Unit - 1

Introduction of 8086 MP's

1.1) Register organisation of 8086 :-

In 8086 MP has Powerful set of Registers known as General Purpose registers and Special Purpose registers. All are 16-bit registers.

The General Purpose registers, can be used as either 8-bit registers (or) 16-bit registers which are used for holding data, Variable and immediate results. temporarily (or) for other Purpose like a Counter (or) for storing offset address for Particular address.

The Special Purpose registers used as segment registers, Pointers, index registers, (or) as offset Storage register for Particular addressing modes.

The Register's are Categorized in to 4-groups :-

a) General Data Registers
b) Segment Registers
c) Pointers and Index Registers
d) Flag Registers.

a) General Data Registers :-

The registers AX, BX, CX, & DX are the 16-bit general Purpose registers.

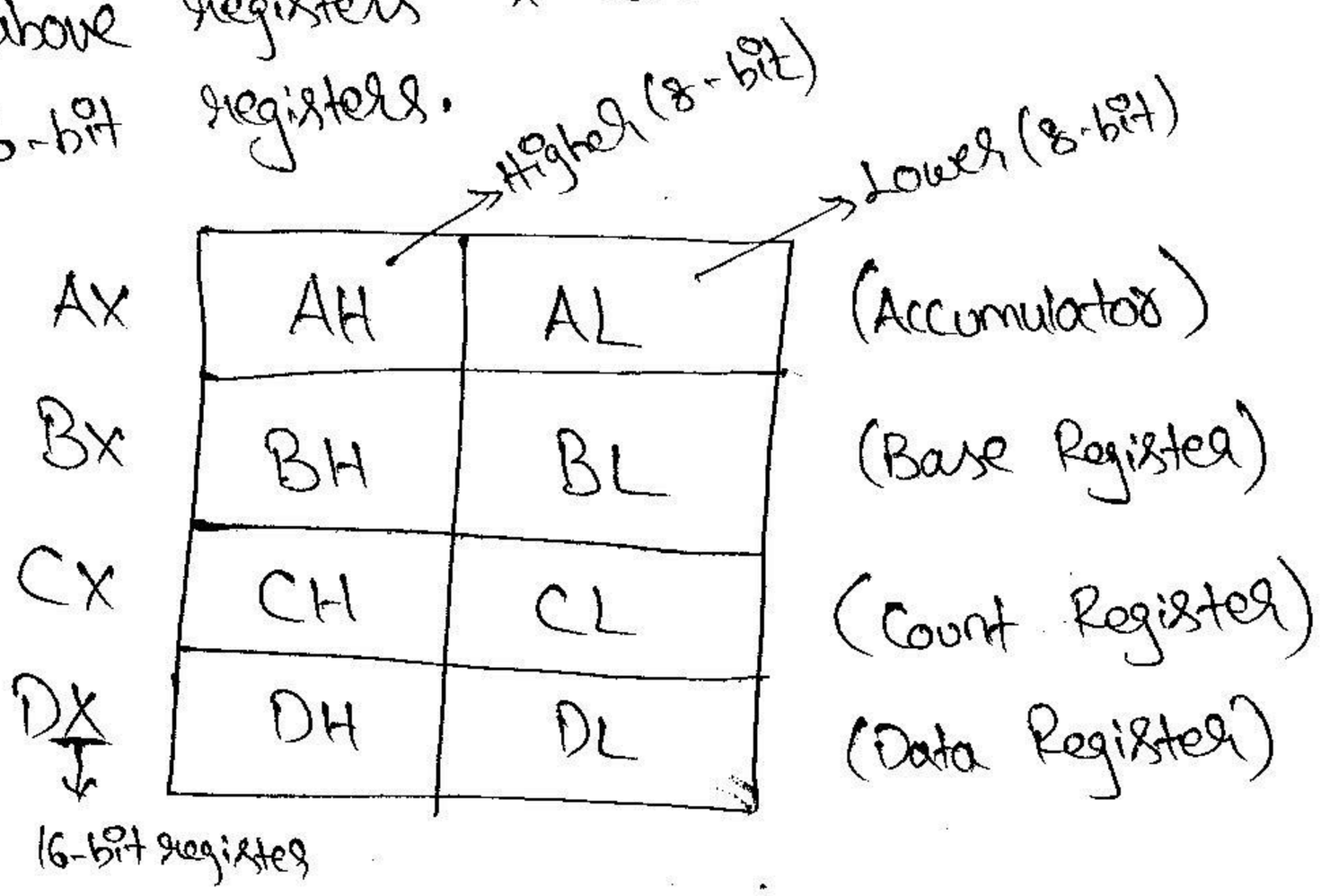
→ Ax is used as 16-bit accumulator with lower 8-bit of Ax designated as "AL" and higher 8-bit designated as "AH". This is the most important general Purpose register having multiple functions.

→ The "Bx" register is used as an offset storage for forming physical address in case of certain addressing modes.

→ The "Cx" register is used as default counter in case of string and loop instructions.

→ The "Dx" register is used as a implicit operand (or) destination in case of a few instructions.

From all above registers x-letter stands for specify the Complete 16-bit registers.

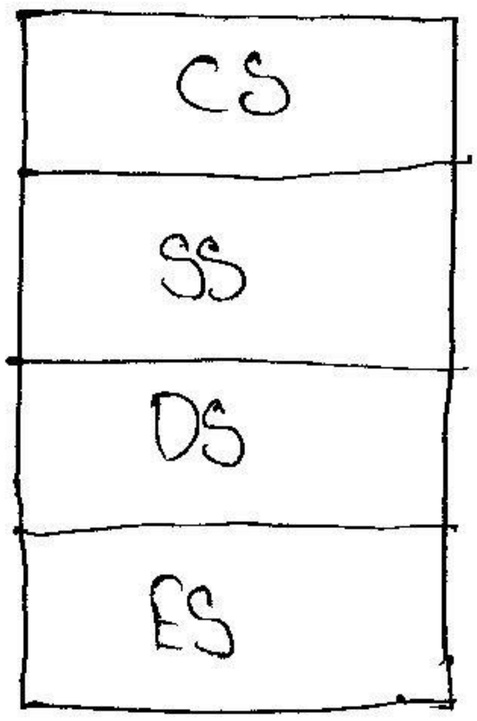


b) Segment Registers :-

The Complete 1 Megabyte memory which the 8086 address, is divided in to 16 logical segments. Each segment thus Contains 64 Kbytes of memory.

There are four (4) segment Registers.

a) Code segment
b) Stack segment
c) Data segment
d) Extra segment



Segment Registers

a) The Code Segment :- This Register is used for addressing a memory location in the Code Segment of the memory, where the executable Program is stored.

b) Data Segment :- This type of register points to the data segment of the memory where the data is resided.

c) Extra Segment :- Refers to a segment which essentially is another data segment of the memory. Thus Extra segment also contains data. ~~Segment~~,

d) Stack Segment :- Register is used for accessing Stack segment of memory i.e memory which is used to store stack data. The CPU uses the stack for temporarily storing important data.

Note:- All these segment registers are logical segments.

General info of Segment registers :-

While addressing any location in the memory bank, the physical address is calculated from two parts,

(i) Segment address (ii) Offset

(i) Segment address contains 16-bit segment base addresses related to different segments.

(ii) In any of pointer and index registers (or) BX may contain offset of the location to be addressed.

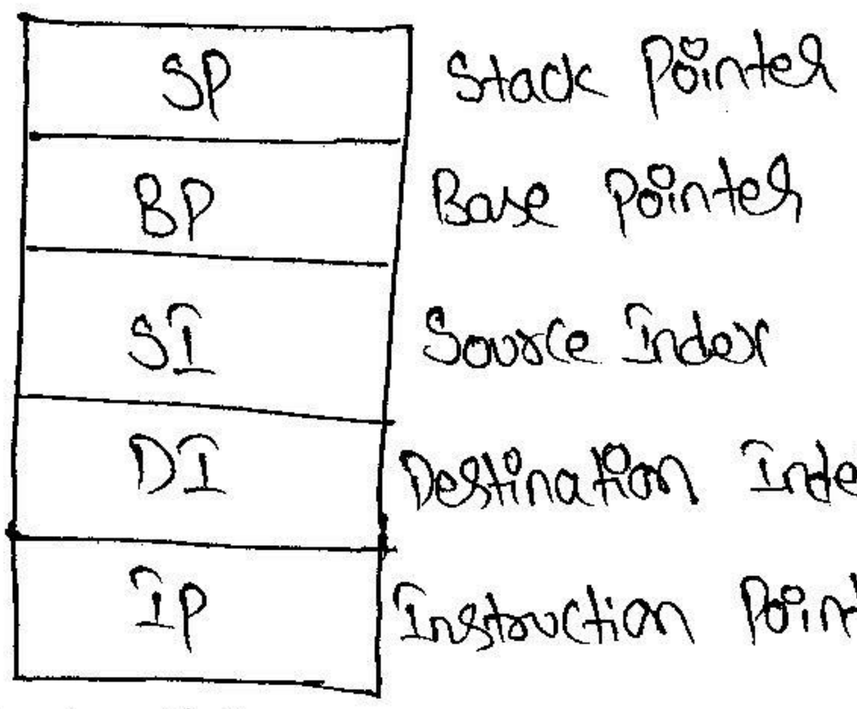
The advantage of this scheme is that instead of maintaining a 20-bit register for a physical address, the Processor just maintains two 16-bit registers which are within the word length capacity of the machine.

They may & may not be physically seperated. In other words, a single segment may require more than one memory chip (or) more than one segment may be accommodated in a single memory chip.

c) Pointers and Index Registers :-

The Pointer Contains offset within the particular segments. The Pointer IP, BP and SP usually Contain offset within the Code (IP) and Stack (BP & SP) segments.

(i) Stack Pointer :- used to hold the address of stack top. Stack top is upper most filled memory location in stack memory



(ii) Base pointer :- Acts as a memory Pointer to Stack Segment register. where BP is manly used to access any location directly in the Stack.

(iii) Source Index & Destination Index :- Acts as memory Pointers relative to Segment register DS. IP will take effective address of the data from SI and stores in DI. (Note :- Index registers used in string manipulation).

(iv) Instruction Pointer :- the offset address of the next instruction is contained in IP. That is IP points to next instruction to be fetched from Code Segment.

d) Flag Registers :-

The 8086 flag registers Contents indicates the results of computation in the ALU. It Contains some flag bits to Control the CPU operations..

