CSC 204 intro Lab

1. In your Linux home directory create a subdirectory where you will create your code.
2. Change into this directory
3. In a text editor such as vi create the following:

.global main

.text

main: #main function

movq $20, %rax #move immediate value 20 into rax reg

movq $x, %rbx #move pointer to rbx reg

movq x, %rcx #move value x into rcx \*x

movq $60, %rax #exit(

movq $0, %rdi #EXIT\_SUCCESS

syscall #);

.data

x: .quad 5 #allocates 8 bytes of memory and places 5 in the location

Name the file “lab.s”

4) Compile the code using the following command: “gcc -g -static -o lab lab.s”

This will create an executable named “lab” with all content in static (no external libraries) and it will also generate debugging symbols and tables.

5) type “objdump -f lab”

What are some of the entries listed? Note the starting addresses.

File Format

Architecture

Executable Privileges

Start Address: 0x0000000000400a30

6) Type “readelf -h lab” What is some of the information present?

Endian type

File type (executable)

Machine AMD x86-64

Entry point address

Start of headers

Flags

Header sizes

7) Type “readelf -S lab” can you find some segments (sections) displayed that we mentioned in class?

Sections such as:

.text – Text section

.rodata – Read-only-data

.data – Data section

.bss

8) From the above question. Look at the bottom of the output.

Key to Flags:

W (write), A (alloc), X (execute), M (merge), S (strings), I (info),

L (link order), O (extra OS processing required), G (group), T (TLS),

C (compressed), x (unknown), o (OS specific), E (exclude),

l (large), p (processor specific)

What are some of the permissions for the: text, data, rodata .

.text – AX – allocate, execute

.rodata – A – allocate only

.data – WA – write, execute

.bss – WA – write, execute

9) In the above question. What is the address of the text section? Is it the same as our entry point address? If not is it higher or lower.?

The address of .text is 00000000004004d0.

The address of text is higher, this is because there is extra instructions added by the compiler that precedes our initial instructions in .text.

10) Type “objdump -d lab” and look for the text section. Does it start at the same address as the segment address?

Yes,

Disassembly of section .text:

00000000004004d0 <backtrace\_and\_maps.constprop.1>:

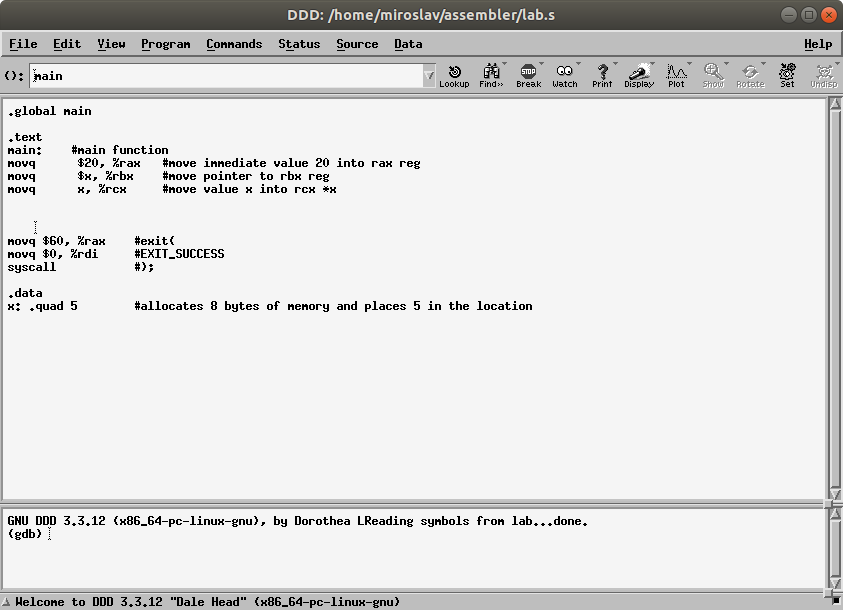
4004d0: c3 retq

11) Was there code inserted by the compiler before our entry point?

The compiler inserted a .\_nit section and .plt section before our .text

BASIC DEBUGGING

12) type “ddd lab”

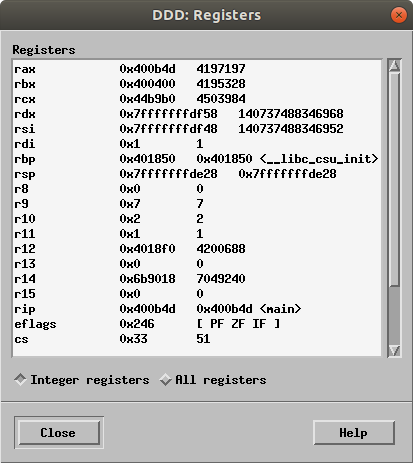


Click on main and click breakpoint on menu.



Click Run.

Go to status → registers.



Click “step” and trace the program.

13) what were some of the changes to the registers?

The value 20 is placed in the rax register

The pointer for the variable x is placed in the rbx register

The value of the x variable, 5, is placed in the rcx register.

Then, to close, the immidiate value 60 is placed in rax

the immidiate value 0 is placed in rdi

syscall is made.