2. 8085 program to load 32H and 48H in registers A & B, add them then display it.

MVI A, 32H

MVI B, 48H

ADD B

OUT 01H

HLT

3. 8085 program to load hexadecimal numbers in registers A & B, subtract them then display it.

MVI A, 23H

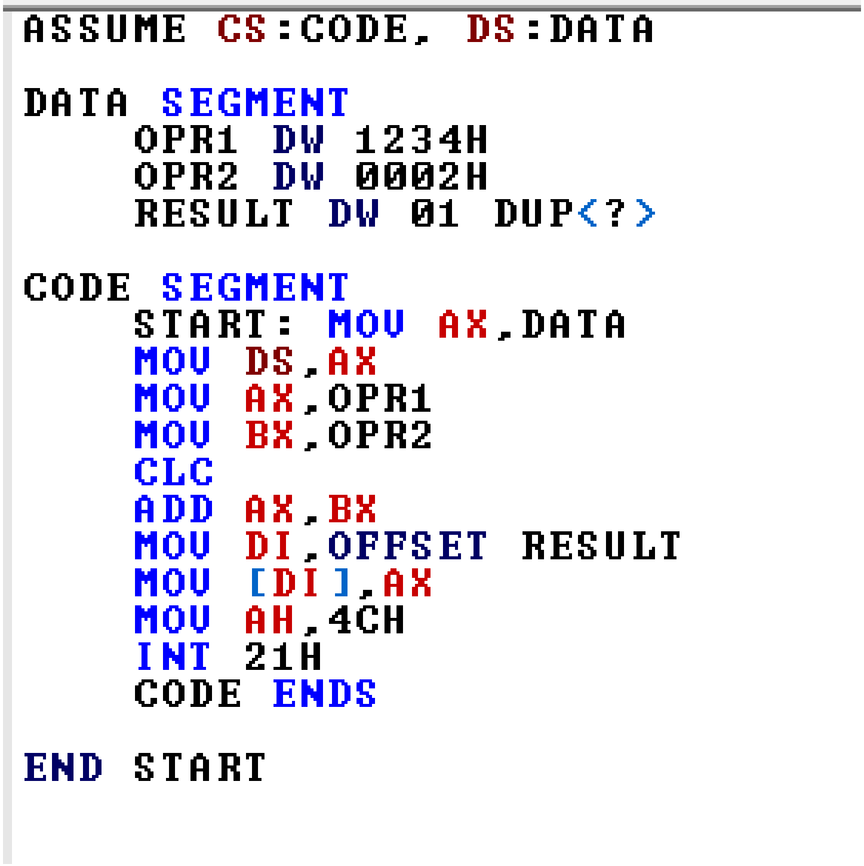
MVI B, 1AH

SUB B

OUT 01H

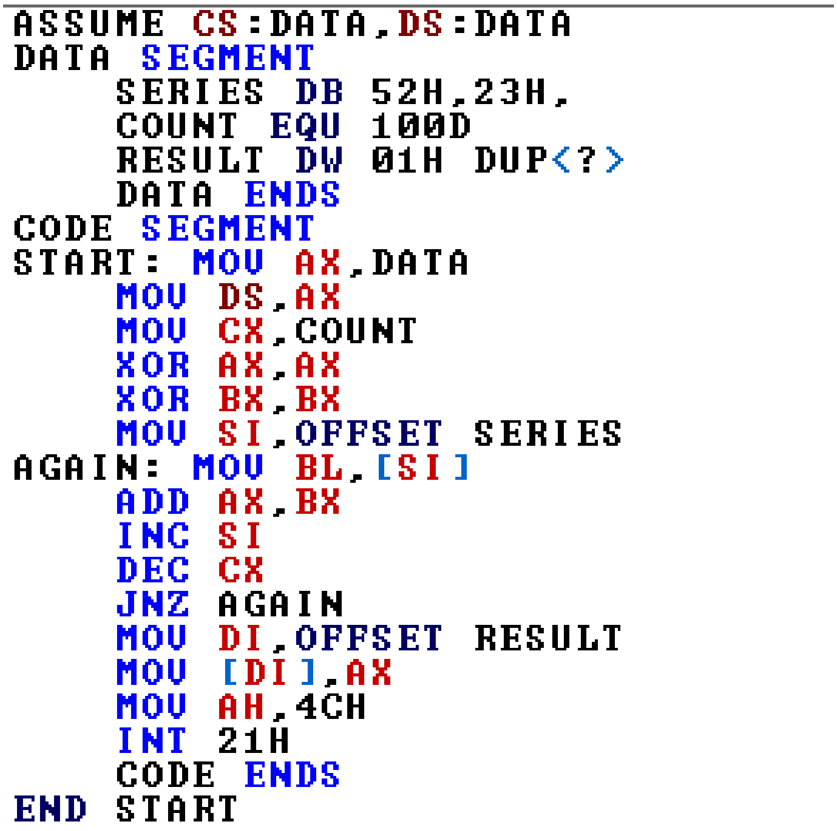
HLT

4. Addition of 2 numbers



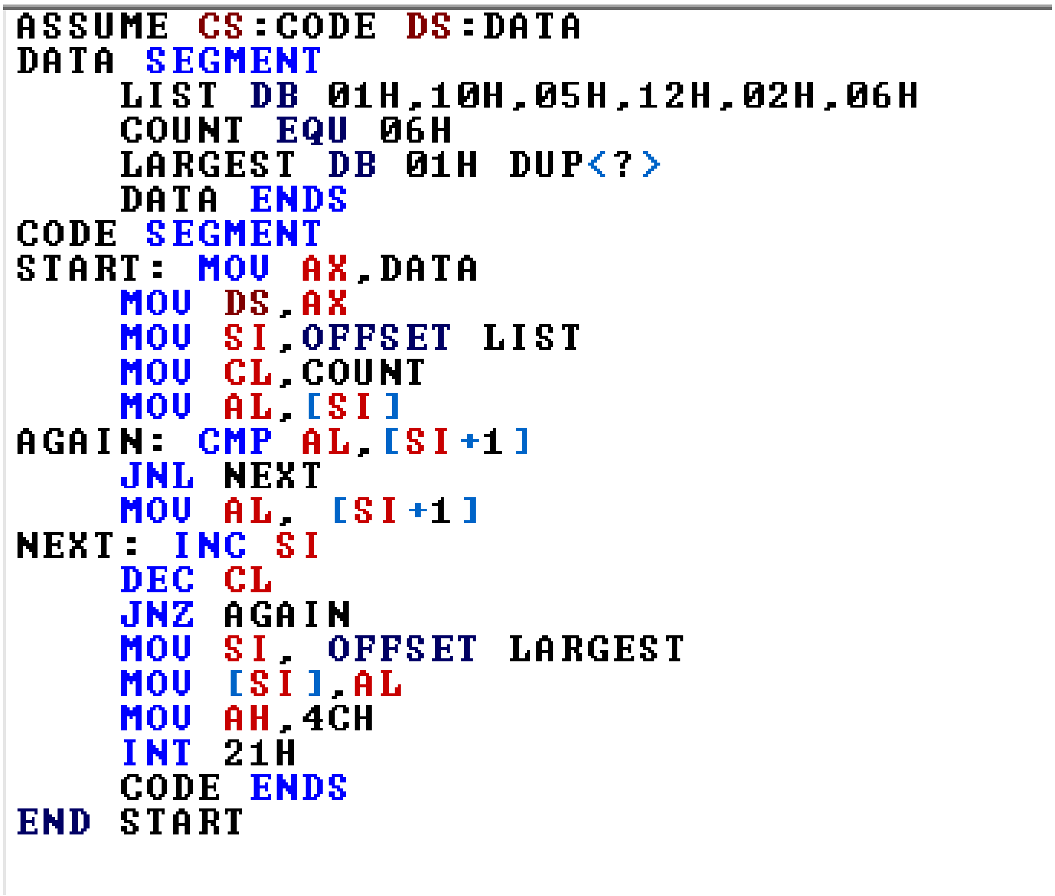
```
ASSUME CS:CODE, DS:DATA

DATA SEGMENT
    OPR1 DW 1234H
    OPR2 DW 0002H
    RESULT DW 01 DUP(?)

CODE SEGMENT
    START: MOV AX,DATA
    MOV DS,AX
    MOV AX,OPR1
    MOV BX,OPR2
    CLC
    ADD AX,BX
    MOV DI,OFFSET RESULT
    MOV [DI],AX
    MOV AH,4CH
    INT 21H
    CODE ENDS

END START
```

