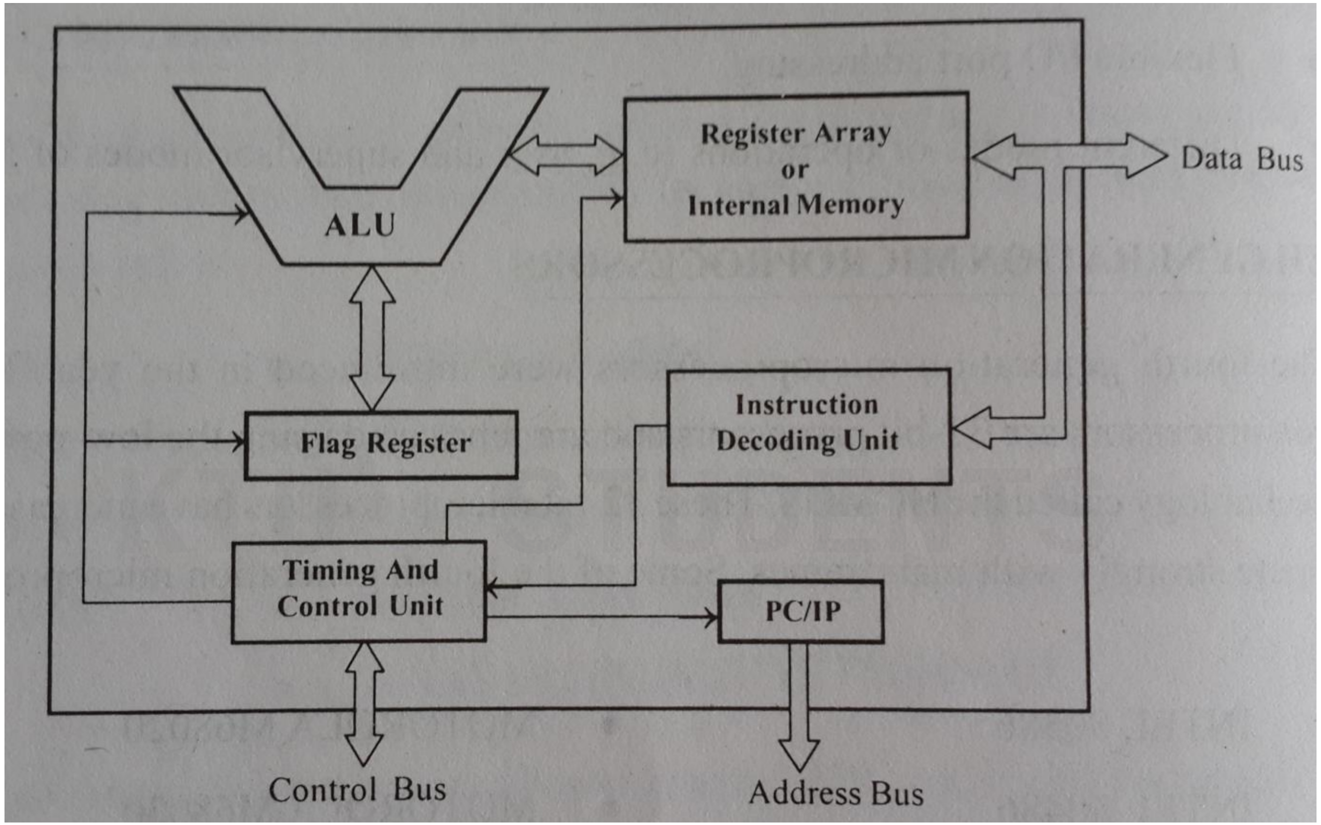
Microprocessors & Microcontrollers

Module 1



Functional blocks of a microprocessor

The ALU and Flag Register

- The Computational unit of μprocessor
- Arithmetic and logical operations on binary data

- Conditions of results are stored as *status bits* called flags in *flag register*

Example is Sign Flag- Status of sign of result of ALU

If result is negative, then 1 is stored in sign flag
If result is positive, then 0 is stored in sign flag

The Register Array

- It is the internal storage device or internal memory
- It stores the input and output of ALU and any other binary information needed for processing

Instruction decoding unit

- There will be a set of instructions provided by the manufacturer for each µprocessor. We have to write a program using these instructions and store them in external memory

- The purpose of decoding unit is to translate the instruction code received to processable data formats

Instruction Pointer/ Program Counter

- It generates the address of the instructions to be fetched from the memory .
- The generated address is sent to the memory through Address bus
- The memory will send instruction codes and data through the Data bus

Instruction codes are decoded by the decoding unit and it sends information to *Timing and Control* unit.

Control unit generates control signals for internal and external operations

Data is stored in the register array for processing in ALU

Microprocessor Based System

It is a system designed using microprocessor as CPU and peripherals.

The Peripherals are

- Semiconductor memories like EPROM and RAM,
- Input-output devices and
- Interfacing units.

Usually Microprocessor is the *Master* (Controls peripherals and initiates all operations) & Peripherals are *Slaves*

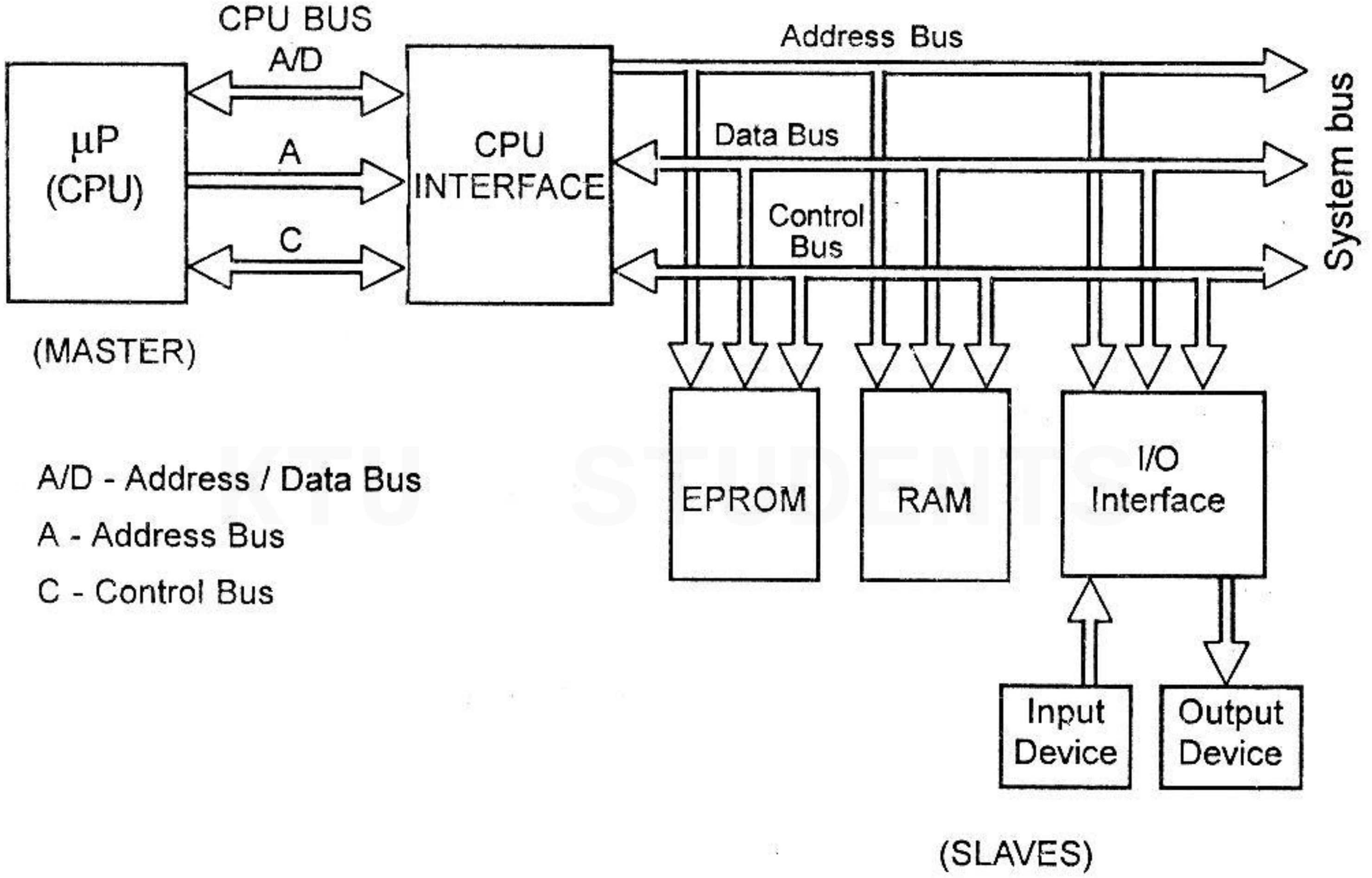


Fig : 1.1 Microprocessor Based System (organisation of microcomputer)

Microprocessor Based System

- All slaves are connected to same system bus
- All slaves have tri-state logic and hence they are normally in **high impedence state**
- When processor selects a slave by sending address, it comes to normal logic and communicates with processor

- *EPROM-* Store permenant programs and data
- RAM memory- Store temporary programs and data
- *Input device-* To enter program/data & to operate the system
- *Output device-* For examining the results
- *I/O interface device-* To match speed of microprocessor and slow I/O devices

The work done by the processor can be classified three groups.

1. Work done internal to the processor
2. Work done external to the processor
3. Operations initiated by the slaves or peripherals.

The work done internal to the processors are *addition, subtraction, logical operations, data transfer operations, etc.*

The work done external to the processor *are reading/writing the memory and reading/writing the I/O devices or the peripherals.*

If the peripheral requires the attention of the master then it can interrupt the master and initiate an operation.

Microprocessor- The Master- Controls all activities of the system

It executes the program stored in memory

Program has a set of instructions stored in consecutive memory location

Microprocessor issues address and control s/g to fetch instruction and data one by one from memory.

It then decodes the instruction and executes the task specified

INTEL 8086 PINS, SIGNALS & ARCHITECTURE

- First 16-bit processor by INTEL in 1978
- HMOS Technology (High Density MOS)
- Approximately 29k transistors
- 40 Pin DIP- dual in-line package
- Power Single 5V supply
- External clock provided by 8284 with 33% duty cycle
- Internal clock range is 5MHz to 10MHz

8086 memory organization

- 8086 uses 20 bit address to access memory

Hence it can access 2^{20}=1 Megabyte i.e. 1 Mb memory space

The 1 Mb space of 8086 is organized as two memory banks of 512 kb each

512 kb + 512 kb = 1 Mb

The memory banks are called even (lower) and odd (upper) bank.

