Univerzitetni študijski program, 3. letnik

Sistemska programska oprema

predavatelj: doc. Tomaž Dobravec

ZBIRNIK (1. del)

Kazalo

O zbiranju – splošno

- Vsak procesor razume le svoj **strojni jezik**
- Pisanje v strojnem jeziku je zamudna naloga → uporabljamo **zbirni jezik.**
- Zbirni jezik = linearna preslikava strojne vsebine.
- Zbirni jezik neločljivo povezan s strojno opremo.

Uporaba zbirnih jezikov

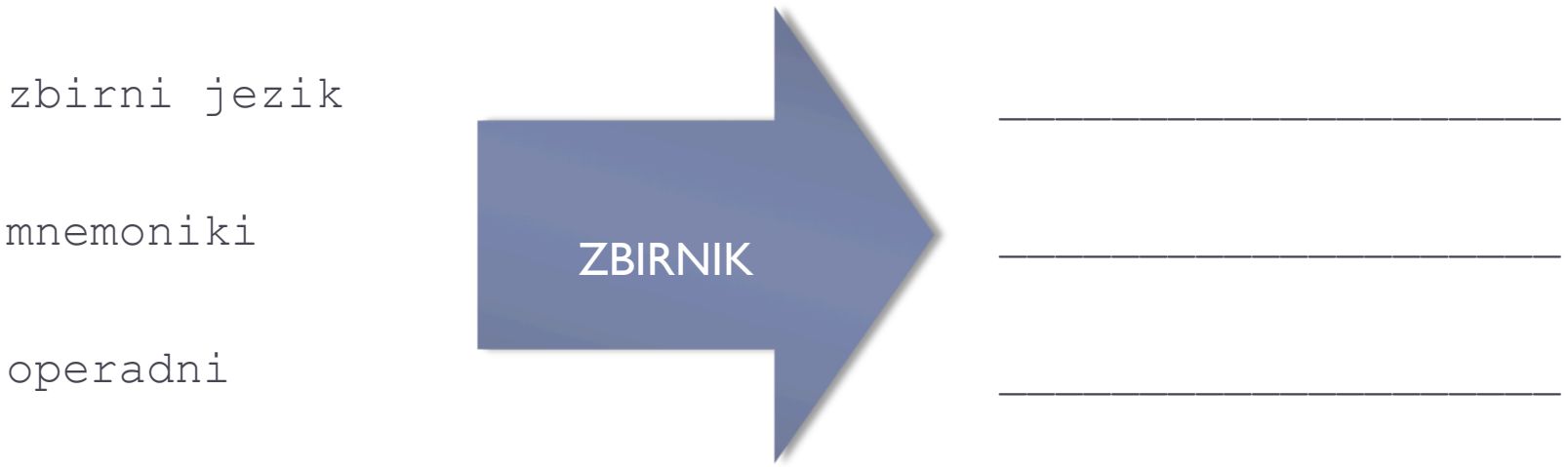
Danes zbirni jezik uporabljamo

- pri sistemskem programiranju,
- za pisanje ključnih delov posameznih programov,
- za reševanje zahtevnih, časovno potratnih problemov,
- …

Zbirni jezik je lahko ena od vmesnih stopenj prevajanja višjenivojskih programskih jezikov.

Glavna naloga zbirnika

- Zbirnik prevede programe, pisane v zbirnem jeziku, v strojni jezik



Specializirane vrste zbirnikov

- modularni zbirnik
- mikro zbirnik
- makro zbirnik
- meta zbirnik
- prečni zbirnik
- zbirnik "naloži in izvedi"

Zbirniški stavek 1/4

```
<zbirniški stavek> ::==
```

Primer:

```
ADDLP    ADDR   X, S          S = S + X
```

Zbirniški stavek 2/4

<zbirniški stavek> ::== [<oznaka stavka>] **<ločilo>**
    <mnemonik> **<ločilo>**{<operand> **<ločilo>**} [<komentar>]

- <mnemonik> - smiselna okrajšava za strojni ukaz

- <ločilo> - dogovorjeni znaki, izbrani tako, da je delo pregledovalnika čimbolj enostavno.

