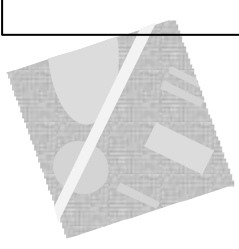


Jakso 10

Ohjelman suoritus järjestelmässä



Käännös

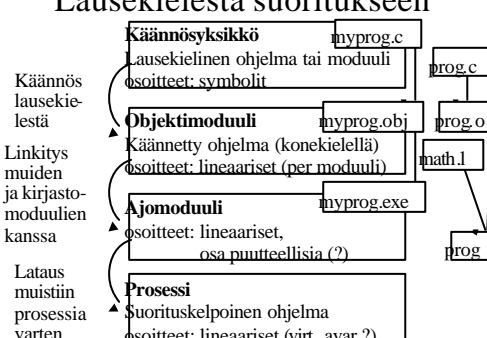
Linkitys

Dynaaminen linkitys

Lataus

21.8.2000 Teemu Kerola, K2000 1

Lausekielestä suoritukseen



21.8.2000 Teemu Kerola, K2000 2

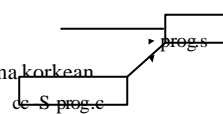
Käännösyksikkö (4)

- 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

21.8.2000 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

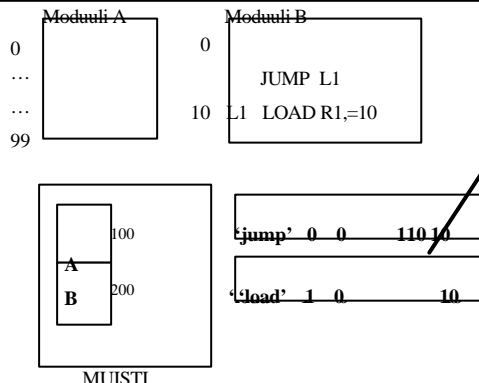


21.8.2000 Teemu Kerola, K2000 4

Objektimoduuli (8)

- Konekielinen koodi
 - moduulin sisäiset viitteet paikallaan (lineaarissa 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 virtuaalimuistin osoitteenmuunnosta)
 - tiedot viittauksista moduulin ulkopuolelle `IMPORT`
 - tiedot kohdista joista tähän moduuliin saa ulkopuolelta `EXPORT`
 - symbolitaulu

21.8.2000 Teemu Kerola, K2000 5



21.8.2000 Liisa Martinen, Ks 2000 6

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

21.8.2000 Teemu Kerola, K2000 7

Lähdekielinen ohjelma (5)

