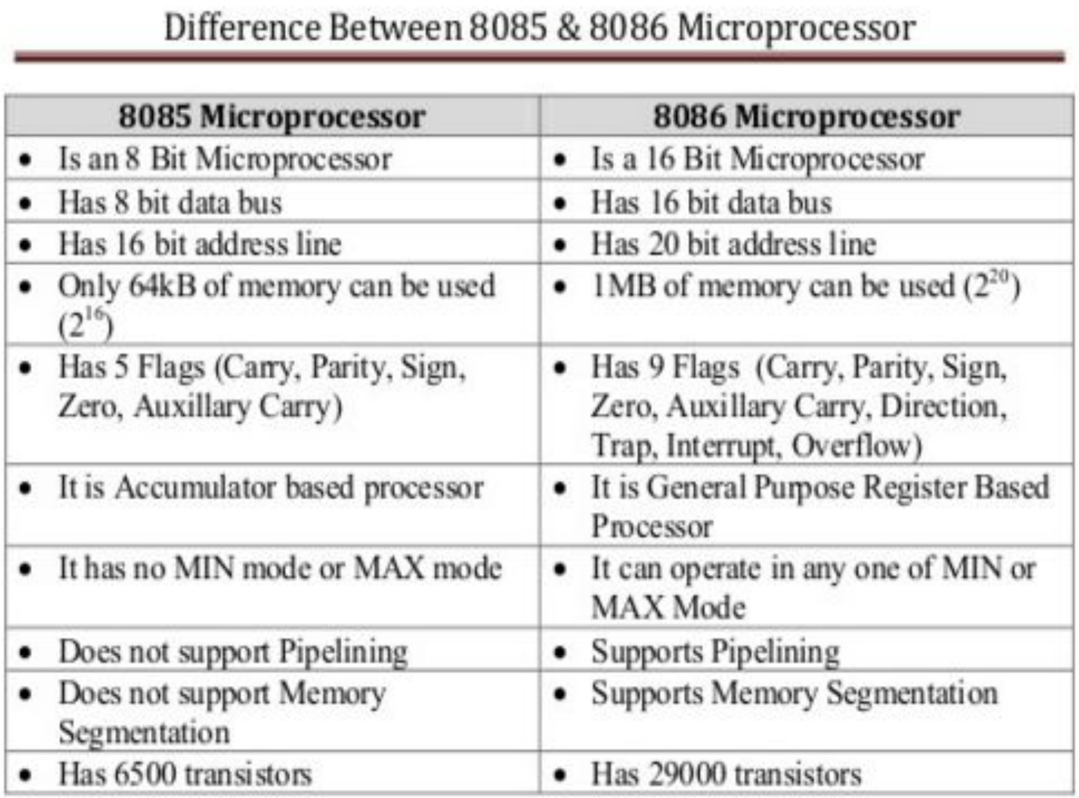


Difference Between 8085 & 8086 Microprocessor

8085 Microprocessor	8086 Microprocessor
• Is an 8 Bit Microprocessor	• Is a 16 Bit Microprocessor
• Has 8 bit data bus	• Has 16 bit data bus
• Has 16 bit address line	• Has 20 bit address line
• Only 64kB of memory can be used (2^{16})	• 1MB of memory can be used (2^{20})
• Has 5 Flags (Carry, Parity, Sign, Zero, Auxillary Carry)	• Has 9 Flags (Carry, Parity, Sign, Zero, Auxillary Carry, Direction, Trap, Interrupt, Overflow)
• It is Accumulator based processor	• It is General Purpose Register Based Processor
• It has no MIN mode or MAX mode	• It can operate in any one of MIN or MAX Mode
• Does not support Pipelining	• Supports Pipelining
• Does not support Memory Segmentation	• Supports Memory Segmentation
• Has 6500 transistors	• Has 29000 transistors

8085 Vs 8086

- There are some of the difference mentioned below:

1.Size:-
8085 is 8 bit microprocessor whereas 8086 is 16 bit microprocessor.

2.Address Bus:-
8085 has 16 bit address bus and 8086 has 20 bit addres bus.

3.Memory:-
8085 can access upto 2^16 = 64 KB of memory whereas 8086 can access upto 2^20 = 1 MB of memory.

4.Instruction Queue:-
8085 doesn't have an instruction queue whereas 8086 has instruction queue.

5.Pipelining:-
8085 does not support pipelined architechture whereas 8086 supports pipelined architechture.

6.Multiprocessing Support:-
8085 does not support multiprocessing support whereas 8086 supports.

- **7.I/O:-**
  8085 can address 2^8 = 256 I/O's and 8086 can access 2^16 = 65,536 I/Os

  8.Arithmetic Support:-
  8085 only supports integer and decimal whereas 8086 supports integer, decimal and ASCII arithmetic.

  9.Multiplication and Division:-
  8085 doesn't support whereas 8086 supports.

  10. Operating Modes:-
  8085 supports only single operating mode whereas 8086 operates in two modes.

  11.External Hardware:-
  8085 requires less external hardware whereas 8086 requires more external hardware.

  12.Cost:-
  The cost of 8085 is low and 8086 is high.

  13.Memory Segmentation:-
  In 8085, memory space is not segmented but in 8086, memory space is segmented.

8086 Microprocessor

Microprocessor

Program controlled semiconductor device (IC) which fetches (from memory), decodes and executes instructions.

It is used as CPU (Central Processing Unit) in computers.

Features

- ❑ ***It is a 16-bit μp.***
- ❑ ***8086 has a 20 bit address bus can access up to 2^{20} memory locations (1 MB).***
- ❑ ***It can support up to 64K I/O ports.***
- ❑ ***It provides 14, 16 -bit registers.***
- ❑ ***Word size is 16 bits.***
- ❑ ***It has multiplexed address and data bus AD0- AD15 and A16 – A19.***
- ❑ ***It requires single phase clock with 33% duty cycle to provide internal timing.***

- ***8086 is designed to operate in two modes, Minimum and Maximum.***

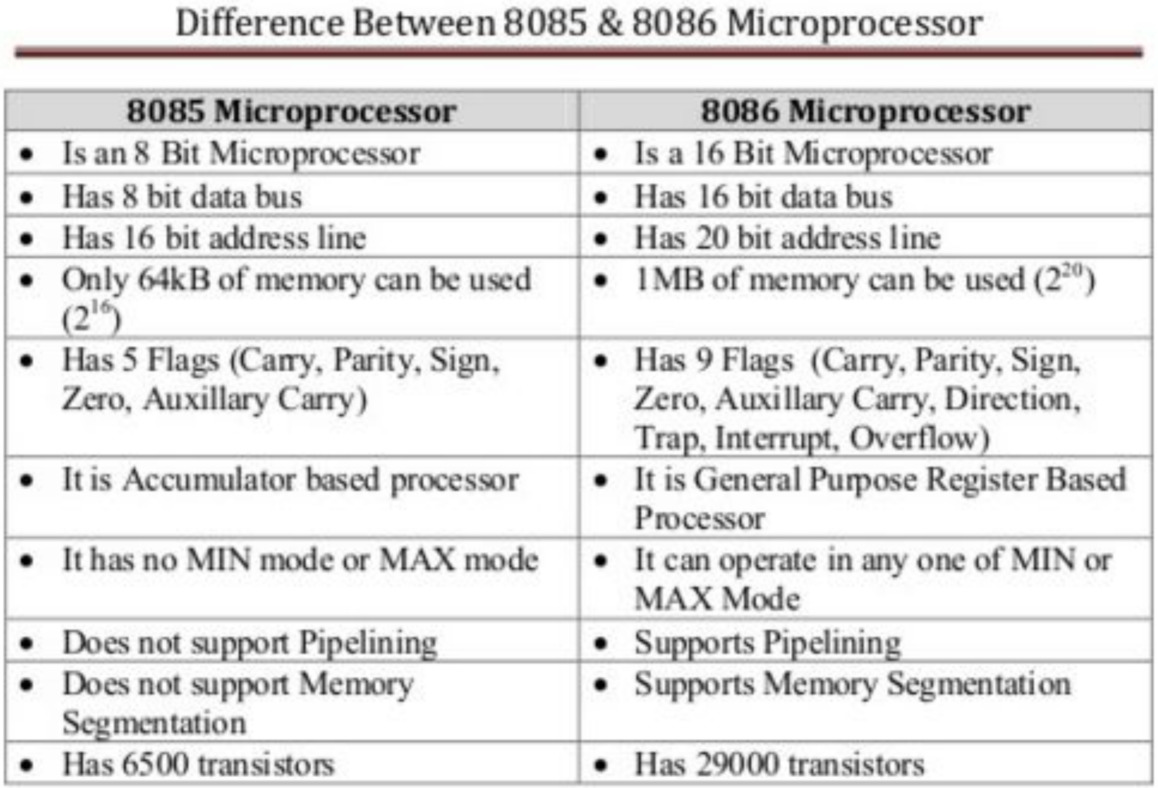
- ***It prefetches up to 6 instruction bytes from memory and queues them in order to speed up instruction execution.***

- ***It requires +5V power supply.***

- ***A 40 pin dual in line package.***

- ***Address ranges from 00000H to FFFFFH***

- ***Memory is byte addressable - Every byte has a separate address.***



Difference Between 8085 & 8086 Microprocessor

8085 Microprocessor	8086 Microprocessor
• Is an 8 Bit Microprocessor	• Is a 16 Bit Microprocessor
• Has 8 bit data bus	• Has 16 bit data bus
• Has 16 bit address line	• Has 20 bit address line
• Only 64kB of memory can be used (2^{16})	• 1MB of memory can be used (2^{20})
• Has 5 Flags (Carry, Parity, Sign, Zero, Auxillary Carry)	• Has 9 Flags (Carry, Parity, Sign, Zero, Auxillary Carry, Direction, Trap, Interrupt, Overflow)
• It is Accumulator based processor	• It is General Purpose Register Based Processor
• It has no MIN mode or MAX mode	• It can operate in any one of MIN or MAX Mode
• Does not support Pipelining	• Supports Pipelining
• Does not support Memory Segmentation	• Supports Memory Segmentation
• Has 6500 transistors	• Has 29000 transistors

Intel 8086 Internal Architecture

Internal architecture of 8086

The architecture of 8086 includes

- Arithmetic Logic Unit (ALU)
- Flags
- General registers
- Instruction byte queue
- Segment registers

- **The 8086 CPU logic has been partitioned into two functional units namely**

- **Bus Interface Unit (BIU) and**
- **Execution Unit (EU)**

- The major reason for this separation is to increase the processing speed of the processor.

- The BIU has to interact with memory and input and output devices in fetching the instructions and data required by the EU.

- EU is responsible for executing the instructions of the programs and to carry out the required processing

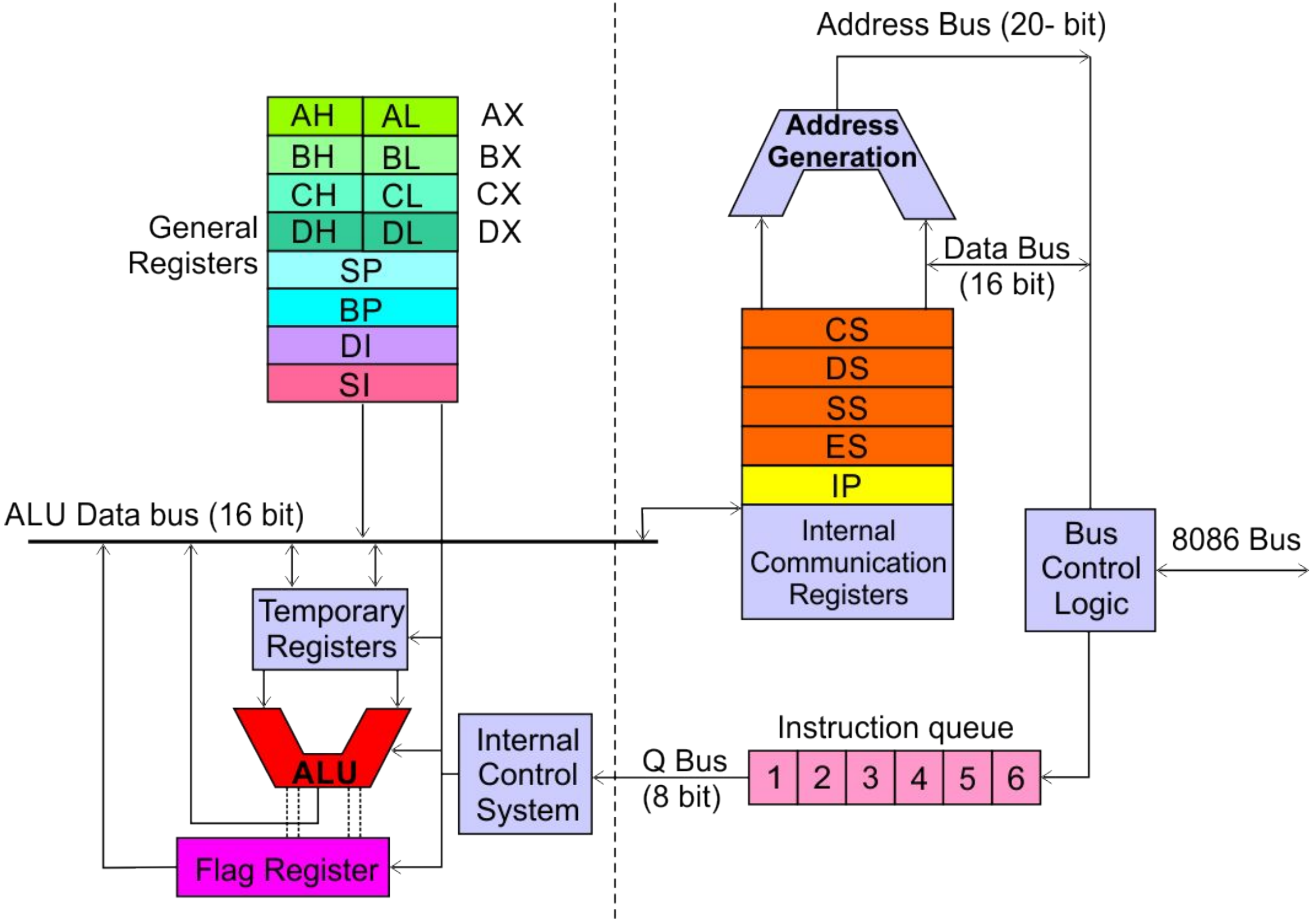
BIU:

- The BIU performs all bus operations for EU
- Fetching instructions
- Responsible for executing all external Bus cycles.
- Read operands and write result

EU:

- Execution unit contains the complete infrastructure required to execute an instruction
- Decodes instructions fetched by the BIU
- Generate control signals

Architecture



Execution Unit (EU)

EU executes instructions that have already been fetched by the BIU.

