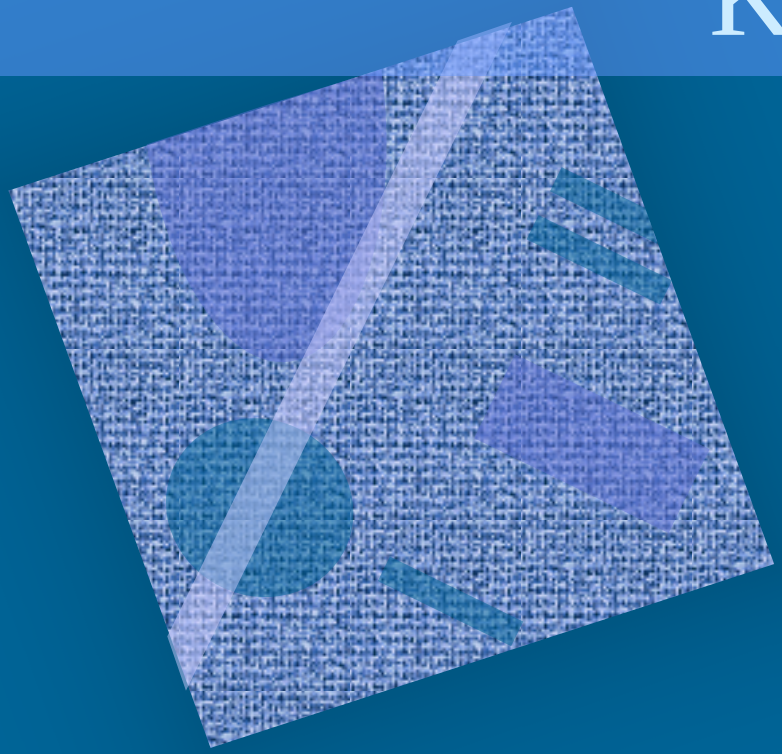
Luvut 3 ja 4
Kertausluento 2

Tiedon esitysmuodot Käyttöjärjestelmä



Monitavuinen tieto

Kokonaisluvut, liukuluvut

Merkit, merkkijonot

Äänet, kuvat, muu tieto

Prosessi ja sen esitysmuoto

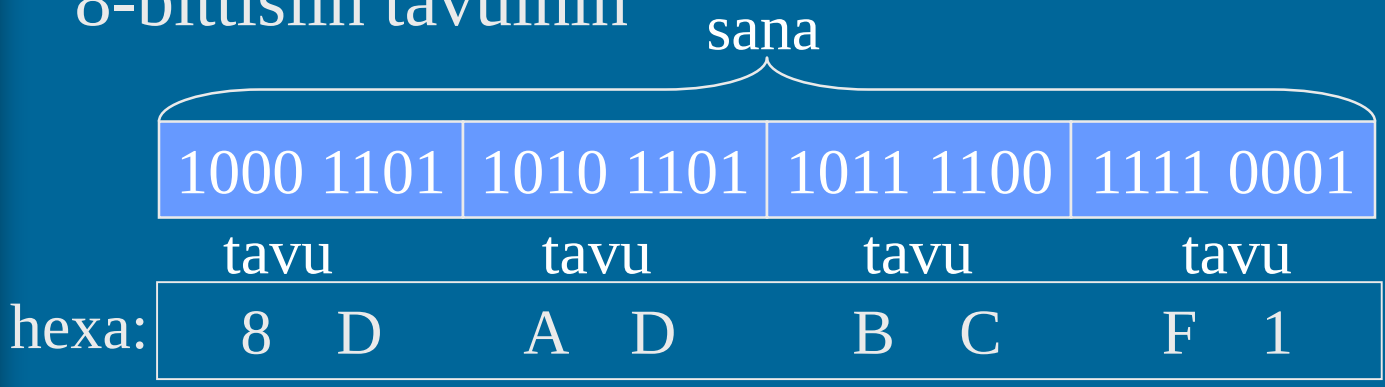
Käyttöjärjestelmä ja sen toteutus

Tiedon esitys

- Kysymys: miten esittää eri tyyppisiä tietoja?
- Vastaus: koodataan ne biteiksi
  - kaikki tieto on koneessa bitteinä
- Kaikelle käsitellylle tiedolle on omat koodausmenetelmänsä
  - kaikkia koodausmenetelmiä ei ole standardoitu
  - samalle tietotyypille voi olla useita koodausmenetelmiä
    - kokonaisluvut, liukuluvut, merkit, merkkijonot, kuvat, ...
  - ongelma: ymmärtävätkö koneet toisiaan?
    - tiedon esitysmuotoa voidaan joutua muuttamaan, kun tietoa siirretään koneelta toiselle

Tiedon esitys laitteistossa

- Kaikki tieto koneessa on binääribitteinä (0 tai 1)
  - binäärijärjestelmän numerot: 0, 1
  - helppo toteuttaa piireillä
  - helppo suunnitella logiikkaa Boolen algebran avulla