2) Architecture of 8086 Micro Processor :-

→ 8086 Provides no:- of improvements over 8085 MP.

→ The 8086 MP supports 16-bit ALU (Arithematic Logical unit), a set of 16-bit registers, and Provides Segmented registers, memory addressing Capability, a rich instruction set, Powerful interrupt architecture, fetched instruction queue for overlapped fetching and execution etc;

→ The 8086 MP Architecture divided in two parts.

(a) Bus Interface unit (BIU)

(b) Execution unit (EU).

→ The 8086 MP a 16-bit MP with 16-bit internal and external data bus, with 20-address lines it Can access

2^{20} = 1MB memory.

(a) Bus Interface Unit (BIU) :-

It Contains the circuit for physical address Calculations and a Predecoding instructions byte queue (6-bytes long). this BIU takes the system's bus signal available for External interfacing of the devices, which this unit is responsible for Establishing Communication with External devices & Peripherals including memory via bus.

(b) Execution Unit (EU):- Executes the Peeviously decoded instructions Concurrently. the BIU along with Execution unit (EU) thus forms a pipeline.

(*) → The BIU thus manages the Complete interface of execution unit with memory and I/O devices, of Course, under Contool of timing and Contool unit.

(*) → The EU Contains the register set of 8086 except segment register and Instruction pointer (IP). It has.

(*) 16-bit ALU, able to Peyform aoithematic and logical operations.

(*) 16-bit Flag register Reflects the result of execution by ALU.

(*) the decoding unit decodes the opcode byte issued from the instruction byte.

(*) the timing and Contool unit devices the neccessaly Contol signals to execute the instruction opCode received from the queue, depending upon the information made available by decoding circuit.

(*) Thus Execution unit may Pass the results to the bus interface unit for staoing them in memory.

Working Process:-

(a) Bus Interface Unit :- It Contains following

(i) the circuit for Physical address Calculations (Adder)
(ii) Pre decoding instruction byte Queue
(iii) Segment registers
(iv) Instruction pointers.

The Main function of BIU are

*) the BIU handles all interfaces with external bus and generates External memory and I/o address.

*) the BIU reads data from the memory and Ports and write data in to memory and Ports.

*) It fetches instructions Codes from the memory and keeps them upto a 6-byte instruction queue.

*) It fills the queue whenever the bus is idle and 2-bytes of the queue are vacant.

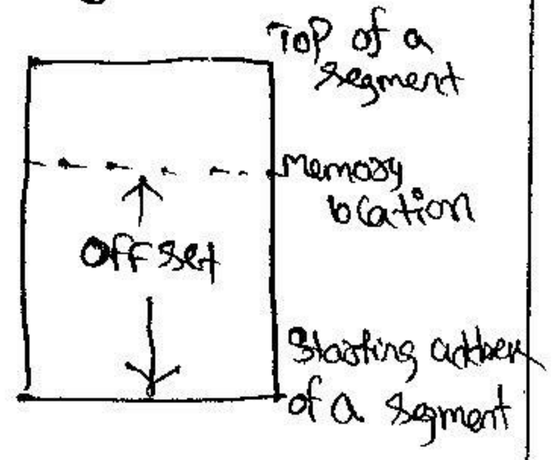
*) In case of Jump and CALL instructions the BIU dumps the queue and then starts reloading it from new address.

i) Adder :- The 8086 MP address a segmented memory. The Complete physical address which is 20 bits long is generated using segment and offset registers,

for generating a Physical address from Contents of these two registers, the Contents of segment registers also called as segment address is shifted left bit wise four times and to this result.

Content of offset register is also called offset Address add to produce 20-bit Physical address.

Eg:- If segment address is 1005H and Offset address is 5555H then Physical address is Calculated as



Segment address → 1005 H
Offset address → 5555 H

where

Segment address → 1005H → 0001 0000 0000 0101

Shifted by 4-bit Position → 0001 0000 0000 0101 0000 } +

+ 0101 0101 0101 0101

0001 0001 0101 1010 0101

1 5 5 A 5

*) The segment register indicates the base address of Particular Segment
*) The offset register indicates the distance of required memory location in segment from base of the address.
*) The segment address by segment value "1005H" can have offset from "0000H" to "FFFFH" within it, i.e maximum 64K locations. may be accomodated in the segment.
* Since the offset is a 16-bit number each segment can have a maximum 64K locations.

The Bus interface unit has a seperate adder to perform this procedure for obtaining Physical address while addressing memory.

*) The segment address value is to be taken from an appropriate segment register depending upon whether code, data (or) stack are to be accessed, while the offset may be Content of IP, BX, DI, SP, BP (or) an immediate 16-bit value depending on addressing mode.

(ii) Instruction Queue :-

In 8085 MP, once the OP-Code is fetched and decoded, the External bus remaining free for some time, while the Processor internally executes the instruction. This time slot is utilized in 8086 to achieve the overlapped fetch and execution cycles, while the fetched instruction is executed internally the External bus is used to fetch the machine code of next instruction and arrange it in a queue called as Pre decoded instruction byte Queue.

Queue is a 6-bytes long, First in - first out Structure the instruction from the queue are taken for decoding sequentially once a byte is decoded, the queue is re-arranged by pushing it out and the queue status is checked for the possibility of the next OP-Code fetch cycle.

iii) Segment Registers :-

*) The Code Segment Contains instructions Codes of a Program.
*) the Stack segment is that segment of memory which is used to store stack data.
*) the Extra segment Contains the destination of source data of certain string instruction.
*) the Data segment Contains Program defined data, Variables and Constants.

iv) Instruction Pointer :-

It acts as a Program Counter. It holds 16-bit offset Pointing to the next instruction Code with in the Current 64KB Code segment.

(b) Execution unit :-(EU)

*) → The EU receives Prefetched instructions from the queue decodes them and then Executes them. The queue acts as first in first out for EU.

*) → If the queue is Empty, the EU waits till the queue gets at least one byte. Such situation arises after the execution of a Jump instruction.

*) → The EU Contains the register set of 8086 Except Segment registers and IP.

*) → The EU has 8 GPR's (General Purpose registers) Called as AH, AL, BH, BL, CH, CL, DH, DL and used individually for temporary Storage of 8-bit data.

The advantage of using internal register for temporary storage is that, since the data is already in the EU, it can be accessed much more quickly than it could be accessed in External memory.

*) → It has ALU (Arthematic Logical unit) of 16-bit able to Perform arithematic and logical operations.

*) → The 16-bit flag register reflects the result of Execution by ALU.

*) → the decoding unit decodes the opcode byte issued from the instruction byte queue.

*) → the timing and Control unit derives the necessary Control Signals to execute the instruction op-code received from the queue.

*) → the EU may Pass the result to the BIU for Storing them in memory.

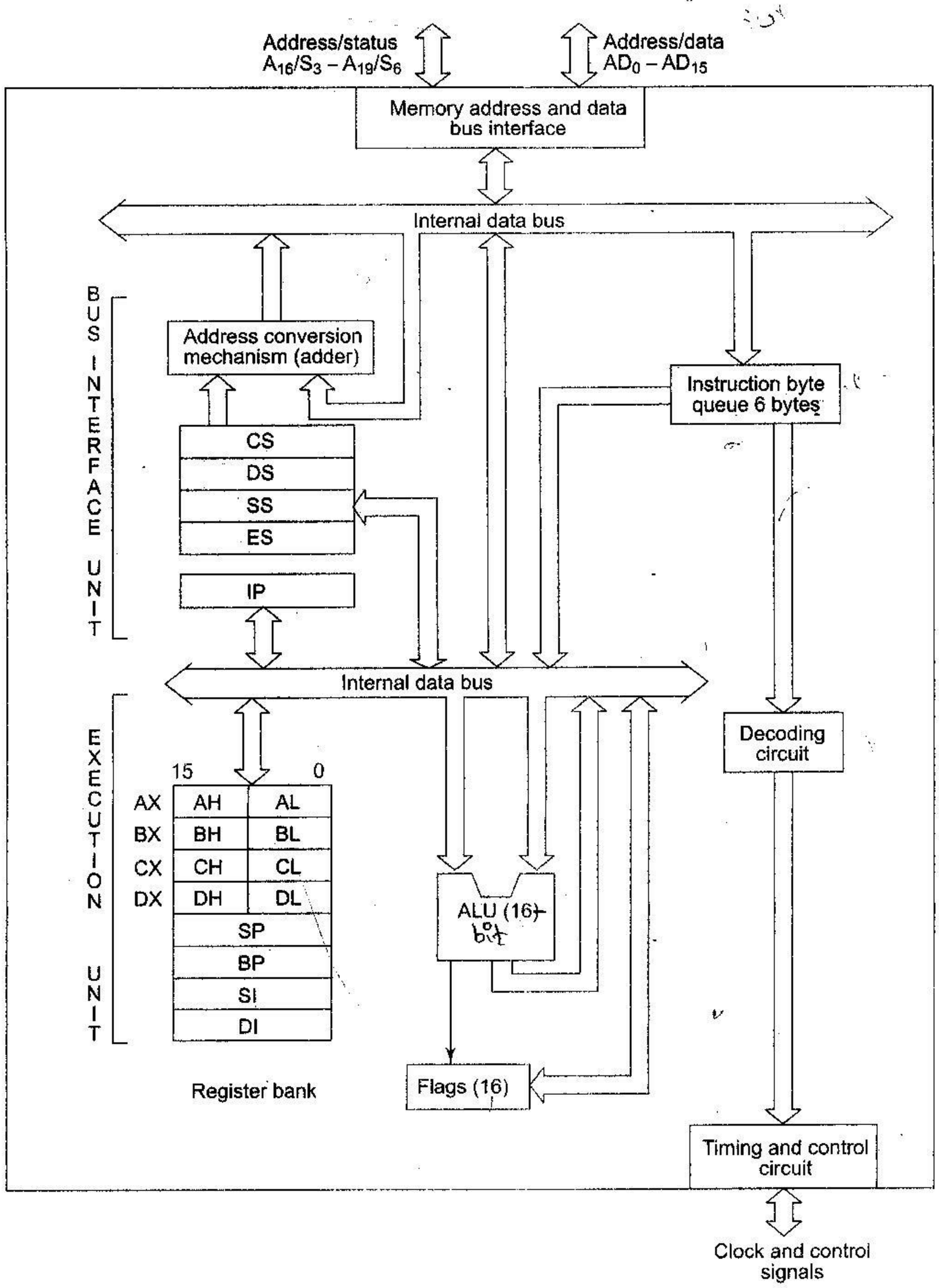


Fig. 1.2 *8086 Architecture*

The complete architecture of 8086 can be divided into two parts (a) Bus Interface Unit (BIU) and (b) Execution Unit (EU). *The bus interface unit contains the circuit for physical address calculations and*

3) Status Registers (or) flag Registers :- (16-bit)

*) → A flag is a flip-flop which indicates some condition produced by the execution of an instruction. The 8086 µp has 16-bit flag register it is also called Status Register (or) Program Status word (PSW).

