**ALP to add two 8 bit numbers (Memory location 2050 and 2051) and store in 3050.**

START: LDA 2050 ; Load the first number from memory location 2050 into register A

MOV B, A ; Move the contents of register A to register B (for later use)

LDA 2051 ; Load the second number from memory location 2051 into register A

ADD B ; Add the contents of register B to register A (A = A + B)

STA 3050 ; Store the result (sum) in memory location 3050

HLT ; Stop the program

ALP to add two numbers, (20 and 30)

START: MVI A, 14H ; Load the immediate value 20 (14H in hexadecimal) into register A

MVI B, 1EH ; Load the immediate value 30 (1EH in hexadecimal) into register B

ADD B ; Add the value in register B to register A (A = A + B)

STA 000 ; Store the result from register A into memory location 3050H

HLT ; Halt the program

ALP to subtract two numbers, (30 and 20)

START: MVI A, 14H ; Load the immediate value 20 (14H in hexadecimal) into register A

MVI B, 1EH ; Load the immediate value 30 (1EH in hexadecimal) into register B

SUB B ; Add the value in register B to register A (A = A + B)

STA 000 ; Store the result from register A into memory location 3050H

HLT ; Halt the program

**Write an assembly language program for 8 bit microprocessor to divide 8 bit data stored in memory location 8050 by 8 bit data stored in 8051 and store the quotient in 8052 and remainder in 8053.**

LDA 8050H ; Load the dividend into the accumulator (A)

MOV B, A ; Move the dividend into register B (Dividend stored in B)

LDA 8051H ; Load the divisor into the accumulator (A)

MOV C, A ; Move the divisor into register C (Divisor stored in C)

; Initialize the quotient and remainder

MVI D, 00H ; Clear D register (D will hold the quotient)

MOV A, B ; Move dividend from B to A for remainder calculation

DIV\_LOOP: ; Division loop label

CMP C ; Compare A (dividend) with C (divisor)

JC DIV\_END ; If A < C (dividend < divisor), jump to end (division complete)

SUB C ; Subtract divisor (C) from A (A = A - C)

INR D ; Increment the quotient stored in D

STA 8053H ; Store the remainder at memory location 8053H

MOV A, D ; Move quotient (content of register D) to A

STA 8052H ; Store the remainder at memory location 8052H

HLT ; Terminate the program

JMP DIV\_LOOP ; Repeat the loop

DIV\_END: ; Division end label

MOV A, D ; Move the quotient from D to A

STA 8052H ; Store the quotient at memory location 8052H

MOV A, B ; Move remainder (content of register B) to A

STA 8053H ; Store the remainder at memory location 8053H

HLT ; Terminate the program

**ALP to multiply two 8 bit numbers.**

MVI B, 05H ; Load the multiplicand into register B (Multiplicand = 05H)

MVI C, 03H ; Load the multiplier into register C (Multiplier = 03H)

MVI A, 00H ; Clear register A (Product = 0)

MVI D, 00H ; Clear register D (Carry = 0)

MULTIPLY\_LOOP:

MOV E, B ; Move multiplicand to register E

ADD A ; Add multiplicand (E) to A

JNC NO\_CARRY ; If no carry, jump to NO\_CARRY

INX D ; Increment D if there is a carry (D = D + 1)

NO\_CARRY:

DCR C ; Decrement multiplier (C = C - 1)

JNZ MULTIPLY\_LOOP ; Repeat until multiplier becomes zero

; At the end, register A will contain the lower byte of the product

; and register D will contain the upper byte (carry, if any).

HLT ; Halt the program

**Write down the output:**

MVI A, AAH ; Load the value AAH into register A

MOV B, A ; Copy the value in A to register B

RRC ; Rotate the accumulator right through the carry

XRA B ; XOR the value of A with the value in B

OUT PORT1 ; Output the result in A to the output port (PORT1)

HLT ; Halt the program

**The output is FF (11111111).**

1. **MVI A, AAH**:
   * The **immediate value** AAH (binary 10101010) is loaded into register A.
   * Now, A = AAH.
2. **MOV B, A**:
   * The value in register A (which is AAH) is copied into register B.
   * Now, B = AAH.
3. **RRC (Rotate Right through Carry)**:
   * The RRC instruction rotates the 8 bits of the accumulator (register A) **right by one position**, and the rightmost bit (bit 0) is copied to both the carry flag and the leftmost bit (bit 7).
   * Original value of A = 10101010B (AAH).
   * After rotation, the new value of A will be 01010101B (55H).
   * **Carry flag** will now hold the value of the bit that was rotated out (bit 0), which was 0.
4. **XRA B (Exclusive OR A with B)**:
   * The **XOR (exclusive OR)** operation is performed between the values in register A and B.
   * Register A = 55H (after the RRC).
   * Register B = AAH.
   * XOR operation between 55H (01010101B) and AAH (10101010B):
     + 01010101B XOR 10101010B = 11111111B (FFH).
   * Now, A = FFH.
5. **OUT PORT1**:
   * The value in register A (now FFH) is sent to the output device or port specified by PORT1.
6. **HLT (Halt)**:
   * The program halts and stops executing.