UNIT-I
8086 Architecture

Introduction to Microprocessors

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.

Generation of Microprocessors:

- **INTEL 4004 (1971)**
  - 4-bit microprocessor
  - 4 KB main memory
  - 45 instructions
  - PMOS technology
  - was first programmable device which was used in calculators
- **INTEL 8008 (1972)**
  - 8-bit version of 4004
  - 16 KB main memory
  - 48 instructions
  - PMOS technology
  - Slow
- **Intel 8080** (1973)
  - 8-bit microprocessor
  - 64 KB main memory
  - 2 microseconds clock cycle time
  - 500,000 instructions/sec
  - 10X faster than 8008
  - NMOS technology
  - Drawback was that it needed three power supplies.

- Small computers (Microcomputers) were designed in mid 1970's
  Using 8080 as CPU.

- **<u>INTEL 8086/8088</u>**

  Year of introduction 1978 for 8086 and 1979 for 8088
  - 16-bit microprocessors
  - Data bus width of 8086 is 16 bit and 8 bit for 8088
  - 1 MB main memory
  - 400 nanoseconds clock cycle time
  - 6 byte instruction cache for 8086 and 4 byte for 8088
  - Other improvements included more registers and additional instructions
  - In 1981 IBM decided to use 8088 in its personal computer

- **<u>INTEL 80186</u>** (1982)

  - 16-bit microprocessor-upgraded version of 8086
  - 1 MB main memory
  - Contained special hardware like programmable counters, interrupt controller etc.
  - Never used in the PC
  - But was ideal for systems that required a minimum of hardware .
- **<u>INTEL 80286</u>** (1983)
  - 16-bit high performance microprocessor with memory management & protection
  - 16 MB main memory
  - Few additional instructions to handle extra 15 MB
  - Instruction execution time is as little as 250 ns
  - Concentrates on the features needed to implement MULTITASKING

- **Intel 80386** (1986)
- **Intel 80486** (1989)
- **Pentium** (1993)
- **Pentium pro**(1995)
- **Pentium ii** (1997)
- **Pentium iii** (1999)
- **Pentium iv** (2002)
- **Latest is Intel i9 processor**

General Architecture of Microprocessors

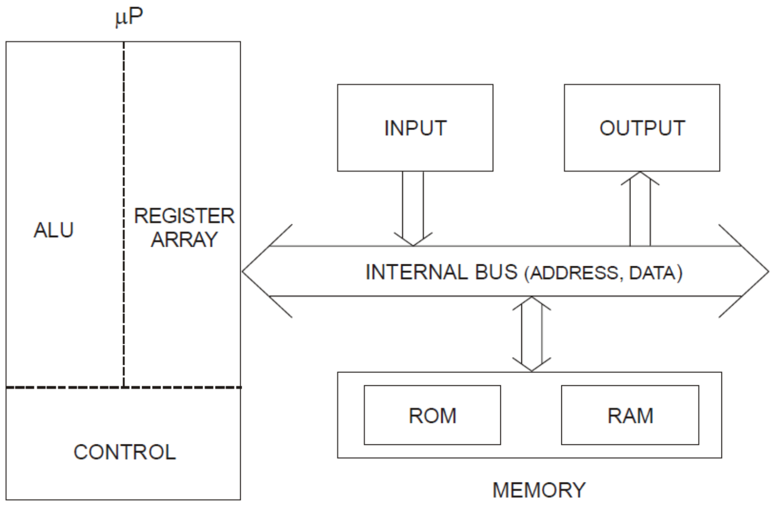


Figure 1.2 Architecture of Microprocessor

Buses

Figure shows busses interconnecting various blocks. These busses allow exchange of words between the blocks. A bus has a wire or line for each bit and thus allows exchange of all bits of a word in parallel. The processing of bits in the µP is also in parallel. The busses can thus be viewed as data highways. The width of a bus is the number of signal lines that constitute the bus.

Arithmetic-Logic Unit (ALU)

The arithmetic-logic unit is a combinational network that performs arithmetic and logical operations on the data.

Internal Registers

A number of registers are normally included in the microprocessor. These are used for temporary storage of data, instructions and addresses during execution of a program. Those in the Intel

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 1.4 shows register organization of 8086. We will categorize the register set into four groups as follows:

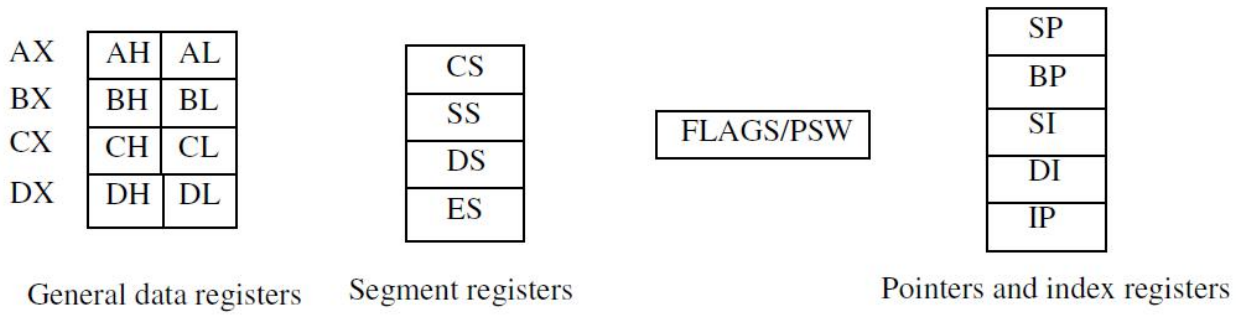


Fig.1.4 Register organization of 8086 Microprocessor

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.