5. Add series of 8 bit numbers, series contains 100 numbers



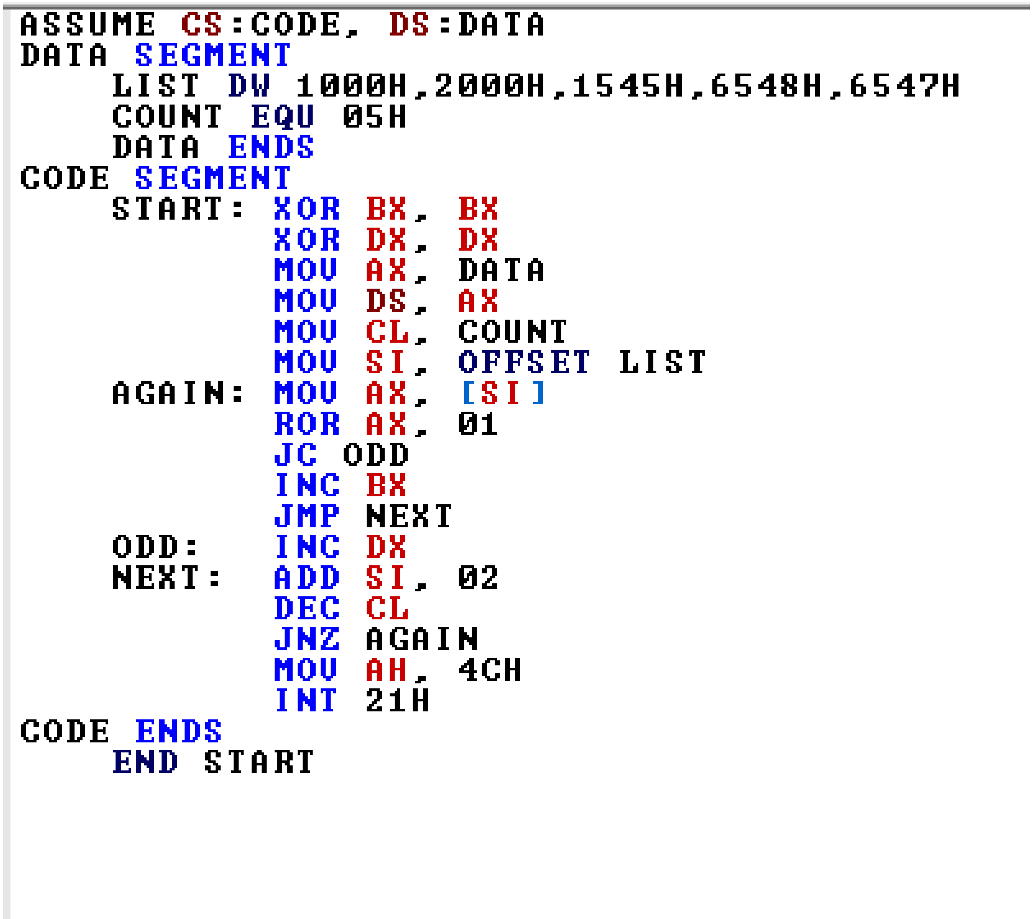
```
ASSUME CS:DATA,DS:DATA
DATA SEGMENT
    SERIES DB 52H,23H,
    COUNT EQU 100D
    RESULT DW 01H DUP(?)
    DATA ENDS
CODE SEGMENT
START: MOV AX,DATA
    MOV DS,AX
    MOV CX,COUNT
    XOR AX,AX
    XOR BX,BX
    MOV SI,OFFSET SERIES
AGAIN: MOV BL,[SI]
    ADD AX,BX
    INC SI
    DEC CX
    JNZ AGAIN
    MOV DI,OFFSET RESULT
    MOV [DI],AX
    MOV AH,4CH
    INT 21H
    CODE ENDS
END START
```

6. Find largest number from a given unordered array of 8 bit numbers



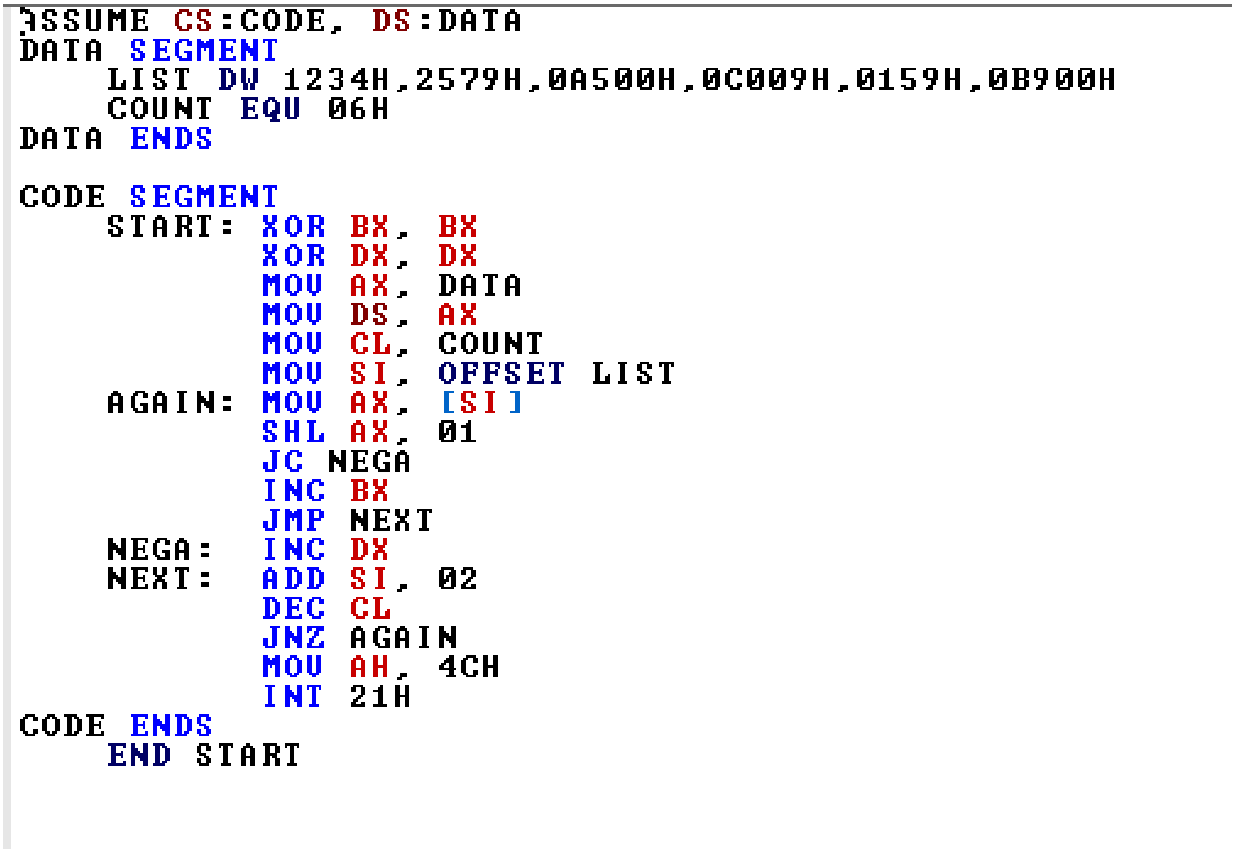
```
ASSUME CS:CODE DS:DATA
DATA SEGMENT
    LIST DB 01H,10H,05H,12H,02H,06H
    COUNT EQU 06H
    LARGEST DB 01H DUP(?)
    DATA ENDS
CODE SEGMENT
START: MOV AX,DATA
    MOV DS,AX
    MOV SI,OFFSET LIST
    MOV CL,COUNT
    MOV AL,[SI]
AGAIN: CMP AL,[SI+1]
    JNL NEXT
    MOV AL, [SI+1]
NEXT: INC SI
    DEC CL
    JNZ AGAIN
    MOV SI, OFFSET LARGEST
    MOV [SI],AL
    MOV AH,4CH
    INT 21H
    CODE ENDS
END START
```

