RANKA
DATE 17 / 1 / 25
PAGE

MODULE 1 - THE INTEL MICROPROCESSORS 8086 ARCHITECTURE

- What is Microprocessor?

It is a small chip used to execute programs.
eg. laptop, computers, etc

- Features of Microprocessor -

1. It is a single chip which is capable of processing the data. It controls all the components of computers. It executes the instructions sequentially using pipeline.
2. MP requires the following buses -
   a. Data bus
      It carries data to and from processor and bigger data bus, faster then processor.
   b. Address bus
      It carries address where the operation has to be performed
   c. Control bus
      It carries the control signals.
      eg. READ, WRITE, etc.

- Difference between Microprocessor & Microcontroller

MICROPROCESSOR	MICROCONTROLLER
1. Processor, memory & I/O devices are seperate	1. Everything are on same / single chip
2. Processor is expensive	2. It is cheap
3. It is used for general purpose	3. It is used for specific purpose
4. High power consumption	4. Low power consumption
5. eg. laptop, computer, etc	5. AC, remote, washing mach, etc.

RANKA
DATE / /
PAGE

- Generation of MP

1. 8085 - 8 bits
2. 8086 - 16 bits
3. 80186 } 16 bits
   80286 }
4. 80386 - 32 bits
5. 80486 - 32 bits
6. 80586 Pentium I, II, III
7. Pentium IV

- Features of 8086 MP

1. Size of data bus = 16 bits
   Size of address bus = 20 bits
   Size of memory = 2^{20} bits = 1048576 = 1 MB
   Size of memory is divided into 2 bank
   One is odd and one is even.
   Size of each memory location = 1 byte = 8 bits
2. It supports pipeline, Pipeline means fetch, decode, execute.
3. It has two operating modes, one is minimum and one is maximum. It supports memory segmentation.

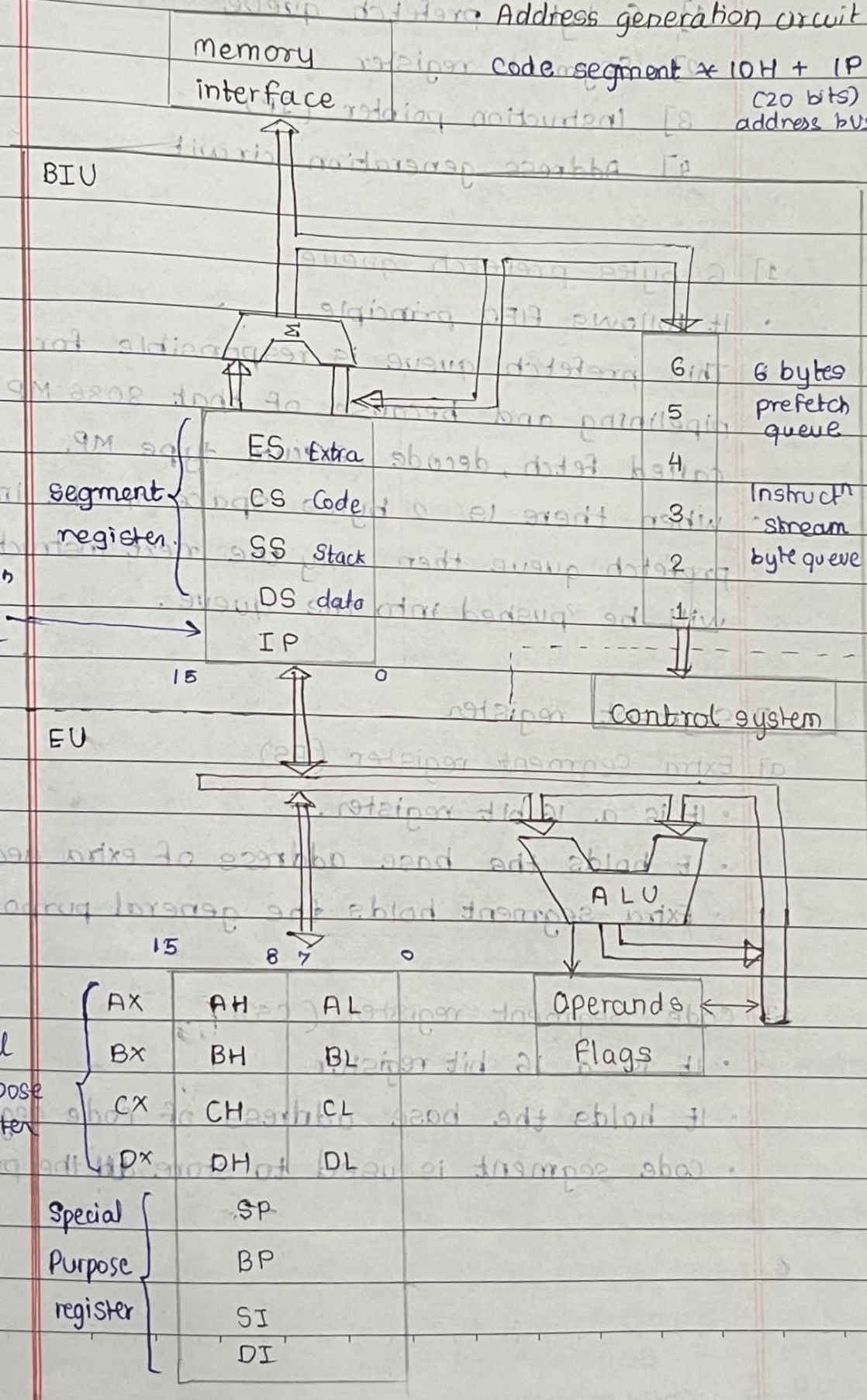
Odd (Higher) 00001H to FFFFFH
Even (Lower) 00000H to FFFFEH

$4H\,4H = 2^0$	A_{19} ...	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
00000H	0	0	0	0	0	0	0	0	0
00001	0	0	0	0	0	0	0	0	1
00002	0	0	0	0	0	0	0	1	0
00003	0	0	0	0	0	0	0	1	1

RANKA
DATE / /
PAGE

- 8086 CPU architecture

It comprises of 2 units 1] BIU 2] EU



RANKA
DATE / /
PAGE

- BIU - BUS INTERFACE UNIT

BIU has following components

1] 6 bytes prefetch queue
2] Segment register
3] Instruction pointer (IP)
4] Address generation circuit

1] 6 bytes prefetch queue

- It follows FIFO principle.
- This prefetch queue is responsible for pipelining and because of that 8086 MP is called fetch, decode, execute type MP.
- When there is 2 bytes space in the instruct prefetch queue then only the next instruction will be pushed into the queue.

