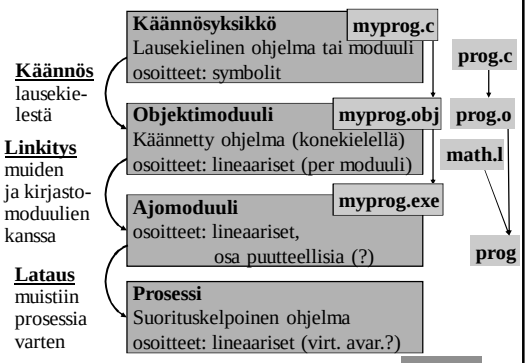
Luento 9 (verkkoluento 10)

# Käännös, linkitys ja lataus

Ohjelmasta prosessiin
Käännösyksikkö
Kääntämisen vaiheet
Makrot, literaalit
Staattinen ja dynaaminen linkitys
Nimien sidonta

## Lausekielestä suoritukseen




15.11.2012 Copyright 2012 Teemu Kerola Keskustele 2

## 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 moduulin kohdista, joihin saa viitata ulkopuolelta **EXPORT**
  - symbolitaulu **SYMBOL TABLE**

15.11.2012 Copyright 2012 Teemu Kerola 3

## 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

15.11.2012 Copyright 2012 Teemu Kerola 4

## Makrot

- Helpottavat ohjelmointia, toistuva koodisarja
- Voivat sisältää parametreja
  - useimmiten nimiparametreja (call-by-name)
- Käsitellään ennen kääntämistä
  - eivät kuulu konekieleen
  - makron ”kutsu” (käyttö) korvataan makron rungolla
- Esimerkkejä
  - swap
  - aliohjelmien prologi ja epilogi
  - itse tehdyt, kääntäjän käyttämät
- Erot aliohjelmiin
  - Kutsu ajankohta, call/return, koodien lukumäärä

15.11.2012 Copyright 2012 Teemu Kerola Keskustele 5

## Literaalit

- Korkean tason kielissä kaikki isot vakiot ovat literaaleja **N := 35000;** **var myStr = ”literal"**
  - kääntäjän pitäisi estää literaalien muuttaminen
    **FortranX: 5 = 6;** **LOAD R1, six**
    **STORE R1, five** **???**
  - literaalia ei saisi välittää viiteparametrina
    - aliohjelma voisi muuttaa sen arvoa? **Java string?**
- Joissakin symb. konekielissä literaalien implisiittinen (automaattinen) määrittely
  - helpommin luettavaa koodia
  - literaalin 234567 tilanvaraus automaattisesti
    **Load R14, =F’234567’**

15.11.2012 Copyright 2012 Teemu Kerola 6

## Assembler käännös

- 1. vaihe:
  - laske käskyjen tilanvaraukset
  - generoi symbolitaulu, muut taulut
- 2. vaihe
  - generoi lopullinen objektimoduuli
  - tulosta symbolinen konekielinen listaus
  - generoi taulut linkitystä varten
  - anna virheilmoitukset
- 3. vaihe
  - koodin optimointi
  - voi olla oikeasti ennen 2. vaihetta tai sen yhteydessä



## 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

- *Uudelleensijoitusvakio*
  - A: 0, B: L, C: L+M
- Lisää vakio kunkin moduulin sisäisiin viitteisiin
- Moduulien väliset viitteet oikein

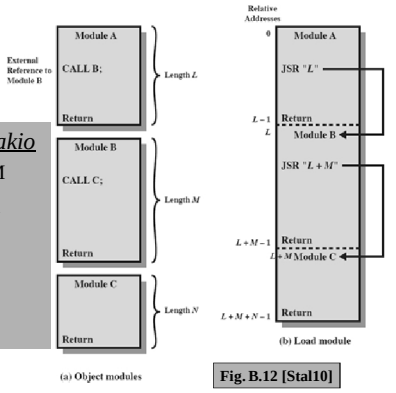



Fig. B.12 [Stal10]



## Staattinen ja dynaaminen linkitys

- Staattinen linkitys
  - Kaikki ohjelmakoodissa viitatut moduulit ja kirjastorutiinit on linkitetty ennen suoritusta
  - Iso ajomoduuli
    - mukana moduuleja, joihin ei yhdellä suorituskerralla tule lainkaan viittauksia
- Dynaaminen linkitys
  - Kutsukohdat muihin moduuleihin jätetään 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 koodi- ja data-alueet ladataan muistiin, prosessi siirretään ready (Ready-to-Run) jonoon
- Eri tyyppejä
  - Absoluuttinen – aina samaan paikkaan muistia
  - Uudelleensijoitettava – joustava paikan valinta
  - Dynaaminen ajoaikainen – paikat vaihtelevat ajoaikana



## -- Luennon -- -- loppu --

- Transistori
  - J. Bardeen, W.B. Shockley ja W. Brattain, Bell Labs, 1948 (Nobel 1956)
  - TX-0, MIT, 1956
- Integroitu piiri (ei enää johtoja)
  - Jack Kilby, Texas Instruments, 1958 (Nobel 2000)
  - Robert Noyce, Fairchild Semiconductor, 1959
  - IBM S/360, 1964