

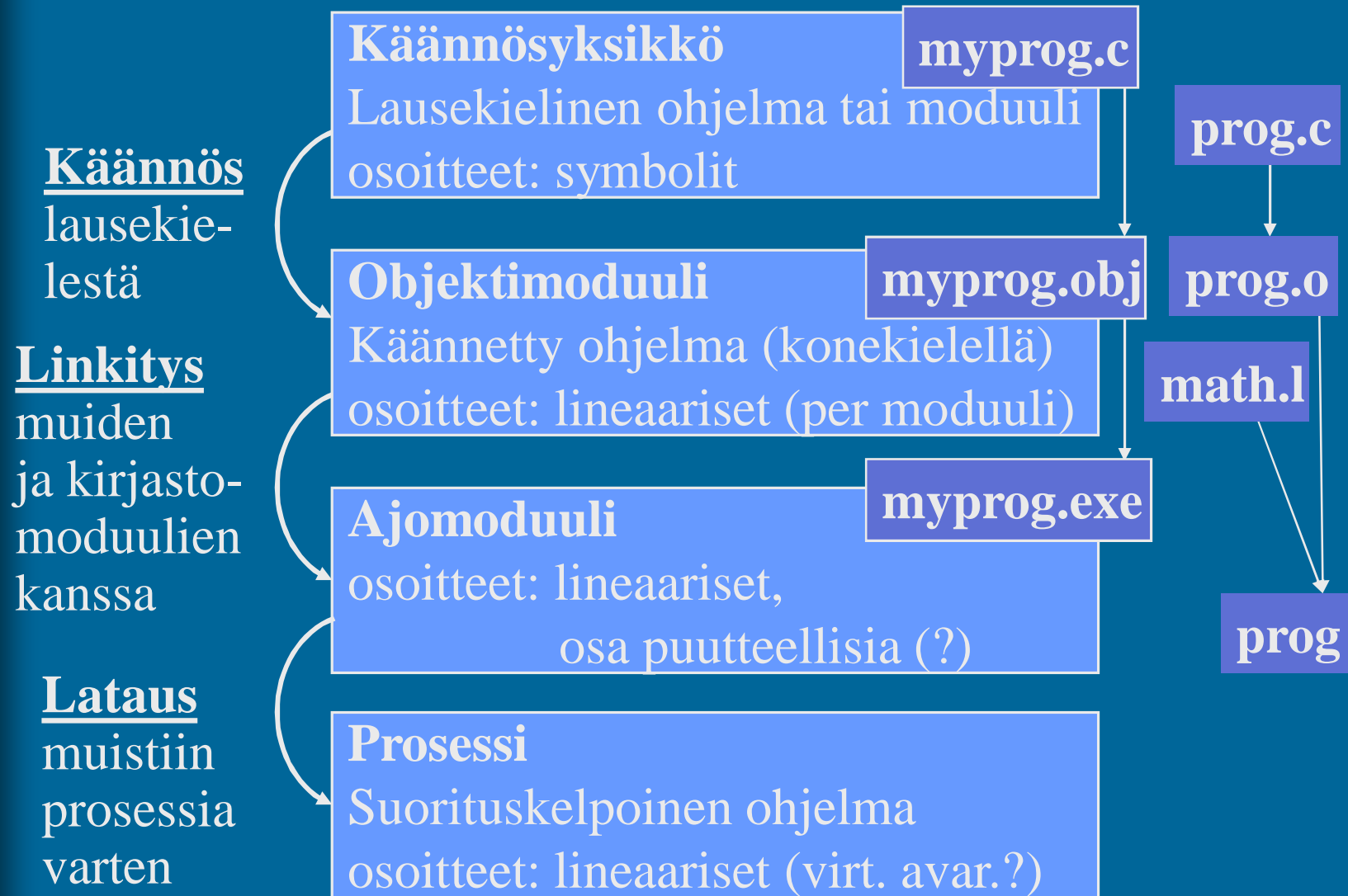
Luento 10 (verkkoluento 10)

Käännös, linkitys ja lataus



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

Lausekielestä suoritukseen



Objektimoduuli

- Konekielinen koodi
 - moduulin sisäiset viitteet paikallaan (linearisessa muistiavaruudessa)
 - moduulin ulkopuoliset viitteet merkitty
- Linkitystä varten:
 - Uudelleensijoitustaulu. 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
 - Symbolitaulu **SYMBOL TABLE** **EXPORT**