7. Find number of even and odd numbers from a given unordered array of 8 bit numbers



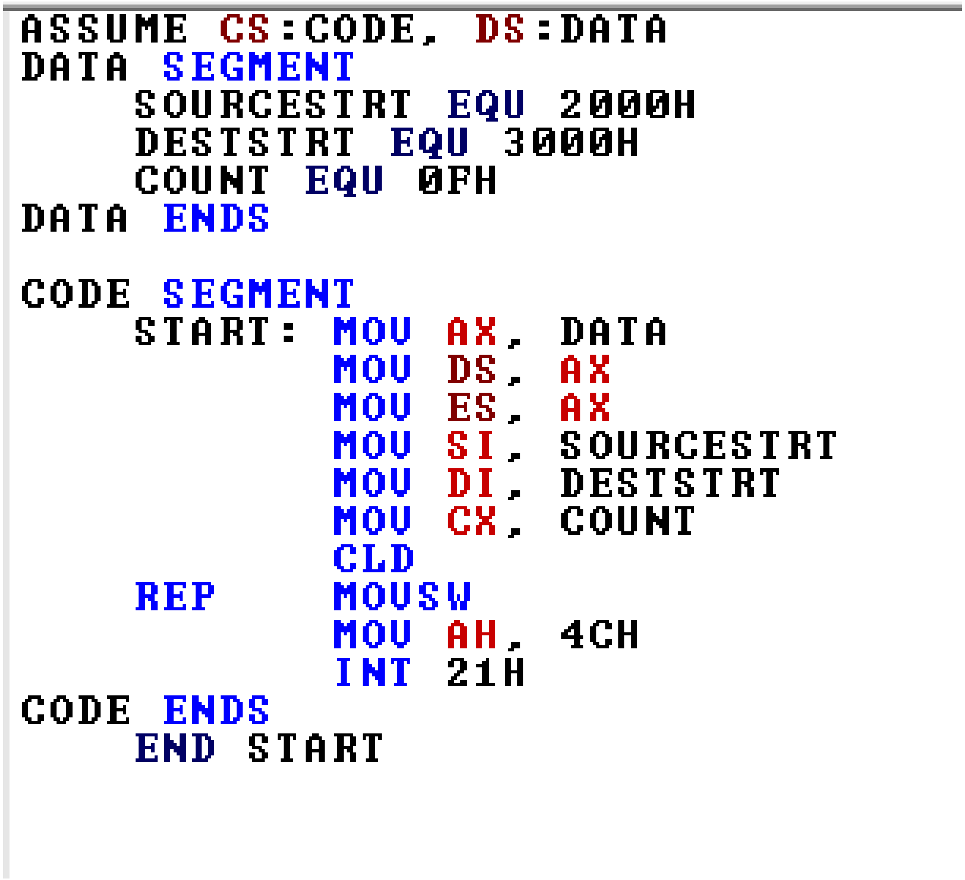
```
ASSUME CS:CODE, DS:DATA
DATA SEGMENT
    LIST DW 1000H,2000H,1545H,6548H,6547H
    COUNT EQU 05H
    DATA ENDS
CODE SEGMENT
    START: XOR BX, BX
           XOR DX, DX
           MOV AX, DATA
           MOV DS, AX
           MOV CL, COUNT
           MOV SI, OFFSET LIST
    AGAIN: MOV AX, [SI]
           ROR AX, 01
           JC ODD
           INC BX
           JMP NEXT
    ODD:   INC DX
    NEXT:  ADD SI, 02
           DEC CL
           JNZ AGAIN
           MOV AH, 4CH
           INT 21H
CODE ENDS
    END START
```

8. Find number of positive and negative numbers from a given series of signed numbers



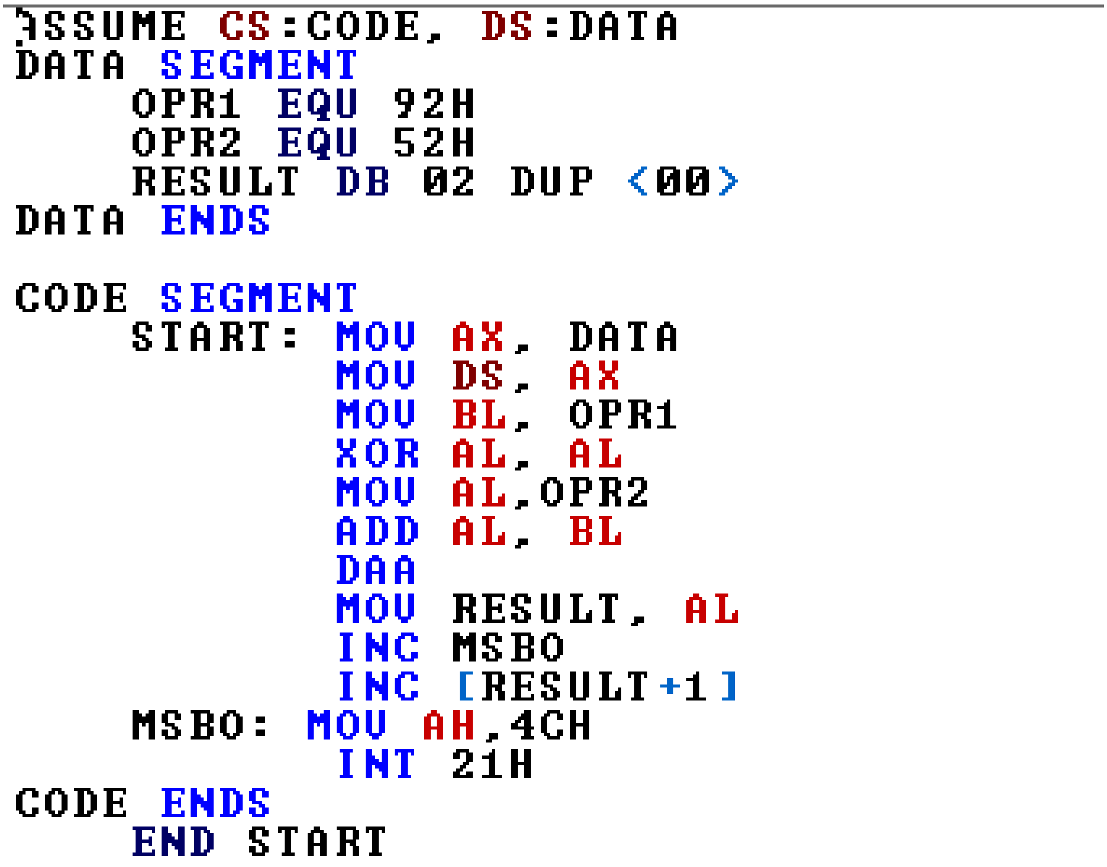
```
ASSUME CS:CODE, DS:DATA
DATA SEGMENT
    LIST DW 1234H,2579H,0A500H,0C009H,0159H,0B900H
    COUNT EQU 06H
DATA ENDS

CODE SEGMENT
    START: XOR BX, BX
           XOR DX, DX
           MOV AX, DATA
           MOV DS, AX
           MOV CL, COUNT
           MOV SI, OFFSET LIST
    AGAIN: MOV AX, [SI]
           SHL AX, 01
           JC NEGA
           INC BX
           JMP NEXT
    NEGA:  INC DX
    NEXT:  ADD SI, 02
           DEC CL
           JNZ AGAIN
           MOV AH, 4CH
           INT 21H
CODE ENDS
    END START
```

