8086 Microprocessors & Peripherals

(06IT/BM - 52)

Sessions handled

by

Prof. M.V. SREENIVAS RAO. *GSSSIETW, MYSORE*

Prof. M. SHIVAKUMAR. *BIET, DAVANGERE*

Prof. VAGEESH.V.KANTLI. *BIET, DAVANGERE*

8086 MICROPROCESSOR AND PERIPHERALS (06IT52)

PART - A

UNIT 1

Introduction, Microprocessor based computer system, Architecture of 8086 Microprocessor, Pin functions, Minimum / Maximum mode of operation.

UNIT 2

Read / Write timing diagrams, 8086 instruction set, Instruction template for data transfer instruction, addressing modes.

UNIT 3

Assembler directives, Programming examples.

UNIT 4

Linking and relocation, Stacks, Procedures, Interrupt and Interrupt routines, Macros.

PART – B

UNIT 5

DOS interrupt 21H function to read a character from keyboard, Write character to console, Creation of a new file, read/write from/to file, Serial / parallel communication. Interfacing devices, Memory devices and interfacing.

UNIT 6

8055 PPI device and interfacing, keyboard, display, ADC, DAC, Stepper motor and Printer interfacing using 8255.

UNIT 7

8279 programmable keyboard/display controller and interfacing, 8253 and interfacing, 8259 programmable interrupt controller and interfacing.

UNIT 8

8257 DMA controller and interfacing, serial communication using 8251 & 8087 Numeric data processor and interfacing, RS 232 serial communication standards.

TEXTBOOKS:

1 **Advanced Microprocessor and Peripherals**- A.K.Ray and K.M. Bhurchandi, Tata McGraw Hill.

2 **Microcomputer systems 8086/8088 family, Architecture, Programming and Design -** Yu-Cheng Liu & Glenn A Gibson, 2nd Edition- July 2003, Prentice Hall of India.

REFERENCE BOOKS:

1. **Microprocessor and Interfacing, Programming & Hardware-** Douglas V Hall, 2nd Edition, Tata McGraw Hill .
2. **Microprocessor Architecture, Programming and Applications with the 8085-** Ramesh S Gaonkar, 4th Edition, Penram International.

Unit 1

Introduction:

Microprocessor:

It is a semiconductor device consisting of electronic logic circuits manufactured by using either a Large scale (LSI) or Very Large Scale (VLSI) Integration Technique. It includes the ALU, register arrays and control circuits on a single chip.

The microprocessor has a set of instructions, designed internally, to manipulate data and communicate with peripherals. This process of data manipulation and communication is determined by the logic design of the microprocessor called the architecture.

The era microprocessors in the year 1971, the Intel introduced the first 4-bit microprocessor is 4004. Using this the first portable calculator is designed. The following table1 shows the list of Intel microprocessors.

Table 1

Year	Name	Bit Size
1971	4004	4
1972	8008	8
1974	8080	8
1977	8085	8
1978	8086	16
1979	8089	16
1982	80286	32
1985	80386	32
1989	80486	32
1993	80586(Pentium)	32
1995	Pentium Pro	32
1997	Pentium II	32
1999	Ecleron and Pentium III	32
2000	Pentium IV	32
2001	Intanium	64
2003	Pentium M processor	64
2005	Pentium IV and Xeon	64
2006	Pentium D 900	64

The different manufacturing companies are introduced different bit size microprocessors in the past decade is shown in the table 2

Table 2

Company Name	Processor Name
AMD	Athlon
Cypress	CY7C601
DEC	ALPHA
Fujitsu	MBL8086
Harris	CS80C286
LSI Logic	LR 30000
National Semiconductor	NS321016N
SGS-Thomson	ST6X86
SUN-Micro	SRP1030
Texas Instruments	TMS390
Toshiba	TC85R4000
Zilog	Z80
Motorola	68000

A microcomputer system just as any other computer system, include two principal components Hardware and Software. The hardware is a course the circuitry, cabinetry etc and the software is the collection of programs which direct the computer while it performs its tasks.

The memory is used to store both data and instructions that are currently being used. It is normally broken into several modules, each module containing several thousand locations. Each location may contain part or all of a datum or instruction and is associated with an identifier called a memory address. The CPU does its work by successfully inputting, or fetching instructions from memory and carrying out the tasks detected them.

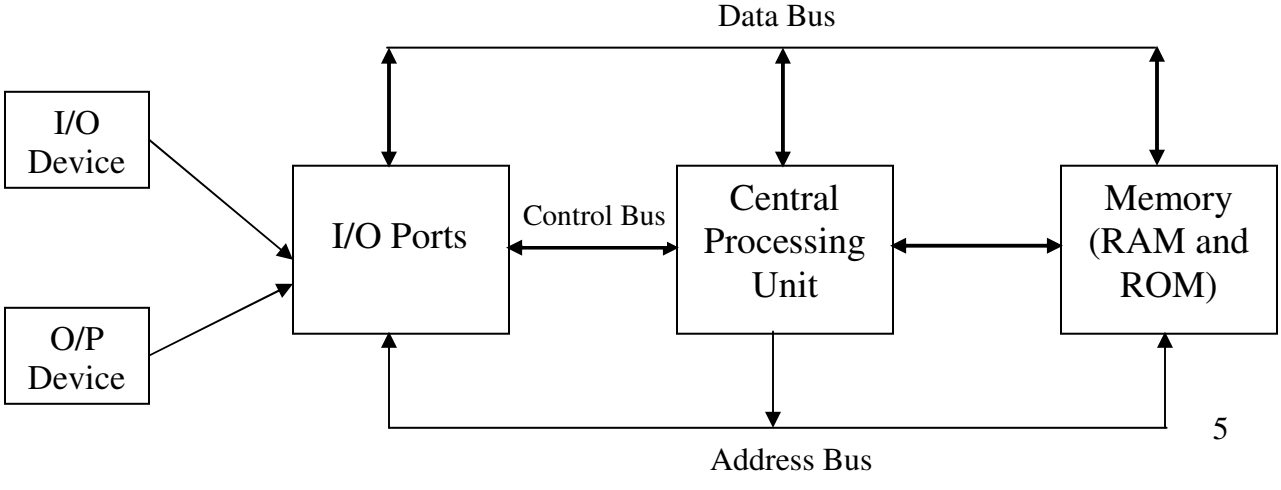


Figure1.1 shows block diagram of a simple microcomputer. The major parts are the central processing unit or CPU, memory and the input and output circuitry or Input/output. Connecting these parts are three sets of parallel line is called buses and control bus. In a microcomputer the CPU is a microprocessor and is often referred to as the microprocessor unit (MPU). Its purpose is to decode the instruction and use them to control the activity with in the system. It performs all arithmetic and logical computations.

Memory: Memory section usually consists of a mixture of RAM and ROM. It may also magnetic floppy disks, magnetic hard disks or optical disks, to store the data.

Input/output: The input/output section allows the computer to take in data from the outside world or send data to the outside world. Peripherals such as keyboards, video display terminals. Printers and modem are connected to the input/output section. These allow the user and computer to communicate with each other. The actual physical devices used to interface the computer buses to external systems are often called ports. An input/output port allows data from keyboard, an analog to digital converter (ADC) or some other source to be read into the computer under the control of the CPU. An output port is used to send data from the computer to some peripheral, such as a video display terminal, a printer or a digital to analog converter (DAC).

Central processing Unit (CPU): CPU controls the operation of the computer .In a microcomputer the CPU is a microprocessor. The CPU fetches the binary coded instructions from memory, decodes the instructions into a series of simple action and carries out these actions in sequence of steps.

CPU contains an a address counter or instruction pointer register which holds the address of the next instruction or data item to be fetched from memory, general purpose register, which are used for temporary storage or binary data and circuitry, which generates the control bus signals.

