The NDP 8087 adds 68 new instructions to the instruction set of 8086, all of which may lie interleaved in an 8086 ALP, as if they were its instructions. The execution of 8087 instructions is transparent to the programmer. These instructions are fetched by 8086 but are executed by 8087. Whenever the 8086 comes across 8087 instruction, it executes the ESCAPE instruction code to pass over the instruction opcode and control of the local bus to 8087. The additional instructions supported by 8087 can be categorized into the following types.

FCSH This instruction changes the sign of the content of the stack top.

The instructions FADD, FSUB, FMUL and FDIV are available with their different options decided by the opcode bits D, P and R. The D-bit decides the source and destination operands as in the case of 8086. The P-bit indicates whether an execution is followed by a pop operation. The R-bit indicates the reverse mode. This is valid only in case of FSUB and FDIV instructions. If R is 0, the destination operand is either subtracted from or divided by the source operand. If R is 1, the source operand is either subtracted from or divided by the destination operand. The result is stored in the destination operand. This definition of R is exactly opposite if D = 1, because if D = 1, it interchanges the source and destination operands.

Transcendental Instructions The 8087 provides five instructions for transcendental calculations described as follows. The operands usually are ST(0) and ST(1) or only ST(0).

FPTAN This instruction calculates the partial tangent of an angle θ, where θ must be in the range from $0 \leq \theta < 90°$. The value of θ must be stored at the stack top (stack top is the implicit operand). The result is given in the form of a ratio of ST/ST(1). If the result is out of capacity of the destination operand, an invalid error occurs.

FPATAN This instruction calculates the arc tangent (inverse tangent) of a ratio ST/ST(1). The stack is popped at the end of the execution and the result (θ) is stored on the top of stack. The contents of ST and ST(1) should follow the inequality,

$$0 \leq ST(1) < ST < 00$$

F2XMI This instruction calculates the expression (2x – 1). The value of x is stored at the top of the stack. The result is stored back at the top of the stack.

FLY2X This instruction calculates the expression 'ST(1)*$\log_2$ ST'. A pop operation is carried out on the top of stack, and the result is stored at the top of stack. The ST must be in the range 0 to +∞, while the ST(1) must be in the range –∞ to +∞.

FLY2XP1 This instruction is used to calculate the expression 'ST(1)*$\log_2$ [(ST)+1]'. The result is stored back on the stack top after a pop operation. The value of |ST| must lie between 0 and $(1-2^{1/2}/2)$ and the value of ST(1) must be between – ∞ to +∞.

Comparison Instructions All the comparison instructions compare the operands and modify the condition code flags, as shown in Tables 8.4 (a) and (b). The instructions available in 8087 for comparison are discussed as follows:

Table 8.4(a) *Condition Code Bits Definition*

Comparison	C_3	C_0
Stack Top > Source	0	0
Stack Top < Source	0	1
Stack Top = Source	1	0
Not Comparable	1	1

FLDZ	Load +0.0 to stack top.
FLD1	Load +1.0 to stack top.
FLDPI	Load π to stack top.
FLD2T	Load the constant $\log_2 10$ to stack top
FLDL2E	Load the constant $\log_2 e$ to stack top
FLDLG2	Load the constant $\log_{10} 2$ to stack top
FLDLN2	Load the constant $\log_e 2$ to stack top.

1. **Explain with the suitable diagram the co-processor configuration of multiprocessing system.**

2.1 Coprocessor configuration

In coprocessor configuration both the CPU and external processor share entire memory and the I/O subsystem. They also share same bus control logic and clock generator. Here the 8086 is the master and supporting processor is the slave. Please refer Fig. 22 which shows the coprocessor configuration.

8086 is supported by a coprocessor 8087 that extends the instruction set which allows the necessary special computation to be accomplished more efficiently. Interaction between CPU and coprocessor is as shown in Fig. 23. TEST = 0 causes CPU to be idle and it is used along with WAIT instruction in multiprocessing environment. The coprocessor does not require any extra logic other than that for maximum mode. Both 8086 and coprocessor execute their instruction from, the same program. An instruction to be executed by COP is indicated when an ESC instruction appears in the program. Only the host processor fetches the instruction, but COP also receives all these instructions and monitors the instruction sequence of the host. ESC instruction contains external opcode indicating what is to be done by COP and is decoded at the same time by both. The host however goes to next instruction or to fetch data from memory for the coprocessor and then to the next instruction. If the memory operand is greater than one word, then COP obtains the remaining words by stealing bus cycle. If the memory operand specified is proceeded by ESC instruction, the COP gets the data and continues its operation. The COP thus sends a busy signal to the host's $\overline{TEST}$ pin. As the host continues processing the instruction, the coprocessor performs operation specified by ESC instruction and this parallel operation continues. At this time the host should execute a WAIT instruction and wait until $\overline{TEST}$ is activated by the coprocessor.

Internal structure of 8087 NDP.

8087 numeric data processor (NDP) or math coprocessor is used to perform floating-point operations. It is a 40-pin DIP (dual-in-line package). As discussed in section 6.2, 8087 is called a coprocessor because it shares the local buses of the 8086/88 host processor.

