

# Luento 10

## Käännös, linkitys ja lataus

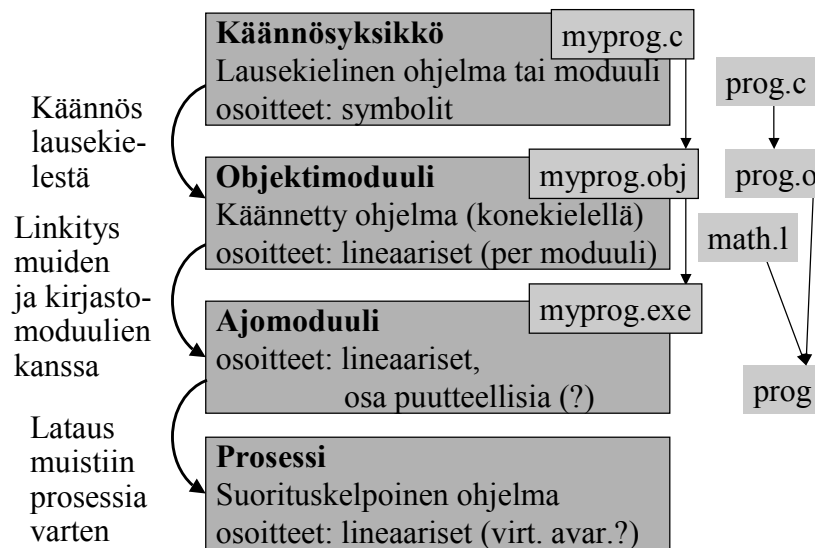
Käännös  
Linkitys  
Dynaaminen linkitys  
Lataus

10/01/2002

Teemu Kerola, Copyright 2002

1

### Lausekielestä suoritukseen (3)



10/01/2002

Teemu Kerola, Copyright 2002

2

## Käännösyksikkö <sup>(4)</sup>

- 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 murehtia ja toteuttaa liitoksia muiden moduulien kanssa
- Käännösyksikön ohjelmointikieli ei ole tärkeä
  - niiden sitominen yhteen tapahtuu objektimoduulien tasolla

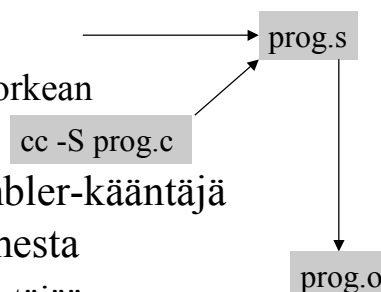
10/01/2002

Teemu Kerola, Copyright 2002

3

## Assembler-kielinen käännösyksikkö <sup>(2)</sup>

- Käännösyksikkö voi olla myös suoraan k.o. koneen symbolisella konekielellä kirjoitettu
  - suoraan käsin
  - kääntäjän generoimana korkean tason kielestä
- Käännöksen tekee assembler-kääntäjä tavallisen kääntäjän asemesta
  - yleensä osa tavallista kääntäjää



10/01/2002

Teemu Kerola, Copyright 2002

4

## Objektimoduuli <sup>(8)</sup>

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

10/01/2002

Teemu Kerola, Copyright 2002

5

## Symbolitaulu <sup>(4)</sup>

- 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

10/01/2002

Teemu Kerola, Copyright 2002

6

## Lähdekieli vs. konekieli <sup>(3)</sup>

- Pascal lauseke:  $N := I+J;$
- C lauseke:  $N = I+J$
- Java lauseke:  $N = I+J;$

TTK-91 symbolinen  
konekieli:

I	DC	3
J	DC	4
N	DC	0
FORMULA LOAD R1, I		
ADD R1, J		
STORE R1, N		

Pentium II, Motorola 680x0 ja  
SPARC symbolinen konekieli:

ks. Fig. 7-2 [Tane99]

