CSO-1

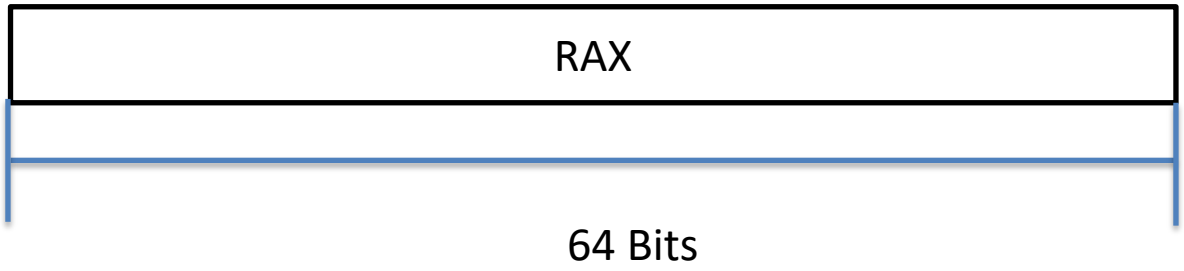
X86 Assembly

Daniel G. Graham PhD

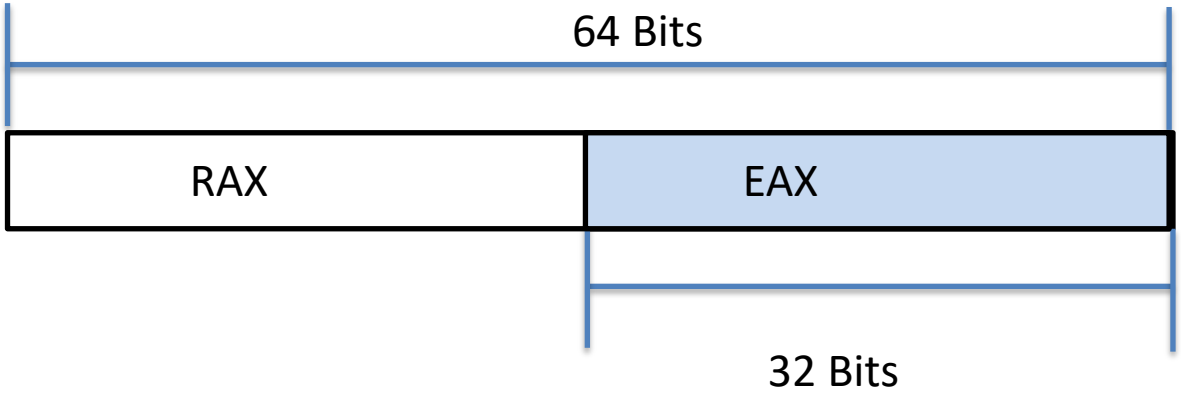
1. X86 Assembly
2. Assemble instructions using Clang
3. Push Encoding
4. Walk through of Push Pop Example in x86
5. Inspect memory

NOW LET'S START TALK ABOUT WRITING ASSEMBLY FOR X86 PROCESSORS

X86 REGISTERS

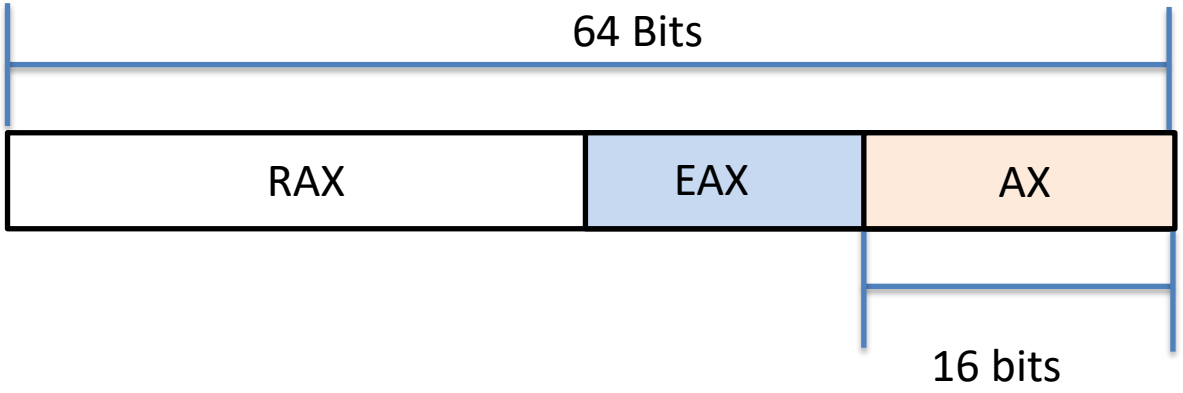


X86 REGISTERS



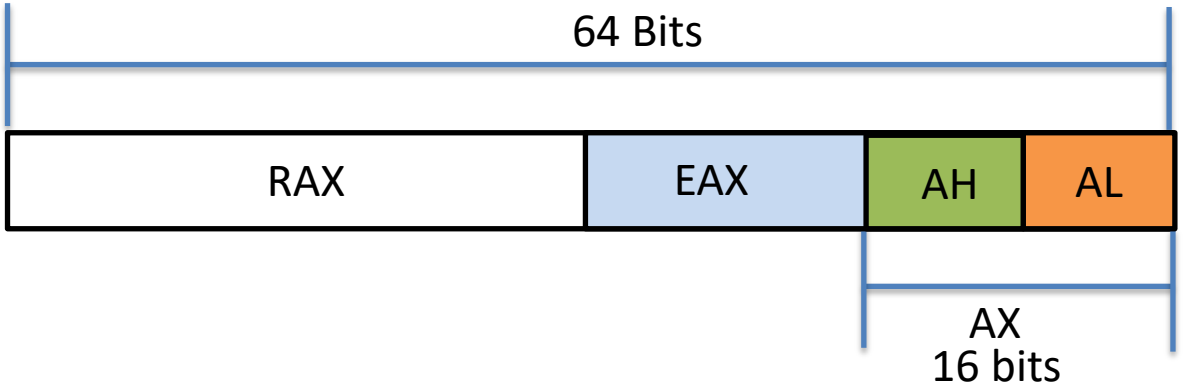
The lowest 32 bits

X86 REGISTERS

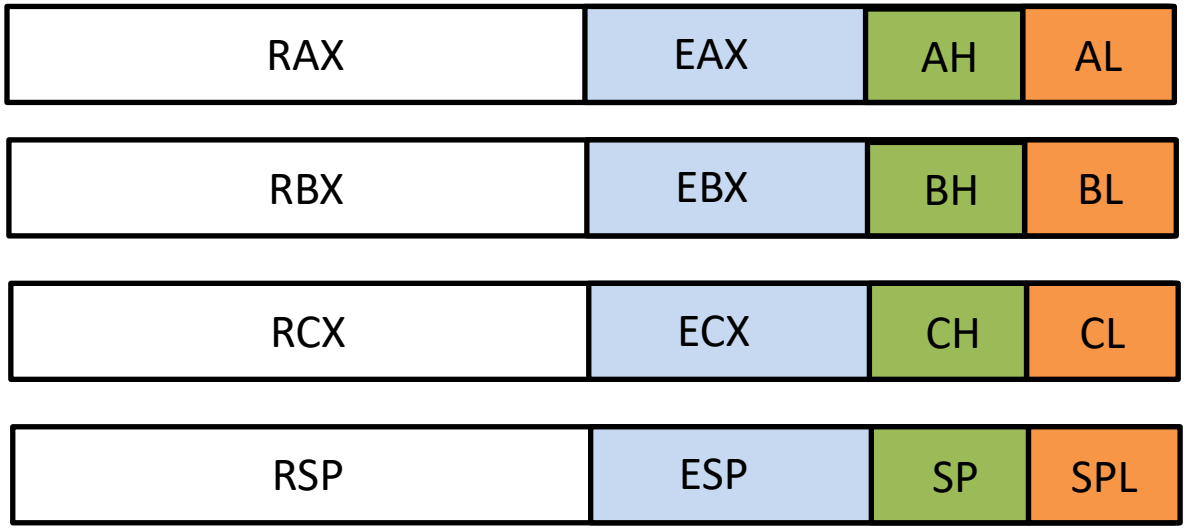


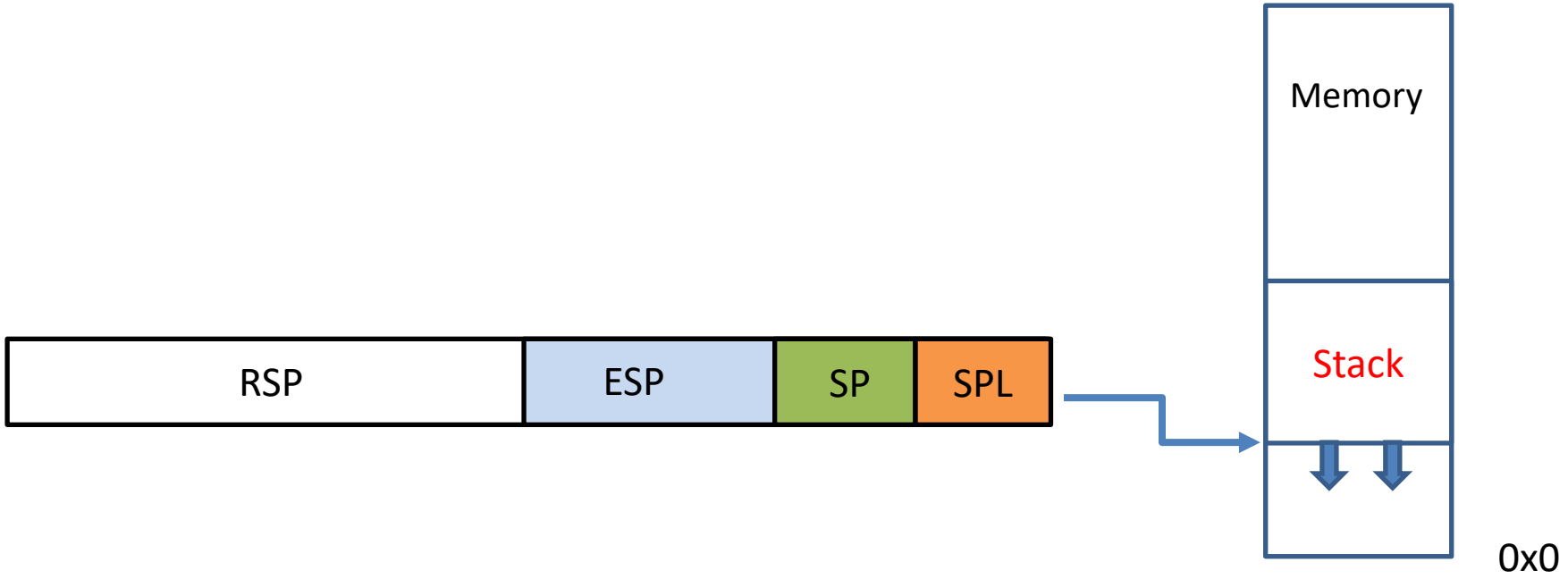
AX can future divided into two registers

X86 REGISTERS



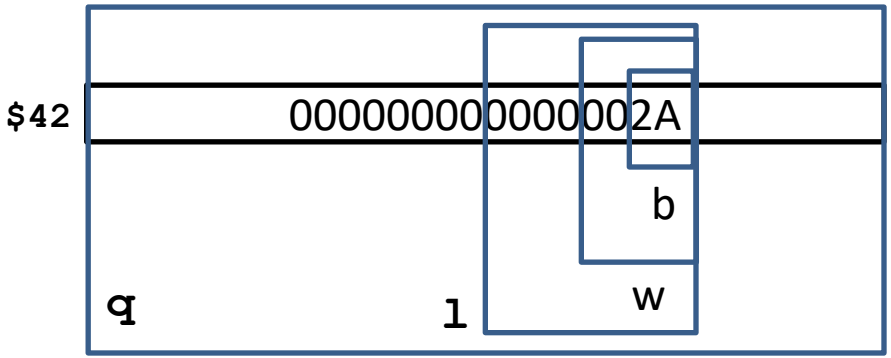
THERE ARE 16 REGISTER





AT&T SYNTAX

```
pushq $42
```



constants start with $

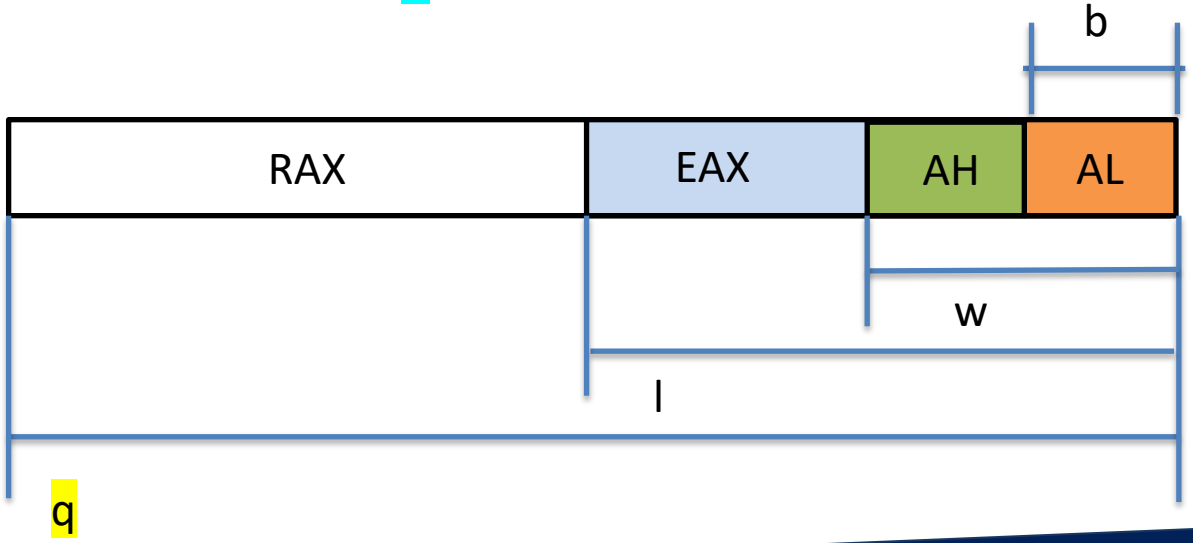
suffix	Meaning
b	“Byte”: 1 byte
w	“Word”: 2 bytes
l	“Long”: 4 bytes
q	“Quad”: 8 bytes (4 words)