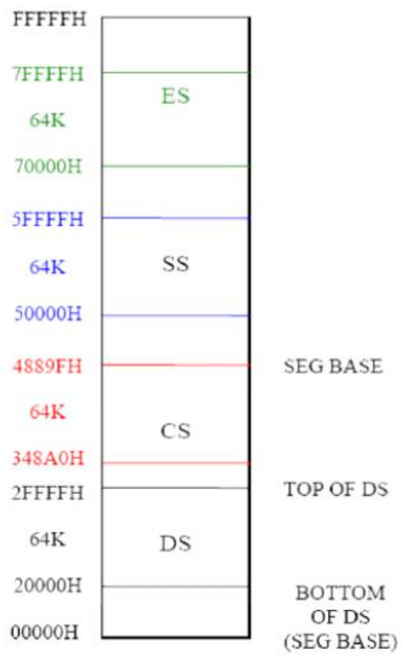


Fig1.5. Memory segmentation

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 Register:

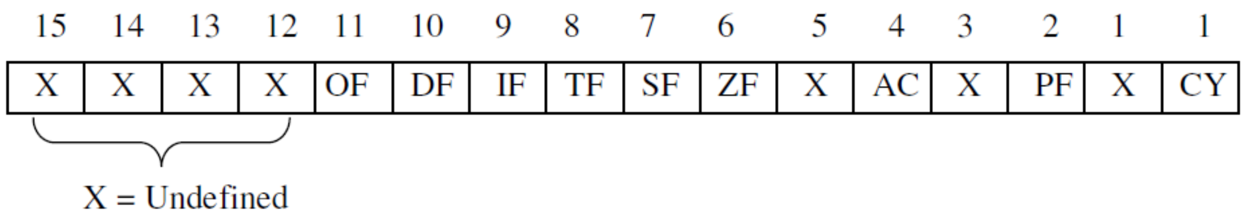


Fig1.6 . Flag Register of 8086

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 1.6. 8086 has 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

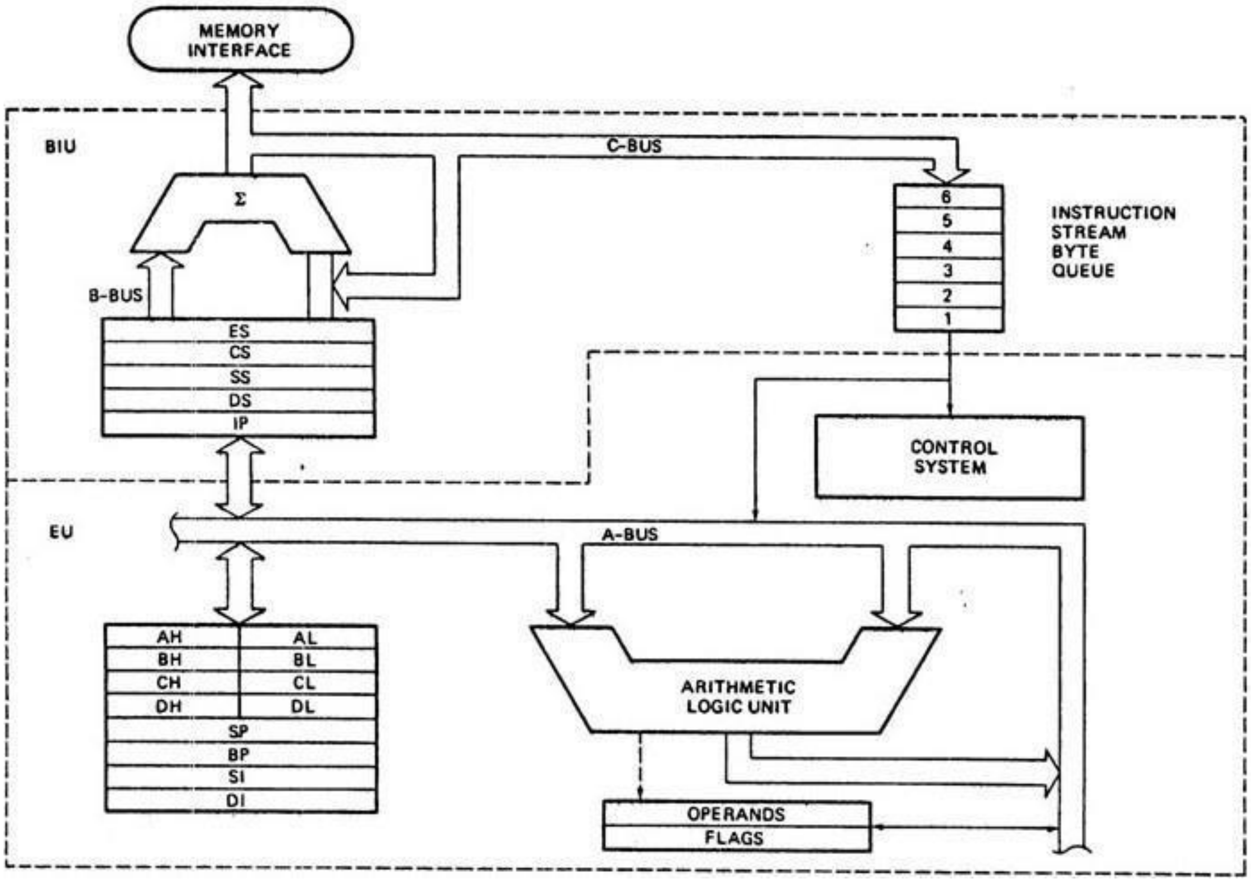
Control flags are set or reset deliberately to control the operations of the execution unit. Control flags are as follows:

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 speedup the processing

1) EXECUTION UNIT(EU)

The Execution unit tells the BIU where to fetch instructions or data from

- decodes instructions and
- Executes instructions

The Execution unit contains:

1) Control circuitry
2) ALU
3) FLAGS
4) General purpose Registers
5) Pointer and Index Registers

Control Circuitry:

- It directs internal operations.

- A decoder in the EU translates instructions fetched from memory Into series of actions which the EU carries out

Arithmetic Logic Unit:

16 bit ALU

Used to carry the operations

- ADD
- SUBTRACT
- XOR
- INCREMENT
- DECREMENT
- COMPLEMENT
- SHIFT BINARY NUMBERS

FLAG REGISTERS:

- A flag is a flip flop that indicates some condition produced by execution of an instruction or controls certain operation of the EU.
- It is 16 bit
- It has nine active flags

Divided into two types

1. Conditional flags
2. Control flags

Conditional Flags

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.

General Purpose Registers:
The 8086 general purpose registers are similar to those of earlier generations 8080 and 8085 .It was designed in such a way that many programs written for 8080 and 8085 could easily be translated to run on 8086.The advantage of using internal registers for the temporary storage of data is that since data already in the EU ., it can be accessed much more quickly than it could be accessed from external memory.
General Purpose 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.

