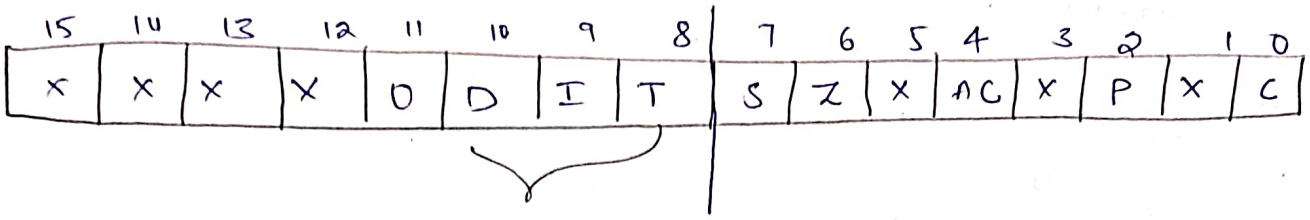
1.Q. Explain the control flags of 8086 Microprocessor.

Ans:- Generally a 16-Register contains 16 bits.

It has 9 flags. and 6 Status flags + 3 control flags.



control flags.

(1). Direction Flag (D):-

used for Strigned (multiple bytes) operations.

if D=0 Auto increment mode : lower Address to higher Address

if D=1 Auto Decrement mode : higher Address to lower Address.

(2). Interrupt flag (I):-

if I = 1 : Enable all maskable interrupts.

if I = 0 : Disable all maskable interrupts.

(No effect on Non-maskable).

(3). Trap/ Trace : (T): Enter into single step execution mode for debugging i.e executes 1 instruction and enters into single step ISR (Interrupt service Routine)

if T=1 CPU automatically generate an internal interrupt after each instructions

if T=0 No function is performed / NO ISR

2. Find out the state of conditional flags of the 8086 microprocessor after execution of following arithmetic operation

(a). 4FH + 48H (using 8-bit operation)

```
 01001111
 01001000
 ---------
 1001  0111
  9     7
```

OF = 0
SF = 0
ZF = 0
ACF = 1
PF = 0
CF = 0

(b). FFFFH + 0001H (using 16-bit operation)

```
  1111  1111  1111  1111
  0000  0000  0000  0001
 ------------------------
 (1) 0000  0000  0000  0000
```

ZF = 1
ACF = 1
PF = 1
CF = 1
OF = 1
SF = 0

(c). F5H - 3FH (using 8-bit operation)

```
  000  101
  xxx  0101
- 0011 1111
-------------
  1011 0110
   B    6
```

OF = 0
SF = 0
ZF = 0
ACF = 0
PF = 0
CF = 0

(3). Derive the logical expression of control signal for following operation in 8086 microprocessor.

(a) memory read operation
(b) memory write operation

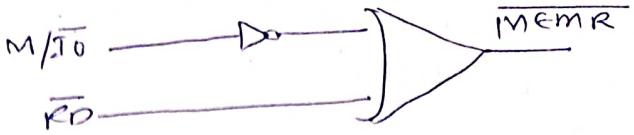
Ans: $\overline{RD}$ = 0, read operation
= 1, no read operation.

$\overline{WR}$ = 0, write operation
1, no write operation.

$M/\overline{IO}$ = 0, from I/O device
= 1, from memory.

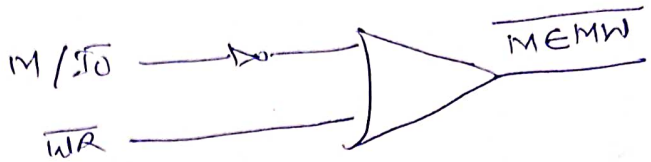
$\overline{RD}$	M/$\overline{IO}$	$\overline{MEMR}$
0	0	1
0	1	0
1	0	1
1	1	1

