Kertausluento 4 (lu9, lu10)

Järjestelmän ulkoinen muisti
Käännös, linkitys ja lataus

Muistihierarkia

Kiintolevyt

I/O:n toteutus

Käännös, Linkitys Lataus

Muistihierarkia

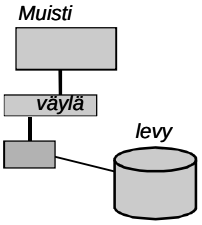
ks. Fig 4.1 [Stal10]
(ks. Fig 4.1 [Stal03])

- Ulkoinen muisti on halvempaa toteuttaa per tavu
- Ulkoinen muisti on hyvin paljon hitaampaa kuin sisäinen muisti

- Aika/tila optimointi
  - suuret tietomäärät täytyy (kannattaa) kustannussyistä pitää ulkoisessa muistissa
  - pienet tietomäärät täytyy (kannattaa) tehokkuussyistä pitää sisäisessä muistissa
- Kaiken viitatun tiedot tulee suoritusaikana olla sisäisessä muistissa!

Virtuaalimuisti

- Osa muistihierarkiaa
- Vastaus ongelmaan
  - miten tehdä suoritusaikaisesta muistista ”yhtä suuri” kuin levymuisti ja ”yhtä nopea” kuin keskusmuisti?
- Kaksitasoinen:
  - keskusmuistissa kulloinkin käytössä olevat alueet
  - levyllä kaikki tiedot
  - kopiointi tarvittaessa



Virtuaalimuistin toteutus

- Toteutustavat
  - kanta- ja rajarekisterit
  - sivutus
  - (segmentointi ja sivuttava segmentointi)
- Pääosa toteutuksesta ohjelmistotasolla
- Laitteistotuki
  - MMU – muistinhallintayksikkö
  - nopeuttaa viitatun muistipaikan todellisen osoitteen laskentaa
    - osoitetta ei tarvitse laskea usealla konekäskyllä, kun MMU tekee sen laitteistotasolla
  - rakenne ja toiminta vaihtelee virtuaalimuistin toteutustavan mukaan

Tiedostojärjestelmä

- KJ:n osa, hallitsee kaikkia tiedostoja
- Valvoo oikeuksia tiedostoa avattaessa
- Muuntaa tiedostonimet fyysisiksi osoitteiksi
- Ylläpitää taulukoita, joista näkee mitä kohtaa mistäkin tiedostosta kukin prosessi on käsittelemässä
- Tiedostojärjestelmä lukee ja kirjoittaa tiedostoja suurina kerralla käsiteltävinä lohkoina (0.5-8 KB?)
  - käyttäjätason prosessit käsittelevät tiedostoja tavuittain eikä niiden tarvitse tietää tiedoston todellista fyysistä rakennetta (KJ:n laiteajuri huolehtii siitä)

Levymuistin saantiaika

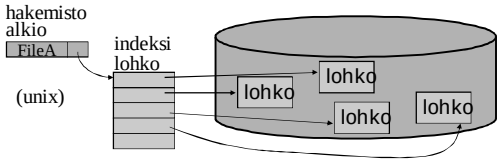
- Tiedon osoite: levypinta + ura + sektori
  - laiteajuri etsii KJ-taulukoista loogisen osoitteen perusteella

ks. Fig 6.2 [Stal10]
(ks. Fig 6.2 [Stal03])

- Saantiaika:
  - hakuvarren siirtoaika (seek time)
    Esim: aver 6.3 ms, min-max 2-15 ms?
  - odota kunnes sektori kohdalla (rotational delay)
    Esim: pyörähdysviive kun 3600 rpm: 8.33 ms
    (keskim. puolen kierroksen aika)
  - siirrä sektorin verran tietoa (data transfer time)
    Esim: pyör.aika / sekt. lkm = 0.42 ms

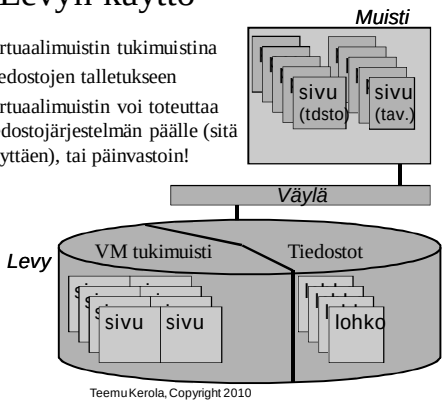
Tiedoston talletus levylle

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

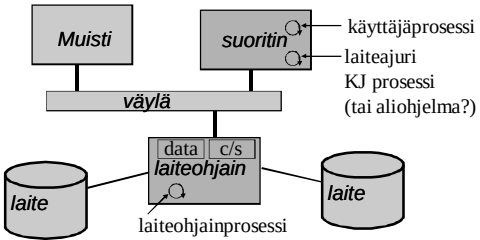


Levyn käyttö

- Virtuaalimuistin tukimuistina
- Tiedostojen talletukseen
- Virtuaalimuistin voi toteuttaa tiedostojärjestelmän päälle (sitä käyttäen), tai päinvastoin!



