5d
Splošne lastnosti ukazov

PO KNJIGI - DUŠAN KODEK: ARHITEKTURA IN ORGANIZACIJA RAČUNALNIŠKIH SISTEMOV

5 dimenzij lastnosti ukazov

Dimenzija
1. Način shranjevanja operandov v CPE
2. Število eksplicitnih operandov v ukazu
3. Lokacija operandov in načini naslavljanja
4. Operacije
5. Vrsta in dolžina operandov

D1. Načini shranjevanja operandov v CPE

- 3 načini shranjevanja operandov v CPE:
  1. **Akumulator**
     - najstarejši način
     - edini register
       - zato ga v ukazih ni treba eksplicitno navajati
     - ukaza LOAD, STORE za prenos v in iz akumulatorja
     - veliko prometa z GP (shranjevanje vmesnih rezultatov), zato počasnost

2. Sklad (stack)

- v danem trenutku je dostopna samo najvišja lokacija
  - podobno kot sklad pladnjev
- LIFO
- ukaza PUSH, POP (ali PULL)
- podobno akumulatorju (takoj dostopen le 1 operand)
  - preprosta realizacija, kratki ukazi, preprosti prevajalniki
  - vendar je prostora za več operandov

3. Množica registrov (register set)

- Najbolje (danes edina rešitev)
  - nekdaj dragi, pa tudi prevajalniki jih niso znali dobro uporabljati
- Register je skupina pomnilnih celic, ki imajo skupne krmilne signale
  - Vsak register ima svoj naslov
- Namen: shranjevanje vmesnih rezultatov
  - pri skladu: v pomnilnik
- 2 rešitvi:
  - splošnonamenski registri (vsi ekvivalentni)
  - 2 skupini: za operande, za naslove
- 2 vrsti:
  - programsko nedostopni
  - programsko dostopni
    - programer jih lahko uporablja kot nek hiter pomnilnik

- **Programsko dostopni registri**
  - majhen pomnilnik, v katerega lahko shranimo operande
  - prednosti pred GP:
    1. Hitrost
       - registri so hitrejši od GP
       - bližji so aritmetično-logični in kontrolni enoti
       - možen je istočasen dostop do več registrov naenkrat
    2. Krajši ukazi
       - krajši naslov (ker je registrov malo) kot pri GP

D2: Število eksplicitnih operandov v ukazu

- **m-operandni** računalnik
  - običajno se podajajo naslovi operandov
  - danes m največ 3
- 4 skupine:
  - **3-operandni**

$$OP3 \leftarrow OP2 + OP1$$

    - operandi so običajno v registrih

- **2-operandni**
  - enostavnejši, a malo počasnejši

$$OP2 \leftarrow OP2 + OP1$$

- **1-operandni**
  - akumulator

$$AC \leftarrow AC + OP1$$

  - mikroprocesorji iz 70. in 80. let
    - Intel 8080, Motorola 6800, Zilog Z80
    - Intel 8086, Intel 80186, Intel 80286

- **Brez-operandni (skladovni)**
  - najkrajši ukazi

$$Sklad_{VRH} \leftarrow Sklad_{VRH} + Sklad_{VRH-1}$$

  - toda: potrebna sta vsaj 2 ukaza z ekspl. operandom!
    - PUSH, POP (prenos med GP in skladom)

D3: Lokacija operandov in načini naslavljanja

- 2 vprašanji:
  - Kje so operandi?
  - Kako je v ukazu podana informacija o njih?

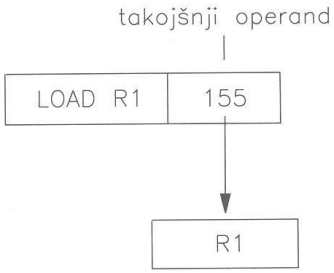
- **Lokacija operandov**
  - registri CPE
  - GP
  - (registri krmilnika V/I naprave)

- 2- in 3-operandni računalniki se delijo še na:
  - **registrsko-registrske** računalnike
    - najbolj razširjeni
    - vsi operandi v registrih CPE
    - reče se tudi **load/store** računalniki (ker rabimo load in store)
  - **registrsko-pomnilniške**
    - en operand v registru, drugi *lahko* v pomnilniku
  - **pomnilniško-pomnilniške**
    - vsak operand *lahko* v pomnilniku
    - zapleteni ukazi, CISC (npr. VAX)