X86 will truncate in constant is larger than the destination

AT&T SYNTAX

```
popq %rax
```

registers start with %



suffix	Meaning
b	“Byte”: 1 byte
w	“Word”: 2 bytes
l	“Long”: 4 bytes
q	“Quad”: 8 bytes (4 words)

ASSEMBLY IS MORE PRECISE THAN C

```
dgg6b@portal06:~/CS01/Assemble/lab$ nano stackExamplePart1.s
```

```
.globl main

main:
    # Push the value 4 onto the stack
    pushq $4

    # Push the value 5 onto the stack
    pushq $5

    # Read the first number from the stack
    popq %rax

    # Read the second number from the stack
    popq %rbx

    # Add the two numbers
    addq %rax, %rbx

    # Push the result back onto the stack
    pushq %rbx
```

[Read 20 lines]

```
dgg6b@portal06:~/CS01/Assemble/lab$ clang -c stackExamplePart1.s -o stackExamplePart1.o
dgg6b@portal06:~/CS01/Assemble/lab$ ls
debugExample.o  registerExample.s    stackExamplePart1.s
debugExample.s  stackExamplePart1.o  stackExamplePart2.s
dgg6b@portal06:~/CS01/Assemble/lab$
```

-c Only run preprocess, compile, and assemble steps

-o <file> Write output to <file>

```
dgg6b@portal02:~/CS01/Assemble/lab$ clang -c stackExamplePart1.s -o stackExamplePart1.o
dgg6b@portal02:~/CS01/Assemble/lab$ ls
debugExample.o  registerExample.s    stackExamplePart1.s
debugExample.s  stackExamplePart1.o  stackExamplePart2.s
dgg6b@portal02:~/CS01/Assemble/lab$ objdump -D stackExamplePart1.o

stackExamplePart1.o:     file format elf64-x86-64


Disassembly of section .text:

0000000000000000 <main>:
   0:   6a 04                   push   $0x4
   2:   6a 05                   push   $0x5
   4:   58                      pop    %rax
   5:   5b                      pop    %rbx
   6:   48 01 c3                add    %rax,%rbx
   9:   53                      push   %rbx
dgg6b@portal02:~/CS01/Assemble/lab$
```

Notice the hex machine instructions just like our toy ISA

Also notice how the address in memory increase based on the size of the instruction

objdump – tool that allows us to inspect the object file

-D, --disassemble-all Display assembler contents of all sections

```
dgg6b@portal02:~/CS01/Assemble/lab$ clang stackExamplePart1.s -o finalprogram
dgg6b@portal02:~/CS01/Assemble/lab$ ls
debugExample.o  finalprogram      stackExamplePart1.o  stackExamplePart2.s
debugExample.s  registerExample.s  stackExamplePart1.s
dgg6b@portal02:~/CS01/Assemble/lab$
```

```
dgg6b@portal02:~/CS01/Assemble/lab$ clang stackExamplePart1.s -o finalprogram
dgg6b@portal02:~/CS01/Assemble/lab$ ls
debugExample.o  finalprogram       stackExamplePart1.o  stackExamplePart2.s
debugExample.s  registerExample.s  stackExamplePart1.s
dgg6b@portal02:~/CS01/Assemble/lab$ lldb finalprogram
(lldb) target create "finalprogram"
Current executable set to '/u/dgg6b/CS01/Assemble/lab/finalprogram' (x86_64).
(lldb)
```

```
dgg6b@portal02:~/CS01/Assemble/lab$ clang stackExamplePart1.s -o finalprogram
dgg6b@portal02:~/CS01/Assemble/lab$ ls
debugExample.o  finalprogram       stackExamplePart1.o  stackExamplePart2.s
debugExample.s  registerExample.s  stackExamplePart1.s
dgg6b@portal02:~/CS01/Assemble/lab$ lldb finalprogram
(lldb) target create "finalprogram"
Current executable set to '/u/dgg6b/CS01/Assemble/lab/finalprogram' (x86_64).
(lldb) b main
Breakpoint 1: where = finalprogram`main, address = 0x0000000000401108
(lldb)
```

```
dgg6b@portal02:~/CS01/Assemble/lab$ lldb finalprogram
(lldb) target create "finalprogram"
Current executable set to '/u/dgg6b/CS01/Assemble/lab/finalprogram' (x86_64).
(lldb) b main
Breakpoint 1: where = finalprogram`main, address = 0x0000000000401108
(lldb) run
Process 3808778 launched: '/u/dgg6b/CS01/Assemble/lab/finalprogram' (x86_64)
Process 3808778 stopped
* thread #1, name = 'finalprogram', stop reason = breakpoint 1.1
    frame #0: 0x0000000000401108 finalprogram`main
finalprogram`main:
->  0x401108 <+0>: pushq  $0x4
    0x40110a <+2>: pushq  $0x5
    0x40110c <+4>: popq   %rax
    0x40110d <+5>: popq   %rbx
(lldb)
```

```
dgg6b@portal02:~/CS01/Assemble/lab$ lldb finalprogram
(lldb) target create "finalprogram"
Current executable set to '/u/dgg6b/CS01/Assemble/lab/finalprogram' (x86_64).
(lldb) b main
Breakpoint 1: where = finalprogram`main, address = 0x0000000000401108
(lldb) run
Process 3808778 launched: '/u/dgg6b/CS01/Assemble/lab/finalprogram' (x86_64)
Process 3808778 stopped
* thread #1, name = 'finalprogram', stop reason = breakpoint 1.1
    frame #0: 0x0000000000401108 finalprogram`main
finalprogram`main:
->  0x401108 <+0>: pushq  $0x4
    0x40110a <+2>: pushq  $0x5
    0x40110c <+4>: popq   %rax
    0x40110d <+5>: popq   %rbx
(lldb) register read $rsp
     rsp = 0x00007fffffffe4c8
(lldb)
```

```
dgg6b@portal02:~/CS01/Assemble/lab$ lldb finalprogram
(lldb) target create "finalprogram"
Current executable set to '/u/dgg6b/CS01/Assemble/lab/finalprogram' (x86_64).
(lldb) b main
Breakpoint 1: where = finalprogram`main, address = 0x0000000000401108
(lldb) run
Process 3808778 launched: '/u/dgg6b/CS01/Assemble/lab/finalprogram' (x86_64)
Process 3808778 stopped
* thread #1, name = 'finalprogram', stop reason = breakpoint 1.1
    frame #0: 0x0000000000401108 finalprogram`main
finalprogram`main:
->  0x401108 <+0>: pushq  $0x4
    0x40110a <+2>: pushq  $0x5
    0x40110c <+4>: popq   %rax
    0x40110d <+5>: popq   %rbx
(lldb) register read $rsp
     rsp = 0x00007fffffffe4c8
(lldb)
```

```
(lldb) target create "finalprogram"
Current executable set to '/u/dgg6b/CS01/Assemble/lab/finalprogram' (x86_64).
(lldb) b main
Breakpoint 1: where = finalprogram`main, address = 0x0000000000401108
(lldb) run
Process 3808778 launched: '/u/dgg6b/CS01/Assemble/lab/finalprogram' (x86_64)
Process 3808778 stopped
* thread #1, name = 'finalprogram', stop reason = breakpoint 1.1
    frame #0: 0x0000000000401108 finalprogram`main
finalprogram`main:
->  0x401108 <+0>: pushq  $0x4
    0x40110a <+2>: pushq  $0x5
    0x40110c <+4>: popq   %rax
    0x40110d <+5>: popq   %rbx
(lldb) register read $rsp
     rsp = 0x00007fffffffe4c8
(lldb) stepi
Process 3808778 stopped
* thread #1, name = 'finalprogram', stop reason = instruction step into
    frame #0: 0x000000000040110a finalprogram`main + 2
finalprogram`main:
->  0x40110a <+2>: pushq  $0x5
    0x40110c <+4>: popq   %rax
    0x40110d <+5>: popq   %rbx
    0x40110e <+6>: addq   %rax, %rbx
(lldb) register read $rsp
     rsp = 0x00007fffffffe4c0
(lldb)
```

stepi tells the debugger to execute the instruction

The arrow points the next instruction that will be executed

```
(lldb) target create "finalprogram"
Current executable set to '/u/dgg6b/CS01/Assemble/lab/finalprogram' (x86_64).
(lldb) b main
Breakpoint 1: where = finalprogram`main, address = 0x0000000000401108
(lldb) run
Process 3808778 launched: '/u/dgg6b/CS01/Assemble/lab/finalprogram' (x86_64)
Process 3808778 stopped
* thread #1, name = 'finalprogram', stop reason = breakpoint 1.1
    frame #0: 0x0000000000401108 finalprogram`main
finalprogram`main:
->  0x401108 <+0>: pushq  $0x4
    0x40110a <+2>: pushq  $0x5
    0x40110c <+4>: popq   %rax
    0x40110d <+5>: popq   %rbx
(lldb) register read $rsp
     rsp = 0x00007fffffffe4c8
(lldb) stepi
Process 3808778 stopped
* thread #1, name = 'finalprogram', stop reason = instruction step into
    frame #0: 0x000000000040110a finalprogram`main + 2
finalprogram`main:
->  0x40110a <+2>: pushq  $0x5
    0x40110c <+4>: popq   %rax
    0x40110d <+5>: popq   %rbx
    0x40110e <+6>: addq   %rax, %rbx
(lldb) register read $rsp
     rsp = 0x00007fffffffe4c0
(lldb)
```

Notice that RSP has been decremented by 8
The size of a quad word

```
Breakpoint 1: where = finalprogram`main, address = 0x0000000000401108
(lldb) run
Process 3808778 launched: '/u/dgg6b/CS01/Assemble/lab/finalprogram' (x86_64)
Process 3808778 stopped
* thread #1, name = 'finalprogram', stop reason = breakpoint 1.1
    frame #0: 0x0000000000401108 finalprogram`main
finalprogram`main:
->  0x401108 <+0>: pushq  $0x4
    0x40110a <+2>: pushq  $0x5
    0x40110c <+4>: popq   %rax
    0x40110d <+5>: popq   %rbx
(lldb) register read $rsp
     rsp = 0x00007fffffffe4c8
(lldb) stepi
Process 3808778 stopped
* thread #1, name = 'finalprogram', stop reason = instruction step into
    frame #0: 0x000000000040110a finalprogram`main + 2
finalprogram`main:
->  0x40110a <+2>: pushq  $0x5
    0x40110c <+4>: popq   %rax
    0x40110d <+5>: popq   %rbx
    0x40110e <+6>: addq   %rax, %rbx
(lldb) register read $rsp
     rsp = 0x00007fffffffe4c0
(lldb) me rea -s8 -fx -c4 $rsp
0x7fffffffe4c0: 0x0000000000000004 0x00007ffff7da8d90
0x7fffffffe4d0: 0x0000000000000000 0x0000000000401108
(lldb)
```

me – memory
rea - read
-s8 8 byte blocks
-fx format hex
-c4 four blocks
$rsp – memory address

```
Breakpoint 1: where = finalprogram`main, address = 0x0000000000401108
(lldb) run
Process 3808778 launched: '/u/dgg6b/CS01/Assemble/lab/finalprogram' (x86_64)
Process 3808778 stopped
* thread #1, name = 'finalprogram', stop reason = breakpoint 1.1
    frame #0: 0x0000000000401108 finalprogram`main
finalprogram`main:
->  0x401108 <+0>: pushq  $0x4
    0x40110a <+2>: pushq  $0x5
    0x40110c <+4>: popq   %rax
    0x40110d <+5>: popq   %rbx
(lldb) register read $rsp
     rsp = 0x00007fffffffe4c8
(lldb) stepi
Process 3808778 stopped
* thread #1, name = 'finalprogram', stop reason = instruction step into
    frame #0: 0x000000000040110a finalprogram`main + 2
finalprogram`main:
->  0x40110a <+2>: pushq  $0x5
    0x40110c <+4>: popq   %rax
    0x40110d <+5>: popq   %rbx
    0x40110e <+6>: addq   %rax, %rbx
(lldb) register read $rsp
     rsp = 0x00007fffffffe4c0
(lldb) me rea -s8 -fx -c4 $rsp
0x7fffffffe4c0: 0x0000000000000004 0x00007ffff7da8d90
0x7fffffffe4d0: 0x0000000000000000 0x0000000000401108
(lldb)
```

