```
-bash-4.1$ gdb sandbox
GNU gdb (GDB) Red Hat Enterprise Linux (7.2-90.el6)
Copyright (C) 2010 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-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /uio/hume/student-u80/tanusanr/Programming Assignments/PA02/sandbox...done.
(gdb) break *_start+1
Breakpoint 1 at 0x8048061: file sandbox.asm, line 7.
(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
   0x08048060 <+0>:     nop
   0x08048061 <+1>:     mov    eax,0x4
   0x08048066 <+6>:     mov    ebx,0x1
   0x0804806b <+11>:    mov    al,0x6f
   0x0804806d <+13>:    and    al,0x2d
   0x0804806f <+15>:    mov    al,0x6d
   0x08048071 <+17>:    and    al,0x4a
   0x08048073 <+19>:    mov    al,0xf
   0x08048075 <+21>:    or     al,0x61
   0x08048077 <+23>:    mov    al,0x94
   0x08048079 <+25>:    xor    al,0x37
   0x0804807b <+27>:    mov    al,0x7a
   0x0804807d <+29>:    sub    al,0x67
   0x0804807f <+31>:    mov    al,0x3d
   0x08048081 <+33>:    add    al,0x34
   0x08048083 <+35>:    mov    al,0x9b
   0x08048085 <+37>:    not    al
   0x08048087 <+39>:    mov    al,0x37
   0x08048089 <+41>:    neg    al
   0x0804808b <+43>:    int    0x80
   0x0804808d <+45>:    mov    eax,0x1
   0x08048092 <+50>:    mov    ebx,0x0
---Type <return> to continue, or q <return> to quit---
   0x08048097 <+55>:    int    0x80
End of assembler dump.
(gdb) r
Starting program: /uio/hume/student-u80/tanusanr/Programming Assignments/PA02/sandbox

Breakpoint 1, _start () at sandbox.asm:7
7           mov eax,4 ; Specify sys_write call
(gdb) si
8           mov ebx,1 ; Specify File Descriptor 1: Standard Output
(gdb) si
9               mov al,01101111b
(gdb) si
10              and al,00101101b
(gdb) info registers
eax            0x6f     111
ecx            0x0      0
edx            0x0      0
```

```
End of assembler dump.
(gdb) r
Starting program: /uio/hume/student-u80/tanusanr/Programming Assignments/PA02/sandbox

Breakpoint 1, _start () at sandbox.asm:7
7           mov eax,4 ; Specify sys_write call
(gdb) si
8           mov ebx,1 ; Specify File Descriptor 1: Standard Output
(gdb) si
9               mov al,01101111b
(gdb) si
10              and al,00101101b
(gdb) info registers
eax            0x6f     111
ecx            0x0      0
edx            0x0      0
ebx            0x1      1
esp            0xffffd150       0xffffd150
ebp            0x0      0x0
esi            0x0      0
edi            0x0      0
eip            0x804806d        0x804806d <_start+13>
eflags         0x202    [ IF ]
cs             0x23     35
ss             0x2b     43
ds             0x2b     43
es             0x2b     43
fs             0x0      0
gs             0x0      0
(gdb) si
12              mov al,6Dh
(gdb) si
13              and al,4Ah
(gdb) info registers
eax            0x6d     109
ecx            0x0      0
edx            0x0      0
ebx            0x1      1
esp            0xffffd150       0xffffd150
ebp            0x0      0x0
esi            0x0      0
edi            0x0      0
eip            0x8048071        0x8048071 <_start+17>
eflags         0x206    [ PF IF ]
cs             0x23     35
ss             0x2b     43
ds             0x2b     43
es             0x2b     43
fs             0x0      0
gs             0x0      0
(gdb) c
Continuing.

Program exited normally.
(gdb) quit
-bash-4.1$
```

```
SECTION .data
SECTION .bss
SECTION .text
      global _start
_start:
                              nop ; This no-op keeps gdb happy...
   mov eax,4 ; Specify sys_write call
   mov ebx,1 ; Specify File Descriptor 1: Standard Output

                              mov al,01101111b ;copying 01101111b to the al registers
                              and al,00101101b ;sets al to be 01101111b

                              mov al,6Dh ; copying 6Dh to the al register
                              and al,4Ah ; ands 4Ah to 6Dh which should give B7h

                              mov al,00001111b ; copying 00001111b to the al register
                              or al,61h ; or-ing 00001111b with 61h, which should give
70h/01110000h

                              mov al,94h ; copying 94h to the al register
                              xor al,37h xor-ing 94h with 37h, which should give A3h

                              mov al,7Ah ; copying 7Ah to the al register
                              sub al,67h  ; subtracts 7Ah with 67h, which should give 13h

                              mov al,3Dh ; copying 3Dh to the al register
                              add al,34h ; adding 3Dh with 34h, which should give 71h

                              mov al,9Bh ; copying 9B to the al register
                              not al ; reverses the bits in al register

                              mov al,37h ; copying 37h to the al register
                              neg al ; makes the value in al to negative with 2's complement,
which should give 9h

   int 80H ; Make kernel call
   mov eax,1 ; Code for Exit Syscall
   mov ebx,0 ; Return a code of zero
   int 80H ; Make kernel call
```

This is the code I ran, I have comented behind what each line is supposed to do / what I think it does. The result was that it exited normally. I ran all of them at once because when I ran them individually some of them started to hang.