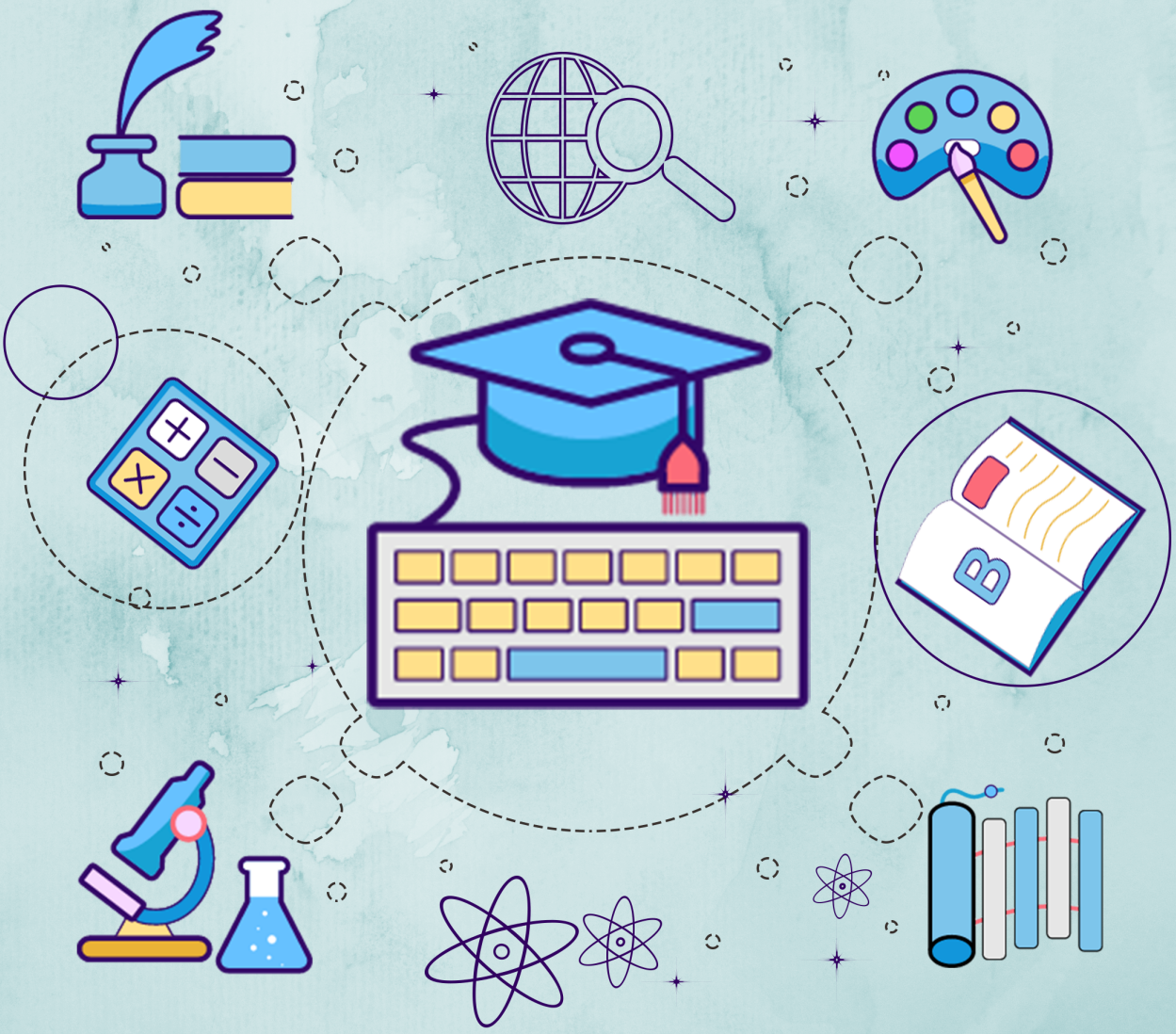
APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

KeralaNotes



SYLLABUS | STUDY MATERIALS | TEXTBOOK

PDF | SOLVED QUESTION PAPERS

www.keralanotes.com

KeralaNotes

KTU STUDY MATERIALS

MICROPROCESSORS AND MICROCONTROLLERS

CST 307

Module 1

Related Link :

- KTU S5 STUDY MATERIALS
- KTU S5 NOTES
- KTU S5 SYLLABUS
- KTU S5 TEXTBOOK PDF
- KTU S5 PREVIOUS YEAR SOLVED QUESTION PAPER

Module 1

What is microprocessor?

- The central processing unit built on a single IC is called Microprocessor.
- A microprocessor is a digital electronic component with miniaturized transistors on a single semiconductor integrated circuit (IC). ***It is a VLSI circuit with CPU, and other structural sections.***
- One or more microprocessors typically serve as a central processing unit (CPU) in a computer system or handheld device.
- A microprocessor is a LSI (large scale integration) IC that does almost all the functions of the CPU.

  Figure shows the structure of a microprocessor chip, or a VLSI core with clock and reset circuit.
- The basic function of the microprocessor is to fetch the instructions stored in the main memory, Identify the operations and the devices involved in it and accordingly generate control signals to determine when a given action is to take place.

Microprocessor chip or VLSI section CPU

Evolution of microprocessors

- 4 bit microprocessors
- -The first microprocessor was introduced in 1971 by Intel Corp.
- It was named Intel 4004 as it was a 4 bit processor.
- It could perform simple arithmetic and logic operations such as addition, subtraction, boolean AND and boolean OR. Has a control unit capable of performing control functions.
- Operates on 4 bit of data.
- The enhanced version of 4004 was Intel 4040. Some other 4 bit processors are International's PPS4 and Thoshiba's T3472.

- 8-bit Microprocessors
- The first 8 bit microprocessor which could perform arithmetic and logic operations on 8 bit words was introduced in 1973 again by Intel.
- This was Intel 8008 and was later followed by an improved version, Intel 8080. Some other 8 bit processors are Zilog-80 and Motorola M6800.
- 16-bit Microprocessors
- The 8-bit processors were followed by 16 bit processors. They are Intel 8086 and 80286.

- 32-bit Microprocessors
- The 32 bit microprocessors were introduced by several companies but the most popular one is Intel 80386.

- Pentium Series
- Instead of 80586, Intel came out with a new processor namely Pentium processor.
- Its performance is closer to RISC performance.
- Pentium was followed by Pentium Pro CPU.It allows multiple CPUs in a single system to achieve multiprocessing.MMX extension was added to Pentium Pro and result was Pentium II.
- The low cost version of Pentium II is celeron.
- The Pentium III provided high performance floating point operations for certain types of computations by using the SIMD extensions to the instruction set.
- Pentium III was faster than high-end RISC CPUs.
- Pentium IV could not execute code faster than the Pentium III when running at the same clock frequency. Pentium IV had to speed up by executing at much higher clock frequency.

- **First Generation Microprocessors**
- The first generation microprocessors were introduced in the year 1971-1972.
- The instructions of these microprocessors were processed serially. They fetched the instruction, decoded it and then executed it. When an instruction of the microprocessor was finished, then microprocessor updates the instruction pointer, and fetched the following instruction, performing this consecutive operation for each instruction in turn.
-

- **Second Generation Microprocessors**
- In the year 1974, small amount of transistors were available on the integrated circuit in the second generation microprocessors.
- Examples are: 16-bit arithmetic 7 pipelined instruction processing, MC68000 Motorola microprocessor.
- These processors were introduced in the year 1979, and Intel 8080 processor is another example.
- The 2nd generation of the microprocessor is defined by overlapped fetch, decode and execute the steps. When the 1st instruction is processed in the EU, then the second instruction is decoded, and the 3rd instruction is fetched.
- The difference between 1st generation and 2nd generation microprocessors was mainly the use of new semiconductor technologies to manufacture the chips. The result of this technology resulted in a five fold increase in instruction, speed, execution and higher chip densities.