Notice that 4 is pushed to the stack

```
    0x40110a <+2>: pushq  $0x5
    0x40110c <+4>: popq   %rax
    0x40110d <+5>: popq   %rbx
(lldb) register read $rsp
     rsp = 0x00007fffffffe4c8
(lldb) stepi
Process 3808778 stopped
* thread #1, name = 'finalprogram', stop reason = instruction step into
    frame #0: 0x000000000040110a finalprogram`main + 2
finalprogram`main:
->  0x40110a <+2>: pushq  $0x5
    0x40110c <+4>: popq   %rax
    0x40110d <+5>: popq   %rbx
    0x40110e <+6>: addq   %rax, %rbx
(lldb) register read $rsp
     rsp = 0x00007fffffffe4c0
(lldb) me rea -s8 -fx -c4 $rsp
0x7fffffffe4c0: 0x0000000000000004 0x00007ffff7da8d90
0x7fffffffe4d0: 0x0000000000000000 0x0000000000401108
(lldb) d
finalprogram`main:
    0x401108 <+0>: pushq  $0x4
->  0x40110a <+2>: pushq  $0x5
    0x40110c <+4>: popq   %rax
    0x40110d <+5>: popq   %rbx
    0x40110e <+6>: addq   %rax, %rbx
    0x401111 <+9>: pushq  %rbx
(lldb)
```

d – disassemble command, great way to get perspective on what you are currently working on.

```
finalprogram`main:
->  0x40110a <+2>: pushq  $0x5
    0x40110c <+4>: popq   %rax
    0x40110d <+5>: popq   %rbx
    0x40110e <+6>: addq   %rax, %rbx
(lldb) register read $rsp
     rsp = 0x00007fffffffe4c0
(lldb) me rea -s8 -fx -c4 $rsp
0x7fffffffe4c0: 0x0000000000000004 0x00007ffff7da8d90
0x7fffffffe4d0: 0x0000000000000000 0x0000000000401108
(lldb) d
finalprogram`main:
    0x401108 <+0>: pushq  $0x4
->  0x40110a <+2>: pushq  $0x5
    0x40110c <+4>: popq   %rax
    0x40110d <+5>: popq   %rbx
    0x40110e <+6>: addq   %rax, %rbx
    0x401111 <+9>: pushq  %rbx
(lldb) stepi
Process 3808778 stopped
* thread #1, name = 'finalprogram', stop reason = instruction step into
    frame #0: 0x000000000040110c finalprogram`main + 4
finalprogram`main:
->  0x40110c <+4>: popq   %rax
    0x40110d <+5>: popq   %rbx
    0x40110e <+6>: addq   %rax, %rbx
    0x401111 <+9>: pushq  %rbx
(lldb)
```

Execute the
push instruction

```
    0x40110c <+4>: popq   %rax
    0x40110d <+5>: popq   %rbx
    0x40110e <+6>: addq   %rax, %rbx
(lldb) register read $rsp
    rsp = 0x00007fffffffe4c0
(lldb) me rea -s8 -fx -c4 $rsp
0x7fffffffe4c0: 0x0000000000000004 0x00007ffff7da8d90
0x7fffffffe4d0: 0x0000000000000000 0x0000000000401108
(lldb) d
finalprogram`main:
    0x401108 <+0>: pushq  $0x4
->  0x40110a <+2>: pushq  $0x5
    0x40110c <+4>: popq   %rax
    0x40110d <+5>: popq   %rbx
    0x40110e <+6>: addq   %rax, %rbx
    0x401111 <+9>: pushq  %rbx
(lldb) stepi
Process 3808778 stopped
* thread #1, name = 'finalprogram', stop reason = instruction step into
    frame #0: 0x000000000040110c finalprogram`main + 4
finalprogram`main:
->  0x40110c <+4>: popq   %rax
    0x40110d <+5>: popq   %rbx
    0x40110e <+6>: addq   %rax, %rbx
    0x401111 <+9>: pushq  %rbx
(lldb) register read $rsp
    rsp = 0x00007fffffffe4b8
(lldb)
```

RSP is again decremented by 8

```
(lldb) register read $rsp
     rsp = 0x00007fffffffe4c0
(lldb) me rea -s8 -fx -c4 $rsp
0x7fffffffe4c0: 0x0000000000000004 0x00007ffff7da8d90
0x7fffffffe4d0: 0x0000000000000000 0x0000000000401108
(lldb) d
finalprogram`main:
    0x401108 <+0>: pushq  $0x4
->  0x40110a <+2>: pushq  $0x5
    0x40110c <+4>: popq   %rax
    0x40110d <+5>: popq   %rbx
    0x40110e <+6>: addq   %rax, %rbx
    0x401111 <+9>: pushq  %rbx
(lldb) stepi
Process 3808778 stopped
* thread #1, name = 'finalprogram', stop reason = instruction step into
    frame #0: 0x000000000040110c finalprogram`main + 4
finalprogram`main:
->  0x40110c <+4>: popq   %rax
    0x40110d <+5>: popq   %rbx
    0x40110e <+6>: addq   %rax, %rbx
    0x401111 <+9>: pushq  %rbx
(lldb) register read $rsp
     rsp = 0x00007fffffffe4b8
(lldb) me rea -s8 -fx -c4 $rsp
0x7fffffffe4b8: 0x0000000000000005 0x0000000000000004
0x7fffffffe4c8: 0x00007ffff7da8d90 0x0000000000000000
(lldb)
```

```
(lldb) register read $rsp
     rsp = 0x00007fffffffe4c0
(lldb) me rea -s8 -fx -c4 $rsp
0x7fffffffe4c0: 0x0000000000000004 0x00007ffff7da8d90
0x7fffffffe4d0: 0x0000000000000000 0x0000000000401108
(lldb) d
finalprogram`main:
    0x401108 <+0>: pushq  $0x4
->  0x40110a <+2>: pushq  $0x5
    0x40110c <+4>: popq   %rax
    0x40110d <+5>: popq   %rbx
    0x40110e <+6>: addq   %rax, %rbx
    0x401111 <+9>: pushq  %rbx
(lldb) stepi
Process 3808778 stopped
* thread #1, name = 'finalprogram', stop reason = instruction step into
    frame #0: 0x000000000040110c finalprogram`main + 4
finalprogram`main:
->  0x40110c <+4>: popq   %rax
    0x40110d <+5>: popq   %rbx
    0x40110e <+6>: addq   %rax, %rbx
    0x401111 <+9>: pushq  %rbx
(lldb) register read $rsp
     rsp = 0x00007fffffffe4b8
(lldb) me rea -s8 -fx -c4 $rsp
0x7fffffffe4b8: 0x0000000000000005 0x0000000000000004
0x7fffffffe4c8: 0x00007ffff7da8d90 0x0000000000000000
(lldb)
```

```
(lldb) register read $rsp
     rsp = 0x00007fffffffe4c0
(lldb) me rea -s8 -fx -c4 $rsp
0x7fffffffe4c0: 0x0000000000000004 0x00007ffff7da8d90
0x7fffffffe4d0: 0x0000000000000000 0x0000000000401108
(lldb) d
finalprogram`main:
    0x401108 <+0>: pushq  $0x4
->  0x40110a <+2>: pushq  $0x5
    0x40110c <+4>: popq   %rax
    0x40110d <+5>: popq   %rbx
    0x40110e <+6>: addq   %rax, %rbx
    0x401111 <+9>: pushq  %rbx
(lldb) stepi
Process 3808778 stopped
* thread #1, name = 'finalprogram', stop reason = instruction step into
    frame #0: 0x000000000040110c finalprogram`main + 4
finalprogram`main:
->  0x40110c <+4>: popq   %rax
    0x40110d <+5>: popq   %rbx
    0x40110e <+6>: addq   %rax, %rbx
    0x401111 <+9>: pushq  %rbx
(lldb) register read $rsp
     rsp = 0x00007fffffffe4b8
(lldb) me rea -s8 -fx -c4 $rsp
0x7fffffffe4b8: 0x0000000000000005 0x0000000000000004
0x7fffffffe4c8: 0x00007ffff7da8d90 0x0000000000000000
(lldb)
```

Stack grows to lower address

Lower address

```
(lldb) register read $rsp
     rsp = 0x00007fffffffe4c0
(lldb) me rea -s8 -fx -c4 $rsp
0x7fffffffe4c0: 0x0000000000000004 0x00007ffff7da8d90
0x7fffffffe4d0: 0x0000000000000000 0x0000000000401108
(lldb) d
finalprogram`main:
    0x401108 <+0>: pushq  $0x4
->  0x40110a <+2>: pushq  $0x5
    0x40110c <+4>: popq   %rax
    0x40110d <+5>: popq   %rbx
    0x40110e <+6>: addq   %rax, %rbx
    0x401111 <+9>: pushq  %rbx
(lldb) stepi
Process 3808778 stopped
* thread #1, name = 'finalprogram', stop reason = instruction step into
    frame #0: 0x000000000040110c finalprogram`main + 4
finalprogram`main:
->  0x40110c <+4>: popq   %rax
    0x40110d <+5>: popq   %rbx
    0x40110e <+6>: addq   %rax, %rbx
    0x401111 <+9>: pushq  %rbx
(lldb) register read $rsp
     rsp = 0x00007fffffffe4b8
(lldb) me rea -s8 -fx -c4 $rsp
0x7fffffffe4b8: 0x0000000000000005 0x0000000000000004
0x7fffffffe4c8: 0x00007ffff7da8d90 0x0000000000000000
(lldb)
```


Higher addresses

```
    0x40110c <+4>: popq   %rax
    0x40110d <+5>: popq   %rbx
    0x40110e <+6>: addq   %rax, %rbx
    0x401111 <+9>: pushq  %rbx
(lldb) stepi
Process 3808778 stopped
* thread #1, name = 'finalprogram', stop reason = instruction step into
    frame #0: 0x000000000040110c finalprogram`main + 4
finalprogram`main:
->  0x40110c <+4>: popq   %rax
    0x40110d <+5>: popq   %rbx
    0x40110e <+6>: addq   %rax, %rbx
    0x401111 <+9>: pushq  %rbx
(lldb) register read $rsp
     rsp = 0x00007fffffffe4b8
(lldb) me rea -s8 -fx -c4 $rsp
0x7fffffffe4b8: 0x0000000000000005 0x0000000000000004
0x7fffffffe4c8: 0x00007ffff7da8d90 0x0000000000000000
(lldb) stepi
Process 3808778 stopped
* thread #1, name = 'finalprogram', stop reason = instruction step into
    frame #0: 0x000000000040110d finalprogram`main + 5
finalprogram`main:
->  0x40110d <+5>: popq   %rbx
    0x40110e <+6>: addq   %rax, %rbx
    0x401111 <+9>: pushq  %rbx
    0x401112:      addb   %al, (%rax)
(lldb)
```

Let's execute the pop in %rax instruction

```
    0x40110e <+6>: addq   %rax, %rbx
    0x401111 <+9>: pushq  %rbx
(lldb) stepi
Process 3808778 stopped
* thread #1, name = 'finalprogram', stop reason = instruction step into
    frame #0: 0x000000000040110c finalprogram`main + 4
finalprogram`main:
->  0x40110c <+4>: popq   %rax
    0x40110d <+5>: popq   %rbx
    0x40110e <+6>: addq   %rax, %rbx
    0x401111 <+9>: pushq  %rbx
(lldb) register read $rsp
     rsp = 0x00007fffffffe4b8
(lldb) me rea -s8 -fx -c4 $rsp
0x7fffffffe4b8: 0x0000000000000005 0x0000000000000004
0x7fffffffe4c8: 0x00007ffff7da8d90 0x0000000000000000
(lldb) stepi
Process 3808778 stopped
* thread #1, name = 'finalprogram', stop reason = instruction step into
    frame #0: 0x000000000040110d finalprogram`main + 5
finalprogram`main:
->  0x40110d <+5>: popq   %rbx
    0x40110e <+6>: addq   %rax, %rbx
    0x401111 <+9>: pushq  %rbx
    0x401112:      addb   %al, (%rax)
(lldb) register read $rax
     rax = 0x0000000000000005
(lldb)
```

The value at the top of stack has been saved to rax.

```
(lldb) stepi
Process 3808778 stopped
* thread #1, name = 'finalprogram', stop reason = instruction step into
    frame #0: 0x000000000040110c finalprogram`main + 4
finalprogram`main:
->  0x40110c <+4>: popq   %rax
    0x40110d <+5>: popq   %rbx
    0x40110e <+6>: addq   %rax, %rbx
    0x401111 <+9>: pushq  %rbx
(lldb) register read $rsp
     rsp = 0x00007fffffffe4b8
(lldb) me rea -s8 -fx -c4 $rsp
0x7fffffffe4b8: 0x0000000000000005 0x0000000000000004
0x7fffffffe4c8: 0x00007ffff7da8d90 0x0000000000000000
(lldb) stepi
Process 3808778 stopped
* thread #1, name = 'finalprogram', stop reason = instruction step into
    frame #0: 0x000000000040110d finalprogram`main + 5
finalprogram`main:
->  0x40110d <+5>: popq   %rbx
    0x40110e <+6>: addq   %rax, %rbx
    0x401111 <+9>: pushq  %rbx
    0x401112:      addb   %al, (%rax)
(lldb) register read $rax
     rax = 0x0000000000000005
(lldb) register read $rsp
     rsp = 0x00007fffffffe4c0
(lldb)
```