2) BUS INTERFACE UNIT (BIU)

The BIU sends out

- Addresses
- Fetches instructions from memory
- Read data from ports and memory

Or

The BIU handles all transfer of data and addresses on the buses for the Execution Unit

The Bus interface unit contains

1) Instruction Queue
2) Instruction pointer
3) Segment registers
4) Address Generator

Instruction Queue:

BIU gets upto 6 bytes of next instructions and stores them in the instruction queue. When EU executes instructions and is ready for its next instruction, then it simply reads the instruction from this instruction queue resulting in increased execution speed. Fetching the next instruction while the current instruction executes is called **pipelining.**(based on FIFO) .This is much faster than sending out an addresses to the system memory and waiting for memory to send back the next instruction byte or bytes .Here the Queue will be dumped and then reloaded from the new Address.

Segment Register:

The 8086 20 bit addresses So it can address upto 2^{20} in memory (1 Mbyte) but at any instant it can address upto 4 64 KB segments. This four segments holds the upper 16 bits of the starting address of four memory segments that the 8086 is working with it at particular time .The BIU always inserts zeros for the lowest 4 bits of the 20 bit starting address

Example : If the code segment register contains 348AH then the code segment starts at 348A0H .In other words a 64Kbyte segment can be located anywhere within 1MByte address Space but the segment will always starts at an address with zeros in the lowest 4 bits

Stack: is a section of memory set aside to store addresses and data while subprogram executes is often called segment base . The stack segment register always holds the upper 16 bit starting address of program stack**.**
The extra segment register and data segment register is used to hold the upper 16 bit starting addresses of two memory segments that are used for data .
Instruction Pointer holds the 16 bit address or offset of the next code byte within the code segment. The value contained in the Instruction Pointer called as Offset because the value must be added to the segment base address in CS to produce the required 20 bit address.

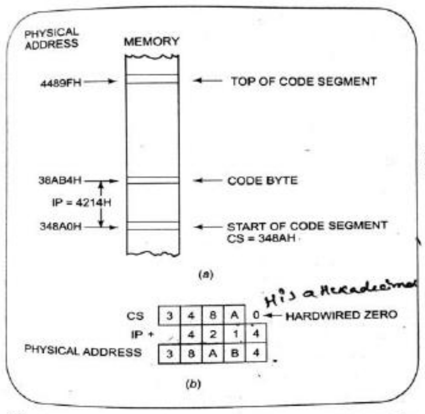


Fig. 2.10 Addition of IP to CS to produce the physical address of the code byte. (a) Diagram. (b) Computation.

CS register contains the Upper 16 bit of the starting address of the code segment in the 1 Mbyte address range the instruction pointer contains a 16 bit offset which tells wherein that 64 Kbyte code segment the next instruction byte has to be fetched from.

Stack Register and Stack Pointer:

Stack: is a section of memory set aside to store addresses and data while subprogram executes is often called segment base . The stack segment register always holds the upper 16 bit starting address of program stack. The Stack pointer (SP) holds the 16 bit offset from the starting of the segment to the memory location where a word was most recently stored .The memory location where the word is stored is called as top of the stack

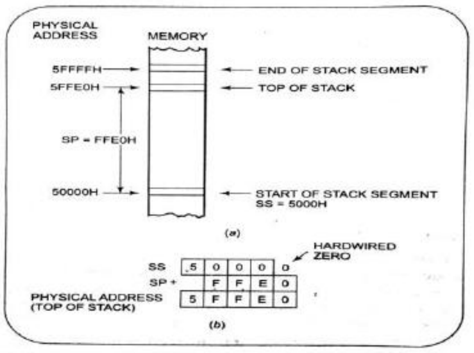


Fig. 2.11 Addition of SS and SP to produce the physical address of the top of the stack. (a) Diagram. (b) Computation.

Pointer and Index registers:
In addition to stack pointer register EU has
Base pointer Register (BP)
Source Pointer Register(SP)
Destination Pointer Register(DP)

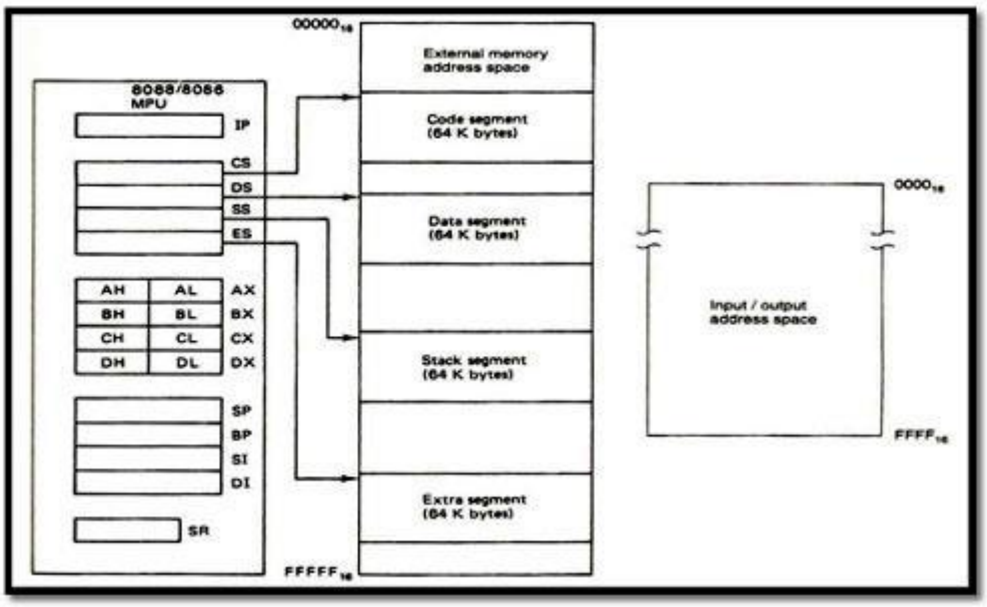
These three registers are used to store temporary storage of data like general purpose registers .They hold the 16 bit offset data of the data word in one of the segment

Programming model

How can a 20-bit address be obtained, if there are only 16-bit registers?
However, the largest register is only 16 bits (64k); so physical addresses have to be calculated. These calculations are done in hardware within the microprocessor.
The 16-bit contents of segment register gives the starting/ base address of particular segment. To address a specific memory location within a segment we need an offset address. The offset address is also 16-bit wide and it is provided by one of the associated pointer or index register.

