

UNIT-II

8086 SYSTEM BUS STRUCTURE

The 8086 Microprocessor is a 16-bit CPU available in 3 clock rates, i.e. 5, 8 and 10MHz, packaged in a 40 pin Cerdip or plastic package. The 8086 Microprocessor operates in single processor or multiprocessor configurations to achieve high performance. The pin configuration is as shown in fig1. Some of the pins serve a particular function in minimum mode (single processor mode) and others function in maximum mode (multiprocessor mode) configuration.

The 8086 signals can be categorized in three groups. The first are the signals having common functions in minimum as well as maximum mode, the second are the signals which have special functions in minimum mode and third are the signals having special functions for maximum mode

2.1 8086 signals

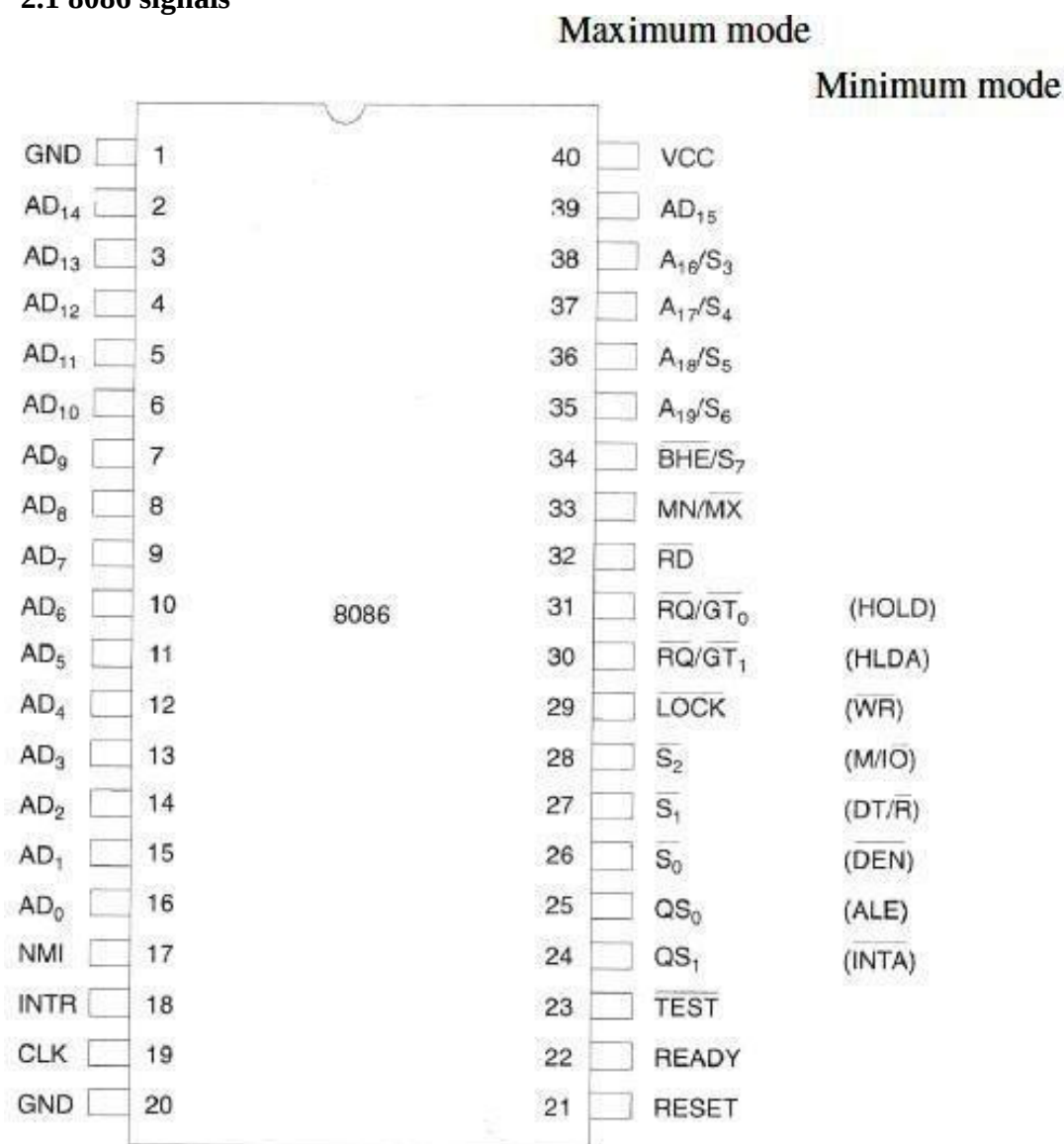


Fig.2.1 Bus signals

[Source: Advanced Microprocessors and Microcontrollers by A.K Ray & K.M. Bhurchandi]

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 T1 state, while the data is available on the data bus during T2, T3, TW and T4. Here T1, T2, T3, T4 and TW are the clock states of a machine cycle. TW is a wait state. These lines are active high and float to a tristate during interrupt acknowledge and local bus hold acknowledge cycles.

A19/S6, A18/S5, A17/S4, A16/S3: These are the time multiplexed address and status lines. During T1, these are the most significant address lines or memory operations. During I/O operations, these lines are low. During memory or I/O operations, status information is available on those lines for T2, T3, TW and T4. The status of the interrupt enable flag bit (displayed on S5) is updated at the beginning of each clock cycle. The S4 and S3 combinedly indicate which segment register is presently being used for memory accesses as shown in Table 2.1.

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

Table 2.1 Bus High Enable / status [Source: Advanced Microprocessors and Microcontrollers by A.K Ray & K.M. Bhurchandi]

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

BHE/S7-Bus High Enable/Status: The bus high enable signal is used to indicate the transfer of data over the higher order (D15-D8) data bus as shown in Table 2.1. It goes low for the data transfers over D15-D8 and is used to derive chip selects of odd address memory bank or peripherals. BHE is low during T1 for read, write and interrupt acknowledge cycles, when- ever a byte is to be transferred on the higher byte of the data bus. The status information is available during T2, T3 and T4. The signal is active low and is tristated during 'hold'. It is low during T1 for the first pulse of the interrupt acknowledge cycle.

Table 2.2 Bus high enable status [Source: Advanced Microprocessors and Microcontrollers by A.K Ray & K.M. Bhurchandi]

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

RD-Read: Read signal, when low, indicates the peripherals that the processor is performing a memory or I/O read operation. RD is active low and shows the state for T2, T3, TW of any read cycle. The signal remains tristated during the 'holdacknowledge'.

READY: This is the acknowledgement from the slow devices or memory that they have completed the data transfer. The signal made available by the devices is synchronized by the 8284A clock generator to provide ready input to the 8086. The signal is active high.

INTR-Interrupt Request: This is a level triggered input. This is sampled during the last clock

cycle of each instruction to determine the availability of the request. If any interrupt request is pending, the processor enters the interrupt acknowledge cycle. This can be internally masked by resetting the interrupt enable flag. This signal is active high and internally synchronized.

TEST: This input is examined by a 'WAIT' instruction. If the TEST input goes low, execution will continue, else, the processor remains in an idle state. The input is synchronized internally during each clock cycle on leading edge of clock.