- **Third Generation Microprocessors:** The third generation microprocessors were introduced in the year 1978, as denoted by Intel's 8086 and the Zilog Z8000.
- These were 16-bit processors with a performance like mini computers. These types of microprocessors were different from the previous generations of the microprocessors.
- **Fourth generation microprocessors**
- As many industries converted from commercial microprocessors to in house designs, the fourth generation microprocessors are entered with outstanding design with a million transistors. Leading edge microprocessors like Motorola's 88100 and Intel's 80960CA could issue and retire more than one instruction per clock cycle.

- **Fifth Generation Microprocessors**
- Fifth generation microprocessors employed decoupled super scalar processing(parallelism in a single processor), and their design soon exceeded 10 million transistors.
- In fifth generation, PCs are a low-margin, high volume business conquered by a single microprocessor.
- Latest microprocessors:-
- **INTEL CORE I7**
  - Introduced in 2008.
  - It is a 64-bit μP.
  - It has 4 physical cores.
  - Its clock speed is from 2.66 GHz to 3.33 GHz.
  - It has 781 million transistors
- **Intel Core i9-9880H Processor**
  - 16 MB Intel Smart Cache.
  - 8 Cores.
  - 16 Threads.
  - 4.80 GHz Max Turbo Frequency.
  - H - High performance graphics.
  - 9th Generation.

Development of microprocessors

Intel 8080 8085	8 bit, NMOS	1974, 2nd gen
Intel 8088 8086	16 bit	1978, 3rd gen
Intel 80186 80286	16 bit	1982
Intel 80386	32 bit, 275000 transistors, HCMOS technology(high density complementary metal oxide semiconductor)	1985, 4th gen
Intel 80486 SX DX	32 bit 32 bit, built in floating point unit	1989
Intel 80586 I MMX Celeron II III IV	64 bit	1993, 5th gen 1997 1999 2000
Z-80(Zilog)	8 bit	1976
Motorola Power PC 601 602 603	32 bit	1993 1995

- Difference between Microprocessor and Microcontroller?
- Microcontroller is a microcomputer with few application specific devices on a single chip/VLSI core.
- Used for real time applications. Eg: industrial control applications, microwave oven
- Microprocessor is a VLSI circuit with CPU, and other structural sections. Used for doing complex calculations.

8086 Microprocessor -Architecture and signals

- **Introduction**
- Intel 8086 is a 16 bit processor, launched in 1978
- Powerful instruction set, and did high speed calculations.
- Programming flexibility
- Speed was more than 8 bit microprocessors
- **Register organisation of 8086**
- 8086 has powerful set of registers known as ***general purpose and special purpose registers***. All of them are 16 bit registers.
- The general purpose registers are used as either **8 bit registers or 16 bit registers**.
- They may be used for holding data, variables and intermediate results temporarily or for other purposes like a 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: Register organisation of 8086
- There are 4 groups of register set
- **General data registers**:-
- The registers **AX, BX, CX, DX** are the general purpose 16bit registers. AX is used as 16bit **accumulator**, with the lower 8bits of AX designated as AL, and higher 8bits as AH. AL can be used as an 8bit accumulator for 8bit operations. Usually the letters L and H specify the lower and higher bytes of a particular register. CH means the higher 8bits of the CX register and CL means the lower 8 bits of CX register.

- The letter X is used to specify the complete 16bit register. The register CX is also used as a default counter in the case of string and loop instructions.
- The register BX is used as an offset storage for forming physical address in case of certain addressing modes.
- DX register is a general purpose register which may be used as an implicit operand or destination incase of a few instructions.
- **Segment registers**
- The 8086 address a segmented memory.
- The complete 1 MB memory which the 8086 addresses is divided into 16 logical segments.
- Each segment contains 64KB of memory.

- There are 4 segment registers, **code segment register(CS), data segment register (DS), extra segment register (ES) and stack segment register (SS)**. The **code segment register** is used for addressing a memory location in the code segment of the memory, where the executable program is stored.
- The **data segment register** points to data segment of the memory, where the data is resided. The extra segment also refers to a segment which essentially is another data segment of the memory. The **stack segment register** is used for addressing stack segment of memory i.e memory which is used to store stack data.
- The CPU uses the **stack** for temporarily storing important data. While addressing any location in the memory bank, the physical address is calculated from 2 parts- the first is the segment address, and the second is the offset.
- The **segment registers** contain the 16bit segment base address, related to different segments.
- Any of the **pointers and index registers** or BX may contain the offset of the location to be addressed.

- *Advantages:-*
- The processor just maintains two 16bit registers which are within the word length capacity of the machine. The CS, DS, SS, and ES segment registers, respectively contain the **segment addresses** for the code, data, stack and extra segments of the memory.
- All these segments are logical elements, they may or may not be physically separated.
- **Pointers and index registers**
- The pointers contain offset within the particular segments. The pointers **IP, BP, and SP** usually contain the offsets within the code(IP) and stack (BP and SP) segments.
- The **index registers** are used as general purpose registers as well as for offset storage in case of indexed, based indexed and relative based indexed addressing modes. The **register SI** is generally used to store the offset of source data in data segment while the **register DI** is used to store the offset of destination in data or or extra segment.
- The index registers are mainly used for string manipulations.

- **Flag register**
- The 8086 flag register contents indicate the results of computations in the ALU. It also contains some flag bits to control the CPU operations.
- 8086 has a 16bit flag register which is divided into 2 parts, **a) condition code or status flags and b)machine control flags.**
- The condition code flag register is the lower byte of the 16bit flag register along with the overflow flag. This part of the flag register of 8086 reflects the results of the operations performed by the ALU.
- The control flag register is the higher byte of the flag register of 8086.
- It contains 3 flags, **direction flag(D), interrupt flag(I) and trap flag(T)**.

Fig: Flag register of 8086

The description of each flag bit is as follows:-

- **S- sign flag**

  This flag is set when the result of any computation is negative. For signed computations the sign flag equals MSB of the result.

- **Z- Zero flag**

  This flag is set if the result of the computation or comparison performed by the previous instruction/instructions is zero.

- **P-Parity flag**

  This flag is set to 1 if the lower byte of the result contains even no of 1's.

- **C-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. If no carry is generated, value will be zero(0)

- **T- Trap flag**

  If this flag is set, the processor enters the single step execution mode. A trap interrupt is generated after execution of each instruction. The processor executes the current instruction and the control is transferred to the Trap interrupt service routine.

- **I- Interrupt flag**

  If this flag is set, the maskable interrupts are recognised by the CPU, otherwise they are ignored.

- **D- Direction flag**

  This is used by string manipulation instructions. If this flag is "o" 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.

- **AC-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.

- **Overflow flag**

  This flag is set if an overflow occurs; i.e if the result of a signed operation is large enough to be accommodated in a destination register. Eg: In case of addition of 2 signed numbers, if the result overflows into the sign bit; i.e the result is of more than 7 bits in size in case of 8 bit signed operations and more than 15 bits in size, in case of 16 bit signed operations, then the overflow flag will be set.

Architecture of 8086

- The architecture of 8086 provides a no of improvements over the 8085 architecture.
- It supports a **16 bit ALU, a set of 16 bit registers** and provides **segmented memory** addressing capability, a rich instruction set, powerful interrupt structure, **fetched instruction queue** for overlapped fetching and execution etc.
- The complete architecture of 8086 can be divided into 2 parts: a) **Bus interface unit(BIU) and** b) **Execution unit(EU)**. The bus interface unit contains the circuit for physical address calculations and a predecoding **instruction byte queue(6** bytes long).
- The bus interface unit makes the system's bus signals available for external interfacing of the devices. This unit is responsible for establishing communications with external devices and peripherals including memory via the bus.
- The 8086 addresses a segmented memory. The complete physical addresses which is 20 bits long is generated using segment and offset registers each 16bits long.

