The 8086 Input/output Interface

This lecture describes the IO interface circuits of an 8086-based microcomputer system. 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.

- 8088/8086 architecture implements independent memory and input/output address spaces
- Memory address space- 1,048,576 bytes long (1M-byte)—00000H-FFFFFH
- Input/output address space- 65,536 bytes long (64K-bytes)—0000H-FFFFH
- Input/output can be implemented in either the memory or I/O address space
- Each input/output address is called a port
- The 8086 microcomputers can employ two different types of input/output (I/O):
    - Isolated I/O.
    - Memory-mapped I/O.

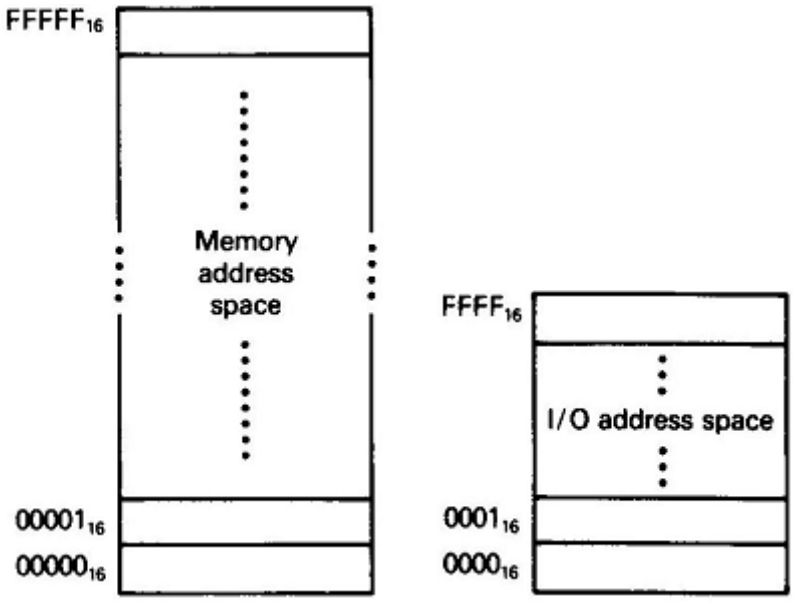

Fig. (1): 8086 memory and I/O Interface

Isolated Input/output

- Using isolated I/O a microcomputer system, the I/O devices are treated **separate from** memory.
- The part of the I/O address space from address 0000H through 00FFH is referred to as **Page 0** as shown in figure (2).
- Supports byte and word I/O ports
- 64K independent byte-wide I/O ports
- 32K independent aligned word-wide I/O ports

- **Advantages of isolated I/O**
    1. Complete memory address space available for use by memory
    2. Special instructions have been provided in the instruction set of the 8086 to perform isolated I/O operation. This instructions tailored to maximize performance

- **Disadvantage of Isolated I/O**

    All inputs/outputs must take place between an I/O port and accumulator (AL or AX) register

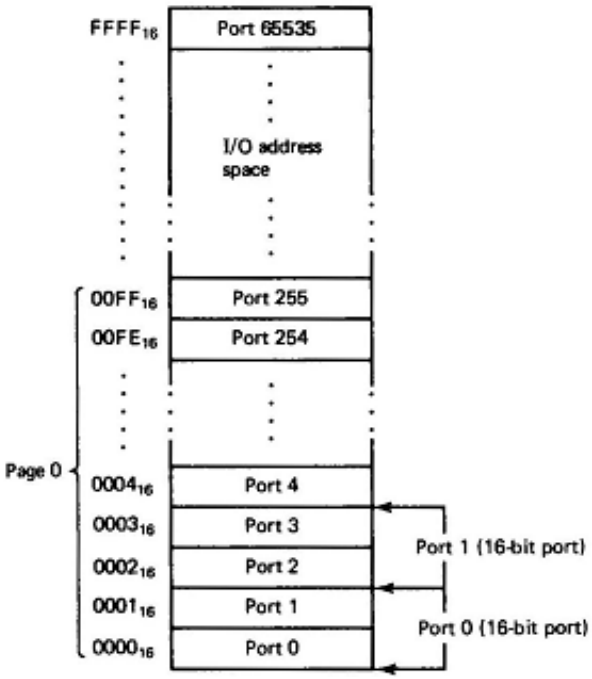

Fig. (2): Isolated I/O port.

Memory-mapped Input/output

- Memory mapped I/O—a part of the memory address space is dedicated to I/O devices
- For Example: (E0000H-E0FFFH) $\rightarrow$ 4096 memory addresses assigned to I/O ports
- E0000H, E0001H, and E0002H correspond to byte wide ports 0,1, and 2
- E0000H and E0001H correspond to word-wide port 0 at address E0000H
- **Advantages of memory mapped I/O**
    1. Instructions that affect data in memory (MOV, ADD, AND, etc.) can be used to perform I/O operations
    2. I/O transfers can take place between I/O port and any of the registers
- **Disadvantage of memory mapped I/O**
    1. Memory instructions perform slower
    2. Part of the memory address space cannot be used to implement memory