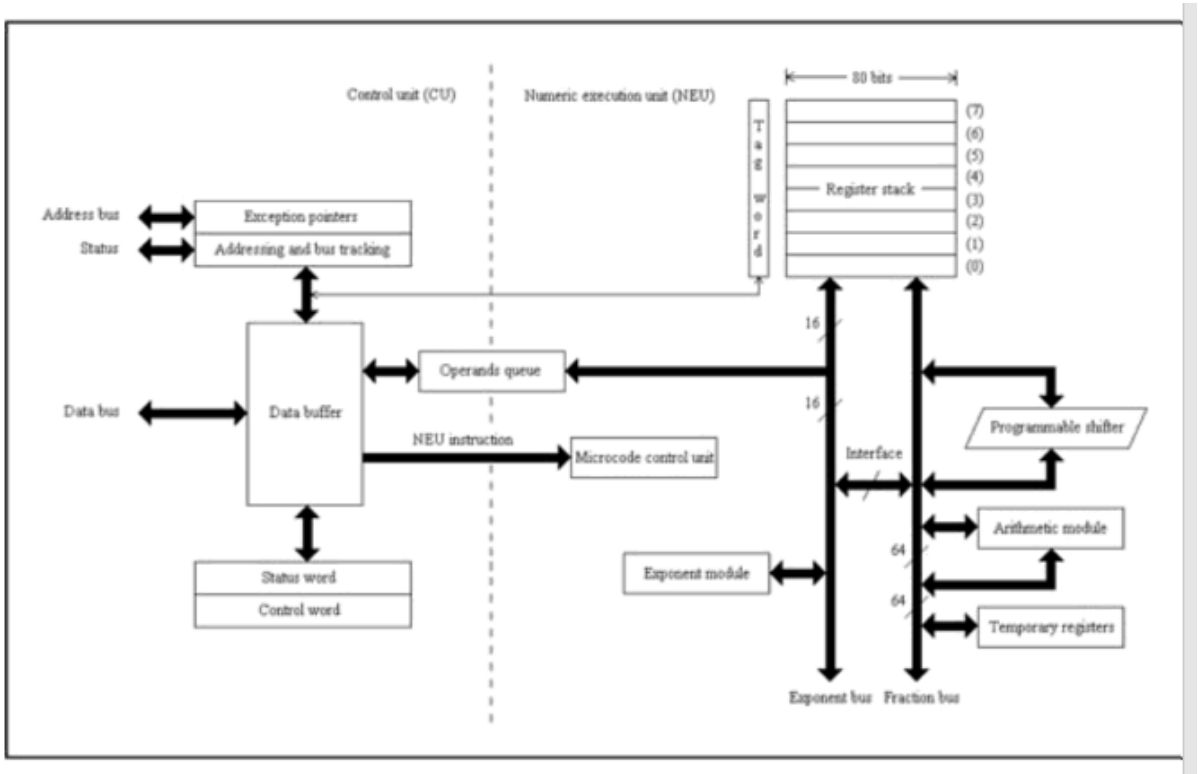
1. *Cycle arbitration:*

   Two or more processors should not be allowed to access a particular memory location or I/O device at the same time. Some means of arbitration is required.

2. *Multicycle usage:*

   Once a processor acquires control of a system resource, it may require multiple cycles to read/ write the data from/to the resource. During this time interval, no other processor should be allowed to access this resource. Some means of indicating the busy resources to other processors is required.

   Unlike in tightly coupled system (TCS), wherein there is a single bus master (8086/88), in loosely coupled system (LCS), each module may be the system bus master. If multiple bus masters access the system bus at the same time, it will result in contention. Hence, some bus arbitration scheme is required. The various system bus arbitration schemes in LCS are daisy chaining, polling and independent requesting.



Co Processor Configuration

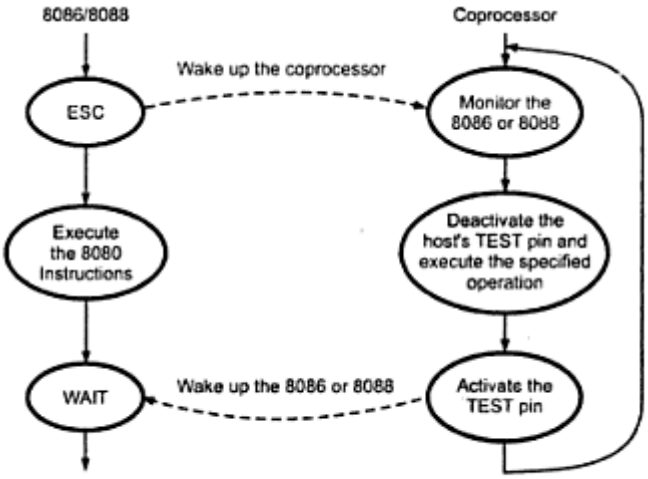


Fig. 2 Interaction between CPU and COP

When 8086 reads an 8087 instruction that needs data from memory or wants to send data to memory, the 8086 sends the memory address coded in the instruction and sends the appropriate memory-read or memory-write signals to transfer a word of data. In case of a memory-read, memory put the addressed data on the data bus. The 8087 then simply reads this word of the data bus and executes its instruction. The 8086 ignores the data word. Many times 8087 needs more than one word. In such situations, the 8086 outputs the address of the first data word on the address bus and outputs the appropriate memory-read or memory-write control signal. The 8087 reads the data word put on the data bus by memory or writes a data word to memory on the data bus. The 8087 then grabs the 20-bit physical address that was send by the 8086. To transfer the additional words, the 8087 then takes over the buses from the 8086. To take over the bus the 8087 sends a low-going pulse on its $\overline{RQ}/\overline{GT}_0$ pin. The 8086 responds to this by sending another low going pulse back to the $\overline{RQ}/\overline{GT}_0$ pin of the 8087 and by floating its buses. The 8087 then increments the address it grabbed during the first transfer and outputs the incremented address on the address bus. When the 8087 outputs a memory-read or memory-write signal, another data word will be transferred to or from the 8087. The 8087 continues this process until it receives/sends all the data words required by the instruction to or from memory. When 8087 finishes its data transfer, it sends another low-going pulse on $\overline{RQ}/\overline{GT}_0$ pin to let the 8086 know it can have the buses back again.