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

TTK-91 symbolinen koneikieli:

```

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

Pentium II, Motorola 680x0 ja SPARC symbolinen koneikieli: ks. Fig. 7-2 [Tane99]

21.8.2000 Teemu Kerola, K2000 8

(Assembler) kääntäjän ohjaukset (4)

- Eivät varsinaista koodia
- Ohjaavat käännöstä

TTK-91: `DC DS EQU`

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

21.8.2000 Teemu Kerola, K2000 9

Makrot (6)

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

21.8.2000 Teemu Kerola, K2000 10

Literaalit (5)

- Vakioita
- Niin suuria, että vakio-osaan ttk-91: käskyn vakiot 2-tavuisia, arvoalue: -128...127
- Halutaan pitää eikä käskyjen yhteyteen talletettuna Pi DC 3.14159265 ; (!!?)
One DC 1
OneMee DC 1024576
- Niitä ei saisi muuttaa LOAD R1, One
ADD R1, =1
STORE R1, One ; ask for trouble

21.8.2000 Teemu Kerola, K2000 11

Literaalit

- Korkean tason kielissä kaikki isot vakiot aina literaaleja
 - kääntäjän pitäisi e m FortranX: 5 = 6; 00000000
 - literaalia ei saisi välittää viiteparametrina
 - aliohjelma voisi muuttaa sen arvoa?
- Myöskään joissakin assemblerkielissä literaaleja ei tarvitse erikseen esitellä
 - helpommin Load R14, =F'234567
 - vakion 234567 tilanvaraus automaattisesti

21.8.2000 Teemu Kerola, K2000 12

Assembler käynnös ⁽¹⁰⁾

- 1. vaihe:
 - laske käskyjen tilanvaraukset
 - ttk-91 helppoa, koska kaikki käskyt 4 tavua!
 - generoi symbolitaulu ks. Kuva 6.2 [Häk98]
 - arvot, arvon vaatima tavumäärä
 - uudelleensijoitustiedot (omana tauluna?)
 - generoi tai käytä muita tauluja
 - literaalitaulu (tilanvaraus loppuksi)
 - kääntäjän ohjauskäskytaulu
 - operaatiokooditaulu

21.8.2000 Teemu Kerola, K2000 13

Assembler käynnös ⁽⁷⁾

- 2. vaihe
 - generoi lopullinen objektimoduuli ks. Kuva 6.3 [Häk98]
 - tulosta symbolinen assembler -listaus ks. Fig. 7-16 [Tane99]
 - generoi taulut linkitystä varten
 - osana objektimoduulia
 - anna virheilmoitukset
- 3. vaihe
 - koodin optimointi

21.8.2000 Teemu Kerola, K2000 14

TTK-91 Assembler käynnös

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

tunnetaan

▼ Taas = 0

▼ i = ?

▼ s = ?

21.8.2000 Teemu Kerola, K2000 15

Symbolitaulu 1. vaiheen aikana ks. kalvo 15

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

Konekäskyt 2 ja 8

2:

OPER	Rj	M	Ri	ADDR
ADD	1	1	0	?

8:

JLES	0	0	0	0
------	---	---	---	---

21.8.2000 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

21.8.2000 Teemu Kerola, K2000 17

Symbolitaulu 1. vaiheen jälkeen ks. kalvo 17

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

Konekäskyt 2 ja 8

2:

OPER	Rj	M	Ri	ADDR
ADD	1	1	0	10

8:

JLES	0	0	0	0
------	---	---	---	---

21.8.2000 Teemu Kerola, K2000 18

TTK-91 objektimoduuli

Moduulin otsake
EXPORT-hakemisto
IMPORT-hakemisto
Uudelleensijoitushakemisto
Koodi ja alustettu data
Moduulin lopuke

(Kuva 6.3 [Häk98])

21.8.2000 Teemu Kerola, K2000 19

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

21.8.2000 Teemu Kerola, K2000 20

TTK-91 objektimoduuli

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

21.8.2000 Teemu Kerola, K2000 21

TTK-91 objektimoduuli

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

21.8.2000 Teemu Kerola, K2000 22

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

21.8.2000 Teemu Kerola, K2000 23

Korkean tason kielen käännös

- Enemmän vaiheita
 - Syntaktisten alkioiden etsintä
 - Syntaksipuun generointi ja jäsennyys
 - Lauseiden tunnistaminen syntaksipuun avulla
 - Välikielen (välikielen)
 - Välikielin generointi

BEGIN 12345 IF ()

(front end)

Välikielin generointi ja symbolitaulut

(back end)

Kääntäjien ja ohj. kielten kurssit

Lisää tietoa? →

ks. syntaksipuun jäsennyyspuun esimerkit

21.8.2000 Teemu Kerola, K2000 24

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

21.8.2000 Teemu Kerola, K2000 25

Linkitys esimerkki

- 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, ja aseta kyseisten viitteiden osoitteet oikein ks. Fig. 7-15 (a) [Tane99] ks. Fig. 7-15 (b) [Tane99]

21.8.2000 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

21.8.2000 Teemu Kerola, K2000 27

Muuttujan X viittaukset linkitettyinä listana

23:	Load	R1, X
...
34	Store	R3, X(R1)
...
555	Add	R4, X
700	DC	0 ; X

23:	Load	1 0	34
...
34:	Store	3 1	555
...
555:	Add	4 0	1

Symb	sij	viittaus
X	700	23

21.8.2000 Teemu Kerola, K2000 28

Staattinen linkitys (5)

- Tavallinen (staattinen) linkitys vaatii, että kaikki ohjelmakoodissa viitatus 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

21.8.2000 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)

21.8.2000 Teemu Kerola, K2000 30

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]

21.8.2000 Teemu Kerola, K2000 31

Windows DLL:n linkityksen kaksi tapaa ⁽³⁾

- 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 käsky, joiden avulla linkitys tapahtuu tarvittaessa
 - DLL ladataan vain jos siihen tulee viittaus
- DLL suoritetaan osana kutsuvaa prosessia käyttäen sen omaa aktivointitietuepinoa

21.8.2000 Teemu Kerola, K2000 32

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?
- Jos käsky siirretään sitomisen jälkeen, mennään metsään ...
- Sijainnista riippumattomassa koodissa kaikki viittaukset ovat joko absoluuttisia, suhteessa PC:hen tai pinossa

virtuaaliosoite	≠	virtuaaliosoite
-----------------	---	-----------------

(position independent)

21.8.2000 Teemu Kerola, K2000 33

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
 - virtuaalimuistia käytettäessä nimien sidonta tehdään viime hetkellä (konekäskyn suoritusaikana) MMU:n (ja TLB:n) avulla

21.8.2000 Teemu Kerola, K2000 34

-- Luennon 10 loppu --

0000	00000000	00000000
0001	00000001	00000001
0002	00000002	00000002
0003	00000003	00000003
0004	00000004	00000004
0005	00000005	00000005
0006	00000006	00000006
0007	00000007	00000007
0008	00000008	00000008
0009	00000009	00000009
000A	0000000A	0000000A
000B	0000000B	0000000B
000C	0000000C	0000000C
000D	0000000D	0000000D
000E	0000000E	0000000E
000F	0000000F	0000000F

Figure 3-42. Hash coding. (a) Symbols, values, and the hash codes derived from the symbols. (b) Eight-byte hash table with linked lists of symbols and values. [Tane99]

21.8.2000 Teemu Kerola, K2000 35