The 8086 Hardware Specifications

The 8086 was the first 16-bit microprocessor introduced by Intel Corporation in 1978. The 8086 is 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 (DIP), 32-pin common & 8-pin (MIN-MAX) Modes. Figure (1-1) shows the pin layout of 8086 microprocessor.

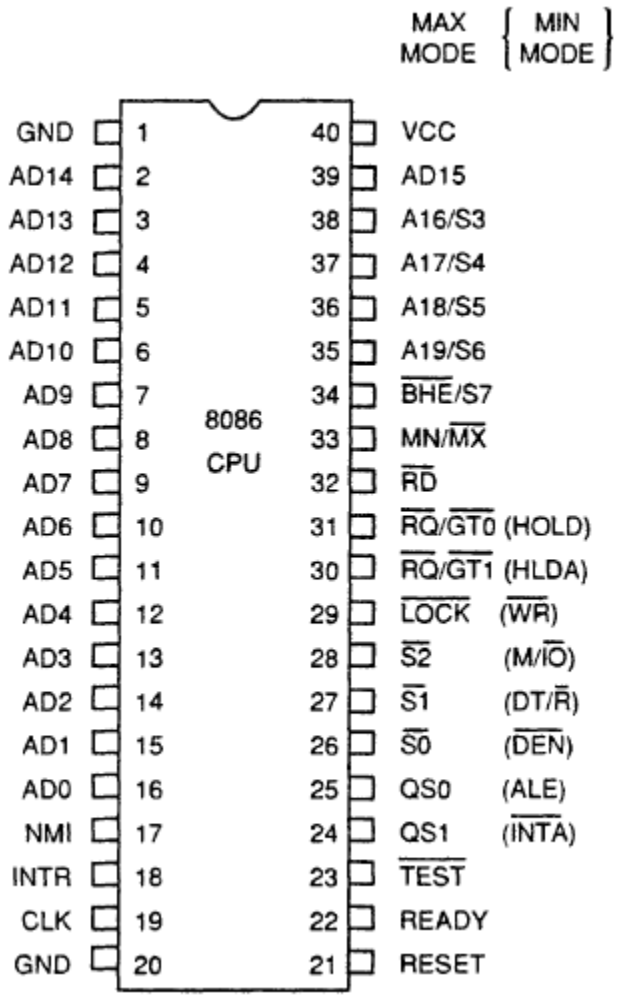


Figure (1-1): The pin layout of 8086 microprocessor.

Minimum mode 8086 system is typically smaller and contains a single processor. All control signals for memory & I/O devices are generated by the microprocessor.

In maximum mode 8086 system, some of the control signals must be externally generated. This requires the addition of an external IC (8288 bus controller). Maximum mode is used only when the system contains external coprocessor such as the 8087 arithmetic coprocessor.

Electrical characteristics of 8086 Microprocessor:

- Power is applied between pin 40 (V_{cc}) and pins 1 (GND) and 20 (GND).
- The nominal value of V_{cc} is specified as +5V dc with a tolerance of ±10%.
- Both 8088 and 8086 draw a maximum of 340mA from the supply.

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}=2.0 mA
V_{OH}	Output high voltage	+2.4 V		I_{OH}=-400 µA

I/O voltage levels

1- Pin Connections (common): The signals pinned out of 8086 are shown in Figure (1-1). Many pins have multiple functions:

1) **VCC (1-pin):** the power supply input provides a +5 V, ±10% signal to the microprocessor.

2) **GND (2pins):** the ground connection is return for power supply. Note that the 8086 microprocessor have two pins labeled GND both must be connected to ground for proper operation.

3) **AD_0-AD_{15} (16-pin)**: the 8086 **Address/Data** bus lines. These lines contain address bits A_0-A_{15} whenever ALE is logic 1, and D_0-D_{15} when ALE is logic 0. These pins are at their high-impedance state during a hold acknowledged.

4) **A_{16}/S_3-A_{19}/S_3 (4-pin)**: the **Address/Status** bits are multiplexed to provide address signals A_{16}-A_{19} and also status bits S_3-S_6. These pins are at their high-impedance state during a hold acknowledged. Status bit S_6 always remains logic 0, bit S_5 indicates the condition of IF flag bit. S_4 and S_3 show which segment is accessed during the current bus cycle as shown in Table (1-1). These two bits can be used to address four separate 1 M byte memory by decoding them as A_{21} and A_{20}.

Table (1-1): Function of status bits $S_3$and S_4.

S_4	S_3	Function
0	0	Extra Segment
0	1	Stack Segment
1	0	Code Segment
1	1	Data Segment

5) **$\overline{BHE}$ / S_7 (1-pin):** the Bus High Enable pin is used in the 8086 to enable the most significant data bus bits (D_8-D_{16}) during a read or a write operation. The state of S_7 is always logic 1.
6) **$\overline{RD}$ (1-pin):** whenever the read signal is logic 0, the data bus is receptive to data from the memory or I/O devices connected to the system. This pin floats to its high-impedance state during a hold acknowledge.
7) **READY (1-pin):** if the READY pin is placed at logic 0 level, the microprocessor enters into wait state and remains idle.
8) **INTR (1-pin):** Interrupt request is used to request hardware interrupts. If INTR is held high when IF=1, the 8086 enters an interrupts acknowledge cycle ($\overline{INTA}$ become active) after the current instruction has complete execution.
9) **NMI (1-pin):** the non-maskable interrupt input is similar to INTR except that the NMI does not check to see whether the IF bit is logic 1. If NMI is activated, this interrupt input uses interrupt vector2.
10) **$\overline{TEST}$ (1-pin):** the test pin is an input that is tested by the WAIT instruction. If $\overline{TEST}$ is logic 0, the WAIT instruction functions as a NOP. If $\overline{TEST}$ is logic 1, the WAIT instruction waits for $\overline{TEST}$ to become logic 0. This pin is most often connected to the 8087 numeric coprocessor.
11) **RESET (1-pin):** the reset input causes the microprocessor to reset itself if this pin is held high for a minimum of four clocking periods. Whenever the 8086 is reset, it begins executing instructions at memory location FFFF0H and disable future interrupts by clearing the IF bit.
12) **CLK (1-pin):** the clock pin provides the basic timing signal to the microprocessor. The clock signal must have a duty cycle of 33% (high for one-third of the clocking period and low for two-thirds) to provide proper internal timing for the 8086.

13) **MN/$\overline{\text{MX}}$ (1-pin)**: the minimum/maximum mode pin selects either minimum mode or maximum mode operation for the microprocessor. If minimum mode is selected, the MN/$\overline{\text{MX}}$ pin must be connected directly to +5V.

Minimum Mode Pins. Minimum mode operation of the 8086/8088 is obtained by connecting the MN/$\overline{\text{MX}}$ pin directly to +5.0 V. Do not connect this pin to +5.0 V through a pull-up resister or it will not function correctly.

IO/$\overline{\text{M}}$ or M/$\overline{\text{IO}}$	The IO/$\overline{\text{M}}$ (8088) or the M/$\overline{\text{IO}}$ (8086) pin selects memory or I/O. This pin indicates that the microprocessor address bus contains either a memory address or an I/O port address. This pin is at its high-impedance state during a hold acknowledge.
$\overline{\text{WR}}$	The **write line** is a strobe that indicates that the 8086/8088 is outputting data to a memory or I/O device. During the time that the $\overline{\text{WR}}$ is a logic 0, the data bus contains valid data for memory or I/O. This pin floats to a high-impedance during a hold acknowledge.
$\overline{\text{INTA}}$	The **interrupt acknowledge** signal is a response to the INTR input pin. The $\overline{\text{INTA}}$ pin is normally used to gate the interrupt vector number onto the data bus in response to an interrupt request.
ALE	**Address latch enable** shows that the 8086/8088address/data bus contains address information. This address can be a memory address or an I/O port number. Note that the ALE signal does not float during a hold acknowledge.
DT/$\overline{\text{R}}$	The **data transmit/receive** signal shows that the microprocessor data bus is transmitting (DT/$\overline{\text{R}}$ = 1) or receiving (DT/$\overline{\text{R}}$ = 0) data. This signal is used to enable external data bus buffers.
DEN	**Data bus enable** activates external data bus buffers.
HOLD	The **hold** input requests a direct memory access (DMA). If the HOLD signal is a logic 1, the microprocessor stops executing software and places its address, data, and control bus at the high-impedance state. If the HOLD pin is a logic 0, the microprocessor executes software normally.
HLDA	**Hold acknowledge** indicates that the 8086/8088 microprocessors have entered the hold state.

Figure (1-2) shows the block diagram of minimum mode 8086 Microprocessor.

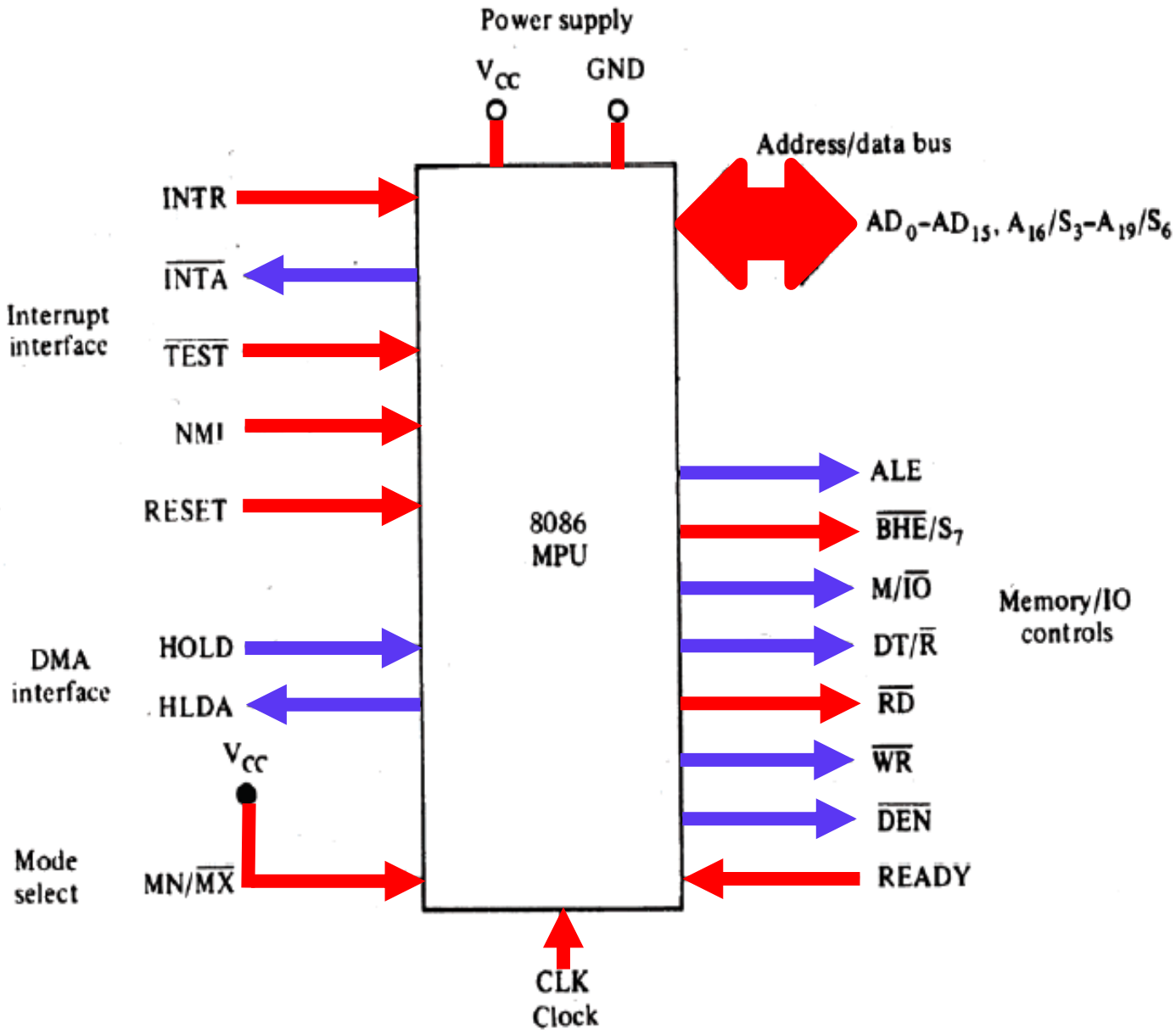


Figure (1-2): The block diagram of minimum mode 8086 MP.

Maximum Mode Pins. In order to achieve maximum mode for use with external coprocessors, connect the MN/$\overline{MX}$ pin to ground.

$\overline{S2}$, $\overline{S1}$, and $\overline{SO}$	The **status bits** indicate the function of the current bus cycle. These signals are normally decoded by the 8288 bus controller described later in this chapter. Table(1-2) shows the function of these three status bits in the maximum mode.
$\overline{RQ}/\overline{GT1}$ and $\overline{RQ}/\overline{GT0}$	The **request/grant** pins request direct memory accesses (DMA) during maximum mode operation. These lines are both bi-directional and are used to request and grant a DMA operation.
$\overline{LOCK}$	The **lock output** is used to lock peripherals off the system. This pin is activated by using the LOCK: prefix on any instruction.
QS_1 and QS_0	The **queue status** bits show the status of the internal instruction queue. These pins are provided for access by the numeric coprocessor (8087). Refer to Table(1-3) for the operation of the queue status bits.

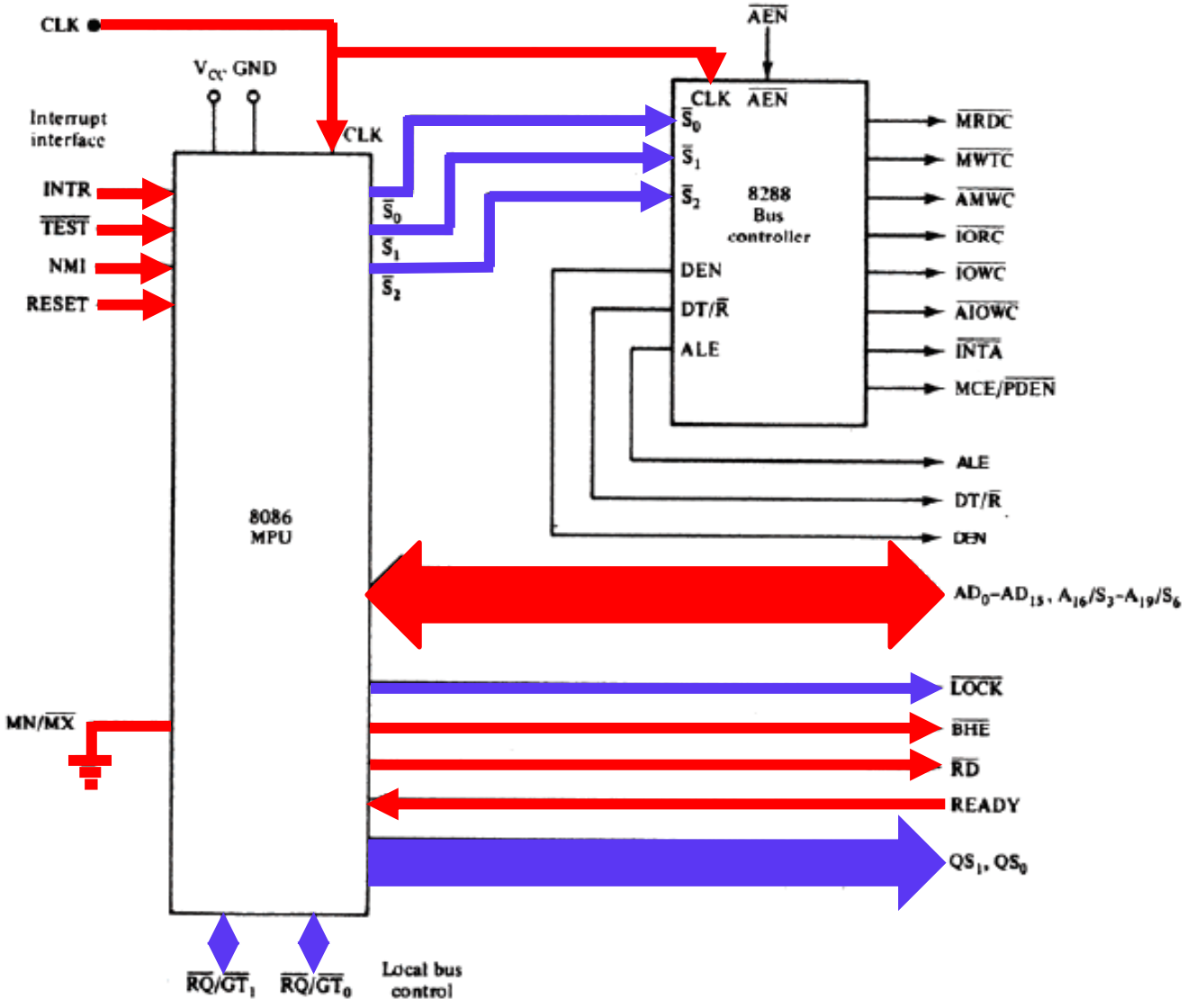
Figure (1-3) shows the block diagram of maximum mode 8086 Microprocessor.

Table (1-2): the Function of status bits ($\bar{S}_2$ $\bar{S}_1$ $\bar{S}_0$) in maximum mode.

Bus cycle status (decoded by 8288 bus controller)		
$\bar{S}_2$ $\bar{S}_1$ $\bar{S}_0$	**CPU cycle**	**8288 command**
0 0 0	**Interrupt acknowledge**	$\overline{INTA}$
0 0 1	**I/O read**	$\overline{IORC}$
0 1 0	**I/O write**	$\overline{IOWC}$, $\overline{AIOWC}$
0 1 1	**Halt**	**None**
1 0 0	**Opcode fetch**	$\overline{MRDC}$
1 0 1	**Memory read**	$\overline{MRDC}$
1 1 0	**Memory write**	$\overline{MWTC}$, $\overline{AMWC}$
1 1 1	**passive**	**None**

Table (1-3): The queue status bits.

QS_1	QS_0	*Function*
0	0	Queue is idle
0	1	First byte of opcode
1	0	Queue is empty
1	1	Subsequent byte of opcode



8086 maximum-mode block diagram

Figure (1-3): The block diagram of maximum mode 8086 MP.

2- CLOCK GENERATOR (8284A)

The 8284A is an ancillary component to the 8086/8088 microprocessors. Without the clock generator, many additional circuits are required to generate the clock (CLK) in an 8086/8088-based system. The 8284A provides the following basic functions or signals: clock generation, RESET synchronization, READY synchronization, and a TTL level peripheral clock signal. Figure (1-4) illustrates the pin-out of the 8284A clock generator.

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

$\overline{\text{AEN1}}$ and $\overline{\text{AEN2}}$ — The **address enable** pins are provided to qualify the bus ready signals, RDY1 and RDY2, respectively. Section 8–5 illustrates the use of these two pins, which are used to cause wait states, along with the RDY1 and RDY2 inputs. Wait states are generated by the READY pin of the 8086/8088 microprocessors, which is controlled by these two inputs.

RDY1 and RDY2 — The **bus ready** inputs are provided in conjunction with the $\overline{\text{AEN1}}$ and $\overline{\text{AEN2}}$ pins to cause wait states in an 8086/8088-based system.

