

## Uputstvo za korišćenje GDB-a

**GNU debager** ili **GDB** je standardan debager za GNU operativni sistem. To je prenosiv (*portable*) debager koji radi na mnogim uniksolikim sistemima i koristi se za mnoge programske jezike. GDB se isporučuje kao sastavni deo većine GNU/Linux distribucija. Ukoliko GDB ne postoji na sistemu, može se instalirati koristeći sledeće dve naredbe:

```
sudo apt-get update
```

```
sudo apt-get install gdb
```

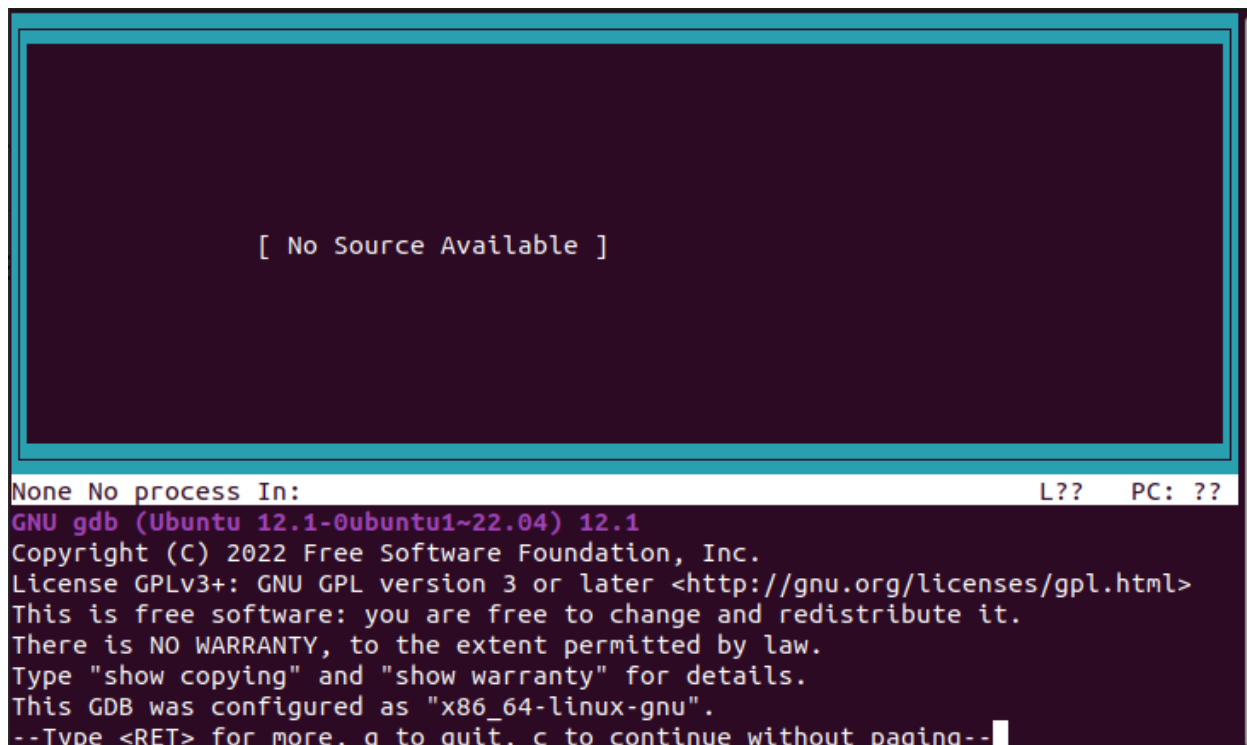
Pre samog korišćenja GDB-a obavezno je kompajliranje programskog koda sa parametrom **-g** iz foldera gde se nalazi fajl sa kodom. (Savet: otvoriti terminal preko celog ekrana zbog veće preglednosti.) U slučaju asemblerskog koda naredba za kompajliranje je:

```
gcc -m32 -g naziv_fajla.S
```

Kompajliranjem se kreira izvršni fajl pod nazivnom a.out (u slučaju da prilikom kompajliranja nije naveden željeni naziv izvršnog fajla). Da bi se započelo korišćenje GDB-a sa njegovim korisničkim interfejsom potrebno je pokrenuti sledeću naredbu:

```
gdb -tui a.out
```

Ako se pri pokretanju GDB-a ispiše poruka *[No Source Available]* (slika 1) potrebno je kliknuti taster **ENTER**.