Načini naslavljanja

- Načini naslavljanja: Kako je v ukazu podana informacija o operandih
  - Tičejo se predvsem pomnilniških operandov
    - pri registrskih je enostavno

1. **Takojšnje naslavljanje** (immediate addressing)
   - operand je v ukazu podan z vrednostjo (je del ukaza)
   - **takojšnji operandi (literali)** so kar konstante
     - LOAD R1, 155, (R1 ← 155)
     - ADD R1, 3 (R1 ← R1 + 3)

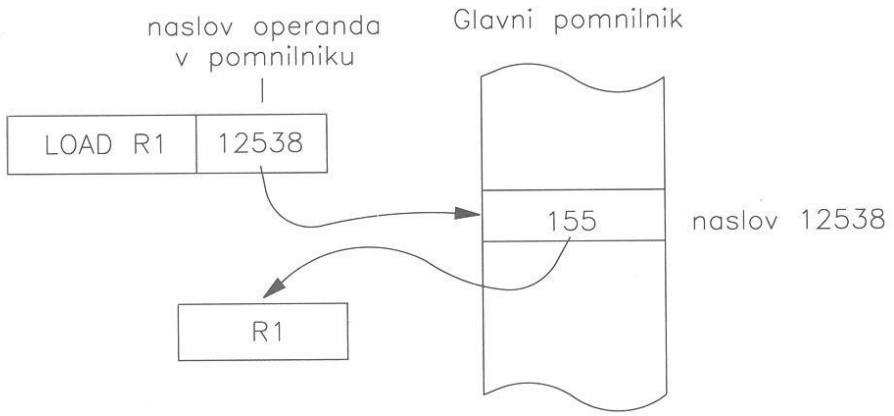


2. **Neposredno naslavljanje** (direct addressing)

- operand je podan z naslovom
  - če je to naslov registra, je to **registrsko naslavljanje**
  - če je to naslov v GP, je to **(neposredno) pomnilniško naslavljanje**
- primerno za operande, ki se jim ne spreminjajo naslovi

Registrsko: ADD R1, R2

Pomnilniško: LOAD R1, (12538) ali pa ADD R1, (1001)



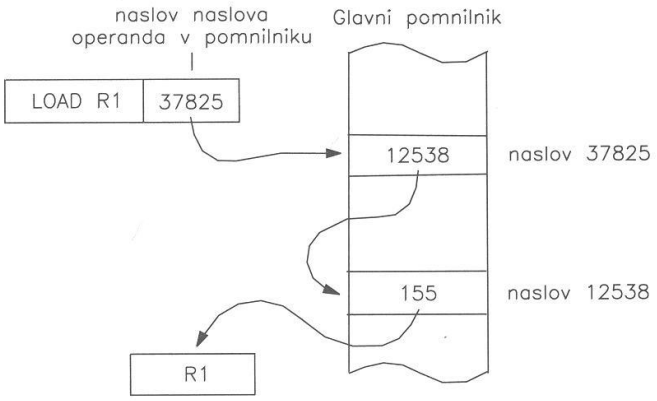
- Težave:
  - velik naslovni prostor → dolg naslov → dolgi ukazi
  - povečanje pom. prostora → drugačni ukazi → nezdružljivost za nazaj
  - primeri, ko operand ni na stalnem naslovu

3. **Posredno naslavljanje** (indirect addressing)

- v ukazu je naslov lokacije, na kateri je shranjen naslov operanda
  - **Pomnilniško posredno naslavljanje**, če gre za naslov pomnilniške lokacije (nerodno, ni pogosto)
    - ADD R1,@(1001) R1 ← R1 + M[M[1001]]
  - **Registrsko posredno naslavljanje**, če gre za naslov registra
    - uporablja se tudi **odmik** (displacement)
      - iz obojega se izračuna pomnilniški naslov
    - imenuje se tudi **relativno naslavljanje**
      - naslov operanda določen relativno na vsebino registra
    - najpogostejši način naslavljanja