While the 8087 is executing an instruction it asserts BUSY pin high. When it completes its instruction it drops its BUSY pin low. Since the BUSY pin from 8087 is connected to the $\overline{TEST}$ pin of the 8086, the 8086 can check this pin to see if 8087 has executed its instruction. The 8086 checks $\overline{TEST}$ pin with the help of WAIT instruction.

The *control block* performs the bus control initialization for the I/O processor operation and provides pointers to the parameter block or data memory for channels 1 and 2. The *Channel Control Word* (CCW) is retrieved and analyzed. The CCW byte is decoded to determine channel operation.

The *parameter block* contains the address of the task block and acts as a message center between the I/O processor and CPU. Parameters information is passed from the CPU to I/O processor in this block to adapt the software interface to the peripheral device. It is also used for transferring data and status information between the I/O processor and CPU.

The task block holds the instructions for the respective channel. This block can reside on the local bus of the I/O processor and the I/O processor can operate concurrently with the CPU. The advantage of the communication between the processor, I/O processor and peripherals is that it allows for a very clear method for the operating system to handle I/O routines.

The 8089 has separate registers for its two different I/O channels as shown in Figure 9.106. Each channel has two sets of registers such as pointers and registers. The pointers are 20-bit registers usually used to address memory, but the registers are 16-bit general-purpose data registers. Each of the pointers excluding PP register has a tag bit. This bit is used to indicate that either the 20-bit register content is to be used or the lower 16-bit register content is to be used as the pointer.

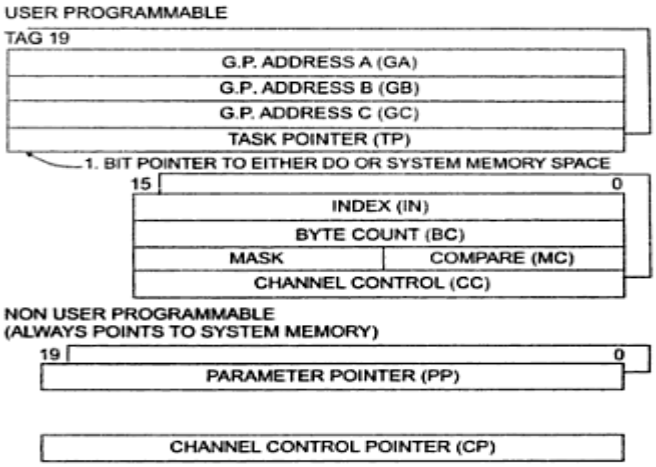
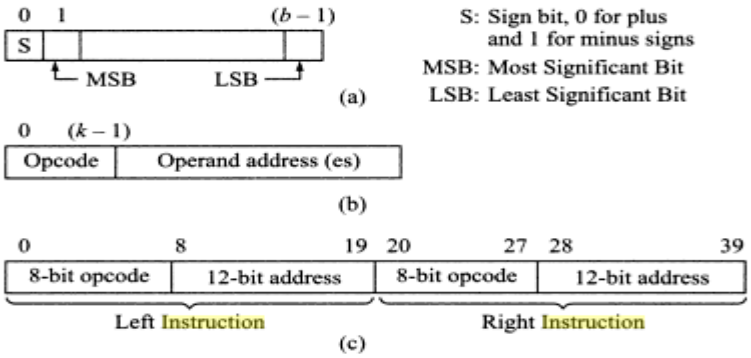
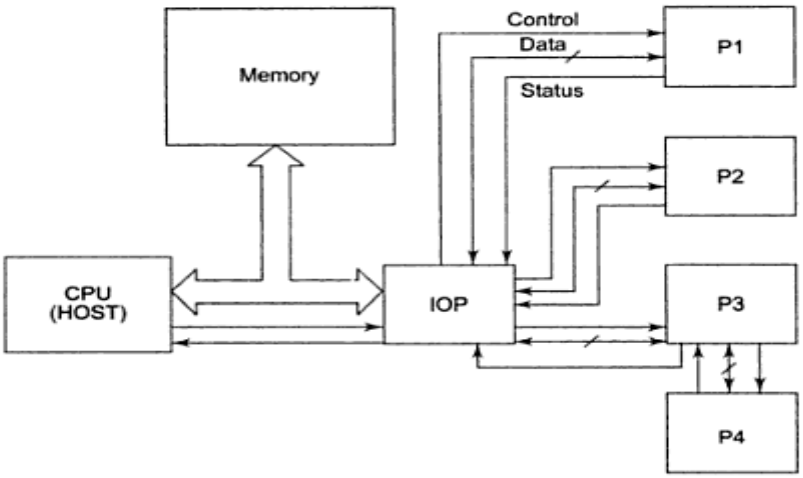