The address line A0 is used to select even bank

$\overline{\text{BHE}}$ is used to select the odd bank

- For accessing I/O mapped devices, 8086 uses a separate 16 bit address And so it can generate 2^{16} =64k addresses

- The signal $M/\overline{IO}$ is used to differentiate the memory and I/O addresses

For Memory address $M/\overline{IO}$ is asserted high and

For I/O address $M/\overline{IO}$ is asserted low by the processer

8086 Operating Modes

- 8086 can operate in two modes, they are Minimum Mode and Maximum mode

- It is selected by the signal $MN/\overline{MX}$

When $MN/\overline{MX}$ is high – Minimum mode called uniprocessor
When $MN/\overline{MX}$ is low – Maximum mode called multiprocessor

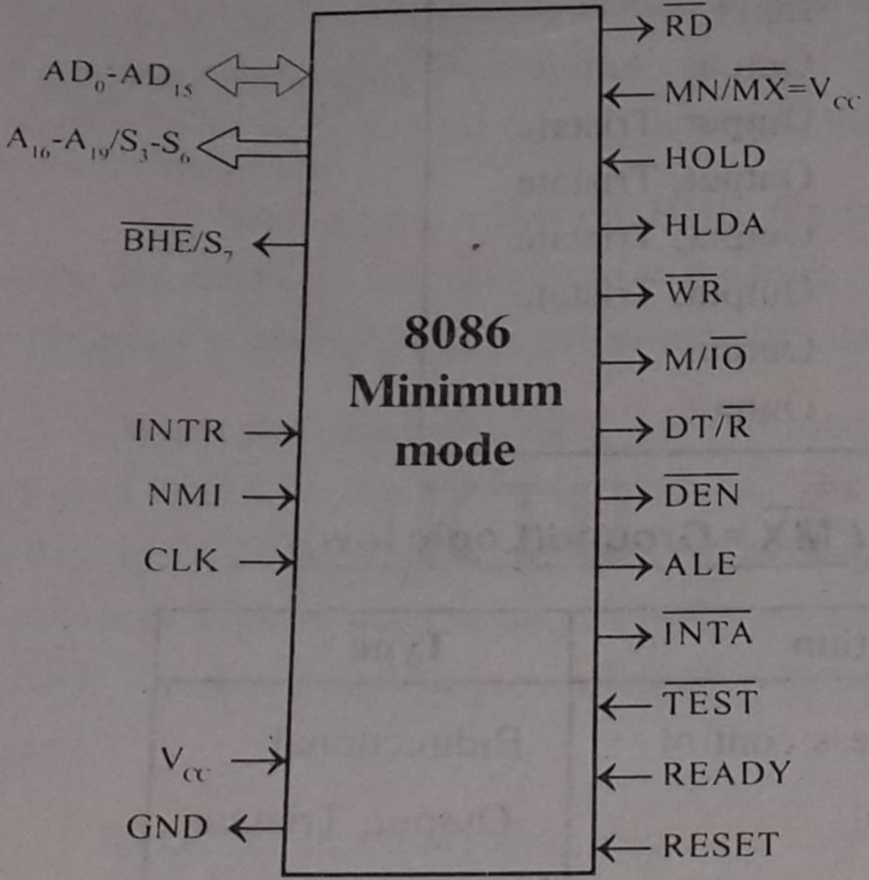


Fig b : 8086-Minimum mode.

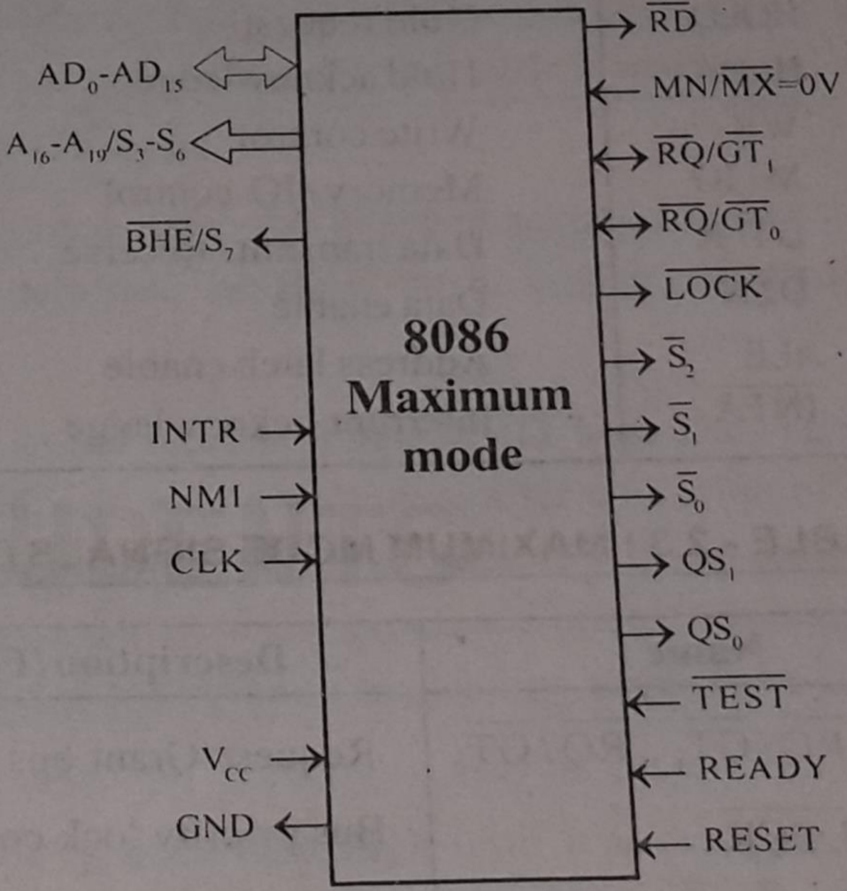


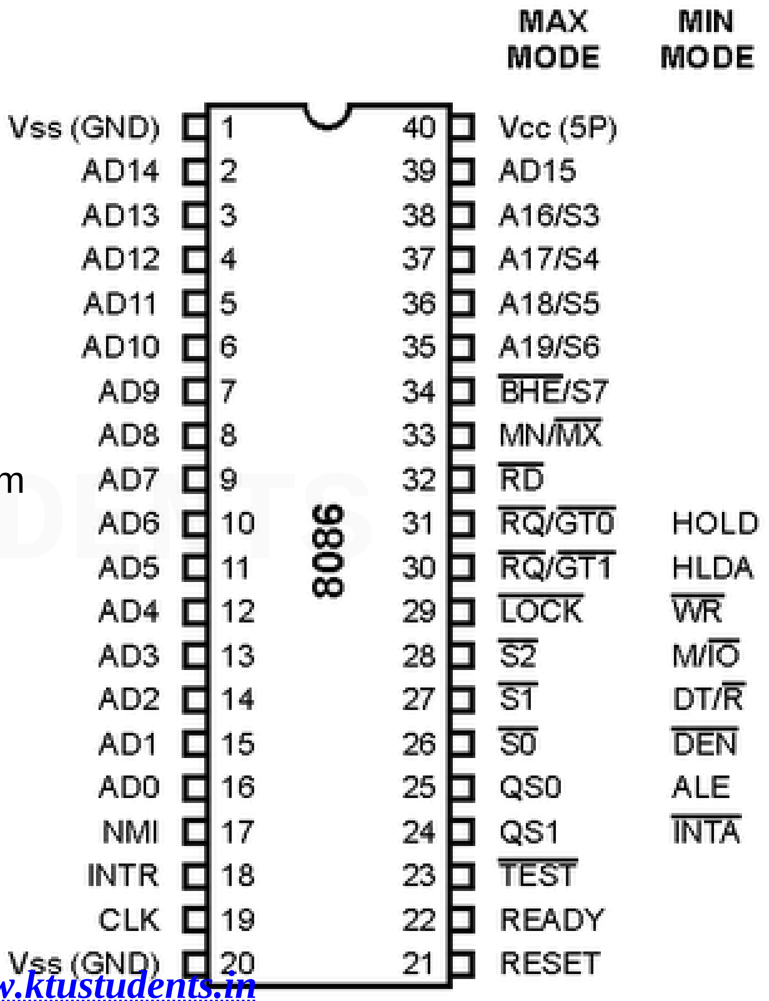
Fig c : 8086-Maximum mode.

Fig 2.1 : 8086 pins and signals.

PINS and Signals of 8086

The signal assigned to pins 24 to 31 will be different for minimum and maximum mode operation

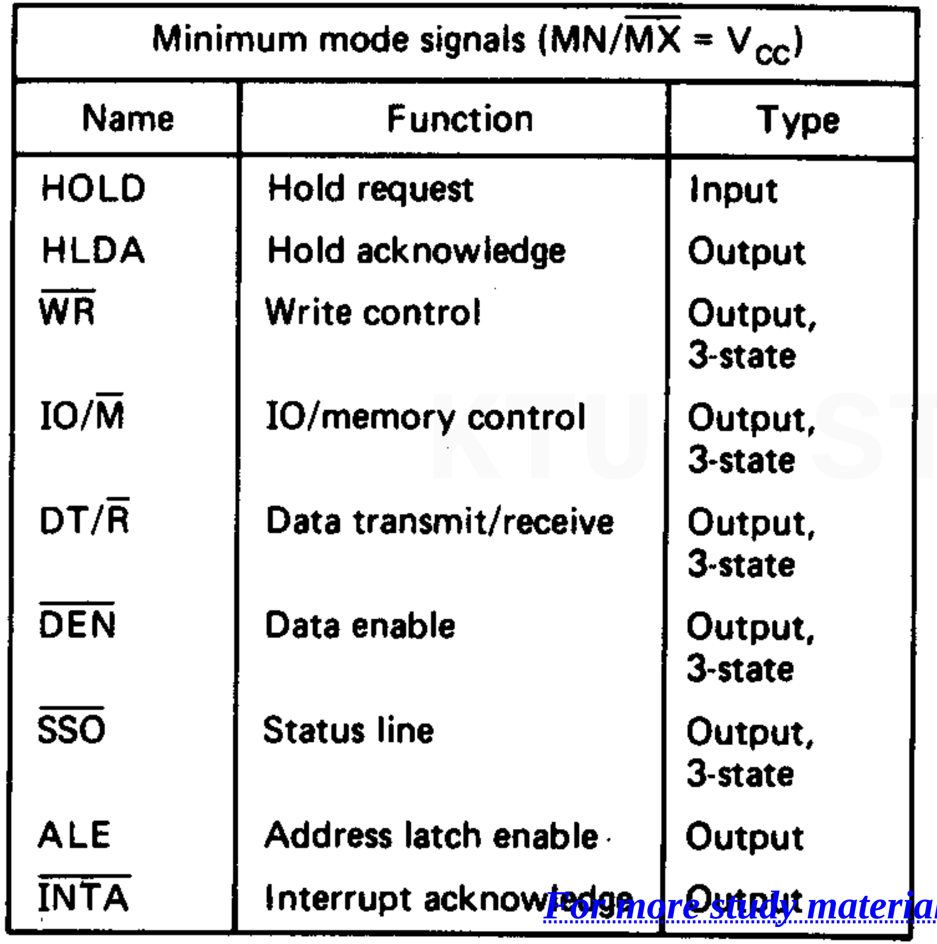
The signal assigned to all other pins are common for minimum and maximum mode operation



