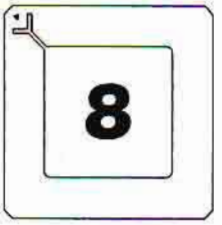


The 8088 and 8086 Microprocessors and Their Memory and Input/Output Interfaces

▲ INTRODUCTION

Up to this point in the book, we have studied the 8088 and 8086 microprocessors from a software point of view. We have covered their software architecture, instruction set, how to write, execute, and debug programs in assembly language, and found that the 8088 and 8086 were identical from the software point of view. This is not true of the hardware architectures of the 8088 and 8086 microcomputer systems. Now we begin examining the 8088 and 8086 microcomputer from the hardware point of view. In this chapter, we cover the 8088/8086's signal interfaces, memory interfaces, input/output interfaces, and bus cycles. The chapters that follow cover other hardware and interfacing aspects of these processors. This chapter includes the following topics:

▲ 8.1 8088 AND 8086 MICROPROCESSORS

The 8086, announced in 1978, was the first 16-bit microprocessor introduced by Intel Corporation. A second member of the 8086 family, the 8088 microprocessor, followed it in 1979. The 8088 is fully software compatible with its predecessor, the 8086. The difference between these two devices is in their hardware architecture. Just like the 8086, the 8088 is internally a 16-bit MPU. However, externally the 8086 has a 16-bit data bus, and the 8088 has an 8-bit data bus. This is the key hardware difference. Both devices have the ability to address up to 1Mbyte of memory via their 20-bit address buses. Moreover, they can address up to 64K of byte-wide input/output ports.

The 8088 and 8086 are both manufactured using *high-performance metal-oxide semiconductor (HMOS) technology*, and the circuitry on their chips is equivalent to approximately 29,000 transistors. They are housed in a 40-pin dual in-line package. This package can be mounted into a socket that is soldered to the circuit board or have its leads inserted through holes in the board and soldered. The signals pinned out to each lead are shown in Figs. 8–1(a) and (b), respectively. Many of their pins have multiple functions. For example, in the pin layout diagram of the 8088, we see that address bus lines A_0 through A_7 and data bus lines D_0 through D_7 are multiplexed. For this reason, these leads are labeled AD_0 through AD_7. By *multiplexed* we mean that the same physical pin carries an address bit at one time and the data bit at another time.

EXAMPLE 8.1

At what pin location on the 8088's package is address bit A_{16} output? With what other signal is it multiplexed? What function does this pin serve on the 8086?

Solution

Looking at Fig. 8–1(a), we find that the signal A_{16} is located at pin 38 on the 8088 and that it is multiplexed with signal S_3. Figure 8–1(b) shows us that pin 38 serves the same functions on the 8086.

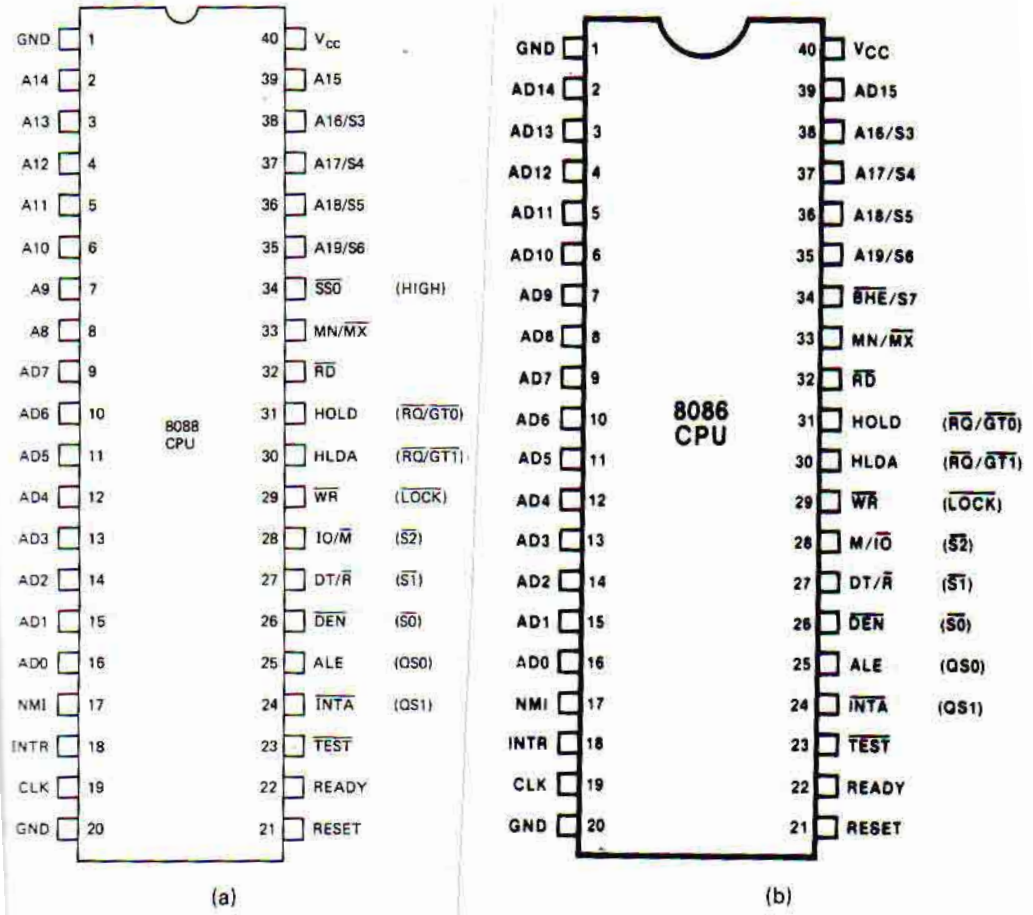


Figure 8–1 (a) Pin layout of the 8088 microprocessor. (Reprinted with permission of Intel Corporation, © 1981) (b) Pin layout of the 8086 microprocessor. (Reprinted with permission of Intel Corporation , © 1979)

8.2 MINIMUM-MODE AND MAXIMUM-MODE SYSTEMS

The 8088 and 8086 microprocessors can be configured to work in either of two modes: the *minimum mode* or the *maximum mode*. The minimum mode is selected by applying logic 1 to the MN/$\overline{\text{MX}}$ input lead. Minimum mode 8088/8086 systems are typically smaller and contain a single microprocessor. Connecting MN/$\overline{\text{MX}}$ to logic 0 selects the maximum mode of operation. This configures the 8088/8086 system for use in larger systems and with multiple processors. This mode-selection feature lets the 8088 or 8086 better meet the needs of a wide variety of system requirements.

Depending on the mode of operation selected, the assignments for a number of the pins on the microprocessor package are changed. As Fig. 8–1(a) shows, the pin functions of the 8088 specified in parentheses pertain to a maximum-mode system.

The signals of the 8088 microprocess sor common to both modes of operation, those unique to minimum mode and those unique to maximum mode, are listed in Figs. 8–2(a), (b), and (c), respectively. Here we find the name, function, and type for each signal. For example, the signal $\overline{RD}$ is in the common group. It functions as a read control output and is used to signal memory or I/O devices when the 8088's system bus is set up to read in data. Moreover, note that the signals hold request (HOLD) and hold acknowledge (HLDA) are produced only in the minimum-mode system. If the 8088 is set up for maximum mode, they are replaced by the request/grant bus access control lines $\overline{RQ}/\overline{GT}_0$ and $\overline{RQ}/\overline{GT}_1$.

Common signals		
Name	Function	Type
AD7–AD0	Address/data bus	Bidirectional, 3-state
A15–A8	Address bus	Output, 3-state
A19/S6–A16/S3	Address/status	Output, 3-state
MN/$\overline{MX}$	Minimum/maximum Mode control	Input
$\overline{RD}$	Read control	Output, 3-state
$\overline{TEST}$	Wait on test control	Input
READY	Wait state control	Input
RESET	System reset	Input
NMI	Nonmaskable Interrupt request	Input
INTR	Interrupt request	Input
CLK	System clock	Input
V_{CC}	+5 V	Input
GND	Ground	

(a)

Minimum mode signals (MN/$\overline{MX}$ = V_{CC})		
Name	Function	Type
HOLD	Hold request	Input
HLDA	Hold acknowledge	Output
$\overline{WR}$	Write control	Output, 3-state
IO/$\overline{M}$	IO/memory control	Output, 3-state
DT/$\overline{R}$	Data transmit/receive	Output, 3-state
$\overline{DEN}$	Data enable	Output, 3-state
$\overline{SSO}$	Status line	Output, 3-state
ALE	Address latch enable	Output
$\overline{INTA}$	Interrupt acknowledge	Output

(b)

Maximum mode signals (MN/$\overline{MX}$ = GND)		
Name	Function	Type
$\overline{RQ}/\overline{GT1}$, 0	Request/grant bus access control	Bidirectional
$\overline{LOCK}$	Bus priority lock control	Output, 3-state
$\overline{S2}$–$\overline{S0}$	Bus cycle status	Output, 3-state
QS1, QS0	Instruction queue status	Output

(c)

Figure 8–2 (a) Signals common to both minimum and maximum modes. (b) Unique minimum-mode signals. (c) Unique maximum-mode signals.

EXAMPLE 8.2

Which pins provide different signal functions in the minimum-mode 8088 and minimum-mode 8086?

Solution

Comparing the pin layouts of the 8088 and 8086 in Fig. 8–1, we find the following:

1. Pins 2 through 8 on the 8088 are address lines A_{14} through A_8, but on the 8086 they are address/data lines AD_{14} through AD_8.
2. Pin 28 on the 8088 is the IO/$\overline{M}$ output and on the 8086 it is the M/$\overline{IO}$ output.
3. Pin 34 of the 8088 is the $\overline{SSO}$ output, and on the 8086 this pin supplies the $\overline{BHE}/S_7$ output.

▲ 8.3 MINIMUM-MODE INTERFACE SIGNALS

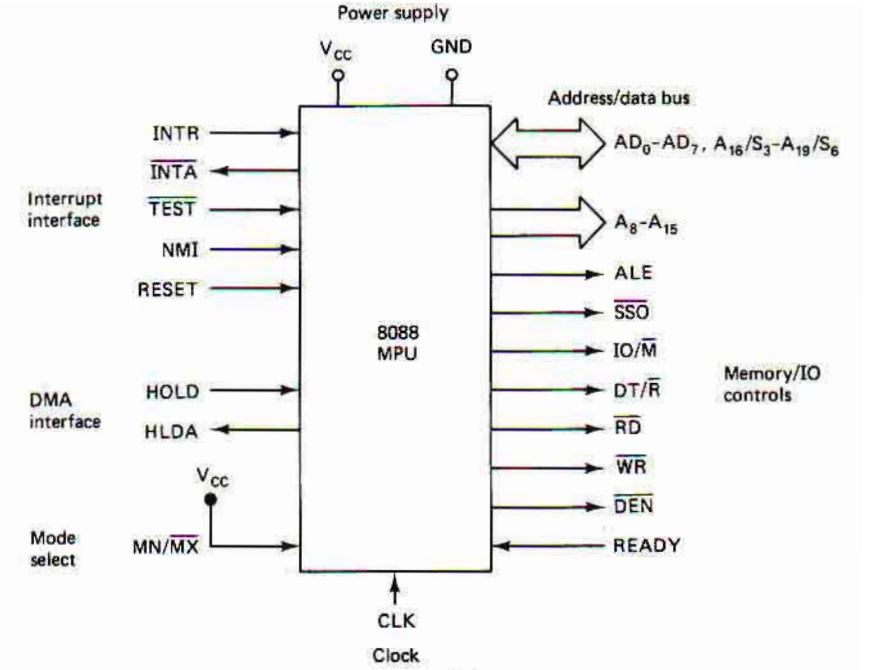
When minimum mode operation is selected, the 8088 or 8086 itself provides all the control signals needed to implement the memory and I/O interfaces. Figures 8–3(a) and (b) show block diagrams of a minimum-mode configuration of the 8088 and 8086, respectively. The minimum-mode signals can be divided into the following basic groups: address/data bus, status, control, interrupt, and DMA.

Address/Data Bus

Let us first look at the address/data bus. In an 8088-based microcomputer system, these lines serve two functions. As an *address bus,* they are used to carry address information to the memory and I/O ports. The address bus is 20 bits long and consists of signal lines A_0 through A_{19}. Of these, A_{19} represents the MSB and A_0 the LSB. A 20-bit address gives the 8088 a 1Mbyte memory address space. However, only address lines A_0 through A_{15} are used when accessing I/O. This gives the 8088 an independent I/O address space that is 64Kbytes in length.

The eight *data bus* lines D_0 through D_7 are actually multiplexed with address lines A_0 through A_7, respectively. For this reason, they are denoted as AD_0 through AD_7. Data line D_7 is the MSB in the byte of data and D_0 the LSB. When acting as a data bus, they carry read/write data for memory, input/output data for I/O devices, and interrupt-type codes from an interrupt controller.

Looking at Fig. 8–3(b), we see that the 8086 has 16 data bus lines instead of 8 as in the 8088. Data lines are multiplexed with address lines A_0 through A_{15} and are therefore denoted as AD_0 through AD_{15}.



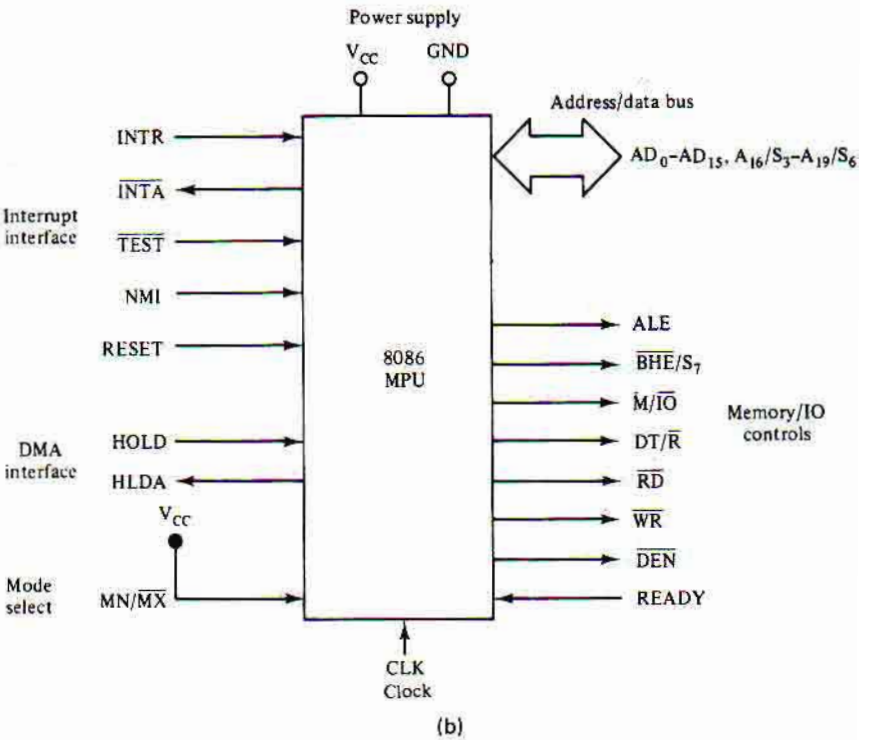


Figure 8–3 (a) Block diagram of the minimum-mode 8088 MPU. (b) Block diagram of the minimum-mode 8086 MPU.

S_4	S_3	Address Status
0	0	Alternate (relative to the ES segment)
0	1	Stack (relative to the SS segment)
1	0	Code/None (relative to the CS segment or a default of zero)
1	1	Data (relative to the DS segment)

Figure 8–4 Address bus status codes. (Reprinted with permission of Intel Corporation, © 1979)

Status Signals

The four most significant address lines, A_{19} through A_{16} of both the 8088 and 8086 are also multiplexed, but in this case with *status signals* S_6 through S_3. These status bits are output on the bus at the same time that data are transferred over the other bus lines. Bits S_4 and S_3 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. These four codes and the registers they represent are shown in Fig. 8–4. Note that the code $S_4S_3 = 00$ identifies the extra segment register as the source of the segment address.

Status line S_5 reflects the status of another internal characteristic of the MPU. It is the logic level of the internal interrupt enable flag. The status bit S_6 is always at the 0 logic level.

Control Signals

The *control signals* are provided to support the memory and I/O interfaces of the 8088 and 8086. They control functions such as when the bus carries a valid address, which direction data are transferred over the bus, when valid write data are on the bus, and when to put read data on the system bus. For example, *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.

Using the IO/$\overline{M}$ (*IO/memory*) line, DT/$\overline{R}$ (*data transmit/receive*) line, and $\overline{SSO}$ (*status output*) line, the 8088 signals which type of bus cycle is in progress and in which direction data are to be transferred over the bus. The logic level of IO/$\overline{M}$ tells external circuitry whether a memory or I/O transfer is taking place over the bus. Logic 0 at this output signals a memory operation, and logic 1 signals an I/O operation. The direction of data transfer over the bus is signaled by the logic level output at DT/$\overline{R}$. When this line is logic 1 during the data transfer part of a bus cycle, the bus is in the transmit mode. Therefore, data are either written into memory or output to an I/O device. On the other hand, logic 0 at DT/$\overline{R}$ signals that the bus is in the receive mode. This corresponds to reading data from memory or input of data from an input port.

Comparing Figs. 8–3(a) and 8–3(b), we find two differences between the minimum-mode 8088 and 8086 microprocessors. First, the 8086's memory/IO control (M/$\overline{IO}$) signal is the complement of the equivalent signal of the 8088. Second, the 8088's $\overline{SSO}$ status signal is replaced by *bank high enable* ($\overline{BHE}$) on the 8086. Logic 0 on this line is used as a memory enable signal for the most significant byte half of the data bus, D_8 through D_{15}. This line also carries status bit S_7.

The signals *read* ($\overline{RD}$) and *write* ($\overline{WR}$) indicate that a read bus cycle or a write bus cycle, respectively, is in progress. The MPU switches $\overline{WR}$ to logic 0 to signal external devices that valid write or output data are on the bus. On the other hand, $\overline{RD}$ indicates that the MPU is performing a read of data off the bus. During read operations, one other control signal, $\overline{DEN}$ (*data enable*), is also supplied. It enables external devices to supply data to the microprocessor.

One other control signal involved with the memory and I/O interface, the READY signal, can be used to insert wait states into the bus cycle so that it is extended by a number of clock periods. This signal is provided by way of an external clock generator device and can be supplied by the memory or I/O subsystem to signal the MPU when it is ready to permit the data transfer to be completed.

Interrupt Signals

The key interrupt interface signals are *interrupt request* (INTR) and *interrupt acknowledge* ($\overline{INTA}$). INTR is an input to the 8088 and 8086 that can be used by an external device to signal that it needs to be serviced. This input is sampled during the final clock period of each *instruction acquisition cycle*. Logic 1 at INTR represents an active interrupt request. When the MPU recognizes an interrupt request, it indicates this fact to external circuits with pulses to logic 0 at the $\overline{INTA}$ output.

The $\overline{TEST}$ input is also related to the external interrupt interface. For example, execution of a WAIT instruction causes the 8088 or 8086 to check the logic level at the $\overline{TEST}$ input. If logic 1 is found at this input, the MPU suspends operation and goes into what is known as the *idle state*. The MPU no longer executes instructions; instead, it repeatedly checks the logic level of the $\overline{TEST}$ input waiting for its transition back to logic 0. As $\overline{TEST}$ switches to 0, execution resumes with the next instruction in the program. This feature can be used to synchronize the operation of the MPU to an event in external hardware.

There are two more inputs in the interrupt interface: *nonmaskable interrupt* (NMI) and *reset* (RESET). On the 0-to-1 transition of NMI, control is passed to a nonmaskable interrupt service routine at completion of execution of the current instruction. NMI is the interrupt request with highest priority and cannot be masked by software. The RESET input is used to provide a hardware reset for the MPU. Switching RESET to logic 0 initializes the internal registers of the MPU and initiates a reset service routine.

DMA Interface Signals

The *direct memory access* (DMA) interface of the 8088/8086 minimum-mode microcomputer system consists of the HOLD and HLDA signals. When an external device wants to take control of the system bus, it signals this fact to the MPU by switching HOLD to the 1 logic level. For example, when the HOLD input of the 8088 becomes active, it enters the hold state at the completion of the current bus cycle. When in the hold state, signal lines AD_0 through AD_7, A_8 through A_{15}, A_{16}/S_3 through A_{19}/S_6, $\overline{SSO}$,

IO/$\overline{M}$, DT/$\overline{R}$, $\overline{RD}$, $\overline{WR}$, $\overline{DEN}$, and INTR are all put into the high-Z state. The 8088 signals external devices that it is in this state by switching its HLDA output to the 1 logic level.

▲ 8.4 MAXIMUM-MODE INTERFACE SIGNALS

When the 8088 or 8086 microprocessor is set for the maximum-mode configuration, it produces signals for implementing a *multiprocessor/coprocessor system environment.* By *multiprocessor environment* we mean that multiple microprocessors exist in the system and that each processor executes its own program. Usually in this type of system environment, some system resources are common to all processors. They are called *global resources.* There are also other resources that are assigned to specific processors. These dedicated resources are known as *local* or *private resources.*

In the maximum-mode system, facilities are provided for implementing allocation of global resources and passing bus control to other microprocessors sharing the system bus.

8288 Bus Controller: Bus Commands and Control Signals

Looking at the maximum-mode block diagram in Fig. 8–5(a), we see that the 8088 does not directly provide all the signals that are required to control the memory, I/O, and interrupt interfaces. Specifically, the $\overline{WR}$, IO/$\overline{M}$, DT/$\overline{R}$, $\overline{DEN}$, ALE, and $\overline{INTA}$ signals are no longer produced by the 8088. Instead, it outputs a status code on three signals lines, $\overline{S}_0$, $\overline{S}_1$, and $\overline{S}_2$, prior to the initiation of each bus cycle. This 3-bit *bus status code* identifies which type of bus cycle is to follow. $\overline{S}_2\overline{S}_1\overline{S}_0$ are input to the external *bus controller* device, the 8288, which decodes them to identify the type of MPU bus cycle. The block diagram and pin layout of the 8288 are shown in Figs. 8–6(a) and (b), respectively. In response, the bus controller generates the appropriately timed command and control signals.

Figure 8–7 shows the relationship between the bus status codes and the types of bus cycles. Also shown are the output signals generated to tell external circuitry which type of bus cycle is taking place. These output signals are *memory read command* ($\overline{MRDC}$), *memory write command* ($\overline{MWTC}$), *advanced memory write command* ($\overline{AMWC}$), *I/O read command* ($\overline{IORC}$), *I/O write command* ($\overline{IOWC}$), *advanced I/O write command* ($\overline{AIOWC}$), and *interrupt acknowledge* ($\overline{INTA}$).

The 8288 produces one or two of these seven command signals for each bus cycle. For instance, when the 8088 outputs the code $\overline{S}_2\overline{S}_1\overline{S}_0 = 001$, it indicates that an I/O read cycle is to be performed. In turn, the 8288 makes its $\overline{IORC}$ output switch to logic 0. On the other hand, if the code 111 is output by the 8088, it is signaling that no bus activity is to take place; the 8288 produces no command signals.

The other control outputs produced by the 8288 consist of DEN, DT/$\overline{R}$, and ALE. These three signals provide the same functions as those described for the minimum mode. Figure 8–5(b) shows that the 8288 bus controller connects to the 8086 in the same way as the 8088, and it also produces the same output signals.