*) → In 16-bit flag Registers 8086 µp has 9-flags and remaining 7-flags bit positions of status registers are undefined (X).

*) → In 9-bit of flags 6-flags are conditional flags they are set (or) Reset by the Processor depending on the result of some arithmatic & logical operations.

*) → The 6-conditional flags are CF, PF, AF, ZF, SF, OF.

*) The remaining 3-flags are control flags these are set (or) reset by programmer as required by certain instructions in a program.

*) → The control flags are TF, DF, IF. 8085 µp compatabile Flags

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	OF	DF	IF	TF	SF	ZF	X	AF	X	PF	X	CF

X → Undefined (7-bits of flags positions of Status Registers)

6-Conditional Flags

CF → Carry Flag
PF → Parity Flag
AF → Auxilary Flag
ZF → Zero Flag
SF → Sign Flag
OF → over flow Flag

3-Control flags.

DF → Direction Flag
IF → Interrupt Enable flag
TF → Trap flag

(i) Conditional flags :-

a) Carry flag (CF) :- This flag is set whenever a Carry/borrow occurs during arithematic operations such as addition (or) Substraction otherwise it is reset.

b) Parity flag (PF) :- This flag is set to '1' if the lower byte of the result Contains even no: of 1's.
If the lower byte of result Contains odd no: then Parity flag is set as zero (0);

c) Auxiliary Carry flag (ACF) :- This is set if there is a Carry from the lowest nibble

Eg:- i.e; Bit three during addition of borrow for lowest nibble (4-bit)

Eg:- ie; Bit three during Substraction into (two bit).

d) Sign flag :- (SF) :- This flag is set when the result of any Computation is negative.
If the flag is set if MSB of the result is zero.
It is set to zero if the MSB of result is zero

e) Zero flag :- (ZF) :- This flag is set if the result of an arithematic (or) logic operation is zero. In Case of non zero result this flag is set zero.

f) Overflow flag (OF) :- This flag is set if an overflow occurs ie if the result of a signed operation is large enough to be accomodated in a destination register overflow flag has no significance in unsigned arithematic operation.

ii) Control flags :-

a) Trap flag (TF) :- It is used for single step control. If this flag is set the Processor enters the single step execution mode. It allows user to execute one instruction of a Program at a time for debugging.

b) Interrupt Flag (IF) :- It is a interrupt Enable/disable flag. If this is set the maskable interrupts are recognized by the CPU, otherwise they are ignored.

c) Direction flag (DF) :- It is used by string manipulation instructions. If this flag bit is '0' the string is Processed begining from the lowest address to the highest address i.e auto incrementing mode otherwise the string is Processed from the highest address towards lowest address i.e auto decrementing mode.

4) Memory Segmentation :-

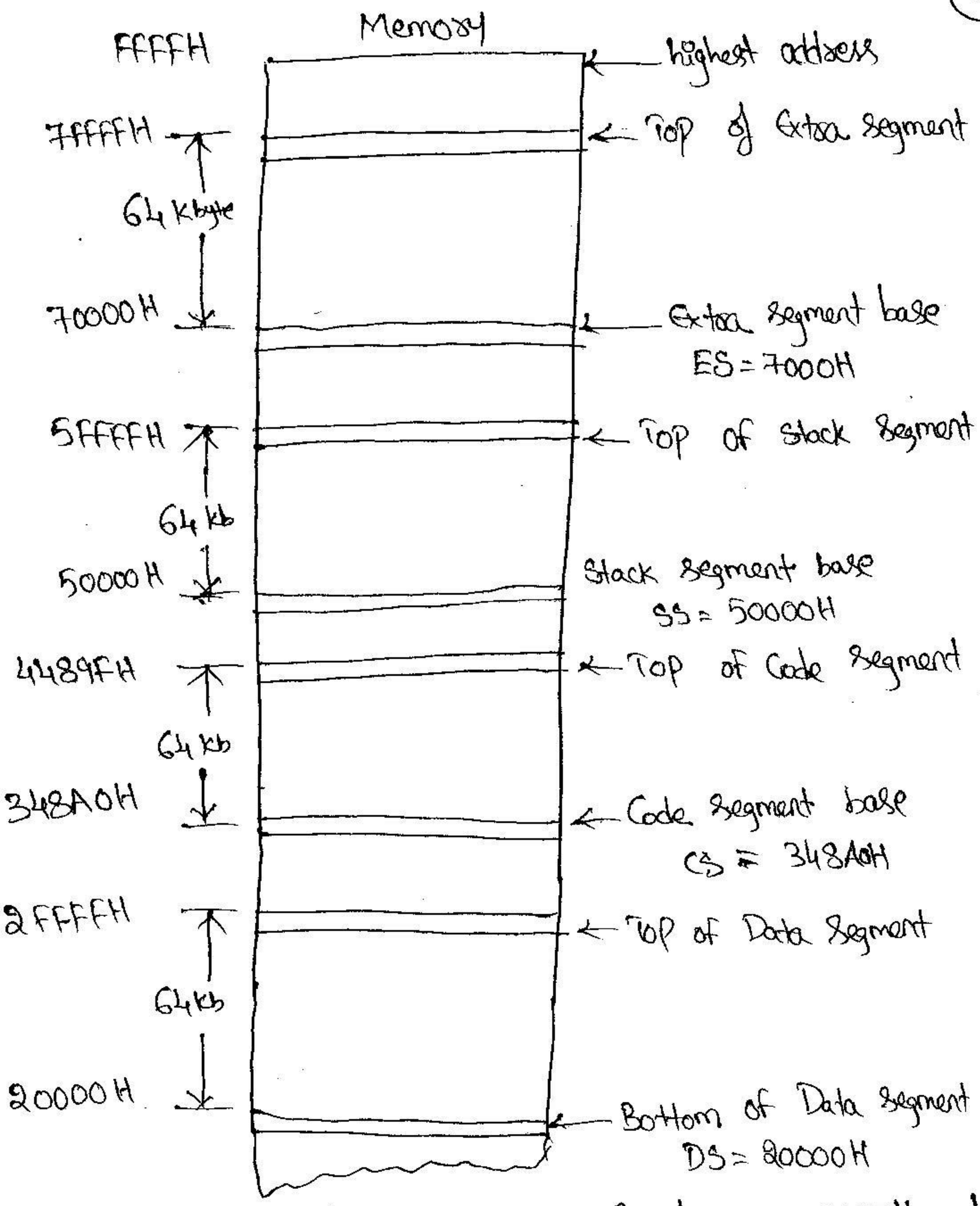
Memory is Physically a linear sequence of address, each address holding a single byte. The lowest memory address is '0', the next address is address '1' and so on until the highest memory address is reached.

The highest memory address for the 8086 MP addres FFFFH

In 8086 available memory space is divided into "chunks" called segments, such a memory is known as segment memory.

The 8086 is able to address 1MBytes of Physical memory.

The Complete 1MB of memory can be divided into 16 segments, Each of 64 Kbytes Size.



*) The address of the segment may be assigned as 0000H to FFFFH respectively.

*) The offset address values are from 0000H to FFFFH respectively.

*) The above Case segments are non-overlaped segments.

*) In Some Cases segments may be overlapping to address a specific memory location within a segment an offset address is needed.

The Main advantages of Segmented memory scheme are

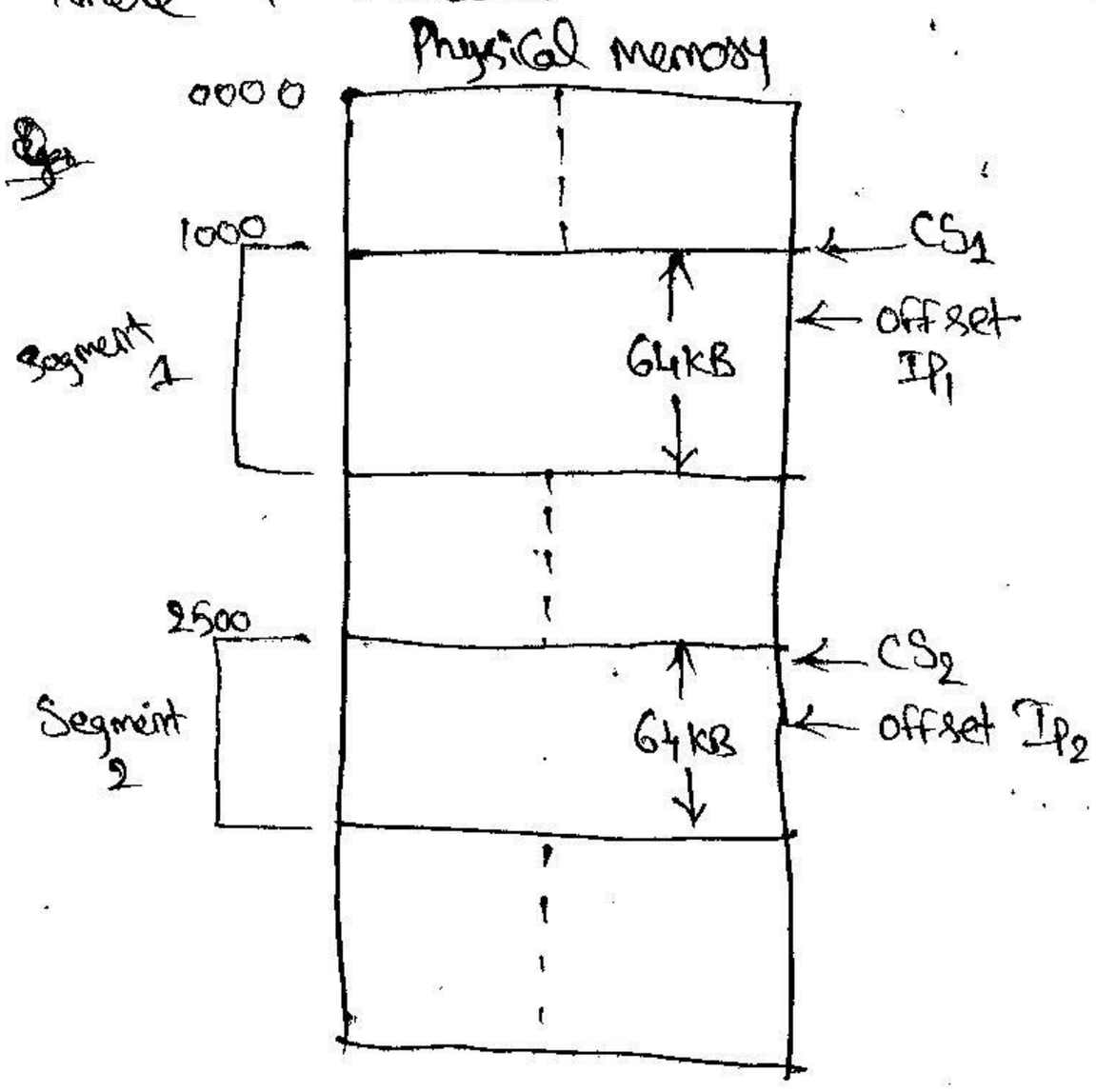
(i) Allows the memory Capacity to be 1Mbyte although the actual address to be handled are of 16-bit size.

