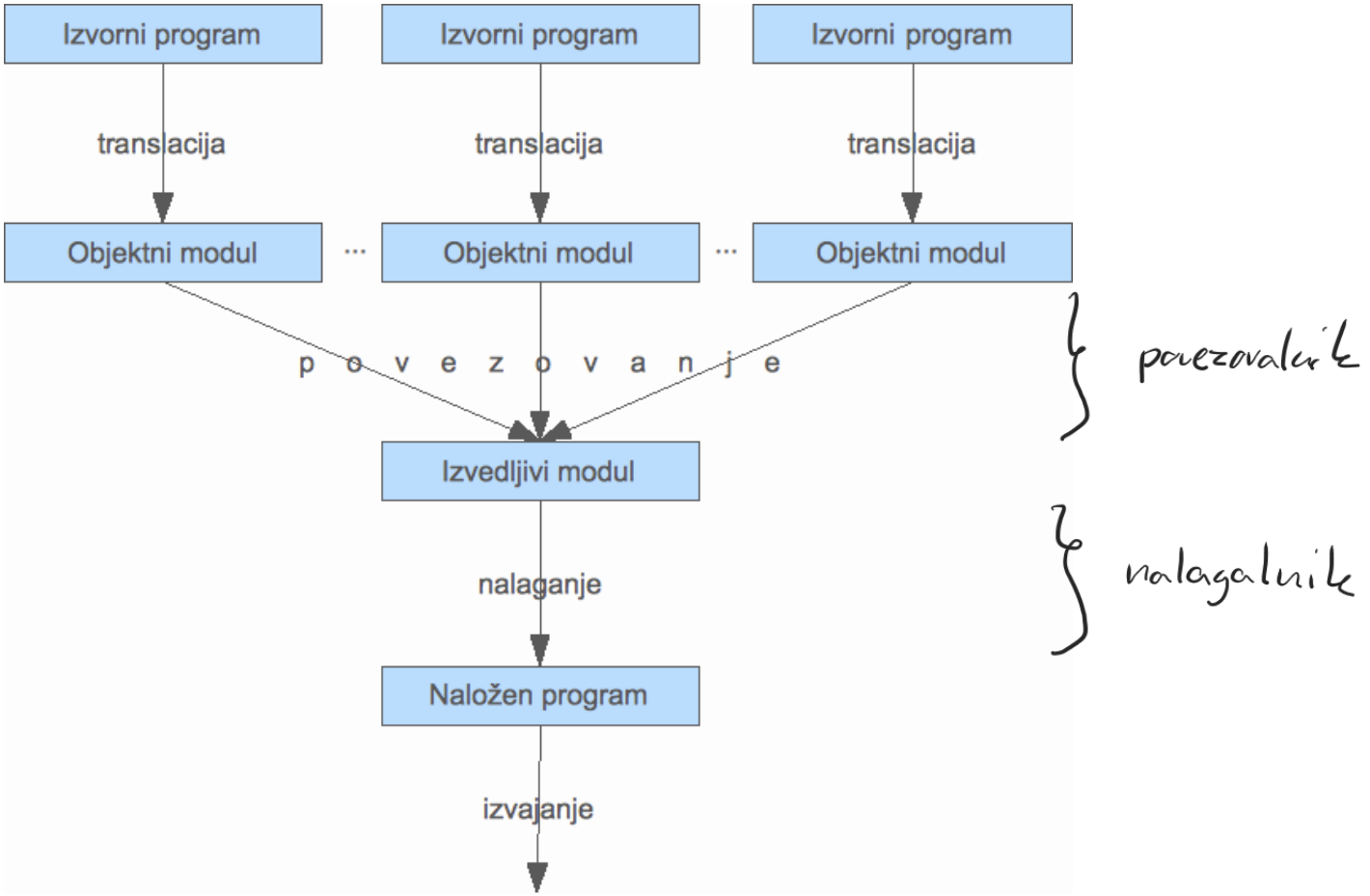
Univerzitetni študijski program, 3. letnik