$\overline{\text{ASYNC}}$ — The **ready synchronization** selection input selects either one or two stages of synchronization for the RDY1 and RDY2 inputs.

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

X1 and X2 — The **crystal oscillator** pins connect to an external crystal used as the timing source for the clock generator and all its functions.

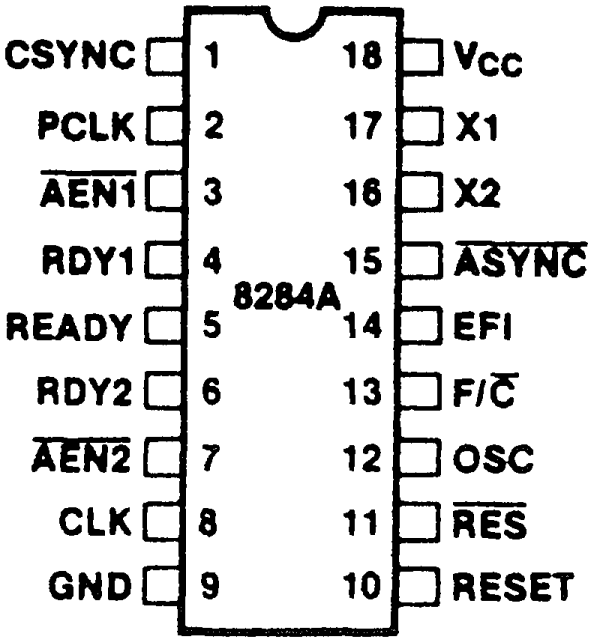


Figure (1-4): The pin layout of 8284A clock generator.

F/$\overline{C}$	The **frequency/crystal select** input chooses the clocking source for the 8284A. If this pin is held high, an external clock is provided to the EFI input pin; if it is held low, the internal crystal oscillator provides the timing signal.
EFI	The **external frequency** input is used when the F/C pin is pulled high. EFI supplies the timing whenever the F/$\overline{C}$ pin is high.
CLK	The **clock** output pin provides the CLK input signal to the 8086/8088 microprocessors and other components in the system. The CLK pin has an output signal that is one-third of the crystal or EFI input frequency and has a 33 percent duty cycle, which is required by the 8086/8088.
PCLK	The **peripheral clock** signal is one-sixth the crystal or EFI input frequency and has a 50 percent duty cycle. The PCLK output provides a clock signal to the peripheral equipment in the system.
OSC	The **oscillator** output is a TTL level signal that is at the same frequency as the crystal or EFI input. The OSC output provides an EFI input to other 8284A clock generators in some multiple-processor systems.
$\overline{RES}$	The **reset** input is an active-low input to the 8284A. The $\overline{RES}$ pin is often connected to an RC network that provides power-on resetting.
RESET	The **reset** output is connected to the 8086/8088 RESET input pin.
CSYNC	The **clock synchronization** pin is used whenever the EFI input provides synchronization in systems with multiple processors. If the internal crystal oscillator is used, this pin must be grounded.
GND	The **ground** pin connects to ground.
V_{CC}	This **power supply** pin connects to +5.0 V with a tolerance of ±10 percent.

Operation of the 8284A

The 8284A is a relatively easy component to understand. Figure (1-5) illustrates the internal logic diagram of the 8284A clock generator.

Operation of the Clock Section. The top half of the logic diagram represents the clock and reset synchronization section of the 8284A clock generator. As the diagram shows, the crystal oscillator has two inputs: X1 and X2. If a crystal is attached to X1 and X2, the oscillator generates a square-wave signal at the same frequency as the crystal. The square-wave signal is fed to an AND gate and also to an inverting buffer that provides the OSC output signal. The OSC signal is sometimes used as an EFI input to other 8284A circuits in a system.

An inspection of the AND gate reveals that when F/$\overline{C}$ is a logic 0, the oscillator output is steered through to the divide-by-3 counter. If F/$\overline{C}$ is a logic 1, then EFI is steered through to the counter.

The output of the divide-by-3 counter generates the timing for ready synchronization, a signal for another counter (divide-by-2), and the CLK signal to the 8086/8088 microprocessors. The CLK signal is also buffered before it leaves the clock generator. Notice that the output of the first counter feeds the second. These two cascaded counters provide the divide-by-6 output at PCLK, the peripheral clock output.

Figure (1-6) shows how an 8284A is connected to the 8086/8088. Notice (1) that F/$\overline{C}$ and CSYNC are grounded to select the crystal oscillator and (2) that a 15 MHz crystal provides the normal 5 MHz clock signal to the 8086/8088 as well as a 2.5 MHz peripheral clock signal.

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

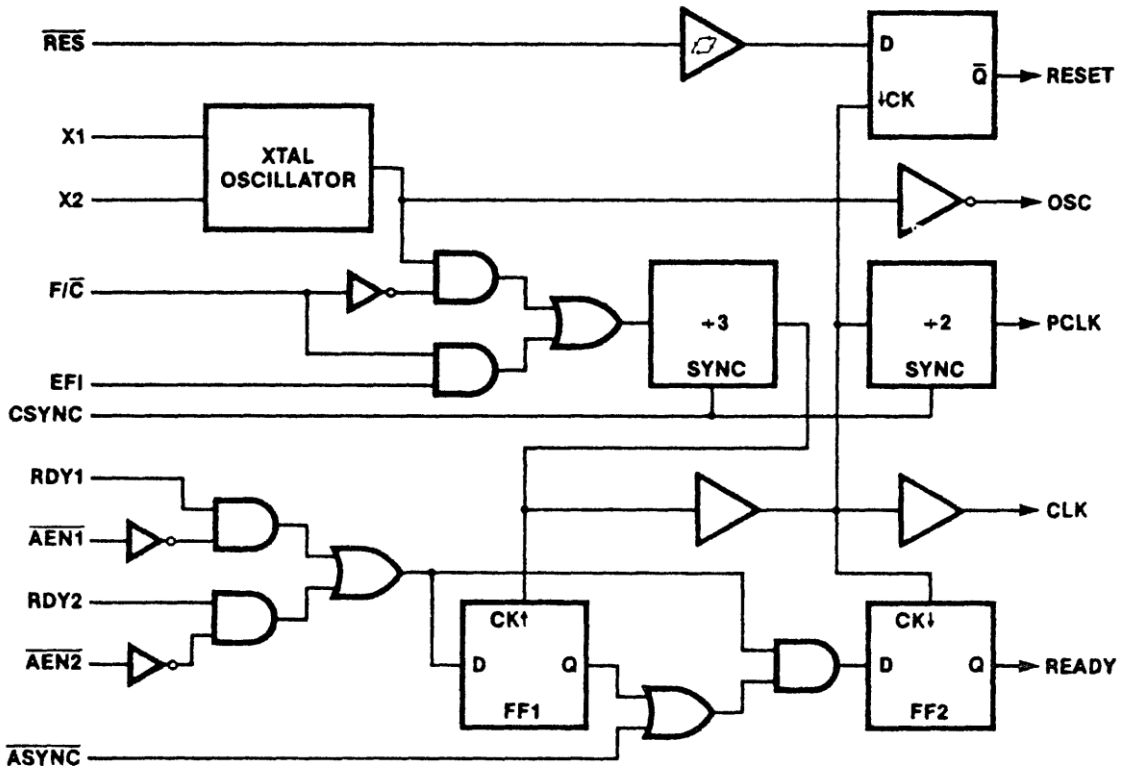


Figure (1-5): The internal logic diagram of 8284A clock generator.

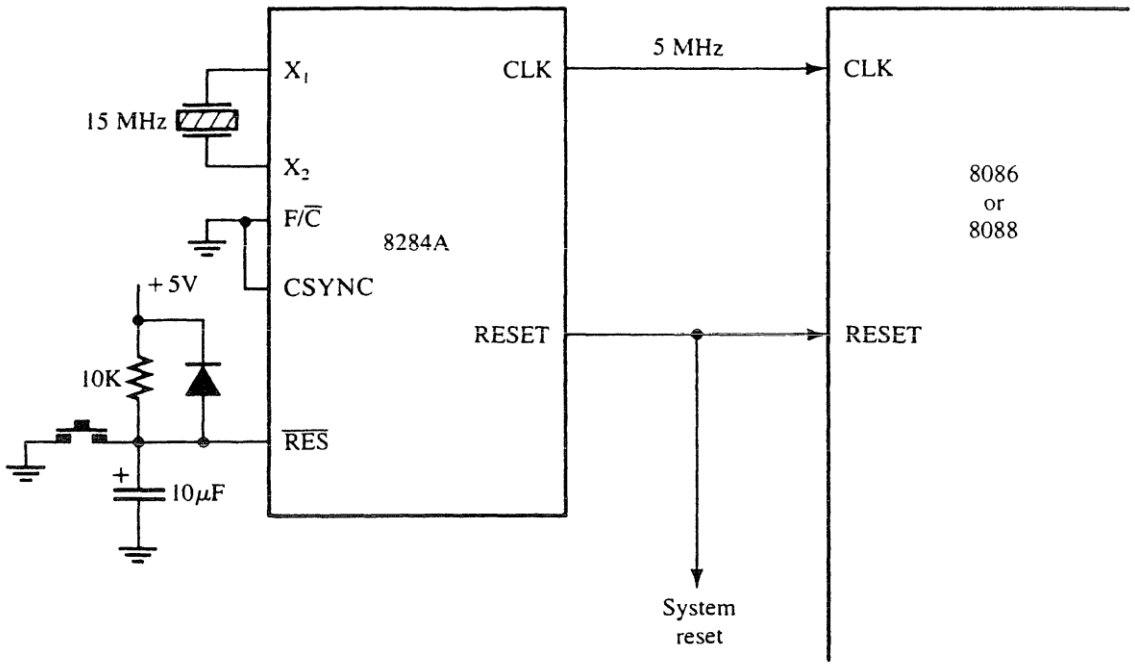


Figure (1-6): The connection of 8284A clock generator with 8086 MP.

Refer again to Figure (1-6). Notice that an RC circuit provides a logic 0 to the $\overline{RES}$ input pin when power is first applied to the system. After a short time, the $\overline{RES}$ input becomes a logic 1 because the capacitor charges toward +5.0 V through the resistor. A push-button switch allows the microprocessor to be reset by the operator. Correct reset timing requires the RESET input to become a logic 1 no later than four clocks after system power is applied and to be held high for at least 50 μs. The flip-flop makes certain that RESET goes high in four clocks, and the RC time constant ensures that it stays high for at least 50 μs.

The READY Input

The READY input is sampled at the end of T2 and again, if applicable, in the middle of Tw. If READY is a logic 0 at the end of T2, then T3 is delayed and Tw is inserted between T2 and T3. READY is next sampled at the middle of Tw to determine if the next state is Tw or T3. It is tested for a logic 0 on the 1-to-0 transition of the clock at the end of T2 and for a 1 on the 0-to-1 transition of the clock in the middle of Tw.

RDY and the 8284A

RDY is the synchronized ready input to the 8284A clock generator.

Figure (1-5) shows the internal structure of the 8284A. The bottom half of this diagram is the READY synchronization circuitry. At the leftmost side, the RDY1 and $\overline{AEN1}$ inputs are ANDed, as are the RDY2 and $\overline{AEN2}$ inputs. The outputs of the AND gates are then ORed to generate the input to the one or two stages of synchronization. In order to obtain a logic 1 at the inputs to the flip-flops, RDY1 ANDed with $\overline{AEN1}$ must be active or RDY2 ANDed with $\overline{AEN2}$ must be active.

3-Bus Cycle and Time States

A bus cycle defines the basic operation that a microprocessor performs to communicate with external devices. Example of bus cycles are the memory read, memory write, input/output read, and input output write.

As shown in Figure (1-7), a bus cycle start with an address being output on the system bus followed by a read or write data transfer. During these operations, the MP produces series of control signals to control the direction and timing of the bus. The bus cycle of the 8086 MP consists of at least four clock periods. These four time states are called T1, T2, T3, and T4. During T1, the MP put an address on the bus. For a write memory cycle, data are put on the bus during state T2 and maintained through T3and T4. When a read cycle is to be performed, the bus is first put in the high-Z state during T2 and then the data to be read must be available on the bus during T3 and T4, These four clock state gives a bus cycle duration of 200 ns *4=800ns in a 5 MHz 8086 system. If no bus cycles are required, the microprocessor performs what are known as idle state. 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.

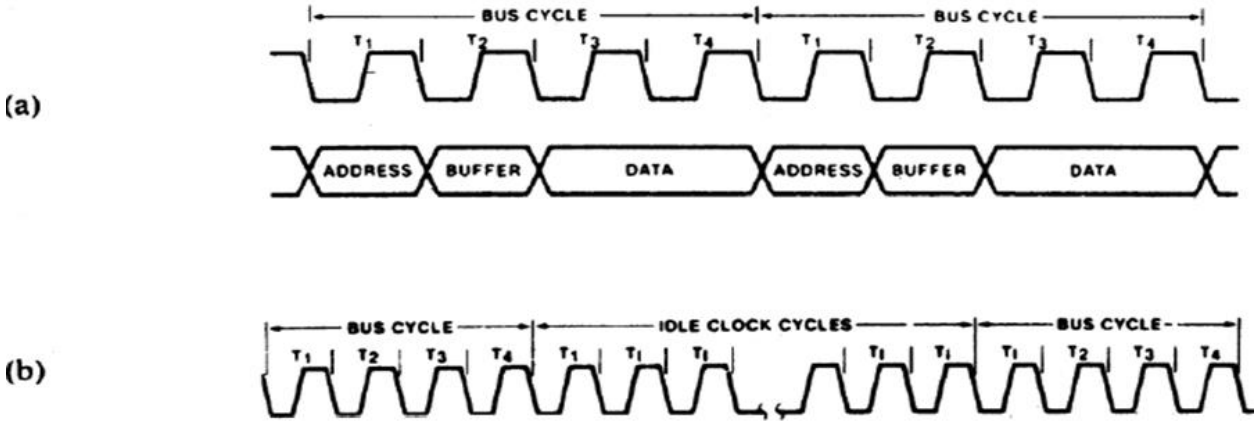


Figure (1-7): a – Bus cycle clock periods. b- Bus cycle with idle states

Basic Bus Operation

The three buses of the 8086 and 8088—address, data, and control—function in exactly the same manner as those of any other microprocessor. If data are written to the memory (see the simplified timing for write in Figure(1-8)), the microprocessor outputs the memory address on the address bus, outputs the data to be written into memory on the data bus, and issues a write ($\overline{WR}$) to memory and IO/$\overline{M}$ = 0 for the 8088 and M/$\overline{IO}$ = 1 for the 8086. If data are read from the memory (see the simplified timing for read in Figure (1-9) , the microprocessor outputs the memory address on the address bus, issues a read ($\overline{RD}$) memory signal, and accepts the data via the data bus.

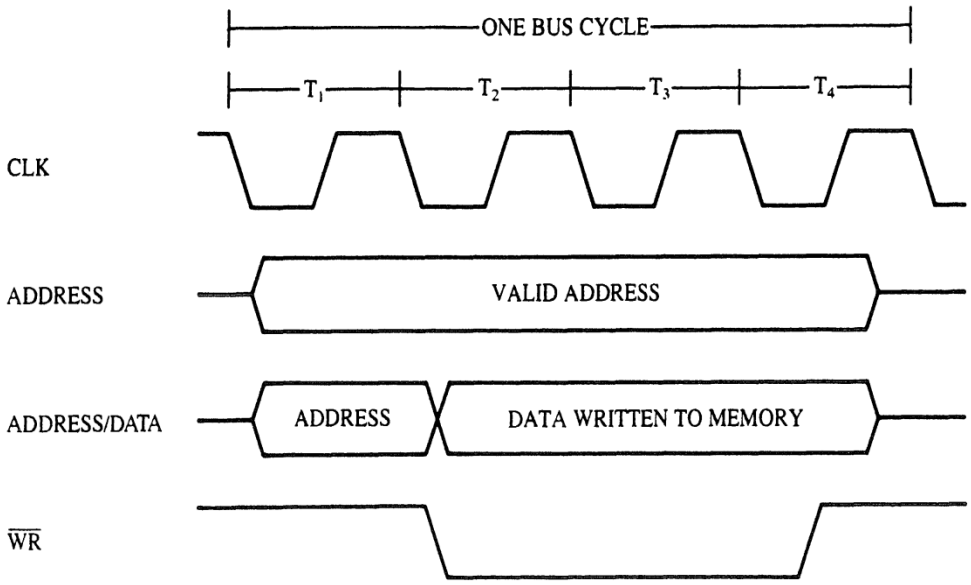


Figure (1-8): Simplified 8086/8088 write bus cycle

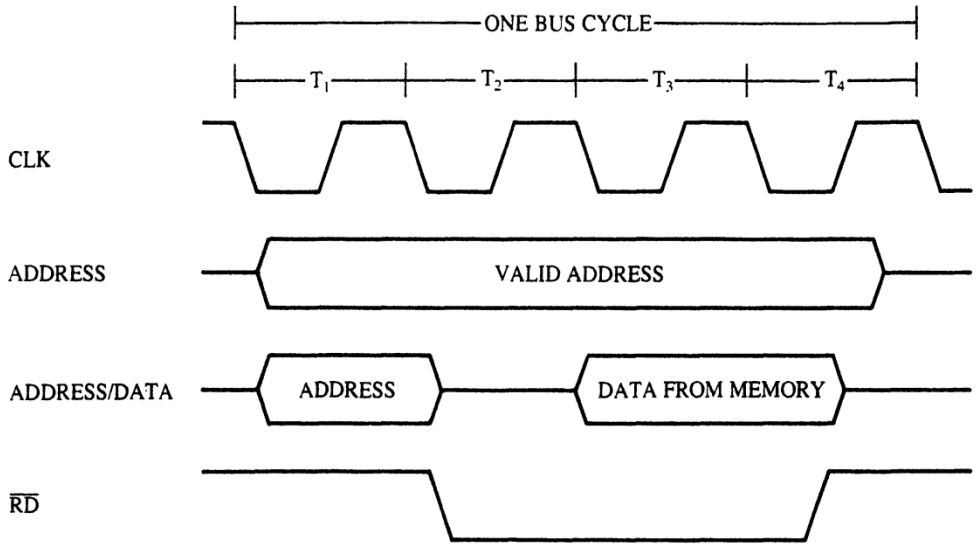


Figure (1-9): Simplified 8086/8088 read bus cycle

During the first clocking period in a bus cycle, which is called T1, many things happen. The address of the memory or I/O location is sent out via the address bus and the address/data bus connections. (The address/data bus is multiplexed and sometimes contains memory-addressing information, sometimes data.) Also output during T1 are the control signals ALE, DT/$\overline{R}$, and IO/$\overline{M}$ (8088) or M/$\overline{IO}$ (8086). The IO/$\overline{M}$ or M/$\overline{IO}$ signal indicates whether the address bus contains a memory address or an I/O device (port) number.

During T2, the 8086/8088 microprocessors issue the $\overline{RD}$ or $\overline{WR}$ signal, $\overline{DEN}$, and, in the case of a write, the data to be written appears on the data bus. These events cause the memory or I/O device to begin to perform a read or a write. The $\overline{DEN}$ signal turns on the data bus buffers, if they are present in the system, so the memory or I/O can receive data to be written or so the microprocessor can accept the data read from the memory or I/O for a read operation. If this happens to be a write bus cycle, then the data are sent out to the memory or I/O through the data bus.