Address bus: The address bus consists of 16, 20, 24 or 32 parallel lines. On these lines the CPU sends out the address of the memory locations that are to be written to or read from. The number of memory locations that the CPU can addresses is determined by the number of address lines, then it can directly address 2n memory location. When the CPU reads data from or writes data to a port, it sends the port address on the address bus.

Ex: CPU has 16 address lines can address 2^{16} or 65536 memory locations.

Data bus: It consists of 8, 16, 32 parallel signal lines. The data bus lines are bidirectional. This means that the CPU can read, data from memory or from a port on these lines, or it can send data out to memory or to port on these lines.

Control bus: The control bus consists of 4 to 10 parallel signals lines. The CPU sends out signals on the control bus enable the outputs of addressed memory devices or port

devices. Typical control bus signal are memory read, memory write, I/O read and I/O write.

Hardware, software and Firmware: hardware is the given to the physical devices and circuitry of the computer. Software refers to collection of programs written for the computer. Firmware is the term given programs stored in ROM's or in other devices which permanently keep their stored information.

Introduction to 16-bit Microprocessor:

The 16-bit Microprocessor families are designed primarily to complete with microcomputers and are oriented towards high-level languages. Their applications sometimes overlap those of the 8-bit microprocessors. The have powerful instruction sets and capable of addressing mega bytes of memory.

The era of 16-bit Microprocessors began in 1974 with the introduction of PACE chip by National Semiconductor. The Texas Instruments TMS9900 was introduced in the year 1976. The Intel 8086 commercially available in the year 1978, Zilog Z800 in the year 1979, The Motorola MC68000 in the year 1980.

The 16-bit Microprocessors are available in different pin packages.

Ex:	Intel 8086/8088	40 pin package
	Zilog Z8001	40 pin package
	Digital equipment LSI-II	40 pin package
	Motorola MC68000	64 pin package
	National Semiconductor NS16000	48 pin package

The primary objectives of this 16-bit Microprocessor can be summarized as follows.

1. Increase memory addressing capability
2. Increase execution speed
3. Provide a powerful instruction set
4. Facilitate programming in high-level languages.

The INTEL iAPX 8086/8088:

It is a 16-bit Microprocessor housed in a 40-pin Dual-Inline-Package (DIP) and capable of addressing 1Megabyte of memory, various versions of this chip can operate with different clock frequencies

i. 8086 (5 MHz)
ii.8086-2 (8 MHz)
iii. 8086-1 (10 MHz).

It contains approximately 29,000 transistors and is fabricated using the HMOS technology . The term 16-bit means that its arithmetic logic unit, its internal registers and most of its instructions are designed to work with 16-bit binary word. The 8086 Microprocessor has a 16-bit data bus, so it can read from or write data to memory and ports either 16-bits or 8-bits at a time. The 8086 Microprocessor has 20-bit address bus, so it can address any one of 220 or 1,048,576 memory locations. Here 16-bit words will be stored in two consecutive memory locations. If the first byte of a word is at an even address, the 8086 can read entire word in one operation, If the first byte of the word is at an odd address the 8086 will read the first byte with one bus operation and the second

byte with another bus operation.

Architecture:

The internal architecture 8086 microprocessor is as shown in the fig 1.2.The 8086 CPU is divided into two independent functional parts, the Bus interface unit (BIU) and execution unit (EU).

The Bus Interface Unit contains Bus Interface Logic, Segment registers, Memory addressing logic and a Six byte instruction object code queue. The execution unit contains the Data and Address registers, the Arithmetic and Logic Unit, the Control Unit and flags.

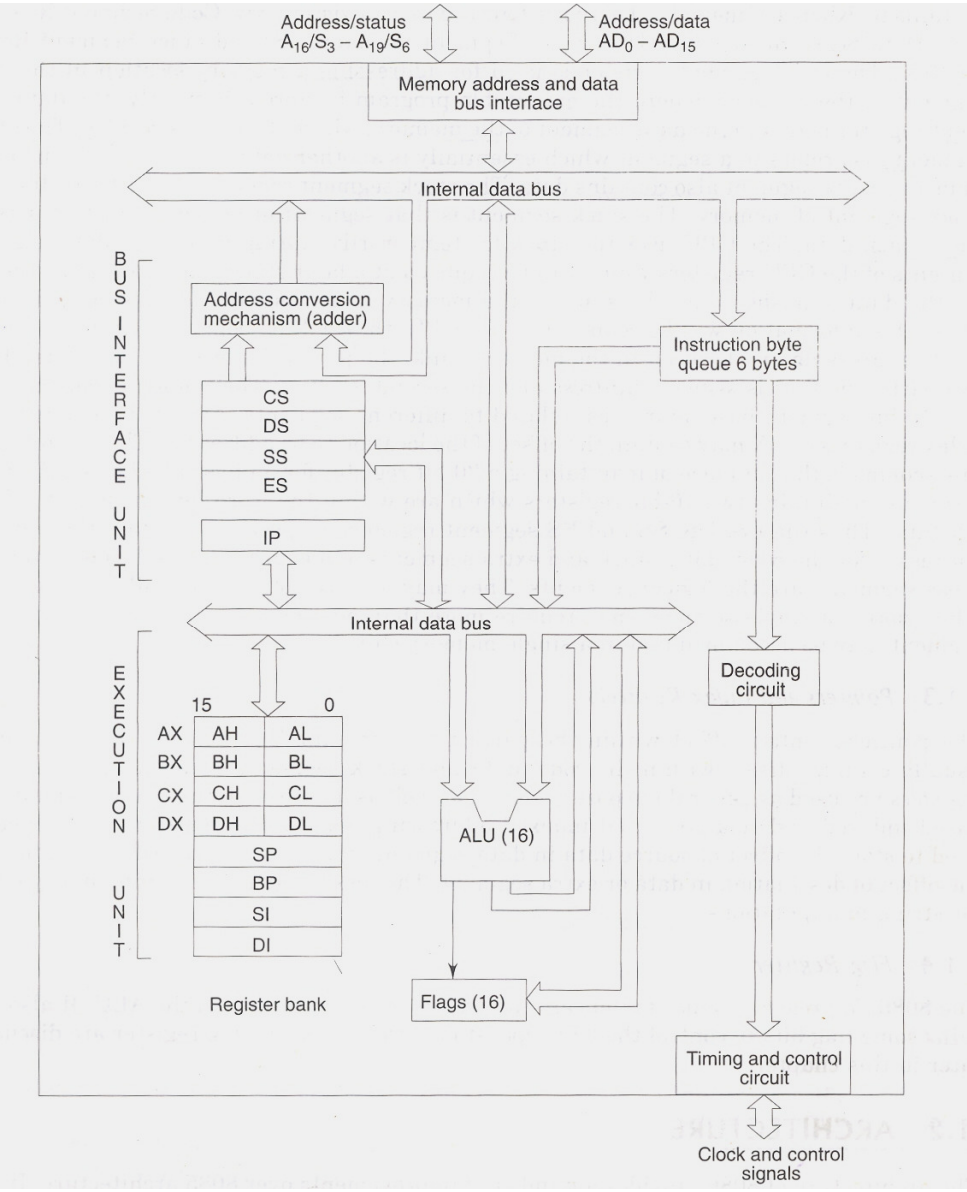


Fig1.2. Internal architecture of 8086 Microprocessor

The BIU sends out address, fetches the instructions from memory, read data from ports and memory, and writes the data to ports and memory. In other words the BIU handles all transfers of data and addresses on the buses for the execution unit.

The execution unit (EU) of the 8086 tells the BIU where to fetch instructions or data from, decodes instructions and executes instruction. The EU contains control circuitry which directs internal operations. A decoder in the EU translates instructions fetched from memory into a series of actions which the EU carries out. The EU is has a 16-bit ALU which can add, subtract, AND, OR, XOR, increment, decrement, complement or shift binary numbers. The EU is decoding an instruction or executing an instruction which does not require use of the buses.