$\overline{MEMR} = \overline{M/IO} + \overline{RD}$



$\overline{WR}$	M/$\overline{IO}$	MEMW
0	0	1
0	1	0
1	0	1
1	1	1

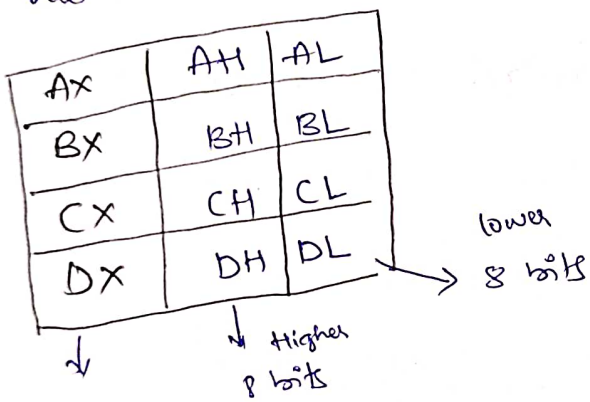
$\overline{MEMW} = \overline{M/IO} + \overline{WR}$



4. Explain general purpose registers of 8086 microprocessor.

Ans: General purpose Registers.

We have 4 General purpose Registers.



* For 8 bit operation;

AX : Default Accumulator

BX : Base address/offset while indexed addressing mode

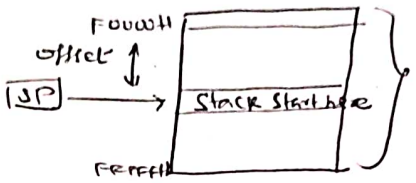
CX : Default counter for loop

DX : Default destination / implicit operand / string instructions.

Q5. What is stack pointer and what is the need of stack?

Ans: In the context of the 8086 processor, the stack pointer is a register that holds the memory address of the top of the stack. The stack is a region of memory used for temporary storage of data during program execution.

- The stack pointer keeps track of the current position on the stack.
- The operation is on LIFO (Last-in-first-out)
- The stack pointer is initially set to point to an empty location at the top of the stack.



SS = FOOOH
PA = [SS] x 10 + [SP]

Needs of stack:

1. Function calls: When a program calls a subroutine or a function, it needs to pass parameters and return addresses.
2. Local variables: The stack can be used to allocate memory for local variable within a subroutine.
3. Interrupt handling: When an interrupt occurs, the processor needs to temporarily suspend the current program execution and handle the interrupt rou
4. Stack-based operations: The stack also facilitates various stack-based operations, such as push and pop instructions, which make it easier to manipulate data and control program flow.

By utilizing stack, the 8086 processor can efficiently manage function calls, local variables, interrupts, and other operations, making it a fundamental component of program execution and memory management.

MID SEMESTER EXAMINATION

1. What do you mean by addressing modes? What are the different addressing modes supported by 8086? Explain each of them with suitable examples.

Ans: Different ways in which source operand is denoted in an instructions is called Addressing modes.

1. Immediate Addressing mode:
The Addressing mode in which the data operand is part of the instruction itself.
- constant value

Ex: MOV CX, 9430 H
MOV AL, EE H

2. Register Addressing mode:
It means that the register is the source of an operand for an instruction.

Ex: MOV CX, AX;
Both are registers. copies the content of 16-bit AX register move it to 16 bit CX register.

3. Direct Register Addressing mode:
- The Address mode in which the effective address of memory location is written directly in the instruction.

Ex: MOV AX, [192H] → offset → direct address
memory location

4. Register indirect Addressing mode:

This Addressing mode allows data to be addressed at any memory location through an offset [logical address] address held in any of the following like

1. Base pointer (BP) Destination index (DI)
   Base Register (BX) Source index (SI)

Ex: MOV AX, [BX];

Suppose the register BX contains 4895 H, then the contents 4895 H moved to AX.

5. Based Addressing mode:

The offset address [logical address] of the operand is given by the sum of contents of BX/BP register and 8/16 bit displacement.

Ex: mov BX, [BX + 04].

Same as Adding the displacement to the indirect Addressing mode.

6. Indexed Addressing mode:

The operand offset [logical address] address is found by adding the contents of source index (SI) / destination index (DI) register & 8 bit / 16 bit displacement.

Ex: mov BX, [SI + 16];

7. Based Indexed Addressing mode:

The offset address of operand that is computed by adding the base register to the content of indexed register

Ex: ADD CX, [AX + SI].

(2). What are the different instruction types of 8086 ? (8)

Ans:- 8086 MPMC supports 8 types of Instructions:

1. Data Transfer instructions
2. Arithmetic instructions
3. Bit manipulation instructions
4. String instructions
5. program execution transfer instruction
6. process control instruction
7. Iteration control instruction
8. Interrupt instruction.

1. Data Transfer Instructions:-

These instructions are used to transfer the data from the source operand to the destination operand.