- Muisti jaettu tasapituisiin sanoihin (word)
  - sana = word = 32 bittiä (ennen 16, 32, 48, 64 bittiä, …)
- Yleensä sana on jaettu tasapituisiin 8-bittisiin tavuihin (byte)



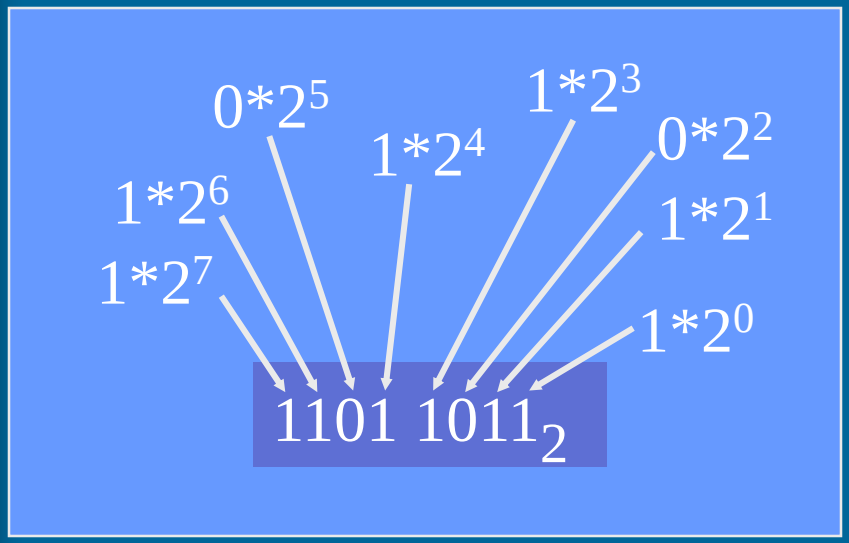
Suorittimen ymmärtämä tieto

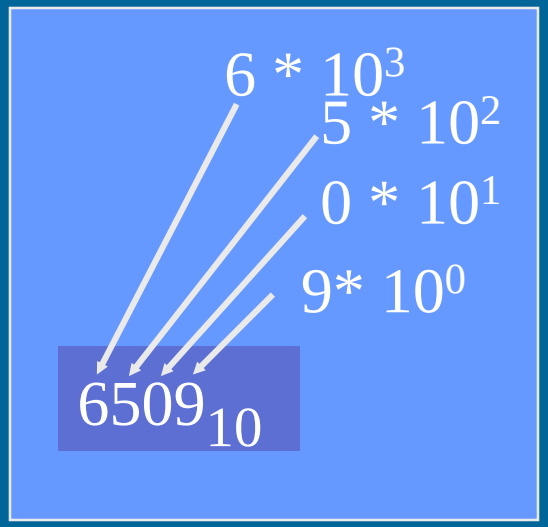
- Kaikki tieto koneessa on koodattuna biteiksi
- Muistissa voidaan esittää kaikki tieto millä tahansa sovitulla esitystavalla (koodauksella)
- Joillekin esitystavoille on omat konekäskyt (eli suoritin ymmärtää niitä)
  - Kokonaisluvut ja liukuluvut (lähes aina)
  - Totuusarvot, merkit ja merkkijonot (joskus)
  - Kuvat ja äänet (ei yleensä ellei erikoistunut suoritin)
- Muiden tietojen käsittely tapahtuu ohjelmallisesti (eli usea käsky tai aliohjelma tai metodi)
  - Esim. merkkejä (merkkien esitysmuotoa) voidaan käsitellään kokonaislukuoperaatioilla ja aliohjelmilla
  - Rationaaliluvut, 128-bittiset kokonaisluvut (?), isot taulukot, tietueet, oliot, sormenjäljet, äänet, kuvat, hajut, …