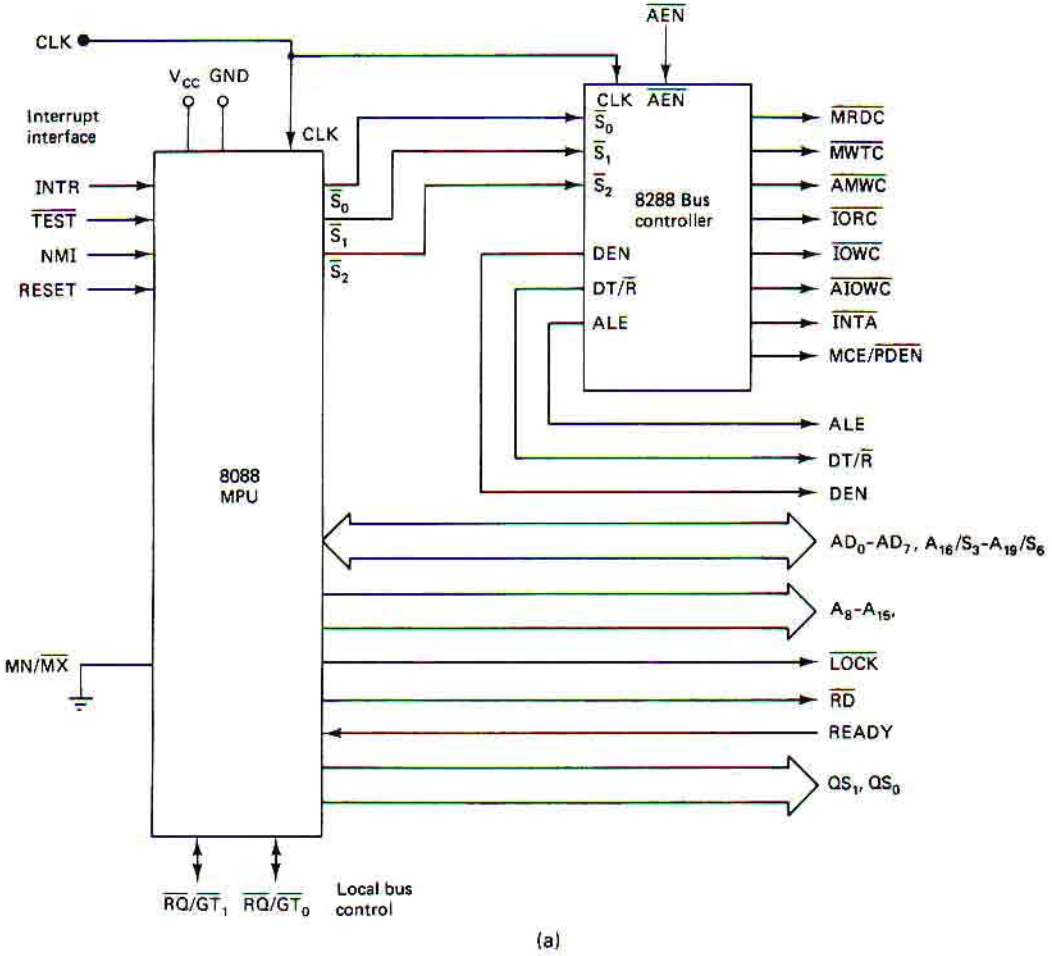


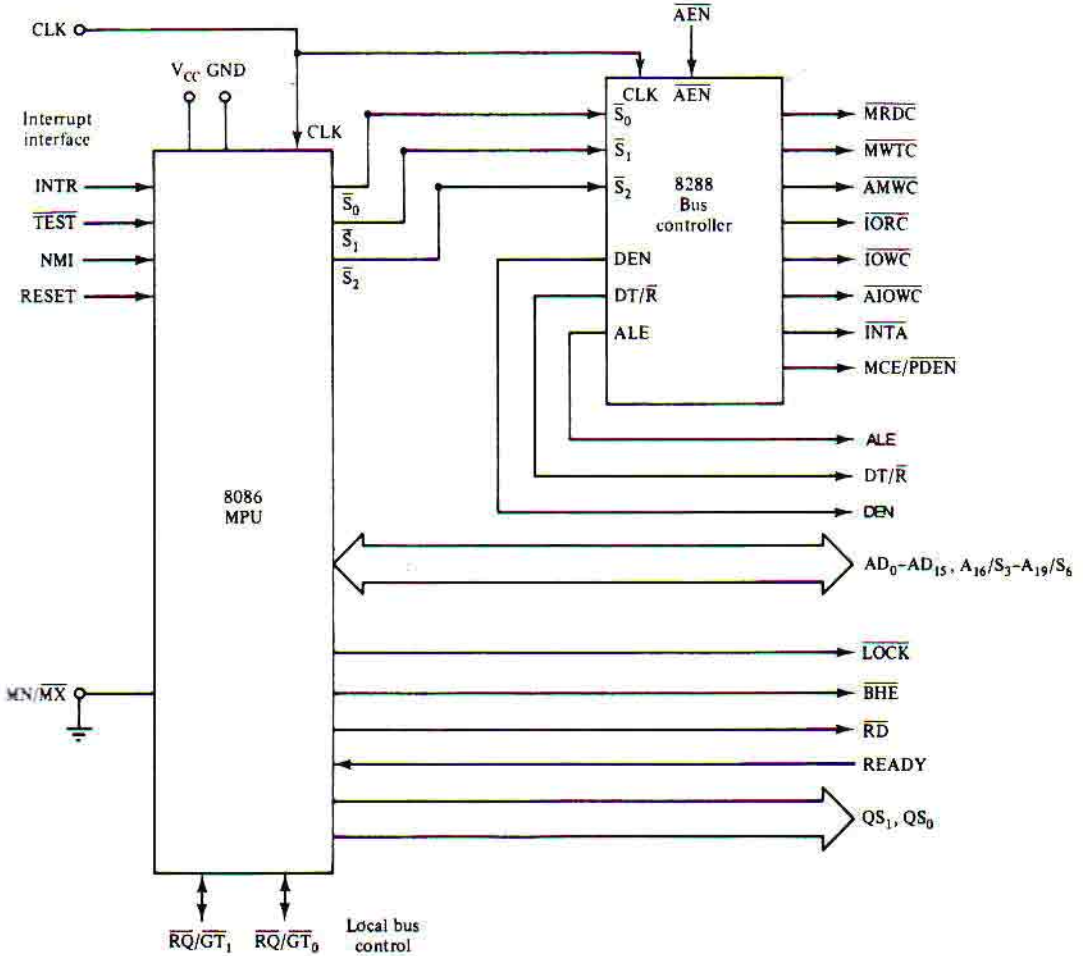
Figure 8–5 (a) 8088 maximum-mode block diagram. (b) 8086 maximum-mode block diagram.

EXAMPLE 8.3

If the bus status code $\overline{S}_2\overline{S}_1\overline{S}_0$ equals 101, what type of bus activity is taking place? Which command output is produced by the 8288?

Solution

Looking at the table in Fig. 8–7, we see that bus status code 101 identifies a read memory bus cycle and causes the $\overline{\text{MRDC}}$ output of the bus controller to be switched to logic 0.



(b)

Figure 8–5 (continued)

Lock Signal

To implement a multiprocessor system, a signal called lock ($\overline{\text{LOCK}}$) is provided on the 8088 and 8086. This signal is meant to be output (logic 0) whenever the processor wants to lock out the other processors from using the bus. This would be the case when a shared resource is accessed. The $\overline{\text{LOCK}}$ signal is compatible with the *Multibus*, an industry standard for interfacing microprocessor systems in a multiprocessor environment.

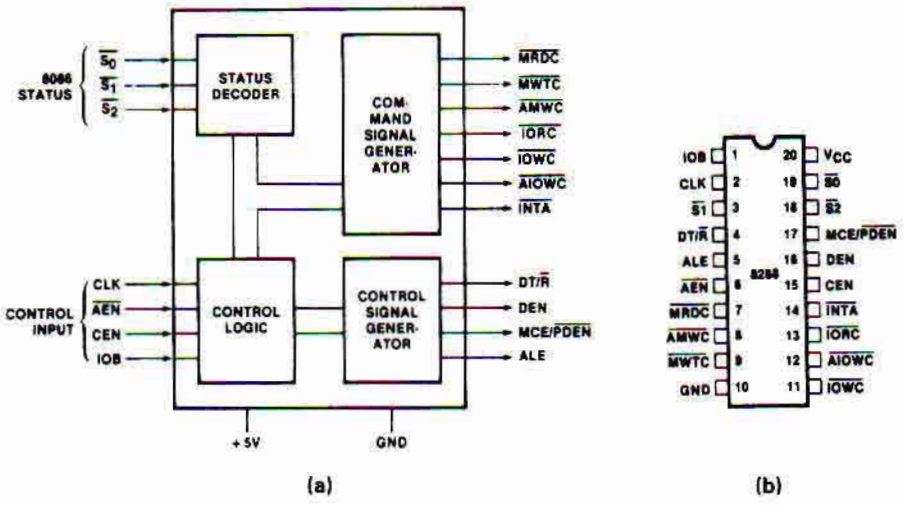


Figure 8–6 (a) Block diagram of the 8288. (Reprinted with permission of Intel Corporation, © 1979) (b) Pin layout. (Reprinted with permission of Intel Corporation, © 1979)

Queue Status Signals

Two other signals produced by the 8088 and 8086, in the maximum-mode microcomputer system, are queue status outputs QS_0 and QS_1 that form a 2-bit *queue status code*, QS_1QS_0. This code tells the external circuitry what type of information was removed from the instruction queue during the previous clock cycle. Figure 8–8 shows the four different queue status codes. Note that $QS_1QS_0 = 01$ indicates that the first byte of an instruction was taken off the queue. As shown, the fetch of the next byte of the instruction is identified by the code 11. Whenever the queue is reset due to a transfer of control, the reinitialization code 10 is output.

Status Inputs			CPU Cycle	8288 Command
$\overline{S2}$	$\overline{S1}$	$\overline{S0}$		
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

Figure 8–7 Bus status codes. (Reprinted with permission of Intel Corporation, © 1979)

QS1	QS0	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.

Figure 8–8 Queue status codes. (Reprinted with permission of Intel Corporation, © 1979)

Local Bus Control Signals

In a maximum-mode configuration, the minimum-mode HOLD and HLDA interface of the 8088/8086 is also changed. These two signals are replaced by *request/grant lines* $\overline{RQ}/\overline{GT}_0$ and $\overline{RQ}/\overline{GT}_1$. They provide a prioritized bus access mechanism for accessing the *local bus.*

▲ 8.5 ELECTRICAL CHARACTERISTICS

In the preceding sections, the pin layout and minimum- and maximum-mode interface signals of the 8088 and 8086 microprocessors were introduced. Here we will first look at the power supply ratings of these processors and then their input and output electrical characteristics.

Looking at Fig. 8–1(a), we find that power is applied between pin 40 (V_{cc}) and pins 1(GND) and 20(GND). Pins 1 and 20 should be connected together. The nominal value of V_{cc} is specified as +5 V dc with a tolerance of ±10%. This means that the 8088 or 8086 will operate correctly as long as the difference in voltage between V_{cc} and GND is greater than 4.5 V dc and less than 5.5 V dc. At room temperature (25°C), both the 8088 and 8086 draw a maximum of 340 mA from the supply.

Let us now look at the dc I/O characteristics of the microprocessor—that is, its input and output logic levels. These ratings tell the minimum and maximum voltages for the 0 and 1 logic states for which the circuit will operate correctly. Different values are specified for the inputs and outputs.

Figure 8–9 shows the I/O voltage specifications for the 8088. Notice that the minimum logic 1 (high-level) voltage at an output (V_{OH}) is 2.4 V. This voltage is specified for a test condition that identifies the amount of current being sourced by the output (I_{OH}) as −400 μA. All processors must be tested during manufacturing to ensure that under this test condition the voltages at all outputs will remain above the value of V_{OHmin}.

Symbol	Meaning	Minimum	Maximum	Test condition
V_{IL}	Input low voltage	−0.5 V	+0.8 V	
V_{IH}	Input high voltage	+2.0 V	V_{cc} + 0.5 V	
V_{OL}	Output low voltage		+0.45 V	I_{OL} = 2.0mA
V_{OH}	Output high voltage	+2.4 V		I_{OH} = −400 uA

Figure 8–9 I/O voltage levels.

Input voltage levels are specified in a similar way; except here the ratings identify the range of voltage that will be correctly identified as a logic 0 or a logic 1 at an input. For instance, voltages in the range $V_{ILmin} = -0.5$ V to $V_{ILmax} = +0.8$ V represent a valid logic 0 (lower level) at an input of the 8088.

The I/O voltage levels of the 8086 microprocessor are identical to those for the 8088 as shown in Fig. 8–9. However, there is one difference in the test conditions. For the 8086, V_{OL} is measured at 2.5 mA instead of 2.0 mA.

▲ 8.6 SYSTEM CLOCK

The time base for synchronization of the internal and external operations of the microprocessor in a microcomputer system is provided by the *clock* (CLK) input signal. At present, the 8088 is available in two different speeds. The standard part operates at 5 MHz and the 8088-2 operates at 8 MHz. On the other hand, the 8086 microprocessor is manufactured in three speeds: the 5-MHz 8086, the 8-MHz 8086-2, and the 10-MHz 8086-1. The 8284 clock generator and driver IC generates CLK. Figure 8–10 is a block diagram of this device.

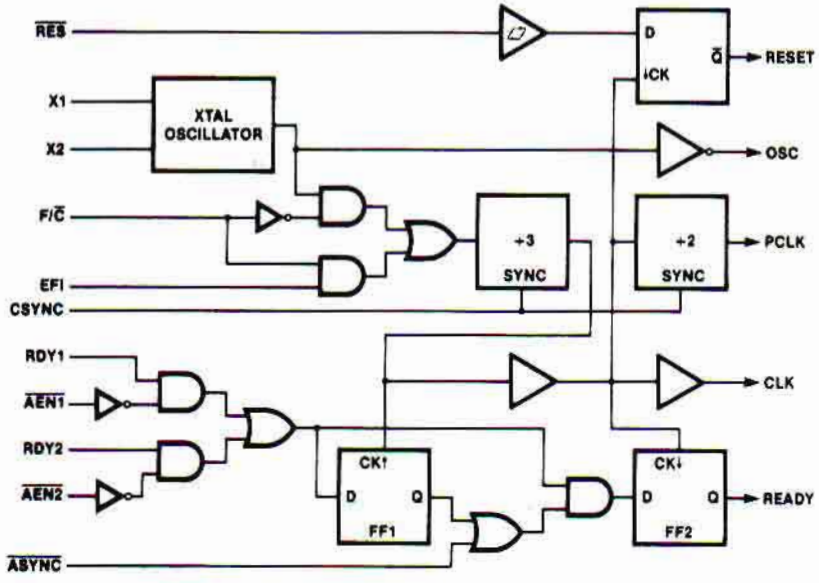


Figure 8–10 Block diagram of the 8284 clock generator. (Reprinted with permission of Intel Corporation, © 1979)

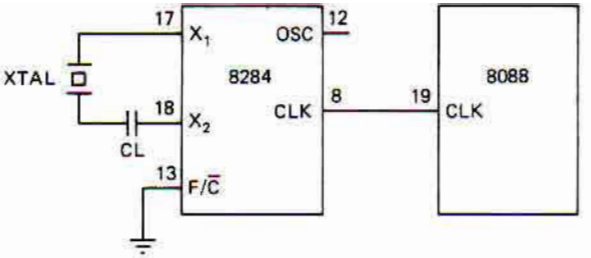


Figure 8–11 Connecting the 8284 to the 8088. (Reprinted with permission of Intel Corporation, © 1979)

The standard way in which this clock chip is used with the 8088 is to connect either a 15- or 24-MHz crystal between its X_1 and X_2 inputs. This circuit connection is shown in Fig. 8–11. Note that a series capacitor C_L is also required. Its typical value when used with the 15-MHz crystal is 12 pF. The *fundamental crystal frequency* is divided by 3 within the 8284 to give either a 5- or 8-MHz clock signal. This signal is internally buffered and output at CLK. The CLK output of the 8284 can be directly connected to the CLK input of the 8088. The 8284 connects to the 8086 in exactly the same way.

Figure 8–12 shows the waveform of CLK. Here we see that the signal is specified at metal oxide semiconductor (MOS)-compatible voltage levels and not transistor transistor logic (TTL) levels. Its minimum and maximum low logic levels are $V_{Lmin} = -0.5$ V and $V_{Lmax} = 0.6$ V, respectively. Moreover, the minimum and maximum high logic levels are $V_{Hmin} = 3.9$ V and $V_{Hmax} = V_{cc} + 1$ V, respectively. The *period* of the clock signal of a 5-MHz 8088 can range from a minimum of 200 ns to a maximum of 500 ns, and the maximum *rise* and *fall times* of its edges equal 10 ns.

Figure 8–10 shows two more clock outputs on the 8284: the *peripheral clock* (PCLK) and *oscillator clock* (OSC). These signals are provided to drive peripheral ICs. The clock signal output at PCLK is half the frequency of CLK. For instance, if an 8088 is operated at 5 MHz, PCLK is 2.5 MHz. Also, it is at TTL-compatible levels rather than MOS levels. On the other hand, the OSC output is at the crystal frequency, which is three times that of CLK. Figure 8–13 illustrates these relationships.

The 8284 can also be driven from an external clock source. The external clock signal is applied to the external frequency input (EFI). Input F/$\overline{C}$ is provided for clock source selection. When it is strapped to the 0 logic level, the crystal between X_1 and X_2 is used. On the other hand, applying logic 1 to F/$\overline{C}$ selects EFI as the source of the clock. The clock sync (CSYNC) input can be used for external synchronization in systems that employ multiple clocks.

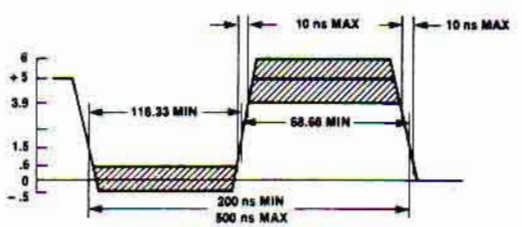


Figure 8–12 CLK voltage and timing characteristics for a 5-MHz processor. (Reprinted with permission of Intel Corporation, © 1979)

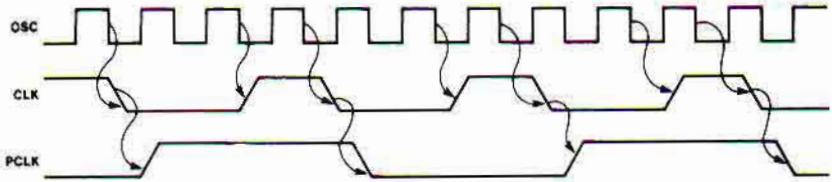


Figure 8–13 Relationship between CLK and PCLK. (Reprinted with permission of Intel Corporation, © 1979)

EXAMPLE 8.4

If the CLK input of an 8086 MPU is to be driven by a 9-MHz signal, what speed version of the 8086 must be used and what frequency crystal must be attached to the 8284?

Solution

The 8086-1 is the version of the 8086 that can be run at 9 MHz. To create the 9-MHz clock, a 27-MHz crystal must be used on the 8284.

▲ 8.7 BUS CYCLE AND TIME STATES

A *bus cycle* defines the basic operation that a microprocessor performs to communicate with external devices. Examples of bus cycles are the memory read, memory write, input/output read, and input/output write. As shown in Fig. 8–14(a), a bus cycle corresponds to a sequence of events that start with an address being output on the system bus followed by a read or write data transfer. During these operations, the MPU produces a series of control signals to control the direction and timing of the bus.

The bus cycle of the 8088 and 8086 microprocessors consists of at least four clock periods. These four time states are called T_1, T_2, T_3, and T_4. During T_1, the MPU puts an address on the bus. For a write memory cycle, data are put on the bus during state T_2 and maintained through T_3 and T_4. When a read cycle is to be performed, the bus is first put in the high-Z state during T_2 and then the data to be read must be available on the bus during T_3 and T_4. These four clock states give a *bus cycle duration* of 125 ns $\times$ 4 = 500 ns in an 8-MHz 8088 system.

If no bus cycles are required, the microprocessor performs what are known as *idle states*. During these states, no bus activity takes place. Each idle state is one clock period long, and any number of them can be inserted between bus cycles. Figure 8–14(b) shows two bus cycles separated by idle states. Idle states are performed if the instruction queue inside the microprocessor is full and it does not need to read or write operands from memory.

Wait states can also be inserted into a bus cycle. This is done in response to a request by an event in external hardware instead of an internal event such as a full queue.

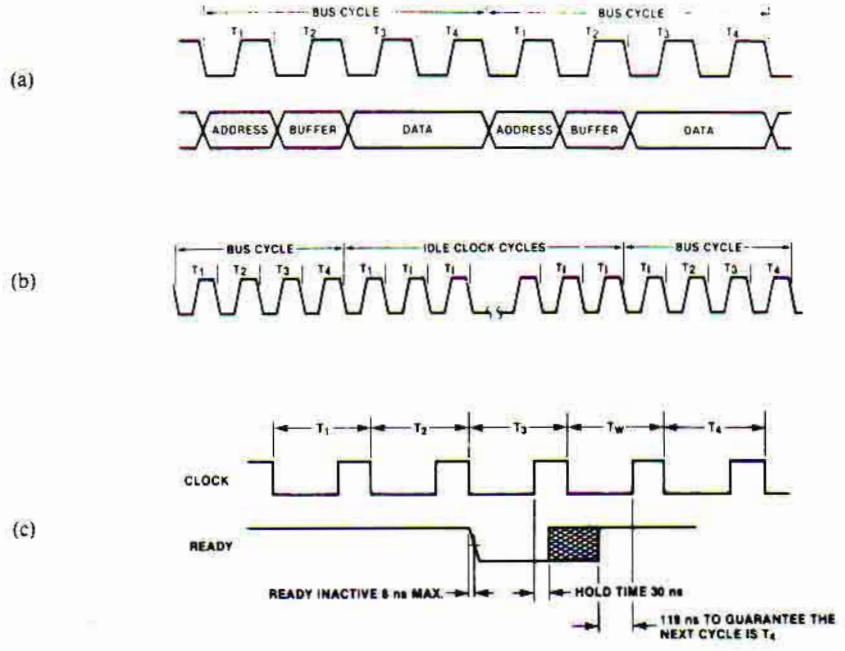


Figure 8–14 (a) Bus cycle clock periods. (Reprinted with permission of Intel Corporation, © 1979) (b) Bus cycle with idle states. (Reprinted with permission of Intel Corporation, © 1979) (c) Bus cycle with wait states. (Reprinted with permission of Intel Corporation, © 1979)

In fact, the READY input of the MPU is provided specifically for this purpose. Figure 8–14(c) shows that logic 0 at this input indicates that the current bus cycle should not be completed. As long as READY is held at the 0 level, wait states are inserted between states T_3 and T_4 of the current bus cycle, and the data that were on the bus during T_3 are maintained. The bus cycle is not completed until the external hardware returns READY back to the 1 logic level. This extends the duration of the bus cycle, thereby permitting the use of slower memory and I/O devices in the system.

EXAMPLE 8.5

What is the duration of the bus cycle in the 8088-based microcomputer if the clock is 8 MHz and two wait states are inserted?

Solution

The duration of the bus cycle in an 8-MHz system is given in general by

$$t_{cyc} = 500 \text{ ns} + N \times 125 \text{ ns}$$

In this expression N stands for the number of wait states. For a bus cycle with two wait states, we get

$$t_{cyc} = 500 \text{ ns} + 2 \times 125 \text{ ns} = 500 \text{ ns} + 250 \text{ ns}$$
$$= 750 \text{ ns}$$

▲ 8.8 HARDWARE ORGANIZATION OF THE MEMORY ADDRESS SPACE

From a hardware point of view, the memory address spaces of the 8088- and 8086-based microcomputers are organized differently. Figure 8–15(a) shows that the 8088's memory subsystem is implemented as a single 1M × 8 memory bank. Looking at the block diagram in Fig. 8–15(a), we see that these byte-wide storage locations are assigned to consecutive addresses over the range from 00000_{16} through $FFFFF_{16}$. During memory operations, a 20-bit address is applied to the memory bank over address lines A_0 through A_{19}. It is this address that selects the storage location that is to be accessed. Bytes of data are transferred between the 8088 and memory over data bus lines D_0 through D_7.

On the other hand, the 8086's 1Mbyte memory address space, as shown in Fig. 8–15(b), is implemented as two independent 512Kbyte banks: the *low (even) bank* and the *high (odd) bank*. Data bytes associated with an even address (00000_{16}, 00002_{16}, etc.) reside in the low bank, and those with odd addresses (00001_{16}, 00003_{16}, etc.) reside in the high bank.

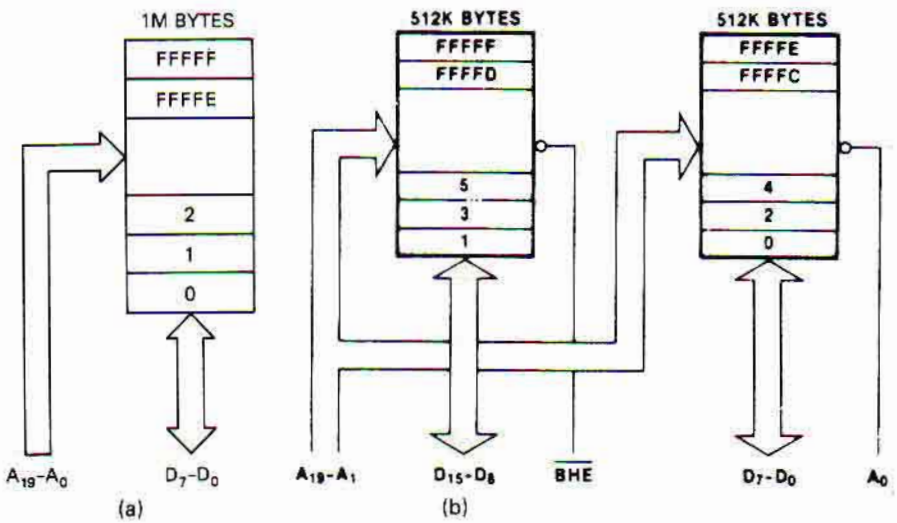


Figure 8–15 (a) 1M × 8 memory bank of the 8088. (b) High and low memory banks of the 8086. (Reprinted with permission of Intel Corporation, © 1979)

The diagram in Fig. 8–15(b) shows that for the 8086 address bits, A_1 through A_{19} select the storage location that is to be accessed. They are applied to both banks in parallel. A_0 and bank high enable ($\overline{BHE}$) are used as bank-select signals. Logic 0 at A_0 identifies an even-addressed byte of data and causes the low bank of memory to be enabled. On the other hand, $\overline{BHE}$ equal to 0 enables the high bank to access an odd-addressed byte of data. Each of the memory banks provides half of the 8086's 16-bit data bus. Notice that the lower bank transfers bytes of data over data lines D_0 through D_7, while data transfers for a high bank use D_8 through D_{15}.

We just saw that the memory subsystem of the 8088-based microcomputer system is actually organized as 8-bit bytes, not as 16-bit words. However, the contents of any two consecutive byte storage locations can be accessed as a word. The lower-addressed byte is the least significant byte of the word, and the higher-addressed byte is its most significant byte. Let us now look at how a byte and a word of data are read from memory.