9. Move a string of data from offset 2000H to 3000H, length of string is 0FH.



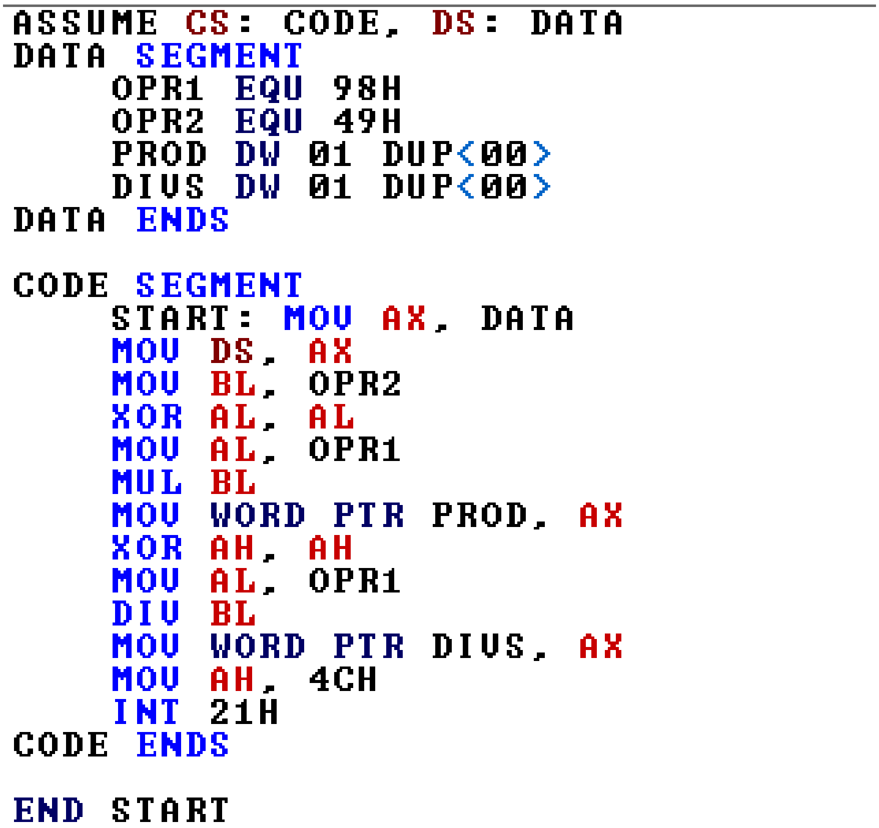
```
ASSUME CS:CODE, DS:DATA
DATA SEGMENT
    SOURCESTRT EQU 2000H
    DESTSTRT EQU 3000H
    COUNT EQU 0FH
DATA ENDS

CODE SEGMENT
    START: MOV AX, DATA
           MOV DS, AX
           MOV ES, AX
           MOV SI, SOURCESTRT
           MOV DI, DESTSTRT
           MOV CX, COUNT
           CLD
    REP    MOVSW
           MOV AH, 4CH
           INT 21H
CODE ENDS
    END START
```

10. Perform 1 byte BCD addition



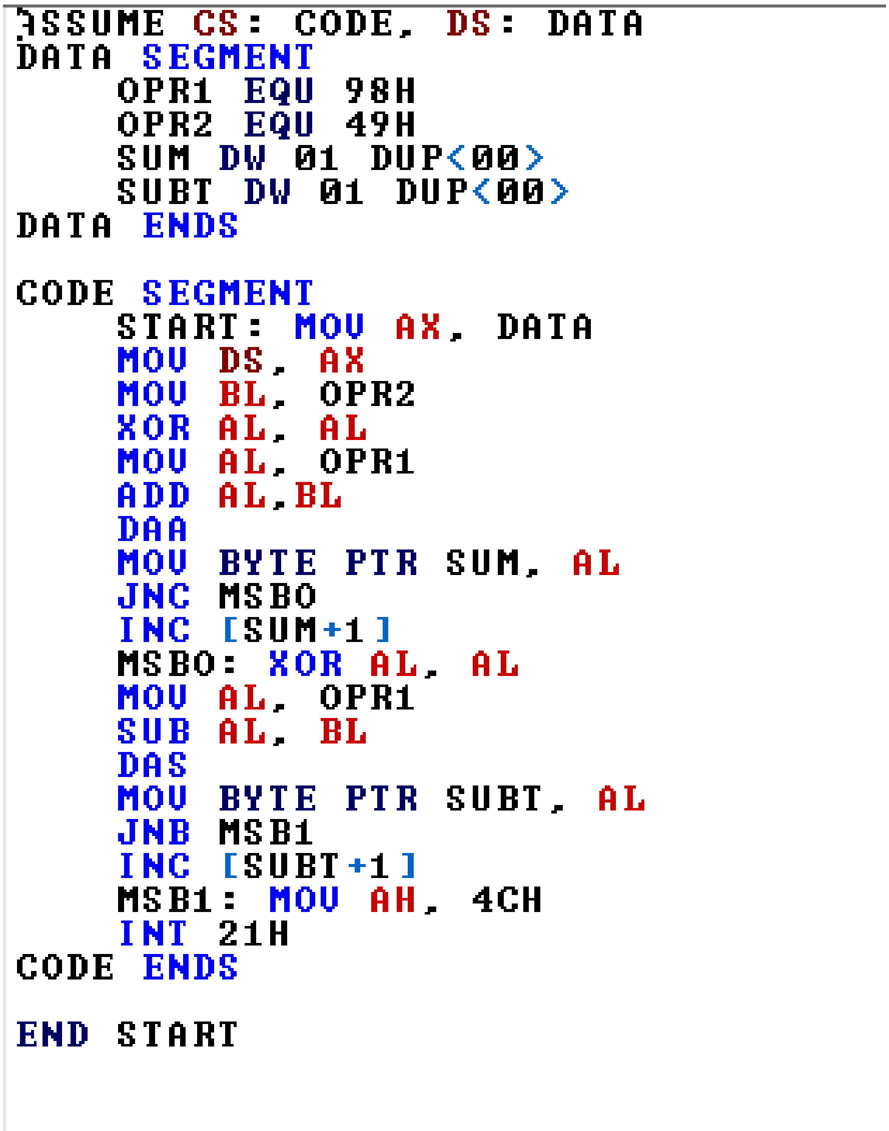
```
ASSUME CS:CODE, DS:DATA
DATA SEGMENT
    OPR1 EQU 92H
    OPR2 EQU 52H
    RESULT DB 02 DUP <00>
DATA ENDS

CODE SEGMENT
    START: MOV AX, DATA
           MOV DS, AX
           MOV BL, OPR1
           XOR AL, AL
           MOV AL,OPR2
           ADD AL, BL
           DAA
           MOV RESULT, AL
           INC MSBO
           INC [RESULT+1]
    MSBO: MOV AH,4CH
           INT 21H
CODE ENDS
    END START
```

