My thought process for all 3 tasks!

Task 1: EBX Counter Loop

Objective: Use EBX as a counter from 0 to 9 and explain what happens.

1. Start Design
   - Recalled from the lecture that loops can be made with inc, cmp, and jl.
   - Decided to use ebx to store the counter instead of ecx.
2. Write Code
   - Initialize ebx to 0.
   - Create a label to loop back to.
   - Use inc ebx, cmp ebx, 10, and jl label for the loop.
   - End with exit syscall.
3. Assemble and Run
   - nasm -f elf32 -g -F dwarf counter_ebx.asm -o counter_ebx.o
   - ld -m elf_i386 counter_ebx.o -o counter_ebx
   - ./counter_ebx
4. Debug
   - Launch gdb ./counter_ebx
   - Set breakpoint at _start
   - Use display $ebx, stepi, and info registers to trace the loop.
   - Verified that EBX goes from 0 to 10.
5. Conclusion
   - EBX is successfully used as a manual counter.
   - Verified correct behavior using GDB.

Task 2: Sum First 10 Fibonacci Numbers

Objective: Calculate Fibonacci sequence (0 to 9) and store final sum (55).

1. Plan the Logic
   - Need two registers to store current and previous Fibonacci numbers.
   - Use a loop to update the sum.
   - Use a counter to repeat 10 times.
2. Write Code
   - Initialize: eax = 0 (first), ebx = 1 (second), ecx = 8 (loop 8 more times).
   - Use edx to hold the sum.
   - In the loop:
     - Add eax and ebx → store in esi
     - Update eax = ebx, ebx = esi
     - Add eax to edx
     - Decrement ecx, loop if not zero
   - Exit syscall.
3. Assemble and Run
   - Compile with NASM and link.
   - Run and check result with echo $? → should return 55.
4. Debug
   - Use gdb, break at start or near end.
   - Use display $eax, $ebx, $edx, step through.
   - Verified Fibonacci values and sum building up to 55.
5. Conclusion
   - Correct sum of 10 Fibonacci numbers achieved.
   - Confirmed with both terminal and GDB.

Task 3: Find Largest Value in an Array

Objective: Use a 3-element array and find the maximum value using a loop.

1. Understand the Lecture
   - Arrays are placed in memory sequentially.
   - Elements can be accessed using [esi], and pointer moved by add esi, 4.
2. Write Code
   - Define array: dd 12, 7, 25
   - Set esi to array base.
   - Move first value into eax as the starting "largest".
   - Loop through next 2 elements using ecx as counter.
   - In loop:
     - Load current element into ebx
     - Compare with eax
     - If greater, move ebx into eax
     - Move esi to next element (add esi, 4)
     - dec ecx, jnz loop
   - Move largest (in eax) to ebx and exit.
3. Assemble and Run
   - Compile with nasm -f elf32 -g -F dwarf array_max.asm -o array_max.o
   - Link and run → echo $? should return 25
4. Debug with GDB
   - Break at label done
   - info registers → check eax or ebx for largest value
   - Use x/3dw 0x804a000 to view array in memory (address from .data section)
   - Confirm 12, 7, 25 are the values and 25 is correctly identified
5. Conclusion
   - Code works, largest element (25) identified using comparisons and pointer arithmetic.
   - Successfully verified using GDB and visual inspection of memory.