Machine Language

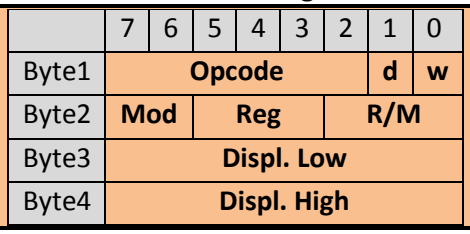
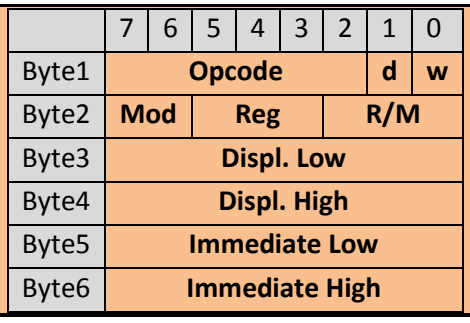
Machine Instruction Format:

	7	6	5	4	3	2	1	0
Byte1	**Opcode**						**d**	**w**
Byte2	**Mod**		**Reg**			**R/M**		
Byte3	**Displ. Low**							
Byte4	**Displ. High**							
Byte5	**Immediate Low**							
Byte6	**Immediate High**							

The Opcode BYTE

- d=0 when source is REG
- d=1 when source is Memory
- w=0 when 8-bit operands are used
- w=1 when 16-bit operands are used

Two Operands Instructions

Assembly Code	Working	Machine Code
1. Mnemonic: R/M, Reg Or Mnemonic: Reg, R/M Reg to R/M or R/M to Reg		
a. ADD AX,[SI]	[00000011b] [00 000 100b]	03 04 h
b. ADD [BX][DI]+1234h, AX	[00000001b] [10 000 001b]	01 81 34 12 h
c. MOV 1234(BP),DX	[10001001b] [10 010 110b] [34h] [12h]	89 96 34 12 h
d. MOV [BX + 10h], CL	[10001000b] [01 001 111b] [10h]	88 4F 10 h
e. MOV AX,BX	[10001001b] [11 011 000b]	89 D8 h
f. MOV CH,BL	[10001000b] [11 011 101b]	88 DD h
2. Mnemonic: Reg, Immediate **Imm. to reg**		[1011wReg] [DataL] [DataH]
a. MOV AX, 1	[10111000b] [01h] [00h]	B8 01 00 h
b. MOV BX, 1234h	[10111011b] [34h] [12h]	BB 34 12 h
c. MOV CL, 3h	[10110001b] [03h]	B1 03 h

3. Mnemonic: R/M, Immediate **Imm. to R/M**	<table><tr><td></td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>Byte1</td><td colspan="6">Opcode</td><td>d</td><td>w</td></tr><tr><td>Byte2</td><td>Mod</td><td colspan="3">Reg</td><td colspan="3">R/M</td></tr><tr><td>Byte3</td><td colspan="8">Displ. Low</td></tr><tr><td>Byte4</td><td colspan="8">Displ. High</td></tr><tr><td>Byte5</td><td colspan="8">Immediate Low</td></tr><tr><td>Byte6</td><td colspan="8">Immediate High</td></tr></table>	[1100 011w] [MOD 000 R/M] [dispL] [dispH][immL] [immH]
a. MOV WORD PTR [BX],100h	[11000111b] [00 000 111b] [00h] [01h]	C7 07 00 01 h
b. MOV BYTE PTR [100h], 10h	[11000110b] [00 000 110b] [00h] [01h] [10h]	C6 06 00 01 10 h
c. MOV WORD PTR [BX+SI], 10h	[11000111b] [00 000 000b] [10h]	C7 00 10 00 h
d. MOV WORD PTR [BX+DI+2], 1234h	[11000111b] [01 000 001b] [00000010b] [34h] [12h]	C7 41 02 34 12 h

Single operand instructions:

- Operand is a register (reg8/16) or a memory operand (mem8/16)
- always 2 bytes for opcode and addressing info
- may have up to 2 more bytes of immediate data

7 1	**0**
Opcode	w

7 6	**5 4 3**	**2 1 0**
mod	opcode	R/M

Opcode bits: 7 bits in the Opcode BYTE and only 3 bits (5,4,3) in the next BYTE.

- w=width of operand
  0= 8-bit
  1= 16-bit
- mod and r/m encode addressing info

POP Instruction

- Structure for memory operands: 8f mod 000 r/m
- Structure for register operands: 01011 reg

POP [DI]	[8fh] [00 000 101b]	[8F 05]h
POP DX	[01011 010]	[5A]h

Your Turn

[Use Tables at the end]

Q1. Convert the following in machine language:

1. INC DH
2. POP AX
3. POP BP
4. INC BYTE PTR [SI-4]

Q2. What will be the equivalent assembly code of given bytes?