- For generating physical address from contents of these 2 registers, the content of a segment register also called as segment address is shifted left bitwise 4 times and to this result , content of an offset register also called as offset address is added, to produce a 20 bit physical address. Eg: If the segment address is 1005H and the offset is 5555H, then the physical address is calculated as below:

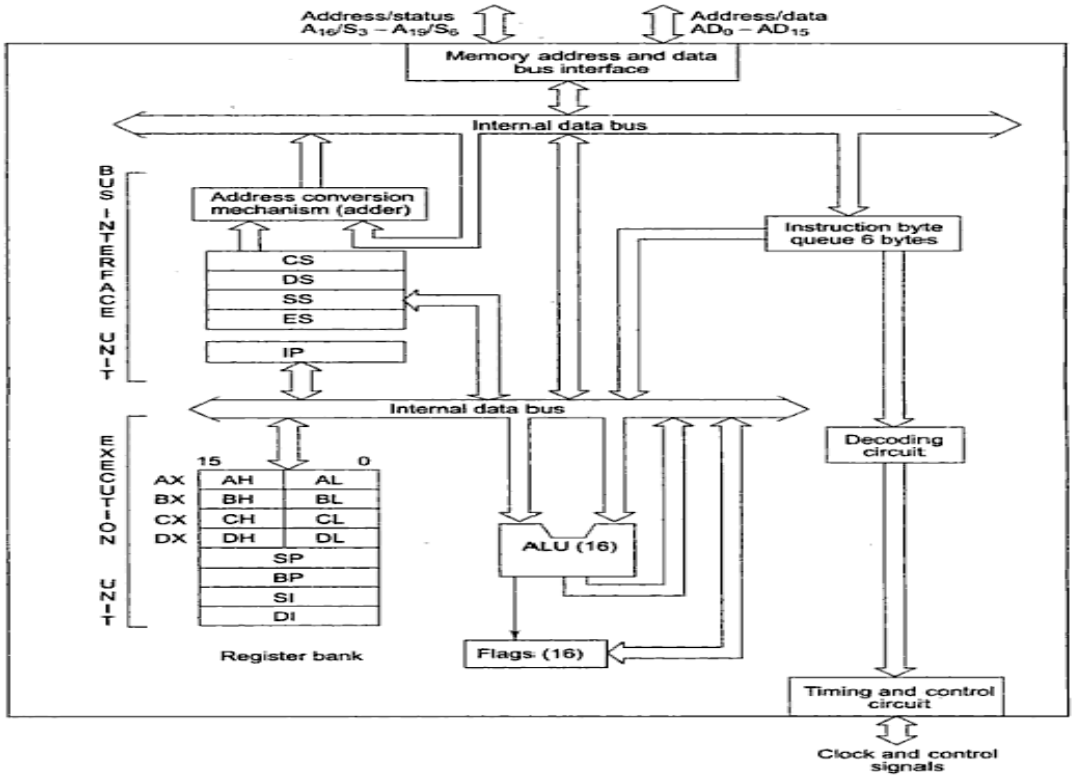
Segment address	⟶ 1005H	
Offset address	⟶ 5555H	
Segment address	⟶ 1005H ⟶	0001 0000 0000 0101
Shifted by 4 bit positions	⟶	0001 0000 0000 0101 0000
		+
Offset address	⟶	0101 0101 0101 0101
Physical address	⟶	0001 0101 0101 1010 0101
		1 5 5 A 5

- Thus the segment addressed by the segment value 1005H can have offset values from 0000H to FFFFH within it i.e maximum 64K locations may be accommodated in the segment. The segment register indicates the base address of a particular segment while offset indicates the distance of the required memory location in the segment from the base address.
- Since the offset is a 16bit number each segment can have maximum of 64bit locations.
- The bus interface unit has a separate adder to perform this procedure for obtaining a physical address while addressing memory.
- The segment address value is to be taken from an appropriate segment register depending upon whether code, data or stack are to be accessed, while the offset may be the content of IP, BX, SI, DI, SP, BP or an immediate 16bit value depending upon the addressing mode.
- In the case of 8085 once the opcode is fetched and decoded, the external bus remains free for sometime while the processor internally executes the instruction.

- This timeslot is utilised in 8086 to achieve the overlapped fetch and execution cycles. While the fetched instruction is executed internally the external bus is used to fetch the machine code of the next instruction and arrange in a queue known as predecoded instruction byte queue.
- It is 6 bytes long, first in first out structure. The instructions from the queue are taken for decoding sequentially.

- Once a byte is decoded, the queue is rearranged by pushing it out and the queue status is checked for the possibility of the next opcode fetch cycle.
- While the opcode is fetched by the Bus Interface Unit(BIU), the Execution Unit(EU) executes the previously decoded instruction concurrently. The BIU along with the Execution Unit(EU) forms a pipeline.
- The BIU manages the complete interface of execution unit with memory and I/O devices under the control of the timing and control unit.
- The execution unit contains the register set of 8086 except segment registers and IP.
- It has a 16bit ALU, able to perform arithmetic and logic operations.
- The 16bit flag register reflects the results of execution by the ALU. The decoding unit decodes the opcode bytes issued from the instruction byte queue.
- The timing and control unit derives the necessary control signals to execute the instruction opcode received from the queue, depending upon the information made available by the decoding circuit. The execution unit may pass the results to the bus interface unit for storing them in the memory.

- Fig: 8086 architecture



MEMORY SEGMENTATION IN 8086

- In segmentation memory scheme ,the complete physically available memory may be divided into a number of logical segments .
- Each segment is 64k Bytes in size and is addressed by one of the segment registers.
- The 16 bit contents of the segment register actually point to the starting location of a particular segments.
- An offset address is needed to address a specific memory location within the segments.
- The offset address is 16 bit long so that maximum offset value can be FFFF H.
- The CPU 8086 is able to address 1M bytes of physical memory .
- The complete 1M bytes of memory can be divided into 16 segments , each of 64K bytes size .
- Addresses of the segments maybe assigned as 0000H to F000H .
- Offset address values are from 0000H to FFFFH .
- Physically address range 00000H to FFFFFH .

- In 8086, memory has four different types of segments. These are:

- Code Segment
- Data Segment
- Stack Segment
- Extra Segment

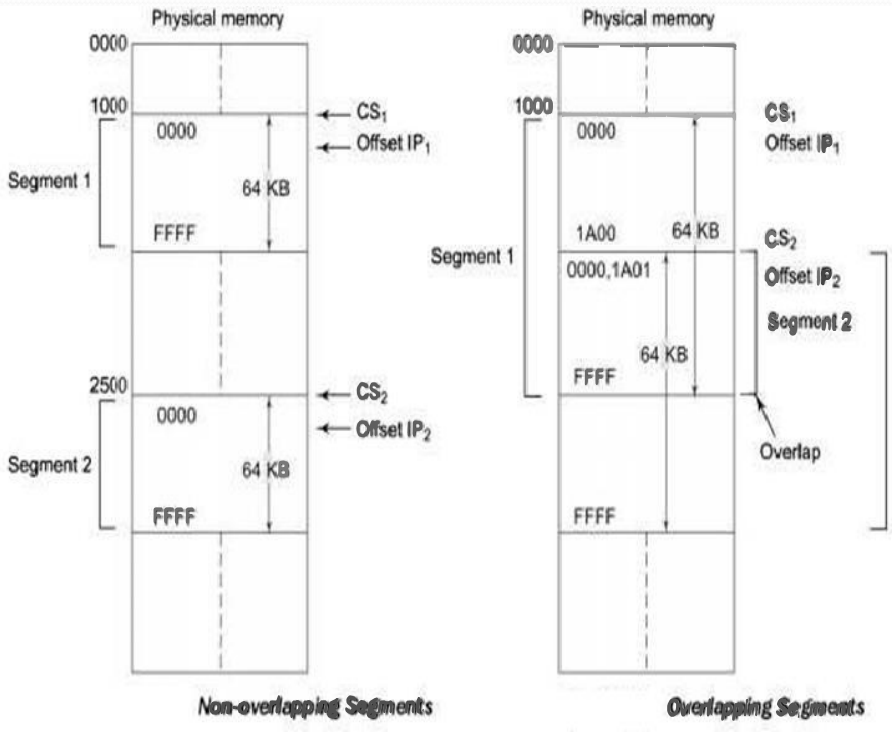
Segment	Offset Registers	Function
CS	IP	Address of the next instruction
DS	BX, DI, SI	Address of data
SS	SP, BP	Address in the stack
ES	BX, DI, SI	Address of destination data (for string operations)

