

# Sistemske programi



# Editor

Poređenje editora: [https://en.wikipedia.org/wiki/Comparison\\_of\\_text\\_editors](https://en.wikipedia.org/wiki/Comparison_of_text_editors)

Kreiranje/izmena sadržaja tekstualnih fajlova

Prikaz (dela fajla) po linijama

– kontrolni znaci, gde počinje i završava se linija?

00000000:	50 72 76 61 20 6C 69 6E 69 6A 61 20 74 65 6B 73	Prva linija teks
00000010:	74 61 2E 20 49 20 6A 6F 73 20 6D 61 6C 6F 2E 2E	ta. I jos malo..
00000020:	2E 0A 44 72 75 67 61 20 6C 69 6E 69 6A 61 2C 20	..Druga linija,
00000030:	70 72 65 74 68 6F 64 6E 6F 20 6A 65 20 7A 6E 61	prethodno je zna
00000040:	6B 20 7A 61 20 45 4E 54 45 52 2E 0A 0A 54 72 65	k za ENTER...Tre
00000050:	63 61 20 6C 69 6E 69 6A 61 20 73 61 20 6D 61 6C	ca linija sa mal
00000060:	6F 20 72 61 7A 6D 61 6B 61 20 6F 64 20 64 72 75	o razmaka od dru
00000070:	67 65 2E 0A	ge..

```
Prva linija teksta. I jos malo...
Druga linija, prethodno je znak za ENTER.

Treca linija sa malo razmaka od druge.
```

# Editor

## Komande

- editorske komande  
(komandni i znakovni režim rada)
- kombinacija tastera
- poseban pokazivački uređaj

## Kursor

- ubaci/prepiši (engl. *insert/overwrite*)
- enter – pomeranje na narednu liniju

**Esc** Normal

Revision 2.0  
Sept. 11, 2011

Vim 7.3+  
:version

## Vim Cheat Sheet for Programmers

Copyleft © 2011  
May be freely distributed!  
Sharing is Caring.

<http://michael.PeopleOfHonorOnly.com/vim/>

**HOW-TO** make Vim not suck: Out of the Box: `:help statusline` `:set nocompatible ruler laststatus=2 showcmd showmode number` **Search** `:set incsearch ignorecase smartcase hlsearch` **Remove useless splash screen** `:set shortmess+=!`

**Best tips:** <http://vim.wikia.com/> **Best scripts:** <http://www.vim.org/scripts/index.php>

~ toggle case	! extern filter	@ play macro	# prev identifier	\$ →	% goto match	^ soft ←	& repeat :s	* next identifier	( begin sentence	) end sentence	_ cur line	+ ↓
• goto mark	1	2	3	4	5	6	7	8	9	0	hard ←	= auto-format

14 block select	^w window...	scroll line ↑	12 :redo	10 ctags return	scroll line ↓	half page ↑	Ctrl I	prev mark	9 ↑	Normal	ctags identifier
Q ex mode	W WORD ↘	E end WORD ↘	R Replace	T ← until char	Y copy line	U undo line	I insert ←	O open ↑	P paste ↑	{ paragraph	} paragraph
q record macro	w word ↘	e end word ↘	r replace char	t ← until char →	y copy	u undo	i ↵ insert	o open ↓	p paste ↓	[ misc.	] misc.

7 incr. #	10 half page ↓	page ↓	file/cursor info	Ctrl H	15 Ctrl J	redraw	Ctrl ;	Ctrl ' 0	Ctrl \
A append →	S subst line	D del →	F ← find char	G goto eof / goto line#	H Top screen	J Join lines	K man page identifier	L Bottom screen	: cmd line
a append ↵	s subst char	d del	f find char →	g extra	h ←	j ↓	k ↑	l →	; "next" / f/F/I/T