TTK-91: kokonaisluvut

Binäärijärjestelmä

- Kantaluku 2, numerot 0 ja 1
  - numeroiden painoarvot oikealta vasemmalle: $1=2^0$, $2=2^1$, $4=2^2$, $8=2^3$, $16=2^4$, $32=2^5$, …
  - kymmenjärjestelmässä painoarvot ovat $1=10^0$, $10=10^1$, $100=10^2$, $1000=10^3$, …





Heksadesimaaliesimerkkejä

binääri: 0100 0111 1001 1010 1111

16-järj: 4 7 9 A F

$= 479AF_{16}$

$= 0004\ 79AF_{16} = \underline{0x}\ 479AF$

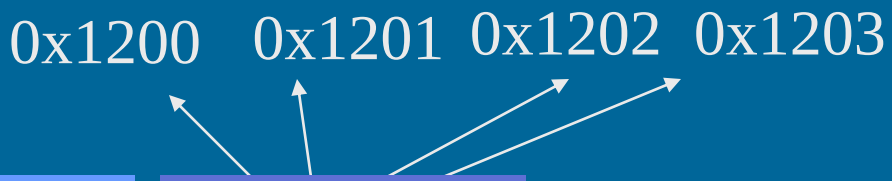
16-järj: $120ADF_{16}$ 1 2 0 A D F

binääri: 0001 0010 0000 1010 1101 1111

Big vs. Little Endian

- Miten monitavuinen tieto talletetaan?
  - Yleensä per sana tai kaksoissana

sanan osoite (pienin tavuosoite)



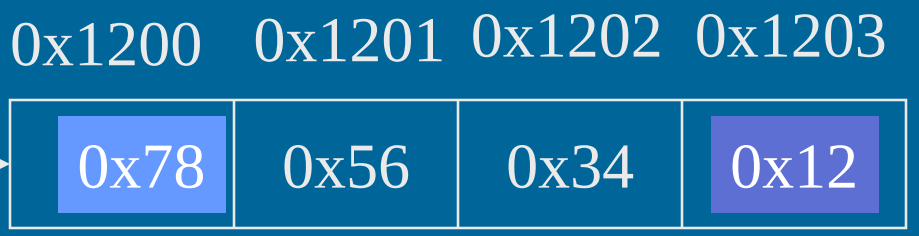
talleta **0x12345678** ??

tavuosoitteet

0x12345678

"normaali" tapa

Big-Endian: eniten merkitsevä tavu pienimpään osoitteeseen



Little-Endian: vähiten merkitsevä **tavu** pienimpään osoitteeseen

0x78563412

Kokonaisluvut

arvo talletus

+57 = 0011 1001

Yleensä positiiviset suoraan binäärinä

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

sign bit = MSB

= most significant bit

luku -57 = 1011 1001 talletus-muoto

-57 = 1100 0110

“sign” bit

+1

-57 = 1100 0111

“sign” bit

-57 = 0100 0110

-57 + 127 =70

+57 = 1011 1000

+57+127=184

Keskustele

Liukuluku

- Vastaa reaalilukua tietokoneessa
- Kompromissi
  - Suuruusluokka vs. tarkkuus
- Aina kiinteä maksimitarkkuus
  - Sama määrä merkitseviä numeroita
- Etumerkki, mantissa, suuruusluokka