2) Allows the placing of code, data and stack portions of the same program in different parts (segment) of memory, for data and code operation.

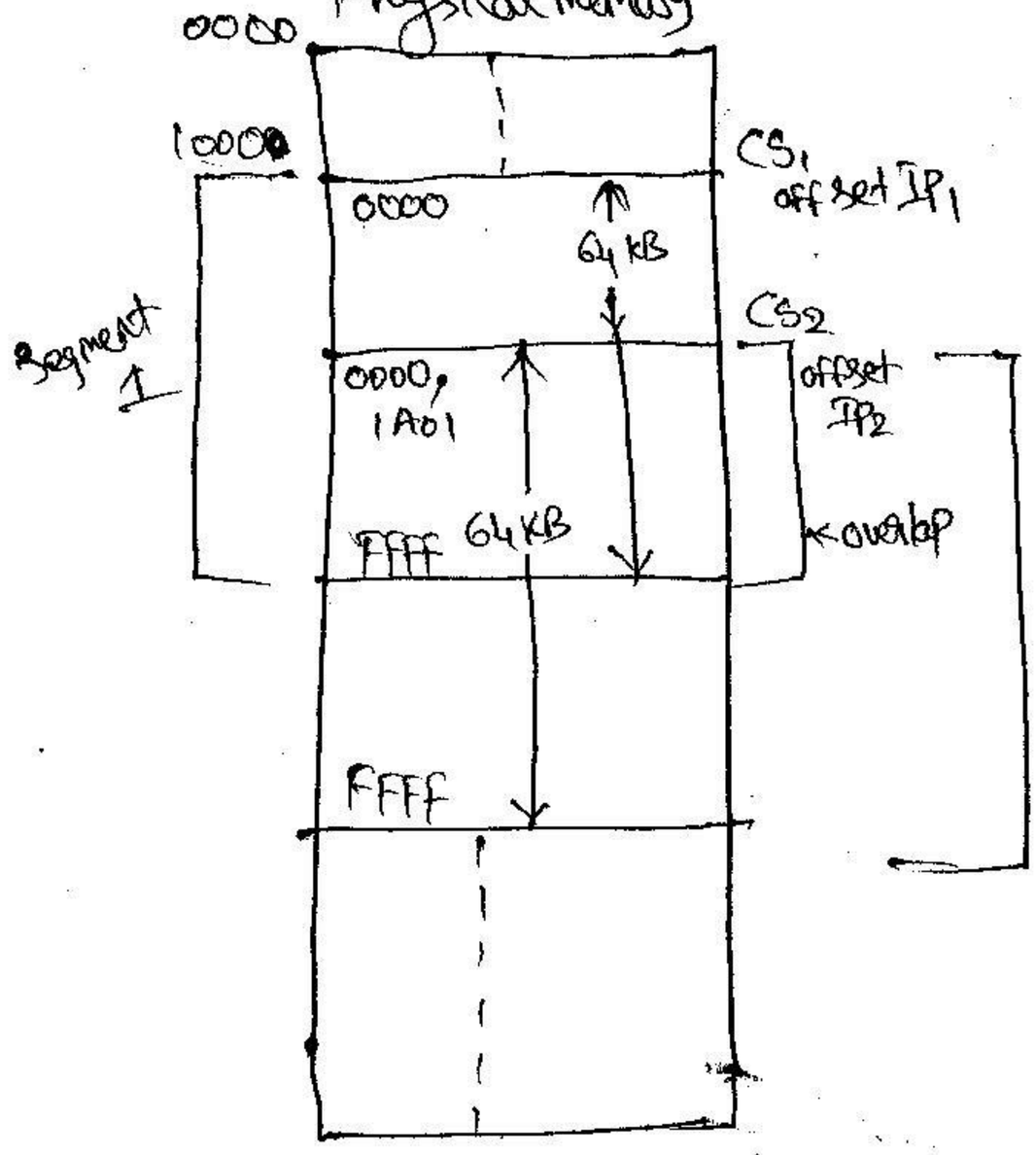
3) Permits a program and/or its data to be put into different areas of memory each time the program is executed i.e provision for relocation is done.

Calculations of Physical address :-

In overlapped area location Physical address $= CS_1 + IP_1 = CS_2 + IP_2$ where '+' indicates the procedure of Physical address formation.



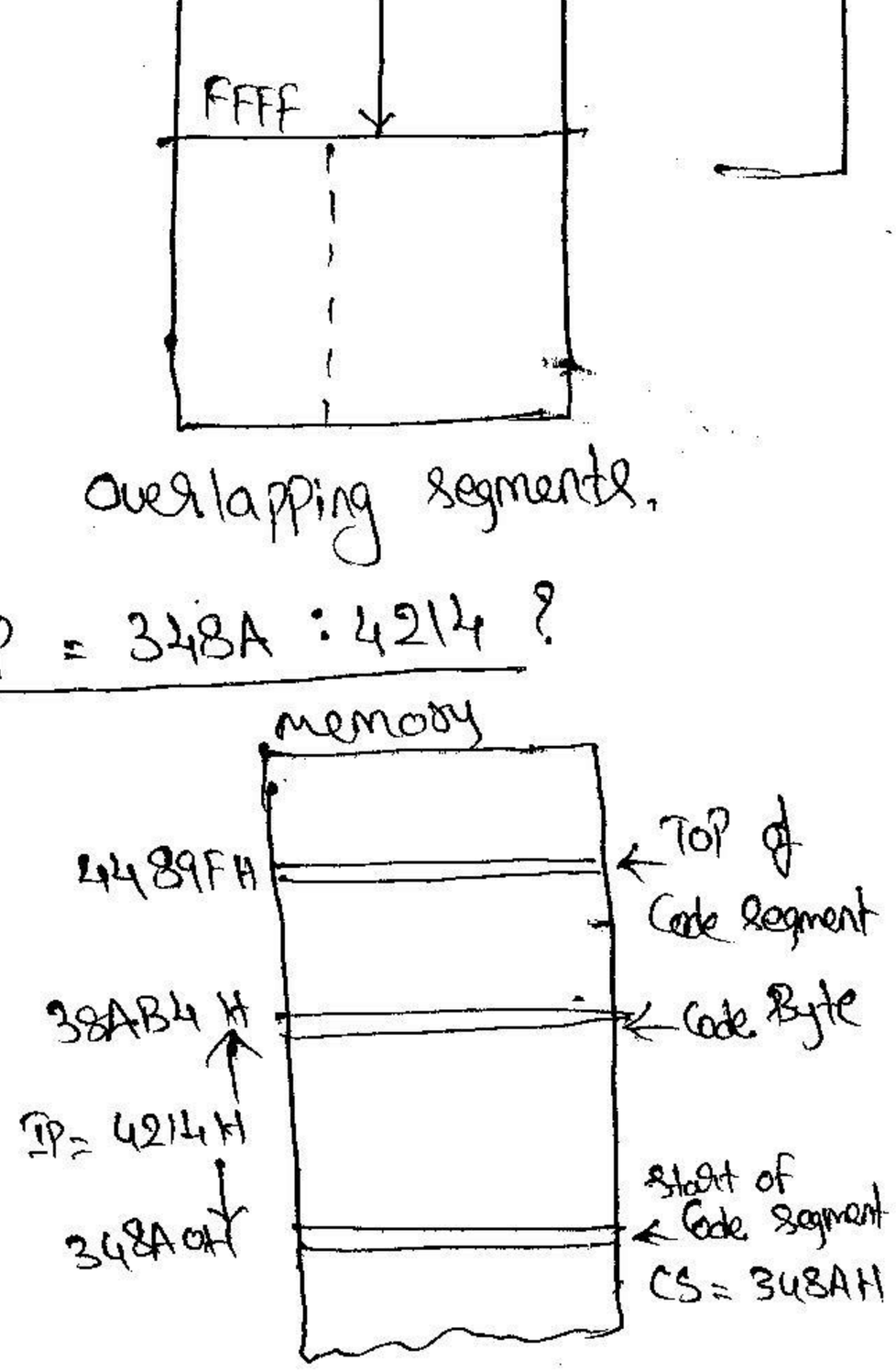
Non overlapping segments



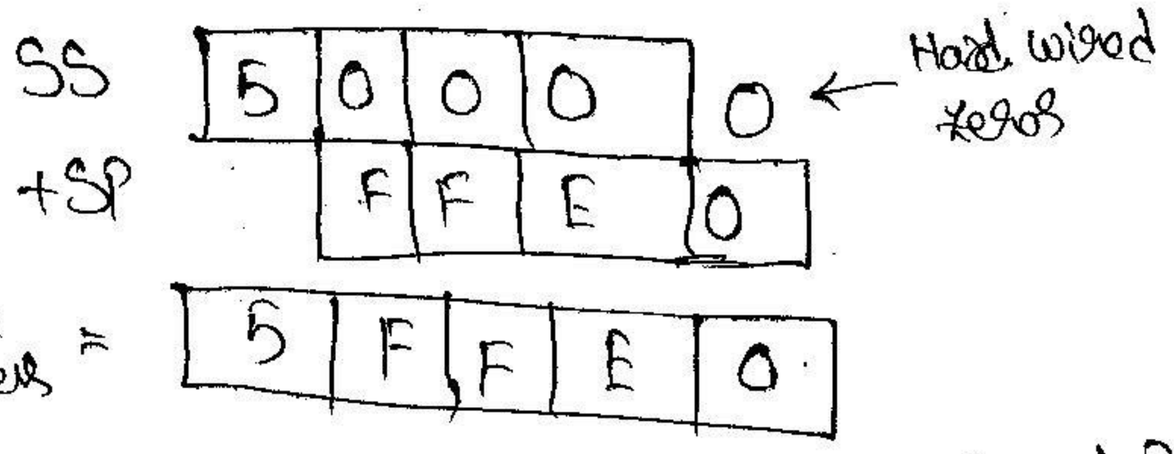
overlapping segments.

