Shift, rotation, bit manipulation and bit masking:

1. Write an assembly program to **multiply AL by 2** using a shift instruction (no MUL instruction allowed).

2. Write an assembly program that:

- Takes a value in BL
- Performs a **logical right shift** by 1
- Stores the result in CL

3. Write an assembly program to **check if the most significant bit (MSB)** of a number is 1.

- Hint: Use SHL or ROL and check the **Carry Flag (CF)**.

4. Suppose AL = 10101100b.

- Perform RCR AL, 2 and determine the final value of AL and CF. (clear the carry flag first)

5. Write a program to **reverse all bits** in an 8-bit number using rotation instructions.

6. Check if the number is even or odd, using shifting or rotation.

7. **Set the 3rd bit (from right) of AL to 1**
Example: AL = 00100100b → 00101100b
(Hint: use OR with bit mask 00000100b)

8. **Toggle (flip) the 2nd bit of AL**
Example: AL = 10011000b → 10011100b
(Hint: use XOR with 00000100b)

9. Pack two 4-bit numbers (nibbles) into a single 8-bit register.
Example: AH = 00000101b (5), AL = 00001010b (10) → BL = 01011010b
(Hint: Shift AH left 4 bits and OR with AL)

10. **Swap the nibbles of AL**
Example: `AL = 10110010b → 00101011b`

11. **Write an assembly program to count how many bits are set to 1** in AL.

15. **Extract lower nibble (last 4 bits).**
From AL, extract only the lower 4 bits and store in BL.
👉 *Hint:* `AND BL, 00001111b`

2D Array:

12. **Sum of all elements in a 2D array.**
Given a 3×3 matrix, find the **sum of all elements** and store the result in a variable SUM.

13. For a 3×3 matrix, find the **sum of each column** and store in a new array.

14. Given two 3×3 matrices A and B, add them element-wise and store the result in matrix C.