11. Multiplication and division of operands



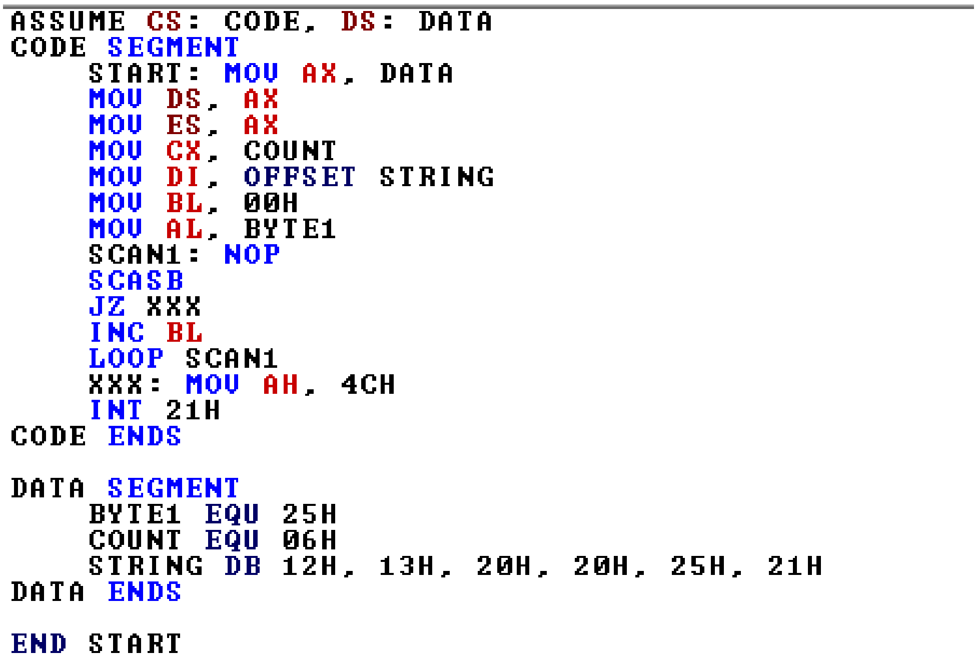
```
ASSUME CS: CODE, DS: DATA
DATA SEGMENT
    OPR1 EQU 98H
    OPR2 EQU 49H
    PROD DW 01 DUP<00>
    DIVS DW 01 DUP<00>
DATA ENDS

CODE SEGMENT
    START: MOV AX, DATA
    MOV DS, AX
    MOV BL, OPR2
    XOR AL, AL
    MOV AL, OPR1
    MUL BL
    MOV WORD PTR PROD, AX
    XOR AH, AH
    MOV AL, OPR1
    DIV BL
    MOV WORD PTR DIVS, AX
    MOV AH, 4CH
    INT 21H
CODE ENDS

END START
```

12. BCD operation of addition and subtraction



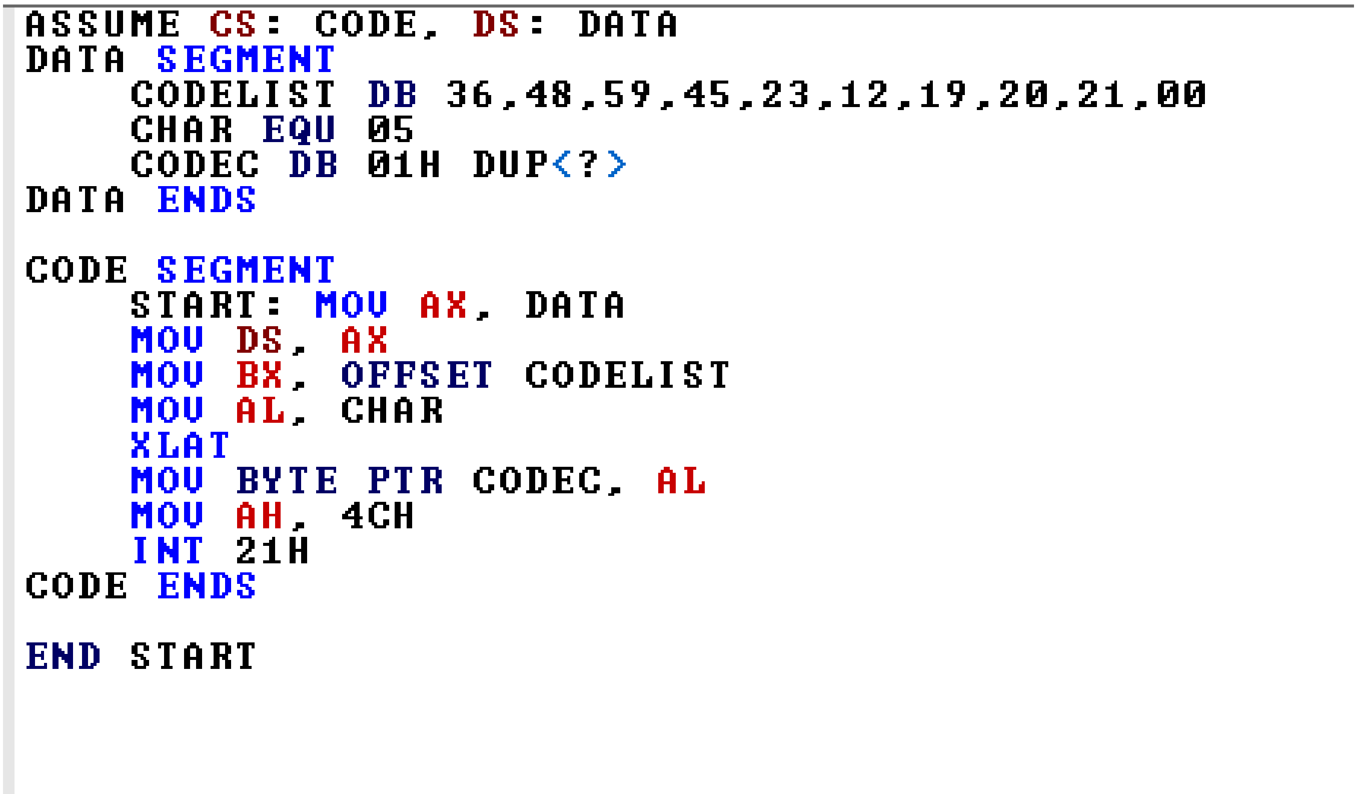
```
ASSUME CS: CODE, DS: DATA
DATA SEGMENT
    OPR1 EQU 98H
    OPR2 EQU 49H
    SUM DW 01 DUP(00)
    SUBT DW 01 DUP(00)
DATA ENDS

CODE SEGMENT
    START: MOV AX, DATA
    MOV DS, AX
    MOV BL, OPR2
    XOR AL, AL
    MOV AL, OPR1
    ADD AL,BL
    DAA
    MOV BYTE PTR SUM, AL
    JNC MSBO
    INC [SUM+1]
    MSBO: XOR AL, AL
    MOV AL, OPR1
    SUB AL, BL
    DAS
    MOV BYTE PTR SUBT, AL
    JNB MSB1
    INC [SUBT+1]
    MSB1: MOV AH, 4CH
    INT 21H
CODE ENDS

END START
```