Ctrl ^	:suspend	7,11 decr. #	Normal / Cancel	block select	page ↑	9,16 ↓	15 Ctrl M	Ctrl ,	Ctrl -	Ctrl /
Shift ↑	Z quit	X ← del char	C change →	V select lines	B ↘ WORD	N "prev" find	M Middle screen	< undent	> indent	? find ↘
	Z extra	X del char →	c change	v select chars	b ↘ word	n find "next"	m set mark	"prev" / f/F/I/T	repeat cmd	/ find ↵

**Unused & Duplicate keys**  
 \ Ctrl-K Ctrl-S (free)  
 Ctrl-L (redraw)  
 ~ near dup of '  
 13 ~ near dup of '  
 14 Ctrl-Q = Ctrl-V  
 15 Ctrl-J = Ctrl-M = ^N

**Legend:**

- Macro Register name (0-9a-zA-Z) required
- Op Motion req.; act between cursor & dst
- Cmd Command
- Ins Command and enter insert mode
- Move Moves cursor or defines range for op
- tag Search (\ = reverse, / = forward)
- tag ctags / diffs / folding
- Code Code formatting, whitespace, etc.
- Extra Extended functionality; req. extra chars
- Char arg req. g z Z ^w ' " ...

**Modes**

`:help modes`

n Normal Esc ^ [ ^c  
i Insert a i r s  
v Visual v V ^v ^q  
o Op pending c d y < >  
c Command Line : / ? !

**word** Foo ( src , b dst , b len ) ;

Note: There is no whitespace in-between 'Foo(src', but before/after 'dst.'

**WORD** Foo ( src , b dst , b len ) ;

**Startup**

vim <filename> +123 goto line 123  
vim <file> ... -t Foo edit at tag 'Foo'  
vim <file> ... -c "/Foo" cmd: find 'Foo' & edit

GUI vim -g or gvim start GUI ver.  
GUI Linux :set guifont=ProggyTinyTT 12  
GUI OSX :set guifont=ProggyTiny\;hll  
diff gvimdiff <file> <file2> [<file3>]

**bug** **Broken Keys** Ctrl-I = Tab, Ctrl-[ = ESC  
Vim is still unable to map certain keys for your own use...

**§ Caps, Ctrl-I, Ctrl-Shift-1, Ctrl-I, Ctrl-I, etc.**

0 See: src/ops.c -c "/valid\_yank\_reg" for " reg. names  
6 See: src/normal.c -c "/nv\_cmds" for g extra cmds  
11 See: src/edit.c -c "/ctrl\_x\_msgs" for ^x insert cmds

**:help cmdline** :r file insert file

!w save !gui switch to GUI  
:q quit !q! quit w/o save  
:e <file> edit file in new buffer  
:source % exec cmds in cur file  
:exec '...' do cmd

**move movement**

soft ^ ← Start of Line 1st non-whitespace  
hard 0 ← Start of Line column 0  
\$ → End of Line

| move col 0 | move col #

^b page ↑ ^f page ↓  
^u 1/2 page ↑ ^d 1/2 page ↓  
^e scroll line ↑ ^y scroll line ↓  
1g start of file 0g end of file  
#g goto line # G end of file

[ [ begin this func {  
] ] begin next func {  
:set matchpairs=(:){,[:],[:>,:?<:  
% goto matching { } <> [ ]

**help range**

:s/Foo/Bar find Foo replace w/ Bar  
:s/Foo/Bar/g ...all instances on line  
:%s/Foo/Bar apply to whole file  
:.,+ # cur line, cur line + # lines  
\$ last line ' < start of select  
' > end of select

**Code** = < > << >>

:set backspace=indent,eol,start  
allow backspace join lines  
:set shiftwidth=# indent width for ai  
:set autoindent! toggle auto-indent  
:set lisp lisp indent mode

**Tags**

!ts list active tags  
^j jump to tag under cursor  
^t restore cursor before tag jump  
^p complete word  
:ta Foo manual jump to tag 'Foo'

**Diff**

[c prev diff :hi DiffAdd guifg=#rrggbb  
]c next diff :hi DiffChange guibg=#rrggbb  
:diffupdate :hi DiffText gui=none  
resync :hi DiffDelete

**Folding**

zR fold remove  
zo fold open  
zc fold close  
zi invert all  
zR fold reduce  
zm fold more

**Changes**

:changes  
g; older change  
g; newer change

**Syntax**

:syntax enable  
:set filetype=  
c cpp sh make perl python  
Note: chose only ONE type!