Zbirniški stavek 3/4

```
<zbirniški stavek> ::== [<oznaka stavka>] <ločilo>
    <mnemonik> <ločilo>{<operand> <ločilo>} [<komentar>]
```

- `<oznaka stavka>` - simbolično ime, ki se lahko pojavlja na dveh mestih:
  - _____
  - _____

Zbirniški stavek 3a/4

```
<zbirniški stavek> ::== [<oznaka stavka>] <ločilo>
    <mnemonik> <ločilo>{<operand> <ločilo>}
  [<komentar>]
```

Primer: računanje produkta `X1 * X2`

Zbirniški stavek 4/4

<zbirniški stavek> ::== [<oznaka stavka>] <ločilo>
    <mnemonik> <ločilo>{**<operand>** <ločilo>} [<komentar>]

- <operandi> **so lahko:**
  - _____
  - _____
  - _____

Osnovne naloge zbirnika 1/7

Zbirnik mora opraviti naslednje naloge:

1) Pregledati in razčleniti zbirniški stavek (pregledovalnik)

2) Mnemonike nadomestiti z operacijskimi kodami

Osnovne naloge zbirnika 2/7

3) Simbolična imena nadomestiti z ustreznimi številskimi ekvivalenti

4) Razrešiti simbolične operande

Osnovne naloge zbirnika 4/7

5) Pretvorba konstant iz izvorne oblike v notranjo strojno predstavitev.

Osnovne naloge zbirnika

6) Pravilno prevede in interpretira "psevdo ukaze"

- psevdo ukazi (direktive), kot so `START, END, BYTE, WORD, RESW, RESB`, se ne prevedejo v strojni ukaz

```
                                                    SIC object code
1.                  P1      START   0000
2.      0000        X1      WORD    5
3.      0003        X2      WORD    7
4.      0006        R       RESW    1

5.      0009        PROG    LDA     X1
6.      000C                ADD     X2
7.      000F                STA     R
8.      0012        HALT    J       HALT
9.                  END     PROG
```

Osnovne naloge zbirnika 6/7

7) Tvori strojne ukaze v predpisani obliki in izpiše kodo.

8) Pripravi in izpiše listo prevajanj
   - vhodni izvorni (zbirni) stavek,
   - objektna koda,
   - številske vrednosti, …

9) Tvori množico tabel (simbolna tabela, tabela sekcij, tabela prečnih referenc, …).

10) Razpozna vrsto specialnih možnosti

- parametrično vodeni zbirnik
- uporaba direktiv in stikal
- določi se lahko način izpisa, privzet številski sistem, format izhodne datoteke, …

Uporaba lokacijskega števca

- Zbirnik za vsak ukaz ugotovi, koliko pomnilnika zasede.
- Trenutno zasedenost pomnilnika vodi v LOKACIJSKEM ŠTEVCU (**LŠ,** angl. *LOCCTR*).
- LŠ v času zbiranja odigra podobno vlogo kot

Pregledovalnik 1/2

- Pregledovalnik bere vhodne vrstice in jih razgradi na posamezne enote.
- Razpoznane elemente okvalificira (oznaka / mnemonik / operand / komentar)
- Pregledovanje je časovno potratna operacija.
- Lahko je implementirano kot končni avtomat.

Pregledovalnik 2/2

- Izraze, ki jih pregledovalnik opredeli kot identifikatorje, obravnava naprej:
  - če gre za desni naslov, preveri prisotnost v tabelah;
  - če gre za levi naslov, zabeleži v simbolni tabeli.
- Pregledovalnik: bralna rutina + konstruktor tabel
- Za vsak simbol pregledovalnik sintaktičnemu analizatorju sporoči tip in stanje.

Tabele zbirnika

- Za pravilen potek zbiranja potrebujemo vsaj dve tabeli:
  - _____

    tabela mnemonikov in navodila zbirniku
  - _____

    vsi levi in desni naslovi
- Tabeli sta po naravi in načinu uporabe zelo različni.

