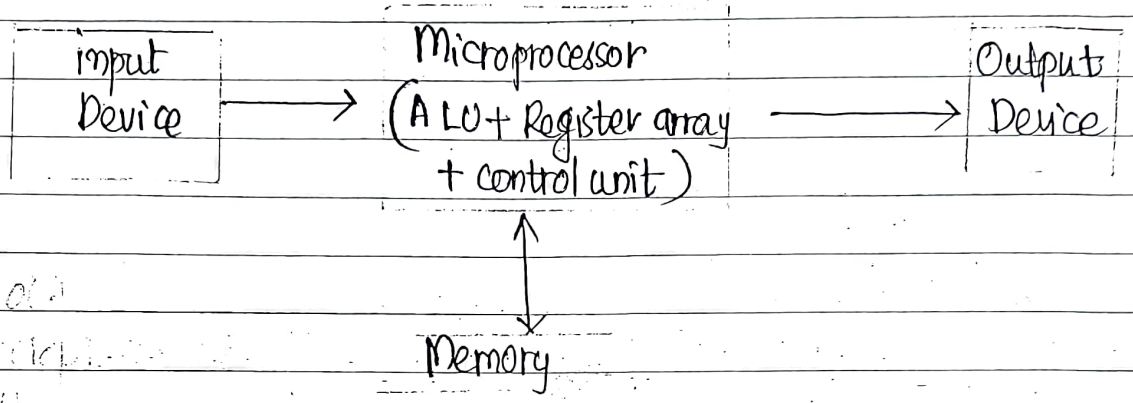
KTU NOTES

The learning companion.

KTU STUDY MATERIALS | SYLLABUS | LIVE NOTIFICATIONS | SOLVED QUESTION PAPERS

Module-1 - Evolution of microprocessors.

Microprocessors : 8085 & 8086



Microprocessor is a controlling unit of a micro-comput fabricated on a small chip capable of performing ALU (Arithmetic Logical unit) operations and communicati with the other devices connected to it.

History :

Invented :

In 1971, the first microprocessor Intel 4004

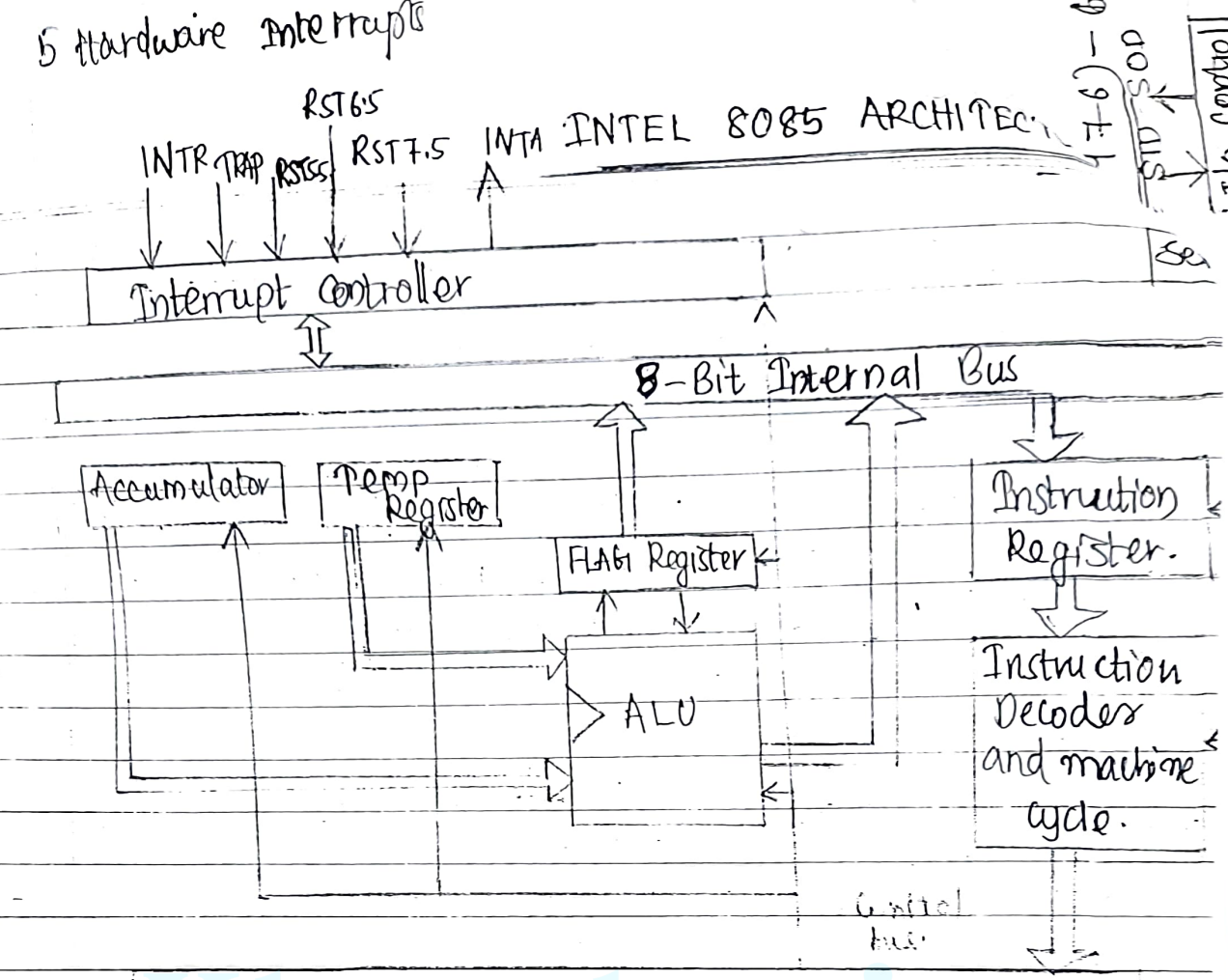
By:

- Federico Faggin.
- Marcian (Ted) Hoff
- Stanley Mazor
- Masatoshi Shima

Generations :

- 1st Generation - Intel 4004 - 4 bit microprocessor
- 2nd Generation - Intel 8008, 8080, 8085 - 8 bit microprocessor
- 3rd Generation - Intel 8086
- 4th Generation - Intel 80386, 80486, 80586 (pentium) - 32 bit
- 5th Generation - pentium dual core, core-i3, core-i5 - 64 bit

5 Hardware Interrupts



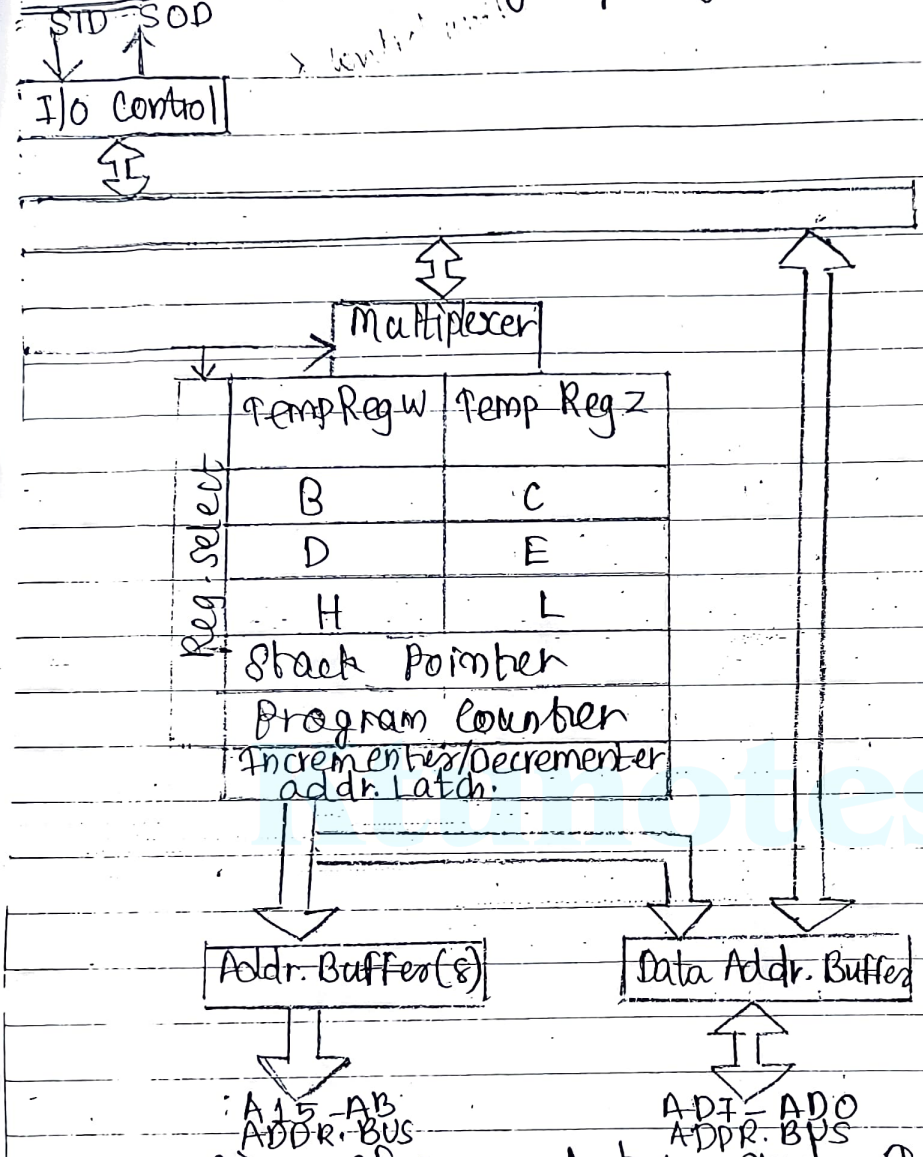
ALU: performing arithmetic and logical operations.

Register Array: Divides into General purpose registers, & special purpose registers, Temporary Registers

- GPR - used for storing data from user or programme
  - Register B, C, D, E, H, L having 8 bit size.
  - (Want to store higher Bits ... all them)
  - BC, CD, DE, EH, HL → pairing possible only like this
- Special purpose registers: . Register A
  - . Each register have a specific use.
  - . Result is always stored in Accumulator for eve operation & also one of the operand is in Accumu

Maximum memory capacity of 8085 is 64 (2^{16} bi

(1976) – 64KB memory capacity.



- Size of accumulator and Instruction Register is 8 bit.
- Current executing instruction will be stored in Instruction register.
- Flag Register – 8 bit size – use to represent the status of the currently executing Instruction or operation.

Having 5 1 bit flip flops which holds either 0 or depending upon the result stored in the accumulator.

Set of 5 flip flops are:- Sign (S), zero (Z), Auxiliary carry (AC), Parity (P), Carry (CY)

Carry generated in D_3 bit, Auxillary carry
Carry generated in D_7 bit, Carry part is

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z		AC		P		CY

S	Z	AC	P	CY
0	0	1	1	1

A 1000 1001
B 1001 1001
0010 0010

Stack pointer (SP) and program Counter (PC)
- Also special purpose register.

- Data Bus - Bi directional
- Address of a peripheral device is 8 bit size
- Processor want to read some data from input device.
- Address of memory location 16 bit
- Address Bus - Uni directional
- Control Bus - Uni directional

- Every memory location, we can store 1 byte of data.
memory location - 16 bit.
Processor take input from memory and write data/result to the memory.

- Memory : • Storing Program code.
  • Input data is stored.
  • Write Output data
- One part is used as stack
  Push and Pop takes place at top.
- Stack used during subroutine. ie, current status of main program stored in stack while

- Stack pointer stores address of the top of the stack which is of size 16 bits
- Program counter stores the address of the instruction to be executed next
  - of size 16 bits.
- PC increments upon executing the very first instruction whose address is stored in IR.
- Incrementer or decrement address Latch:
  - To increment or decrement address in PC or stack pointer
  - In PC only incrementing happens
  - stack pointer - both increment and decrement
- Temporary Register: Only accessed by processor and not by users (when ALU executes instructi
  - Used for storing temporary datas in operatio
- 3 temporary Registers ↑ ↑ going ALU
  1) Between Accumulator and flag Registers
  2) and 3) Above B and C - used to store tempora data between execution of 2 instructions
  Can pair W&Z of total size 16 bits

Temp Reg W&Z:2

- Instruction decoder and machine cycle encoding
  Every - instruction has 3 stages
  1) Instruction fetch: reading instruction from memory
  2) Instruction Decode - Understanding meaning of instruction (Adding/subtracting/other operatio
  3) Instruction Execute Running the instruction

- Instruction decoder takes instruction from IR and pass timing to control signal
- Timing & control unit: Receives the decoded instruction from decoder & generate control signals related to instruction.
- what are the control signal generated

1st control signal: i) RESET IN
used for reseting microprocessor (8085).
ie all register value will become 0.
eg: like factory Reset.

ii) RESET OUT:
- used by microprocessor
- resets all the connected devices.

iii) IO/$\overline{M}$
- decides whether processor is accessing I/O device or the memory.

If value is 1, then using I/O device
value is 0 → using memory.

iv) RD & WR
→ RD - processor reading (1), if 0, not reading
WR - writing (1)

v) READY: input signal to the processor which represents connected I/O device is ready for transferring data.

vi) X1, X2 & CLK OUT
- X1 & X2 are input signals (clock in)
- CLK OUT is o/p signal
- every device in microcomputer are of different speed.
- inorder to avoid waiting of microprocessor.
- to avoid that external clock given to

Address bits of I/O devices is 8 bits.

processor using pins X1 & X2 (to match speeds)
- 6 MHz is given (3 to X1 & 3 to X2)
- Through CLK OUT, given to connected devices.

vii) S1 & S0:

Shows status of operation

S_1	S_0	Operation performed on microprocessor
1	1	memory fetch / Instruction fetch
1	0	memory Read operation
0	1	memory Write.
0	0	Halt Operation or stop.

* Bus Structure

3 Buses
1) Address
2) Data
3) Control Bus.

- Address Bus: size or no. of pins in Address Bus is 16 Bits. : $A_0 - A_{15}$ (represented as)
- Data Bus of size 8 bits: $D_0 - D_7$

Address Bus divide into 2 (higher and lower address)
($A_0 - A_7$ and $A_8 - A_{15}$)
lower address combined to reduce size.

$AD_0 - AD_7$, $A_8 - A_{15}$

↳ multiplexing concept to reduce bits.

pins to 16 bits from 24 bits

A_8 – A_{15} : only pass address.
Lower stages pass both Address and data.