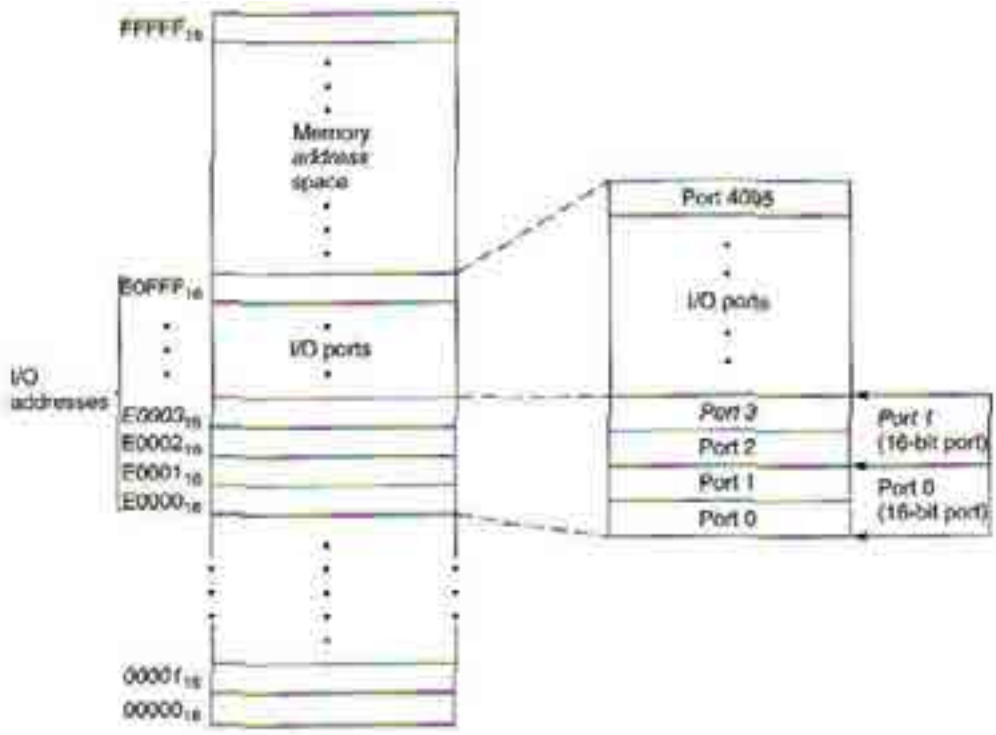

Fig. (3): Memory Mapped I/O port.

Differences between Isolated I/O and Memory Mapped I/O:

Isolated I/O	No.	Memory Mapped I/O
Isolated I/O uses separate memory space.	**01**	Memory mapped I/O uses memory from the main memory.
Limited instructions can be used. Those are IN, OUT, INS, OUTS.	**02**	Any instruction which references to memory can be used. (MOV, AND XCHG, SUB …….)
Faster because I/O instructions is specifically designed to run faster than memory instructions	**03**	Slower because memory instructions execute slower than the special I/O instructions
The memory address space is not affected	**04**	Part of the memory address space is lost
The addresses for Isolated I/O devices are called ports.	**05**	Memory mapped I/O devices are treated as memory locations on the memory map

Minimum Mode Interface

- Similar in structure and operation to memory interface
- I/O devices—can represent LEDs, switches, keyboard, serial communication port, printer port, etc.
- I/O data transfers take place between I/O devices and MPU over the multiplexed-address data bus AD0-AD7, A8-A15
- This interface use the control signals review
    - ALE = pulse to logic 1 tells bus interface circuitry to latch I/O address
    - $\overline{\mathbf{RD}}$ = logic 0 tells the I/O interface circuitry that an input (read) is in progress
    - $\overline{\mathbf{WR}}$ = logic 0 tells the I/O interface circuitry that an output (write) is in progress
    - $\mathbf{M/\overline{IO}}$ = logic 0 tells I/O interface circuits that the data transfer operation is for the IO subsystem
    - $\mathbf{DT/\overline{R}}$ = sets the direction of the data bus for input (read) or output (write) operation
    - $\overline{\mathbf{DEN}}$ = enables the interface between the I/O subsystem and MPU data bus

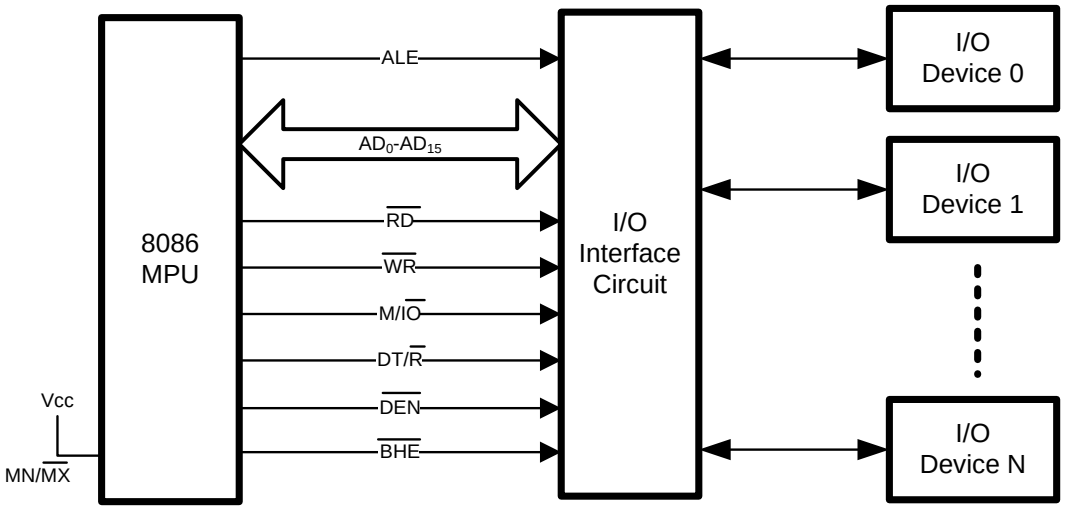

Fig. (4): Minimum mode 8086 system I/O Interface.