Eg:- Calculate Physical address of CS : IP = 348A : 4214 ?

CS	3	4	8	A	0 ← Hard wired zeros
+ IP		4	2	1	4
Physical Address =	3	8	A	B	4



Eg:- Calculate the Physical address of SS:SP = 5000 : FFE0 ?



Eg:- If an absolute address of the type 6A3D9 is given express it in the form of CS:IP ?

Let us assume that CS Contains 6000H the offset address is Calculated as

Offset = 6A3D9H − 6000 X 10H = A3D9H

Hence CS:IP = 6000 : A3D9 //.

5) Addressing Modes:-

Each instruction Performs an operation on the specified data (operand). Hence an operand must be specified for an instruction to be executed. The operand may be reside in accumulator, in a general purpose register (s) in a memory location.

*) The way by which an operand is specified for an instruction is Called "Addressing Mode".

A/c to the flow of instruction Execution the Constructions may be Categorized as

(i) Sequential Control flow instructions.

(ii) Control transfer instructions.

ADDRESSING MODES :-

- An instruction performs an operation on the specified data is known as **operand**.
- Addressing modes indicates a way of locating data or operand which is accessed by an instruction that to be executed.

categories of addressing modes:

1. sequential control flow
2. control transfer instruction

1. Sequential control flow instruction:-

which offers execution, control transfer to the next instruction appearing immediately after it in the program.

Eg:- arithmetic, logical , data transfer, processor control

2. Control transfer instruction

Instructions transfer control to the some predefined address or the address some how specified in the instructions ,after their execution.

Eg:- INT,CALL,RET,& JUMP

Addressing modes in sequential control flow

There are 8 addressing modes for sequential control flow

- In this 2-addressing modes are provided for instructions which operate on Registers (or) immediate operands
- The reaming 6- addressing modes are for specifying an operand stored in the memory

Types of Addressing modes for Register (or) operand instructions

1. Immediate Addressing Mode
2. Direct Addressing Mode
3. Register Addressing Mode
4. Register Indirect Addressing Mode
5. Indexed Addressing Mode
6. Register Relative Addressing Mode
7. Based Indexed Addressing Mode
8. Relative Based Indexed Addressing Modes

Register addressing mode :

The operand to be accessed is specified as residing in an internal register of 8086.

Fig. below shows internal registers, any one can be used as a source or destination operand, however only the data registers can be accessed as either a byte or word.

REGISTER	OPERAND	SIZES
	BYTE(REG-8)	WORD (reg-16)
ACCUMULATOR	AL, AH	AX
BASE	BL,BH	BX
COUNT	CL,CH	CX
DATA	DL,DH	DX
STACK POINTER	-	SP
BASE POINTER	-	BP

SOURCE INDEX	-	SI
DESTINATION INDEX	-	SI
CODE SEGMENT	-	CS
DATA SEGMENT	-	DS
STACK SEGMENT	-	SS
EXTRA SEGMENT	-	ES

1. **Immediate Addressing Mode**

The data is part of instruction itself and is available in successive bytes of instruction code.

Example: -

MOV AL,26H //***** copies 8-bit data 26H in to AL register
(since AL is 8-bit register)

Example:-

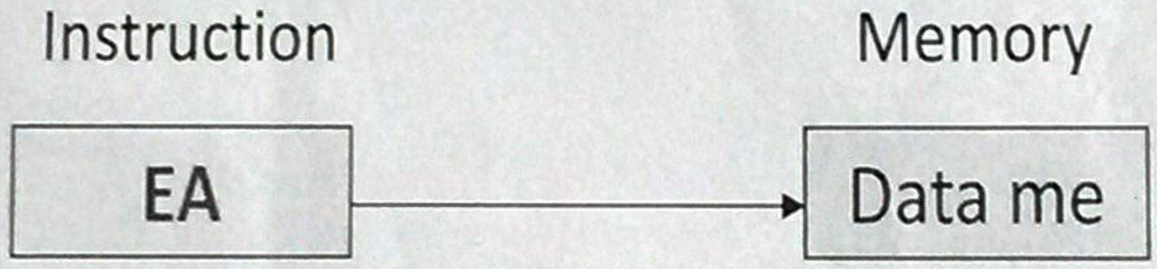
MOV AX, 4653H ///*****copies 16-bit data 4653H in to ax register pair
(Since AX is pair of two 8-bit AL & AH registers)

2. Direct Addressing Mode

The instruction operands specifies in memory address (offset) where data is located.

Example:-

MOV AX, [5000] ; 5000 is a Effective Address [offset] directly written in the address.



Data resides in a memory location in data segment(DS), whose Effective Address is computed using **5000H** as the **offset address &** content **DS** as segment address

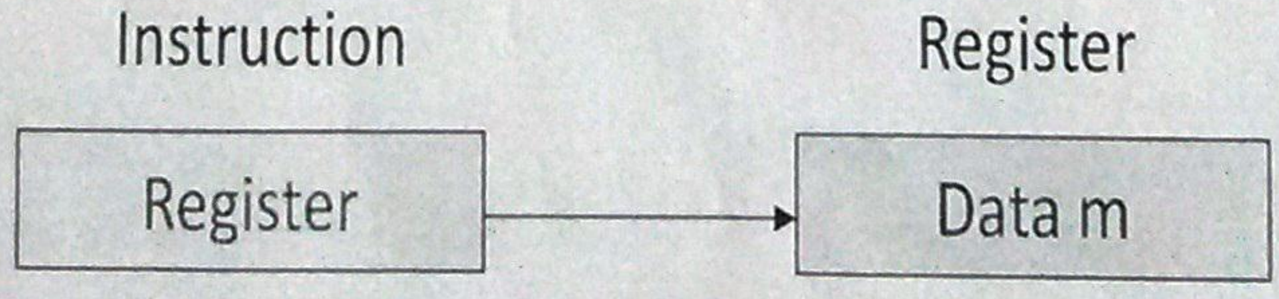
Effective Address {EA} here is = **10H * DS+ 5000H**

3. Register Addressing Mode

Refers the data in a register (or) in a register pair . All the registers, except IP, may be used in this mode.

Eg: -

MOV AX, CX //****copies 16-bit content of CX into AX register pair
MOV AL, CL //**** copies 8-bit content of CL into AL register



4.Register Indirect Addressing Mode

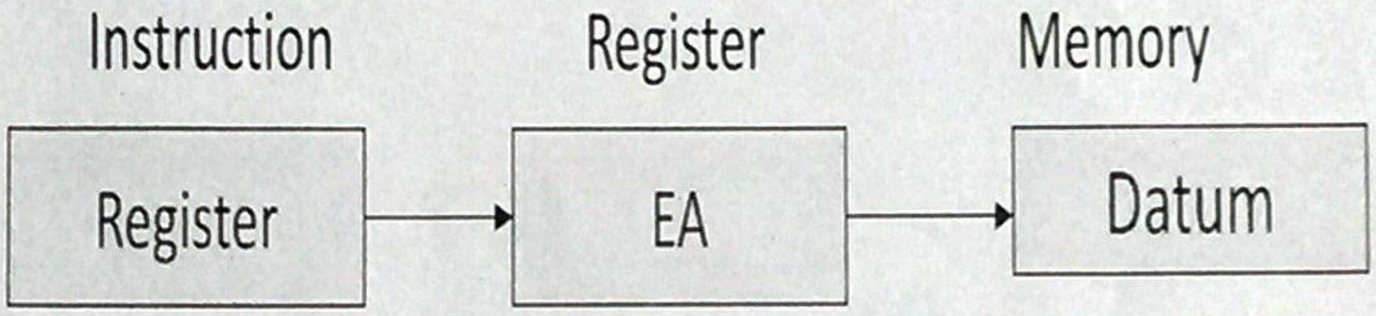
The address of the memory location which contain data or operand is determined as indirect way. Using the offset register, this mode of addressing is known as register indirect mode.

In this mode the offset address is either BX (or) SI (or) DI register.
The default segment is either DS (or) ES .

Eg:

MOV AX, [BX] //** data present in memory location in DS whose offset address is i BX.

Effective address = **10H*DS+[BX]**



5. Indexed Addressing Mode

- In this addressing mode, offset of the operand is stores in one of the index registers. DS is the default segment for index register SI and DI . in this case DS and ES are default segment for SI and DI respectively .
- This mode is special case of the above discussed register indirect addressing mode.