Fig. 9.106 *Register model*

Instruction set of IOP



Functional block diagram of the I/O Processor (8089)



The 8089 has two internal IO channels which can be programmed independently, to handle two separate IO tasks for the host CPU. The common ALU is shared by both the channels. The control unit derives the control signals required for the operation of the IOP channels. The IOP channels also share the common control unit. The bus interface and control unit handles all the bus activities, under the control of the control unit. The CCP, i.e. channel control pointer, available for programmers, automatically gets loaded with the 20-bit address of a memory table for the channel. This table is prepared by the host CPU to allot a task to the IOP. The address of the memory table for channel 2 is calculated by adding 8 to the contents of CCP, i.e. the memory table address for channel 1. For allotting a task to the IOP, the CPU first prepares the memory table in the memory that contains the details of the task. It then, asserts the Channel Attention signal (CA) and simultaneously selects one of the two channels using the SEL line. The SEL line is usually connected to the A_0 line of the host CPU, so that two consecutive addresses are assigned to the two channels.

The two channels are identical in their organization and may be used interchangeably for each other. Each of the channels has two sets of registers, viz. pointers and registers. The pointers are 20-bit registers normally used to address memory, while the registers are 16-bit general purpose data registers. Each of the pointers, except PP, has a tag bit assigned with each of them. This bit indicates whether the 20-bit register content is to be used (i.e. for addressing 1Mbyte system memory) or the lower 16-bit register content is to be used as the pointer (i.e. for addressing within 64 Kbyte local memory).

Communication with iop:

Ans: A processor which directly shares the local buses (data, address and control buses) of the host processor is called a *coprocessor*. 8087 and 8089 are called coprocessors because they share the local buses of the 8086/88 host processor.

Communication Co processor:

Ans: 8087 has no effective address (EA) generation capability. Hence, it relies on 8086/88 for the generation of EA. 8086/88 fetches the instructions in the program and puts them in its instruction queue. 8087 instructions are interspersed with the 8086/88 instructions in the program. 8087 reads the instructions prefetched into the instruction queue of 8086/88 and puts all of them into its own internal queue. Hence, the instructions prefetched into the instruction queue of 8086/88 as well as the instructions in the internal queue of 8087 are a mix of 8086/88 and 8087 instructions. However, the 8087 instructions are prefixed with the code 11011 so that they can be distinguished from 8086/88 instructions. When 8086/88 finds an 8086/88 instruction in its instruction queue, it executes it, whereas when it finds an 8087 instruction in its instruction queue, it executes a no operation (NOP) or reads one additional word from memory for the 8087. Similarly, when 8087 finds an 8087 instruction in its internal queue, it executes it, whereas when it finds an 8086/88 instruction in its internal queue, it executes a no operation (NOP). When 8087 is busy executing some instruction, it asserts its BUSY output, which is connected to the TEST# input of the 8086/88. A high on the TEST# input signals the 8086/88 not to request the 8087 to perform another calculation till TEST# is asserted low. Also, if WAIT instruction of 8086/88 or FWAIT instruction of 8087 is encountered, execution of the

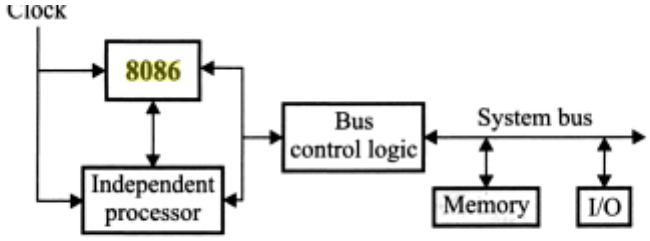


Figure 9.5 Block diagram of closely coupled configuration.

A closely coupled configuration is similar to a coprocessor configuration in the sense that both these configurations share the same memory, IO system, bus and bus control logic and clock generator with the host processor. But unlike coprocessor configuration (which consists of host processor and coprocessor), the closely coupled configuration consists of the host processor and an independent processor or may be an independent processor and a coprocessor. The independent processor, unlike a coprocessor, fetches and executes its own instruction. Coprocessor cannot take control of the bus; it does everything through the processor. Closely coupled processor may take control of the bus independently. Two 8086 microprocessors cannot be closely coupled. For instruction fetches and data references, the independent processor accesses the bus through the processors $\overline{RQ}/\overline{GT}$ lines. A simple block diagram of a closely coupled configuration is shown in Figure 9.5.

Minimum mode:

The pin functions which are unique for the minimum mode of operation are discussed in this section. The other pin functions are same as already discussed in the Section 2.2. For minimum mode of operation pin MN/$\overline{MX}$ is kept high. The pins 24-31 have unique pin functions for the minimum mode of operation as shown in Fig. 2.1. (a) which are as follows:

$\overline{INTA}$ *(Output). Pin 24 Interrupt Acknowledge.* It is active low. On receiving interrupt signal the processor issues an interrupt acknowledge signal through this line.

ALE (Output). Pin 25 Address Latch Enable. It is a high pulse issued by the processor during T_1 state of a bus cycle. The processor sends this signal to latch the address into the 8282/8283 address latch.

$\overline{DEN}$ *(Output). Pin 26 Data Enable.* In the minimum mode of operation this signal is issued by the processor to enable Intel 8286/8287 bus transceiver. It is an active low signal.

