2 a). Perform 16-bit Addition operation.

```
MOV AX, 1234H     ; Load first 16-bit number
MOV BX, 1111H     ; Load second 16-bit number
ADD AX, BX        ; Add BX to AX
; Result in AX
HLT
```

2 b). Perform 16-bit Subtraction operation.

```
MOV AX, 4567H     ; Minuend
MOV BX, 1234H     ; Subtrahend
SUB AX, BX        ; AX = AX - BX
; Result in AX
HLT
```

3 a). Perform 16-bit Multiplication operation

```
MOV AX, 0005H     ; Load lower 16-bit multiplicand
MOV BX, 0003H     ; Load multiplier
MUL BX            ; AX * BX → result in DX:AX
; For small values, result fits in AX
HLT
```

3 b). Perform 16-bit Division operation

```
MOV AX, 0010H     ; Dividend (16-bit)
MOV BX, 0004H     ; Divisor
```

```
DIV BL          ; Divide AX by BL (result in AL, remainder in AH)
HLT
```

4. Calculate the length of a string.

```
MOV SI, OFFSET STR
MOV CX, 0

NEXT_CHAR:
CMP BYTE PTR [SI], '$'    ; End of string
JE END
INC CX
INC SI
JMP NEXT_CHAR

END:
; Length in CX
HLT

STR DB 'HELLO$', 0
```

5. Find the sum of the numbers in a word array.

```
MOV SI, OFFSET ARR
MOV CX, 05        ; Number of elements
MOV AX, 0000      ; Clear AX for sum
```

```
SUM_LOOP:
ADD AX, [SI]      ; Add word at SI to AX
ADD SI, 2         ; Move to next word (2 bytes)
LOOP SUM_LOOP

; Result in AX
HLT

ARR DW 1000H, 2000H, 3000H, 4000H, 5000H
```

6. Sorting number in ascending order of an unsorting array

```
MOV CX, 05        ; Number of elements
DEC CX            ; Passes = n-1
MOV SI, OFFSET NUMS

OUTER:
MOV BX, CX
MOV SI, OFFSET NUMS

INNER:
MOV AX, [SI]
CMP AX, [SI+2]
JBE SKIP
```

```
XCHG AX, [SI+2]
MOV [SI], AX

SKIP:
ADD SI, 2
DEC BX
JNZ INNER

LOOP OUTER

HLT

NUMS DW 4321H, 1234H, 5678H, 1111H, 1000H
```

7. Move a byte String from source to destination.

```
LEA SI, SRC
LEA DI, DEST
MOV CX, 05

REP MOVSB     ; Moves CX bytes from DS:SI to ES:DI
HLT

SRC DB 'HELLO'
DEST DB 5 DUP (?)
```