$+5678901.2345678$ vs. $+5.678901 * 10^{6}$

-0.000012345678 vs. $-1.234568 * 10^{-5}$

$+111.1010101111$ vs. $+1.11101011 * 2^{2}$

IEEE 32-bit FP Standard

vakiolisäys +127

$+6144.0_{10} = +1\ 1000\ 0000\ 0000.0_2 = +0.0011 * 2^{15} = \ldots$

(+0x7F)

1b	8b (+127)	23b
"+"	"15"	"0.1875" = "0.0011"
sign	exponent	**mantissa** or significand

1/8 = 0.1250
1/16= 0.0625
0.1875

0 = '+'
1 = '-'

- 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)

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

24 bitin mantissa!

Miksi käytetään piilobittiä?

Keskustele

UCS ja Unicode

- UCS - Universal Character Set
- Samat merkistöt, eri standardit
- 2 tavua eli 16 bittiä per merkki
  - 65536 merkkiä koko maailmassa käytössä oleville n. 200000 symbolille
- (32-bit UCS-4 sisältää myös kaikki kiinalaiset merkit)
- Kontrollimerkit
  - 0x0000-001F and 0x0080-009F
  - 0x007F = DELETE, 0x0020 = SPACE
- UCS:ssä myös 8-bittiset koodi ”rivit”
  - eri alueille tai tarkoituksiin (zone) omat 8-bittiset koodinsa, esim. UTF-8

Merkkijonot

- Yleensä peräkkäin talletettu joukko tavuja
- Lisäksi tarvitsee jollain tavalla koodata merkkijonon pituus:
  - laitetaan loppuun erikoismerkki
    - C-kieli: ’\0’ = 0x00
  - toteutetaan tietueena

pituus merkkijono

- ei (yleensä enää) omia konekäskyjä, manipulointi aliohjelmilla
  - kokonaisluku- ja bittimanipulointikäskyt
  - joissakin koneissa ”strcopy” ja ”strcmp” konekäskyt (oletettu jokin tietty merkkijonojen esitysmuoto)

Totuusarvot

- Boolean TRUE ja FALSE
- Yleensä koodattu TRUE=1, FALSE=0
  - muttei aina!
  - Totuusarvolauseke ***A and B*** = kokon.lukulauseke ***A*B***
- Usein Boolean arvo per sana
  - loput 31 bittiä nollia
  - ohjelmointikielten Boolean muuttujat
- Joskus pakatussa muodossa 32 arvoa per sana
- Ei omia konekäskyjä, manipulointi bittimanipulaatiokäskyillä ja aliohjelmilla
  - Bittimanipulaatiokäskyt ovat yleensä kaikille sanan (tavun) biteille!

Kuvat, videot, äänet

- Monta standardia
  - Yleisyys, siirrettävyys, pakkaustiheys
- Pakattu (usein) mahdollisimman vähän tilaa vievää muotoon
  - Optimoitu tilan (tiedonsiirron), ei laskennan mukaan
  - Purkaminen voi vaatia paljon laskentaa
- Ei omia konekäskyjä, manipulointi aliohjelmilla ja/tai erikoissuorittimilla (jotka voi olla integroitu emolevyyn)

Maku, haju, tunto ja muu data

- Tähtien kirkkaus, hajut,
  veneen tyyppi, tunteen palo, ….
- Toteutus sovelluskohtaisesti, ei vielä yleisiä standardeja
  - kokonaisluvut (diskreetti data)
  - liukuluvut (jatkuva data)
- Ei omia konekäskyjä, manipulointi omilla aliohjelmilla