- **Types Of Segmentation**

1. **Overlapping Segment**
   - • A segment starts at a particular address and its maximum size can go up to 64kilobytes.
   - • if another segment starts along with this 64kilobytes location of the first segment, then the two are said to be *Overlapping Segment*
2. **Non-Overlapping Segment**
   - • A segment starts at a particular address and its maximum size can go up to 64kilobytes.
   - • if another segment starts before this 64kilobytes location of the first segment, then the two segments are said to be *Non-Overlapped Segment*.

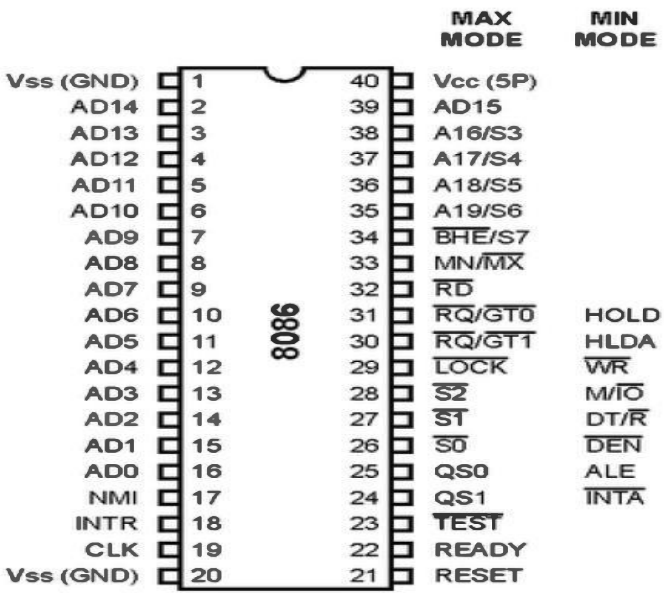
- **Advantages of segmented memory scheme**

1. Allows the memory capacity to be 1MB although the actual addresses to b e handled are of 16 bit size.
2. Allows the placing of code,data and stack portions of the same program in different parts(segments) of memory each time the program is executed i.e provision for relocation is done.
3. Permits a program and data to be put into different area of memory each time the program is executed i.e. provision for relocation is done

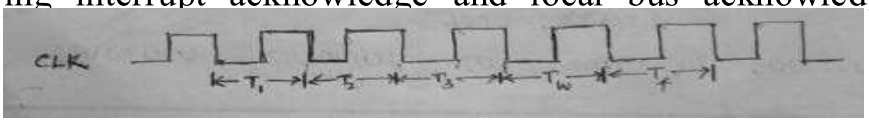


Signal Description of 8086

- Intel 8086 was launched in 1978.
- It was the first 16-bit microprocessor.
- This microprocessor had major improvement over the execution speed of 8085.
- It is available as 40-pin and Dual-Inline-Package (DIP).
- It is available in three versions:
  - 8086 (5 MHz)
  - 8086-2 (8 MHz)
  - 8086-1 (10 MHz)
- It consists of 29,000 transistors.
- It has a 16 line data bus and 20 line address bus.
- It could address up to 1 MB of memory.
- It has more than 20,000 instructions.
- It supports multiplication and division.



- **8086 signals can be categorized in three groups**
- Signals having common functions in minimum as well as maximum mode.
- Signals having special functions for minimum mode.
- Signals having special functions for maximum mode.
- **$AD_{15} - AD_0$:**
- Multiplexed memory I/O address and data lines.
- Address remains on the lines during T_1 state, while data is available on the data bus during T_2 ,T_3 , Tw and T_4.
- T_1,T_2,T_3,T_4 and Tw are the clock states of a machine cycle.
- Tw is a wait state.
- These lines are active high and float to a tri-state (high impedance state) during interrupt acknowledge and local bus acknowledge cycles.



- **A19/S6 ,A18/S5, A17/S4, A16/S3**
- Time multiplexed address and status lines.
- During T_1 , these are the most significant address lines for 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.
- S_6 is always low.
- S_5 denotes the status of interrupt enable flag bit and is updated at the beginning of each clock cycle.
- S_4 and S_3 together indicate which segment register is presently being used for memory accesses.
- These lines float to tri-state off during the local bus acknowledge.
- The address bits are separated from the status bits using latches controlled by the ALE signal

S_4	S_3	*Indications*
0	0	Alternate Data
0	1	Stack
1	0	Code or none
1	1	Data

- **BHE / S7:- Bus High Enable / Status**
- Indicate the transfer of data over the higher order (D15 - D8)data.
- It goes low for the data transfers over D15-D8 and is used to derive chip select of odd address memory bank or peripherals.
- BHE is low during T1 for read ,write and interrupt acknowledge cycles , whenever a byte is to be transferred on the higher byte of the data bus.
  - The status information is available during T2,T3,and T4.
  - Status is active low and is tri-stated during 'hold'.
- **RD** – Read :-
- Indicates the peripheral that the processor is performing a memory or I/O operation.
- Active low –T_2,T_3,Tw of read cycle.
- Tri-stated during hold acknowledge.
- **READY:-**
- Acknowledgement from the slow devices or memory that they have completed data transfer. Signal is active high.

BHE	A_0	Indication
0	0	Whole word
0	1	Upper byte from or to even address
1	0	Lower byte from or to even address
1	1	None

INTR:- Interrupt Request

- Sampled during last cycle of each instruction to determine the availability of the request.
- If any interrupt request is pending, the processor enters the interrupt acknowledge cycle.
- Internally marked by resetting the interrupt enable flag.

NMI:-Non Maskable Interrupt

- Causes Type-2 interrupt.
- Not maskable internally by software.
- Transition from low to high initiates the interrupt response at the end of each instruction.

- **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.
- It is used to test the status of math coprocessor 8087.

- **RESET**:-
- Causes the processor to terminate the current activity and start execution from FFFFOH.
- Active high for atleast 4 clock cycles.
- Restarts execution when the RESET returns low.

- **CLK**- Clock input:-
- Provide the basic timing for processor operation and bus control activity.
- Asymmetric square wave with 33% duty cycle.

- **VCC:-** +5 power supply for the operation of the internal circuit.
- **GND**:- ground for the internal circuit

- MN/$\overline{MX}$- if low then maximum mode/multiprocessor mode, if high then minimum mode/single processor mode.
- **Minimum Mode Operation-** Pin Function
- **M / I/O:-**
- Memory I/O (logically equivalent to S2 in maximum mode).
- Low indicates that CPU is having an I/O operation.
- High indicates that CPU is having a memory operation.
- Active in the previous T4 and remains active till final T4 of the current cycle.
- Tri-stated during local bus “hold acknowledge”.

- **INTA**:- Interrupt Acknowledge
- Read strobe for interrupt acknowledge cycles.
- Low means processor has accepted the interrupt.
- Active low during T2,T3 and Tw of each interrupt acknowledge cycle.
- **ALE**:-
- Indicate the availability of the valid address on address/data lines.
- Connected to latch enable input of latches.
- Active high and is never tri-stated.