**Convert <eol>**

q start recording  
@ playback  
q stop recording  
@@ repeat

then :w to convert

**help recording**

:set tabstop=# set tab stop every #th col  
:set expandtab! toggle hard/soft tabs  
:set listchars= tab>:~:trail:-,nbsp:%,eol:\$ toggle whitespace  
:set list! toggle whitespace  
:set colorcolumn=80 visible right margin indicator  
noremap + :s/\|\\|/|<CR> block comment

**Insert mode**

9 ^p prev auto-complete ^n next  
10 ^d undent ^t indent  
\$11 ^x ^f filename completion  
^s spelling :set spell!  
^k dictionary ]s next bad  
^t thesaurus :help spell

12 ^r paste register 0-9a-zA-Z or ...  
+ clipboard (or '\*') :help c\_CTRL-R  
" last del/copy % filename

**buffer #**

:buffers list files  
:new blank file/buffer  
:bn new file  
:bp prev file  
:bd close file  
:bd! force close  
:set lines=#  
:set columns=#  
:winpos # # # GUI

**Windows**

**help windows**

^w or :wincmd \*  
w :switch to next  
c :close!  
n :new  
s :split horz.  
v :vsplit vertical  
o :only maximize  
= all same size  
h move to win ←  
j move to win ↓  
k move to win ↑  
l move to win →  
:sp [<filename>]  
edit in split window

**Cursor Bookmarks**

:marks  
:o

ma mark local 'a'  
'A goto global 'A'  
'' prev location

**File / Directory**

:Explore Of :e .  
:set browsefor=...  
one of buffer last

# Makro pretprocesor

Poziva se pre asemblera, analizira fajl i prepoznaje:

- makro definicije
- makro pozive
- uslovne direktive

Menja sadržaj fajla pre prosleđivanja asembleru

Zasniva se na leksičkoj i sintaksoj analizi

Prepoznavanje makro definicija

- **tabela makro imena**
- **tabela makro tela**

# Makro pretprocesor

Radi brže pretrage, parametri makroa se zamenjuju rednim brojevima ispred kojih je znak &

IZBACI

```

MAKRO          R
PREBACI_RP    R, (%0)
DODAJ_1       %0
KRAJ
  
```

UBACI

```

MAKRO          R
PREBACI_PR    (%1) ,R
DODAJ_1       %1
KRAJ
  
```

Tabela makro imena		
IZBACI	1	2
UBACI	3	4

Tabela makro tela		
1	PREBACI_RP	&1, (%0)
2	DODAJ_1	%0
3	PREBACI_PR	(%1) , &1
4	DODAJ_1	%1

# Makro pretprocesor

Prepoznavanje makro poziva

- argumenti se smeštaju u **tabelu argumenata**

**IZBACI %2**

redni broj	Argument
1	%2

- pretraga tabele imena i nalaženje linija koje odgovaraju makro definiciji
- zamena poziva telom (uz zamenu parametara)
- ponovna analiza od prve ubačene linije

# Asembler

Analiza programa u vidu tekstualnog fajla

- prepoznavanje naredbi i direktiva
- leksička, sintaksna i semantička analiza

SABERI %3, #2

SABERI %3, \$2

PREBACI\_DR prom, %l

(pri čemu prom ne postoji)