Maximum Mode Interface

- Maximum-mode interface differences review
- 8288 bus controller produces the control signals for I/O subsystem:
- Signal changes
    - $\overline{\mathbf{IORC}}$ replaces $\overline{\mathbf{RD}}$
    - $\overline{\mathbf{IOWC}}$ and $\overline{\mathbf{AIOWC}}$ replace $\overline{\mathbf{WR}}$
    - **DEN** is complement of $\overline{\mathbf{DEN}}$
    - $\mathbf{M/\overline{IO}}$ no longer needed (bus controller creates separate IO read/write controls)

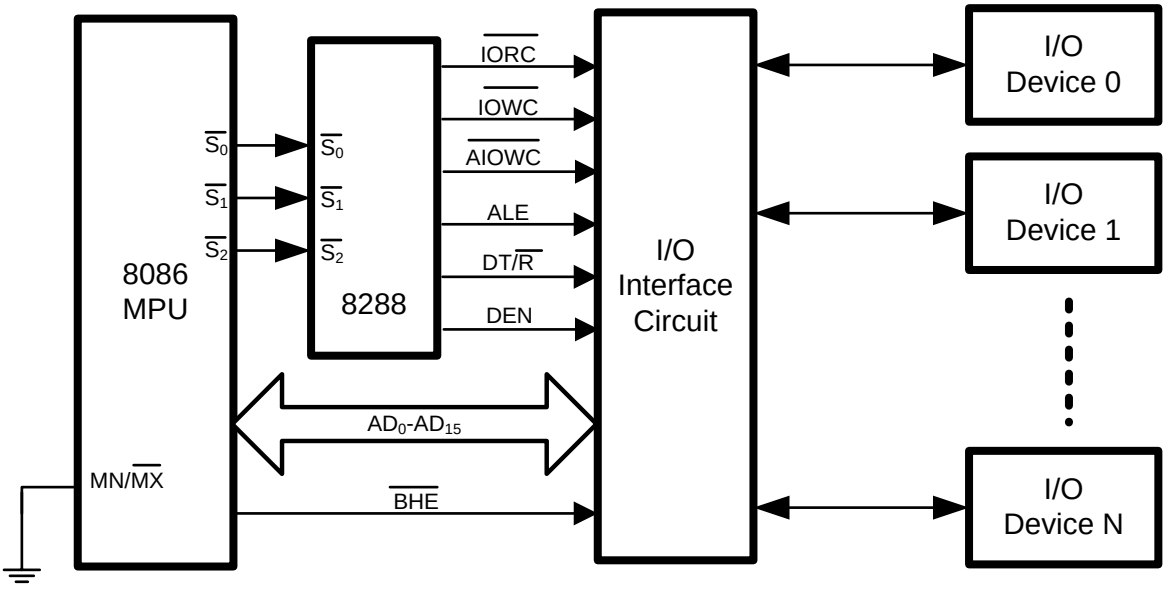

Fig. (5): Maximum mode 8086 system I/O Interface.

IN and OUT Instruction

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 a word of data. All data transfers take place between an I/O device and the MPU's accumulator register. The general form of this instruction is as shown below:

Mnemonic	Meaning	Format	Operation	Flags Effected
IN	**Input direct** **Input variable**	**IN Acc, Port** **IN Acc, DX**	**(Acc) ← (Port)** **(Acc) ← (DX)**	**none**
OUT	**Output direct** **Output variable**	**OUT Port, Acc** **OUT DX , Acc**	**(Port) ← (Acc)** **(DX) ← (Acc)**	**none**

Example:

```
IN AL,0C8H          ;Input a byte from port 0C8H to AL
IN AX, 34H          ;Input a word (two byte) from port 34H, 35H to AX
OUT 3BH, AL         ;Copy the contents of the AL to port 3Bh
OUT 2CH,AX          ;Copy the contents of the AX to port 2CH, 2DH
```

For a variable port IN instruction, the port address is loaded in DX register before IN instruction. DX is 16 bit. Port address range from 0000H – FFFFH.

Example: (a)

```
MOV DX, 0FF78H      ;Initialize DX point to port
IN AL, DX           ;Input a byte from a 8 bit port 0FF78H to AL
IN AX, DX           ;Input a word from 16 bit port to 0FF78H,0FF79H to AX.
```

Example: write a series of instructions that will output FFH to an output port located at address B000H of the I/O address space.