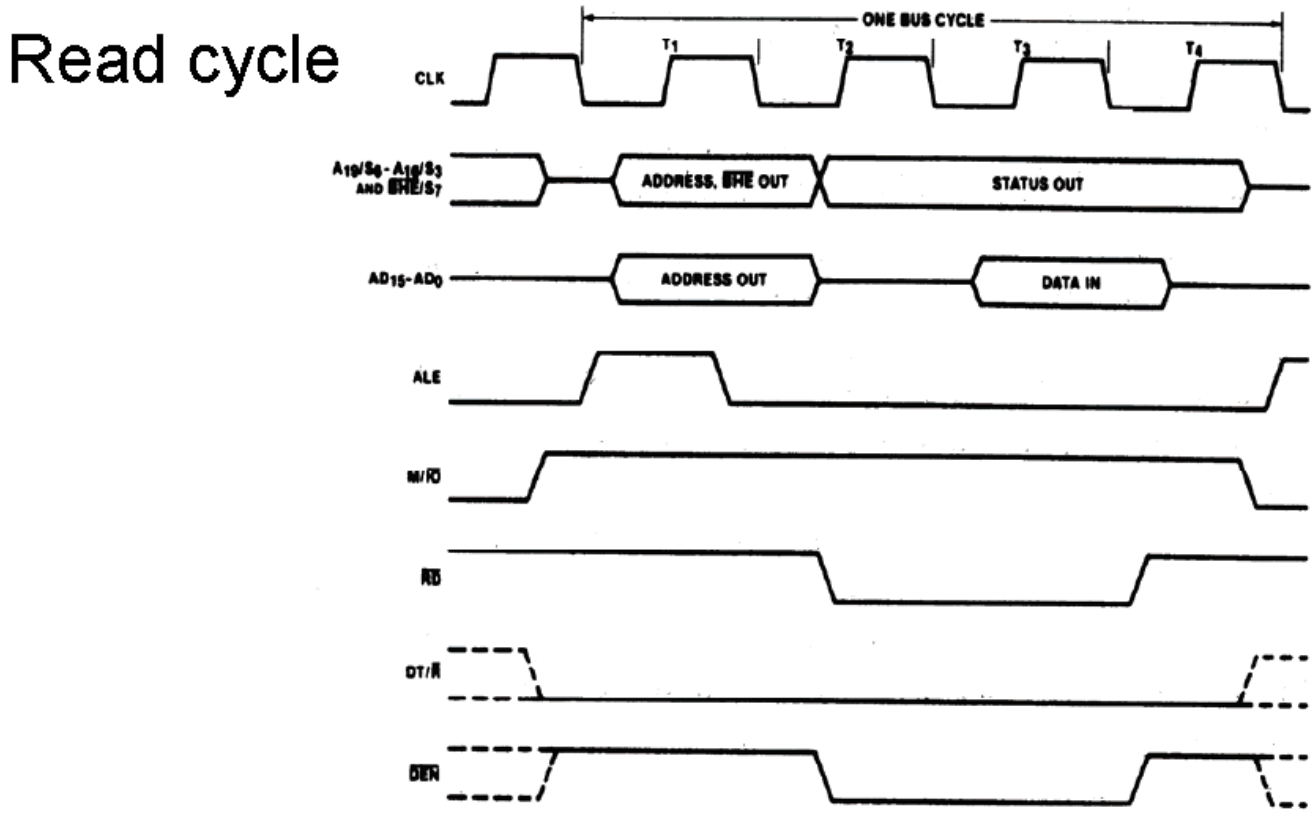


Figure (1-10): Minimum-mode memory read bus cycle of the 8086

- Write cycle

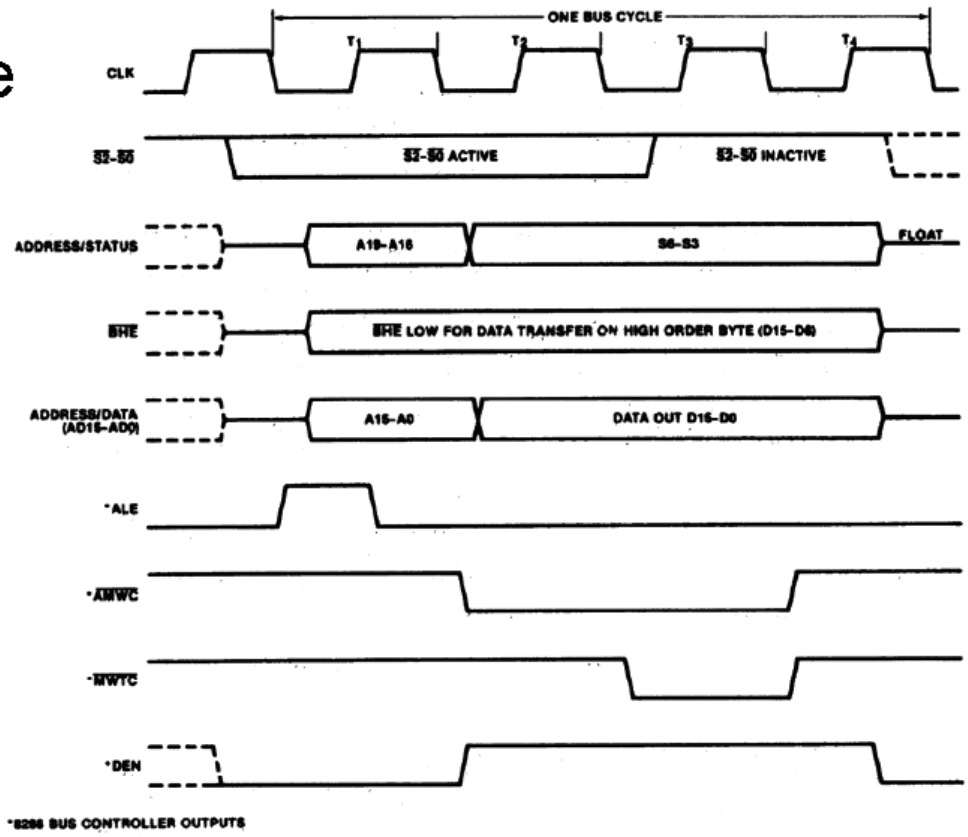


Figure (1-11):Maximum-mode memory write bus cycle of the 8086

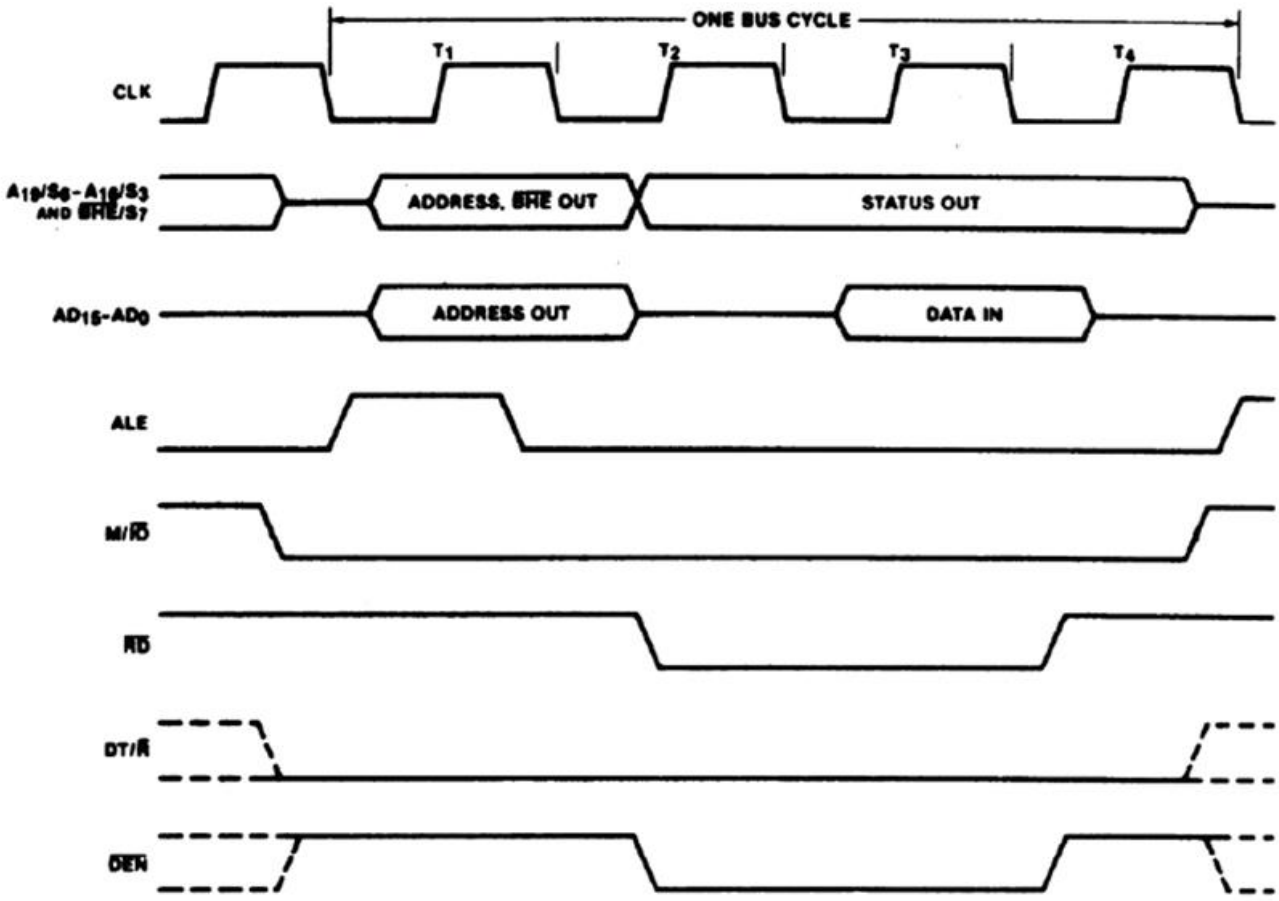


Figure (1-12): Input bus cycle of 8086 MP.

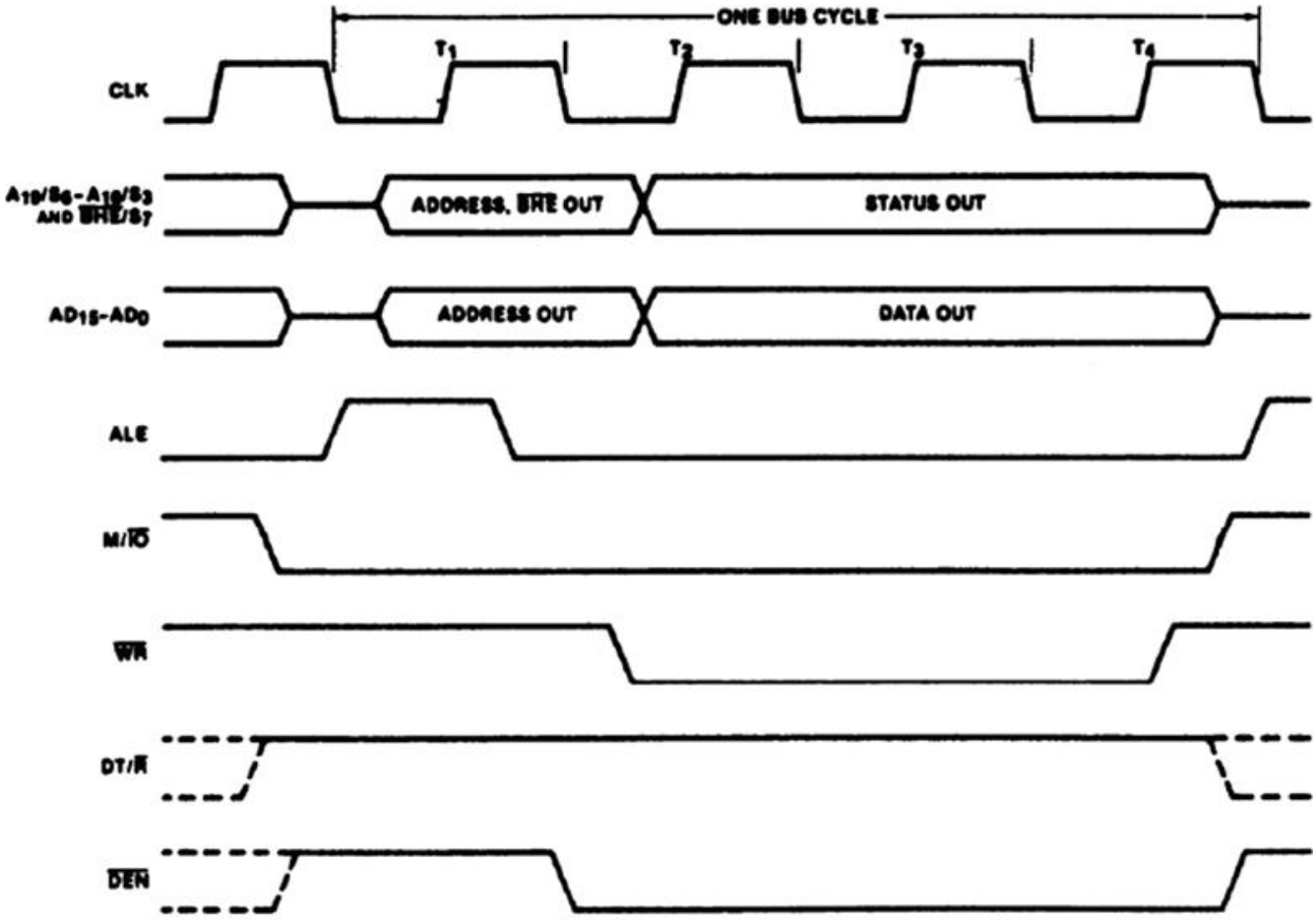


Figure (1-11): Output bus cycle of 8086 MP.

4-BUS BUFFERING AND LATCHING

Before the 8086/8088 microprocessors can be used with memory or I/O interfaces, their multiplexed buses must be demultiplexed. This section provides the detail required to demultiplex the buses and illustrates how the buses are buffered for very large systems. (Because the maximum fan-out is 10, the system must be buffered if it contains more than 10 other components.)

Demultiplexing the Buses

The address/data bus on the 8086/8088 is multiplexed (shared) to reduce the number of pins required for the 8086/8088 integrated circuit. Unfortunately, this burdens the hardware designer with the task of extracting or demultiplexing information from these multiplexed pins.

Why not leave the buses multiplexed? Memory and I/O require that the address remains valid and stable throughout a read or write cycle. If the buses are multiplexed, the address changes at the memory and I/O, which causes them to read or write data in the wrong locations.

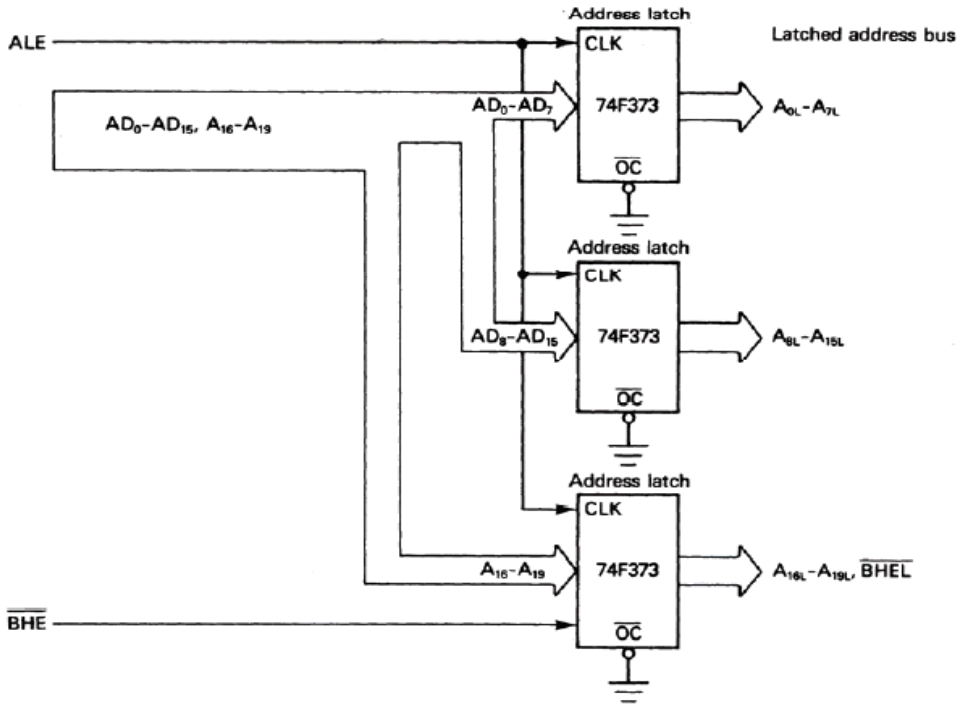
All computer systems have three buses: (1) an address bus that provides the memory and I/O with the memory address or the I/O port number, (2) a data bus that transfers data between the microprocessor and the memory and I/O in the system, and (3) a control bus that provides control signals to the memory and I/O. These buses must be present in order to interface to memory and I/O.

Demultiplexing the 8086. the 8086 system requires separate address, data, and control buses. It differs primarily in the number of multiplexed pins. In the 8088, only AD_7–AD_0 and A_{19}/S_6–A_{16}/S_3 are multiplexed. In the 8086, the multiplexed pins include AD_{15}–AD_0, A_{19}/S_6–A_{16}/S_3, and $\overline{BHE}/S_7$. All of these signals must be demultiplex

Figure (1-14) illustrates a demultiplexed 8086 with all three buses: address (A_{19}–A_0 and $\overline{BHE}$), data (D_{15}–D_0), and control (M/$\overline{IO}$, $\overline{RD}$, and $\overline{WR}$).