**Skener** – prepoznaje ispravne reči jezika

**Parser** – prepoznaje ispravne rečenice jezika

**Semantička analiza** – obično deo parsera

# EBNF definicije

malo\_slovo  $\rightarrow$  a|b|c|č|ć|d|đ|e|f|g|h|i|j|k|l|m|n|o|p|r|s|š|t|u|v|z|ž

cifra  $\rightarrow$  0|1|2|3|4|5|6|7|8|9

decimalni\_broj  $\rightarrow$  cifra{cifra}

heksa\_cifra  $\rightarrow$  cifra|A|B|C|D|E|F

heksadecimalni\_broj  $\rightarrow$  0x(heksa\_cifra){heksa\_cifra}

broj  $\rightarrow$  decimalni\_broj|heksadecimalni\_broj

labela  $\rightarrow$  malo\_slovo{malo\_slovo|cifra|\_}

# EBNF definicije

- direktiva -> nova\_linija [labela:]  
razmak (ZAUZMI|NAPUNI)  
razmak broj
- telo -> { direktiva  
| osnovna\_naredba  
| naredba\_prebacivanja  
| upravljačka\_naredba }
- program -> POČETAK razmak labela telo  
nova\_linija KRAJ

# Asembler

## Greške prilikom asembliranja

- pojava neočekivanog znaka
- pojava neočekivane reči
- kršenje semantičkog pravila
- oporavak od greške – sledeća naredba/direktiva

## Generisanje mašinskog koda

- ako je naredba uspešno prepoznata
- kod naredbe – iz **tabele naredbi** (engl. *opcode table*)

kod tipa naredbe (4 bita)	relativni kod naredbe (4 bita)	kod 1. registra (4 bita)	kod 2. registra (4 bita)	obavezna reč dodatna reč

Ime naredbe	Heksadecimalni kod naredbe i njena dužina		Ime naredbe	Heksadecimalni kod naredbe i njena dužina	
DESNO	34	1	SKOČI	C0	2
DODAJ_1	30	1	SKOČI_ZA_<	D2	2
I	14	1	SKOČI_ZA_<=	D5	2
ILI	15	1	SKOČI_ZA_!=	D1	2
LEVO	33	1	SKOČI_ZA_==	D0	2
NATRAG	F0	1	SKOČI_ZA_>	D4	2
NE	32	1	SKOČI_ZA_>=	D3	2
ODBIJ_1	31	1	SKOČI_ZA_±_<	D6	2
ODUZMI	12	1	SKOČI_ZA_±_<=	D9	2
ODUZMI_P	13	1	SKOČI_ZA_±_>	D8	2
POZOVI	E0	2	SKOČI_ZA_±_>=	D7	2
PREBACI_DR	60	2	SKOČI_ZA_M	DA	2
PREBACI_IR	80	2	SKOČI_ZA_N	D0	2
PREBACI_NR	50	2	SKOČI_ZA_NE_M	DB	2
PREBACI_PR	70	1	SKOČI_ZA_NE_N	D1	2
PREBACI_RD	90	2	SKOČI_ZA_NE_P	D3	2
PREBACI_RI	B0	2	SKOČI_ZA_NE_V	DD	2
PREBACI_RP	A0	1	SKOČI_ZA_P	D2	2
PREBACI_RR	40	1	SKOČI_ZA_V	DC	2
SABERI	10	1	UPOREDI	20	1
SABERI_P	11	1			

# Asembler

Viši bajt obavezne reči – kod naredbe

Niži bajt obavezne reči – registri

– SABERI %3, %2 -> 1023

Dodatna reč

– ako je broj, nema problema

– ako je labela, određivanje njene adrese mora prethoditi popunjavanju dodatne reči

Labela ispred naredbe

Labela ispred direktive

# Asembler

## **Tabela labela/simbola** (engl. *symbol table*)

- Uz svaku labelu postoji polje sa njenom adresom
- Zbog referenciranja unapred (engl. *forward reference*), asembliranje obično ide u dva prolaza
  1. Analiza teksta i popunjavanje tabele simbola
  2. Generisanje mašinskog koda
- Tabela labela često ima i polje sa tekućim stanjem:
  - definisana, definisana i korišćena,
  - nedefinisana i korišćena
- Određivanje adrese labela zahteva brojač lokacija

# Asembler

## **Brojač lokacija** (engl. *location counter*)

- zna se dužina svake naredbe i direktive
- prilikom analize programa se može izračunati adresa naredne naredbe/direktive
- u brojaču lokacija se uvek nalazi adresa početka naredbe ili direktive čija analiza sledi
- inicijalna vrednost?
- kada se naiđe na novu labelu, njena adresa je?