The Queue: The BIU fetches up to 6 instruction bytes for the following instructions. The BIU stores these prefetched bytes in first-in-first-out register set called a queue. When the EU is ready for its next instruction it simply reads the instruction byte(s) for the instruction from the queue in the BIU. This is much faster than sending out an address to the system memory and waiting for memory to send back the next instruction byte or bytes. Except in the case of JMP and CALL instructions, where the queue must be dumped and then reloaded starting from a new address, this prefetch-and-queue scheme greatly speeds up processing. Fetching the next instruction while the current instruction executes is called pipelining.

Word Read

Each of 1 MB memory address of 8086 represents a byte wide location.16-bit words will be stored in two consecutive memory locations. If first byte of the data is stored at an even address**,** 8086 can read the entire word in one operation.

For example if the 16 bit data is stored at even address 00520H is 9634H

MOV BX, [00520H]

8086 reads the first byte and stores the data in BL and reads the 2nd byte and stores the data in BH

BL= (00520H) i.e. BL=34H
BH= (00521H) BH=96H

If the first byte of the data is stored at an odd address, 8086 needs two operations to read the 16 bit data.

For example if the 16 bit data is stored at even address 00521H is 3897H

MOV BX, [00521H]

In first operation, 8086 reads the 16 bit data from the 00520H location and stores the data of 00521H location in register BL and discards the data of 00520H location In 2nd operation, 8086 reads the 16 bit data from the 00522H location and stores the data of 00522H location in register BH and discards the data of 00523H location.

BL= (00521H) i.e. BL=97H
BH= (00522H) BH=38H

Byte Read:

MOV BH, [Addr]

For Even Address:

Ex: MOV BH, [00520H]

8086 reads the first byte from 00520 location and stores the data in BH and reads the 2nd byte from the 00521H location and ignores it

BH =[00520H]

For Odd Address

MOV BH, [Addr]

Ex: MOV BH, [00521H]

8086 reads the first byte from 00520H location and ignores it and reads the 2nd byte from the 00521 location and stores the data in BH

BH = [00521H]

Physical address formation:

The 8086 addresses a segmented memory. The complete physical address which is 20-bits long is generated using segment and offset registers each of the size 16-bit.The content of a segment register also called as segment address, and content of an offset register also called as offset address. To get total physical address, put the lower nibble 0H to segment address and add offset address. The fig 1.3 shows formation of 20-bit physical address.

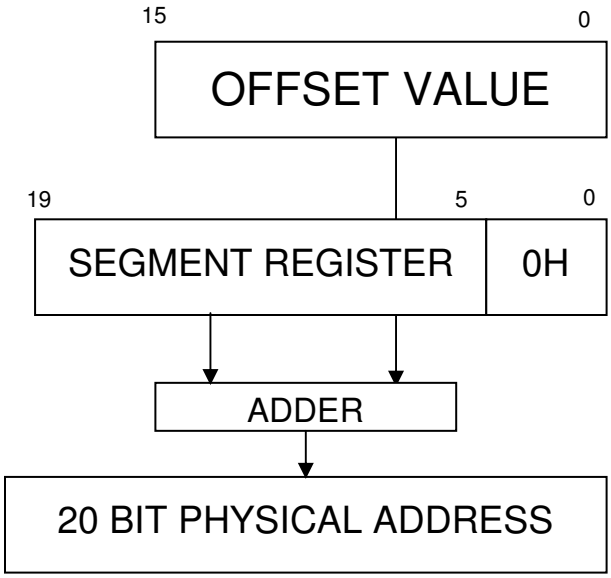


Fig. 1.3. Physical address formation

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:

General data registers:

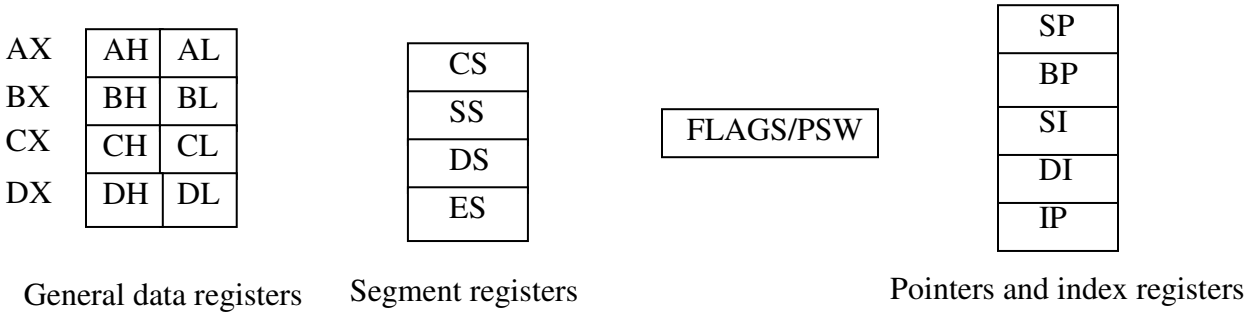


Fig.1.4 Register organization of 8086 Microprocessor

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.

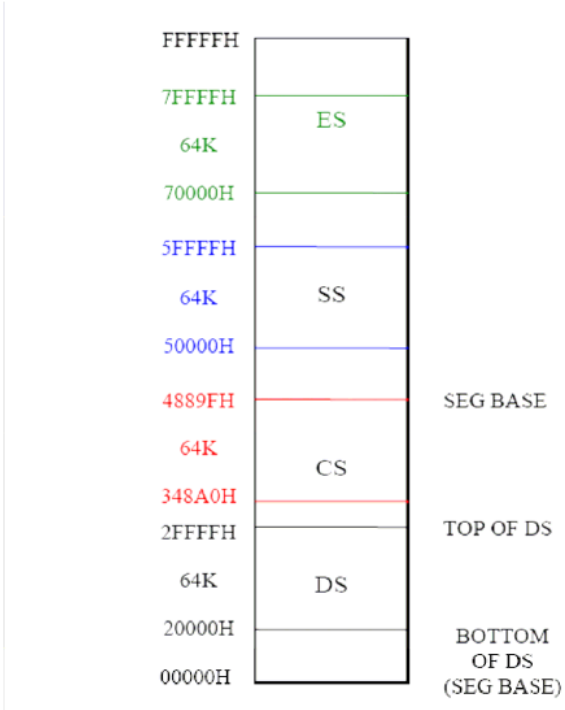


Fig1.5. Memory segmentation

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

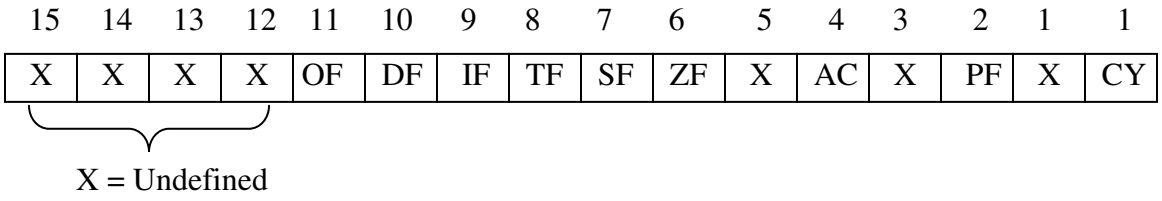


Fig1.6 . Flag Register of 8086

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

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. $D_0 - D_3$) to upper nibble (i.e. $D_4 - D_7$), the AC flag is set i.e. carry given by D_3 bit to D_4 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.