13. Find whether given byte is in string or not. If present, then find relative address of byte form starting location



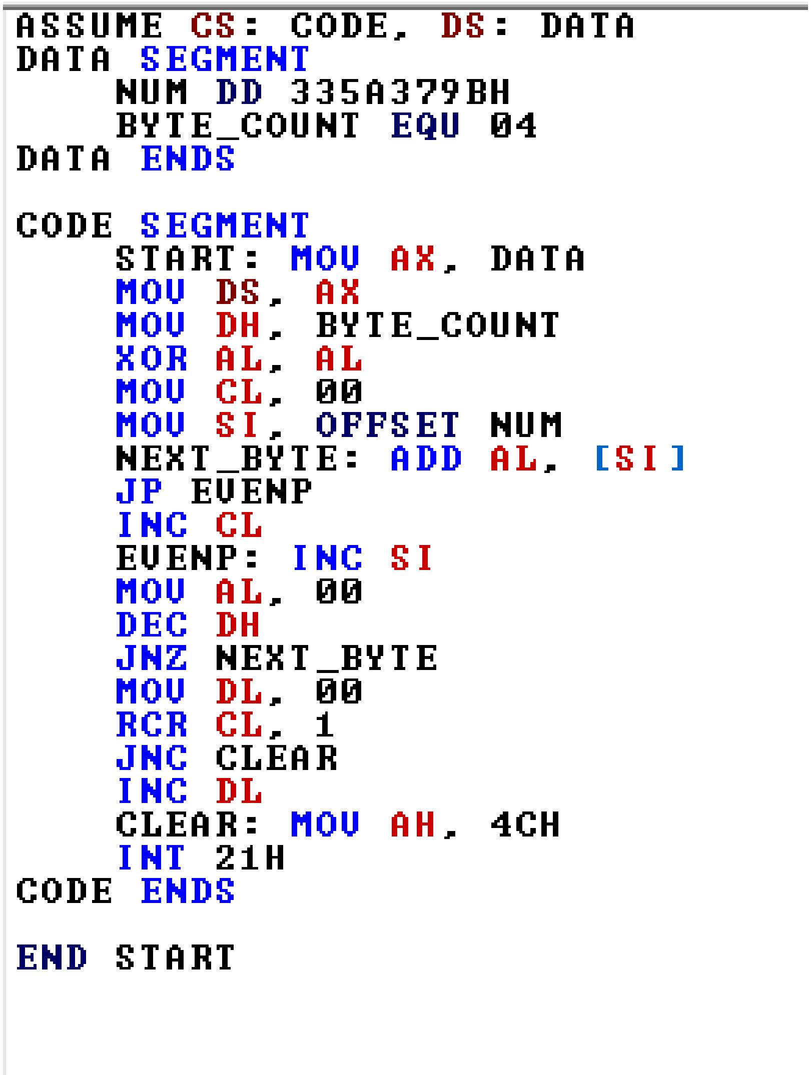
```
ASSUME CS: CODE, DS: DATA
CODE SEGMENT
    START: MOV AX, DATA
    MOV DS, AX
    MOV ES, AX
    MOV CX, COUNT
    MOV DI, OFFSET STRING
    MOV BL, 00H
    MOV AL, BYTE1
    SCAN1: NOP
    SCASB
    JZ XXX
    INC BL
    LOOP SCAN1
    XXX: MOV AH, 4CH
    INT 21H
CODE ENDS

DATA SEGMENT
    BYTE1 EQU 25H
    COUNT EQU 06H
    STRING DB 12H, 13H, 20H, 20H, 25H, 21H
DATA ENDS

END START
```

14. BCD no. 0 to 9 to equivalent 7 segment using lookup table



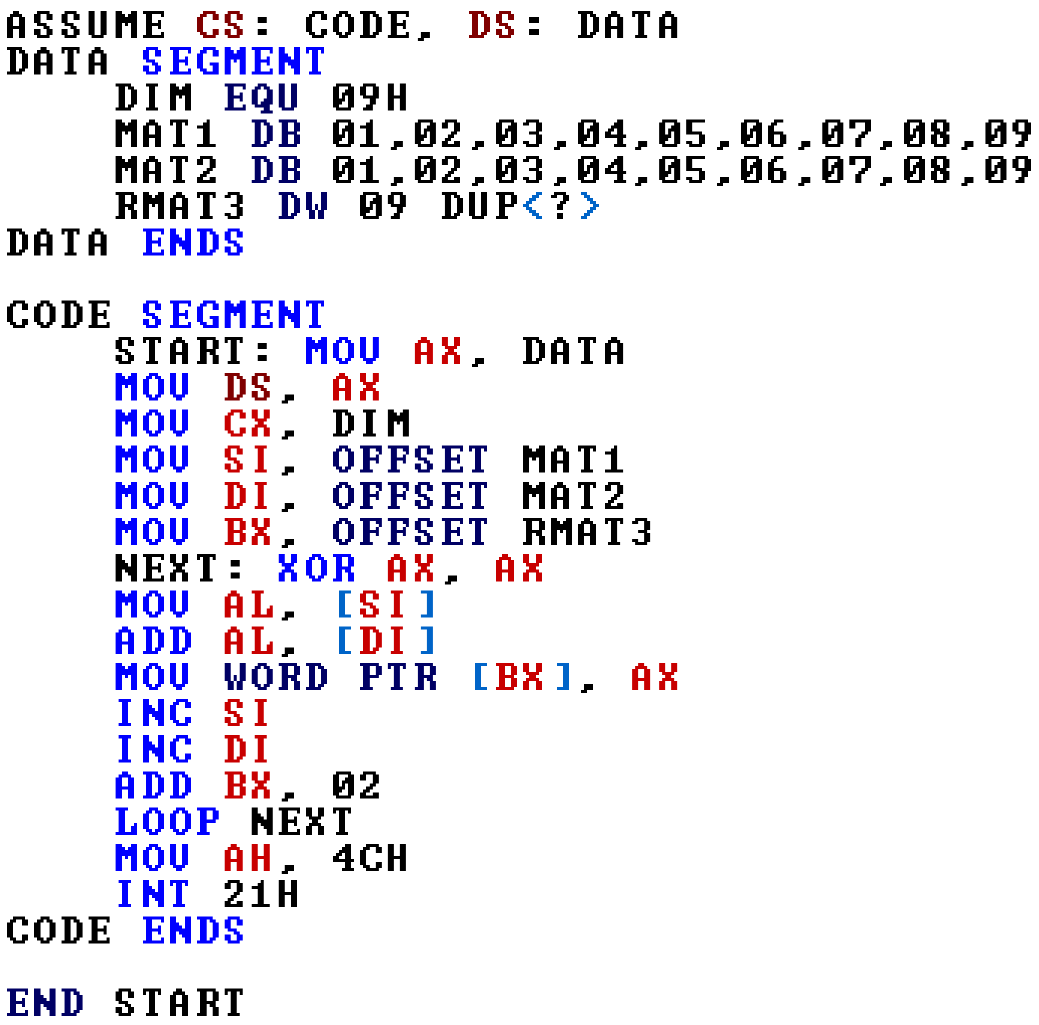
```
ASSUME CS: CODE, DS: DATA
DATA SEGMENT
    CODELIST DB 36,48,59,45,23,12,19,20,21,00
    CHAR EQU 05
    CODEC DB 01H DUP<?>
DATA ENDS

CODE SEGMENT
    START: MOV AX, DATA
    MOV DS, AX
    MOV BX, OFFSET CODELIST
    MOV AL, CHAR
    XLAT
    MOV BYTE PTR CODEC, AL
    MOV AH, 4CH
    INT 21H
CODE ENDS

END START
```