Eg:-

MOV AX, [SI]

Here data is available at an offset address stored in SI in DS. The Effective Address, in this case is computed as

EA=10H*DS+[SI]

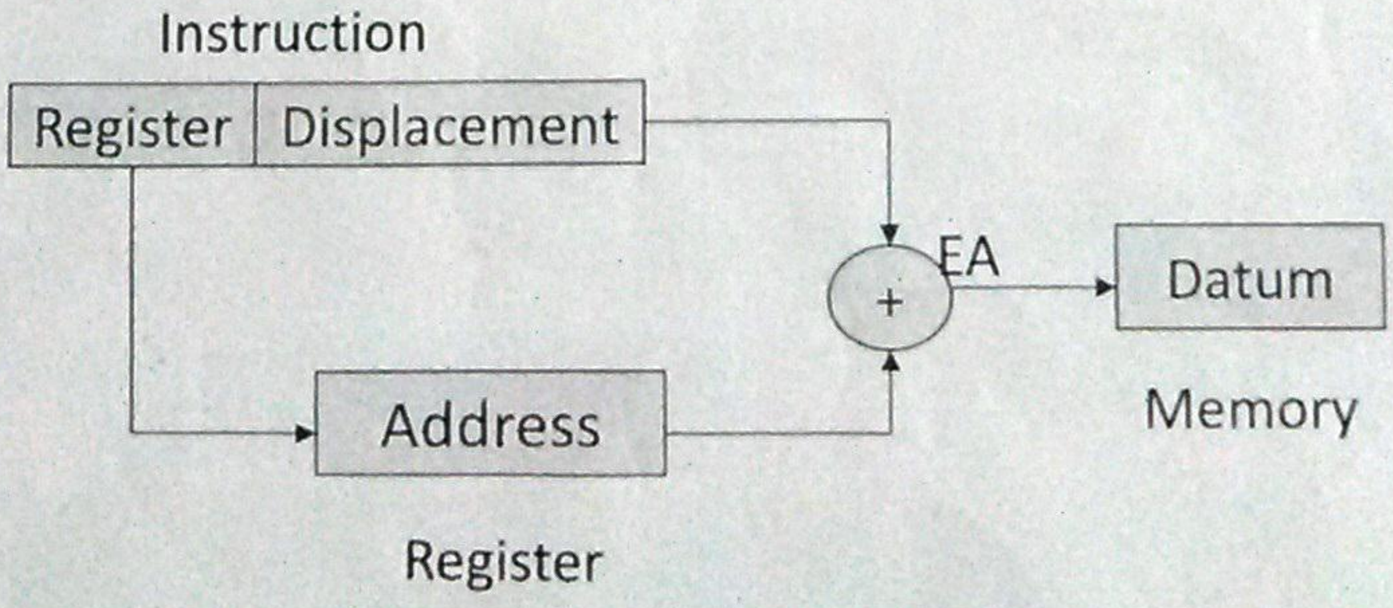
6. Register Relative Addressing Mode

- In this addressing mode , data is available at EA formed by adding 8-bit (or) 16-bit displacement With content of any of registers BX,BP,SI and DI in the default (either DS or ES) segment.

Eg:-

MOV AX, 50H[BX]

Here **Effective Address = 10H*DS+50H+[BX]**



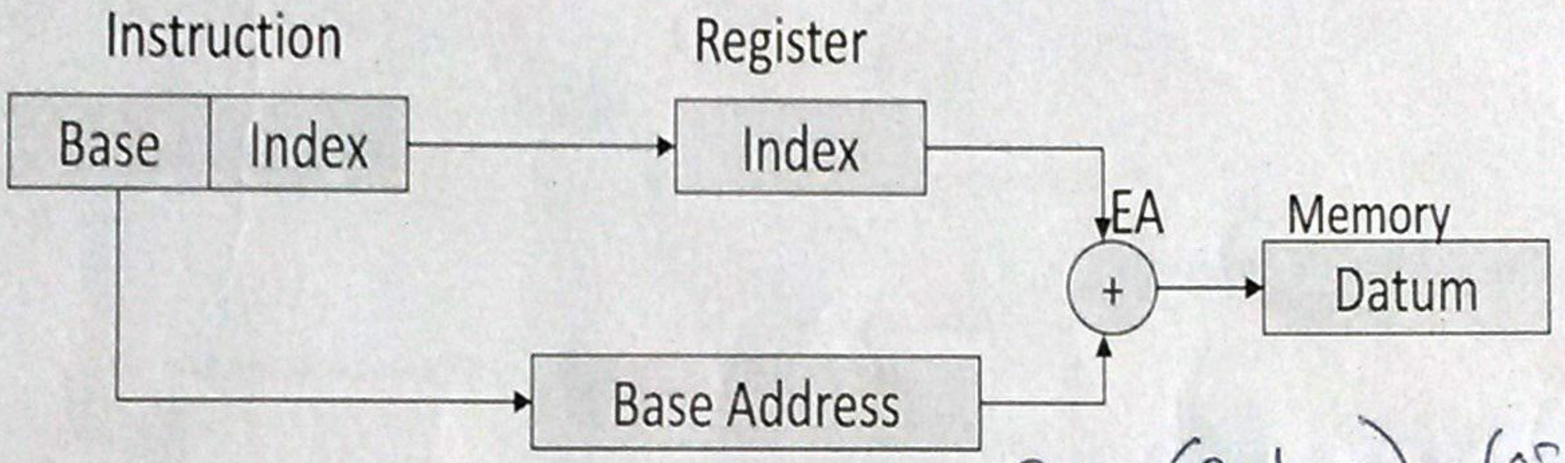
7.Based Indexed Addressing Mode

- **Based Addressing Mode:**

The effective address will be the contents of base register (or) base pointer register.

- **Based Indexed Addressing Mode:**
  - This is same as register indirect
  - The effective address of the operand is formed by adding contents of base register to the index registers.
  - DS and ES are default segmen s,
  - SI and DI are used as index registers and BX and BP are used as base registe s.

Example: MOV AX, [BX][SI]

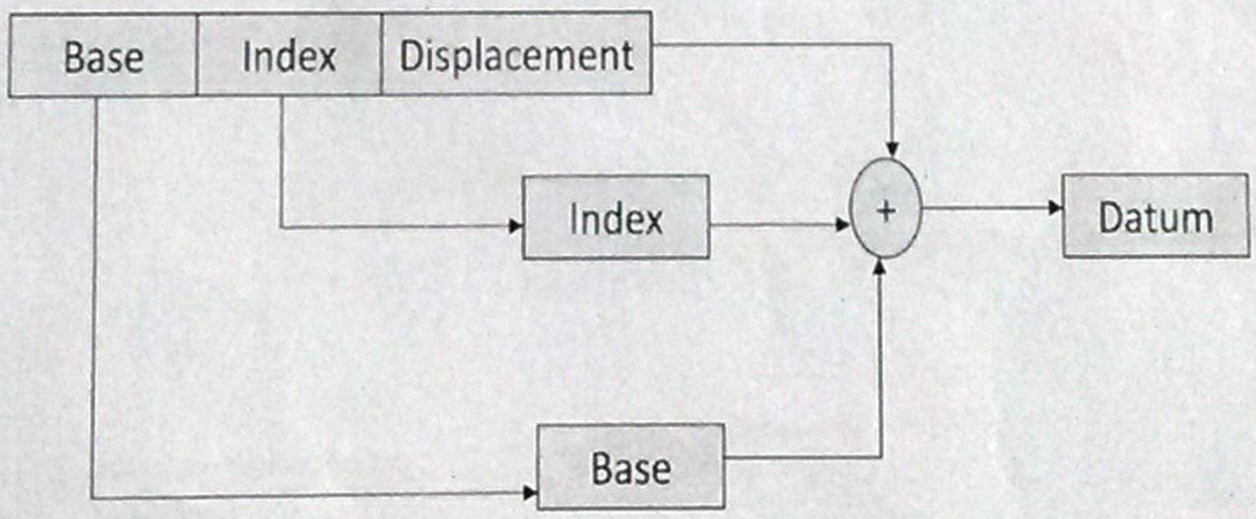


8. Relative Based Indexed Addressing Modes

The Effective address of the operand is formed by adding 8-bit or 16-bit displacement with the contents of any one of the base registers(i.e. BX/BP) and index register (i.e. SI/DI) in the default segments.

Example: MOV AX, 50H [BX][SI]

Here 50H is immediate displacement



ii) Control transfer instructions :-(or) Program Memory access A.M :-

* → The Program memory access addressing modes explain about the transfer of control to some other location which is not in sequence hence applicable to control transfer instruction and some times known as control transfer addressing modes.

* → The control transfer addressing modes depends upon whether the destination location is within same segment (or) in different on. It is also depends up on the method of Parsing the destination address to the Processor.

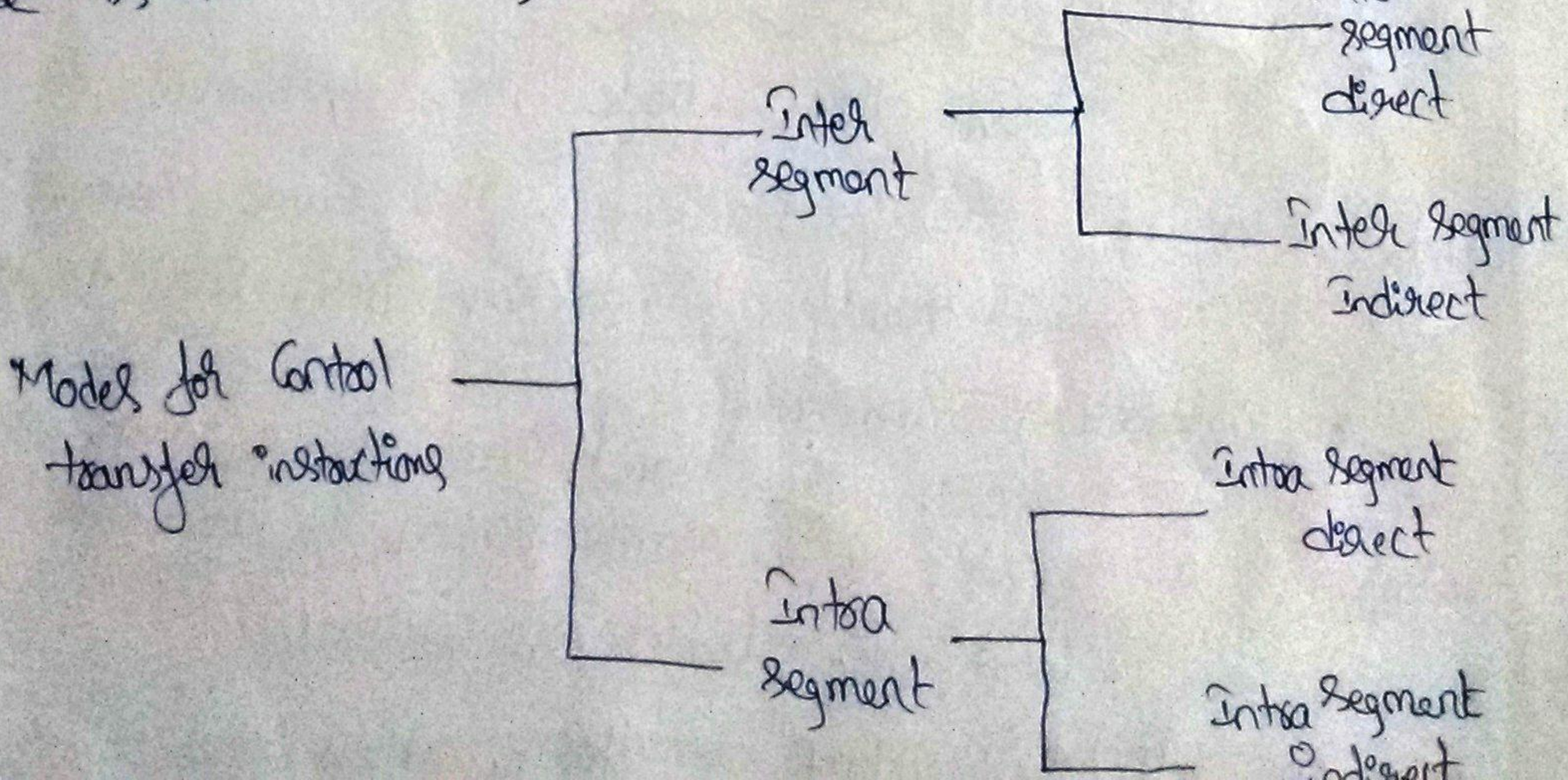
There are 2-types of Addressing modes for control transfer instructions.