Common signals

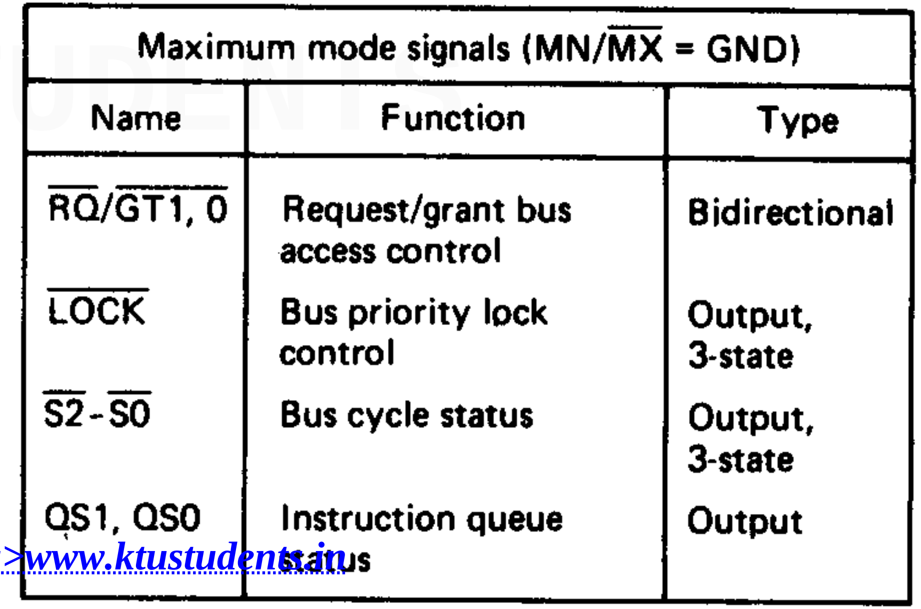
Name	Function	Type
AD7–AD0	Address/data bus	Bidirectional, 3-state
A15-A8	Address bus	Output, 3-state
A19/S6-A16/S3	Address/status	Output, 3-state
MN/$\overline{MX}$	Minimum/maximum Mode control	Input
$\overline{RD}$	Read control	Output, 3-state
$\overline{TEST}$	Wait on test control	Input
READY	Wait state control	Input
RESET	System reset	Input
NMI	Nonmaskable Interrupt request	Input
INTR	Interrupt request	Input
CLK	System clock	Input
V_{CC}	+5 V	Input
GND	Ground	

Minimum and maximum mode Signals of 8086



Minimum mode signals ($MN/\overline{MX} = V_{CC}$)

Name	Function	Type
HOLD	Hold request	Input
HLDA	Hold acknowledge	Output
$\overline{WR}$	Write control	Output, 3-state
$IO/\overline{M}$	IO/memory control	Output, 3-state
$DT/\overline{R}$	Data transmit/receive	Output, 3-state
$\overline{DEN}$	Data enable	Output, 3-state
$\overline{SSO}$	Status line	Output, 3-state
ALE	Address latch enable	Output
$\overline{INTA}$	Interrupt acknowledge	Output



Maximum mode signals ($MN/\overline{MX}$ = GND)

Name	Function	Type
$\overline{RQ}/\overline{GT1, 0}$	Request/grant bus access control	Bidirectional
$\overline{LOCK}$	Bus priority lock control	Output, 3-state
$\overline{S2}-\overline{S0}$	Bus cycle status	Output, 3-state
QS1, QS0	Instruction queue status	Output

•**Pin 1, 20**	Ground
•**Pin 40**	Vcc
•**Pin 19**	CLK
•**Pin 17**	INTR
•**Pin 18**	NMI

AD_{15} –AD_0 [Pin 2- 16 &39]

The pins have dual function. They act as address bus during the first part of machine cycle and as data bus in the later part.

A_{19}/S_6 –A_{16}/S_3 [Pin 35- 38]

Contains address information in the first part and status bits in the later part. The status bits, when decoded, indicates the type of operations (eg. Memory access) being performed and the segment register being used.

Status Signal		Segment Register
S4	S3	
0	0	Extra Segment ES
0	1	Stack Segment SS
1	0	Code Segment CS
1	1	Data Segment DS

Common Signals

- **$\overline{\text{BHE}}$-** Bus High Enable- When data is to be transferred. It is an active low signal
- **$\overline{\text{RD}}$** - When processor reads from a memory or I/O location RD is asserted low

- **$\overline{\text{TEST}}$-** It is tested by WAIT instruction. 8086 enters wait state after execution of WAIT instruction and it will resume execution only when $\overline{\text{TEST}}$ is made low by external hardware. It is synchronized during each clock cycle on clock leading edge.

- **INTR-** It is maskable interrupt. INTR must be held high until it is recognized as interrupt signal
- **NMI-** Non Maskable Interrupt, activated by leading edge of clock signal

Common Signals

- **READY-** It is an input s/g to the processor, used by memory or I/O to get extra time for data transfer or to introduce wait states in bus cycles
- **RESET-** It is the system reset input signal. For Power ON reset, it is held high for 50ms. While working, to Reset, it is held high for atleast 4 clock cycles
- **CLK –** Input signal which provides basic timing for 8086 and bus controller. 8086 doesn't have on chip clock generation. It is done by 8284. 8284 also provides READY & RESET signals too

MIN mode Signals

- **DT/$\overline{\text{R}}$-** *Data Transmit/ Receive* – Output signal to control direction of data flow
- **$\overline{\text{DEN}}$-** *Data Enable* – Output signal, used as output enable for data transceivers
- **ALE** – *Address Latch Enable* – Demultiplex address and data signals using external latches
- **M/$\overline{\text{IO}}$** – To Differentiate memory or I/O access.

  For memory the signal is high

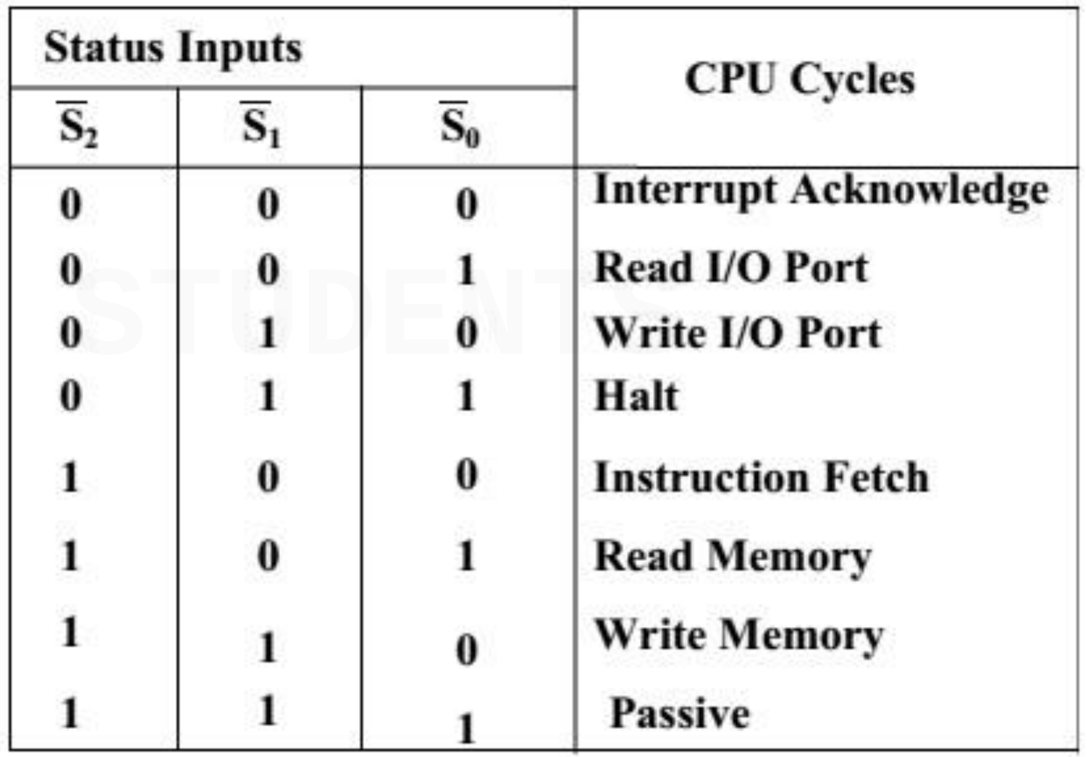
  For I/O signal is low

MIN mode Signals

- **$\overline{\text{WR}}$** – *Write Control signal-* Asserted low when processor writes data into M/IO
- **$\overline{\text{INTA}}$** – *Interrupt Acknowledge* – When interrupt request is accepted, it is made low
- **HOLD** – Input signal to processor from other bus masters as a request to grant the control of the bus. It is usually used by *DMA controller* to get full control of the bus
- **HLDA** – *Hold Acknowledge* – Acknowledgement signal from processor to the master requesting control of bus through HOLD
- When HOLD is received, the processor sets the tri-state pin to high impedence and sends acknowledgement signal to the requested device, on receipt of HLDA, the other master will take control of the bus

MAX mode Signals

- $\overline{S2}$ $\overline{S1}$ $\overline{S0}$ are status signals used by 8288 bus controller to generate bus timing and control signals



Status Inputs			CPU Cycles
$\overline{S}_2$	$\overline{S}_1$	$\overline{S}_0$	
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	Instruction Fetch
1	0	1	Read Memory
1	1	0	Write Memory
1	1	1	Passive

MAX mode Signals

- **$\overline{\text{RQ}}/\overline{\text{GT0}}$, $\overline{\text{RQ}}/$ $\overline{\text{GT1}}$** – *Bus Request/ Bus Grant* – The request made by other bus masters to force release the local bus at the end of the processor's current bus cycle. The pins are bi-directional

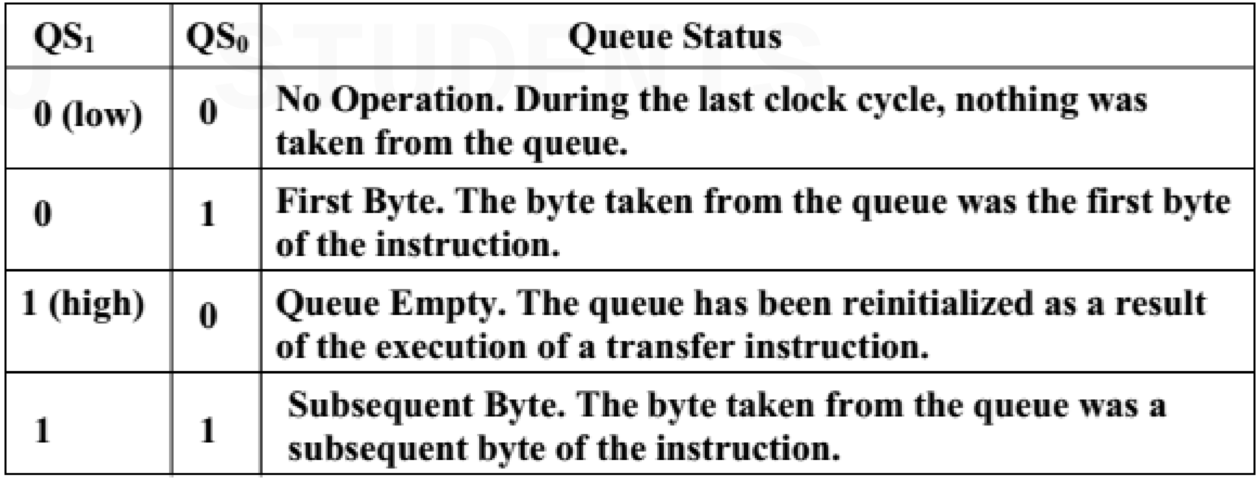
  $\overline{\text{GT0}}$ will have a higher prority than $\overline{\text{GT1}}$

- **$\overline{\text{LOCK}}$** – *Output signal* - Activated by LOCK prefix instruction, active low signal