To be able to program a microprocessor, one does not need to know all of its hardware architectural features. What is important to the programmer is being aware of the various registers within the device and to understand their purpose, functions, operating capabilities, and limitations.

The above figure illustrates the software architecture of the 8086 microprocessor. From this diagram, we see that it includes fourteenl6-bit internal registers: the instruction pointer (IP), four data registers (AX, BX, CX, and DX), two pointer registers (BP and SP), two index registers (SI and DI), four segment registers (CS, DS, SS, and ES) and status register (SR), with nine of its bits implemented as status and control flags.


00000₁₆
External memory address space
8088/8086 MPU
IP
CS
DS
SS
ES
Code segment (64 K bytes)
Data segment (64 K bytes)
Stack segment (64 K bytes)
Extra segment (64 K bytes)
AH
AL
AX
BH
BL
BX
CH
CL
CX
DH
DL
DX
SP
BP
SI
DI
SR
FFFFF₁₆
0000₁₆
Input / output address space
FFFF₁₆

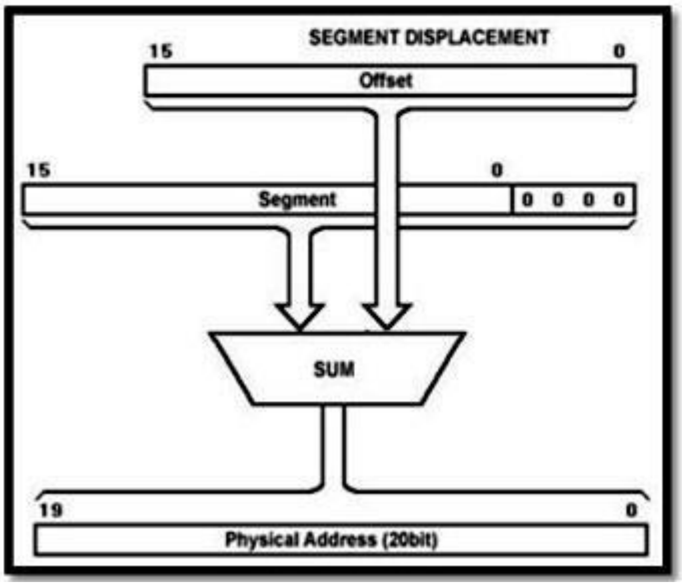
The point to note is that the beginning segment address must begin at an address divisible by 16.Also note that the four segments need not be defined separately. It is allowable for all four segments to completely overlap (CS = DS = ES = SS).

Logical and Physical Address

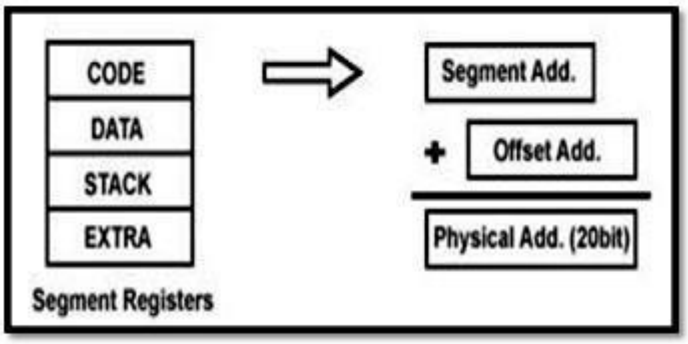
Addresses within a segment can range from address 00000h to address 0FFFFh. This corresponds to the 64K-bytelength of the segment. An address within a segment is
called an offset or logical address.

A logical address gives the displacement from the base address of the segment to the desired location within it, as opposed to its "real" address, which maps directly anywhere into the 1 MByte memory space. This "real" address is called the physical address.

What is the difference between the physical and the logical address? The physical address is 20 bits long and corresponds to the actual binary code output by the BIU on the address bus lines. The logical address is an offset from location 0 of a given segment.

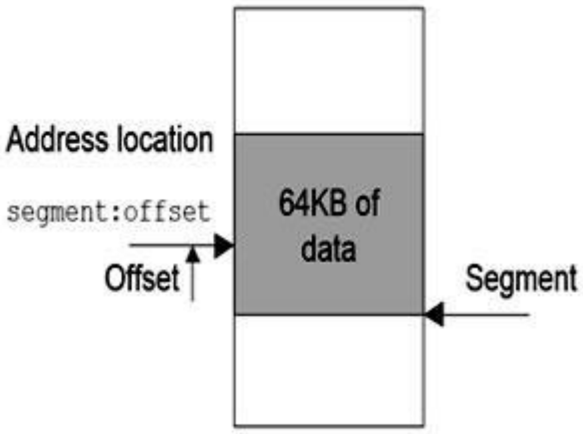


You should also be careful when writing addresses on paper to do so clearly. To specify the logical address XXXX in the stack segment, use the convention SS:XXXX, which is equal to [SS] * 16 + XXXX.



Logical address is in the form of: Base Address: Offset Offset is the displacement of the memory location from the starting location of the segment. To calculate the physical address of the memory, BIU uses the following formula:

Physical Address = Base Address of Segment * 16 + Offset



Example:

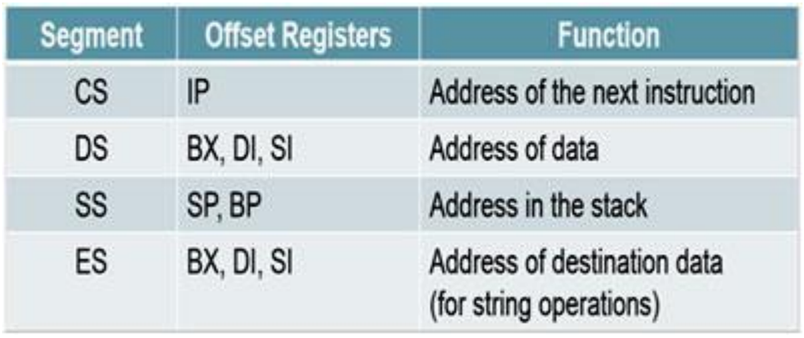
The value of Data Segment Register (DS) is 2222H.