BIU and EU functions separately.

Bus Interface Unit (BIU)

BIU fetches instructions, reads data from memory and I/O ports, writes data to memory and I/ O ports.

EXECUTION UNIT

The main parts are:

- Control Circuitry
- Instruction decoder
- ALU
- General purpose registers Ax,Bx,Cx,Dx
- Pointer Registers SP,BP
- Index Registers SI,DI
- Flag Register

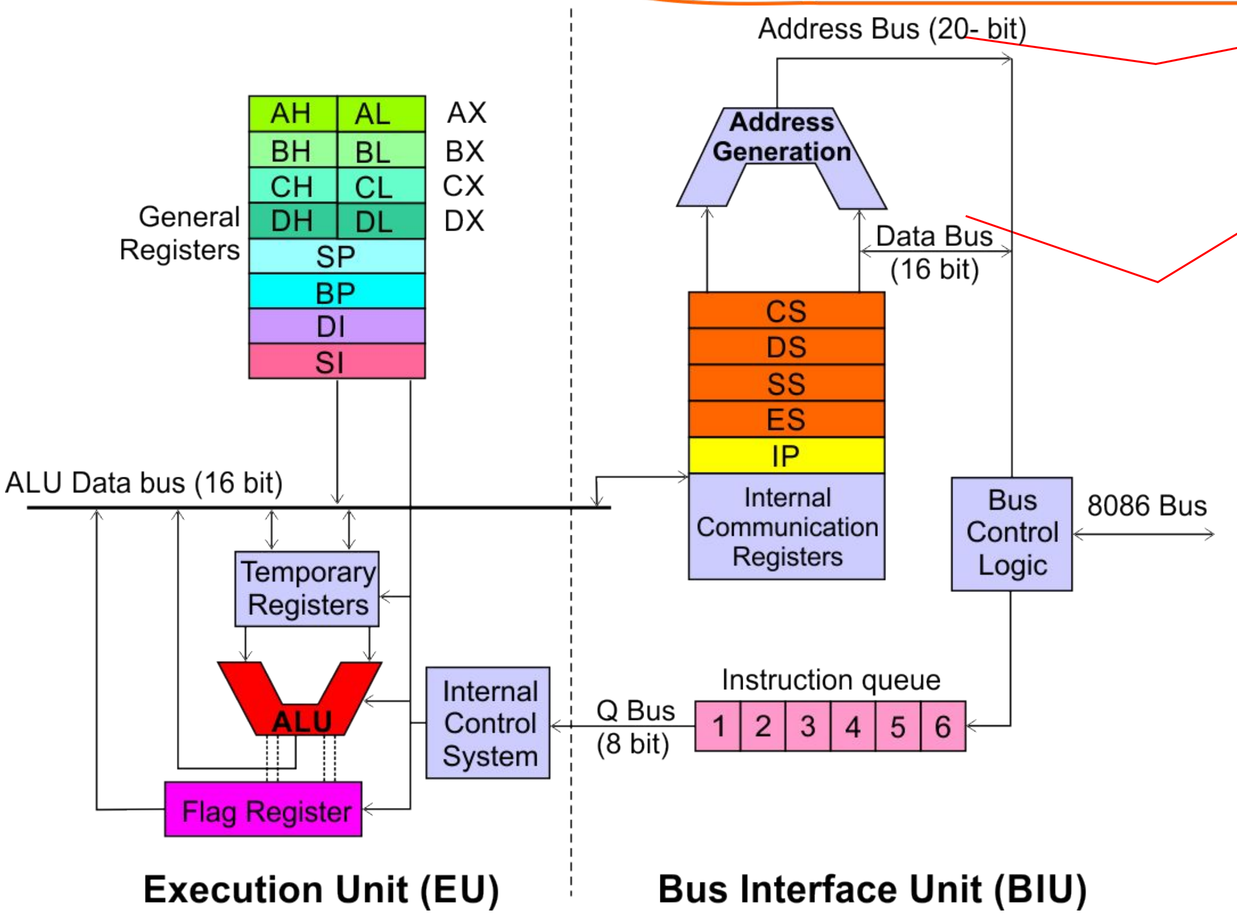
BUS INTERFACE UNIT (BIU)

Contains

- 6-byte Instruction Queue (Q)
- The Segment Registers (CS, DS, ES, SS).
- The Instruction Pointer (IP).
- The Address Summing block (Σ)

Architecture

Bus Interface Unit (BIU)



Dedicated Adder to generate 20 bit address

Four 16-bit segment registers

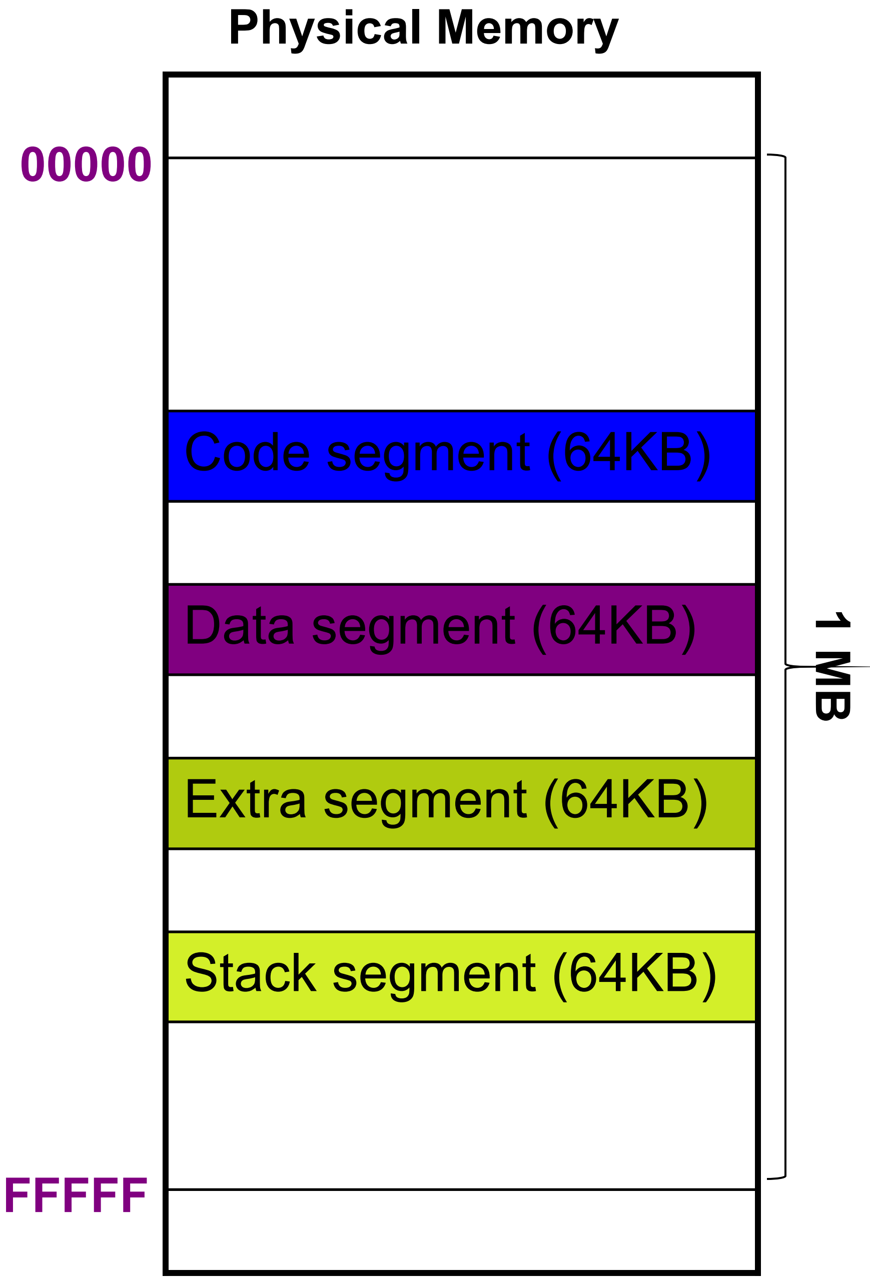
Code Segment (CS)
Data Segment (DS)
Stack Segment (SS)
Extra Segment (ES)

Segmented Memory

- The memory in an 8086/88 based system is organized as segmented memory.

- The CPU 8086 is able to address 1Mbyte of memory.

- The Complete physically available memory may be divided into a number of logical segments.



SEGMENTATIOM

Segmentation is the process in which the main memory of the processor is logically divided into different segments and each segment has its own base address.

It is basically used to enhance the speed of execution of the system, so that the processor is able to fetch and execute the data from the memory easily and fast.

- The number of address lines in 8086 is 20, 8086 BIU will send 20bit address, so as to access one of the 1MB memory locations.

- The four segment registers actually contain the upper 16 bits of the starting addresses of the four memory segments of 64 KB each with which the 8086 is working at that instant of time.

The Bus Interface Unit (BIU) contains four 16 bit special purpose registers (mentioned below) called as Segment Registers.

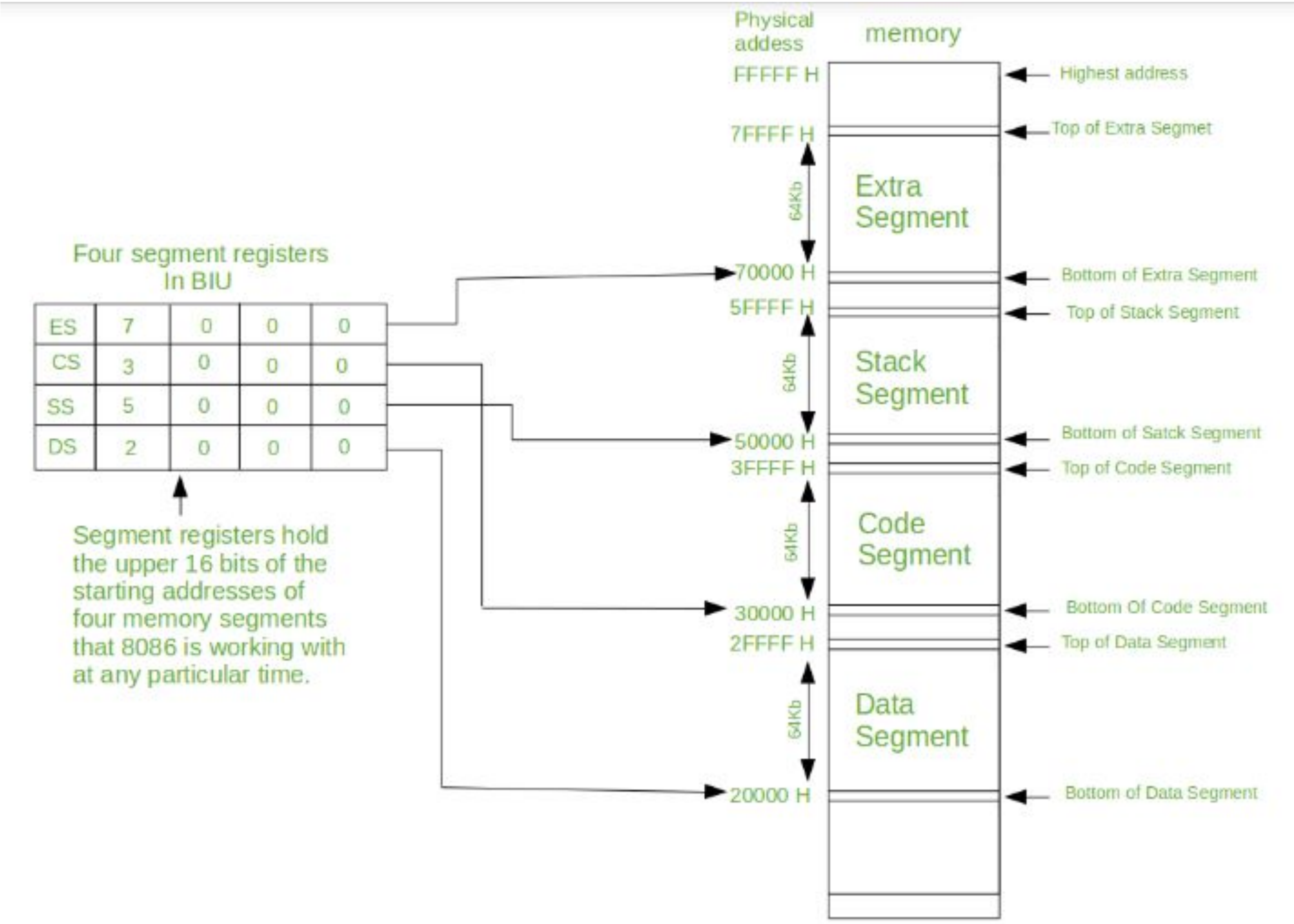
Code segment register (CS): is used for addressing memory location in the code segment of the memory, where the executable program is stored.

Data segment register (DS): points to the data segment of the memory where the data is stored.

Extra Segment Register (ES): also refers to a segment in the memory which is another data segment in the memory.

Stack Segment Register (SS): is used for addressing stack segment of the memory. The stack segment is that segment of memory which is used to store stack data.

Memory Segmentation



Four segment registers In BIU

ES	7	0	0	0
CS	3	0	0	0
SS	5	0	0	0
DS	2	0	0	0

Segment registers hold the upper 16 bits of the starting addresses of four memory segments that 8086 is working with at any particular time.

Advantages of Segmented memory Scheme

The main advantages of segmentation are as follows:

- It provides a powerful memory management mechanism.
- Data related or stack related operations can be performed in different segments.
- Code related operation can be done in separate code segments.
- It allows to processes to easily share data.
- Allows the placing of code, data and stack portions of the same program in different parts (segments) of the memory for data and code protection.
- It allows to extend the address ability of the processor, i.e. segmentation allows the use of 16 bit registers to give an addressing capability of 1 Megabytes. Without segmentation, it would require 20 bit registers.
- It is possible to enhance the memory size of code data or stack segments beyond 64 KB by allotting more than one segment for each area.