2] Segment register

a] Extra Segment register (ES)

- It is a 16 bit register.
- It holds the base address of extra segment
- Extra segment holds the general purpose dat

b] Code Segment register (CS)

- It is a 16 bit register.
- It holds the base address of code segment
- Code segment is used to store the progra

c] Stack segment register (SS)
- It is a 16 bit register.
- It holds base address of stack segment.
- Stack segment holds the stack memory.

d] Data segment register (DS)
- It is a 16 bit register.
- It holds the base address of data segment
- Data segment is used for storing the general purposed data.

3] Instruction pointer (IP)
- It is 16 bit register.
- It holds the offset value of code segment.
- IP is incremented after every instruction byte is fetched.

4] Address generation circuit
- A BIU has a physical address generation circuit.
- It generates 20 bits physical address using segment and offset addresses.

Code segment × 10H + IP = Physical address
(16 bits ⇒ 20 bits)

NOTE ① ES × 10H + DI ② SS × 10H + BP ③ DS × 10H + SI

eg. CS = 1234 H and IP = 0005 H

Physical address = 1234 × 10 + 0005
= 12340 + 0005
= 12345 H (20 bits)

$$\begin{array}{r} 12340 \\ +\ 0005 \\ \hline 12345 \end{array}$$

RANKA
DATE 21 / 1 / 25
PAGE

- EU - Execution Unit

EU has following components

1] General Purpose Register
2] Special Purpose Register
3] Arithmetic Logic Unit
4] Control System
5] Flags.

1] General Purpose Register

a] AX (Accumulator)

- It is 16 bit register.
- It is combination of AL and AH register. where AL and AH are 8 bit registers.
- Accumulator is most useful register which holds the operand and results during multiplication and division operation.

b] BX (Base)

- It is a combination of BL and BH register 8 bits each.
- It holds the memory address in indirect addressing modes

c] CX (Counter Register)

- It holds the count of instruction like a loop, rotate, shift and string operation

d] DX

- It is used with AX register to hold 32 bit value during multiplicat^{n} & division values.
- It acts as extension of AX register.

RANKA
DATE / /
PAGE

2] Special Purpose Register

a] SP (Stack Pointer)

- It points to the stack top.

b] BP (Base Pointer)

- It holds the offset address of any locat^{n} in stack segment.

c] SI (Source Index)

- It holds offset address in data segment during string operation.

d] DI (Destination Index)

It holds offset address of extra segment during string operation

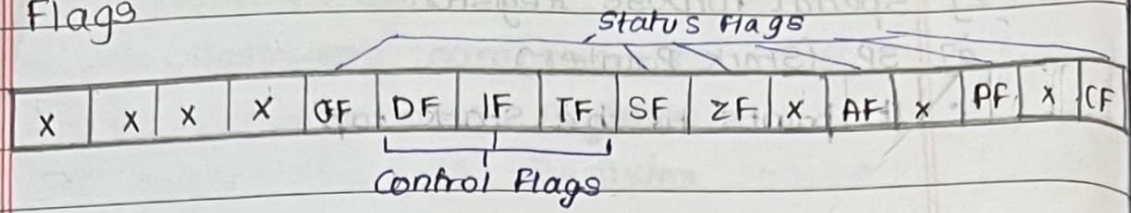
3] Arithmetic Logic Unit (ALU)

- 8086 have 16 bit ALU
- This ALU perform both 8 bits and 16 bits operation.

4] Control System

- It is also 16 bit component.
- Instructions are already fetched by BIU and stored in prefetched queue. The instruction is removed by prefetched queue and goes to Control System for decoding purpose.
- The Control System then generates control signal that inform all part of architecture that what need to be done for exewtion.

5] Flags



The Flag register is divided into 2 part-

1] Status Flags

2] Control Flags

1] Status Flag

a] Carry flag-

Set 1 = Carry out of MSB

Reset 0 = No such carry

eg. 1100 0011

1011 1100

[1] 0111 1111

↳ Carry generated, Hence Carry Flag = 1.

b] Parity Flag-

1 = Even Parity

0 = odd Parity.

eg. No. of 1's (0111 1111) = 7

∴ Parity flag = 0.

c] Auxiliary Carry Flag-

1 = Carry from Lower Nibble to Higher Nibble

0 = No such Carry

eg. 1100 1011 CF = 1

[1] 0011 1100 AF = 1

0000 0111

d] Zero flag -

1 = Result = 0

0 = Result ≠ 0

e] Sign flag -

1 = MSB of result is 1 (-ve)

0 = MSB of result is 0 (+ve)

(used for signed nos)

f] Overflow flag :

1 = overflow occured

0 = No overflow occurred.

~~of is calculated as E7~~ .

Range { -80H to 7F H

{ $(-128)_{10}$ to $(127)_{10}$

2] Control flag

a] Trap flag -

This flag is used for single-stepping purpose

1 = ~~Sti~~ Single stepping enable

0 = Single stepping disable.

b] Interrupt flag -

This flag is used to enable and disable the interrupt

1 = Enable

0 = Disable.

c] Direction flag -

It is used ~~to~~ in string manipulation.