10/01/2002

Teemu Kerola, Copyright 2002

7

## (Assembler) kääntäjän ohjauskäskyt <sup>(4)</sup>

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

TTK-91: DC  
DS  
EQU

Pentium II:

ks. Fig. 7-3 [Tane99]

10/01/2002

Teemu Kerola, Copyright 2002

8

## Makrot <sup>(6)</sup>

- 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” korvataan makron rungolla
- Esimerkkejä
  - swap ks. Fig. 7-4 ja 7-6 [Tane99]
  - aliohjelmien prologi ja epilogi
  - itse tehdyt, kääntäjän käyttämät
- Erot aliohjelmiin ks. Fig. 7-5 [Tane99]

10/01/2002

Teemu Kerola, Copyright 2002

9

## Literaalit <sup>(5)</sup>

- 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
OneMeg	DC	1024576
- Niitä ei saisi muuttaa
 

LOAD R1, One
ADD R1,=1
STORE R1, One ; ask for trouble

10/01/2002

Teemu Kerola, Copyright 2002

10

## Literaalit <sup>(2)</sup>

- Korkean tason kielissä kaikki isot vakiot ovat literaaleja `N := 35000;` `var myStr = "literal"`
  - kääntäjän pitäisi estää literaalien muuttamisen
    - `FortranX: 5 = 6;` `???????`
  - 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

10/01/2002

Teemu Kerola, Copyright 2002

11

## Assembler käännös <sup>(10)</sup>

- 1. vaihe:
  - laske käskyjen tilanvaraukset
    - ttk-91 helppoa, koska kaikki käskyt 4 tavua!
  - generoi symbolitaulu `ks. Kuva 6.2 [Häkk98]`
    - arvot, arvon vaatima tavumäärä
    - uudelleensijoitustiedot (omana tauluna?)
  - generoi tai käytä muita tauluja
    - literaalitaulu (tilanvaraus loppuksi)
    - kääntäjän ohjauskäskytaulu
    - operaatiokooditaulu

10/01/2002

Teemu Kerola, Copyright 2002

12

## Assembler käänös <sup>(8)</sup>

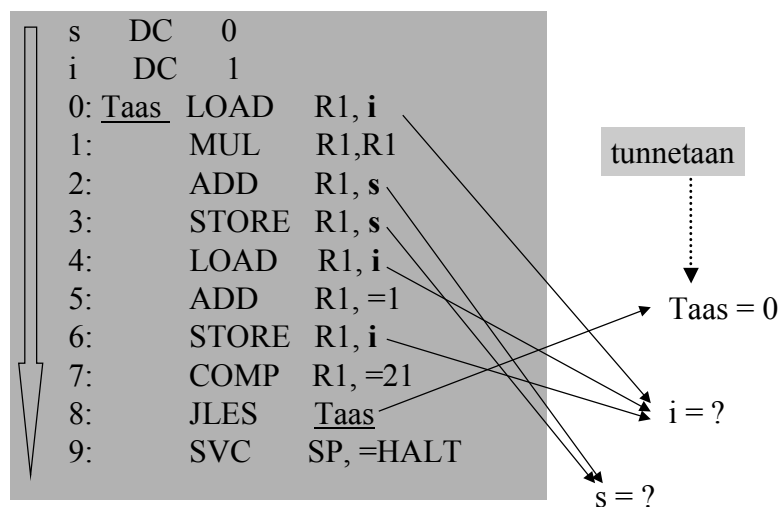
- 2. vaihe
  - generoi lopullinen objektimoduuli ks. Kuva 6.3 [Häkk98]
  - tulosta symbolinen konekielinen listaus ks. Fig. 7-16 [Tane99]
  - generoi taulut linkitystä varten
    - osana objektimoduulia
  - anna virheilmoitukset
- 3. vaihe
  - koodin optimointi
  - voi olla oikeasti ennen 2. vaihetta tai sen yhteydessä

