

CSCI 2330 – x86-64 GDB Exercises

1. What GDB command (just one) should you use for each of the following situations when debugging an assembly program (without the source code)?

- (a) You are paused on `callq foo`, and you want to execute the entire function and then pause after returning.
- (b) You are paused on `callq foo`, and you want to execute **into** the function and then pause execution again.
- (c) You accidentally stepped into a call to `malloc` and want to return to the calling function (i.e., back into your own code).
- (d) You want to know what calling `foo(20)` would return (assume the program isn't about to make that call on its own).
- (e) You are at a breakpoint within a loop and want to run the next loop iteration (you can assume there is only the one breakpoint set).

2. Write a single GDB "**x**" command ("examine memory") to do each of the following (you must use the **x** command, not `print`):

- (a) Print a 4-byte int stored in memory at address `%rax`, in decimal.
- (b) Print an 8-byte int stored in memory at address `%rax`, in hex.
- (c) Print a string stored in memory at address `%rax`.
- (d) Print a string stored in memory at address `0x123456`.
- (e) Print an array of 5 chars starting at address `%rax`, showing their decimal values.
- (f) Print an array of 5 chars starting at address `%rax`, showing their textual values.
- (g) Print an array of 5 pointers starting at address `%rax`.

3. Suppose you're trying to reverse-engineer the x86-64 code snippet below. When executing the program within GDB, what GDB command should you run to try to understand what this code is doing?

```
subq    $0x18,%rsp
leaq    0x8(%rsp),%rcx
leaq    0xc(%rsp),%rdx
movq    $0x4f8ab1,%rsi
movq    $0xdf35c0,%rdi
movl    $0x0,%eax
callq   isoc99_sscanf@plt
```