Symbolitaulu

- Mikä arvo kullakin symbolilla on?
 - Staattinen arvo voi olla (myös) muistiosoite
- 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, ei tarvita normaalisuorituksessa

Makro

- Usein toistuva koodisarja, helpottaa ohjelmointia
- Voi sisältää parametreja
 - 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

Literaalit

- Korkean tason kielissä kaikki isot vakiot ovat literaaleja

```
tmp1 dc 35000
load r1, tmp1
store r1, N
```

```
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 symbolisissa konekielissä literaalien implisiittinen (automaattinen) määrittely

- helpommin kirjoitettavaa/luettavaa koodia
- literaalin 234567 tilanvaraus automaattisesti

```
Load R14, =F'234567'
```

Symbolin F'234567' määrittely:
F'234567' ≡ "sen muistipaikan osoite,
jonka arvo on 234567"

Assembler käänös

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

Korkean tason kielen käännös

- Enemmän vaiheita

BEGIN

123.45

IF

(

- Syntaktisten alkioiden etsintä

(front end)

- Syntaksipuun generointi ja jäsenitys

- Lauseiden tunnistaminen syntaksipuun avulla

- Välikielen (välikoodin) generointi (ei aina)

P-code, bytecode, ...

Välikieliesitys ja symbolitaulut

- Koodin optimointi

- Koodin generointi

(back end)

- ei (yleensä) Java-ohjelmille

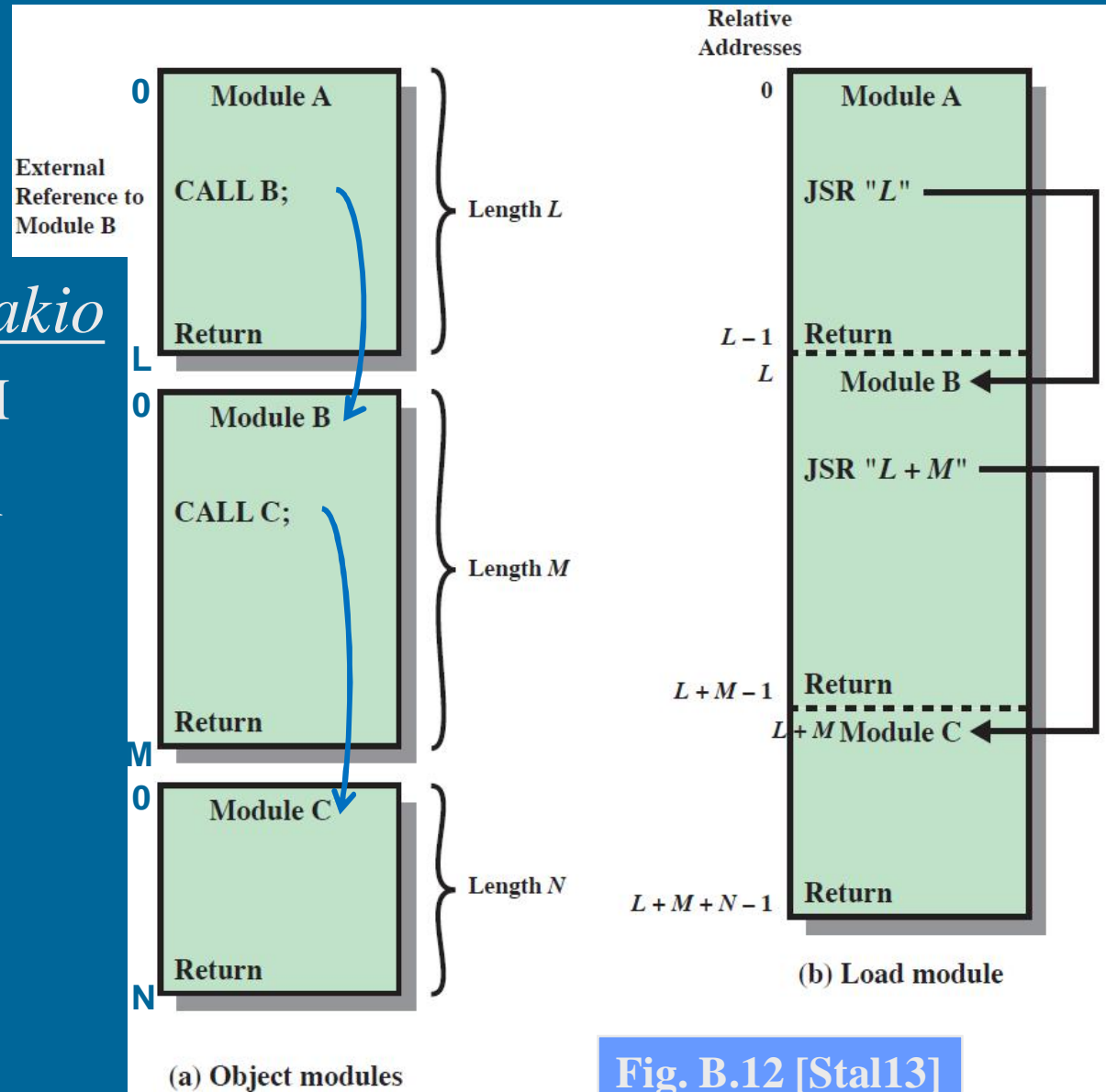