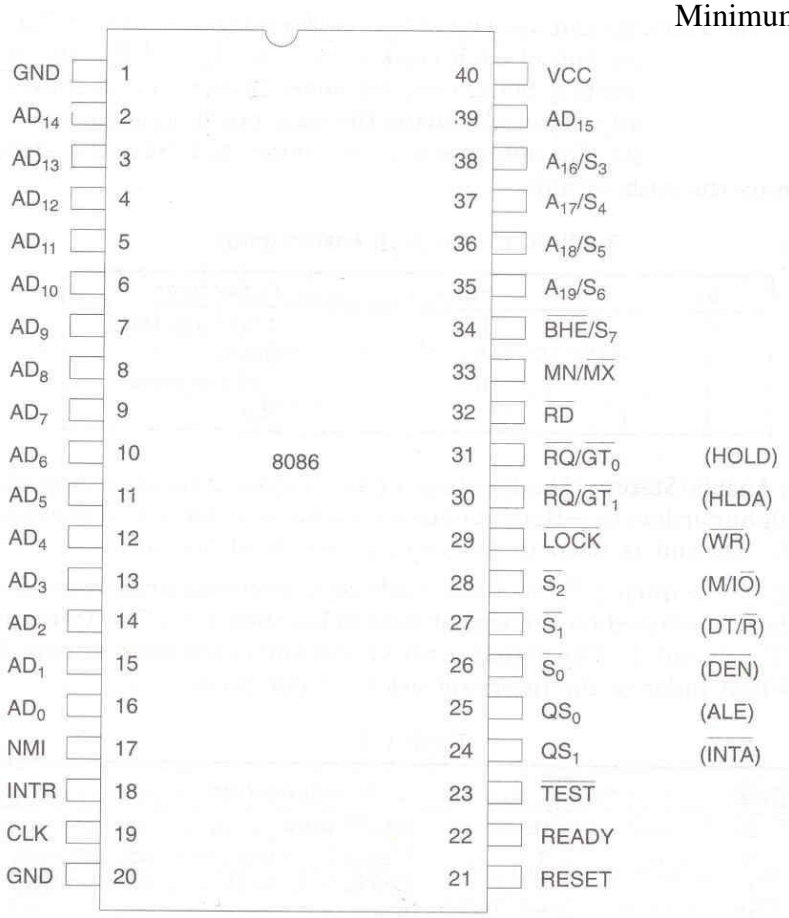
Direction Flag (DF): It is used in string operation. If it is set, string bytes are accessed from higher memory address to lower memory address. When it is reset, the string bytes are accessed from lower memory address to higher memory address.

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.

Maximum mode

Minimum mode



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 are 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 T_1 state, while the data is available on the data bus during T_2, T_3, T_W and T_4. Here T_1, T_2, T_3, T_4 and T_W are the clock states of a machine cycle. T_W is a wait state. These lines are active high and float to a tristate during interrupt acknowledge and local bus hold acknowledge cycles.

A_{19}/S_6, A_{18}/S_5, A_{17}/S_4, A_{16}/S_3: These are the time multiplexed address and status lines. During T_1, 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 T_2, T_3, T_W and T_4 .The status of the interrupt enable flag bit(displayed on S_5) is updated at the beginning of each clock cycle. The S_4 and S_3 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 S_6 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

$\overline{BHE}/S_7$-Bus High Enable/Status: The bus high enable signal is used to indicate the transfer of data over the higher order (D_{15}-D_8) data bus as shown in Table 1.2. It goes low for the data transfers over D_{15}-D_8 and is used to derive chip selects of odd address memory bank or peripherals. $\overline{BHE}$ is low during T_1 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 T_2, T_3 and T_4. The signal is active low and is tristated during 'hold'. It is low during T_1 for the first pulse of the interrupt acknowledge cycle.

$\overline{BHE}$	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

Table 1.2

$\overline{\text{RD}}$-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 T_2, T_3, T_W 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-lnterrupt 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.

V_{CC} : +5V power supply for the operation of the internal circuit. GND ground for the internal circuit.

MN/$\overline{\text{MX}}$: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/$\overline{\text{IO}}$ -Memory/IO: This is a status line logically equivalent to S_2 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 T_4 and remains active till final T_4 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 T_2, T_3 and T_W 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.

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 T_4 $\overline{\text{DEN}}$ 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.

If the DMA request is made while the CPU is performing a memory or I/O cycle, it will release the local bus during T 4 provided:

1. The request occurs on or before T 2 state of the current cycle.

2. The current cycle is not operating over the lower byte of a 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.

So far we have presented the pin descriptions of 8086 in minimum mode.

The following pin functions are applicable for maximum mode operation of 8086.

S_2, S_1, S_0 -Status Lines: These are the status lines which reflect the type of operation, being carried out by the processor. These become 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. 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.

QS_1, QS_0-Queue Status: These lines give information about the status of the code-prefetch queue. These are active during the CLK cycle after which the queue operation is performed. 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

This modification in a simple fetch and execute architecture of a conventional microprocessor offers an added advantage of *pipelined processing* of the instructions. The 8086 architecture has a 6-byte instruction prefetch queue. Thus even the largest (6-bytes) instruction can be prefetched from the memory and stored in the prefetch queue. This results in a faster execution of the instructions. In 8085, an instruction (opcode and operand) is fetched, decoded and executed and only after the execution of this instruction, the next one is fetched. By prefetching the instruction, there is a considerable speeding up in instruction execution in 8086. This scheme is known as *instruction pipelining*. At the starting 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 (the first byte) of instruction code, if the CS:IP address is odd or two bytes at a time, if the CS:IP address is even. The first byte is a complete opcode in case of some instructions (one byte opcode instruction) and it is a part of opcode, in case of other instructions (two byte long opcode instructions), the remaining part of opcode may lie in the second byte. But invariably the first byte of an instruction is an opcode. These opcodes along with data are fetched and arranged 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 at least two bytes of the instruction queue are emptied. The instruction execution cycle is never broken for fetch operation. 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 are treated as data bytes depending upon the decoded instruction length, other wise, 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 the number 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 at least, two bytes of the queue are empty and the EU may be concurrently executing the fetched instructions.

The next byte after the instruction is completed is again the first opcode byte of the next instruction. A similar procedure is repeated till the complete execution of the program. The main point to be noted here is, that the fetch operation of the next instruction is overlapped with the execution of the current instruction. As shown in the architecture, there are two separate units, namely, execution unit and bus interface unit. While the execution unit is busy in executing an instruction, after it is completely decoded, the bus interface unit may be fetching the bytes o(the next instruction from memory, depending upon the queue status. Figure 1.6 explains the queue operation.

$\overline{RQ}/\overline{GT_0}$, $\overline{RQ}/\overline{GT_1}$-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 $\overline{RQ}/\overline{GT_1}$, RQ/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 T_4 (current) or T_1 (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.
Thus each master to master exchange of the local bus is a sequence of 3 pulses. There must be at least one dead clock cycle after each bus exchange. The request and grant pulses are active low. For the bus requests those are received while 8086 is performing memory or I/O cycle, the granting of the bus is governed by the rules as discussed i~ case of HOLD, and HLDA in minimum mode.

Until now, we have described the architecture and pin configuration of 8086. In the next section, we will study some operational features of 8086 based systems.

8284A Clock Generator

The 8284A is an ancillary component to the 8086/8088 microprocessor. Without the clock generator, many additional circuits to generate the clock (CLK in an 8086/8088 based system. The clock Generator 8284A provides the following basic functions or signals: clock generation, RESEST synchronization, READY synchronization, and a TTL level peripheral clock signal.