Ukazna tabela 1/3

- Ukazna tabela – **UKTAB** (angl. operation code table; optab)
- **UKTAB** vsebuje seznam mnemonikov s pripadajočimi operacijskimi kodami, tipom ukaza in številom parametrov
- Primer: ukazna tabela za SIC (imamo na listih)

Mnemonik	Operacijska koda	Tip ukaza	Število operandov
LDA	00	3/4	1
FLOAT	C0	1	0
DIVR	9C	2	2
CLEAR	B4	2	1
...	...	...	...

Ukazna tabela 2/3

- Poleg mnemonikov v UKTAB pišemo še
  - _____
  - _____
- Uporaba UKTAB:
  - _____
  - _____

Ukazna tabela 3/3

- UKTAB je lahko "zapečena" v sam zbirnik, lahko pa se naloži ob začetku izvajanja
- UKTAB je statična tabela, se ne spreminja

Simbolna tabela

- Simbolna tabela – **SIMTAB** (angl. *SYMTAB*)
- Na začetku zbiranja je SIMTAB prazna!
- Pri zbiranju simbole vpisujemo v SIMTAB skupaj z njihovo vrednostjo (trenutna vrednost `LŠ` ali konstanta pri `EQU`)
- Običajna implementacija: _____

Simbolna tabela - primer

```
vrednost LŠ                    ukaz         format ukaza
-----------------------------------------------------
                               LDS    #0
                               LDA    X1
                               COMP   #0
                               JLT    ENDL
                               LDX    X2
                    ADDLP      ADDR   X, S
                               SUB    #1
                               COMP   #0
                               JGT    ADDLP
                    ENDL       RSUB

                    X1         WORD   7
                    X2         WORD   5
```

Simbolično ime	Vrednost

Uporaba simbolne tabele

- Uporaba SIMTAB: za razreševanje simboličnih imen
- Potek razreševanja:
  - _____
  - _____
- MOŽNE TEŽAVE:
  - _____
  - _____

Reševanje problema "*desni naslov pred levim*"

- Problem "*desni naslov pred levim*" rešimo na različne načine:
  - _____
  - _____

Enoprehodni zbirnik 1/5

- Z enoprehodnim zbirnikom problem "*desni naslov pred levim*" rešimo na enega od treh načinov:
  - _____
  - _____
  - _____

Enoprehodni zbirnik 2/5

Ka) Omejevanje uporabnika

- Zahteva: levi naslov se mora vedno pojaviti pred desnim

- Nekateri zbirniki omejujejo le vnaprejšnja sklicevanja na podatke

Enoprehodni zbirnik 3/5

Kb) Turbo princip ("naloži in izvedi", angl. *load-and-go*)

- Predpostavka: vsa orodja (zbirnik/povezovalnik/nalagalnik) in celoten preveden program so ves čas v delovnem pomnilniku

- Ideja:

Enoprehodni zbirnik

Kc) Naslavljanje s pomočjo simbolne tabele

- Ideja:
  - vsako ime (levo in desno) vpišemo v simbolno tabelo,
  - namesto na simbol se sklicujemo na vpis v simbolni tabeli,
  - celotno simbolno tabelo pridružimo prevedenemu programu.

- Slabosti:

Enoprehodni zbirnik 5/5

Kc) Naslavljanje s pomočjo simbolne tabele

Primer:

```
LŠ        koda programa                popravljena koda programa
-------------------------------------------------------------------
000000           LDA   #ENA
000003           J     LOOP
000006   ZANKA    ADD   #ENA
000009   LOOP    J     ZANKA

ENA        EQU   1
```

Dvoprehodni zbirnik

- Z dvoprehodnim zbirnikom problem "*desni naslov pred levim*" rešimo tako, da kodo beremo dvakrat:
  - pri prvem branju

    _____
  - pri drugem branju

    _____

Dvoprehodni zbirnik – 1. prehod

IZVORNA VRSTICA je lahko:

- UKAZ v zbirnem jeziku
- NAVODILO zbirniku

  Rezervacija (RESB, RESW, WORD, BYTE)

  Izenačitev, konstanta (EQU)

  Izhodišče (ORG)

  Konec (END)

Dvoprehodni zbirnik – 2. prehod

IZVORNA VRSTICA je lahko:

- UKAZ v zbirnem jeziku
- NAVODILO zbirniku

  Rezervacija (RESB, RESW)

  Rezervacija z inicializacijo (WORD, BYTE)

  Izenačitev, konstanta (EQU)

  Izhodišče (ORG)

  Konec (END)

Komunikacija med 1. in 2. prehodom

- Poznamo dva načina komunikacije med obema prehodoma:

- v obeh prehodih beremo izvorno kodo, edina komunikacija med prehodoma je simbolna tabela

- v prvem prehodu beremo originalno izvorno kodo in poleg simbolne tabele ustvarimo še vmesno datoteko, ki poleg izvorne kode vsebuje še
  - naslove posameznega ukaza v ukazni tabeli,
  - dodatne že razvozlane informacije o posameznem ukazu,
  - naslove simbolov v simbolni tabeli;

  v drugem prehodu beremo le vmesno datoteko.

Dvoprehodni zbirnik – 1. prehod; psevdokoda

```
Pass 1:

 begin
   read first input line
   if OPCODE = 'START' then
       begin
1          save #[OPERAND] as starting address
           initialize LOCCTR to starting address
           write line to intermediate file
           read next input line
       end {if START}
   else
       initialize LOCCTR to 0
*  while OPCODE ≠ 'END' do
       begin
           if this is not a comment line then
               begin
                   if there is a symbol in the LABEL field then
                       begin
                           search SYMTAB for LABEL
2                          if found then
                               set error flag (duplicate symbol)
                           else
                               insert (LABEL,LOCCTR) into SYMTAB
                       end {if symbol}
3                  search OPTAB for OPCODE
                   if found then
                       add 3 {instruction length} to LOCCTR
                   else if OPCODE = 'WORD' then
                       add 3 to LOCCTR
                   else if OPCODE = 'RESW' then
                       add 3 * #[OPERAND] to LOCCTR
4                  else if OPCODE = 'RESB' then
                       add #[OPERAND] to LOCCTR
                   else if OPCODE  = 'BYTE' then
                       begin
                           find length of constant in bytes
                           add length to LOCCTR
                       end {if BYTE}
                   else
                       set error flag (invalid operation code)
               end {if not a comment}
           write line to intermediate file
           read next input line
       end {while not END}
   write last line to intermediate file
5  save (LOCCTR - starting address) as program length
 end {Pass 1}
```

Dvoprehodni zbirnik – 2. prehod; psevdokoda

Pass 2:

```
begin
   read first input line {from intermediate file}
   if OPCODE = 'START' then
      begin
         write listing line
         read next input line
      end {if START}
   write Header record to object program
   initialize first Text record
*  while OPCODE ≠ 'END' do
      begin
         if this is not a comment line then
            begin
1              search OPTAB for OPCODE
               if found then
                  begin
                     if there is a symbol in OPERAND field then
                        begin
2                          search SYMTAB for OPERAND
                           if found then
                              store symbol value as operand address
                           else
                              begin
                                 store 0 as operand address
                                 set error flag (undefined symbol)
                              end
                        end {if symbol}
                     else
                        store 0 as operand address
3                    assemble the object code instruction
                  end {if opcode found}
               else if OPCODE = 'BYTE' or 'WORD' then
                  convert constant to object code
               if object code will not fit into the current Text record then
                  begin
4                    write Text record to object program
                     initialize new Text record
                  end
               add object code to Text record
            end {if not comment}
         write listing line
         read next input line
5     end {while not END}
   write last Text record to object program
   write End record to object program
   write last listing line
end {Pass 2}
```

Tvorba operacijske kode in operandov

Zbirnik na podlagi zbirniškega ukaza tvori operacijsko kodo strojnega ukaza, ki ji pridruži primerno obdelane operande.