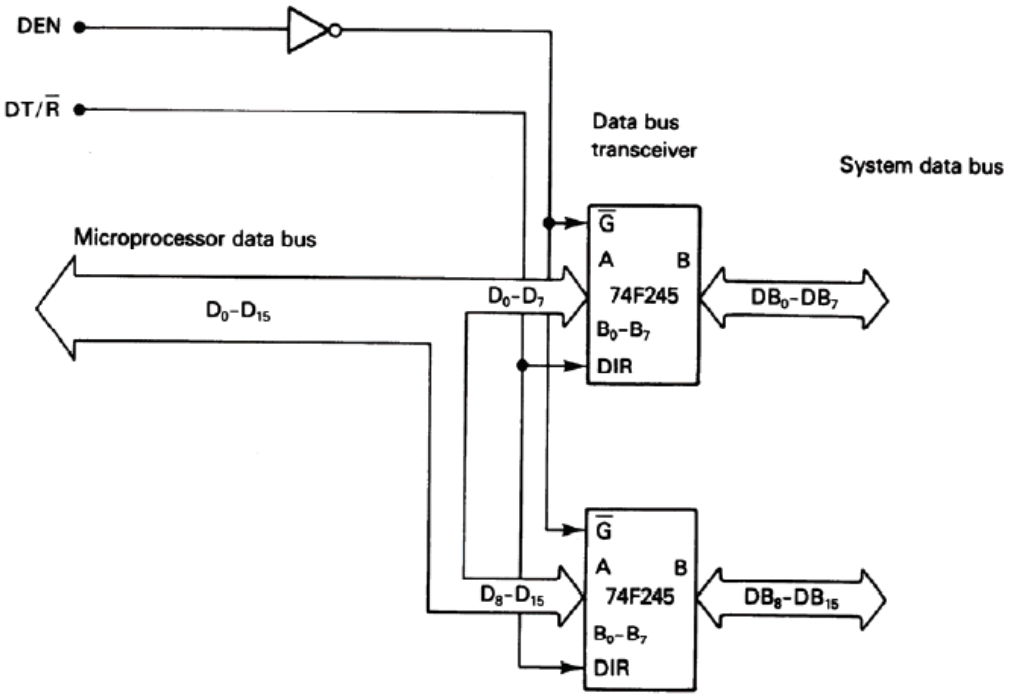
The Fully Buffered 8086. Figure (1-14) illustrates a fully buffered 8086 microprocessor. Its address pins are already buffered by the 74LS373 address latches; its data bus employs two 74LS245 octal bi-directional bus buffers; and the control bus signals, IO/$\overline{M}$, M/$\overline{IO}$, $\overline{RD}$, and $\overline{WR}$ use a 74LS244 buffer. A fully buffered 8086 system requires one 74LS244, two 74LS245s, and three 74LS373s. The 8086 requires one more buffer than the 8088 because of the extra eight data bus connections, D_{15}–D_8. It also has a $\overline{BHE}$ signal that is buffered for memory-ban selection.

Address bus latches and buffers



Address latch circuit

Data bus transceivers



Data bus transceiver circuit

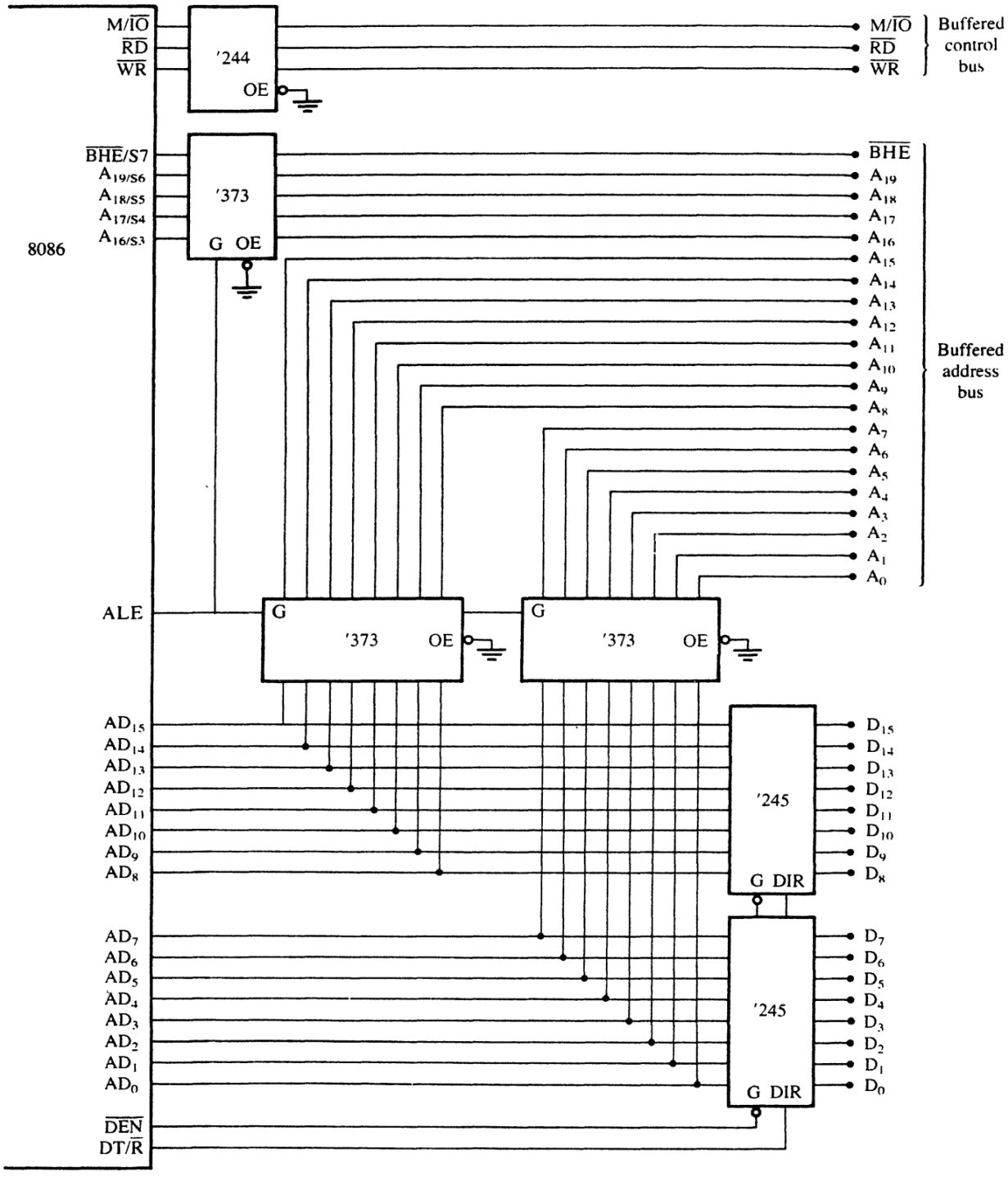


Figure (1-14): A demultiplexed address bus and fully buffered of 8086 MP buses

5-Memory Pin Connections

Pin connections common to all memory devices are the address inputs, data outputs or input/outputs, some type of selection input, and at least one control input used to select a read or write operation. See Figure (1-15) for ROM and RAM generic-memory devices.

Address Connections. All memory devices have address inputs that select a memory location within the memory device. Address inputs are almost always labeled from A_0, the least significant address input, to A_n, where subscript $_n$ can be any value but is always labeled as one less than the total number of address pins. For example, a memory device that has 10 address pins has its address pins labeled from A_0 to A_9. The number of address pins found on a memory device is determined by the number of memory locations found within it.

Today, the more common memory devices have between 1K (1,024) to 16M (16,777,216) memory locations, with 256M memory location devices on the horizon. A 1K memory device has 10 address pins (A_0–A_9); therefore, 10 address inputs are required to select any of its 1,024 memory locations. It takes a 10-bit binary number (1,024 different combinations) to select any single location on a 1,024-location device. If a memory device has 11 address connections (A_0–A_{10}), it has 2,048 (2K) internal memory locations. The number of memory locations can thus be extrapolated from the number of address pins. For example, a 4K memory device has 12 address connections, an 8K device has 13, and so forth. A device that contains 1M locations requires a 20-bit address (A_0–A_{19}).

Data Connections. All memory devices have a set of data outputs or input/outputs. The device illustrated in Figure (1-15) has a common set of input/output (I/O) connections. Today, many memory devices have bi-directional common I/O pins.

The data connections are the points at which data are entered for storage or extracted for reading. Data pins on memory devices are almost always labeled D_0 through D_7 for an 8-bit-wide memory device. In this sample memory device, there are eight I/O connections, which means that the memory device stores 8-bits of data in each of its memory locations. An 8-bit-wide memory device is often called a **byte-wide** memory. Although most devices are currently 8-bits wide, not all memory devices are 8-bits wide. Some devices are 16-bits, 4-bits, or just 1-bit wide.

Selection Connections. Each memory device has an input—sometimes more than one—that selects or enables the memory device. This kind of input is most often called a **chip select** ($\overline{CS}$), **chip enable** ($\overline{CE}$), or simply **select** ($\overline{S}$) input. RAM memory generally has at least one $\overline{CS}$ or $\overline{S}$ input, and ROM at least one $\overline{CE}$. If the $\overline{CE}$, $\overline{CS}$, or $\overline{S}$ input is active (a logic 0 in this case, becaus of the over-bar), the memory device performs a read or a write; if it is inactive (a logic 1 in this case), the memory device cannot do a read or a write because it is turned off or disabled. If more than one $\overline{CS}$ connection is present, all must be activated to read or write data.

Control Connections. All memory devices have some form of control input or inputs. A ROM usually has only one control input, while a RAM often has one or two control inputs.

The control input most often found on a ROM is the **output enable** ($\overline{OE}$) or **gate** ($\overline{G}$) connection, which allows data to flow out of the output data pins of the ROM. If $\overline{OE}$ and the selection input are both active, then the output is enabled; if $\overline{OE}$ is inactive, the output is disabled at its high-impedance state. The $\overline{OE}$ connection enables and disables a set of three-state buffers located within the memory device and must be active to read data.

A RAM memory device has either one or two control inputs. If there is one control input, it is often called R/$\overline{W}$. This pin selects a read operation or a write operation only if the device is selected by the selection input ($\overline{CS}$). If the RAM has two control inputs, they are usually labeled $\overline{WE}$ (or $\overline{W}$) and $\overline{OE}$ (or $\overline{G}$). Here, $\overline{WE}$ **(write enable)** must be active to perform a memory write operation, and OE must be active to perform a memory read operation. When these two controls ($\overline{WE}$) and $\overline{OE}$) are present, they must never both be active at the same time. If both control inputs are inactive (logic 1's), then data are neither written nor read and the data connections are at their high-impedance state.

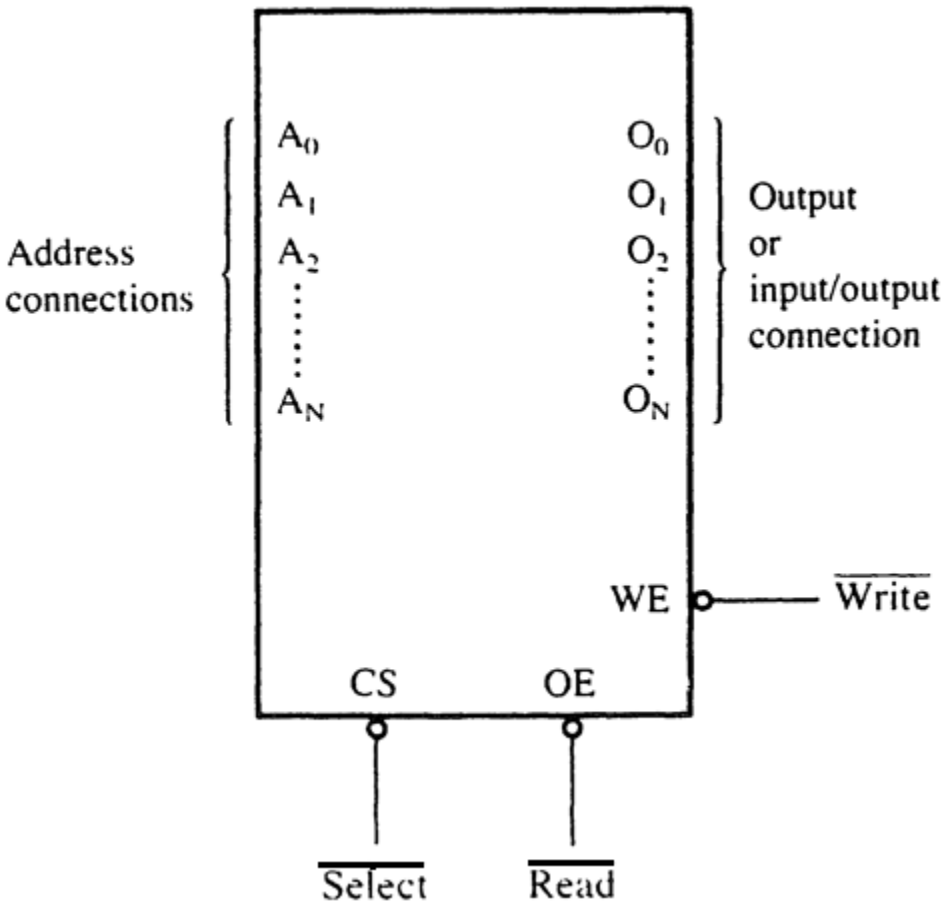


Figure (1-15): Memory Pin Connection.

ROM Memory

The read-only memory (ROM) permanently stores programs and data that are resident to the system and must not change when power is disconnected. The ROM is permanently programmed so data are always present, even when power is disconnected. This type of memory is often called *nonvolatile memory.*

The ROM is available in many forms today. A device we call a ROM is purchased in mass quantities from a manufacturer and programmed during its fabrication at the factory. The EPROM **(erasable programmable read-only memory),** a type of ROM, is more commonly used when software must be changed often or when too limited a number are in demand to make the ROM economical. For a ROM to be practical, we usually must purchase at least 10,000 devices. An EPROM is programmed in the field on a device called an *EPROM programmer.* The EPROM is also erasable if exposed to high-intensity ultraviolet light for about 20 minutes or less, depending on the type of EPROM.

PROM memory devices are also available, but they are not as common today. The PROM **(programmable read-only memory)** is also programmed in the field by burning open tiny Nichrome or silicon oxide fuses, but once programmed it cannot be erased.

Still another, newer type of **read-mostly memory** (RMM) is called the *flash memory.* The flash memory is also often called an EEPROM *(electrically erasable programmable ROM),* EAROM *(electrically alterable ROM),* or a NOVRAM *(nonvolatile RAM).* These memory devices are electrically erasable in the system, but require more time to erase than a normal RAM. The flash memory device is used to store setup information for systems such as the video card in the computer. It may also soon replace the EPROM in the computer for the BIOS memory. Some systems contain a password stored in the flash memory device.

Static Ram (SRAM) Devices

Static RAM memory devices retain data for as long as DC power is applied. Because no special action (except power) is required to retain stored data, these devices are called *static memory.* They are also called *volatile memory* because they will not retain data without power. The main difference between a ROM and a RAM is that a RAM is written under normal operation, while a ROM is programmed outside the computer and is only normally read.

Dynamic Ram (DRAM) Memory

About the largest static RAM available today is a 128K × 8. Dynamic RAMs, on the other hand, are available in much lager sizes: up to 16M × 1. In all other respects, DRAM is essentially the same as SRAM, except that it retains data for only 2 or 4 ms on an integrated capacitor. After 2 or 4 ms, the contents of the DRAM must be completely rewritten *(refreshed)* because the capacitors, which store a logic 1 or logic 0, lose their charges.

PLD

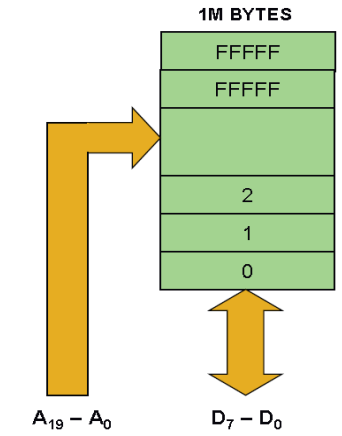
This section of the text explains the use of the programmable logic device or PLD as a decoder. Recently, the PAL has replaced PROM address decoders in the latest memory interfaces. There are three PLD devices that function in basically the same manner, but have different names: PLA **(programmable logic array),** PAL **(programmable array logic),** and GAL **(gated array logic).** Although these devices have been in existence since the mid-1970s, they have only recently appeared in memory systems and digital designs. The PAL and the PLA are fuse programmed, as is the PROM, and some PLD devices are erasable, as are EPROMs. In essence, all three devices are arrays of logic elements that are programmable.

- Major types of programmable logic architecture
  - Simple Programmable Logic Devices (**SPLDs**)
    - PAL, GAL, PLA
  - Complex Programmable Logic Devices (**CPLDs**)
  - Field Programmable Gate Arrays (**FPGAs**)

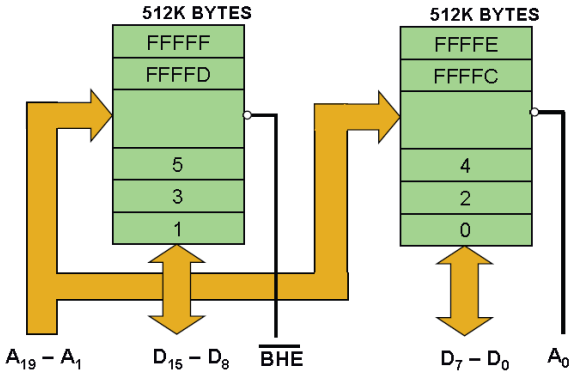
- Memory Organization of 8086 MP:

Figure (1-16) shows that the 8086's 1M byte memory address space is implemented as two independent 512k byte 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 high bank.

- A_1-A_{19} selects the storage location that is to be accessed. They are applied to both banks in parallel.
- A_0 and $\overline{BHE}$ are used as bank-select signals.
- The lower bank transfers bytes of data over data lines D_0-D_7, while data transfer for a high bank use D_8-D_{15}.

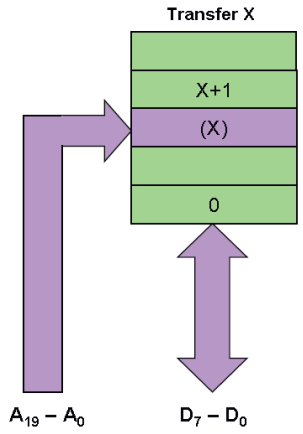


1Mx8 memory bank of the 8088

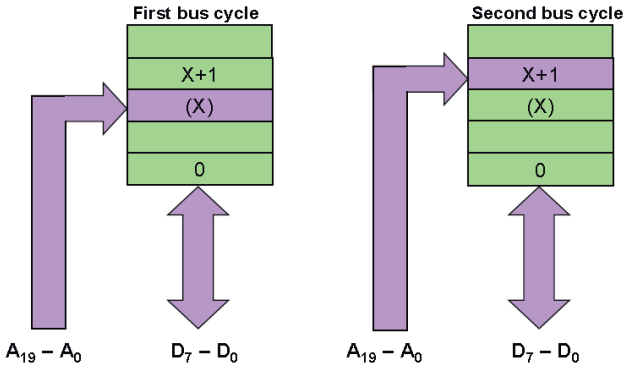


High and low memory banks of the 8086

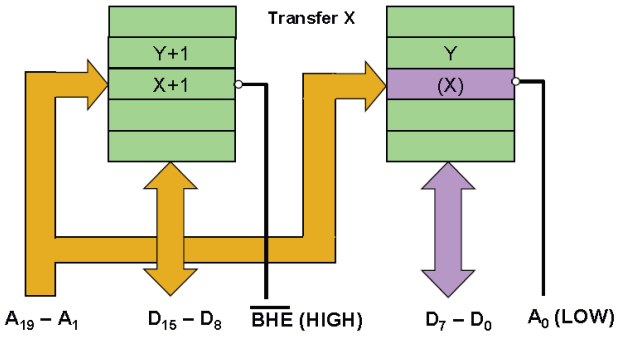
Figure (1-16): High and low memory banks of the 8086 MP.