Asemblerski program			Brojač lokacija
	POČETAK	ulaz	0
ulaz:	PREBACI_NR	\$12,%0	0
	PREBACI_NR	\$10,%1	2
ponovo:	UPOREDI	%1,%0	4
	SKOČI_ZA_==	kraj	5
	SKOČI_ZA_<	manje	7
veće:	ODUZMI	%1,%0	9
	SKOČI	ponovo	10
manje:	ODUZMI	%0,%1	12
	SKOČI	ponovo	13
kraj:	SKOČI	kraj	15
	KRAJ		15

Labela	Adresa
kraj	15
manje	12
ponovo	4
ulaz	0
veće	9

# Asembler

Nakon II prolaza – mašinski oblik programa  
objektna sekvenca

Semantičke greške

– duplirana labela

– nedefinisana labela

Algoritamske

greške?

Tabela objektna sekvenca				
Adrese lokacija	Objektna sekvenca	Komentar		
		POČETAK	ulaz	
0000	5000	ulaz:	PREBACI_NR	\$12,%0
0001	000C			
0002	5010		PREBACI_NR	\$10,%1
0003	000A			
0004	2001	ponovo:	UPOREDI	%1,%0
0005	D000		SKOČI_ZA_==	kraj
0006	000F			
0007	D200		SKOČI_ZA_<	manje
0008	000C			
0009	1201	veće:	ODUZMI	%1,%0
000A	C000		SKOČI	ponovo
000B	0004			
000C	1210	manje:	ODUZMI	%0,%1
000D	C000		SKOČI	ponovo
000E	0004			
000F	C000	kraj:	SKOČI	kraj
0010	000F			
			KRAJ	