- operacijska koda
- operand
  - _____
  - _____
  - _____
  - _____

Tvorba operacijske kode in operandov

Sestavljeni operandi:

Sestavljeni so iz

- _____
- _____
- _____

Operand se razreši (ovrednoti) z računsko rutino.

Tvorba operacijske kode in operandov

- Tvorjena operacijska koda je včasih odvisna od načina uporabe; isti mnemonik se ob različnih okoliščinah lahko prevede v različno operacijsko kodo.

- Dejavniki, ki lahko vplivajo na operacijsko kodo:
  1. _____
  2. _____
  3. _____
  4. ...

Tvorba operacijske kode in operandov

1. Sprememba operacijske kode zaradi načina naslavljanja.

Primer (Intel x86) Osnovna op. koda za ukaz `RET` je `0xC3`; koda se spremeni, če je naslov daleč od trenutnega naslova

```
RET near_addr        ...   0xC3
RET far_addr         ...   0xC8
```

Tvorba operacijske kode in operandov

2. Sprememba operacijske kode zaradi tipa operandov

Primer (Intel x86): ukaz `ADD` se prevede v `000000sd`

`s` ... size
- `0` ... velikost operanda je 8 bitov
- `1` ... velikost operand je 16 ali 32 bitov

`d` ... direction
- `0` ... prvi operand je register, drugi pomnilnik
- `1` ... prvi operand je pomnilnik, drugi register

Tvorba operacijske kode in operandov

3) Sprememba operacijske kode zaradi velikosti konstante

Primer (MC6800):

`BOR  EQU  128`
`…`
`LDA  A, BOR`
	Gre za direktno naslavljanje, zato se ta koda prevede v
	`X'9680`

`BOR  EQU  4096`
`…`
`LDA  A, BOR`
	Razširjeno naslavljanje, koda se prevede v
	`X'B61000`

Kako pa je s tem pri SICu?

Objektna koda

- Zbirnik program prevede v **objektno kodo** in jo zapiše v **objektno datoteko**; pri tem uporablja dogovorjeni **objektni format**

- Objektna datoteka se lahko uporabi kot
  - samostojen program ali
  - modul v večjem programu.

- Preden objektna datoteka dejansko zaživi, potrebujemo še povezovalnik in/ali nalagalnik.

SIC objektna datoteka

SIC uporablja preprost *tekstovni* objektni format, ki predvideva tri vrste zapisov:

1) **Zaglavje (Header – H)** vsebuje ime, začetni naslov in dolžino programa

SIC objektna datoteka

2) Programski zapis (Text – T) vsebuje prevedene ukaze (operacijske kode skupaj s pripadajočimi naslovi)

3) Zapis za konec (End – E) označuje konec programa vsebuje (opcijsko) naslov prvega izvršljivega ukaza

SIC objektna datoteka – primer 1

Program

```
Naslov    koda                                  SIC koda
-----------------------------------------------------
0000      PRVI    START    0
0000              LDA      X            00000C
0003              SUB      Y            1C000F
0006              ADD      INCR         180012
0009              STA      Z            0C0015

000C      X       WORD     X'000009'    000009
000F      Y       WORD     X'000005'    000005
0012      INCR    WORD     X'000001'    000001
0015      Z       RESW     1
```

Objektna datoteka:

SIC objektna datoteka – primer 2


```
5    1000 COPY      START     1000
10   1000 FIRST     STL       RETADR     141033
15   1003 CLOOP     JSUB      RDREC      482039
20   1006           LDA       LENGTH     001036
25   1009           COMP      ZERO       281030
30   100C           JEQ       ENDFIL     301015
35   100F           JSUB      WRREC      482061
40   1012           J         CLOOP      3C1003
45   1015 ENDFIL    LDA       EOF        00102A
…    …              …         …          manjkajo 4 ukazi
70   1024           LDL       RETADR     081033
75   1027           RSUB                 4C0000
80   102A EOF       BYTE      C'EOF'     454F46
85   102D THREE     WORD      3          000003
90   1030 ZERO      WORD      0          000000
95   1033 RETADR    RESW      1
100  1036 LENGTH    RESW      1
105  1039 BUFFER    RESB      4096

125  2039 RDREC     LDX       ZERO       041030
…                   …         …          manjka 10 ukazov
180  205A           RSUB                 4C0000
185  205D INPUT     BYTE      X'F1'      F1
190  205E MAXLEN    WORD      4096       001000

210  2061 WRREC     LDX       ZERO       041030
…                   …         …          manjkata 2 ukaza
225  206A           LDCH      BUFFER, X  509039
…                   …         …          manjkajo 3 ukazi

245  2076           RSUB                 4C0000
250  2079 OUTPUT    BYTE      X'05'      05
255                 END       FIRST
```

SIC objektna datoteka – primer 2 2/2

Objektna datoteka programa na prejšnji strani (Fig 2.1)

Posebnosti pri tvorbi kode za SIC/XE

- Tvorba objektne kode za SIC/XE se precej razlikuje od tvorbe objektne kode za SIC
  - SIC zelo preprosto

    koda je vedno 3-bajtna, en sam bit za določanje indeksnega načina naslavljanja
  - SIC/XE mnogo bolj zapleteno
    - več formatov ukazov

      koda je lahko 1-, 2-, 3- ali 4-bajtna
    - več načinov naslavljanja

      identificirati je treba način naslavljanja in modificirati osnovno kodo ukaza

Posebnosti tvorbe SIC/XE kode si bomo ogledali na primeru iz slike 2.5

```
5     0000 COPY       START     0
10    0000 FIRST      STL       RETADR
12    0003            LDB       #LENGTH
13                    BASE      LENGTH
15    0006 CLOOP      +JSUB     RDREC
20    0009            LDA       LENGTH
…
40    0017            J         CLOOP
…
55    0020            LDA       #3
…
70    002A            J         @RETADR
…
95    0030 RETADR     RESW      1
100   0033 LENGTH     RESW      1
105   0036 BUFFER     RESB      4096
…
125   1036 RDREC      CLEAR     X
…
133   103C            +LDT      #4096
…
150   1049            COMPR     A,S
…
160   104E            STCH      BUFFER, X
…
175   1056 EXIT       STX       LENGTH
```

Del programa iz slike 2.5 (SIC/XE)

Posebnosti pri tvorbi kode za SIC/XE
Registrsko-registrski ukazi

- Zbirnik mora poznati numerične vrednosti (interna reprezentacija) vseh registrov (`A=0, X=1, L=2, B=3, S=4, T=5, F=6`),
- registre v številke spremeni v 2. fazi zbiranja,
- lahko uporabi simbolno tabelo (pred začetkom zbiranja jo napolni z imeni in vrednostmi za vse registre).

Primer: prevajanje ukazov v vrsticiah 125 in 150:

```
125  1036 RDREC   CLEAR   X        ________
150  1049 COMPR   A,S              ________
```

Posebnosti pri tvorbi kode za SIC/XE
Relativno naslavljanje

- Večina registrsko-pomnilniških ukazov se prevede bodisi kot PC-relativno bodisi kot bazno-relativno naslavljanje.
- V obeh primerih mora zbirnik izračunati *odmik*
- Odmik mora biti dovolj majhen
  - predznačeno (-2048…2047) za PC relativno
  - nepredznačeno (0…4095) za bazno-relativen način.
- Način naslavljanja izbere zbirnik; najprej poskusi s PC-relativnim

Posebnosti pri tvorbi kode za SIC/XE
Relativno naslavljanje

Računanje odmika pri *-relativnem naslavljanju poteka obratno kot računanje uporabnega naslova (UN).