DT/$\overline{R}$ *(Output). Pin 27 Data Transmit/receive.* When the minimum mode system incorporates Intel 8286/8287 octal bus transceiver, this signal is required for the data flow control. When this signal is high, data are sent out and when it is low, data are received.

M/$\overline{IO}$ *(Output). Pin 28 Status Signal.* It is issued by the processor to distinguish a memory access from an I/O access. When this signal is high memory is accessed and when it is low, an I/O device is accessed.

$\overline{WR}$ *(Output). Pin 29 Write.* When this signal is low, the processor performs memory write or I/O write operation depending on the state of M/$\overline{IO}$ signal.

HLDA (Output). Pin 30 HOLD Acknowledge. On receiving HOLD signal, the processor issues a HOLD acknowledge signal through this pin. It is an active high signal.

HOLD (Input). Pin 31 When another device in the system wants to use the address and data bus, it sends a HOLD request to the processor through this line. It is an active high signal.

Maximum mode:

For the maximum mode of operation, the pin MN/$\overline{MX}$ is kept low and is grounded. The pins 24-31 have unique functions for the maximum mode of operation as shown in Fig.2.1(b) which are as follows:

QS_1, QS_0 (Output). Pin 24, 25 Queue Status. The queue status is valid during the clock cycle after which the queue operation is performed. These signals provide status to allow external tracking of the internal 8086 instruction queue. Table 2.3 shows the status of QS_0 and QS_1.

Table 2.3 Status of QS_0 and QS_1

QS_1	QS_0	*Characteristics*
0	0	No Operation
0	1	1st byte of opcode from queue
1	0	Empty queue
1	1	Subsequent byte from queue

$\overline{S}_0$, $\overline{S}_1$, $\overline{S}_2$ ***(Output). Pin 26, 27, 28 Status Signals.*** These signals are required by Intel 8288 bus controller to generate all memory and I/O access control signals. Table 2.4 shows the encoding of these signals.

$\overline{LOCK}$ ***(Output). Pin 29*** The LOCK, a prefix instruction, activates $\overline{LOCK}$ signal. It is an active low signal. It remains active until the completion of the next instruction. When it goes low, all interrupts are masked and HOLD request is not granted. Consequently other devices do not get control over the system bus while $\overline{LOCK}$ is low. In a multiprocessor system, the other devices are informed through this signal that they should not issue HOLD request.

$\overline{RQ}/\overline{GT}_0$, $\overline{RQ}/\overline{GT}_1$ ***(Bidirectional). Pin 30, 31 Local Bus Priority Control.*** An external device sends a hold request to the CPU through these lines to release the local bus at the end of CPU's current bus cycle. $\overline{RQ}/\overline{GT}_0$ has higher priority over $\overline{RQ}/\overline{GT}_1$. After receiving hold request, the CPU sends hold acknowledge signal through these lines.

In the maximum mode of operation $\overline{WR}$, ALE, $\overline{DEN}$, DT/$\overline{R}$, etc. are not directly available from the processor. Rather they are available from the bus controller 8288.

The other pin functions are same as already discussed in the Section 2.2.

1. *Tightly coupled system (TCS):*

A processor which directly shares the local buses (data, address and control buses) of the host processor is called a *coprocessor*. In tightly coupled system (TCS), the 8086/88 microprocessor and the coprocessor share all the system resources, including the system bus, system memory, system I/O, clock generator and bus control logic [23]. The 8086/88 microprocessor acts as the host processor or master and the coprocessor acts as the slave. The bus access control is given by the 8086/88. Hence, the bus request line of the coprocessor is connected to the 8086/88.

i. Shared single-bus mode:

In shared single-bus mode, a single bus is shared between multiple processors. In fact, the processors share all the system resources. e.g., 8087 numeric data processor (NDP) or math coprocessor connected to 8086 microprocessor (host processor). Intel calls this two-chip CPU configuration an iAPX 86/21. 8087 is called a coprocessor because it shares the local buses of the 8086/88 host processor. 8086 – 8087 interface is shown in fig. 5.37 [12].

ii. I/O bus mode:

In I/O bus mode, a special processor handles the I/O operations. e.g., 8089 I/O processor (IOP) connected to 8086 host processor [12]. Two sets of buses are used – *local I/O bus* and *system bus*. The local I/O bus is accessible to 8089 only. The system bus is used to access system memory and also provides a common communications path between 8086 and 8089. Like 8087, 8089 is also called a coprocessor because it also shares the local buses of the 8086/88 host processor.

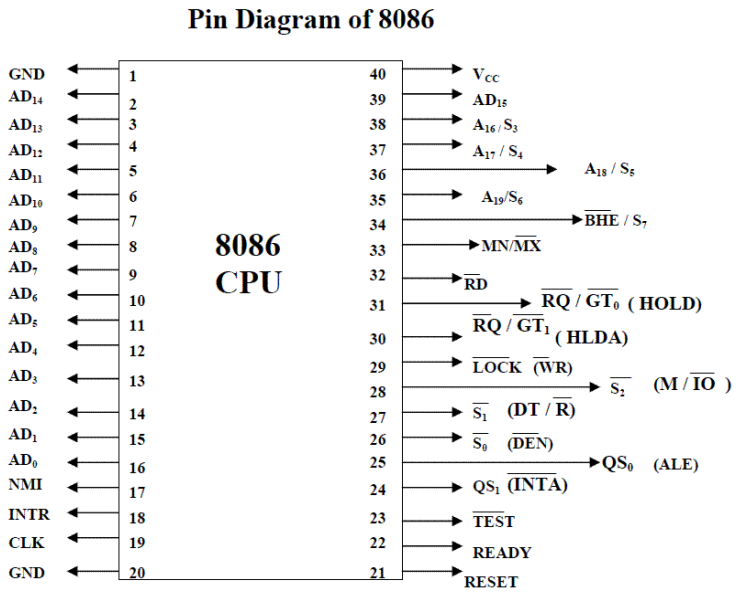
The difference between shared single-bus mode and I/O bus mode is that in shared single-bus mode, the coprocessor is dependent on the 8086/88 and interacts directly with the 8086/88 [23]. Hence, there are more direct lines between the host processor and the coprocessor. As against this, in I/O bus mode, the coprocessor may act independently.

