

# Asemblersko programiranje za x86\_64 arhitekturu

Nikola Milev

Poslednja izmena: 29/12/2017

## Sintaksa

Na kursu ćemo koristiti Intelovu neprefiksnu sintaksu.

Svaki red assemblera može biti:

- Prazan red: prazni redovi se ignorišu
- Direktiva
- Linije koje nisu ni prazne ni direktive smatraju se instrukcijama

Pri nailasku na simbol #, ostatak linije se ignoriše (komentari). Svaka linija može početi labelom.

## Labele

Definicija labele sastoji se iz identifikatora iza kog se navodi simbol :. Identifikator mora početi slovom ili simbolom \_, dok može sadržati slova, simbol \_ i cifre. Labele se prilikom prevođenja programa prevode u memorijske adrese. Labele mogu označavati adrese podataka, kao i adrese instrukcija.

## Direktive

Direktive počinju simbolom . i imaju specijalno značenje.

- *.intel\_syntax noprefix* – Označava se da se koristi Intelova neprefiksna sintaksa
- *.global identifikator* ili *.global identifikator* – Navodi se da je *identifikator* globalni simbol
- *.data* – Počinje se sekcija inicijalizovanih podataka
- *.text* – Počinje se sekcija koda
- *.ascii* – Kreira se ASCII niska na čijem se kraju automatski navodi terminirajuća nula
- *.byte* – Kreira se jedan ili niz bajtova; članovi niza razdvojeni su zapetom
- *.word* – Kreira se jedan ili niz slogova od 2 bajta
- *.long* – Kreira se jedan ili niz slogova od 4 bajta
- *.quad* – Kreira se jedan ili niz slogova od 8 bajtova

## Instrukcije

Instrukcija se sastoji od koda instrukcije i operan(a)da. Svaki kod instrukcije ima svoju simboličku oznaku. Opšti oblik instrukcije sa dva operanda je: *kod op1, op2*. Načini zadavanja operanada:

- Registarski operandi: navodi se simbolička oznaka registra
- Neposredni operandi: direktno se navodi vrednost sa kojom se radi
- Memorijski operandi: navodi se adresa sa kojom se radi. Opšti sintaksni oblik je:  $[B + S * I + D]$ .  $B$  je bazna adresa,  $D$  je pomeraj,  $I$  je indeks, dok je  $S$  veličina "elementa". Svaki od navedenih elemenata može se izostaviti i tada se dobijaju specijalni slučajevi:
  - $[B]$  – Bazno adresiranje
  - $[B + D]$  – Bazno adresiranje sa pomerajem
  - $[B + S * I]$  – Indeksno adresiranje

# Registri

U toku kursa, korišćiće se registri opšte namene, veličine 8 bajtova (64 bita). Njih ima 16:

- *rax, rbx, rcx, rdx, rsi, rdi, rsp, rbp, r8, ..., r15*

Moguć je pristup niža 4 bajta:

- *eax, ebx, ecx, edx, esi, edi, esp, ebp, r8d, ..., r15d*

Moguć je pristup najniža 2 bajta:

- *ax, bx, cx, dx, si, di, sp, bp, r8w, ..., r15w*

Moguć je pristup najnižem bajtu:

- *al, bl, cl, dl, sil, dil, spl, bpl, r8b, ..., r15b*

Registri *rbp* i *rsp* imaju specijalnu svrhu pri radu sa stekom:

- Registar *rbp* služi za čuvanje trenutnog okvira steka.
- Registar *rsp* služi za čuvanje trenutnog vrha steka. Pri izvršavanju instrukcije *pop*, vrednost registra *rsp* automatski se uvećava za veličinu operanda, dok se pri instrukciji *push* vrednost registra *rsp* automatski umanjuje za veličinu operanda.