To convert this 16-bit address into 20-bit, the BIU appends 0H to the LSB (by multiplying with 16) of the address. After appending, the starting address of the Data Segment becomes 22220H.

Data at any location has a logical address specified as:2222H: 0016H

Where 0016H is the offset, 2222 H is the value of DS Therefore the physical address:22220H + 0016H
: 22236 H

The following table describes the default offset values to the corresponding memory segments.



Segment	Offset Registers	Function
CS	IP	Address of the 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)

Some of the advantages of memory segmentation in the 8086 are as follows:

- With the help of memory segmentation a user is able to work with registers having only 16-bits.
- The data and the user’s code can be stored separately allowing for more flexibility.

- Also due to segmentation the logical address range is from 0000H to FFFFH the code can be loaded at any location in the memory.

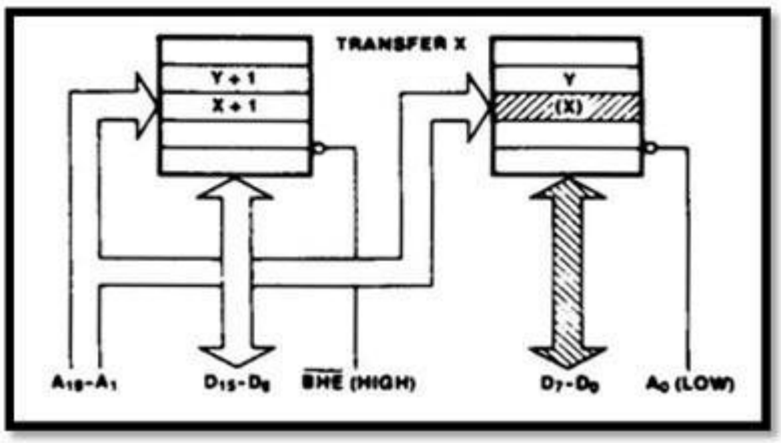
Physical memory organization:

The 8086's 1Mbyte memory address space is divided in to two independent 512Kbyte banks: the low (even) bank and the high (odd) bank. Data bytes associated with an even address (0000016, 0000216, etc.) reside in the low bank, and those with odd addresses (0000116, 0000316, etc.) reside in the high bank.

Address bits A1 through A19 select the storage location that is to be accessed. They are applied to both banks in parallel. A0and bank high enable (BHE) are used as bank-select signals.

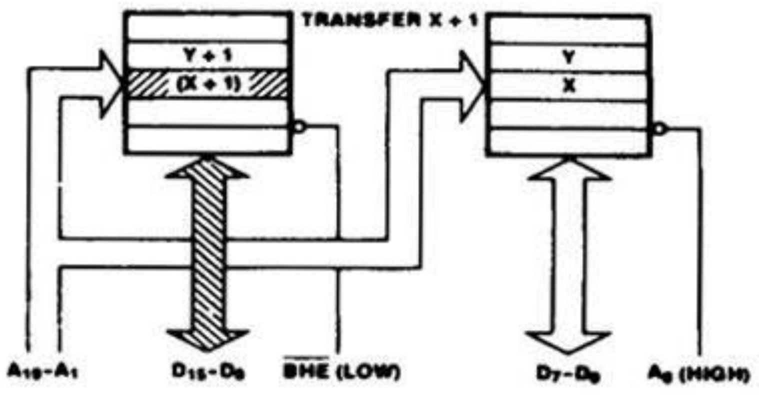
The four different cases that happen during accessing data:

Case 1: When a byte of data at an even address (such as X) is to be accessed:



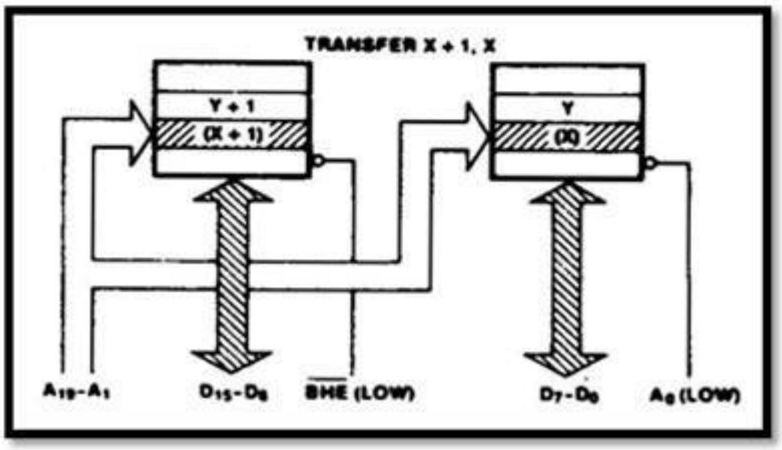
- A0 is set to logic 0 to enable the low bank of memory.
- BHE is set to logic 1 to disable the high bank.

Case 2: When a byte of data at an odd addresses (such as X+1) is to be accessed:



- A0is set to logic 1 to disable the low bank of memory.
- BHE is set to logic 0 to enable the high bank.

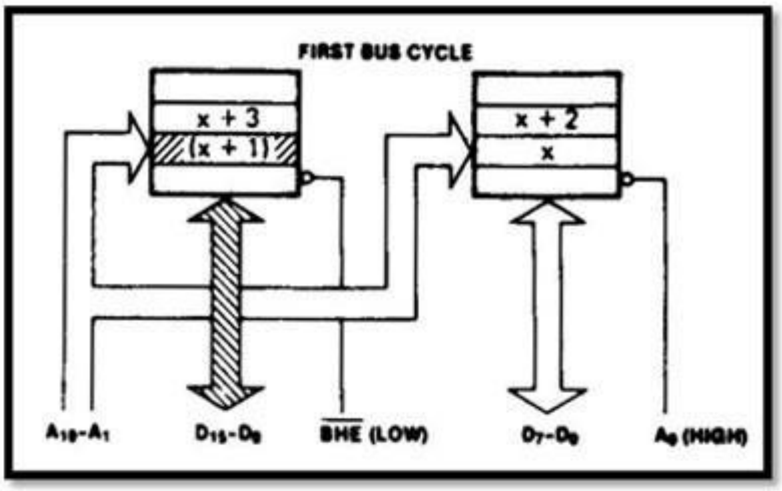
Case 3: When a word of data at an even address (aligned word) is to be accessed:



- A0 is set to logic 0 to enable the low bank of memory.
- BHE is set to logic 0 to enable the high bank.