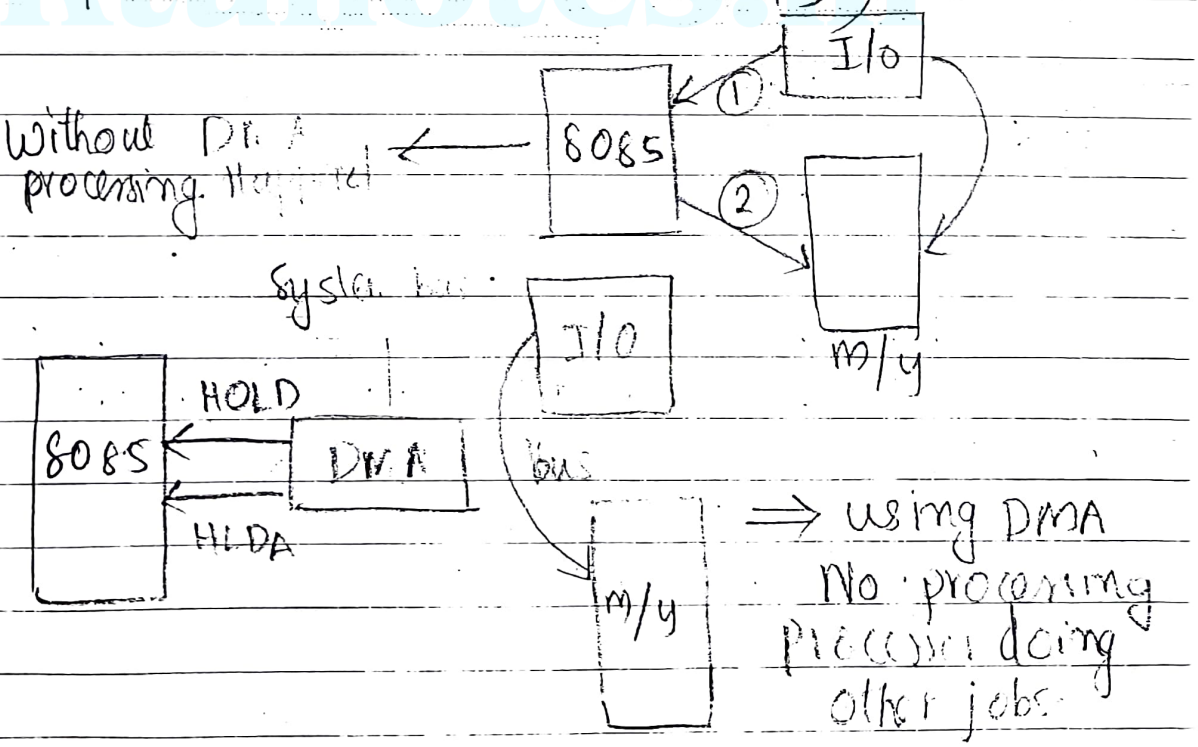
- Address Latch Enable : output signal
  - Shows the status of AD_0 – AD_7 lines or pins.

ALE = 1, AD_0 – AD_7 has the content as Address
ALE = 0, content in AD_0 – AD_7 as data.

- HOLD & HLDA.

HOLD → Input signal to the processor
HLDA → Output signal from the processor.

(want to pass data from I/O to memory
first pass data from I/O to processor,
From processor data pass to m/y)



HLDA - Hold acknowledging

00011000

- Serial IO Control:
  - can send data in serial or parallel mode.
  - to send serial input data - SID
  - to send serial output dat - SOD
    (At a time 1 bit)

- Interrupt controller:
  External request comes to the processor during the normal execution of programs

Types of Interrupt Request.

1) INTR
2) TRAP
3) RST5.5
4) RST6.5
5) RST7.5

} used for sending Interrupt request to the processor.

6) INTA → Interrupt Acknowledgement.
   Used for granting the request for the interrupt

If interrupt comes to the TRAP, it grants the request.

TRAP has highest priority and is non-maskable (Processor can't hide it)

Processor can hide or ignore INTR, RST5.5, RST6.5 and RST7.5

- There is some buffer associated with Address bus and databus. (It no registers mainly use latches)
  - Address Buffer
  - Data Buffer

INTEL 8086 MICROPROCESSOR:

- 8086 (Also called iAPX 86)
- Designed & released by intel in 1978.
- 16 bit microprocessor
- has a 16 bit data bus : $D_0 - D_{15}$
- Address bus of 20 bit : $A_0 - A_{19}$
- memory capacity - 1MB

Can multiplex Address bus with data bus

$A_0 - A_{15}$ $A_{16} - A_{19}$

$D_0 - D_{15}$

$AD_0 - AD_{15}$ $A_{16} - A_{19}$

1 bit - 2 location

→ 0
→ 1

- $2^{\text{no. of bits}}$ → No. of locations

2 bits → 4 locations

00
01
10
11

there 2^{20} bits

$= 2^{10} \cdot 2^{10}$ bits

$= 2^{10} \cdot 1KB$

$= 1MB$

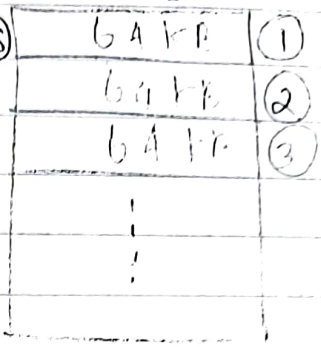
3 versions of 8086, depending on clock speed:

1) 8086 → 5MHz
2) 8086-2 → 8MHz
3) (C) 8086-1 → 10MHz

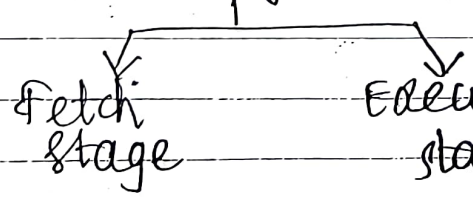
- Supports memory segmentation.
  - 1MB divided into segment each of size 64kB

∴ there are total $\frac{2^{20}}{2^{16}} = 2^4 = 16$ segments

① Code segment – storing program/code (CS)
② Data segment – storing data (DS)
③ Stack segment – used as stack m/y (SS)
④ Extra segment (ES) → used for storing string data



- General purpose register based processors.
  4 general purpose registers:
  1) AX – used as accumulator with size 16 bit
  2) BX
  3) CX
  4) DX
- 2 modes:
  1) minimum mode
  2) maximum mode ∴ co-processors availa in this mod
- contains 29000 transistors
- Uses two stages of pipelining

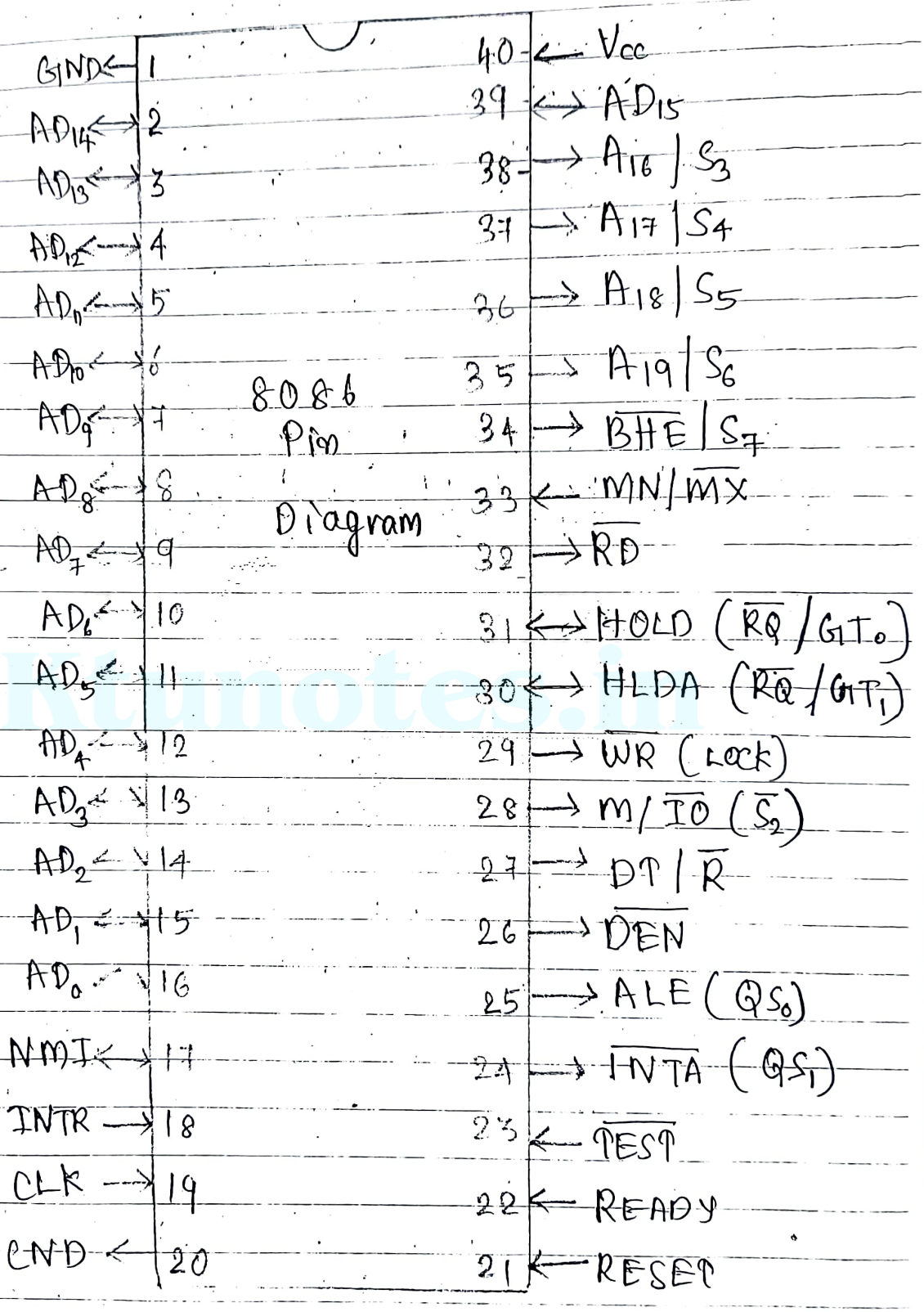


: which improves performance.

using parallel or pipelining concept
- At the time of execution of first instructi next instructions fetching happens.
  (8085 – sequential)

- 8086 can acess $2^{16} = 65,536$ I/O's
  Address bus of I/O device are of 16 bits.

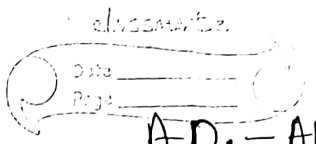
8086 PINS Configuration



24 – 31 pins depends on mode

minimum modes are represented by:

: $\overline{INTA}$, ALE, $\overline{DEN}$, DT/$\overline{R}$, M/$\overline{IO}$, $\overline{WR}$, HLDA, HOLD



$AD_0 - AD_1$

$AD_0 - AD_7$ $AD_8 - AD_{15}$

Lower Data Bus

maximum modes:

QS_1, QS_0, $\overline{S_0}$, $\overline{S_1}$, $\overline{S_2}$, LOCK,

Pin 1 & 20 represents ground

40 represents V_{cc} → power supply.

pin 2 – 16 represent address or data bus

ie $AD_0 \rightarrow AD_{14}$; Pin 16 – 2

$A_{16} - A_{19}$ - multiplexed with the status signals. (pins 35–38)

ie multiplexing of status signal & Address lines.

pin 22 - READY - connected device is ready or not to transfer data

Two interrupts pin in 8086

NMI - Non maskable interrupts - pin 17 (process cannot ignore interrupt)

pin 18 represents INTR.

pin 24 : $\overline{INTA}$ - acknowledgement

pin 33 : MN/$\overline{MX}$: two modes, if pin 1 or 0

MN represents minimum (pin 33 is 1)

$\overline{MX}$ represent maximum (pin 33 is 0)

pin 32 : $\overline{RD}$ used for reading.

pin 34 : $\overline{BHE}$: Bus High Enable

Represents whether the higher data bus ($AD_8 - AD_{15}$) contain data or not.

$\overline{BHE} \rightarrow 0$ $AD_8 - AD_{15}$ contains data

$\overline{BHE} \rightarrow 0$, does not contain data in $AD_8 - AD_{15}$.

pin 23 : $\overline{TEST}$ - lower active

8086 transfers control to other external devices during execution for the execution of external devices. After the execution in the external devices, the signal is produced TEST and give back the control.

From 24 to 31 → mode dependant signals.

Minimum mode pins:

$\overline{INTR}$: interrupt acknowledgement in minimum mode : pin 24

pin 25 ALE: Address latch Enable : whether AD bus $AD_0 - AD_{15}$ contains address or data.
ALE = 1, then content in $AD_0 - AD_{15}$ is address
ALE = 1, then content in $AD_0 - AD_{15}$ is data.

$\overline{DEN}$: Data Enable pin 26
Data is available in the buffers for transmi -ng or receiving
$\overline{DEN}$ = 0, Data is in buffer
$\overline{DEN}$ = 1 → Data not in buffer

pin 27 → DT/$\overline{R}$ (Data Transmit/Receive)
Whether processor transmit or receive data
1 0

M/$\overline{IO}$ = 0 → Accessing IO devices. - pin 28
M/$\overline{IO}$ = 1 → Accessing memory

pin 29 → $\overline{WR}$ - writing
pin 30 & 31 are HLDA & HOLD
requesting system bus using DMA controller

Maximum mode pins:

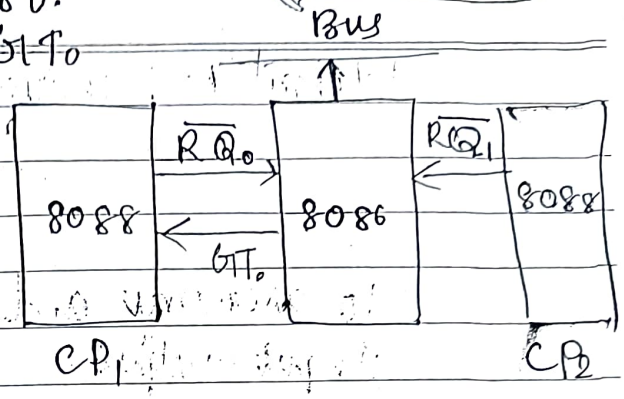
pin 31 → ($\overline{RQ}$/GT$_0$)
Pin 31 → ($\overline{RQ}$/GT$_1$)

two or more processor are there in maximum mode, system bus is common and 8086 has control and other processors request 8086 throug

$\overline{RQ_0}$/ GT_0 (Pin 31) - Coprocessor request system bus through $\overline{RQ}$ to 8086. grant request through GT_0

Request & grant

$\overline{RQ_1}$/ GT_1 (pin 30) - another coprocessor request bus through $\overline{RQ_1}$ & grant through GT_1.