Konekäskyjen esitysmuoto

- Konekohtainen, jokaisella omansa
- Käskyt ovat 1 tai useamman tavun mittaisia
  - SPARC, kaikki käskyt: 1 sana eli 4 tavua
  - ARM, kaikki käskyt: 1 sana eli 4 tavua
  - Pentium II: 1-16 tavua, paljon variaatioita
- Käskyillä on yksi tai useampi muoto, kussakin tietty määrä erilaisia kenttiä
  - opcode, Ri, Rj, Rk, osoitusmoodi
  - pitkä tai lyhyt vakio

TTK-91, kaikki käskyt: 1 sana, 1 muoto

Taulukkojen esitysmuoto

- Peräkkäisrakenteena, kuten esimerkit aikaisemmin
- Riveittäin tai sarakkeittain tai linkitettynä rakenteena
- Ei omia konekäskyjä, manipulointi aliohjelmilla tai toistorakenteilla
  Poikkeus: vektorisuorittimet, joilla
  - vektorirekisterit (esim. 8 tai 64 liukulukua) tavallisten rekistereiden lisäksi
  - Multimediakäskyt: 64b = 8 * 8b vektori (Intel MMX)
  - omia konekäskyjä vektorioperaatioita varten
- Indeksoitu tiedonosoitusmoodi tukee peräkkäin muistiin talletettujen 1-ulotteisten taulukoiden käyttöä

Keskustele

Tietueiden esitysmuoto

record

- Peräkkäisrakenteena
- Osoite on jonkin osoitinmuuttujan arvo
- Ei omia konekäskyjä, manipulointi aliohjelmilla tai kääntäjän generoimien vakiolisäysten avulla
- Indeksoitu tiedonosoitusmoodi tukee tietueiden käyttöä