1 = Auto decrement

0 = Auto increment

RANKA
DATE 24 / 1 / 25
PAGE

① 43H + 22H

	43	0100 0011
+	22	0010 0010
	65	0110 0101

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

② 41H + 51H

	41	0100 0001
+	51	0101 0001
	92	1001 0010

CF = 0 ZF = 0
PF = 0 SF = 1
AF = 0 OF = 1 $(92)_{16} = (146)_{10}$

GPR

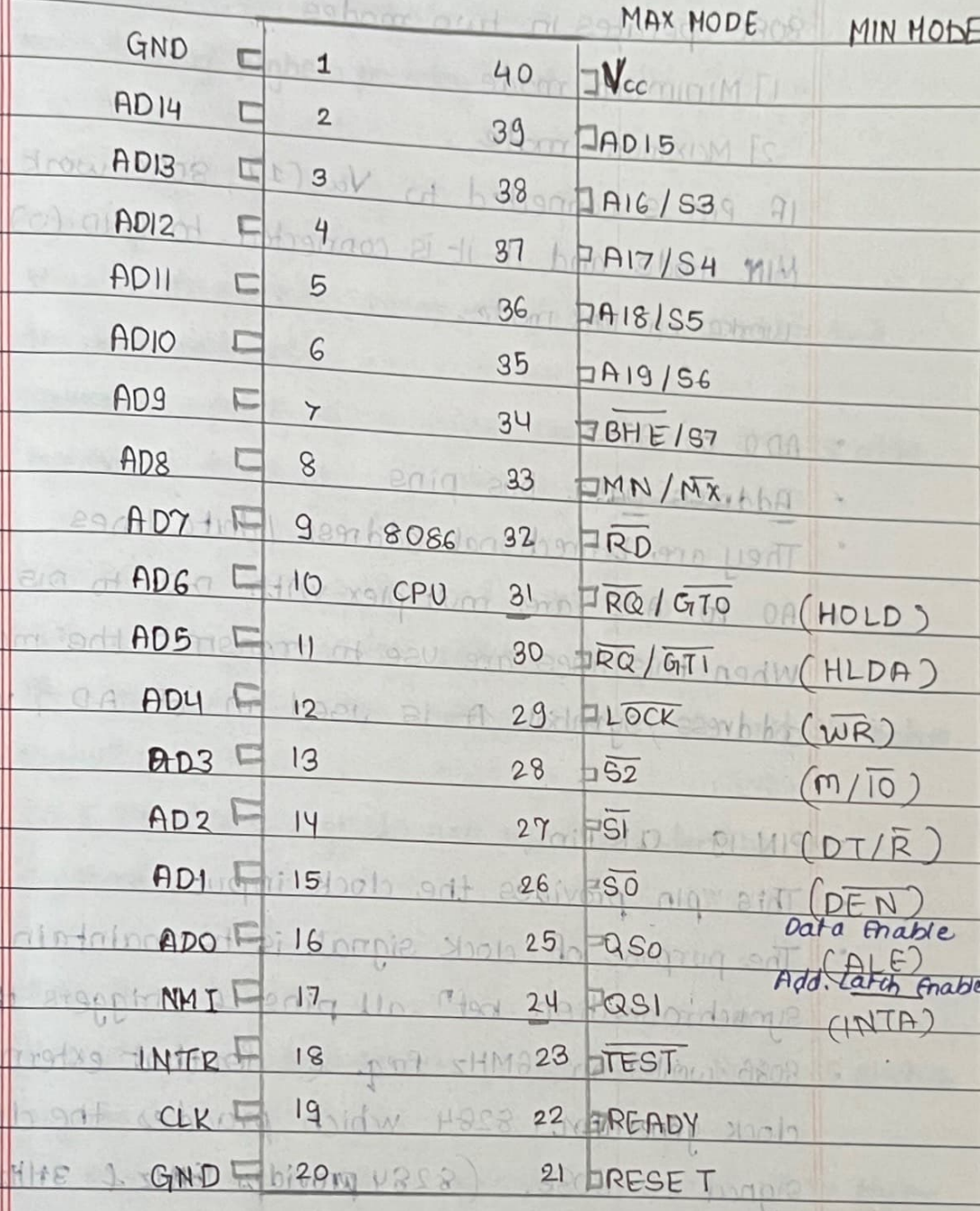
SPR

SEGMENT

PLAGS

Programmer's model.

- 8086 Pin diagram.



— active when I/p = 0 - active low pins.

S - Status signal

NMI - Non Maskable Interrupt

INTR

RANKA
DATE / /
PAGE

- PIN 33 - MN/$\overline{MAX}$ PIN
- 8086 operates in two modes
  1] Minimum mode
  2] Maximum mode
- If Pin is connected to V_{cc} (1), 8086 work on MIN. mode and if it is connected to GND. (0) it works on MAX mode.

- AD0 to AD15
- Address Data bus pins
- They are bidirectional address / data lines.
- A0 to A15 are multiplex with D-0 to D15 data bus
- When this lines are use to transmit the memory address, symbol A is used in of AD, vice versa.

- PIN 19 CLK Pin
- This pin provides the clock input.
- The purpose of clock signal is to maintain synchronisation bet^n all pins & triggers the operat^n.
- 8086 works on 8MHz freq. & for that external clock generator 8284 which provides the clock signal to 8086. (8284 provides 18MHz & 3MHz → 05;is, 18/3=6)

- PIN 1 and 20 - GND PIN. (Ground Pin)
- It is enable when Ground pin is zero.

- PIN 40 - V_{cc}
- 8086 require +5 volt DC supply & enable when VCC = 1

RANKA
DATE 27 / 1 / 25
PAGE

- NMI - PIN 17
- Non Maskable interrupt
- NMI having higher priority and it is hardware interrupt.
- INTR - PIN 18
- Interrupt.
- It is a maskable interrupt (user can enable and disable request).
- Maskable INTR is a lower priority intr & it is also hardware interrupt.

- PIN 38 / 37 - A16/S3 & A17/S4
- This are higher order address lines.
- This address line are multiplex with S3 and S4 status signal.
- S3 & S4 status signals are used to identify the segment.

- PIN 36 A18/S5
- Here address line bit is multiplex with S5 status signal.
- S5 status signal gives interrupt status if S5=1 then intr enable, else intr disable.

- PIN 35 A19/S6
- S6 status signal is used for data transfer purpose.

RANKA
DATE / /
PAGE

- PIN 34 $\overline{BHE}$ /S7
- BHE stands for Bus High Enable.
- BHE & A0 is used for Memory Banking
- It is multiplex with S7 status signal & it is also used for data transfer purpose.
- $\overline{BHE}$ = 0 → odd A0 = 0 → even

- PIN 32 $\overline{RD}$ (Read)
- It is active during read operation.
- MP 8086 performs I/p read & memory read
- It is active low pin.

- PIN 23- $\overline{TEST}$
- It is active low pin.
- Wait for test control when TEST = 0, the MP continue it's execution & when TEST = 1, MP waits

- PIN 22 - READY
- This signal is used to synchronise the MP with peripherals.
- When READY = 1; peripheral is ready to transfer data
- When READY = 0; peripheral is not ready.

- PIN 21 - RESET
- It is used to RESET the MP 8086.