NMI-Non-maskable Interrupt: This is an edge-triggered input which causes a Type2 interrupt. 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. It is an asymmetric square wave with 33% duty cycle. The range of frequency for different 8086 versions is from 5MHz to 10MHz.

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

MN/MX: The logic level at this pin decides whether the processor is to operate in either minimum (single processor) or maximum (multiprocessor) mode.

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

M/IO - Memory/IO: This is a status line logically equivalent to S2 in maximum mode. When it is low, it indicates the CPU is having an I/O operation, and when it is high, it indicates that the CPU is having a memory operation. This line becomes active in the previous T4 and remains active till final T4 of the current cycle. It is tristated during local bus "hold acknowledge".

INTA -Interrupt Acknowledge: This signal is used as a read strobe for interrupt acknowledge cycles. In other words, when it goes low, it means that the processor has accepted the interrupt. It is active low during T2, T3 and TW of each interrupt acknowledge cycle.

ALE-Address latch Enable: This output signal indicates the availability of the valid address on the address/data lines, and is connected to latch enable input of latches. This signal is active high and is nevertristated.

DT /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'.

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 of T2 until the middle of T4. DEN is tristated during 'hold acknowledge' cycle.

HOLD, HLDA-Hold/Hold Acknowledge: When the HOLD line goes high, it indicates to the processor that another master is requesting the bus access. The processor, after receiving the HOLD request, issues the hold acknowledge signal on HLDA pin, in the middle of the next clock cycle after completing the current bus (instruction) cycle. At the same time, the processor floats the local bus and control lines. When the processor detects the HOLD line low, it lowers the HLDA signal. HOLD is an asynchronous input and it should be externally synchronized.

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.

S₂, S₁, S₀ -Status Lines: These are the status lines which reflect the type of operation, being carried out by the processor. These become active during T₄ of the previous cycle and remain active during T₁ and T₂ of the current bus cycle. The status lines return to passive state during T₃ of the current bus cycle so that they may again become active for the next bus cycle during T₄. Any change in these lines during T₃ indicates the starting of a new cycle, and return to passive state indicates end of the bus cycle. These status lines are encoded in table 2.3.

Table 2.3. Status lines [Source: Advanced Microprocessors and Microcontrollers by A.K Ray & K.M. Bhurchandi]

S ₂	S ₁	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: This output pin indicates that other system bus masters will be prevented from gaining the system bus, while the LOCK signal is low. The 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 buscontroller.

QS₁, QS₀-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 2.4.

Table 2.4. Queue Status [Source: Advanced Microprocessors and Microcontrollers by A.K Ray & K.M. Bhurchandi]

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

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 bytelong 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 of the next instruction from memory, depending upon the queue status. Figure 1.6 explains the queue operation.

RQ/GT0, RQ/GT1-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 RQ/GT0 having higher priority than RQ/ GT1, 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 T4 (current) or T1 (next) clock cycle, a pulse one clock wide from 8086 to the requesting master, indicates that the 8086 has allowed the local bus to float and that it will enter the "hold acknowledge" state at next clock cycle. The CPU's bus interface unit is likely to be disconnected from the local bus of the system.
3. A one clock wide pulse from 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 in case of HOLD, and HLDA in minimum mode.

2.2 Basic configurations : Read Write Timing Diagram

✓ General Bus Operation

The 8086 has a combined address and data bus commonly referred as a time multiplexed address and data bus. The main reason behind multiplexing address and data over the same pins is the maximum utilization of processor pins and it facilitates the use of 40 pin standard DIP package. The bus can be demultiplexed using a few latches and transreceivers, whenever required.

Basically, all the processor bus cycles consist of at least four clock cycles. These are referred to as T1, T2, T3, T4. The address is transmitted by the processor during T1, It is present on the bus only for one cycle. The negative edge of this ALE pulse is used to separate the address and the data or status information.

In maximum mode, the status lines S0, S1 and S2 are used to indicate the type of operation. Status bits S3 to S7 are multiplexed with higher order address bits and the BHE signal. Address is valid during T1 while status bits S3 to S7 are valid during T2 through T4.

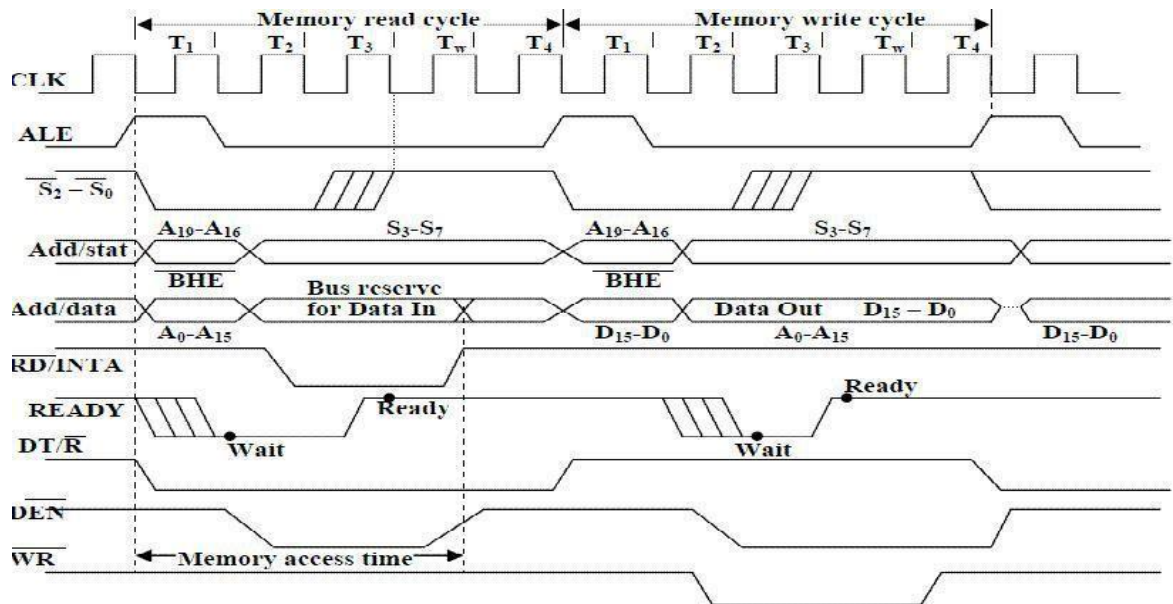


Fig.2.2. General Bus operation cycle [Source: Advanced Microprocessors and Microcontrollers by A.K Ray & K.M. Bhurchandi]

2.3 System Bus timings: 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, transceivers, clock generator, memory and I/O devices.