Sistemska programska oprema

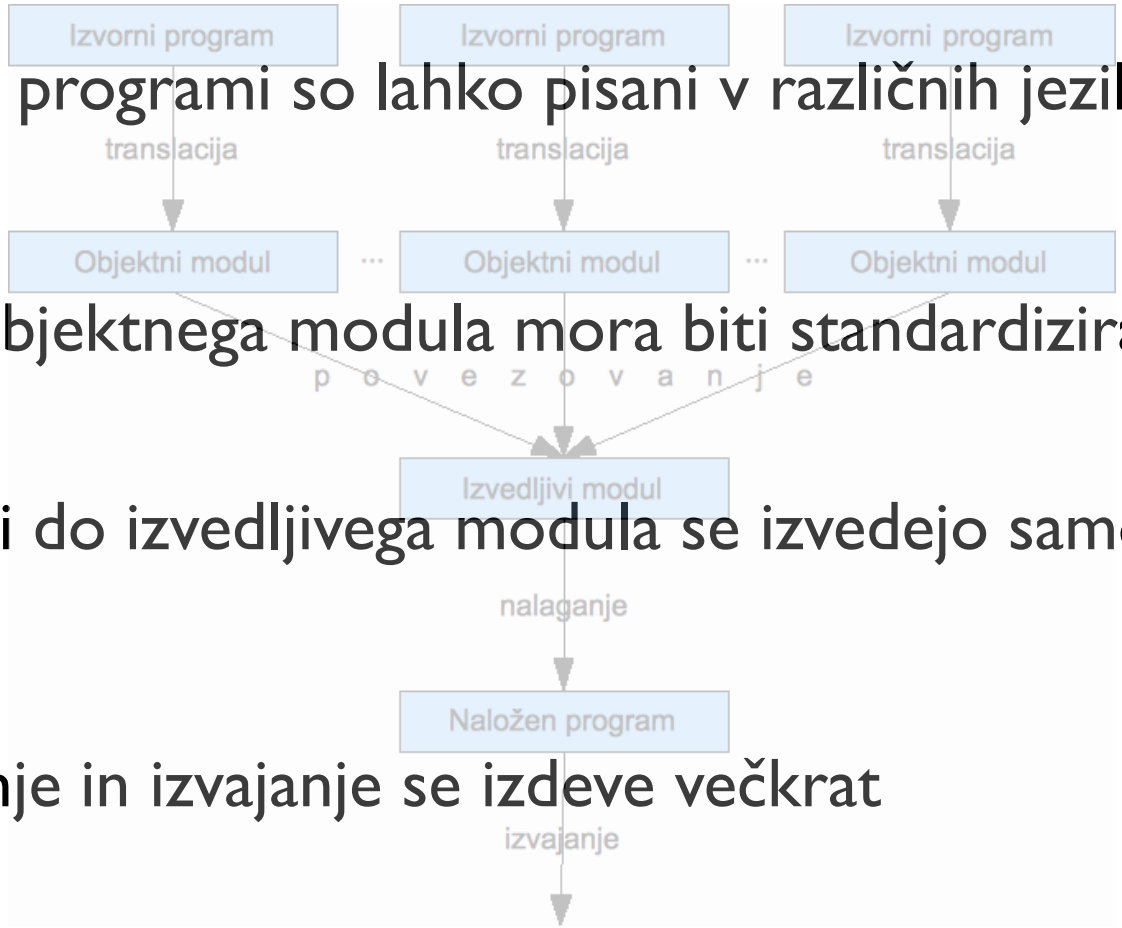
predavatelj: doc. Tomaž Dobravec

Povezovalnik

Življenjska pot programa



Življenjska pot programa



- izvorni programi so lahko pisani v različnih jezikih
- zapis objektnega modula mora biti standardiziran
- procesi do izvedljivega modula se izvedejo samo enkrat
- nalaganje in izvajanje se izdeve večkrat

Življenjska pot programa

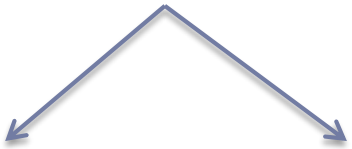
Po translaciji na objektni nivo, pred izvajanjem:

- posameznemu modulu dodeliti pomnilniški prostor,
- opraviti povezovanje med moduli,
- prenasloviti vse prenaslovljive operande,
- dopolniti izvedljivi program s sistemskimi rutinami,
- naložiti program na pravo lokacijo.

povezovalnik

nalagalnik

Vse to lahko opravi en ali več programov.



povezovalnik nalagalnik

Povezovalnik in/ali nalagalnik

- Klasičen povezovalnik ne pozna končnega nalagalnega naslova (tega pozna šele nalagalnik)

- Povezovalnik in nalagalnik včasih združijo v en program.

  Prednosti:

  preprostejši algoritem za povezovanje

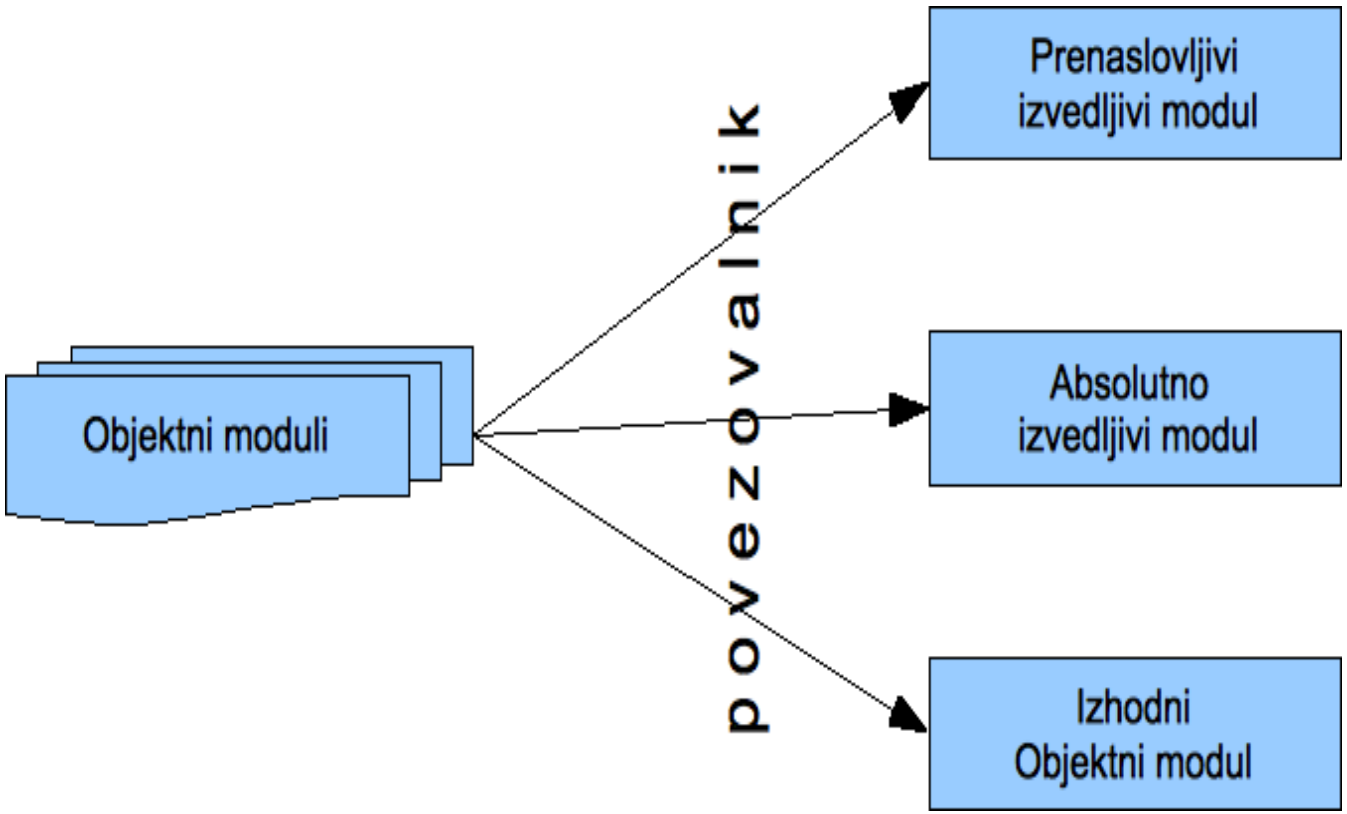
  Slabosti:

  pred izvajanjem je treba program vedno znova povezati.

