Assignment 5

余北辰 519030910245

1. (Array initialization) Allocate 100 bytes of data, and assign value from 0-99 to those bytes.

```
.MODEL SMALL
.STACK 64
;-------------------
        .DATA
ARRAY   DB  100 DUP(?)

;-------------------
        .CODE
MAIN    PROC    FAR
        MOV     AX, @DATA
        MOV     DS, AX
        MOV     BX, OFFSET ARRAY
        MOV     CX, 100
        MOV     AL, 0H
BACK:   MOV     [BX], AL
        INC     AL
        INC     BX
        LOOP    BACK
        MOV     AH, 4CH
        INT     21H
MAIN    ENDP
        END     MAIN
```

2. (If/else translation) Allocate a word in the memory with any value and another word for storing its absolute value. Write the program to perform the conversion.

```
.MODEL SMALL
.STACK 64
;------------------
    .DATA
NUM   DW    -1
ANS   DW    ?
;------------------
    .CODE
MAIN   PROC   FAR
    MOV    AX, @DATA
    MOV    DS, AX
    MOV    BX, NUM
    CMP    BX, 0
    JGE    FINISH
    MOV    AX, 0
    SUB    AX, BX
    MOV    BX, AX
FINISH: MOV    ANS, BX
    MOV    AH, 4CH
    INT    21H
MAIN   ENDP
    END    MAIN
```

3. (Function argument and return value) Convert the following C code to the 8086 assembly code. You need to handle the function call argument passing and return value properly. [Hint: you can use stack for passing the argument and register AX for receiving the return value of the function.]

```
.MODEL SMALL
.STACK 64
;------------------
    .DATA
DATA_A   DW   -1
DATA_B   DW   1
DATA_C   DW   ?
DATA_D   DW   ?
;------------------
    .CODE
MAIN   PROC   FAR
    MOV    AX, @DATA
    MOV    DS, AX
```

```
14        MOV     BX, DATA_A
15        PUSH    BX
16        CALL    ABS
17        POP     BX
18        MOV     DATA_C, AX
19
20        MOV     BX, DATA_B
21        PUSH    BX
22        MOV     BX, DATA_A
23        PUSH    BX
24        CALL    ADD1
25        POP     BX
26        MOV     DATA_D, AX
27
28        MOV     AH, 4CH
29        INT     21H
30    MAIN    ENDP
31
32
33    ABS     PROC
34      MOV BP, SP
35        MOV     AX, [BP] + 2
36        CMP     AX, 0
37        JGE     FINISH
38        MOV     BX, 0
39        SUB     BX, AX
40        MOV     AX, BX
41    FINISH: RET
42
43    ABS     ENDP
44
45
46    ADD1    PROC
47      MOV BP, SP
48        MOV     AX, [BP] + 2
49        MOV     BX, [BP] + 4
50        ADD     AX, BX
51
52      RET
53    ADD1    ENDP
54
55    END     MAIN
```

4. (Function local variable) Convert the foo() function to assembly (no need to convert main). You need to handle the local variables properly. [Hint: you can use the stack for local variables and remember to clean the stack before return.]

```
.MODEL SMALL
.STACK 64
;-------------------
    .DATA
DATA_A   DW   ?
;-------------------
    .CODE
MAIN   PROC   FAR
    MOV    AX, @DATA
    MOV    DS, AX
    MOV    BX, DATA_A
    CALL   FOO
    MOV    AH, 4CH
    INT    21H
MAIN   ENDP


FOO    PROC
    PUSH   CX
    PUSH   BX
    PUSH   AX
    ...; Use AX as a, BX as b and CX as c in the foo()
    POP    AX
    POP    BX
    POP    CX
FOO    ENDP

END    MAIN
```