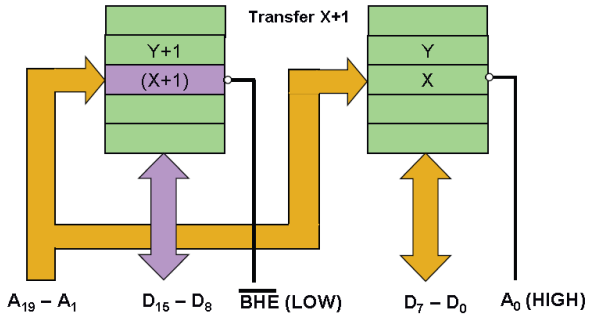
Byte transfer by the 8088



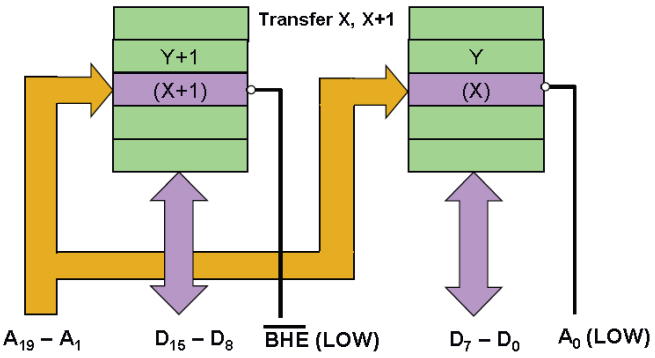
Word transfer by the 8088



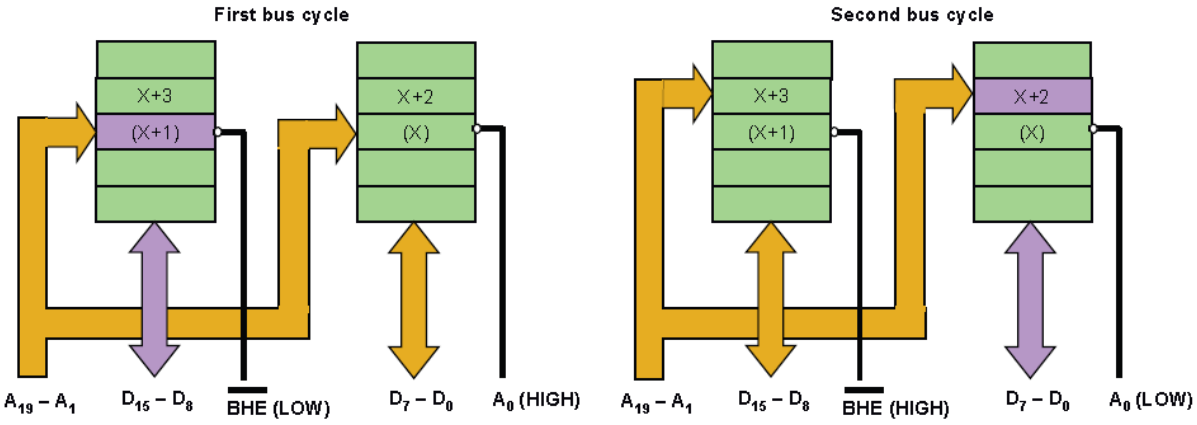
Even address byte transfer by the 8086



Odd address byte transfer by the 8086



Even address word transfer by the 8086

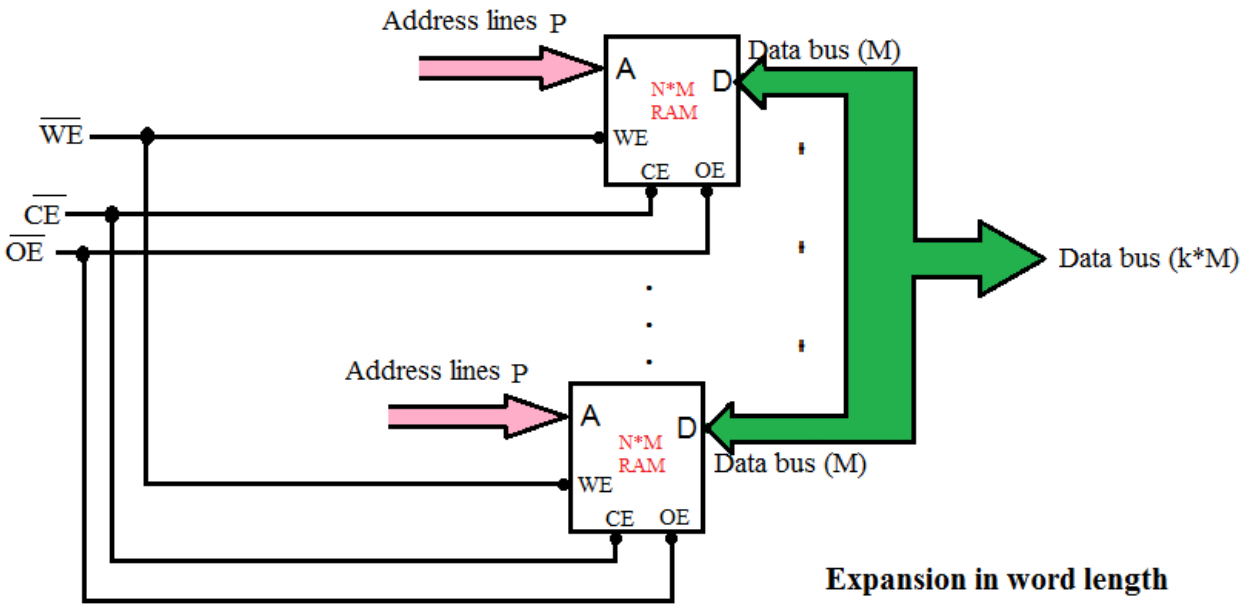


Odd-address word transfer by the 8086

- Memory System Design:

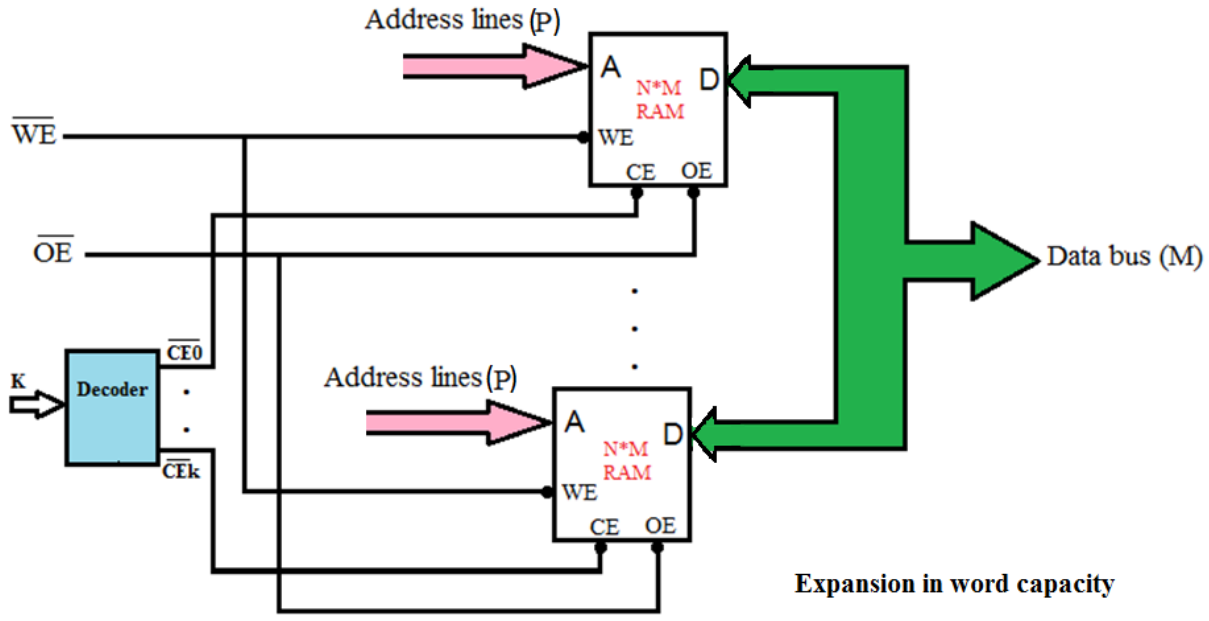
ICs are organized as N*M bits array (Memory size = N*M bits arrays). Where N is no. of Storage Location (Word Capacity) and M is no. of Bits (Word Length). It is possible to increase the storage locations N or increase number of bits M or increase both.

- Increase No. of Bits (M): Expansion in Word Length Design (N * k M) RAM by using (N*M) RAM.



Expansion in word length

- Increase the storage locations (N): Expansion in Word Capacity.
  Design (2^k N * M) RAM by using (N*M) RAM.



Expansion in word capacity

*** Address Bus Status Codes:** When a bus cycle is a memory bus cycle an address bus status code S4S3 is output by the processor. Bits S4 and S3 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 table (1-1). These status codes are output in both the minimum and the maximum modes.

*** Memory Control Signals:** In this section we will explain the memory control signals and there function with respect to memory interface operation.

Minimum-mode Memory Control Signals: a minimum mode memory interface of 8086 based microcomputer, we find that the control signals provided to support the interface to the memory subsystem are ALE, M/$\overline{IO}$, DT/$\overline{R}$, $\overline{RD}$, $\overline{WR}$, DEN, and $\overline{BHE}$.

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.

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.

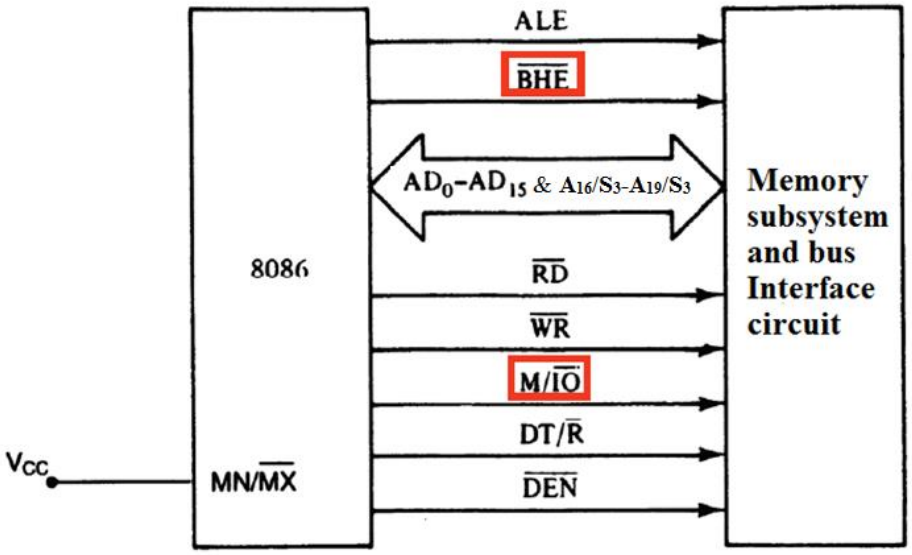
The input-output/memory (M/$\overline{IO}$) and data transmit/receive (DT/$\overline{R}$) lines signal external circuitry whether a memory or I/O bus cycle is active and whether the 8086 will transmit or receive data over the bus.

The signals *read* ($\overline{RD}$) and *write* ($\overline{WR}$) identify that a read or write bus cycle respectively, is in progress. During all memory operations, the 8086 produces one other control signal, *data enable* ($\overline{DEN}$). Logic 0 at this output is used to enable the data bus.

$\overline{BHE}$ 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.

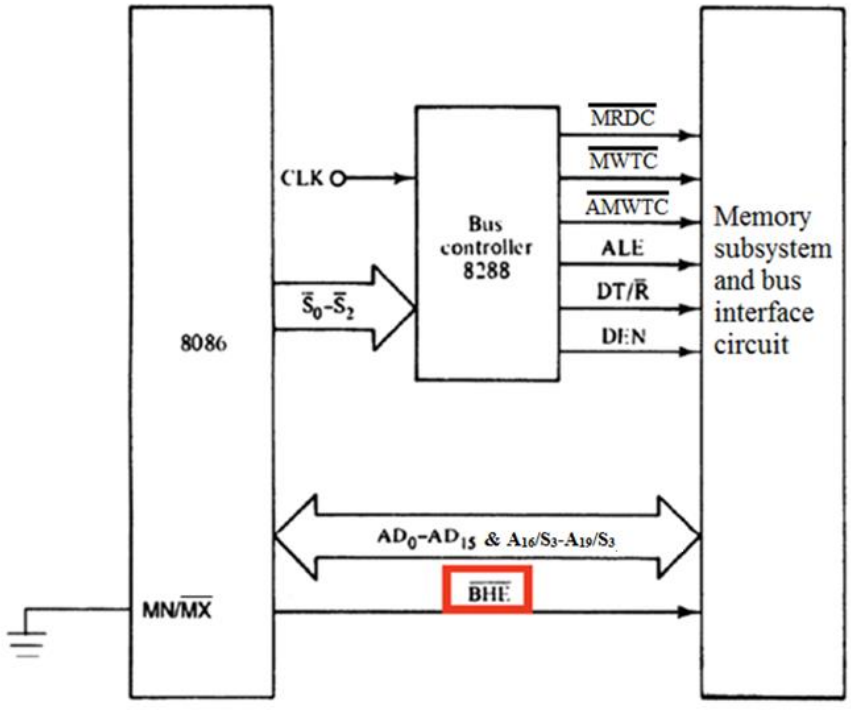
Maximum-mode Memory Control Signals: When the 8086 is configured to work in the maximum mode, it does not directly provide all the control signals to support the memory interface. Instead, an external bus controller, the 8288, provides memory commands and control signals.

Minimum-mode interface



Minimum-mode 8086 system memory interface.

Maximum-mode interface



Maximum-mode 8086 system memory interface.

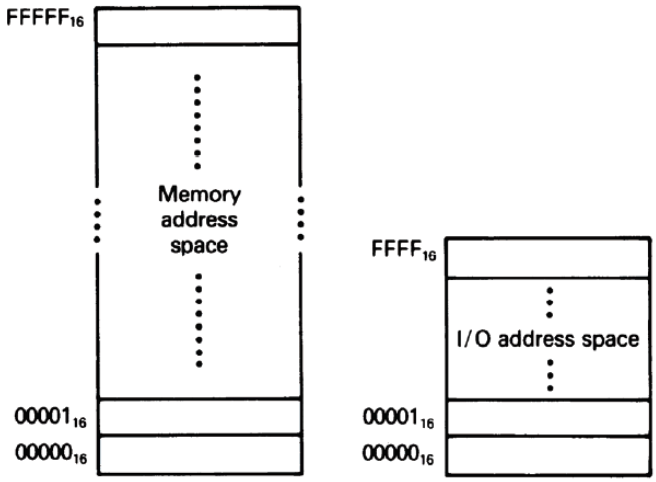
Basic I/O Interface

Isolated and Memory-Mapped I/O

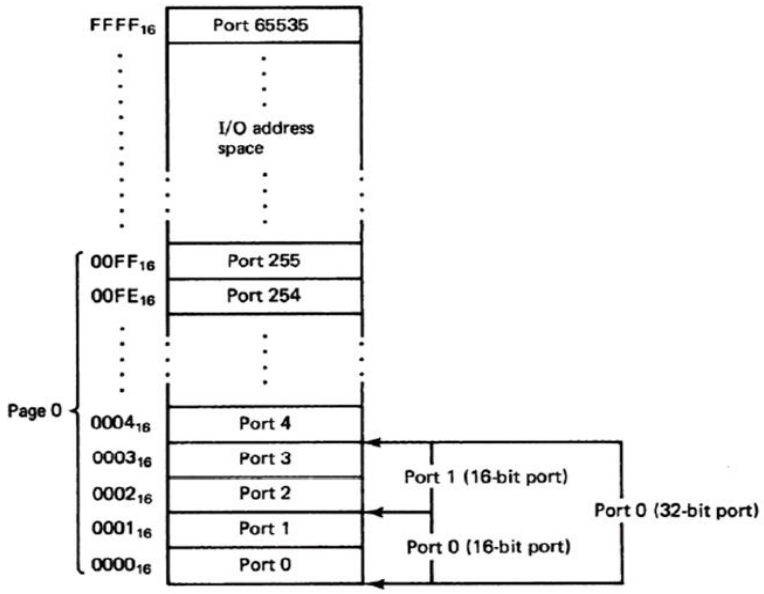
There are two completely different methods of interfacing I/O to the microprocessor: **isolated I/O** and **memory-mapped I/O.** In isolated I/O, the IN, INS, OUT, and OUTS instructions transfer data between the microprocessor accumulator or memory and the I/O device. In memory-mapped I/O, any instruction that references memory can accomplish the transfer. Both isolated and memory-mapped I/O are in use, so both are discussed in this text.