1. FF C7
2. FE 84 80 00
3. 8F 06 00 12

Q3. Convert the following 2-operand instructions in machine language:

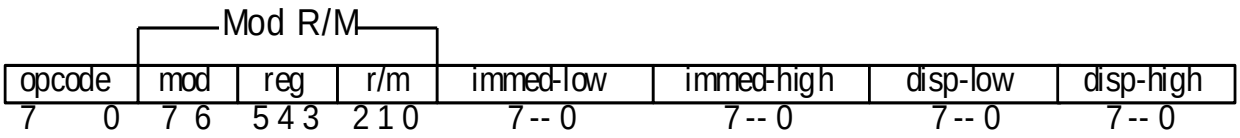
1. MOV WORD PTR [BX+SI], 10h
2. MOV BH, 3h
3. MOV AL, [34F4h]

Function	*Format*				*80186 Clock Cycles*	*80188 Clock Cycles*	*Comments*
Data Transfer							
Mov = Move:							
Register to Register/ Memory	1000100w	mod reg r/m			2/12	2/12*	
Register/memory to register	1000101w	mod reg r/m			2/9	2/9*	
Immediate to register/memory	1100011w	mod 000 r/m	data	data if w = 1	12/13	12/13	
Immediate to register	1011w reg	data	data if w = 1		3/4	3/4	8/16-bit
Memory to accumulator	1010000w	addr-low	addr-high		8	8*	8/16-bit
Accumulator to memory	1010001w	addr-low	addr-high		9	9*	
Register/memory to segment register	10001110	mod 0 reg r/m			2/9		2/13
Segment register to register/memory	10001100	mod 0 reg r/m			2/11	2/15	
PUSH = Push:							
Memory	11111111	mod 110 r/m			16	20	
Register	01010 reg				10	14	
Segment register	000 reg 1				10	9	13
Immediate	011010s0	data	data if s = 0		10	14	
PUSH = Push All	01100000				36	68	
POP = pop:							
Memory	10001111	mod 000r/m			20	24	
Register	01011	reg			10	14	
Segment register	000 reg 111	(reg ≠ 01)			8	12	
POPA = Pop All	01100001				51	83	
XCHG = Exchange:							
Register/memory with register	1000011W	mod reg r/m			4/17	4/17*	
Register with accumulator	10010 reg				3	3	
XLAT = Translate byte to AL	11010111				11	15	
LEA = Load EA to register	10001101	mod reg r/m			6	6	
LDS = Load pointer to DS	11000101	mod reg r/m	(mod ≠ 11)		18	26	
LES = Load pointer to ES	11000100	mod reg r/m	(mod ≠ 11)		18	26	
LAHF = Load AH with flags	10011111				2	2	
SAHF = Store AH into flags	10011110				3	3	
PUSHF = Push flags	10011100				9	13	
POPF = Pop flags	10011101				8	12	

Arithmetic							
ADD = Add:							
Reg/memory with register to either	000000dw	mod reg r/m	3/10	3/10*			
Immediate to register/memory	100000sw	mod 000 r/m	data	data if sw = 01	4/16	4/16*	
Immediate to accumulator	0000010w	data	data if w = 1		3/4	3/4	8/16-bit
ADC = Add with carry:							
Reg/memory with register to either	000100dw	mod reg r/m	3/10	3/10*			
Immediate to register/memory	100000sw	mod010 r/m	data	data if sw = 01	4/16	4/16*	
Immediate to accumulator	0001010w	data	data if w = 1		3/4	3/4	8/16-bit
INC = Increment:							
Register/memory	1111111w	mod 000 r/m			3/15	3/15*	
Register	01000 reg				3	3	
SUB = Subtract:							
Reg/memory and register to either	001010 dw	mod reg r/m	3/10	3/10*			
Immediate from register/memory	100000 sw	mod 101 r/m	data	data if s w = 01	4/16	4/16*	
Immediate from accumulator	0010110w	data	data if w = 1		3/4	3/4	8/16-bit
SBB = Subtract with borrow:							
Reg/memory and register to either	000110dw	mod reg r/m			3/10*	3/10*	
Immediate from register/memory	100000sw	mod 011 r/m	data	data if sw = 01	4/16*	4/16*	
Immediate from accumulator	0001110w	data	data if w = 1		3/4	3/4	8/16-bit
DEC = Decrement							
Register/memory	1111111w	mod 001 r/m			3/15	3/15*	
Register	01001 reg				3	3	
CMP = Compare:							
Register/memory with register	0011101w	mod reg r/m			3/10	3/10*	
Register with register/memory	0011100w	mod reg r/m			3/10	3/10*	
Immediate with register/memory	100000sw	mod 111 r/m	data	data if sw = 01	3/10	310*	
Immediate with accumulator	0011110w	data	data if w = 1		3/4	3/4	8/16-bit