- instruction to transfer a word : MOV, PUSH, POP, XCHG ... etc.
- instruction to transfer a address : LEA, LDS, LES ... etc...
- Instruction to transfer input/output port : IN, OUT.
- Instruction to transfer flag register : LAHF, SAHF, PUSHF, POP F.

2. Arithmetic Instructions:-

These instruction are used to perform arithmetic operation like addition, subtraction, multiplication, division .. etc..

- Instruction to perform addition : ADD, ADC, INC, AAA .. etc.
- Instruction to perform multiplication : MUL, IMUL, AAM .. etc.
- Instruction to perform substraction : SUB, SBB, DEC .. etc.
- Instruction to perform division : DIV, IDIV, AAD .. etc.

(3) Bit Manipulation Instruction:

These instruction are used to perform operations where data bits are involved i.e operations like logical shift .. etc...

(i) NOT, AND, XOR, TEST.
- SHL, SAL, SHR, SAR
- ROL, ROR, RCR, RCL.

(4) String Instructions:

String is a group of bytes/words and their memory is always allocated in a sequential order.

- REP
- REPE/REPZ
- REPNE/REPNZ
- MOVS/MOVSB/SMOVSW
- COMS/COMPSB/COMPSW
- INS/INSB/INSW
- OUTS/OUTSB/OUTSW

(5) program Execution Transfer instructions:

- Used to transfer/branch the instruction during an execution
- CALL, RET, JMP.
- JC, JE, JZ, JNC, JS, JO .. etc..

(6) processor control instructions.

used to control the processor action by setting/resetting the flag values.

- STC, CLC, CMC, STD, CLD, STI, CLI..

(7) Iteration control instruction:

Used to execute the given instruction for number of times.

- LOOPE/LOOPZ
- LOOPNE/LOOPNZ
- JCXZ.

(3) Draw the register organization of 8086 and explain typical applications of each register. (10)

Ans: The 8086 microprocessor has a complex register organization, consisting of several general-purpose registers, segment registers, index registers, and control registers. Here is a simplified representation of the register organization of the 8086:

1. General-Purpose Registers:

- AX: Accumulator register (16 bits)

Typical applications: Arithmetic operations, data storage, and data manipulation.

- BX: Base Register (16 bits)

Typical applications: Used as a pointer to data in memory or as an offset for addressing.

- CX: Count register (16 bits)

Typical applications:- Used as a counter in loop operations and shifts.

- DX: Data register (16 bits)

Typical applications: Used in I/O operations, as a temporary storage, or as a divisor in multiplication and division instructions.

Note: Each general-purpose register can be accessed as two 8-bit registers, such as AH/AL, BH/BL, CH/CL, and DH/DL.

(2) Segment Registers:

1 MB of memory segmented to 16 parts each of 64 KB.

- memory can be classified as 4 types of segments.

(i) code segment:

- Store codes
- stores address of its line of code.

② Data segment:

- stores Data / values
- stores address of its Data in memory segment.

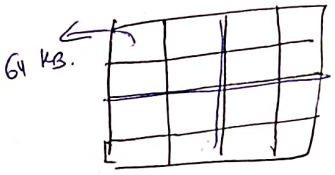
(iii) stack segment:

- used as stack LIFO memory
- used to store address of top of stack.

(iv) Extra segment: To store the extra data it needed

- store address of extract data.

- Each code, Data, stack and Extra segments can hold 16 bit of data
- But physical address is of 20 bits.



2^{16} = 64 KB memory location

③ pointer and index registers:

- pointer: We have three pointer.
  - (i) Instruction pointer (IP)
  - (ii) segment pointer (SP)
  - (iii) Base pointer (BP)

After each instruction IP will be increasing by 2 (not CS)

- CS x 10H + IP
- DS x 10H + BP (Base pointer)
- SS x 10H + SP (Segment pointer)

④ Indexed Registers:

We have two types of Index Registers.