  It prevent other bus masters from gaining the control of system bus

MAX mode Signals

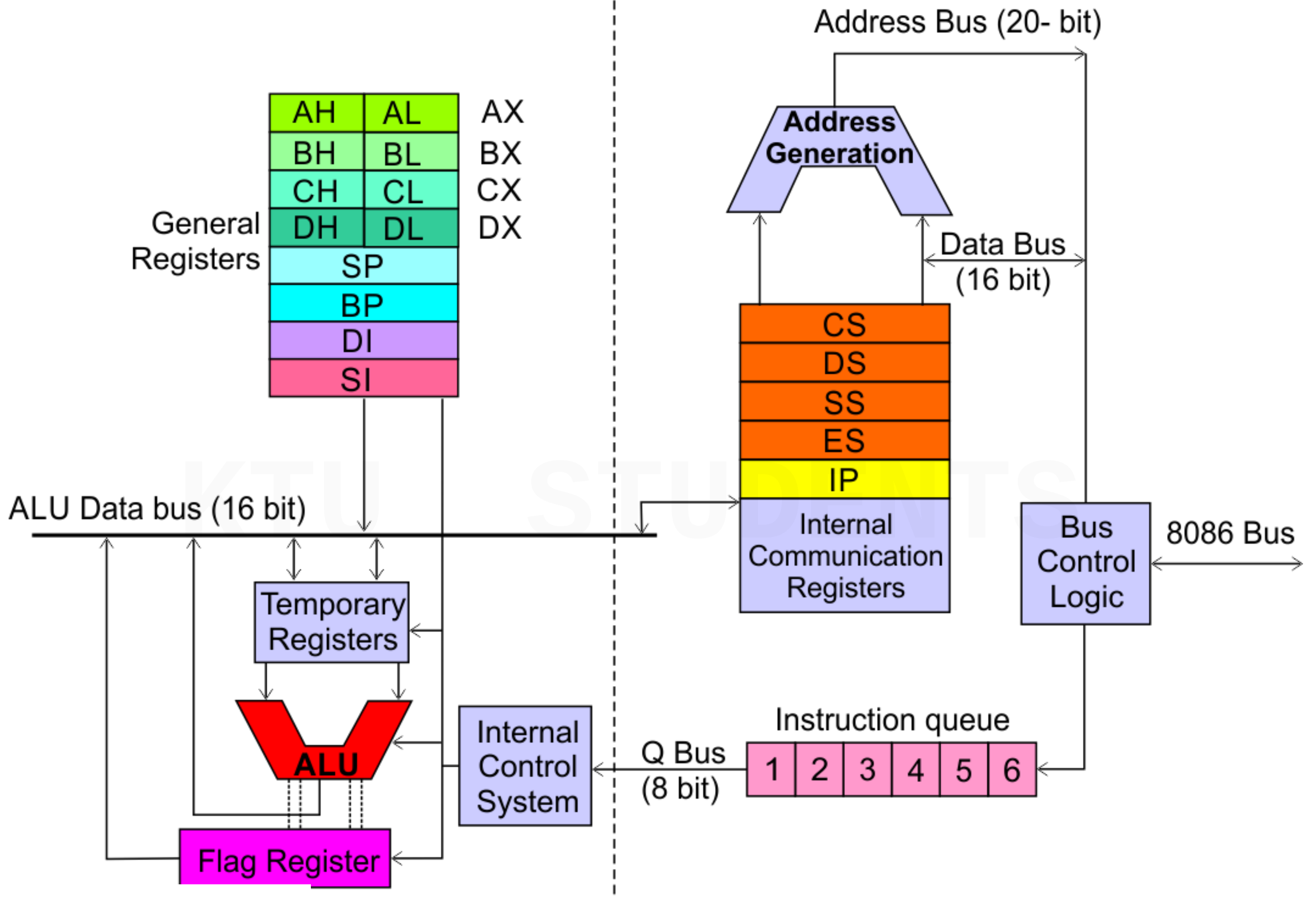
- **QS1- QS0-** The processor provides status of queue to the external devices



QS_1	QS_0	Queue Status
0 (low)	0	No Operation. During the last clock cycle, nothing was taken from the queue.
0	1	First Byte. The byte taken from the queue was the first byte of the instruction.
1 (high)	0	Queue Empty. The queue has been reinitialized as a result of the execution of a transfer instruction.
1	1	Subsequent Byte. The byte taken from the queue was a subsequent byte of the instruction.

Queue status codes

8086 Architecture



Execution Unit (EU)

Bus Interface Unit (BIU)

INTEL 8086 Architecture

- 8086 has pipelined architecture, where functional units and execution time of functional units are overlapped.

The functional units work independently most of the time

- 8086 is internally divided as 2 functional units
  - **Bus Interface Unit (BIU)**
  - **Execution Unit (EU)**

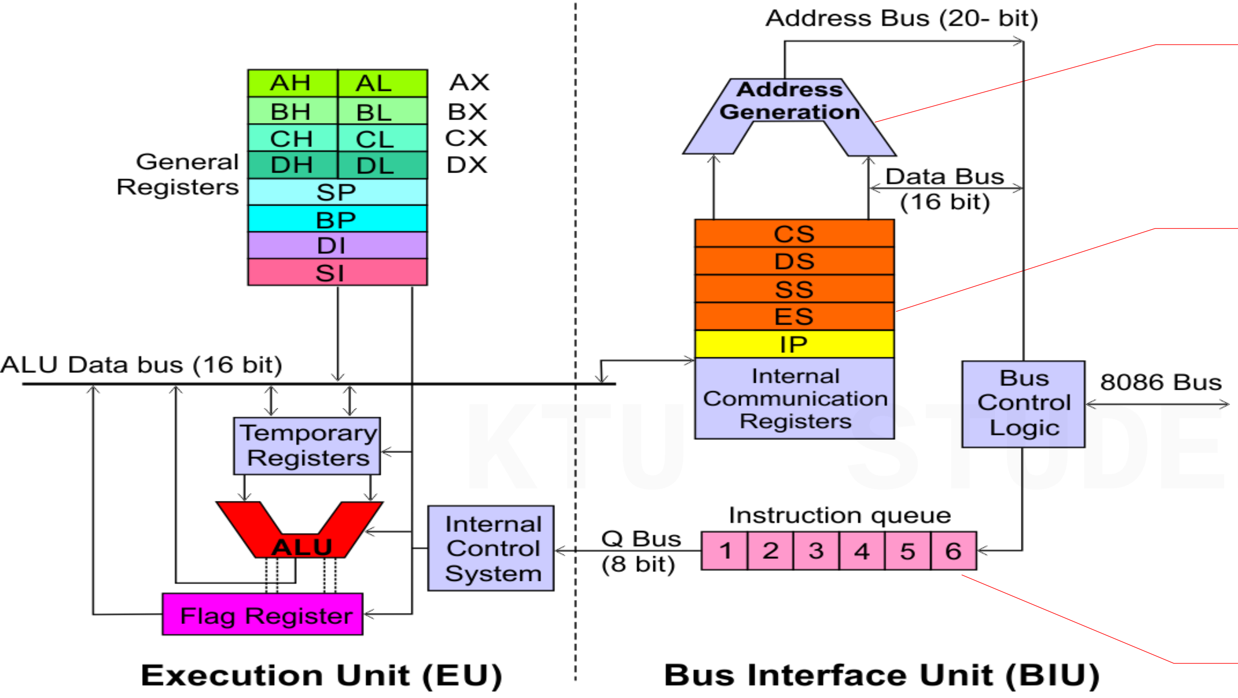
INTEL 8086 Architecture

- Bus Interface Unit BIU – It fetches, reads data from memory and I/O ports, writes data to memory and I/O ports.

  It contains Segment Registers, Instruction Pointer, Instruction queue, Address generation unit and bus control unit
- Execution Unit EU executes instruction that have already beeen fetched by the BIU

8086 Microprocessor

Bus Interface Unit (BIU)



Dedicated Adder to generate 20 bit address

Four 16-bit segment registers

Code Segment (CS)
Data Segment (DS)
Stack Segment (SS)
Extra Segment (ES)

IP – Instruction Pointer

Instruction queue- pipeline

8086 Microprocessor

Execution Unit (EU)

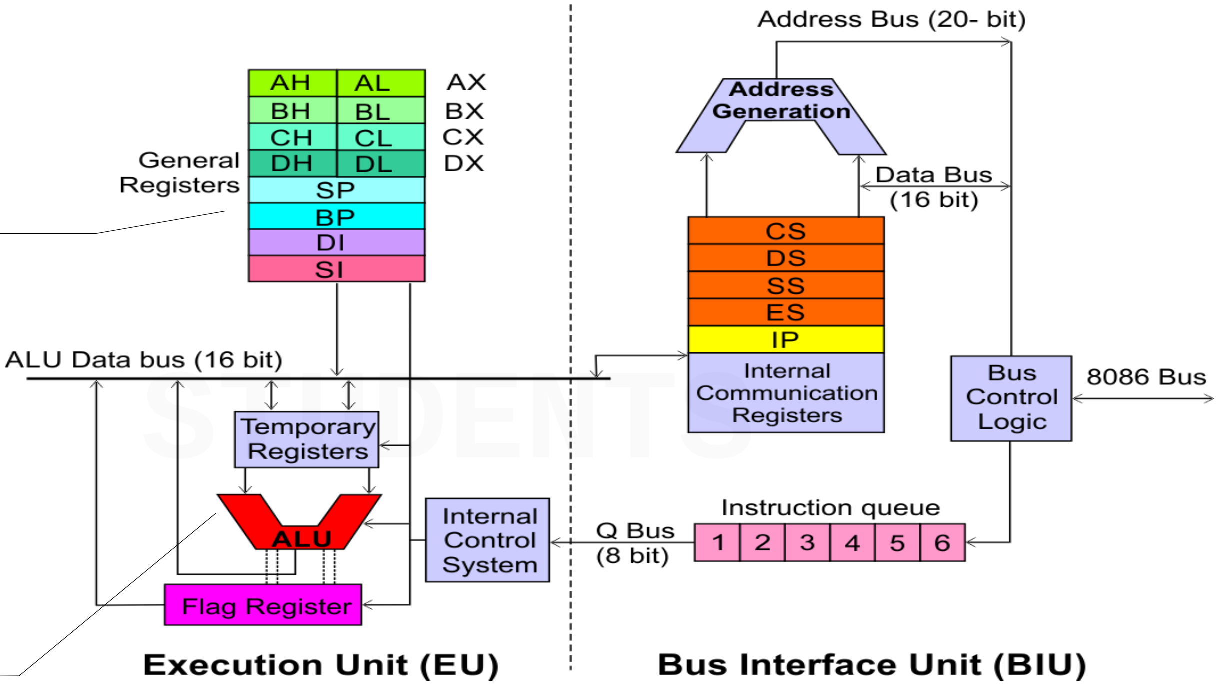
Four general purpose registers(AX, BX, CX, DX);