Figure 8–16(a) shows how a byte-memory operation is performed to the storage location at address X. As shown in the diagram, the address is supplied to the memory

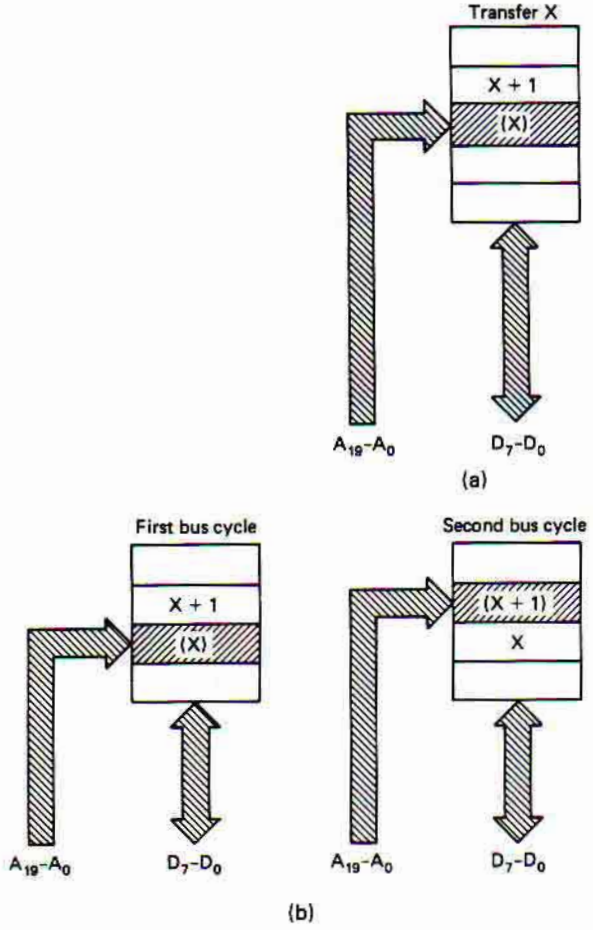


Figure 8–16 (a) Byte transfer by the 8088. (b) Word transfer by the 8088.

bank over lines A_0 through A_{19}, and the byte of data is written into or read from storage location X over lines D_0 through D_7. D_7 carries the MSB of the byte of data, and D_0 carries the LSB. This shows that a byte of data is accessed by the 8088 in one bus cycle. A memory cycle for an 8088 running at 5 MHz with no wait states takes 800 ns.

When a word of data is to be transferred between the 8088 and memory, we must perform two accesses of memory, reading or writing a byte in each access. Figure 8–16(b) illustrates how the word storage location starting at address X is accessed. Two bus cycles are required to access a word of data. During the first bus cycle, the least significant byte of the word, located at address X, is accessed. Again the address is applied to the memory bank over A_0 through A_{19}, and the byte of data is transferred to or from storage location X over D_0 through D_7.

Next, the 8088 automatically increments the address so that it now points to byte address X + 1. This address points to the next consecutive byte storage location in memory, which corresponds to the most significant byte of the word of data at X. Now a second memory bus cycle is initiated. During this second cycle, data are written into or read from the storage location at address X + 1. Since word accesses of memory take two bus cycles instead of one, it takes 1.6 ms to access a word of data when the 8088 is operating at a 5-MHz clock rate with no wait states.

The 8086 microprocessor performs byte and word data transfers differently from the 8088. Let us next examine the data transfers that can take place in an 8086-based microcomputer.

Figure 8–17(a) shows that when a byte-memory operation is performed to address X, an even-addressed storage location in the low bank is accessed. Therefore, A_0 is set to logic 0 to enable the low bank of memory and $\overline{BHE}$ to logic 1 to disable the high bank. As shown in the block diagram, data are transferred to or from the lower bank over data bus lines D_0 through D_7. Line D_7 carries the MSB of the byte, and D_0 the LSB.

On the other hand, to access a byte of data at an odd address such as X + 1 in Fig. 8–17(b), A_0 is set to logic 1 and $\overline{BHE}$ to logic 0. This enables the high bank of memory and disables the low bank. Data are transferred between the 8086 and the high bank over bus lines D_8 through D_{15}. Here D_{15} represents the MSB and D_8 the LSB.

Whenever an even-addressed word of data is accessed, both the high and low banks are accessed at the same time. Figure 8–17(c) illustrates how a word at even address X is accessed. Note that both A_0 and $\overline{BHE}$ equal 0; therefore, both banks are enabled. In this case, bytes of data are transferred from or to both banks at the same time. This 16-bit word is transferred over the complete data bus D_0 through D_{15}. The bytes of an even-addressed word are said to be aligned and can be transferred with a memory operation that takes just one bus cycle.

A word at an odd-addressed boundary is said to be unaligned. That is, the least significant byte is at the lower address location in the high memory bank. This is demonstrated in Fig. 8–17(d). Here we see that the odd byte of the word is located at address X + 1 and the even byte at address X + 2.

Two bus cycles are required to access an unaligned word. During the first bus cycle, the odd byte of the word, which is located at address X + 1 in the high bank, is accessed. This is accompanied by select signals $A_0 = 1$ and $\overline{BHE} = 0$ and a data transfer over D_8 through D_{15}. Even though the data transfer uses data lines D_8 through D_{15}, to the processor it is the low byte of the addressed data word.

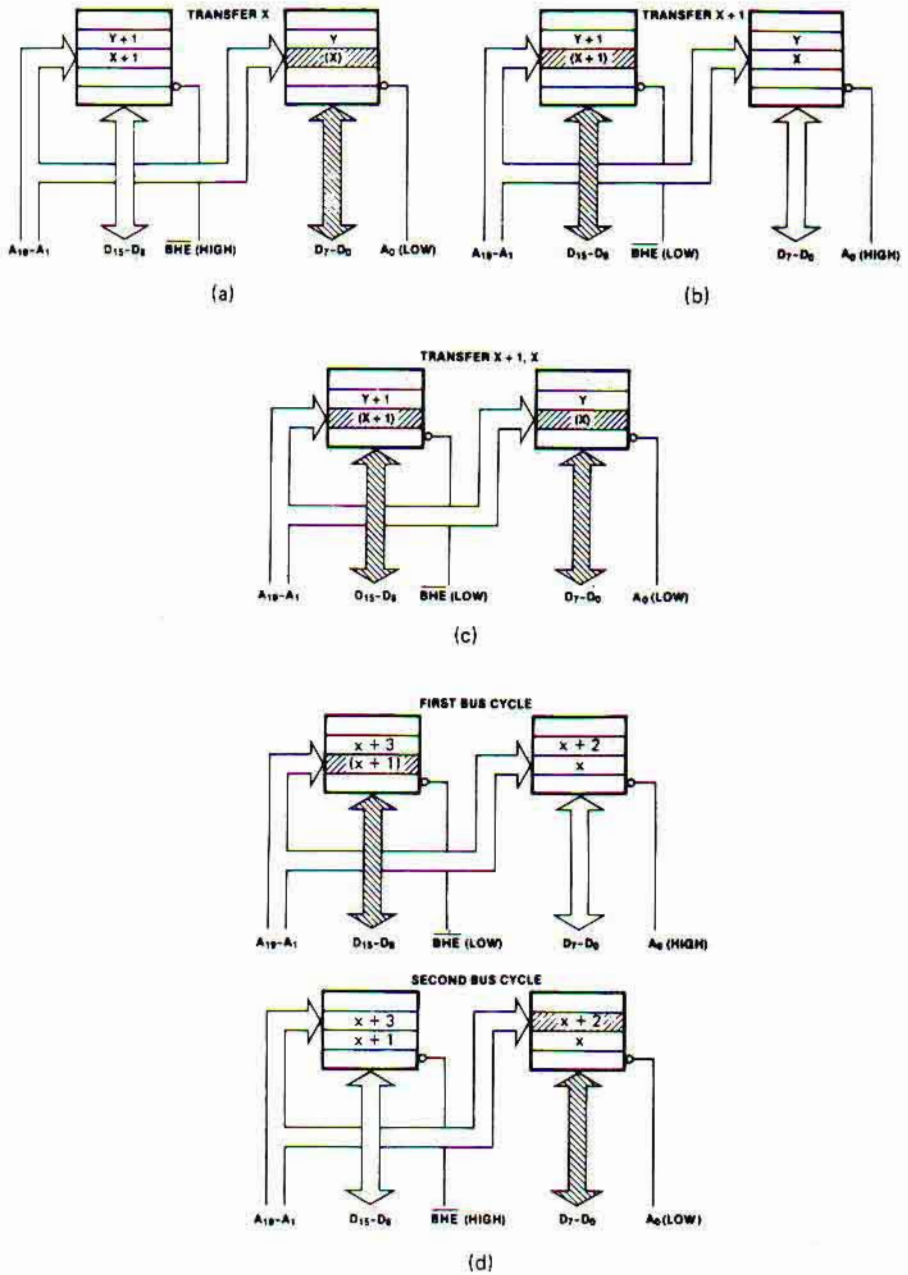


Figure 8–17 (a) Even-address byte transfer by the 8086. (Reprinted with permission of Intel Corporation, © 1979) (b) Odd-address byte transfer by the 8086. (Reprinted with permission of Intel Corporation, © 1979) (c) Even-address word transfer by the 8086. (Reprinted with permission of Intel Corporation, © 1979) (d) Odd-address word transfer by the 8086. (Reprinted with permission of Intel Corporation, © 1979)

Next, the 8086 automatically increments the address so that $A_0 = 0$. This represents the next address in memory, which is even. Then a second memory bus cycle is initiated. During this second cycle, the even byte located at X + 2 in the low bank is accessed. The data transfer takes place over bus lines D_0 through D_7. This transfer is accompanied by $A_0 = 0$ and $\overline{BHE} = 1$. To the processor, this is the high byte of the word of data.

EXAMPLE 8.6

Is the word at memory address 01231_{16} of an 8086-based microcomputer aligned or misaligned? How many bus cycles are required to read it from memory?

Solution

The first byte of the word is the second byte at the aligned-word address 01230_{16}. Therefore, the word is misaligned and requires two bus cycles to be read from memory.

▲ 8.9 ADDRESS BUS STATUS CODES

Whenever a memory bus cycle is in progress, an address bus status code S_4S_3 is output by the processor. The status code is multiplexed with address bits A_{17} and A_{16}. This two-bit code is output at the same time the data are carried over the data lines.

Bits S_4 and S_3 together form a 2-bit binary code that identifies which one of the four segment registers was used to generate the physical address that was output during the address period in the current bus cycle. The four *address bus status codes* are listed in Fig. 8–4. Here we find that code $S_4S_3 = 00$ identifies the extra segment register, 01 identifies the stack segment register, 10 identifies the code segment register, and 11 identifies the data segment register.

These status codes are output in both the minimum and the maximum modes. The codes can be examined by external circuitry. For example, they can be decoded with external circuitry to enable separate 1Mbyte address spaces for ES, SS, CS, and DS. In this way, the memory address reach of the microprocessor can be expanded to 4Mbytes.

▲ 8.10 MEMORY CONTROL SIGNALS

Earlier in the chapter we saw that similar control signals are produced in the maximum and minimum mode. Moreover, we found that in the minimum mode, the 8088 and 8086 microprocessors produce all the control signals. But in the maximum mode, the 8288 bus controller produces them. Here we will look more closely at each of these signals and their functions with respect to memory interface operation.

Minimum-Mode Memory Control Signals

In the 8088 microcomputer system shown in Fig. 8–18, which is configured for the minimum mode of operation, we find that the control signals provided to support the interface to the memory subsystem are ALE, IO/$\overline{M}$, DT/$\overline{R}$, $\overline{RD}$, $\overline{WR}$, and $\overline{DEN}$. These

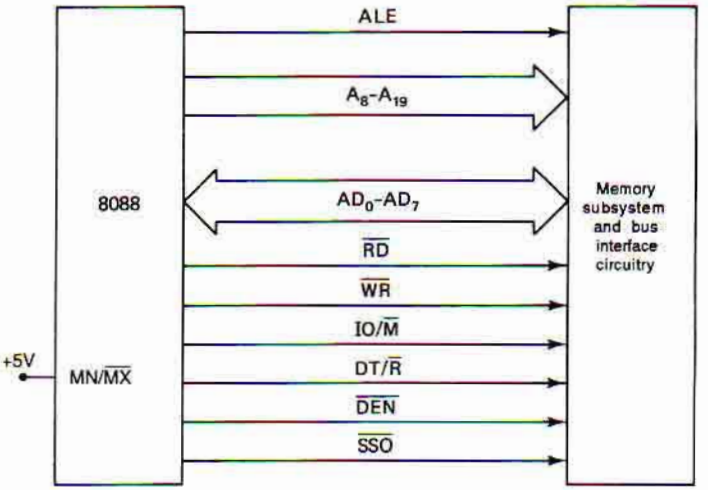


Figure 8–18 Minimum-mode 8088 memory interface.

control signals are required to tell the memory subsystem when the bus is carrying a valid address, in which direction data are to be transferred over the bus, when valid write data are on the bus, and when to put read data on the bus. For example, *address latch enable* (ALE) signals external circuitry that a valid address is on the bus. It is a pulse to the 1 logic level and is used to latch the address in external circuitry.

The *input-output/memory* (IO/$\overline{M}$) and *data transmit/receive* (DT/$\overline{R}$) lines signal external circuitry whether a memory or I/O bus cycle is in progress and whether the 8088 will transmit or receive data over the bus. During all memory bus cycles, IO/$\overline{M}$ is held at the 0 logic level. The 8088 switches DT/$\overline{R}$ to logic 1 during the data transfer part of the bus cycle, the bus is in the transmit mode, and data are written into memory. On the other hand, it sets DT/$\overline{R}$ to logic 0 to signal that the bus is in the receive mode, which corresponds to reading of memory.

The signals *read* ($\overline{RD}$) and *write* ($\overline{WR}$) identify that a read or write bus cycle, respectively, is in progress. The 8088 switches $\overline{WR}$ to logic 0 to signal memory that a write cycle is taking place over the bus. On the other hand, $\overline{RD}$ is switched to logic 0 whenever a read cycle is in progress. During all memory operations, the 8088 produces one other control signal, *data enable* ($\overline{DEN}$). Logic 0 at this output is used to enable the data bus.

Status line $\overline{SSO}$ is also part of the minimum-mode memory interface. The logic level that is output on this line during read bus cycles identifies whether a code or data access is in progress. $\overline{SSO}$ is set to logic 0 whenever instruction code is read from memory.

The control signals for the 8086's minimum-mode memory interface differ in three ways. First, the 8088's IO/$\overline{M}$ signal is replaced by the memory/input-output (M/$\overline{IO}$) signal. Whenever a memory bus cycle is in progress, the M/$\overline{IO}$ output is switched to logic 1. Second, the signal $\overline{SSO}$ is removed from the interface. Third, a new signal, *bank high enable* ($\overline{BHE}$), has been added to the interface. $\overline{BHE}$ is used as a select input for the high bank of memory in the 8086's memory subsystem. That is, logic 0 is output on this line

during the address part of all the bus cycles in which data in the high-bank part of memory is to be accessed.

Maximum-Mode Memory Control Signals

When the 8088 is configured to work in the maximum mode, it does not directly provide all the control signals to support the memory interface. Instead, an external bus controller, the 8288, provides memory commands and control signals. Figure 8–19 shows an 8088 connected in this way.

Specifically, the $\overline{WR}$, IO/$\overline{M}$, DT/$\overline{R}$, $\overline{DEN}$, ALE, and $\overline{SSO}$ signal lines on the 8088 are changed. They are replaced with *multiprocessor lock* ($\overline{LOCK}$) signal, a *bus status code* ($\overline{S}_2\overline{S}_1\overline{S}_0$), and a *queue status code* (QS_1QS_0). The 8088 still does produce the signal $\overline{RD}$, which provides the same function as it did in minimum mode.

The 3-bit bus status code $\overline{S}_2\overline{S}_1\overline{S}_0$ is output prior to the initiation of each bus cycle. It identifies which type of bus cycle is to follow. This code is input to the 8288 bus controller. Here it is decoded to identify which type of bus cycle command signals must be generated.

Figure 8–20 shows the relationship between the bus status codes and the types of bus cycles produced. Also shown in this chart are the names of the corresponding command signals that are generated at the outputs of the 8288. For instance, the input code $\overline{S}_2\overline{S}_1\overline{S}_0$ equal to 100 indicates that an instruction fetch bus cycle is to take place. Since the instruction fetch is a memory read, the 8288 makes the *memory read command* ($\overline{MRDC}$) output switch to logic 0.

Another bus command provided for the memory subsystem is $\overline{S}_2\overline{S}_1\overline{S}_0$ equal to 110. This represents a memory write cycle and it causes both the *memory write command*

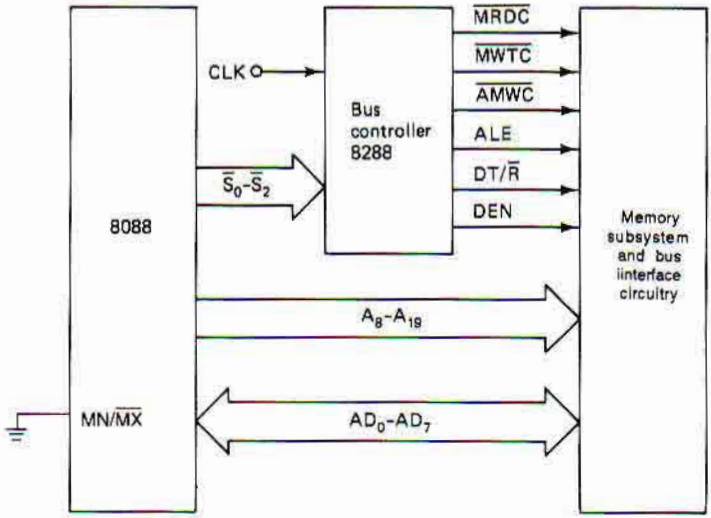


Figure 8–19 Maximum-mode 8088 memory interface.

Status Inputs			CPU Cycle	8288 Command
$\overline{S}_2$	$\overline{S}_1$	$\overline{S}_0$		
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

Figure 8–20 Memory bus cycle status codes produced in maximum mode. (Reprinted with permission of Intel Corporation, © 1979)

($\overline{MWTC}$) and *advanced memory write command* ($\overline{AMWC}$) outputs to switch to the 0 logic level.

The other control outputs produced by the 8288 are DEN, DT/$\overline{R}$, and ALE. These signals provide the same functions as those produced by the corresponding pins on the 8088 in the minimum system mode.

The two status signals, QS_0 and QS_1, form an instruction queue code. This code tells the external circuitry what type of information was removed from the queue during the previous clock cycle. Figure 8–8 shows the four different queue statuses. For instance, $QS_1QS_0 = 01$ indicates that the first byte of an instruction was taken from the queue. The next byte of the instruction that is fetched is identified by queue status code 11. Whenever the queue is reset (e.g., due to a transfer of control) the reinitialization code 10 is output. Similarly, if no queue operation occurred, status code 00 is output.

The *bus priority lock* ($\overline{LOCK}$) signal, as shown in the interface, can be used as an input to a bus arbiter. The bus arbiter is used to lock other processors off the system bus during accesses of common system resources such as *global memory* in a multiprocessor system. The READY signal is used to interface slow memory devices.

All of the memory control signals we just described for the 8088-based microcomputer system serve the same function in the maximum-mode 8086 microcomputer. However, there is one additional control signal in the 8086's memory interface, the $\overline{BHE}$. The $\overline{BHE}$ performs the same function as it did in the minimum-mode system. That is, it is used as an enable input to the high bank of memory.

8.11 READ AND WRITE BUS CYCLES

In the preceding section we introduced the status and control signals associated with the memory interface. Here we continue by studying the sequence in which they occur during the read and write bus cycles of memory.

Read Cycle

Figure 8–21 shows the memory interface signals of a minimum-mode 8088 system. Here their occurrence is illustrated relative to the four time states T_1, T_2, T_3, and T_4 of the 8088's bus cycle. Let us trace the events that occur as data or instructions are read from memory.

The *read bus cycle* begins with state T_1. During this period, the 8088 outputs the 20-bit address of the memory location to be accessed on its multiplexed address/data bus AD_0 through AD_7, A_8 through A_{15}, and multiplexed lines A_{16}/S_3 through A_{19}/S_6. Note that at the same time a pulse is also produced at ALE. The trailing edge or the high level of this pulse should be used to latch the address in external circuitry.

Also we see that at the start of T_1, signals IO/$\overline{M}$ and DT/$\overline{R}$ are set to the 0 logic level. This indicates to circuitry in the memory subsystem that a memory cycle is in progress and that the 8088 is going to receive data from the bus. Status $\overline{SSO}$ is also out-

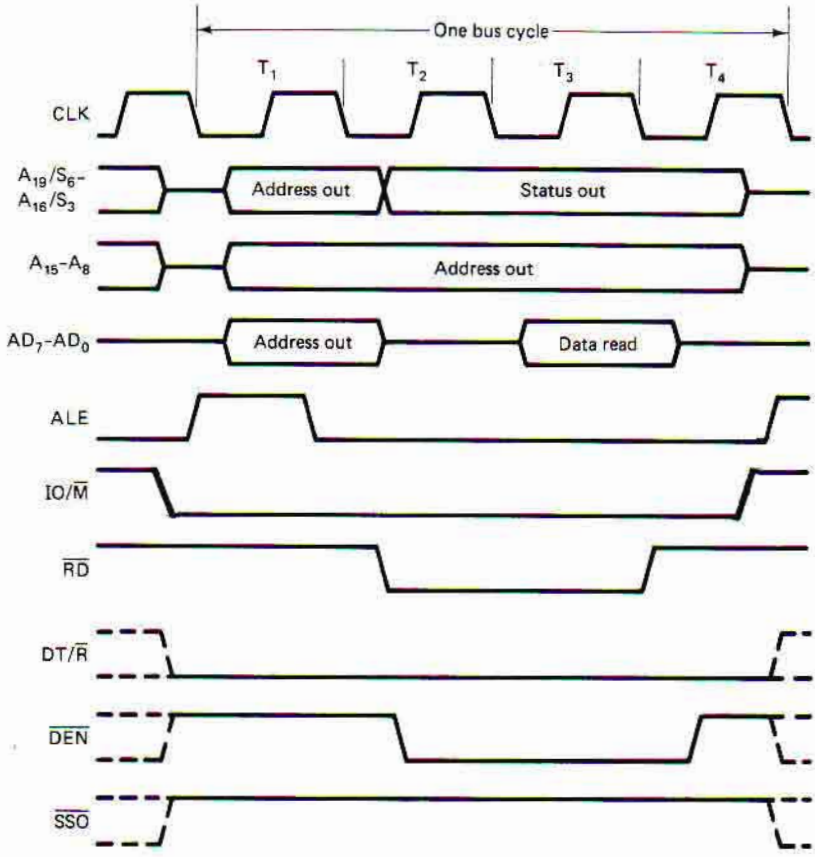


Figure 8–21 Minimum-mode memory read bus cycle of the 8088. (Reprinted with permission of Intel Corporation, © 1979)

put at this time. Note that all three of these signals are maintained at these logic levels throughout all four periods of the bus cycle.

Beginning with state T_2, status bits S_3 through S_6 are output on the upper four address bus lines A_{16} through A_{19}. Remember that bits S_3 and S_4 identify to external circuitry which segment register was used to generate the address just output. This status information is maintained through periods T_3 and T_4. The part of the address output on address bus lines A_8 through A_{15} is maintained through states T_2, T_3, and T_4. On the other hand, address/data bus lines AD_0 through AD_7 are put in the high-Z state during T_2.

Late in period T_2, $\overline{RD}$ is switched to logic 0. This indicates to the memory subsystem that a read cycle is in progress. $\overline{DEN}$ is switched to logic 0 to enable external circuitry to allow the data to move from memory onto the microprocessor's data bus.

As shown in the waveforms, input data are read by the 8088 during T_3. The memory must provide valid data during T_3 and maintain it until after the processor terminates the read operation. As Fig. 8–21 shows, it is in T_4 that the 8088 switches $\overline{RD}$ to the inactive 1 logic level to terminate the read operation. $\overline{DEN}$ returns to its inactive logic level late during T_4 to disable the external circuitry, which allows data to move from memory to the processor. The read cycle is now complete.