Olioiden esitysmuoto

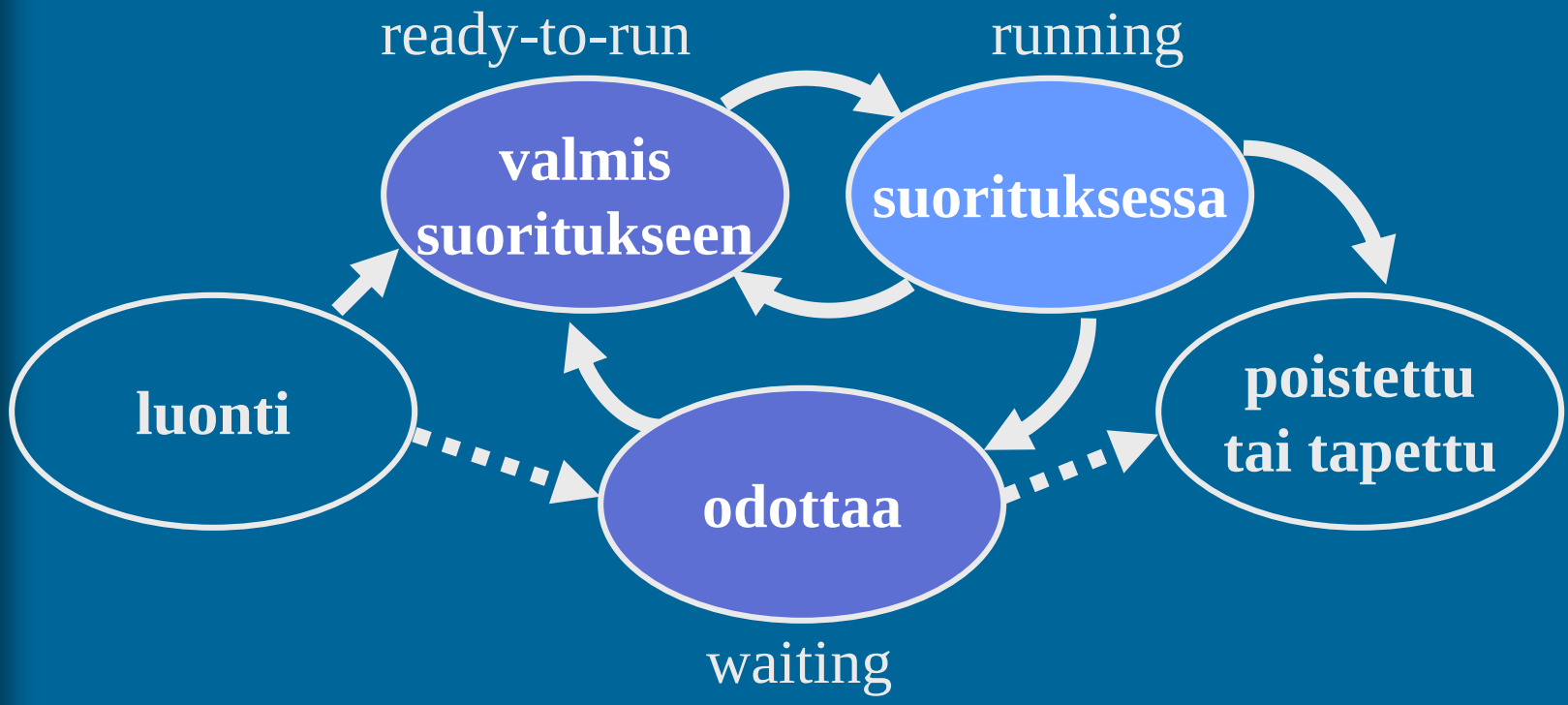
object

- Kuten tietueet, yleensä varattu keosta (heap)
- Useat olion kentistä sisältävät vuorostaan osoitteen keosta suoritusaikana varattuun toiseen olioon
- Metodit ovat aliohjelmien osoitteita
- Ei omia konekäskyjä, manipulointi aliohjelmilla

Prosessi = ohjelma järjestelmässä

- Järjestelmässä voi olla ”samalla kertaa” **monta prosessia** joko samasta tai eri ohjelmasta
  - Käyttäjän (ihmisen) näkökulma ja aikaskaala (1 min, 1 sek, 10 ms)
- Suorittimella **suorituksessa on vain yksi** prosessi kerrallaan
  - Oletus: 1 ytiminen (1 core) suoritin
  - Laitteiston näkökulma ja aikaskaala
    - 1 ns, 1 µs, 1 ms?
- **Muut prosessit ovat odottamassa** jotakin
  - suoritinta? I/O:ta? viestiä toiselta prosessilta?
  - vapaata muistitilaa?

Prosessin elinkaari (perusmalli)



- Prosessin 5 suoritustilaa
  - Milloin mikäkin tilanvaihto tapahtuu?
    - Mikä tapahtuma saa aikaan tilanvaihdon?
    - Mitä tilanvaihdossa tapahtuu?
    - Kuka jatkaa suoritusta tilanvaihdon jälkeen?

Keskustele

Prosessin kuvaaja (PCB)