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

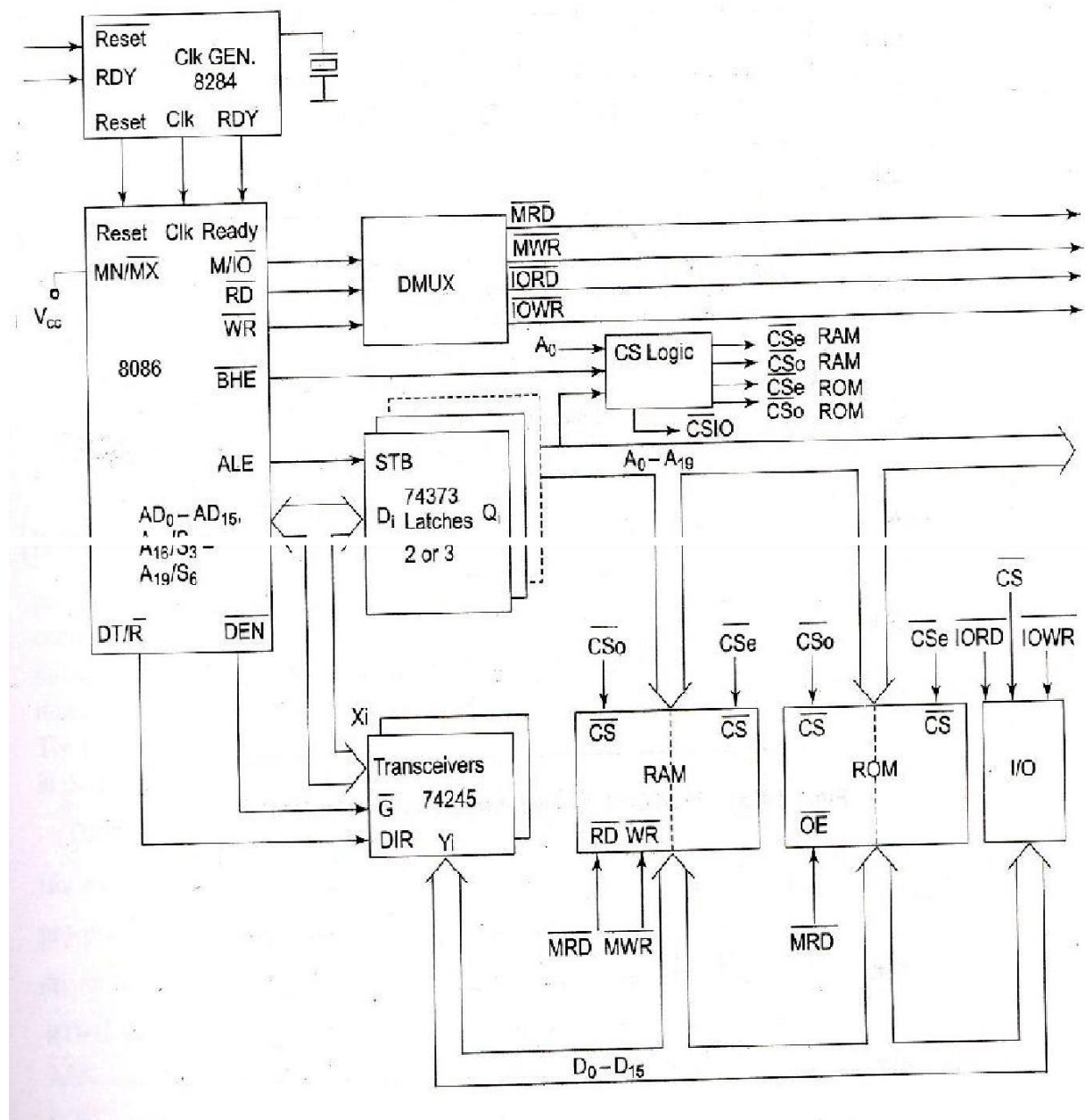


Fig 2.3. Minimum Mode 8086 System [Source: Advanced Microprocessors and Microcontrollers by A.K Ray & K.M. Bhurchandi]

A write cycle also begins with the assertion of ALE and the emission of the address. The M/IO* signal is again asserted to indicate a memory or I/O operation. In T2 after sending the address in T1 the processor sends the data to be written to the addressed location. The data remains on the bus until middle of T4 state. The WR* becomes active at the beginning of T2 (unlike RD* is somewhat delayed in T2 to provide time for floating). The BHE* and A0 signals are used to select the proper byte or bytes of memory or I/O word to be read or written. The M/IO*, RD* and WR* signals indicate the types of data transfer as specified in Table

Table 2.5 Read write cycle [Source: Advanced Microprocessors and Microcontrollers by A.K Ray & K.M. Bhurchandi]

M/I/O	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

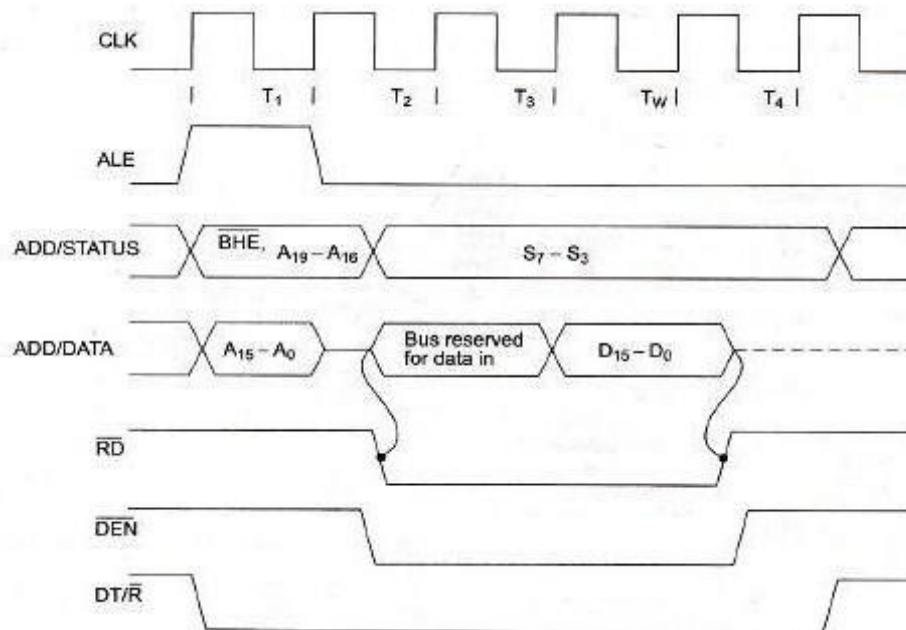


Fig. 2.4 Read cycle timing diagram for minimum mode [Source: Advanced Microprocessors and Microcontrollers by A.K Ray & K.M. Bhurchandi]