Solution:

```
MOV DX, B000H
MOV AL, FF
OUT DX, AL
```

Example: Data are to be read from two byte-wide input ports at addresses **AAH** and **A9**H and then output as a word to a word-wide output port at address **B000H.** Write a series of instructions to perform this input/output operation.

Solution:

```
IN  AL, AAH
MOV AH, AL
IN  AL, A9H
MOV DX, B000H
OUT DX, AX
```

Input/Output Bus Cycles

The input/output bus cycles are essentially the same as those involved in the memory interface. Figure show the output bus cycle of the 8086. It's similar to the write cycle except for the signal $\mathbf{M}/\overline{\mathbf{IO}}$.

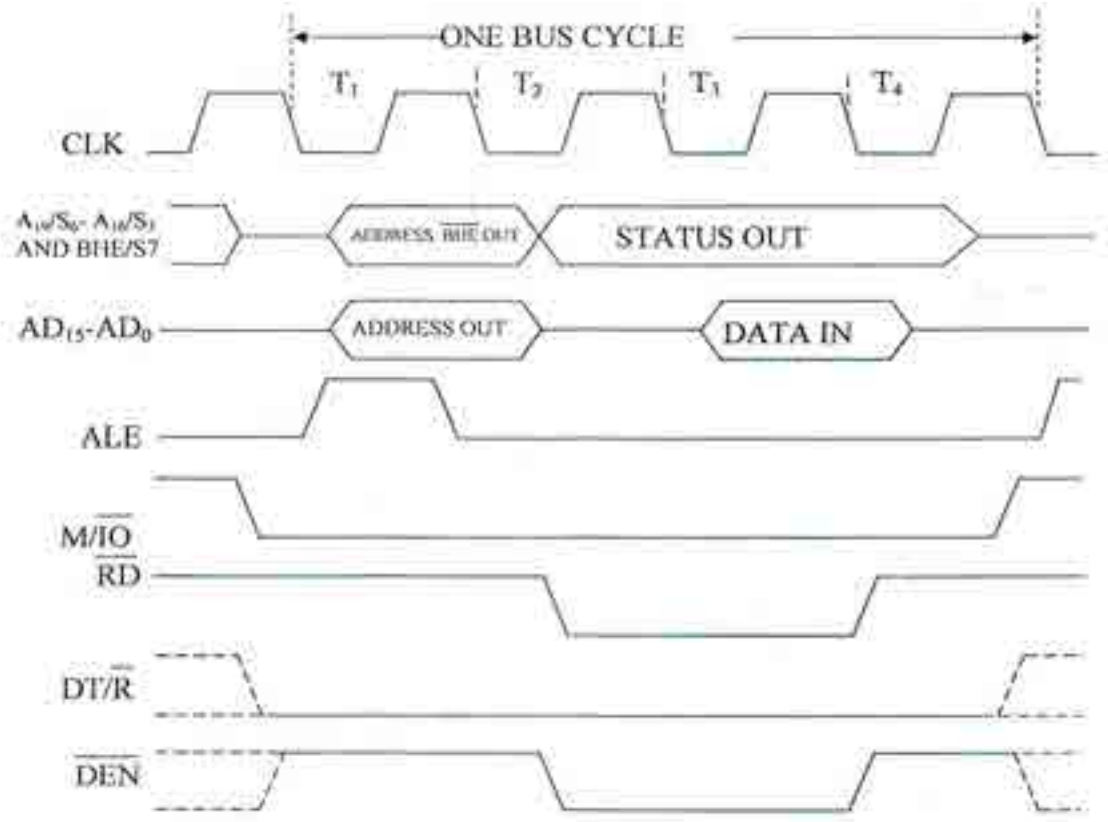

Fig. (6): Input bus cycle of 8086.