Lisää tietoa?



Kääntäjien ja ohj. kielten kurssit

Linkitys

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



Staattinen ja dynaaminen linkitys

- Staattinen linkitys
 - Kaikki ohjelmakoodissa viitatus moduulit ja kirjastorutiinit on linkitetty ennen suoritusta
 - Iso ajomoduuli
 - mukana moduuleja, joihin ei yhdellä suorituskerralla tule lainkaan viittauksia
- Dynaaminen linkitys
 - Kutsukohtat 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)

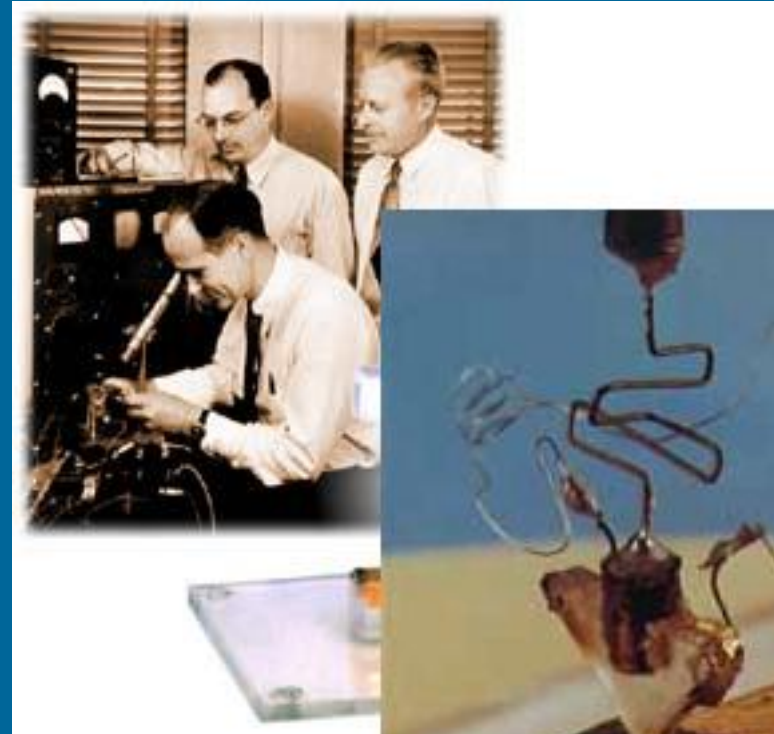
Lataus

- Ajomodulistista luodaan suorituskelpoinen prosessi (rakennetaan PCB ja sen viitteet kuntoon)
- Prosessin koodi- ja data-alueet ladataan muistiin, prosessi siirretään jonoon Valmis suoritukseen (Ready, Ready-to-Run)
- Eri tyyppejä
 - Absoluuttinen – aina samaan paikkaan muistia
 - Uudelleensijoitettava – joustava sijainnin valinta muistissa
 - Milloin ja miten osoitteet muutetaan?
 - Dynaaminen ajoaikainen – muistisijainti vaihtelee ajoaikana
 - Milloin ja miten osoitteet muutetaan?

-- Loppu --

Nobel
1956

- Transistori
 - J. Bardeen, W.B. Shockley ja W. Brattain, Bell Labs, 1948
 - TX-0, MIT, 1956
 - 1900-luvun tärkeimpiä keksintöjä maailmassa



Nobel
2000

- Integroitu piiri (ei enää johtoja)
 - Jack Kilby, Texas Instruments, 1958
 - Robert Noyce, Fairchild Semiconductor, 1959
 - IBM S/360, 1964