(i) Inter Segment Am

(ii) Intra segment Am.

* → If the location to which the control is to be transferred lies in a different segment other than the current one, the mode is called intersegment mode.

* → If the destination location lies in the same segment the mode is called intrasegment mode.



(i) Intra segment Direct mode :-

In this mode the address to which the Control is to be transfered lies in the same segment in which the Control transfer instruction lies and appears directly in instruction as an immediate displacement value.

In this addressing mode the displacement is Computed relative to the Content of Instruction Pointer IP.

The Effective Address to which the Control will be transfered is given by sum of 8 (8) 16-bit displacement and Current Content of IP.

In Case of Jump instruction, if the signed displacement (d) is of 8-bit (ie $-128 < d < +127$), we term it as short Jump.

If it is 16-bits (ie $-32768 < d < +32767$) it is termed as long Jump.

Eg:- JMP SHORT LABEL; LABEL lies within -128 to 127 from the Current IP Content thus SHORT LABEL is 8-bit signed displacement.

---

ii) Intra segment Indirect mode :-

In this mode, the displacement to which the Control to be transfered, is in the same segment in which the Control transfer instructions lies, but it is passed to the instruction indirectly.

Here, the branch address is found as the Content of a register or a memory location. The addressing mode may be used in unconditional branch instructions.

JMP [BX] ; Jump to effective address stored in BX

JMP [BX + 5000H].

(iii) Intersegment Direct :-

In this mode, the address to which the control is to be transferred is in a different segment. This addressing mode provides a means of branching from one code segment to another code segment.

Here the CS and IP of the destination address are specified directly in the instruction.

Eg:- JMP 5000H : 2000H ; Jump to effective address 2000H in segment 5000H.

(iv) Intersegment Indirect :-

In this mode, the address to which the control is to be transferred lies in a different segment and it is passed to the instruction indirectly.

Contents of a memory block containing four bytes ie IP (LSB), IP (MSB), CS (LSB) and CS (MSB) sequentially. The starting address of the memory block may be referred using any of the addressing modes, except immediate mode.

Eg:- JMP [2000H] ;

Jump to an address in the other segment specified at effective address 2000H in DS, that points to the memory block as said above.

MSB → Most Significant Bit

LSB → Least " "

6) Forming the Effective Address :-

Eg:- The Contents of different registers are given below. Form EA for different addressing modes.

offset (displacement) = 5000H

[AX] - 1000H , [BX] - 2000H , [SI] - 3000H , [DI] - 4000H,

[BP] - 5000H , [SP] - 6000H , [CS] - 0000H , [DS] - 1000H,

[SS] - 2000H , [IP] - 7000H

shifting a number four times is equivalent to multiply it by 16_D (or) 10_H

(i) Direct addressing mode :-

MOV Ax, [5000H] → offset address

DS : OFFSET ⇔ 1000H : 5000H

10H * DS = 10000

offset = +5000

15000H → Effective address

(ii) Register indirect :-

MOV Ax, [Bx]

DS : Bx ⇔ 1000H : 2000H

10H * DS = 10000

[Bx] = 2000

12000H → EA

(iii) Register relative :-

MOV Ax, 5000[Bx]

DS : [5000 + Bx]

10H * DS ⇒ 10000

offset ⇒ 5000

[Bx] ⇒ 2000

17000H → EA

(iv) Based indexed :-

MOV AX, [BX] [SI]

DS : [BX] + [SI] = 1000H : 2000H + 3000H

10H * DS ⇒ 10000
[BX] ⇒ 2000
[SI] ⇒ 3000
15000 H → EA

(v) Relative based indexed :-

MOV AX, 5000 [BX] [SI]

DS : [BX + SI + 5000]

10H * DS ⇒ 10000
[BX] ⇒ 2000
[SI] ⇒ 3000
offset ⇒ 5000
1A000 → EA //.

7) MACHINE LANGUAGE INSTRUCTION FORMATS :-

*) A machine language format has one (or) more no:- of fields.
*) The first field is called as operation code (or) Op-code field, which indicates type of operation to be performed by CPU.

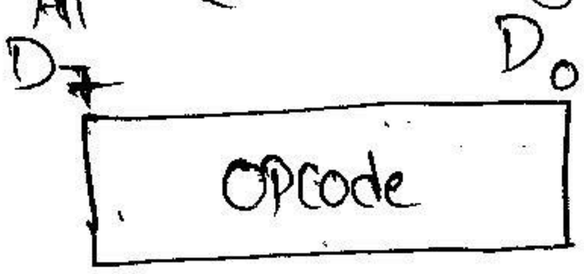
There are 6-general instruction formats in 8086.
*) The length of instruction format may vary from 1 byte to 6 byte.

1.) One-byte instruction :-

This format is only 1-byte (8-bits) and may have the implied data (or) Register operands.

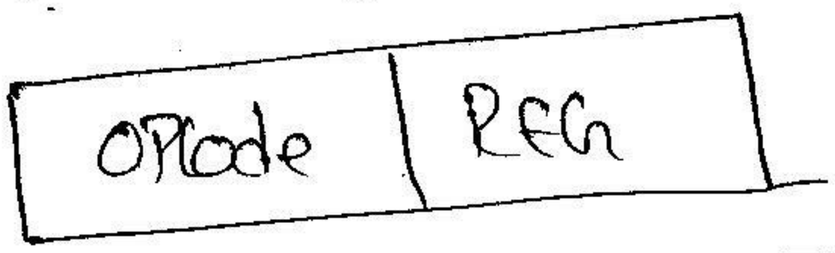
the least significant (3-bits) of op-code are used for specifying the register operand.

All the 8-bits from an OPcode and the operands are implied.



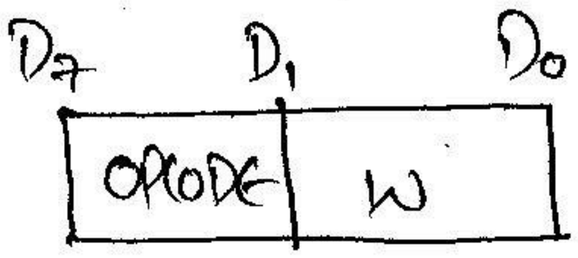
one byte instructions - implicit operands

One byte Instruction - Register mode :-

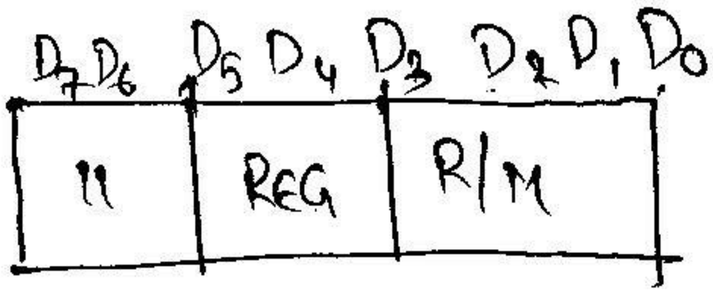


2) Register to Register :- This is 2-byte long.

*) The 1st byte specifies the operation code and width of operand specified by 'W' bit.
*) The 2nd byte specifies the code shows the Register operands and Register/memory R/M field.

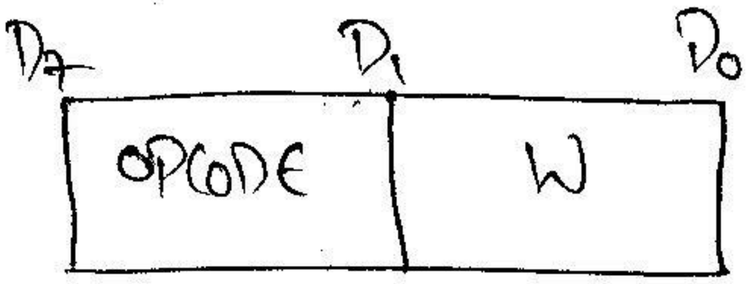


1st byte

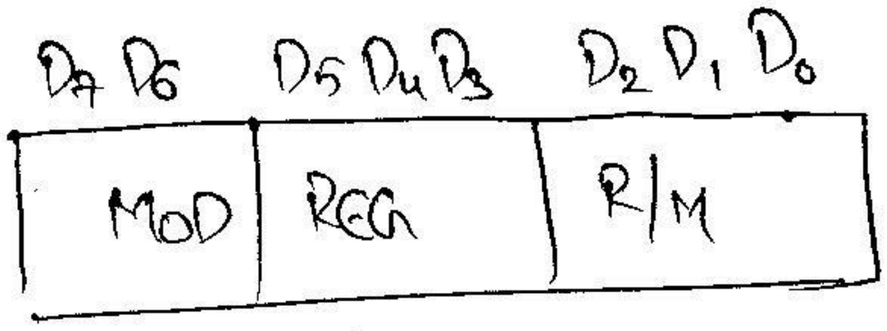


3.) Register to/from Memory with no Displacement.

This format is also 2-byte long and similar to the Register to Register format except for the MOD field.