Loosely coupled configuration

2. *Loosely coupled system (LCS) or resident bus mode:*

In loosely coupled system (LCS) or resident bus mode, each module in the system may be the system bus master. There are multiple general-purpose CPUs connected in parallel. Each CPU has access to its own *resident (private) bus* as well as the *system bus*. Thus, expensive system resources can be shared by all processors [12]. It is used for medium-size to large systems.

Pin details of 8086



AD15±AD0

ADDRESS DATA BUS: These lines constitute the time multiplexed

memory/IO address and data bus.

ALE

Address Latch Enable. A HIGH on this line causes the lower order 16bit address bus to be

latched that stores the addresses and then, the lower order 16bit of the address bus can be used

as data bus.

READY

READY is the acknowledgement from the addressed memory or I/O device that it will complete the data transfer.

INTR

INTERRUPT REQUEST: is a level triggered input which is sampled during the last clock cycle of each instruction to determine if the processor should enter into an interrupt acknowledge operation. A subroutine is vectored to via an interrupt vector lookup table located in system memory. It can be internally masked by software resetting the interrupt enable bit. INTR is internally synchronized. This signal is active HIGH.

INTA

Interrupt Acknowledge from the MP

NMI

NON-MASKABLE INTERRUPT: an edge triggered input which causes
an interrupt request to the MP. A subroutine is vectored to via an interrupt vector
lookup table located in system memory. NMI is not maskable internally
by software.

RESET: causes the processor to immediately terminate its present
activity. The signal must be active HIGH for at least four clock cycles. It
restarts execution

MN/MX

MINIMUM/MAXIMUM: indicates what mode the processor is to operate in.

• Minimum mode The 8086 processor works in a single processor environment.
All control signals for memory and I/O are generated by the microprocessor.

• Maximum mode is designed to be used when a coprocessor exists in the system.

• 8086 works in a multiprocessor environment. Control signals for memory and

I/O are generated by an external BUS Controller.

M/IO : Differentiate between the Memory and I/O operation. A LOW on this pin indicated I/O operation and a HIGH indicated a Memory Operation

HOLD : The 8086 has a pin called HOLD. This pin is used by external devices to gain control of the busses.

HLDA :

When the HOLD signal is activated by an external device, the 8086 stops executing instructions and stops using the busses. This would allow external devices to control the information on the

COMMON SIGNALS		
Name	**Function**	**Type**
AD_{15}– AD_0	Address/ Data Bus	Bidirectional 3-state
A_{19}/S_6– A_{16}/S_3	Address / Status	Output 3-State
$\overline{BHE}$ /S7	Bus High Enable / Status	Output 3-State
MN / $\overline{MX}$	Minimum / Maximum Mode Control	Input
$\overline{RD}$	Read Control	Output 3-State
TEST	Wait On Test Control	Input
READY	Wait State Controls	Input
RESET	System Reset	Input
NMI	Non - Maskable Interrupt Request	Input
INTR	Interrupt Request	Input
CLK	System Clock	Input
Vcc	+ 5 V	Input
GND	Ground	

Minimum Mode Signals ($\overline{MN}/MX$ = Vcc)		
Name	Function	Type
HOLD	Hold Request	Input
HLDA	Hold Acknowledge	Output
$\overline{WR}$	Write Control	Output 3- state
M/$\overline{IO}$	Memory or IO Control	Output 3-State
DT/$\overline{R}$	Data Transmit / Receiver	Output 3- State
$\overline{DEN}$	Date Enable	Output 3-State
ALE	Address Latch Enable	Output
$\overline{INTA}$	Interrupt Acknowledge	Output

Maximum mode signals (MN / $\overline{MX}$ = GND)

Name	Function	Type
RQ / $\overline{GT1, 0}$	Request / Grant Bus Access Control	Bidirectional
$\overline{LOCK}$	Bus Priority Lock Control	Output, 3- State
$\overline{S_2} - \overline{S_0}$	Bus Cycle Status	Output, 3- State
QS1, QS0	Instruction Queue Status	Output

Minimum Mode Interface

Address/Data bus:

20 bits vs 8 bits multiplexed

Status signals:

A16-A19 multiplexed with status signals S3-S6 respectively

S3 and S4 together form a 2 bit binary code that identifies which of the internal segment registers was used to generate the physical address that was output on the address bus during the current bus cycle.

S5 is the logic level of the internal interrupt enable flag, s6 is always logic 0.

Control Signals:

Address Latch Enable (ALE)

is a pulse to logic 1 that signals external circuitry when a valid address is on the bus. This address can be latched in external circuitry on the 1-to-0 edge of the pulse at ALE.