Isolated I/O. The most common I/O transfer technique used in the Intel microprocessor-based system is isolated I/O. The term *isolated* describes how the I/O locations are isolated from the memory system in a separate I/O address space. (Figure(1-17)illustrates both the isolated and memory-mapped address spaces

- Isolated input/output
  - When using isolated I/O in a microcomputer system, the I/O device are treated separate from memory.
  - The memory address space contains 1 M consecutive byte address 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 $FFFF_{16}$.
  - All input and output data transfers must take place between the AL or AX register and I/O port.
    - Isolated input/output

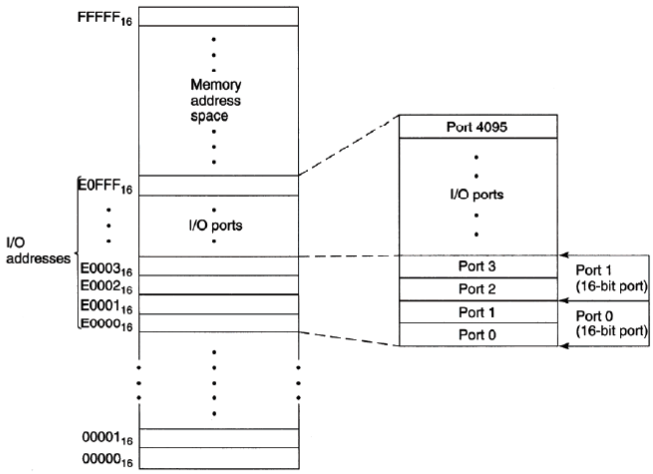


8086 memory and I/O address spaces



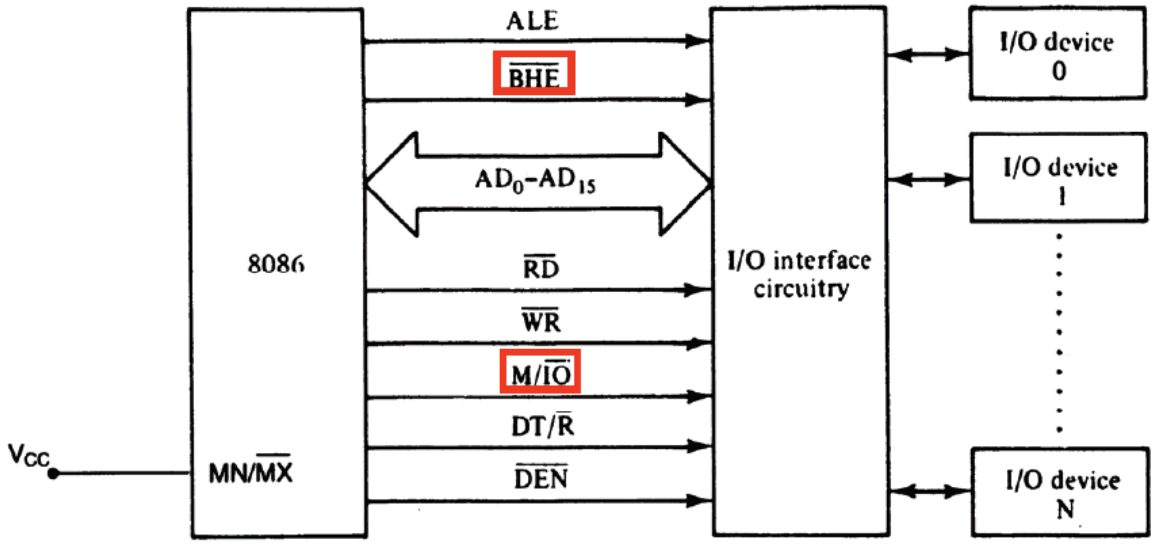
Isolated I/O ports

- Memory-mapped input/output
  - In the case of memory-mapped I/O, MPU looks at the I/O port as though it is a storage location in memory.
  - Some of the memory address space is dedicated to I/O ports.
  - Instructions that affect data in memory are used instead of the special I/O instructions.
  - The memory instructions tend to execute slower than those specifically designed for isolated I/O.
- Memory-mapped input/output



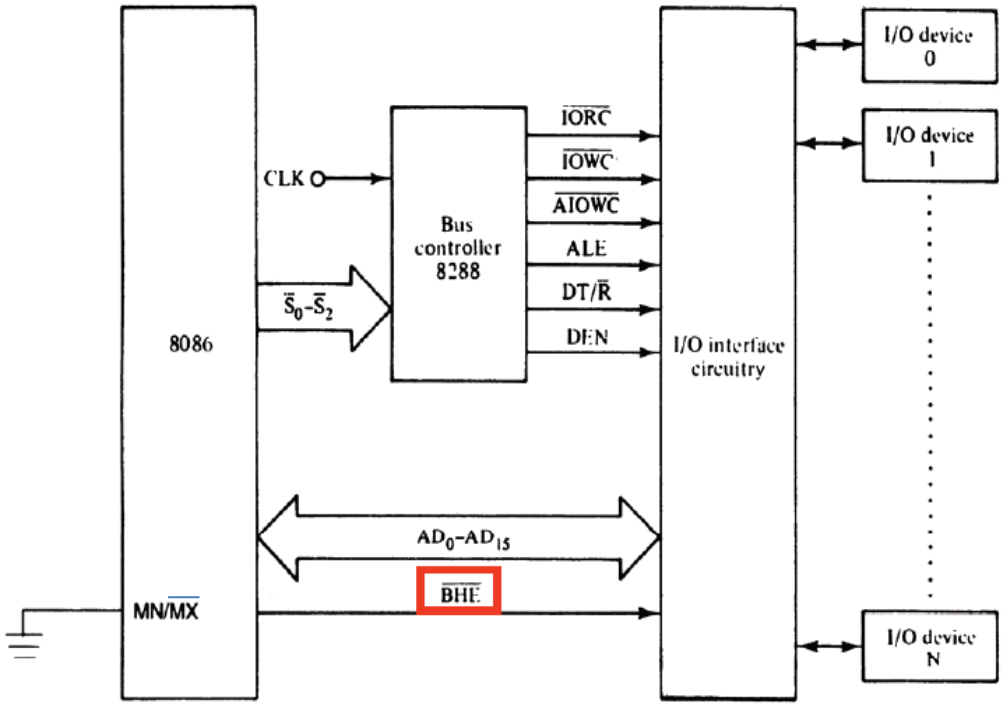
Memory mapped I/O ports

Minimum-mode interface



Minimum-mode 8086 system I/O interface

Maximum-mode interface



Maximum-mode 8086 system I/O interface

Input/Output Instructions

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)

Input/Output Instructions

EXAMPLE

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
```

EXAMPLE

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-wide output port at address $B000_{16}$. Write a sequence of instructions to perform this input/output operation.

Solution:

First read in the byte at address AA_{16} into AL and move it into AH.

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

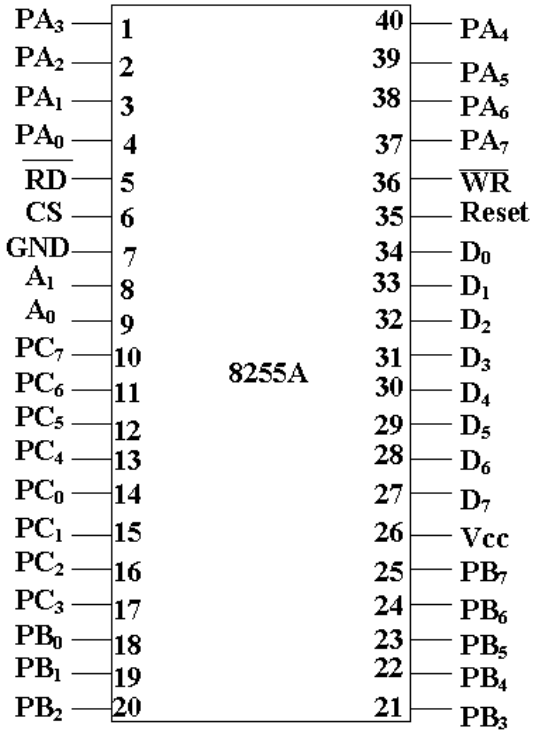
Now the other byte can be read into AL by the instruction

