Debugando código criado em Assembly x86

\+ Aqui o debbuger usado será o gdb
→ Para entrar no gdp de forma mai limpa e rápida:

```
gdb -q ./ass
```

```
┌──(root㉿DESKTOP-NJHHNK6)-[/home/kali]
└─# gdb -q ./ass
Reading symbols from ./ass...
(No debugging symbols found in ./ass)
(gdb)
```

→ Podemos colocar um breakpoint no nosso entrypoint (função main - onde começa o programa): break _main
→ Eaí damos um run pra que ele rode nosso programa até a mainrun

```
(gdb) break _main
Breakpoint 1 at 0x8049000
(gdb) run
Starting program: /home/kali/ass

Breakpoint 1, 0x08049000 in _main ()
```

→ Para mostrar as informações dos registradores: info registers ou i r

```
eax            0x0                 0
ecx            0x0                 0
edx            0x0                 0
ebx            0x0                 0
esp            0xffffd440          0xffffd440
ebp            0x0                 0x0
esi            0x0                 0
edi            0x0                 0
eip            0x8049000           0x8049000 <_main>
eflags         0x202               [ IF ]
cs             0x23                35
ss             0x2b                43
ds             0x2b                43
es             0x2b                43
fs             0x0                 0
```

→ Ele mostra, em eip, qual será o próximo executado
→ O disassembly é o disas

```
(gdb) disas
Dump of assembler code for function _main:
=> 0x08049000 <+0>:     mov    $0x4,%eax
   0x08049005 <+5>:     mov    $0x1,%ebx
   0x0804900a <+10>:    mov    $0x804a000,%ecx
   0x0804900f <+15>:    mov    $0xf,%edx
   0x08049014 <+20>:    int    $0x80
```

→ Só que ele veio na syntax da AT&T, mas queremos na da intel:

```
set disassembly-flavor intel
```

```
(gdb) set disassembly-flavor intel
(gdb) disas
Dump of assembler code for function _main:
=> 0x08049000 <+0>:     mov    eax,0x4
   0x08049005 <+5>:     mov    ebx,0x1
   0x0804900a <+10>:    mov    ecx,0x804a000
   0x0804900f <+15>:    mov    edx,0xf
   0x08049014 <+20>:    int    0x80
   0x08049016 <+22>:    mov    eax,0x1
   0x0804901b <+27>:    mov    ebx,0x0
   0x08049020 <+32>:    int    0x80
```

→ Para executar passo a passo: stepi

→ Para saber o que tem em um endereço:

```
x/s 0x8049000
```

x → examine, s → tipo: string