1st byte

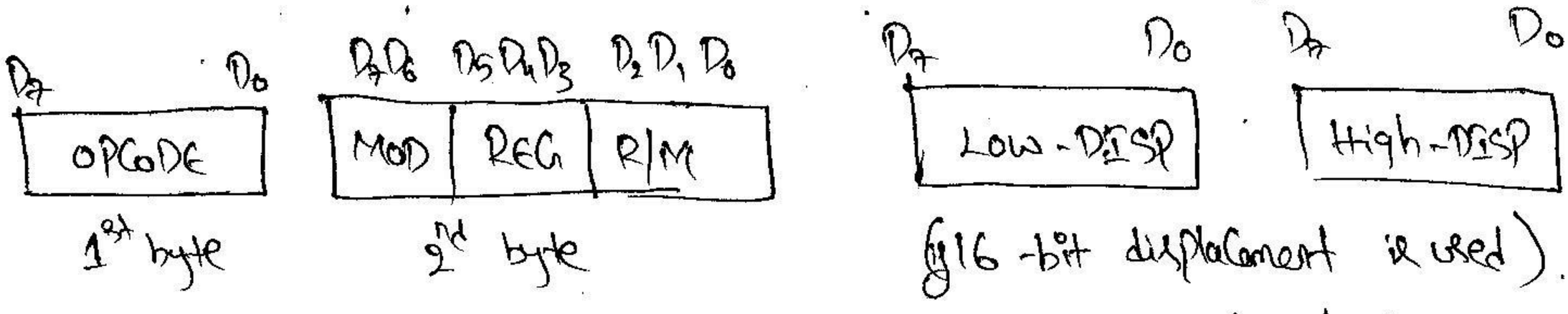


2nd byte

The MOD field shows the Mode of addressing.

4) Register to/from Memory with Displacement:-

This type of instruction format contains 1 (or) 2 additional bytes for displacement along with 2-byte the format of Register to/from memory without displacement.



1st byte

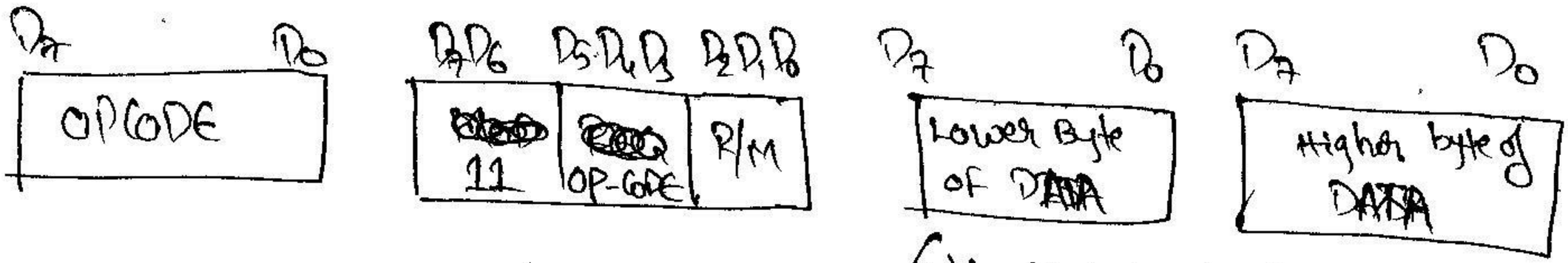
2nd byte

(if 16-bit displacement is used)

Additional 1/2 bytes.

5) Immediate operand to Register :-

In this first byte as well as the 6-bits from the 2nd byte which are used for REG field in case to register to register format are used for opcode.



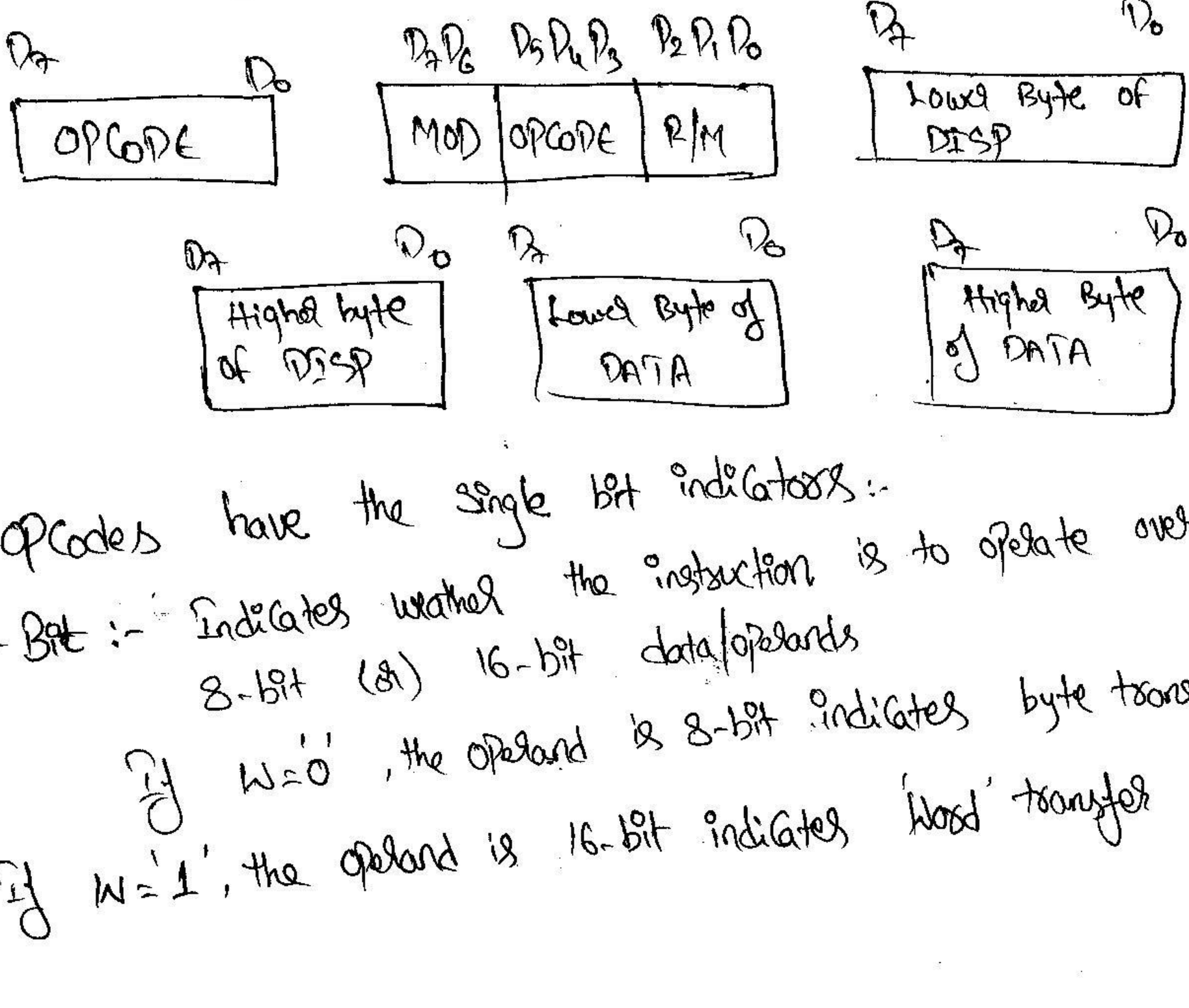
(if 16-bit ~~displacement~~ data used)

6) Immediate operand to memory with 16-bit Displacement :-

*) This type of instruction requires 5 to 6 Bytes for coding.

*) The first 2-bytes contains information regarding OPCODE, MOD, and R/M fields

*) The remaining 4 byte contains 2-byte of displacement and 2-byte of data



The opcodes have the single bit indicators :-

W-Bit :- Indicates weather the instruction is to operate over an 8-bit (or) 16-bit data/operands

If W='0', the operand is 8-bit indicates byte transfer

If W='1', the operand is 16-bit indicates 'word' transfer

D-bit :- This is valid Case of Double operand instructions.
One of the operands must be register specified by the REG field
If D='0' indicates from Register (Source operand)
D='1' " to Register (Destination operand).

S-Bit :- This bit is sign extention bit. The S-bit is along with W-bit to show the type of the operation.

8-bit operation with immediate operand is indicated by S=0 ; W=0

16-bit operation with 16-bit immediate operand is indicated S=0 ; W=1

16-bit " with Sign extended immediate data is S=1 ; W=1

S,W = 01 "16-bit immediate data from operand".

= 11 "An immediate data type is sign extended to form 16-bit operand.

V-bit :- Used in Cases of Shift and rotate instructions
V=0 indicates COUNT = 1
V=1 " " in 'CL' register.

Z-bit :- Used by REP instruction to Control the loop
Z=1 instruction with REP Prefix is executed until zero flag matches the Z-bit.

Code for MOD field	Name of the Mode
00	Memory Mode with no displacement
01	" " 8-bit signed "
10	" " 16-bit unsigned "
11	Register Mode

Register Codes

16-bit Address Code	(W = 1) Width	8-bit Reg Code	(W = 0) Reg Code
000	AX	000	AL
001	CX	001	CL
010	DX	010	DL
011	BX	011	BL
100	SP	100	AH
101	BP	101	CH
110	SI	110	DH
111	DI	111	BH

Segment (2-bit) Register Code	Segment Register
00	ES
01	CS
10	SS
11	DS

d → displacement

R/M \ MOD	00	01	10	11 W=1	11 W=0
000	[BX]+[SI]	[BX]+[SI]+d_8	[BX]+[SI]+d_{16}	AX	AL
001	[BX]+[DI]	[BX][DI]+d_8	[BX][SI]+d_{16}	CX	CL
010	[BP]+[DI]	[BP]+[SI]+d_8	[BP]+[SI]+d_{16}	DX	DL
011	[BP]+[DI]	[BP]+[DI]+d_8	[BP]=[SI]+d_{16}	BX	BL
100	[SI]	[SI]+d_8	[SI]+d_{16}	SP	AH
101	[DI]	[DI]+d_8	[DI]+d_{16}	BP	CH
110	d_{16} direct Address	[BP]+d_8	[BP]+d_{16}	SI	DH
111	[BX]	[BX]+d_8	[BX]+d_{16}	DI	BH

To find out the MOD and R/M fields of particular instructions one should first decide the addressing mode of the instruction. The A.M depends upon the operands and suggests how the effective address may be computed for locating the operand, if it lies in memory.

Note:- Usually All the addressing modes have DS as default data segment. However the A.M using BP and SP have SS as the default segment register.

Eg:- (1) MOV AL, 20H

1011	W	REG	DATA
1011	0	000	20H

It is Immediate A.M

from above instruction the instruction code is

1011	W	REG	data

As the data given is not 16-bit, W=0 and the data is being transfered to AL and its code is "000".

∴ Code is 1011 0 000 20H

Now binary code is 1011 0000 20H is

Machine code = B0, 20H //.