RANKA
DATE / /
PAGE

PIN 24 - 31 work in both Minimum & maximum mode.

Minimum mode.

- PIN 24 ($\overline{INTA}$)
- Interrupt Acknowledgement pin
- On receiving intr signal, the processor issue an ~~INTA~~ INTA signal.
- PIN 25 - ALE.

  Address latch Enable pin
- When ALE = 1; it is use to transfer addresses
  ALE = 0; It is not use to carry addresses.
- PIN 26 - $\overline{DEN}$

  Data Enable
- It is active low pin.
- When DEN = 0; it is use to transfer ~~data~~ data.
  ~~DEN = 1;~~
- PIN 27 - D T/$\overline{R}$

  Data Transmission/Receiver pin
- When it is high, data is transmitted (WRITE).
  When it is low, data is received (READ).
- PIN 28 (M/$\overline{IO}$)

  Memory / Input Output
- When this signal is high, CPU wants to access memory.
  When this signal is low, CPU wants to access i/p devices.

- PIN 29 - $\overline{WR}$ Write pin
- It is active low pin
- If write = 0, then write operation executes

- PIN 31 - HOLD
- When other device wants to use address & data bus, it sends hold request to the MP 8086 using this pin.

- PIN 30 - HLDA
- Hold Acknowledgement pin
- HLDA is sent by the processor when it receives HOLD signal.

Maximum mode

28/1/25

- PIN 26, 27, 28 ⇒ $\overline{S0}$ $\overline{S1}$ $\overline{S2}$
- S0, S1, S2 are status signals which are used to generate control signals.
- This status signals are decoded as follows

S2	S1	S0	Operations
0	0	0	Interrupt ack.
0	0	1	I/O read
0	1	0	I/O write
0	1	1	Halt
1	0	0	Code access
1	0	1	Memory read
1	1	0	Memory write
1	1	1	Inactive.

- PIN 30, 31 $\overline{RQ}$/GT0 $\overline{RQ}$/$\overline{GT1}$ Request Grant
- RQ pin is used by other processor to force the current processor 8086 for releasing the local bus.
- This pins are bi-directional.
- GT pin is used to grant request
- The request on GT0 have higher priority then GT1

- PIN 29 - $\overline{LOCK}$
- It is an o/p signal activated by LOCK prefix and remain active until the completion of instruction prefixed by LOCK
- This pin is used to prevent other bus master from gaining control of the bus.

- PIN 24, 25 QS0 QS1 Queve status
- These signals indicate the status of 8086 queve according to the following table.

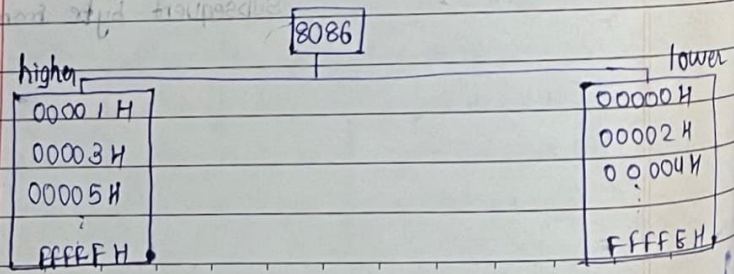
QS1	QS0	Operation
0	0	No operation
0	1	First byte of op code from queve
1	0	Empty the queve
1	1	Subsequent byte from queve

- 8086 memory banking.
- In 8086 processor, memory is stored in memory location. Each location can hold 1 byte of data.
- Why? One location stores only 1 byte of data?

→ Minimum byte operation needs 1 byte so it is prefered to assign min. memory to one location so that memory is not wasted.

- whenever 16 bit data is stored in memory, it is stored in 2 consecutive memory location.
- So whenever processor needs to fetch this it first fetches lower bytes and then higher bytes.
- So overall 16 bits processor needs 2 cycles to perform 16 bits operation which leads to memory banking.
- If the memory can be divided into 2 parts, with some algorithm, two 8 bit data can be fetched in one cycle.
- ~~As or~~ In 8086, memory is divided into 2 parts-
  1] Odd bank (Higher bank)
  2] Even bank (Lower bank)

Even bank starts from 00000H to FFFFEH
Odd bank starts from 00001H to FFFFFH



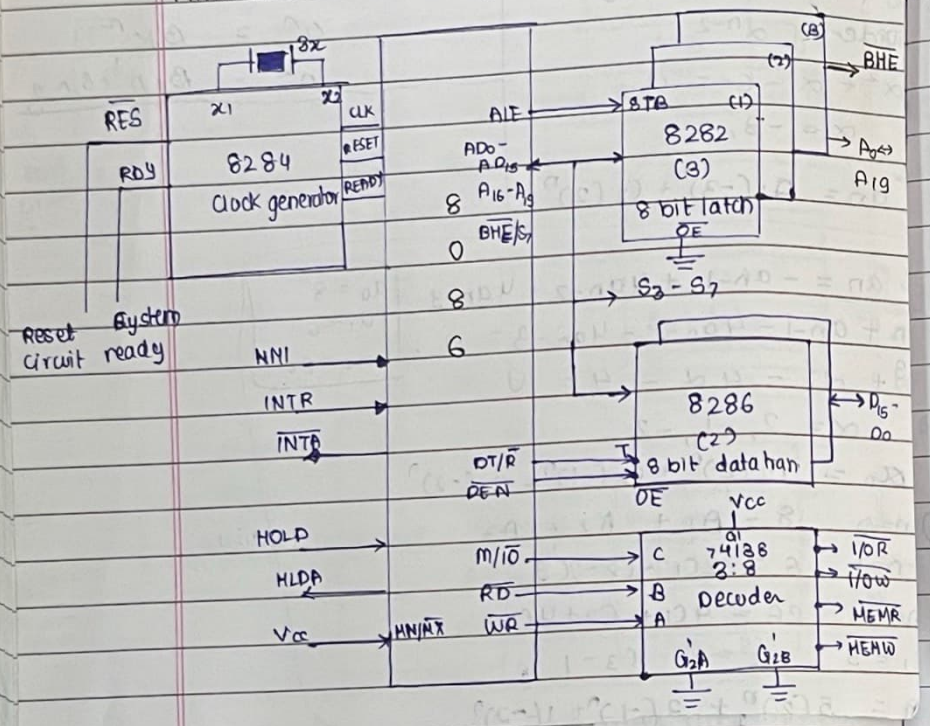
- Align data
- Location 00000H and 00001H is example of align data, means 8086 can fetch the data from both the location in one cycle.
- Location 00001H and 00002H is example of misalign data, means 8086 requires 2 cycles to access the misalign data.
- A0 and BHE is use to select the bank.