CS Register

This register contains the initial address of the code segment. This address plus the offset value contained in the instruction pointer (IP) indicates the address of an instruction to be fetched for execution.

SS Register

The stack segment register contains the initial address of the stack segment. This address plus the value contained in the stack pointer (SP) is used for stack operations.

DS Register

The Data segment register contains the initial address of the current data segment. This address plus the offset value in instruction causes a reference to a specific location in the data segment.

ES Register

Extra segment is used by some string operations. The Extra segment register contains the initial address of the extra segment. String instructions always use the ES and DI registers to calculate the physical address for the destination.

SEGMENT:OFFSET NOTATION

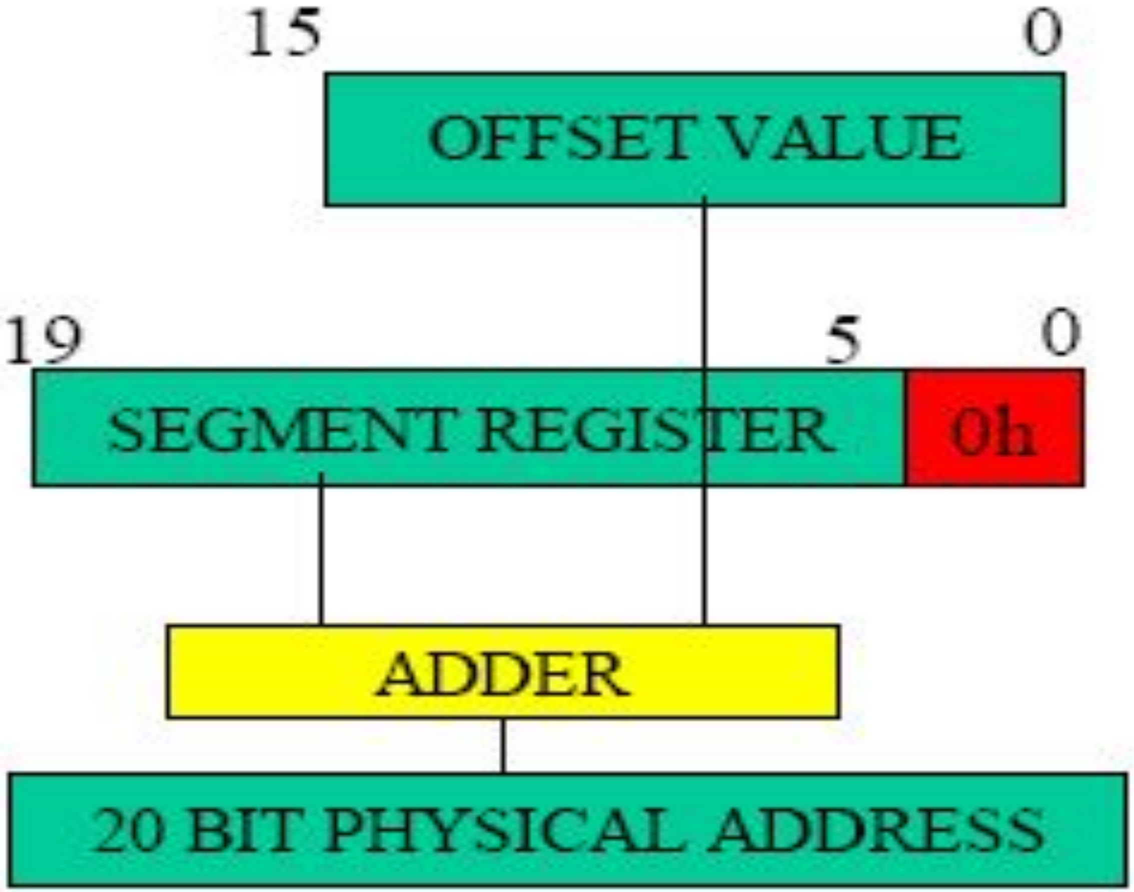
- The total addressable memory size is 1MB.

- Most of the processor instructions use 16-bit pointers the processor can effectively address only 64 KB of memory.

- To access memory outside of 64 KB the CPU uses special segment registers to specify where the code, stack and data 64 KB segments are positioned within 1 MB of memory

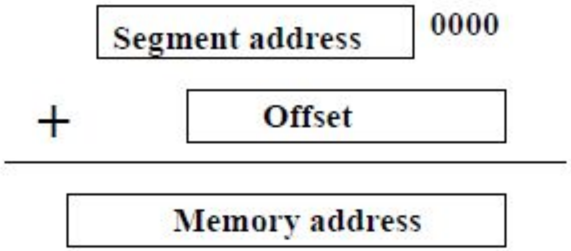
SEGMENT:OFFSET NOTATION

- A simple scheme would be to order the bytes in a serial fashion and number them from 0 (or 1) to the end of memory.
- The scheme used in the 8086 is called segmentation.
- Every address has two parts, a SEGMENT and an OFFSET (Segment:Offset)
- The segment indicates the starting of a 64KB of memory, in multiples of 16
- The offset indicates the position within the 64k portion
- Absolute address = (segment * 16) + offset


15
0
OFFSET VALUE
19
5
0
SEGMENT REGISTER
0h
ADDER
20 BIT PHYSICAL ADDRESS

Memory Address Calculation

- ❑ Segment addresses must be stored in segment registers
- ❑ Offset is derived from the combination of pointer registers, the Instruction Pointer (IP), and immediate values



- ❑ Examples

CS=348A
IP=4214

CS	3	4	8	A	0
IP +		4	2	1	4
Instruction address	3	8	A	B	4

SS	5	0	0	0	0
SP +		F	F	E	0
Stack address	5	F	F	E	0

DS	1	2	3	4	0
DI +		0	0	2	2
Data address	1	2	3	6	2

- **The following examples shows the CS:IP scheme of address formation:**

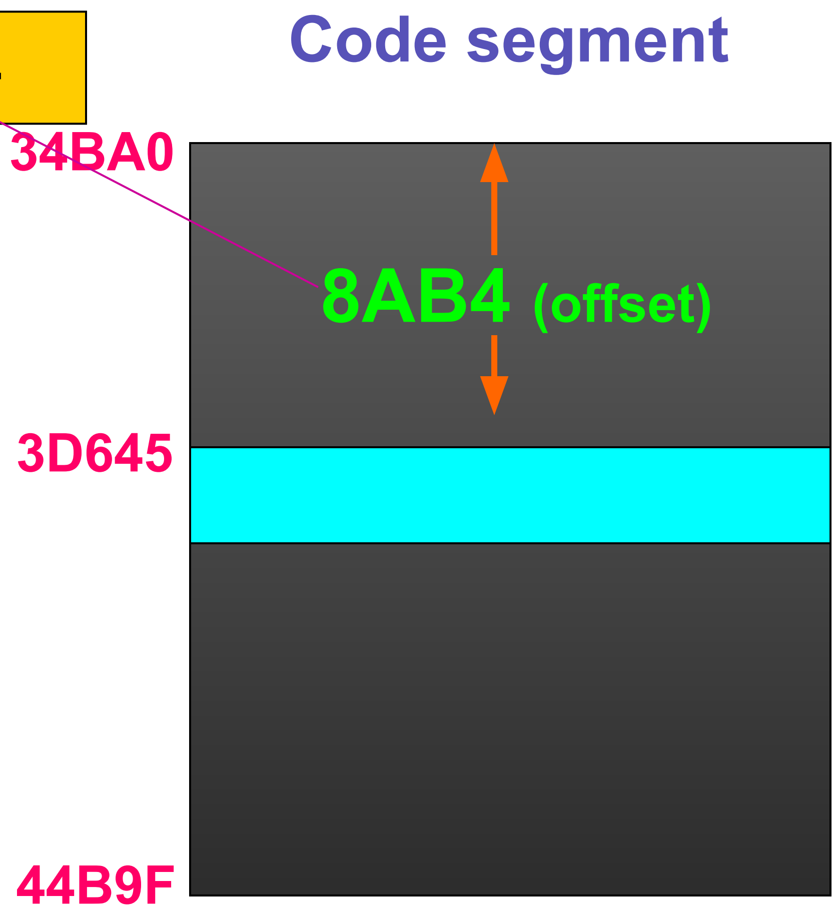
CS 34BA

IP 8AB4

Inserting a hexadecimal 0H (0000B) with the CSR or shifting the CSR four binary digits left

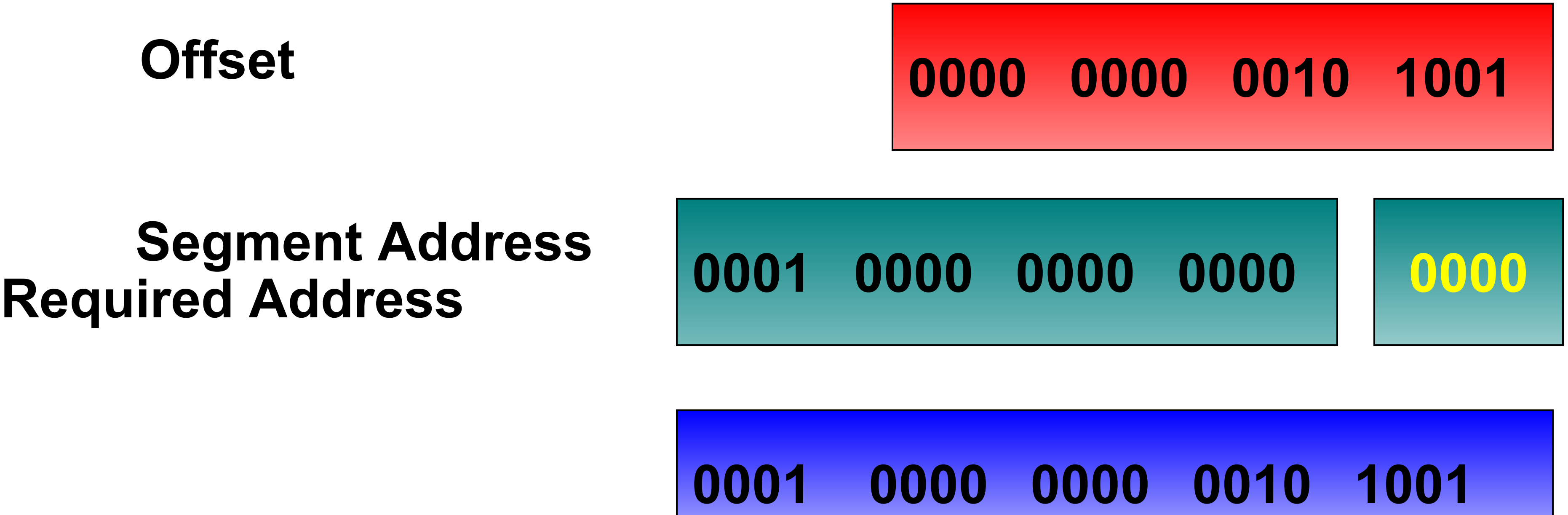
34BA0 (CS) +
8AB4 (IP)

3D654 (next address)



- **Example For Address Calculation (segment: offset)**

- If the data segment starts at location 1000h and a data reference contains the address 29h where is the actual data?



Offset 0000 0000 0010 1001

Segment Address 0001 0000 0000 0000 0000

Required Address

0001 0000 0000 0010 1001

- CS=5000   50000
- IP=8500   8500
-                     58500
- CS:IP
- 20BIT MEMORY ADDRESS=48530
-                     OFFSET= 8530
-   =4000
- BASE ADDRESS IN SEGMENT REGISTER

»

Segment and Address register combination

- CS:IP
- SS:SP SS:BP
- DS:BX
  DS:SI (for string operations)
- DS:DI (for other than string operations)
- ES:DI (for string operations)

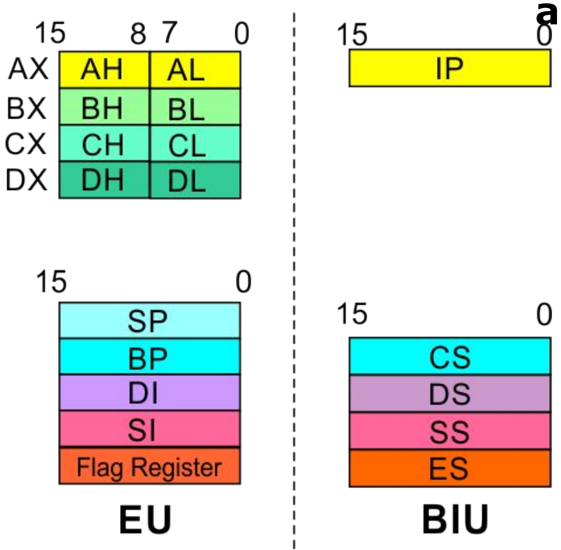
Architecture

Bus Interface Unit (BIU)

Segment Registers

Code Segment Register

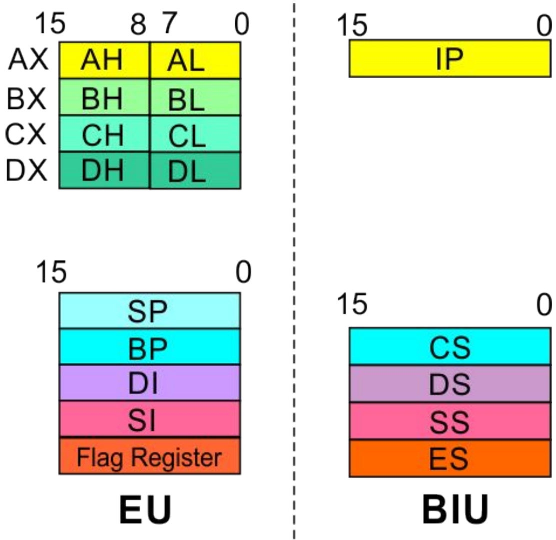
- 16-bit
- CS contains the base or start of the current code segment; IP contains the distance or offset from this address to the next instruction byte to be fetched.
- BIU computes the 20-bit physical address by logically shifting the contents of CS 4-bits to the left and then adding the 16-bit contents of IP.
- That is, all instructions of a program are relative to the contents of the CS register multiplied by 16 and then offset is added provided by the IP.