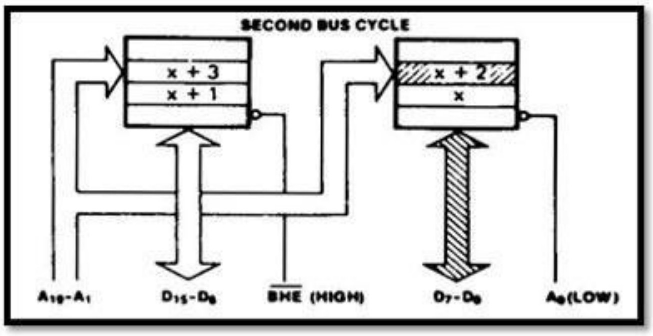
Case 4: When a word of data at an odd address (misaligned word) is to be accessed, then the 8086 need two bus cycles to access it:

a) During the first bus cycle, the odd byte of the word (in the high bank) is addressed



- A0 is set to logic 1 to disable the low bank of memory
- BHE is set to logic 0 to enable the high bank.

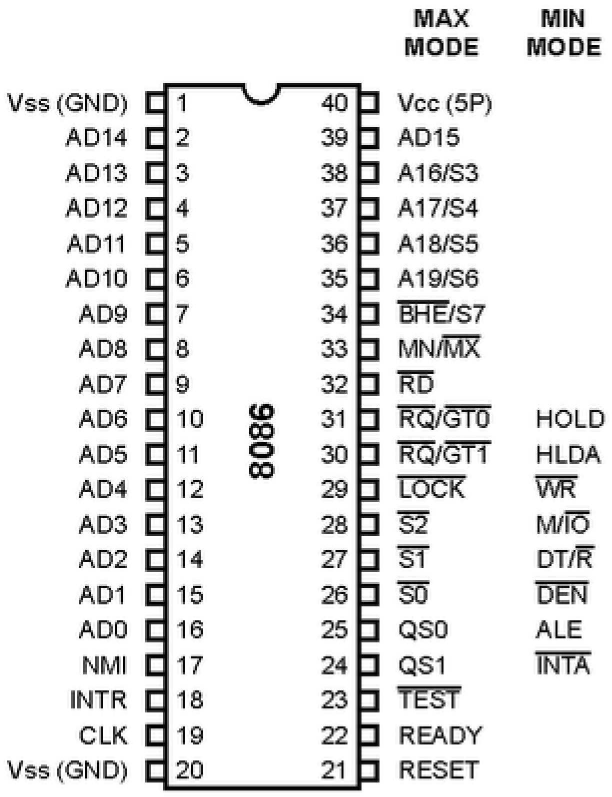
b) During the second bus cycle, the odd byte of the word (in the low bank) is addressed



- A0is set to logic 0 to enable the low bank of memory.
- BHE is set to logic 1 to disable the high bank.

Signal Description of 8086 Microprocessor

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. The pin configuration is as shown in fig1. 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.

The following signal description is common for both the minimum and maximum modes.

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. These lines are active high and float to a tristate during interrupt acknowledge and local bus hold acknowledge cycles.

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. The S4 and S3 combinedly indicate which segment register is presently being used for memory accesses as shown in Table 1.1.

These lines float to tri-state off (tristated) during the local bus hold acknowledge. The status line S6 is always low(logical). The address bits are separated from the status bits using latches controlled by the ALE signal.

S4	S3	Indication
0	0	Alternate Data
0	1	Stack
1	0	Code or none
1	1	Data

Table 1.1 Bus High Enable/Status

BHE/S7 (Active Low):

The bus high enable signal is used to indicate the transfer of data over the higher order (D15-D8) data bus as shown in Table 1.2. It goes low for the data transfers over D15-D8 and is used to derive chip selects of odd address memory bank or peripherals. $\overline{\text{BHE}}$ is low during T1 for read, write and interrupt acknowledge cycles, when- ever a byte is to be transferred on the higher byte of the data bus. The status information is available during T2, T3 and T4. The signal is active low and is tristated during 'hold'. It is low during T1 for the first pulse of the interrupt acknowledge cycle.

$\overline{\text{BHE}}$	A_0	Indication
0	0	Whole Word
0	1	Upper byte from or to odd address
1	0	Upper byte from or to even address
1	1	None

$\overline{\text{RD}}$-Read:

Read signal, when low, indicates the peripherals that the processor is performing a memory or I/O read operation. $\overline{\text{RD}}$ is active low and shows the state for T2, T3, TW of any read cycle. The signal remains tristated during the 'hold acknowledge'.

READY:

This is the acknowledgement from the slow devices 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.

INTR-Interrupt Request:

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. This can be internally masked by resetting the interrupt enable flag. This signal is active high and internally synchronized.

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. The input is synchronized internally during each clock cycle on leading edge of clock.

NMI-Non-maskable Interrupt:

This is an edge-triggered input which causes a Type2 interrrupt. 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. This input is internally synchronized.

RESET:

This input causes the processor to terminate the current activity and start execution from FFFF0H. The signal is active high and must be active for at least four clock cycles. It restarts execution when the RESET returns low. RESET is also internally synchronized.

CLK-Clock Input:

The clock input provides the basic timing for processor operation and bus control activity. Its an asymmetric square wave with 33% duty cycle. The range of frequency for different 8086 versions is from 5MHz to 10MHz.

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. The following pin functions are for the minimum mode operation of 8086.

M/IO -Memory/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. This line becomes active in the previous T4 and remains active till final T4 of the current cycle. It is tristated during local bus "hold acknowledge".

$\overline{\text{INTA}}$-Interrupt Acknowledge:

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. It is active low during T2, T3 and TW of each interrupt acknowledge cycle.

ALE-Address latch Enable:

This output signal indicates the availability of the valid address on the address/data lines, and is connected to latch enable input of latches. This signal is active high and is never tristated.