Naloge povezovalnika

- Iz enega ali več vhodnih objektnih modulov zgradi kompakten izvedljiv program ali izhodni objektni modul;
- objektnim modulom dodeli pomnilniški prostor,
- vse prenaslovljive module združi v enoten prenaslovljiv del kode,
- vse prenaslovitvene tabele združi v eno tabelo,
- poskrbi, da enaka simbolična imena v celotnem programu pomenijo ISTO pomnilniško lokacijo ali konstanto

Vrste izhodnih modulov



Vrste izhodnih modulov

Izhod v obliki absolutno izvedljivega modula

- razrešene so vse reference iz prenaslovitvene tabele in vsi globalni simboli,
- program se vedno naloži na isto pomnilniško lokacijo.

Izhod v obliki izhodnega objektnega modula

- več objektnih modulov se združi v en modul,
- razrešene so vse medsebojne reference,
- tak modul lažje vključimo v druge programe.

Vrste izhodnih modulov

Izhod v obliki prenaslovljivega izvedljivega modula

V tem primeru povezovalnik

- razreši vse zunanje reference in odstrani tabele globalnih simbolov,
- vse prenaslovljive sekcije združi v eno prenaslovljivo sekcijo,
- ustvari nove prenaslovitvene zapise,
- iz več vhodnih objektnih datotek ustvari eno izhodno objektno datoteko.

Delovanje povezovalnika

Povezovalnik datoteke bere dvakrat.

1. prehod: zgradi povezovalne tabele

Control Section Table

- tabela razdelitev pomnilnika (CSTAB)
- tabela zunanjih simbolov (ESTAB)

External Symbol Table

2. prehod: na podlagi podatkov iz CSTAB in ESTAB vse sestavne dele poveže v izvedljiv modul.

Prvi prehod statičnega povezovanja

Tvorba tabele razdelitev pomnilnika (CSTAB)

- posamezne sekcije "razporedimo" po pomnilniku, v CSTAB si zapomnimo njihove nalagalne naslove

Ime sekcije	Naslov	Dolžina
S 1	0	d1
S 2	d 1	d 2
S 3	d 1 + d 2	d 3

naslov [0] = 0

naslov [i] = naslov [i - 1] + dolžina [i - 1]

Prvi prehod statičnega povezovanja

Tvorba tabele zunanjih simbolov (ESTAB)

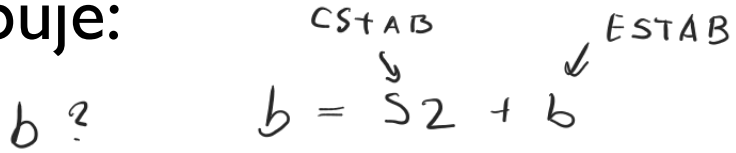
- v ESTAB za vsak zunanji simbol hranimi ime sekcije in njegovo vrednost (naslov)

naslov

Ime sekcije	Ime simbola	Vrednost
S1	X	10
S1	y	20
S1	z	1000
S2	b	50

Drugi prehod statičnega povezovanja

V drugem prehodu povezovalnik zgradi izvedljiv modul, ki vsebuje:



- **popravljeno (prenaslovljivo) kodo**

  na podlagi razpoložljivih podatkov popravi

  vse prenaslovljive naslove v prave naslove

- **popravljeno tabelo prilagoditvenih zapisov**

  prilagoditvene zapise spremeni glede na

  nove naslove kontrolnih sekcij

Delovanje povezovalnika - primer

- Delovanje povezovalnika si bomo ogledali na primeru spodnjih treh programov:

```
Loc       Source statement

0000  PROGA   START   0
              EXTDEF  LISTA,ENDA
              EXTREF  LISTB,ENDB,LISTC,ENDC
              .
              .
              .
0020  REF1    LDA     LISTA
0023  REF2   +LDT     LISTB+4
0027  REF3    LDX     #ENDA-LISTA
              .
              .
              .
0040  LISTA   EQU     *
              .
              .
              .
0054  ENDA    EQU     *
0054  REF4    WORD    ENDA-LISTA+LISTC
0057  REF5    WORD    ENDC-LISTC-10
005A  REF6    WORD    ENDC-LISTC+LISTA-1
005D  REF7    WORD    ENDA-LISTA-(ENDB-LISTB)
0060  REF8    WORD    LISTB-LISTA
              END
```

```
Loc       Source statement

0000  PROGB   START   0
              EXTDEF  LISTB,ENDB
              EXTREF  LISTA,ENDA,LISTC,ENDC
              .
              .
              .
0036  REF1   +LDA     LISTA
003A  REF2    LDT     LISTB+4
      +LDX    #ENDA-LISTA
              .
              .
              .
              EQU     *
              .
              .
              .
              EQU     *
              WORD    ENDA-LISTA+LISTC
              WORD    ENDC-LISTC-10
              WORD    ENDC-LISTC+LISTA-1
              WORD    ENDA-LISTA-(ENDB-LISTB)
              WORD    LISTB-LISTA
              END
```

```
Loc       Source statement

0000  PROGC   START   0
              EXTDEF  LISTC,ENDC
              EXTREF  LISTA,ENDA,LISTB,ENDB
              .
              .
              .
0018  REF1   +LDA     LISTA
001C  REF2   +LDT     LISTB+4
0020  REF3   +LDX     #ENDA-LISTA
              .
              .
              .
0030  LISTC   EQU     *
              .
              .
              .
0042  ENDC    EQU     *
0042  REF4    WORD    ENDA-LISTA+LISTC
0045  REF5    WORD    ENDC-LISTC-10
0048  REF6    WORD    ENDC-LISTC+LISTA-1
004B  REF7    WORD    ENDA-LISTA-(ENDB-LISTB)
004E  REF8    WORD    LISTB-LISTA
```

CS TAB

Ime sekcije	Naslov	Dolžina
PROGA	0	0x63
PROGB	0x63	0x7F
PROGC	0xE2	0x51

ESTAB:

Ime selecije	Ime simbola	Vrednost
PROGA	LISTA	0x40
	ENDA	0x54
PROGB	LISTB	0x60
	ENDB	0x70
PROGC	LISTC	0x30
	ENDC	0x42

Zbirnik: v PROGA

0x24 ↓ - 5 →

① +LDT LISTB + 4 → 771 00004

M 000024 05 + LISTB

Povezovalnik:

LISTB = 0x63 + 0x60 = 0xC3

↳ 771000C7 , M 000024 5 + PROG

To bo začetni naslov po nalaganju

(P) PROGB:

REF4 WORD ENDA - LISTA + LISTC

↓ zbirnik

00 00 00 00

M 00 00 70 06 + ENDA
M 00 00 70 06 - LISTA
M 00 00 70 06 + LISTC

Linker

ENDA = PROGA + 54 = 54
LISTA = PROGA + 40 = 40
LISTC = PROGC + 30 = 0xE2 + 0x30 = 0x112

ENDA - LISTA + LISTC = 0x126

↳ 00 01 26

M 00 00 03 | 06 + PROG
↓
0x70
4 0x63

Izvedba povezovanja (algoritem)

- Algoritem predpostavlja uporabo prilagoditvenih zapisov.
- **Vhod algoritma**: je množica prenaslovljivih objektnih programov, vsak vsebuje eno ali več kontrolnih sekcij, zunanje reference ter prenaslovitveno tabelo.
- **Izhod algoritma**: en objekten program z eno prenaslovitveno tabelo.
- Algoritem predpostavi, da je **začetni nalagalni naslov 0**
  - vsi direktni naslovi, ki jih vsebuje izhodni objektni program, so relativni glede na končni nalagalni naslov,
  - za vsak direktni naslov je izhodnemu objektnemu programu dodan prilagoditveni zapis.