HOLD Response Sequence

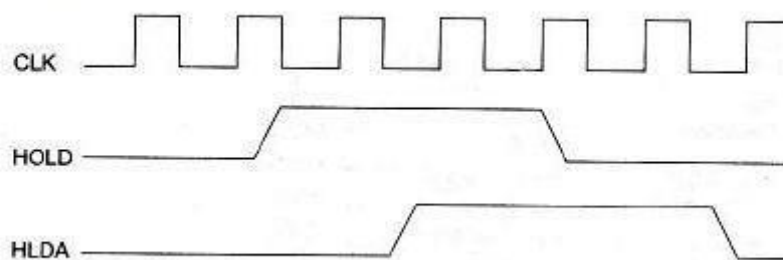
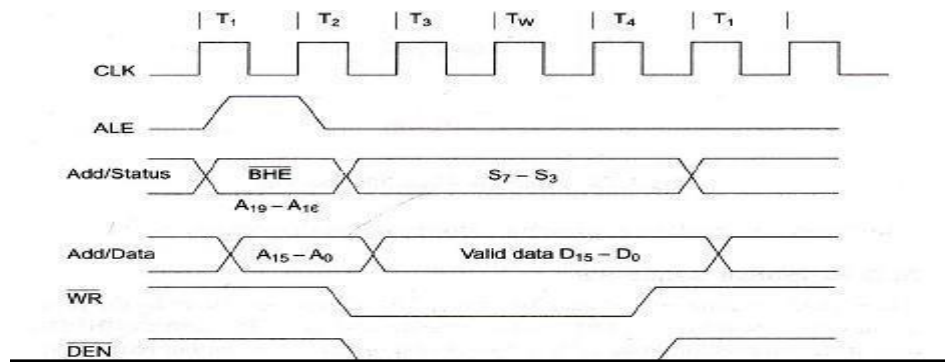


Fig 2.5 Bus request and bus grant timings in minimum mode system [Source: Advanced Microprocessors and Microcontrollers by A.K Ray & K.M. Bhurchandi]



[Source: Advanced Microprocessors and Microcontrollers by A.K Ray & K.M. Bhurchandi]

2.3 System Design using 8086: 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 S₂*, S₁* and S₀*. 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 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 S₂*, S₁* and S₀* and CLK. The CPU drives these inputs to 8288. It derives the outputs ALE, DEN*, DT/R*, MWTC*, AMWC*, IORC*, IOWC* and AIOWC*. The AEN*, IOB and CEN pins are especially useful for multiprocessor systems. AEN* and IOB are generally grounded. CEN pin is usually tied to +5V.

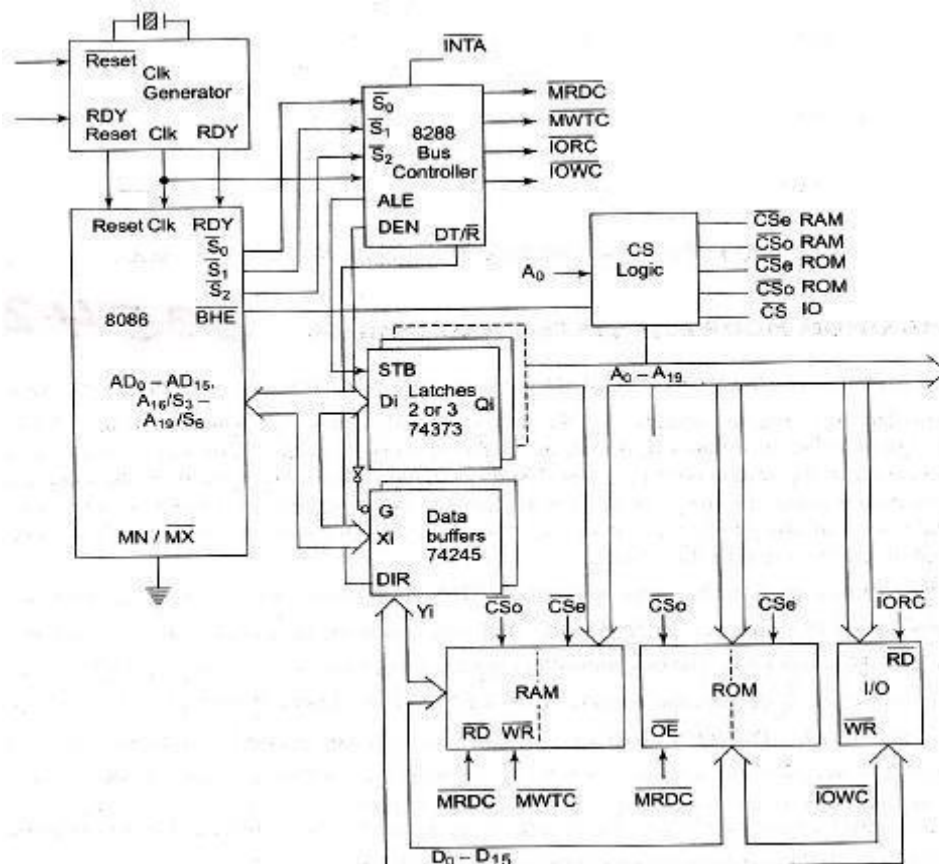


Fig. 2.6 Maximum mode configuration [Source: Advanced Microprocessors and Microcontrollers by A.K Ray & K.M. Bhurchandi]

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. 2.1.

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 T1, 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 writeoperation.

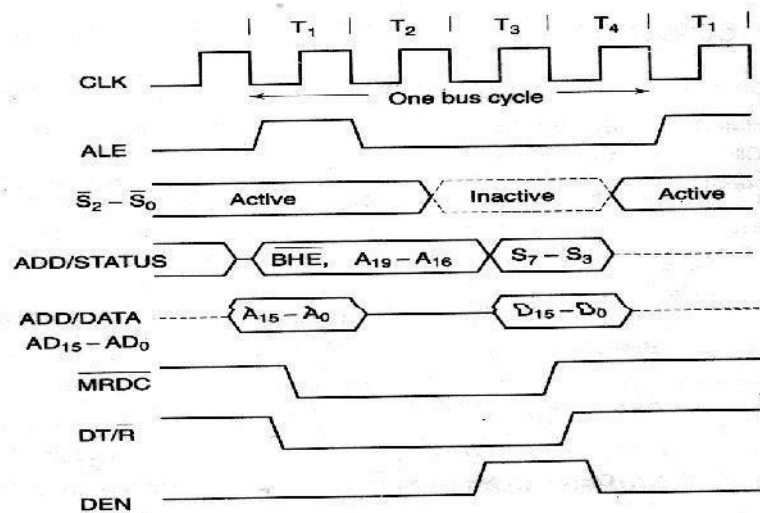


Fig.2.7 Memory read cycle [Source: Advanced Microprocessors and Microcontrollers by A.K Ray & K.M. Bhurchandi]

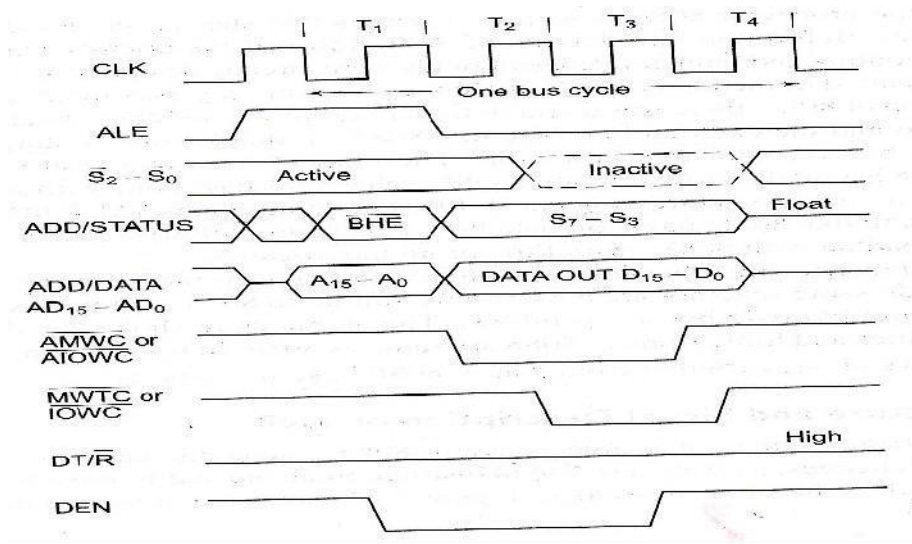


Fig.2.8 Memory write cycle [Source: Advanced Microprocessors and Microcontrollers by A.K Ray & K.M. Bhurchandi]