IO/M line:

memory or I/O transfer is selected (complement for 8086)

DT/R line: direction of data is selected

SSO (System Status Output) line: =1 when data is read from memory and =0 when code isread from memory (only for 8088)

BHE (Bank High Enable) line: =0 for most significant byte of data for8086 and also carries S7

RD line: =0 when a read cycle is in progress

WR line:=0 when a write cyle is in progress

DEN line: (Data enable) Enables the external devices to supply datato the processor.

Ready line:can be used to insert wait

states into the bus cycle so that it is extended by anumber of clock periods

Interrupt signals:INTR (Interrupt request):=1 shows there is a service request, sampled at the final clock cycle of each instruction acquisition cycle.

INTA: Processor responds with two pulses going to 0 when it services the interrupt and waitsfor the interrupt service number after the second pulse.

TEST: Processor suspends operation when =1. Resumes operation when=0. Used tosyncronize the processor to external events.

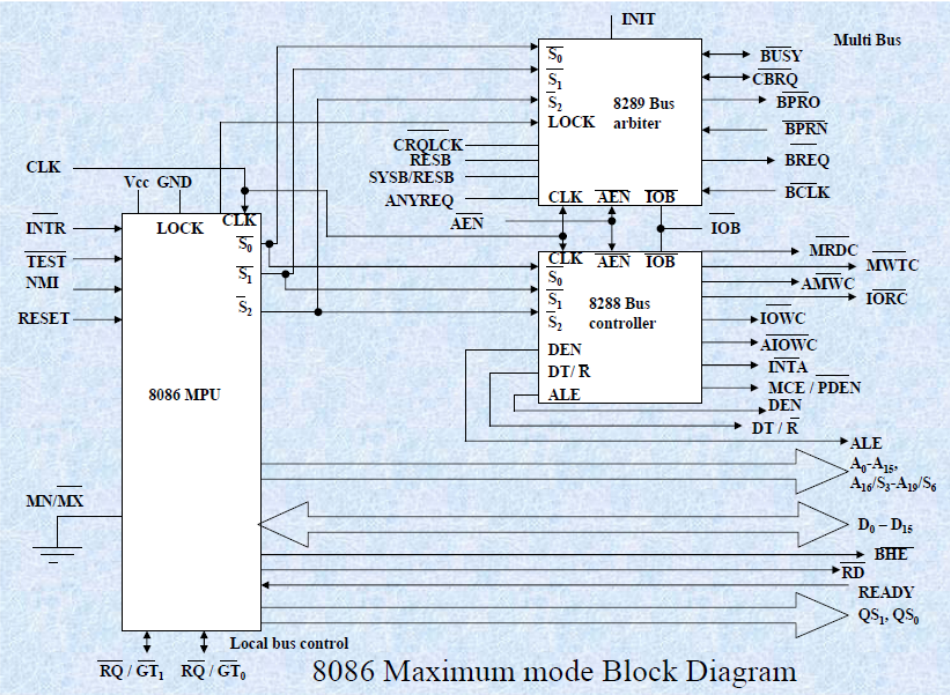
NMI (Nonmaskable interrupt) :A leading edge transition causes the processor go to theinterrupt routine after the current instruction is executed.

RESET : =0 Starts the reset sequence.

The maximum mode is selected by applying logic 0 to the MN / MX# input pin. This is a multi micro processors configuration. When the 8086 is set for the maximum-mode configuration, it provides signals for implementing a multiprocessor / coprocessor system environment.

Maximum mode

The maximum mode is selected by applying logic 0 to the MN / MX# input pin. This is a multi micro processors configuration. When the 8086 is set for the maximum-mode configuration, it provides signals for implementing a multiprocessor / coprocessor system environment.

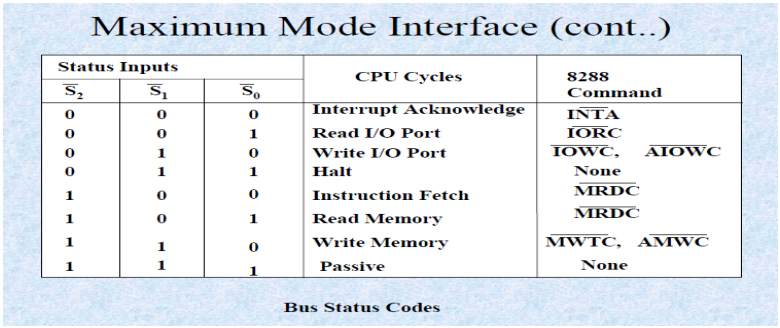


8086 Maximum mode Block Diagram

8288 Bus Controller – Bus Command and Control

Signals: 8086 does not directly provide all the signals that are required to control the memory, I/O and interrupt interfaces. Specially the WR, M/IO, DT/R, DEN, ALE and INTA, signals are no longer produced by the 8086. Instead it outputs three status signals S0, S1, S2 prior to the initiation of each bus cycle. This 3- bit bus status code identifies which type of bus cycle is to follow. S2S1S0 are input to the external

bus controller device, the bus controller generates the appropriately timed command and control signals.