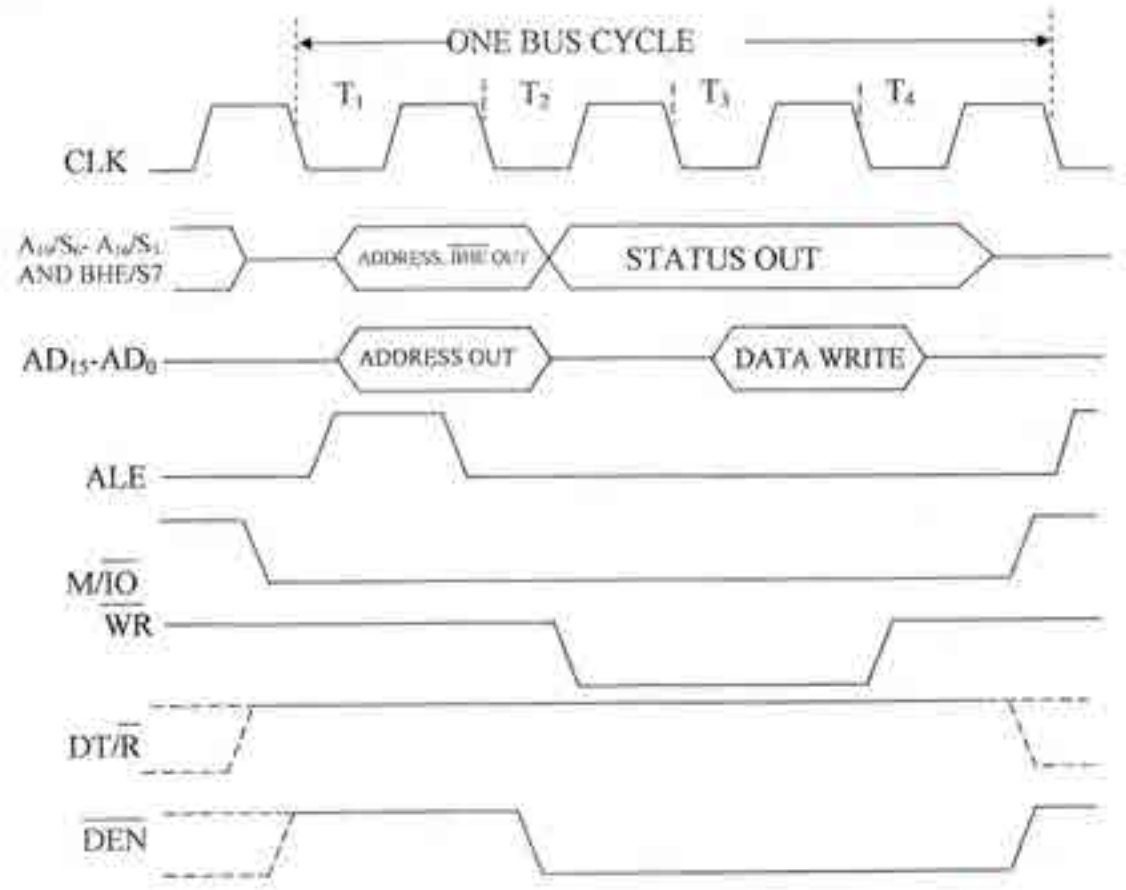

Fig. (7): Output bus cycle of 8086.

Byte-Wide Input and Output Ports using Isolated I/O

Figure (8) shows a circuits diagram of a byte-wide input and output ports (8 bit) using isolated I/O for an 8086 based microcomputer system. From this diagram there is four parts:

Demultiplexing circuit:

- Two 74F373 octal latches are used to form a 16-bit address latch. These devices latch the address A_0 through A_{15} synchronously with the ALE pulse. The latched address outputs are labeled A_{oL} through A_{15L}.
- Remember that address lines A_{16} through A_{19} are not involved in the I/O interface.
- Data bus transceiver buffer in 8086 system is implemented using 74F245 octal bus IC's, where the control inputs '**DIR**' and '$\overline{\mathbf{G}}$' is used to control the data flow (An $\rightarrow$ Bn) or (Bn $\rightarrow$An).
- Figure (9) shows the block and circuit diagram of the 8-bit Data bus transceiver buffer IC. Also note that $\overline{\mathbf{G}}$ input is used to enable the buffer operation, whereas **DIR** input selects the direction of intended data transfer