## Instrukcije

### Instrukcije za rad sa stekom

- *push op* – Na vrh steka smešta se vrednost sadržana u operandu. Registar *rsp* umanjuje se za veličinu operanda. Na stek je moguće smestiti samo dvobajtne i osmobajtne podatke.
- *pop op* – Sa vrha steka uklanja se vrednost i smešta se u operand. registar *rsp* uvećava se za veličinu operanda. Sa steka je moguće ukloniti samo dvobajtne i osmobajtne podatke.

### Instrukcije transfera

- *mov op1, op2* – U prvi operand smešta se vrednost drugog operanda. Očekuje se da oba operanda budu iste širine.
- *movzx op1, op2* – U prvi operand smešta se vrednost drugog operanda, proširena nulama (transfer sa neoznačenim proširivanjem). Drugi operand ne može biti dat neposredno. Očekivano je da je drugi operand manje širine od prvog.
- *movsx op1, op2* – U prvi operand smešta se vrednost drugog operanda, proširena u skladu sa znakom (transfer sa označenim proširivanjem). Drugi operand ne može biti dat neposredno. Očekivano je da je drugi operand manje širine od prvog.
- *lea op1, op2* – U prvi operand smešta se adresa koja je predstavljena drugim operandom.

### Aritmetičke instrukcije

- Sabiranje: *add op1, op2* – sabira argumente i rezultat smešta u prvi argument
- Oduzimanje: *sub op1, op2* – oduzima argumente i rezultat smešta u prvi argument
- Neoznačeno množenje: *mul op* – Ukoliko je operand osmobajtni, neoznačeno množi sadržaj registra *rax* operandom *op* i rezultat se smešta u *rdx : rax*. Ukoliko je operand četvorobajtni, neoznačeno množi sadržaj registra *eax* operandom *op* i rezultat se smešta u *edx : eax*.
- Označeno množenje: *imul op* – Ukoliko je operand osmobajtni, označeno množi sadržaj registra *rax* operandom *op* i rezultat se smešta u *rdx : rax*. Ukoliko je operand četvorobajtni, označeno množi sadržaj registra *eax* operandom *op* i rezultat se smešta u *edx : eax*.
- Proširivanje znaka: *cdqe* – označeno proširivanje registra *eax* na *rax*
- Proširivanje znaka: *cdq* – označeno proširivanje registra *eax* na *edx : eax*
- Proširivanje znaka: *cqo* – označeno proširivanje registra *rax* na *rdx : rax*
- Neoznačeno deljenje: *div op* – ukoliko je operand osmobajtni, neoznačeno deli sadržaj registara *rdx : rax* operandom *op*; količnik se nalazi u *rax* dok se ostatak pri deljenju nalazi u *rdx*. Ukoliko je operand četvorobajtni, neoznačeno deli sadržaj registara *edx : eax* operandom *op*; količnik se nalazi u *eax* dok se ostatak pri deljenju nalazi u *edx*.

- Označeno deljenje: *div op* –ukoliko je operand osmobaajtni, označeno deli sadržaj registara *rdx : rax* operandom *op*; količnik se nalazi u *rax* dok se ostatak pri deljenju nalazi u *rdx* . Ukoliko je operand četvorobaajtni, označeno deli sadržaj registara *edx : eax* operandom *op*; količnik se nalazi u *eax* dok se ostatak pri deljenju nalazi u *edx*.
- Negiranje: *neg op*
- Uvećanje operanda za 1: *inc op*
- Umanjenje operanda za 1: *dec op*

## Logičke instrukcije

- Konjunkcija: *and op1, op2*
- Disjunkcija: *or op1, op2*
- Ekskluzivna disjunkcija: *xor op1, op2*
- Negacija: *not op*
- Šiftovanje (pomeranje) u levo: *shl op1, op2* – pomera prvi argument za broj mesta koji je sadržan u drugom argumentu. Drugi argument mora biti dat neposredno.
- Šiftovanje (pomeranje) u desno:
  - Logičko: *shr op1, op2* ; Drugi argument mora biti dat neposredno.
  - Aritmetičko: *sar op1, op2* ; Drugi argument mora biti dat neposredno.