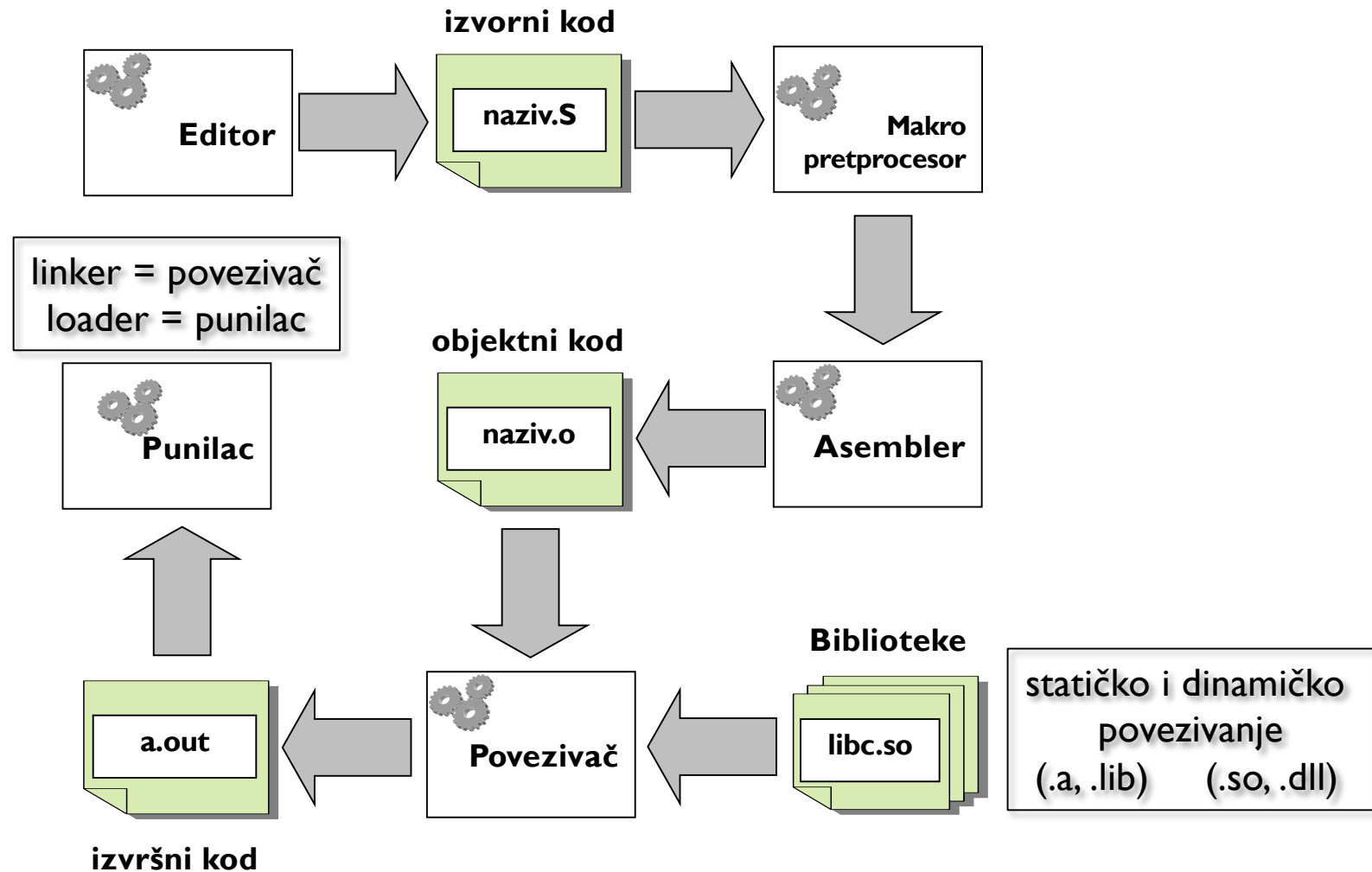
# Asembler

**Objektna sekvenca** se nalazi u objektnoj datoteci

- kod koji još uvek nije spreman za izvršavanje
- nije čitljiv kao tekstualna datoteka
- sadrži adresu ulazne naredbe (engl. *entry point*) koja odgovara ulaznoj labeli

Za izvršnu sekvencu je neophodan **povezivač** (engl. *linker*)

# Prevođenje i pokretanje programa



# Linker (povezivač)

**Od objektne sekvence kreira izvršnu sekvencu**

- povezivanje više fajlova
- dodavanje koda sistemskih potprograma

## **Problem relokacije**

- sve objektne sekvence počinju od iste adrese
  - samo jedna može od inicijalne adrese
- ređanje sekvenci jedna iza druge
  - konstanta relokacije
- problem apsolutnih adresa u kodu

Tabela objektne sekvence				
Adrese lokacija	Objektna sekvenca	Apsolutna adresa	Komentar	
			POČETAK	ulaz
0000 0001	5000 000C		ulaz: PREBACI_NR	\$12,%0
0002 0003	5010 000A		PREBACI_NR	\$10,%1
0004	2001		ponovo: UPOREDI	%1,%0
0005 0006	D000 000F	<-	SKOČI_ZA_==	kraj
0007 0008	D200 000C	<-	SKOČI_ZA_<	manje
0009	1201		veće: ODUZMI	%1,%0
000A 000B	C000 0004	<-	SKOČI	ponovo
000C	1210		manje: ODUZMI	%0,%1
000D 000E	C000 0004	<-	SKOČI	ponovo
000F 0010	C000 000F	<-	kraj: SKOČI	kraj
			KRAJ	

# Linker (povezivač)

Apsolutne adrese se takođe moraju relocirati

- za koliko?
- **statička relocacija** – korekcija apsolutnih adresa za konstantu relocacije

## Tabela relocacije

Heksadecimalne adrese
0006
0008
000B
000E
0010

- generiše je assembler, sadrži **logičke adrese lokacija objektne sekvence** koje sadrže **apsolutne adrese**