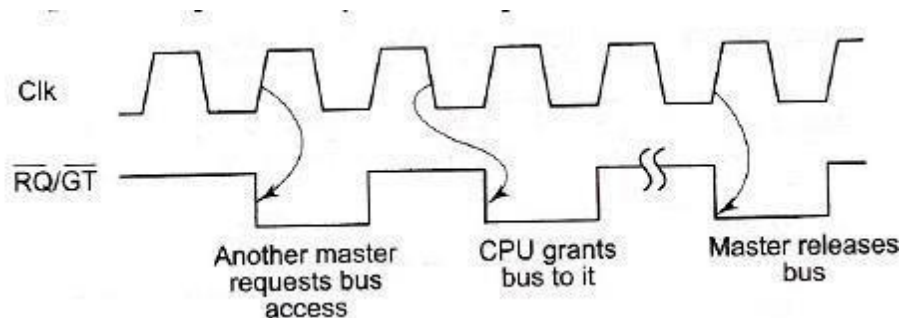


Fig 2.9 $\overline{RG}/\overline{GT}$ Timings in maximum mode [Source: Advanced Microprocessors and Microcontrollers by A.K Ray & K.M. Bhurchandi]

2.5 I/O programming

On the 8086, all programmed communications with the I/O ports is done by the IN and OUT instructions defined in Fig. 6-2.

✓ IN and OUT instructions

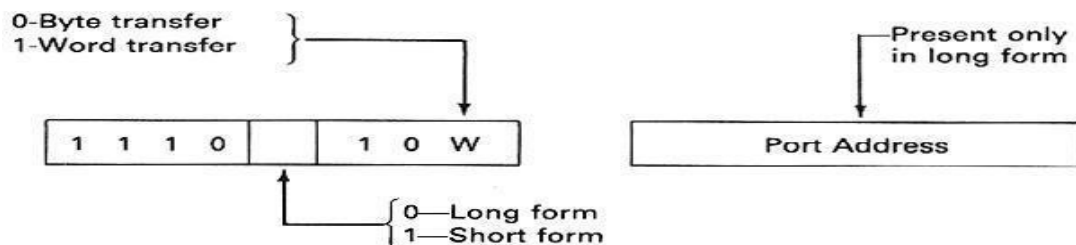
Name	Mnemonic and Format	Description
Input		
Long form, byte	IN AL, PORT	(AL) <- (PORT)
Long form, word	IN AX, PORT	(AX) <- (PORT+1: PORT)
Short form, byte	IN AL, DX	(AL) <- ((DX))
Short form, word	IN AX, DX	(AX) <- ((DX) + 1: (DX))
Output		
Long form, byte	OUT PORT, AL	(PORT) <- (AL)
Long form, word	OUT PORT, AX	(PORT+1: PORT) <- (AX)
Short form, byte	OUT DX, AL	((DX)) <- (AL)
Short form, word	OUT DX, AX	((DX)+1: (DX)) <- (AX)

Note: PORT is a constant ranging from 0 to 255

Flags: No flags are affected

Addressing modes: Operands are limited as indicated above.

If the second operand is DX, then there is only one byte in the instruction and the contents of DX are used as the port address. Unlike memory addressing, the contents of DX are not modified by any segment register. This allows variable access to I/O ports in the range 0 to 64K. The machine language code for the IN instruction is:



Although AL or AX is implied as the first operand in an IN instruction, either AL or AX must be specified so that the assembler can determine the W-bit.

Similar comments apply to the OUT instruction except that for it the port address is the destination and is therefore indicated by the first operand, and the second operand is either AL or AX. Its machine code is:



Note that if the long form of the IN or OUT instruction is used the port address must be in the range 0000 to 00FF, but for the short form it can be any address in the range 0000 to FFFF (i.e. any address in the I/O address space). Neither IN nor OUT affects the flags.

The IN instruction may be used to input data from a data buffer register or the status from a status register. The instructions

IN AX, 28H

MOV DATA_WORD, AX

would move the word in the ports whose address are 0028 and 0029 to the memory location DATA_WORD.

2.6 Introduction to Multiprogramming

In order to adapt to as many situations as possible both the 8086 and 8088 have been given two modes of operation, the minimum mode and the maximum mode. The minimum mode is used for a small system with a single processor, a system in which the 8086/8088 generates all the necessary bus control signals directly (thereby minimizing the required bus control logic). The maximum mode is for medium-size to large systems, which often include two or more processors.

2.7 System Busstructure

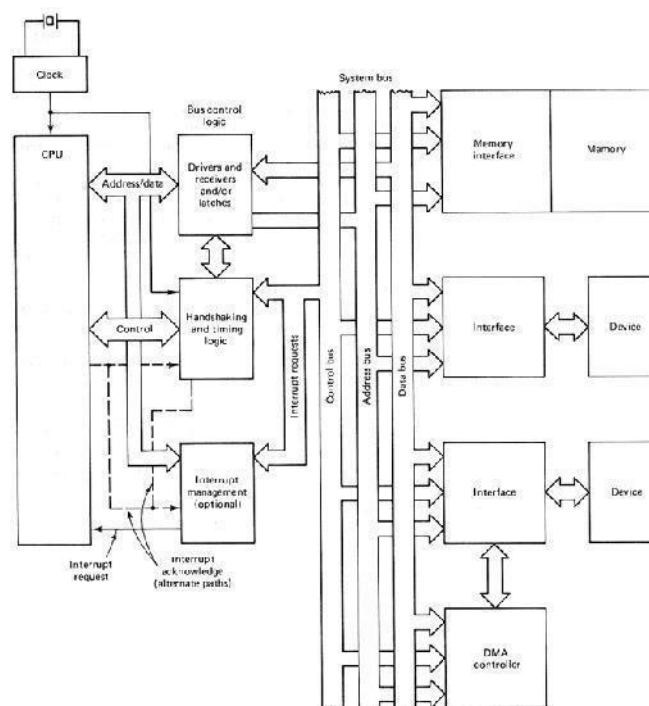


Fig.2.10 Typical system Bus architecture [Source: Advanced Microprocessors and Microcontrollers by A.K Ray & K.M. Bhurchandi]

2.8 Multiprocessor Systems

Multiprocessor Systems refer to the use of multiple processors that execute instructions simultaneously and communicate using mailboxes and semaphores

Maximum mode of 8086 is designed to implement 3 basic multiprocessor configurations:

1. Coprocessor(8087)
2. Closely coupled (dedicated I/O processor:8089)
3. Loosely coupled (Multibus)

Coprocessors and closely coupled configurations are similar - both the CPU and the external processor share:

- Memory
- I/O system
- Bus & bus control logic
- Clock generator

2.9 Coprocessor configurations

Coprocessor Configuration:

WAIT instruction allows the processor to synchronize itself with external hardware, eg., waiting for 8087 math co-processor.