Process Control Block

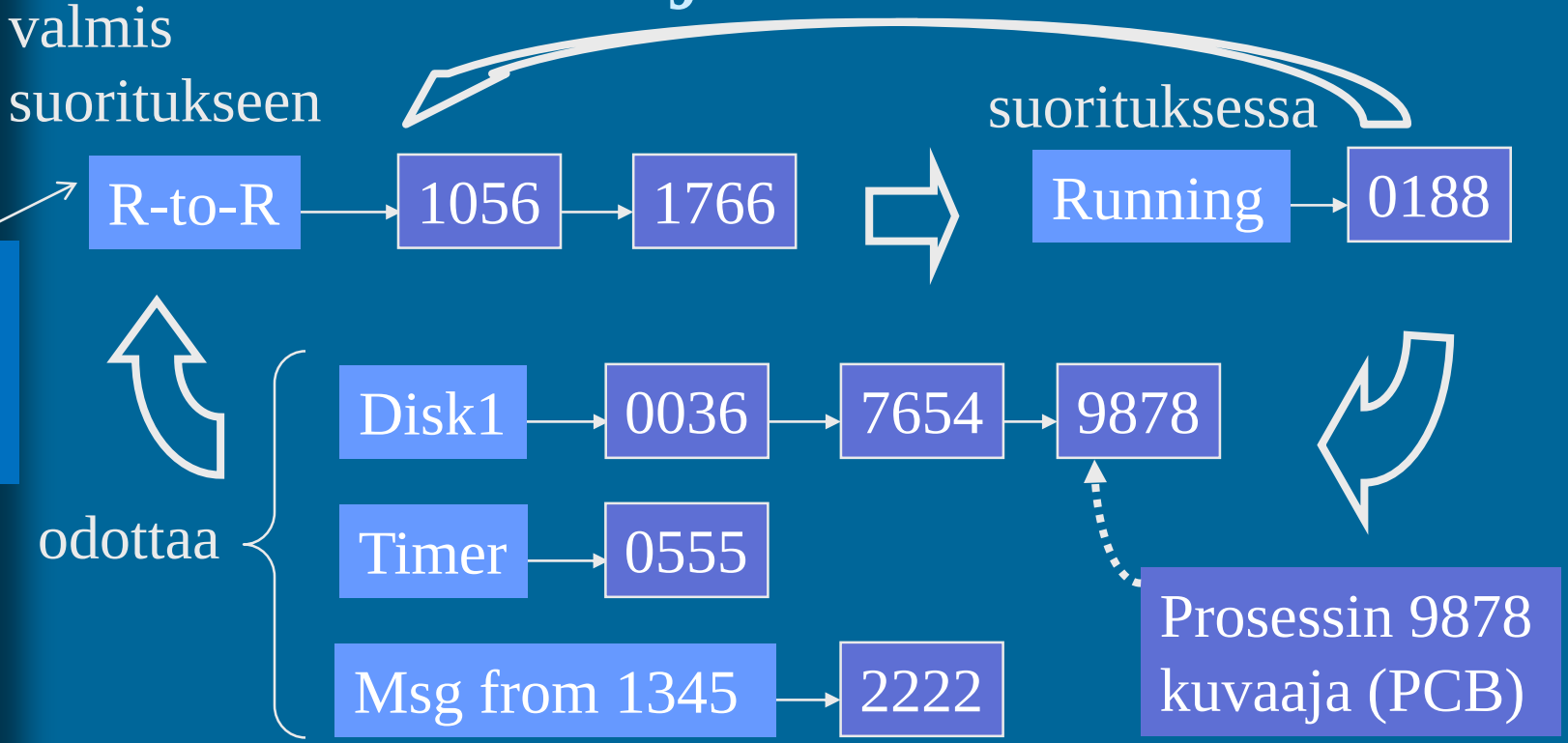
- Prosessin tunniste
  - 14023
- Prioriteetti suorittimen vuoronantoa varten
  - 143
- Prosessin tila ja/tai odottamisen syy
  - R-to-R
- Suoritinympäristö talletettuna odottamisen aikana
  - Työrekisterit, SP, FP, tilarekisterit, …
  - PC, seuraavaksi suoritettavan käskyn osoite
    - Prosessi vaihtuu, kun tämä ladataan
    - aluksi main {}
- Poikkeuskäsittelijöiden osoitteet (ellei oletusarv.)
- Aikaviipale (milloin KJ antaa vuoron toiselle)
- Käytössä olevat muistialueet, auki olevat tiedostot
- KJ:n hallintotietoa (kokonaisaika, etc etc)