# Relativno adresiranje

Bez apsolutnih adresa, nema ni problema relokacije

- ako se kod skokova umesto apsolutne adrese navede rastojanje (u lokacijama) do naredbe na koju se skače

**Relativna adresa** predstavlja razliku adrese dodatne reči naredbe skoka i obavezne reči ciljne naredbe

**Stvarna adresa** se dobija **sabiranjem sadržaja programskog brojača i relativne adrese**  
**(PC + relativna adresa)**

# Relativno adresiranje

Naredba SKOČI sa relativnim adresiranjem:

1. ciklus: programski brojač  $\rightarrow$  adresne linije (P2)  
1  $\rightarrow$  č (P41)  
linije podataka  $\rightarrow$  pomoćni registar (P3)
2. ciklus programski brojač  $\rightarrow$  registar 1. podatka (P2, P37, P42)
3. ciklus pomoćni registar  $\rightarrow$  registar 2. podatka (P4, P37, P43)
4. ciklus: saberi (P52)  
linije podataka  $\rightarrow$  programski brojač (P1)

Relativna adresa – označen ili neoznačen broj?

Tabela objektne sekvence				
Adrese lokacija	Objektna sekvence	Relativna adresa	Komentar	
			POČETAK	ulaz
0000	5000		ulaz: PREBACI_NR	\$12,%0
0001	000C			
0002	5010		PREBACI_NR	\$10,%1
0003	000A			
0004	2001		ponovo: UPOREDI	%1,%0
0005	D000		SKOČI_ZA_==	kraj
0006	0009	<-		
0007	D200		SKOČI_ZA_<	manje
0008	0004	<-		
0009	1201		veće: ODUZMI	%1,%0
000A	C000		SKOČI	ponovo
000B	FFF9	<-		
000C	1210		manje: ODUZMI	%0,%1
000D	C000		SKOČI	ponovo
000E	FFF6	<-		
000F	C000		kraj: SKOČI	kraj
0010	FFFF	<-		
			KRAJ	

# Problem spoljašnjih referenci

Korišćenje labele definisane u drugom fajlu

U toku asembliranja koda koji koristi spoljašnju labelu, takva labela ostaje nedefinisana

Asembler formira **tabelu nedefinisanih labela** (engl. *external reference table*) koju koristi linker

- pored naziva labele, mora sadržati sve adrese na kojima se ta labela koristi

Asembler formira i **tabelu ulaznih labela** (engl. *entry point table*)

# Problem spoljašnjih referenci

Tabela objektne sekvence			
Adrese lokacija	Objektna sekvencija	Nedefinisana adresa	Komentar
			POČETAK      primer
0000 0001	5010 000C		primer: PREBACI_NR      \$12,%1
0002 0003	5020 000A		PREBACI_NR      \$10,%2
0004 0005	EOF0 0000	<-	POZOVI      nzd
0006 0007	C000 FFFE		kraj: SKOČI      kraj
			KRAJ
Tabela relokacije			
-			
Tabela nedefinisanih labela			
nzd	0005		
Tabela ulaznih labela			
primer	0000		

Tabela objektne sekvence		
Adrese lokacija	Objektna sekvence	Komentar
		POČETAK      nzd
0000	2012	nzd: UPOREDI      %2, %1
0001 0002	D000 0009	SKOČI_ZA_==      kraj
0003 0004	D200 0004	SKOČI_ZA_<      manje
0005	1212	veće: ODUZMI      %2, %1
0006 0007	C000 FFF9	SKOČI      nzd
0008	1221	manje: ODUZMI      %1, %2
0009 000A	C000 FFF6	SKOČI      nzd
000B	4001	kraj: PREBACI_RR      %1, %0
000C	F0F0	NATRAG
		KRAJ
Tabela relokacije		
-		
Tabela nedefinisanih labela		
-	-	
Tabela ulaznih labela		
nzd	0000	