LOCK: pin 29
is used by the microprocessor to lock the system Bus

Status signals S_2 S_1 and S_0 : pins 26, 27, 2

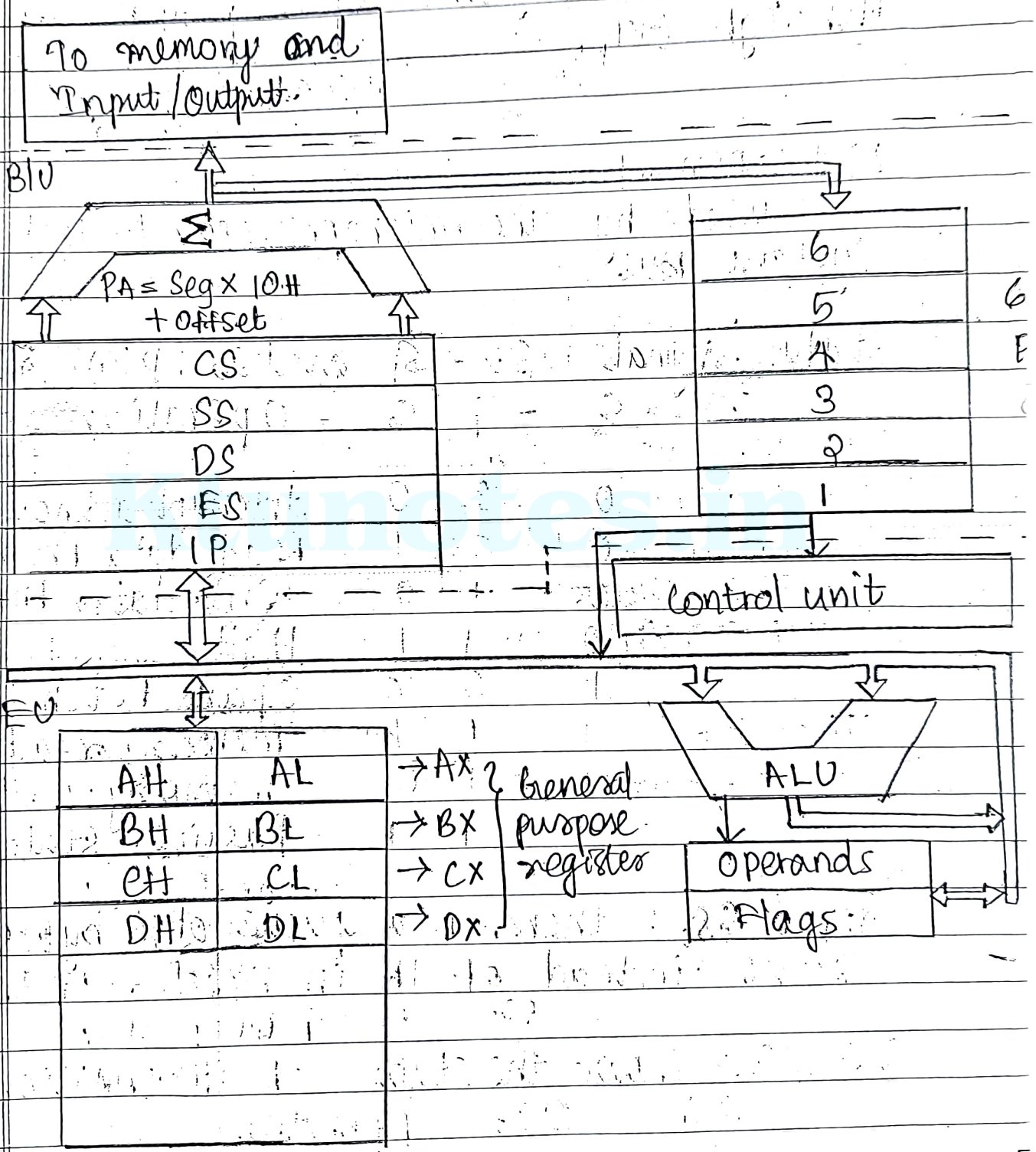
S_2	S_1	S_0	Operation
0	0	0	Interrupt acknowledgment
0	0	1	Read data from I/O po
0	1	0	Write data from I/O por
0	1	1	Halt
1	0	0	Opcode fetch
1	0	1	Memory read
1	1	0	Memory write
1	1	1	Passive state.

QS_1 & QS_0 : there is a queue of 6 bytes in 8086 (instead of IR in 8085) → Instruction Queue

QS_1 & QS_0 shows the status of Instruction Queue

QS_1	QS_0	Operation
0	0	No operation
0	1	1st byte of opcode from queue
1	0	Empty the queue
1	1	Subsequent byte from queue

Internal Architecture of 8086



fill ചെയ്യാനുള്ള 2 slot എടുക്കുന്നു free ആയിരിക്കും.
2 byte = 16 bits (address bus)

Entire architecture is
Divided into two units:
1) BIU - Bus interface unit
2) EU - Execution unit.

BIU - communicated with I/O device through the system bus.
eg: data/Instruction taken from input devices through bus interface unit.
EU - used for the execution of instruction.

byte fetch queue

BIU

contains 6 byte instruction queue or pre-fetch queue. When 1st instruction executed in EU, then 6 byte of instruction copied to prefetch queue.

It contains 5 registers - 16 bits.
They are CS, SS, DS, ES, IP

CS

- Starting address of code segment is F0000H
- Code segment Register
- Stores the starting address of code segment. (Base address)
- Register of size 16 bits. But starting address of CS is 20 bits. We will remove the last 4 bits and then it will become 16 bits size and then stored in CS.
- Remove zero and store F000H (16 bits)

FFFFFH
F0000H
00000H

Code segment	64
Data segment	
Stack segment	
Extra segment	

↳ Address start with total 20 bits. (for each location having address)
1 MB

- Actual location's address is of size 20 bits (starting address)
- Each segment has a register in BIU.

Extra Segment:

Starting address of ES is E0000H and Ending address is EFFFFH.
∴ Content in ES is E000H (since Starting address is 20 bits and ES can afford on 16 bits, so removing 4 bits)

IP - Instruction Pointer:

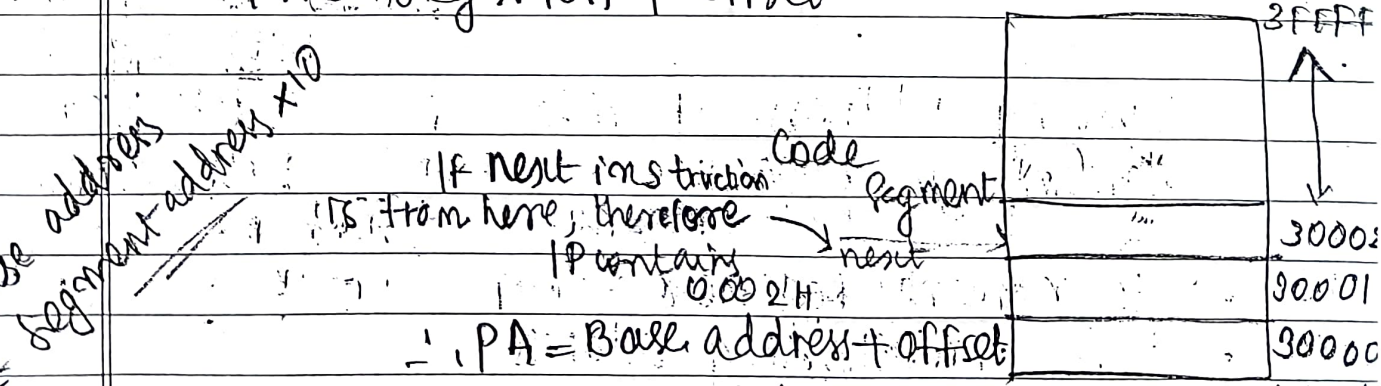
- Stores address of next instruction to be execute

Σ : Sigma.

Calculating the physical address using segment address and offset

PA = Seg × 10H + Offset.

Base address = segment address × 10



∴ PA = Base address + offset

30002H = 30000 + 0002H

CS	3000H	IP	0003H

PA = Segment Address × 10H + offset address

= 3000 × 10H + 0003H

$= 30000 + 0003$

$= 30003 \longrightarrow AD_0 - AD_{15}, A_{16} - A_{19}$

Use of BIU:

- Interact with System Bus.
  - Read data from I/O or memory
  - Write data from I/O or memory
  - physical address generation

$PA = Seg \times 10H + offset$

EU: Execution unit

General purpose registers:
- AX, BX, CX and DX
- Each having size 16 bits no pairing, but can be divided.

(16) AX → A_H (8), A_L (8)
(16) BX → B_H, B_L
(16) CX → C_H, C_L
(16) DX → D_H, D_L

AX → Also called accumulator having a special purpose

IN 0008H (moving the content and no register mentioned so storing in Accumulator)

→ Mov AX, 0002H

A_H 00
A_L 02

MOV A_L, 03H
→ MOV A_H, 08H

00021
00014
00000
offset address

BX → Also known as base register
CX → Can be used as counter in some case of Instructions.

During Division, Remainder → DX
Quotient stored in AX

Top - * SP (Stack pointer)
- Stack pointer stored in stack segment.
- Shows offset value in stack segment

* BP (Base Pointer)
- Stores offset of stack and data, it can be used.

stores any offset

Index Registers:

In data & extra segment

* SI (Source Index) - It is an offset used in DS
* DI (Destination Index) - assign in Extra segme
stores offset corresponding to ES (DI & SI can be used in both DS and ES)

SP, BP → pointers

Flag Register:
- 16 bit size
- 4 additional
total number of flags are 9
7 bits are unused (used for future purpose)
9 flags are classified into two categories
• Conditional flag:
These flags are set or reset when the result of the operation satisfy some condition
1) Carry flag (bit number D_0)
- Represented by CY
- If operation in ALU set, then 1, then CY=
2) Parity (D_2)
- This flag is set when the operation result contains even number of 1's
3) Auxillary carry
In case of auxillary carry when the lower nipple generate some carry then auxillary carry will be set.

4) Zero flag – D_6
- Represented as Z
- when result of operation is zero, the zero flag will be set.

5) Sign flag – S – D_7
when result is -ve number, sign flag will be set.

6.) (Bit Number – D_{11}) – Over flow flag
This flag will be set when result of the operation exceeds the system capacity.

- Control flags

These flags actually used for controlling the processor. So they are called machine control flags. 3 control flags

1) D_9 – I – Interrupt flag
It will be set, then processor can accept interrupts.

2) D_8 – T – Trap flag
When this flag is set, the operation is performed in single step control (debugging)

3) D_{10} – D – Direction flag
Used in string handling whether string read from left to right(1) or right to left(

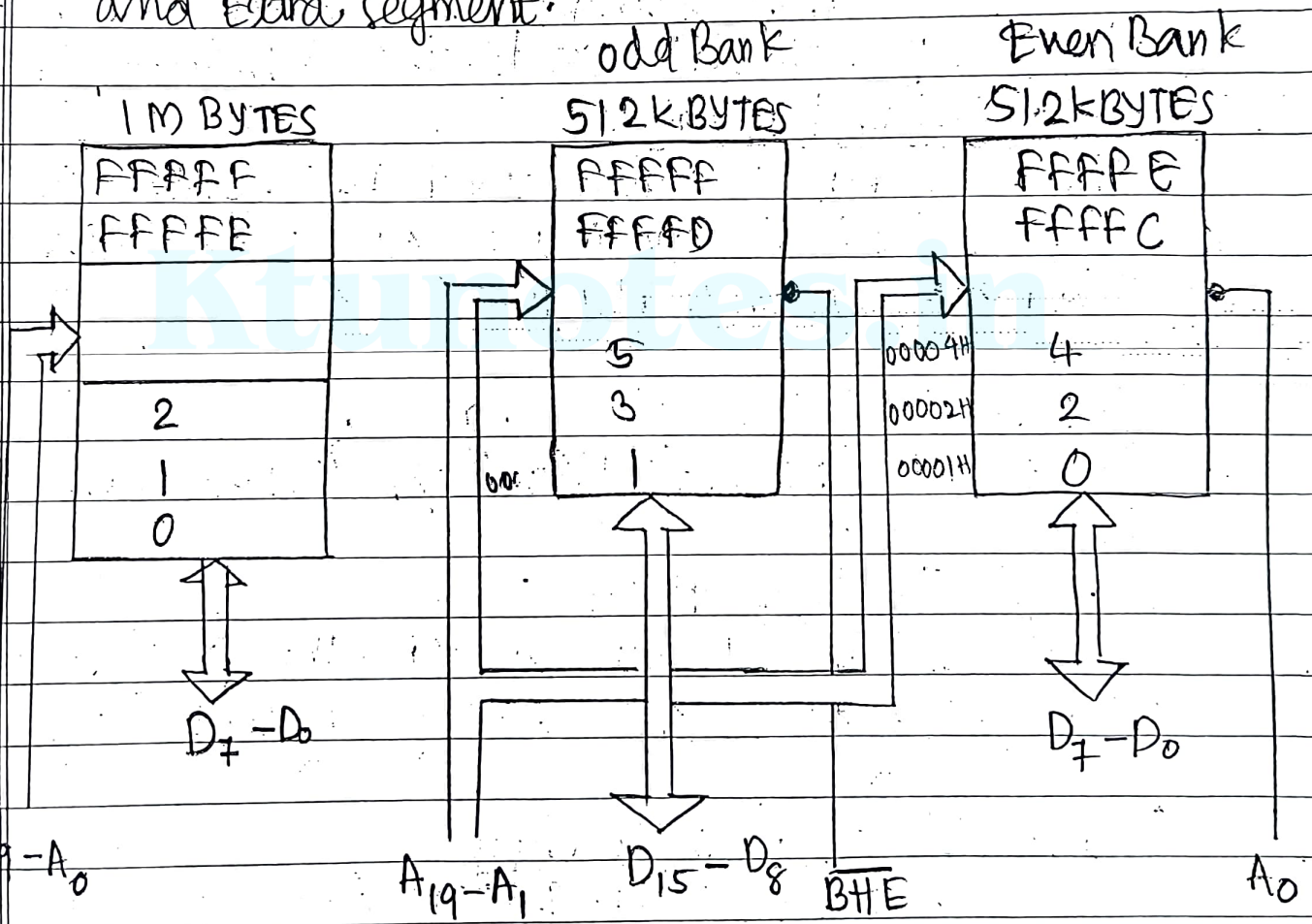
D_{15}	D_{14}	D_{13}	D_{12}	D_{11}	D_{10}	D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
				0	D	I	T	S	Z		AC		P		CY

Memory Organisation in 8086

Dividing memory organisation in 8086 into

1) Conceptual / Logical memory organisation
2) physical memory organisation.

- 1MB - divided into 16 segments of 64kB.
- Code segment, Data segment, stack segment and Extra segment.



(Even and odd memory banks of 8086)

* FFFFEH E = 14

$A_0 - A_{19}$

$A_{19} A_{18} A_{17} A_{16} \ldots\ldots A_3 A_2 A_1 A_0$

1 1 1 1 0

Even Address

↳ $A_0 = 0$

When $A_0 = 0$, then address Data is in $D_7 - D_0$ pins.

$\overline{BHE} = 0$, data present in $D_8 - D_{15}$

∴ Data in $D_0 - D_{15}$

* FFFFCH

$A_0 - A_{19}$

$A_{19} A_{18} A_{17} A_{16} \ldots A_3 A_2 A_1 A_0$

1 1 0 0

C = 12 1100

Even Address

↳ $A_0 = 0$

If $A_0 = 0$, then address is in even Bank

If $A_0 = 1$, then address is in odd Bank.