A timing diagram for the 8086's memory read cycle is given in Fig. 8–22(a). Comparing these waveforms to those of the 8088 in Fig. 8–21, we find just four differences; $\overline{BHE}$ is output along with the address during T_1; the data read by the 8086 during T_3 can be carried over all 16 data bus lines; M/$\overline{IO}$, which replaces IO/$\overline{M}$, is switched to logic 1

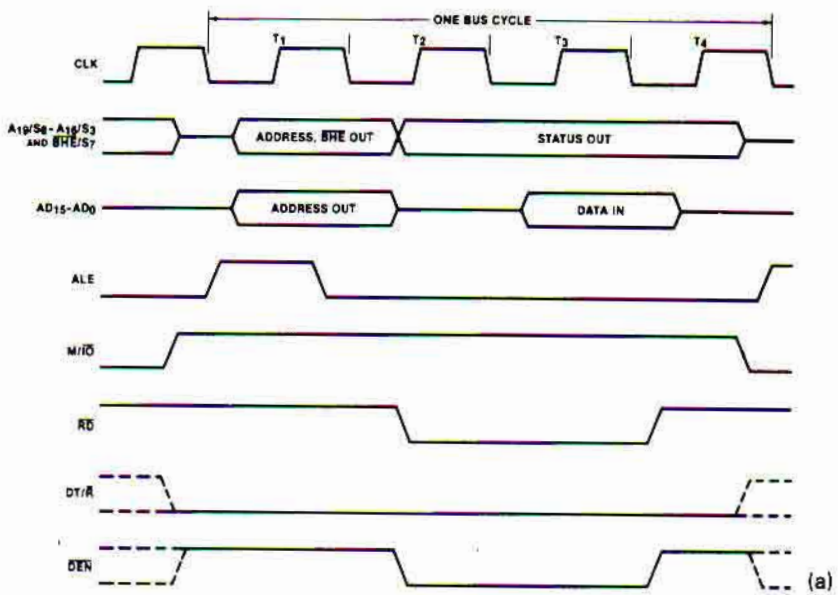


Figure 8–22 (a) Minimum-mode memory read bus cycle of the 8086. (Reprinted with permission of Intel Corporation, © 1979) (b) Maximum-mode memory read bus cycle of the 8086. (Reprinted with permission of Intel Corporation, © 1979)

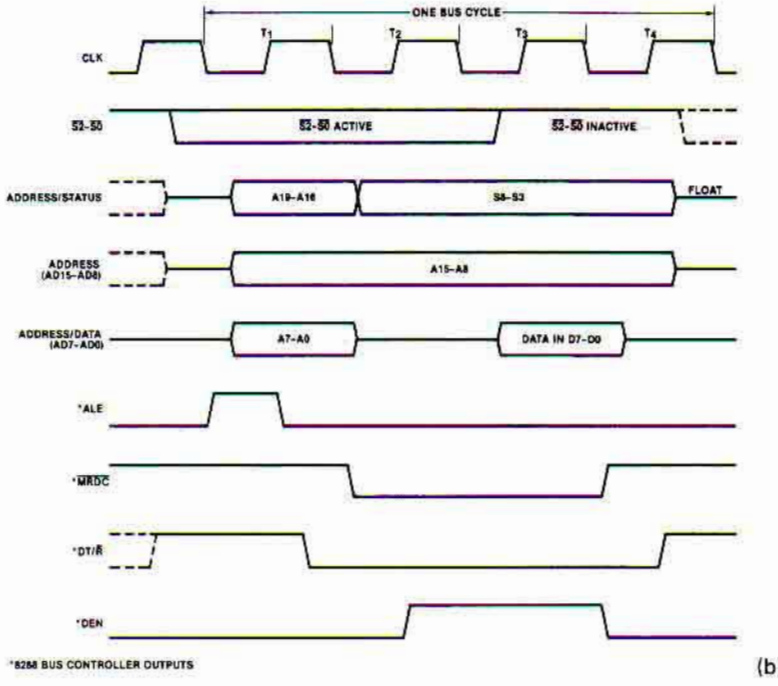


Figure 8–22 (continued)

at the beginning of T_1 and is held at this level for the duration of the bus cycle; and the $\overline{SSO}$ status signal is not produced.

Figure 8–22(b) shows a read cycle of 8-bit data in a maximum-mode 8086-based microcomputer system. These waveforms are similar to those given for the minimum-mode read cycle in Fig. 8–22(a). Comparing these two timing diagrams, we see that the address and data transfers that take place are identical. In fact, the only difference found in the maximum-mode waveforms is that a bus cycle status code, $\overline{S}_2\overline{S}_1\overline{S}_0$, is output just prior to the beginning of the bus cycle. This status information is decoded by the 8288 to produce control signals ALE, $\overline{MRDC}$, DT/$\overline{R}$, and DEN.

Write Cycle

Figure 8–23(a) illustrates the *write bus cycle* timing of the 8088 in minimum mode. It is similar to that given for a read cycle in Fig. 8–21. Looking at the write cycle waveforms, we find that during T_1 the address is output and latched with the ALE pulse. This is identical to the read cycle. Moreover, IO/$\overline{M}$ is set to logic 0 to indicate that a memory cycle is in progress and status information is output at $\overline{SSO}$. However, this time DT/$\overline{R}$ is switched to logic 1. This signals external circuits that the 8088 is going to transmit data over the bus.

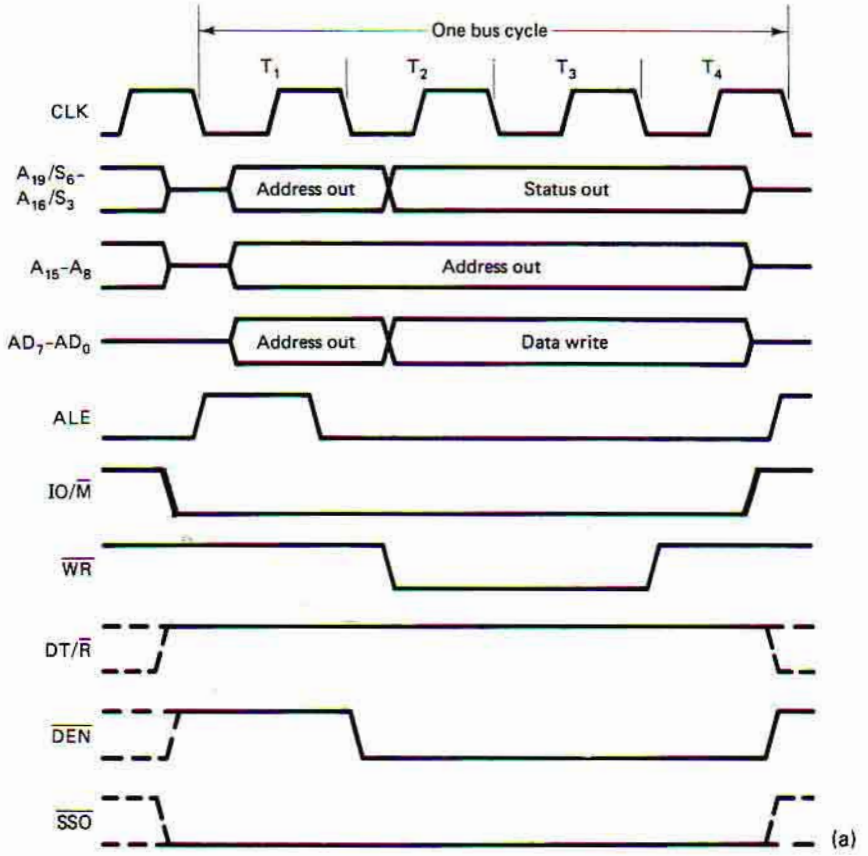


Figure 8–23 (a) Minimum-mode memory write bus cycle of the 8088. (Reprinted with permission of Intel Corporation, © 1979) (b) Maximum-mode memory write bus cycle of the 8086. (Reprinted with permission of Intel Corporation, © 1979)

As T_2 starts, the 8088 switches $\overline{WR}$ to logic 0. This tells the memory subsystem that a write operation is to follow over the bus. The 8088 puts the data on the bus late in T_2 and maintains the data valid through T_4. The writing of data into memory starts as $\overline{WR}$ becomes 0, and continues as it changes to 1 early in T_4. $\overline{DEN}$ enables the external circuitry to provide a path for data from the processor to the memory. This completes the write cycle.

Just as we described for the read bus cycle, the write cycle of the 8086 differs from that of the 8088 in four ways; again, $\overline{SSO}$ is not produced; $\overline{BHE}$ is output along with the address; data are carried over all 16 data bus lines; and finally, M/$\overline{IO}$ is the complement of the 8088's IO/$\overline{M}$ signal. The waveforms in Fig. 8–23(b) illustrate a write cycle of word data in a maximum-mode 8086 system.

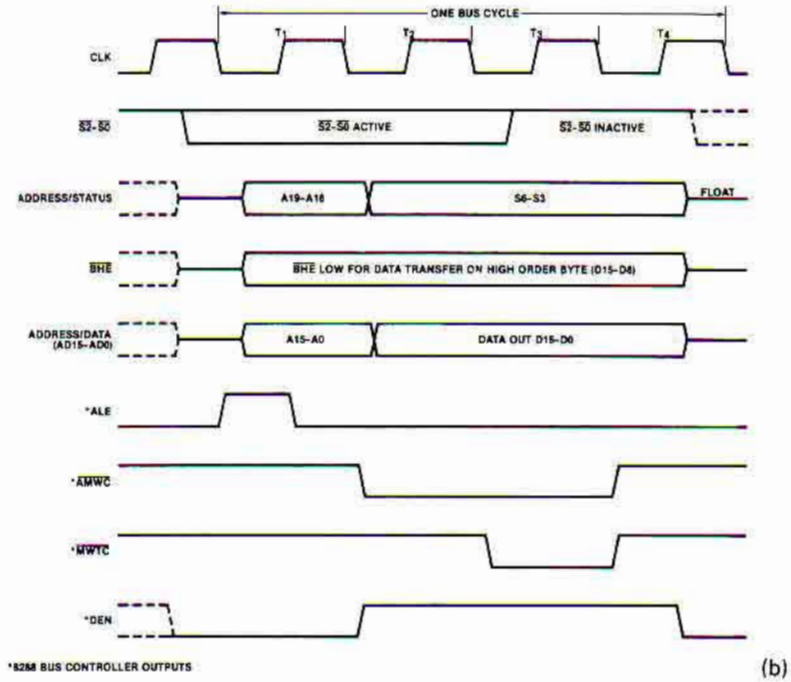


Figure 8–23 (continued)

▲ 8.12 MEMORY INTERFACE CIRCUITS

This section describes the memory interface circuits of an 8086-based microcomputer system. The 8086 system was selected instead of an 8088 microcomputer because it is more complex. Figure 8–24 shows a memory interface diagram for a maximum-mode 8086-based microcomputer system. Here we find that the interface includes the 8288 bus controller, address bus latches and an address decoder, data bus transceiver/buffers, and bank read and write control logic. The 8088 microcomputer is simpler in that the interface does not require bank write control logic because its address space is organized as a single bank.

Looking at Fig. 8–24, we see that bus status code signals $\overline{S}_2$, $\overline{S}_1$, and $\overline{S}_0$, which are outputs of the 8086, are supplied directly to the 8288 bus controller. Here they are decoded to produce the command and control signals needed to coordinate data transfers over the bus. Figure 8–20 highlights the status codes that relate to the memory interface. For example, the code $\overline{S}_2\overline{S}_1\overline{S}_0 = 101$ indicates that a data memory read bus cycle is in progress. This code makes the $\overline{\text{MRDC}}$ command output of the bus control logic switch to logic 0. Note in Fig. 8–24 that $\overline{\text{MRDC}}$ is applied to the bank read control logic.

Next let us look at how the address bus is latched, buffered, and decoded. Looking at Fig. 8–24, we see that address lines A_0 through A_{19} are latched along with control signal $\overline{\text{BHE}}$ in the address bus latch. The latched address lines A_{17L} through A_{19L} are

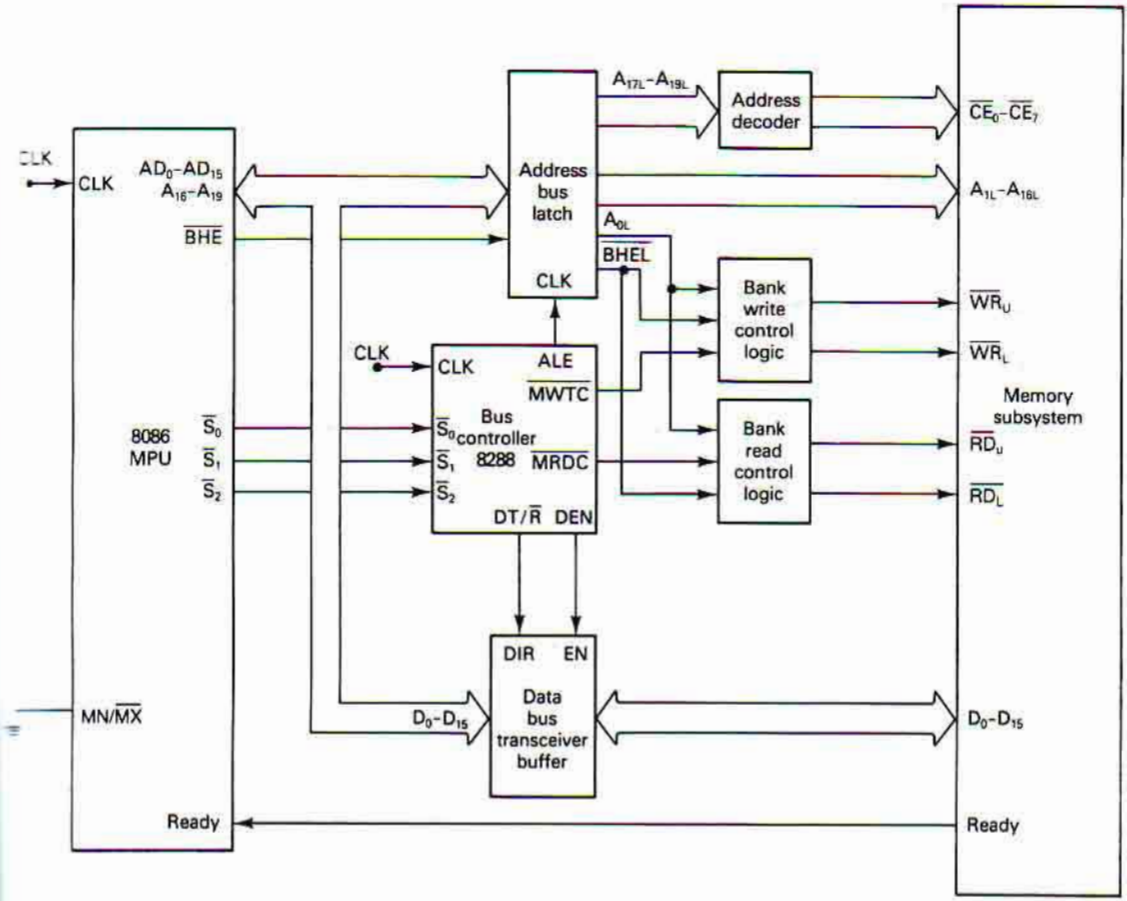


Figure 8–24 Memory interface block diagram.

decoded to produce chip enable outputs $\overline{CE}_0$ through $\overline{CE}_7$. Notice that the 8288 bus controller produces the address latch enable (ALE) control signal from $\overline{S}_2\overline{S}_1\overline{S}_0$. ALE is applied to the CLK input of the latches and strobes the bits of the address and bank high enable signal into the address bus latches. The address latch devices buffer these signals. Latched address lines A_{1L} through A_{16L} and $\overline{CE}_0$ through $\overline{CE}_7$ are applied directly to the memory subsystem.

During read bus cycles, the $\overline{MRDC}$ output of the bus control logic enables the bytes of data at the outputs of the memory subsystem onto data bus lines D_0 through D_{15}. During read operations from memory, the bank read control logic determines whether the data are read from one of the two memory banks or from both. This depends on whether a byte- or word-data transfer is taking place over the bus.

Similarly during write bus cycles, the $\overline{MWTC}$ output of the bus control logic enables bytes of data from the data bus D_0 through D_{15} to be written into the memory. The bank write control logic determines to which memory bank the data are written.

Note in Fig. 8–24 that in the bank write control logic the latched bank high enable signal $\overline{BHEL}$ and address line A_{0L} are gated with the memory write command signal $\overline{MWTC}$ to produce a separate write enable signal for each bank. These signals are denoted as $\overline{WR}_U$ and $\overline{WR}_L$. For example, if a word of data is to be written to memory over data bus lines D_0 through D_{15}, both $\overline{WR}_U$ and $\overline{WR}_L$ are switched to their active 0 logic level. Similarly the memory read control logic uses $\overline{MRDC}$, A_{0L}, and $\overline{BHEL}$ to generate $\overline{RD}_U$ and $\overline{RD}_L$ signals for bank read control.

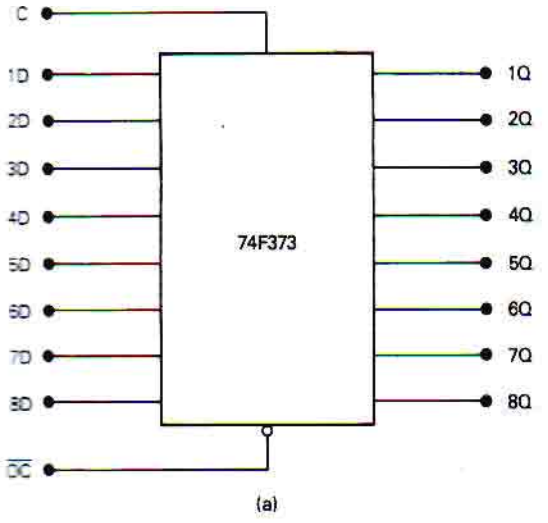
The bus transceivers control the direction of data transfer between the MPU and memory subsystem. In Fig. 8–24, we see that the operation of the transceivers is controlled by the DT/$\overline{R}$ and DEN outputs of the bus controller. DEN is applied to the EN input of the transceivers and enables them for operation. This happens during all read and write bus cycles. DT/$\overline{R}$ selects the direction of data transfer through the devices. Note that it is supplied to the DIR input of the data bus transceivers. When a read cycle is in progress, DT/$\overline{R}$ is set to 0 and data are passed from the memory subsystem to the MPU. On the other hand, when a write cycle is taking place, DT/$\overline{R}$ is switched to logic 1 and data are carried from the MPU to the memory subsystem.

Address Bus Latches and Buffers

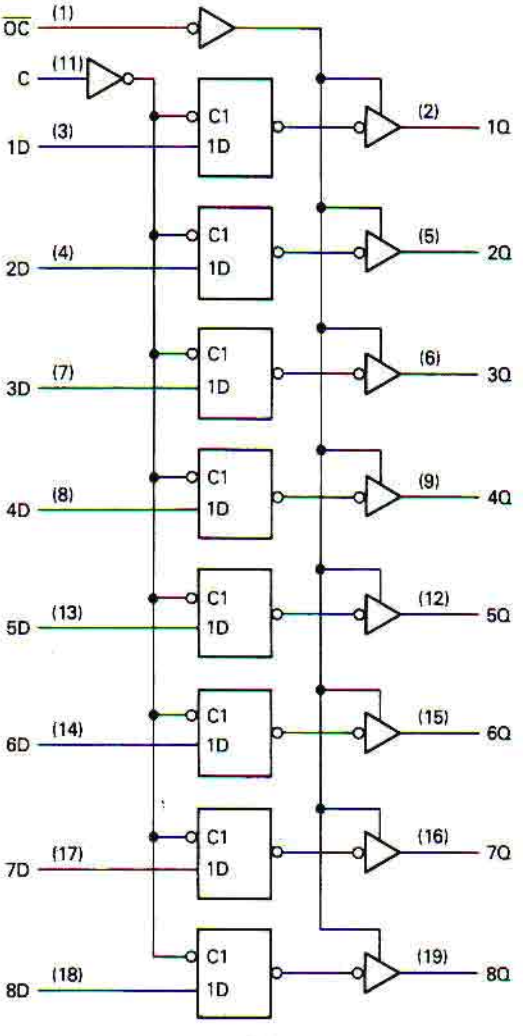
The 74F373 is an example of an octal latch device that can be used to implement the *address latch* section of the 8086's memory interface circuit. A block diagram of this device is shown in Fig. 8–25(a) and its internal circuitry is shown in Fig. 8–25(b). Note that it accepts eight inputs: 1D through 8D. As long as the clock (C) input is at logic 1, the outputs of the D-type flip-flops follow the logic level of the data applied to their corresponding inputs. When C is switched to logic 0, the current contents of the D-type flip-flops are latched. The latched information in the flip-flops is not output at data outputs 1Q through 8Q unless the output-control ($\overline{OC}$) input of the buffers that follow the latches is at logic 0. If $\overline{OC}$ is at logic 1, the outputs are in the high-impedance state. Figure 8–25(c) summarizes this operation.

In the 8086 microcomputer system, the 20 address lines (AD_0–AD_{15}, A_{16}–A_{19}) and the bank high enable signal $\overline{BHE}$ are normally latched in the address bus latch. The circuit configuration shown in Fig. 8–26 can be used to latch these signals. Fixing $\overline{OC}$ at the 0 logic level permanently enables latched outputs A_{0L} through A_{19L} and $\overline{BHEL}$. Moreover, the address information is latched at the outputs as the ALE signal from the bus controller returns to logic 0—that is, when the CLK input of all devices is switched to logic 0.

In general, it is important to minimize the propagation delay of the address signals as they go through the bus interface circuit. The switching property of the 74F373 latches that determine this delay for the circuit of Fig. 8–26 is called *enable-to-output propagation delay* and has a maximum value of 13 ns. By selecting fast latches—that is, latches with a shorter propagation delay time—a maximum amount of the 8086's bus cycle time is preserved for the access time of the memory devices. In this way slower, lower cost memory ICs can be used. These latches also provide buffering for the 8086's address lines. The outputs of the latch can sink a maximum of 24 mA.



(a)



(b)

Inputs			Output
$\overline{OC}$	Enable C	D	Q
L	H	H	H
L	H	L	L
L	L	X	Q_0
H	X	X	Z

(c)

Figure 8–25 (a) Block diagram of an octal D-type latch. (b) Circuit diagram of the 74F373. (Courtesy of Texas Instruments Incorporated) (c) Operation of the 74F373. (Courtesy of Texas Instruments Incorporated).

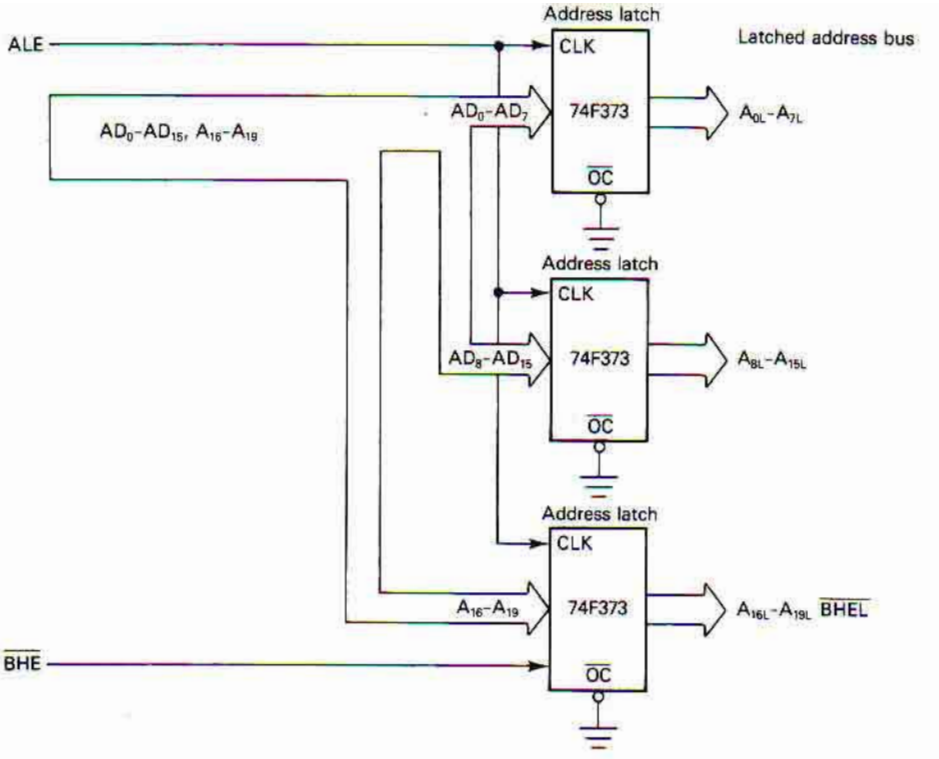


Figure 8–26 Address latch circuit.

Bank Write and Bank Read Control Logic

The memory of the 8086 microcomputer is organized in upper and lower banks. It requires separate write and read control signals for the two banks. The logic circuit in Fig. 8–27 shows how the bank write control signals, $\overline{WR}_U$ for the upper bank and $\overline{WR}_L$ for the lower bank can be generated from the bus controller signals $\overline{WRTC}$, the address bus latch signals A_{0L} and $\overline{BHEL}$. Two OR gates are used for this purpose.

Similar to the bank write control logic circuit, the bank read control logic circuit can be designed to generate $\overline{RD}_U$, the read for the upper bank of memory, and $\overline{RD}_L$, the read for the lower bank. Figure 8–28 illustrates such a circuit. Note that the circuit uses the $\overline{MRDC}$ signal from the bus controller.

Data Bus Transceivers

The *data bus transceiver* block of the bus interface circuit can be implemented with 74F245 octal bus transceiver ICs. Figure 8–29(a) shows a block diagram of this device. Note that its bidirectional input/output lines are called A_1 through A_8 and B_1 through B_8. Looking at the circuit diagram in Fig. 8–29(b), we see that the $\overline{G}$ input is used to enable

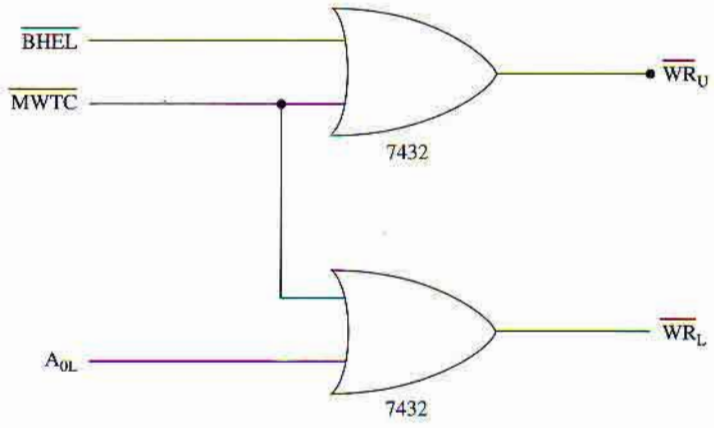


Figure 8–27 Bank write control logic.

the buffer for operation. On the other hand, the logic level at the direction (DIR) input selects the direction in which data are transferred through the device. For instance, logic 0 at this input sets the transceiver to pass data from the B lines to the A lines. Switching DIR to logic 1 reverses the direction of data transfer.

