[Public]

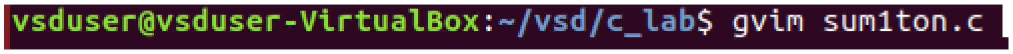
VSDSquadron_Labs

*Task 1. C-lab & RISC-V lab: counting sum of numbers from 1 to n *

Note: we already installed RISC-V toolchain.

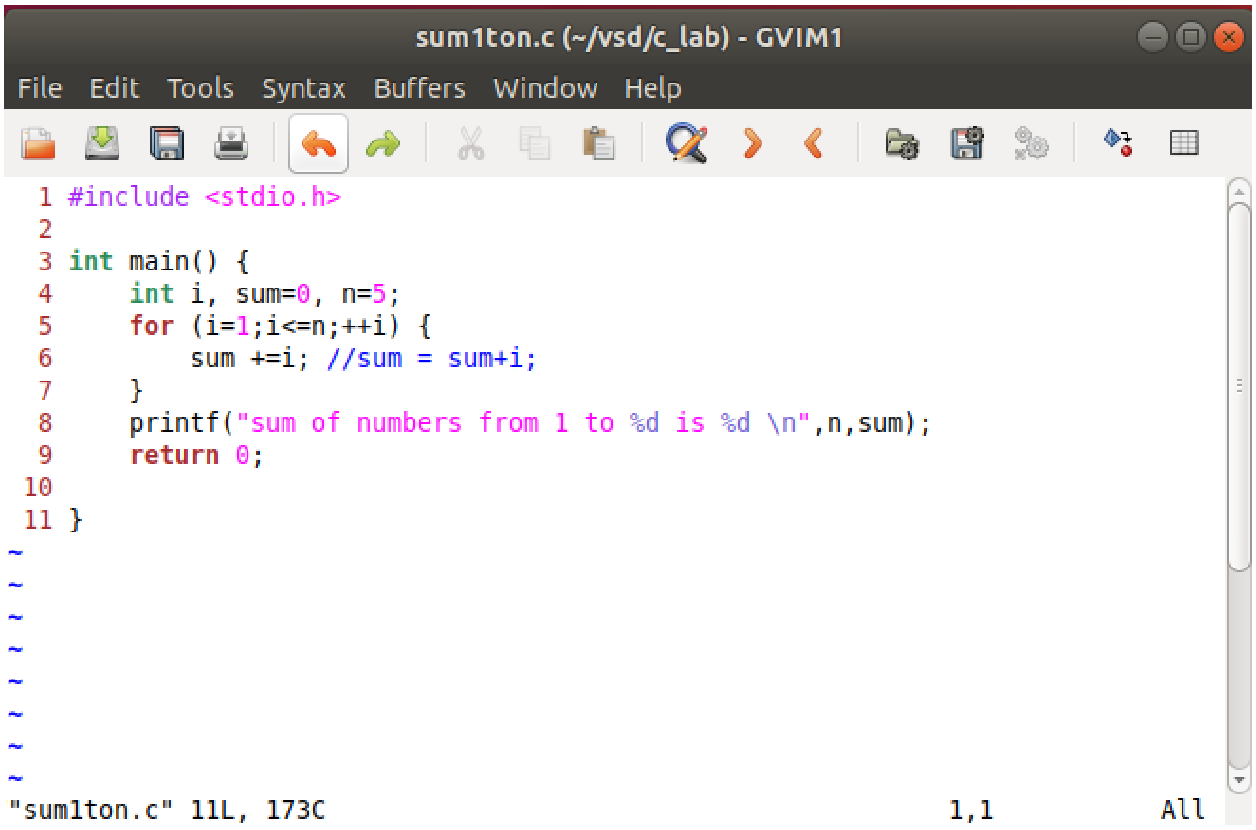
1. write c-code file with gvim editor

   cmd: **gvim sum1ton.c**



```
vsduser@vsduser-VirtualBox:~/vsd/c_lab$ gvim sum1ton.c
```

It will open gvim editor and you can write your c source code. Here I'm writing **counting sum of numbers from 1 to n**.

