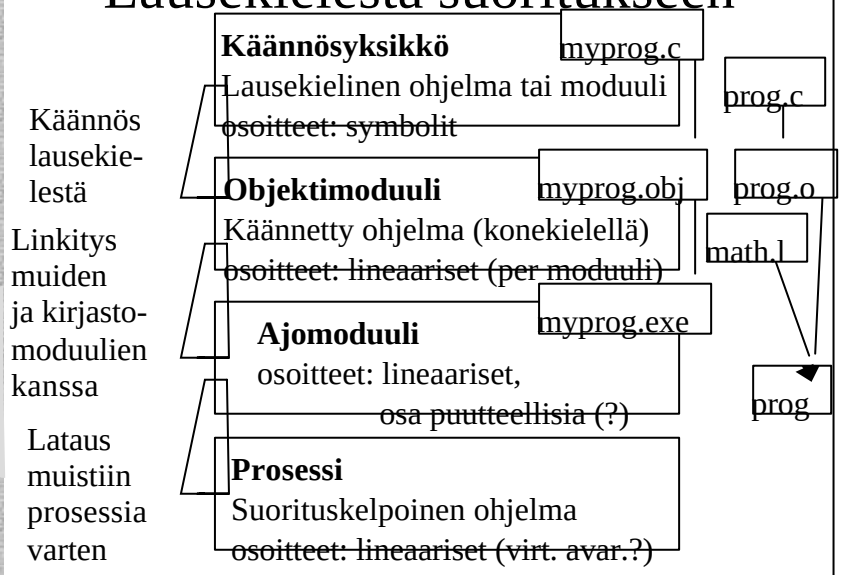
Jakso 10
Ohjelman suoritus järjestelmässä

Käännös
Linkitys
Dynaaminen linkitys
Lataus

Lausekielestä suoritukseen



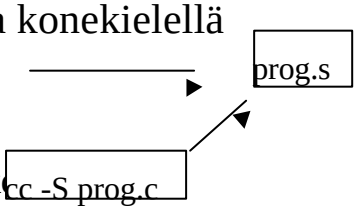
Käännösyksikkö (4)

- 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

31.5.2001 Teemu Kerola, K2000 3

Assembler-kielinen käännösyksikkö

- 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

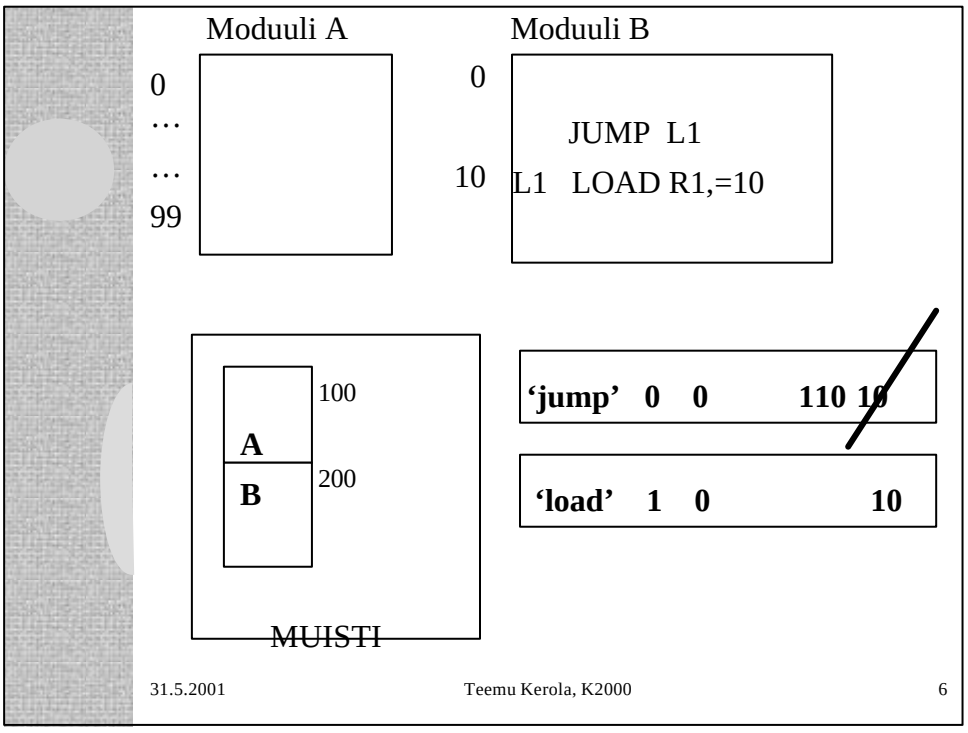
31.5.2001 Teemu Kerola, K2000 4

Objektimoduuli (8)