- **DT/R**:- Data Transmit/Receive
- To decide the direction of data flow through transreceivers (bi- directional buffers).
- High when processor sends out data.
- Low when processor receives data.
- Tri-stated during ‘hold acknowledge’.

- **DEN** :-Data Enable
- • Indicates the availability of valid data over the address/data lines.
- • Enable the data transreceivers to separate the data from the multiplexed address/data signal.
- • Active from middle of T2 to T4.
- • Tri-stated during 'hold acknowledge' cycle
- **HOLD, HLDA**:- Hold/Hold Acknowledge
- Hold line goes high indicates to the processor that another master is requesting the bus access.
- The processor, after receiving the HOLD request, issues the HLDA signal in the middle of the next clock cycle after completing the current bus 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.
- If a DMA request is made while the CPU is performing a memory or I/O cycle, it will release the local bus during T4 provided.
    1. The request occurs on or before T2 state of the current cycle.
    2. The current cycle is not operating over the lower byte of the word (or operating on an odd address).
    3. The current cycle is not the first acknowledge of an interrupt acknowledge sequence.
    4. A lock instruction is not being executed.

- **MAXIMUM MODE:** SIGNAL DESCRIPTION

- S_2, S_1, S_0 – status lines
- Indicates the type of operation carried out by the processor.
- Becomes active during T_4 of the previous cycle and remain active during T_1 and T_2 of the current bus cycle.
- The status lines return to passive state during T_3 of the current bus cycle so that they may again become active for the next bus cycle during T_4.
- Any change in these lines during T_3 indicates the starting of a new cycle and return to passive state indicates end of the bus cycle.
-

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

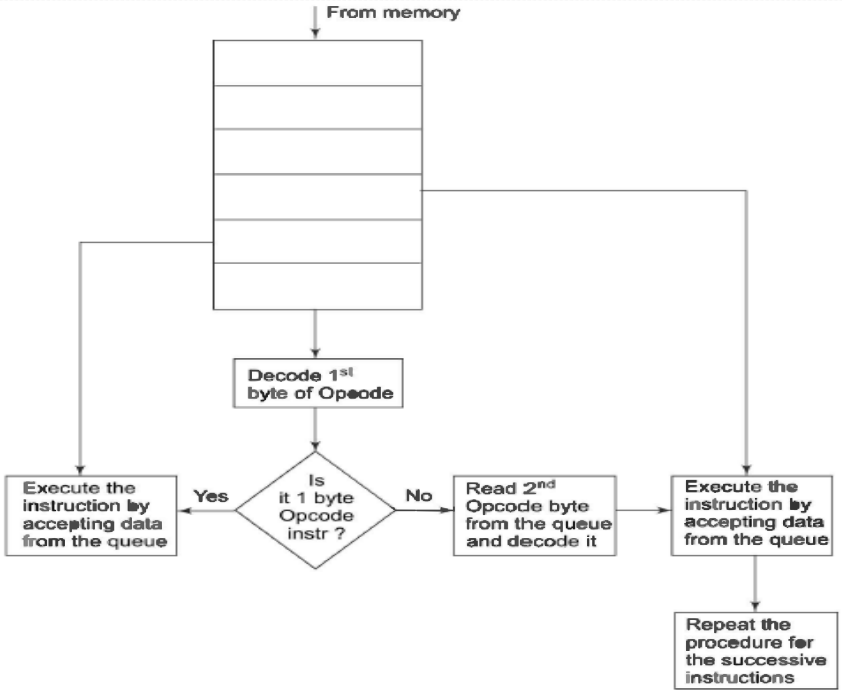
- **LOCK**
- • indicates that other system bus masters will be prevented from gaining the system bus, while the lock signal is low.
- • Activated by the 'lock' prefix instruction and remains active until the completion of the next instruction.
- • This floats to tri state off during hold acknowledge.
- **QS1, QS0-** queue status
- • Status of the code - prefetch queue.
- • Active during the clock cycle after which the queue operation is performed.

QS_1	QS_0	Queue Status
0	0	No operation
0	1	First byte of opcode from queue
1	0	Empty queue
1	1	Subsequent byte from queue

- Microprocessor offers, **pipelined processing** of instructions. The architecture has a 6byte instruction prefetch queue.
- Even the largest instruction can be **prefetched and stored** in this queue.
- Speed of execution has been increased. This scheme is **instruction pipelining**.
- CS:IP is loaded with required address for starting execution. Queue is empty initially ,and the microprocessor fetches the **first byte of instruction** code if CS:IP address is odd, or 2 bytes at a time if CS:IP **address is even.**
- Opcodes along with data is fetched and arranged in the queue.
- If 2 bytes of instruction queue are empty then, BIU starts **next fetch operation.** If the instruction is single opcode byte then next bytes are **data bytes**, else next byte is the second byte of instruction opcode.
- While execution unit is busy in executing the instruction, after it is decoded, the BIU will be fetching the next instruction bytes depending on the **queue status**.

- **RQ/GT0, RQ/GT1-** Request/grant
- • Used by other local bus masters, to force the processor to release the local bus at the end of the processors current bus cycle.
- • Bidirectional.
- • RQ/GT0 has higher priority than RQ/GT1.

- **THE REQUEST/GRANT SEQUENCE IS AS FOLLOWS:**
  - 1. A pulse one clock wide from another bus master **requests the bus acce**ss 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 in the 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 8086 may **regain control of the local bus** at the next clock cycle.



The Queue Operation

- **STEPS**

- The cs: ip is loaded with the required address from which the execution is to be started.
- Initially ,the queue will be empty and the microprocessor starts a fetch operation to bring one byte of (first byte)instruction code ,**if the cs:ip address is odd**, or 2 bytes at a time if CS:IP **address is even.**
- The first byte is a complete opcode in case of some instructions (**one byte opcode**)and it is a part of opcode in case of **two byte long opcode** instruction, and remaining part of opcode lie in 2nd byte.
- Opcodes along with data are fetched in the queue .when the first byte from the queue goes for decoding, and interpretation, one byte in the queue becomes empty and subsequently the queue is updated.
- The microprocessor does not perform the next fetch operation till **atleast two bytes of the instruction queue are emptied.**
- After decoding the first byte ,the **decoding circuit** decides whether the instruction is of single opcode byte or double opcode byte.
- If it is single opcode byte, the next bytes in the queue are treated as **data bytes**. otherwise the next byte in the queue is treated as the **second byte** of the instruction opcode.
- The second byte is then ,decoded in continuation with the first byte to decide the instruction length and no of subsequent bytes to be treated as instruction data.
- The queue is updated after every byte is read from the queue but the **fetch cycle** is initiated by BIU only if atleast two bytes of the queue are empty and **EU** may concurrently executing the fetched instructions.

Physical memory organisation

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

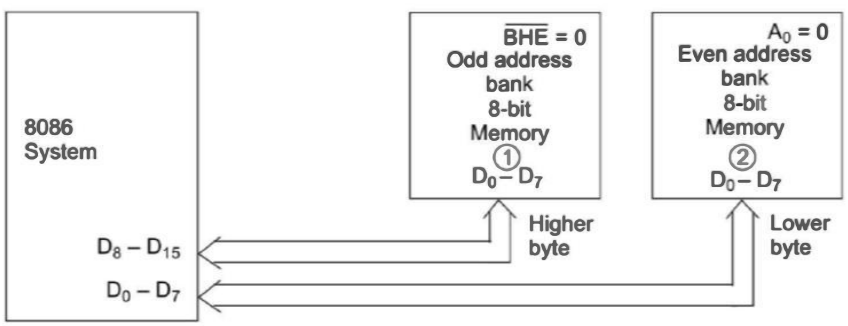
1. Both the bytes may be data operands
2. Both the bytes may contain opcode bits
3. One of the bytes may be opcode while the other may be data

All the above possibilities are taken care of by the internal decoder circuit of the microprocessor. The opcodes and operands are identified by the internal decoder circuit which further derives the signals those act as input to the timing and control unit. The timing and control unit then derives all the signals required for execution of the instruction.