10/01/2002

Teemu Kerola, Copyright 2002

13

## TTK-91 Assembler käänös, 1. vaihe



10/01/2002

Teemu Kerola, Copyright 2002

14

Symbolitaulu 1. vaiheen aikana ks. kalvo 14

Symboli	tyyppi	arvo	uud. sij.tietoa
s	data	?	2, 3
i	data	?	0, 4, 6
Taas	viite	0	8
HALT	vakio	11	

	OPER	Rj	M	Ri	ADDR
2:	ADD	1	1	0	?
	....				
8:	JLES	0	0	0	0

Konekäsyt  
2 ja 8

10/01/2002 Teemu Kerola, Copyright 2002 15

Koodi & data 1. vaiheen jälkeen

s	DC	0	
i	DC	1	
0:	Taas	LOAD	R1, i
1:		MUL	R1, R1
2:		ADD	R1, s
3:		STORE	R1, s
4:		LOAD	R1, i
5:		ADD	R1, =1
6:		STORE	R1, i
7:		COMP	R1, =21
8:		JLES	Taas
9:		SVC	SP, =HALT
10:	0		; siis s = 10
11:	1		; i = 11

Kaikilla  
symboleilla  
tunnettu arvo

10/01/2002 Teemu Kerola, Copyright 2002 16



Symbolitaulu 1. vaiheen jälkeen ks. kalvo 16

Symboli	tyyppi	arvo	uud. sij.tietoa
s	data	10	2, 3
i	data	11	0, 4, 6
Taas	viite	0	8
HALT	vakio	11	

	OPER	Rj	M	Ri	ADDR
2:	ADD	1	1	0	10
	....				
8:	JLES	0	0	0	0

10/01/2002 Teemu Kerola, Copyright 2002 17

## TTK-91 objektimoduuli

Moduulin otsake	( Kuva 6.3 [Häkk98] )
EXPORT-hakemisto	
IMPORT-hakemisto	
Uudelleensijoitushakemisto	
Koodi ja alustettu data	
Moduulin lopuke	

10/01/2002 Teemu Kerola, Copyright 2002 18

## TTK-91 objektimoduuli

- Moduulin otsakeosa
  - moduulin nimi
  - linkittäjän tarvitsemia tietoja
    - objektimoduulin osien pituudet
    - käänös päivämäärä
    - kääntäjän nimi ja versio
    - ensimmäisen suoritettavan käskyn osoite
      - ellei aina 0

10/01/2002

Teemu Kerola, Copyright 2002

19

## TTK-91 objektimoduuli

- EXPORT-hakemisto
  - tunnukset, joihin voidaan viitata muista moduuleista
    - rutiinit, aliohjelmat, (oliot, metodit)
    - yhteiskäyttöinen data
  - tunnuksen osoite (= symbolin arvo)
  - mahdollinen käyttöoikeus
    - R/W/E/RW

10/01/2002

Teemu Kerola, Copyright 2002

20

## TTK-91 objektimoduuli

- **IMPORT-hakemisto**
  - muissa moduuleissa määritellyt tunnukset
    - tunnus
    - niiden käskyjen osoitteet, jossa tunnus esiintyy
- **Koodi ja alustettu data**
  - alustamattomille muuttujille ei tarvitse varata (vielä) tilaa, mutta ne on otettava huomioon data-alueen koossa

10/01/2002

Teemu Kerola, Copyright 2002

21

## TTK-91 objektimoduuli

- **Uudelleensijoitushakemisto**
  - niiden käskyjen osoitteet, joiden osoiteosaa on korjattava, kun siirrytään moduulien yhteiseen osoitevaruuteen
    - suoraviivainen lisäys (joka käskyyn) ei toimi, sillä käskyn osoiteosa voi olla vakio, jota ei saa muuttaa
    - erikseen (a) paikalliseen dataan viittaavat ja (b) hyppykäskyt, sillä linkittäessä yhdistetään erikseen data- ja koodialueet