Pointer registers (Stack Pointer, Base Pointer);

and

Index registers (Source Index, Destination Index) each of 16-bits

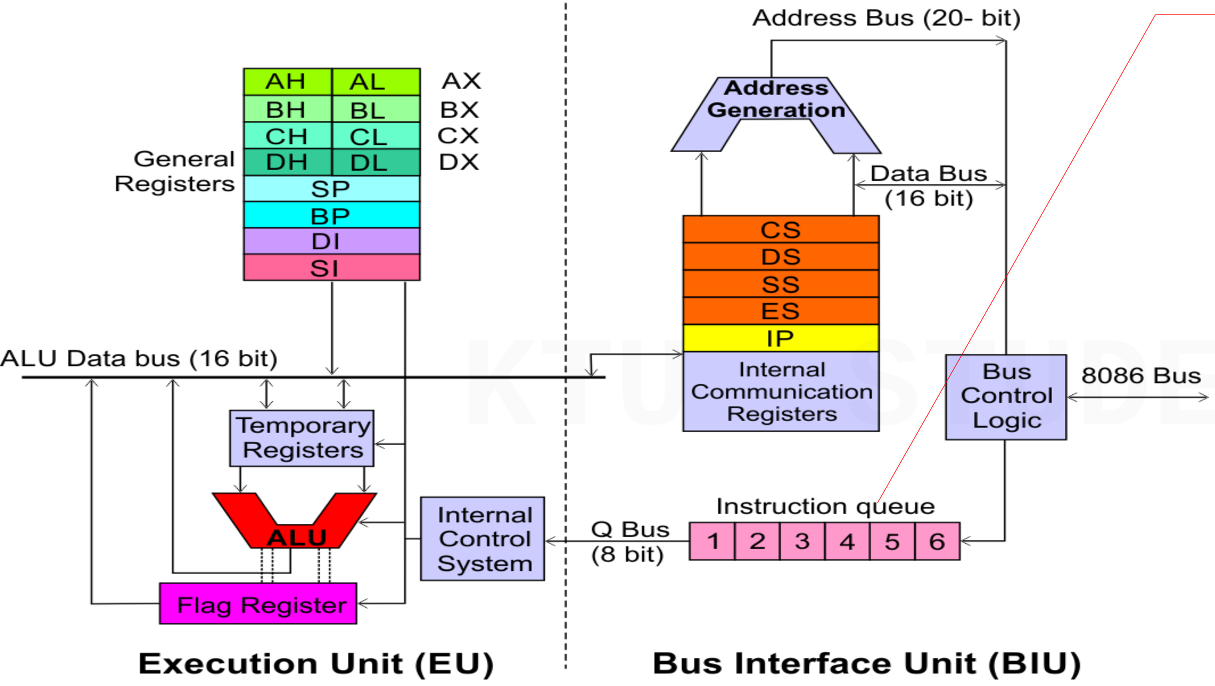
16-bit ALU for performing arithmetic and logic operation



EU decodes and executes instructions.

A decoder in the EU control system translates instructions.

Bus Interface Unit (BIU)



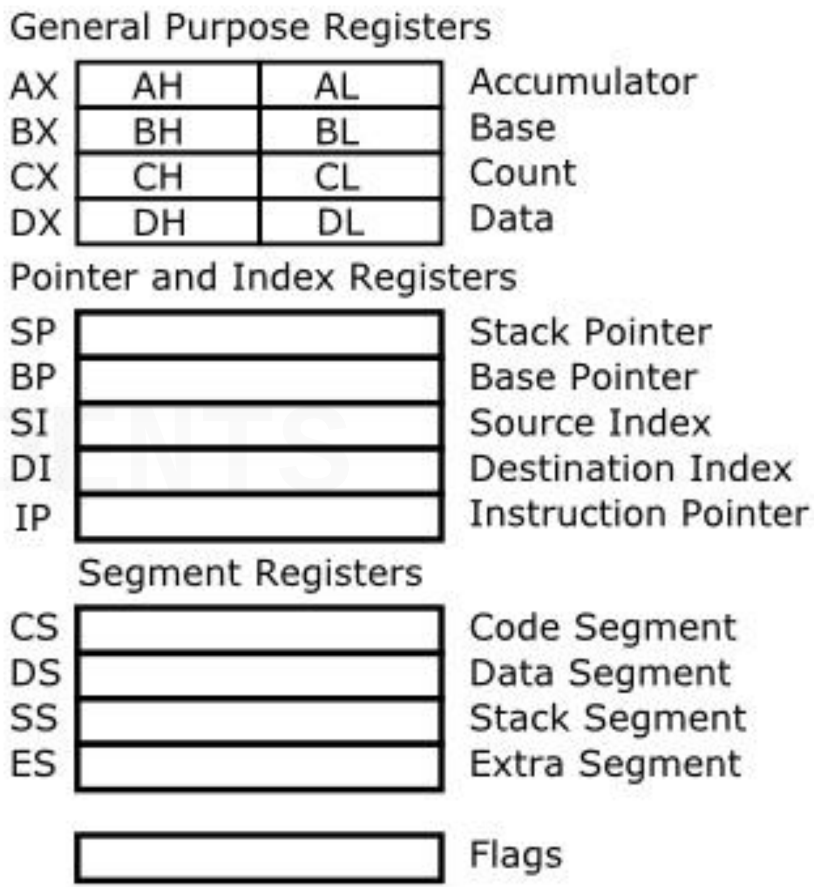
Instruction queue

- A group of First-In-First-Out (FIFO) in which up to 6 bytes of instruction code are pre fetched from the memory ahead of time.
- This is done in order to speed up the execution by overlapping instruction fetch with execution.
- This mechanism is known as <u>pipelining</u>.

The Registers

Register units are classified as

1. General purpose registers
2. Pointer and Index Registers
3. Memory Segment Registers
4. Flag Registers



Types if Registers in 8086

Sl.No.	Type	Register width	Name of register
1	General purpose register	16 bit	AX, BX, CX, DX
		8 bit	AL, AH, BL, BH, CL, CH, DL, DH
2	Pointer register	16 bit	SP, BP
3	Index register	16 bit	SI, DI
4	Instruction Pointer	16 bit	IP
5	Segment register	16 bit	CS, DS, SS, ES
6	Flag Register	16 bit	O,D,I,T,S,Z, AC,P,C

INTEL 8086

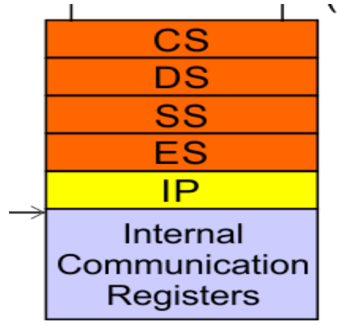
Segment Registers

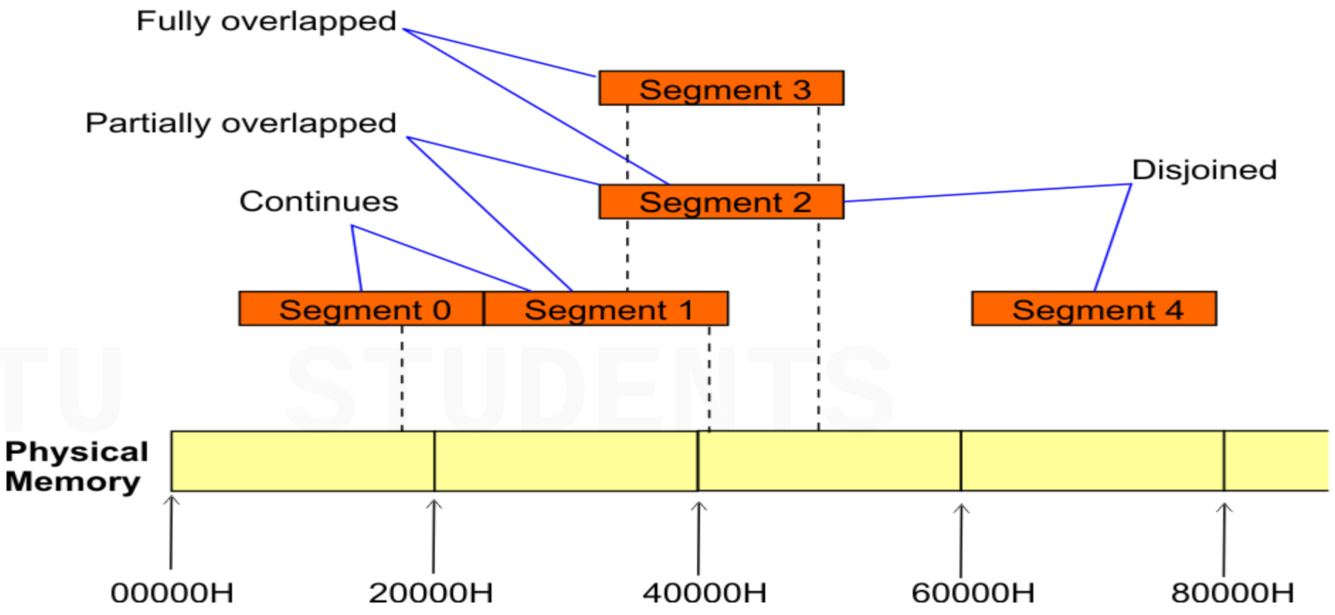
- 8086 BIU sends out 20 bit address, so it can access 2^{20}=1Mb memory
- However, at a given time 8086 works only with *four 64kb segments* within 1Mb
- Four Segment registers in BIU are used to hold the **upper 16 bits of starting address** of the 4 memory segments

1. Code Segment (CS) Register
2. Stack Segment (SS) Register
3. Data Segment (DS) Register
4. Extra Segment (ES) Register

Bus Interface Unit (BIU)

Segment Registers





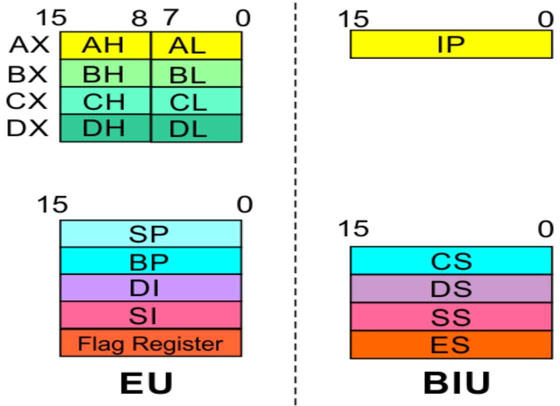
- The *four 64kb* may be continuous partially overlapped, fully overlapped or disjoint

Bus Interface Unit (BIU)

Segment Registers

Code Segment Register

- 16-bit
- CS contains the *upper 16 bits* of the starting address for segment from which BIU is fetching the instruction code bytes
- IP contains the *16 bit address offset* to the next instruction byte to be fetched.

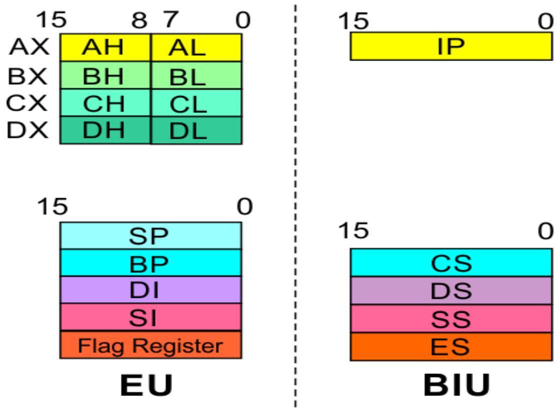


Bus Interface Unit (BIU)

Segment Registers

Data Segment Register

- 16-bit
- DS contains the *upper 16 bits* of the starting address for memory segment used for data
- Operands for most instructions are fetched from this segment.