When the CPU executes WAIT
waiting state.

TEST input is asserted (low), the waiting state is completed and execution will resume.

ESC instruction:

ESC opcode, operand, opcode: immediate value recognizable to a coprocessor as an instruction opcode

Operand: name of a register or a memory address (in any mode)

When the CPU executes the ESC instruction, the processor accesses the memory operand by placing the address on the address bus.

If a coprocessor is configured to share the system bus, it will recognize the ESC instruction and therefore will get the opcode and the operand

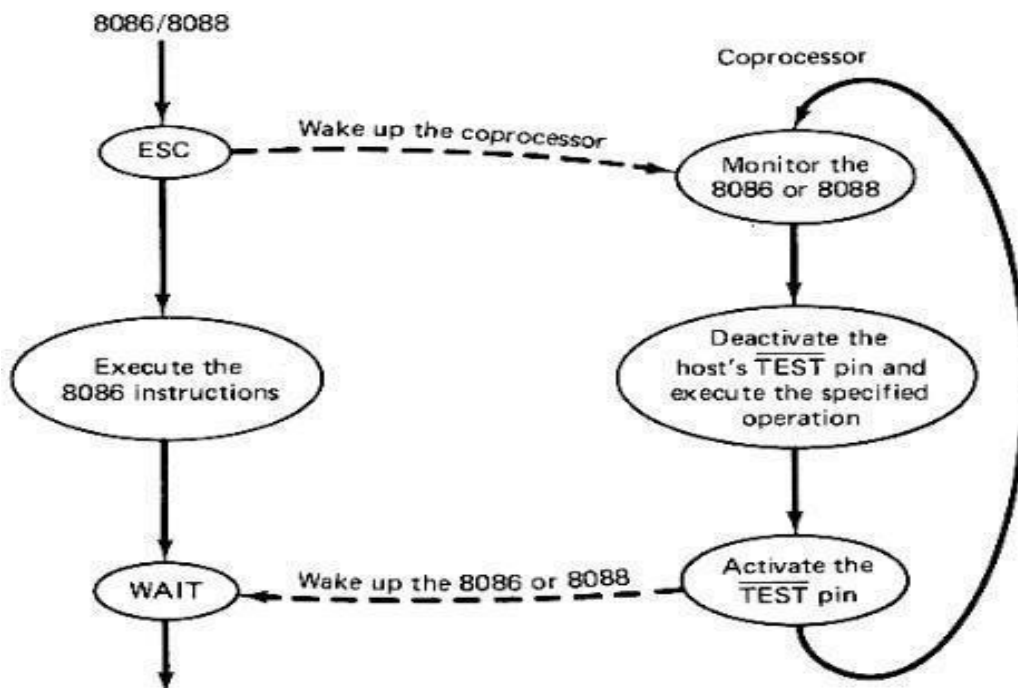


Fig. 2.11 Synchronisation between the 8086 and its coprocessor

[Source: Yu-Cheng Liu, Glenn A. Gibson, "Microcomputer Systems: The 8086 / 8088 Family - Architecture, Programming and Design", Second Edition, Prentice Hall of India, 2007]

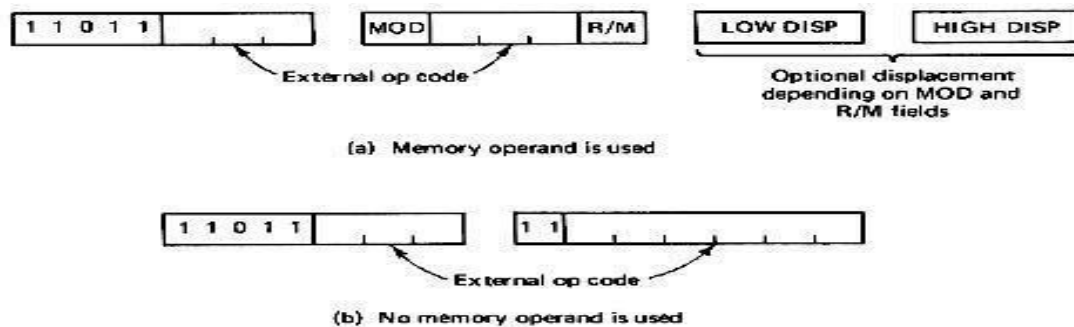


Fig 2.12 Machine code formats for the ESC instruction[Source:Yu-Cheng Liu, Glenn A.Gibson, “Microcomputer Systems: The 8086 / 8088 Family - Architecture, Programming and Design”, Second Edition, Prentice Hall of India,2007]