15. Check if parity is even or odd. If even, then set DL to 00 else 01



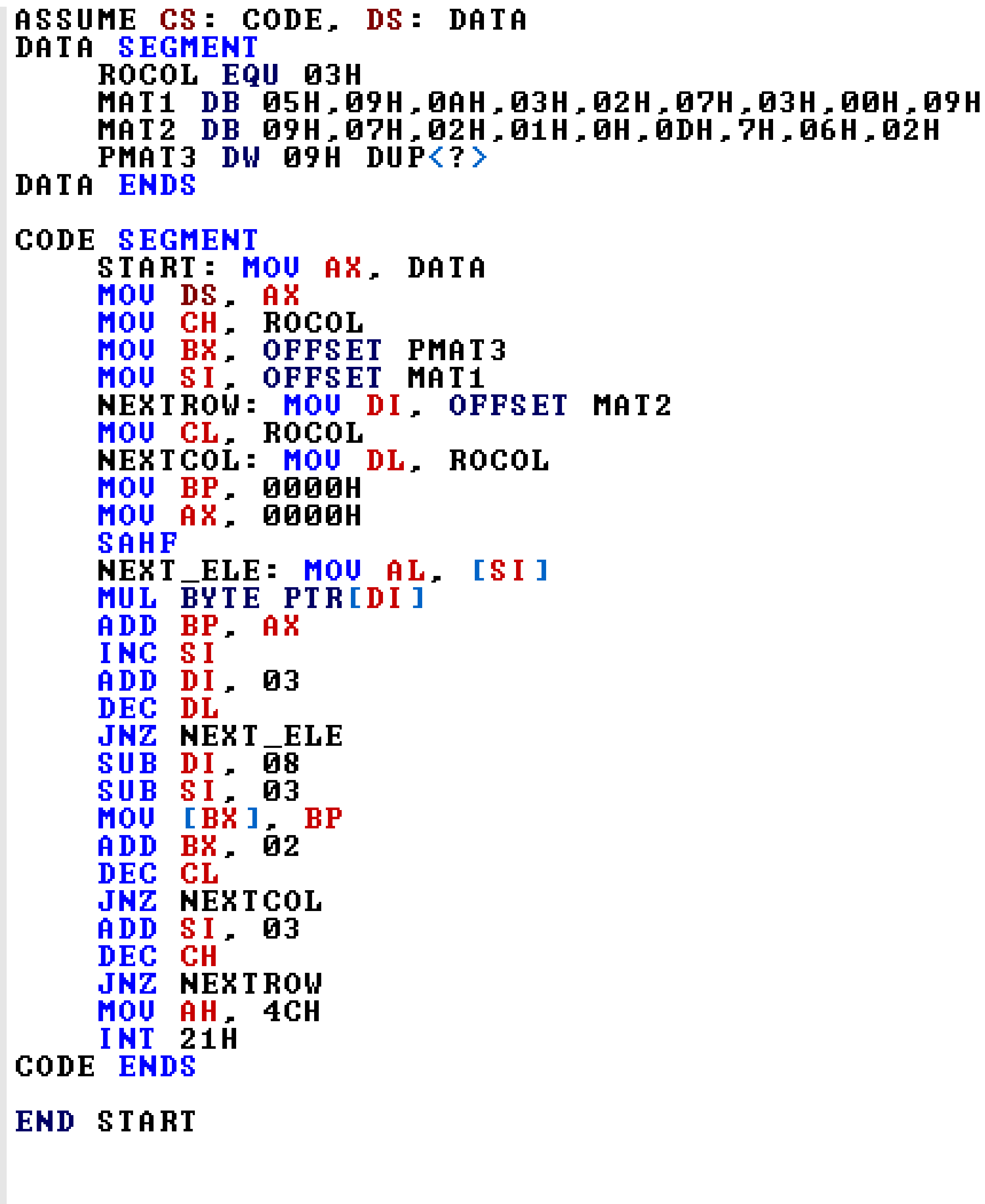
```
ASSUME CS: CODE, DS: DATA
DATA SEGMENT
    NUM DD 335A379BH
    BYTE_COUNT EQU 04
DATA ENDS

CODE SEGMENT
    START: MOV AX, DATA
    MOV DS, AX
    MOV DH, BYTE_COUNT
    XOR AL, AL
    MOV CL, 00
    MOV SI, OFFSET NUM
    NEXT_BYTE: ADD AL, [SI]
    JP EVENP
    INC CL
    EVENP: INC SI
    MOV AL, 00
    DEC DH
    JNZ NEXT_BYTE
    MOV DL, 00
    RCR CL, 1
    JNC CLEAR
    INC DL
    CLEAR: MOV AH, 4CH
    INT 21H
CODE ENDS

END START
```

16. Addition of two 3X3 matrices



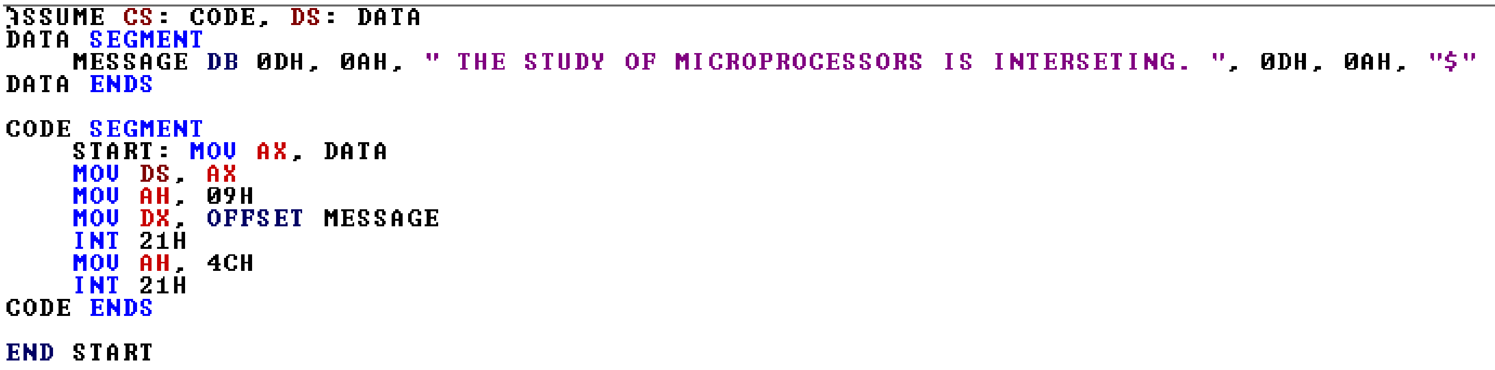
```
ASSUME CS: CODE, DS: DATA
DATA SEGMENT
    DIM EQU 09H
    MAT1 DB 01,02,03,04,05,06,07,08,09
    MAT2 DB 01,02,03,04,05,06,07,08,09
    RMAT3 DW 09 DUP<?>
DATA ENDS

CODE SEGMENT
    START: MOV AX, DATA
    MOV DS, AX
    MOV CX, DIM
    MOV SI, OFFSET MAT1
    MOV DI, OFFSET MAT2
    MOV BX, OFFSET RMAT3
    NEXT: XOR AX, AX
    MOV AL, [SI]
    ADD AL, [DI]
    MOV WORD PTR [BX], AX
    INC SI
    INC DI
    ADD BX, 02
    LOOP NEXT
    MOV AH, 4CH
    INT 21H
CODE ENDS

END START
```