processor context

time slice

Keskustele

Prosessit jonoissa



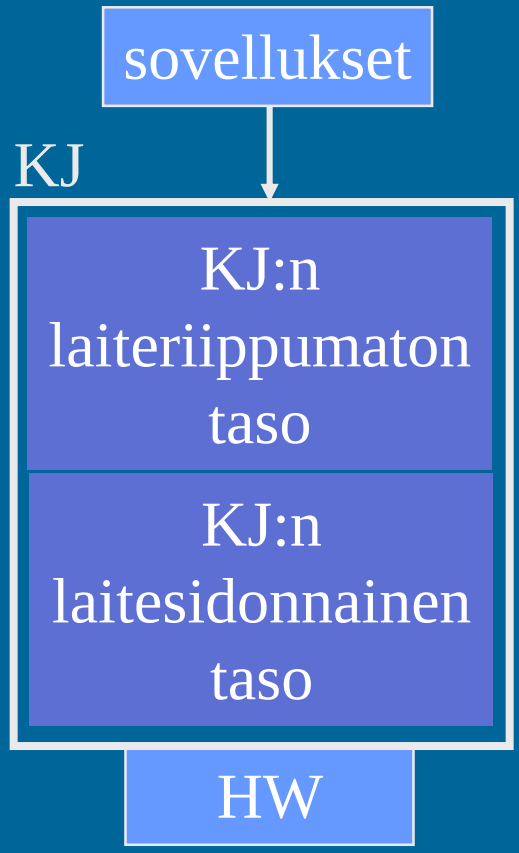
Vuoronanto:
valitse seuraava prosessi Ready-to-Run -jonosta ja
siirrä se suoritukseen CPU:lle
(kopioi tämän prosessin **suoritinympäristö** suorittimelle)

Prosessin vaihdon toteutus

- Vaihdon tekee KJ rutiini sillä hetkellä suorittavan prosessin ympäristössä (esim. keskeytyskäsittelijässä)
- Talleta vanhan prosessin suoritinympäristö suorittimelta omalle talletusalueelle (PCB:ssä) muistiin
  - Rekisterit, myös PC (tästä jatketaan joskus …)
  - Ei tarvitse laittaa talteen, jos vanha prosessi tapetaan
- Kopioi uuden prosessin suoritinympäristö omalta talletusalueeltaan (PCB:ssä) suorittimelle
  - Lataa rekisterit (viimeisenä PC)
- Uuden prosessin suoritus jatkuu täsmälleen siitä mihin viime kerralla jäätiin
  - Sama konekäsky, käytännössä sama suoritusympäristö
  - Yleensä keskellä prosessin vaihtoa suorittavaa KJ rutiinia
    - Keskeytyskäsittelijässä?
  - Seuraavaksi palataan tästä rutiinista ja jatketaan laskentaa

Käyttöjärjestelmä (KJ)

- Laiteriippumaton (HW-riippumaton) sovellusten rajapinta laitteistoon
  - Helpottaa laitteiston käyttöä
  - Jakaa palvelua kaikille reilusti
  - Resurssien hallinta ja valvonta
  - Sovellukset on helpompi toteuttaa ja siirtää muualta



Käyttöjärjestelmän rakenne

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

Käyttöjärjestelmän (KJ) toteutus

- Joukko prosesseja ja/tai aliohjelmia
  - prosessit elävät omaa elämäänsä
    - käyttäjätason prosessi
    - etuoikeutettu prosessi, root-prosessi
    - esim. laiteajuri
  - aliohjelmat suoritetaan sen hetkisen prosessin ympäristössä (etuoikeutetussa tilassa?)
    - esim. keskeytyskäsittelijä, laiteajuri
  - saavat kontrollin aina tarvittaessa
    - aliohjelmakutsut, SVC, viestit
    - ajastimet ja muut keskeytykset
    - KJ ei tee mitään, ellei sen koodia ole suorituksessa!

daemon
windows service
windows palvelu

Keskustele

KJ palvelun kontrollin palautus

- Aliohjelmakutsut
  - CALL → RETURN
- SVC
  - SVC → IRET
- Viestit
  - viesti → vastausviesti
    (lähettäjä odottaa vastausta RECEIVE:ssä)
- Ajastimet ja muut keskeytykset
  - Keskeytys → IRET
    (vuoro keskeytyneelle tai vuoronantajan valitsemalle seuraavalle prosessille)

Explisiittinen KJ-palvelun kutsu = Palvelupyyntö

Implisiittinen KJ-palvelun kutsu

KJ palvelu suoritetaan keskeytetty prosessin sisällä. Prosessi ei odota Ready-jonossa.