RSP has been incremented.

```
    frame #0: 0x000000000040110c finalprogram`main + 4
finalprogram`main:
->  0x40110c <+4>: popq   %rax
    0x40110d <+5>: popq   %rbx
    0x40110e <+6>: addq   %rax, %rbx
    0x401111 <+9>: pushq  %rbx
(lldb) register read $rsp
     rsp = 0x00007fffffffe4b8
(lldb) me rea -s8 -fx -c4 $rsp
0x7fffffffe4b8: 0x0000000000000005 0x0000000000000004
0x7fffffffe4c8: 0x00007ffff7da8d90 0x0000000000000000
(lldb) stepi
Process 3808778 stopped
* thread #1, name = 'finalprogram', stop reason = instruction step into
    frame #0: 0x000000000040110d finalprogram`main + 5
finalprogram`main:
->  0x40110d <+5>: popq   %rbx
    0x40110e <+6>: addq   %rax, %rbx
    0x401111 <+9>: pushq  %rbx
    0x401112:      addb   %al, (%rax)
(lldb) register read $rax
     rax = 0x0000000000000005
(lldb) register read $rsp
     rsp = 0x00007fffffffe4c0
(lldb) me rea -s8 -fx -c4 $rsp
0x7fffffffe4c0: 0x0000000000000004 0x00007ffff7da8d90
0x7fffffffe4d0: 0x0000000000000000 0x0000000000401108
(lldb)
```

Top stack now contains 4

```
    0x40110d <+5>: popq   %rbx
    0x40110e <+6>: addq   %rax, %rbx
    0x401111 <+9>: pushq  %rbx
(lldb) register read $rsp
     rsp = 0x00007fffffffe4b8
(lldb) me rea -s8 -fx -c4 $rsp
0x7fffffffe4b8: 0x0000000000000005 0x0000000000000004
0x7fffffffe4c8: 0x00007ffff7da8d90 0x0000000000000000
(lldb) stepi
Process 3808778 stopped
* thread #1, name = 'finalprogram', stop reason = instruction step into
    frame #0: 0x000000000040110d finalprogram`main + 5
finalprogram`main:
->  0x40110d <+5>: popq   %rbx
    0x40110e <+6>: addq   %rax, %rbx
    0x401111 <+9>: pushq  %rbx
    0x401112:      addb   %al, (%rax)
(lldb) register read $rax
     rax = 0x0000000000000005
(lldb) register read $rsp
     rsp = 0x00007fffffffe4c0
(lldb) me rea -s8 -fx -c4 $rsp
0x7fffffffe4c0: 0x0000000000000004 0x00007ffff7da8d90
0x7fffffffe4d0: 0x0000000000000000 0x0000000000401108
(lldb) me rea -s8 -fx -c4 $rsp-8
0x7fffffffe4b8: 0x0000000000000005 0x0000000000000004
0x7fffffffe4c8: 0x00007ffff7da8d90 0x0000000000000000
(lldb)
```

If we look at lower address in memory we'll that 5 is still there we haven't deleted it. We've just moved RSP

```
finalprogram`main:
->  0x40110d <+5>: popq   %rbx
    0x40110e <+6>: addq   %rax, %rbx
    0x401111 <+9>: pushq  %rbx
    0x401112:      addb   %al, (%rax)
(lldb) register read $rax
     rax = 0x0000000000000005
(lldb) register read $rsp
     rsp = 0x00007fffffffe4c0
(lldb) me rea -s8 -fx -c4 $rsp
0x7fffffffe4c0: 0x0000000000000004 0x00007ffff7da8d90
0x7fffffffe4d0: 0x0000000000000000 0x0000000000401108
(lldb) me rea -s8 -fx -c4 $rsp-8
0x7fffffffe4b8: 0x0000000000000005 0x0000000000000004
0x7fffffffe4c8: 0x00007ffff7da8d90 0x0000000000000000
(lldb) stepi
Process 3808778 stopped
* thread #1, name = 'finalprogram', stop reason = instruction step into
    frame #0: 0x000000000040110e finalprogram`main + 6
finalprogram`main:
->  0x40110e <+6>: addq   %rax, %rbx
    0x401111 <+9>: pushq  %rbx
    0x401112:      addb   %al, (%rax)
finalprogram`_fini:
    0x401114 <+0>: endbr64
(lldb) register read $rbx
     rbx = 0x0000000000000004
(lldb)
```

Now register rbx contains the value 4.

```
     rsp = 0x00007fffffffe4c0
(lldb) me rea -s8 -fx -c4 $rsp
0x7fffffffe4c0: 0x0000000000000004 0x00007ffff7da8d90
0x7fffffffe4d0: 0x0000000000000000 0x0000000000401108
(lldb) me rea -s8 -fx -c4 $rsp-8
0x7fffffffe4b8: 0x0000000000000005 0x0000000000000004
0x7fffffffe4c8: 0x00007ffff7da8d90 0x0000000000000000
(lldb) stepi
Process 3808778 stopped
* thread #1, name = 'finalprogram', stop reason = instruction step into
    frame #0: 0x000000000040110e finalprogram`main + 6
finalprogram`main:
->  0x40110e <+6>: addq   %rax, %rbx
    0x401111 <+9>: pushq  %rbx
    0x401112:      addb   %al, (%rax)
finalprogram`_fini:
    0x401114 <+0>: endbr64
(lldb) register read $rbx
     rbx = 0x0000000000000004
(lldb) d
finalprogram`main:
    0x401108 <+0>: pushq  $0x4
    0x40110a <+2>: pushq  $0x5
    0x40110c <+4>: popq   %rax
    0x40110d <+5>: popq   %rbx
->  0x40110e <+6>: addq   %rax, %rbx
    0x401111 <+9>: pushq  %rbx
(lldb)
```

Next we will add rax and rbx and store the result in rbx.

In At&T syntax the designation register is always last.

```
0x7fffffffe4d0: 0x0000000000000000 0x0000000000401108
(lldb) me rea -s8 -fx -c4 $rsp-8
0x7fffffffe4b8: 0x0000000000000005 0x0000000000000004
0x7fffffffe4c8: 0x00007ffff7da8d90 0x0000000000000000
(lldb) stepi
Process 3808778 stopped
* thread #1, name = 'finalprogram', stop reason = instruction step into
    frame #0: 0x000000000040110e finalprogram`main + 6
finalprogram`main:
->  0x40110e <+6>: addq   %rax, %rbx
    0x401111 <+9>: pushq  %rbx
    0x401112:      addb   %al, (%rax)
finalprogram`_fini:
    0x401114 <+0>: endbr64
(lldb) register read $rbx
     rbx = 0x0000000000000004
(lldb) d
finalprogram`main:
    0x401108 <+0>: pushq  $0x4
    0x40110a <+2>: pushq  $0x5
    0x40110c <+4>: popq   %rax
    0x40110d <+5>: popq   %rbx
->  0x40110e <+6>: addq   %rax, %rbx
    0x401111 <+9>: pushq  %rbx
(lldb) register read $rax $rbx
     rax = 0x0000000000000005
     rbx = 0x0000000000000004
(lldb)
```

Here is the state of the registers before we execute the instruction

```
    0x401114 <+0>: endbr64
(lldb) register read $rbx
     rbx = 0x0000000000000004
(lldb) d
finalprogram`main:
    0x401108 <+0>: pushq  $0x4
    0x40110a <+2>: pushq  $0x5
    0x40110c <+4>: popq   %rax
    0x40110d <+5>: popq   %rbx
->  0x40110e <+6>: addq   %rax, %rbx
    0x401111 <+9>: pushq  %rbx
(lldb) register read $rax $rbx
     rax = 0x0000000000000005
     rbx = 0x0000000000000004
(lldb) stepi
Process 3808778 stopped
* thread #1, name = 'finalprogram', stop reason = instruction step into
    frame #0: 0x0000000000401111 finalprogram`main + 9
finalprogram`main:
->  0x401111 <+9>: pushq  %rbx
    0x401112:      addb   %al, (%rax)
finalprogram`_fini:
    0x401114 <+0>: endbr64
    0x401118 <+4>: subq   $0x8, %rsp
(lldb) register read $rax $rbx
     rax = 0x0000000000000005
     rbx = 0x0000000000000009
(lldb)
```

Notice that the result is stored in rbx. Remember the destination is always last

```
   0x401112:      addb   %al, (%rax)
finalprogram`_fini:
   0x401114 <+0>: endbr64
   0x401118 <+4>: subq   $0x8, %rsp
(lldb) register read $rax $rbx
     rax = 0x0000000000000005
     rbx = 0x0000000000000009
(lldb) d
finalprogram`main:
   0x401108 <+0>: pushq  $0x4
   0x40110a <+2>: pushq  $0x5
   0x40110c <+4>: popq   %rax
   0x40110d <+5>: popq   %rbx
   0x40110e <+6>: addq   %rax, %rbx
-> 0x401111 <+9>: pushq  %rbx
(lldb) stepi
Process 3808778 stopped
* thread #1, name = 'finalprogram', stop reason = instruction step into
    frame #0: 0x0000000000401112 finalprogram
   0x401112:      addb   %al, (%rax)
finalprogram`_fini:
   0x401114 <+0>: endbr64
   0x401118 <+4>: subq   $0x8, %rsp
   0x40111c <+8>: addq   $0x8, %rsp
(lldb) me rea -s8 -fx -c4 $rsp
0x7fffffffe4c0: 0x0000000000000009 0x00007ffff7da8d90
0x7fffffffe4d0: 0x0000000000000000 0x0000000000401108
(lldb)
```