$\text{DT}/\overline{\text{R}}$-Data Transmit/Receive:

This output is used to decide the direction of data flow through the transreceivers (bidirectional buffers). When the processor sends out data, this signal is high and when the processor is receiving data, this signal is low. Logically, this is equivalent to S1 in maximum mode. Its timing is the same as M/I/O. This is tristated during 'hold acknowledge'.

$\overline{\text{DEN}}$-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. It is active from the middle ofT2 until the middle of T4 DEN is tristated during 'hold acknowledge' cycle.

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. HOLD is an asynchronous input, and it should be externally synchronized.

S2, S1, S0 -Status Lines:

These are the status lines which reflect the type of operation, being carried out by the processor. These become active during T4 of the previous cycle and remain active during T1 and T2 of the current bus cycle. The status lines return to passive state during T3 of the current bus cycle so that they may again become active for the next bus cycle during T4. Any change in these lines during T3 indicates the starting of a new cycle, and return to passive state indicates end of the bus cycle. These status lines are encoded in table 1.3

$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	**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

Table 1.3

$\overline{\text{LOCK}}$:

This output pin indicates that other system bus masters will be prevented from gaining the system bus, while the $\overline{\text{LOCK}}$ signal is low. The $\overline{\text{LOCK}}$ signal is activated by the 'LOCK' prefix instruction and remains active until the completion of the next instruction. This floats to tri-state off during "hold acknowledge". When the CPU is executing a critical instruction which requires the system bus, the LOCK prefix instruction ensures that other processors connected in the system will not gain the control of the bus. The 8086, while executing the prefixed instruction, asserts the bus lock signal output, which may be connected to an external bus controller.

QS1, QS0-Queue Status:

These lines give information about the status of the codeprefetch queue. These are active during the CLK cycle after which the queue operation is performed. These are encoded as shown in Table 1.4.

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

Table 1.4

$\overline{RQ}/\overline{GT_0}$, $\overline{RQ}/\overline{GT_1}$-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. Each of the pins is bidirectional with $\overline{RQ}/\overline{GT_0}$ having higher priority than $\overline{RQ}/\ \overline{GT_1}$, $\overline{RQ}/\overline{GT}$ pins have internal pull-up resistors and may be left unconnected. The request! Grant sequence is as follows:

1. A pulse one clock wide from another bus master requests the bus access to 8086.

2. During T4 (current) or T1 (next) clock cycle, a pulse one clock wide from 8086 to the requesting master, indicates that the 8086 has allowed the local bus to float and that it will enter the "hold acknowledge" state at next clock cycle. The CPU's bus interface unit is likely to be disconnected from the local bus of the system.

3. A one clock wide pulse from the another master indicates to 8086 that the 'hold' request is about to end and the 8086 may regain control of the local bus at the next clock cycle.

Minimum Mode 8086 System and Timings

In a minimum mode 8086 system, the microprocessor 8086 is operated in minimum mode by strapping its MN/MX* pin to logic1. In this mode, all the control signals are given out by the microprocessor chip itself. There is a single microprocessor in the minimum mode system. The remaining components in the system are latches, transreceivers, clock generator, memory and I/O devices. Some type of chip selection logic may be required for selecting memory or I/O devices, depending upon the address map of the system.

Latches:

The latches are generally buffered output D-type flip-flops, like, 74LS373 or 8282. They are used for separating the valid address from the multiplexed address/data signals and are controlled by the ALE signal generated by 8086.

Transreceivers

Transreceivers are the bidirectional buffers and some times they are called as data amplifiers. They are required to separate the valid data from the time multiplexed address/data signal. They are controlled by two signals, namely, DEN* and DT/R*. The DEN* signal indicates that the valid data is available on the data bus, while DT/R indicates the direction of data, i.e. from or to the processor.

Memory:

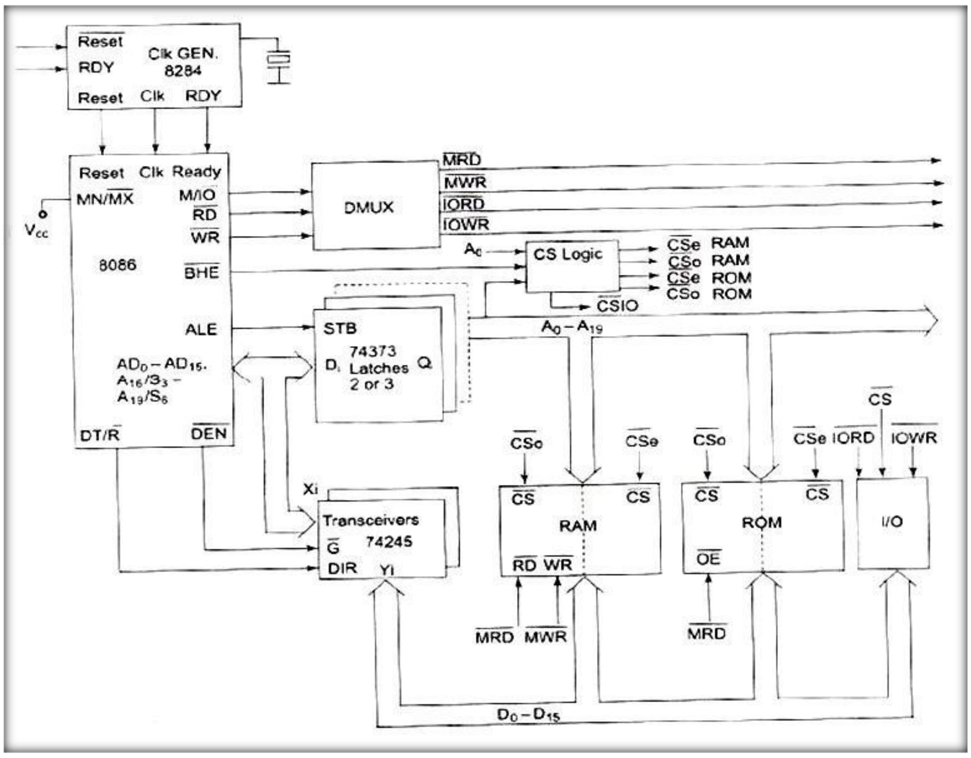
The system contains memory for the monitor and users program storage. Usually, EPROMS are used for monitor storage, while RAMs for users program storage.