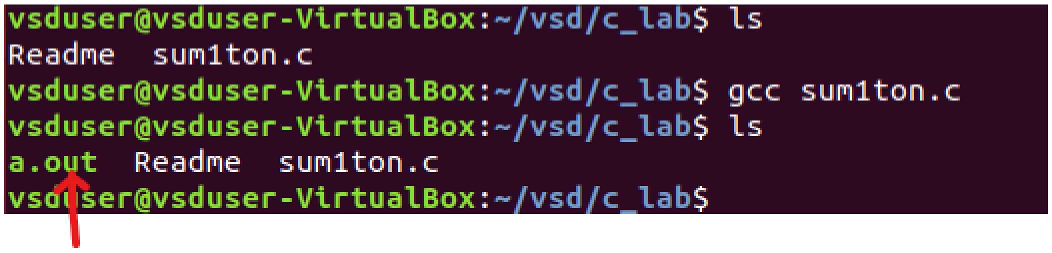

sum1ton.c (~/vsd/c_lab) - GVIM1
File Edit Tools Syntax Buffers Window Help

```
#include <stdio.h>

int main() {
    int i, sum=0, n=5;
    for (i=1;i<=n;++i) {
        sum +=i; //sum = sum+i;
    }
    printf("sum of numbers from 1 to %d is %d \n",n,sum);
    return 0;

}
~
~
~
~
~
~
~
~
"sum1ton.c" 11L, 173C                                   1,1           All
```

I. c-file compile with GCC and run

2. compile c file (below cmd does 4 steps: prepocess, compile, assemble, link to create executable file)

cmd: **gcc file.c**

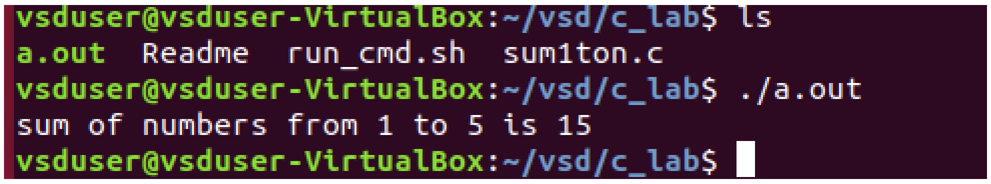


```
vsduser@vsduser-VirtualBox:~/vsd/c_lab$ ls
Readme  sum1ton.c
vsduser@vsduser-VirtualBox:~/vsd/c_lab$ gcc sum1ton.c
vsduser@vsduser-VirtualBox:~/vsd/c_lab$ ls
a.out  Readme  sum1ton.c
vsduser@vsduser-VirtualBox:~/vsd/c_lab$
```

It will create a.out executable file

3. run executable file

cmd: **./a.out**



```
vsduser@vsduser-VirtualBox:~/vsd/c_lab$ ls
a.out  Readme  run_cmd.sh  sum1ton.c
vsduser@vsduser-VirtualBox:~/vsd/c_lab$ ./a.out
sum of numbers from 1 to 5 is 15
vsduser@vsduser-VirtualBox:~/vsd/c_lab$ 
```

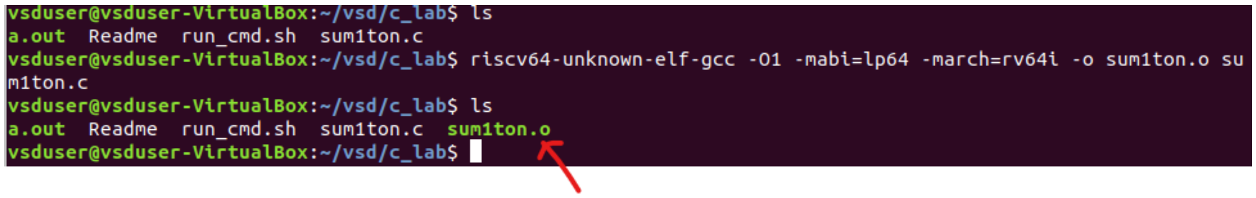
check output in the terminal: **sum of numbers from 1 to 5 is 15**

