10. Write a program that uses the `execve` system call to start a shell, meaning that it executes the program `/bin/sh`.

11. What does it mean for a program in Unix/Linux to have its suid bit set?

12. Take the program from problem #10, and make it suid root. Run the program. Do you have a root shell? Can you explain why?

13. Modify the program from #10 by using the `setreuid` syscall so that when run suid it returns a root shell. Why would this ability be useful to an attacker?

14. Write a program that uses `execve` to print out the current date and time, using the command `/bin/bash -c date`

15. Write a program that contains the numbers 7, 4, 11, and 16 in the `.data` section. The code should use `push` commands to put this data on the stack; it should then use `pop` commands to put these numbers in `eax, ebx, cx` and `dl` respectively.

   Run your program in a debugger. Break it after the last number is put on the stack
   - What is the value of `esp`?
   - Where on the stack are the values stored?

   Continue the program, and break it again after all of the values have been pushed to the registers.
   - Is the data still on the stack? Explain.
   - What is the value of `esp`?
   - What are the values of `ecx` and `edx`. Explain.

16. Modify the code for `larg.s` presented in class to use the `loop` and/or `jcxz` opcodes.

17. Write a program that
   - Contains two unsigned integers in the `.data` section,
   - Pushes these two numbers onto the stack,
   - Reads them from the stack,
   - Adds them, and
   - Uses the result as the exit code from the program.

   What happens if the numbers are negative? What happens if the addition overflows?

18. Repeat the previous, but return the product.

19. Repeat the previous, but return the quotient; ignore the remainder.