SOLUTIONS FOR CHAPTER 2

1.

```
        .MODEL SMALL
        .STACK 64
        .DATA
                ORG     10H
DATA_IN         DW              2525H,4FFFH,8555H,1F00H,2BBBH,0C4H
                ORG     28H
COPY            DW              6 DUP(?)
;-------------------------------------
        .CODE
MAIN    PROC    FAR
        MOV     AX,@DATA
        MOV     DS,AX
        MOV     SI,OFFSET DATA_IN       ;SI points to data to be copied
        MOV     DI,OFFSET COPY          ;DI points to copy of data
        MOV     CX,06H                  ;loop counter = 6
MOV_LOOP: MOV  AX,[SI]                  ;move the next word from DATA area to AL
        MOV     [DI],AX                 ;move the next word to COPY area
        INC     SI                      ;increment DATA pointer
        INC     SI
        INC     DI                      ;increment COPY pointer
        INC     DI
        DEC     CX                      ;decrement LOOP counter
        JNZ     MOV_LOOP                ;jump if loop counter not zero
        MOV     AH,4CH                  ;set up to return
        INT     21H                     ;return to DOS
MAIN    ENDP
        END     MAIN
```

2. first the source file (extension "asm") must be produced with a word processor which produces an ASCII file
   then the program is assembled to produce the object (extension "obj") file
   then the program is linked to produce the executable (extension "exe") file

3. the linker program

4. the assembler program

5. false

7. after

8. when the procedure is called, IP (which points to the next instruction to be executed after the CALL) is saved on the stack since it is a NEAR procedure. After the CALL and all PUSH instructions have been executed, the stack is as follows with SP = 1278.

   1278 <- flag register
   127A <- DI
   127C <- SI
   127E <- DX
   1280 <- CX
   1282 <- BX
   1284 <- AX 1285 = (AH) 1284 = (AL)
   1286 <- IP 1287 = (04) 1286 = (53)
   1288

9. SP = 1278
   POPF;now SP = 127A
   POP DI ;now SP = 127C
   POP SI ;now SP = 127E
   POP DX ;now SP = 1280
   POP CX ;now SP = 1282
   POP BX ;now SP = 1284
   POP AX ;now SP = 1286
   ;SP = 1288 after the RET

10. the address of the instruction immediately following the CALL is stored on the stack. The last instruction of a called subroutine must be RET in order to tell the system to pop off the return address from the stack.

11. CS and IP, IP

12. NEAR calls require two bytes to store IP
    FAR calls require four bytes to store CS and IP

13. IP = 673D will be stored in the stack at 1295 and 1294, therefore
    SS:1295 = 67 and SS:1294 = 3D

14. (a) 3F (displacement) + 6E (instruction after JNC) = E0AD, the offset of ERROR1
    (b) 39 (displacement) + 74 (instruction after JNO) = E0AD, the offset of ERROR1
    (c) E3 (displacement) + A9 (instruction after JMP) = E08C, the offset of C8

15. the following notation indicates "offset location: byte contents of location"

for DATA1			
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		
for DATA2			
0040:4E	0041:61	0042:6D	0043:65
0044:3A	0045:20	0046:4A	0047:6F
0048:68	0049:6E	004A:20	004B:4A
004C:6F	004D:6E	004E:65	004F:73
for DATA3			
0060:35	0061:39	0062:35	0063:36
0064:33	0065:34	0066:32	
for DATA4			
0070:60	0071:25	0072:06	0073:10
for DATA5			
0074:31	0075:00		
for DATA6			
0080:6E	0081:7F	0082:69	0083:25
for DATA7			
0084:F2	0085:99	0086:1C	0087:A2
0088:7B	0089:9E		
for DATA8			
0090:28	0091:98	0092:99	0093:24
0094:79	0095:99	0096:39	0097:04
0098:00	0099:00		
for DATA9			
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  32
;-----------------------------
        .DATA
DATA    DW      234DH,0DE6H,3BC7H,566AH     ;leading zero
        ORG      10H
SUM     DW      ?
;-----------------------------
        .CODE
START   PROC  FAR               ;no colon after START
        MOV  AX,DATA            ;should be @DATA
        MOV  DS,AX
        MOV  CX,04
        MOV  BX,0
        MOV  DI,OFFSET DATA
LOOP1:  ADD  BX,[DI]
        INC  DI
        INC  DI
        DEC  CX                 ;DEC CX NOT BX
        JNZ  LOOP1
        MOV  SI,OFFSET SUM      ;SUM  instead of RESULT
        MOV  [SI],BX
        MOV  AH,4CH
        INT  21H
START   ENDP
        END  START              ;START not STRT
```