Task 01

Take input in AX and move to BX.

```
MOV AX, 5     ; Store 5 in AX
MOV BX, AX    ; Copy AX to BX
```

Task 02

Swap two numbers using 3 registers.

```
MOV AX, 5      ; First number
MOV BX, 10     ; Second number
MOV CX, AX     ; Temporary store AX
MOV AX, BX
MOV BX, CX
```

Task 03

Add two numbers using registers.

```
MOV AX, 3
MOV BX, 4
ADD AX, BX     ; AX = AX + BX = 7
```

Task 04

Subtract two numbers using registers.

```
MOV AX, 3
MOV BX, 5
SUB AX, BX     ; Result will be negative, stored in 2's complement
```

Note: for neg value options,

1. NEG AX ; Now AX = +ve
2. SUB AX, BX
3. JS NEGATIVE_RESULT ; Jump if sign flag is set (i.e., result is negative)

Task 05 Swap two numbers using ADD and SUB only.

```
MOV AX, 5
MOV BX, 10
ADD AX, BX     ; AX = 15
SUB BX, AX     ; BX = -5 → actually 10 - 15 = -5
NEG BX         ; BX = 5
SUB AX, BX     ; AX = 10
```

Task 06

.DATA

A DW ? ; Declare variable A

B DW ? ; Declare variable B

C DW ? ; Declare variable C

```
; 1. A = B - A
MOV AL, B
SUB AL, A
MOV A, AL

; 2. A = -(A + 1)
INC A
NEG A

; 3. C = A + (B + 1)
MOV AL, B
INC AL
ADD AL, A
MOV C, AL

; 4. A = B − (A − 1)
DEC A
MOV AL, B
SUB AL, A
MOV A, AL
```

Task 07

Multiply and Divide X, Y, Z

1. X * Y

2. X / Y

3. X * Y / Z

```
.DATA
X DW ?    ; Declare variable X
Y DW ?    ; Declare variable Y
Z DW ?    ; Declare variable Z
RESULT DW ?  ; Variable for storing the result

.CODE
MOV AX, X   ; Load X into AX
MUL Y       ; Multiply AX by Y, result is in DX:AX
MOV RESULT, AX  ; Store lower 16-bit result in RESULT

MOV AX, X   ; Load X into AX
DIV Y       ; Divide AX by Y, quotient goes to AL, remainder to AH
MOV RESULT, AX  ; Store quotient in RESULT

MOV AX, X   ; Load X into AX
MUL Y       ; Multiply AX by Y, result is in DX:AX
DIV Z       ; Divide DX:AX by Z, result goes to AX
MOV RESULT, AX  ; Store quotient in RESULT
```

Task 08

1. 36DF * AF
2. F4D5 / C9A5
3. CA92 * BAF9
4. C2A2 * ABCD / BED

```
.DATA
RESULT DW ?  ; Variable to store results

.CODE
MOV AX, 36DFh    ; Load first number
MUL AFh          ; Multiply AX by AF, result stored in DX:AX
MOV RESULT, AX   ; Store lower 16-bit result in RESULT  ; (36DF * AF)

MOV AX, F4D5h    ; Load dividend into AX
MOV DX, 0000h    ; Clear DX for accurate division
DIV C9A5h        ; Divide AX by C9A5h
MOV RESULT, AX   ; Store quotient in RESULT           ; (F4D5 / C9A5)

MOV AX, CA92h    ; Load first number
MUL BAF9h        ; Multiply AX by BAF9h, result stored in DX:AX
MOV RESULT, AX   ; Store lower 16-bit result in RESULT  ;(CA92 * BAF9)

MOV AX, C2A2h    ; Load first number
MUL ABCDh        ; Multiply AX by ABCDh, result stored in DX:AX
DIV BEDh         ; Divide DX:AX by BEDh
MOV RESULT, AX   ; Store quotient in RESULT      ;(C2A2 * ABCD / BED)
```

Task 09

MOV instruction register combinations:

```
; 1. mov AX, BX
; 2. mov BX, CX
; 3. mov DX, AX
; 4. mov CX, DX
```

Task 10

ADD/SUB register combinations:

```
; ADD examples
ADD AX, BX
ADD CX, DX

; SUB examples
SUB AX, CX
SUB DX, BX
```

Task 11

(1 + 2) * (3 - 1) / 5 + 3 + 2 - (1 * 2)

```
MOV AL, 1
ADD AL, 2       ; AL = 3
MOV BL, 3
SUB BL, 1       ; BL = 2
MUL BL          ; AX = 6
MOV BL, 5
DIV BL          ; AL = 1 (quotient)
ADD AL, 3       ; AL = 4
ADD AL, 2       ; AL = 6
MOV BL, 1
MUL BL          ; AX = 6
SUB AL, 2       ; Final answer = 4
```