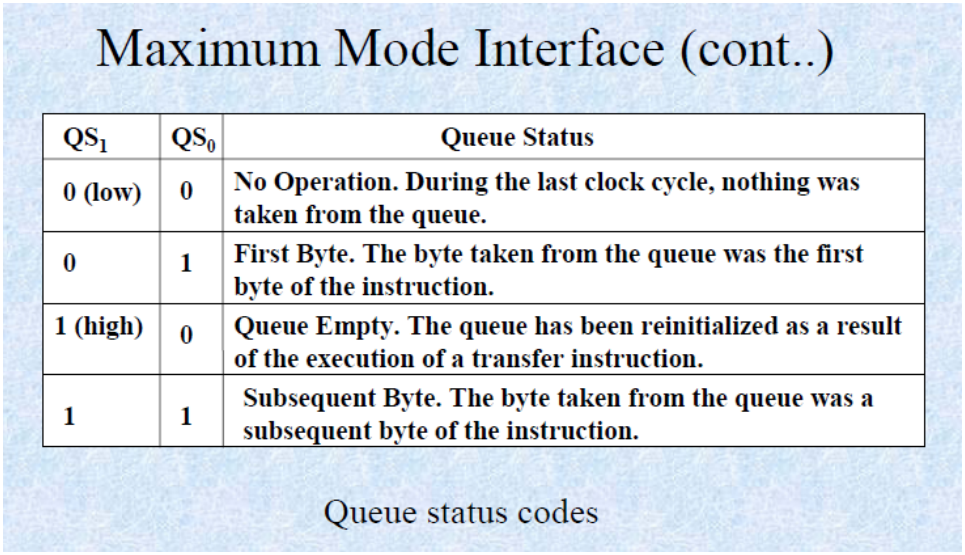
Maximum Mode Interface (cont..)

Status Inputs $\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	CPU Cycles	8288 Command
0	0	0	Interrupt Acknowledge	$\overline{INTA}$
0	0	1	Read I/O Port	$\overline{IORC}$
0	1	0	Write I/O Port	$\overline{IOWC}$, $\overline{AIOWC}$
0	1	1	Halt	None
1	0	0	Instruction Fetch	$\overline{MRDC}$
1	0	1	Read Memory	$\overline{MRDC}$
1	1	0	Write Memory	$\overline{MWTC}$, $\overline{AMWC}$
1	1	1	Passive	None

Bus Status Codes

The 8288 produces one or two of these eight command signals for each bus cycles. For instance, when the

Queue Status Signals : Two new signals that are produced by the 8086 in the maximum-mode system are queue status outputs QS0 and QS1. Together they form a 2-bit queue status code, QS1QS0.



Maximum Mode Interface (cont..)

QS_1	QS_0	Queue Status
0 (low)	0	No Operation. During the last clock cycle, nothing was taken from the queue.
0	1	First Byte. The byte taken from the queue was the first byte of the instruction.
1 (high)	0	Queue Empty. The queue has been reinitialized as a result of the execution of a transfer instruction.
1	1	Subsequent Byte. The byte taken from the queue was a subsequent byte of the instruction.

Queue status codes

Local Bus Control Signal – Request / Grant Signals: In a maximum mode configuration, the minimum mode HOLD, HLDA interface is also changed. These two are replaced by request/grant lines RQ/ GT0 and RQ/ GT1, respectively. They provide a prioritized bus access mechanism for accessing the local bus.

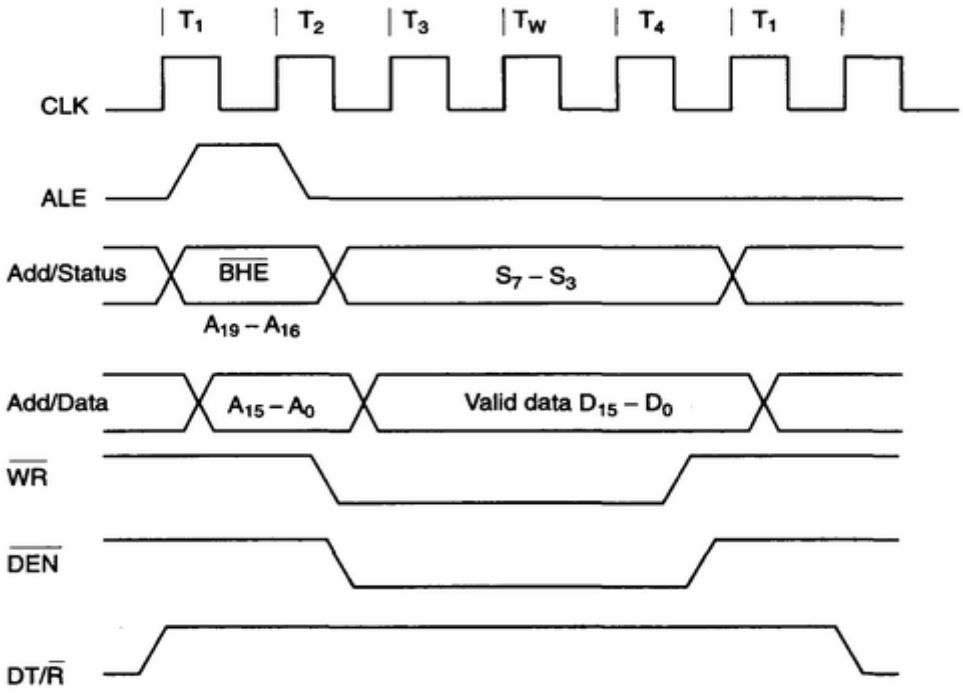


Fig.1.9(b) *Write Cycle Timing Diagram for Minimum Operation*