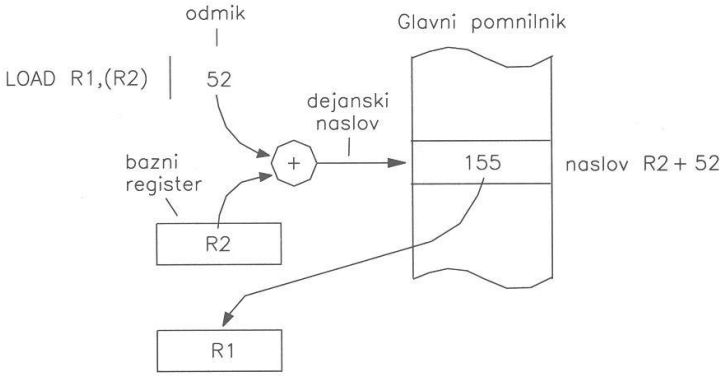
Posredno naslavljanje:

pomnilniško



a) Pomnilniško posredno naslavljanje

registrsko



Glavne vrste relativnega naslavljanja

3.1 Bazno naslavljanje (base addressing)

- reče se tudi **naslavljanje z odmikom** (displacement addressing)
- najpogostejše
- naslov operanda A = R2 + D
  - k vsebini registra R2 prištejemo odmik D
- R2 je **bazni register**, A pa **dejanski naslov** (effective address)
- Npr.: ADD R1,100(R2) R1 ← R1 + M[R2+100]
- Če D=0: Bazno brez odmika
  - ADD R1,(R2) R1 ← R1 + M[R2]

3.2 **Indeksno naslavljanje** (indexed addressing)

- odmik D
- $A = R2 + R3 + D = R2 + D_1$
- R3 je indeksni register
- glavno področje uporabe so polja, strukture in seznami
  - elementi se običajno obdelujejo zaporedoma po naraščajočih (ali padajočih) indeksih, zato sta pogosti operaciji

    $R3 \leftarrow R3 + \Delta$ in $R3 \leftarrow R3 - \Delta$
  - Δ je dolžina operanda, merjena v številu pomnilniških besed (*korak indeksiranja*)
- Npr.:
  - ADD R1,100(R2+R3), R1 $\leftarrow$ R1 + M[R2+R3+100] (dostop do elementov polja)

3.3 Pred-dekrementno naslavljanje (pre-decrement addressing)

- $R3 \leftarrow R3 - \Delta$
- A = R2 + D ali A = R2 + R3 + D
- bazno ali indeksno

3.4 Po-inkrementno naslavljanje (post-increment addressing)

- A = R2 + D ali A = R2 + R3 + D
- $R3 \leftarrow R3 + \Delta$

3.5 Velikostno indeksno naslavljanje (scaled indexed addressing)

- $A = R2 + R3 \times \Delta + D$
- dovolj je inkrementirati R3

- Pred-dekrementno in po-inkrementno naslavljanje v paru tvorita **skladovno naslavljanje** (stack addressing)
  - sklad je v GP
  - določeni računalniki imajo register **skladovni kazalec** (stack pointer)

Še 2 pojma:

- **Pozicijsko neodvisno naslavljanje**
  - pozicijsko neodvisni programi
    - lahko jih premestimo v drug del pomnilnika
    - ne smejo vsebovati absolutnih naslovov
      - neposredno, pomnilniško posredno nasl.
  - možna rešitev je preslikovanje naslovov
    - če program ni pozicijsko neodvisen

- **PC-relativno naslavljanje**
  - kot bazni register služi kar programski števec (PC)

D4: Operacije

- Operacije niso ključnega pomena
  - Npr., možno je narediti računalnik, ki ima en sam ukaz:

    SBN A,B,C

    Pomen: M[A] ← M[A] - M[B]; če M[A] < 0, skoči na C

- Operacij je manj kot ukazov
- Imena ukazov so **mnemoniki**
  - okrajšava ang. imena ukaza
    - vsebuje tudi operacijo
    - npr. A, D, AD, ADD, S ... za seštevanje v fiksni vejici

Skupine operacij

1. **Prenosi podatkov** (data transfer)
   - izvor, ponor
   - v resnici gre za kopiranje
   - Običajni **mnemoniki**:
     - LOAD: GP $\rightarrow$ R
     - STORE: R $\rightarrow$ GP
     - MOVE: R $\rightarrow$ R ali GP $\rightarrow$ GP
     - PUSH: GP ali R $\rightarrow$ Sklad
     - POP (PULL): Sklad $\rightarrow$ GP ali R
   - tudi CLEAR in SET