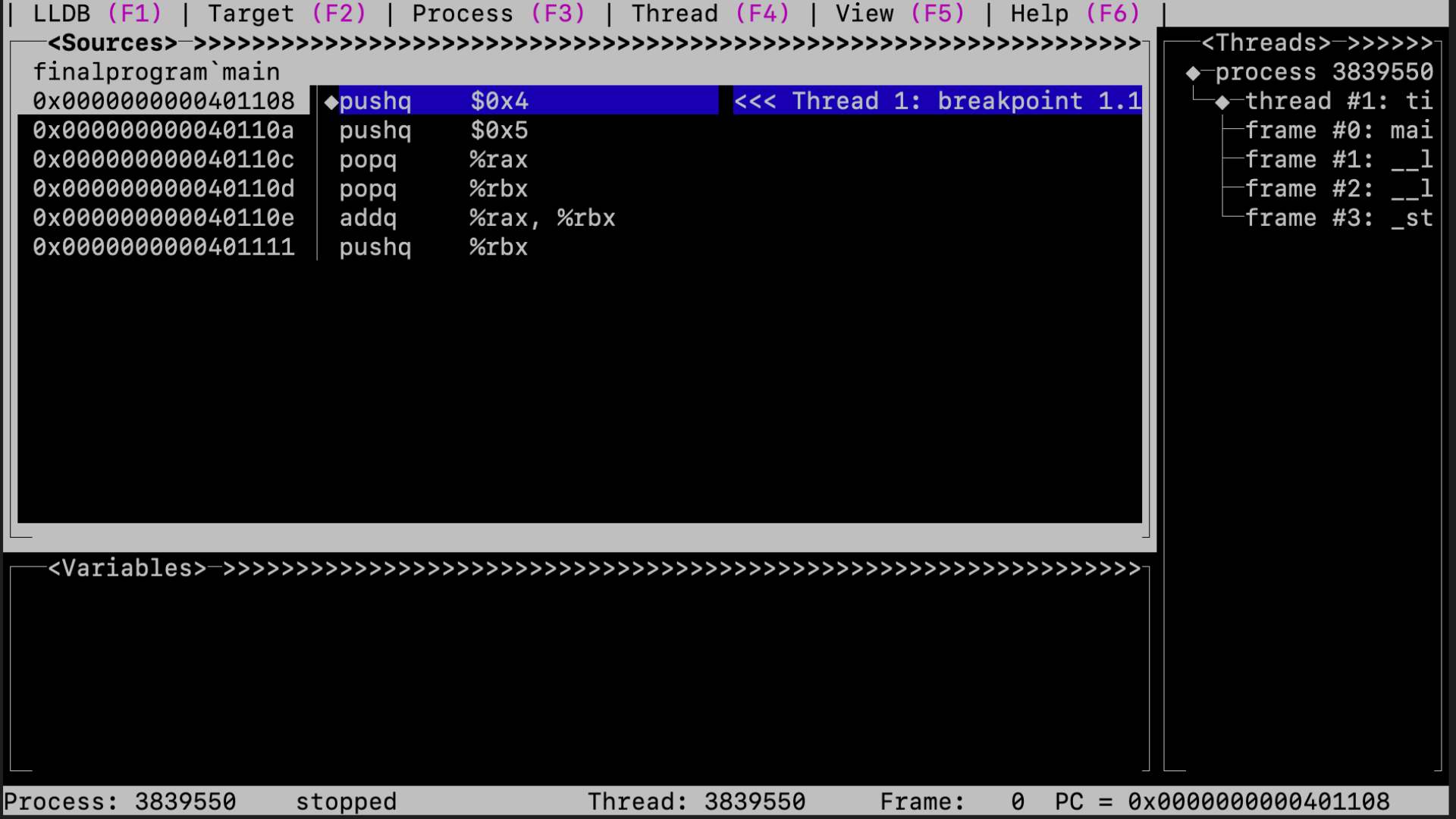
Value in rbx store to the stack

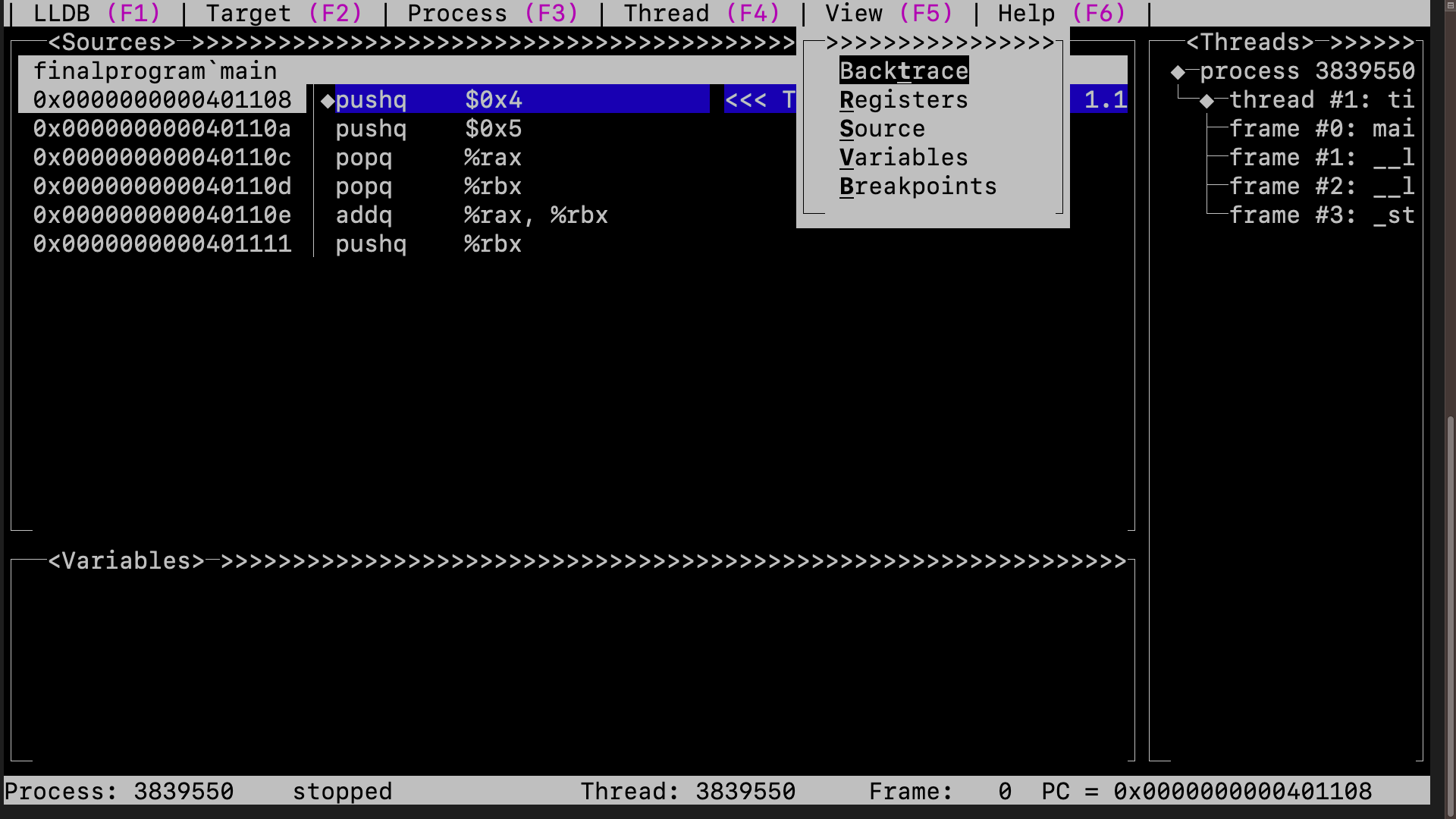
NOW LET’S DO THAT AGAIN WITH THE GUI

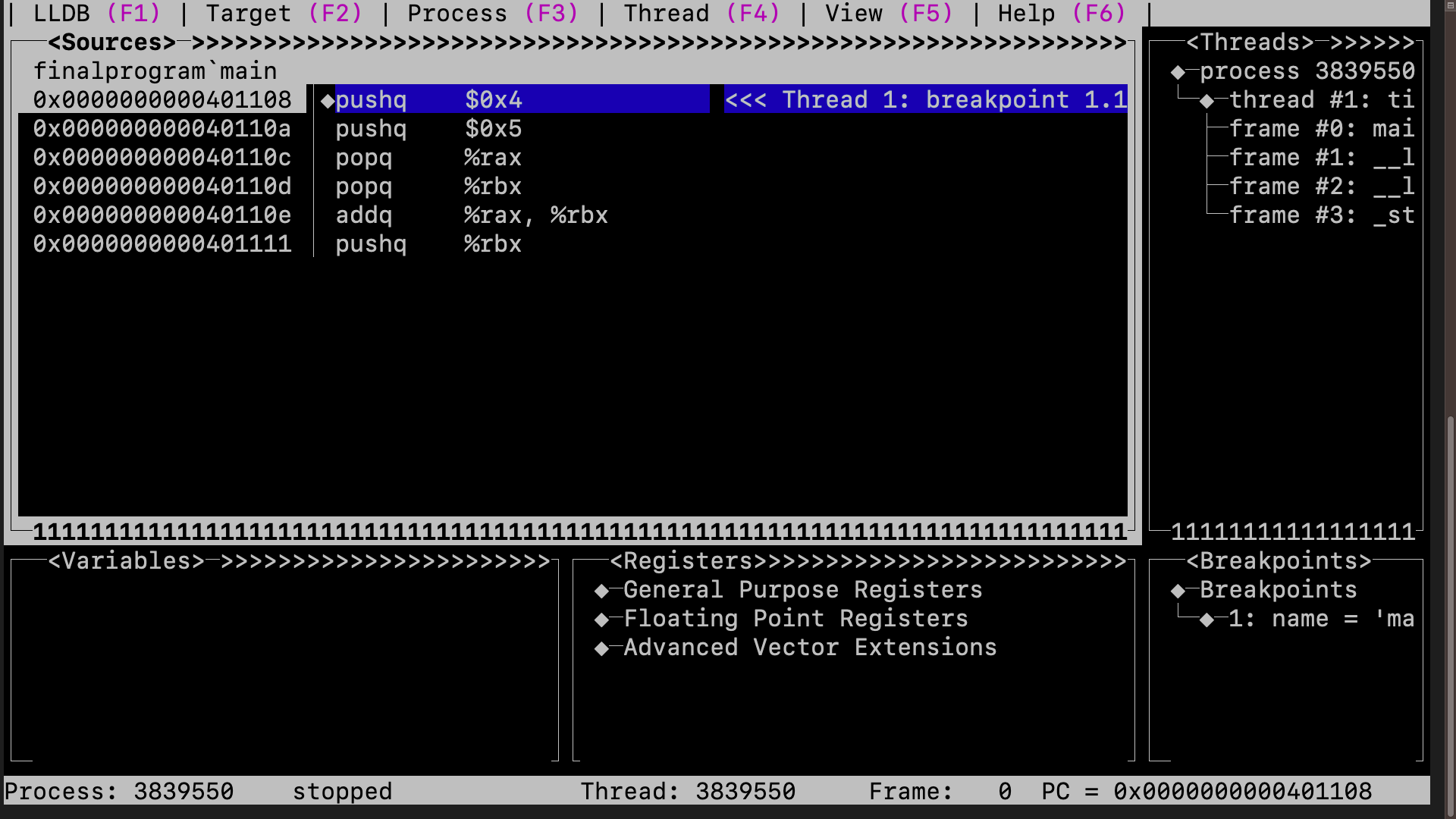
YOU CAN MOVE BACK AN FORTH IF YOU WANT TO INSPECT MEMORY

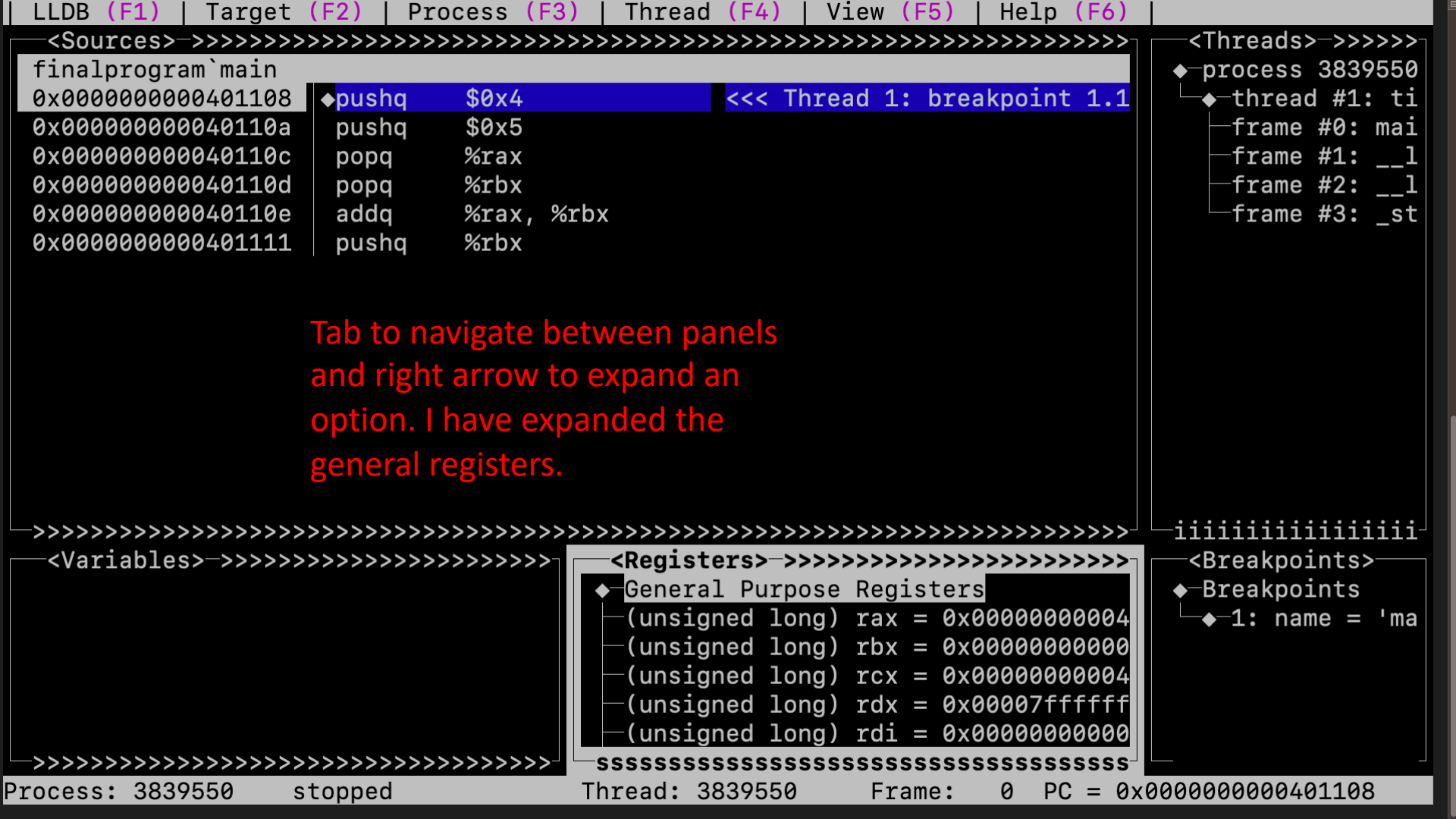
```
dgg6b@portal02:~/CS01/Assemble/lab$ lldb finalprogram
(lldb) target create "finalprogram"
Current executable set to '/u/dgg6b/CS01/Assemble/lab/finalprogram' (x86_64).
(lldb) b main
Breakpoint 1: where = finalprogram`main, address = 0x0000000000401108
(lldb) run
Process 3839550 launched: '/u/dgg6b/CS01/Assemble/lab/finalprogram' (x86_64)
Process 3839550 stopped
* thread #1, name = 'finalprogram', stop reason = breakpoint 1.1
    frame #0: 0x0000000000401108 finalprogram`main
finalprogram`main:
->  0x401108 <+0>: pushq  $0x4
    0x40110a <+2>: pushq  $0x5
    0x40110c <+4>: popq   %rax
    0x40110d <+5>: popq   %rbx
(lldb) gui
```