$\overline{BHE}$ = 0 ⇒ odd ⇒ Higher bank
A0 = 0 ⇒ even ⇒ Lower bank

BHE	A0	Mode
0	0	Both the banks are selected & 8086 performs 16 bit operations
0	1	Higher bank is selected & 8086 performs 8 bit operations with higher bank only
1	0	Lower bank is selected & 8086 performs 8 bit operations with lower bank only
1	1	Both the banks are kept idle.

Latch - storage.

- Minimum Mode Configuration of 8086



- 8086 works in minimum mode when MN/MX pin is connected to Vcc.
- In minimum mode, 8086 is only processor and other components used in min. mode are as follows -
  1] 8284 Clock generator
  2] 8282 (3) 8bit Address latch
  3] 8286 (2) 8bit data transreceiver
  4] 74138 3:8 Decoder

1] 8282 (3) 8bit address latch
- If ALE = 1, multiplex pins act as a address lines
- If ALE = 1, then 8282 IC allows the value (add) from this multiplex lines and then store it in 8282 latch
- Following Imp. things are used in 8282 -

1] STB - Strobe
- This pin is equivalent to ALE.
- Whenever STB = 1, then address is allowed inside 8282

2] $\overline{OE}$ - Output Enable
- Active low. pin
- This pin is used to select 8282 chip

2] 8284 clock generator.
- This provides the clock signals to 8086 MP which are essential for synchronising the operations of 8086 MP.
- 8086 is powered with 6MHz frequency. This 6MHz frequency is provided by 8284 clock generator.
- 8284 have 18MHz clock signals and 3MHz crystal oscillator.

So, $\frac{18}{3}$ = 6 MHz frequency is passed to 8086

3] 8286 (2) 8 bit data transreceiver.

- Whenever ALE = 1 data addresses also enters to 8286, to solve this issue 8286 should be logically disconnected. (However, disconnection is not possible) this is done by $\overline{OE}$ pin.
- This $\overline{OE}$ pin of 8286 is connected to the $\overline{DEN}$ pin of 8086.
- $\overline{DEN}$ pin decides whether signal from multiplex bus should enter the 8286 transreceiver or not.
- Once ALE = 0, $\overline{DEN}$ = 0, $\overline{OE}$ = 0 which means 8286 allow the data from multiplex bus.
- T pin T = 1 transmit T = 0 receive
- As 8286 is a bidirectional device it should be specified whether the data is transmitted or received and which is done by T pin of 8286.

4/02/25

4] IC 74138 3:8 Decoder

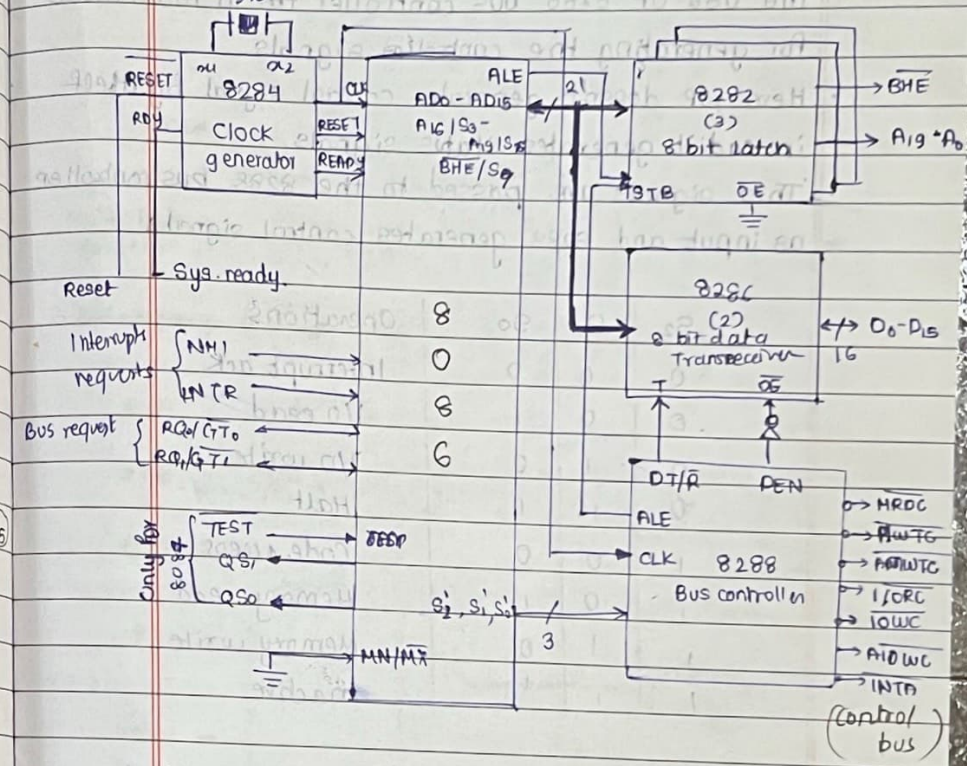
- This is 3:8 decoder
- For this i/ps are

1] M/$\overline{IO}$ 2] $\overline{RD}$ 3] $\overline{WR}$

M/$\overline{IO}$	$\overline{RD}$	$\overline{WR}$	Action
1	0	1	Memory read
1	1	0	Memory write
0	0	1	I/O read
0	1	0	I/O write

Refer min. mode pins for 8086 pin diagram for NMI, INTR, $\overline{INTA}$, HOLD, HLDA.

- Maximum Mode Configuration of 8086.



- In maximum mode, MN/$\overline{MX}$ pin is connected to the ground.
- Following components are used in Maximum mode -

1] 8284 clock generator
2] 8282 (3) 8 bit latch
3] 8286 (2) 8 bit data transreceiver
4] 8288 bus controller

For 1], 2], 3] refer the explaination from minimum mode.