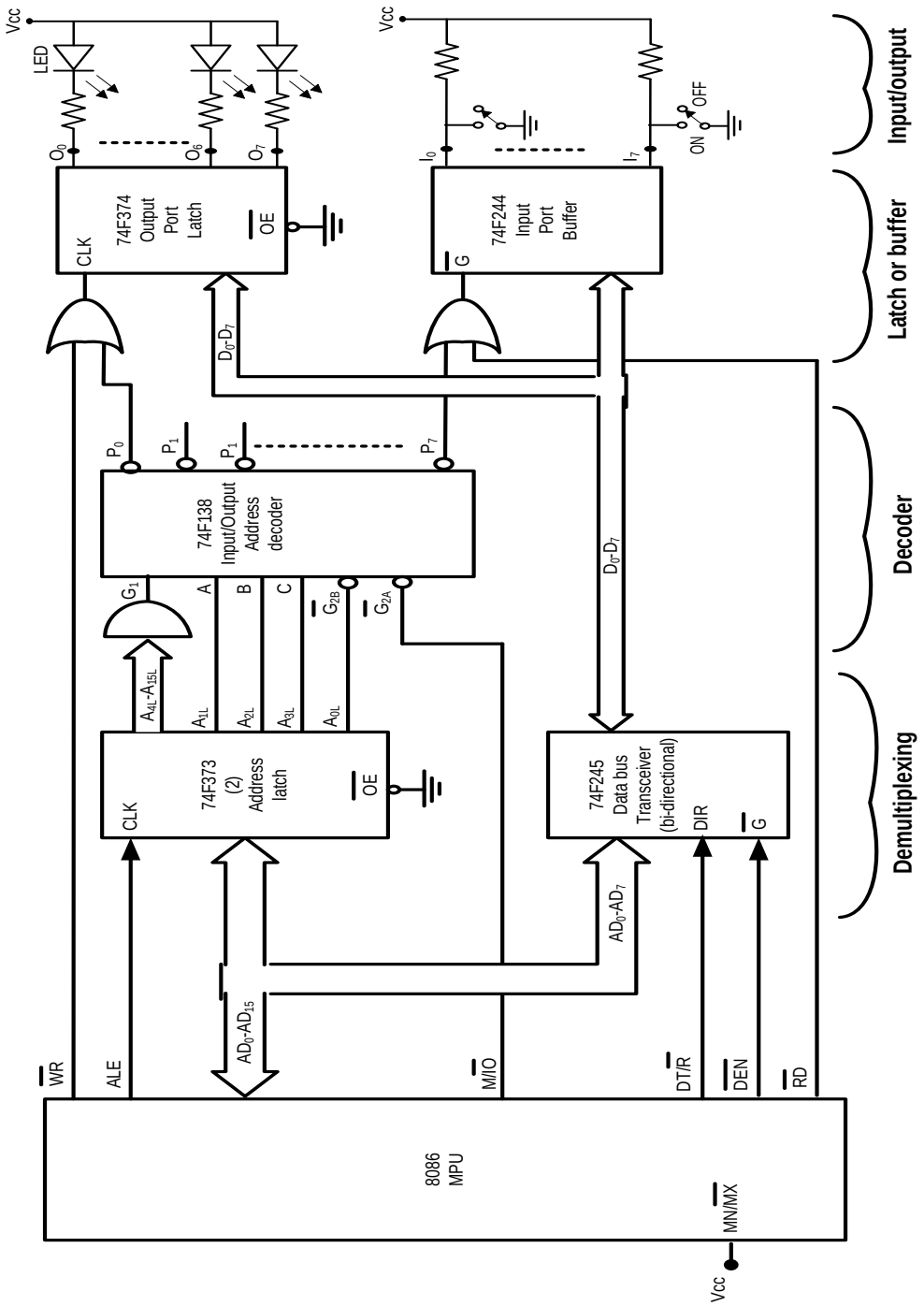

Fig. (8): A byte-wide input and output ports using Isolated I/P for an 8086 based.

- Assume that the device is enabled by applying $\overline{\mathbf{G}}$ = 0. Now if **DIR** is set to logic 0, the output of AND gate 1 will be 0 and all the odd numbered buffers (G3, G5, G7 and so on) will be off. So the data path from An to Bn will be disabled. But the output of AND gate 2 will be logic 1 and all the even numbered buffers (G4, G6, G8 and so on) will be ON. Consequently, the data path from Bn to An will be ENABLED.
- Similarly, for $\overline{\mathbf{G}}$ = 0 and **DIR** = logic 1, data path from An to Bn will be ENABLED.