10/01/2002

Teemu Kerola, Copyright 2002

22

## Korkean tason kielen käännös <sup>(7)</sup>

- Enemmän vaiheita

- Syntaktisten alkioiden etsintä

- Syntaksipuun generointi ja jäsenys

- Lauseiden tunnistaminen syntaksipuun avulla

- Välikielen (välikoodin) generointi (ei aina)

- Koodin generointi

- ei (yleensä) Java-ohjelmille

BEGIN

123.45

IF

(

(front end)

Välikieliesitys ja symbolitaulut

(back end)

Lisää  
tietoa?Kääntäjien  
ja ohj. kielten  
kurssitks. syntaksipuun,  
jäsenyspuun  
esimerkit

10/01/2002

Teemu Kerola, Copyright 2002

23

## Linkitys

- Uudelleensijoitusongelma

(relocation problem)

- jokaisen objektimoduulin osoitteet alkavat 0:sta

- tulosmoduulissa kaikki yhdessä lineaarisessa osoitevaruudessa

- useimpien moduulien kaikkia osoitteita täytyy muuttaa

- käskyjen osoitteet
- datan osoitteet

10/01/2002

Teemu Kerola, Copyright 2002

24

## Linkitys esimerkki <sup>(4)</sup>

- Neljä moduulia: A, B, C ja D ks. Fig. 7-14 [Tane99]
- Laske joka moduulille *uudelleen-*  
*sijoitusvakio* (moduulin alkuosoite) (relocation  
constant)
- Lisää k.o. vakio kunkin moduulin sisäisiin viitteisiin
- Etsi kaikki moduulien väliset viitteet, ks. Fig. 7-15 (a) [Tane99]  
ja aseta kyseisten viitteiden osoitteet oikein ks. Fig. 7-15 (b) [Tane99]

10/01/2002

Teemu Kerola, Copyright 2002

25

## Muuttujan X viittausten päivitys <sup>(6)</sup>

- Miten löytää linkityksen aikana kaikki kohdat, jossa muuttujaan X viitataan?
- Vastaus 1: iso taulukko, jossa kaikki kohdat listattu
- Vastaus 2: Muuttujan X viittaukset on kaikki linkitetty keskenään linkitetyksi listaksi objektimoduulissa
  - vain linkitetyn listan alkuosoite taulukossa
  - X:n osoitteen paikalla aluksi linkki seuraavaan käskyyn, missä X:ään viitataan
  - listan voi käyttää vain yhden kerran?

10/01/2002

Teemu Kerola, Copyright 2002

26

## Muuttujan X viitaukset linkitettyinä listana <sup>(1)</sup>

lähdekoodi			symbolitaulu, moduuli ABC			objektimoduuli		
23:	Load	R1, X	Symb	sij	1. viittaus	23:	Load	1 0 34
...	...	...	X	700	23	...	...	...
34	Store	R3, X(R1)				34:	Store	3 1 555
...	...	...				...	...	...
555	Add	R4, X				555:	Add	4 0 -1
...	...	...						
700	DC	0 ; X						

10/01/2002

Teemu Kerola, Copyright 2002

27

## Staattinen linkitys <sup>(5)</sup>

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

10/01/2002

Teemu Kerola, Copyright 2002

28

## Dynaaminen linkitys <sup>(4)</sup>

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

10/01/2002

Teemu Kerola, Copyright 2002

29

## Windows DLL <sup>(4)</sup>

- DLL - Dynamically Linked Library
  - koodia, dataa, molempia
- Säästää tilaa myös yhteiskäytön vuoksi
 

.dll	yleinen tapaus
.drv	driver
.fon	font
- Helpompi korjata virheitä
  - ei tarvita uutta käännettä!
  - riittää kun DLL vaihdetaan uuteen
  - seuraavassa suorituksessa uusi versio käyttöön
- Ajomoduuli kootaan kuten tavallinen objektimoduuli
  - DLL moduulit merkitty erikoislipukkeella (huomioidaan linkityksen yhteydessä)

ks. Fig. 7.19 [Tane99]