While referring to word data, the BIU requires one or two memory cycles, depending upon whether the starting byte is located at an even or odd address. It is always better to locate the word data at an even address. To read or write a complete word from/to memory, if it is located at an even address, only one read or write cycle is required. If the word is located at an odd address, the first read or write cycle is required for accessing the lower byte while the second one is required for accessing the upper byte. Thus, two bus cycles are required, if a word is located at an odd address. It should be kept in mind that while initialising the structures like stack they should be initialised at an even address for efficient operation.

8086 is a 16-bit microprocessor and hence can access two bytes of data in one memory or I/O read or write operation. But the commercially available memory chips are only byte size, i.e. they can store only one byte in a memory location. Obviously, to store 16-bit data, two successive memory locations are used and the lower byte of 16-bit data can be stored in the first memory location while the second byte is stored in the next location. In a sixteen bit read or write operation both of these bytes will be read or written in a single machine cycle.

A map of an 8086 memory system starts at 00000H and ends at FFFFFH. 8086 being a 16-bit processor is expected to access 16-bit data to / from 8-bit commerically available memory chips in parallel, as shown in Fig.



Physical Memory Organisation

Thus, bits D_0 D_7 of a 16-bit data will be transferred over D_0–D_7 (lower byte) of 16-bit data bus to / from 8-bit memory (2) and bit D_8–D_{15} of the 16-bit data will be transferred over D_8–D_{15} (higher byte) of the 16-bit data bus of the microprocessor, to / from 8-bit memory (1). Thus to achieve 16-bit data transfer using 8-bit memories, in parallel, the map of the complete system byte memory addresses will obviously be divided into the two memory banks as shown in Fig.

The lower byte of a 16-bit data is stored at the first address of the map 00000H and it is to be transferred over D_0–D_7 of the microprocessor bus so 00000H must be in 8-bit memory (2). Higher byte of the 16-bit data is stored in the next address 00001H; it is to be transferred over D_8–D_{15} of the microprocessor bus so the address 00001H must be in 8-bit memory (1) of Fig. On similar lines, for the next 16-bit data stored in the memory, immediately after the previous one, the lower byte will be stored at the next address ●●●●2N and it must be in 8-bit memory (2) while the higher byte will be stored at the next address 00003H that must be in 8-bit memory (1). Thus, if it is imagined that the complete memory map of 8086 is filled with 16-bit data, all the lower bytes (D_0–D_7) will be stored in the 8-bit memory bank (2) and all the higher bytes (D_8–D_{15}) will be stored in the 8-bit memory bank (1). Consequently, it can be observed that all the lower bytes have to be stored at even addresses and all the higher bytes have to be stored at odd addresses. Thus, the 8-bit memory bank (1) will be called an odd address bank and the 8-bit memory bank (2) will be called an even address bank. The complete memory map of 8086 system is thus divided into even and odd address memory banks.

BHE'	A0	Function
0	0	Whole word
0	1	Upper byte/ odd address
1	0	Lower byte/even address
1	1	none

If 8086 transfers a 16-bit data to / from memory, both of these banks must be selected for the 16-bit operation. However, to maintain an upward compatibility with 8085, 8086 must be able to implement 8-bit operations. In which case, two possibilities arise; the first being 8-bit operation with even memory bank, i.e. with an even address and the second one is 8-bit operation with odd address memory bank, i.e. with an odd address. The two signals A_0 and $\overline{BHE}$ solve the problem of selection of appropriate memory banks as presented in Table

Certain locations in memory are reserved for specific CPU operations. The locations from FFFF0H to FFFFFH are reserved for operations including jump to initialisation programme and I/O-processor initialisation. The locations 00000H to 003FFH are reserved for *interrupt vector table*. The interrupt structure provides space for a total of 256 interrupt vectors. The vectors, i.e. CS and IP for each interrupt routine requires 4 bytes for storing it in the interrupt vector table. Hence, 256 types of interrupt require 256 C 4 = 03FFH (1Kbyte) locations for the complete interrupt vector table.

MINIMUM MODE 8086 SYSTEM AND TIMINGS

In a minimum mode 8086 system, the microprocessor 8086 is operated in minimum mode by strapping its MN/$\overline{MX}$ pin to logic 1. 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.

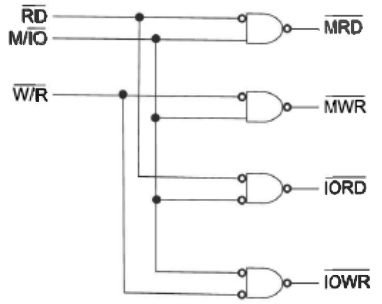


Fig. (a) *Deriving 8086 Control Signals*

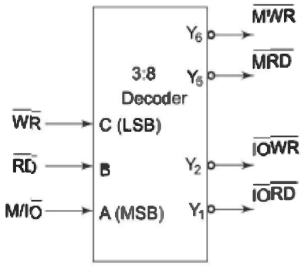


Fig. : (b) *Deriving 8086 Control Signals*

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 are the bidirectional buffers and sometimes they are called data amplifiers. They are required to separate the valid data from the time multiplexed address/data signal. They are controlled by two signals, namely, $\overline{DEN}$ and DT/$\overline{R}$. The $\overline{DEN}$ signal indicates that the valid data is available on the data bus, while DT/$\overline{R}$ indicates the direction of data, i.e. from / to the processor. The system contains memory for the monitor and users program storage. Usually, EPROMS are used for monitor storage, while RAMs for users' program storage. A system may contain I/O devices for communication with the processor as well as some special purpose I/O devices. The clock generator (IC8284) generates the clock from the crystal oscillator and then shapes it 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 organisation is shown in 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, the first is the timing diagram for *read cycle* and the second is the timing diagram for *write cycle*.

The read cycle begins in T_1 with the assertion of the Address Latch Enable (ALE) signal and M/$\overline{IO}$ signal. During the negative going edge of this signal, the valid address is latched on the local bus. The $\overline{BHE}$ and A_0 signals address low, high or both bytes. From T_1 to T_4, the M/$\overline{IO}$ signal indicates a memory or I/O operation. At T_2, the address is removed from the local bus and is sent to the output. The bus is then tristated. The Read ($\overline{RD}$) control signal is also activated in T_2. This signal causes the addressed device to enable its data bus drivers. After $\overline{RD}$ goes low, the valid data is available on the data bus. 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. CS logic indicates chip select logic and 'e' and 'O' sufixes indicate even and odd address memory banks.

A write cycle also begins with the assertion of ALE and the emission of the address. The M/$\overline{IO}$ signal is again asserted to indicate a memory or I/O operation. In T_2, after sending the address in T_1, the processor sends the data to be written to the addressed location. The data remains on the bus until the middle of T_4 state. The $\overline{WR}$ becomes active at the beginning of T_2 (unlike $\overline{RD}$ is somewhat delayed in T_2 to provide time for floating).