JP/JPE = Jump on parity/ parity even	01111010	disp			4/13	4/13	
JO = Jump on overflow	01110000	disp			4/13	4/13	
JS = Jump on sign	01111000	disp			4/13	4/13	
JNE/JNZ = Jump on not equal/not zero	01110101	disp			4/13	4/13	
JNL/JGE = Jump on not less/greater or equal	01111101	disp			4/13	4/13	
JNLE/JG = Jump on not less or equal/greater	01111111	disp			4/13	4/13	
JNB/JAE = Jump on not below/above or equal	01110011	disp			4/13	4/13	
JNBE/JA = Jump on not below or equal/above	01110111	disp			4/13	4/13	
JNP/JPO = Jump on not par/par odd	01111011	disp			4/13	4/13	
JNO = Jump on not overflow	01110001	disp			4/13	4/13	
JNS = Jump on not sign	01111001	disp			4/13	4/13	
JCXZ = Jump on CX zero	11100011	disp			5/15	5/15	
LOOP = Loop CX times	11100010	disp			6/16	6/16	
LOOPZ/LOOPE = Loop while zero/equal	11100001	disp			6/16	6/16	
LOOPNZ/LOOPNE = Loop while not zero/ equal	11100000	disp			6/16	6/16	Loop not taken/ Loop taken
ENTER = Enter Procedure	11001000	data-low	data-high	L			
L = 0					15	19	
L = 1					25	29	
L > 1					22+16 (n – 1)	26 + 20 (n–1)	
LEAVE = Leave Procedure	11001001				8	8	
INT = Inerrupt:							
Type specified	11001101	type			47	47	if
Type 3	11001100				45	45	INT.
INTO = Interrupt on overflow	11001110				48/4	48/4	taken/ if
IRET = Interrupt return	11001111				28	28	INT.
BOUND = Detect value out of range	01100010	mod reg/m			33–35	33–35	not taken
PROCESSOR CONTROL							
CLC = Clear carry	11111000				2	2	
CMC = Complement carry	11110101				2	2	

STC = Set carry	11111001				2	2	
CLD = Clear direction	11111100				2	2	
STD = Set direction	11111101				2	2	
CLI = Clear interrupt	11111010				2	2	
STI = Set interrupt	11111011				2	2	
HLT = Halt	11110100				2	2	if $\overline{\text{TEST}}$ = 0
WAIT = Wait	10011011				6	6	
LOCK = Bus lock prefix	11110000				2	3	
ESC = Processor Extension Escape	11011 TTT	mod LLL r/m			6	6	
	(TTT LLL are opcode to processor extension)						
NOP = No Operation	10010000				3	3	



(The opcode indicates whether or not the immediate value field is present, as well as its size.)

CODE	EXPLANATION
00	Memory Mode, no displacement follows*
01	Memory Mode, 8-bit displacement follows
10	Memory Mode, 16-bit displacement follows
11	Register Mode (no displacement)

*Except when R/M = 110, then 16-bit displacement follows

(a)

R/M	Register	R/M	Register
000	AX or AL	100	SP or AH
001	CX or CL	101	BP or CH
010	DX or DL	110	SI or DH
011	BX or BL	111	DI or BH

MOD = 11			EFFECTIVE ADDRESS CALCULATION			
R/M	W = 0	W = 1	R/M	MOD = 00	MOD = 01	MOD = 10
000	AL	AX	000	(BX) + (SI)	(BX) + (SI) + D8	(BX) + (SI) + D16
001	CL	CX	001	(BX) + (DI)	(BX) + (DI) + D8	(BX) + (DI) + D16
010	DL	DX	010	(BP) + (SI)	(BP) + (SI) + D8	(BP) + (SI) + D16
011	BL	BX	011	(BP) + (DI)	(BP) + (DI) + D8	(BP) + (DI) + D16
100	AH	SP	100	(SI)	(SI) + D8	(SI) + D16
101	CH	BP	101	(DI)	(DI) + D8	(DI) + D16
110	DH	SI	110	DIRECT ADDRESS	(BP) + D8	(BP) + D16
111	BH	DI	111	(BX)	(BX) + D8	(BX) + D16