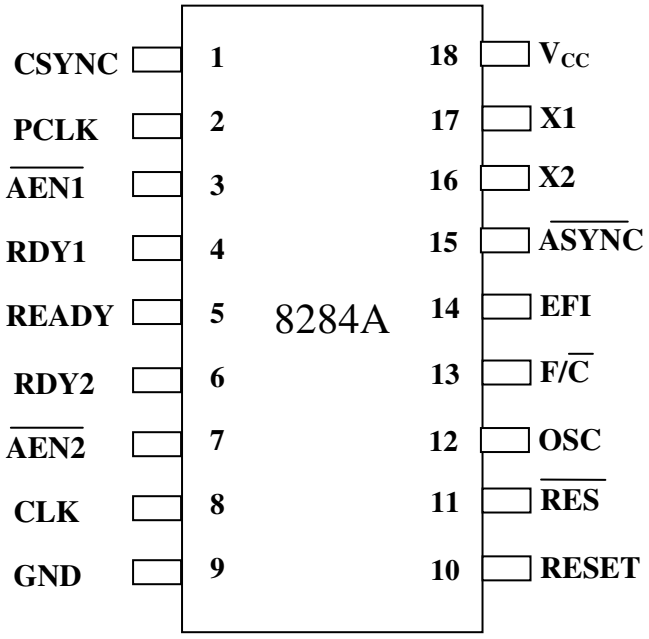


Fig1. Pin diagram of the 8284A clock generator

Pin Functions: The 8284A is an 18-pin integrated circuit designed specifically for use with the 8086/8088 microprocessors as shown in fig1. The following is a list of each pin and its function.

AEN1* and AEN2*: The address enable pins are provided to qualify the ready signals. RDY1 and RDY2, respectively. Which are used to cause wait states, along with the RDY1 and RDY2 inputs. Wait states are generated by the READY pin of the 8086/8088 microprocessor. This is controlled by these two inputs.

RDY1 and RDY2: The bus ready inputs are provided in conjunction with the AEN1* and AEN2* pins to cause wait states in an 8086/8088 microprocessor based system.

ASYNC*: The ready synchronization selection input selects either one or two stages of synchronization for the RDY1 and RDY2 inputs.

READY: Ready is an output pin that connects to the 8086/8088 microprocessor READY input. This signal is synchronized with the RDY1 and RDY2 inputs.

X1 and X2: The Crystal Oscillator pins connect to an external crystal used as the timing source for the clock generator and all its functions.

F/C*: The Frequency/Crystal select input results the clocking source for the 8284A. If this pin is held high, an external clock is provided to the EFI input pin, and if it is held low, the internal crystal oscillator provides the timing signal.

EFI: The External Frequency input is used when the F/C is pulled high. EFI supplies the timing whenever the F/C* pin is high.
.

CLK: The clock output pin provides CLK input signal to the 8086/8088 microprocessors and other components in the system. The CLK pin has an output signal that is one-third of the crystal or EFI input frequency and has a 33 percent duty cycle, which is required by the 8086/8088 microprocessors.

PCLK: The Peripheral Clock signal is one-sixth the crystal or EFI input frequency and has a 50 percent duty cycle. The PCLK output provides a clock signal to the peripheral equipment in the system.

OSC: The Oscillator output is a TTL level signal that is at the same frequency as the crystal or EFI input. (The OSC output provides and EFI input to other 8284A clock generators in some multiple processor systems).

RES*: The reset input is an active-low input to the 8284A. The RES* pin is often connected an RC network that provides power-on resetting.

RESET: The Reset output is connected to the 8086/8088 microprocessors RESET input pin.

CSYNC: The clock synchronization pin is used whenever the EFI input provides synchronization in systems with multiple processors. When the internal crystal oscillator is used, this pin must be grounded.

GND: The ground pin is connects to ground.

V_{cc}: This power supply pin connects to + 5.0V with a tolerance of ± 10 percent

Operation of the 8084A:

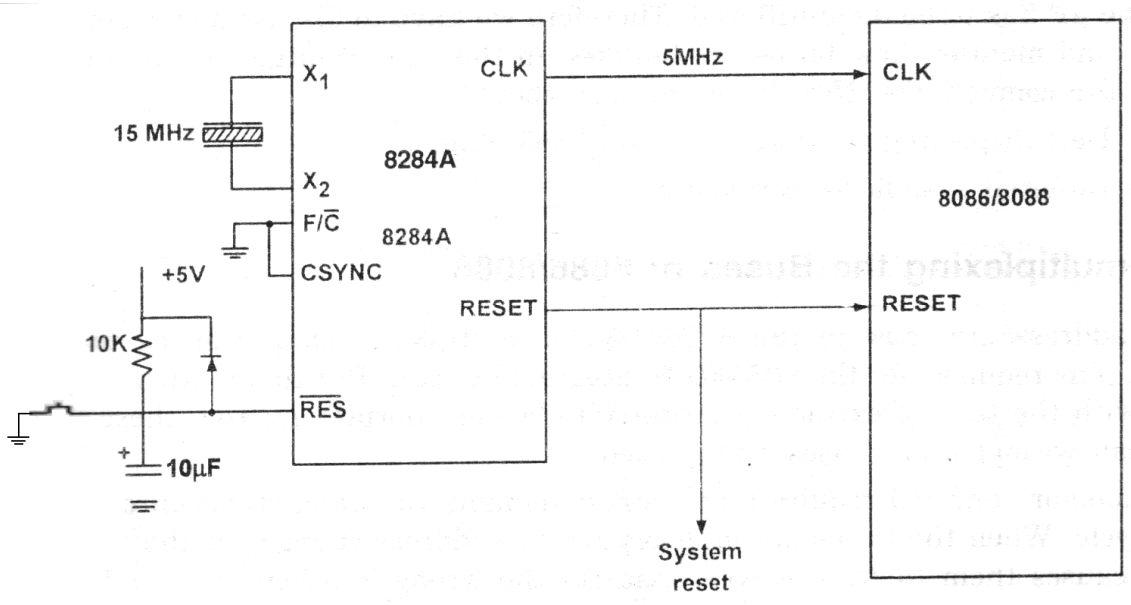


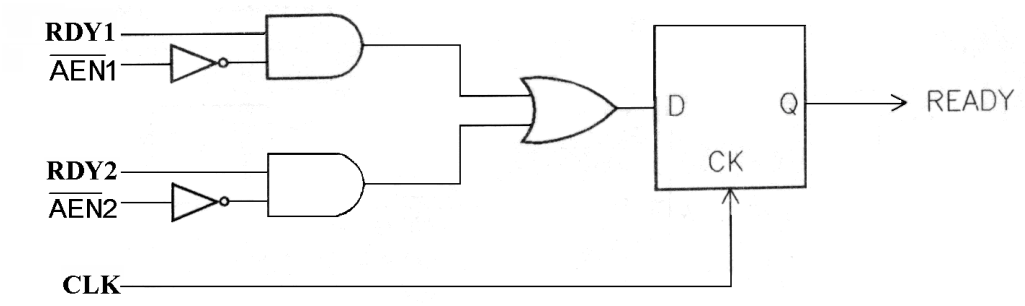
Fig1.1. The clock generator (8284A) and the 8086/8088 microprocessor illustrating the connection for the clock and reset signal. (A 15 MHz of crystal provides the 5MHz clock for the microprocessor.)

The Reset section of the 8284A is very simple. It consists of a Schmitt trigger buffer and a single D-type flip flop circuit. The D-type flip flop ensures that the timing requirements of the 8086/8088 microprocessor RESET input are met. The circuit applies the RESET signal to the microprocessor negative edge (1-to-0 transition) of each clock. The 8086/8088 microprocessor sample RESET at the positive edge (0-to-1 transition) of the clocks; therefore, the circuit meets the timing requirements of the 8086/8088 microprocessor.