IO Devices:

A system may contain I/O devices for communication with the processor as well as some special purpose I/O devices.

Clock Generator:

The clock generator generates the clock from the crystal oscillator and then shapes it and divides to make it more precise so that it can be used as an accurate timing reference for the system. The clock generator also synchronizes some external signals with the system clock.



The general system organization is shown in above fig .Since it has 20 address lines and 16 data lines, the 8086 CPU requires three octal address latches and two octal data buffers for the complete address and data separation.

The working of the minimum mode configuration system can be better described in terms of the timing diagrams rather than qualitatively describing the operations. The opcode fetch and read cycles are similar. Hence the timing diagram can be categorized in two parts.

1) Timing diagram for read cycle
2) Timing diagram for write cycle.

Timing diagram for Read cycle :

The read cycle begins in T1 with the assertion of the address latch enable (ALE) signal and also M/IO* signal. During the negative going edge of this signal, the valid address is latched on the local bus. The BHE* and A0 signals address low, high or both bytes. From Tl to T4, the M/IO* signal indicates a memory or I/O operation. At T2 the address is removed from the local bus and is sent to the output. The bus is then tristated. The read (RD*) control signal is also activated in T2 .

The read (RD) signal causes the addressed device to enable its data bus drivers. After RD* goes low, the valid data is available on the data bus.

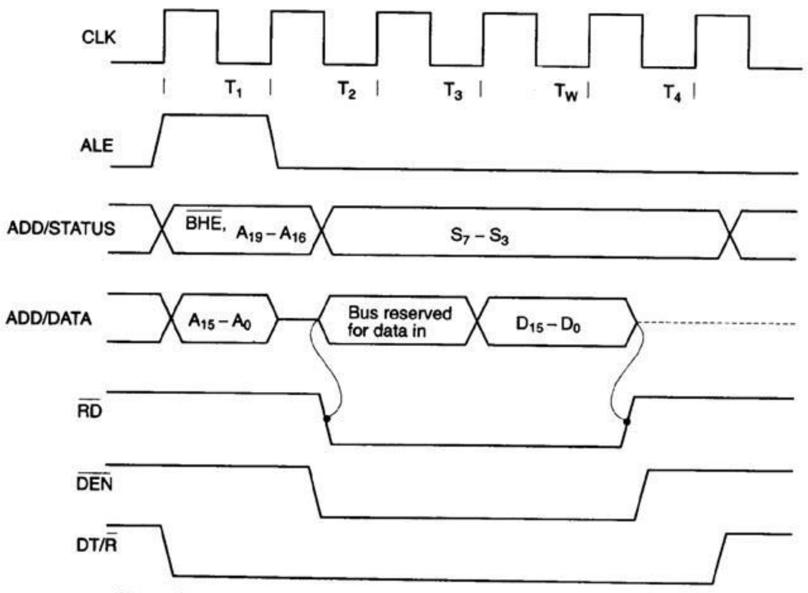
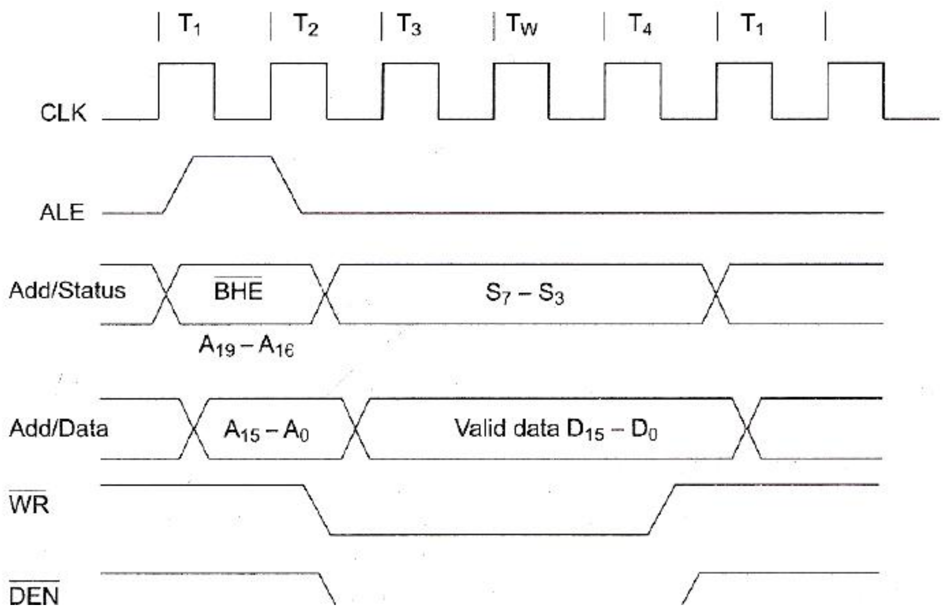


Fig. 1.9(a) *Read Cycle Timing Diagram for Minimum Mode*

The addressed device will drive the READY line high, when the processor returns the read signal to high level, the addressed device will again tristate its bus drivers.

Timing diagram for write cycle:

A write cycle also begins with the assertion of ALE and the emission of the address. The M/IO* signal is again asserted to indicate a memory or I/O operation. In T2 after sending the address in Tl the processor sends the data to be written to the addressed location. The data remains on the bus until middle of T4 state. The WR* becomes active at the beginning of T2.



The BHE* and A0 signals are used to select the proper byte or bytes of memory or I/O word to be read or written. The M/IO*, RD* and WR* signals indicate the types of data transfer as specified in Table

M/IO	RD	WR	Transfer Type
0	0	1	I/O read
0	1	0	I/O write
1	0	1	Memory read
1	1	0	Memory write

HOLD Response Sequence

The HOLD pin is checked at the end of the each bus cycle. If it is received active by the processor before T4 of the previous cycle or during T1 state of the current cycle, the CPU activities HLDA in the next clock cycle and for the succeeding bus cycles, the bus will be given to another requesting master The control control of the bus is not regained by the processor until the requesting master does not drop the HOLD pin low. When the request is dropped by the requesting master, the HLDA is dropped by the processor at the trailing edge of the next clock as shown in fig