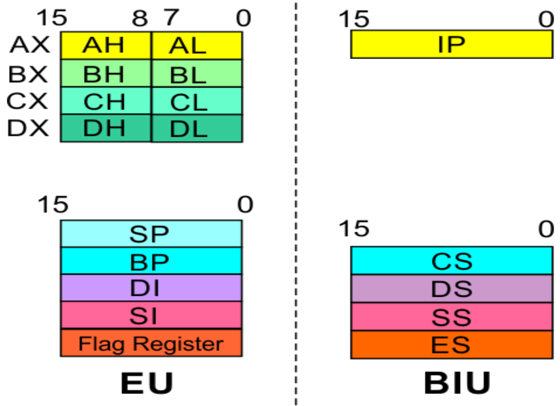


Bus Interface Unit (BIU)

Segment Registers

Stack Segment Register

- 16-bit
- Stack is a section of memory, set aside to store address & data while a sub program executes
- 8086 has set aside an entire 64kb segment as Stack
- The *upper 16 bit* of starting address of this segment is kept in Stack Segment Register

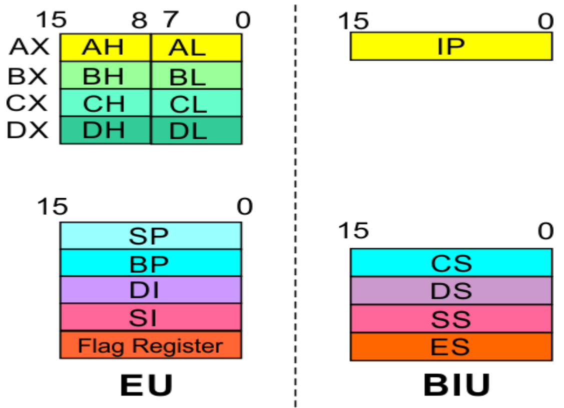


Bus Interface Unit (BIU)

Segment Registers

Extra Segment Register

- 16-bit
- Points to the extra segment in which data is stored.
- ES contains the *upper 16 bits* of the starting address for memory segment used for data *in excess pointed to by DS*

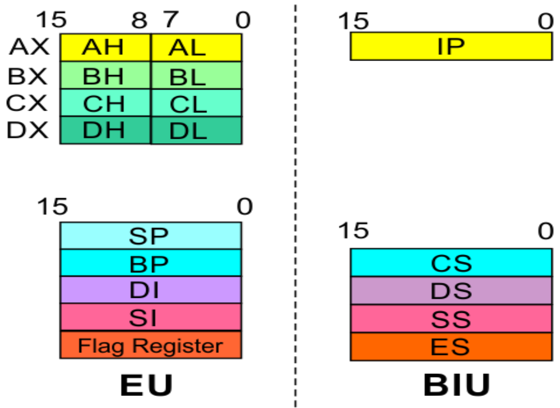


Bus Interface Unit (BIU)

Segment Registers

Instruction Pointer

- 16-bit
- Always points to the next instruction to be executed within the currently executing code segment.
- So, this register contains the 16-bit offset address pointing to the next instruction code within the 64Kb of the code segment area.
- Its content is automatically incremented as the execution of the next instruction takes place.

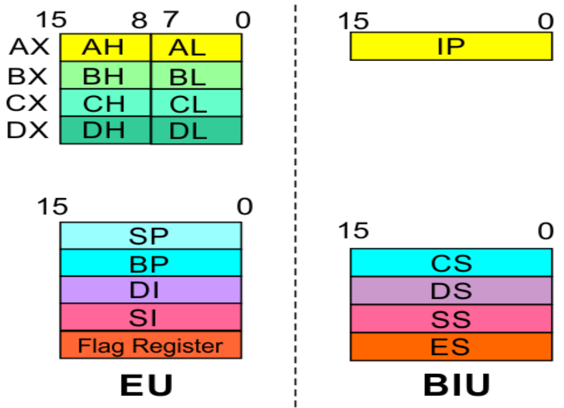


Execution Unit (EU)

EU Registers

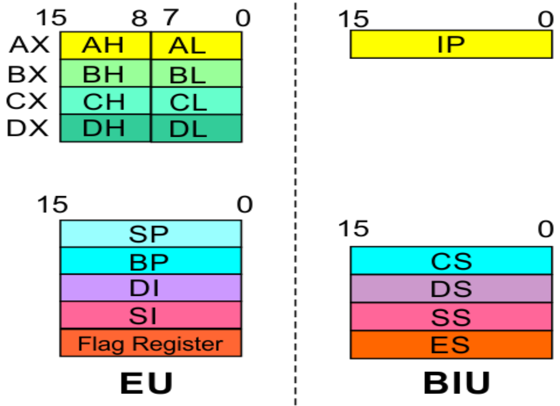
Accumulator Register (AX)

- 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.
- The I/O instructions use the AX or AL for inputting / outputting 16 or 8 bit data to or from an I/O port.



Execution Unit (EU)

EU Registers



Base Register (BX)

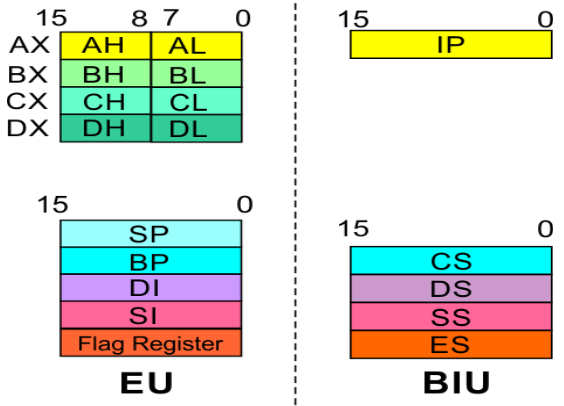
- Consists of two 8-bit registers BL and BH, which can be combined together and used as a 16-bit register BX.
- BL in this case contains the low-order byte of the word, and BH contains the high-order byte.
- Used to store base value in base addressing mode
- All memory references of this register content uses DS as the default segment register.

Execution Unit (EU)

EU Registers

Counter Register (CX)

- Consists of two 8-bit registers CL and CH, which can be combined together and used as a 16-bit register CX.
- When combined, CL register contains the low order byte of the word, and CH contains the high-order byte.
- Instructions such as SHIFT, ROTATE and LOOP use the contents of CX as a counter.



Example:

The instruction LOOP START automatically decrements CX by 1 without affecting flags and will check if [CX] = 0.

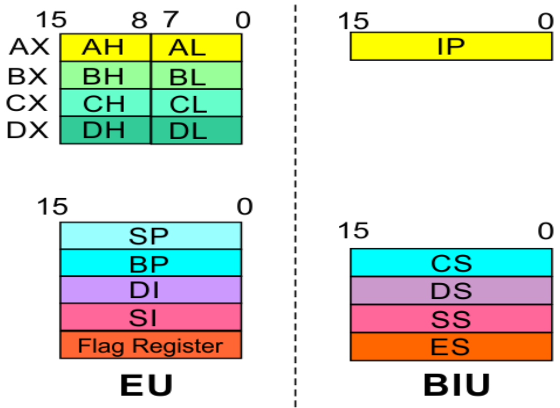
If it is zero, 8086 executes the next instruction; otherwise the 8086 branches to the label START.

Execution Unit (EU)

EU Registers

Data Register (DX)

- Consists of two 8-bit registers DL and DH, which can be combined together and used as a 16-bit register DX.
- When combined, DL register contains the low order byte of the word, and DH contains the high-order byte.
- It is usually used to hold data for multiplication and division

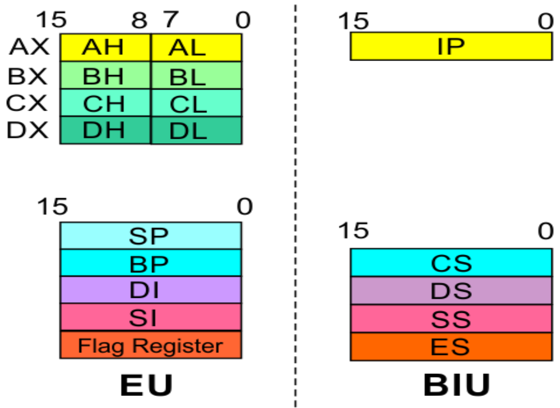


Execution Unit (EU)

EU Registers

Stack Pointer (SP) and Base Pointer (BP)

- SP and BP are used to access data in the stack segment.
- Stack is a section of memory, set aside to store address & data while a sub program executes
- The location where a word was stored most recently is called top of stack
- SP register holds 16 bit offset address from Start of Segment to Top of stack
- BP register holds base value of stack used in Base Addressing mode

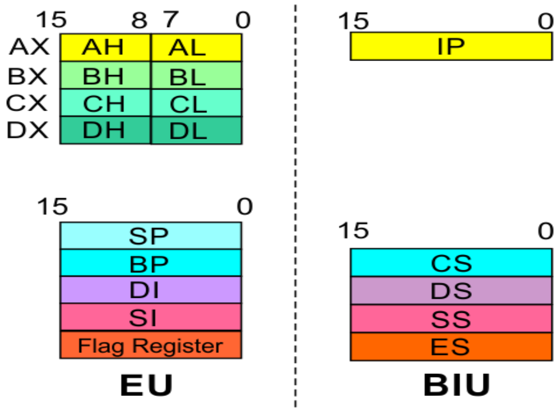


Execution Unit (EU)

EU Registers

Source Index (SI) and Destination Index (DI)

- Used in indexed addressing modes.
- SI- Used to hold index value of source operand (data) for string instructions
- DI- Used to hold the index value of destination operand (data) for string operations



Register	Name of the Register	Special Function
AX	16-bit Accumulator	Stores the 16-bit results of arithmetic and logic operations
AL	8-bit Accumulator	Stores the 8-bit results of arithmetic and logic operations
BX	Base register	Used to hold base value in base addressing mode to access memory data
CX	Count Register	Used to hold the count value in SHIFT, ROTATE and LOOP instructions
DX	Data Register	Used to hold data for multiplication and division operations
SP	Stack Pointer	Used to hold the offset address of top stack memory
BP	Base Pointer	Used to hold the base value in base addressing using SS register to access data from stack memory
SI	Source Index	Used to hold index value of source operand (data) for string instructions
DI	Data Index	Used to hold the index value of destination operand (data) for string operations

Flag Registers

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

0	CF	**Carry Flag** This flag is set, when there is a carry out of MSB in case of addition or a borrow in case of subtraction
2	PF	**Parity Flag** This flag is set to 1, if the result contains even parity; ie. number of 1's ; for odd parity flag is zero.
4	AC	**Auxiliary Carry Flag** This is set, if there is a carry from the low nibble to high nibble, during addition, or borrow for the low nibble to high nibble, during subtraction. i.e, carry or borrow from bit three
6	ZC	**Zero Flag** This flag is set, if the result of the computation or comparison performed by an instruction is zero

7	SF	**Sign Flag** This flag is set to 1, when the result of any computation is negative
8	TF	**Trap Flag** If this flag is set, the processor enters the *single step execution mode* by generating internal interrupts after the execution of each instruction. It is used for debugging the program
9	IF	**Interrupt Flag** Setting IF to 1 causes 8086 to recognize external mask able interrupts, reset to 0 disables interrupts
10	DF	**Direction Flag** This is set to 1 for auto decrement and is reset to 0 for auto increment of SI and DI during string data accessing
11	OF	**Over flow Flag** This flag is set, if an arithmetic overflow occurs, i.e, if the size of the result exceeds he capacity of destination location *For more study materials>www.ktustudents.in*