```
IN      AL, 09AH
```

And to write out the word of data

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

THE PROGRAMMABLE PERIPHERAL INTERFACE



8255A Pin Configuration

The parallel input-output port chip 8255 is also called as programmable ***peripheral input-output port.*** The Intel's 8255 is designed for use with Intel's 8-bit, 16-bit and higher capability microprocessors. It has 24 input/output lines which may be individually programmed in two groups of twelve lines each, or three groups of eight lines. The pin configuration of 8255 is shown in figure above.

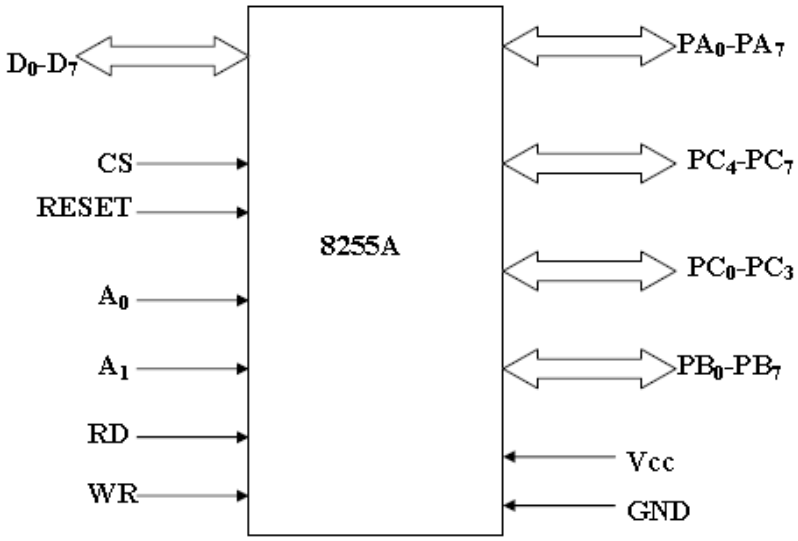
The two groups of I/O pins are named as Group A and Group B. Each of these two groups contains a subgroup of eight I/O lines called as 8-bit port and another subgroup of four lines or a 4-bit port. Thus Group A contains an 8-bit port A along with a 4-bit port C upper.

The port A lines are identified by symbols PA0-PA7 while the port C lines are identified as PC4-PC7. Similarly, Group B contains an 8-bit port B, containing lines PB0-PB7 and a 4-bit port C with lower bits PC0- PC3. The port C upper and port C lower can be used in combination as an 8-bit port C.

Both the port C are assigned the same address. Thus one may have either three 8-bit I/O ports or two 8-bit and two 4-bit ports from 8255. All of these ports can function independently either as input or as output ports. This can be achieved by programming the bits of an internal register of 8255 called as control word register (CWR).

The internal block diagram and is shown in figure below. The 8-bit data bus buffer is controlled by the read/write control logic. The read/write control logic manages all of the internal and external transfers of both data and control words.

RD, WR, A1, A0 and RESET are the inputs provided by the microprocessor to the READ/ WRITE control logic of 8255. The 8-bit, 3-state bidirectional buffer is used to interface the 8255 internal data bus with the external system data bus. This buffer receives or transmits data upon the execution of input or output instructions by the microprocessor. The control words or status information is also transferred through the buffer.



Signals of 8255

The signals descriptions of 8255 are briefly presented as follows:

• **PA7-PA0**: These are eight port A lines that acts as either latched output or buffered input lines depending upon the control word loaded into the control word register.

• **PC7-PC4:** Upper nibble of port C lines. They may act as either output latches or input buffers lines.

• This port also can be used for generation of handshake lines in mode 1 or mode 2.

• **PC3-PC0:** These are the lower port C lines; other details are the same as PC7-PC4 lines.

• **PB0-PB7:** These are the eight port B lines which are used as latched output lines or buffered input lines in the same way as port A.

• **RD**: This is the input line driven by the microprocessor and should be low to indicate read operation to 8255.

• **WR**: This is an input line driven by the microprocessor. A low on this line indicates write operation.

• **CS**: This is a chip select line. If this line goes low, it enables the 8255 to respond to RD and WR signals, otherwise RD and WR signal are neglected.

• **A1-A0:** These are the address input lines and are driven by the microprocessor.

These lines A1-A0 with RD, WR and CS form the following operations for 8255. These address lines are used for addressing any one of the four registers, i.e. three ports and a control word register as given in table below.

$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	A_1	A_0	Input (Read) cycle
0	1	0	0	0	Port A to Data bus
0	1	0	0	1	Port B to Data bus
0	1	0	1	0	Port C to Data bus
0	1	0	1	1	CWR to Data bus

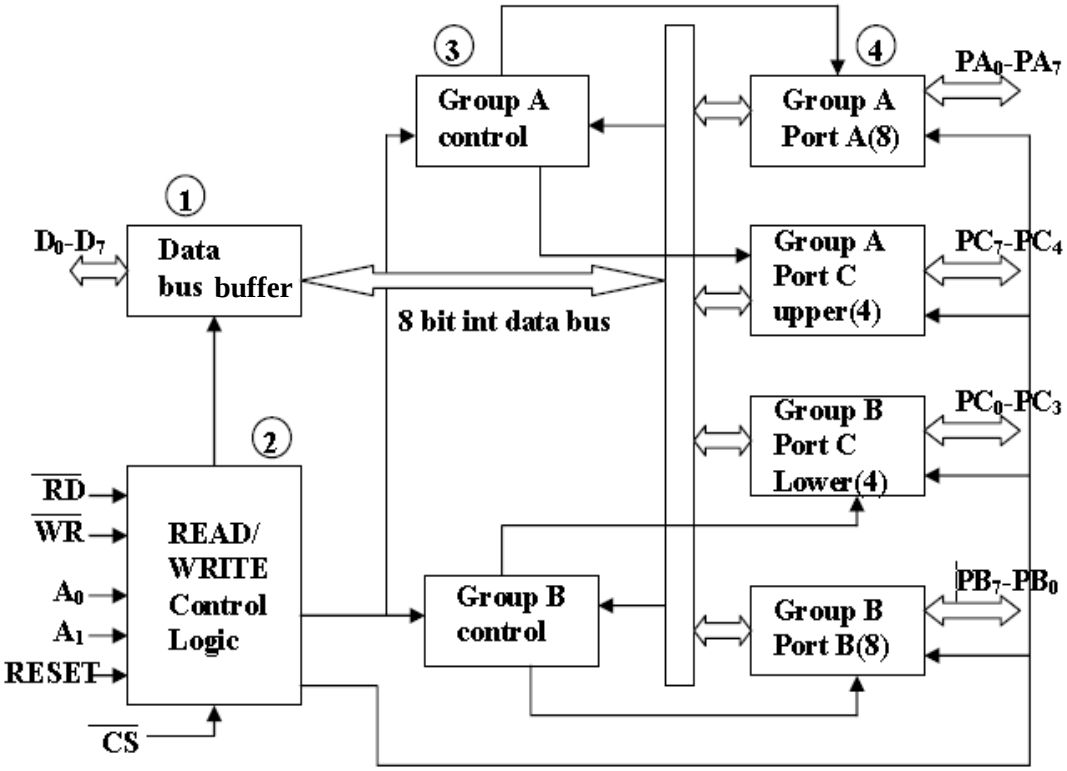
$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	A_1	A_0	Output (Write) cycle
1	0	0	0	0	Data bus to Port A
1	0	0	0	1	Data bus to Port B
1	0	0	1	0	Data bus to Port C
1	0	0	1	1	Data bus to CWR

• **D0-D7:** These are the data bus lines those carry data or control word to/from the microprocessor.

• **RESET:** Logic high on this line clears the control word register of 8255. All ports are set as input ports by default after reset.

Block Diagram of 8255 (Architecture):

1. Data bus buffer.
2. Read Write control logic.
3. Group A and Group B controls.
4. Port A, B and C.



Block Diagram of 8255

• ***Data bus buffer***: This is a tristate bidirectional buffer used to interface the 8255 to system data bus. Data is transmitted or received by the buffer on execution of input or output instruction by the CPU. Control word and status information are also transferred through this unit.

• ***Read/Write control logic***: This unit accepts control signals (RD, WR) and also inputs from address bus and issues commands to individual group of control blocks (Group A, Group B). It has the following pins:

a) CS – Chip select: A low on this PIN enables the communication between CPU and 8255.

b) RD (Read): A low on this pin enables the CPU to read the data in the ports or the status word through data bus buffer.

c) WR (Write): A low on this pin, the CPU can write data on to the ports or on to the control register through the data bus buffer.

d) RESET: A high on this pin clears the control register and all ports are set to the input mode

e) **A0** and **A1** (Address pins): These pins in conjunction with RD and WR pins control the selection of one of the three ports.

• ***Group A and Group B controls***: These blocks receive control from the CPU and issues commands to their respective ports.

• Group A - PA and PCU (PC7 –PC4)

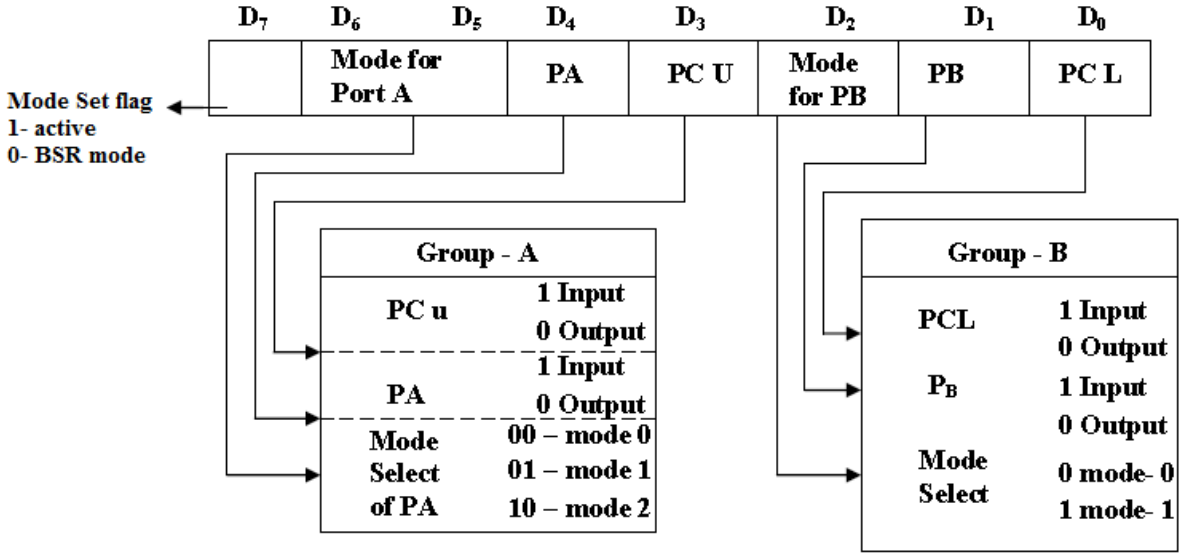
• Group B - PCL (PC3 – PC0)

• Control word register can only be written into no read operation of the CW register is allowed.

a) **Port A**: This has an 8 bit latched/buffered O/P and 8 bit input latch. It can be programmed in three modes – mode 0, mode 1, and mode 2.

b) **Port B**: This has an 8 bit latched / buffered O/P and 8 bit input latch. It can be programmed in mode 0, mode1.

c) **Port C**: This has an 8 bit latched input buffer and 8 bit output latched/buffer. This port can be divided into two 4 bit ports and can be used as control signals for port A and port B. it can be programmed in mode 0.



Control Word Format of 8255

To communicate with peripherals through the 82C55A, three steps are necessary:

1. Determine the addresses of ports A, B, and C and the control word of the control register according to Chip select logic and address lines A0 and A1.
2. Write a control word in the control register.
3. Write I/O instruction to communicate with peripherals through ports A, B, and C.

Modes of Operation of 8255

These are two basic modes of operation of 8255, I/O mode and Bit Set-Reset mode (BSR). In I/O mode, the 8255 ports work as programmable I/O ports, while in BSR mode only port C (PC0-PC7) can be used to set or reset its individual port bits. Under the I/O mode of operation, further there are three modes of operation of 8255, so as to support different types of applications, mode 0, mode 1 and mode 2.

All these modes can be selected by programming a register internal to 8255 known as CWR. The control word register has two formats. The first format is

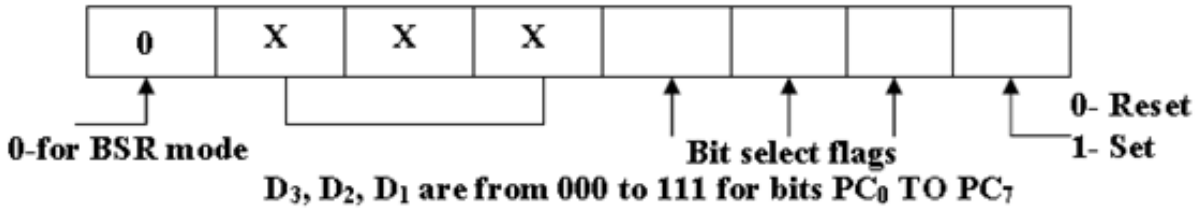
valid for I/O modes of operation, i.e. modes 0, mode 1 and mode 2 while the second format is valid for bit set/reset (BSR) mode of operation.

BSR Mode: In this mode any of the 8-bits of port C can be set or reset depending on D0 of the control word. The bit to be set or reset is selected by bit select flags D3, D2 and D1 of the CWR as given in table.

Table: D3, D2 and D1 of the CWR.

D_3	D_2	D_1	Selected bits of port C
0	0	0	PC_0
0	0	1	PC_1
0	1	0	PC_2
0	1	1	PC_3
1	0	0	PC_4
1	0	1	PC_5
1	1	0	PC_6
1	1	1	PC_7

BSR mode control word register format is shown in following figure.



BSR Mode Control Word Register Format

I/O Modes:

a) ***Mode 0 (Basic I/O mode):*** This mode is also called as basic input/output mode. This mode provides simple input and output capabilities using each of the three ports. Data can be simply read from and written to the input and output ports respectively, after appropriate initialization. The salient features of this mode are as listed below:

1. Two 8-bit ports (port A and port B) and two 4-bit ports (port C upper and lower) are available. The two 4-bit ports can be combined used as a third 8-bit port.

2. Any port can be used as an input or output port.

3. Output ports are latched. Input ports are not latched.

b) Mode 1*(Strobed input/output mode)*: in this mode the handshaking control the input and output action of the specified port. Port C lines PC0-PC2; provide strobe or handshake lines for port B. This group which includes port B and PC0-PC2 is called as group B for Strobed data input/output. Port C lines (PC3-PC5) provide strobe lines for port A.

This group is including port A and PC3-PC5 from group A. Thus port C is utilized for generating handshake signals. The salient features of mode 1 are listed as follows:

1. Two groups – group A and group B are available for strobed data transfer.
2. Each group contains one 8-bit data I/O port and one 4-bit control/data port.
3. The 8-bit data port can be either used as input and output port. The inputs and outputs both are latched.
4. Out of 8-bit port C, PC0-PC2 are used to generate control signals for port B and PC3-PC5 are used to generate control signals for port A. The lines PC6, PC7 may be used as independent data lines.

The control signals for both the groups in input and output modes are explained as follows:

Input control signal definitions (mode 1):

• STB(Strobe input) – If this lines falls to logic low level, the data available at 8-bit input port is loaded into input latches.

• **IBF** (Input buffer full) – If this signal rises to logic 1, it indicates that data has been loaded into latches, i.e. it works as an acknowledgement. IBF is set by a low on STB and is reset by the rising edge of RD input.

• **INTR** (Interrupt request) – This active high output signal can be used to interrupt the CPU whenever an input device requests the service. INTR is set by a high STB pin and a high at IBF pin. INTE is an internal flag that can be controlled by the bit set/reset mode of either PC4 (INTEA) or PC2 (INTEB) as shown in the figure.

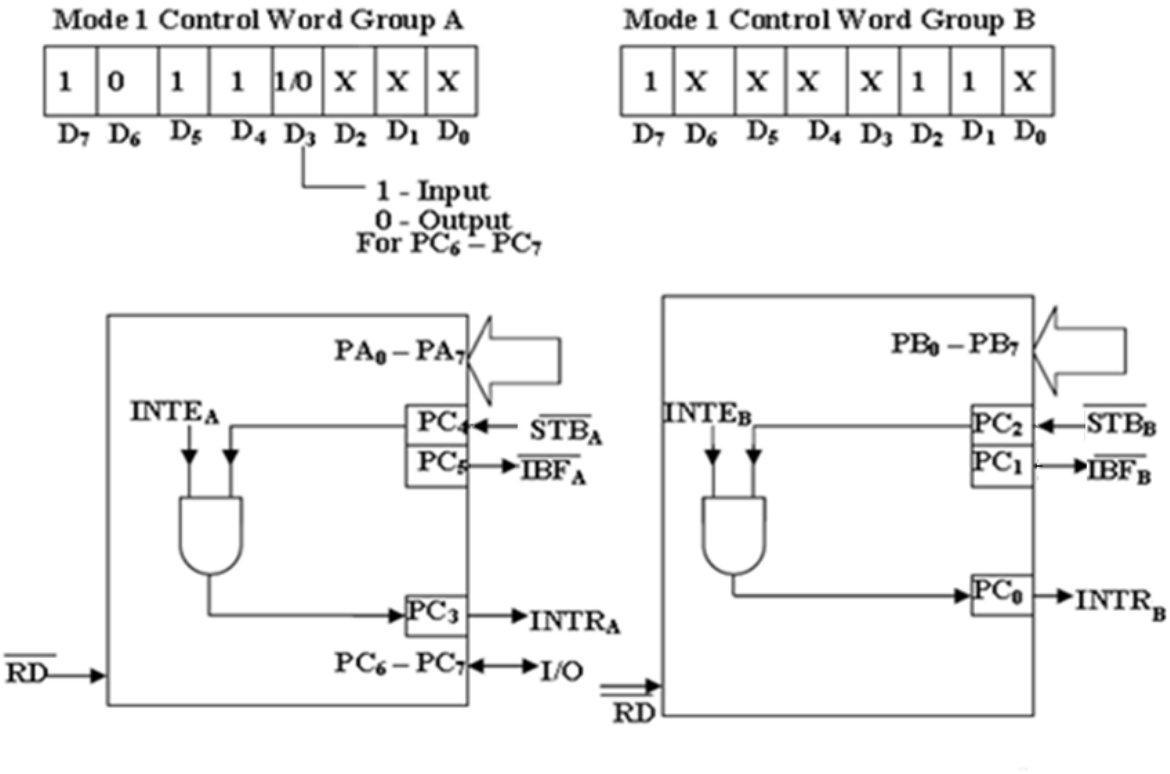
INTR is reset by a falling edge of RD input. Thus an external input device can be request the service of the processor by putting the data on the bus and sending the strobe signal.

Output control signal definitions (mode 1):

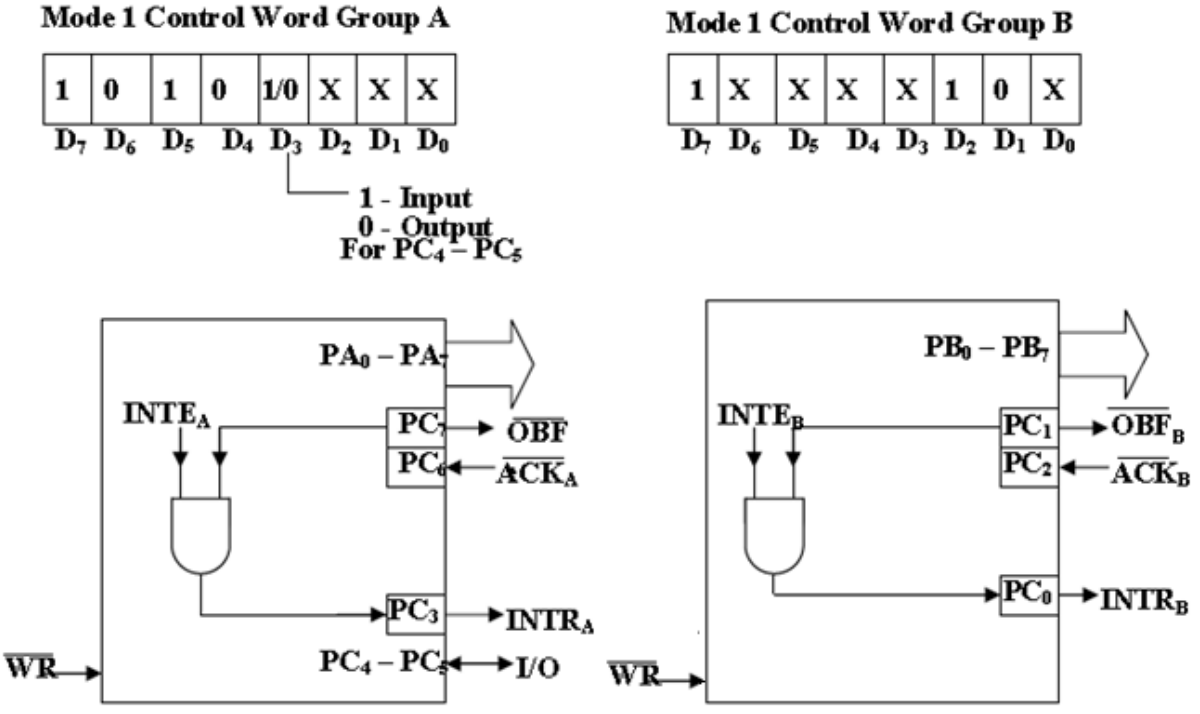
• **OBF** (Output buffer full) – This status signal, whenever falls to low, indicates that CPU has written data to the specified output port. The OBF flip-flop will be set by a rising edge of WR signal and reset by a low going edge at the ACK input.

• ACK (Acknowledge input) – ACK signal acts as an acknowledgement to be given by an output device. ACK signal, whenever low, informs the CPU that the data transferred by the CPU to the output device through the port is received by the output device.

• **INTR** (Interrupt request) – Thus an output signal that can be used to interrupt the CPU when an output device acknowledges the data received from the CPU. INTR is set when ACK, OBF and INTE are 1. It is reset by a falling edge on WR input. The INTEA and INTEB flags are controlled by the bit set-reset mode of PC6 and PC2 respectively.



Input control signal definitions in Mode 1



Output control signal definitions Mode 1

c) Mode 2 ***(Strobed bidirectional I/O):*** This mode of operation of 8255 is also called as strobed bidirectional I/O. This mode of operation provides 8255 with an additional features for communicating with a peripheral device on an 8-bit data bus. Handshaking signals are provided to maintain proper data flow and synchronization between the data transmitter and receiver. The interrupt generation and other functions are similar to mode 1. In this mode, 8255 is a bidirectional 8-bit port with handshake signals. The Rd and WR signals decide whether the 8255 is going to operate as an input port or output port.

The Salient features of Mode 2 of 8255 are listed as follows:

1. The single 8-bit port in group A is available.
2. The 8-bit port is bidirectional and additionally a 5-bit control port is available.
3. Three I/O lines are available at port C (PC2 – PC0).
4. Inputs and outputs are both latched.
5. The 5-bit control port C (PC3-PC7) is used for generating / accepting handshake signals for the 8-bit data transfer on port A.

Control signal definitions in mode 2:

