1. **Read Intel 8086/88 User Manual page 6-53 to 6-55: understand how the instruction is encoded to and decoded from machine code. Use the two tables that followed (table 6-22 and 6-23); do the following two questions. Convert the following hexadecimal machine codes to assembly language mnemonics. State what each of the byte fields mean (Table 6-23 from page 6-64 to page 6-69).**

   a. B8 00 20

   b. 8E D8

   c. 46

   d. 90

   e. 89 7C FE

   f. 75 F7

   g. E2 EF

   h. 26 80 07 78

a. **B8 00 20**

B8 = MOV AX, IMMED16
00 = Data-lo
20 = Data-hi

Answer:	MOV AX, 2000h

b. **8E D8**

8E = MOV SEGREG, REG16 / MEM16
D8 = 2nd byte: MOD 0 SR R/M
11 0 11 000

MOD = 11 Register Mode
SR = 11 Segment register DS (see page 3-57) – operand 1
R/M = 000 Register AX – operand 2

Answer:	MOV DS, AX

c. **46**

46 = INC SI

Answer:	INC SI

d. **90**

90 = NOP

Answer:	NOP

e. **89 7C FE**

89 = MOV REG16 / MEM16, REG16 (note error in table 6-23)
7C = 2nd byte: MOD REG R/M
01 111 100
MOD = 01 Memory Mode, 8-bit displacement follows
REG = 111 DI – operand #2
R/M = 100 (SI) + D8 - operand #1
FE = 8-bit signed displacement
11111110
2's complement: 00000010 = 2
Thus the displacement is –2

Answer:	MOV [SI]-2, DI

f. **75 F7**

75 = JNE/JNZ short-label
F7 = IP – INC8 (8-bit signed offset to add to IP)
11110111
2's complement: 00001001 = 9
Thus the offset is –9

Answer:	JNE –9	(that is, jump back 9 machine code bytes if not equal)
or	JNZ -9	

g. **E2 EF**

E2 = LOOP short-label
EF = IP – INC8 (8-bit signed offset to add to IP)
11101111
2's complement: 00010001 = 17
Thus the offset is –17

Answer:	LOOP –17	(loop back 17 machine bytes)

h. **26 80 07 78**

26 = Segment override – 'ES:'
80 = One of several choices; look at bits 3, 4, & 5 of next byte
07 = 00000111 Bits 3, 4, & 5 are '000', so instruction is
ADD REG8 / MEM8, IMMED8
MOD = 00 Memory mode, no displacement follows
R/M = 111 (BX)
78 = 8-bit immediate value

Note: instruction **must** have BYTE PTR to indicate an 8-bit operation

Answer:	ADD BYTE PTR ES:[BX], 78h

2. Convert the following instructions to machine code – give your answers in hexadecimal. State what each of the bit fields mean.

```
i. PUSH BX
j. MOV [SI+490], SP
k. OUT DX, AL
l. POPF
m. AND AX, [BX+DI+2Dh]
n. ADD DS:[BP], DX        Note: you will have to 'add' a displacement
o. XOR AL, [BX+DI-36H]
p. MOV [DI+476], ES
```

a. PUSH BX – two possible answers

Memory or Register Operand

11111111	mod 110 r/m

mod = 11 Register Mode (r/m is treated as a "reg" field)
r/m = 011 Register BX

Answer #1:	FF F3

Register Operand

01010 reg

reg = 011 Register BX

Answer #2:	53

b. MOV [SI+490], SP

Memory or Register Operand to/from Register Operand

100010 d w	mod reg r/m	disp-lo	disp-hi

d = 0 From register
w = 1 Word operands (SP is a 2 byte register)
mod = 10 Memory Mode, 16-bit displacement follows
reg = 100 Register SP
r/m = 100 EA = (SI) + D16
disp = 1EAh Displacement = 490 = 1EAh

Answer:	89 A4 EA 01

c. OUT DX, AL

Variable Port

1110111w

w = 0 Byte operand (AL is a 1 byte register)

Answer:	EE

d. POPF

10011101

Answer: 9D

e. AND AX, [BX+DI+2Dh]

Memory or Register Operand with Register Operand

001000 d w | mod reg r/m | disp-lo

d = 1	To register
w = 1	Word operands (AX is 2 bytes large)
mod = 01	Memory Mode, 8-bit displacement follows (2Dh)
reg = 000	Register AX
r/m = 001	(BX) + (DI) + D8
disp = 2D	Displacement is 2Dh, which can fit in a 1 byte signed number

Answer: 23 41 2D

f. ADD DS:[BP], DX

Segment override: It is a bit tricky finding the prefix byte for segment overrides. If you look in Table 6-22 of the Intel User's Manual, on page 6-61, you will see that the last entry is:

SEGMENT=Override prefix 001 reg 110

The **reg** field is actual a segment register field, and you can use the **Segment** column of the **"reg" Field Bit Assignments** chart on page 3-57 to determine how to set it.

