1. Understanding the Objective

→ Recognized that the lab requires passing a number to a procedure, which must check if it's odd or even and print the result to the terminal.
→ Recalled from lecture that arguments are passed via the stack, and printing is done using Linux syscalls.

2. Designing the Program Structure

→ Decided on using three main sections:

- .data to hold the strings for "odd" and "even"
- .text for _start (entry point) and the custom _check_even_odd procedure
- Use of section .bss was not necessary

→ Mapped out that the number would be:

- pushed to the stack
- accessed via [ebp+8] in the procedure
- checked using bitwise AND with 1 to determine odd/even.

3. Implementing the Procedure Logic

→ Set up standard function prologue with push ebp and mov ebp, esp.
→ Retrieved the passed value using [ebp + 8].
→ Checked odd/even with:

- and eax, 1
- cmp eax, 0
- je print_even
-

→ Used mov eax, 4 and int 0x80 to print the right message.

4. Exiting the Program

→ After printing, returned to _start using ret.
→ Exited program cleanly with syscall exit (eax=1, ebx=0, int 0x80).

5. Testing and Validation

→ Hardcoded different numbers into eax before pushing.
→ Verified correct terminal output for both even and odd numbers.
→ Ensured stack and procedure cleanup with leave and ret.