Segment Registers

Data Segment Register

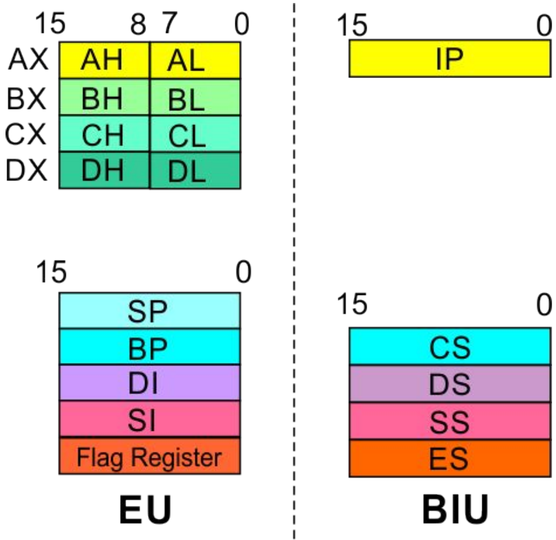
- 16-bit
- Points to the current data segment; operands for most instructions are fetched from this segment.
- The 16-bit contents of the Source Index (SI) or Destination Index (DI) or a 16-bit displacement are used as offset for computing the 20-bit physical address.



Segment Registers

Stack Segment Register

- 16-bit
- Points to the current stack.
- The 20-bit physical stack address is calculated from the Stack Segment (SS) and the Stack Pointer (SP) for stack instructions such as PUSH and POP.
- In based addressing mode, the 20-bit physical stack address is calculated from the Stack segment (SS) and the Base Pointer (BP).



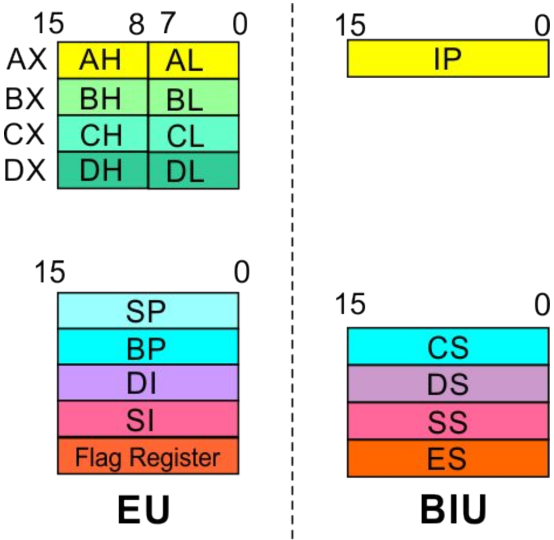
Architecture

Bus Interface Unit (BIU)

Segment Registers

Extra Segment Register

- 16-bit
- Points to the extra segment in which data (in excess of 64K pointed to by the DS) is stored.
- String instructions use the ES and DI to determine the 20-bit physical address for the destination.



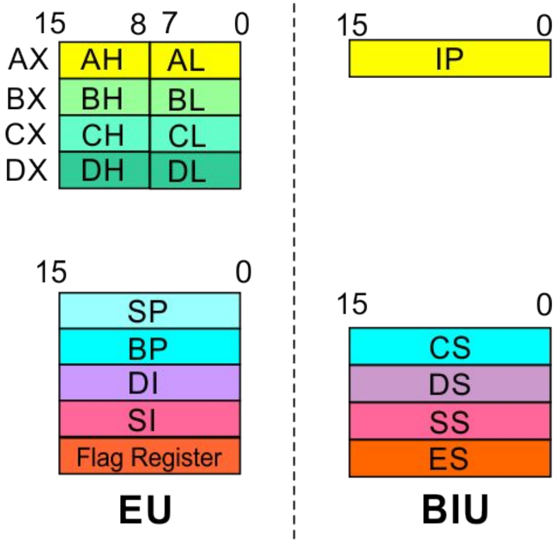
Architecture

Bus Interface Unit (BIU)

Segment Registers

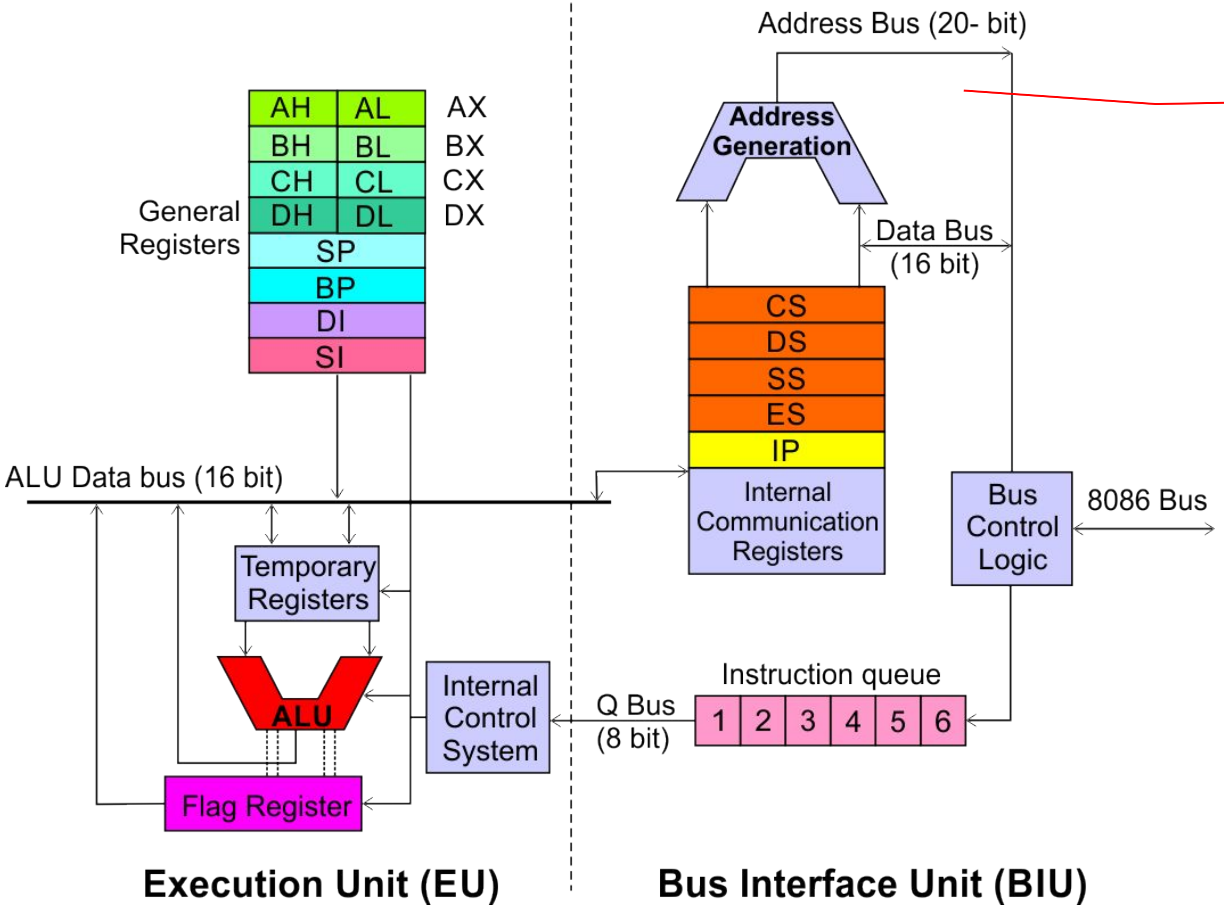
Instruction Pointer

- **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.**



Architecture

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>.**

Architecture

Execution Unit (EU)

EU decodes and executes instructions.

A decoder in the EU control system translates instructions.

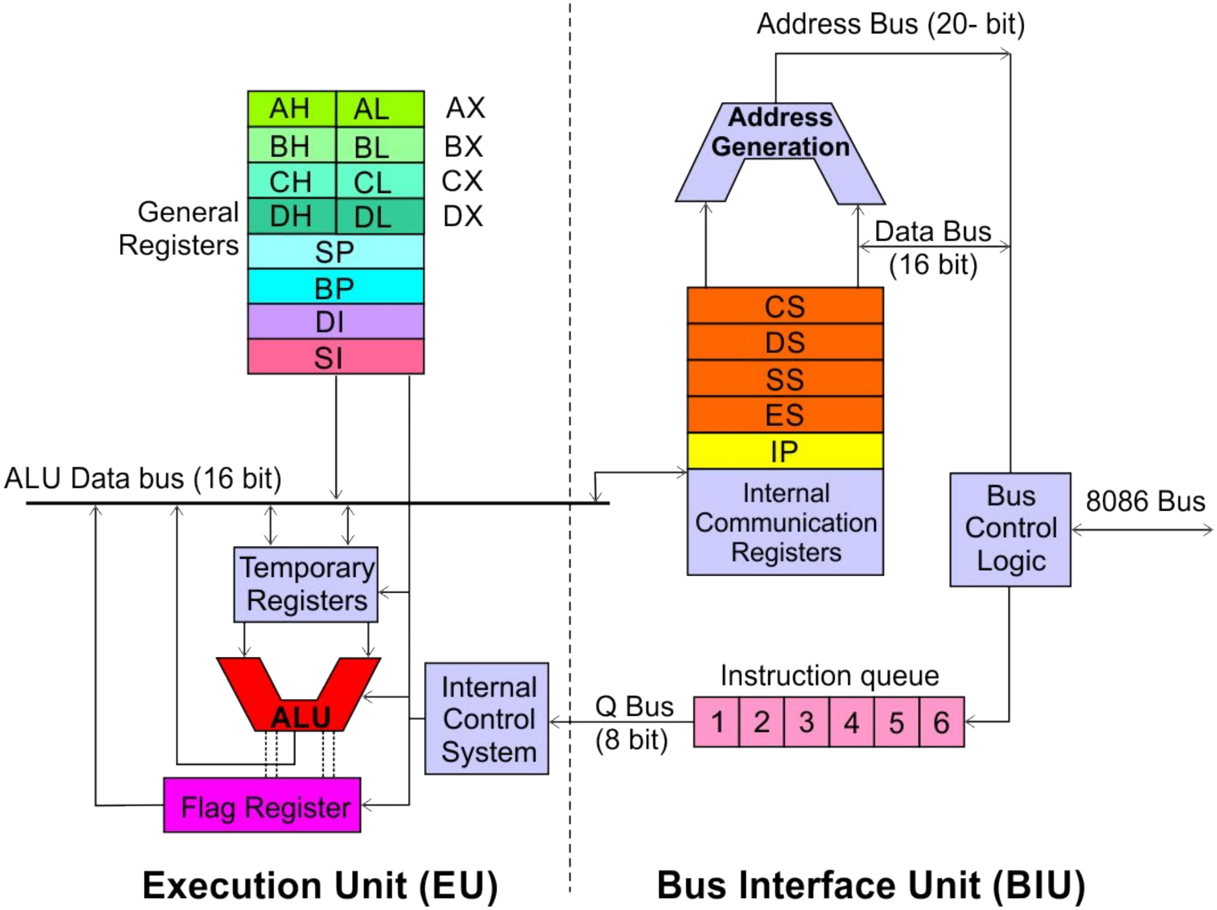
16-bit ALU for performing arithmetic and logic operation

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



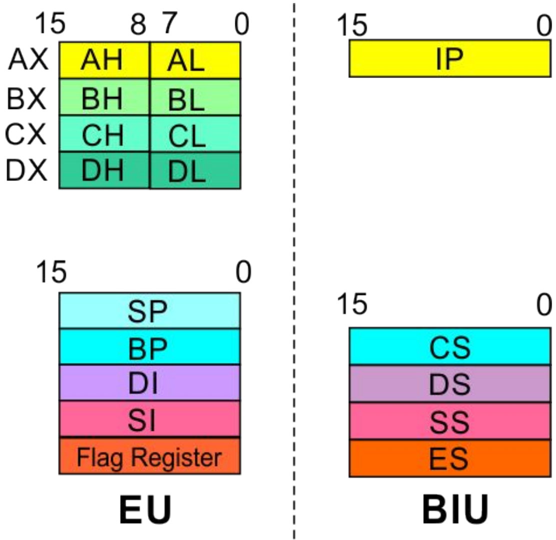
Some of the 16 bit registers can be used as two 8 bit registers as :

AX can be used as AH and AL
BX can be used as BH and BL
CX can be used as CH and CL
DX can be used as DH and DL

EU Registers

Accumulator Register (AX)

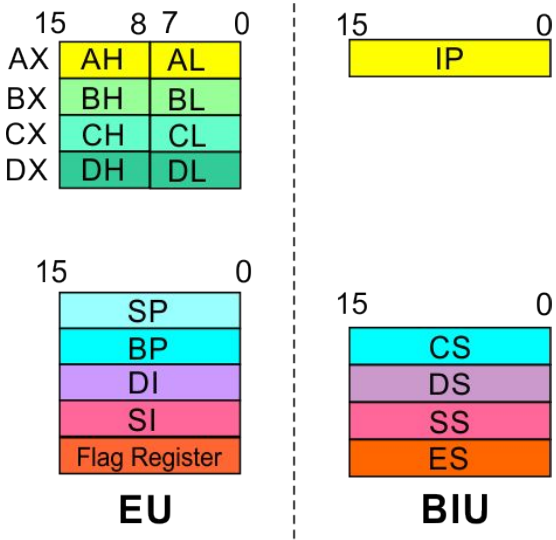
- Consists of two 8-bit registers AL and AH, which can be combined together and used as a 16-bit register AX.
- AL in this case contains the low order byte of the word, and AH contains the high-order byte.
- The I/O instructions use the AX or AL for inputting / outputting 16 or 8 bit data to or from an I/O port.
- Multiplication and Division instructions also use the AX or AL.



EU Registers

Base Register (BX)

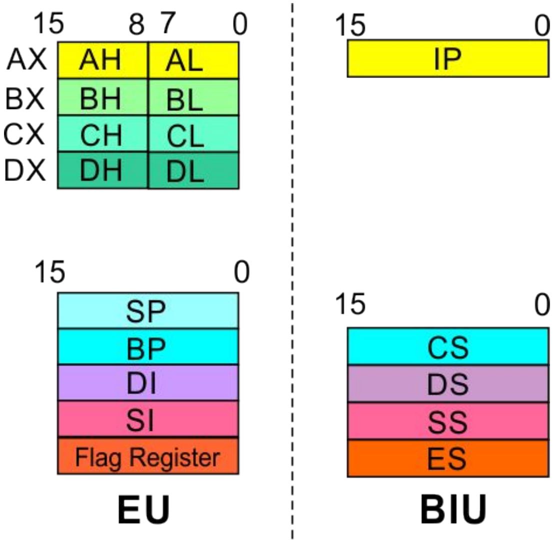
- Consists of two 8-bit registers BL and BH, which can be combined together and used as a 16-bit register BX.
- BL in this case contains the low-order byte of the word, and BH contains the high-order byte.
- This is the only general purpose register whose contents can be used for addressing the 8086 memory.
- All memory references utilizing this register content for addressing use DS as the default segment register.