* FFFFB

Value of $A_0 = 1$

then stored in Odd Bank

B = 1011

$\overline{BHE}$	A_0	(16 bit)
0	0	Data is in $D_0 - D_7$ and $D_8 - D_{15}$
0	1	Data is in odd bank : $D_8 - D_{15}$
1	0	Data is in $D_0 - D_7$: Even bank
1	1	No operation. (No data)

A_0 → Used for Enabling Even Bank

$\overline{BHE}$ → Used for Enabling odd Bank.

Memory Segmentation:

Segment	Offset Registers	Function
CS	IP	Address of next Instruction
DS	BX, DI, SI	Address of data
SS	SP, BP	Address in the stack
ES	BX, DI, SI	Address of destination data (for string operations)

physical address = Segment Address * 10H + offset

- The content of following registers are,

CS = 1111H DS = 1678 H ES = 1298H SS = 6789H
IP = 6721 BX = 7865H DI = 1235H BP = 7821H

Calculate the corresponding physical address bytes in CS, DS, ES and SS.

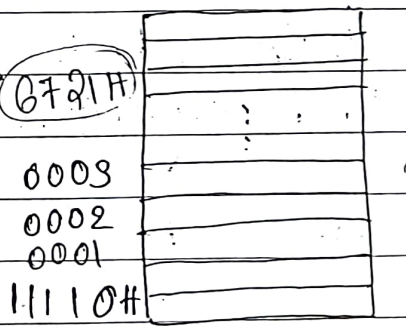
CS | 1111H | IP | 6721H |

PA_{CS} = Segment Add X10H + offset

= 1111H X 10H + 6721H

= 11110H + 6721H

PA_{CS} = 17831H



DS | 1678H | BX | 7865H |

PA_{DS} = 1678H X 10H + 7865H

= 16780H + 7865H
= 1DFE5

16780 +
7865
1DFE5

PA_{ES} = 1298H × 10 + 1235H

= 12980H + 1235H
= 13BB5

12980
1235
13BB5

PA_{SS} = 6789H × 10 + 7821
= 67890 + 7821
= 6F0B1

67890 +
7821
6F0B1

- A memory location has physical address 80FD2H. In what segment does it have offset BFD2H

P.A = Seg.Add × 10H + Offset.

∴ Seg.Add = $\frac{PA - Offset}{10H}$

80FD2H − BFD2H / 10H
= 75000H / 10H = 7500H

80FD2
BFD2
75000

∴ Segment Address = 7500H

Starting physical addess = 75000H

- Calculate the physical address corresponding to logical address D470H in the Extra Segment. Repeat for logical address 2D90H in the

Stack segment. Assume that segment definitions ES = 52B9H and SS = 5D27H

PA = Seg. Add * 10H + Offset

PA_{ES} = 52B9H × 10H + D470H

= 52B90H + D470H

= 60000H

PA_{SS} = 5D27H × 10H + 2D90H

= 60000H

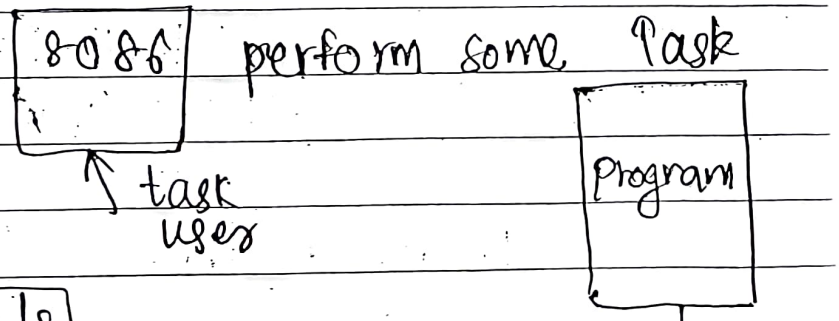
Segment definition = segment address

```
  1 1 1
  52B90+
   D470
  -----
  60000
```

```
  1 1 1
  5D270+
   2D90
  -----
  60000
```

ADDRESSING MODES IN 8086

The different ways in which a source operand is denoted in an instruction is known as addressing modes.



Instruction:

Opcode	operands

Eg: ADD AX, BX

AX → Destination, BX → Source

opcode operands.

mov AX, BX

1. Immediate addressing mode

The addressing mode in which the data operand is a part of the instruction itself is known as immediate addressing mode.

- Example

```
MOV CX, 4929H → data
ADD AX, 387FH
MOV AL, FFH
```

2. Register Addressing mode

It means that the register is the source of an operand for an instruction.

Eg: MOV CX, AX — move data in AX (is stored) to CX

ADD BX, AX — (content in AX added with BX and stored in BX)

3. Direct Addressing mode

The addressing mode in which the effective addre of the memory location is written directly in the instruction.

Example

ADD AL, [1005H]

offset / Effective Address

AL [03H]

DS [5000H]

1008	02H
1007	03H
1005	05H
00H	

PA = Seg x 10H + offset

= 5000 x 10 + 1005H

= 50000 + 1005H

= 51005H

AL = 05H + 03H = 08H

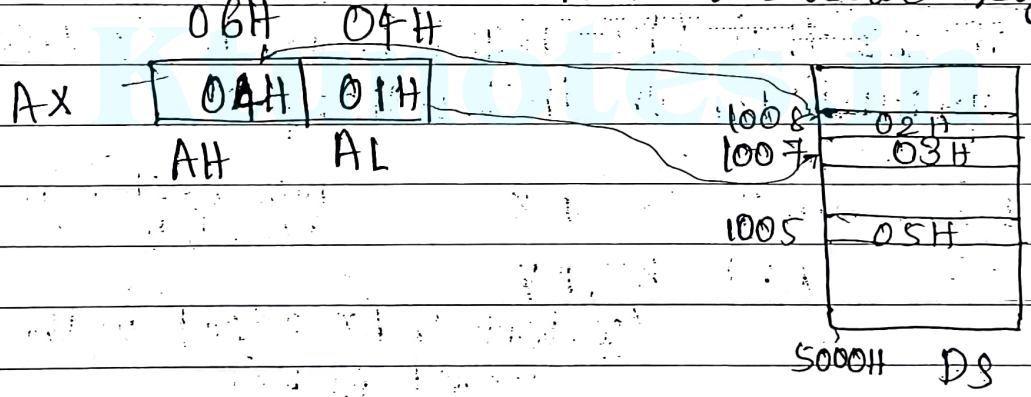
AL 08H

stored in $A_0 - A_{19}$

ADD AX, [1007H] — size of AX is 16 bit.

PA = 51007H

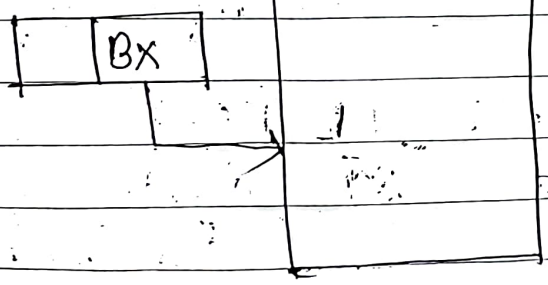
We need 16 bit data so we take next data from the data segment



Register Indirect addressing mode

Here the offset address is stored in a register and that register is represented in the instruction.

① ADD AX, [BX]



AX 0509H

BX 1005H

AL ← AL ← PA

AH ← AH ← PA+1

Content in BX is considered as offset address.

Example.

MOV AX,[BX] : Suppose the register BX contains 4895H, then contents of 4895H are moved to AX.

5. Based addressing mode

In this addressing mode, the offset addr of the operand is given by the sum of contents of the BX / BP registers and 8 bit/ 16 bit displacement.

Eg: MOV DX,[BX+0004]

BX, BP

displacement

BX 1004H 0003H

Offset 1004 + 0003 = 1007H

The displacement will be either 4 bit or 8 bit

MOV AX,[BX+04H] - 8 bit.

6. Indexed addressing mode

In this addressing mode, the operands offset addr... found by adding the contents

of SI or DI register and 8 bit [16 bit displacements

```
MOV BX, [SI + 0016]
ADD AL [DI + 0016]
```

1) MOV AX, 0016H
2) ADD AX, [0023H]
3) ADD AL, [0108H]
4) ADD AX, [CX] → offset present in CX
5) ADD AX, [BP + 0108H] → 16 bit displacement
6) ADD AL, [BX + 1C1H] → 8 bit displacement
7) ADD BX, [SI + 0016H] → 16 bit

8 find the addressing modes of each instruction

→ 1) Immediate Addressing mode
2) Direct Addressing mode → 16 bit
3) Direct addressing mode → 8 bit
4) Register Indirect addressing mode
5) & 6) Based addressing mode
7) Indexed addressing mode.

7) Based-Index addressing mode

In this addressing mode, offset addres of the operand is computed by summing the base register to the contents of an Index register.

```
ADD CX [BX+SI]
ADD AX [BX+DI]
```

8. Based indexed with displacement mode

In this mode, the operands offset is computed by adding the base register contents, An index registers contents and 8 or 16 bit displacement.

```
MOV AX, [BX + DI + 0008]
ADD CX, [BX + SI + 0016]
```

9. Implied addressing mode (Implicit)

Instructions using this mode have no operand The instruction itself will specify the data to be operated by the instruction. (not the operand)

Eg:

```
CLC // clear carry flag.
STC // set carry flag.
```

10. Relative Addressing mode

In this addressing mode, the effective addre of a program instruction is specified relativ to Instruction Pointer (IP) by an 8-bit signe displacement

Eg: JZ 0AH Jump zero 0AH

If zero flag one jump to 0AH

Like goto in C.

IP [1000H]

0AH

JNC - Jump No Carry
JC - Jump Carry.

11. Direct I/O port Addressing.

These addressing modes are used to access data from standard I/O mapped devices or ports.

In direct port addressing mode, an 8 bit port address is directly specified in the instruction.

Eg:

- IN
- OUT

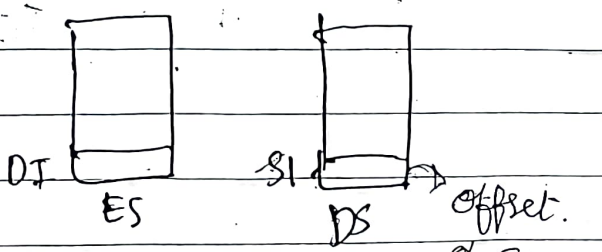
IN AL, [09H] Reading data.
OUT [0CH] Writing.

12. String Addressing.

Employed in string operations to operate on string data. The effective address (EA) of source data is stored in SI register and the EA of destination is stored in DI register. Segment register for calculating base address of source data is DS and that of the destination data is ES.

Eg:

MOVS BYTE

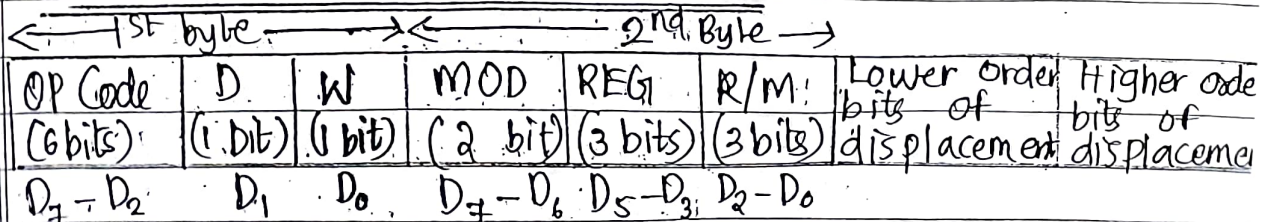


Offset : SI

Comparison between 8086 & 8088 Microprocessor

8086 microprocessor	8088 microprocessor
• The data bus is of 16 bits.	• The data bus is of 8 bits
• It has 3 available clock speeds (5MHz, 8MHz (8086-2) and 10 MHz (8086-1)).	• It has 2 available clock speeds (5MHz, 8MH:
• The memory capacity is 512kB chips	• The memory capacity is implemented as a single 1 mx8 memory banks
• It has memory control pin (M/IO) signal	• It has complemented memory control pin (IO/M signal of 8086
• It has Bus/Bank High Enable (BHE) signal	• It has status signal (SS0
• It can read or write either 8-bit or 16-bit word at the same time	• It can read only 8-bi word at the same tim
• Input/output voltage level is measured at 2.5mA	• Input/Output voltage level is measured at 2.0mA
• It has 6 byte Instruction queue	• It has 4 byte instructi queue as it can fetch only 1 byte at a time.

Instruction Format in 8086



OP Code (6 bits)	D (1 bit)	W (1 bit)	MOD (2 bit)	REG (3 bits)	R/M (3 bits)	Lower order bits of displacement	Higher order bits of displacement
$D_7 - D_2$	D_1	D_0	$D_7 - D_6$	$D_5 - D_3$	$D_2 - D_0$		

• every instruction that is executed in 8086 microprocessor, an instruction is binary representation of that instruction. format is available that

1st Byte

This instruction format can be coded in 2 bytes depending upon the addressing mode for instruction.

D-Bit:

D Stands for direction.

If D=0, then direction is from register.
If D=1, then the direction is to register.

Register as destination.

Eg: MOV BL, AL → Source. D=0
MOV BL, [0103H] D= → Destination

W-Bit:

W Stands for word.

If W=0, then only a byte is being transferred ie, 8 bits.
If W=1, then a whole word is being transferred; ie, 16 bits.

The opCode stands for Operation code. Every instruction has a unique 6-bit opcode. For example, the opCode for MOV is 100010.

2nd Byte

MOD:

D_7	D_6 D_5 D_4	D_3 D_2 D_1	D_0
MOD	Reg	R/M	

Value of MOD = 00 → no displacement

CODE	EXPLANATION
00	Memory mode, no displacement follow
01	Memory mode 8 bit displacement follow
10	Memory mode, 16 bit displacement follow
11	Register mode (no displacement)

D_6	D_7	
0	0	MOV AX,[BX]
0	1	MOV AX,[BX+08H]
1	0	MOV AX,[BX+080CH] 16b
1	1	MOV AX, BX

Reg: means Register

REG	W=0	W=1
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

R/M :

R/M	Operand Address
000	EA = (BX) + (SI) + Displacement
001	EA = (BX) + (DI) + Displacement
010	EA = (BP) + (SI) + Displacement
011	EA = (BP) + (DI) + Displacement
100	EA = (SI) + Displacement
101	EA = (DI) + Displacement
110	EA = (BP) + Displacement
111	EA = (BX) + Displacement

Direct addre

1 Byte instruction:

eg: CLC (implicit)

only opcode (No D or W Bit)

1 Byte

opcode (8bits)

Opcode for CLC ⇒ 11111000 (F8 H)

eg:2: INC CX (increment)

increment content in CX register.

data in CX is 16 bits.

opcode	D	W
010000	0	1

(41 H)

from the register, D = 0

eg:3: INC CL

opcode	D	W
010000	0	0

(40 H)

2 Byte Instructions

Mov BL, AL

data moving to the register

→ Data bits

opcode	D	W
100010	1	0

(here considering BL register Otherwise D = 0.