II. compile same c file with RISC-V gcc compiler & see genearted assemmbly code

4. cmd: **riscv64-unknown-elf-gcc -O1 -mabi=lp64 -march=rv64i -o sum1ton.o sum1ton.c**

above cmd will create **sum1ton.o** file

```
other options

 -O1/-Ofast

 -time                        Time the execution of each subprocess.

 -o <file>                    Place the output into <file>.

 -march=rv32i -mabi=ilp32 to the linker.
```



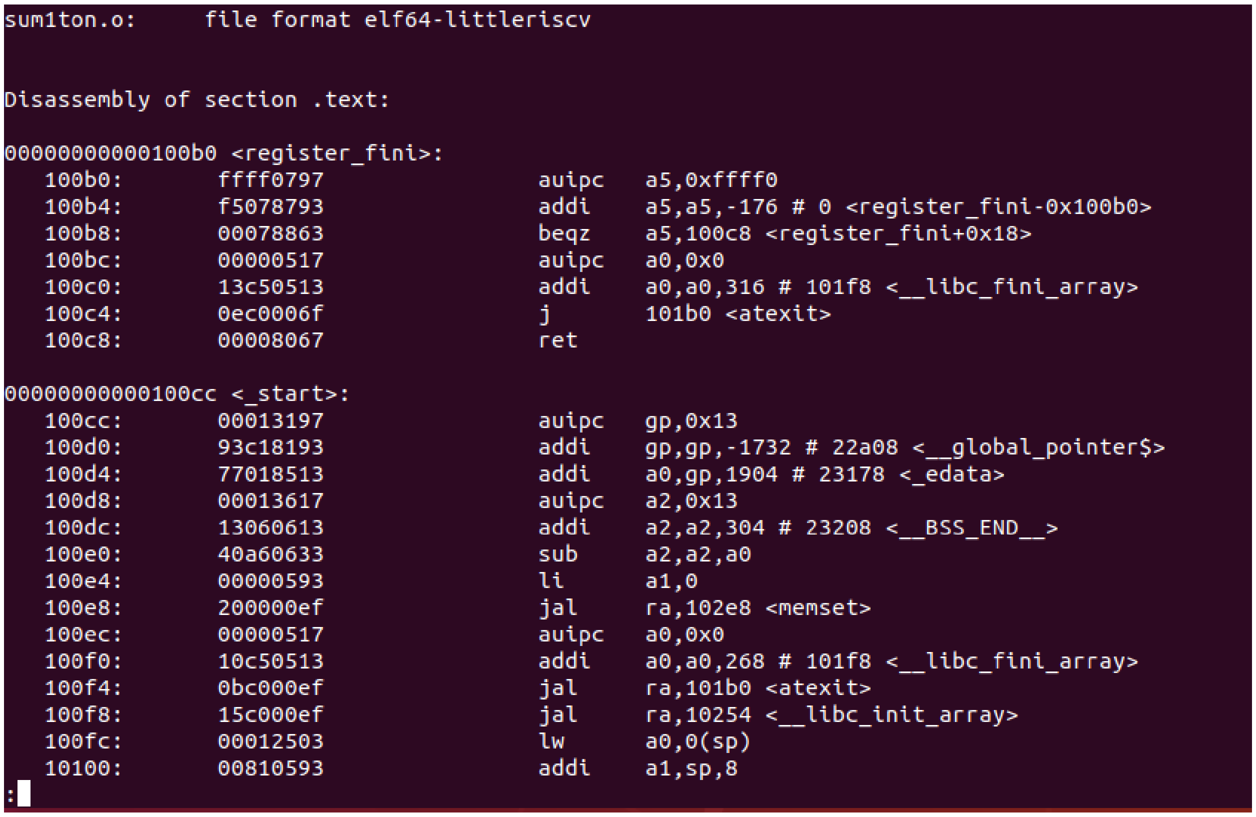
```
vsduser@vsduser-VirtualBox:~/vsd/c_lab$ ls
a.out  Readme  run_cmd.sh  sum1ton.c
vsduser@vsduser-VirtualBox:~/vsd/c_lab$ riscv64-unknown-elf-gcc -O1 -mabi=lp64 -march=rv64i -o sum1ton.o su
m1ton.c
vsduser@vsduser-VirtualBox:~/vsd/c_lab$ ls
a.out  Readme  run_cmd.sh  sum1ton.c  sum1ton.o
vsduser@vsduser-VirtualBox:~/vsd/c_lab$ 
```