Figure 8–30 shows a circuit that implements the data bus transceiver block of the bus interface circuit using the 74F245. For the 16-bit data bus of the 8086 microcomputer, two devices are required. Here the DIR input is driven by the signal data transmit/receive (DT/$\overline{R}$), and $\overline{G}$ is supplied by data bus enable (DEN). These signals are outputs of the 8288 bus controller.

Another key function of the data bus transceiver circuit is to buffer the data bus lines. This capability is defined by how much current the devices can sink at their outputs. The I_{OL} rating of the 74F245 is 64 mA.

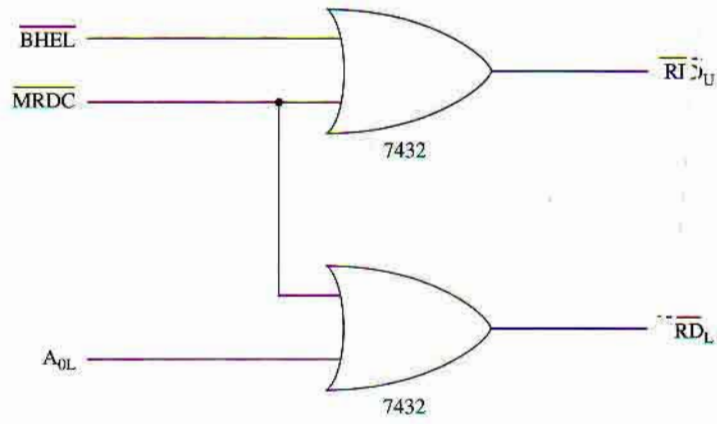


Figure 8–28 Bank read control logic.

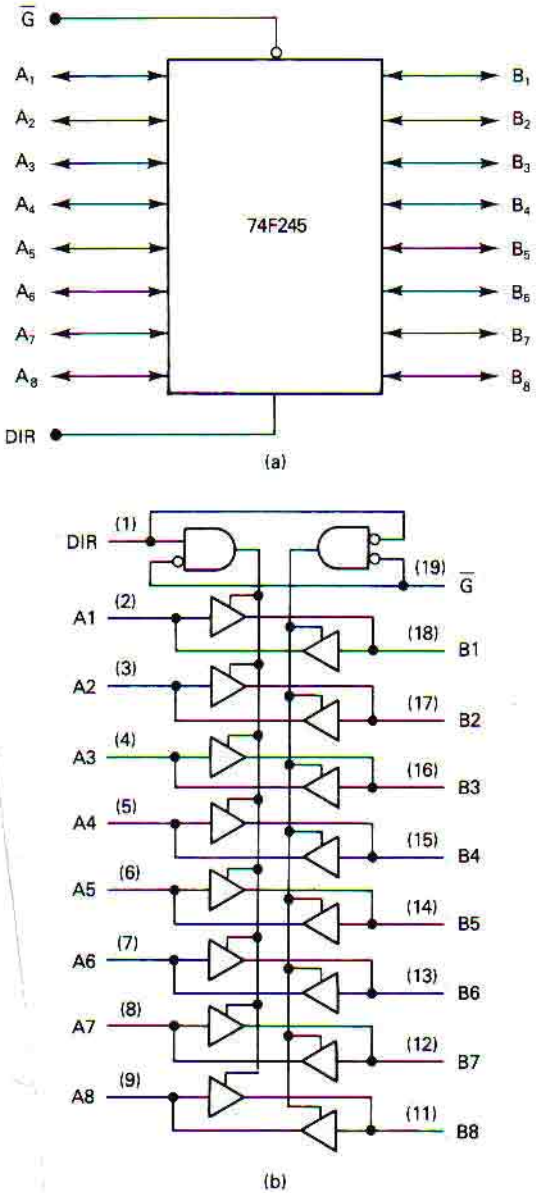


Figure 8–29 (a) Block diagram of the 74F245 octal bidirectional bus transceiver. (b) Circuit diagram of the 74F245. (Courtesy of Texas Instruments Incorporated)

Address Decoders

As shown in Fig. 8–31, the *address decoder* in the 8086 microcomputer system is located at the output side of the address latch. A typical device used to perform this decode function is the 74F139 dual 2-line to 4-line decoder. Figures 8–32(a) and (b) show a block diagram and circuit diagram for this device, respectively. When the enable ($\overline{G}$) input is at its active 0 logic level, the output corresponding to the code at the BA inputs

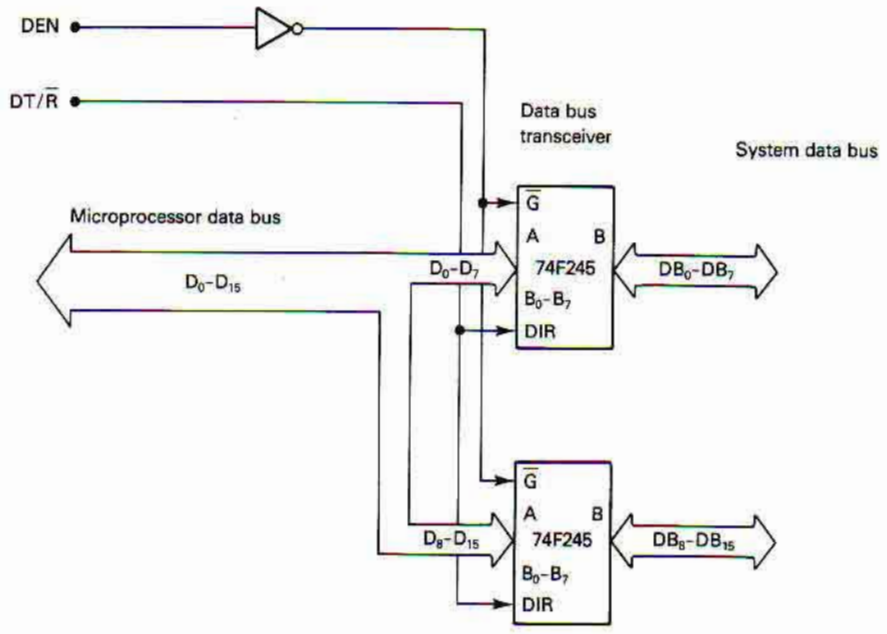


Figure 8–30 Data bus transceiver circuit.

switches to the 0 logic level. For instance, when BA = 01, output Y_1 is logic 0. The table in Fig. 8–32(c) summarizes the operation of the 74F139.

The circuit in Fig. 8–33 employs the address decoder configuration shown in Fig. 8–31. Note that address lines A_{17L} and A_{18L} are applied to the A and B inputs of the 74F139 decoder. The address line A_{19L} is used to enable one of the decoders and $\overline{A}_{19L}$, obtained using an inverter, enables the second decoder of the 74F139. Each decoder generates four chip enable (CE) outputs. Thus both decoders of the 74F139 together produce the eight outputs $\overline{CE}_0$ through $\overline{CE}_7$.

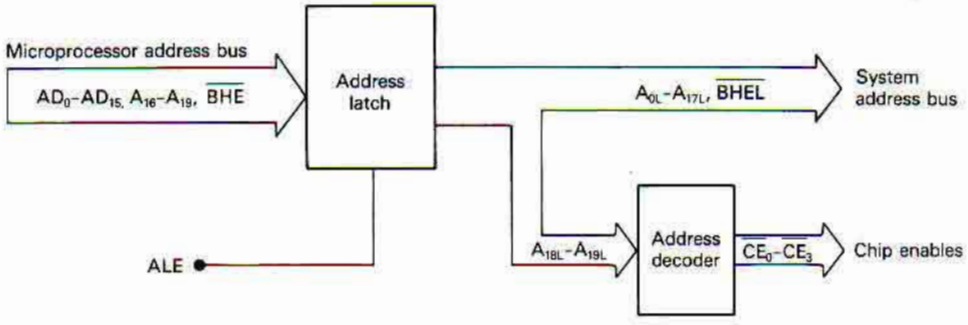
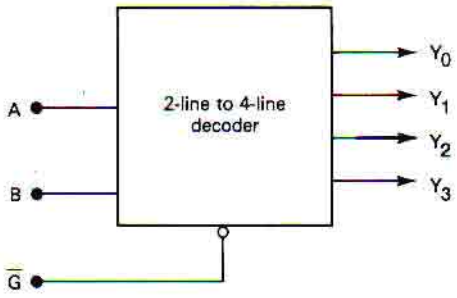
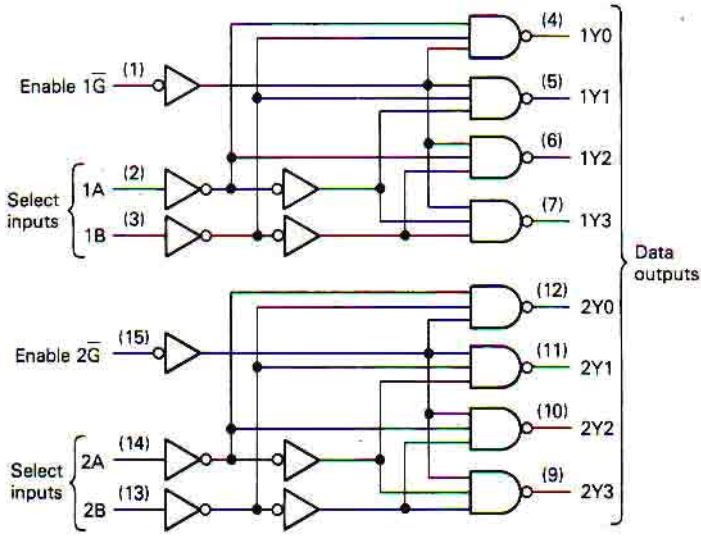


Figure 8–31 Address bus configuration with address decoding.



(a)



(b)

INPUTS			OUTPUTS			
ENABLE	SELECT					
$\overline{G}$	B	A	Y0	Y1	Y2	Y3
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	L	H	H	L	H	H
L	H	L	H	H	L	H
L	H	H	H	H	H	L

(c)

Figure 8–32 (a) Block diagram of the 74F139 2-line to 4-line decoder/demultiplexer. (b) Circuit diagram of the 74F139. (Courtesy of Texas Instruments Incorporated) (c) Operation of the 74F139 decoder. (Courtesy of Texas Instruments Incorporated)

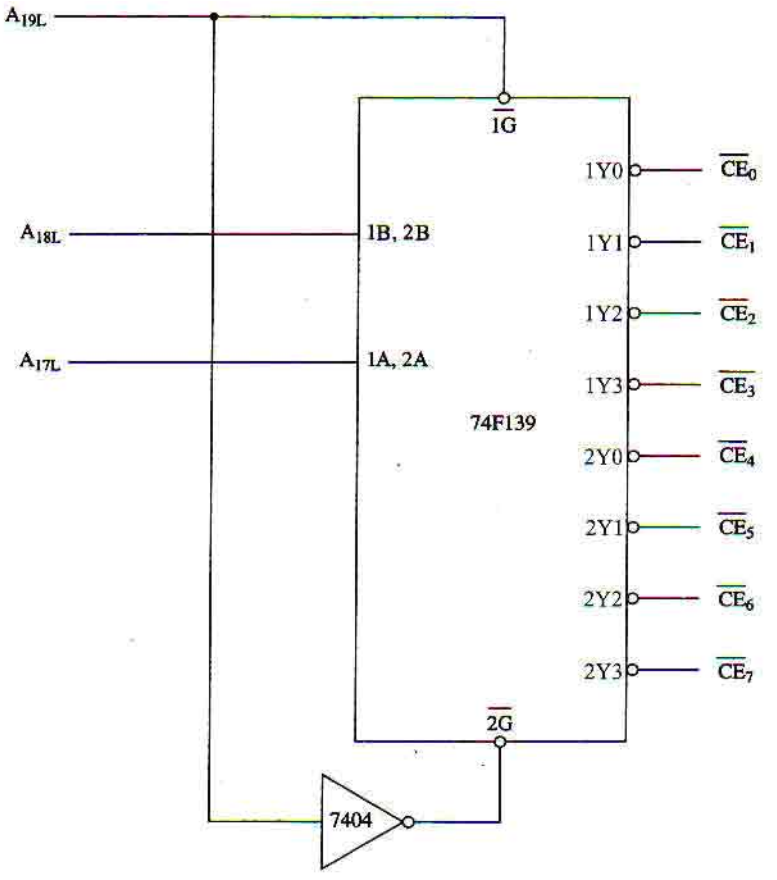
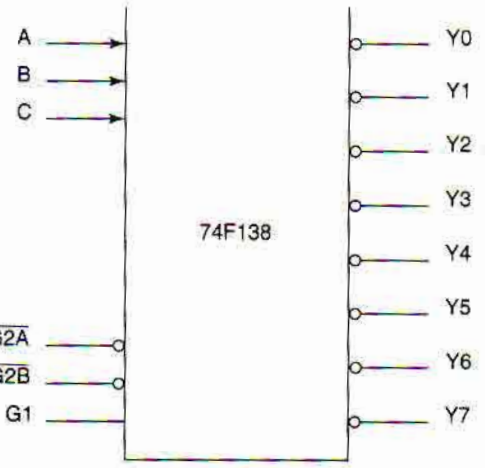


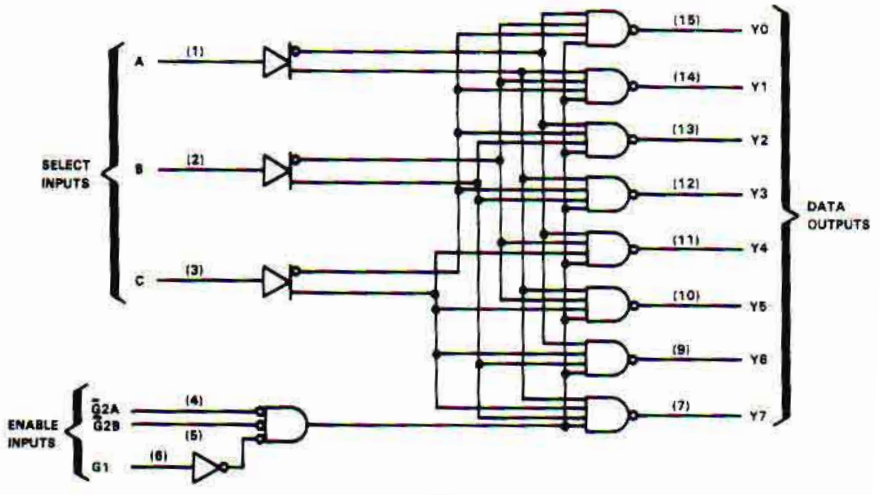
Figure 8–33 Address decoder circuit.

The block diagram of another commonly used decoder, the 74F138, is shown in Fig. 8–34(a). The 74F138 is similar to the 74F139, except that it is a single three-line to eight-line decoder. The circuit used in this device is shown in Fig. 8–34(b). Note that it can be used to produce eight $\overline{CE}$ outputs. The table in Fig. 8–34(c) describes the operation of the 74F138. Here we find that when enabled, only the output that corresponds to the code at the CBA inputs switches to the active 0 logic level.

The circuit in Fig. 8–35 uses the 74F138 to generate chip enable signals $\overline{CE}_0$ through $\overline{CE}_7$ by decoding address lines A_{17L}, A_{18L}, and A_{19L}. Connecting the enable inputs to +5V and ground permanently enables the decoder. The advantage of using the 74F138 over the 74F139 for decoding is that it does not require an extra inverter to generate eight chip enable signals.



(a)



(b)

ENABLE INPUTS			SELECT INPUTS			OUTPUTS							
G1	$\overline{G2A}$	$\overline{G2B}$	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	X	H	H	H	H	H	H	H	H
X	X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	L	H	H	H	H	H	L	H	H	H	H
H	L	L	H	L	L	H	H	H	H	L	H	H	H
H	L	L	H	L	H	H	H	H	H	H	L	H	H
H	L	L	H	H	L	H	H	H	H	H	H	L	H
H	L	L	H	H	H	H	H	H	H	H	H	H	L

(c)

Figure 8–34 (a) Block diagram of 74F138. (b) 74F138 circuit diagram. (Courtesy of Texas Instruments Incorporated) (c) Operation of the 74F138. (Courtesy of Texas Instruments Incorporated)

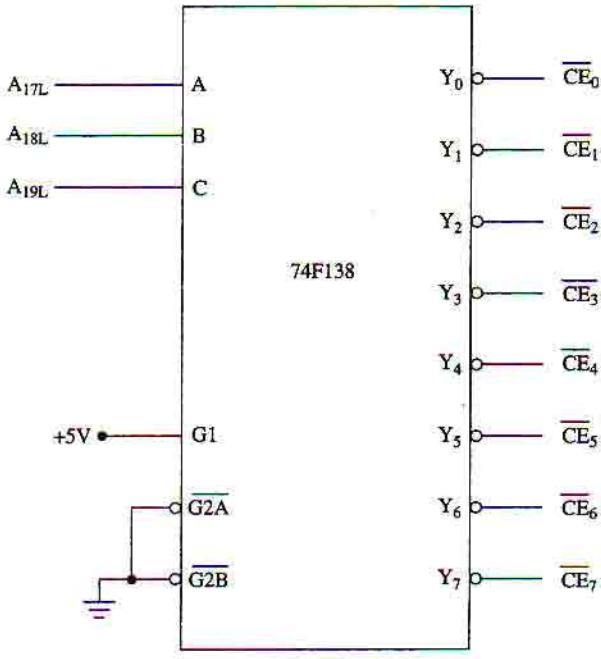


Figure 8–35 Address decoder circuit using 74F138.

▲ 8.13 PROGRAMMABLE LOGIC ARRAYS

In the last section we found that basic logic devices such as latches, transceivers, and decoders are required in the bus interface section of the 8086 microcomputer system. We showed that these functions were performed with standard logic devices such as the 74F373 octal transparent latch, 74F245 octal bus transceiver, and 74F139 two-line to four-line decoder, respectively. Today *programmable logic array* (PLA) devices are becoming very important in the design of microcomputer systems. For example, address and control signal decoding in the memory interface in Fig. 8–24 can be implemented with PLAs, instead of with separate logic ICs. Unlike the earlier mentioned devices, PLAs do not implement a specific logic function. Instead, they are general-purpose logic devices that have the ability to perform a wide variety of specialized logic functions. A PLA contains a general-purpose AND-OR-NOT array of logic gate circuits. The user has the ability to interconnect the inputs to the AND gates of this array. The definition of these inputs determines the logic function that is implemented. The process used to connect or disconnect inputs of the AND gate array is known as *programming*, which leads to the name programmable logic array.

PLAs, GALs, and EPLDs

A variety of different types of PLA devices are available. Early devices were all manufactured with the bipolar semiconductor process. These devices are referred to as *PALs* and remain in use today. Bipolar devices are programmed with an interconnect pattern by burning out fuse links within the device. In the initial state, all of these fuse links are intact. During programming, unwanted links are open-circuited by injecting a current

through the fuse to burn it out. For this reason, once a device is programmed it cannot be reused. If a design modification is required in the pattern, a new device must be programmed and substituted for the original device. Since PALs are made with an older bipolar technology, they are limited to simpler functions and characterized by slower operating speeds and high power consumption.

Newer PLA devices are manufactured with the CMOS process. With this process, very complex, high-speed, low-power devices can be made. Two kinds of CMOS PLAs are in wide use today: the *GAL* and the *EPLD*. These devices differ in the type of CMOS technology used in their design. GALs are designed using *electrically erasable read-only memory* (E^2ROM) technology. The input/output operation of this device is determined by the programming of cells. These electrically programmable cells are also electrically erasable. For this reason, a GAL can be used for one application, erased, and then re-programmed for another application. EPLDs are similar to GALs in that they can be programmed, erased, and reused; however, the erase mechanism is different. They are manufactured with *electrically programmable read only memory* (EPROM) technology. That is, they employ EPROM cells instead of E^2ROM cells. Therefore, to be erased an EPLD must be exposed to ultraviolet light. GALs and EPLDs are currently the most rapidly growing segments of the PLA marketplace.

Block Diagram of a PLA

The block diagram in Fig. 8–36 represents a typical PLA. Looking at this diagram, we see that it has 16 input leads, marked I_0 through I_{15}. There are eight output leads, labeled F_0 through F_7. This PLA is equipped with three-state outputs. For this reason, it has a chip-enable control lead. In the block diagram, this control input is marked $\overline{CE}$. The logic level of $\overline{CE}$ determines if the outputs are enabled or disabled.

When a PLA is used to implement random logic functions, the inputs represent Boolean variables, and the outputs are used to provide eight separate random logic functions. The internal AND-OR-NOT array is programmed to define a sum-of-product equation for each of these outputs in terms of the inputs and their complements. In this way,

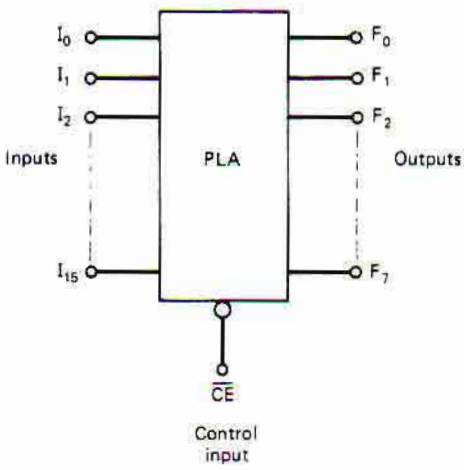


Figure 8–36 Block diagram of a PLA. (Reprinted with the permission of Walter A. Triebel)

we see that the logic levels applied at inputs I_0 through I_{15} and the programming of the AND array determine what logic levels are produced at outputs F_0 through F_7. Therefore, the capacity of a PLA is measured by three properties: the number of inputs, the number of outputs, and the number of product terms (P-terms).

Architecture of a PLA

We just pointed out that the circuitry of a PLA is a general purpose AND-OR-NOT array. Figure 8–37(a) shows this architecture. Here we see that the input buffers supply input signals A and B and their complements $\overline{A}$ and $\overline{B}$. Programmable connections in the AND array permit any combination of these inputs to be combined to form a product term. The product term outputs of the AND array are supplied to fixed inputs of the

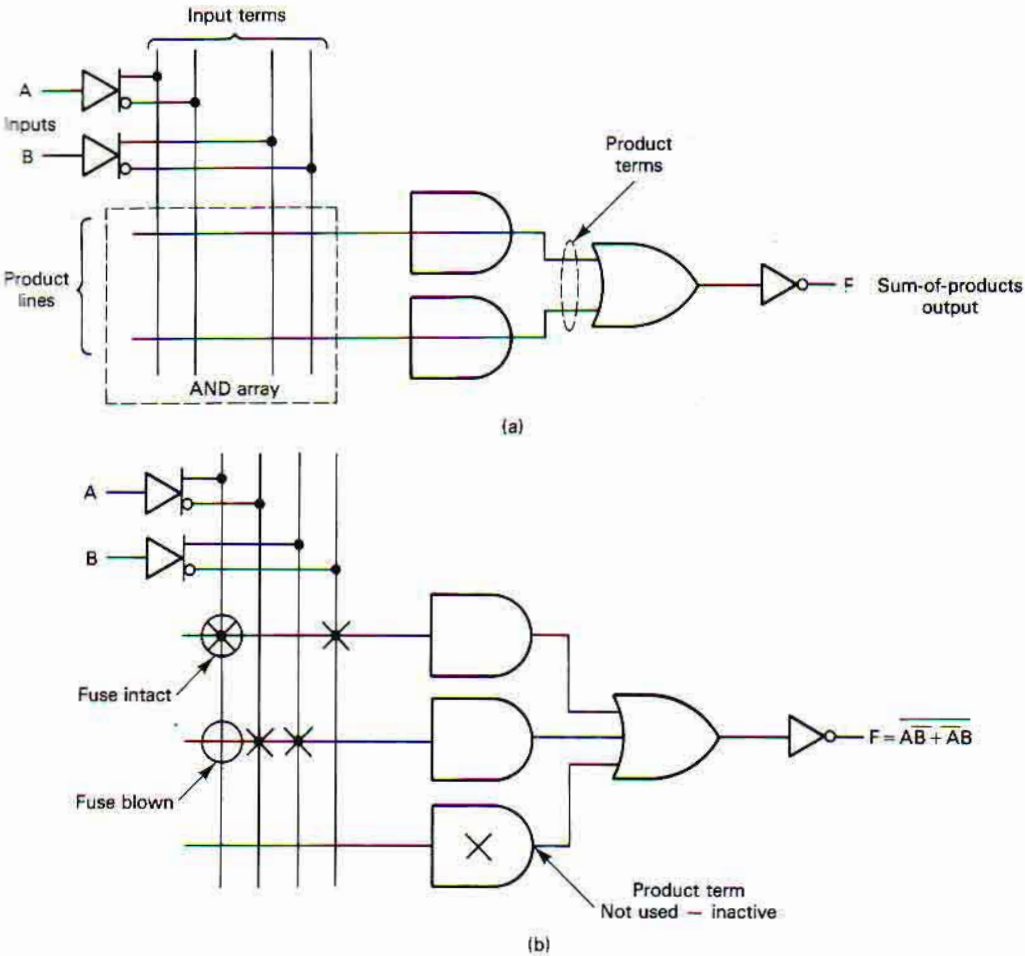


Figure 8–37 (a) Basic PLA architecture. (b) Implementing the logic function $F = \overline{(A\overline{B} + \overline{A}B)}$.