• **INTR** – (Interrupt request) as in mode 1, this control signal is active high and is used to interrupt the microprocessor to ask for transfer of the next data byte to/from it. This signal is used for input (read) as well as output (write) operations.

-Control Signals for Output operations:

• OBF (Output buffer full): this signal, when falls to low level, indicates that the CPU has written data to port A.

• ACK (Acknowledge): this control input, when falls to logic low level, acknowledges that the previous data byte is received by the destination and next byte may be sent by the processor. This signal enables the internal tristate buffers to send the next data byte on port A.

• **INTE1** (A flag associated with OBF): this can be controlled by bit set/reset mode with PC6.

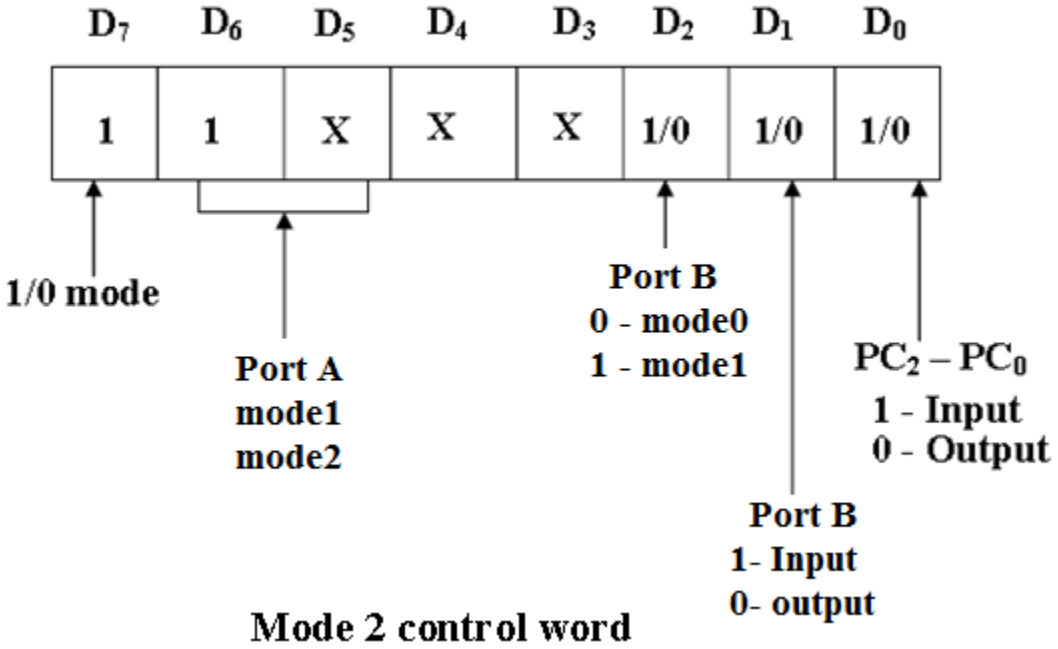
-Control signals for input operations:

• STB (Strobe input): a low on this line is used to strobe in the data into the input latches of 8255.

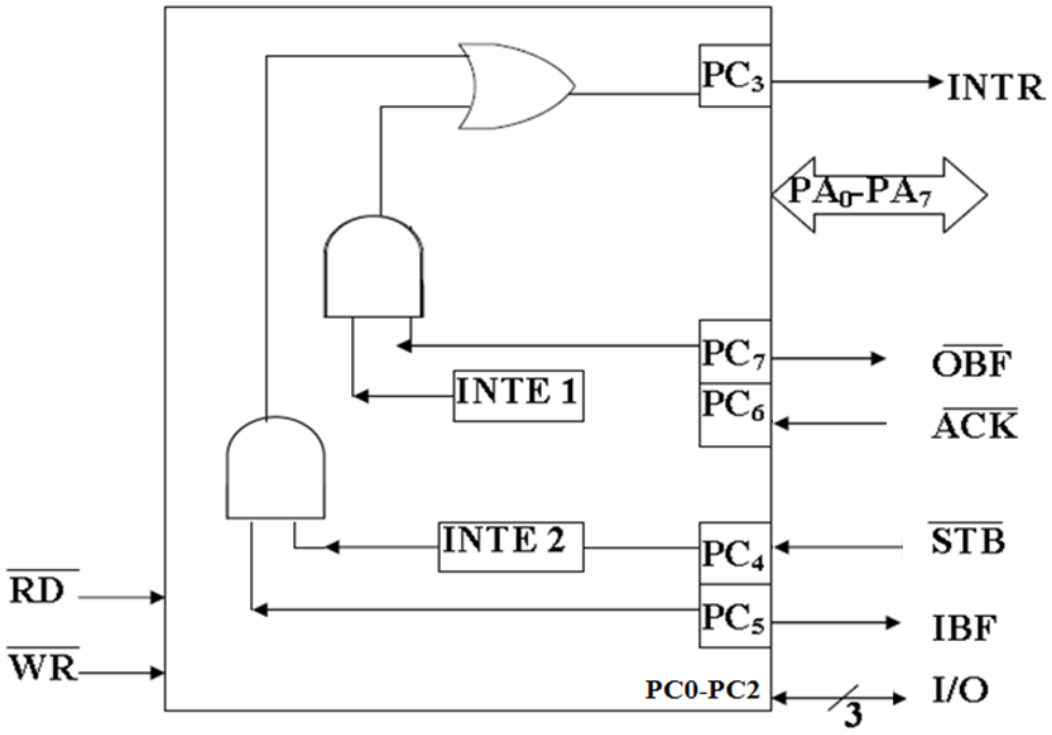
• **IBF** (Input buffer full): when the data is loaded into input buffer, this signal rises to logic '1'. This can be used as an acknowledge that the data has been received by the receiver.

Mode 2 Bidirectional Data Transfer:

The following figure shows a schematic diagram containing an 8-bit bidirectional port, 5-bit control port and the relation of INTR with the control pins. Port B can either be set to Mode 0 or 1 with port A (Group A) is in Mode 2. Mode 2 is not available for port B. The following figure shows the control word. The INTR goes high only if IBF, INTE2, STB and RD go high or OBF, INTE1, ACK and WR go high. The port C can be read to know the status of the peripheral device, in terms of the control signals, using the normal I/O instructions.



Mode 2 control word



Mode 2 pins

The Table below shows a summary of the port connections for the 82C55A PPI.

		Mode 0		Mode 1		Mode 2
Port A		IN	OUT	IN	OUT	I/O
Port B		IN	OUT	IN	OUT	Not used
Port C	0	IN	OUT	$INTR_B$	$INTR_B$	I/O
	1			IBF_B	$\overline{OBF}_B$	I/O
	2			$\overline{STB}_B$	$\overline{ACK}_B$	I/O
	3			$INTR_A$	$INTR_A$	INTR
	4			$\overline{STB}_A$	I/O	$\overline{STB}$
	5			IBF_A	I/O	IBF
	6			I/O	$\overline{ACK}_A$	$\overline{ACK}$
	7			I/O	$\overline{OBF}_A$	$\overline{OBF}$

82C55A Implementation of Parallel Input/Output Ports:

The circuit in **figures below** shows how PPI devices can be connected to the bus of 8086 to implement parallel input/output ports. This circuit configuration is for a minimum mode 8086 microcomputer. Here we find two groups of eight 82C55A devices one connected to the lower eight data bus lines, and the other to the upper eight data bus line. Each of these groups is capable of implementing up to 192 I/O lines to give a total I/O capability of 384 I/O lines data bus. Each of the groups of 82C55As has its own 74F138 I/O address decoder. The address decoder is used to select devices in a group one at a time. The ports in the upper group are connected at odd-address boundaries and those in the lower group are at even-address boundaries.

Example1: what is the mode and I/O configuration for ports A, B and C of an 82C55A after its control register is loaded with 82 H.

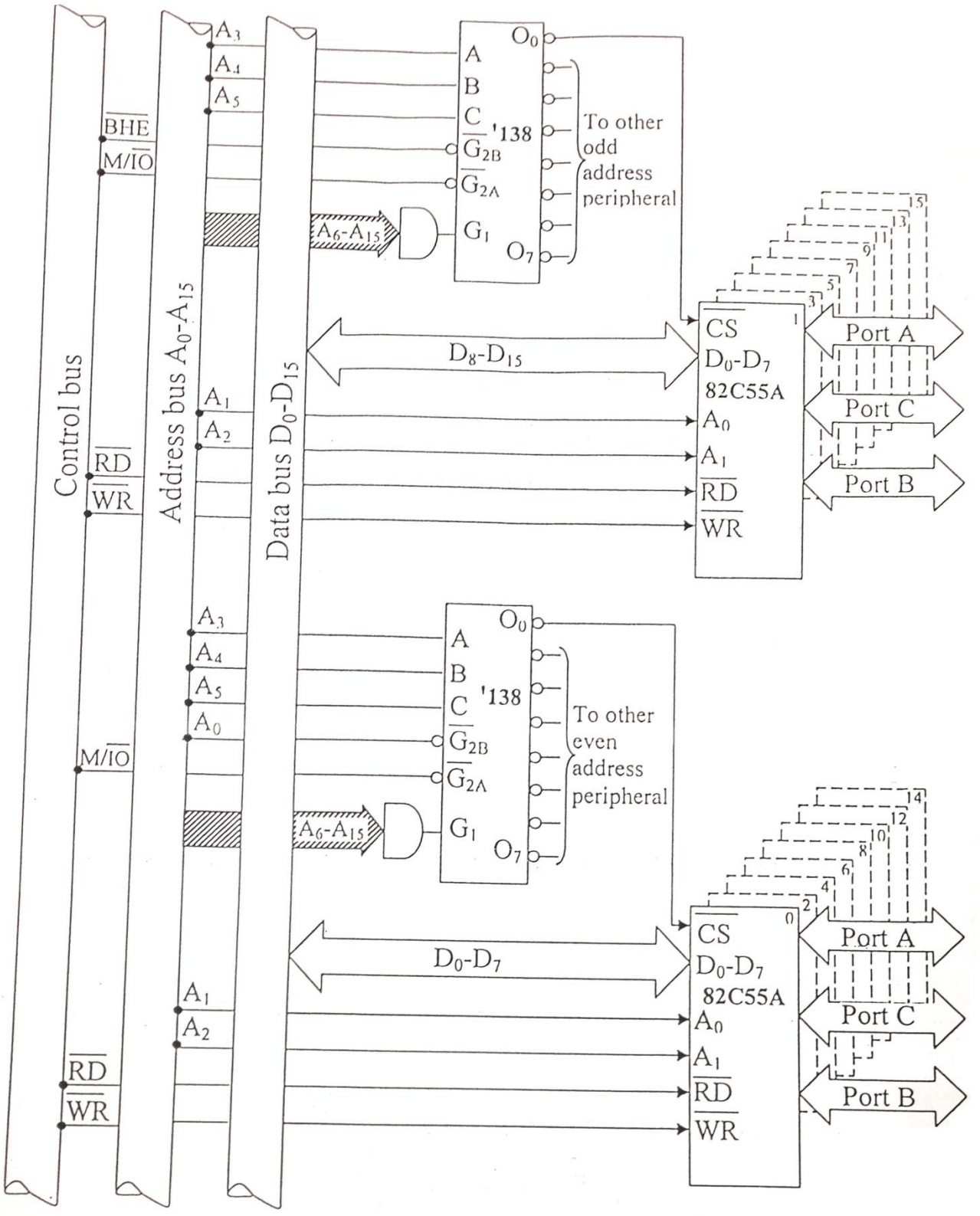
Example2: In the isolated parallel input/output ports of 82C55A implementation, assume that PPI 15 is configured so that port A is an output port, both port B and C are input ports, and all three ports are set up for mode 0 operation. Write a program that will input the data at port B and C, find the difference (port C)-(port B), and output this difference to port A.

Example3: In memory mapped parallel input/output ports of 82C55A implementation, assume that PPI0 is configured so that port A is an output port, both port B and C are input ports, and all three ports are set up for mode 0 operation. Write a program to perform the following:

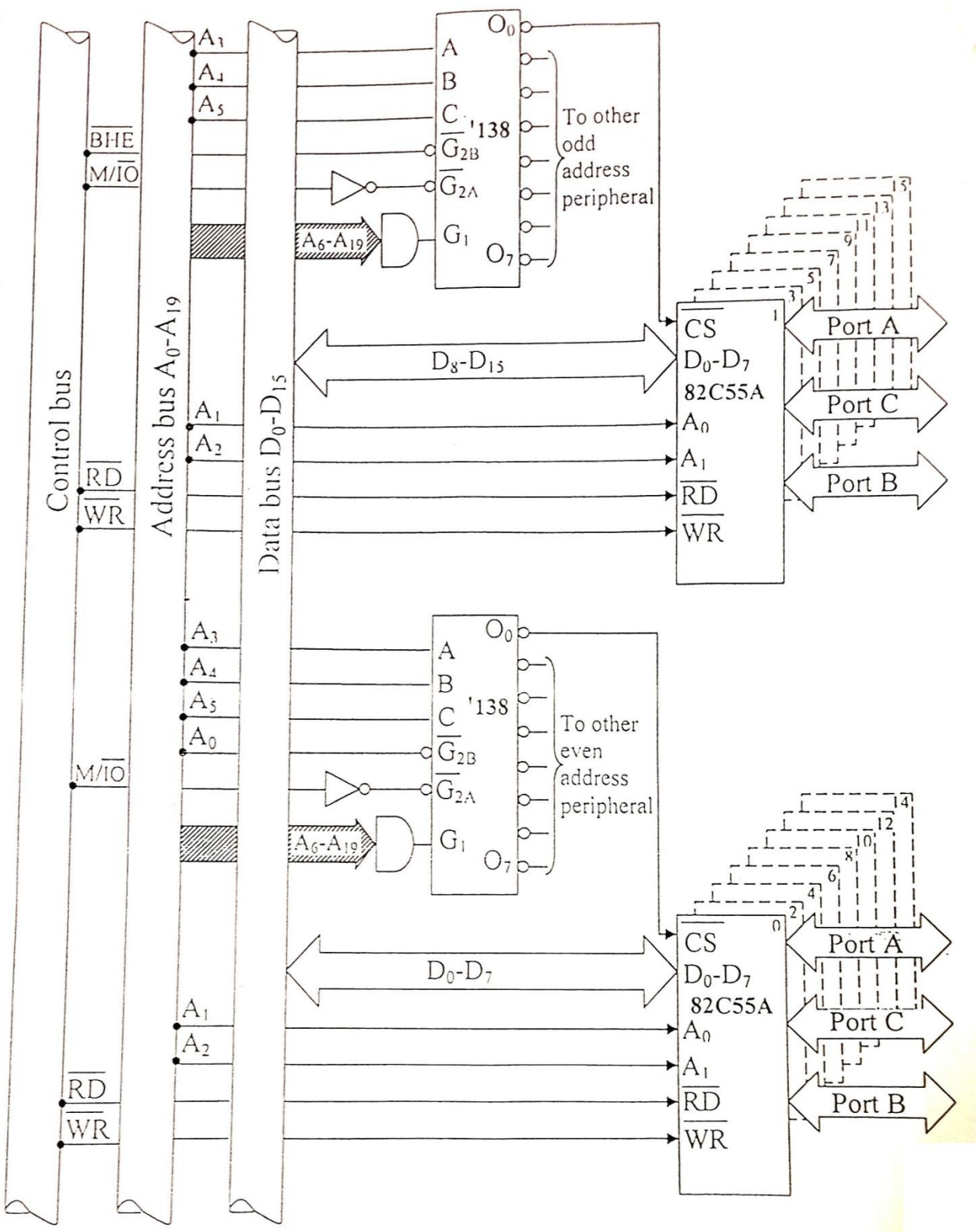
1- initialize the control register of PPI0.

2- input the contents of port B and C, AND them together, and output the result to port A.

Example4: Write a BSR program to set bits PC7 and PC3 and reset them after (10ms). Use PPI0 of parallel input/output ports of 82C55A implementation, and assume that a delay subroutine is available.



Isolated 82C55A I/O ports at even- and odd- address boundaries in an 8086 MP.



Memory mapped 82C55A parallel I/O port at even- and odd-address boundaries in an 8086-base MP.

Ex1: The interrupt control flag INTEA is controlled by bit set/reset of PC6. what is command code must be written to the control register of the 82C55A to set its value to logic 1?

D7	D6	D5	D4	D3	D2	D1	D0		
0	x	x	x	1	1	0	1		Let x = 1
0	1	1	1	1	1	0	1	=	7D H

Ex2: what control word must be written into the control register of the 82C55A so that port A is configured for bidirectional operation and port B is setup with mode1 output?

D7 =1

Port A port A is configured for bidirectional operation: D6=1

D5 D4 D3 = x

Port B is setup with mode1 output:

D2= 1

D1= 0

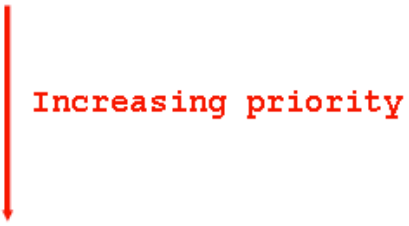
D0= x

D7	D6	D5	D4	D3	D2	D1	D0		
1	1	x	x	x	1	0	x		Let x = 0
1	1	0	0	0	1	0	0	=	C4 H

Interrupt Mechanism, Types, and Priority

- A mechanism for quickly changing program environment
  - Transfer of program control is initiated by the occurrence of either an internal event or an event in its external hardware.
- Interrupt-service routine
  - A section of program to which control is passed when an interrupt signal occurs

- The 8086 MPU is capable of implementing any combination of up to 256 (0-0FFH) interrupts.
  - External hardware interrupts
  - Nonmaskable interrupt
  - Software interrupts
  - Internal interrupts
  - Reset



- Interrupt priority
  - Each of interrupts is given a different priority level by assigning it a type number.
  - Type 0 identifies the highest-priority
  - Type 255(0FFH) identifies the lowest
  - A few of the type numbers are reserved/dedicated for special interrupt functions. (e.g., divide error)
  - Only devices with higher priority are allowed to interrupt the active service routine.
  - For hardware interrupts, the priority scheme is implemented in external hardware.

Interrupts

The processor has the following interrupts:

•**INTR** is a maskable hardware interrupt. The interrupt can be enabled/disabled using STI/CLI instructions or using more complicated method of updating the FLAGS register with the help of the POPF instruction.

•When an interrupt occurs, the processor stores FLAGS register into stack, disables further interrupts, fetches from the bus one byte representing interrupt type, and jumps to interrupt processing routine address of which is stored in location 4 * <interrupt type>. Interrupt processing routine should return with the IRET instruction.

•**NMI** is a non-maskable interrupt. Interrupt is processed in the same way as the INTR interrupt. Interrupt type of the NMI is 2, i.e. the address of the NMI processing routine is stored in location 0008h. This interrupt has higher priority than the maskable interrupt.

•**Software interrupts** can be caused by:

•INT instruction - breakpoint interrupt. This is a type 3 interrupt.

•INT <interrupt number> instruction - any one interrupt from available 256 interrupts.

•INTO instruction - interrupt on overflow

•Single-step interrupt - generated if the TF flag is set. This is a type 1 interrupt. When the CPU processes this interrupt it clears TF flag before calling the interrupt processing routine.

•**Processor exceptions**: Divide Error (Type 0), Unused Opcode (type 6) and Escape opcode (type 7).

•Software interrupt processing is the same as for the hardware interrupts.

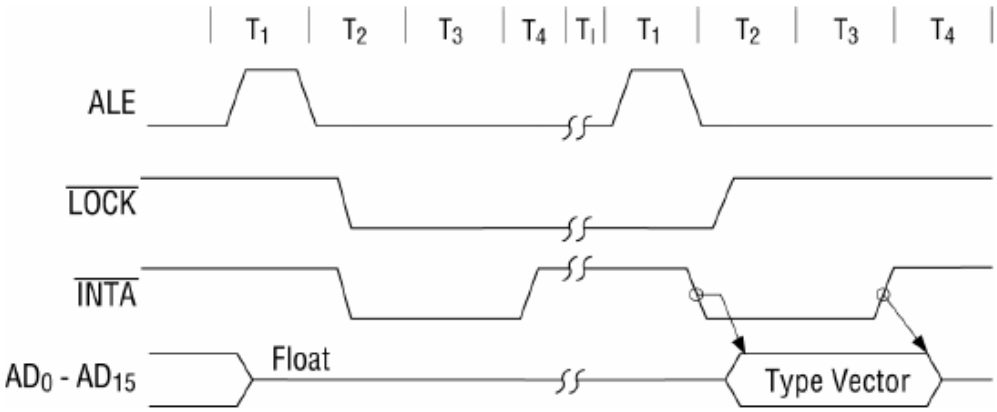
Interrupt Instructions

Format	Meaning	Operation	Flag Affected
CLI	Clear interrupt flag	0 → (IF)	IF
STI	Set interrupt flag	1 → (IF)	IF
INT n	Type n S/W interrupt	(Flags) → ((SP)-2) 0 → TF,IF (CS) → ((SP)-4) (2+4·n) →(CS) (IP) →((SP-6) (4·n) →(IP)	TF,IF
IRET	Interrupt Return	((SP)) →(IP) ((SP+2) →(CS) ((SP)+4) →(Flags) (SP)+6 →(SP)	All
INTO	Interrupt on Overflow	INT 4 steps	TF, IF
HLT	Halt	Wait for an external interrupt or reset to occur	None
WAIT	Wait	Wait for TEST/ input to go active	None

Enabling/Disabling of Interrupts

- IF (interrupt-enable flag) bit is provided within the CPU.
  - It affects only the external hardware interrupt interface.
  - STI, CLI
- During the initiation sequence of a service routine for an external hardware interrupt, the MPU automatically clears IF.
  - This masks out the occurrence of any additional external hardware interrupts.
  - In some applications, it may be necessary to permit other interrupts to interrupt the active service routine. In the case, an STI instruction in the service routine to reenable the INTR input. Otherwise, the external H/W-interrupt interface is reenabled by the IRET instruction at the end of the service routine.

- Interrupt acknowledge bus cycle

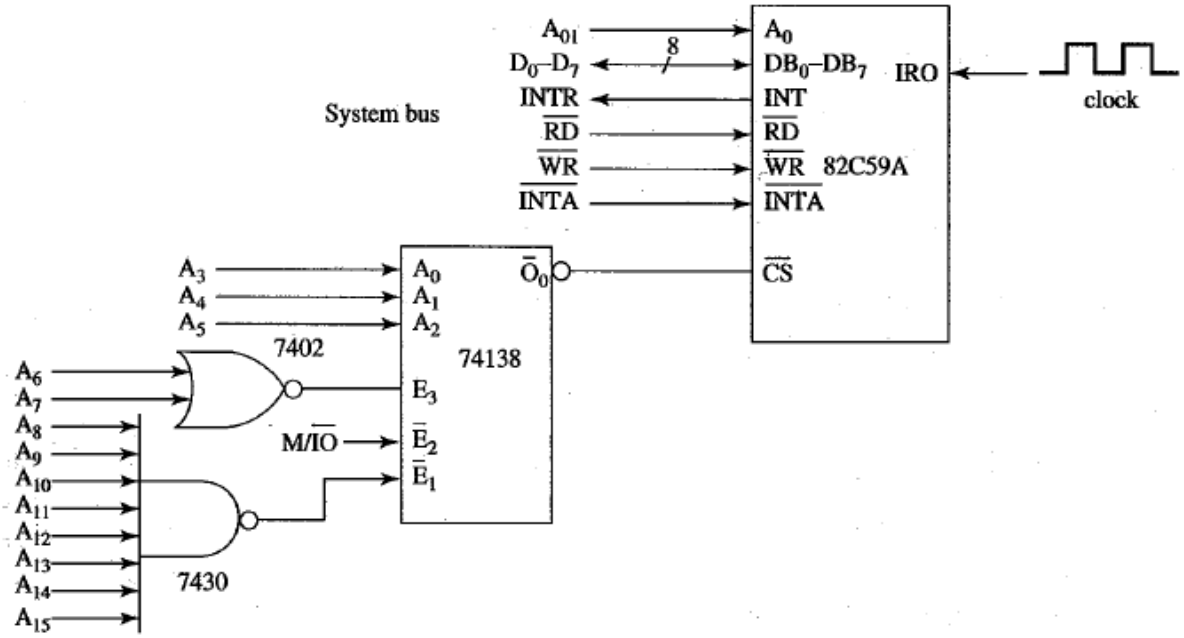


82C59A Programmable Interrupt Controller

- PIC is an LSI designed to simplify the implementation of the interrupt interface in 8086-based systems.
- The operation of the PIC is programmable under software control.
  - Level-sensitive or edge-triggered inputs
  - Easy expansion from 8 to 64 interrupt inputs by cascading
  - Wide variety of priority schemes.

Interrupt Interface Circuits Using the PIC

- Example



Software interrupts

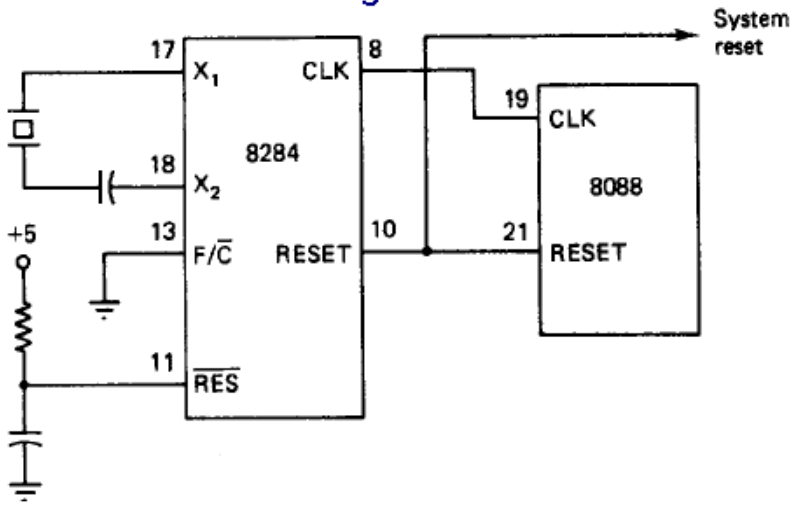
- The software interrupts differ from the external hardware interrupts in that their service routines are initiated in response to the execution of a software interrupt instruction.
- *INT n* instruction is used to initiate a software interrupt, where *n* is the type number.
- The mechanism by which a software interrupt is initiated is similar to that described for the external hardware interrupts.
  - No external interrupt-acknowledge bus cycles are initiated.
  - Control is passed to the start of the service routine immediately upon completion of execution of the interrupt instruction.
- Software interrupts are of higher priority than the external interrupts and are not masked out by IF.

Nonmaskable Interrupt

- NMI is an interrupt initiated from external hardware.
  - It cannot be masked out with the interrupt flags.
  - The NMI input is used, not the INTR input.
  - Positive edge-triggered.
- NMI has a dedicated type number – type 2
- Typically, the NMI is assigned to hardware events that must be responded to immediately.
  - The detection of a power failure.
  - The detection of a memory read error.

RESET

- A hardware means for initializing the MPU.
- Reset is typically done at power-up to provide an orderly start-up of the system.
  - RESET must be held at logic 1 for a minimum of 4 clock cycles.



RESET

- At completion of initialization, the flags are all cleared;
  - The IP is set to 0000H and the CS is set to FFFFH.
- Program execution begins at address FFFF0H after reset.

Components	Content
Flags	Clear
Instruction pointer	0000H
CS Register	FFFFH
DS Register	0000H
SS Register	0000H
ES Register	0000H
Queue	Empty

Internal Interrupt Functions

- The internal interrupts are not masked out with the IF.

- Divide Error
  - The error function represents an error condition that can occur in the execution of the division instruction.
  - If the results from a DIV is larger than the specified destination an error has occurred.
  - The condition causes automatic initiation of a type 0 interrupt and passes control to a service routine: IP is defined at address 00000H and CS is defined at address 00002H.

Internal Interrupt Functions

- Overflow Error
  - Whenever an overflow occurs, the overflow flag gets set.
  - However, the transfer of program control to a service routine is not automatic. INTO instruction must be executed to test the flag(OF) and determine if the overflow service routine should be initiated.
  - A type 4 interrupt service routine is initiated.

- Single Step
  - If the trap flag bit is set, the single-step mode of operation is enabled.
  - A type 1 interrupt service routine is initiated.
  - The single-step operation can be used as a valuable software debugging tool.

Internal Interrupt Functions

- Breakpoint Interrupt
  - The function can be used to implement a software diagnostic tool.
  - A breakpoint interrupt is initiated by execution of the breakpoint instruction(code = CCH). The instruction can be inserted at strategic points in a program that is being debugged to cause execution to be stopped automatically.
  - The service routine can stop execution of the main program, permit the programmer to examine the contents of registers and memory, and allow for the resumption of execution of the program down to the next breakpoint.