Laiteohjain (I/O Moduuli)



Ks. Fig 7.4 [Stal10]
(ks. Fig 7.4 [Stal03])

Laitteiden käytön toteutus

ks. laiteohjainkuva (ed. kalvo)

- Käyttäjäohjelma kutsuu käyttöjärjestelmän laiteajuria tekemään I/O:n. Laiteajuri suoritetaan samalla suorittimella kuin käyttäjäohjelmakin.
- Laiteajuri ohjaa laitteen toimintaa laitteen laiteohjaimella olevien kontrollirekisterien (muistialue ”c”) avulla
- Laiteajuri voi lukea laitteen tilatietoa laiteohjaimella olevien statusrekisterien (muistialue ”s”) avulla
- Laiteajuri voi lukea (kirjoittaa) laitteen lukemaa (laitteelle kirjoitettavaa) tietoa laiteohjaimella olevien datarekistereiden (muistialue ”data”) avulla
- Kontrolli-, status- ja datarekisteri kolmikko muodostaa ”I/O portin” suorittimen näkökulmasta

Laiteohjaimen rekistereihin viittaaminen

- Ongelma: miten suorittimella suorittavan laiteajuri viittaa eri kortilla oleviin ”rekistereihin” (muistiin)?

ks. laiteohjainkuva

- Ratkaisu 1: omat I/O-konekäskyt tätä tarkoitusta varten
  - käskyssä annetaan laiteohjaimen identifikaatio ja laiterekisterin nro (oma I/O osoiteavaruus)
  - vaikea laajentaa käyttöä uusiin laitteisiin, joilla ”laiterekisterit” voivat olla hyvinkin erilaisia
  - suorittimen konekäskyjä ei voi muuttaa

x86: IN, OUT INS, OUTS

KOKSI: IN, OUT

Ratkaisu 2: muistiinkuvattu I/O

ks. laiteohjainkuva

- Laiteajuri lukee/kirjoittaa laiteohjaimella olevia rekistereitä (data, status/kontrolli) tavallisilla muistin luku/kirjoitus käskyillä

load R1,=DiskRd
store R2, DiskCtr

  - ei tarvita erillisiä I/O-konekäskyjä!
  - laiteohjaimella olevat ”laiterekisterit” ovat samanlaista viitattavaa muistia kuin ”normaali muisti”
  - muistisoitteen ensimmäiset bitit (ei siis käskykoodi) valitsevat, mille laitteelle (vai tavallisen muistiin) viittaus kohdistuu

DiskCtr EQU 0x80000001

  - voidaan käyttää rinnan I/O käskyjen kanssa (laiterekistereihin voi siis viitata sekä I/O-käskyillä että muistiinkuvatun I/O:n avulla)

esim. Intelin arkkitehtuurit

I/O tyypit

ks. laiteohjainkuva

- **Suora I/O**: laiteajuri odottaa tiukassa silmukassa, kunnes laiteohjaimen statusrekisteri ilmoittaa I/O-pyynnön valmistuneen (direct I/O, programmed I/O)
  - laiteajuri siirtää tietoa muistin ja datarekisterin välillä
- **Epäsuora I/O**: I/O:n odotusaikana suorittimella suoritetaan jotain muuta ohjelmaa (indirect I/O interrupt driven I/O)
  - Kun I/O-pyyntö valmistuu, laiteohjain antaa keskeytyksen (laitekeskeytys, I/O interrupt) suorittimelle, joka (jonkin ajan kuluttua) jatkaa kesken jäänyttä I/O-pyynnön esittänyttä ohjelmaa.
  - laiteajuri siirtää tietoa muistin ja datarekisterin välillä

I/O tyypit (jatkoa)