Izvedba povezovanja (algoritem)

- Kontrolne sekcije so lahko medseboj prepletene preko zunanjih referenc.
- Pri branju posamezne kontrolne sekcije povezovalnik še ne pozna naslova vseh zunanjih referenc.

Težavo rešimo s pomočjo dveh prehodov:

1. prehod:

   Določi naslove kontrolnih sekcij in zunanjih simbolov

2. prehod:

   Prenaslavljanje vseh direktnih naslovov.

1. prehod povezovalnika

- V prvem prehodu algoritem obravnava le H (začetek nove kontrolne sekcije) in D (zunaji simboli) zapise.

- Po koncu prvega prehoda morajo biti

  - pravilno napolnjeni tabeli CSTAB in ESTAB,
  - pravilno nastavljene vrednosti spremenljivk
    - `PROGNAME` (ime programa);
    - `PROGLEN` (dolžina celotnega programa);
    - `STARTADDR` (naslov prvega ukaza, ki se mora izvršiti);

2. prehod povezovalnika

- V drugem prehodu povezovalnik
  - še enkrat prebere vse kontrolne sekcije,
  - popravi direktne naslove (upošteva nalagalni naslov sekcije in vrednosti vseh zunanjih simbolov) in
  - tvori tabelo novih prilagoditvenih zapisov (`NEWMODREC`).

2. prehod povezovalnika

Pri tvorbi tabele `NEWMODREC` je treba:

- Vsem prilagoditvenim zapisom popraviti ime zunanjega simbola.

  M 00 00 70 06 + LIST C → M 00 00 70 06 + PROG

- Popraviti naslov prilagoditvenega zapisa tako, da se originalni naslov poveča za vrednost nalagalnega naslova kontrolne sekcije.

  → M 00 00 D3 06 + PROG

- Paziti, da se medseboj izničujoči zapisi ne zapišejo v izhodno datoteko.

  M 00 00 D3 06 + PROG } odstranimo
  M 00 00 D3 06 - PROG }

Delovanje povezovalnika - primer

- Povezovanje treh kontrolnih sekcij (primer s slike 2.16):

```
a   HCOPY 000000 001033
b   D BUFFER 000033 BUFEND 001033 LENGTH 00002D
c   R RDREC  WRREC
d   T 000000 1D 172027 4B100000 032023 …
e   T 00001D 0D 000003 0f200A 4B100000 …
    …
f   M 000004 05 +RDREC
g   M 000024 05 +WRREC

h   H RDREC 000000 00002B
i   R BUFFER LENGTH BUFEND
j   T 000000 1D B410 B400 …
k   T 00001D 0E 3B2FE9 13100000 4F0000 F1 000000
    …
l   M 000028 06 +BUFEND
m   M 000028 06 –BUFFER

n   H WRREC 000000 00001C
o   R LENGTH BUFFER
p   T 000000 1C B410 77100000 E32012 …
r   M 000003 05 +LENGTH
```

① CSTAB

	Naslov	Dolžina
COPY	0	1033
RDREC	1033	2B
WRREC	105E	1C

ESTAB

	Ime simbola	Vrednost
COPY	BUFFER	33
COPY	BUFEND	1033
COPY	LENGTH	2D

PROGNAME = COPY
PROGLEN = 105E + 1C = 107A
STARTADDR = 0

② HCOPY 00 00 00 00107A
T 00 00 00 1D 172027 4B101033 . . .
T 00 00 1D 0D 000003 0F200A 4B10105E . . .

M 00 00 04 05 +COPY
M 00 00 24 05 +COPY

T 00 1033 1D . . .
T 00 1050 0E . . . 001000

M 00105B 06 +COPY
M 00105B 06 −COPY
} skenslamo, tega ni v izhodni datoteki

T 00105E 1C B410 7710002D ...
M 001061 05 +COPY