Task1

```
.MODEL SMALL
.STACK 100H

.DATA
msg DB 'Please insert a character: $'
char DB ?

.CODE
MAIN PROC
    MOV AX, @DATA
    MOV DS, AX

    ; Display message
    LEA DX, msg
    MOV AH, 09H
    INT 21H

    ; Take character input
    MOV AH, 01H
    INT 21H
    MOV char, AL   ; Store the character

    ; Move to new line
    MOV AH, 02H
    MOV DL, 0AH
    INT 21H
    MOV DL, 0DH
    INT 21H

    ; Display the character
    MOV DL, char
    MOV AH, 02H
    INT 21H

    ; Exit
    MOV AX, 4C00H
    INT 21H
MAIN ENDP
END MAIN
```

Task2

```
.MODEL SMALL
.STACK 100H

.DATA
msg1 DB 'Enter first digit: $'
msg2 DB 0DH, 0AH, 'Enter second digit: $'
msg3 DB 0DH, 0AH, 'Sum: $'
msg4 DB 0DH, 0AH, 'Difference: $'
msg5 DB 0DH, 0AH, 'Product: $'
msg6 DB 0DH, 0AH, 'Quotient: $'
num1 DB ?
num2 DB ?
res DB ?

.CODE
MAIN PROC
    MOV AX, @DATA
    MOV DS, AX

    ; Input first digit
    LEA DX, msg1
    MOV AH, 09H
    INT 21H

    MOV AH, 01H
    INT 21H
    SUB AL, '0'
    MOV num1, AL

    ; Input second digit
    LEA DX, msg2
    MOV AH, 09H
    INT 21H

    MOV AH, 01H
    INT 21H
    SUB AL, '0'
    MOV num2, AL

    ; ===== Addition =====
    MOV AL, num1
    ADD AL, num2
    ADD AL, '0'

    LEA DX, msg3
    MOV AH, 09H
    INT 21H
    MOV DL, AL
    MOV AH, 02H
    INT 21H

    ; ===== Subtraction =====
    MOV AL, num1
    SUB AL, num2
    ADD AL, '0'

    LEA DX, msg4
    MOV AH, 09H
    INT 21H
    MOV DL, AL
    MOV AH, 02H
    INT 21H

    ; ===== Multiplication =====
    MOV AL, num1
    MOV BL, num2
    MUL BL
    ADD AL, '0'

    LEA DX, msg5
    MOV AH, 09H
    INT 21H
    MOV DL, AL
    MOV AH, 02H
    INT 21H

    ; ===== Division =====
    MOV AL, num1
    MOV BL, num2
    XOR AH, AH
    DIV BL
    ADD AL, '0'

    LEA DX, msg6
    MOV AH, 09H
    INT 21H
    MOV DL, AL
    MOV AH, 02H
    INT 21H

    ; Exit
    MOV AX, 4C00H
    INT 21H
MAIN ENDP
END MAIN
```

Task3

[a]

```
.MODEL SMALL
.STACK 100H

.DATA
msg DB 'Enter a character: $'

.CODE
MAIN PROC
    MOV AX, @DATA
    MOV DS, AX

    ; Display message
    LEA DX, msg
    MOV AH, 09H
    INT 21H

    ; Read character
    MOV AH, 01H
    INT 21H
    MOV BL, AL        ; Store input character in BL

    ; Display space
    MOV DL, 20H
    MOV AH, 02H
    INT 21H

    ; Display character
    MOV DL, BL
    MOV AH, 02H
    INT 21H

    ; Exit
    MOV AX, 4C00H
    INT 21H
MAIN ENDP
END MAIN
```