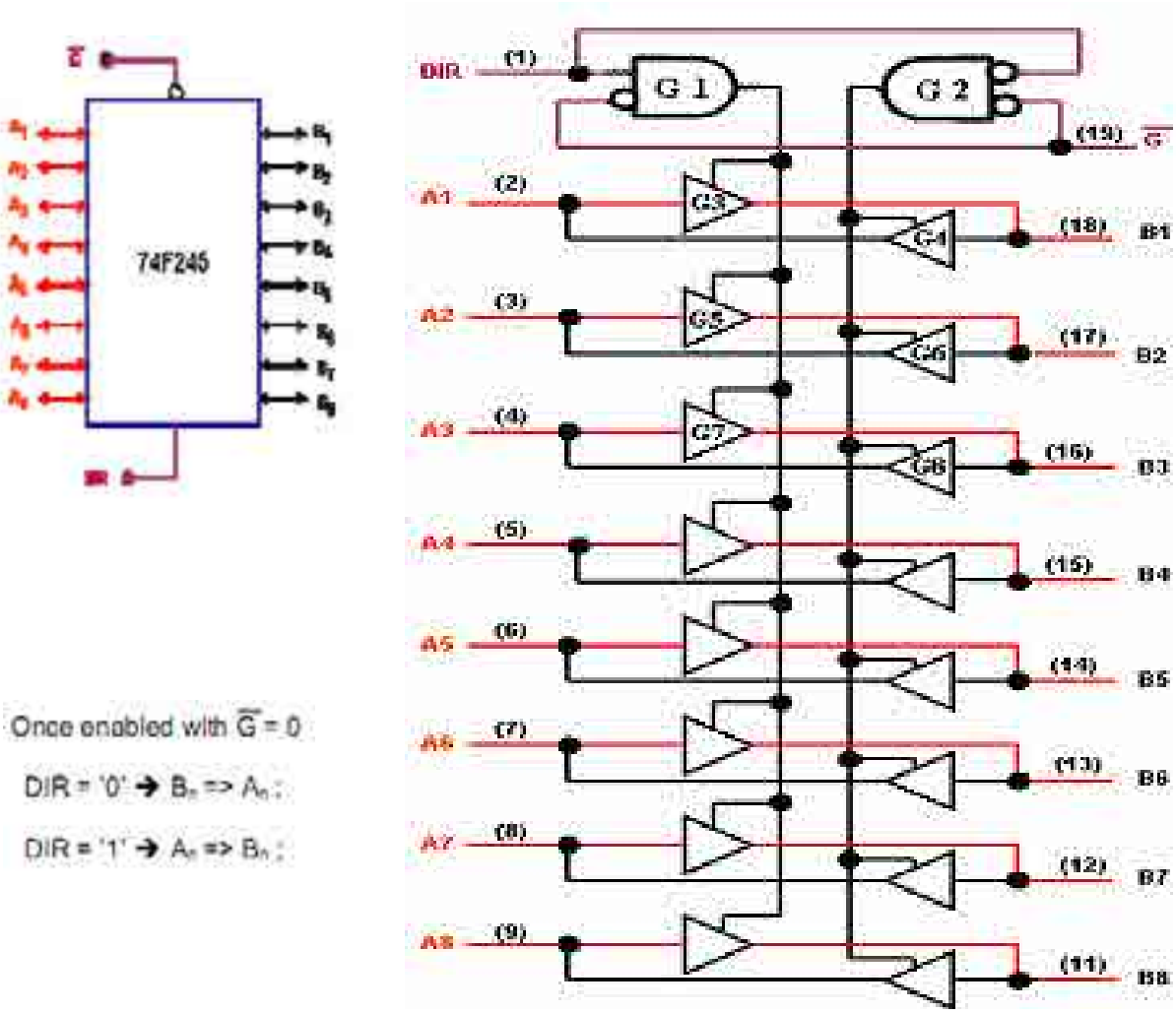

Fig. (9): The block and circuit diagram of the 8-bit Data bus transceiver buffer IC.

Decoder circuit:

- A 74F138 (3 Line-to- 8 Line decoder) is used for decoder circuit. Address lines A_{4L}-A_{15L} are applied to AND gate, output of AND gate and address line A_{0L} provide two of the three enable inputs of the 74F138 input/output address decoder. These signals are applied to enable input $\mathbf{G_1}$ and $\overline{\mathbf{G_{2B}}}$ respectively.

- The decoder requires one more enable signal at its $\overline{G_{2A}}$ input, which is supplied by the complement of $M/\overline{IO}$.
- The enable inputs must be $\overline{G_{2B}}$ $\overline{G_{2A}}$ G_1 =001 to enable the decoder for operation.
- The condition $\overline{G_{2B}}$ =0 corresponds to an even address, and $\overline{G_{2A}}$ =0 represent the fact that an I/O bus cycle is in progress. G_1 =1 is achieved if the output of AND gate is logic 1.
- The three address lines $A_{3L}A_{2L}A_{1L}$ are applied to select inputs **CBA** of the 74F138 decoder.
- When the decoder is enabled, the P output corresponding to these select inputs switches to logic 0.

The Latch

- For output circuit we need to store the output data so we use latch, for this purpose 74F374 device is selected.
- For Figure (8), the gate at the **CLK** input of 74F374 has its inputs P_0 and $\overline{WR}$.
- When valid output data are on the bus, $\overline{WR}$ switches to logic 0.
- Since P_0 is also 0, the **CLK** input of the 74F374 for port 0 switches to logic 0.
- At the end of the $\overline{WR}$ pulse, the clock switches from 0 to 1, a positive transition.
- This causes the data on D_0 –D_7 to be latched and become available at output lines O_0 –O_7 of port 0
- $\overline{OE}$ =0 enable the output , the latched data appears at the appropriate port outputs.

The Buffer

- Buffer is used with input ports. In figure (8) the 74F244 octal buffer is used to implement the port.
- The outputs of the buffer are applied to the data bus for input to the MPU. This buffer has three-state outputs.
- For Figure (8), the gate at the **G** input of 74F244 has its inputs P_7 and $\overline{RD}$.
- When an input bus cycle is in progress, $\overline{RD}$ switches to logic 0.
- Since P_7 is also 0, the **G** input of the 74F244 for port 7 switches to logic 0 and the outputs of 74F244 are enabled.
- In this case, the logic levels at inputs I_0 – I_7 are passed onto data bus lines D_0 – D_7 respectively.
- This byte of data is carried through the enable data bus transceiver to the data bus of the 8086.
- As part of the input operation, the 8086 reads this byte of data into the AL register.

INPUT/OUTPUT Device

- The circuit in figure (8) has 8 LEDs attached to outputs O_0 – O_7 of output port 0. These LEDs represent output device.

- Also, in figure (8) there is 8 switches attached to input $I_0 - I_7$ of input port 7. These switches represent input device.

Example 1: For figure (8), what is the I/O address of

a. Port 0 (P_0)
b. Port 7 (P_7)

Assume all unused address bit are at logic 0.

Solution:

To enable ports $P_0 - P_7 \rightarrow$ 74F138 decoder must be enabled $\rightarrow$ $\overline{G_{2B}}$ $\overline{G_{2A}}$ G_1 =001

$G_1 = 1 \rightarrow$ output of AND gate = $1 \rightarrow A_{4L} - A_{15L}$ = 111111111111

$\overline{G_{2B}} = 0 \rightarrow A_{0L} = 0$

$\overline{G_{2A}} = 0 \rightarrow M/\overline{IO} = 0$

(a) To enable Port 0 (P_0) :

Input of decoder ABC = 000 $\rightarrow A_{1L}\, A_{2L}\, A_{3L}$ = 000

A_{15L}	A_{14L}	A_{13L}	A_{12L}	A_{11L}	A_{10L}	A_{9L}	A_{8L}	A_{7L}	A_{6L}	A_{5L}	A_{4L}	A_{3L}	A_{2L}	A_{1L}	A_{0L}
1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0

I/O address of Port 0 = FFF0H

(b) To enable Port 7 (P_7)

Input of decoder ABC = 111 $\rightarrow A_{1L}\, A_{2L}\, A_{3L}$ = 111

A_{15L}	A_{14L}	A_{13L}	A_{12L}	A_{11L}	A_{10L}	A_{9L}	A_{8L}	A_{7L}	A_{6L}	A_{5L}	A_{4L}	A_{3L}	A_{2L}	A_{1L}	A_{0L}
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

I/O address of Port 0 = FFFEH

Example 2: For the circuit of figure (8), write an instruction sequence that inputs the byte contents of input port 7 to the memory DS:A000H.