MOD	REG	R/M
11	011	000

→ code of Register (Destination is AL)

register mode

code for BL

(8AD8 H)

ata taken from memory here

↓

MOV CL, [BX]

opcode for MOV = 1000 10

D = 1

W = 0

MOD = 00

REG = 001

R/M = 011

opcode	D	W	MOD	REG	R/M
100010	1	0	00	001	111

8A 0F

ADD AX, [SI]

opcode for ADD = 000000

D = 1

W = 1

opcode	D	W	MOD	REG	R/M
000000	1	1	00	000	100

MOD = 00

REG = 000

R/M = 100

0304

- MOV AL, [1234H]

opcode	D		MOD	REG	R/M	
100010	1	0	00	000	110	→ direct addre

34	12

∴ 8A 06 3412 (4 byte instruction)

D: Data coming to register : 1

16 bits : W = 1

- writing the instruction;

ADD AX, BX → register.

Find the machine code:

opcode for ADD is 000000

opcode	D	W	MOD	REG	R/M
000000	1	1	11	000	01

1st Byte — 2nd Byte.

03C3 H

- ADD AX, [BX] memory

opcode	D	W	MOD	REG	R/M
000000	1	1	00	000	111

03C7 H

- ADD CL, [SI + 08H]

operating in 8 bit

opcode	D	W	MOD	REG	R/M	lower byte
000000	1	0	01	001	100	08

higher byte

(024C 08H)

- ADD CX, [BP + SI + 1234H]

opcode	D	W
000000	1	1

MOD	REG	R/M	lower byte
10	001	010	34

higher byte: 12

(038A3412H)

Add content in AL with contents in memory location & store in the memory.

- ADD [SI+1234H], AL

000000	0	0
opcode	D	W

MOD	REG	R/m
10	000	100

34 12

(00843412H)

D: Data from register

MOD : 8+8 : 16 bits : 10

- ADD AL, [1234H] (no displacement here)

opcode	D	W
000000	1	0

MOD	REG	R/m
00	000	110

low byte higher byte

34 12

(00063412H)

Timing Diagram: (Graphical representation of process)

Two modes

1) minimum (8086 present)

2) Maximum (8086 and co-processors are there)

ADD AX, [1234H] : 8086

(1) Opcode fetch
(2) Read the content from EA 1234H
(3) Addition performs
(4) Store the result in AX

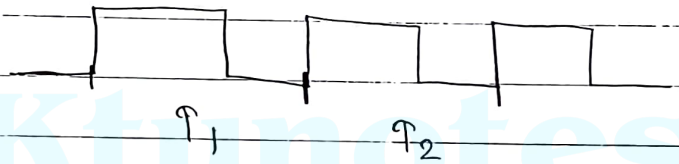
The total time taken to execute an instruction is

called instruction cycle.

Machine cycle:
Time required to execute the instructions such as I/O read or I/O write or M/y read or Memory Write.

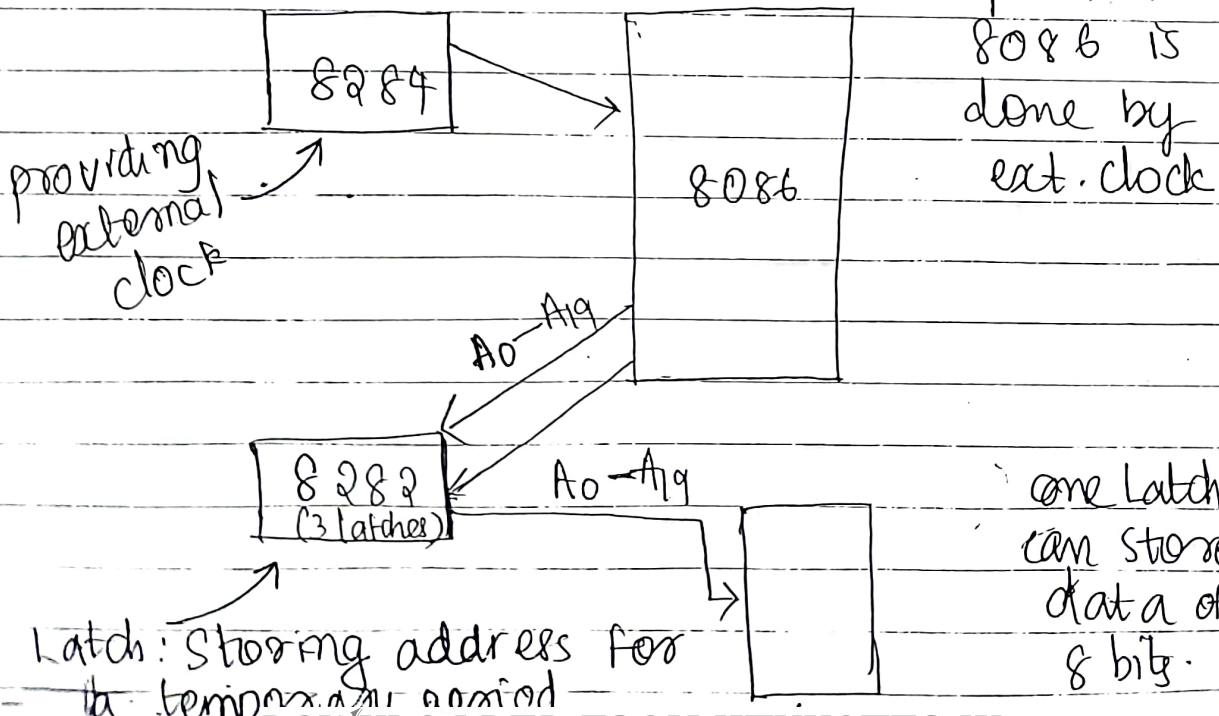
- 2 machine cycle in ADD AX, [1234H]
- ADD AL, BL : One machine cycle (Only opcode fetch)

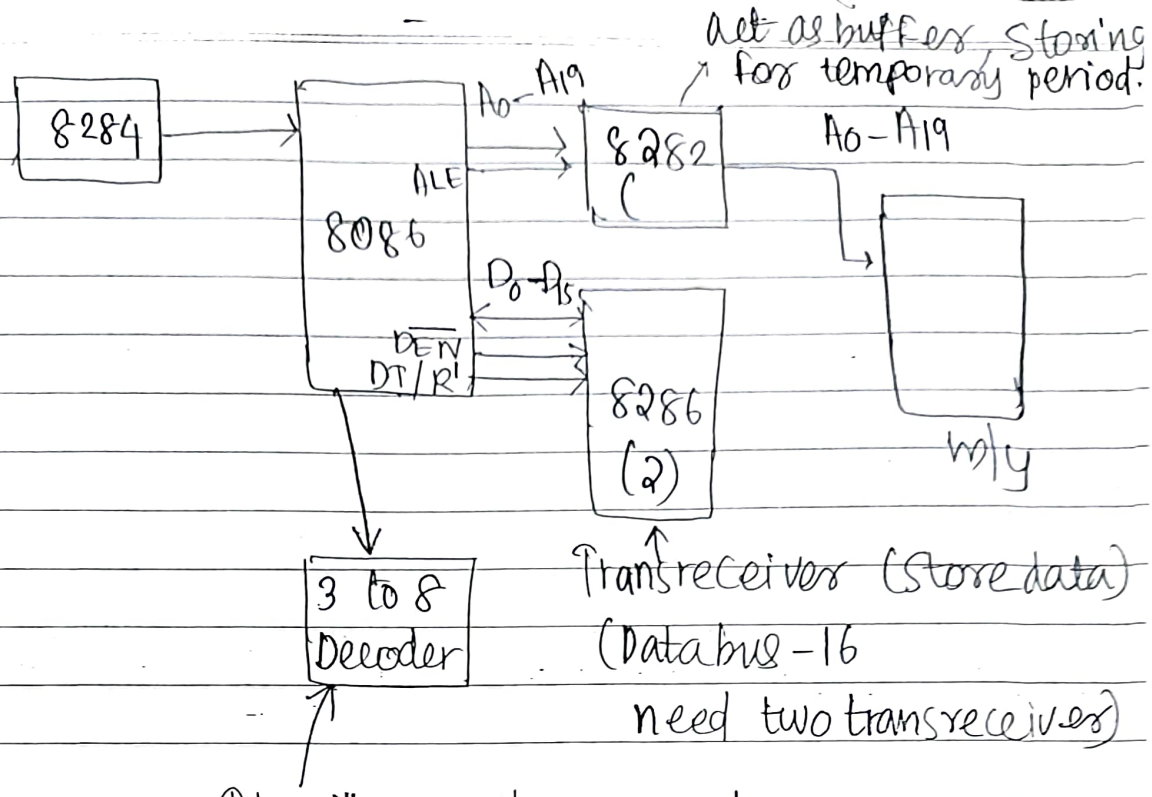
T-State:



20 bits & bits present

Minimum mode Timing diagram:





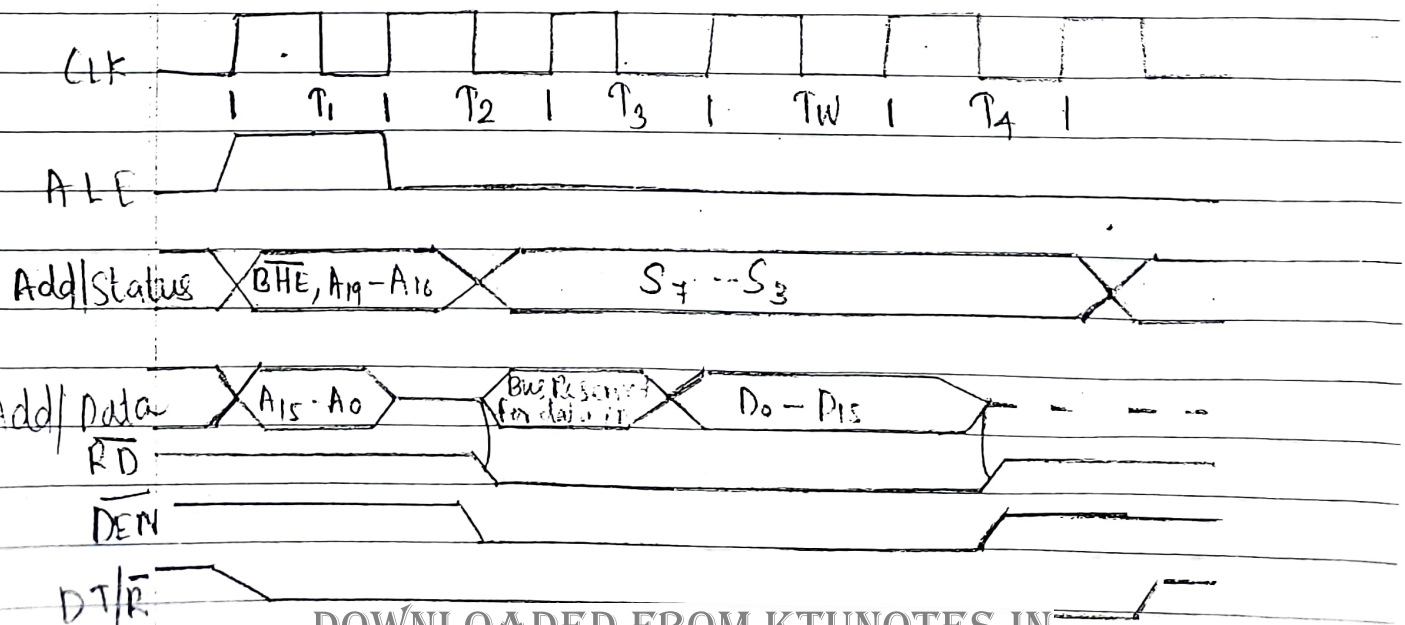
Act as buffer storing for temporary period.

Transreceiver (Store data)
(Databus - 16
need two transreceivers)

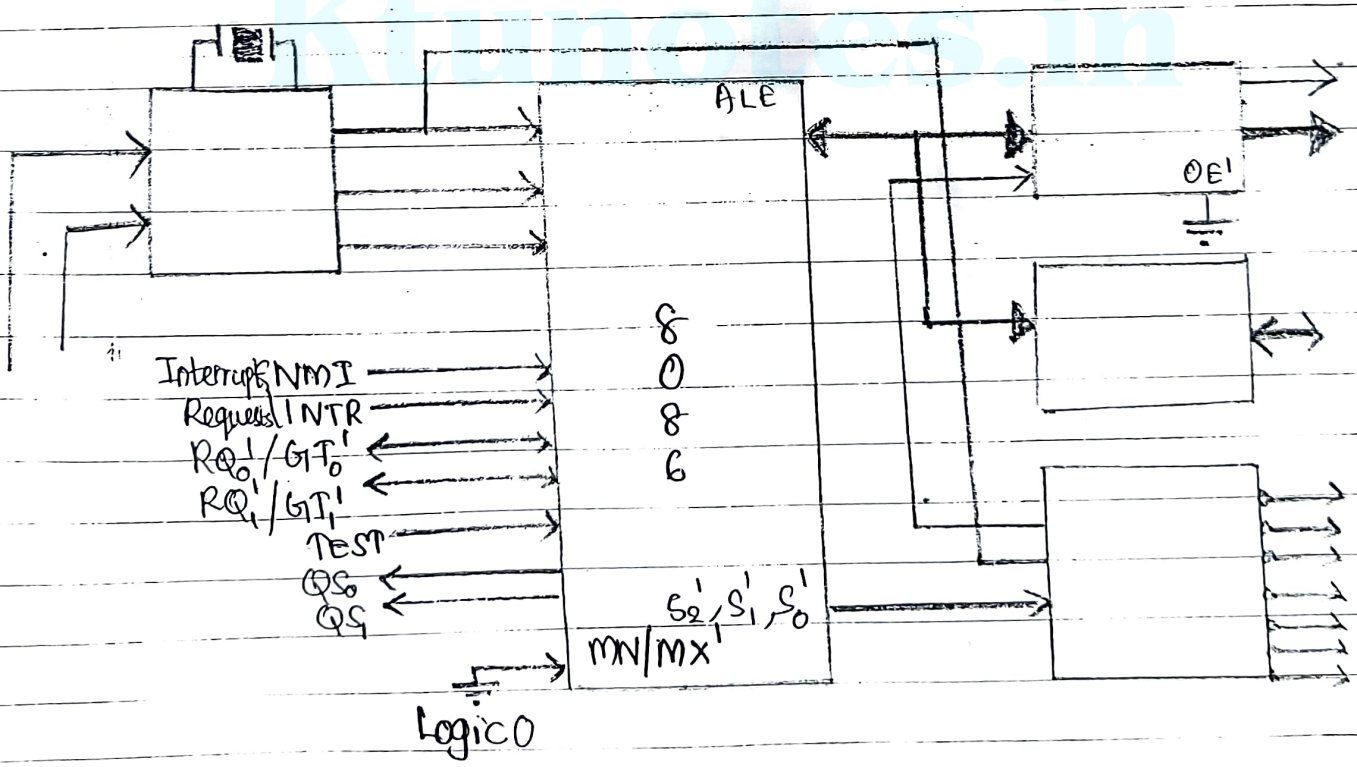
It will accepts 3 inputs and generates 8 outputs.
The inputs are IO/M, RD, WR that provide 4 outputs IOR, IOW, MEMR, MEMW (control signals)
(control bus)

DEN' or DT/R' - used for enabling Transreceiver.