5. check how assembly level instructions for the c-code getting generated with below command

   cmd: **riscv64-unknown-elf-objdump -d sum1ton.o**

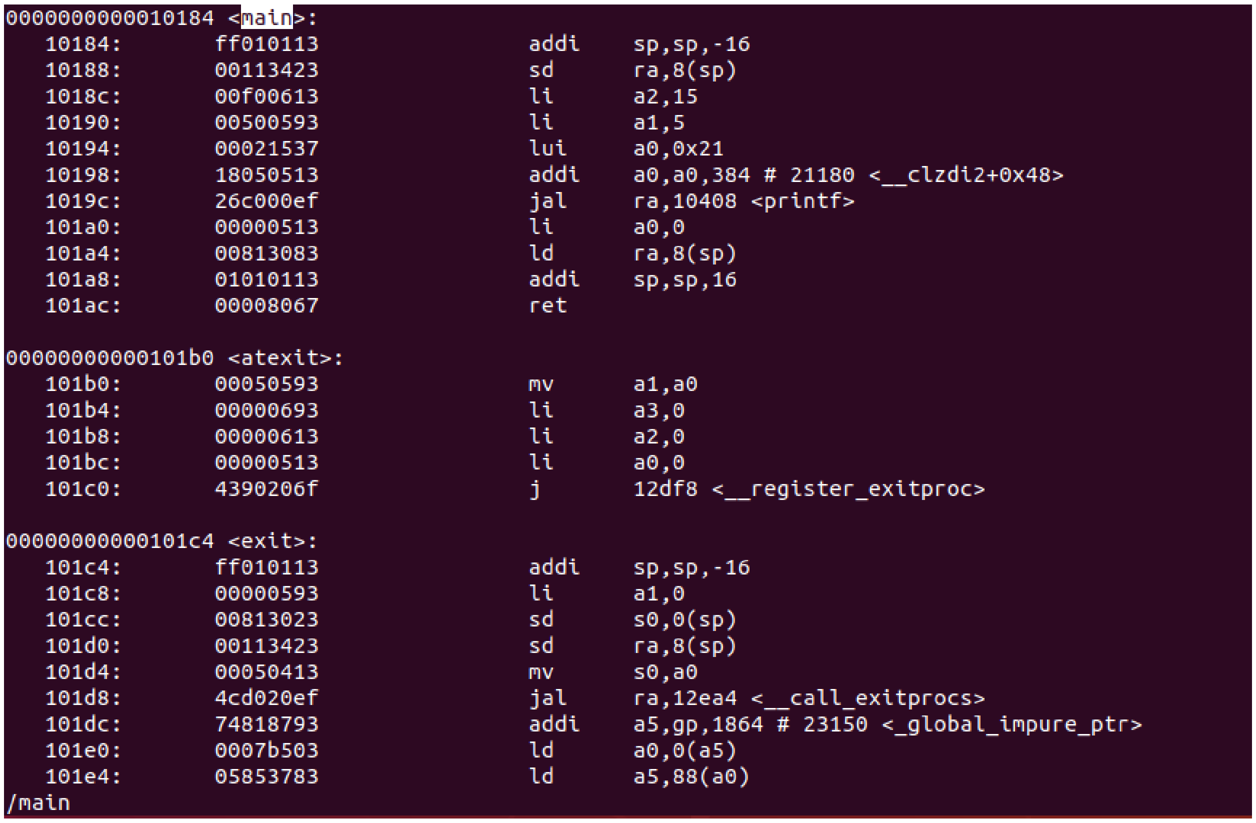
   ```
   here
     objdump is object dump
     d is disassemble
     use |less cmd at end & search for /main
   ```