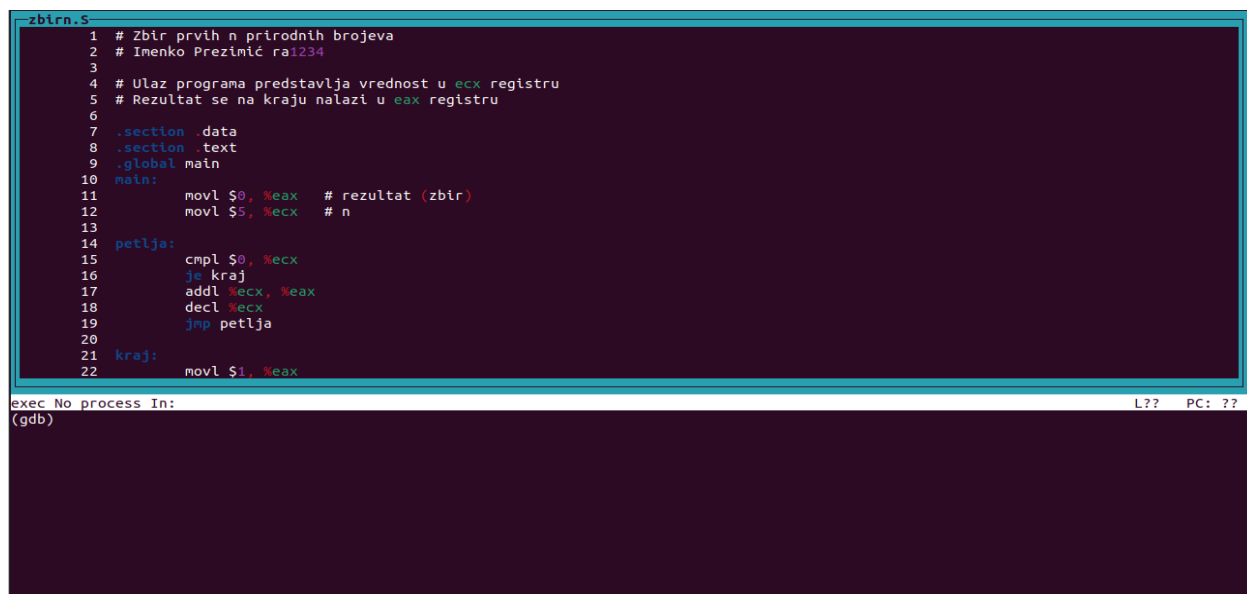


```
[ No Source Available ]

None No process In: L?? PC: ??
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04) 12.1
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
--Type <RET> for more, q to quit, c to continue without paging--
```

Slika 1

Kada je GDB uspešno pokrenut, u gornjem delu terminala trebalo bi da bude vidljiv kod programa, dok bi u donjem delu (prostor predviđen za komande) trebalo da u poslednjoj liniji piše **(gdb)** što znači da je GDB spreman da prima naredbe (slika 2).