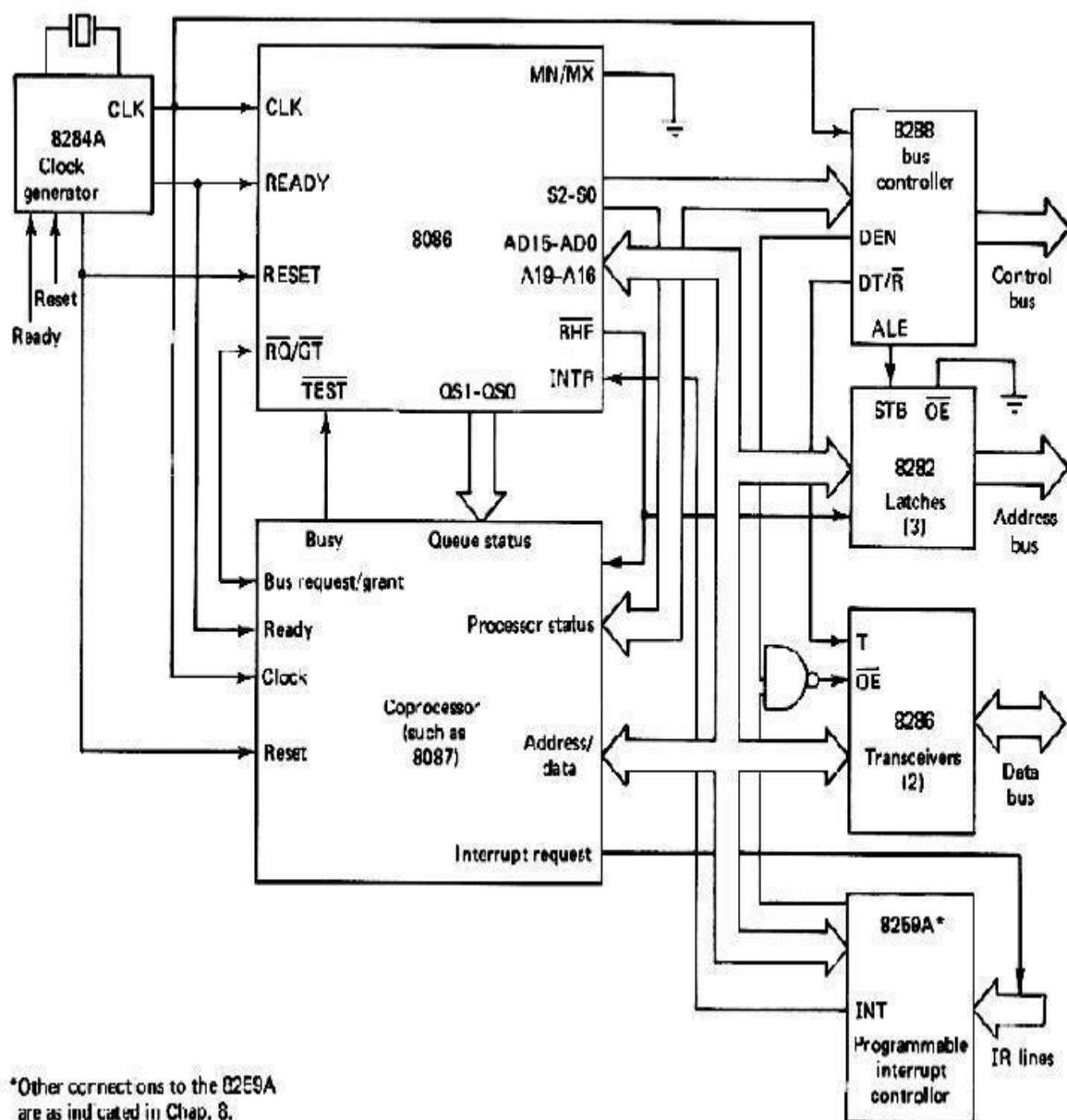


Fig.2.13 Coprocessor configuration

[Source:Yu-Cheng Liu, Glenn A.Gibson, “Microcomputer Systems: The 8086 / 8088 Family - Architecture, Programming and Design”, Second Edition, Prentice Hall of India,2007]

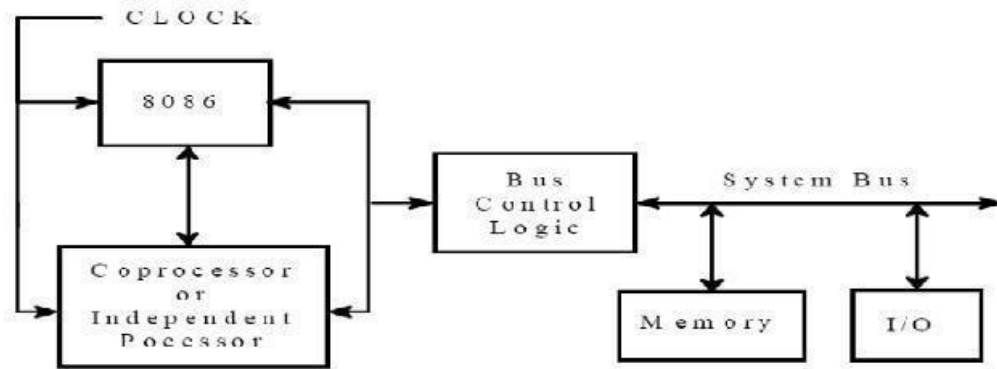


Fig.2.14 closely coupled configuration [Source:Yu-Cheng Liu, Glenn A.Gibson, “Microcomputer Systems: The 8086 / 8088 Family - Architecture, Programming and Design”, Second Edition, Prentice Hall of India,2007]

- ✓ Coprocessor cannot take control of the bus, it does everything through the CPU
- ✓ Closely Coupled processor may take control of the bus independently - 8089 shares CPU's clock and bus control logic.
- ✓ communication with host CPU is by way of shared memory
- ✓ host sets up a message (command) in memory
- ✓ independent processor interrupts host on completion
- ✓ Two 8086's cannot be closely coupled

2.10 Closely Coupled Configuration

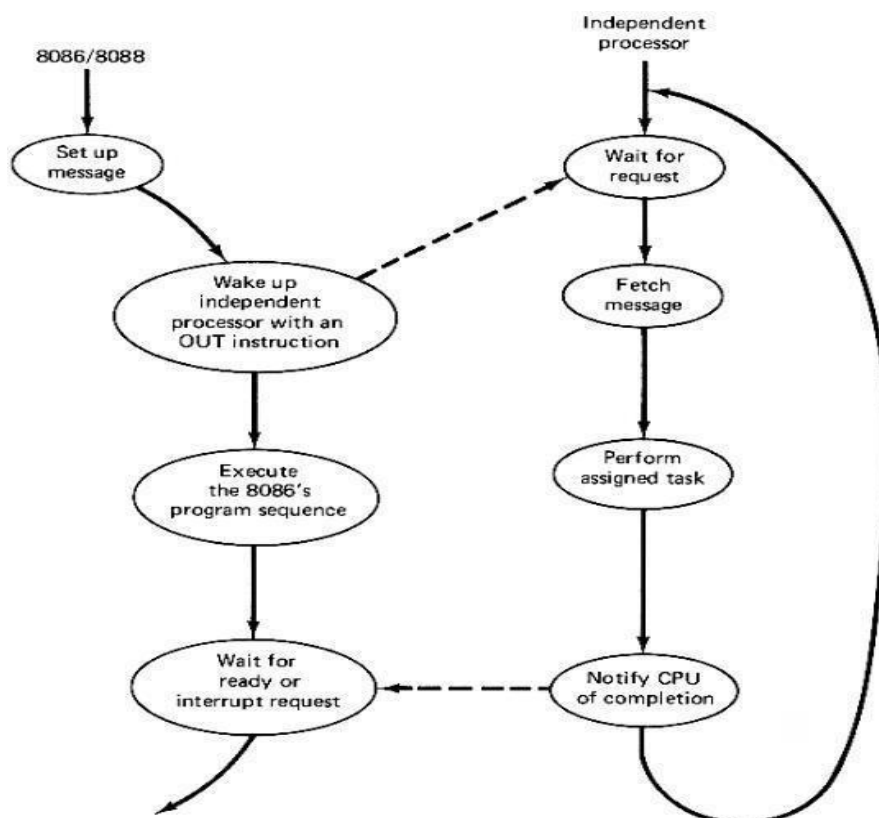
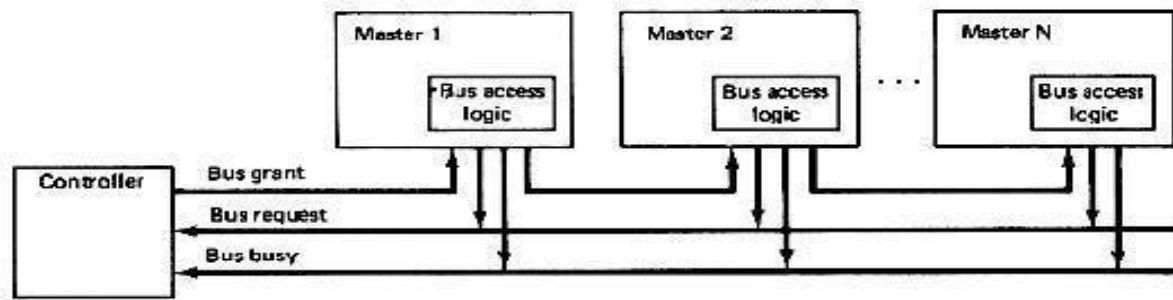
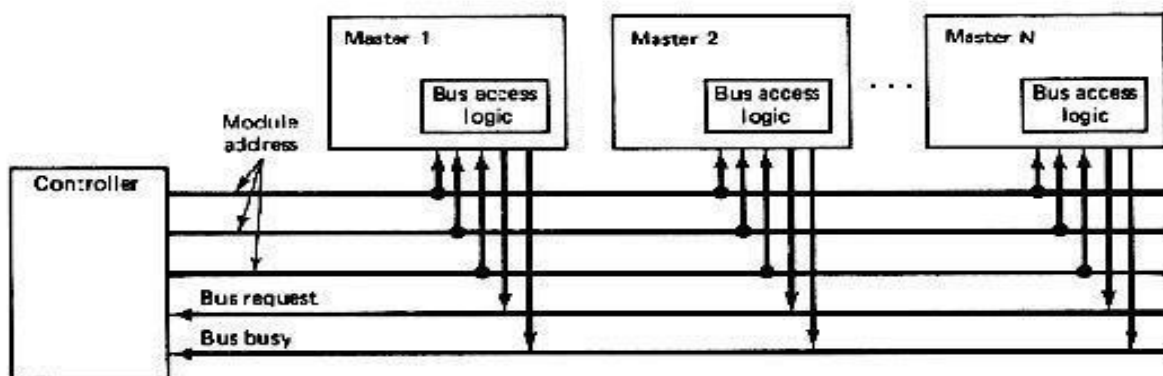