- Konekielinen koodi
  - moduulin sisäiset viitteet paikallaan (lineaarisessa muistiavaruudessa)
  - moduulin ulkopuoliset viitteet merkitty
- Linkitystä varten tiedot
  - tiedot niiden osoitteiden sijainnista, jotka täytyy päivittää, jos moduulin sijainti muistissa vaihtuu (suorat muistiosoitteet, joihin ei käytetä virtuaalista osoitteenmuunnosta)
  - tiedot viittauksista moduulin ulkopuolelle
  - tiedot kohdista joista tähän moduuliin saa viitata ulkopuolelta
  - symbolitaulu

IMPORT

EXPORT



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

Lähdekielinen ohjelma (5)

- 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]

(Assembler) kääntäjän ohjauskäskyt (4)

- Eivät varsinaista koodia
- Ohjaavat käännöstä

TTK-91: DC
DS
EQU

Pentium II: ks. Fig. 7-3 [Tane99]

31.5.2001 Teemu Kerola, K2000 9

Makrot (6)

- Helpottavat ohjelmointia
- Usein toistuville koodisarjoille annetaan nimi makro
- Makroilla voi olla parame ks. Fig. 7-4 ja 7-6 [Tane99]
- Esimerkkejä
  - swap
  - aliohjelmien prologi ja epilogi
  - itse tehdyt, kääntäjän käyttämät
- Makrot käsitellään ennen kä ks. Fig. 7-5 [Tane99]
- Erot aliohjelmiin

31.5.2001 Teemu Kerola, K2000 10

Literaalit (5)

- Vakioita
- Niin suuria, että eivät mahdu konekäskyn vakio-osaan

ttk-91: käskyn vakiot 2-tavuisia, arvoalue: -128…127

- Halutaan pitää datan joukossa eikä käskyjen yhteyteen talletettuna
- Niitä ei saisi muuttaa

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

```
LOADR1, One
ADD   R1,=1
STORE R1, One  ; ask for trouble
```


Literaalit

- Korkean tason kielissä kaikki isot vakiot aina literaaleja
  - kääntäjän pitäis[illegible]n m

```
FortranX:  5 = 6;
```

???????

  - literaalia ei saisi välittää viiteparametrina
    - aliohjelma voisi muuttaa sen arvoa?
- Myöskään joissakin assemblerkielissä literaaleja ei tarvitse er[illegible]

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

  - helpommin luettavaa koodia
  - vakion 234567 tilanvaraus automaattisesti

Assembler käännös (10)

- 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

ks. Kuva 6.2 [Häkk98]

31. toukokuuta 2001 13

Assembler käännös (7)

- 2. Vaihe
  - generoi lopullinen objektimoduuli
  - tulosta symbolinen assembler -listaus
  - generoi taulut linkitystä varten
    - osana objektimoduulia
  - anna virheilmoitukset
- 3. Vaihe
  - koodin optimointi

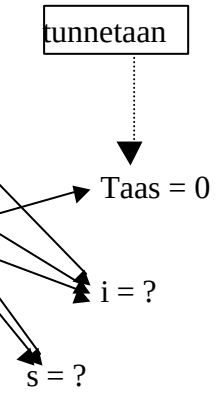
ks. Kuva 6.3 [Häkk98]

ks. Fig. 7-16 [Tane99]

31.5.2001 Teemu Kerola, K2000 14

TTK-91 Assembler käännös

```
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
```



