```
; 8086 Assembly Programs with Explanations

;----------------------------------------
; 1. Addition of Two Numbers
;----------------------------------------

; (i) 8-bit addition - Immediate and Register addressing
MOV AL, 05H       ; Load immediate value 5 into AL
ADD AL, 03H       ; Add immediate value 3 to AL (AL = 08H)

; (ii) 16-bit addition - Immediate and Register addressing
MOV AX, 1234H     ; Load 1234H into AX
ADD AX, 1111H     ; AX = 2345H

; (b) Direct and Register addressing
MOV AL, [2000H]   ; Load value from memory 2000H into AL
ADD AL, BL        ; Add contents of BL to AL

MOV AX, [3000H]   ; Load 16-bit value from 3000H into AX
ADD AX, BX        ; Add BX to AX

; (c) Direct and Immediate addressing
MOV AL, [2000H]   ; Load from memory
ADD AL, 04H       ; Add immediate 4

MOV AX, [3000H]
ADD AX, 0A0AH     ; Add 0A0AH to AX

;----------------------------------------
; 2. Subtraction of Two Numbers
;----------------------------------------

; (i) 8-bit - Immediate and Register
MOV AL, 09H
SUB AL, 02H       ; AL = 07H

; (ii) 16-bit - Direct and Register
MOV AX, [3000H]
SUB AX, BX        ; AX = AX - BX

; (c) Direct and Immediate
MOV AX, [3000H]
SUB AX, 1234H     ; AX = AX - 1234H

;----------------------------------------
; 3. Multiplication
;----------------------------------------

; (i) 8-bit using repeated addition
MOV AL, 03H       ; Multiplicand
MOV BL, 04H       ; Multiplier
MOV CL, 00H       ; Result

L1: ADD CL, AL    ; Add AL to result
    DEC BL        ; Decrease counter
    JNZ L1

; (i) 8-bit using MUL
```

```
MOV AL, 05H
MOV BL, 04H
MUL BL            ; AX = AL * BL

; (ii) 16-bit using MUL
MOV AX, 0200H
MOV BX, 0100H
MUL BX            ; DX:AX = AX * BX

;----------------------------------------
; 4. Division
;----------------------------------------

; (i) 8-bit using repeated subtraction
MOV AL, 09H
MOV BL, 02H
MOV CL, 00H

L2: SUB AL, BL
    INC CL
    CMP AL, BL
    JAE L2         ; Continue if AL >= BL

; (i) 8-bit using DIV
MOV AL, 08H
MOV BL, 02H
DIV BL            ; AL = Quotient, AH = Remainder

; (ii) 16-bit using DIV
MOV AX, 0400H
MOV BX, 0200H
DIV BX

;----------------------------------------
; 5. Smallest/Largest in Array (8-bit)
;----------------------------------------
MOV SI, 1000H
MOV CL, 05H
MOV AL, [SI]
INC SI

L3: CMP AL, [SI]   ; Compare current AL with next value
    JAE SKIP1
    MOV AL, [SI]   ; AL = new max
SKIP1:
    INC SI
    DEC CL
    JNZ L3         ; Loop till end

; AL contains largest number

;----------------------------------------
; 6. Sorting Array (Bubble Sort)
;----------------------------------------
MOV CX, 09
LOOP1: MOV SI, 1000H
       MOV DX, 0
LOOP2: MOV AL, [SI]
```

```
       CMP AL, [SI+1]
       JBE SKIP2
       XCHG AL, [SI+1]
       MOV [SI], AL
SKIP2: INC SI
       INC DX
       CMP DX, CX
       JL LOOP2
       DEC CX
       JNZ LOOP1

;----------------------------------------
; 7. Copy String using String Instructions
;----------------------------------------
LEA SI, source
LEA DI, dest
MOV CX, 05
CLD
REP MOVSB         ; Byte copy

LEA SI, source
LEA DI, dest
MOV CX, 05
CLD
REP MOVSW         ; Word copy

;----------------------------------------
; 8. Count Even and Odd Numbers
;----------------------------------------
MOV CX, 05
MOV SI, 1000H
MOV DL, 0
MOV DH, 0

NEXT: MOV AL, [SI]
      TEST AL, 01
      JZ EVEN
      INC DH       ; Odd
      JMP SKIP3
EVEN: INC DL       ; Even
SKIP3: INC SI
       LOOP NEXT

;----------------------------------------
; 9. Display Name
;----------------------------------------
MOV AH, 09H
LEA DX, name
INT 21H
MOV AH, 4CH
INT 21H

name DB 'BUNNY$', 0

;----------------------------------------
; 10. Multiply Two 3x3 Matrices
;----------------------------------------
; Concept: Result[i][j] = sum of A[i][k]*B[k][j]
```

```
; Loop unrolling needed for nested iteration.
; Simplified structure below.

; (Pseudocode-level):
; Loop i=0 to 2
;    Loop j=0 to 2
;        AX=0
;        Loop k=0 to 2
;            AL = A[i][k]
;            BL = B[k][j]
;            MUL BL
;            ADD AX to Result[i][j]

;----------------------------------------
; 11. Use of Procedures and Macros
;----------------------------------------
ADD_TWO MACRO a, b
    MOV AL, a
    ADD AL, b
ENDM

PROC_ADD:
    MOV AL, 05H
    ADD AL, 03H
    RET

CALL PROC_ADD
ADD_TWO 04H, 05H

;----------------------------------------
; 12. Reverse Array
;----------------------------------------
MOV CX, 05
LEA SI, ARRAY1
LEA DI, ARRAY2
ADD DI, 4         ; Point DI to last

L4: MOV AL, [SI]
    MOV [DI], AL
    INC SI
    DEC DI
    LOOP L4

;----------------------------------------
```