The RC circuit provides a logic 0 to the RES* input pin when power is first applied to the system. After a short time, the RES* input becomes a logic 1 because the capacitor charges toward +5.0V through the resistor. A push-button switch allows the microprocessor to be reset by the operator. Correct reset timing requires the RESET input

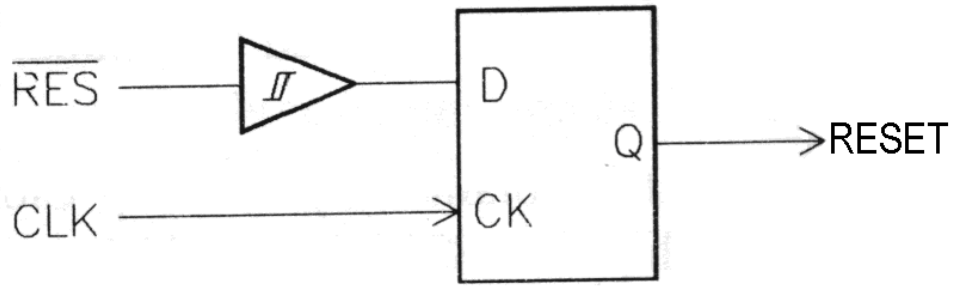
to become logic 1 no later than four clocks after system power is applied and to be held high for at least 50µs. The flop-flop makes certain that RESET goes high in four clocks, and the RC time constant ensures that it stays high for at least 50µs.

Ready Logic



The Ready Logic generates the Ready signal for the 8086/8088 microprocessor. If the Ready signal is made low by this circuit during T2 state of a machine cycle, the microprocessor introduces a wait state between T3 and T4 states of the machine cycle. The Ready logic is indicated in the figure. There are two pairs of signals in 8284 which can make the Ready output of 8284 to go low. If (RDY1=0 or AEN1*=1) and (RDY2=0 or AEN2*=1), the Ready output becomes low when the next clock transition takes place. In PCs, RDY2 and AEN2* are not used, and as such RDY2 is tied to Ground and /or AEN2* is tied to +5V. AEN1* is used for generating wait states in the 8086/8088 bus cycle, and RDY1 is used for generating wait state in the DMA bus cycle.

Reset Logic



The Reset logic generates the Reset input signal for the 8086/8088. When the RESET* pin goes low, the Reset output is generated by the 8284 when the next clock transition takes place.

In PCs, the RES* input is activated by one of the following.

- ✓ From the manual Reset button on the front panel.
- ✓ From the 'Power on Reset' circuit, which uses RC components.
- ✓ If the 'Power Good' signal from the SMPS is not active.

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.

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 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. 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 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 Fig. 1.1. 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.

Fig 1.2 shows the read cycle timing diagram. The read cycle begins in T_1 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 A_0 signals address low, high or both bytes. From T_1 to T_4, 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 T_2 .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. 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.

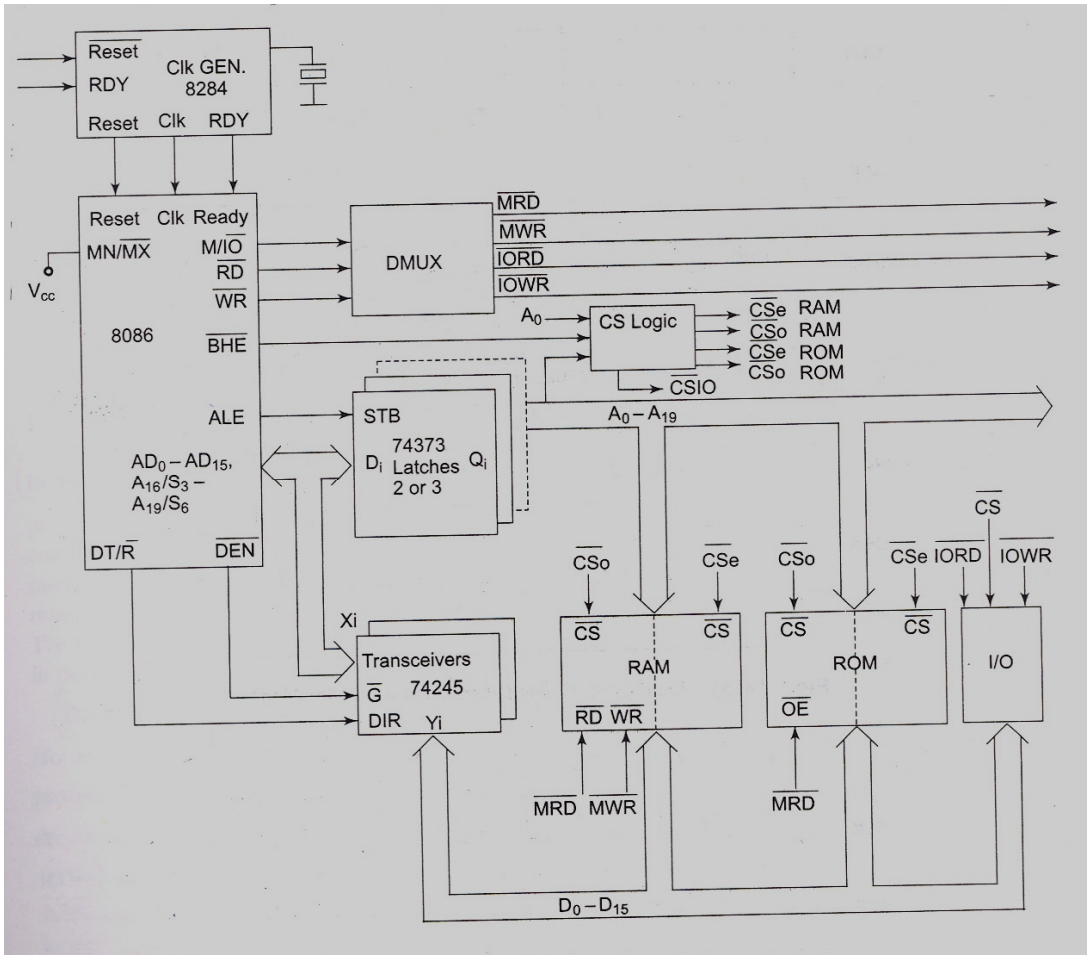


Fig 1.1. Minimum Mode 8086 System

Fig 1.3 shows the write cycle timing diagram. 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 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 middle of T_4 state. The WR* becomes active at the beginning ofT_2 (unlike RD* is somewhat delayed in T_2 to provide time for floating).

The 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/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