15

Symbolitaulu 1. vaiheen aikana

ks. kalvo 15

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

Kone-käskyt 2 ja 8

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

31.5.2001 Teemu Kerola, K2000 16

Koodi & data 1. vaiheen jälkeen

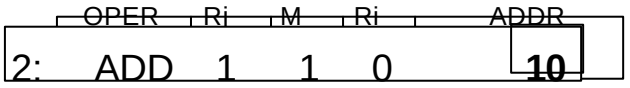
```
s     DC     0
i     DC     1

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

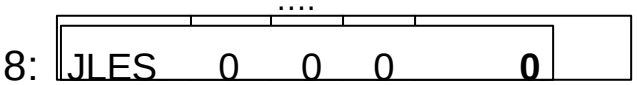
Symbolitaulu 1. vaiheen jälkeen

ks. kalvo 17

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



….



TTK-91 objektimoduuli

(Kuva 6.3 [Häkk98])

TTK-91 objektimoduuli

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

TTK-91 objektimoduuli

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

TTK-91 objektimoduuli

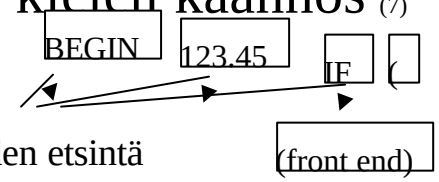
- IMPORT-hakemisto
  - muissa moduuleissa määritellyt tunnukset
    - tunnus
    - niiden käskyjen osoitteet, jossa tunnus esiintyy
- Koodi ja alustettu data
  - alustamattomille muuttujille ei tarvitse varata tilaa
    - otettava huomioon data-alueen koossa

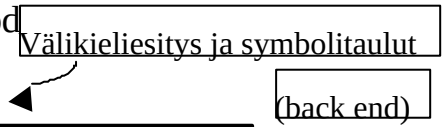
TTK-91 objektimoduuli

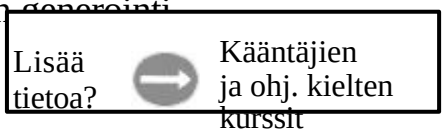
- Uudelleensijoitushakemisto
  - niiden käskyjen osoitteet, joiden osoiteosaa on korjattava, kun siirrytään moduulien yhteiseen osoiteavaruuteen
    - suoraviivainen lisäys ei toimi, sillä käskyn osoiteosa voi olla vakio, jota ei saa muuttaa
    - erikseen paikalliseen dataan viittaavat ja hyppykäskyt,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

Korkean tason kielen käännös (7)

- Enemmän vaiheita
  - Syntaktisten alkioiden etsintä
    - Syntaksipuun generointi ja jäsennys
  - Lauseiden tunnistaminen syntaksipuun avulla
  - Välikielen (välikoodi)
  - Koodin generointi







Linkitys

- Uudelleensijoitusongelma (relocation problem)
  - jokaisen objektimoduulin osoitteet alkavat 0:sta
  - tulosmoduulissa kaikki yhdessä lineaarisessa osoiteavaruudessa
  - useimpien moduulien kaikkia osoitteita täytyy muuttaa
    - käskyjen osoitteet
    - datan osoitteet

31.5.2001 Teemu Kerola, K2000 25

Linkitys esimerkki

ks. Fig. 7-14 [Tane99]

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

ks. Fig. 7-15 (a) [Tane99]

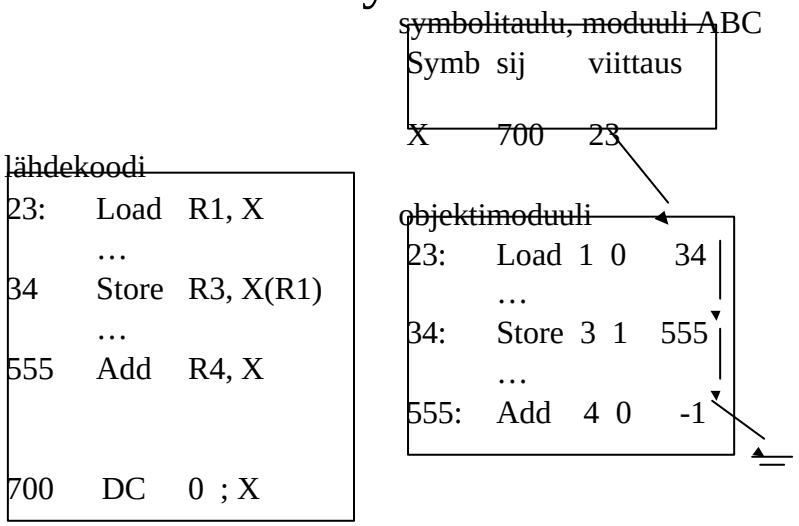
ks. Fig. 7-15 (b) [Tane99]

31.5.2001 Teemu Kerola, K2000 26

Muuttujan X viittausten päivitys (3)

- Miten löytää 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 alkuosoite taulukossa

Muuttujan X viitaukset linkitettynä listana



Staattinen linkitys (5)

