1.

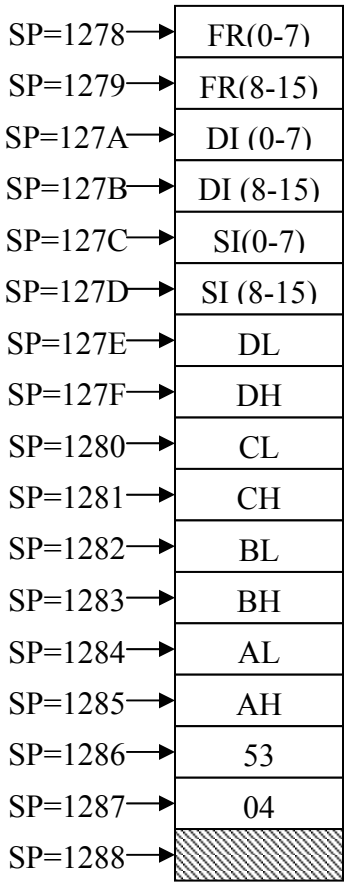
```
; This program transfers 6 bytes of data from memory locations with offset of 0010H to
; Memory locations with offset of 0028H.

TITLE          PROG2-3  (EXE)   PURPOSE: TRANSFERS 6 BYTES OF DATA
PAGE  60,132
                .MODEL SMALL
                .STACK 64
                .DATA
                ORG    10H
DATA_IN         DB              25H, 4FH, 85H, 1FH, 2BH, 0C4H
                ORG    28H
COPY            DB              6 DUP (?)
;--------------
            .CODE
MAIN            PROC            FAR
                MOV   AX,@DATA
                MOV   DS, AX
                MOV     SI, OFFSET DATA_IN
                MOV     DI, OFFSET COPY
                MOV     CX, 03H
MOV_LOOP:       MOV     Ax, [SI]
                MOV     [DI], Ax
                INC     SI
                INC     SI
                INC     DI
                INC     DI
                DEC     CX
                JNZ     MOV_LOOP
                MOV     AH,4CH
                INT     21H
   MAIN         ENDP
                END    MAIN
```

2.

The three steps to create an executable Assembly language program are outlined as follows:

Step	Input	Program	Output
1. Edit the program	Keyboard	Editor	Myfile.asm
2. Assemble the program	Myfile.asm	MASM or TASM	Myfile.obj
3. Link the program	Myfile.obj	LINK or TLINK	Myfile.exe

3. LINK or TLINK produces ".exe" file
4. MASM or TASM produces ".obj" file
5. False
6. Yes
7. After
8.



9.

Sequence of POP	**SP after each POP**
POPF	SP=127A
POP DI	SP=127C
POP SI	SP=127E
POP DX	SP=1280
POP CX	SP=1282
POP BX	SP=1284
POP AX	SP=1286

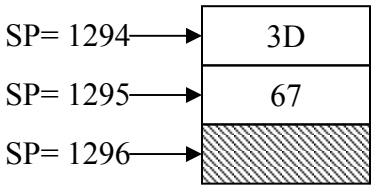
10.

It must be noted that in the **NEAR** call only the IP is saved on the stack, and in the case of **FAR** call both CS and IP are saved. After finishing execution of the subroutine, for control to be transferred back to the caller, the last instruction in the called subroutine must be **RET** (return). In the same way that the assembler generates different opcode for FAR and NEAR calls, the opcode for the RET instruction in the case of NEAR and FAR is different, as well. For Near calls, the IP is restored; for FAR calls, both CS and IP are restored.

11. In a Far call, CS and IP are saved on the stack, whereas in a near call, IP is saved on the stack.

12. In case of NEAR call 2 bytes and in case of FAR call 4 bytes.

13.



14.

a. JNC ERROR1

Address = IP of next instruction + the operand

= (E06C+2) +3F= E0AD

b. JNO ERROR1

Address = IP of next instruction + the operand

= (E072+2) +39= E0AD

c. JMP C8

Address = IP of next instruction + the operand (neglect carry)

= (E0A7+2) +E3=E08C

15.

Offset	Data
0020	31
0021	2D
0022	38
0023	30
0024	30
0025	2D
0026	35
0027	35
0028	35
0029	2D
002A	31
002B	32
002C	33
002D	34
0040	4E
0041	61
0042	6D
0043	65
0044	3A
0045	20
0046	4A
0047	6F
0048	68
0049	6E
004A	4A
004B	6F
004C	73

Offset	Data
0060	35
0061	39
0062	35
0063	36
0064	33
0065	34
0066	32
0070	60
0071	25
0072	06
0073	10
0074	49
0075	00
0080	6E
0081	7F
0082	69
0083	25
0084	F2
0085	99
0086	1C
0087	A2
0088	7B
0089	9E
008A	00
008B	00

Offset	Data
0090	28
0091	98
0092	99
0093	24
0094	79
0095	99
0096	39
0097	04
0098	00
0099	00
009A	EE
009B	EE
009C	EE
009D	EE
009E	EE
009F	EE

16.

```
TITLE     Problem (EXE) PROBLEM 16 PROGRAM
PAGE      60,132
            .MODEL SMALL
            .STACK 64
            .DATA
DATA      DW 234DH, 0DE6H, 3BC7H, 566AH
          ORG 10H
SUM       DW    ?
          .CODE
START     PROC   FAR
          MOV AX, @DATA
          MOV DS, AX
          MOV CX, 04H
          MOV BX, 0H
          MOV DI, OFFSET DATA
LOOP1:    ADD BX, [DI]
          INC DI
          DEC CX
          JNZ  LOOP1
          MOV SI, OFFSET SUM
          MOV [SI], BX
          MOV AH, 4CH
          INT 21H
START     ENDP
          END   START
```