4] 8288 Bus controller

- The use of 8288 bus controller in max. mode for generating the control signals.
- Here MP doesn't generate control signal by itself instead it generates status signals.
- This signals are passed to the 8288 bus controller as input and 8288 generates control signals

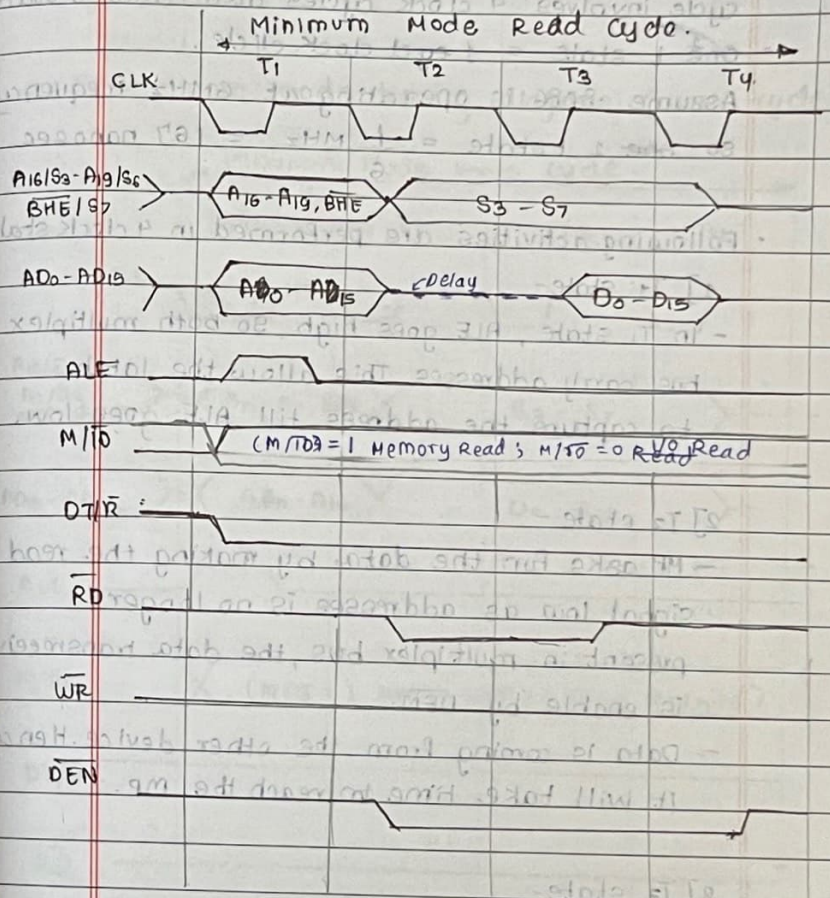
S_2	S_1	S_0	Operations
0	0	0	Interrupt ack.
0	0	1	I/O read
0	1	0	I/O write
0	1	1	Halt
1	0	0	Code access
1	0	1	Memory read
1	1	0	Memory write
1	1	1	Inactive

- The output signals generated by status signals (according to diagram).

MRDC Memory Read
MWTC Memory Write
AMWTC Advanced memory write.
IORDC Input read
IOWTC Input write
AIOWC Advanced input output write
INTA Interrupt Acknowledge.

- For $\overline{RQ_0}/GT_0$, RQ_1, GT_1, NMI, INTR, TEST refer 8086 pin diagram.

- Timing diagrams
- Minimum mode Timing diagram (Read cycle),



Q. 10m Give description about pulses. Refer 8086 pin diagram.

- In read machine cycle, data is transferred from memory or i/o device to the MP. One machine cycle involves 4 clock cycles also called t-state.
- 1 t-state = 1 clock cycle.
Assume 8086 is operating at 6MHz frequency so 1 t-state = $\frac{1}{6}$ MHz = 167 nanosec.

- Following activities are performed in 4 clock states.

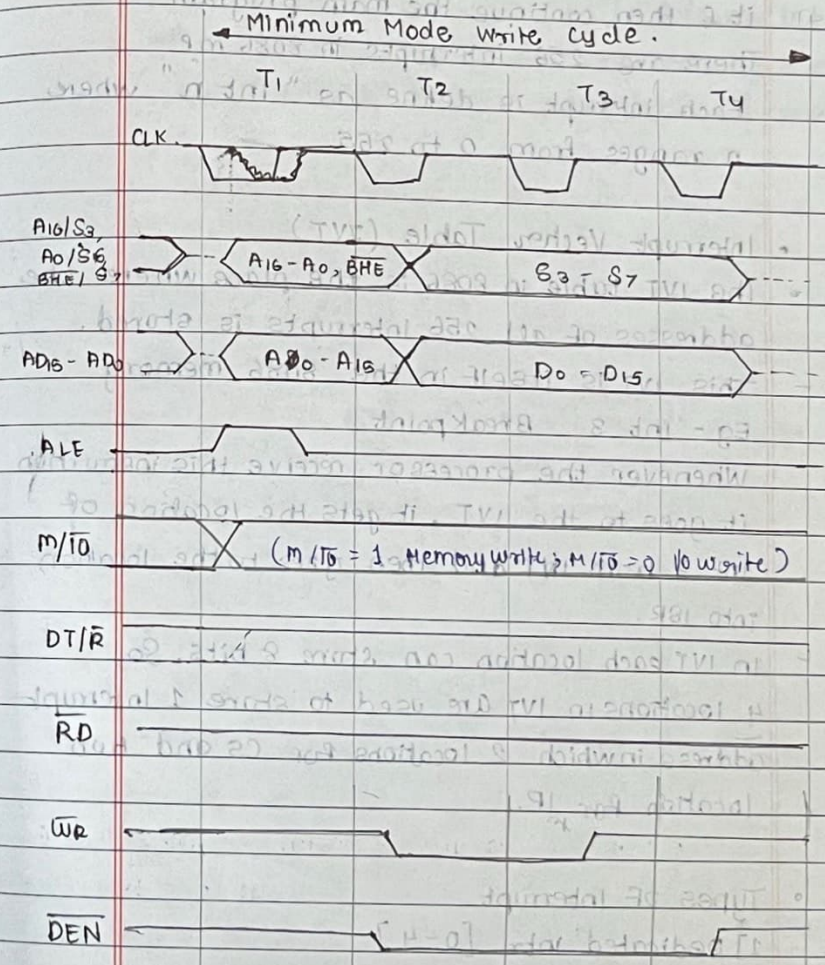
1] T_1 state -
- In T_1 state, ALE goes high. So both multiplex bus carry addresses. This allow the latch to capture the address till ALE goes low.