   cmd: **riscv64-unknown-elf-objdump -d sum1ton.o | less**



```
sum1ton.o:     file format elf64-littleriscv


Disassembly of section .text:

00000000000100b0 <register_fini>:
   100b0:       ffff0797                auipc   a5,0xffff0
   100b4:       f5078793                addi    a5,a5,-176 # 0 <register_fini-0x100b0>
   100b8:       00078863                beqz    a5,100c8 <register_fini+0x18>
   100bc:       00000517                auipc   a0,0x0
   100c0:       13c50513                addi    a0,a0,316 # 101f8 <__libc_fini_array>
   100c4:       0ec0006f                j       101b0 <atexit>
   100c8:       00008067                ret

00000000000100cc <_start>:
   100cc:       00013197                auipc   gp,0x13
   100d0:       93c18193                addi    gp,gp,-1732 # 22a08 <__global_pointer$>
   100d4:       77018513                addi    a0,gp,1904 # 23178 <_edata>
   100d8:       00013617                auipc   a2,0x13
   100dc:       13060613                addi    a2,a2,304 # 23208 <__BSS_END__>
   100e0:       40a60633                sub     a2,a2,a0
   100e4:       00000593                li      a1,0
   100e8:       200000ef                jal     ra,102e8 <memset>
   100ec:       00000517                auipc   a0,0x0
   100f0:       10c50513                addi    a0,a0,268 # 101f8 <__libc_fini_array>
   100f4:       0bc000ef                jal     ra,101b0 <atexit>
   100f8:       15c000ef                jal     ra,10254 <__libc_init_array>
   100fc:       00012503                lw      a0,0(sp)
   10100:       00810593                addi    a1,sp,8
:
```

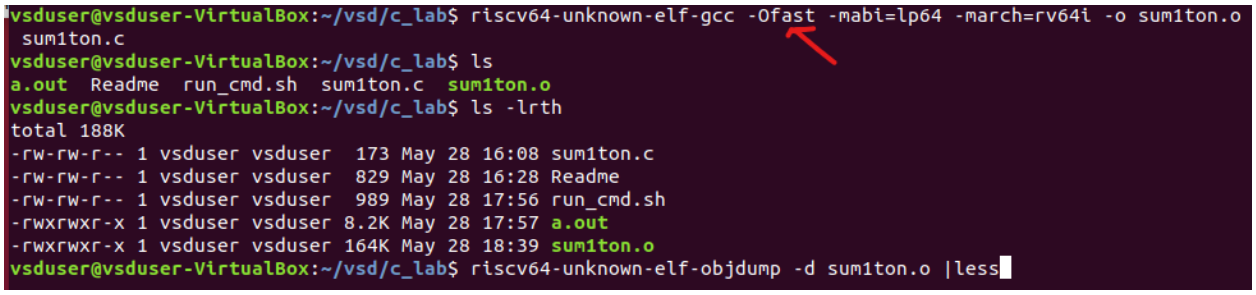
```
serach for main function in this disassembly code
```