2. Aritmetične in logične operacije

- izvajajo se v ALE (nad operandi v fiksni vejici)
- Aritmetične operacije: seštevanje, odštevanje, množenje, deljenje, aritm. negacija, absolutna vrednost, inkrement, dekrement
  - za vsako je več ukazov (različne dolžine operandov)
- Logične operacije: AND, OR, NOT, XOR, pomiki

3. Kontrolne operacije

- spreminjajo vrstni red ukazov

3.1 Pogojni skoki (conditional branches).

- 3 načini za izpolnjenost pogoja:
  - **Pogojni biti** se postavijo kot rezultat določenih operacij.
    - Z (zero), N (negative), C (carry), V (overflow), itd.
    - Npr. ukaz BEQ (branch if equal) skoči, če je Z=1
  - **Pogojni register**
    - poljuben register
    - Npr. ali je njegova vsebina 0
  - **Primerjaj in skoči** (compare and branch)
    - skok, če je primerjava izpolnjena

3.2 Brezpogojni skoki (unconditional branch, jump)

3.3 Klici in vrnitve iz podprogramov

- ukaz za klic podprograma mora shraniti **povratni naslov** (return address)
- tipična mnemonika sta CALL in JSR (jump to subroutine)
- RET (return) za vrnitev

4. **Operacije v plavajoči vejici**.
   - izvaja jih posebna enota (FPU – Floating Point Unit), ki ni del ALE
   - poleg osnovnih štirih operacij so še koren, logaritem, eksponentna in trigonometrične funkcije

5. **Sistemske operacije**.
   - vplivajo na način delovanja računalnika
   - običajno spadajo med **privilegirane ukaze**

6. **Vhodno/izhodne operacije.**
   - obstajajo na nekaterih računalnikih
     - na drugih se uporabljajo običajni ukazi za prenos podatkov
   - prenosi med GP in V/I ter med CPE in V/I

- Ukaze lahko delimo tudi na
  - **skalarne** in
  - **vektorske**
    - na vektorskih računalnikih se lahko ista operacija izvrši na *N* skupinah operandov
    - pri skalarnih je treba za to uporabiti zanko
    - vektorske ukaze srečamo na superračunalnikih

D5: Vrsta in dolžina operandov

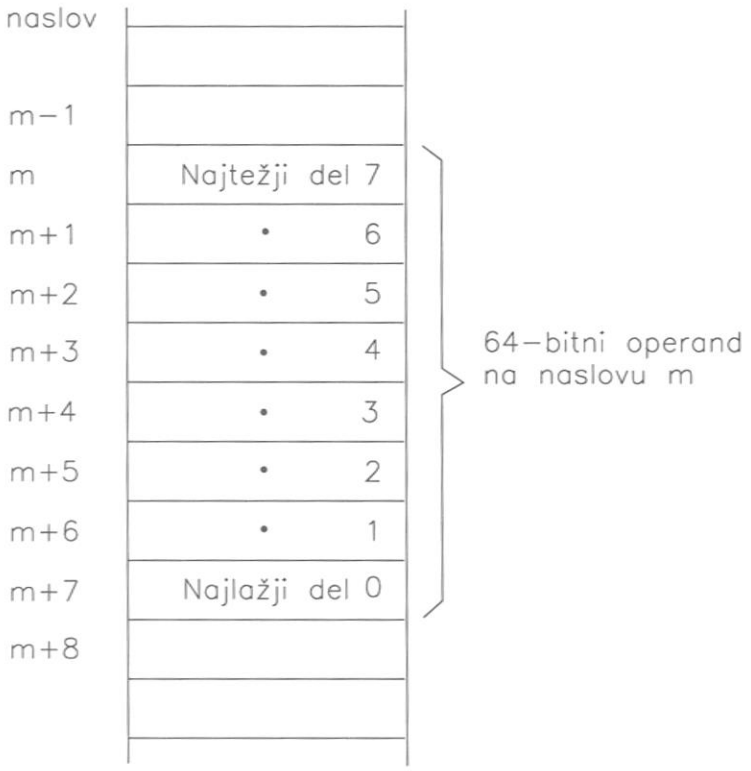
- Vrste operandov:

1. **bit**
   - v višjih jezikih jih običajno ni
   - koristno pri sistemskih operacijah
2. **znak**
   - običajno 8-bitni ASCII
   - več znakov tvori **niz** (string)
3. **celo število**
   - predznačeno ali nepredznačeno
   - dolžine 8, 16, 32, 64 bitov
4. **realno število**
   - št. v plavajoči vejici (običajno po standardu IEEE 754)
   - enojna natančnost 32 bitov, dvojna natančnost 64 bitov; obstajajo tudi 128-bitna
5. **desetiško število**
   - v 8 bitih 2 BCD števili ali 1 ASCII znak

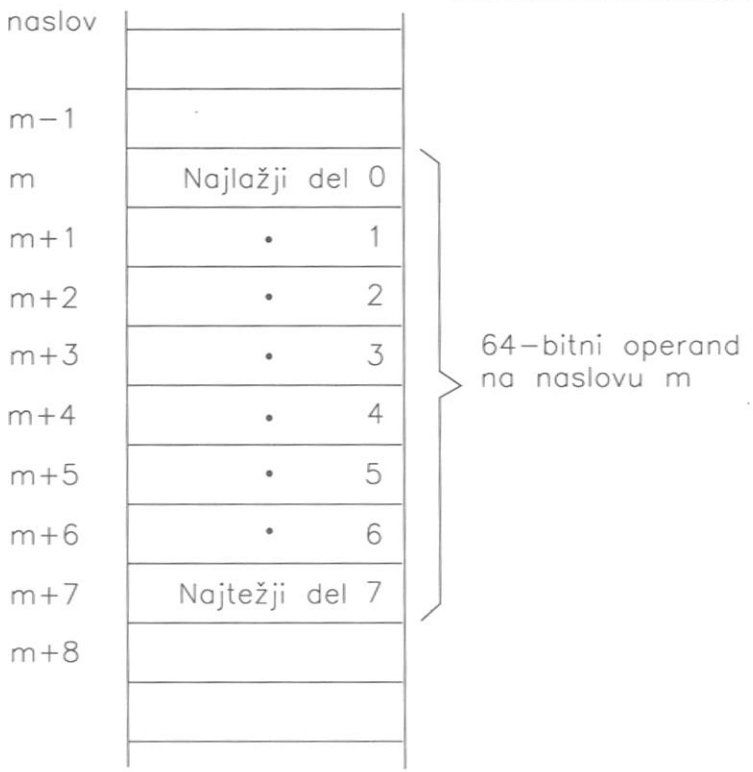
- Operandi dolžin večkratnikov 2 imajo posebna imena:
  - 8 Bajt (byte)
  - 16 Polovična beseda (halfword)
  - 32 Beseda (word)
  - 64 Dvojna beseda (double word)
  - 128 Štirikratna beseda (quad word)

    - to sicer ne velja za vse računalnike

- **Sestavljeni pomnilniški operandi** so sestavljeni iz več pomnilniških besed
  - v pomnilniku morajo biti na zaporednih lokacijah, sicer bi težko podali naslov takega operanda
- Obstajata 2 načina (glede na vrstni red), kako jih shranimo v pomnilnik:
  - **pravilo debelega konca** (Big Endian Rule)
    - najtežji del operanda na najnižjem naslovu
  - **pravilo tankega konca** (Little Endian Rule)
    - najlažji del operanda na najnižjem naslovu



a) Pravilo debelega konca (Big Endian)



b) Pravilo tankega konca (Little Endian)

Problem poravnanosti

- pomnilnik, ki omogoča dostop do 8 8-bitnih besed hkrati, je narejen kot 8 paralelno delujočih pomnilnikov
- istočasen dostop do *s* besed dolgega operanda na naslovu *A* je možen le, če je *A* deljiv z *s* (*A* mod *s* = 0)
  - pri 8-bitni pomnilniški besedi mora imeti 64-bitni operand zadnje 3 bite enake 0
    - **poravnan** (aligned) operand
    - sicer **neporavnan** (misaligned)
      - potreben več kot en dostop
      - pri nekaterih računalnikih se sproži past

Zgradba ukazov

Ukaz je shranjen v eni ali več (sosednih) pomnilniških besedah

Vsak ukaz vsebuje

1. Operacijsko kodo (informacijo o operaciji, ki naj se izvrši)
2. Informacijo o operandih, nad katerimi naj se izvrši operacija