(i) Source Index (SI):
(ii) Destination Index (DI):

(i) Source Index: store / locate / pointer source data

DS x 10H + SI

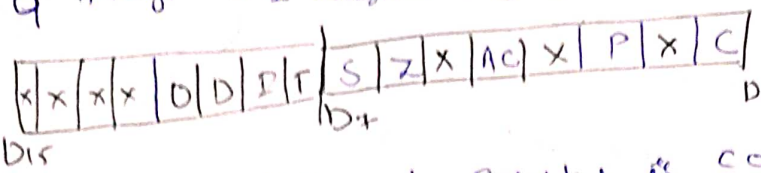
(ii) Destination Index (DI): pointer Destination data

DS x 10H + DI

4. Draw and discuss flag registers of 8086 in brief. (12)

Flags in 8086 :-

9 flags - 1 Register - 8 bits



- Sign Flag(S): MSB of Result is copied
  1 → neg Number
  0 → positive Number.

- Zero flag(Z):
  if resultant is 0. then flag is 1.

- (Auxiliary carry) AC = 1 : whenever carry from lower nibble to higher nibble (3rd bit to 4th bit {index starting from 0}) during BCD. ; used internally by processors.

  (1)
  0011 | 1001
  0101 | 1001

  0010

  AC = 1.

- parity flag (P): It check the no. of one's in result.

  even parity : No. of 1's in Result = 1
  odd parity : No. of 1's is odd then → 0.

- carry flag (C): carry / borrow if generated during arithemetic operation then it is '1' else if is '0'.

- Over flow (O): The flag is set to one, if an over flow occurs. ie if the result of signed operation is greater than the destination Register.

  MSB
  1 → negative
  0 → positive

- Resultant is > 15 bits, signed operation means MSB is reserved for sign → 0 = 1.

* control flags:

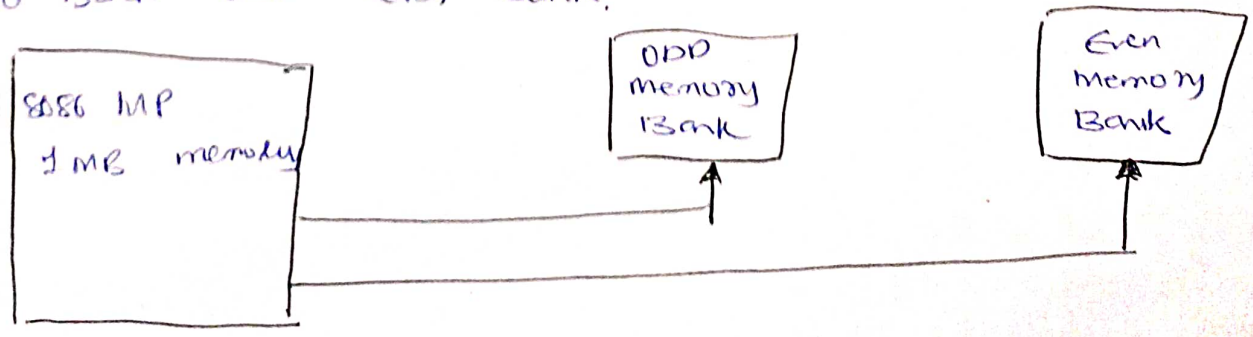
- Direction Flag (D):
  - D=0 : Auto increment mode: lower address to higher address
  - D=1 : Auto decrement mode: higher address to lower address.
- interrupt flag (I):
  - 1 : Enable all maskable interrupts
  - 0 : Disable all maskable interrupts
  - (No effect on non-maskable)
- Trap/Trace: enter into single step execution mode for debugging i.e executes 1 instruction and enter into single step ISR (interrupt service Routine).
  - 1 → CPU automatically generates.
  - 0 → No function is performed / No ISR.

(5). Explain the physical memory organization in an 8086 system:

There are 20 address lines on the 8086 microprocessor. This gives us 2^20 different memory locations. However the total size is 2^20 bytes, which is equal to 1 MB.

- Even the memory is byte-addressable yet the 8086 microprocessor can easily handle up to 16 bits of data at a time through its 16 data lines so, the to organize the memory efficiently, the entire memory in 8086 divided into two memory banks.

odd Bank and Even Bank.



OR

Concept of segmented memory:

(14)

The concept of segmented memory is a memory organization scheme used in the 8086 microprocessor and similar architecture. In segmented memory.

The Total addressable memory space is divided into segments, each identified by a segment address and an offset address.

In the 8086 architecture, a segment is a continous block of memory. is divided into multiple segments. and each segment is addressed using a 16-bit segment register and a 16-bit offset register.

physical Address = (Segment Reg $\times$ 10) + offset Register.