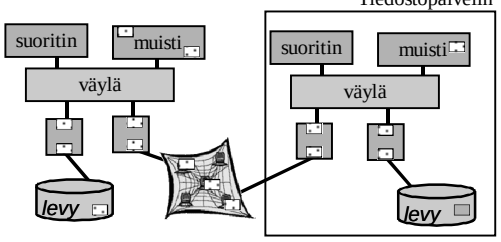
ks. laiteohjainkuva

- **DMA** - Direct Memory Access
  - Älykkäämpi laiteohjain
  - Laiteohjain voi suoraan kopioida tiedot keskusmuistiin
    - laiteajurin ei tarvitse laiterekistereitä käyttäen siirtää tietoa muistin ja datarekisterin välillä
    - Tieto kulkee väylän kautta vain yhden kerran!
  - Laiteohjain tekee paljon suuremman määrän työtä itsenäisesti (kuin epäsuorassa I/O:ssa) ennen suorittimelle annettavaa laitekeskeytystä

Tiedostopalvelin

- (Lähi)verkossa oleva palvelin
- Käytettäessä tiedoston (osien) kopio on muistissa (ja ehkä myös paikallisella levyllä)

orig. tiedosto
tiedoston kopio?



Esimerkki: kirjoittimen laiteajuri ttk-91 koneelle

- Laitteella voi tulostaa kokonaislukuja yksi kerrallaan
- Muistiinkuvattu I/O, suora I/O
- Laiteportti
  - kontrollirekisteri muistipaikka 1048576 = 0x80000
  - tilarekisteri muistipaikka 1048577 = 0x80001
  - datarekisteri muistipaikka 1048578 = 0x80002
- Laiteajuri Print toimii etuoikeutetussa tilassa
- Kutsu:

```
PUSH  SP, =0      ; space for return value
PUSH  SP, X       ; parameter to print
SVC   SP, =Print  ; returns Success/Failure
POP   SP, R1
JNZER R1, TakeCareOfTrouble
```


Esim: laiteajurin toteutus (12)

(kesk.)

ks. laiteohjainkuva

Solution with no timeout

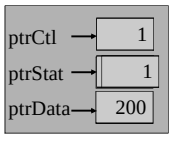
```
ptrCtr   DC  1048576 ; control register address
ptrStat  DC  1048577 ; status register address
ptrData  DC  1048578
retVal   EQU -3
parData  EQU -2

Print  PUSHR SP              ;save regs
       LOAD  R1, parData(FP)
       STORE R1, @ptrData ; data to print
       LOAD  R1, =0
       STORE R1, @ptrStat  ; init (clear) state register
       LOAD  R1, =1
       STORE R1, @ptrCtr  ; give command to print

Wait   LOAD  R1, @PtrStat ; check state register
       JNZER R1, Done
       JUMP  Wait          ; wait until I/O done

Done   LOAD  R1, =0        ; return "Success"
       STORE R1, retVal(FP)
       POPR  SP            ; recover regs
       IRET  SP, =1
```

Oleta: SVC:n ja IRET:n toteutus samalla tavalla kuin CALL ja EXIT

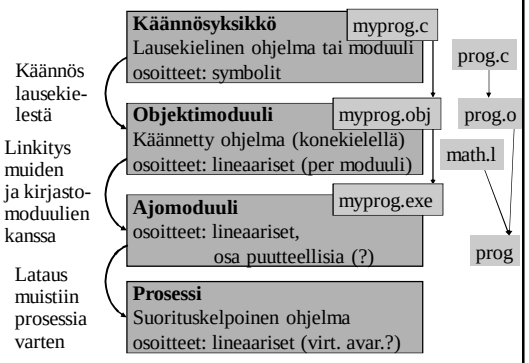


See: driver.k91

Luento 10
Käännös, linkitys ja lataus

Käännös
Linkitys
Dynaaminen linkitys
Lataus

Lausekielestä suoritukseen



Käännösyksikkö

- Jollain ohjelmointikielellä kuvattu eheä kokonaisuus, joka halutaan aina kääntää yhdessä
  - kaikki yhteen liittyvät aliohjelmat
  - olioperustainen luokka
- Liian suuri kokonaisuus?
  - turhaa aikaa kääntämiseen joka muutoksen jälkeen
- Liian pieni kokonaisuus?
  - turhaa aikaa liitoksien suunnitteluun ja toteutukseen muiden moduulien kanssa
- Käännösyksikön ohjelmointikieli ei ole tärkeä
  - niiden sitominen yhteen tapahtuu objektimoduulien tasolla