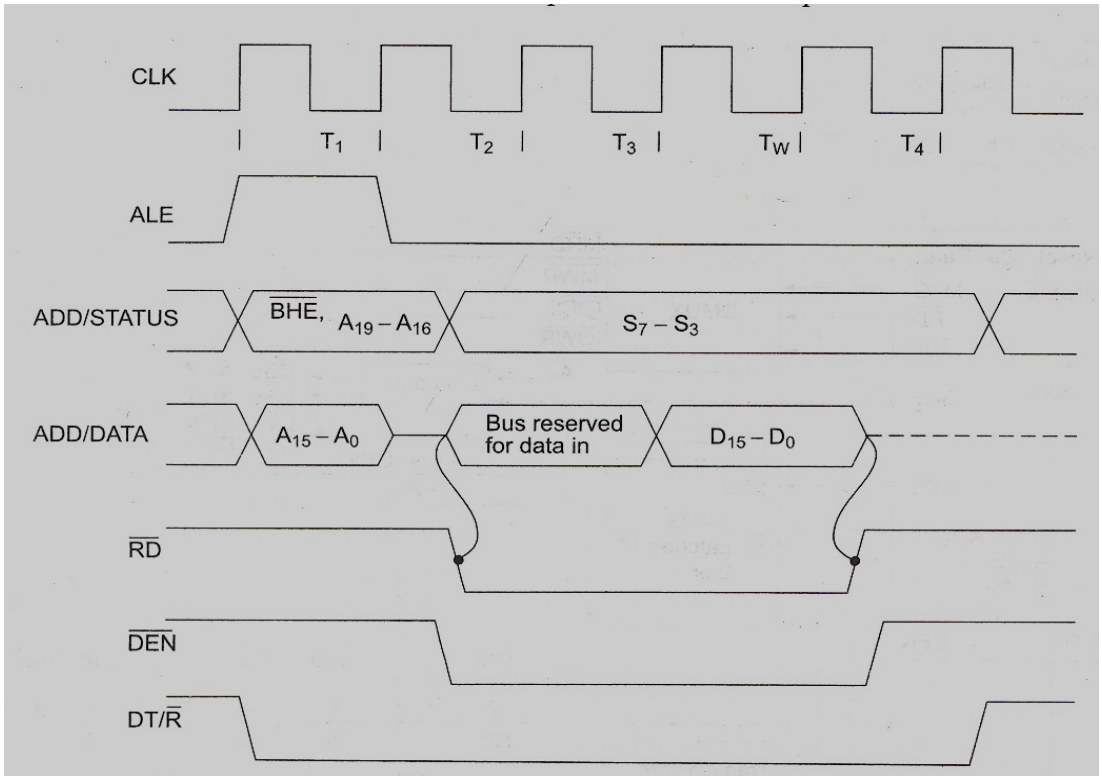
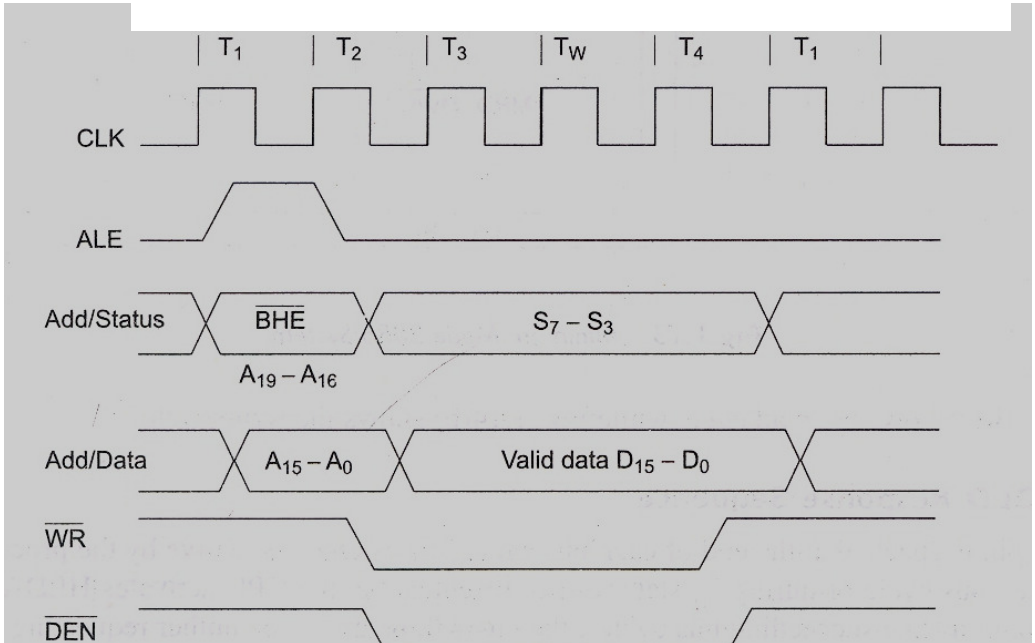


Fig.1.2. Read Cycle Timing Diagram for Minimum Mode



HOLD Response Sequence

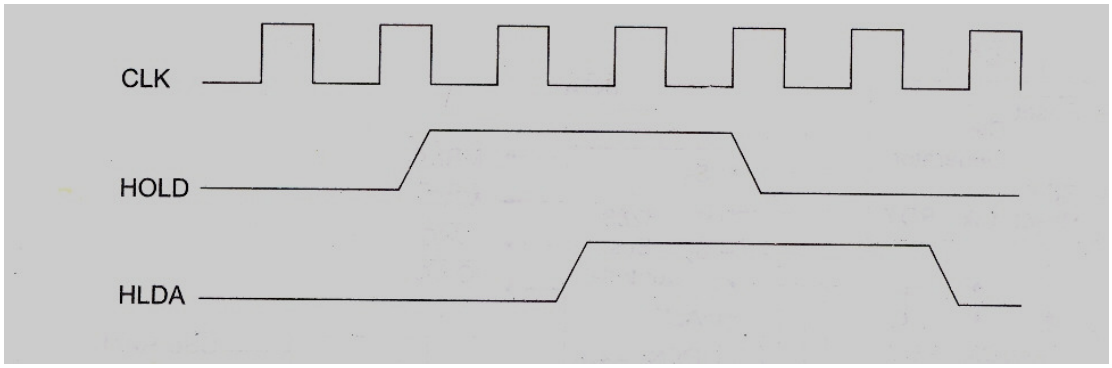


Fig 1.4. Bus Request and Bus Grant Timings in Minimum Mode System

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 1.4.

MAXIMUM MODE 8086 SYSTEM AND TIMINGS

In the maximum mode, the 8086 is operated by strapping the MN/MX* pin to ground. In this mode, the processor derives the status signals S_2*, S_1* and 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 general system organization is as shown in the fig1.1

The basic functions of the bus controller chip IC8288, is to derive control signals like RD* and WR* (for memory and I/O devices), DEN*, DT/R*, ALE, etc. using the information made available by the processor on the status lines. The bus controller chip has input lines S_2*, S_1* and S_0* and CLK. These inputs to 8288 are driven by the CPU. It derives the outputs ALE, DEN*, DT/R*, MWTC*, AMWC*, IORC*, IOWC* and AIOWC*. The AEN*, IOB and CEN pins are specially useful for multiprocessor systems. AEN* and IOB are generally grounded. CEN pin is usually tied to +5V.