* Advantages of segmented memory:

1. Increased Addressable Memory
2. Flexibility in memory management
3. code and Data Separation
4. Efficient memory Access
5. Support for real mode and protected mode

6. Answer the following questions.

(i). What is the range of physical address if CS = FF59H?

PA = FF590 + 0000 = FF590H

Last PA = FF590 + FFFF = 0F58F

```
1111 1111 0101 1001 0000
     1111 1111 1111 1111
-----------------------
1 0000 1111 0101 1000 1111
```

(ii) Which registers are used to access the stack?

Ans: the base pointer and stack pointer registers

(iii). if control is transferred outside the current code segment, is it NEAR or FAR?

Ans: If the control is outside the current code segment it is FAR.

(iv). In unsigned multiplication of AX and BX. in which registers the product is placed in?

Ans: AX and DX

(v). To set all bits of an operand to 1, it could be ORed with 1.

(vi). To set all bits of an operand to 0, it could be ANDed with 0.

7. Explain the following instruction of 8086 with suitable examples

(i) MOV. - MOV is used to transfer the data from source to destination.

En: MOV AX, 5000H
MOV DS, AX
MOV AH, 2H.

(ii) PUSH: Push is used to add the data in stack.

Ex: MOV BX, 2000 H
PUSH BX.

(iii) POP F: pop is used to remove or delete the data in stack.

Ex: MOV BX, 2050H
PUSH BX
POP BX.

(iv) ADD: used to addition of destination and source.

Ex: ADD AL, 20H.
ADD AH, AL
ADD, AX, BX.

(v) MUL: used to multiple two instructions .. only the source

- If the source is of 8 bits, it will multiply with AL and result will be stored in AX register.
- If the source is of 16 bits, it will multiply with AX and result will be stored in DX-AX register.

Ex: MUL BL;
MUL BX;

(vi) DIV: Source.

- It divides a word(16) by byte (8) and a double word by word.
- If divisor is of 8 bits, it will divides with AX and result of quotient will be store in AL register and remainder stored in AH

Ex: DIV BL
DIV BX

(vii) CMP: Destination, Source.

- Used to comparison
- If Destination is greater than source, carry flag = 0
- If Destination is less than source, carry flag = 1.
- If Destination is equal to source, zero flag = 1.

Ex:- CMP AL, BL
CMP, CL, 20H
CMP BX, AX

(viii) DAA: Decimal Adjust after Addition

- In any microprocessor, we perform addition in Binary/Hexa-decimal But in real world, we understand things in decimal addition.
- So DAA instruction is used after addition to show adjustment which gives decimal additions.
- Add 06H with AL. if lower nibble > 9 or Auxiliary carry = 1
- Add 60H with AL, Higher nibble > 9 or carry = 1.

Ex:- ADD AL, BL
DAA

if AL = 20 H
BL = 40 H
AL. 60 H

(ix) AND: Destination, Source.

- Perform logical AND operation b/w destination and source.
- Destination can be register or memory location
- Source can be register or memory location

Ex:- AND AL, BL.

If AL = 43 H
BL = 56 H

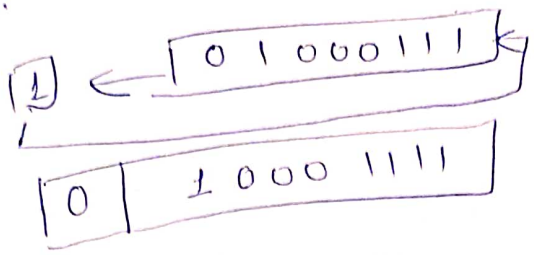
0100 0011 b
0011 0110 b
0000 0110 = 06 H

RCL: destination, count.

- This instruction performs Rotate bits left with carry.
- If count is greater than 1 then it has to be stored in CL register.

Eg: RCL, DH, 1

if DH = 47 H. = 0100 0111 b and carry = 1.



DH = 8FH.
CF = 0.

After execution RCL.