Assembler-kielinen käännösyksikkö

- Käännösyksikkö voi olla myös suoraan k.o. koneen symbolisella konekielellä kirjoitettu
  - suoraan käsin → prog.s
  - kääntäjän generoiman korkean tason kielestä (cc -S prog.c)
- Käännöksen tekee assembler-kääntäjä tavallisen kääntäjän asemesta
  - yleensä osa tavallista kääntäjää

prog.o

Objektimoduuli

- Konekielinen koodi
  - moduulin sisäiset viitteet paikallaan (lineaarisessa muistiavaruudessa)
  - moduulin ulkopuoliset viitteet merkitty
- Linkitystä varten:
  - tiedot niiden osoitteiden sijainneista, jotka täytyy päivittää, kun moduulin osoiteavaruus yhdistetään jonkin toisen moduulin osoiteavaruuden kanssa linkityksessä — RELOCATION TABLE
  - tiedot viittauksista moduulin ulkopuolelle — IMPORT
  - tiedot kohdista, joista tähän moduuliin saa viitata ulkopuolelta — EXPORT
  - symbolitaulu — SYMBOL TABLE

Symbolitaulu

- Kääntäjä generoi
- Ylläpidetään linkityksen aikana
- Joskus ylläpidetään myös latauksen jälkeen virheilmoitusten tekemistä varten
  - ohjelmien kehitysympäristöt ylläpitävät symbolitaulua koko ajan
- Jätetään pois valmiista ohjelmasta
  - vie turhaa tilaa

(Assembler) kääntäjän ohjauskäskyt

- Eivät varsinaista koodia
  - niistä ei tule konekäskyjä
- Ohjaavat käännöstä

TTK-91: DC
DS
EQU

Pentium 4: ks. Fig. 7-3 [Tane10]

Makrot (kesk.)

- Helpottavat ohjelmointia
- Usein toistuville koodisarjoille annetaan nimi ⇒ makro
- Makroilla voi olla parametreja
  - useimmiten nimiparametreja (call-by-name)
- Makrot käsitellään ennen kääntämistä
  - eivät kuulu konekieleen
  - makron ”kutsu” (käyttö) korvataan makron rungolla
- Esimerkkejä
  - swap
  - aliohjelmien prologi ja epilogi
  - itse tehdyt, kääntäjän käyttämät
- Erot aliohjelmiin ks. Fig. 7-5 [Tane10]
  - Kutsu ajankohta, call/return, koodien lukumäärä

Literaalit

- Vakioita
- Niin suuria, että eivät mahdu konekäskyn vakio-osaan ... ttk-91: käskyn vakiot 2-tavuisia, arvoalue: -32767 … 32767
- ... tai muuten vain halutaan pitää datan joukossa eikä käskyjen yhteyteen talletettuna

```
Pi      DC    3.14159265 ; (!!??)
One     DC    1          vrt. One EQU 1
OneMeg  DC    1024576
```

- Niitä ei saisi muuttaa

```
LOAD R1, =2
STORE R1, One  ; ask for trouble
```


Literaalit

- Korkean tason kielissä kaikki isot vakiot ovat literaaleja N := 35000; var myStr = ”literal"
  - kääntäjän pitäisi estää literaalien muuttamisen

FortranX: 5 = 6;

```
LOAD  R1, six
STORE R1, five
```

???

  - literaalia ei saisi välittää viiteparametrina
    - aliohjelma voisi muuttaa sen arvoa? Java string?
- Myös joissakin assemblerkielissä literaalien implisiittinen (automaattinen) määrittely
  - helpommin luettavaa koodia

```
Load  R14, =F’234567’
```

  - literaalin 234567 tilanvaraus automaattisesti

Assembler käännös

- 1. vaihe:
  - laske käskyjen tilanvaraukset
    - ttk-91 helppoa, koska kaikki käskyt 4 tavua!
  - generoi symbolitaulu
    - arvot, arvon vaatima tavumäärä
    - uudelleensijoitustiedot (omana tauluna?)
  - generoi tai käytä muita tauluja
    - literaalitaulu (tilanvaraus lopuksi)
    - kääntäjän ohjauskäskytaulu
    - operaatiokooditaulu

Assembler käännös (kesk.)