- 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
    - kääntäjässä koodin optimointikoodi, vaikka optimointia ei suoriteta
    - pelissä tasojen 8-22 moduulit, kun aloittelija ei pääse tasoa 3 ylemmäksi vielä kuukausiin

31.5.2001 Teemu Kerola, K2000 29

Dynaaminen linkitys (4)

- 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)

31.5.2001 Teemu Kerola, K2000 30

Lataus (4)

- Ajomoduulista luodaan suorituskelpoinen prosessi (rakennetaan PCB ja sen viitteet kuntoon)
- Prosessin koodialueet (tai ainakin sen pääohjelma) ja tarvittava ladataan muistiin, prosessi siirretään R-to-R jonoon
- Sitten kun prosessi saa suoritusvuoron suorittimella, MMU ladataan PCB:n avulla tämän prosessin tiedoilla

Windows DLL

- DLL - Dynamically Linked Library
  - koodia, dataa, molempia
- Säästää tilaa myös yhteiskäytön vuoksi
- Helpompi korjata virheitä
  - ei tarvita uutta käännöstä!
  - riittää kun DLL vaihdetaan uuteen
- Kootaan kuten tavallinen objektimoduuli
  - erikoislipuke merkitsee sen DLL:ksi (huomioidaan linkityksen yhteydessä)

.dll	yleinen tapaus
.drv	driver
.fon	font

ks. Fig. 7.19 [Tane99]

Windows DLL:n linkityksen kaksi tapaa (3)

(implicit linking)

- Epäsuora dynaaminen linkitys
  - kaikki viitatut 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 suoritetaan osana kutsuvaa prosessia käyttäen sen omaa aktivointitietuepinoa

Nimien sidonta (3)

- Milloin symbolin L suoritusaikainen muistiosoite sidotaan (lasketaan valmiiksi)? (binding time)
  - ohjelman kirjoitusaikana?
  - käännösaikana?
  - linkityksessä?
  - latauksessa?
  - kantarekisterin asetuksen aikana?
  - osoitteen sisältämän konekäskyn suoritusaikana?
  - …

virtuaaliosoite

- Sijainnista riippumattomassa koodissa kaikki viittaukset ovat suhteessa PC:hen tai pinossa (position independent)

Nimen sidonta voi tapahtua jo ohjelmaa kirjoitettaessa esim. kun koodissa viitataan porttinumeroon tai keskeytyskäsittelijään, joka sijaitsee sovitussa paikassa.

Virtuaalimuistia käytettäessä nimien sidonta

tehdään viime hetkellä (konekäskyn suoritusaikana)

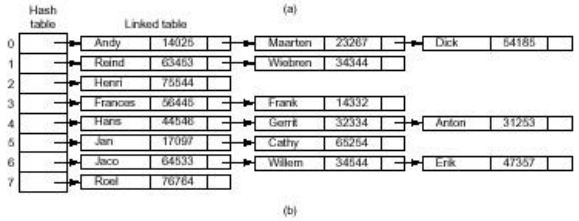
MMU:n (ja TLB:n) avulla

31.5.2001 Teemu Kerola, K2000 35

-- Luennon 10 loppu --

Andy	14025	0
Anton	31253	4
Cathy	65264	5
Dick	64185	0
Erik	47357	6
Frances	56445	3
Frank	14332	3
Gerrit	32334	4
Hans	44546	4
Henri	75544	2
Jan	17097	5
Jaco	64533	6
Maarten	23267	0
Reind	63463	1
Roel	76764	7
Willem	34544	6
Wiebren	34344	1

(a)



(b)

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]

31.5.2001 Teemu Kerola, K2000 36