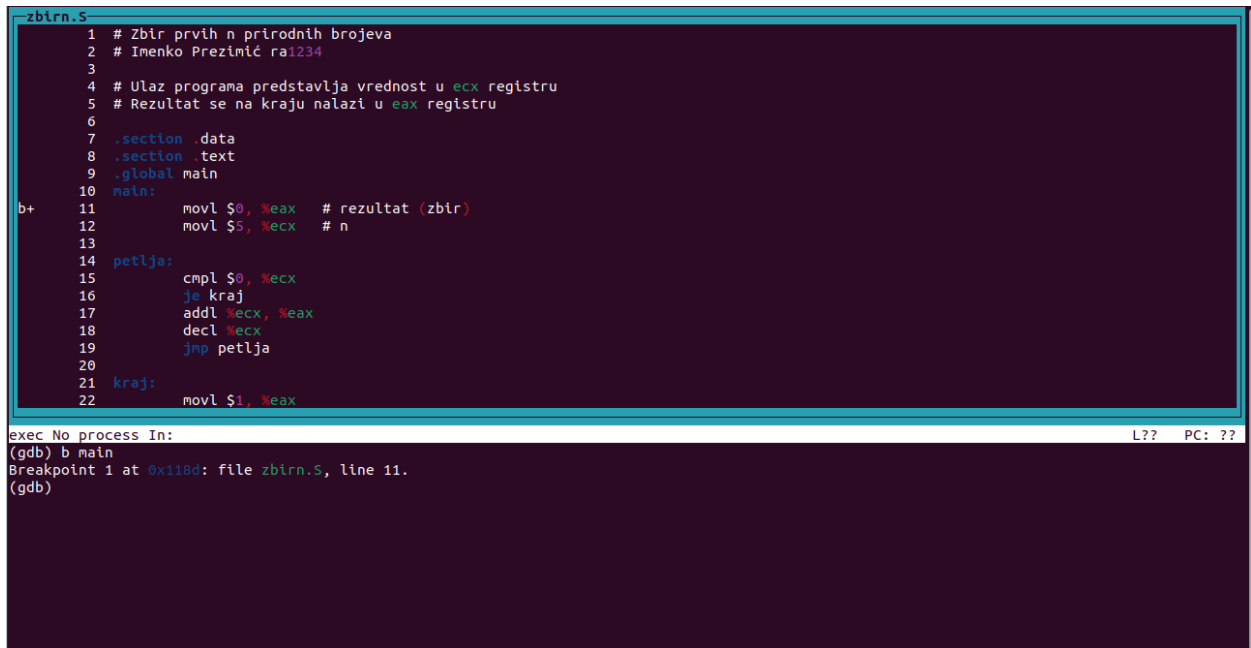
```
zbrn.s
1 # Zbir prvih n prirodnih brojeva
2 # Imenko Prezimić ra1234
3
4 # Ulaz programa predstavlja vrednost u ecx registru
5 # Rezultat se na kraju nalazi u eax registru
6
7 .section data
8 .section text
9 .global main
10 main:
11     movl $0, %eax # rezultat (zbir)
12     movl $5, %ecx # n
13
14 petlja:
15     cmpl $0, %ecx
16     je kraj
17     addl %ecx, %eax
18     decl %ecx
19     jmp petlja
20
21 kraj:
22     movl $1, %eax

exec No process In: L?? PC: ??
(gdb)
```

Slika 2

Da bi se počelo sa debugovanjem koda potrebno je postaviti *breakpoint* na početak *main*-a (slika 3), što je moguće naredbom:

**break main ili b main**



```
zbrn.S
1 # Zbir prvih n prirodnih brojeva
2 # Imenko Prezimić ra1234
3
4 # Ulaz programa predstavlja vrednost u ecx registru
5 # Rezultat se na kraju nalazi u eax registru
6
7 .section .data
8 .section .text
9 .global main
10 main:
b+ 11     movl $0, %eax # rezultat (zbir)
12     movl $5, %ecx # n
13
14 petlja:
15     cmpl $0, %ecx
16     je kraj
17     addl %ecx, %eax
18     decl %ecx
19     jmp petlja
20
21 kraj:
22     movl $1, %eax

exec No process in: L?? PC: ??
(gdb) b main
Breakpoint 1 at 0x118d: file zbrn.S, line 11.
(gdb)
```

Slika 3

Naredba za pokretanje izvršavanja programa:

**run ili r**

Kada se ova naredba izvrši, u prozoru u kom je prikazan kod programa linija koja treba da bude izvršena sledeća će biti osenčena (slika 4).

```
zbrn.S
7  .section data
8  .section text
9  .global main
10 main:
B+> 11  movl $0, %eax # rezultat (zbr)
12  movl $5, %ecx # n
13
14  petlja:
15      cpl $0, %ecx
16      je kraj
17      addl %ecx, %eax
18      decl %ecx
19      jmp petlja
20
21  kraj:
22      movl $1, %eax
23      movl $0, %ebx
24      int $0x80
25
26
27
28

multi-thre Thread 0xf77bf500 ( In: main L11 PC: 0x5655618d
(gdb) b main
Breakpoint 1 at 0xi18d: file zbrn.S, line 11.
(gdb) r
Starting program: /home/ognjen/Desktop/Arhitektura/Vezbe1/Resenja/a.out
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, main () at zbrn.S:11
(gdb) █
```

Slika 4

Otvaranje prozora sa vrednostima registara:

### layout reg

Nakon ove naredbe u gornjem delu terminala se otvara novi prozor kroz koji je moguće pratiti vrednosti registrara (slika 6). U većini slučajeva kada se ovaj prozor otvori u njemu bude ispisana poruka [Registers Values Unavailable] (slika 5), sve što je potrebno uraditi u tom slučaju je pomeriti se na sledeću liniju koda što se obavlja naredbom **step**.

```

[ Register Values Unavailable ]

zbirn.S
B+> 11      movl $0, %eax  # rezultat (zbir)
    12      movl $5, %ecx  # n
    13
    14      petlja:
    15          cmpl $0, %ecx
    16          je kraj
    17          addl %ecx, %eax
    18          decl %ecx
    19          jmp petlja
    20
    21      kraj:

multi-thre Thread 0xf7fbf500 ( In: main) L11 PC: 0x5655618d
Type "apropos word" to search for commands related to "word"...
Reading symbols from a.out...
(gdb) b main
Breakpoint 1 at 0x118d: file zbirn.S, line 11.
(gdb) r
Starting program: /home/ognjen/Desktop/Arhitektura/Vezbe1/Resenja/a.out
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, main () at zbirn.S:11
(gdb) layout reg
(gdb)

```

Slika 5

```

--Register group: general--
eax      0x0          0
ecx      0x5f67f55b  1600648539
edx      0xffffcb0  -13360
ebx      0xf7e26000  -136159232
esp      0xffffcbac  0xffffcbac
ebp      0xf7ffd020  0xf7ffd020 <_rtld_global>
esi      0xffffcc64  -13212
edi      0xf7ffc80  -134231168
eip      0x56556192  0x56556192 <main+5>

B+> 11      movl $0, %eax  # rezultat (zbir)
    > 12      movl $5, %ecx  # n
    13
    14      petlja:
    15          cmpl $0, %ecx
    16          je kraj
    17          addl %ecx, %eax
    18          decl %ecx
    19          jmp petlja
    20
    21      kraj:

multi-thre Thread 0xf7fbf500 ( In: main) L12 PC: 0x56556192
Reading symbols from a.out...
(gdb) b main
Breakpoint 1 at 0x118d: file zbirn.S, line 11.
(gdb) r
Starting program: /home/ognjen/Desktop/Arhitektura/Vezbe1/Resenja/a.out
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, main () at zbirn.S:11
(gdb) layout reg
(gdb) s
(gdb)

```

Slika 6

Ako je u nekom trenutku potrebno posebno ispisati vrednost određenog registra to je moguće naredbom:

**print \$eax ili p \$eax**

- podrazumevano je da se vrednost smeštena u registru prikazuje u decimalnom i heksadecimalnom obliku, ako je potrebno prikazati vrednost u binarnom obliku koristi se sledeća naredba:

**print /t \$eax ili p /t \$eax**

- takođe, za ispis vrednosti registrara manjeg kapaciteta koristi se ista naredba:

**print \$al ili p \$al**

Naredbe za kretanje kroz kod:

- izvršavanje sledeće naredbe bez ulaska u funkciju: **next** ili **n**
- izvršavanje sledeće naredbe sa ulaskom u funkciju: **step** ili **s**
- postavljanje *breakpoint*-a na početak određenje funkcije (labele):

**break <naziv\_labela> ili b <naziv\_labela>**

- postavljanje *breakpoint*-a na određenu liniju koda:

**break <broj\_linije> ili b <broj\_linije>**

- nastavak izvršavanja programa do sledećeg *breakpoint*-a: **continue** ili **c**
- brisanje svih postavljenih *breakpoint*-a: **clear**
- kada poslednja izvršena naredba treba ponovo da se izvrši dovoljno je samo pritisnuti taster **ENTER**, bez potrebe za ponovnim kucanjem naredbe

Rad sa promenljivama:

- prikaz vrednosti promenljive: **x &<naziv\_promenljive>**
  - ako je potrebno konvertovati vrednost promenljive u određeni brojni sistem:

**x /<brojni\_sistem> &<naziv\_promenljive>**

- **/t** - za binarni brojni sistem
- **/d** - za decimalni brojni sistem

- **/x** - za heksadecimalni brojni sistem
- **/o** - za oktalni brojni sistem
- **/c** - za prikaz karaktera koji odgovara *ascii* vrednosti smeštene u promenljivoj
- primer: Prikaz vrednosti promenljive *a* u decimalnom obliku
  - `x/d &a`

- prikaz niza:

**`x/<broj_elementa_niza><velicina_elementa><brojni_sistem> &<naziv_niza>`**

- moguće veličine elemenata niza:
  - bytes (1 bajt): **/b**
  - halfwords (2 bajta): **/h**
  - words (4 bajta): **/w**
  - giants (8 bajta): **/g**
- primer: Prikaz vrednosti elemenata niza pod nazivom *niz* koji sadrži 10 elemenata gde je svaki veličine 4 bajta (words - w) u heksadecimalnom obliku (x)
  - `x/10wx &niz`
- kada se radi sa nizovima često je potrebno prikazati vrednost trenutnog elementa niza
  - indirektno adresiranje (ako je adresa niza smeštena u registru *eax*)
 

**`x/<velicina_elementa_niza><brojni_sistem> $eax`**

    - primer: Prikaz vrednosti trenutnog elementa niza (ako je adresa niza smeštena u registru *eax*), gde je svaki element veličine 1 bajt (bytes) u binarnom obliku (t)
      - `x/bt $eax`
  - indeksno adresiranje (gde je indeks smešten u registru *eax*)
 

**`x ((char *)&niz+$eax*4)`**

- prikaz stringa: **x/s &<naziv\_stringa>**
  - prikaz stringa kao niza karaktera:

**x /<broj\_karaktera>c &<naziv\_stringa>**

Kod rada sa potprogramima često je potrebno prikazati vrednost prosleđenih argumenata potprograma pomoću registra ebp

- pošto se vrednosti prvog parametra potprogram pristupa kao 8(%ebp), njegovu vrednost je moguće prikazati pomoću naredbe:

**x /<brojni\_sistem> \$ebp+8**

- za prikaz ostalih parametara potrebno je pratiti šablon:

**x \$ebp+broj\_bajtova\_na\_steku\_u\_odnosu\_na\_ebp**

- na isti način se pristupa i lokalnim promenljivama, pa tako se vrednost prve lokalne promenljive dobija kao: **x \$ebp-4**

Prilikom rada sa gdb fokus je postavljen na deo sa kodom i nije moguće pristupiti prethodnim naredbama pritiskom strelice na gore. Da bi se fokus prebacio na komandnu liniju pokrenuti komandu:

**focus cmd**

Da bi vratili fokus nazad na kod pokrenuti komandu:

**focus src**

Izlaz iz GDB-a: **quit** ili **q**