OR array. The output of the OR gate produces a sum-of-products function. Finally, the inverter complements this function.

The circuit in Fig. 8–37(b) shows how the function $F = \overline{(A\overline{B} + \overline{A}B)}$ is implemented with the AND-OR-NOT array. Notice that an X marked into the AND array means that the fuse is left intact, and no marking means that it has been blown to form an open circuit. For this reason, the upper AND gate is connected to A and $\overline{B}$ and produces the product term $A\overline{B}$. The second AND gate from the top connects to $\overline{A}$ and B to produce the product term $\overline{A}B$. The bottom AND gate is marked with an X to indicate that it is not in use. Gates like this that are not to be active should have all of their input fuse links left intact.

Figure 8–38(a) shows the circuit structure that is most widely used in PLAs. It differs from the circuit shown in Fig. 8–37(a) in two ways. First, the inverter has a programmable three-state control and can be used to isolate the logic function from the output. Second, the buffered output is fed back to form another set of inputs to the AND array. This new output configuration permits the output pin to be programmed to work as a *standard output, standard input,* or *logic-controlled input/output.* For instance, if the upper AND gate, which is the control gate for the output buffer, is set up to permanently enable the inverter, and the fuse links for its inputs that are fed back from the outputs are all blown open, the output functions as a standard output.

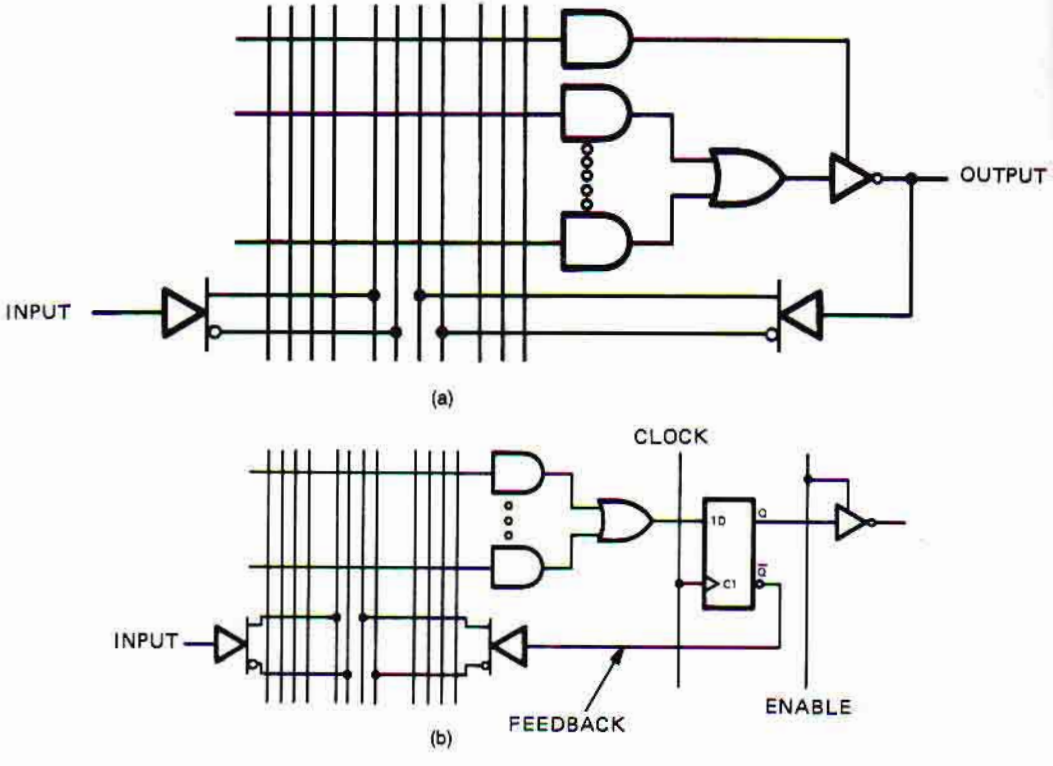


Figure 8–38 (a) Typical PLA architecture. (Courtesy of Texas Instruments Incorporated) (b) PLA with output latch. (Courtesy of Texas Instruments Incorporated)

PLAs are also available in which the outputs are latched with registers. Figure 8–38(b) shows a circuit for this type of device. Here we see that the output of the OR gate is applied to the D input of a clocked D-type flip-flop. In this way, the logic level produced by the AND-OR array is not presented at the output until a pulse is first applied at the CLOCK input. Furthermore, the feedback input is produced from the complemented output of the flip-flop, not the output of the inverter. This configuration is known as a *PLA with registered outputs* and is designed to simplify implementation of *state machine* designs.

Standard PAL™ Devices

Now that we have introduced the types of PLAs, block diagram of the PLA, and internal architecture of the PLA, let us continue by examining a few of the widely used PAL devices. A PAL, or a programmable array logic, is a PLA in which the OR array is fixed; only the AND array is programmable.

The 16L8 is a widely used PAL IC. Its internal circuitry and pin numbering are shown in Fig. 8–39(a). This device is housed in a 20-pin package, as shown in Fig. 8–39(b). Looking at this diagram, we see that it employs the PLA architecture illustrated in Fig. 8–38(a). Note that it has 10 dedicated input pins. All of these pins are labeled I. There are also two dedicated outputs, which are labeled with the letter O, and six programmable I/O lines, which are labeled I/O. Using the programmable I/O lines, the number of input lines can be expanded to as many as 16 inputs or the number of outputs can be increased to as many as eight lines.

All the 16L8's inputs are buffered and produce both the original form of the signal and its complement. The outputs of the buffer are applied to the inputs of the AND array. This array is capable of producing 64 product terms. Note that the AND gates are arranged into eight groups of eight. The outputs of seven gates in each of these groups are used as inputs to an OR gate, and the eighth output is used to produce an enable signal for the corresponding three-state output buffer. In this way, we see that the 16L8 is capable of producing up to seven product terms for each output, and the product terms can be formed using any combination of the 16 inputs.

The 16L8 is manufactured with bipolar technology. It operates from a $+5V \pm 10\%$ dc power supply and draws a maximum of 180 mA. Moreover, all its inputs and outputs are at TTL-compatible voltage levels. This device exhibits high-speed input-output propagation delays. In fact, the maximum I-to-O propagation delay is rated as 7 ns.

Another widely used PAL is the 20L8 device. Looking at the circuitry of this device in Fig. 8–40(a), we see that it is similar to that of the 16L8 just described. However, the 20L8 has a maximum of 20 inputs, eight outputs, and 64 P-terms. The device's 24-pin package is shown in Fig. 8–40(b).

The 16R8 is also a popular 20-pin PLA. The circuit diagram and pin layout for this device are shown in Figs. 8–41(a) and (b), respectively. From Fig. 8–41(a), we find that its eight fixed I inputs and AND-OR array are essentially the same as those of the 16L8. There is one change. The outputs of eight AND gates, instead of seven, are supplied to the inputs of each OR gate.

A number of changes have been made at the output side of the 16R8. Note that the outputs of the OR gates are first latched in D-type flip-flops with the CLK signal. They are then buffered and supplied to the eight Q outputs. Another change is that the enable

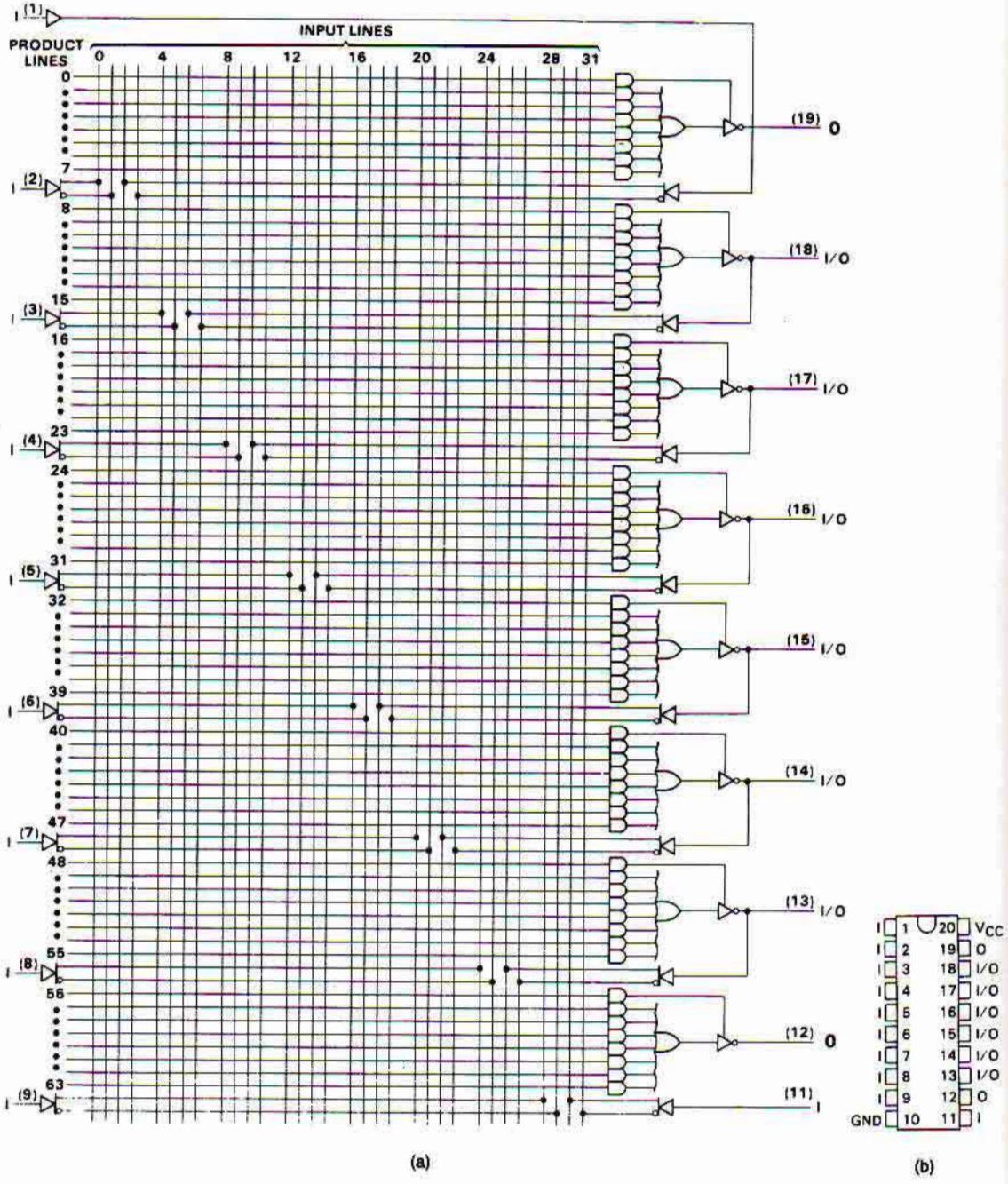


Figure 8–39 (a) 16L8 circuit diagram. (Courtesy of Texas Instruments Incorporated) (b) 16L8 pin layout. (Courtesy of Texas Instruments Incorporated)

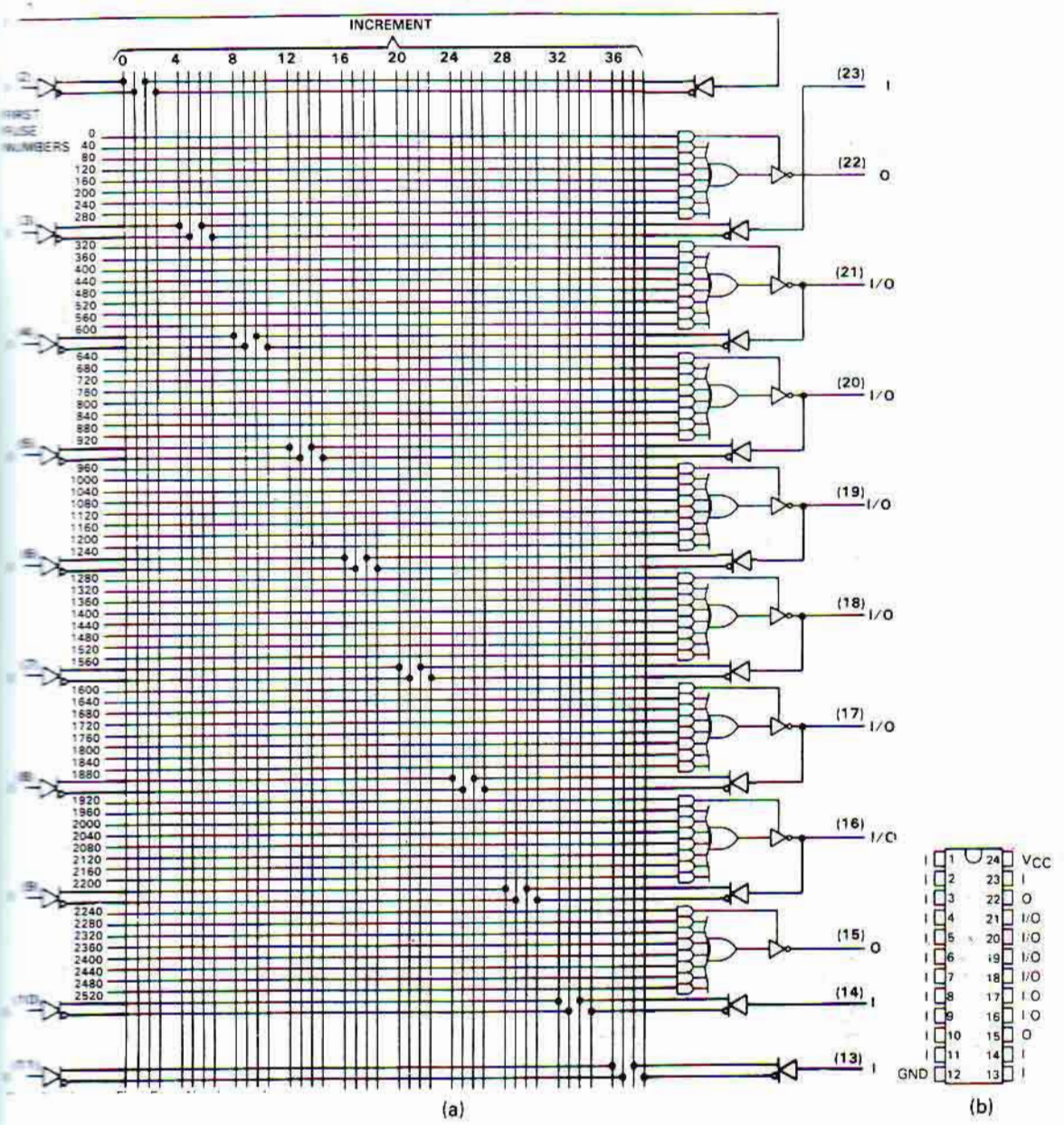


Figure 8–40 (a) 20L8 circuit diagram. (Courtesy of Texas Instruments Incorporated) (b) 20L8 pin layout. (Courtesy of Texas Instruments Incorporated)

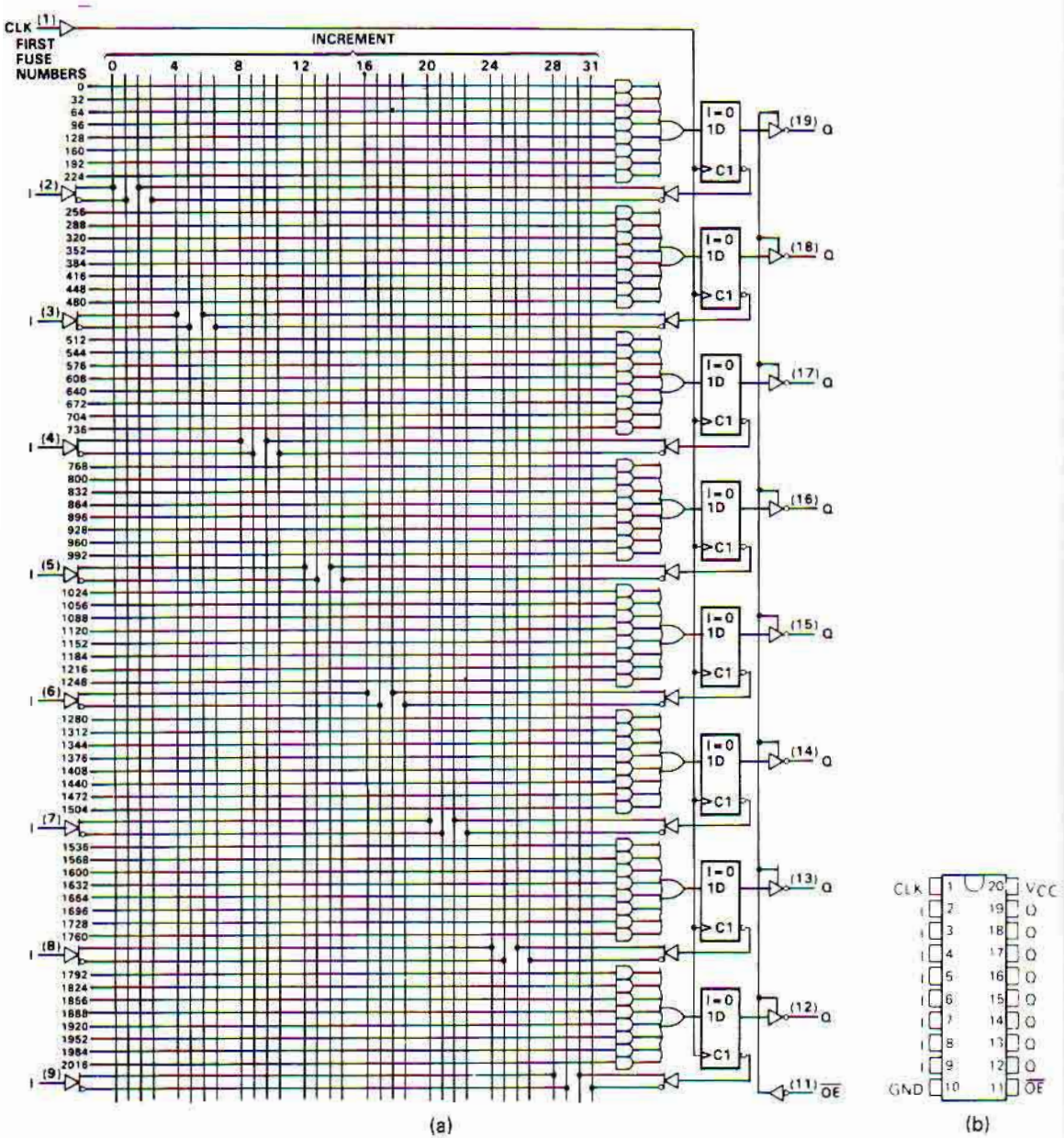


Figure 8–41 (a) 16R8 circuit diagram. (Courtesy of Texas Instruments Incorporated) (b) 16R8 pin layout. (Courtesy of Texas Instruments Incorporated)

signals for the output inverters are no longer programmable. Now the logic level of the $\overline{OE}$ control input enables all three-state outputs.

The last change is in the part of the circuit that produces the feedback inputs. In the 16R8, these eight input signals are derived from the complementary output of the corresponding latch instead of the output of the buffer. For this reason, the output leads can no longer be programmed to work as direct inputs.

The 20R8 is the register output version of the 20L8 PAL. Its circuit diagram and pin layout are given in Figs. 8–42(a) and (b), respectively.

Expanding PLA Capacity

Some applications have requirements that exceed the capacity of a single PLA IC. For instance, a 16L8 device has the ability to supply a maximum of 16 inputs, 8 outputs, and 64 product terms. Connecting several devices together can expand capacity. Let us now look at the way in which PLAs are interconnected to expand the number of inputs and outputs.

If a single PLA does not have enough outputs, two or more devices can be connected together into the configuration of Fig. 8–43(a). Here we see that the inputs I_0 through I_{15} on the two devices are individually connected in parallel. This connection does not change the number of inputs.

On the other hand, the eight outputs of the two PLAs are separately used to form the upper and lower bytes of a 16-bit output word. The bits of this word are denoted as O_0 through O_{15}. So with this connection, we have doubled the number of outputs.

When data are applied to the inputs, PLA 1 outputs the eight least significant bits of data. At the same instant PLA 2 outputs the eight most significant bits. These outputs can be used to represent individual logic functions.

Another limitation on the application of PLAs is the number of inputs. The maximum number of inputs on a single 16L8 is 16. However, additional ICs can be connected to expand the capacity of inputs. Figure 8–43(b) shows how one additional input is added. This permits a 17-bit input denoted as I_0 through I_{16}. The new bit I_{16} is supplied through inverters to the $\overline{CE}$ inputs on the two PLAs. At the output side of the PLAs, outputs O_0 through O_7 of the two devices are individually connected in parallel. To implement this connection, PLA devices with open-collector or three-state outputs must be used.

When I_{16} is logic 0, $\overline{CE}$ on PLA 1 is logic 0. This enables the device for operation, and the output functions coded for input I_0 through I_{15} are output at O_0 through O_7. At the same instant, $\overline{CE}$ on PLA 2 is logic 1 and it remains disabled. Making the logic level of I_{16} equal to 1 disables PLA 1 and enables PLA 2. Now the input at I_0 through I_{15} causes the output function defined by PLA 2 to be output at O_0 through O_7. Actually, this connection doubles the number of product terms as well as increases the number of inputs.

▲ 8.14 TYPES OF INPUT/OUTPUT

The input/output system of the microprocessor allows peripherals to provide data or receive results of processing the data. This is done using I/O ports. The 8088 and 8086 microcomputers can employ two different types of input/output (I/O): *isolated I/O* and *memory-mapped I/O*. These I/O methods differ in how I/O ports are mapped into the 8088/8086's address spaces. Some microcomputer systems employ both kinds of I/O—

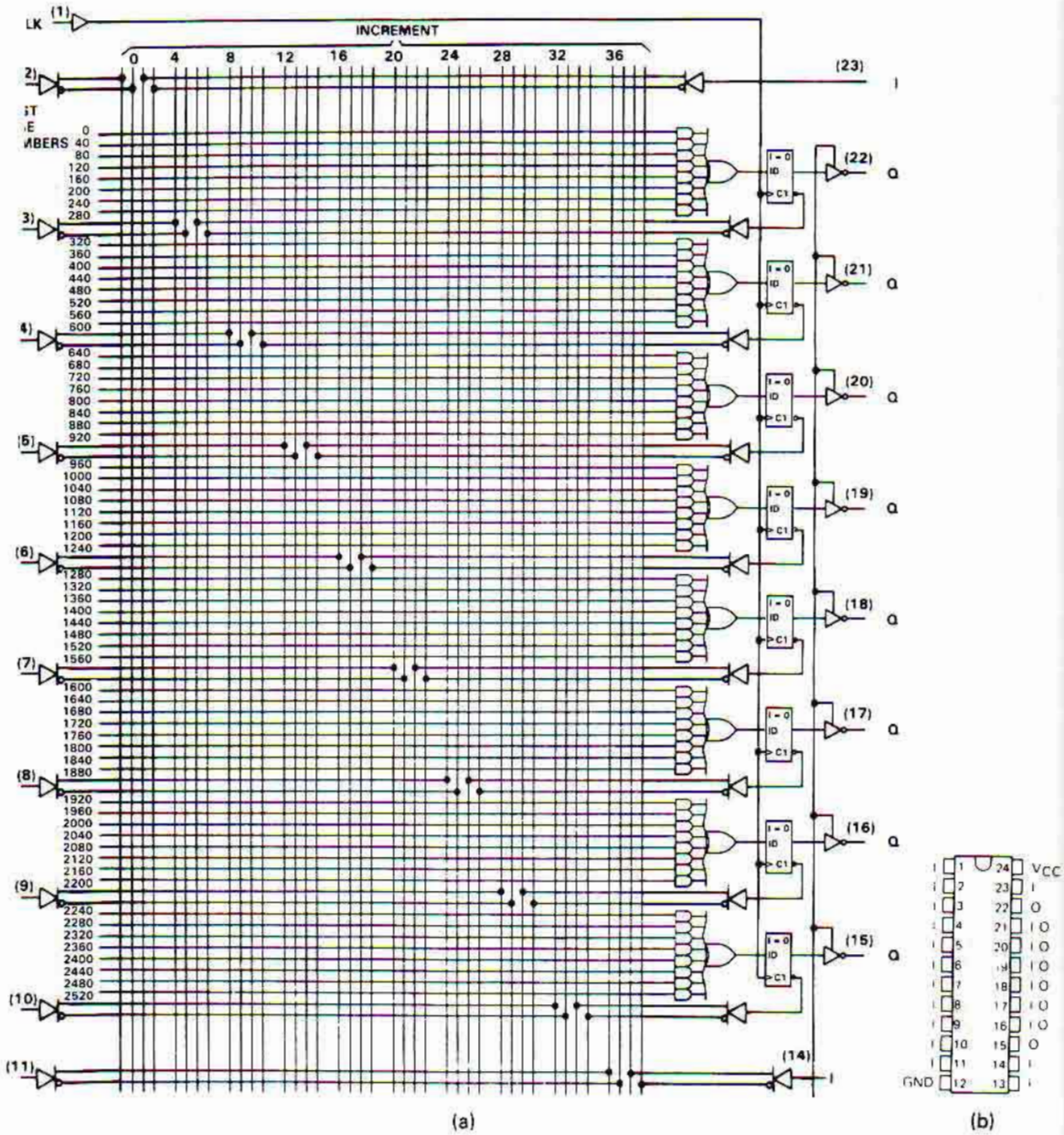


Figure 8–42 (a) 20R8 circuit diagram. (Courtesy of Texas Instruments Incorporated) (b) 20R8 pin layout. (Courtesy of Texas Instruments Incorporated)

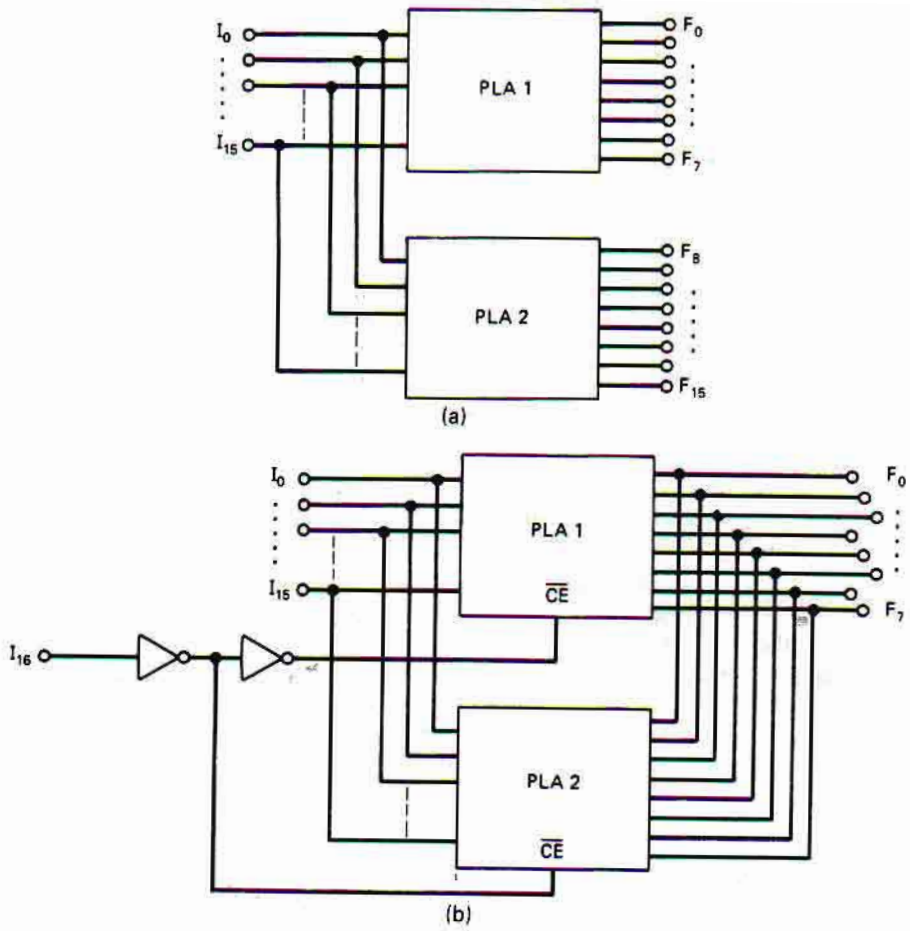


Figure 8–43 (a) Expanding output word length. (Reprinted with the permission of Walter A. Triebel) (b) Expanding input word length. (Reprinted with the permission of Walter A. Triebel)

that is, some peripheral ICs are treated as isolated I/O devices and others as memory-mapped I/O devices. Let us now look at each of these types of I/O.