Posebnosti pri tvorbi kode za SIC/XE
Relativno naslavljanje

PC-relativno naslavljanje

Naslov se izračuna glede na vrednost PC

Primer:

```
10      0000   FIRST  STL           RETADR        _____

40      0017          J             CLOOP         _____
```

```
6 - 26    = -20
0x006-0x001A = 0x0FEC
```

Posebnosti pri tvorbi kode za SIC/XE
Relativno naslavljanje

Bazno-relativno naslavljanje

- naslov se določi glede na vrednost baznega registra
- zbirnik vrednosti baznega registra ne pozna, uporabnik mu jo sporoči s posebnim navodilom (direktivo) `BASE`

Primer:

```
12      0003            LDB             #LENGTH
13                      BASE            LENGTH
...
160     104E            STCH            BUFFER, X     _____
```

Posebnosti pri tvorbi kode za SIC/XE
Relativno naslavljanje

Bazno-relativno naslavljanje

- Opomba 1: v primeru, da bazni register uporabljamo za drug namen (recimo: začasno shranjevanje rezultata aritmetičnih operacij) moramo to zbirniku sporočiti! Uporabimo direktivo `NOBASE`.

- Opomba 2: vrstici

```
20    0009          LDA LENGTH
175   1056  EXIT    STX LENGTH
```

se kljub podobnosti prevedeta različno (prva s PC-relativnim, druga z bazno-relativnim naslavljanjem)

Posebnosti pri tvorbi kode za SIC/XE
Uporaba 4-zlogovnega razširjenega formata

- Kadar je odmik prevelik za uporabo *-relativnega naslavljanja, se uporabi 4-zlogovni zapis ukaza (20 bitov za opis naslova).

- Programer mora uporabo razširjenega formata zbirniku sporočiti z uporabo znaka + pred ukazom

Primer:

```
15  0006 CLOOP      +JSUB   RDREC     _________
```

Posebnosti pri tvorbi kode za SIC/XE
Takojšnje naslavljanje

- Uporabimo znak # pred operandom

- Za zbirnik je takojšnje naslavljanje zelo preprosto opravilo: operand pretvori v notranjo predstavitev in ga vstavi v operacijsko kodo.

Primeri:

```
55    0020      LDA      #3          _____

133   103C      +LDT     #4096       _____

12    0003      LDB      #LENGTH     _____
```

Posebnosti pri tvorbi kode za SIC/XE
Posredno naslavljanje

- Uporabimo znak @ pred operandom

Primer:

```
70      002A    J       @RETADR    _______
```

Naloga za vajo

- "Ročno" prevedi program in ustvari objektno datoteko.

```
SESTEJ      START       0
            LDS         #3
            LDT         #9
            LDX         #0
ZANKA       LDA         ALFA,X
            ADD         BETA,X
            STA         GAMA,X
            ADDR        S,X
            COMPR       X,T
            JLT         ZANKA
KONEC       J           KONEC

ALFA        WORD        1
            WORD        2
            WORD        3
BETA        WORD        4
            WORD        5
            WORD        6
GAMA        RESW        3
```

Naloga za vajo – preveri pravilnost rešitve

Ukaz `java -cp sictools.jar sic.Asm sestej.asm` ustvari `lst` datoteko:

```
01000           SESTEJ  START   4096
01000  6D0003           LDS     #3
01003  750009           LDT     #9
01006  050000           LDX     #0
01009  03A010   ZANKA   LDA     ALFA,X
0100C  1BA016           ADD     BETA,X
0100F  0FA01C           STA     GAMA,X
01012  9041             ADDR    S,X
01014  A015             COMPR   X,T
01016  3B2FF0           JLT     ZANKA
01019  3F2FFD   KONEC   J       KONEC

0101C  000001   ALFA    WORD    1
0101F  000002           WORD    2
01022  000003           WORD    3
01025  000004   BETA    WORD    4
01028  000005           WORD    5
0102B  000006           WORD    6
0102E  00....00 GAMA    RESW    3
```

Naloga za vajo – preveri pravilnost rešitve

Ukaz `java -cp sictools.jar sic.Asm sestej.asm` ustvari `obj` datoteko:

```
HSESTEJ001000000037
T0010001F6D0003750009050000003A0101BA0160FA01C9041A0153B2FF03F2FFD000001
T00101F0F000002000003000004000005000006
E001000
```