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.

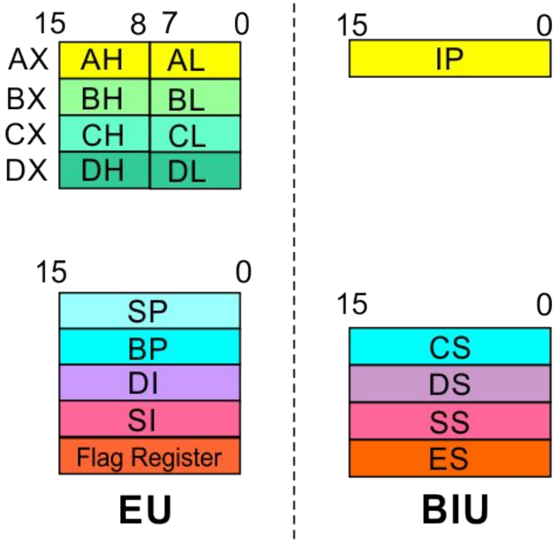
EU Registers

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.

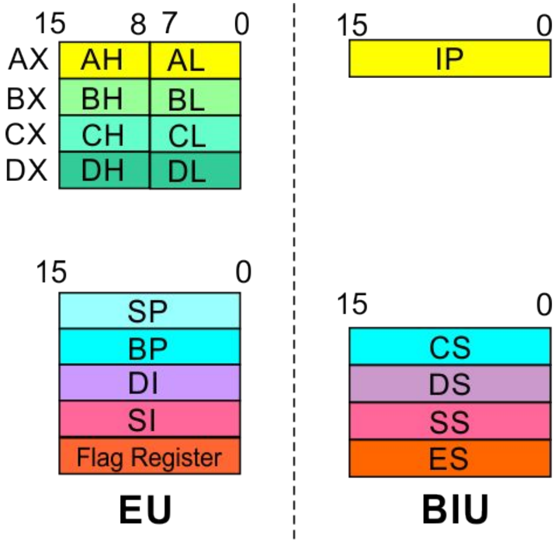
- Used to hold the high 16-bit result (data) in 16 X 16 multiplication or the high 16-bit dividend (data) before a 32 ÷ 16 division and the 16-bit reminder after division.



EU Registers

Stack Pointer (SP) and Base Pointer (BP)

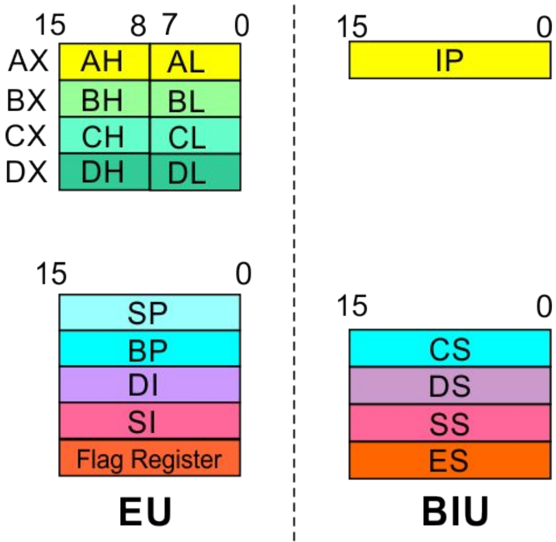
- SP and BP are used to access data in the stack segment.
- SP is used as an offset from the current SS during execution of instructions that involve the stack segment in the external memory.
- SP contents are automatically updated (incremented/ decremented) due to execution of a POP or PUSH instruction.
- BP contains an offset address in the current SS, which is used by instructions utilizing the based addressing mode.



EU Registers

Source Index (SI) and Destination Index (DI)

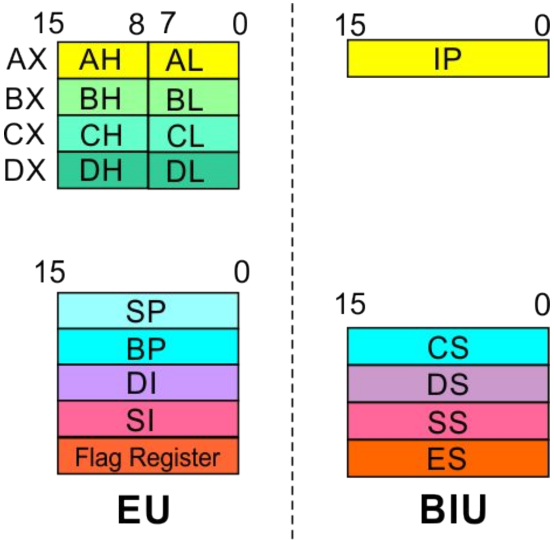
- Used in indexed addressing.
- Instructions that process data strings use the SI and DI registers together with DS and ES respectively in order to distinguish between the source and destination addresses.



EU Registers

Source Index (SI) and Destination Index (DI)

- **Used in indexed addressing.**
- **Instructions that process data strings use the SI and DI registers together with DS and ES respectively in order to distinguish between the source and destination addresses.**



Architecture

Execution Unit (EU)

Flag Register

Auxiliary Carry Flag

This is set, if there is a carry from the lowest nibble, i.e, bit three during addition, or borrow for the lowest nibble, i.e, bit three, during subtraction.

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.

Sign Flag

This flag is set, when the result of any computation is negative

Zero Flag

This flag is set, if the result of the computation or comparison performed by an instruction is zero

Parity Flag

This flag is set to 1, if the lower byte of the result contains even number of 1's ; for odd number of 1's set to zero.

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

Over flow Flag

This flag is set, if an overflow occurs, i.e, if the result of a signed operation is large enough to accommodate in a destination register. The result is of more than 7-bits in size in case of 8-bit signed operation and more than 15-bits in size in case of 16-bit sign operations, then the overflow will be set.

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

Direction Flag

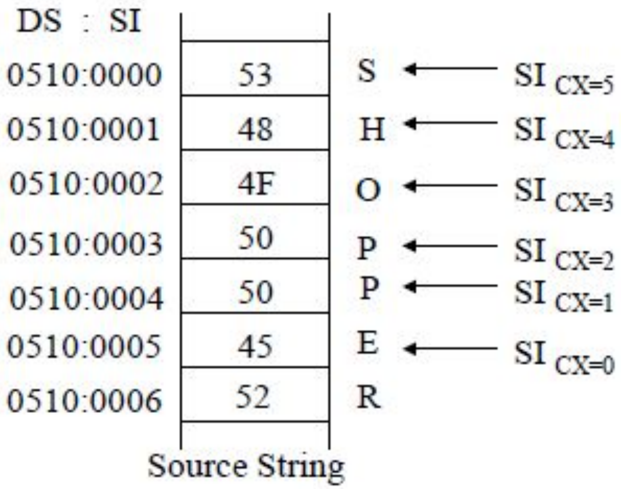
This is used by string manipulation instructions. If this flag bit is '0', the string is processed beginning from the lowest address to the highest address, i.e., auto incrementing mode. Otherwise, the string is processed from the highest address towards the lowest address, i.e., auto decrementing mode.

Interrupt Flag

Causes the 8086 to recognize external mask interrupts; setting IF enables these interrupts and clearing IF disables these interrupts.

```
CLD
MOV CX, 5
REP MOVSB
```

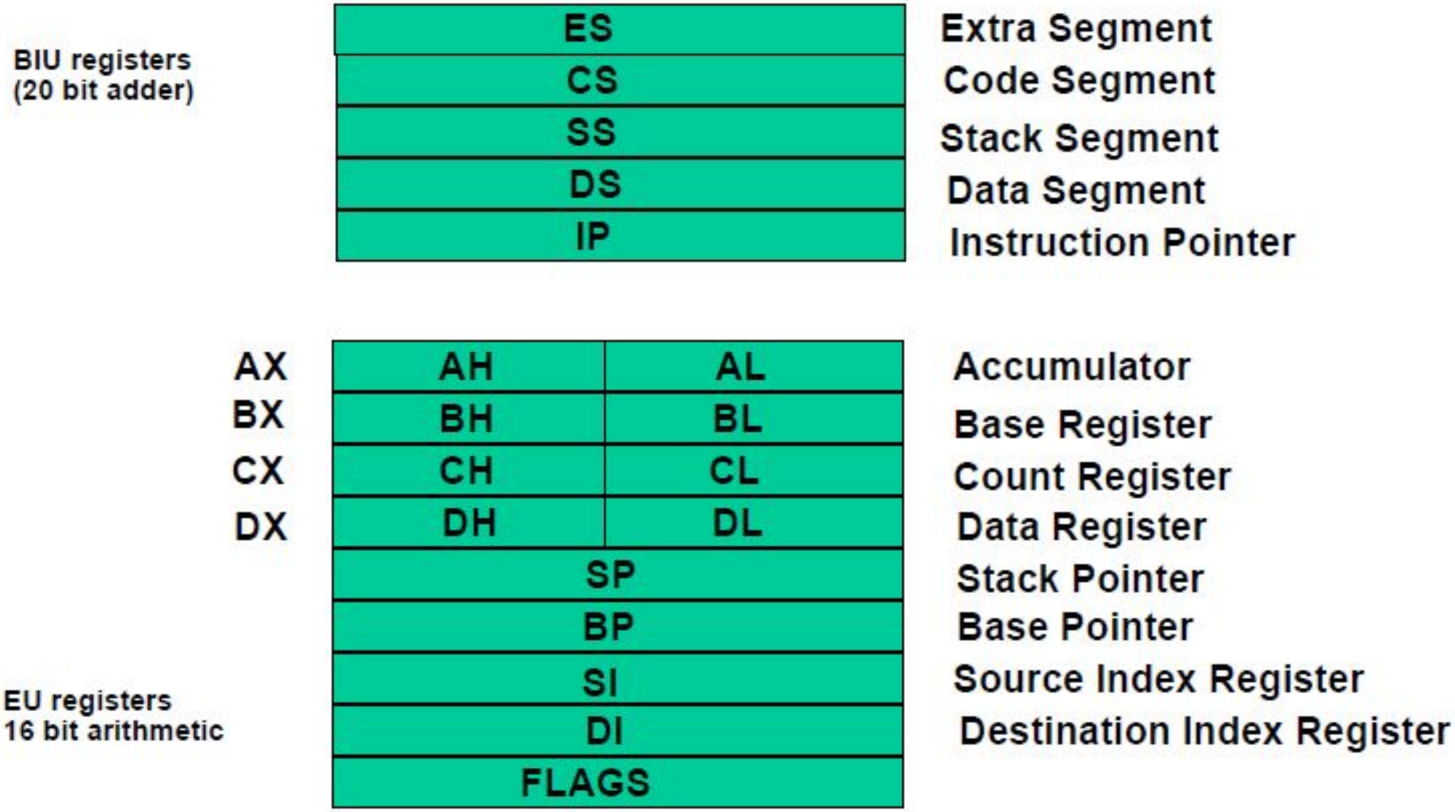
At the beginning of execution,
DS=0510H and SI=0000H



DS : SI			
0510:0000	53	S	SI $_{CX=5}$
0510:0001	48	H	SI $_{CX=4}$
0510:0002	4F	O	SI $_{CX=3}$
0510:0003	50	P	SI $_{CX=2}$
0510:0004	50	P	SI $_{CX=1}$
0510:0005	45	E	SI $_{CX=0}$
0510:0006	52	R	

Source String

8086 Programmer’s Model



BIU registers
(20 bit adder)

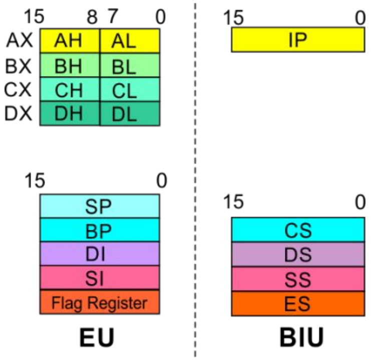
ES	Extra Segment
CS	Code Segment
SS	Stack Segment
DS	Data Segment
IP	Instruction Pointer

AX	AH	AL	Accumulator
BX	BH	BL	Base Register
CX	CH	CL	Count Register
DX	DH	DL	Data Register
	SP		Stack Pointer
	BP		Base Pointer
	SI		Source Index Register
	DI		Destination Index Register
	FLAGS		

EU registers
16 bit arithmetic

Architecture

8086 registers categorized into 4 groups



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

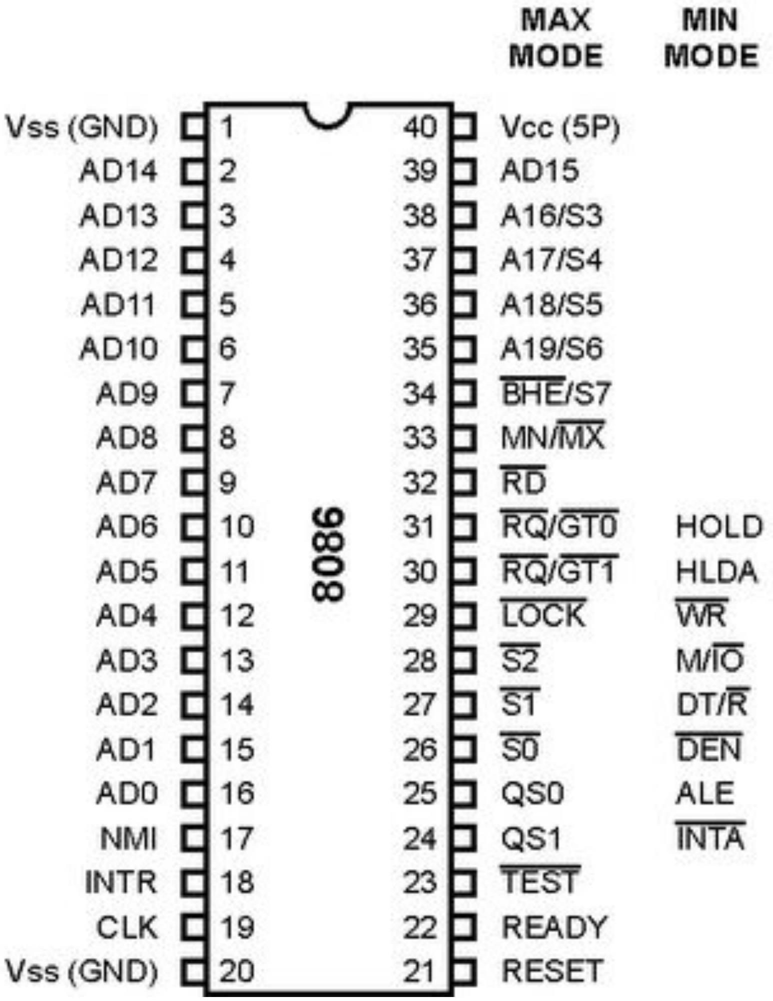
Sl.No.	Type	Register width	Name of register
1	General purpose register	16 bit	AX, BX, CX, DX
		8 bit	AL, AH, BL, BH, CL, CH, DL, DH
2	Pointer register	16 bit	SP, BP
3	Index register	16 bit	SI, DI
4	Instruction Pointer	16 bit	IP
5	Segment register	16 bit	CS, DS, SS, ES
6	Flag (PSW)	16 bit	Flag register