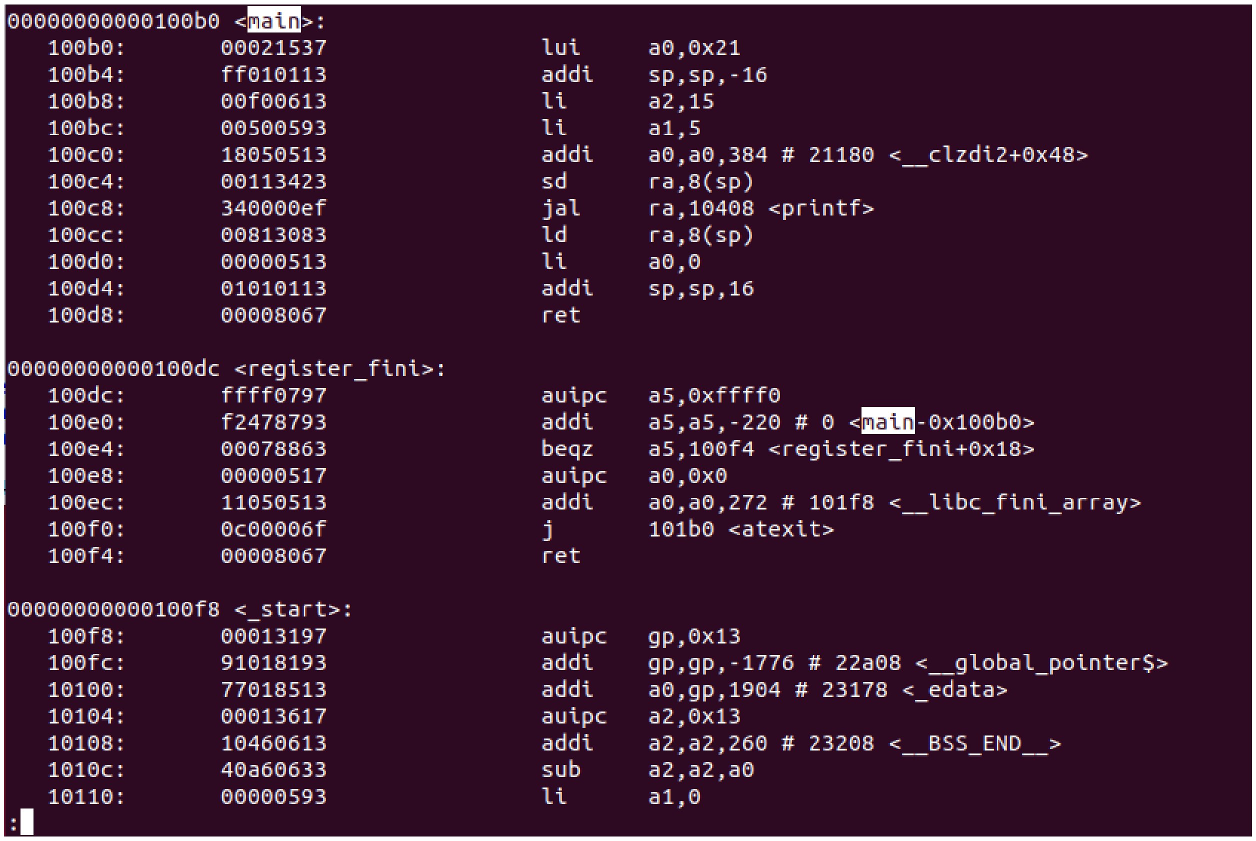
```
0000000000010184 <main>:
   10184:       ff010113                addi    sp,sp,-16
   10188:       00113423                sd      ra,8(sp)
   1018c:       00f00613                li      a2,15
   10190:       00500593                li      a1,5
   10194:       00021537                lui     a0,0x21
   10198:       18050513                addi    a0,a0,384 # 21180 <__clzdi2+0x48>
   1019c:       26c000ef                jal     ra,10408 <printf>
   101a0:       00000513                li      a0,0
   101a4:       00813083                ld      ra,8(sp)
   101a8:       01010113                addi    sp,sp,16
   101ac:       00008067                ret

00000000000101b0 <atexit>:
   101b0:       00050593                mv      a1,a0
   101b4:       00000693                li      a3,0
   101b8:       00000613                li      a2,0
   101bc:       00000513                li      a0,0
   101c0:       4390206f                j       12df8 <__register_exitproc>

00000000000101c4 <exit>:
   101c4:       ff010113                addi    sp,sp,-16
   101c8:       00000593                li      a1,0
   101cc:       00813023                sd      s0,0(sp)
   101d0:       00113423                sd      ra,8(sp)
   101d4:       00050413                mv      s0,a0
   101d8:       4cd020ef                jal     ra,12ea4 <__call_exitprocs>
   101dc:       74818793                addi    a5,gp,1864 # 23150 <_global_impure_ptr>
   101e0:       0007b503                ld      a0,0(a5)
   101e4:       05853783                ld      a5,88(a0)
/main
```