Isolated Input/Output

When using isolated I/O in a microcomputer system, the I/O devices are treated separate from memory. This is achieved because the software and hardware architectures of the 8088/8086 support separate memory and I/O address spaces. Figure 8–44 illustrates these memory and I/O address spaces.

In our study of 8088/8086 software architecture in Chapter 2, we examined these address spaces from a software point of view. We found that information in memory or at

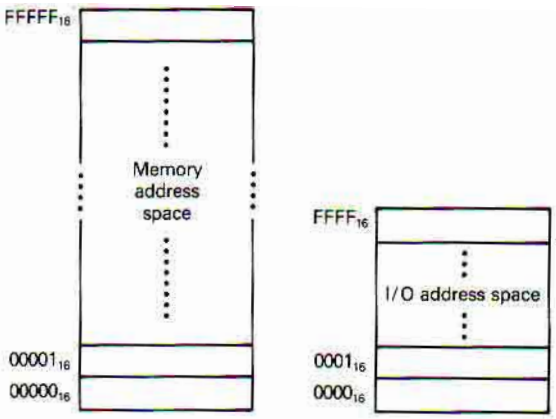


Figure 8–44 8088/8086 memory and I/O address spaces.

I/O ports is organized as bytes of data; that the memory address space contains 1M consecutive byte addresses in the range 00000_{16} through $FFFFF_{16}$; and that the I/O address space contains 64K consecutive byte addresses in the range 0000_{16} through $FFFF_{16}$.

Figure 8–45(a) shows a more detailed map of this I/O address space. Here we find that the bytes of data in two consecutive I/O addresses could be accessed as word-wide data. For instance, I/O addresses 0000_{16}, 0001_{16}, 0002_{16}, and 0003_{16} can be treated as independent byte-wide I/O ports, ports 0, 1, 2, and 3, or ports 0 and 1 may be considered together as word-wide port 0.

Note that the part of the I/O address space in Fig. 8–45(a) from address 0000_{16} through $00FF_{16}$ is referred to as *page 0*. Certain I/O instructions can only perform operations to ports in this part of the address range. Other I/O instructions can input or output data for ports anywhere in the I/O address space.

This isolated method of I/O offers some advantages. First, the complete 1Mbyte memory address space is available for use with memory. Second, special instructions have been provided in the instruction set of the 8088/8086 to perform isolated I/O input and output operations. These instructions have been tailored to maximize I/O performance. A disadvantage of this type of I/O is that all input and output data transfers must take place between the AL or AX register and the I/O port.

Memory-Mapped Input/Output

I/O devices can be placed in the memory address space of the microcomputer as well as in the independent I/O address space. In this case, the MPU looks at the I/O port as though it is a storage location in memory. For this reason, the method is known as *memory-mapped I/O*.

In a microcomputer system with memory-mapped I/O, some of the memory address space is dedicated to I/O ports. For example, in Fig. 8–45(b) the 4096 memory addresses in the range from $E0000_{16}$ through $E0FFF_{16}$ are assigned to I/O devices. Here the contents of address $E0000_{16}$ represent byte-wide I/O port 0, and the contents of addresses $E0000_{16}$ and $E0001_{16}$ correspond to word-wide port 0.

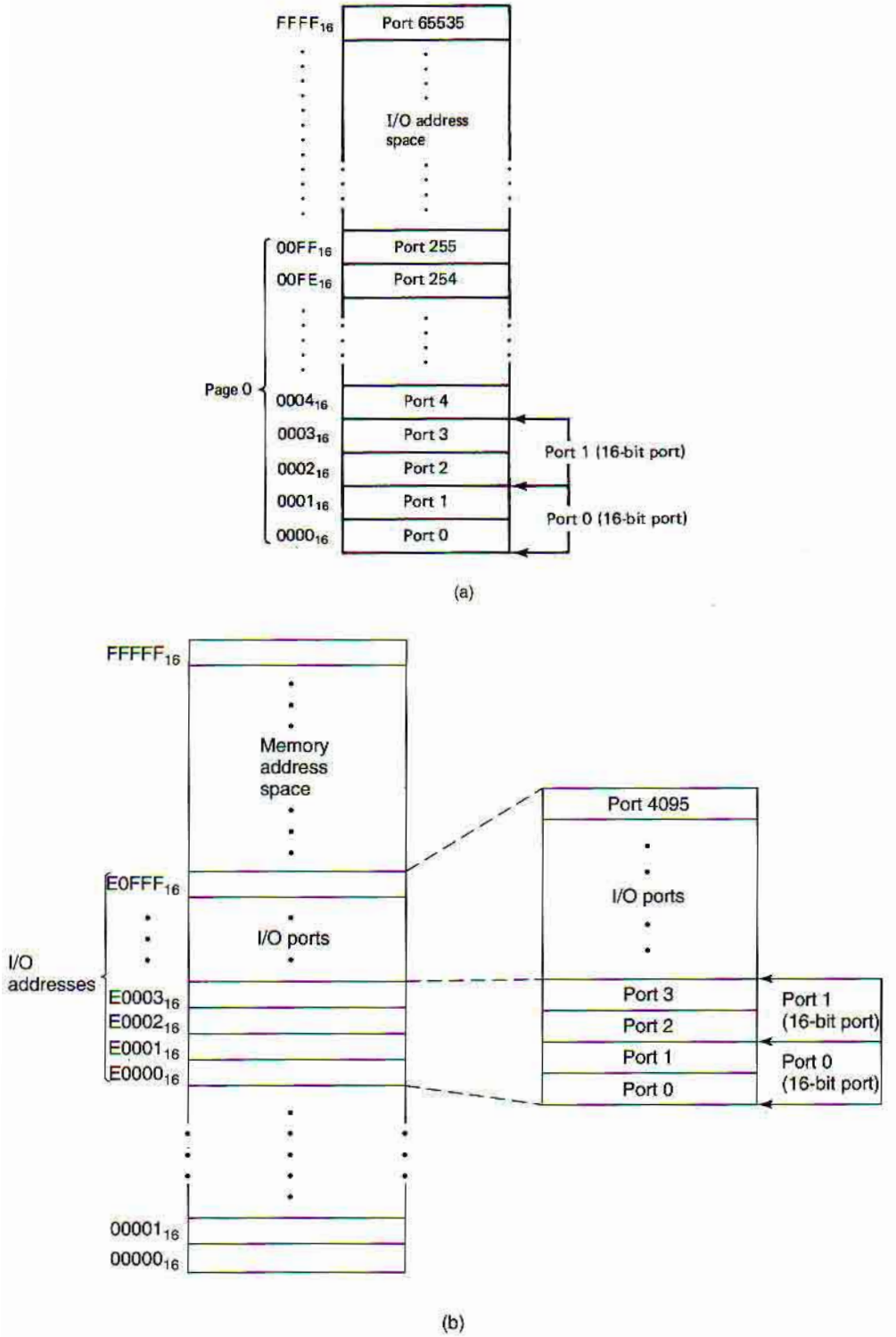


Figure 8–45 (a) Isolated I/O ports. (b) Memory-mapped I/O ports.

When I/O is configured in this way, instructions that affect data in memory are used instead of the special input/output instructions. This is an advantage in that many more instructions and addressing modes are available to perform I/O operations. For instance, the contents of a memory-mapped I/O port can be directly ANDed with a value in an internal register. In addition, I/O transfers can now take place between an I/O port and an internal register other than just AL or AX. However, this also leads to a disadvantage. That is, the memory instructions tend to execute slower than those specifically designed for isolated I/O. Therefore, a memory-mapped I/O routine may take longer to perform than an equivalent program using the input/output instructions.

Another disadvantage of using this method is that part of the memory address space is lost. For instance, in Fig. 8–45(b) addresses in the range from $E0000_{16}$ through $E0FFF_{16}$, allocated to I/O, cannot be used to implement memory.

▲ 8.15 ISOLATED INPUT/OUTPUT INTERFACE

The *isolated input/output interface* of the 8088 and 8086 microcomputers permits them to communicate with the outside world. The way in which the MPU deals with input/output circuitry is similar to the way in which it interfaces with memory circuitry. That is, input/output data transfers also take place over the multiplexed address/data bus. This parallel bus permits easy interface to LSI peripherals such as parallel I/O expanders, interval timers, and serial communication controllers. Through this I/O interface, the MPU can input or output data in bit, byte, or word (for the 8086) formats. Let us continue by looking at how an isolated I/O interface is implemented for minimum- and maximum-mode 8088 and 8086 microcomputer systems.

Minimum-Mode Interface

Let us begin by looking at the isolated I/O interface for a minimum-mode 8088 system. Figure 8–46(a) shows this minimum-mode interface. Here we find the 8088, interface circuitry, and I/O ports for devices 0 through N. I/O devices 0 through N can represent input devices such as a keyboard, output devices such as a printer, or input/output devices such as an asynchronous serial communications port. An example of a typical I/O device used in the I/O subsystem is a programmable peripheral interface (PPI) IC, such as the 82C55A. This type of device is used to implement parallel input and output ports. The circuits in the interface section must perform functions such as select the I/O port, latch output data, sample input data, synchronize data transfers, and translate between TTL voltage levels and those required to operate the I/O devices.

The data path between the 8088 and I/O interface circuits is the multiplexed address/data bus. Unlike the memory interface, this time just the 16 least significant lines of the bus, AD_0 through AD_7 and A_8 through A_{15}, are in use. This interface also involves the control signals that we discussed as part of the memory interface—that is ALE, $\overline{SSO}$, $\overline{RD}$, $\overline{WR}$, IO/$\overline{M}$, DT/$\overline{R}$, and $\overline{DEN}$.

Figure 8–46(b) shows the isolated I/O interface of a minimum-mode 8086-based microcomputer system. Looking at this diagram, we find that the interface differs from that of the 8088 microcomputer in several ways. First, the complete data bus AD_0 through

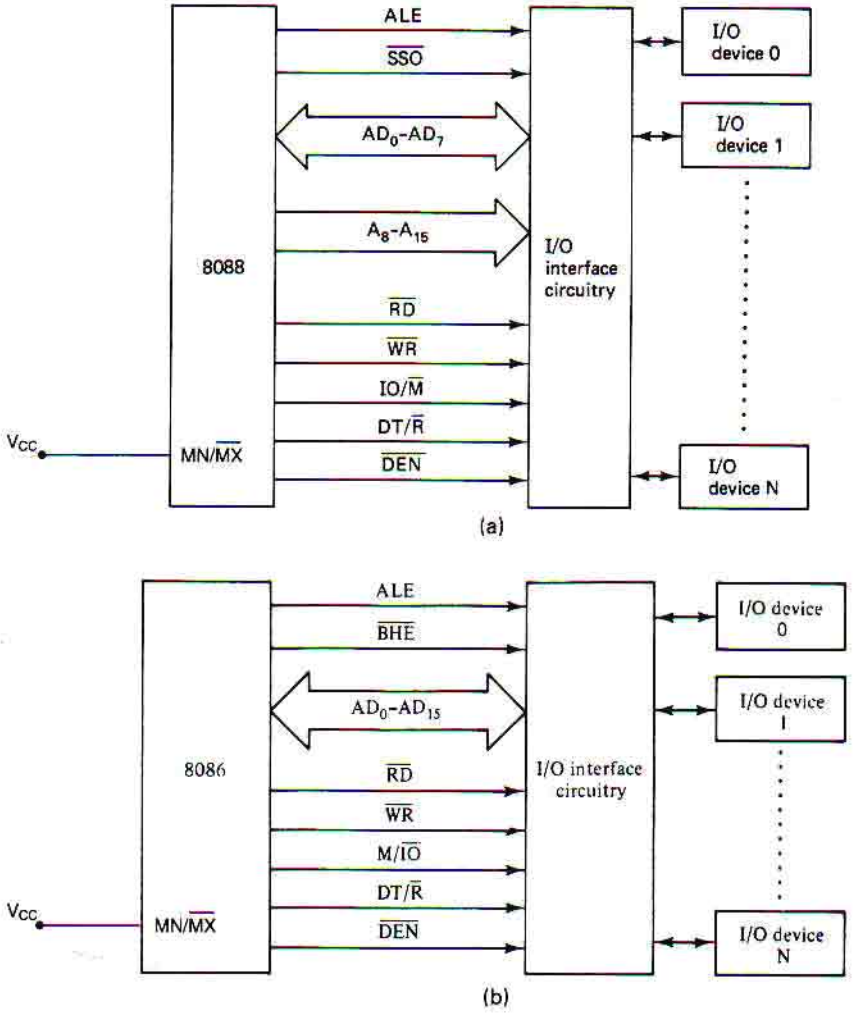


Figure 8–46 (a) Minimum-mode 8088 system I/O interface. (b) Minimum-mode 8086 system I/O interface.

AD_{15} is used for input and output data transfers; second, the $M/\overline{IO}$ control signal is the complement of the equivalent signal $IO/\overline{M}$ in the 8088's interface; and third, status signal $\overline{SSO}$ is replaced by $\overline{BHE}$.

Maximum-Mode Interface

When the 8088 is strapped to operate in the maximum mode ($MN/\overline{MX}$ connected to ground), the interface to the I/O circuitry changes. Figure 8–47(a) illustrates this configuration.

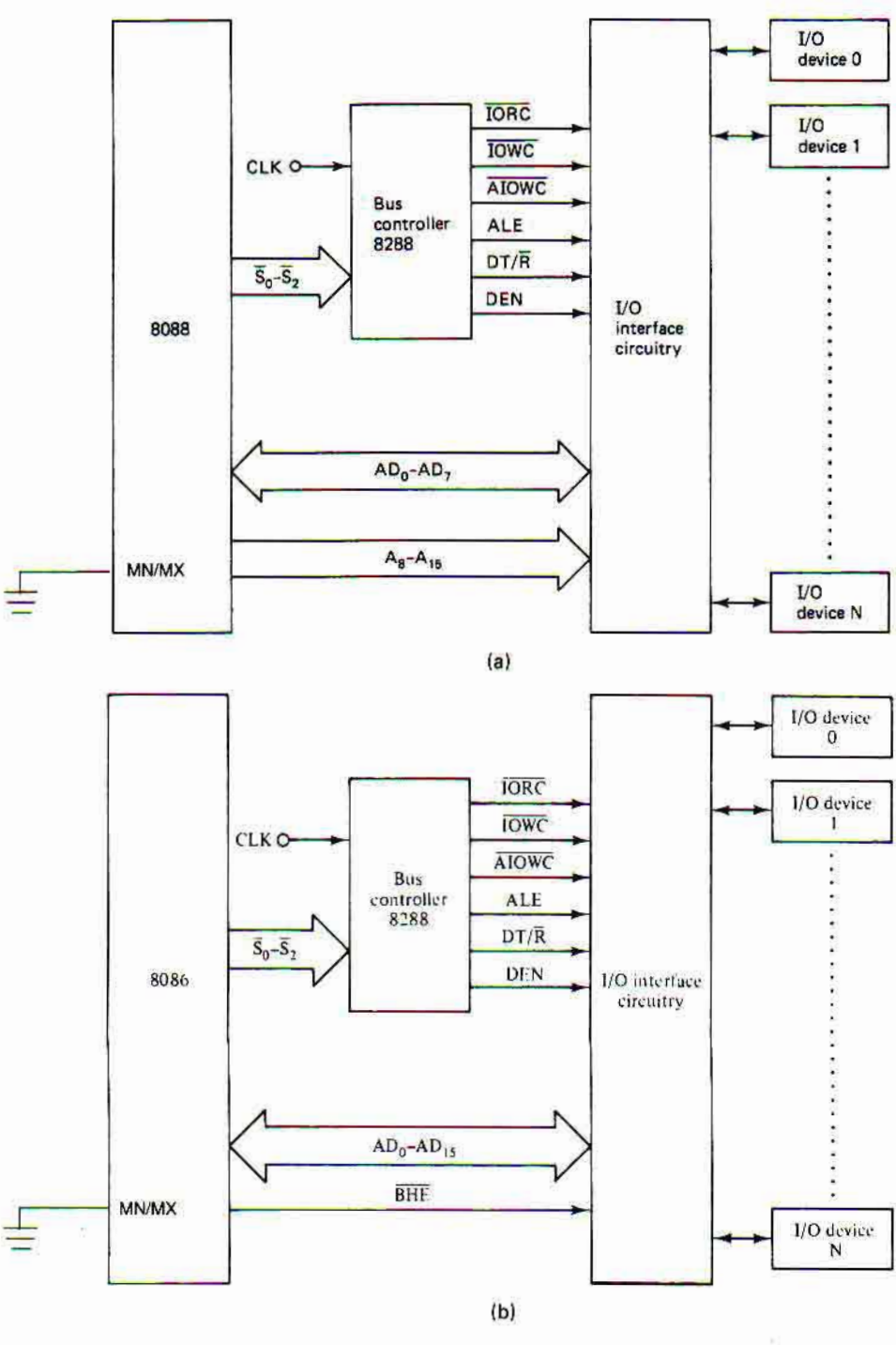


Figure 8–47 (a) Maximum-mode 8088 system I/O interface. (b) Maximum-mode 8086 system I/O interface.

Status inputs			CPU cycle	8288 command
$\overline{S}_2$	$\overline{S}_1$	$\overline{S}_0$		
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

Figure 8–48 I/O bus cycle status codes. (Reprinted with permission of Intel Corporation, Copyright/Intel Corp. 1979)

As in the maximum-mode memory interface, the 8288 bus controller produces the control signals for the I/O subsystem. The 8288 decodes bus command status codes output by the 8088 at $\overline{S}_2\overline{S}_1\overline{S}_0$. These codes tell which type of bus cycle is in progress. If the code corresponds to an I/O read bus cycle, the 8288 generates the *I/O read command output* ($\overline{IORC}$) and for an I/O write cycle it generates *I/O write command outputs* ($\overline{IOWC}$) and ($\overline{AIOWC}$). The 8288 also produces the control signals ALE, DT/$\overline{R}$, and DEN. The address and data transfer path between 8088 and maximum-mode I/O interface remains address/data bus lines AD_0 through AD_7 and A_8 through A_{15}.

Figure 8–47(b) shows the maximum-mode isolated I/O interface of an 8086 microprocessor system. There are only two differences between this interface diagram and that for the 8088 microprocessor. As in the minimum mode, the full 16-bit data bus is the path for data transfers, and the signal $\overline{BHE}$, which is not supplied by the 8088, is included in the interface.

The table in Fig. 8–48 shows the bus command status codes together with the command signals that they produce. Those for I/O bus cycles are highlighted. The MPU indicates that data are to be input (read I/O port) by code $\overline{S}_2\overline{S}_1\overline{S}_0 = 001$. This code causes the bus controller to produce control output I/O read command ($\overline{IORC}$). There is one other code that represents an output bus cycle, the write I/O port code $\overline{S}_2\overline{S}_1\overline{S}_0 = 010$. It produces two output command signals: I/O write cycle ($\overline{IOWC}$) and advanced I/O write cycle ($\overline{AIOWC}$). These command signals are used to enable data from the I/O ports onto the system bus during an input operation and from the MPU to the I/O ports during an output operation.

▲ 8.16 INPUT/OUTPUT DATA TRANSFERS

Input/output data transfers in the 8088 and 8086 microcomputers can be either byte-wide or word-wide. The port that is accessed for input or output of data is selected by an *I/O address*. This address is specified as part of the instruction that performs the I/O operation.

I/O addresses are 16 bits in length and are output by the 8088 to the I/O interface over bus lines AD_0 through AD_7 and A_8 through A_{15}. AD_0 represents the LSB and A_{15} the

MSB. The most significant address lines, A_{16} through A_{19}, are held at the 0 logic level during the address period (T_1) of all I/O bus cycles. Since 16 address lines are used to address I/O ports, the 8088's I/O address space consists of 64K byte-wide I/O ports.

The 8088 signals to external circuitry that the address on the bus is for an I/O port instead of a memory location by switching the IO/$\overline{M}$ control line to the 1 logic level. This signal is held at the 1 level during the complete input or output bus cycle. For this reason, it can be used to enable the address latch or address decoder in external I/O circuitry.

Data transfers between the 8088 and I/O devices are performed over the data bus. Data transfers to byte-wide I/O ports always require one bus cycle. Byte data transfers to a port are performed over bus lines D_0 through D_7. Word transfers also take place over the data bus, D_0 through D_7. However, this type of operation is performed as two consecutive byte-wide data transfers and takes two bus cycles.

For the 8086 microcomputer, I/O addresses are output on address/data bus lines AD_0 through AD_{15}. The logic levels of signals A_0 and $\overline{BHE}$ determine whether data are input/output for an odd-addressed byte-wide port, even-addressed byte-wide port, or a word-wide port. For example, if $A_0\overline{BHE}$ = 10, an odd-addressed byte-wide I/O port is accessed. Byte data transfers to a port at an even address are performed over bus lines D_0 through D_7 and those to an odd-addressed port are performed over D_8 through D_{15}. Data transfers to byte-wide I/O ports always take place in one bus cycle.

Word data transfers between the 8086 and I/O devices are accompanied by the code $A_0\overline{BHE}$ = 00 and are performed over the complete data bus, D_0 through D_{15}. A word transfer can require either one or two bus cycles. To ensure that just one bus cycle is required for the word data transfer, word-wide I/O ports should be aligned at even-address boundaries.

▲ 8.17 INPUT/OUTPUT INSTRUCTIONS

Input/output operations are performed by the 8088 and 8086 microprocessors that employ isolated I/O using special input and output instructions together with the I/O port addressing modes. These instructions, *in* (IN) and *out* (OUT), are listed in Fig. 8–49. Their mnemonics and formats are provided together with a brief description of their operations.