2] T_2 state -
- MP asks for the data by making the read signal low as addresses is no longer present in multiplex bus, the data transreceiver is enable by $\overline{DEN}$.
- Data is coming from the other device. Hence, it will take time to reach the MP.

3] T_3 state -
- Memory starts sending data on databus.
- IF READY = 1, it means device is ready & MP completes the machine cycle in next t-state, i.e, T_4 state.
- If READY = 0, it means device is not ready & MP will insert into wait state between T_3 and T_4.

4] T_4 state -
- MP captures the data sent by the memory device.

- Minimum mode timing diagram (write cycle),



Maximum mode timing diagram → Read
→ Write

- Interrupt structure

Interrupt in 8086 is a special condition that arises while 8086 is executing the main program.

- Then 8086 stops & goes to the ISR (Interrupt Service Routine), MP executes ISR and complete it & then continue the main program.
- There are 256 interrupts in 8086 mp.
  Each interrupt is define as "Int n" where n ranges from 0 to 255.

- Interrupt Vector Table (IVT)
- The IVT table in 8086 is the place where the addresses of all 256 interrupts is stored.
- This IVT is itself in the 8086 memory
  Eg - Int 3 Breakpoint
  Whenever the processor receive this instruction it goes to the IVT, it gets the location of Int 3 from IVT and then goes to the location into ISR.
- In IVT each location can store 8 bits. So, 4 locations in IVT are used to store 1 Interrupt address inwhich 2 locations for CS and two location for IP.

- Types of Interrupt.

1] Dedicated intr [0-4]

- 0 - Divide error
- 1 - Single Step
- 2 - NMI
- 3 - Break point intr
- 4 - Overflow intr

2] Reserved intr [5-31]

These intrupts are reserved by INTEL which is used in higher processor and they are not available for user.

3] User defined intr [32-255] / Software intr.

ISR For this interrupts are written by the user to service various user-defined conditions.

MODULE 2 - INSTRUCTION SET & PROGRAMMING.

* Explain the addressing modes of 8086 MP.

1] Immediate Addressing Mode
The addressing mode in which the data operand itself is a part of the instruction. is known as immediate addressing mode.
eg MOV AX, 1234 H

2] Register Addressing Mode.
The data is referred using specific register in which data is stored.
eg. MOV AX, BX.

3] Direct addressing mode.
In this type of addressing mode the effective address is directly given into the instruction.
eg. MOV AX, [1234] // value at this location is used
The square bracket around 1234 indicate that it is memory location and not a 16 bit data.

4] Implied addressing mode
In this addressing mode, operands are implied (not specified)
eg. LAHF (Load AH reg. from Flag), STC (Set Carry Flag).
CLD (Clear Direction Flag)

5] Indirect addressing mode.
a] Register indirect addressing mode.
- Operand address is the add. of register
eg. MOV AX, [BX].
Here CS = 2000H & BX = 1135
- Physical address = CS add * 10H + Offset
= 20000 + 1135
= 21135
- Data store in address 21135 H is 234H. So 234H will be stored in AX.

b] Register Relative addressing mode.
- The address of the operand is given as a sum of register address and a displacement (8 or 16 bit value)
- eg. MOV AX, [BX + 4]
Physical address = DS * 10H + [offset + Relative]
Let DS = 4000; BX = 2225.
∴ Physical address = 40000 + [2225 + 4]
= 42229 H.

c] Base Indexed Addressing mode
- operand address is given as a sum of base register plus an index register (SI, DI)
eg. MOV AX, [BX + SI]
Physical address = CS * 10H + [BX + SI]

d] Base Relative plus addressing mode.
- Operand add. is given as a sum of base register plus an index register (SI, DI) plus displacement
eg. MOV AX, [BX + SI + 20H]
Physical add = CS * 10H + [BX + SI + 20]

- DATA TRANSFER INSTRUCTION.

1) MOV

MOV destination, source

2) PUSH

PUSH source

3) POP

POP destination

4) XCHG (exchange)

XCHG R1, R2

* 5) LAHF (Load AH from flag register)

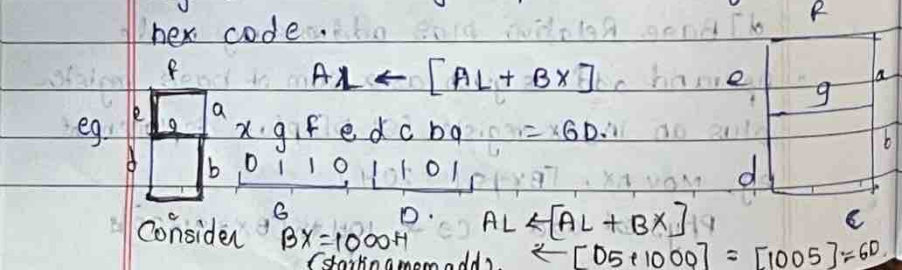
load AH with content of lower 8 bits (8 bits) of flag register (16 bits).

- Flags SF, ZF, AF, PF, CF
  7, 6, 4, 2, 0

LAHF

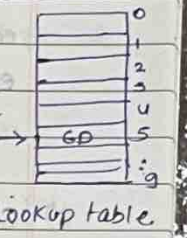
6) XLATB

- If this instruction is use then no need to mention operand.
- It is used to translate the content of AL register into it's ~~reg~~ respective 7 segment hex code.



- Hex value of each no. from 0 to 9 is stored in DS (Lookup table) in sequential manner.

eg. MOV BX, ~~OFFSET~~ OFFSET value ;
XLATB ;



Lookup table

7) XLAT translation table address

- This instruction required to mention the operand
  [AL ← AL + translation table add]

8) LEA (Load Effective Address)

LEA destination, source add.

eg. LEA BX, FF03

Here FF03 is effective add. means offset value of segment.

Stack	
24	FF05
54	FF04
	FF03

- ARITHMETIC INSTRUCTION.