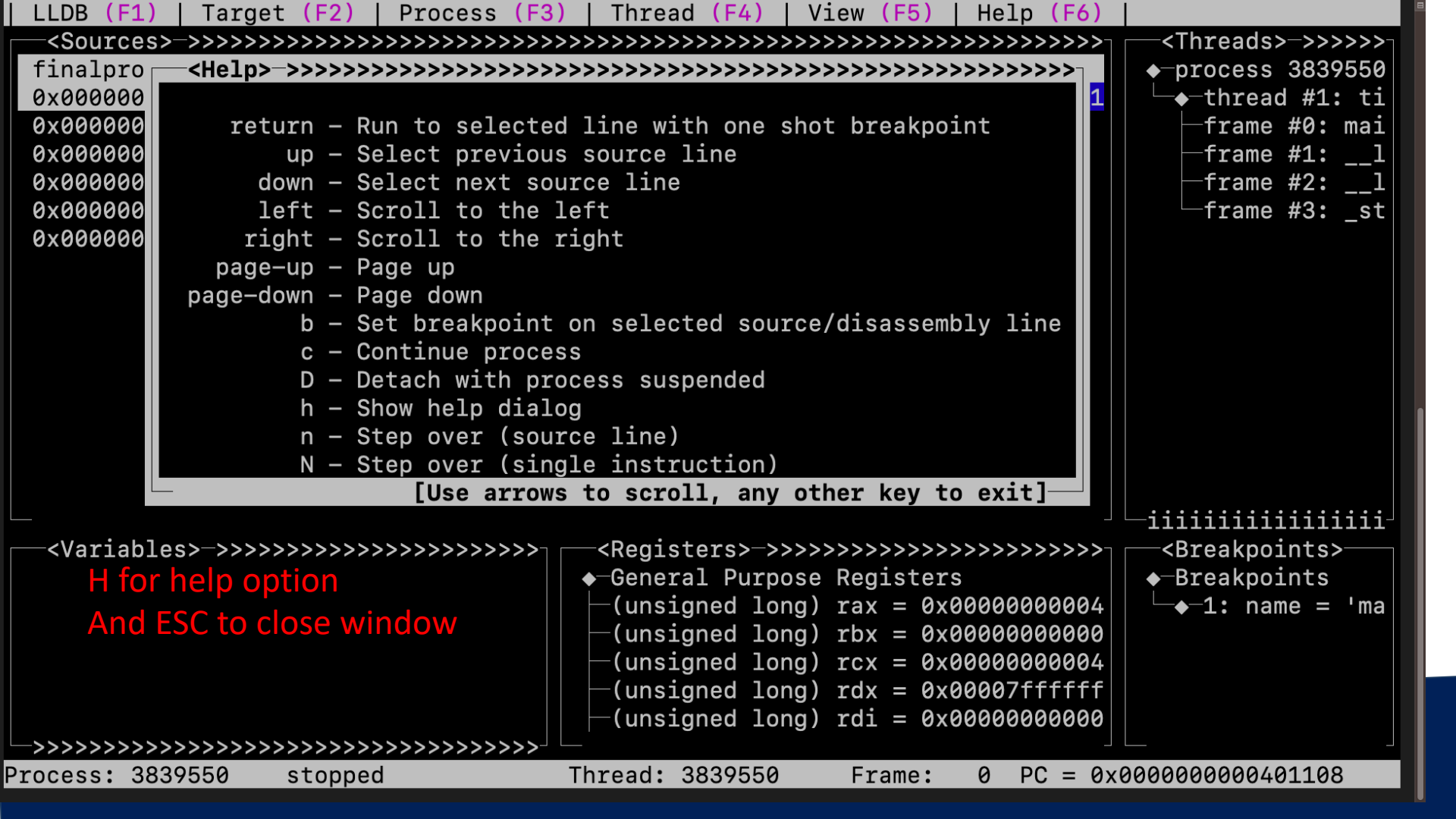
Set up just like you did before but this time instead of using stepi step over an instruction launch the gui

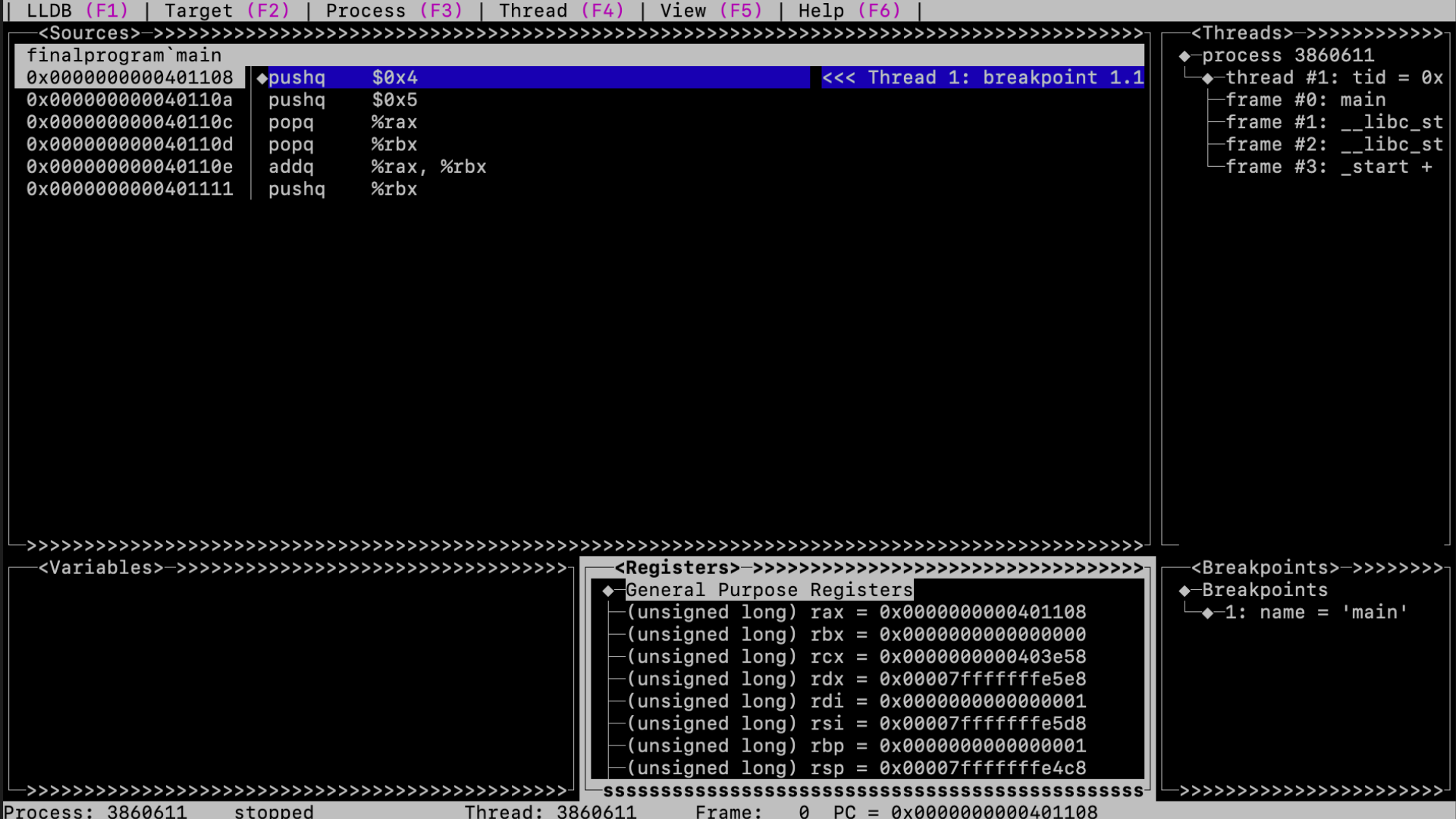

| LLDB (F1) | Target (F2) | Process (F3) | Thread (F4) | View (F5) | Help (F6) |
<Sources>
finalprogram`main
0x0000000000401108 ◆pushq $0x4 <<< Thread 1: breakpoint 1.1
0x000000000040110a pushq $0x5
0x000000000040110c popq %rax
0x000000000040110d popq %rbx
0x000000000040110e addq %rax, %rbx
0x0000000000401111 pushq %rbx
<Threads>
◆process 3839550
◆thread #1: ti
frame #0: mai
frame #1: __l
frame #2: __l
frame #3: _st
<Variables>
Process: 3839550 stopped Thread: 3839550 Frame: 0 PC = 0x0000000000401108

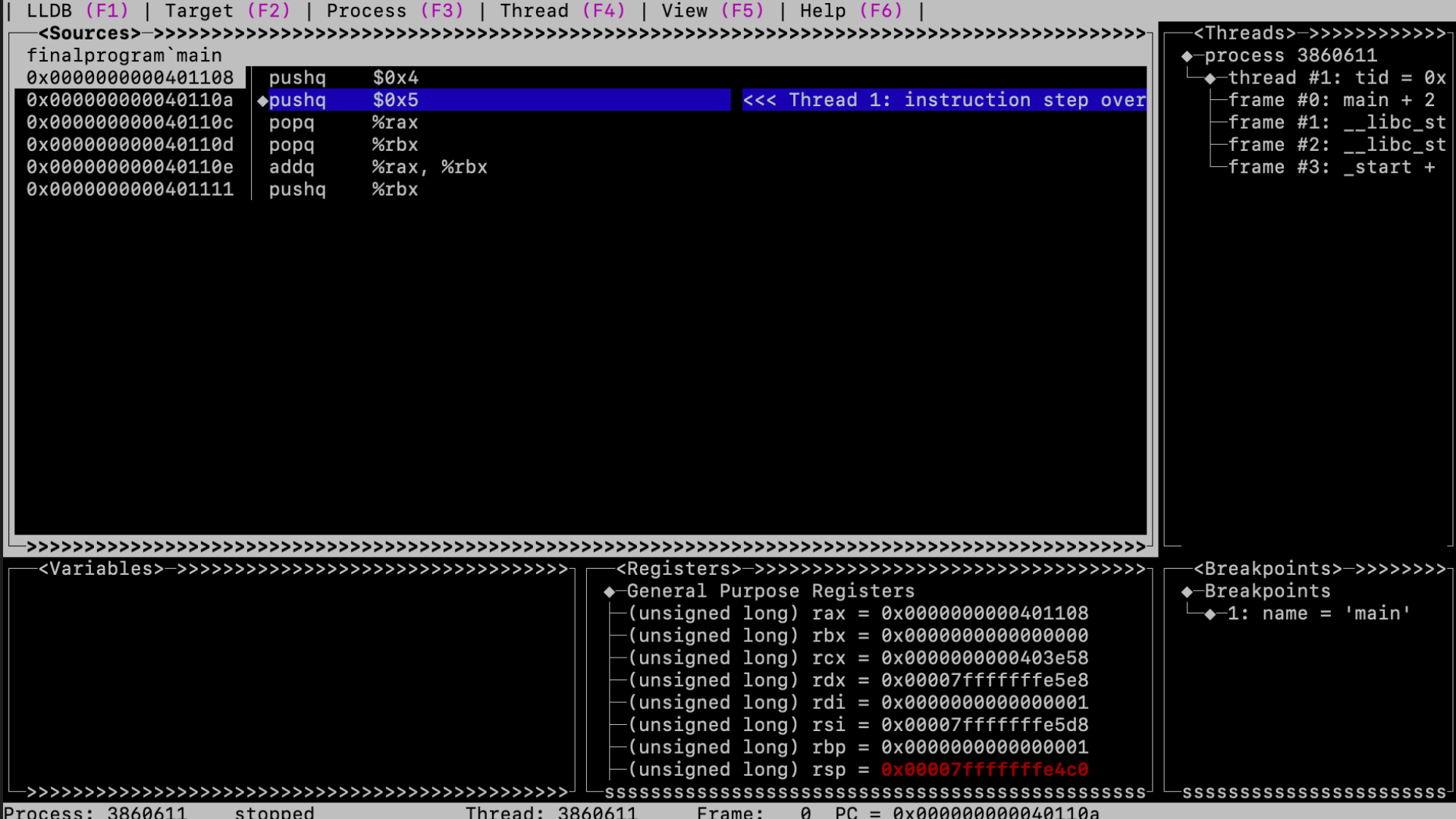

| LLDB (F1) | Target (F2) | Process (F3) | Thread (F4) | View (F5) | Help (F6) |
<Sources>
finalprogram`main
0x0000000000401108 pushq $0x4 <<< T 1.1
0x000000000040110a pushq $0x5
0x000000000040110c popq %rax
0x000000000040110d popq %rbx
0x000000000040110e addq %rax, %rbx
0x0000000000401111 pushq %rbx
Backtrace
Registers
Source
Variables
Breakpoints
<Threads>
process 3839550
thread #1: ti
frame #0: mai
frame #1: __l
frame #2: __l
frame #3: _st
<Variables>
Process: 3839550 stopped Thread: 3839550 Frame: 0 PC = 0x0000000000401108