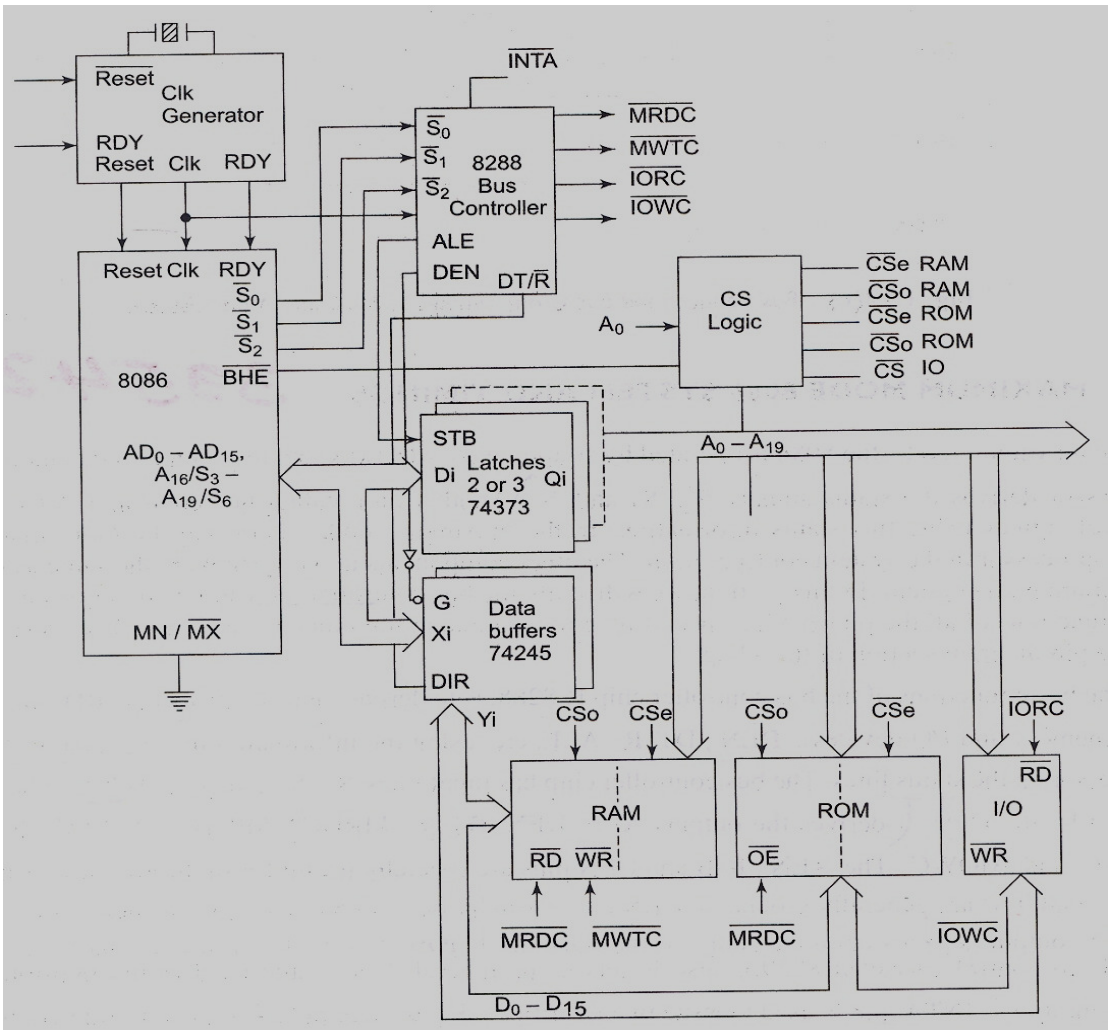


Fig 1.1 Maximum Mode 8086 System

The significance of the MCE/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. INTA* pin is used to issue two interrupt acknowledge pulses to the interrupt controller or to an interrupting device.

IORC*, 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 MRDC*, 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 from or to the bus. For both of these write command signals, the advanced signals namely AIOWC* and AMWTC* are available. They also serve the same purpose, but are activated one clock cycle earlier than the IOWC* and MWTC* signals, respectively. The maximum mode system is shown in fig. 1.1.

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. The fig. 1.2 shows the maximum mode timings for the read operation while the fig. 1.3 shows the same for the write operation.

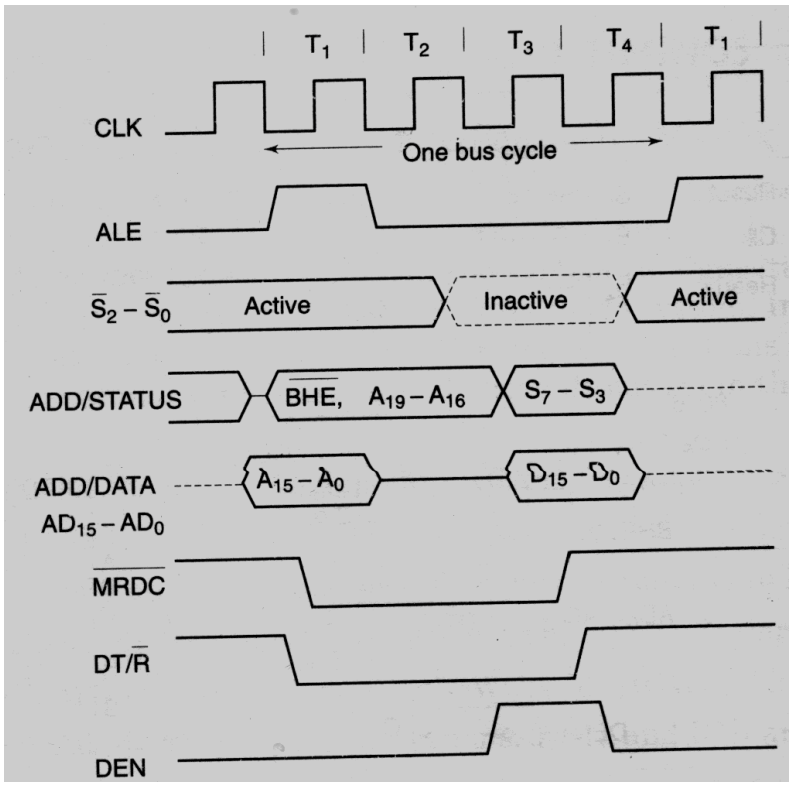
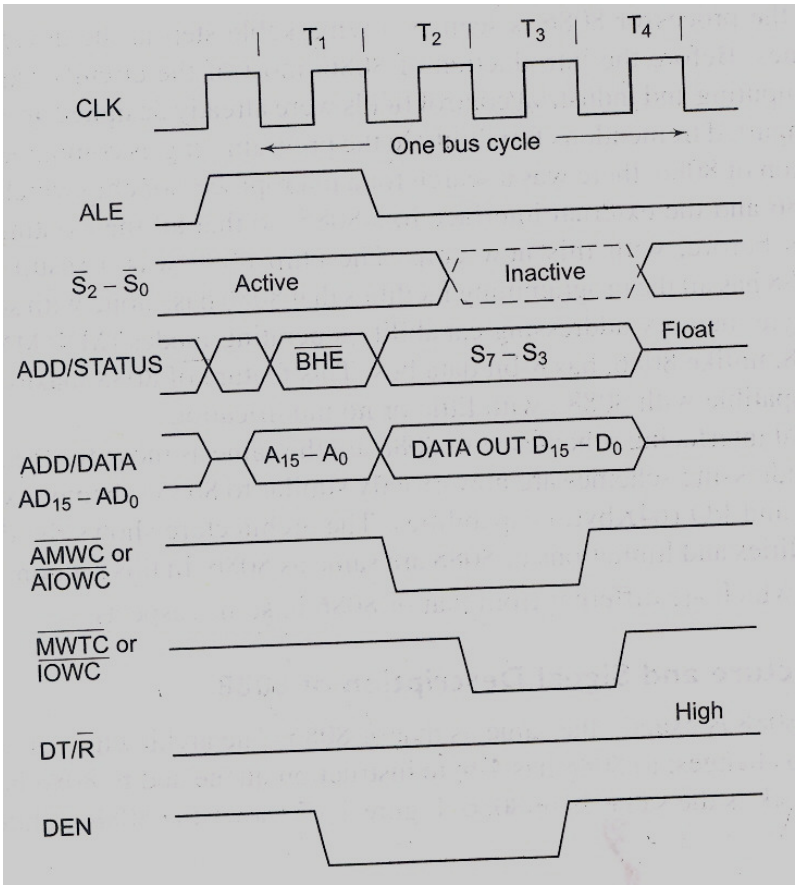


Fig. 1.2 Memory Read Timing in Maximum Mode



Timings for RQ Fig. 1.3 Memory Write Timing in Maximum Mode

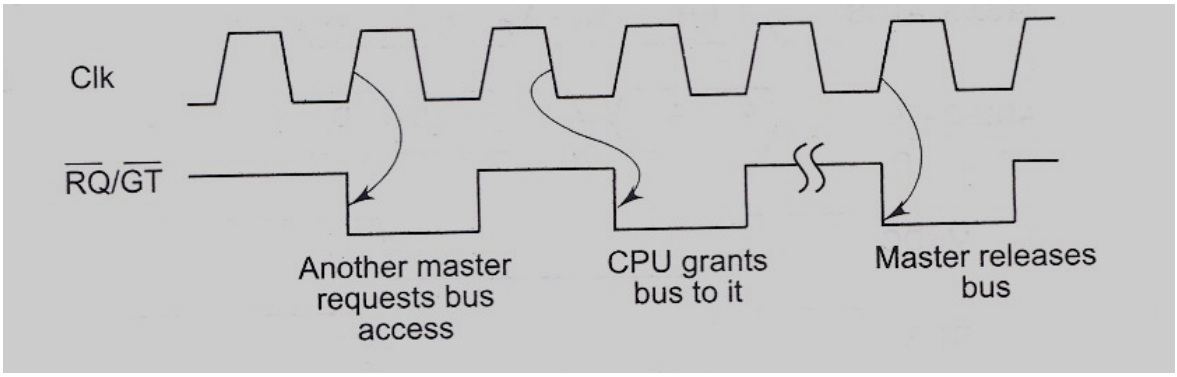


Fig1.4. RQ*/GT* Timings in Maximum Mode