17. Product of two matrices



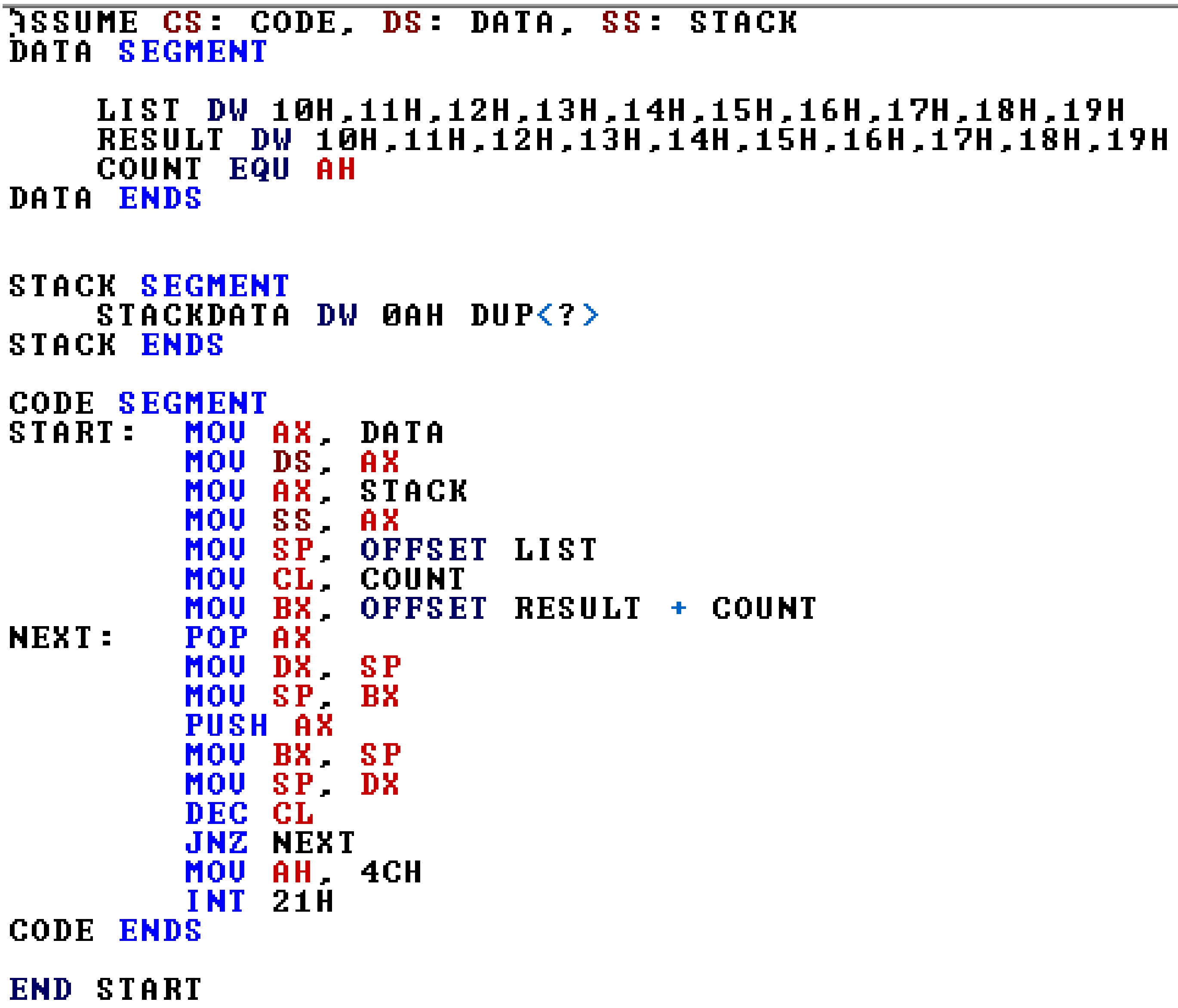
```
ASSUME CS: CODE, DS: DATA
DATA SEGMENT
    ROCOL EQU 03H
    MAT1 DB 05H,09H,0AH,03H,02H,07H,03H,00H,09H
    MAT2 DB 09H,07H,02H,01H,0H,0DH,7H,06H,02H
    PMAT3 DW 09H DUP(?)
DATA ENDS

CODE SEGMENT
    START: MOV AX, DATA
    MOV DS, AX
    MOV CH, ROCOL
    MOV BX, OFFSET PMAT3
    MOV SI, OFFSET MAT1
    NEXTROW: MOV DI, OFFSET MAT2
    MOV CL, ROCOL
    NEXTCOL: MOV DL, ROCOL
    MOV BP, 0000H
    MOV AX, 0000H
    SAHF
    NEXT_ELE: MOV AL, [SI]
    MUL BYTE PTR[DI]
    ADD BP, AX
    INC SI
    ADD DI, 03
    DEC DL
    JNZ NEXT_ELE
    SUB DI, 08
    SUB SI, 03
    MOV [BX], BP
    ADD BX, 02
    DEC CL
    JNZ NEXTCOL
    ADD SI, 03
    DEC CH
    JNZ NEXTROW
    MOV AH, 4CH
    INT 21H
CODE ENDS

END START
```

18. Display message “Study of microprocessor is interesting” on screen



```
ASSUME CS: CODE, DS: DATA
DATA SEGMENT
    MESSAGE DB 0DH, 0AH, " THE STUDY OF MICROPROCESSORS IS INTERSETING. ", 0DH, 0AH, "$"
DATA ENDS

CODE SEGMENT
    START: MOV AX, DATA
    MOV DS, AX
    MOV AH, 09H
    MOV DX, OFFSET MESSAGE
    INT 21H
    MOV AH, 4CH
    INT 21H
CODE ENDS

END START
```

19. Change sequence of 16 2 byte numbers from ascending to descending order. Numbers are stored in data segment. Store number series at address starting from 6000H. Use LIFO property of stack.



```
ASSUME CS: CODE, DS: DATA, SS: STACK
DATA SEGMENT

    LIST DW 10H,11H,12H,13H,14H,15H,16H,17H,18H,19H
    RESULT DW 10H,11H,12H,13H,14H,15H,16H,17H,18H,19H
    COUNT EQU AH
DATA ENDS


STACK SEGMENT
    STACKDATA DW 0AH DUP(?)
STACK ENDS

CODE SEGMENT
START:  MOV AX, DATA
        MOV DS, AX
        MOV AX, STACK
        MOV SS, AX
        MOV SP, OFFSET LIST
        MOV CL, COUNT
        MOV BX, OFFSET RESULT + COUNT
NEXT:   POP AX
        MOV DX, SP
        MOV SP, BX
        PUSH AX
        MOV BX, SP
        MOV SP, DX
        DEC CL
        JNZ NEXT
        MOV AH, 4CH
        INT 21H
CODE ENDS

END START
```