Count no of instructions in the code. main() function started at address 0x1_0184, next subtask executed at address 0x1_01b0. difference between 0x1_01b0 - 0x1_0184 = 2c/4 we are diving here becoz of byte aligned address, so address jumps by 4. = 'd11. So total 11 instructions present in main to next subtask.

Now lets try other option like -Ofast and check the assembly code



```
vsduser@vsduser-VirtualBox:~/vsd/c_lab$ riscv64-unknown-elf-gcc -Ofast -mabi=lp64 -march=rv64i -o sum1ton.o
 sum1ton.c
vsduser@vsduser-VirtualBox:~/vsd/c_lab$ ls
a.out  Readme  run_cmd.sh  sum1ton.c  sum1ton.o
vsduser@vsduser-VirtualBox:~/vsd/c_lab$ ls -lrth
total 188K
-rw-rw-r-- 1 vsduser vsduser  173 May 28 16:08 sum1ton.c
-rw-rw-r-- 1 vsduser vsduser  829 May 28 16:28 Readme
-rw-rw-r-- 1 vsduser vsduser  989 May 28 17:56 run_cmd.sh
-rwxrwxr-x 1 vsduser vsduser 8.2K May 28 17:57 a.out
-rwxrwxr-x 1 vsduser vsduser 164K May 28 18:39 sum1ton.o
vsduser@vsduser-VirtualBox:~/vsd/c_lab$ riscv64-unknown-elf-objdump -d sum1ton.o |less
```



```
00000000000100b0 <main>:
   100b0:       00021537                lui     a0,0x21
   100b4:       ff010113                addi    sp,sp,-16
   100b8:       00f00613                li      a2,15
   100bc:       00500593                li      a1,5
   100c0:       18050513                addi    a0,a0,384 # 21180 <__clzdi2+0x48>
   100c4:       00113423                sd      ra,8(sp)
   100c8:       340000ef                jal     ra,10408 <printf>
   100cc:       00813083                ld      ra,8(sp)
   100d0:       00000513                li      a0,0
   100d4:       01010113                addi    sp,sp,16
   100d8:       00008067                ret

00000000000100dc <register_fini>:
   100dc:       ffff0797                auipc   a5,0xffff0
   100e0:       f2478793                addi    a5,a5,-220 # 0 <main-0x100b0>
   100e4:       00078863                beqz    a5,100f4 <register_fini+0x18>
   100e8:       00000517                auipc   a0,0x0
   100ec:       11050513                addi    a0,a0,272 # 101f8 <__libc_fini_array>
   100f0:       0c00006f                j       101b0 <atexit>
   100f4:       00008067                ret

00000000000100f8 <_start>:
   100f8:       00013197                auipc   gp,0x13
   100fc:       91018193                addi    gp,gp,-1776 # 22a08 <__global_pointer$>
   10100:       77018513                addi    a0,gp,1904 # 23178 <_edata>
   10104:       00013617                auipc   a2,0x13
   10108:       10460613                addi    a2,a2,260 # 23208 <__BSS_END__>
   1010c:       40a60633                sub     a2,a2,a0
   10110:       00000593                li      a1,0
:
```

now count number of instructions 100dc-100b0= 2C/4=B='d11. No of instrctions didnt changed even though if we change -O1/-Ofast.