[b]

```
.MODEL SMALL
.STACK 100H

.DATA
msg DB 'Enter an uppercase letter: $'

.CODE
MAIN PROC
    MOV AX, @DATA
    MOV DS, AX

    ; Display message
    LEA DX, msg
    MOV AH, 09H
    INT 21H

    ; Read uppercase letter
    MOV AH, 01H
    INT 21H
    MOV BL, AL

    ; Convert to lowercase (ASCII + 32)
    ADD BL, 32

    ; Display space
    MOV DL, 20H
    MOV AH, 02H
    INT 21H

    ; Display lowercase letter
    MOV DL, BL
    MOV AH, 02H
    INT 21H

    ; Exit
    MOV AX, 4C00H
    INT 21H
MAIN ENDP
END MAIN
```

Task4(same as 3b)

```
MOV AH, 01H       ; Read a character
INT 21H           ; Wait for user input (ASCII character to AL)

; Convert to lowercase: 'A' to 'Z' are 65 to 90, 'a' to 'z' are 97 to 122
ADD AL, 20H       ; Add 32 to convert uppercase to lowercase (ASCII difference)

MOV DL, AL        ; Move the lowercase letter to DL for output
MOV AH, 02H       ; Prepare to output a single character
INT 21H           ; Output the character

; Move to next line (Carriage return and Line feed)
MOV DL, 0DH       ; Carriage return (CR)
MOV AH, 02H       ; Function to output a single character
INT 21H

MOV DL, 0AH       ; Line feed (LF)
MOV AH, 02H
INT 21H
```

Task5

```
.model small
.stack 100h

.data
sum_msg db 'THE SUM OF ', 0
and_msg db ' AND ', 0
is_msg db ' IS $'

.code
main:
    ; Initialize the data segment
    mov ax, @data
    mov ds, ax

    ; Part (a) - Display a '?'
    mov ah, 02H          ; Function to display a character
    mov dl, '?'          ; ASCII value of '?'
    int 21H              ; Call interrupt to display '?'

    ; Part (b) - Read first digit
    mov ah, 01H          ; Function to read a character
    int 21H              ; Get user input (character in AL)
    sub al, '0'          ; Convert ASCII to decimal (subtract '0' ASCII value)
    mov bl, al           ; Store first digit in BL

    ; Read second digit
    mov ah, 01H          ; Function to read another character
    int 21H              ; Get user input (character in AL)
    sub al, '0'          ; Convert ASCII to decimal (subtract '0' ASCII value)
    mov cl, al           ; Store second digit in CL

    ; Calculate the sum
    add bl, cl          ; Add the first and second digits (sum in BL)

    ; Part (c) - Display the sum and digits
    ; Display "THE SUM OF"
    mov ah, 09H
    lea dx, sum_msg
    int 21H

    ; Display first digit
    add bl, '0'         ; Convert the first digit back to ASCII
    mov dl, bl
    mov ah, 02H
    int 21H

    ; Display " AND "
    lea dx, and_msg
    mov ah, 09H
    int 21H

    ; Display second digit
    mov dl, cl          ; Move second digit to DL
    add dl, '0'         ; Convert to ASCII
    mov ah, 02H
    int 21H

    ; Display " IS "
    lea dx, is_msg
    mov ah, 09H
    int 21H

    ; Display sum
    add bl, '0'         ; Convert sum back to ASCII
    mov dl, bl
    mov ah, 02H
    int 21H

    ; Exit the program
    mov ah, 4Ch         ; Exit to DOS
    int 21H
```

Task6

```
.MODEL SMALL
.STACK 100H
.DATA
MSG DB 'ENTER THREE INITIALS: $'
NEWLINE DB 0DH, 0AH, '$'
INITIALS DB 3 DUP('$')  ; Space for three initials

.CODE
MAIN PROC
    MOV AX, @DATA
    MOV DS, AX

    ; Display prompt message
    MOV DX, OFFSET MSG
    MOV AH, 09H
    INT 21H

    ; Read three initials
    MOV AH, 08H
    INT 21H
    MOV INITIALS, AL

    INT 21H
    MOV INITIALS+1, AL

    INT 21H
    MOV INITIALS+2, AL

    ; Print initials vertically
    MOV SI, OFFSET INITIALS
    MOV CX, 3

    PRINT_LOOP:
    MOV DL, [SI]   ; Load initial
    MOV AH, 02H    ; Print character
    INT 21H

    MOV DX, OFFSET NEWLINE   ; New line
    MOV AH, 09H
    INT 21H

    INC SI
    LOOP PRINT_LOOP

    MOV AH, 4CH   ; Terminate program
    INT 21H
MAIN ENDP
END MAIN
```