Solution: The address of Port P7 =FFFEH. The instruction sequence needed to input the byte is:

```
MOV DX, FFFEH
IN AL, DX
MOV [A000], AL
```

Example 3: For the circuit of figure (8), write an instruction sequence that outputs contents of memory location DS:8000H to output port 0.

Solution: The address of Port P0 =FFF0H. The instruction sequence needed to output the contents of memory location DS:8000H to output port 0 is:

```
MOV DX, FFF0H
MOV  AL , [8000]
OUT DX , AL
```

Time Delay Loop and Blinking an LED at an Output Port

The circuit in Figure 1 show how to attach a LED to output port $\mathbf{O_7}$ of parallel **port 0**. The port address is FFF0H, and the LED corresponds to **bit 7** of the byte of data that is written to port 0. The circuit use 74LS374 (edge clocked octal latch).

For the LED to turn on, $\mathbf{O_7}$ must be switched to logic 0, and it will remain on until this output is switched back to 1. The 74LS374 is not an inverting latch, therefore, to make $\mathbf{O_7}$ logic 0, simply write 0 to that bit of the octal latch.

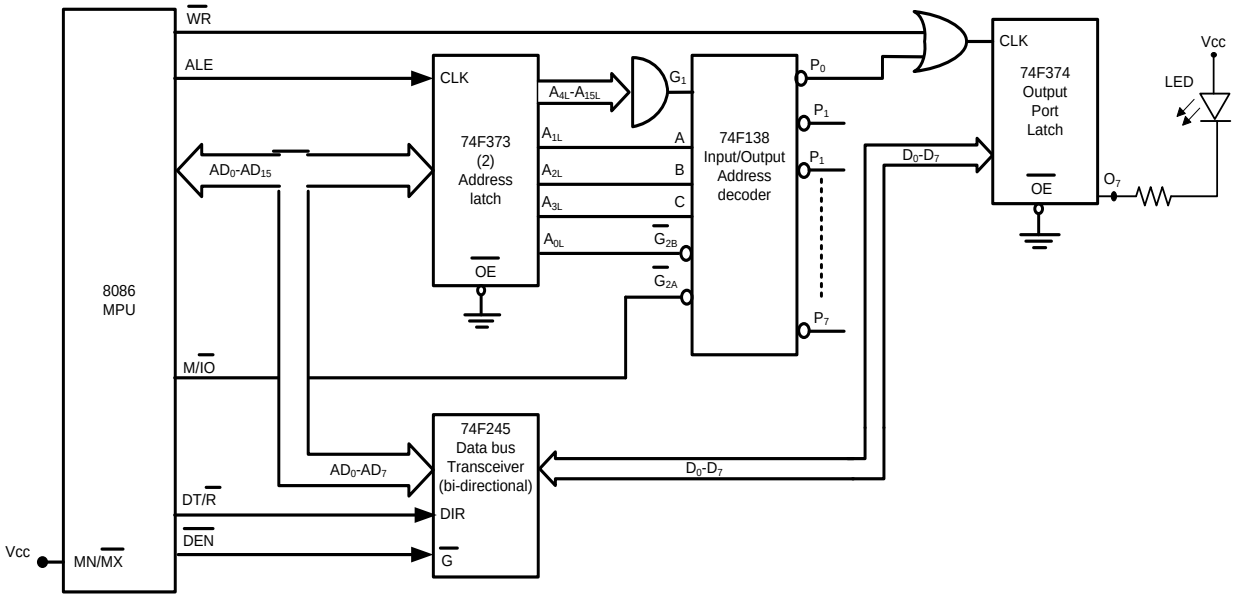

Fig. (1): Time Delay Loop and Blinking an LED at an Output Port

Example 1: Write instruction sequence to make the LED (in Figure 1) blink.
Solution: we must write a program that first makes $\mathbf{O_7}$ logic 0 to turn on the LED, delays for a short period of time, and then switches $\mathbf{O_7}$ back to 1 to turn off the LED. This piece of program can run as a loop to make the LED continuously blink. This is done as follows:

Sequence of instructions needed to initialize0 7 to logic 0.

```
        MOV DX, FFF0H   ; Initialize address of port0
        MOV AL, 00H     ; Load data with bit 7 as logic 0
ON_OFF: OUT DX, AL      ; Output the data to port 0
```

Delay for a short period of time so as to maintain the data written to the LED

```
        MOV CX, FFFFH   ; Load delay count of FFFFH
HERE:   LOOP HERE       ; Time delay loop
```

The value in bit 7 of AL is complemented to 1 and then a jump is performed to return to the output operation that writes the data to the output port:

```
        XOR  AL, 80H    ; Complement bit 7 of AL
        JMP ON_OFF      ; Repeat to Output the new bit 7
```