Timing Diagram - Memory Read in minimum mod

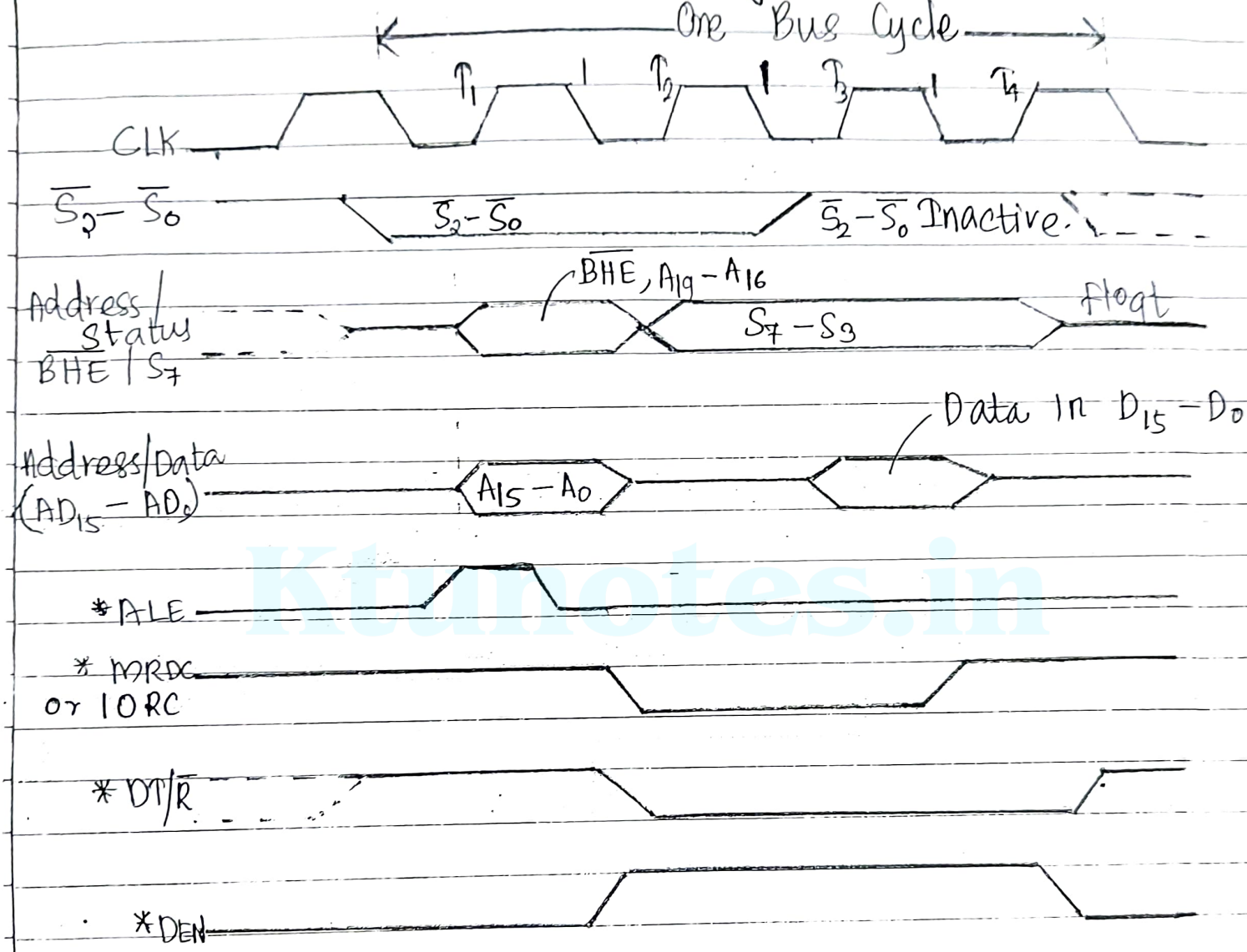


Maximum mode configuration of 8086 microprocessor



Controller generates * DEN, * DT/$\overline{R}$ etc. (8288) and not by 8086.

Bus controller is providing ALE



Input (Read Operation)

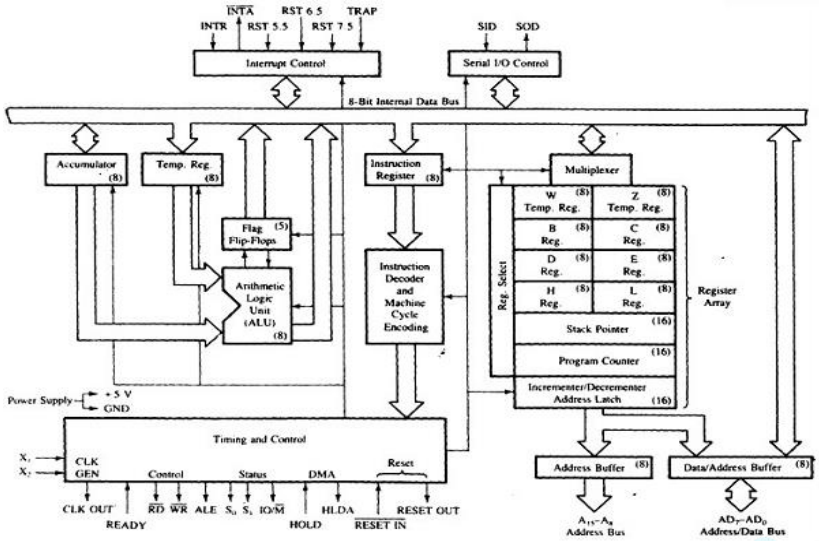
$\overline{DEN}$ → Min mode
DEN → Max mode.

MRDC
MWTC

MODULE 1 EVOLUTION OF MICROPROCESSOR

A microprocessor is a computer processor which incorporates the functions of a computer's central processing unit (CPU) on a single integrated circuit (IC), or at most a few integrated circuits The microprocessor is a multipurpose, clock driven, register based, digital-integrated circuit which accepts binary data as input, processes it according to instructions stored in its memory, and provides results as output. Microprocessors contain both combinational logic and sequential digital logic. Microprocessors operate on numbers and symbols represented in the binary numeral system.

8085 ARCHITECTURE



ALU: The Arithmetic and Logic Unit**,** ALU performs the arithmetic and logical operations:

- Addition
- Subtraction
- Logical AND
- Logical OR
- Logical EXCLUSIVE OR
- Complement (Logical NOT)
- Increment (add 1)
- Decrement (subtract 1)
- Left shift, Rotate left, Rotate right
- Clear, etc.

Timing and Control Unit: The timing and control unit is the section of the CPU.

- It is used to generate timing and control signals which are necessary for the execution of instructions.
- It is used to control data flow between CPU and peripherals (including memory).
- It is used to provide status, control and timing signals which are required for the operation of memory and I/O devices.
- It is used to control the entire operations of the microprocessor and peripherals connected to it.

Registers: Registers are used for temporary storage and manipulation of data and instructions by the microprocessor. Data remain in the registers till they are sent to the I/O devices or memory. Intel 8085 microprocessor has the following registers:

- One 8-bit accumulator (ACC) i.e. register A
- Six general purpose registers of 8-bit, these are B,C, D, E, H and L

- One 16-bit stack pointer, SP
- One 16-bit Program Counter, PC
- Instruction register
- Temporary register

In addition to the above mentioned registers the 8085 microprocessor contains a set of five flip-flops which serve as flags (or status flags). A flag is a flip-flop which indicates some conditions which arises after the execution of an arithmetic or logical instruction.

1. **Accumulator (ACC):** The accumulator is an 8-bit register associated with the ALU. The register 'A' is an accumulator in the 8085. It is used to hold one of the operands of an arithmetic and logical operation. The final result of an arithmetic or logical operation is also placed in the accumulator.
2. **General-Purpose Registers:** The 8085 microprocessor contains six 8-bit general purpose registers. They are: B, D, C, E, H and L register.
   To hold data of 16-bit a combination of two 8-bit registers can be employed.
   The combination of two 8-bit registers is called **register pair**. The valid register pairs in the 8085 are: D-E, B-C and H-L. The H-L pair is used to act as a memory pointer.
3. **Program Counter (PC):** It is a 16-bit special purpose register. It is used to hold the address of memory of the next instruction to be executed. It keeps the track of the instruction in a program while they are being executed. The microprocessor increments the content of the next program counter during the execution of an instruction so that at the end of the execution of an instruction it points to the next instructions address in the program.
4. **Stack Pointer (SP):** It is a 16-bit special function register used as memory pointer. A stack is nothing but a portion of RAM. In the stack, the contents of only those registers are saved, which are needed in the later part of the program.
   The stack pointer (SP) controls the addressing of the stack. The Stack Pointer contains the address of the top element of data stored in the stack.
5. **Instruction Register:** The instruction register holds the opcode (operation code or instruction code) of the instruction which is being decoded and executed.
6. **Temporary Register:** It is an 8-bit register associated with the ALU. It holds data during an arithmetic/logical operation. It is used by the microprocessor. It is not accessible to programmer.
7. **Flags:** The Intel 8085 microprocessor contains five flip-flops to serve as a status flags. The flip-flops are reset or set according to the conditions which arise during an arithmetic or logical operation. The five status flags of Intel 8085 are: Carry Flag (CS), Parity Flag (P), Auxiliary Carry Flag (AC), Zero Flag(Z) Sign Flag(S). If a flip-flop for a particular flag is set, then it indicates 1. When it is reset, it indicates 0.

Data and Address Bus

- The Intel 8085 is an 8-bit microprocessor. Its **data bus** is 8-bit wide and therefore, 8 bits of data can be transmitted in parallel from or to the microprocessor.
- The Intel 8085 requires an **address bus** of 16-bit wide as the memory addresses are of 16-bits.
- The 8 most significant bits of the address are transmitted by the address bus, A-bus (pins A_8 ? A_{15}).
- The 8 least significant bits of the address are transmitted by data/address bus, AD-bus (pins AD_0 ? AD_7).

Interrupt control : Whenever a microprocessor is executing a main program and if suddenly an interrupt occurs, the microprocessor shifts the control from the main program to process the incoming request. After the request is completed, the control goes back to the main program. There are 5 interrupt signals in 8085 microprocessors: INTR, TRAP, RST 7.5, RST 6.5, RST 5.5

Serial Input/output control : It controls the serial data communication by using Serial input data and Serial output data.

8085 microprocessor	8086 microprocessor
The data bus is of 8 bits.	The data bus is of 16 bits.
The address bus is of 16 bits.	The address bus is of 20 bits.
The memory capacity is 64 KB. Also 8085 Can Perform Operation upto 2^8 i.e. 256 numbers. A number greater than this is to taken multiple times in 8 bit data bus.	The memory capacity is 1 MB. Also 8086 Can Perform Operation upto 2^{16} i.e. 65,536 numbers.
The input/output port addresses are of 8 bits.	The input/output port addresses are of 8 bits.
The operating frequency is 3.2 MHz.	The operating frequency is 5 MHz, 8MHZ,10MHZ.
8085 MP has Single Mode of Operation.	8086 MP has Two Modes Of Operation. 1. Minimum Mode = Single CPU PROCESSOR 2. Maximum Mode = Multiple CPU PROCESSOR.
It not have multiplication and division instructions.	It have multiplication and division instructions.
It does not support pipe-lining.	It supports pipe-lining as it has two independent units Execution Unit (EU) and Bus Interface Unit (BIU).
It does not support instruction queue.	It supports instruction queue.
Memory space is not segmented.	Memory space is segmented.
It consists of 5 flags(Sign Flag, Zero Flag, Auxiliary Carry Flag, Parity Flag, Carry Flag).	It consists of 9 flags(Overflow Flag, Direction Flag, Interrupt Flag, Trap Flag, Sign Flag, Zero Flag, Auxiliary Carry Flag, Parity Flag, Carry Flag)

8086 MICROPROCESSORS

Register Organization of 8086

- 8086 has a powerful set of registers containing general purpose and special purpose registers.
- All the registers of 8086 are 16-bit registers.
- The general purpose registers, can be used either 8-bit registers or 16-bit registers. The general purpose registers are either used for holding the data, variables and intermediate results temporarily or for other purpose like counter or for storing offset address for some particular addressing modes etc.
- The special purpose registers are used as segment registers, pointers, index registers or as offset storage registers for particular addressing modes.
- Fig shows register organization of 8086. We will categorize the register set into four groups as follows:

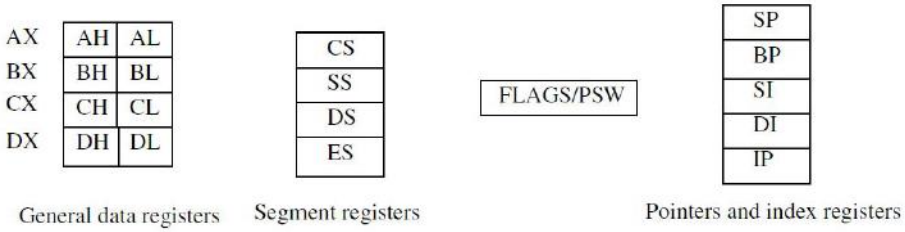


Fig.1.4 Register organization of 8086 Microprocessor

General Data Registers: The registers AX, BX, CX, and DX are the general 16-bit registers.

- **AX Register:** Accumulator register consists of two 8-bit registers AL and AH, which can be combined together and used as a 16- bit register AX. AL in this case contains the low-order byte of the word, and AH contains the high order byte. Accumulator can be used for I/O operations, rotate and string manipulation.

- **BX Register:** This register is mainly used as a base register. It holds the starting base location of a memory region within a data segment. It is used as offset storage for forming physical address in case of certain addressing mode.

- **CX Register:** It is used as default counter or count register in case of string and loop instructions.

- **DX Register:** Data register can be used as a port number in I/O operations and implicit operand or destination in case of few instructions. In integer 32-bit multiply and divide instruction the DX register contains high-order word of the initial or resulting number.

Segment Registers: To complete 1Mbyte memory is divided into 16 logical segments. The complete 1Mbyte memory segmentation is as shown in fig 1.5. Each segment contains 64Kbyte of memory. There are four segment registers.

- **Code segment (CS)** is a 16-bit register containing address of 64 KB segment with processor instructions. The processor uses CS segment for all accesses to instructions referenced by instruction pointer (IP) register. CS register cannot be changed directly. The CS register is automatically updated during far jump, far call and far return instructions. It is used for addressing a memory location in the code segment of the memory, where the executable program is stored.

- **Stack segment (SS)** is a 16-bit register containing address of 64KB segment with program stack. By default, the processor assumes that all data referenced by the stack pointer (SP) and base pointer (BP) registers is located in the stack segment. SS register can be changed directly using POP instruction. It is used for addressing stack segment of memory. The stack segment is that segment of memory, which is used to store stack data.