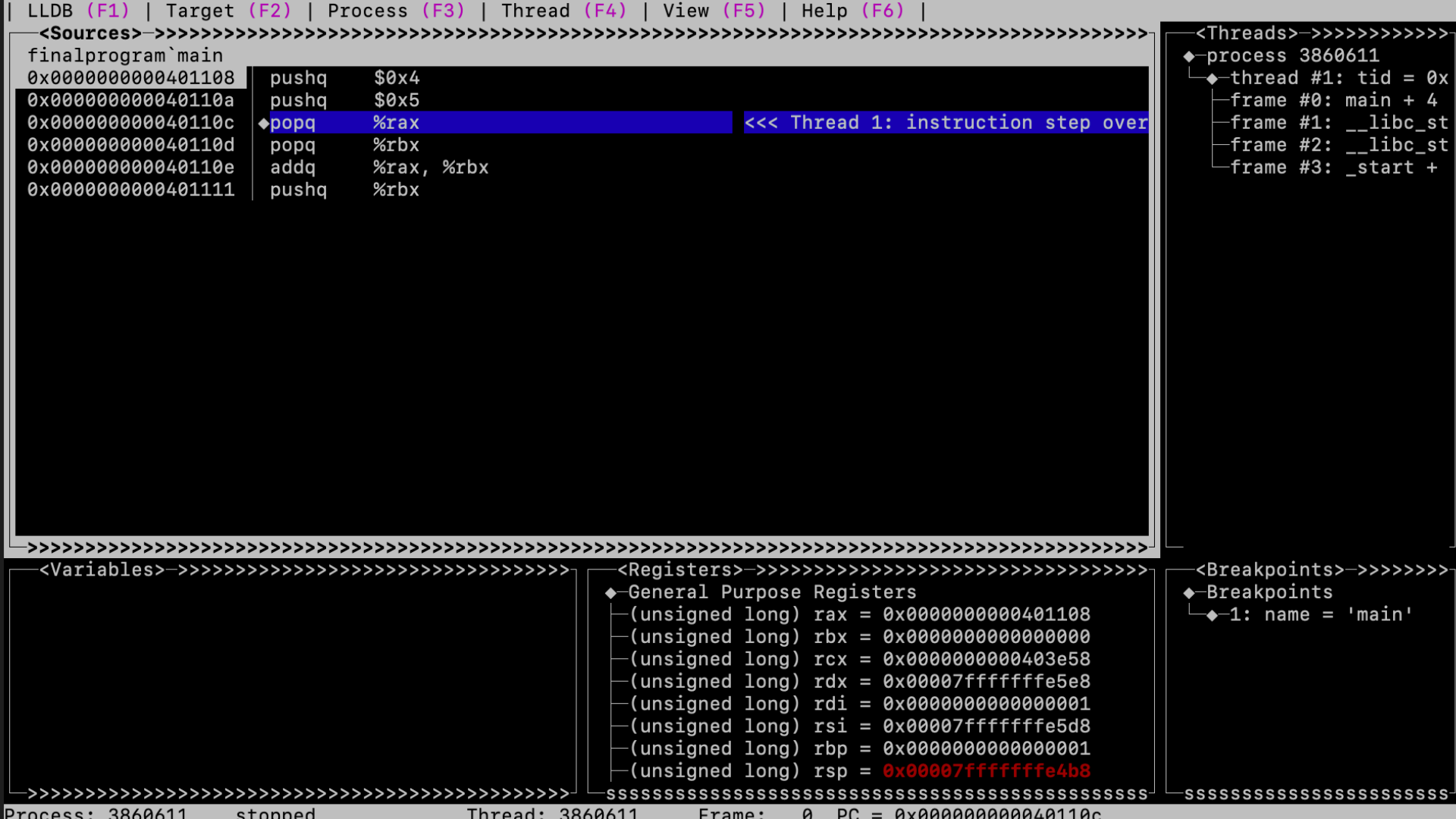

| LLDB (F1) | Target (F2) | Process (F3) | Thread (F4) | View (F5) | Help (F6) |
<Sources>
finalprogram`main
0x0000000000401108 ◆pushq $0x4 <<< Thread 1: breakpoint 1.1
0x000000000040110a pushq $0x5
0x000000000040110c popq %rax
0x000000000040110d popq %rbx
0x000000000040110e addq %rax, %rbx
0x0000000000401111 pushq %rbx
<Threads>
◆process 3839550
◆thread #1: ti
frame #0: mai
frame #1: __l
frame #2: __l
frame #3: _st
<Variables>
<Registers>
◆General Purpose Registers
◆Floating Point Registers
◆Advanced Vector Extensions
<Breakpoints>
◆Breakpoints
◆1: name = 'ma
Process: 3839550 stopped Thread: 3839550 Frame: 0 PC = 0x0000000000401108

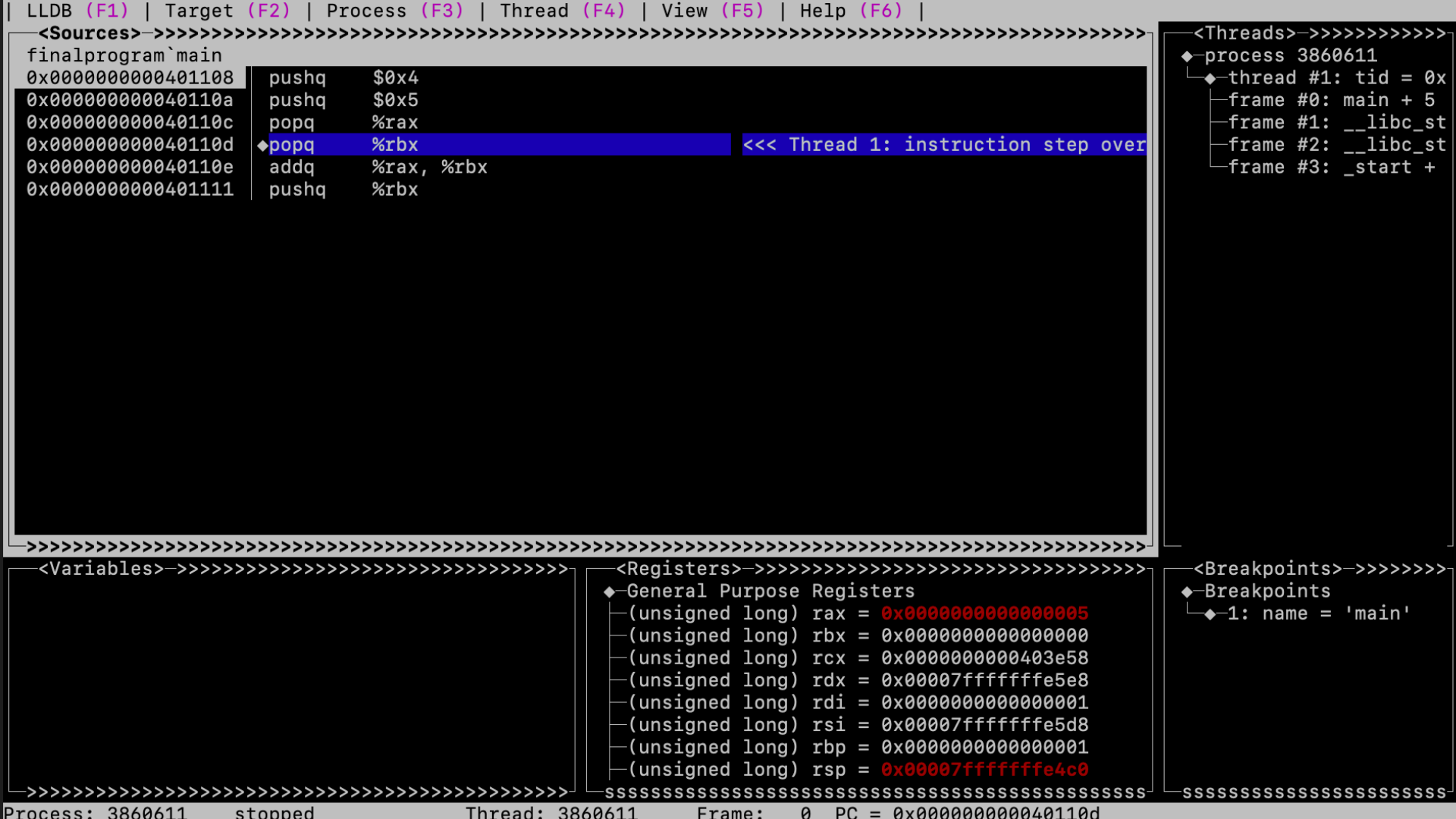

LLDB (F1) | Target (F2) | Process (F3) | Thread (F4) | View (F5) | Help (F6)
<Sources>
finalprogram`main
0x0000000000401108 pushq $0x4 <<< Thread 1: breakpoint 1.1
0x000000000040110a pushq $0x5
0x000000000040110c popq %rax
0x000000000040110d popq %rbx
0x000000000040110e addq %rax, %rbx
0x0000000000401111 pushq %rbx
Tab to navigate between panels and right arrow to expand an option. I have expanded the general registers.
<Threads>
process 3839550
thread #1: ti
frame #0: mai
frame #1: __l
frame #2: __l
frame #3: _st
<Variables>
<Registers>
General Purpose Registers
(unsigned long) rax = 0x00000000004
(unsigned long) rbx = 0x00000000000
(unsigned long) rcx = 0x00000000004
(unsigned long) rdx = 0x00007fffffff
(unsigned long) rdi = 0x00000000000
<Breakpoints>
Breakpoints
1: name = 'ma
Process: 3839550 stopped
Thread: 3839550 Frame: 0 PC = 0x0000000000401108

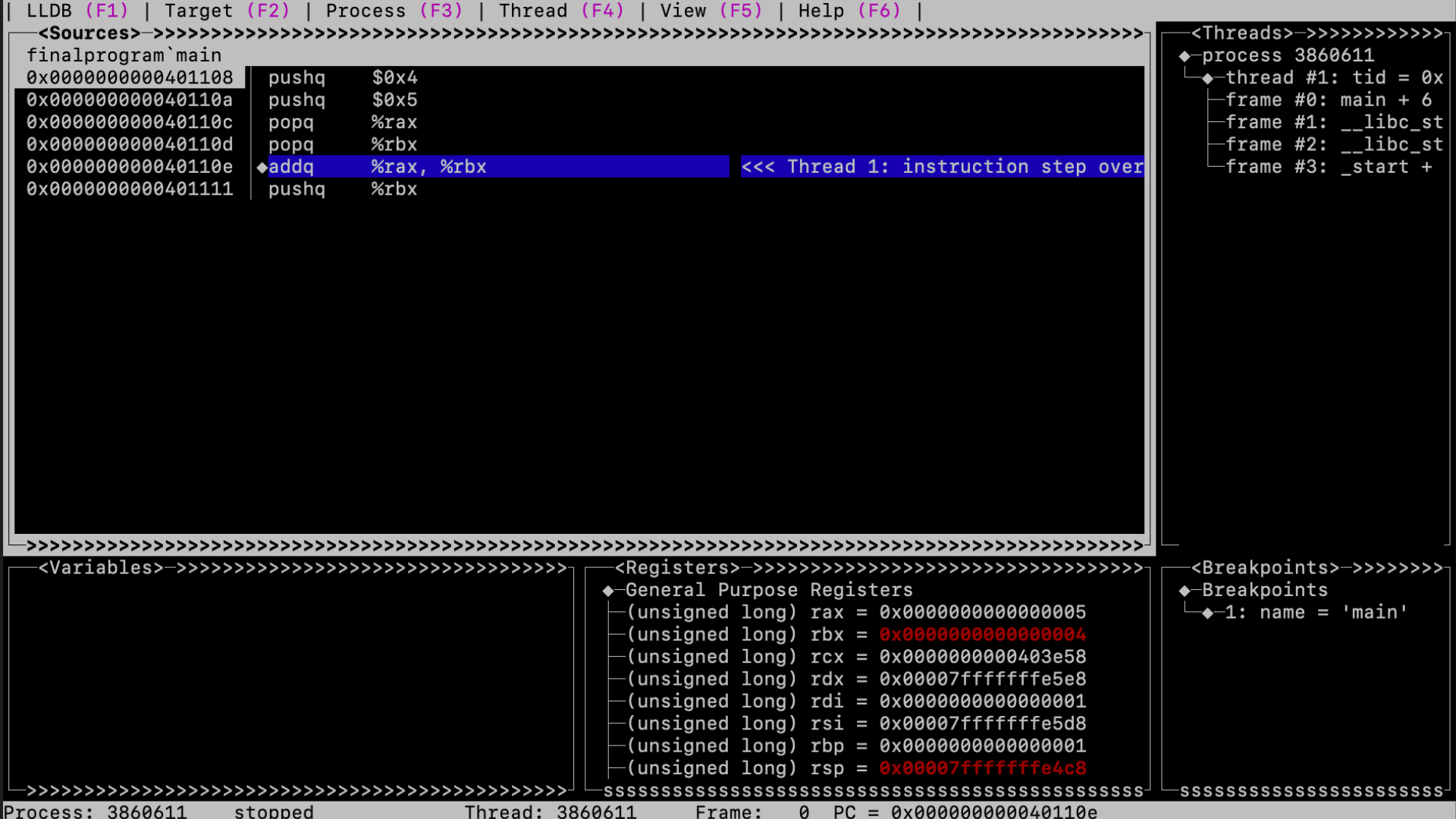

LLDB (F1) | Target (F2) | Process (F3) | Thread (F4) | View (F5) | Help (F6)
<Sources>
finalpro
0x000000
0x000000
0x000000
0x000000
0x000000
0x000000
<Help>
return - Run to selected line with one shot breakpoint
up - Select previous source line
down - Select next source line
left - Scroll to the left
right - Scroll to the right
page-up - Page up
page-down - Page down
b - Set breakpoint on selected source/disassembly line
c - Continue process
D - Detach with process suspended
h - Show help dialog
n - Step over (source line)
N - Step over (single instruction)
[Use arrows to scroll, any other key to exit]
<Threads>
process 3839550
thread #1: ti
frame #0: mai
frame #1: __l
frame #2: __l
frame #3: _st
<Variables>
H for help option
And ESC to close window
<Registers>
General Purpose Registers
(unsigned long) rax = 0x00000000004
(unsigned long) rbx = 0x00000000000
(unsigned long) rcx = 0x00000000004
(unsigned long) rdx = 0x00007fffffff
(unsigned long) rdi = 0x00000000000
<Breakpoints>
Breakpoints
1: name = 'ma
Process: 3839550 stopped
Thread: 3839550
Frame: 0
PC = 0x0000000000401108