Registers and Special Functions

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

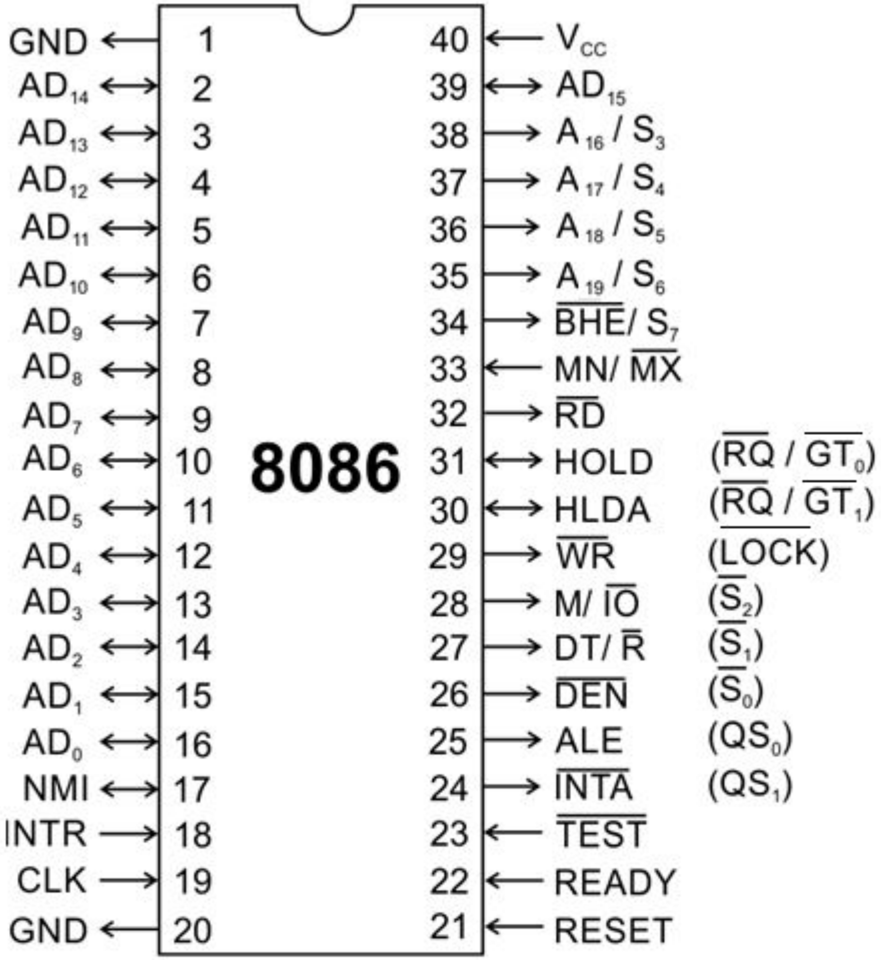
Pins and signals

PIN DIAGRAM of 8086 Microprocessor



Pins and Signals

Common signals



AD_0-AD_{15} (Bidirectional)

Address/Data bus

Low order address bus; these are multiplexed with data.

When AD lines are used to transmit memory address the symbol A is used instead of AD, for example A_0-A_{15}.

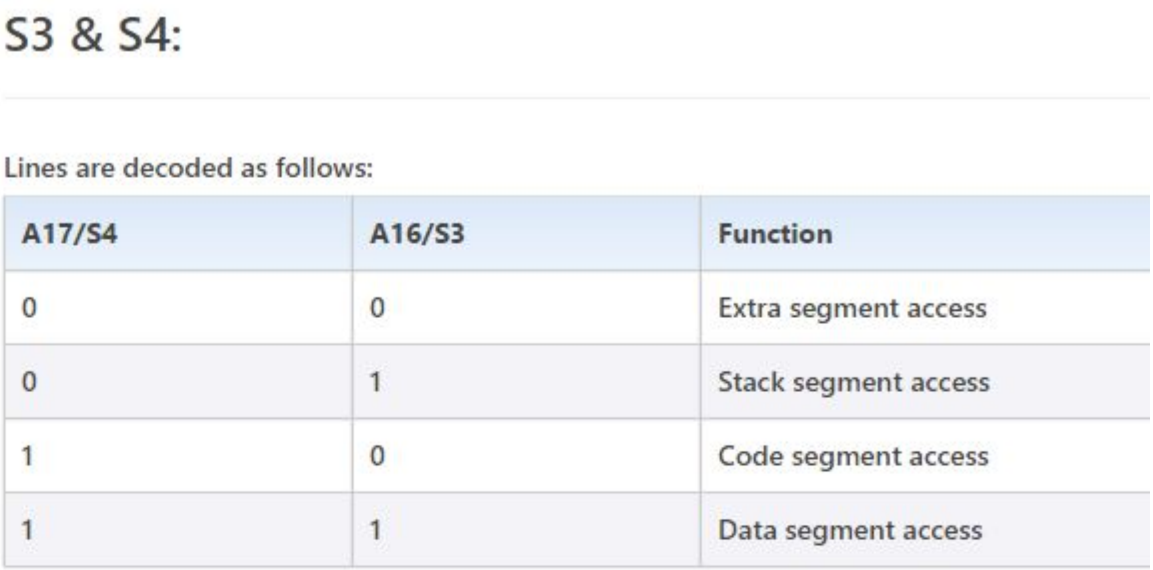
When data are transmitted over AD lines the symbol D is used in place of AD, for example D_0-D_7, D_8-D_{15} or D_0-D_{15}.

A_{16}/S_3, A_{17}/S_4, A_{18}/S_5, A_{19}/S_6

High order address bus. These are multiplexed with status signals

QS1,QS0

After the first clock cycle of an instruction execution, the A17/S4 and A16/S3 pins specify which segment register generates the segment portion of the 8086 address.



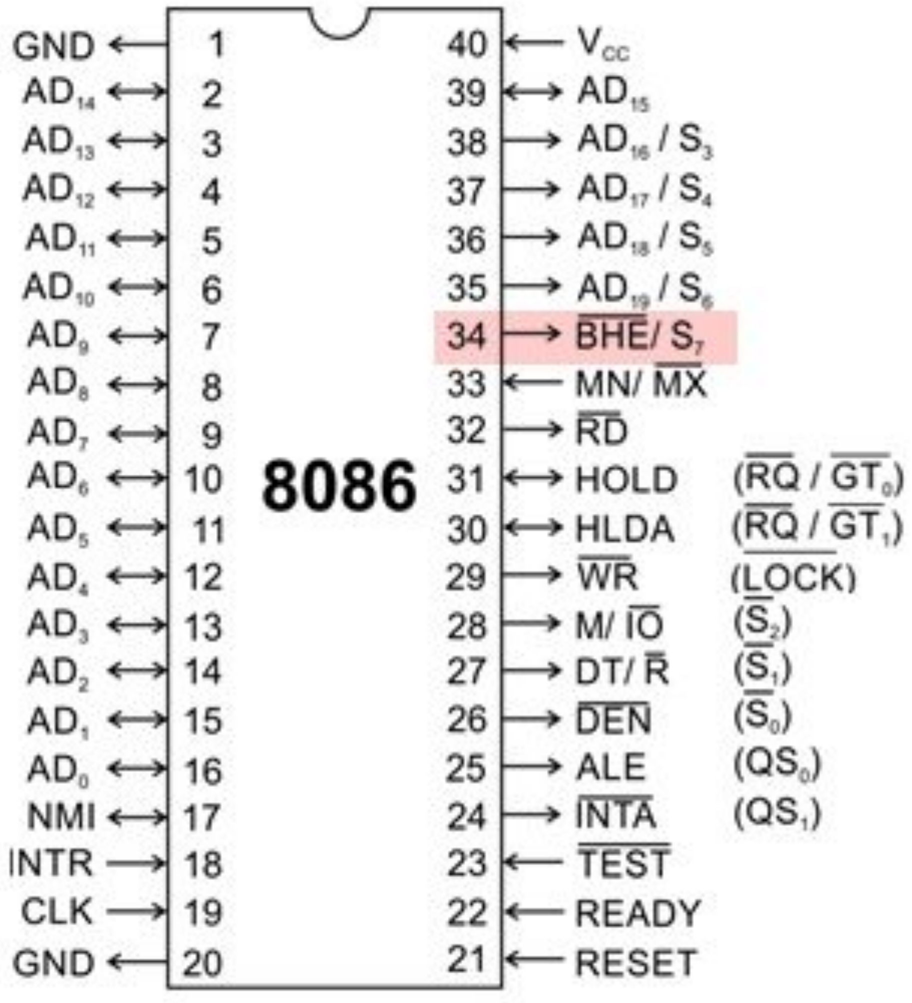
S3 & S4:

Lines are decoded as follows:

A17/S4	A16/S3	Function
0	0	Extra segment access
0	1	Stack segment access
1	0	Code segment access
1	1	Data segment access

Pins and Signals

Common signals



BHE’(Active Low)/S_7 (Output)

Bus High Enable/Status

It is used to enable data onto the most significant half of data bus, D_8-D_{15}. 8-bit device connected to upper half of the data bus use BHE (Active Low) signal. It is multiplexed with status signal S_7.

MN/ MX’

MINIMUM / MAXIMUM

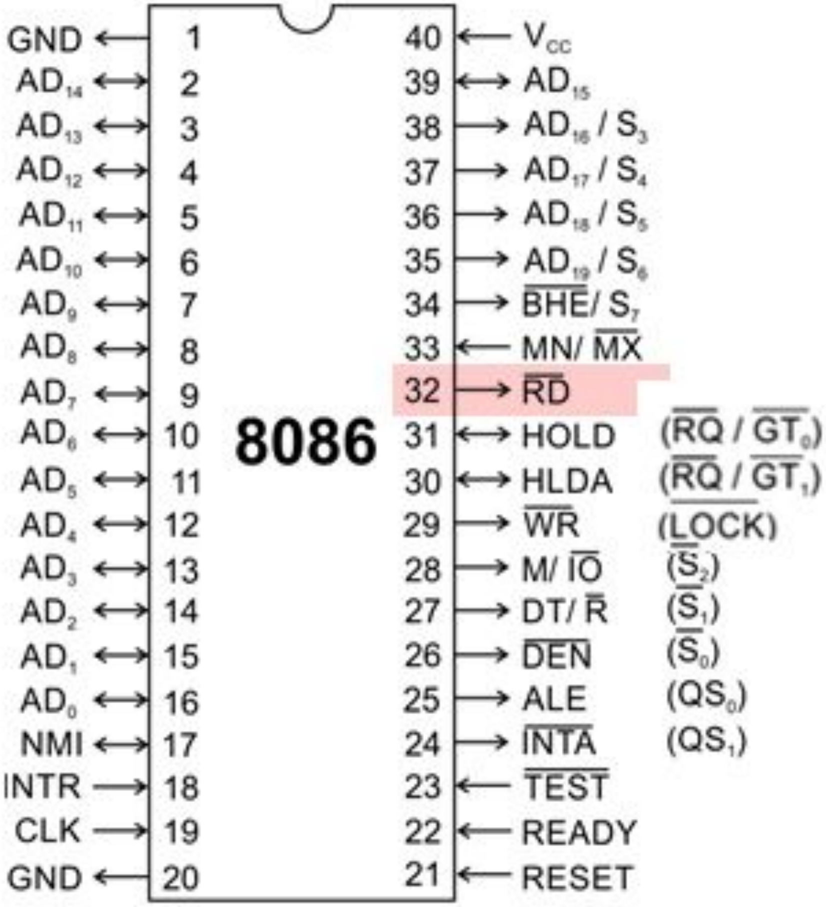
This pin signal indicates what mode the processor is to operate in.

RD’(Read) (Active Low)

The signal is used for read operation.
It is an output signal.
It is active when low.

Pins and Signals

Common signals



BHE’(Active Low)/S_7 (Output)

Bus High Enable/Status

It is used to enable data onto the most significant half of data bus, D_8-D_{15}. 8-bit device connected to upper half of the data bus use BHE (Active Low) signal. It is multiplexed with status signal S_7.