JMP Used to Jump

Eg:
```
JMP   MLT2P;
JMP   BX;
JMP   WORD PTR [BX];
JMP   DWORD PTR [32];
```

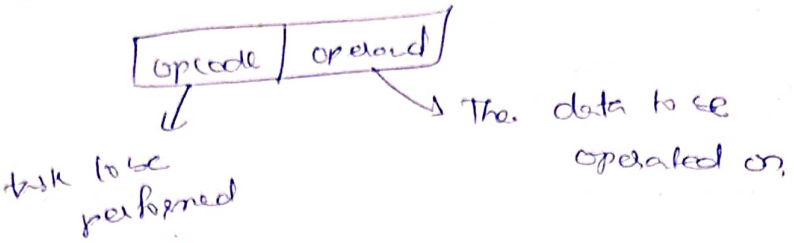
xii) **JB/JC/JNAE:** (Jump if above or equal / Jmp if not below/ Jump if no carry)

Eg:
```
CMP   AX, 437H;
JNB   NEXT;
```

INSTRUCTION FARMATS:-

Instruction: is a command given to the up to perform a specific task on specified data.

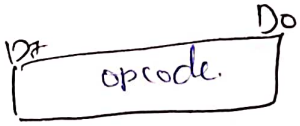
Each instruction has 2 - parts



- There are 6 - instruction formats for 8086:

(i) One byte instruction
(ii) Registers to registers
(iii) Register to / from memory with no displacement
iv) Register to / from memory with displacement
v) Immediate operand to Register
vi) Immediate operand to memory with displacement.

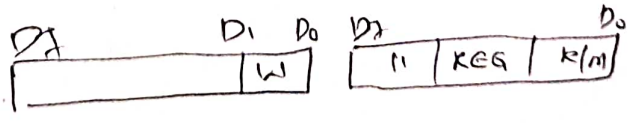
(i) one byte Instruction:-



- This format is only one byte long.
- It may have implied data (or) register operands.
- The least significant 3-bits of opcode represent the register operand, if any.

Eg: DAA

(ii) Registers to Registers:

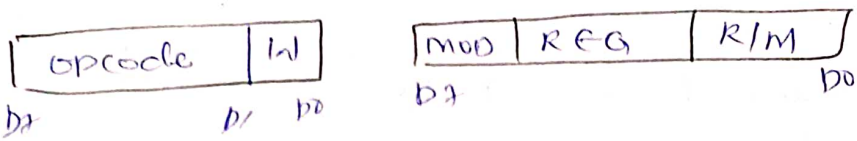


- This format is two bytes long.
- The first byte represents opcode and width of the operand specified by 'w' bit.

W=1 for 16-bit operand
W=0 for 8-bit operand

Eg: MOV AX, BX.

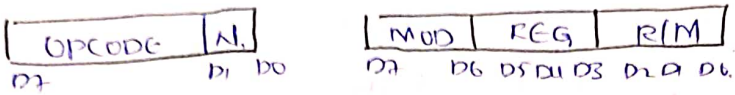
Registers to / from memory with no displacement.



- This formate is also two byte long, and similar to Register to Register formats except mod filed.

Ex: MOV AX, [SI]

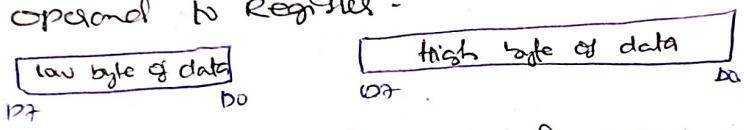
(iv). Register to/from memory with DISP (Displacement)



- This format contains one (or) two additional bytes for DISP along with 2-byte formate of Register to/from memory without displacement.

Ex: MOV AX, [SI + 2000H].

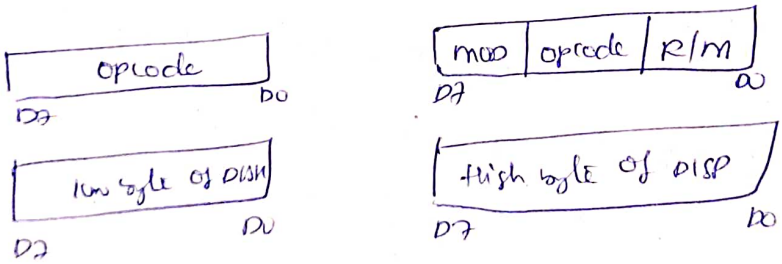
(v). Immediate operand to Registers :



- In this format, the first byte and 3-bits from the second byte are used to represent the opcode.

Ex: mov AX, 1234 H.

(vi). Immediate operand to memory with displacement ?



- This format is 5 or 6 bytes long.
- The first 2-bytes represent opcode, mod and R/M fields.
- The next 2-bytes represents displacements
- The last 2-byte represents Immediate data.

— THE END —