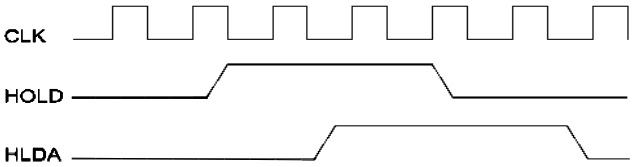
The $\overline{BHE}$ and A_0 signals are used to select the proper byte or bytes of memory or I/O word to be read or written

The M/$\overline{IO}$, $\overline{RD}$ and $\overline{WR}$ signals indicate the types of data transfer as specified in Table

$M/\overline{IO}$	$\overline{RD}$	$\overline{DEN}$	*Transfer Type*
●	0	1	I/● read
0	1	0	I/O write
1	●	1	Memory read
1	1	●	Memory write

HOLD Response Sequence

The HOLD pin is checked at the end of each bus cycle. If it is received active by the processor before T_4 of the previous cycle or during T_1 state of the current cycle, the CPU activates HLDA in the next clock cycle and for the succeeding bus cycles, the bus will be given to another requesting master. The 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,



Bus Request and Bus Grant Timings in Minimum Mode System

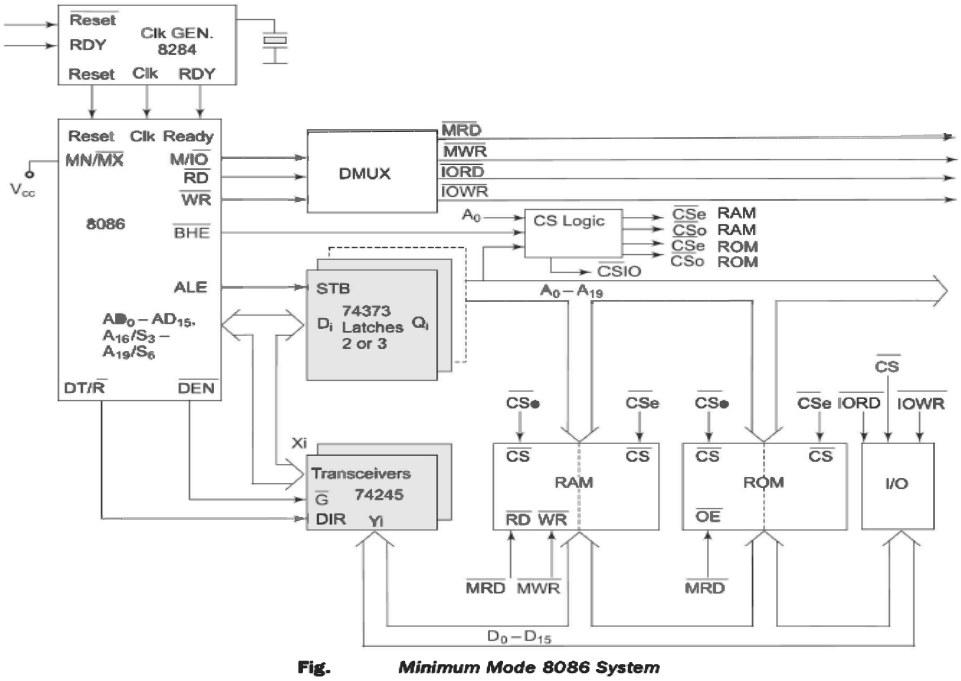


Fig. *Minimum Mode 8086 System*

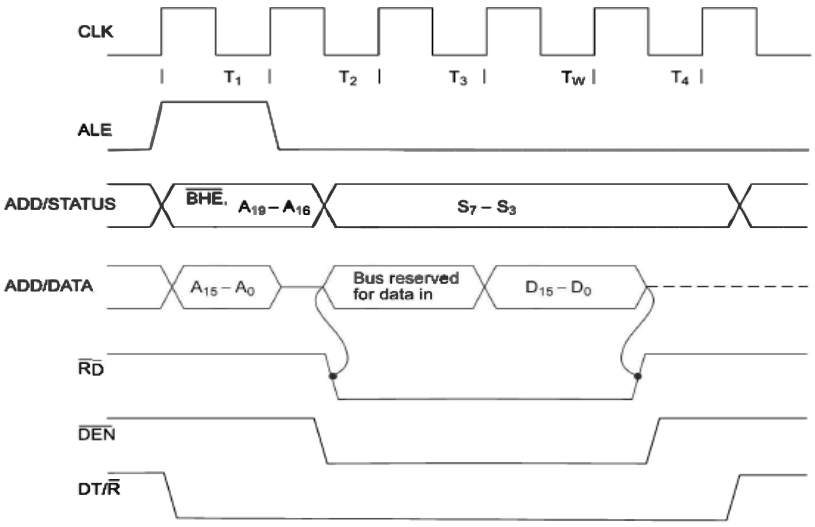


Fig. (a) ***Read Cycle Timing Diagram for Minimum Mode***

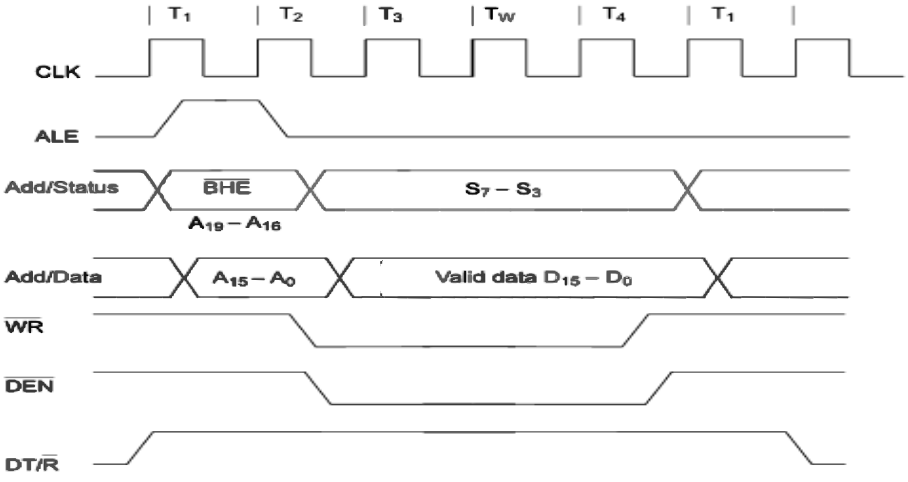


Fig. (b) ***Write Cycle Timing Diagram for Minimum Mode Operation***

Maximum mode 8086 system and timings

In the maximum mode, the 8086 is operated by strapping the MN/$\overline{MX}$ pin to ground. In this mode, the processor derives the status signals $\overline{S}_2$, $\overline{S}_1$ and $\overline{S}_0$. Another chip called bus controller derives the control signals using this status information. In the maximum mode, there may be more than one microprocessor in the system configuration. The other components in the system are the same as in the minimum mode system.

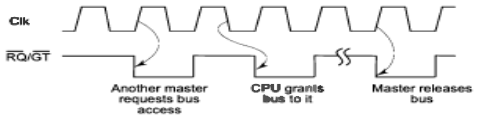
The basic functions of the bus controller chip IC8288, is to derive control signals like $\overline{RD}$ and $\overline{WR}$ (for memory and I/O devices), $\overline{DEN}$, DT/$\overline{R}$, ALE, etc. using the information made available by the processor on the status lines. The bus controller chip has input lines $\overline{S}_2$, $\overline{S}_1$ and $\overline{S}_0$ and CLK, which are driven by the CPU. It derives the outputs ALE, DEN, DT/$\overline{R}$, $\overline{MRDC}$, $\overline{MWTC}$, $\overline{AMWC}$, $\overline{IORC}$, $\overline{IOWC}$ and $\overline{AIOWC}$. The $\overline{AEN}$, IOB and CEN pins are specially useful for multiprocessor systems. $\overline{AEN}$ and IOB are generally grounded. CEN pin is usually tied to +5V. The significance of the MCE/$\overline{PDEN}$ output depends upon the status of the IOB pin. If IOB is grounded, it acts as master cascade enable to control cascaded 8259A, else it acts as peripheral data enable used in the multiple bus configurations. $\overline{INTA}$ pin is used to issue two interrupt acknowledge pulses to the interrupt controller or to an interrupting device.