- 2. vaihe ks. seur. kalvo, Kuva 6.3 [Häkk98]
  - generoi lopullinen objektimoduuli ks. Fig. 7-16 [Tane10]
  - tulosta symbolinen konekielinen listaus
  - generoi taulut linkitystä varten
    - osana objektimoduulia
  - anna virheilmoitukset
- 3. vaihe
  - koodin optimointi
  - voi olla oikeasti ennen 2. vaihetta tai sen yhteydessä

TTK-91 objektimoduuli

(Kuva 6.3 [Häkk98])

Korkean tason kielen käännös

- Enemmän vaiheita
  - Syntaktisten alkioiden etsintä (front end)
    - Syntaksipuun generointi ja jäsennys
  - Lauseiden tunnistaminen syntaksipuun avulla
  - Välikielen (välikoodin) generointi (ei aina)
  - Koodin generointi (back end)
    - ei (yleensä) Java-ohjelmille

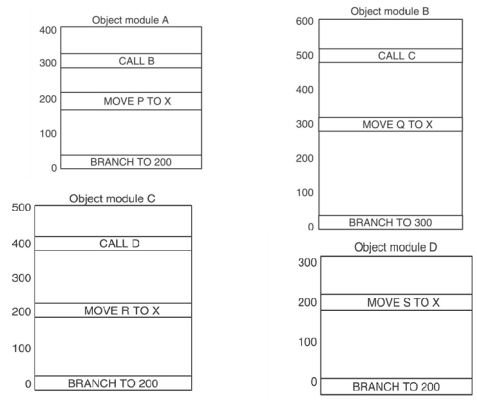
BEGIN 123.45 IF (

Välikieliesitys ja symbolitaulut

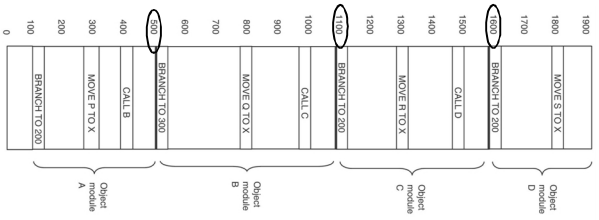
Lisää tietoa? Kääntäjien ja ohj. kielten kurssit

Linkitys esimerkki

- Neljä moduulia: A, B, C ja D (ks. Fig. 7-14 [Tane10])
- Laske joka moduulille *uudelleen-sijoitusvakio* (moduulin alkuosoite) (relocation constant)
- Lisää k.o. vakio kunkin moduulin sisäisiin viitteisiin (ks. Fig. 7-15 [Tane10])
- Etsi kaikki moduulien väliset viitteet, ja aseta kyseisten viitteiden osoitteet oikein



Tasks Performed by the Linker



The object modules of Fig. 7-14 after being positioned in the binary image but before being relocated and linked.

Tasks Performed by the Linker

The same object modules after linking and after relocation has been performed. Together they form an executable binary program, ready to run

Staattinen linkitys

- Tavallinen (staattinen) linkitys vaatii, että kaikki ohjelmakoodissa viitatut moduulit ja kirjastorutiinit on linkitetty ennen suoritusta
- Ajomoduulista tulee hyvin iso
  - mukana myös paljon moduuleja, joihin ei yhdellä suorituskerralla tule lainkaan viittauksia
    - esim: kääntäjässä koodin optimointikoodi, vaikka koodin optimointia ei suoriteta joka kerta
    - esim: pelissä tasojen 8-22 moduulit, kun aloittelija ei pääse tasoa 3 ylemmäksi vielä kuukausiin

Dynaaminen linkitys

- Jätetään linkityksessä kutsukohdat muihin moduuleihin auki
- Pienempi ajomoduuli, mutta hitaampi suorittaa
- Viittaus ”ratkaisemattomaan” (eli ei-linkitettyyn) moduuliin ratkotaan suoritusaikana
  - suoritus keskeytyy ja puuttuva moduuli linkitetään paikalleen (kaikki viittaukset siihen korjataan kuntoon)

Lataus

- Ajomoduulista luodaan suorituskelpoinen prosessi (rakennetaan PCB ja sen viitteet kuntoon)
- Prosessin koodialueet (tai ainakin sen pääohjelma) ja tarvittava data-alue ladataan muistiin, prosessi siirretään ready (Ready-to-Run) jonoon
- Sitten kun prosessi saa suoritusvuoron suorittimella, MMU ja laiterekisterit ladataan PCB:n avulla tämän prosessin tiedoilla
  - virtuaalimuistia käytettäessä joidenkin nimien sidonta tehdään viime hetkellä (konekäskyn suoritusaikana) MMU:n avulla

Yhteenveto

- Muistihierarkia
- Kiintolevyt ja niiden toiminta
- Tiedostojärjestelmä
- I/O:n toteutus
- Käännös, linkitys ja lataus