Task7

```
.MODEL SMALL
.STACK 100H
.DATA
MSG DB 'ENTER A HEX DIGIT: $'
DECMSG DB 'IN DECIMAL IT IS: $'
NEWLINE DB 0DH, 0AH, '$'

.CODE
MAIN PROC
    MOV AX, @DATA
    MOV DS, AX

    MOV DX, OFFSET MSG  ; Display prompt
    MOV AH, 09H
    INT 21H

    MOV AH, 08H  ; Read single hex digit
    INT 21H

    SUB AL, 'A'  ; Convert HEX A-F to decimal (10-15)
    ADD AL, 10

    MOV DL, AL
    MOV DX, OFFSET NEWLINE
    MOV AH, 09H
    INT 21H

    MOV DX, OFFSET DECMSG
    MOV AH, 09H
    INT 21H

    ADD DL, '0'  ; Convert numeric to ASCII
    MOV AH, 02H
    INT 21H

    MOV AH, 4CH
    INT 21H
MAIN ENDP
END MAIN
```

Task8

```
.MODEL SMALL
.STACK 100H
.DATA
BOX DB '**********', 0DH, 0AH, '**********', 0DH, 0AH, '**********', 0DH, 0AH, '**********', 0DH, 0AH
    DB '**********', 0DH, 0AH, '**********', 0DH, 0AH, '**********', 0DH, 0AH, '**********', 0DH, 0AH
    DB '**********', 0DH, 0AH, '$'

.CODE
MAIN PROC
    MOV AX, @DATA
    MOV DS, AX

    MOV DX, OFFSET BOX
    MOV AH, 09H
    INT 21H

    MOV AH, 4CH
    INT 21H
MAIN ENDP
END MAIN
```

Task9

```
.MODEL SMALL
.STACK 100H
.DATA
NUM1 DW 1234H   ; First multi-digit decimal number (example)
NUM2 DW 5678H   ; Second multi-digit decimal number (example)
RESULT DW ?

.CODE
MAIN PROC
    MOV AX, NUM1
    ADD AX, NUM2
    MOV RESULT, AX

    MOV AH, 4CH
    INT 21H
MAIN ENDP
END MAIN
```

Task 10: Status Flags for `ADD AL, BL` (80h + 80h)

When AL = 80h and BL = 80h, the sum is 100h, which exceeds an 8-bit register. The flags will be:

- Carry Flag (CF) = 1 → Overflow beyond 8-bit limit.
- Overflow Flag (OF) = 1 → Signed overflow occurs.
- Zero Flag (ZF) = 0 → Result is not zero.
- Sign Flag (SF) = 0 → Result is positive in unsigned context.
- Auxiliary Carry Flag (AF) = 1 → Carry from bit 3 to bit 4.

Task 11: Proving Carry into MSB but No Carry Out

If AX and BX contain positive numbers and ADD AX, BX results in a signed overflow (i.e., the sum is greater than 7FFFh for 16-bit signed numbers), then:

- Carry Flag (CF) = 0 → No carry out of MSB.
- Overflow Flag (OF) = 1 → Signed overflow occurs.
- MSB Carry-In Occurs (Internal Calculation).

Task 12: Proving Carry Out of MSB but No Carry Into MSB

If AX and BX contain negative numbers (e.g., both values are above 8000h), then after executing ADD AX, BX:

- Carry Flag (CF) = 1 → Carry out of MSB.
- Overflow Flag (OF) = 0 → No signed overflow (result stays negative).