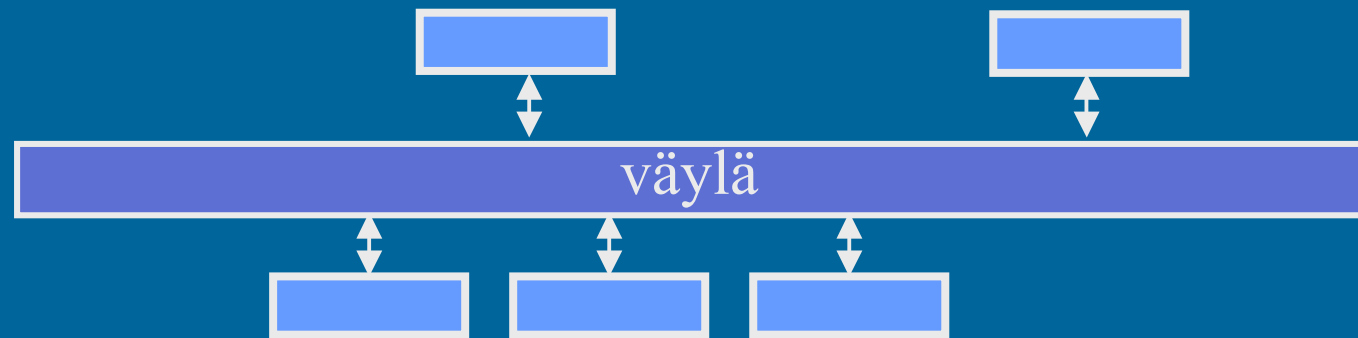
omalla lastulla,  
tason 1  
välimuistin  
kanssa



PCI to SCSI bridge

Fig. 3-50 [Tane99]

# Väylät (5)



- Kullakin laitteella oma osoite
- Yksi lähettää, kaikki kuulevat, vain ”oikea” laite vastaanottaa
- Paljon erilaisia
- Lähellä suoritinta olevat ovat nopeampia

Lisää  
tietoa?



Tietokoneen  
rakenne-  
kurssi

# TTK-91 koneen KOKSI simulaattori <sup>(6)</sup>

- Tavallinen Pascalilla kirjoitettu ohjelma
- TTK-91 koneen osat tietorakenteina
  - rekisterit, MMU, CU, muisti
- Simuloi käskyjen suoritussykliä käsky kerrallaan
- Toteuttaa myös TTK-91 koneen käyttöjärjestelmän osat osana tavallista ohjelmaa
  - assembler käääntäjä, lataaja, debugger, kesk. käsittelijät
- Graafinen käyttöliittymä

ks. suoritussyklin toteutus Koksissa (6 kalvoa)

# TTK-91 käskyn suoritus sykli (5)

hae käsky simuloidusta muistista

$IR = \text{mem}[PC]$

pura käsky osiin (OPER, Rj, M, Ri, ADDR) ja laske osoiteosan arvo TR (ADDR tai  $\text{regs}[Ri] + \text{ADDR}$ )

$\text{ADDR} = IR \% 65536$

$TR = \text{regs}[Ri] + \text{ADDR}$

tee tarvittava määrä (M) operandin hakuja muistista rekisteriin TR

$TR = \text{mem}[TR]$

valitse aliohjelma operaatiokoodin (OPER) perusteella

$\text{if } (\text{opcodeOK}[\text{OPER}] = \text{FALSE}) \text{ then } \text{SR.U} = 1;$

simuloi konekäskyn suorituksen muutokset rekistereihin (R0...R7, SR, PC, MAR, MBR)

$\text{ADD } Rj, M \text{ ADDR}(Ri) \Rightarrow \text{regs}[Rj] += TR;$

lopetta suoritus jos SVC tai keskeytys

$\text{SR.O} = \dots$

# -- Luennon 5 loppu --

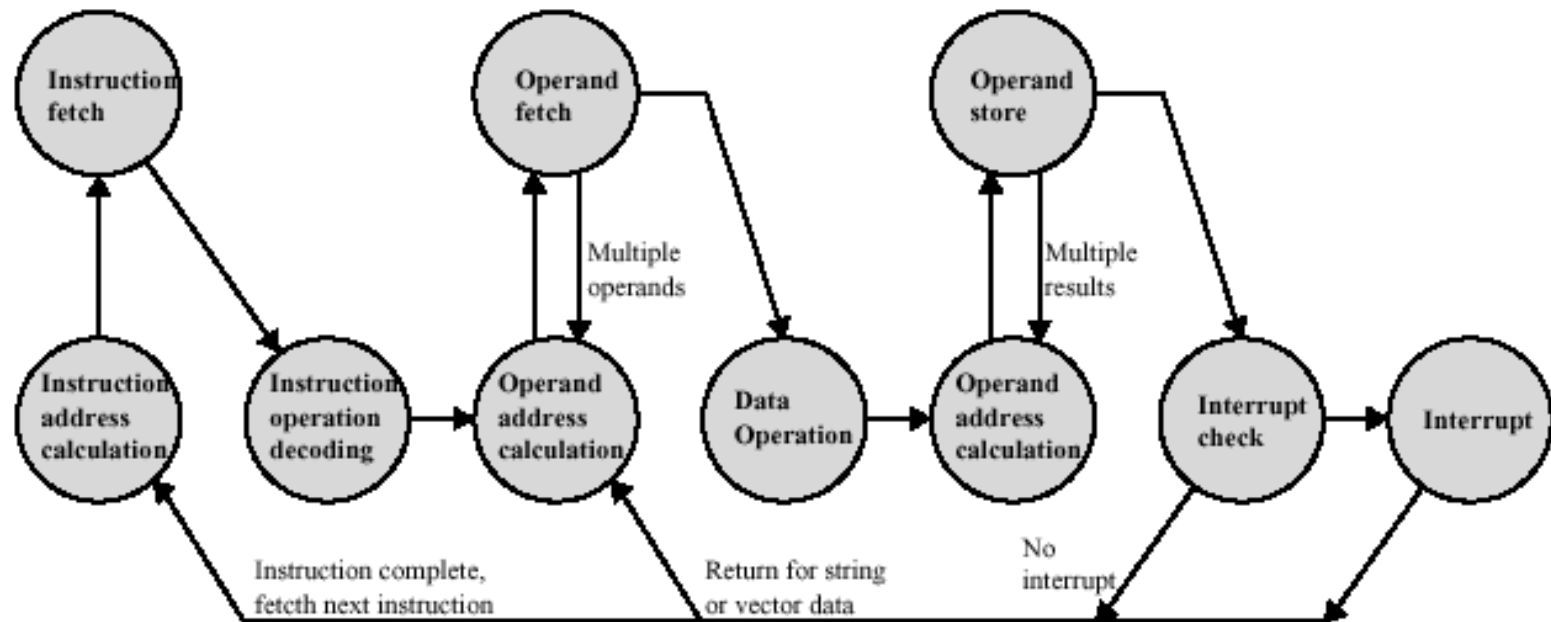


Figure 3.12 Instruction Cycle State Diagram, With Interrupts

[Sta199]