Segment DS Override: 001 11 110 = 3E

Note: you will have to 'add' a displacement

The addition of a displacement to the memory reference (that is, [BP]), is needed because there is no encoding for [BP]. Logically, the **mod** field should be 00, and the **r/m** field should be 110. But that is a special case, for when a direct address is used (something like [1000h]). To encode, you will have to rewrite the instruction into the functionally equivalent form:

ADD DS:[BP+0], DX

Memory or Register Operand with Register Operand

000000 d w | mod reg r/m | disp-lo

d = 0	From register
w = 1	Word operands (DX is 2 bytes large)
mod = 01	Memory Mode, 8-bit signed displacement follows
reg = 010	Register DX
r/m = 110	(BP) + D8
disp = 00	Displacement is 0

Answer: 3E 01 56 00

g. XOR AL, [BX+DI-36H]

Memory or Register Operand with Register Operand

001100 d w | mod reg r/m | disp-lo

d = 1	To register
w = 0	Byte operands (AL is 1 byte large)
mod = 01	Memory Mode, 8-bit displacement follows
reg = 000	Register AL
r/m = 001	(BX) + (DI) + D8
disp	Displacement is –36h, which can fit in an 8-bit 2's complement form

36h = 0011 0110b -> (2's comp) -> 11001010b = CAh = disp

Answer:	32 41 CA

h. MOV [DI + 476], ES

Segment Register to Memory or Register Operand

10001100	mod 0 reg r/m	disp-lo	disp-hi

mod = 10	Memory Mode, 16-bit displacement follows - Note that 476 is too large for an 8-bit signed number!
reg = 00	Register ES (note that this is a segment register, and thus is 2 bits large)
r/m = 101	(DI) + D16
disp = 01DC	Displacement is $476_{10} = 1DC_{16}$

Answer:	8C 85 DC 01

3. The following bytes are found in order somewhere in memory. Assuming they are machine codes, decode the values into meaningful assembly language mnemonics.

```
B9 00 12 D0 C0 E8 C8 E2 F9
```

B9 – MOV CX, ImmeD16

The next two bytes are the immediate 16-bit value loaded into CX (00 12 -> 1200H)

MOV CX, 1200H

D0 – One of eight possibilities (ROL, ROR, RCL, etc.), so use the next byte.

C0 = 11000000. The first two digit (MSB) "11" is MOD and "11" means that r/m = reg field. The next three digits "000" indicates that ROL Reg8, 1. The last three digits (LSB) is "000", it is R/M field and represent AL.

ROL AL, 1

E8 – CALL Near-proc

Indicates that there is a call to a subroutine in the same segment. The next two bytes (IP-INC-Lo and IP-INC-Hi) give a signed 16-bit displacement from the current value of IP.

Disp = E2C8 (negative)
= 1110001011001000b -> -0001110100111000b
= - 1D38H = -7480D

CALL [IP-7480]

F9 – STC

4. Use full segment definition, write a DOS compatible program that: a) clears the screen, b) set the cursor to screen position row = 10 and column = 5, c) displays the prompt "Please enter an 8-digit number: ", d) get the keyboard input and save the number to a buffer area in the memory (you define), e) sort the number on its ascending order and save them to another buffer for display. For example, if the input number is 29034765, then after your sort, the result should be 02345679. You can assume that the number for each digit is non-repeat but actually the repeated case is the same, f) after your sort, change to the start of next new line, output "The sorted number is: " and the number, g) exit use DOS function 4CH. Write task a) and b) using subroutines. Test your code on PC by yourself.

```
;Tasks:
;(1) Clear the screen use subroutine
;(2) Set the cursor to ROW 10 and COLUMN 5 on the screen use subroutine
;(3) Output a prompt string: "Please enter an 8-digit number:"
;(4) Accept keyboard input: (put to buffer INPUT_BUF)
;(5) Sort the number on its ascending order
;(6) move the sorted number to display buffer (OUTPUT_BUF)
;(7) Change to a new line and Output string: "The sorted number is :" and the number and then exit
```