MN/ MX’

MINIMUM / MAXIMUM

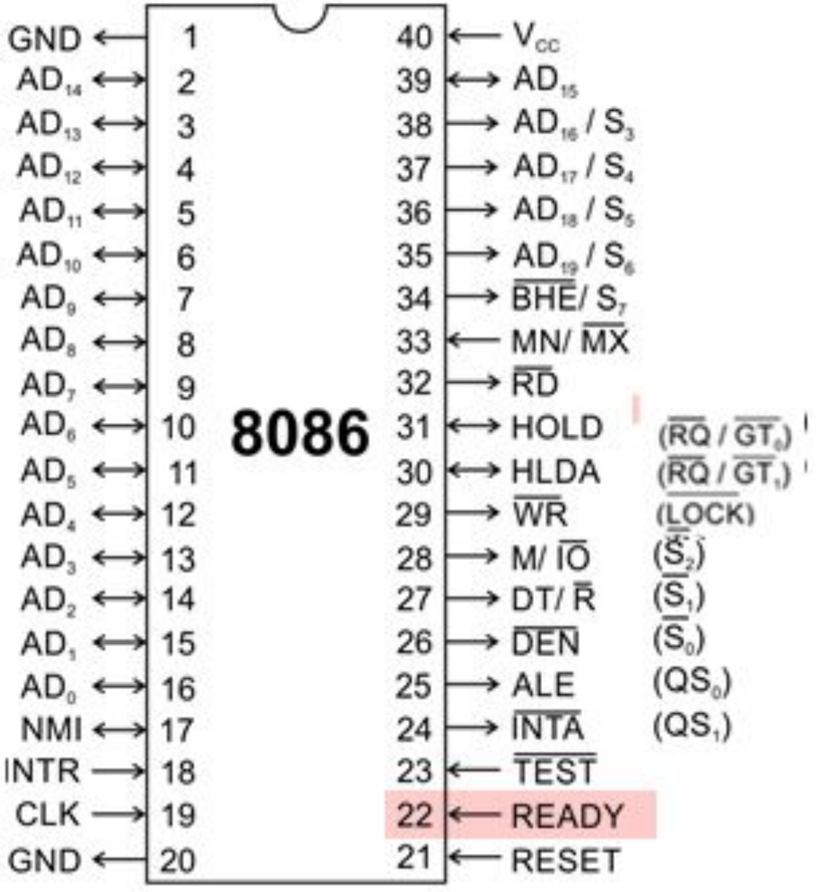
This pin signal indicates what mode the processor is to operate in.

RD’(Read) (Active Low)

The signal is used for read operation.
It is an output signal.
It is active when low.

Pins and Signals

Common signals



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.
- Used to hold the high 16-bit result (data) in 16 X 16 multiplication or the high 16-bit dividend (data) before a 32 ÷ 16 division and the 16-bit reminder after division.

READY

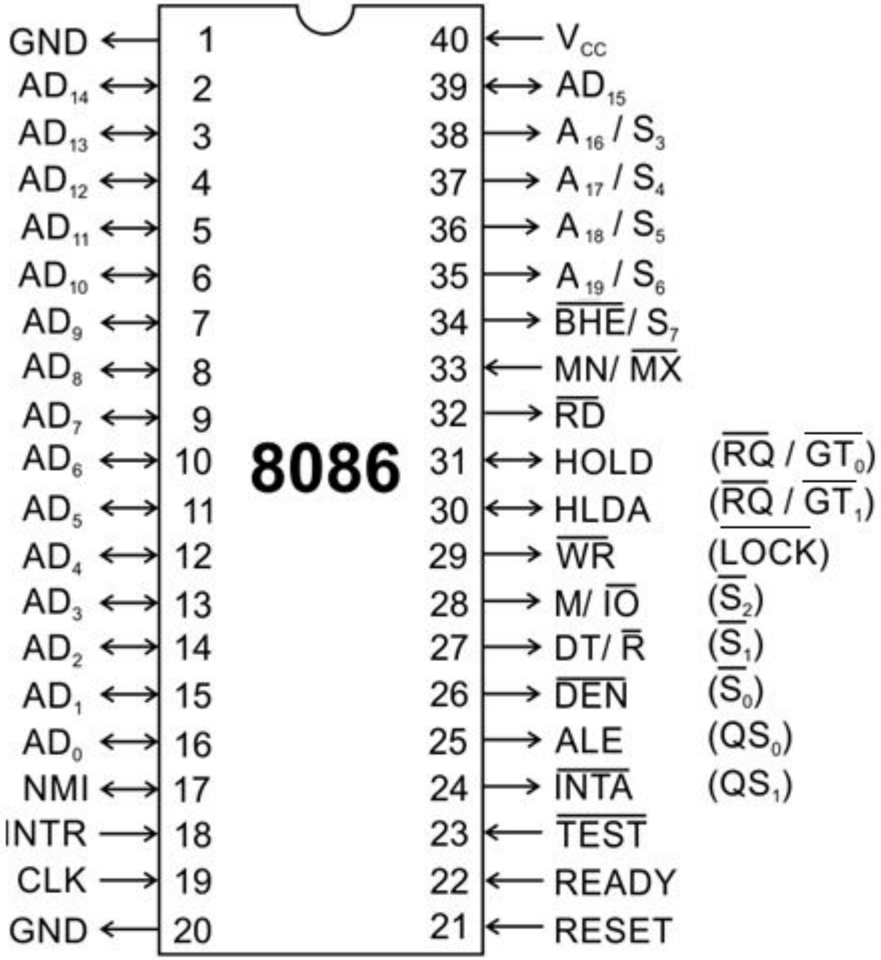
This is the acknowledgement from the slow device or memory that they have completed the data transfer.

The signal made available by the devices is synchronized by the 8284A clock generator to provide ready input to the 8086.

The signal is active high.

Pins and Signals

Common signals



RESET (Input)

Causes the processor to immediately terminate its present activity.

The signal must be active HIGH for at least four clock cycles.

CLK

The clock input provides the basic timing for processor operation and bus control activity. Its an asymmetric square wave with 33% duty cycle.

INTR Interrupt Request

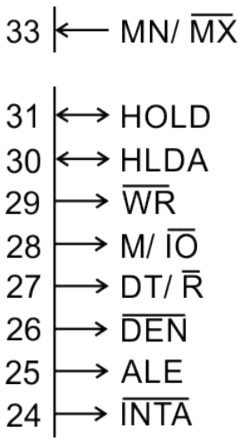
This is a triggered input. This is sampled during the last clock cycles of each instruction to determine the availability of the request. If any interrupt request is pending, the processor enters the interrupt acknowledge cycle.

This signal is active high and internally synchronized.

Pins and Signals for

Min and Max Operating Modes

The 8086 microprocessor can work in two modes of operations : Minimum mode and Maximum mode.



In the minimum mode of operation the microprocessor do not associate with any co-processors and can not be used for multiprocessor systems.

In the maximum mode the 8086 can work in multi-processor or co-processor configuration.

Minimum or maximum mode operations are decided by the pin MN/ MX'(Active low).

When this pin is high 8086 operates in minimum mode otherwise it operates in Maximum mode.

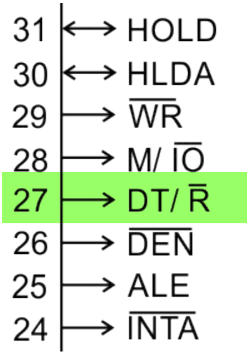
Pins and Signals

Minimum mode signals

Data Register (DX)

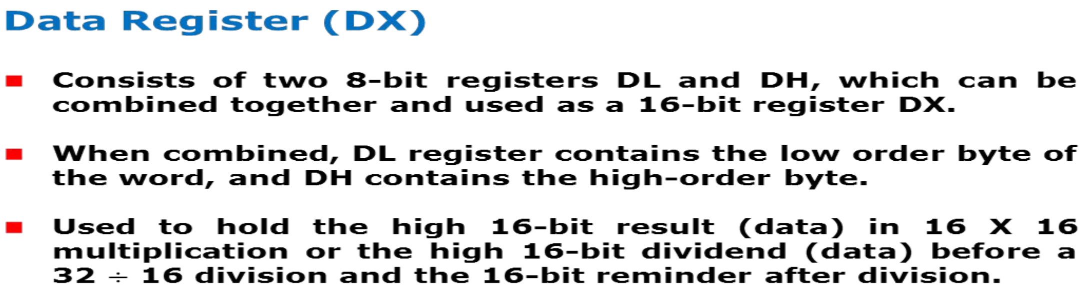
- Consists of two 8-bit registers DL and DH, which can be combined together and used as a 16-bit register DX.
- When combined, DL register contains the low order byte of the word, and DH contains the high-order byte.
- Used to hold the high 16-bit result (data) in 16 X 16 multiplication or the high 16-bit dividend (data) before a 32 ÷ 16 division and the 16-bit reminder after division.

$DT/\overline{R}$	(**Data Transmit/ Receive**) Output signal from the processor to control the direction of data flow through the data transceivers
$\overline{DEN}$	(**Data Enable**) Output signal from the processor used as output enable for the transceivers
ALE	(**Address Latch Enable**) Used to demultiplex the address and data lines using external latches
$M/\overline{IO}$	Used to differentiate memory access and I/O access. For memory reference instructions, it is **high**. For IN and OUT instructions, it is **low**.
$\overline{WR}$	Write control signal; asserted **low** Whenever processor writes data to memory or I/O port
$\overline{INTA}$	(**Interrupt Acknowledge**) When the interrupt request is accepted by the processor, the output is **low** on this line.



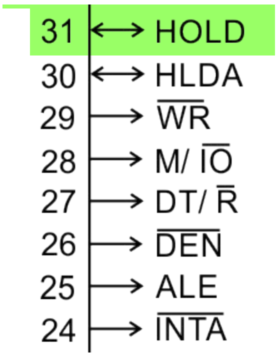
Pins and Signals

Minimum mode signals



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.
- Used to hold the high 16-bit result (data) in 16 X 16 multiplication or the high 16-bit dividend (data) before a 32 ÷ 16 division and the 16-bit reminder after division.


31 HOLD
30 HLDA
29 $\overline{\text{WR}}$
28 M/ $\overline{\text{IO}}$
27 DT/ $\overline{\text{R}}$
26 $\overline{\text{DEN}}$
25 ALE
24 $\overline{\text{INTA}}$

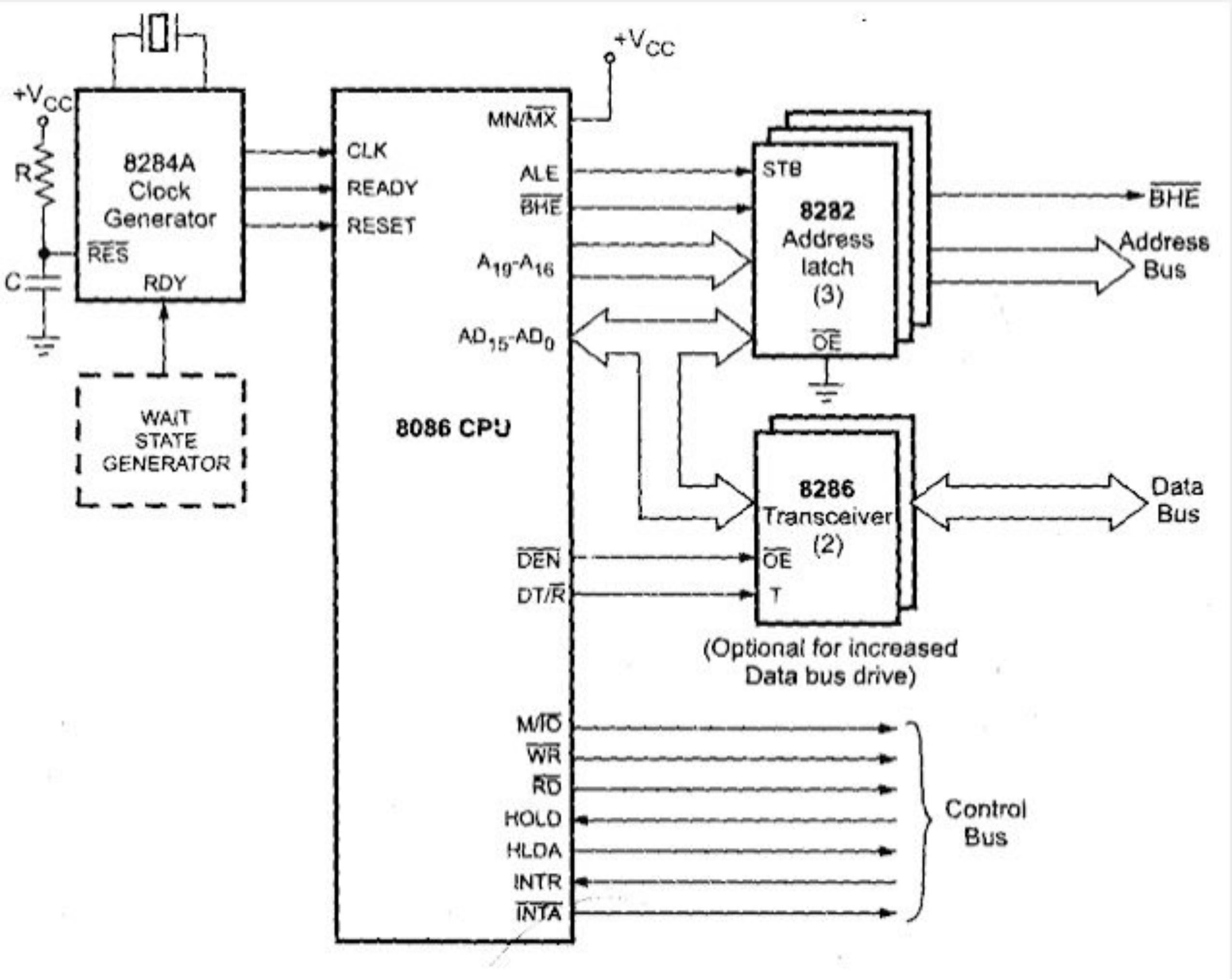
HOLD **Input signal to the processor form the bus masters as a request to grant the control of the bus.**

Usually used by the DMA controller to get the control of the bus.

HLDA **(Hold Acknowledge) Acknowledge signal by the processor to the bus master which requested the control of the bus through HOLD.**

The acknowledge is asserted high, when the processor accepts HOLD.

For Reference Only(Minimum Mode Configuration)



Minimum mode of 8086

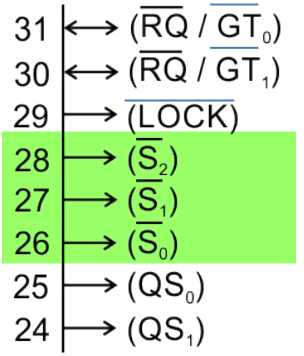
Pins and Signals

Maximum mode signals

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.
- Used to hold the high 16-bit result (data) in 16 X 16 multiplication or the high 16-bit dividend (data) before a 32 ÷ 16 division and the 16-bit reminder after division.