$\overline{IORC}$, $\overline{IOWC}$ are I/O read command and I/O write command signals respectively. These signals enable an IO interface to read or write the data from or to the addressed port. The $\overline{MRDC}$, $\overline{MWTC}$ are memory read command and memory write command signals respectively and may be used as memory read and write signals. All these command signals instruct the memory to accept or send data to or from the bus. For both of these write command signals, the advanced signals namely $\overline{AIOWC}$ and $\overline{AMWTC}$ are available. They also serve the same purpose, but are activated one clock cycle earlier than the $\overline{IOWC}$ and $\overline{MWTC}$ signals, respectively. The maximum mode system is shown in Fig.

The maximum mode system timing diagrams are also divided in two portions as read (input) and write (output) timing diagrams. The address/data and address/status timings are similar to the minimum mode. ALE is asserted in T_1, just like minimum mode. The only difference lies in the status signals used and the available control and advanced command signals. Figure (a) shows the maximum mode timings for the read operation while the Fig. (b) shows the same for the write operation. The CS Logic block represents chip select logic and the 'e' and 'O' suffixes indicate even and odd address memory bank.

Timings for $\overline{RQ}$ / $\overline{GT}$ Signals

The request/grant response sequence contains a series of three pulses as shown in the timing diagram Fig. (c). The request/grant pins are checked at each rising pulse of clock input. When a request is detected and if the conditions discussed in pin diagram section of this chapter for valid HOLD request are satisfied, the processor issues a grant pulse over the $\overline{RQ}/\overline{GT}_0$ pin immediately during the T_4 (current) or T_1 (next) state. When the requesting master receives this pulse, it accepts the control of the bus. The requesting master uses the bus till it requires. When it is ready to relinquish the bus, it sends a release pulse to the processor (host) using the $\overline{RQ}/\overline{GT}$ pin. This sequence is shown in Fig. (c).



(c) $\overline{RQ}/\overline{GT}$ Timings in Maximum Mode

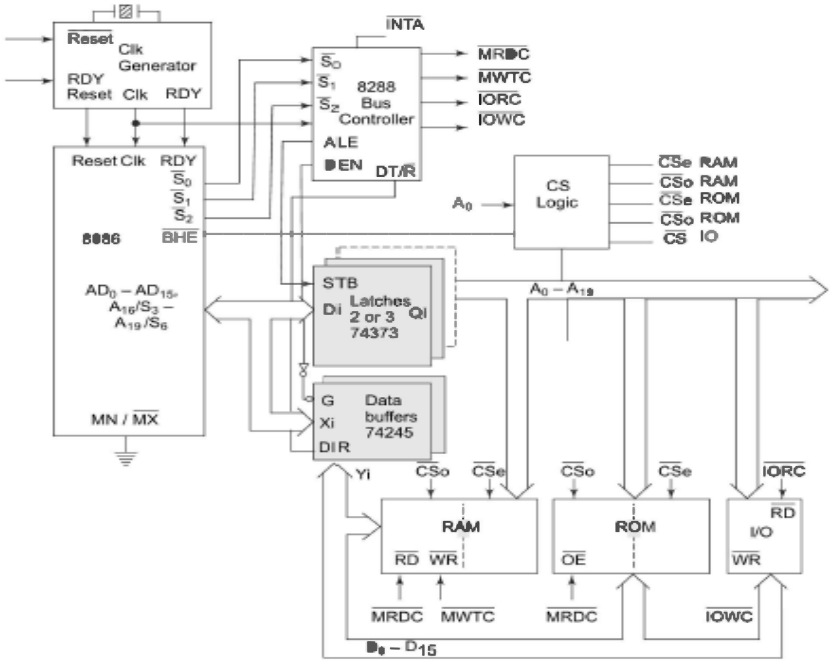


Fig. ***Maximum Mode 8086 System***

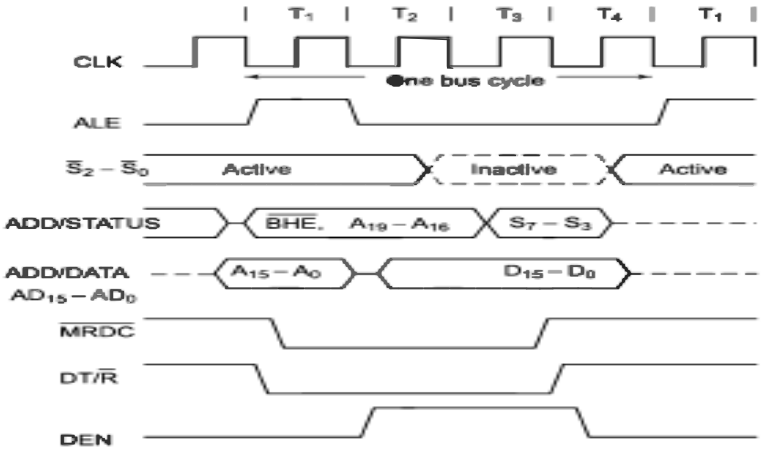


Fig (a) ***Memory Read Timing in Maximum Mode***

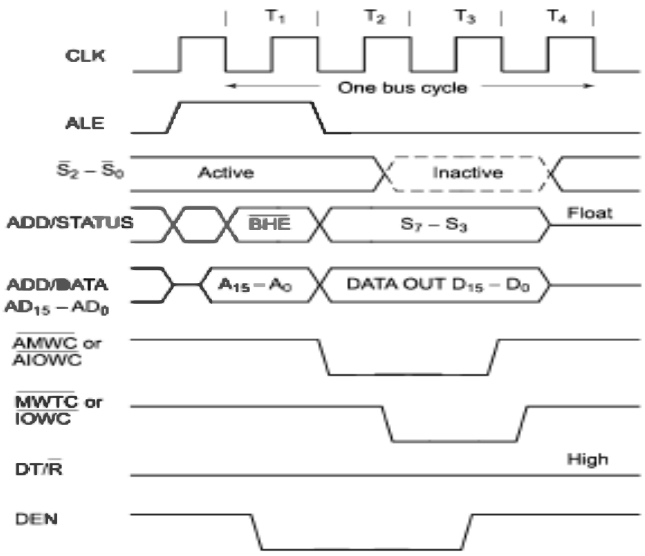


Fig. (b) *Memory Write Timing in Maximum Mode*

Comparison between 8086 and 8088

The 8088, with an 8-bit external data bus, has been designed for internal 16-bit processing capability. Nearly all the internal functions of 8088 are identical to 8086. The 8088 uses the external bus in the same way as 8086, but only 8 bits of external data are accessed at a time. While fetching or writing the 16-bit data, the task is performed in two consecutive bus cycles. As far as the software is concerned, the chips are identical,

except in case of timings. The 8088, thus may take more time for execution of a particular task as compared to 8086.

All the changes in 8088 over 8086 are directly or indirectly related to the 8-bit, 8085 compatible data and control bus interface.

1. The predecoded code queue length is reduced to 4 bytes in 8088, whereas the 8086 queue contains 6 bytes. This was done to avoid the unnecessary prefetch operations and optimize the use of the bus by BIU while prefetching the instructions.
2. The 8088 bus interface unit will fetch a byte from memory to load the queue each time, if at least 1 byte is free. In case of 8086, at least 2 bytes should be free for the next fetch operation.
3. The overall execution time of the instructions in 8088 is affected by the 8-bit external data bus. All the 16-bit operations now require additional 4 clock cycles. The CPU speed is also limited by the speed of instruction fetches.

The pin assignments of both the CPUs are nearly identical, however, they have the following functional changes.

1. A_8–A_{15} already latched, all time valid address bus.
2. $\overline{BHE}$ has no meaning as the data bus is of 8-bits only.
3. $\overline{SS}_0$ provides the S_0 status information in minimum mode.
4. $\overline{IO}$/M has been inverted to be compatible with 8085 bus structure.