- **Data segment (DS)** is a 16-bit register containing address of 64KB segment with program data. By default, the processor assumes that all data referenced by general registers (AX, BX, CX, DX) and index register (SI, DI) is located in the data segment. DS register can be changed directly using POP and LDS instructions. It points to the data segment memory where the data is resided.

- **Extra segment (ES)** is a 16-bit register containing address of 64KB segment, usually with program data. By default, the processor assumes that the DI register references the ES segment in string manipulation instructions. ES register can be changed directly using POP and LES instructions. It also refers to segment which essentially is another data segment of the memory. It also contains data.

Pointers and Index Registers: The pointers contain within the particular segments. The pointers IP, BP, SP usually contain offsets within the code, data and stack segments respectively

- **Stack Pointer (SP)** is a 16-bit register pointing to program stack in stack segment.

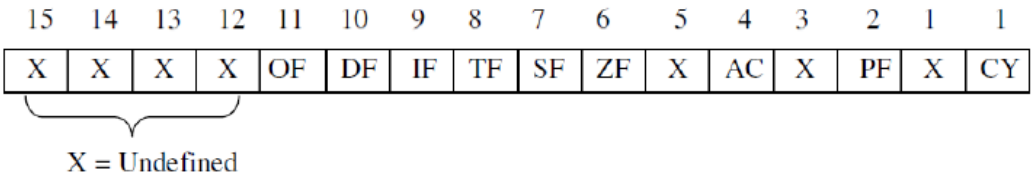
- **Base Pointer (BP)** is a 16-bit register pointing to data in stack segment. BP register is usually used for based, based indexed or register indirect addressing.

- **Source Index (SI)** is a 16-bit register. SI is used for indexed, based indexed and register indirect addressing, as well as a source data addresses in string manipulation instructions.

- **Destination Index (DI)** is a 16-bit register. DI is used for indexed, based indexed and register indirect addressing, as well as a destination data address in string manipulation instructions

Flag Registers: Flags Register determines the current state of the processor.
They are modified automatically by CPU after mathematical operations, this allows to determine the type of the result, and to determine conditions to transfer control to other parts of the program. The 8086 flag

register as shown in the fig, 9 active flags and they are divided into two categories: 1. Conditional Flags 2. Control Flags



Conditional flags are as follows:

- **Carry Flag (CY):** This flag indicates an overflow condition for unsigned integer arithmetic. It is also used in multiple-precision arithmetic.

- **Auxiliary Flag (AC):** If an operation performed in ALU generates a carry/barrow from lower nibble (i.e. D0 – D3) to upper nibble (i.e. D4 – D7), the AC flag is set i.e. carry given by D3 bit to D4 is AC flag. This is not a general-purpose flag; it is used internally by the Processor to perform Binary to BCD conversion.

- **Parity Flag (PF):** This flag is used to indicate the parity of result. If lower order 8-bits of the result contains even number of 1's, the Parity Flag is set and for odd number of 1's, the Parity flag is reset.

- **Zero Flag (ZF):** It is set; if the result of arithmetic or logical operation is zero else it is reset.

- **Sign Flag (SF):** In sign magnitude format the sign of number is indicated by MSB bit. If the result of operation is negative, sign flag is set.

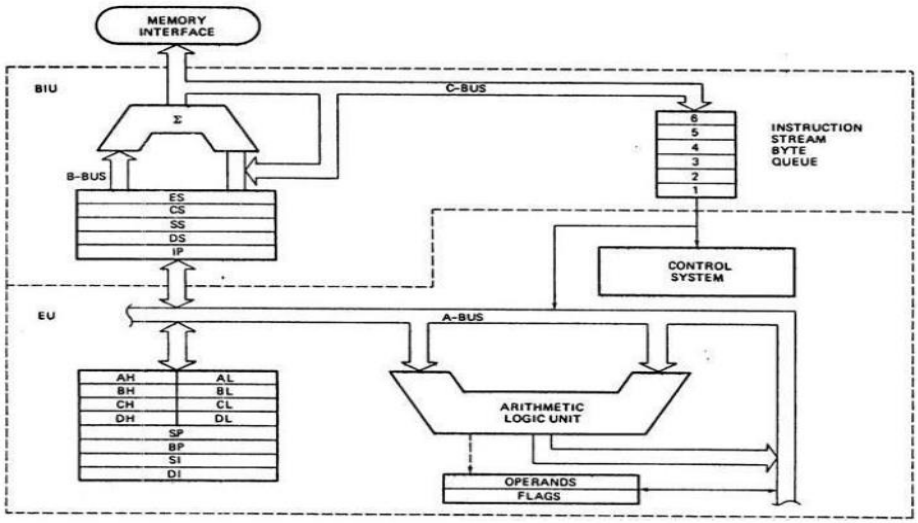
Control Flags: Control flags are set or reset deliberately to control the operations of the execution unit. Control flags are as follows:

- **Trap Flag (TF):** It is used for single step control. It allows user to execute one instruction of a program at a time for debugging. When trap flag is set, program can be run in single step mode.

- **Interrupt Flag (IF):** It is an interrupt enable/disable flag. If it is set, the maskable interrupt of 8086 is enabled and if it is reset, the interrupt is disabled. It can be set by executing instruction sit and can be cleared by executing CLI instruction.

- **Direction Flag (DF):** It is used in string operation. If it is set, string bytes are accessed from higher memory address to lower memory address. When it is reset, the string bytes are accessed from lower memory address to higher memory address.

8086 Architecture



- The 8086 is mainly divided into mainly two blocks
    1. Execution Unit (EU)
    2. Bus interface Unit (BIU)

- Dividing the work between these two will speed up the processing

Bus Interface Unit:

It provides the interface of 8086 to external memory and I/O devices via the System Bus. It performs various machine cycles such as memory read, I/O read etc. to transfer data between memory and I/O devices.

BIU performs the following functions-

- It generates the 20 bit physical address for memory access.
- It fetches instructions from the memory.
- It transfers data to and from the memory and I/O.
- Maintains the 6 byte prefetch instruction queue (**supports pipelining**).

BIU mainly contains the **4 Segment registers**, the **Instruction Pointer**, an **Instruction Queue** and an **Address Generation Circuit**.

Instruction Pointer (IP):

- It is a 16 bit register. It holds offset of the next instructions in the Code Segment.
- IP is incremented after every instruction byte is fetched.
- IP gets a new value whenever a branch instruction occurs.
- CS is multiplied by 10H to give the 20 bit physical address of the Code Segment.
- Address of the next instruction is calculated as CS x 10H + IP.

Address Generation Circuit:

- The BIU has a Physical Address Generation Circuit.
- It generates the 20 bit physical address using Segment and Offset addresses using the formula:
  Physical Address = Segment Address x 10H + Offset Address

Instruction Queue:

- It is a 6 byte queue (FIFO).
- Fetching the next instruction (by BIU from CS) while executing the current instruction is called pipelining.
- Gets flushed whenever a branch instruction occurs.

Segment Registers:

- **Code Segment register:** CS holds the base address for the Code Segment. All programs are stored in the Code Segment and accessed via the IP.
- **Data Segment registers:** DS holds the base address for the Data Segment.
- **Stack Segment register:** SS holds the base address for the Stack Segment.
- **Extra Segment register:** ES holds the base address for the Extra Segment.

The Execution Unit (EU): The main components of the EU are General purpose registers, the ALU, Special purpose registers, Instruction Register and Instruction Decoder and the Flag/Status Register.

1. Fetches instructions from the Queue in BIU, decodes and executes arithmetic and logic operations using the ALU.
2. Sends control signals for internal data transfer operations within the microprocessor.
3. Sends request signals to the BIU to access the external module.
4. It operates with respect to T-states (clock cycles) and not machine cycles.

- **Instruction Register and Instruction Decoder:** The EU fetches an opcode from the queue into the instruction register. The instruction decoder decodes it and sends the information to the control circuit for execution.

- **Arithmetic Logic Unit (16 bit):** Performs **8 and 16 bit** arithmetic and logic operations.

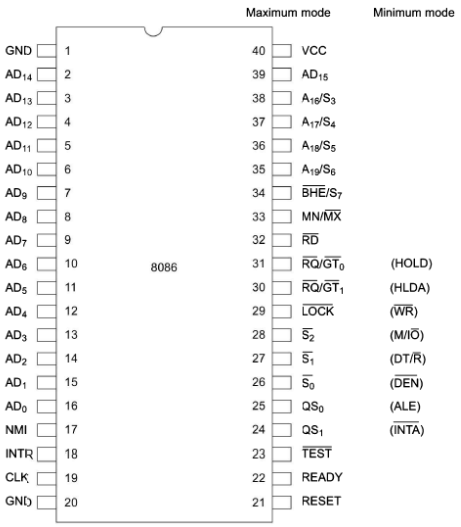
- **Flag Registers:** the 16 bit flag register reflects the result of ALU.

- **Timing and control unit:** it derives necessary control signals to execute instruction opcode received from queue depending upon information available in decoding unit.

Memory Segmentation:

- The memory in 8086 is organized as segments. In this scheme, the complete physical memory is divided into number of logical segments.
- Each segment is 64 K Bytes in size and is addressed by one of the segment registers.
- The CPU of 8086 is able to address 1 MB of memory.
- The complete 1 MB is divided into 16 segments each of 64 K bytes.
- The address of segments may be addressed as 0000H to F000H
- The offset address values are from 0000H to FFFFH so that physical address ranges from 00000H to FFFFFH. In the above case , segments is called non – overlapping segments.
- Suppose a segment start at particular address and its maximum size is 64 K bytes. But, if another segment starts before this 64 K bytes location of first segment, then the segment is said to be overlapping segment.
- The advantages include:
    - Allow the memory capacity to be 1 Mb although actual address to be handled are of 16 bits.
    - Allow the placing of code, data and stack portion of same program in different segments of memory for data and code protection.
    - Permits a program and/ or its data to be put into different areas of memory each time it is executed i.e. provision for relocation is done.

PIN OUT DIAGRAM OF 8086



- The 8086 Microprocessor is a 16-bit CPU available in 3 clock rates, i.e. 5, 8 and 10MHz, packaged in a 40 pin CERDIP or plastic package. The 8086 Microprocessor operates in single processor or multiprocessor configurations to achieve high performance.
- Some of the pins serve a particular function in minimum mode (single processor mode) and others function in maximum mode (multiprocessor mode) configuration.

- The 8086 signals can be categorized in three groups. The first are the signals having common functions in minimum as well as maximum mode, the second are the signals which have special functions in minimum mode and third are the signals having special functions for maximum mode.

Pins	Description
AD15-AD0:	These are the time multiplexed memory I/O address and data lines. Address remains on the lines during T1 state, while the data is available on the data bus during T2, T3, TW and T4. Here T1, T2, T3, T4 and TW are the clock states of a machine cycle. TW is await state.
A19/S6,A18/S5, A17/S4,A16/S3	These are the time multiplexed address and status lines. During T1, these are the most significant address lines or memory operations. During I/O operations, these lines are low. During memory or I/O operations, status information is available on those lines for T2, T3, TW and T4. The status of the interrupt enable flag bit(displayed on S5) is updated at the beginning of each clock cycle.
BHE/S7 (Bus High Enable/Status)	The bus high enable signal is used to indicate the transfer of data over the higher order (D15-D8). t goes low for the data transfers over D15-D8 and is used to derive chip selects of odd address memory bank or peripherals.
RD	Read signal, when low, indicates the peripherals that the processor is performing a memory or I/O read operation.
READY	This is the acknowledgement from the slow devices or memory that they have completed the data transfer.
INTR	This is a level triggered input. This is sampled during the last clock cycle of each instruction to determine the availability of the request. If any interrupt request is pending, the processor enters the interrupt acknowledge cycle.
TEST	This input is examined by a 'WAIT' instruction. If the TEST input goes low, execution will continue, else, the processor remains in an idle state.
NMI	This is an edge-triggered input which causes a Type2 interrupt. The NMI is not maskable internally by software. A transition from low to high initiates the interrupt response at the end of the current instruction.
RESET	This input causes the processor to terminate the current activity and start execution from FFFF0H
CLK	The clock input provides the basic timing for processor operation and bus control activity.
VCC	+5V power supply for the operation of the internal circuit. GND ground for the internal circuit.
MN/MX	The logic level at this pin decides whether the processor is to operate in either minimum (single processor) or maximum (multiprocessor) mode.
M/IO	This is a status line logically equivalent to S2 in maximum mode. When it is low, it indicates the CPU is having an I/O operation, and when it is high, it indicates that the CPU is having a memory operation.
INTA	This signal is used as a read strobe for interrupt acknowledge cycles. In other words, when it goes low, it means that the processor has accepted the interrupt.
ALE	This output signal indicates the availability of the valid address on the address/data lines and is connected to latch enable input of latches.
DT/$\bar{R}$: (Data transmit/Receive)	This output is used to decide the direction of data flow through the transreceivers (bidirectional buffers).
$\overline{\text{DEN}}$ (Data Enable)	This signal indicates the availability of valid data over the address/data lines. It is used to enable the transreceivers (bidirectional buffers) to separate the data from the multiplexed address/data signal
HOLD, HLDA-Hold/Hold Acknowledge	When the HOLD line goes high, it indicates to the processor that another master is requesting the bus access. The processor, after receiving the HOLD request, issues the hold acknowledge signal on HLDA pin, in the middle of the next clock cycle after completing the current bus (instruction) cycle. At the same time, the processor floats the local bus and control lines. When the processor detects the HOLD line low, it lowers the HLDA signal.

S2, S1, S0 - Status Lines	These are the status lines which reflect the type of operation, being carried out by the processor.
$\overline{LOCK}$:	This output pin indicates that other system bus masters will be prevented from gaining the system bus, while the signal is low.
QS1, QS0-Queue Status	These lines give information about the status of the code prefetch queue. These are active during the CLK cycle after which the queue operation is performed
$\overline{RQ}/\overline{GT_0}$ (Request/Grant)	These pins are used by other local bus masters, in maximum mode, to force the processor to release the local bus at the end of the processor's current bus cycle.

$\overline{S}_2$	$\overline{S}_1$	$\overline{S}_0$	*Indication*
0	0	0	Interrupt acknowledge
0	0	1	Read I/O port
0	1	0	Write I/O port
0	1	1	Halt
1	0	0	Code access
1	0	1	Read memory
1	1	0	Write memory
1	1	1	Passive

QS_1	QS_0	*Indication*
0	0	No operation
0	1	First byte of opcode from the queue
1	0	Empty queue
1	1	Subsequent byte from the queue

PHYSICAL MEMORY ORGANIZATION OF 8086

In an 8086 based system, the 1Mbytes memory is physically organised as an odd bank and an even bank, each of 512 Kbytes, addressed in parallel by the processor. Byte data with an even address is transferred on D_7-D_0, while the byte data with an odd address is transferred on $D_{15}-D_8$ buss lines. The processor provides two enable signals, $\overline{BHE}$ and A_0 for selection of either even or odd or both the banks. The instruction stream is fetched from memory as words and is addressed internally by the processor as necessary. In other words, if the processor fetches a word (consecutive two bytes) from memory, there are different possibilities, like:

1. Both the bytes may be data operands
2. Both the bytes may contain opcode bits
3. One of the bytes may be opcode while the other may be data

- All of the above operations are handled by the internal decoder circuit.
- 8086 is a 16 bit microprocessor so it can access two bytes of data in one memory or I/O read or write operation. But commercially available chips are only one byte size.
- A map of memory 8086 system starts from 00000H and ends at FFFFFH.