1) ADD

ADD R1, R2

2) SUB

SUB R1, R2

3) MUL

MUL R1, R2

4) DIV

DIV R1, R2

5) ADC (Add with Carry)
- It performs ADD instruction but adds the carry flag to result.
- eg. ADC AL, 25H
(AL ← AL + 25H + carry)

6) SBB (Subtraction with Borrow)
- It subtracts borrow also.
- eg. SBB AL, 25H,
(AL ← AL - 25H - carry)

7) INC
eg. INC AX
(AX ← AX+1)

8) DEC
eg. DEC AX
(AX ← AX-1)

9) NEG
This will negate instruction (2's complement)
eg. NEG BX
BX = 35 = 0011 0101
2's complement = 1100 1011
C B

10) CMP
Compare instruction result isn't stored but flags will be affected. [result is not stored]
OP1 = OP2 ; ZF = 1
OP1 < OP2 ZF. CF = 1

11) DAA (Decimal Adjust for addition)
- It works only for on AL register
- Used when we want to perform addition of 2 decimal no.
- Logic : If lower nibble of AL > 9,
then add 06
If higher nibble of AL ≥ 9
then add 60.
eg. AL = 15H, CL = 28H
ADD AL, CL 15H + 28H = 3DH
DAA 3DH + 06H = 43.
// To convert to decimal answer add 06 to 3D because in 3D lower nibble is greater than 9.
This instruction is used after addition.

12) DAS (Decimal Adjust for Subtraction)
- It works only on AL register.
- Used when we want to perform subtraction of 2 decimal no.
- Logic - If lower nibble of AL > 9 then subtract 06
If higher nibble of AL > 9 then subtract 60
- Here we first perform subtraction of 2 no & result is stored in AL register. Then perform DAS instruction.
eg. AL = 86H CL = 57H
SUB AL, CL 86H - 57H = 2FH
86 - 57 = 29
To convert decimal and subtract 06 from 2F bcz in 2F lower nibble is greater than 9.
2F - 06 = 29

Imp 13) AAA (ASCII adjustment for addition)

Case I: If there was no carry, CF & AF are cleared & all register is unchanged and clear the higher nibble of AL.

80 + 45 = CF (No carry) so AH = 00,
AF = CF = 0, AL = 05

Case II: If addition produces a decimal carry, then increment AH by 1 & CF & AF are set.

80 + C5 = 145 (decimal carry generator)
so AH = 01, AL = 45, AF = CF = 1

Case III: If addition produces a decimal carry and lower nibble is greater than 9, then increment AH by 1 and CF & AF are set clear higher nibble of AL & add 06 with lower nibble. 88 + 05 = 14D (decimal carry) ~~generate~~

AH = 01, AF = CF = 1, AL = 0D
D + 6 = 13 ∴ AL = 03

NOTE: higher bit doesn't consider

14) AAS (ASCII adjustment for subtraction)

Case I: AH = 05, AL = 88
SUB = AL, CL
AA S;
88 − C1 = C7 (Borrow = 1)
AH = 05 − 01 CF = AF = 1 AL = 07
AH = 04 clear higher nibble of AL

By its
Words 16 bits
Double word 32 bits.

Case 2: 88 − 08 = 80 (No carry)
AH = 05, CF = AF = 0, AL = 00

15) CBW (Convert BYTE to word)

This instruction extend the sign bit of AL register into AH register.

Before execution

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	0	1	0	0

AH register

D7	D6	D5	D4	D3	D2	D1	D0
1	1	0	0	0	1	1	0

AL register

After execution
AH = 11111111
AL = 11000110

16) CWD (Convert Word to Double word)
copies sign bit of word in AX to all bits of DX.

Before execution

1	0	0	0	0	1	0	0	1	1	0	0	0	1	1	0

DX register.

1	0	0	0	1	1	0	0	1	1	1	1	0	1	1	0

AX register

After execution,

DX = 1111 1111 1111 1111

AX = 1000 1100 1111 0110

17] IMUL (Multiplication of 2 signed no.)

- Allows multiplication of 2 signed operands
- operands can be +ve/-ve.
- Operation IMUL & MUL are same

MOV AL, -04H (FC)

MOV BL, 4H

IMUL BL

BL * AL = -4 * 4

= $(-16)_{10}$

= (FO)H

AL = F0 AH = FF

When operand is in byte it's multiply with AL register and result is stored in AX register. When operand is word it is multiply with AX register and result is stored in DX, AX register.

18) AAM (ASCII adjustment For multiplication instruct)

MOV AL, 05H; ① | AH 00 | AL 35 | AAM

MOV BL, 07H; 35/10

MUL BL → AL = 23H Quotient = 03 remainder = 05

AAM → $AL = (35)_{10}$. ② | AH 03 | AL 05 |

RET

05H * 07H = 23H = $(35)_{10}$.

AAM instruction divides the content of AL by 10. After that it stores quotient in AH and remainder in AL.

19) IDIV. (Division of two signed no.s)

- MOV AL, -06H (FA)

MOV BL, 3H

DIV BL (IDIV BL) AX = -6/3 = -2 = -2H = FF

AL = FE AH = FF

1111 1110 1111 1111

signed bit

20) AAD (ASCII adjustment For Division)

When AAD is executed, the two BCD digits are combined into a single binary no. by setting

AL = (AH * 10) + AL

and clearing AH to 0,

eg CH = 09

MOV AX, 0205H (AH 02, AL 05)

AAD; AH = 0 and AL = 19H (25)

DIV CH, AL = Quotient = 02 } Decimal.

AH = Remainder 07

- Mix language program

```
#include<stdio.h>
int a, b, c;
void main()
{
  printf("Enter two nos");
  scanf("%d %d", &a, &b);
  c = a + b;
  printf("The addition is %d", c);
}
```

Output

```
Enter two nos 2 9
The addition is 11
```

#include <stdio.h>

c = a+b. ⇒ Convert using assembly language.

```
asm {
mov ax, a
mov bx, b
add ax, bx
mov c, ax.
};
printf("Result = %d", c)
getch()
```

MUL, DIV, SUB can also come.

- Logical instructions.

1] AND.

AND BL, AL

AL = 1001 0011 = 93H

BL = 0111 0101 = 75H

BL = 0001 0001 = 11H.

2] OR.

OR BL, AL

AL = 1001 0011 = 93H

BL = 0111 0101 = 75H

BL = 1111 0111 = F7H

3] NOT.

NOT AL.

AL = 1001 0011 = 93H

$\bar{AL}$ = 0110 1100 = 75H

4] TEST (logical AND)

TEST BL, AL.

AL = 1001 0011 = 93H

BL = 0111 0101 = 75H

Ans = 0001 0001 = 11H

Flag registers are set according to answer value. Result is not stored.