Note that there are two different forms of IN and OUT instructions: the *direct I/O instructions* and *variable I/O instructions*. Either of these two types of instructions can be used to transfer a byte or word of data. All data transfers take place between an I/O device and the MPU's accumulator register. For this reason, this method of performing I/O is known as *accumulator I/O*. Byte transfers involve the AL register, and word transfers the AX regis-

Mnemonic	Meaning	Format	Operation	
IN	Input direct	IN Acc,Port	(Acc) ← (Port)	Acc = AL or AX
	Input indirect (variable)	IN Acc,DX	(Acc) ← ((DX))	
OUT	Output direct	OUT Port,Acc	(Port) ← (Acc)	
	Output indirect (variable)	OUT DX,Acc	((DX)) ← (Acc)	

Figure 8–49 Input/output instructions.

ter. In fact, specifying AL as the source or destination register in an I/O instruction indicates that it corresponds to a byte transfer. That is, byte-wide or word-word input/output is selected by specifying the accumulator (Acc) in the instruction as AL or AX, respectively.

In a direct I/O instruction, the address of the I/O port is specified as part of the instruction. Eight bits are provided for this direct address. For this reason, its value is limited to the address range from 0_{10} equals 00_{16} to 255_{10} equals FF_{16}. This range is referred to as page 0 in the I/O address space.

An example is the instruction

```
IN AL, 0FEH
```

As Fig. 8–49 shows, execution of this instruction causes the contents of the byte-wide I/O port at address FE_{16} of the I/O address space to be input to the AL register. This data transfer takes place in one input bus cycle.

EXAMPLE 8.7

Write a sequence of instructions that will output the data FF_{16} to a byte-wide output port at address AB_{16} of the I/O address space.

Solution

First, the AL register is loaded with FF_{16} as an immediate operand in the instruction

```
MOV  AL, 0FFH
```

Now the data in AL can be output to the byte-wide output port with the instruction

```
OUT  0ABH, AL
```

The difference between the direct and variable I/O instructions lies in the way in which the address of the I/O port is specified. We just saw that for direct I/O instructions an 8-bit address is specified as part of the instruction. On the other hand, the variable I/O instructions use a 16-bit address that resides in the DX register within the MPU. The value in DX is not an offset. It is the actual address that is to be output on AD_0 through AD_7 and A_8 through A_{15} during the I/O bus cycle. Since this address is a full 16 bits in length, variable I/O instructions can access ports located anywhere in the 64K-byte I/O address space.

When using either type of I/O instruction, the data must be loaded into or removed from the AL or AX register before another input or output operation can be performed. In the case of variable I/O instructions, the DX register must be loaded with the address. This requires execution of additional instructions. For instance, the instruction sequence

```
MOV  DX, 0A000H
IN   AL, DX
MOV  BL, AL
```

inputs the contents of the byte-wide input port at $A000_{16}$ of the I/O address space into AL and then saves it in BL.

EXAMPLE 8.8

Write a series of instructions that will output FF_{16} to an output port located at address $B000_{16}$ of the I/O address space.

Solution

The DX register must first be loaded with the address of the output port. This is done with the instruction

```
MOV  DX, 0B000H
```

Next, the data that are to be output must be loaded into AL with the instruction

```
MOV  AL, 0FFH
```

Finally, the data are output with the instruction

```
OUT  DX, AL
```

EXAMPLE 8.9

Data are to be read in from two byte-wide input ports at addresses AA_{16} and $A9_{16}$ and then output as a word to a word-wide output port at address $B000_{16}$. Write a sequence of instructions to perform this input/output operation.

Solution

We can first read in the byte from the port at address AA_{16} into AL and move it to AH. This is done with the instructions

```
IN   AL, 0AAH
MOV  AH, AL
```

Now the other byte, which is at port $A9_{16}$, can be read into AL by the instruction

```
IN  AL, 0A9H
```

The word is now held in AX. To write out the word of data, we load DX with the address $B000_{16}$ and use a variable output instruction. This leads to the following:

```
MOV  DX, 0B000H
OUT  DX, AX
```

▲ 8.18 INPUT/OUTPUT BUS CYCLES

In Section 8.15, we found that the isolated I/O interface signals for the minimum-mode 8088 and 8086 microcomputer systems are essentially the same as those involved in the memory interface. In fact, the function, logic levels, and timing of all signals other

than IO/$\overline{M}$ (M/$\overline{IO}$) are identical to those already described for the memory interface in Section 8.11.

Waveforms for the 8088's *I/O input (I/O read) bus cycle* and *I/O output (I/O write) bus cycle* are shown in Figs. 8–50 and 8–51, respectively. Looking at the input and output bus cycle waveforms, we see that the timing of IO/$\overline{M}$ does not change. The 8088 switches it to logic 1 to indicate that an I/O bus cycle is in progress. It is maintained at the 1 logic level for the duration of the I/O bus cycle. As in memory cycles, the address is output together with ALE during clock period T_1. For the input bus cycle, $\overline{DEN}$ is switched to logic 0 to signal the I/O interface circuitry when to put the data onto the bus and the 8088 reads data off the bus during period T_3.

On the other hand, for the output bus cycle in Fig. 8–51, the 8088 puts write data on the bus late in T_2 and maintains it during the rest of the bus cycle. This time $\overline{WR}$ switches to logic 0 to signal the I/O system that valid data are on the bus.

The waveforms of the 8086's input and output bus cycles are shown in Figs. 8–52 and 8–53, respectively. Let us just look at the differences between the input cycle of the 8086 and that of the 8088. Comparing the waveforms in Fig. 8–52 to those in Fig. 8–50, we see that the 8086 outputs the signal $\overline{BHE}$ along with the address in T-state T_1. Remem-

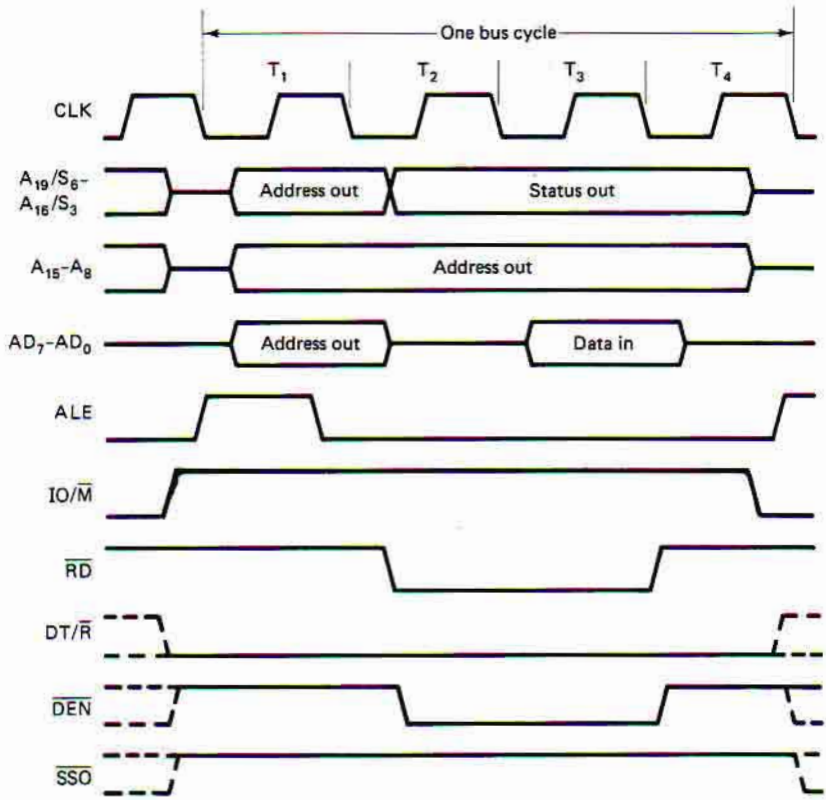


Figure 8–50 Input bus cycle of the 8088.

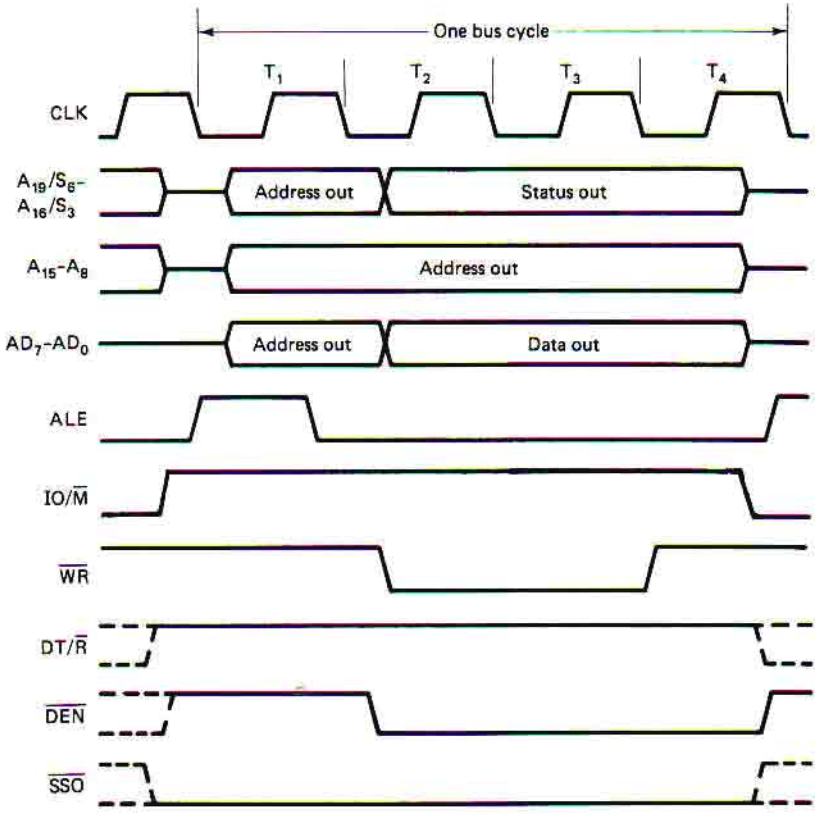


Figure 8–51 Output bus cycle of the 8088.

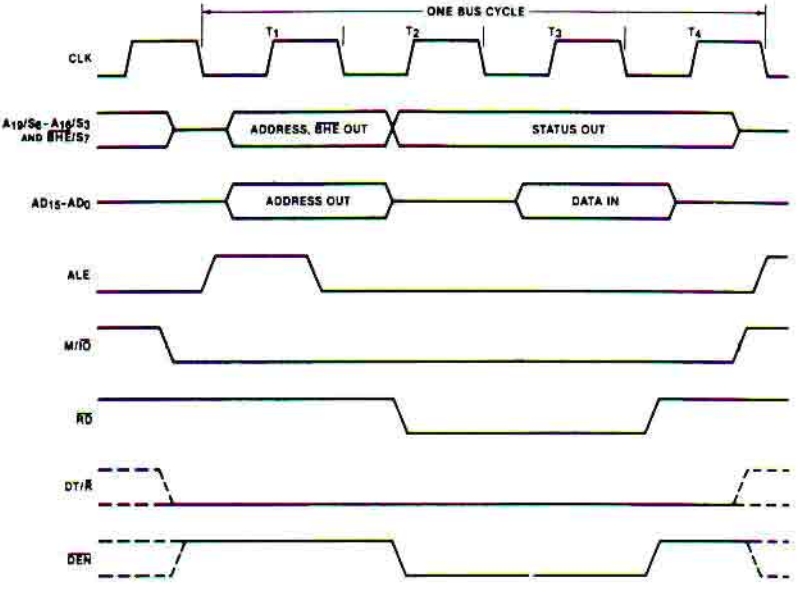


Figure 8–52 Input bus cycle of the 8086.

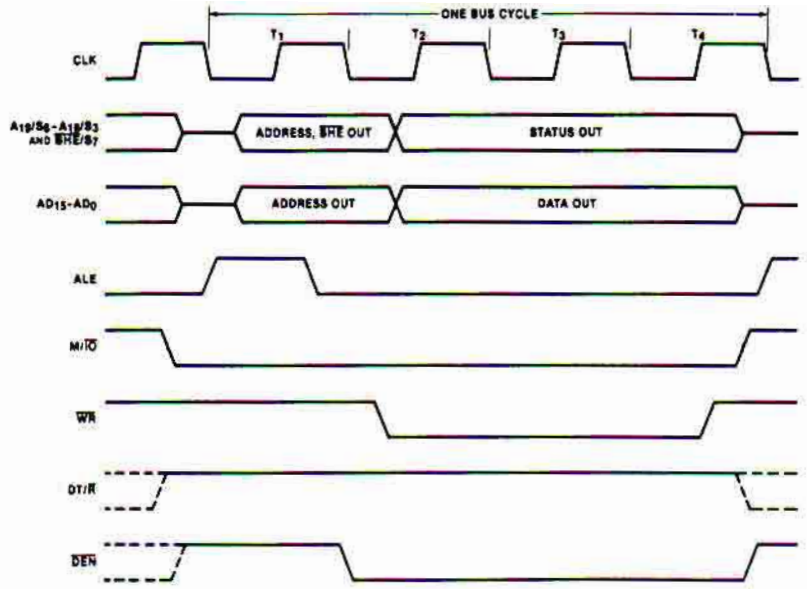


Figure 8–53 Output bus cycle of the 8086.

ber that for the 8086 microprocessor this signal is used along with A_0 to select the byte-wide or the word-wide port. Next, the 8086's data transfer path to the I/O interface is the 16-bit address/data bus, not 8 bits as in the 8088 system. Therefore, data transfers, which take place during T_3, can take place over the lower 8 data bus lines, upper 8 data bus lines, or all 16 data bus lines. Third, the 8086 outputs logic 0 on the M/$\overline{IO}$ line, while the 8088 outputs logic 1 on the IO/$\overline{M}$ line. That is, the M/$\overline{IO}$ control signal of the 8086 is the complement of that of the 8088. Finally, the 8086 does not produce an $\overline{SSO}$ output signal like the one in the 8088.

REVIEW PROBLEMS

Section 8.1

1. Name the technology used to fabricate the 8088 and 8086 microprocessors.
2. What is the transistor count of the 8088?
3. Which pin is used as the NMI input on the 8088?
4. Which pin provides the $\overline{BHE}/S_7$ output signals on the 8086?
5. How much memory can the 8088 and 8086 directly address?
6. How large is the I/O address space of the 8088 and 8086?

Section 8.2

7. How is minimum or maximum mode of operation selected?
8. Describe the difference between the minimum-mode 8088 system and maximum-mode 8088 system.

9. What output function is performed by pin 29 of the 8088 when in the minimum mode? Maximum mode?
10. Is the signal M/$\overline{IO}$ an input or output of the 8086?
11. Name one signal that is supplied by the 8088 but not by the 8086.
12. Are the signals QS_0 and QS_1 produced in the minimum mode or maximum mode?

Section 8.3

13. What are the word lengths of the 8088's address bus and data bus? The 8086's address bus and data bus?
14. Does the 8088 have a multiplexed address/data bus or independent address and data buses?
15. What mnemonic is used to identify the least significant bit of the 8088's address bus? The most significant bit of the 8088's data bus?
16. What does status code $S_4S_3 = 01$ mean in terms of the memory segment being accessed?
17. Which output is used to signal external circuitry that a byte of data is available on the upper half of the 8086's data bus?
18. What does the logic level on M/$\overline{IO}$ signal to external circuitry in an 8086 microcomputer?
19. Which output is used to signal external circuitry in an 8088-based microcomputer that valid data is on the bus during a write cycle?
20. What signal does a minimum-mode 8088 respond with when it acknowledges an active interrupt request?
21. Which signals implement the DMA interface in a minimum-mode 8088 or 8086 microcomputer system?
22. List the signals of the 8088 that are put in the high-Z state in response to a DMA request.

Section 8.4

23. Identify the signal lines of the 8088 that are different for the minimum-mode and maximum-mode interfaces.
24. What status outputs of the 8088 are inputs to the 8288?
25. What maximum-mode control signals are generated by the 8288?
26. What function does the $\overline{LOCK}$ signal serve in a maximum-mode 8088 microcomputer system?
27. What status code is output by the 8088 to the 8288 if a memory read bus cycle is taking place?
28. What command output becomes active if the status inputs of the 8288 are 100_2?
29. If the 8088 executes a jump instruction, what queue status code would be output?
30. What signals are provided for local bus control in a maximum-mode 8088 system?

Section 8.5

31. What is the range of power supply voltage over which the 8088 is guaranteed to work correctly?

32. What is the maximum value of voltage that is considered a valid logic 0 at bit D_0 of the 8088's data bus? Assume that the output is sinking 2 mA.

33. What is the minimum value of voltage that would represent a valid logic 1 at the INTR input of the 8088?

34. At what value current is $V_{OL\,max}$ measured on the 8086?

Section 8.6

35. At what speeds are 8088s generally available?

36. What frequency crystal must be connected between the X_1 and X_2 inputs of the clock generator if an 8088-2 is to run at full speed?

37. What clock outputs are produced by the 8284? What would be their frequencies if a 30-MHz crystal were used?

38. What are the logic levels of the clock waveforms applied to the 8088?

Section 8.7

39. How many clock states are in an 8088 bus cycle that has no wait states? How are these states denoted?

40. What is the duration of the bus cycle for a 5-MHz 8088 that is running at full speed and with no wait states?

41. What is an idle state?

42. What is a wait state?

43. If an 8086 running at 10 MHz performs bus cycles with two wait states, what is the duration of the bus cycle?

Section 8.8

44. How is the memory of an 8088 microcomputer organized from a hardware point of view? An 8086 microcomputer?

45. Give an overview of how a byte of data is read from memory address $B0003_{16}$ of an 8088-based microcomputer, and list the memory control signals along with their active logic levels that occur during the memory read bus cycle.

46. Give an overview of how a word of data is written to memory starting at address $A0000_{16}$ of an 8088-based microcomputer, and list the memory control signals together with their active logic levels that occur during the memory write cycle.

47. In which bank of memory in an 8086-based microcomputer are odd-addressed bytes of data stored? What bank select signal is used to enable this bank of memory?

48. Over which of the 8086's data bus lines are even-addressed bytes of data transferred and which bank select signal is active?

49. List the memory control signals together with their active logic levels that occur when a word of data is written to memory address $A0000_{16}$ in a minimum-mode 8086 microcomputer system.

50. List the memory control signals together with their active logic levels that occur when a byte of data is written to memory address $B0003_{16}$ in a minimum-mode 8086 microcomputer. Over which data lines is the byte of data transferred?

Section 8.9

51. In a maximum-mode 8088 microcomputer, what code is output on S_4S_3 when an instruction-fetch bus cycle is in progress?

52. What is the value of S_4S_3 if the operand of a pop instruction is being read from memory? Assume the microcomputer employs the 8088 in the maximum mode.

Section 8.10

53. Which of the 8088's memory control signals is the complement of the corresponding signal on the 8086?

54. What memory control output of the 8088 is not provided on the 8086? What signal replaces it on the 8086?

55. In a maximum-mode 8088-based microcomputer, what memory bus status code is output when a word of instruction code is fetched from memory? Which memory control output(s) is (are) produced by the 8288?

56. In maximum mode, what memory bus status code is output when a destination operand is written to memory? Which memory control output(s) is (are) produced by the 8288?

57. When the instruction PUSH AX is executed, what address bus status code and memory bus cycle code are output by the 8088 in a maximum-mode microcomputer system? Which command signals are output by the 8288?

Section 8.11

58. How many clock states are in a read bus cycle that has no wait states? What would be the duration of this bus cycle if the 8086 were operating at 10 MHz?

59. What happens in the T_1 part of the 8088's memory read or write bus cycle?

60. Describe the bus activity that takes place as an 8088, in minimum mode, writes a byte of data into memory address $B0010_{16}$.

61. Which two signals can be used to determine that the current bus cycle is a write cycle?

62. Which signal can be used to identify the start of a bus cycle?

Section 8.12

63. Give an overview of the function of each block in the memory interface diagram shown in Fig. 8–24.

64. When the instruction PUSH AX is executed, what bus status code is output by the 8086 in maximum mode, what are the logic levels of A_0 and $\overline{BHE}$, and what read/write control signals are produced by the bus controller?

65. What type of basic logic devices is provided by the 74F373?

66. Specify the logic levels of $\overline{BHEL}$, $\overline{MWRC}$, $\overline{MRDC}$, and A_{0L} when the 8086 in Fig. 8–24 reads a word of data from address 12340H.

67. Make a truth table, using the circuits in Figs. 8–27 and 8–28, to specify the logic levels of $\overline{RD}_U$, $\overline{RD}_L$, $\overline{WR}_U$, $\overline{WR}_L$, $\overline{BHEL}$, $\overline{MRDC}$, $\overline{MWTC}$, and A_{0L} when the processor
    (a) reads a byte from address 01234H
    (b) writes a byte to address 01235H
    (c) reads a word from address 01234H
    (d) writes a word to address 01234H
68. What logic devices are provided by the 74F245?
69. In the circuit of Fig. 8–30, what logic levels must be applied to the $\overline{DEN}$ and DT/$\overline{R}$ inputs to cause data on the system data bus to be transferred to the microprocessor data bus?
70. Make a drawing like that shown in Fig. 8–30 to illustrate the data bus transceiver circuit needed in an 8088-based microcomputer system.
71. How many address lines must be decoded to generate five chip select signals?
72. Name an IC that implements a two-line to four-line decoder logic function.
73. If the inputs to a 74F138 decoder are $G_1 = 1$, $\overline{G}_{2A} = 0$, $G_{2B} = 0$, and CBA = 101 , which output is active?
74. Make a drawing for a minimum-mode 8088-based microcomputer for which a 74F138 decoder is used to generate $\overline{MEMR}$ and $\overline{MEMW}$ from the $\overline{RD}$, $\overline{WR}$, and IO/$\overline{M}$ signals.

Section 8.13

75. What does PLA stand for?
76. List three properties that measure the capacity of a PLA.
77. What is the programming mechanism used in the PAL called?
78. What does PAL stand for? Give the key difference between a PAL and a PLA.
79. Redraw the circuit shown in Fig. 8–37(b) to illustrate how it can implement the logic function $F = (A\,\overline{B} + AB)$.
80. How many dedicated inputs, dedicated outputs, programmable input/outputs, and product terms are supported on the 16L8 PAL?
81. What is the maximum number of inputs on a 20L8 PAL? The maximum number of outputs?
82. How do the outputs of the 16R8 differ from those of the 16L8?
83. Use a 16L8 to decode address lines A_{17L} through A_{19L} to generate $\overline{CE}_0$ through $\overline{CE}_7$.

Section 8.14

84. Name the two types of input/output.
85. What type of I/O is in use when peripheral devices are mapped to the 8088's I/O address space?
86. Which type of I/O has the disadvantage that part of the address space must be given up to implement I/O ports?
87. Which type of I/O has the disadvantage that all I/O data transfers must take place through the AL or AX register?

Section 8.15

88. What are the functions of the 8088's address and data bus lines relative to an isolated I/O operation?

89. In a minimum-mode 8088 microcomputer, which signal indicates to external circuitry that the current bus cycle is for the I/O interface and not the memory interface?

90. List the differences between the 8088's minimum-mode I/O interface in Fig. 8–46(a) and that of the 8086 in Fig. 8–46(b).

91. What is the logic relationship between the signals IO/$\overline{M}$ and M/$\overline{IO}$?

92. In a maximum-mode system, which device produces the input (read), output (write), and bus control signals for the I/O interface?

93. Briefly describe the function of each block in the I/O interface circuit in Fig. 8–47(a).

94. In a maximum-mode 8086 microcomputer, what status code identifies an input bus cycle?

95. In the maximum-mode I/O interface shown in Fig. 8–47(a), what are the logic levels of $\overline{IORC}$, $\overline{IOWC}$, and $\overline{AIOWC}$ during an output bus cycle?

Section 8.16

96. How many bits are in the 8088's I/O address?

97. What is the range of byte addresses in the 8088's I/O address space?

98. What is the size of the 8086's I/O address space in terms of word-wide I/O ports?

99. In an 8086-based microcomputer system, what are the logic levels of A_0 and $\overline{BHE}$ when a byte of data is being written to I/O address $A000_{16}$? If a word of data is being written to address $A000_{16}$?

100. In an 8088 microcomputer system, how many bus cycles are required to output a word of data to I/O address $A000_{16}$? In an 8086 microcomputer system?

Section 8.17

101. Describe the operation performed by the instruction IN AX, 1AH.

102. Write an instruction sequence to perform the same operation as that of the instruction in problem 101, but this time use variable or indirect I/O.

103. Describe the operation performed by the instruction OUT 2AH, AL.

104. Write an instruction sequence that outputs the byte of data $0F_{16}$ to an output port at address 1000_{16}.

105. Write a sequence of instructions that inputs the byte of data from input ports at I/O addresses $A000_{16}$ and $B000_{16}$, adds these values together, and saves the sum in memory location IO_SUM.

106. Write a sequence of instructions that will input the contents of the input port at I/O address $B0_{16}$ and jump to the beginning of a service routine identified by the label ACTIVE_INPUT if the least significant bit of the data is 1.

Section 8.18

107. In the 8088's input bus cycle, during which T state do the IO/$\overline{M}$, ALE, $\overline{RD}$, and $\overline{DEN}$ control signals become active?

108. During which T state in the 8088's input bus cycle is the address output on the bus? Are data read from the bus by the MPU?

109. If an 8088 is running at 5 MHz, what is the duration of the output bus operation performed by executing the instruction OUT 0C0H, AX?

110. If an 8086 running at 10 MHz inserts two wait states into all I/O bus cycles, what is the duration of a bus cycle in which a byte of data is being output?

111. If the 8086 in problem 110 outputs a word of data to a word-wide port at I/O address $1A1_{16}$, what is the duration of the bus cycle?