## Instrukcije poređenja

- *cmp* – poređenje operanada oduzimanjem; ne menja operande nego *rflags* registar
- *test* – bitovsko poređenje operanada konjukcijom; ne menja operande nego *rflags* registar

## Instrukcije kontrole toka

- Bezuslovni skok: *jmp op* – bezuslovni skok na adresu
- Poziv funkcije: *call op* – bezuslovni skok uz pamćenje povratne adrese na steku
- *ret* – skidanje adrese sa steka i skok na nju
- *je op* – Skok na adresu ukoliko je rezultat prethodne instrukcije nula
- *je op* – Skok na adresu ukoliko je rezultat prethodnog poređenja jednako (ekvivalentno sa *je*)
- *jnz op* – Skok na adresu ukoliko rezultat prethodnog poređenja nije nula
- *jne op* – Skok na adresu ukoliko rezultat prethodnog poređenja nije jednako (ekvivalentno sa *jnz*)
- *ja op* – Skok ukoliko je rezultat prethodnog poređenja veće (neoznačeni brojevi)
- *jb op* – Skok ukoliko je rezultat prethodnog poređenja manje (neoznačeni brojevi)
- *jae op* – Skok ukoliko je rezultat prethodnog poređenja veće ili jednako (neoznačeni brojevi)
- *jbe op* – Skok ukoliko je rezultat prethodnog poređenja manje ili jednako (neoznačeni brojevi)
- *jg op* – Skok ukoliko je rezultat prethodnog poređenja veće (označeni brojevi)
- *jl op* – Skok ukoliko je rezultat prethodnog poređenja manje (označeni brojevi)
- *jge op* – Skok ukoliko je rezultat prethodnog poređenja veće ili jednako (označeni brojevi)
- *jle op* – Skok ukoliko je rezultat prethodnog poređenja manje ili jednako (označeni brojevi)
- Postoje i negacije navedenih instrukcija: *jna, jnb, jnae, jnbe, jng, jnl, jnge, jnle*

## Širina operanada u instrukciji

Ukoliko kao jedan od operanada navedemo registar, njegova širina implicitno određuje širinu drugog operanda. Međutim, ukoliko nijedan operand nije registarski, neophodno je eksplicitno naglasiti veličinu bar jednog operanada:

- *byte ptr* – Tretira operand kao jednobajtni podatak.
- *word ptr* – Tretira operand kao dvobajtni podatak.
- *dword ptr* – Tretira operand kao četvorobajtni podatak.
- *qword ptr* – Tretira operand kao osmobajtni podatak.

## Konvencije za pozivanje i pisanje funkcija

- Poziv se vrši instrukcijom *call op*
- Prenos parametara: Celobrojni parametri prenose se redom (s leva na desno) u registrima: *rdi, rsi, rdx, rcx, r8, r9*. Ukoliko funkciji prenosimo više od 6 argumenata, tada se preostali smeštaju na stek, redom s desna na levo.
- Povratna vrednost nalazi se u *rax* registru.
- Registri koji pripadaju pozvanoj funkciji:
  - *rax, rdi, rsi, rdx, rcx, r8 – r11*
- Registri koji pripadaju pozivajućoj funkciji:
  - *rbx, rbp, rsp, r12 – r15*
- Ukoliko u našem programu pozivamo neku funkciju, neophodno je da u trenutku poziva adresa vrha steka (sadržana u registru *rsp*) bude deljiva sa 16.
- Pisanje prologa: instrukcija *enter n, 0* gde je *n* broj bajtova koji odvajamo za lokalne promenljive
- Pisanje epiloga: instrukcija *leave*, koja vraća stek u stanje u trenutku ulaska u funkciju.
- Povratak iz funkcije: instrukcija *ret*

## Prevođenje

Za prevođenje asemblerskog koda dovoljno je koristiti *gcc*. Na primer, ukoliko prevodimo dva fajla, *1.c* i *1.s*, tada bi naredba za prevođenje bila *gcc 1.c 1.s*.