Min mode 8086 system

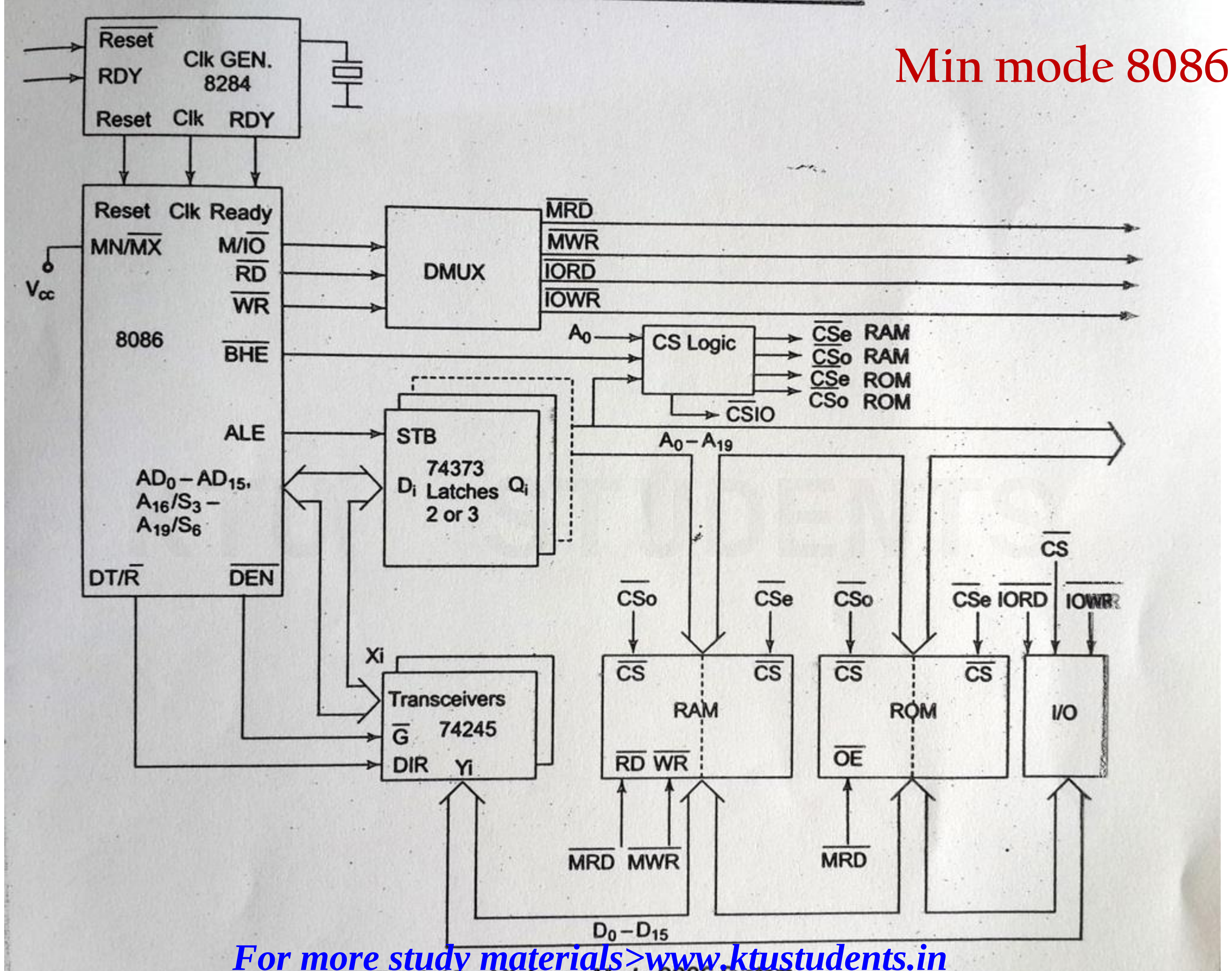


Fig. 1.13 Minimum Mode 8086 System

Read Cycle Minimum mode

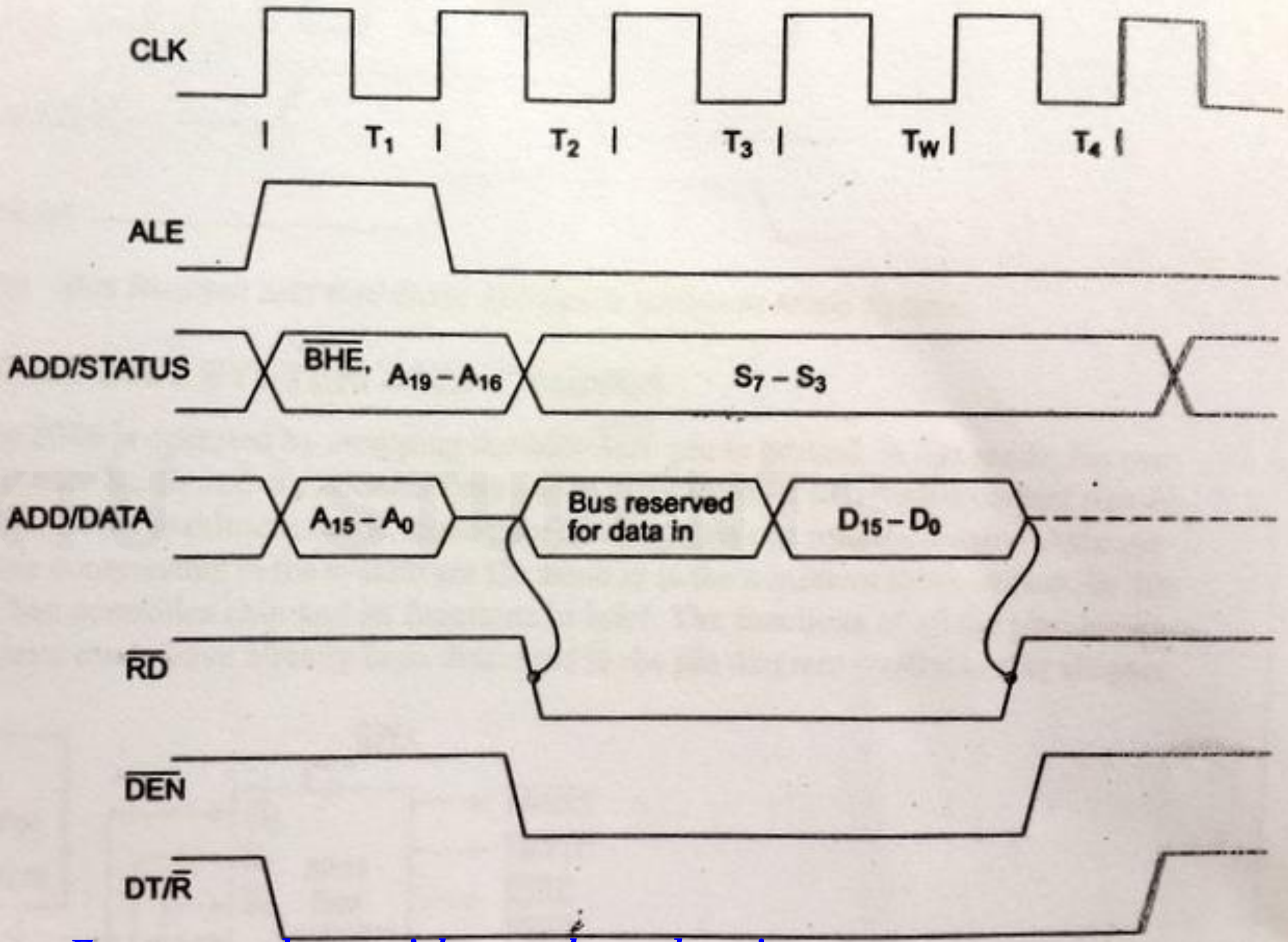


Fig. 1.14(a) ***Read Cycle Timing Diagram for Minimum Mode***

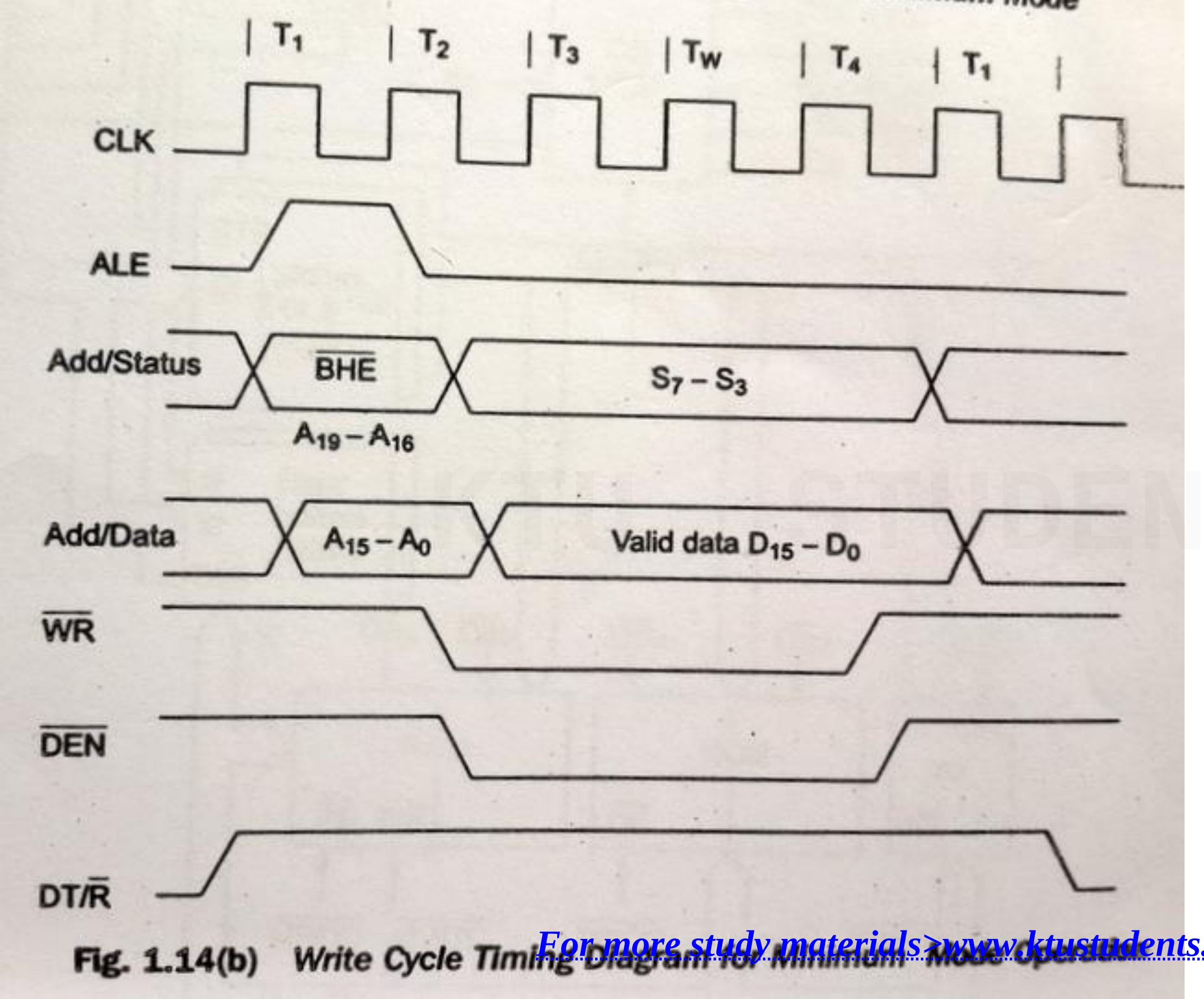


Fig. 1.14(b) Write Cycle Timing Diagram for Minimum Mode Operation

Write Cycle Minimum mode

Maximum mode 8086 system

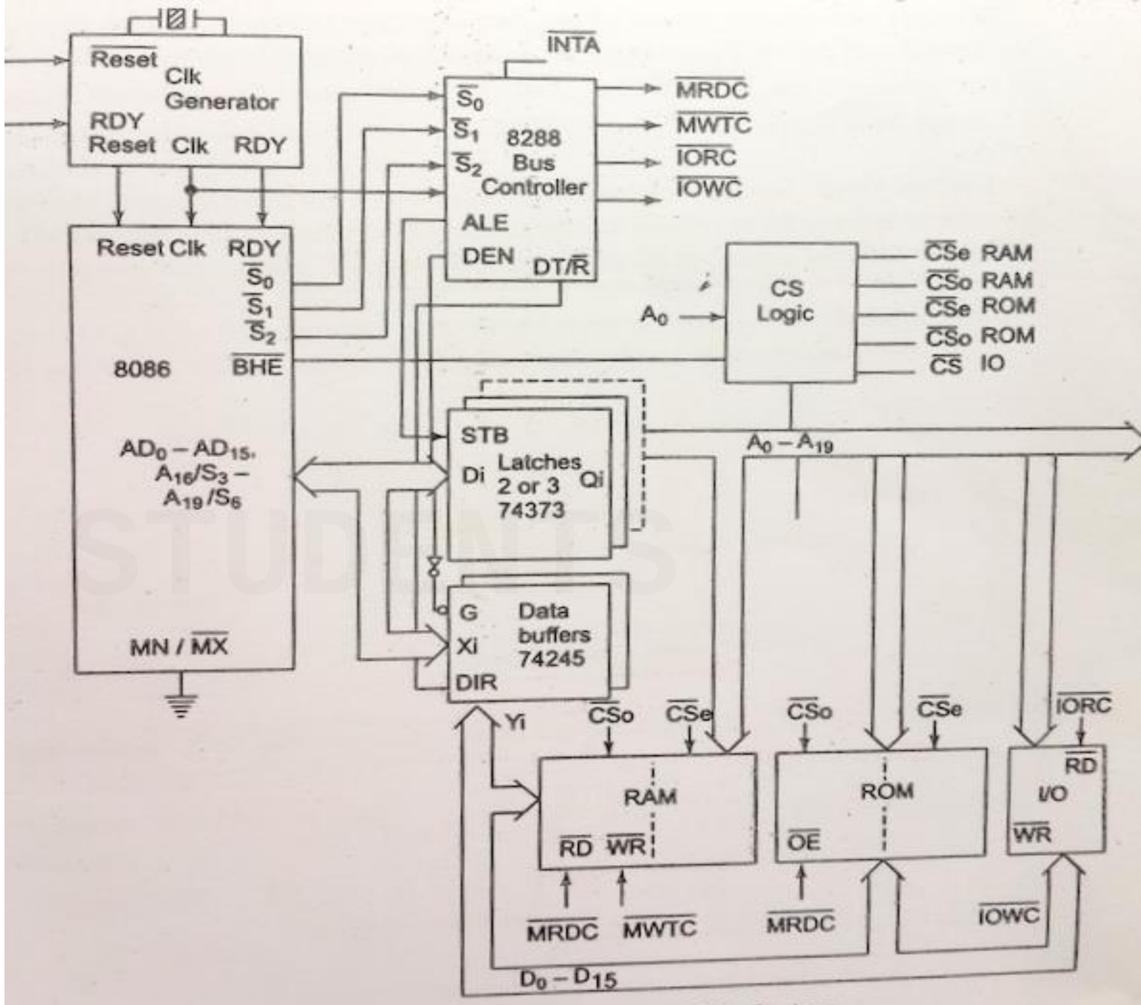


Fig. ... Maximum Mode 8086 System

Read Cycle Maximum mode

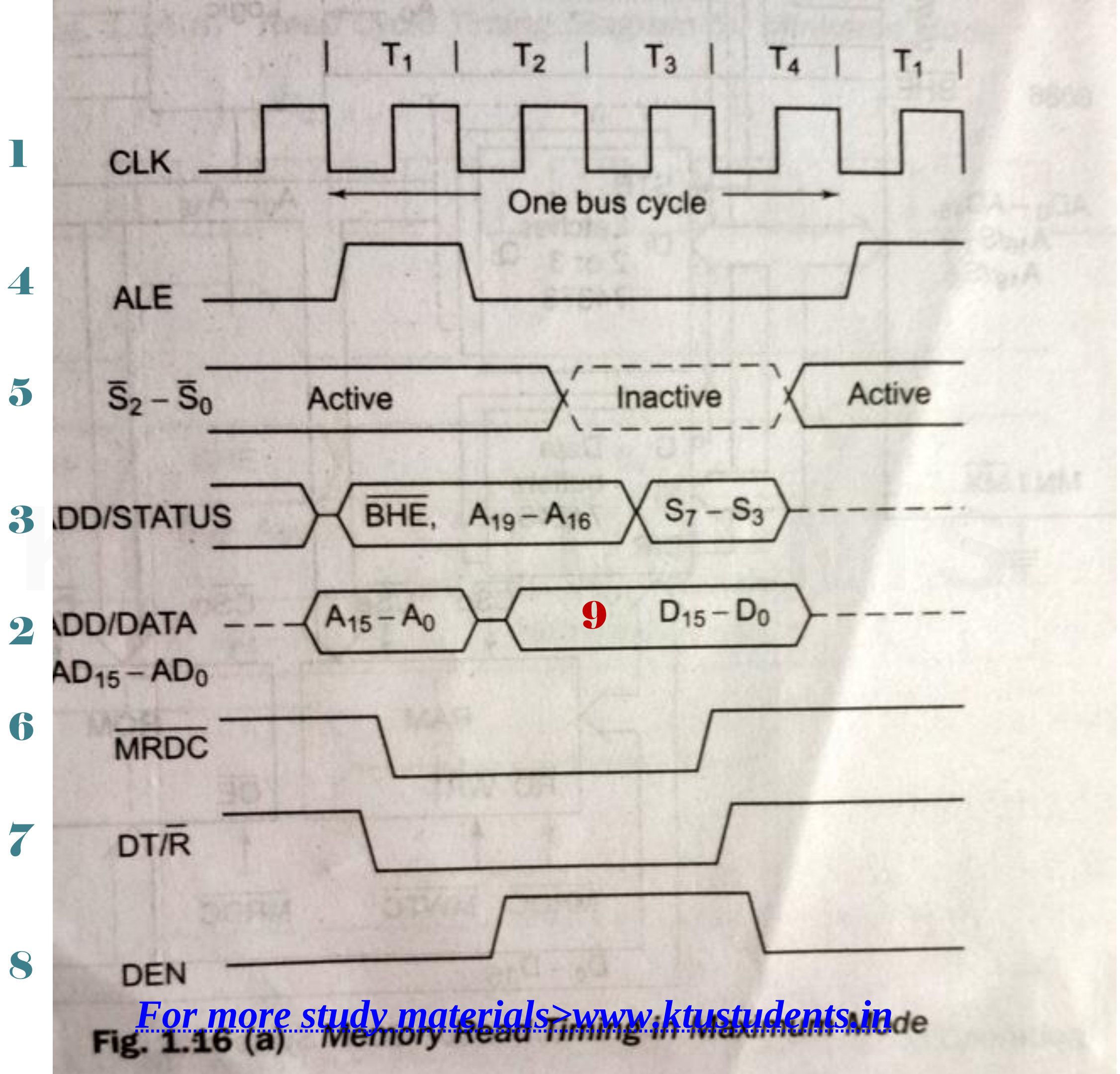


Fig. 1.16 (a) Memory Read Timing in Maximum Mode

Write Cycle Maximum mode

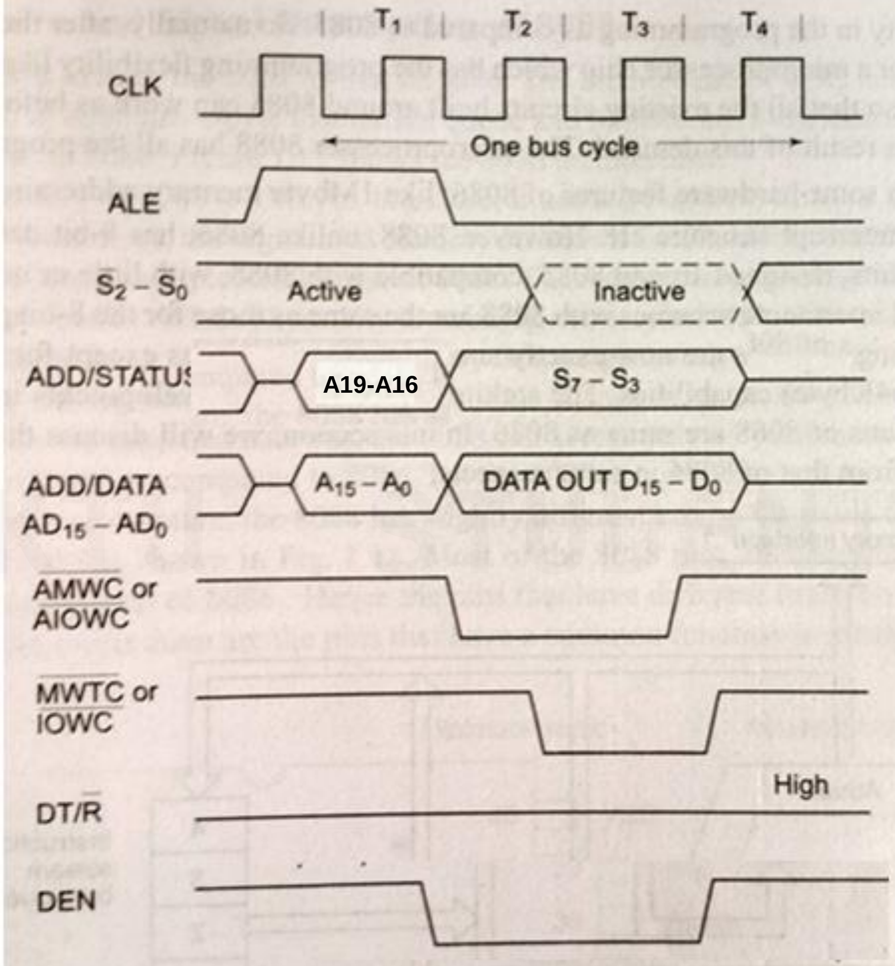


Fig. [illegible] Memory Write Timing in Maximum Mode

8086 Family

- 8086 has two family of processors. They are 8086 and 8088.

- 8088 uses 8 bit data bus externally, 8086 uses 16 it data bus externally
- 8088 memory access in bytes, 8086 memory access in words

IBMs first PC was using 8088 microprocessor as CPU

Comparison of 8086 and 8088

No	8086	8088
1	16 bit data bus, 20 bit address bus	8 bit data bus, 20 bit address bus
2	Access memory in bytes and words	Access memory in bytes only
3	Uses $\overline{\text{BHE}}$ signal to access higher bye	No higher byte, hence $\overline{\text{BHE}}$ signal is not there
4	6 byte instruction queue	4 byte instruction queue
5	M/$\overline{\text{IO}}$ for memory or IO access	$\overline{\text{IO}}$/M for memory or IO access