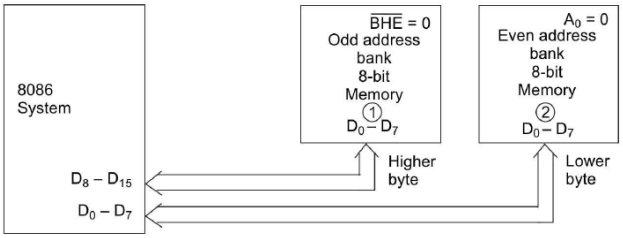


Fig. 1.7 *Physical Memory Organisation*

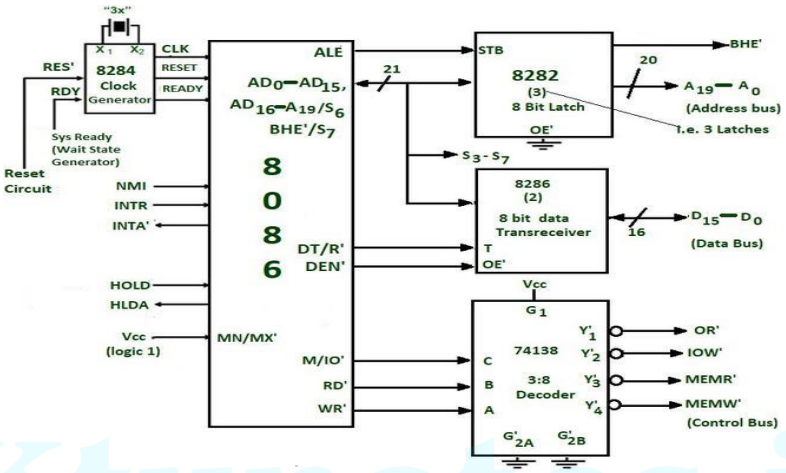
- Thus bits D0 – D7 of a 16 bit data will be transferred over D0 – D7 bus of 16 bit microprocessor to/ from 8 bit memory (1) and D8 – D15 will be transferred over D8 – D15 of 16 bit microprocessor to/ from memory (2).
- The lower byte is stored at 00000H and is transferred over D0 – D7, so 00000H must be in memory 2
- Higher byte is stored in 00001H and is transferred over D8 – D15, so 00001H must be in memory 1.
- All lower bytes are stored in even memory bank (memory bank 2) and all higher bytes are in odd memory bank (memory bank 1).
- Thus, complete memory map of 8086 is divided into odd and even memory banks.
- If 8086 transfers a 16-bit data to/ from memory both these banks must be selected for operation.
- Two signals A0 and $\overline{BHE}$ solve the problem of selection of memory banks.
- Certain locations are reserved for specific CPU purposes.

- The locations from FFFF0H to FFFFFH are reserved for operations including jump to initialization program and I/O processor initialization.
- The locations 00000H to 003FFH is reserved for interrupt vector table.

MINIMUM MODE OF OPERATION IN 8086

- The 8086 microprocessor operates in minimum mode when MN/MX' = 1.
- In minimum mode,8086 is the only processor in the system which provides all the control signals which are needed for memory operations and I/O interfacing.
- Here the circuit is simple but it does not support multiprocessing.

- The address bus of 8086 is 20 bits long. By this we can access 220 bytes memory i.e. 1MB. Out of 20 bits, 16 bits A0 to A15(or 16 lines) are multiplexed with a data bus.



8282 (8 bits) latches: The latches are buffered D FF. They are used to separate the valid address from the multiplexed Address/data bus by using the control signal ALE, which is connected to strobe (STB) of 8282. The ALE is active high signal. Here three such latches are required because the address is 20 bits.

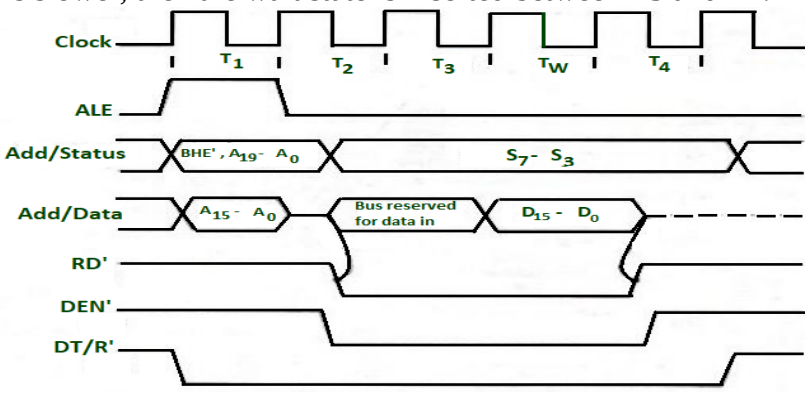
8286 (8 bits) transceivers: They are bidirectional buffers and also known as data amplifiers. They are used to separate the valid data from multiplexed add/data bus. Two such transceivers are needed because the data bus is 16 bits long. 8286 is connected to DT/R' and DEN' signals. They are enabled through the DEN signal. The direction of data on the data bus is controlled by the DT/R' signal. DT/R' is connected to T and DEN' is connected to OE'.

- 8284 clock generator is used to provide the clock.
- M/IO'= 1, then I/O transfer is performed over the bus. and when M/IO' = 0, then I/O operation is performed.
- The signals RD' and write WR' are used to identify whether a read bus cycle or a write bus cycle is performing. When WR' = 0, then it indicates that valid output data on the data bus.
- RD' indicates that the 8086 is performing a read data or instruction fetch process is occurring .During read operations, one other control signal is also used, which is DEN (data enable) and it indicates the external devices when they should put data on the bus.
- Control signals for all operations are generated by decoding M/IO', RD', WR'. They are decoded by 74138 3:8 decoder.

Timing diagram: The working of min mode can be easily understood by timing diagrams.

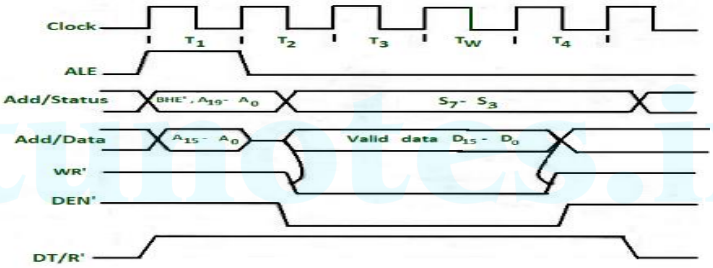
- All processors bus cycle is of at least 4 T-states (T1, T2, T3, T4). The address is given by processor in the T1 state. It is available on the bus for one T-state.
- In T2, the bus is tristated for changing the direction of the bus (in the case of a data read cycle.)
- The data transfer takes place between T3 and T4.

- If the addressed device is slower, then the wait state is inserted between T3 and T4.



Read Memory cycle

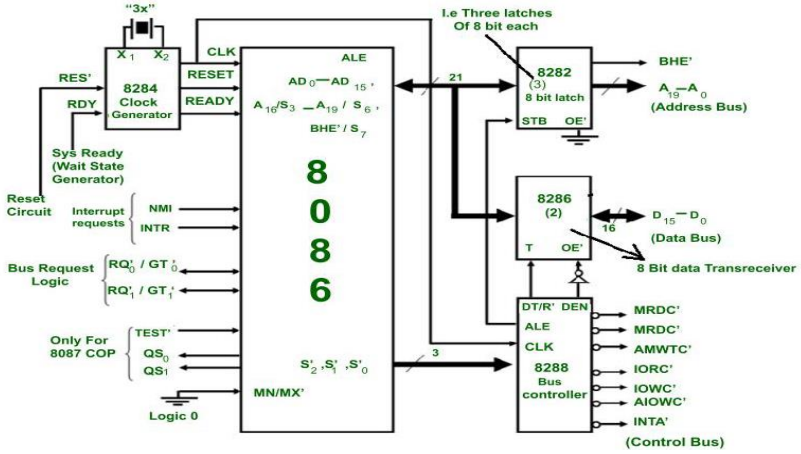
- At T1 state ALE =1 ,this indicates that a valid address is latched on the address bus and also M / IO'= 1, which indicates the memory operation is in progress.
- In T2, the address is removed from the local bus and is sent to the addressed device. Then the bus is tristated.
- When RD' = 0, the valid data is present on the data bus.
- During T2 DEN' =0, which enables transceivers and DT/R' = 0, which indicates that the data is received.
- During T3, data is put on the data bus and the processor reads it.
- The output device makes the READY line high. This means the output device has performed the data transfer process. When the processor makes the read signal to 1, then the output device will again tristate its bus drivers.



Memory Write cycle

MAXIMUM MODE OPERATION OF 8086

- In this we can connect more processors to 8086 (8087/8089)
- 8086 max mode is basically for implementation of allocation of global resources and passing bus control to other co – processor (i.e. second processor in the system), because two processors cannot access system bus at same instant.
- All processors execute their own program.
- The resources which are common to all processors are known as global resources.
- The resources which are allocated to a particular processor are known as local or private resources.



- When MN/ MX' = 0, 8086 works in max mode.

- Clock is provided by 8284 clock generator.
- 8288 bus controller- Address form the address bus is latched into 8282 8-bit latch. Three such latches are required because address bus is 20 bits. The ALE (Address latch enable) is connected to STB(Strobe) of the latch. The ALE for latch is given by 8288 bus controller.
- The data bus is operated through 8286 8-bit transceiver. Two such transceivers are required, because data bus is 16-bit. The transceivers are enabled the DEN signal, while the direction of data is controlled by the DT/R signal. DEN is connected to OE' and DT/ R' is connected to T. Both DEN and DT/ R' are given by 8288 bus controller.
- Control signals for all operations are generated by decoding S'2, S'1 and S'0 using 8288 bus controller.
- Bus request is done using RQ' / GT' lines interfaced with 8086. RQ0/GT0 has more priority than RQ1/GT1.
- INTA' is given by 8288, in response to an interrupt on INTR line of 8086.
- In max mode, the advanced write signals get enabled one T-state in advance as compared to normal write signals. This gives slower devices more time to get ready to accept the data, therefore it reduces the number of cycles.

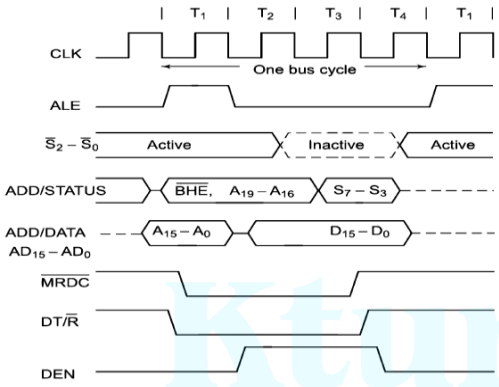


Fig. 1.16 (a) *Memory Read Timing in Maximum Mode*

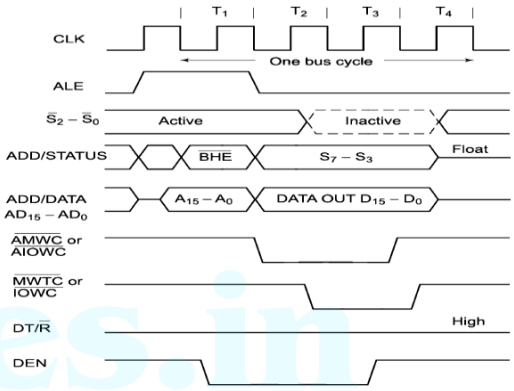


Fig. 1.16(b) *Memory Write Timing in Maximum Mode*

DIFFERENCE BETWEEN 8086 AND 8088 MICROPROCESSORS

Sl no	8086	8088
1	The data bus is of 16 bits.	The data bus is of 8 bits.
2	It has 3 available clock speeds (5 MHz, 8 MHz & 10 MHz)	It has 2 available clock speeds (5 MHz, 8 MHz)
3	The memory capacity is 512 kB.	The memory capacity is implemented as a single 1 MX 8 memory banks.
4	It has memory control pin (M/IO) signal.	It has complemented memory control pin (IO/M) signal of 8086.
5	It has Bank High Enable (BHE) signal.	It has Status Signal (SSO).
6	It can read or write either 8-bit or 16-bit word at the same time.	It can read only 8-bit word at the same time.
7	Input/Output voltage level is measured at 2.5 mA.	Input/Output voltage level is measured at 2.0 mA
8	It has 6 byte instruction queue.	It has 4 byte instruction queue as it can fetch only 1 byte at a time.
9	It draws a maximum supply current of 360 mA.	It draws a maximum supply current of 340 mA.

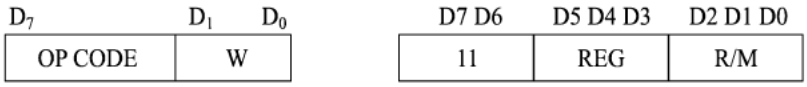
MACHINE LANGUAGE INSTRUCTION FORMAT

- A Machine language instruction format has one or more fields associated with it.

- The first field is called operation code field or opcode field, which indicates the type of operation to be performed by the CPU.
- It also contains other fields known as operand fields.
- The CPU executes the instruction using the information available in instruction fields.
- There are 6 general format of instruction fields in 8086. The length vary from 1 byte to 6 bytes.

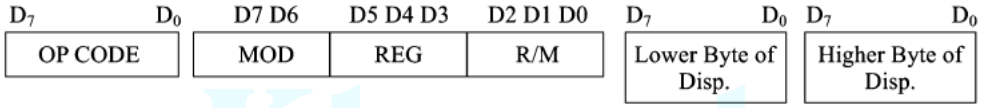
1. One byte Instruction This format is only one byte long and may have the implied data or register operands. The least significant 3-bits of the opcode are used for specifying the register operand, if any. Otherwise, all the 8-bits form an opcode and the operands are implied.

2. Register to Register This format is 2 bytes long. The first byte of the code specifies the operation code and width of the operand specified by w bit. The second byte of the code shows the register operands and R/M field, as shown below.



3. Register to/from Memory with no Displacement This format is also 2 bytes long and similar to the register to register format except for the MOD field as shown.

4. Register to/from Memory with Displacement This type of instruction format contains one or two additional bytes for displacement along with 2-byte the format of the register to/from memory without displacement. The format is as shown below.



5. Immediate Operand to Register In this format, the first byte as well as the 3-bits from the second byte which are used for REG field in case of register to register format are used for opcode. It also contains one or two bytes of immediate data. The complete instruction format is as shown below.

6. Immediate Operand to Memory with 16-bit Displacement This type of instruction format requires 5 or 6 bytes for coding. The first 2 bytes contain the information regarding OPCODE, MOD, and R/M fields. The remaining 4 bytes contain 2 bytes of displacement and 2 bytes of data as shown.