10/01/2002

Teemu Kerola, Copyright 2002

30

## Windows DLL:n linkityksen kaksi tapaa <sup>(3)</sup>

- Epäsuora dynaaminen linkitys (implicit linking)
  - kaikki viitatus moduulit ladataan (lataus aloitetaan) virtuaalimuistiin ja niihin viitataan staattisesti linkitetyn pienemmän liitospalikan (import library) avulla
- Suora dynaaminen linkitys (explicit linking)
  - koodiin generoidaan suoraan viitepaikalle käskyt, joiden avulla linkitys tapahtuu tarvittaessa
  - DLL ladataan vain jos siihen tulee viittaus
- DLL:ssä oleva koodi suoritetaan osana kutsuvaa prosessia käyttäen sen omaa aktivointitietuepinoa

10/01/2002

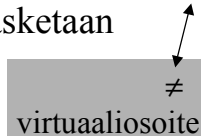
Teemu Kerola, Copyright 2002

31

## Nimien sidonta <sup>(2)</sup>

(name binding)

- Milloin symbolin L suoritusaikainen muistiosoite tai muu lopullinen arvo sidotaan (lasketaan valmiiksi)?
  - ohjelman kirjoitusaikana?
  - käännösaikana?
  - linkityksessä?
  - latauksessa?
  - kantarekisterin asetuksen aikana?
  - osoitteen sisältämän konekäskyn suoritusaikana?
- Jos sijaintipaikkaa siirretään sitomisen jälkeen, mennään metsään ...


 virtuaaliosoite

10/01/2002

Teemu Kerola, Copyright 2002

32



## Sijainnista riippumaton koodi <sup>(3)</sup>

(position independent code)

- Jos koodi siirretään toiseen paikkaan, niin mitään osoitetta ei tarvitse päivittää
- Kaikki muistiviittaukset ovat
  - absoluuttisia (esim. keskeytys käsittelijän osoite),
  - suhteessa PC:hen, tai
  - pinossa
- Siellä ei ole viittauksia mihinkään koodiin tai tietorakenteeseen suorien (fyysisten) muistiosoitteiden avulla

10/01/2002

Teemu Kerola, Copyright 2002

33

## Lataus <sup>(4)</sup>

- Ajomoduulista luodaan suorituskelpoinen prosessi (rakennetaan PCB ja sen viitteet kuntoon)
- Prosessin koodialueet (tai ainakin sen pääohjelma) ja tarvittava data-alue ladataan muistiin, prosessi siirretään R-to-R 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

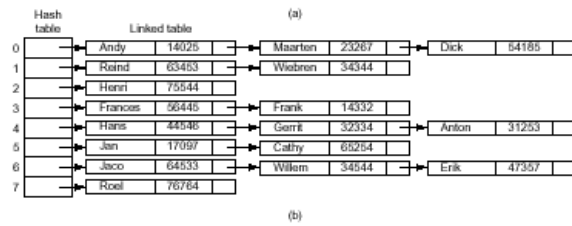
10/01/2002

Teemu Kerola, Copyright 2002

34

## -- Luennon 10 loppu --

Andy	14025	0
Anton	31253	4
Cathy	65254	5
Dick	54185	0
Erik	47357	6
Francois	56445	3
Frank	14332	3
Gerrit	52334	4
Hans	44546	4
Henri	75544	2
Jan	17097	5
Jaco	64533	6
Maarten	23267	0
Raimd	63453	1
Roel	76764	7
Willem	34544	6
Wietren	34344	1



**Figure 7-12.** Hash coding. (a) Symbols, values, and the hash codes derived from the symbols. (b) Eight-entry hash table with linked lists of symbols and values.

[Tane99]

10/01/2002

Teemu Kerola, Copyright 2002

35