$\overline{S_0}, \overline{S_1}, \overline{S_2}$	**Status signals; used by the 8086 bus controller to generate bus timing and control signals. These are decoded as shown.**



Status Signal			Machine Cycle
$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	
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	Code access
1	0	1	Read memory
1	1	0	Write memory
1	1	1	Passive/Inactive

Pins and Signals

Maximum mode signals

Data Register (DX)

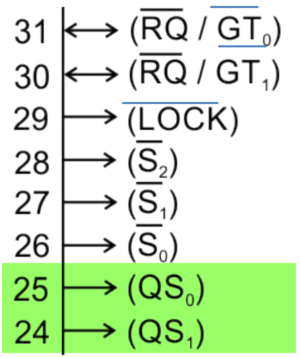
- Consists of two 8-bit registers DL and DH, which can be combined together and used as a 16-bit register DX.
- When combined, DL register contains the low order byte of the word, and DH contains the high-order byte.
- Used to hold the high 16-bit result (data) in 16 X 16 multiplication or the high 16-bit dividend (data) before a 32 ÷ 16 division and the 16-bit reminder after division.

QS1,QS0

(Queue Status) The processor provides the status of queue in these lines.

The queue status can be used by external device to track the internal status of the queue in 8086.

The output on QS_0 and QS_1 can be interpreted as shown in the table.



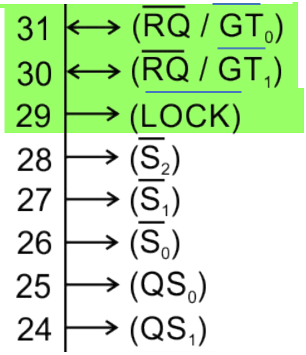
Queue status		Queue operation
QS_1	QS_0	
0	0	No operation
0	1	First byte of an opcode from queue
1	0	Empty the queue
1	1	Subsequent byte from queue

Pins and Signals

Maximum mode signals

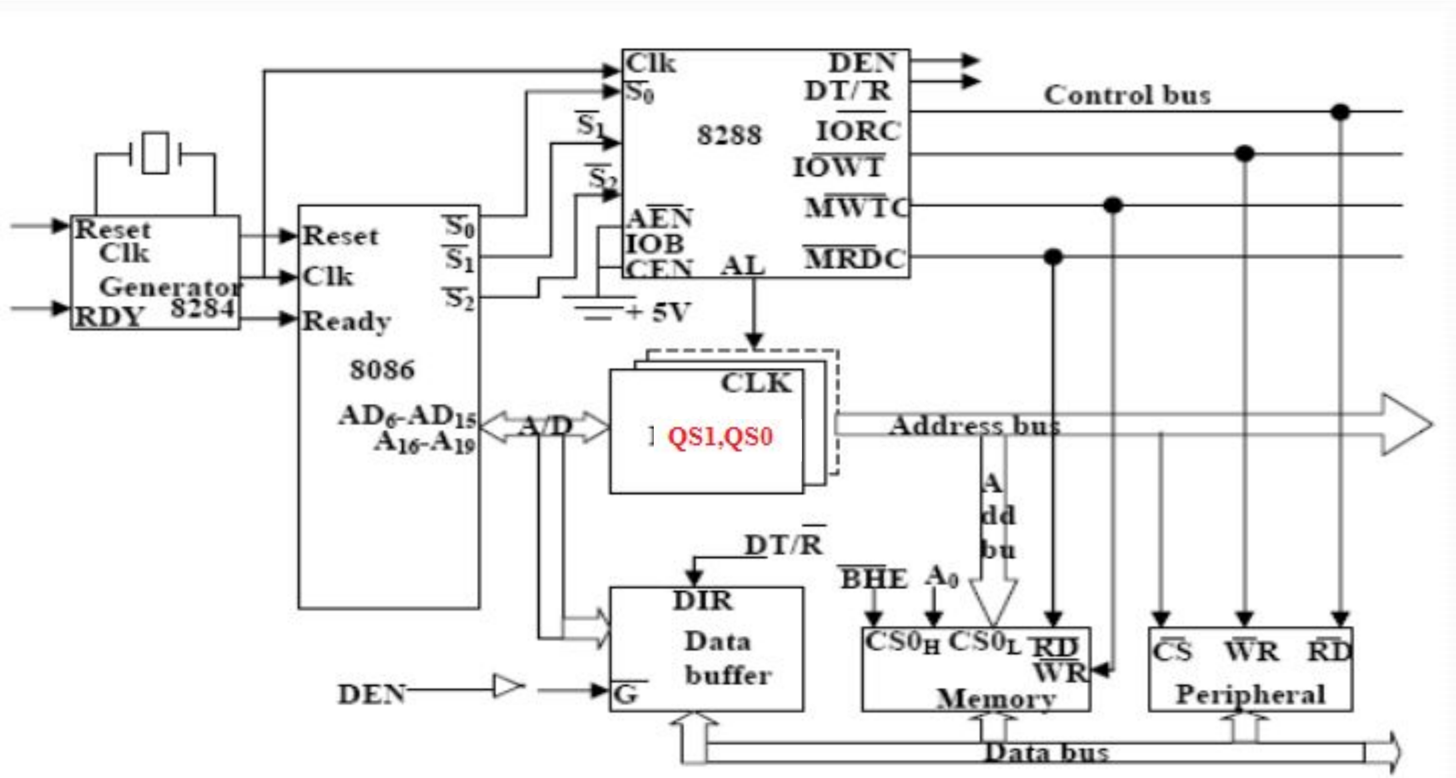
Data Register (DX)

- 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.
- Used to hold the high 16-bit result (data) in 16 X 16 multiplication or the high 16-bit dividend (data) before a 32 ÷ 16 division and the 16-bit reminder after division.



$\overline{RQ}/\overline{GT_0}$, $\overline{RQ}/\overline{GT_1}$	(Bus Request/ Bus Grant) These requests are used by other local bus masters to force the processor to release the local bus at the end of the processor's current bus cycle. These pins are bidirectional. The request on $\overline{GT_0}$ will have higher priority than $\overline{GT_1}$
$\overline{LOCK}$	An output signal activated by the LOCK prefix instruction. Remains active until the completion of the instruction prefixed by LOCK. The 8086 output low on the $\overline{LOCK}$ pin while executing an instruction prefixed by LOCK to prevent other bus masters from gaining control of the system bus.

For Reference Only (Maximum Mode Configuration)



Maximum Mode 8086 System.

Memory Addressing by 8086 (Physical Memory Organisation)

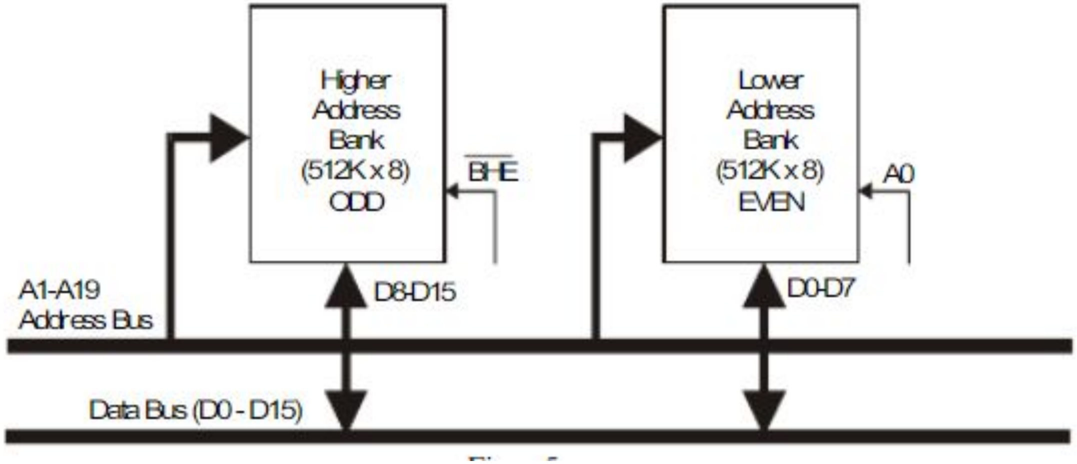

Higher Address Bank (512K x 8) ODD
BHE
Lower Address Bank (512K x 8) EVEN
A0
A1-A19 Address Bus
D8-D15
D0-D7
Data Bus (D0 - D15)

Fig 1: Memory Addressing

Most of the memory ICs are byte oriented i.e. each memory location can store only one byte of data. The 8086 is a 16-bit microprocessor, it can transfer 16-bit data. So in addition to byte, word (16-bit) has to be stored in the memory. This is stored by using two consecutive memory locations, one for least significant byte and other for most significant byte.

Physical Memory Organisation

The entire memory is divided into two memory banks : bank0 and bank1. Fig.1 shows the interfacing diagram to these memory banks.

- Bank0 is selected only when A_0 is zero and
- Bank1 is selected only when BHE’ is zero.
- A_0 is zero for all even addresses. So Bank0 is usually referred as **even addressed memory** bank.
- BHE’ is used to access higher order memory bank, referred to as **odd addressed memory** bank.
- The address space is physically connected to a 16-bit data bus by dividing the address space into two 8-bit banks of upto 512K bytes each.

- One bank is connected to the lower half of the 16-bit data bus (D0 – D7) and contains even address bytes. *i.e.* when A0 bit is low, the bank is selected.

- The other bank is connected to the upper half of the data bus (D8-D15) and contains odd address bytes. *i.e.* when A0 is high and BHE’ (Bus High Enable)is low, the odd bank is selected. A specific byte within each bank is selected by address lines A1-A19.

Physical Memory Organisation

Data can be accessed from the memory in four different ways. They are:

- 8-bit data from Lower (Even) address Bank.
- 8-bit data from Higher (Odd) address Bank.
- 16-bit data starting from Even Address.
- 16-bit data starting from Odd Address.

·

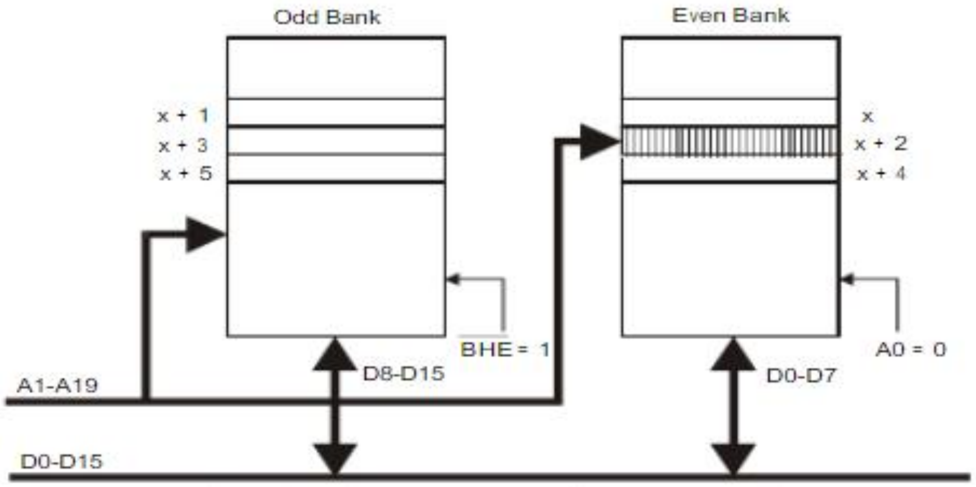
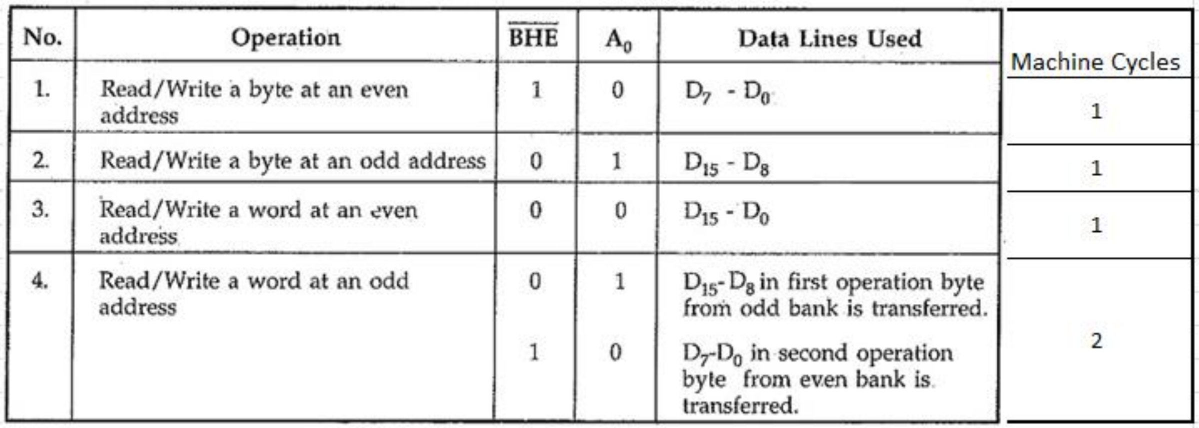


Fig 2: 8-Bit data Access from Even Ba



No.	Operation	$\overline{BHE}$	A_0	Data Lines Used	Machine Cycles
1.	Read/Write a byte at an even address	1	0	D_7 - D_0	1
2.	Read/Write a byte at an odd address	0	1	D_{15} - D_8	1
3.	Read/Write a word at an even address	0	0	D_{15} - D_0	1
4.	Read/Write a word at an odd address	0 1	1 0	D_{15}-D_8 in first operation byte from odd bank is transferred. D_7-D_0 in second operation byte from even bank is transferred.	2

Physical Memory Organisation

8-bit data access from Even Address Bank takes one machine cycle

To access memory bytes from Even address, information is transferred over the lower half of the data bus(D0 - D7) A 0 is output LOW and BHE’ is output HIGH enabling only the even address bank. Fig 2 shown for this.

8-bit Data access from Odd Address Bank takes one machine cycle

To access memory byte from an odd address, information is transferred over the higher half of the data bus (D8-D15). The BHE’ output low enables the upper memory bank. A0 is output high to disable the lower memory bank.

16-bit data starting from an even address takes one machine cycle

- 16-bit data from an even address is accessed in a single bus cycle. Address lines A1-A19 select the appropriate byte within each bank. A0 low and BHE’ low enables both banks simultaneously.

16-bit Data Access starting from Odd Address takes two machine cycles

- A 16-bits word located at an odd address (two consecutive bytes with the least significant byte at an odd byte address) is accessed using two bus cycles.

- During the first bus cycle the lower byte is accessed. A1-A19 address bus specifies the address and A0 as1 and BHE is low. Therefore the even memory bank is disabled and odd memory bank is enabled.
- During the second bus cycle, the upper byte (with the even address 0006H is accessed, the address is incremented. Therefore A0 is zero and BHE is made high. The even memory bank is enabled and the odd memory bank is disabled.