- **Zgradba ali format ukaza**
  - pove, kako so biti ukaza razdeljeni na operacijsko kodo in operande
    - število polj, njihova velikost in pomen posameznih bitov v njih
- Možni so različni formati
- Parametri, ki najbolj vplivajo na format:
  1. Dolžina pom. besede
     - pri 8: dolžina ukaza večkratnik 8
     - pri dolgih pom. besedah: dolžina ukaza ½ ali ¼ besede
  2. Število eksplicitnih operandov v ukazu
  3. Vrsta in število registrov v CPE
     - št. registrov vpliva na št. bitov za naslavljanje
  4. Dolžina pom. naslova
     - predvsem, če se uporablja neposredno naslavljanje
  5. Število operacij

- Optimalne rešitve za format ukazov ni
  - kaj je kriterij?
  - neke vrste umetnost
  - medsebojna odvisnost parametrov
  - možno je minimizirati velikost programov
    - pogostost ukazov, Huffmanovo kodiranje
    - v praksi se ni izkazalo (Burroughs)

- 3 načini:

1. Spremenljiva dolžina
   - št. eksplicitnih operandov spremenljivo
   - različni načini naslavljanja
   - veliko formatov
     - npr. 1..15 bajtov pri 80x86, 1..51 VAX
   - kratki formati za pogoste ukaze

Op. koda	Način naslavljanja 1	Naslovno polje 1	. . .	Način naslavljanja n	Naslovno polje n

2. Fiksna dolžina
   - št. eksplicitnih operandov fiksno
   - majhno št. formatov (RISC)
     - Alpha, ARM, MIPS, PowerPC, SPARC

Op. koda	Naslovno polje 1	Naslovno polje 2	Naslovno polje 3

3. Hibridni način

Op. koda	Način naslavljanja	Naslovno polje

Op. koda	Naslovno polje 1	Način naslavljanja 2	Naslovno polje 2

Op. koda	Način naslavljanja	Naslovno polje 1	Naslovno polje 2

- **Ortogonalnost ukazov** (medsebojna neodvisnost parametrov ukaza)
  1. Informacija o operaciji neodvisna od info. o operandih
  2. Informacija o enem operandu neodvisna od info. o ostalih operandih

Število ukazov in RISC

- **CISC računalniki**
  - Complex Instruction Set Computer
  - imajo veliko število ukazov
  - IBM 370, VAX, Intel
- **RISC računalniki**
  - Reduced Instruction Set Computer
  - imajo majhno število ukazov
  - MIPS, ARM, DEC Alpha, IBM/Motorola Power PC
- Oboji imajo svoje prednosti in slabosti
  - na začetku so bili računalniki tipa CISC, RISC pa so se pojavili kasneje
  - RISC so enostavnejši in imajo hitrejše ukaze, vendar pa program potrebuje več ukazov

- 2 ugotovitvi v 80. letih:
  1. **Stalno povečevanje števila ukazov**
     - IAS (1951): 23 ukazov in 1 način nasl.
     - 70. leta: stotine ukazov
  2. **Velik del ukazov redko uporabljan**

Razlogi za povečevanje števila ukazov

- Semantični prepad
  - v 60. letih so proizvajalci zato povečevali št. ukazov
- Mikroprogramiranje
  - dodajanje novih ukazov preprosto
- Razmerje med hitrostjo CPE in GP
  - faktor vsaj 10
  - kompleksen ukaz hitrejši kot zaporedje preprostih ukazov

Razlogi za zmanjševanje števila ukazov

- Težave prevajalnikov
  - velik del ukazov redko uporabljan
- Pojav predpomnilnikov
  - v primeru zadetka v PP je dostop skoraj enako hiter kot do mikroukazov
- Uvajanje paralelizma v CPE
  - cevovod (lažja realizacija pri preprostih ukazih)

Definicija arhitekture RISC

- Večina ukazov se izvrši v enem ciklu CPE
  - lažja real. cevovoda
- Registrsko-registrska zasnova (load/store)
  - zaradi zahteve 1
- Ukazi realizirani s trdo ožičeno logiko
  - ne mikroprogramsko
- Malo ukazov in načinov naslavljanja
  - hitrejše in enostavnejše dekodiranje in izvrševanje
- Enaka dolžina ukazov
- Dobri prevajalniki
  - upoštevajo zgradbo CPE