Fig.2.15 Interprocessor communication through shared memory

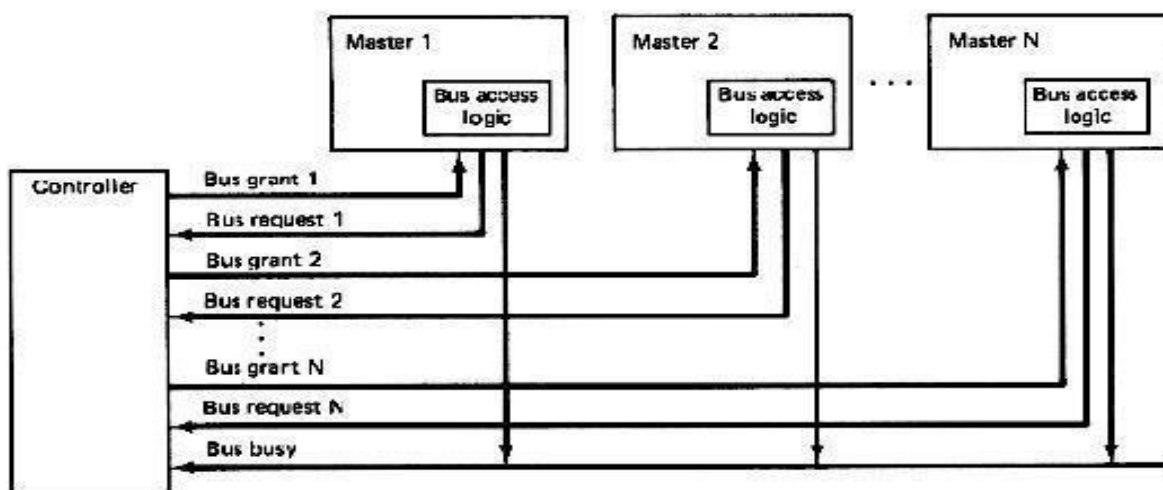
[Source:Yu-Cheng Liu, Glenn A.Gibson, “Microcomputer Systems: The 8086 / 8088 Family - Architecture, Programming and Design”, Second Edition, Prentice Hall of India,2007]



(a) Daisy chain method



(b) Polling method



(c) Independent requests method

Fig.2.16 Bus allocation schemes [Source:Yu-Cheng Liu, Glenn A.Gibson, “Microcomputer Systems: The 8086 / 8088 Family - Architecture, Programming and Design”, Second Edition, Prentice Hall of India,2007]

Loosely Coupled Configurations:

A loosely coupled configuration provides the following advantages:

1. High system throughput can be achieved by having more than one CPU.
2. The system can be expanded in a modular form. Each bus master module is an independent unit and normally resides on a separate PC board. Therefore, a bus master module can be added or removed without affecting the other modules in the system.
3. A failure in one module normally does not cause a breakdown of the entire system and the faulty module can be easily detected and replaced.

4. Each bus master may have a local bus to access dedicated memory or I/O devices so that a greater degree of parallel processing can be achieved. More than one bus master module may have access to the shared system bus.

Extra bus control logic must be provided to resolve the bus arbitration problem.

The extra logic is called bus access logic and it is its responsibility to make sure that only one bus master at a time has control of the bus.

Simultaneous bus requests are resolved on a **priority basis**: There are three schemes for establishing priority:

1. Daisy chaining.
2. Polling.
3. Independent requesting.

2.11 Introduction to Advanced processors: 80286 Microprocessor

2.11.1 Salient Features of 80286

✓ The 80286 is the first member of the family of advanced microprocessors with memory management and protection abilities. The 80286 CPU, with its 24-bit address bus is able to address 16 Mbytes of physical memory. Various versions of 80286 are available that runs on 12.5 MHz, 10 MHz and 8 MHz clock frequencies. 80286 is upwardly compatible with 8086 in terms of instruction set.

✓ 80286 has two operating modes namely real address mode and virtual address mode. In real address mode, the 80286 can address up to 1 Mb of physical memory address like 8086. In virtual address mode, it can address up to 16 Mb of physical memory address space and 1 GB of virtual memory address space.

✓ The instruction set of 80286 includes the instructions of 8086 and 80186. 80286 has some extra instructions to support operating system and memory management. In real address mode, the 80286 is object code compatible with 8086. In protected virtual address mode, it is source code compatible with 8086. The performance of 80286 is five times faster than the standard 8086.

2.11.1 Need for Memory Management

The part of main memory in which the operating system and other system programs are stored is not accessible to the users. It is required to ensure the smooth execution of the running process and also to ensure their protection. The memory management which is an important task of the operating system is supported by a hardware unit called memory management unit.

Swapping in of the Program

Fetching of the application program from the secondary memory and placing it in the physical memory for execution by the CPU.

Swapping out of the executable Program

Saving a portion of the program or important results required for further execution back to the secondary memory to make the program memory free for further execution of another required portion of the program.

2.11.2 Concept of Virtual Memory

Large application programs requiring memory much more than the physically available 16 Mbytes of memory, may be executed by dividing it into smaller segments. Thus for the user, there exists a very large logical memory space which is not actually available. Thus there exists a virtual memory which does not exist physically in a system. This complete process of virtual memory management is taken care of by the 80286 CPU and the supporting operating system.