Objektna  
sekvence za  
Nzd  
potprogram  
(prenošenje param.  
preko %1, %2)

# Obrazovanje izvršne sekvence

Linker preuzima sve objektne sekvence i sve tabele u njima i pravi **tabelu objektnih sekvenci** (engl. *object module table*)

Ulazna labela objektne sekvence	Dužina objektne sekvence	Adresa početka objektne sekvence
<code>primer</code>	8	0000
<code>nzd</code>	13	0008

**Adresa početka** objektne sekvence je i **konstanta relokacije**

Vrši se relokacija adresa u tabelama relokacije i ulaznih labela

Potom se vrši i relokacija apsolutnih adresa u svim objektnim sekvencama

# Obrazovanje izvršne sekvence

Sve tabele ulaznih labela se spajaju u **tabelu spoljašnjih labela** (engl. *global symbol table*)

Tabela spoljašnjih labela	
<b>primer</b>	0000
<b>nzd</b>	0008

Prolazi se kroz sve tabele nedefinisanih labela i koriguju se sve adrese koje pripadaju svakoj od nedefinisanih labela

Potom se sve objektne sekvence mogu spojiti u jednu izvršnu sekvencu

Tabela izvršne sekvence			
Adrese lokacija	Izvršna sekvenca	Komentar	
0000 0001	5010 000C	primer: PREBACI_NR	\$12,%1
0002 0003	5020 000A	PREBACI_NR	\$10,%2
0004 0005	E0F0 0003	POZOVI	nzd
0006 0007	C000 FFFE	kraj: SKOČI	kraj
0008	2012	nzd: UPOREDI	%2,%1
0009 000A	D000 0009	SKOČI_ZA_==	kraj
000B 000C	D200 0004	SKOČI_ZA_<	manje
000D	1212	veće: ODUZMI	%2,%1
000E 000F	C000 FFF9	SKOČI	nzd
0010	1221	manje: ODUZMI	%1,%2
0011 0012	C000 FFF6	SKOČI	nzd
0013	4001	kraj: PREBACI_RR	%1,%0
0014	F0F0	NATRAG	

# Linker (povezivač)

Linker obično radi u **dva prolaza**

## **I prolaz:**

- formiranje tabele objektnih sekvenci
- relokacija tabela
- formiranje tabele spoljašnjih labela

## **II prolaz:**

- relokacija apsolutnih adresa
- rešavanje spoljašnjih referenci
- stvaranje izvršne sekvence

**Ulazna adresa** izvršne sekvence je **jednaka ulaznoj adresi** njene **prve objektno sekvence**

# Loader (punilac)

Zauzimanje (dovoljno) radne memorije

Kopiranje izvršne sekvence iz izvršne datoteke u radnu memoriju (RAM)

Formiranje **slike procesa** (popunjavanje atributa)

Podешavanje baznog i graničnog registra

Pokretanje programa počevši od njegove ulazne adrese

Postupak pretvaranja logičkih u fizičke adrese se naziva i **dinamička relokacija**

# Dibager (engl. *debugger*)

Program koji omogućava nadgledanje izvršavanja drugih programa

Neophodno je da postoji mogućnost prekida izvršavanja programa nakon svake ili nakon unapred odabranih naredbi, nakon čega se poziva dibager

## **Koračni režim rada procesora** (engl. *single step*)

- bit traga (engl. *trace bit*)
- SR<sub>6</sub> kod KONCEPT-a
- pre dobavljanja svake naredbe se dešava izuzetak u okviru čije obrade se poziva dibager

# Dibager

## **Naredba zamke** (engl. *trap*)

- direktno dovodi do izvršavanja izuzetka
- omogućuje rad dibagera i ako nema koračnog režima rada
- pozivanje dibagera nakon samo nekih naredbi
  - zamena naredbe naredbom zamke
  - vraćanje originalne naredbe kada se aktivira dibager

## **Dibagerski registri**

- dibager se aktivira kada se adresa u programskom brojaču poklopi sa sadržajem nekog od dibagerskih registara