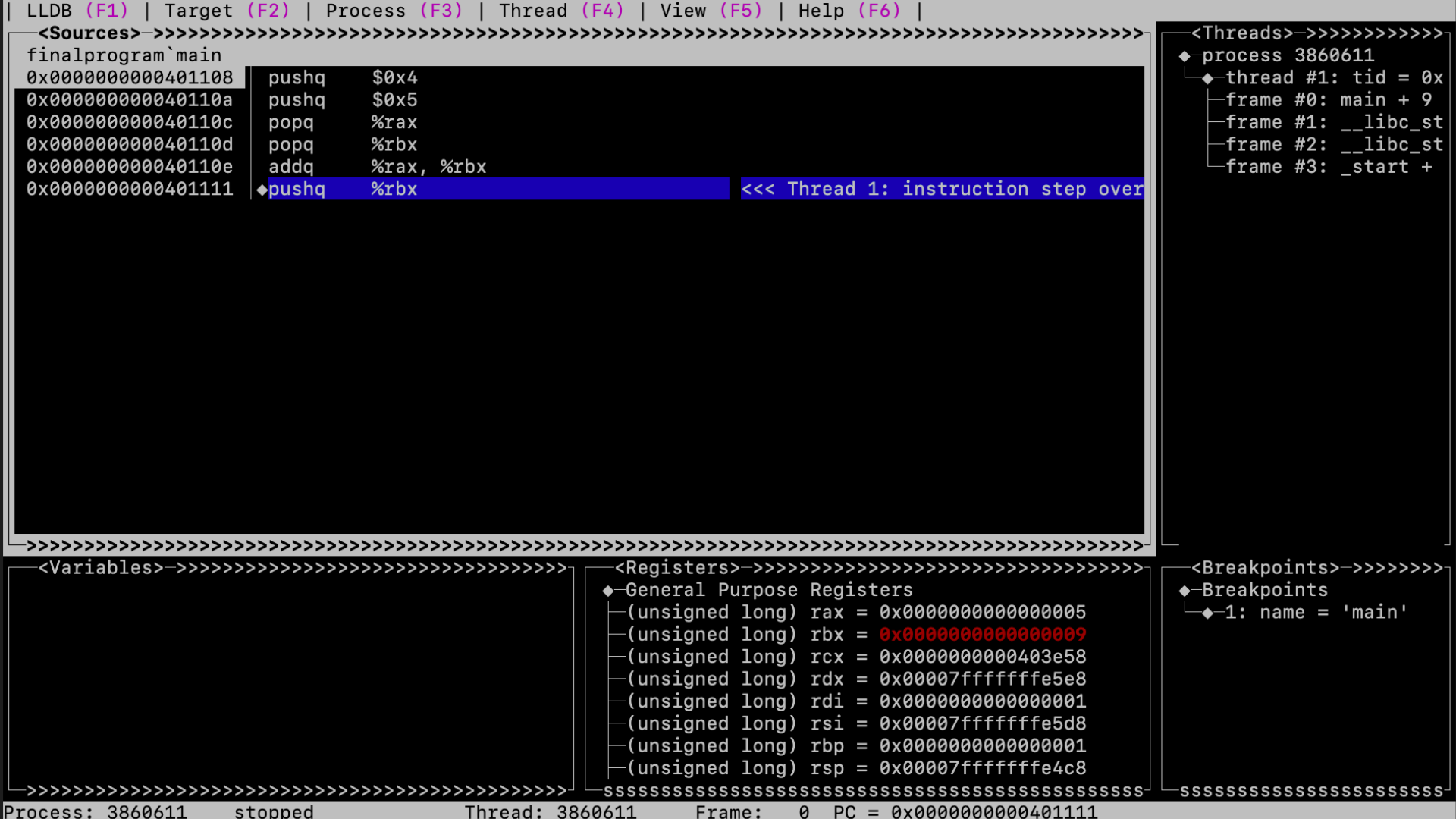

| LLDB (F1) | Target (F2) | Process (F3) | Thread (F4) | View (F5) | Help (F6) |
<Sources>
finalprogram`main
0x0000000000401108 pushq $0x4 <<< Thread 1: breakpoint 1.1
0x000000000040110a pushq $0x5
0x000000000040110c popq %rax
0x000000000040110d popq %rbx
0x000000000040110e addq %rax, %rbx
0x0000000000401111 pushq %rbx
<Threads>
process 3860611
thread #1: tid = 0x
frame #0: main
frame #1: __libc_st
frame #2: __libc_st
frame #3: _start +
<Variables>
<Registers>
General Purpose Registers
(unsigned long) rax = 0x0000000000401108
(unsigned long) rbx = 0x0000000000000000
(unsigned long) rcx = 0x0000000000403e58
(unsigned long) rdx = 0x00007fffffffe5e8
(unsigned long) rdi = 0x0000000000000001
(unsigned long) rsi = 0x00007fffffffe5d8
(unsigned long) rbp = 0x0000000000000001
(unsigned long) rsp = 0x00007fffffffe4c8
<Breakpoints>
Breakpoints
1: name = 'main'
Process: 3860611 stopped Thread: 3860611 Frame: 0 PC = 0x0000000000401108


| LLDB (F1) | Target (F2) | Process (F3) | Thread (F4) | View (F5) | Help (F6) |
<Sources>
finalprogram`main
0x0000000000401108 pushq $0x4
0x000000000040110a pushq $0x5 <<< Thread 1: instruction step over
0x000000000040110c popq %rax
0x000000000040110d popq %rbx
0x000000000040110e addq %rax, %rbx
0x0000000000401111 pushq %rbx
<Threads>
process 3860611
thread #1: tid = 0x
frame #0: main + 2
frame #1: __libc_st
frame #2: __libc_st
frame #3: _start +
<Variables>
<Registers>
General Purpose Registers
(unsigned long) rax = 0x0000000000401108
(unsigned long) rbx = 0x0000000000000000
(unsigned long) rcx = 0x0000000000403e58
(unsigned long) rdx = 0x00007fffffffe5e8
(unsigned long) rdi = 0x0000000000000001
(unsigned long) rsi = 0x00007fffffffe5d8
(unsigned long) rbp = 0x0000000000000001
(unsigned long) rsp = 0x00007fffffffe4c0
<Breakpoints>
Breakpoints
1: name = 'main'
Process: 3860611 stopped Thread: 3860611 Frame: 0 PC = 0x000000000040110a


| LLDB (F1) | Target (F2) | Process (F3) | Thread (F4) | View (F5) | Help (F6) |
<Sources>
finalprogram`main
0x0000000000401108 pushq $0x4
0x000000000040110a pushq $0x5
0x000000000040110c popq %rax
<<< Thread 1: instruction step over
0x000000000040110d popq %rbx
0x000000000040110e addq %rax, %rbx
0x0000000000401111 pushq %rbx
<Threads>
process 3860611
thread #1: tid = 0x
frame #0: main + 4
frame #1: __libc_st
frame #2: __libc_st
frame #3: _start +
<Variables>
<Registers>
General Purpose Registers
(unsigned long) rax = 0x0000000000401108
(unsigned long) rbx = 0x0000000000000000
(unsigned long) rcx = 0x0000000000403e58
(unsigned long) rdx = 0x00007fffffffe5e8
(unsigned long) rdi = 0x0000000000000001
(unsigned long) rsi = 0x00007fffffffe5d8
(unsigned long) rbp = 0x0000000000000001
(unsigned long) rsp = 0x00007fffffffe4b8
<Breakpoints>
Breakpoints
1: name = 'main'
Process: 3860611 stopped Thread: 3860611 Frame: 0 PC = 0x000000000040110c


| LLDB (F1) | Target (F2) | Process (F3) | Thread (F4) | View (F5) | Help (F6) |
<Sources>
finalprogram`main
0x0000000000401108 pushq $0x4
0x000000000040110a pushq $0x5
0x000000000040110c popq %rax
0x000000000040110d popq %rbx <<< Thread 1: instruction step over
0x000000000040110e addq %rax, %rbx
0x0000000000401111 pushq %rbx
<Threads>
process 3860611
thread #1: tid = 0x
frame #0: main + 5
frame #1: __libc_st
frame #2: __libc_st
frame #3: _start +
<Variables>
<Registers>
General Purpose Registers
(unsigned long) rax = 0x0000000000000005
(unsigned long) rbx = 0x0000000000000000
(unsigned long) rcx = 0x0000000000403e58
(unsigned long) rdx = 0x00007fffffffe5e8
(unsigned long) rdi = 0x0000000000000001
(unsigned long) rsi = 0x00007fffffffe5d8
(unsigned long) rbp = 0x0000000000000001
(unsigned long) rsp = 0x00007fffffffe4c0
<Breakpoints>
Breakpoints
1: name = 'main'
Process: 3860611 stopped Thread: 3860611 Frame: 0 PC = 0x000000000040110d

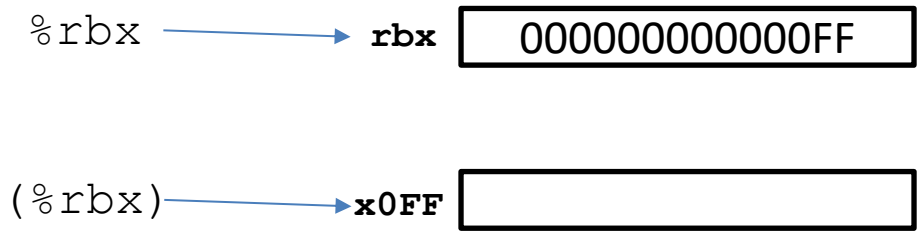

| LLDB (F1) | Target (F2) | Process (F3) | Thread (F4) | View (F5) | Help (F6) |
<Sources>
finalprogram`main
0x0000000000401108 pushq $0x4
0x000000000040110a pushq $0x5
0x000000000040110c popq %rax
0x000000000040110d popq %rbx
0x000000000040110e addq %rax, %rbx <<< Thread 1: instruction step over
0x0000000000401111 pushq %rbx
<Threads>
process 3860611
thread #1: tid = 0x
frame #0: main + 6
frame #1: __libc_st
frame #2: __libc_st
frame #3: _start +
<Variables>
<Registers>
General Purpose Registers
(unsigned long) rax = 0x0000000000000005
(unsigned long) rbx = 0x0000000000000004
(unsigned long) rcx = 0x0000000000403e58
(unsigned long) rdx = 0x00007fffffffe5e8
(unsigned long) rdi = 0x0000000000000001
(unsigned long) rsi = 0x00007fffffffe5d8
(unsigned long) rbp = 0x0000000000000001
(unsigned long) rsp = 0x00007fffffffe4c8
<Breakpoints>
Breakpoints
1: name = 'main'
Process: 3860611 stopped Thread: 3860611 Frame: 0 PC = 0x000000000040110e


| LLDB (F1) | Target (F2) | Process (F3) | Thread (F4) | View (F5) | Help (F6) |
<Sources>
finalprogram`main
0x0000000000401108 pushq $0x4
0x000000000040110a pushq $0x5
0x000000000040110c popq %rax
0x000000000040110d popq %rbx
0x000000000040110e addq %rax, %rbx
0x0000000000401111 pushq %rbx
<<< Thread 1: instruction step over
<Threads>
process 3860611
thread #1: tid = 0x
frame #0: main + 9
frame #1: __libc_st
frame #2: __libc_st
frame #3: _start +
<Variables>
<Registers>
General Purpose Registers
(unsigned long) rax = 0x0000000000000005
(unsigned long) rbx = 0x0000000000000009
(unsigned long) rcx = 0x0000000000403e58
(unsigned long) rdx = 0x00007fffffffe5e8
(unsigned long) rdi = 0x0000000000000001
(unsigned long) rsi = 0x00007fffffffe5d8
(unsigned long) rbp = 0x0000000000000001
(unsigned long) rsp = 0x00007fffffffe4c8
<Breakpoints>
Breakpoints
1: name = 'main'
Process: 3860611 stopped Thread: 3860611 Frame: 0 PC = 0x0000000000401111


| LLDB (F1) | Target (F2) | Process (F3) | Thread (F4) | View (F5) | Help (F6) |
<Sources>
finalprogram
0x0000000000401108 pushq $0x4
0x000000000040110a pushq $0x5
0x000000000040110c popq %rax
0x000000000040110d popq %rbx
0x000000000040110e addq %rax, %rbx
0x0000000000401111 pushq %rbx
<Threads>
process 3860611
thread #1: tid = 0x
frame #0:
<Variables>
<Registers>
General Purpose Registers
Floating Point Registers
Advanced Vector Extensions
<Breakpoints>
Breakpoints
1: name = 'main'
Process: 3860611 stopped Thread: 3860611 Frame: 0 PC = 0x0000000000401112

NEXT TIME

- () s represent value in memory



COMPUTED ADDRESS

Base + (Index * Scale) + Displacement

disp(base, index, scale)

AT&T Syntax	Pseudo code
100(%rbx, %rcx, 4)	memory[rbx+rcx*4 + 100]
100(%rbx)	memory[rbx + 100]
100(%rbx,8)	memory[rbx * 8 + 100]
100(,%rbx,8)	memory[rbx * 8 + 100]
100(%rbx,%rcx)	memory[rbx+rcx+100]
100	memory[100]

COMPUTED ADDRESSES