Lecture 10

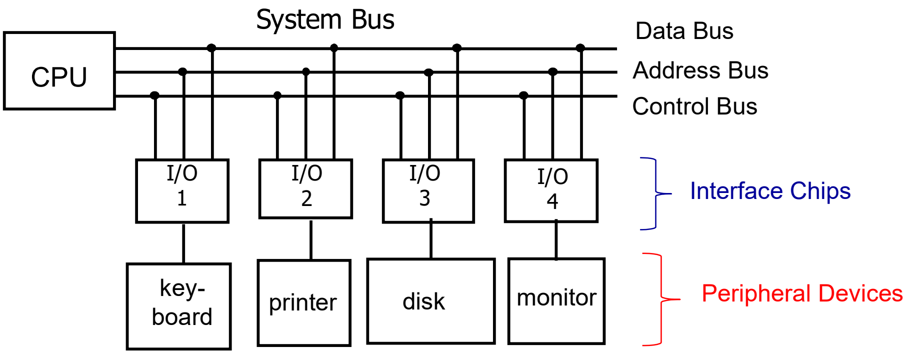
Parallel Input/Output

Topics

- Parallel Input/Output Interfacing
- 8086 Basic I/O Interfaces

Input/Output Interfacing

- **Peripheral Devices** are I/O devices that exchange data with a CPU.
  - Examples : Switch, button, LED, monitor, LCD screen, printer, mouse, keyboard, disk drive, sensor, motor, audio, etc.
- **Interface Chips** are used to resolve the differences between CPU and I/O devices.



System Bus

CPU

Data Bus

Address Bus

Control Bus

I/O 1

I/O 2

I/O 3

I/O 4

Interface Chips

key-board

printer

disk

monitor

Peripheral Devices

I/O Interfacing Methods

- An interface chip is used as bridge between CPU and peripheral device.
- Data formats in peripherals may be different than CPU.
- Data transfer rate of peripherals are slower than CPU.
  A synchronization mechanism is used.
- Communication between CPU and Interface Chip is always parallel.
- Communication between Interface Chip and Peripheral Device can be either parallel or serial.
- There are two I/O interfacing methods:
  - Isolated I/O Method
  - Memory-Mapped I/O Method

Isolated I/O Interfacing Method

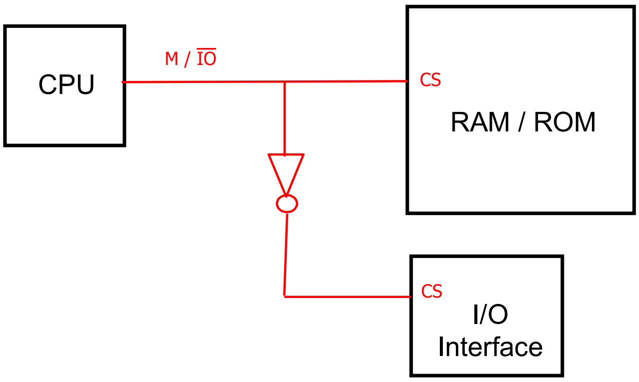
- The M / $\overline{\text{IO}}$ control signal of the CPU is used as a selector between Memory chips and Input/Output chips.
- Separate memory and I/O address spaces are used.
- There are distinct input and output instructions.
- Example 8086 Assembly language instructions : **IN , OUT**

Using M/$\overline{\text{IO}}$ Signal for Isolated I/O Interfacing

The M / $\overline{\text{IO}}$ control signal of the CPU is used as the Memory chip selection and also as the Input-Output Interface chip selection.

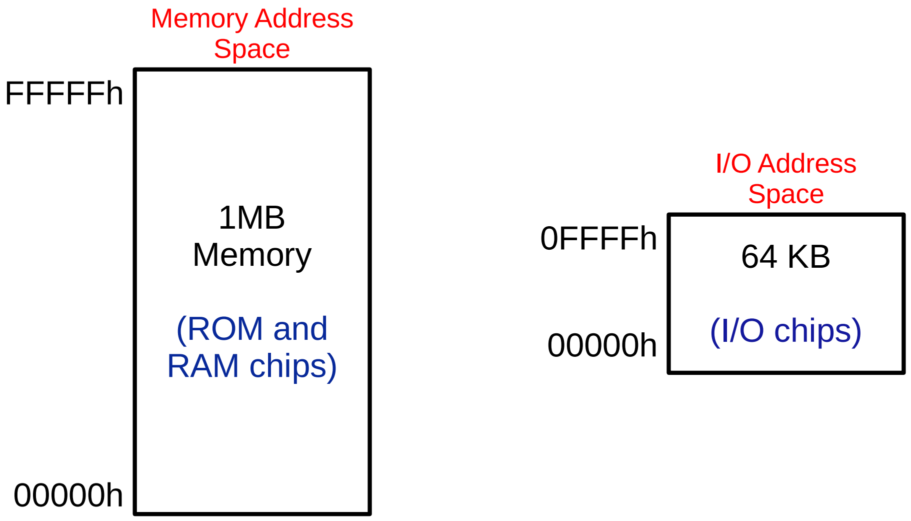
M / $\overline{\text{IO}}$ = 1 means Memory chip is selected

M / $\overline{\text{IO}}$ = 0 means Input-Output chip is selected



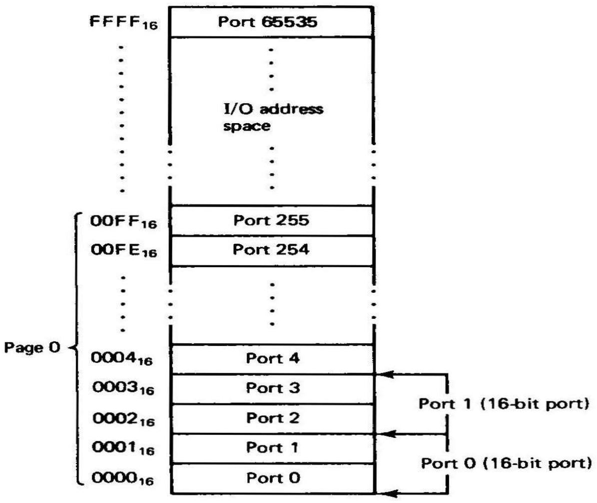
Address Spaces in Isolated I/O Interfacing

- The entire memory address space is used by memory chips only.
- There are no reserved addresses for I/O chips in memory address space, because the I/O addresses are separated from memory addresses.



I/O Address Space in Isolated I/O

- The bytes of data in two consecutive I/O addresses could be accessed as either byte data (8-bit), or as word data (16-bit).
- Example: I/O addresses 0000h, 0001h, 0002h, 0003h can be treated as independent byte ports 0, 1, 2 and 3.
- Or they can be treated as as word ports 0 and 1.

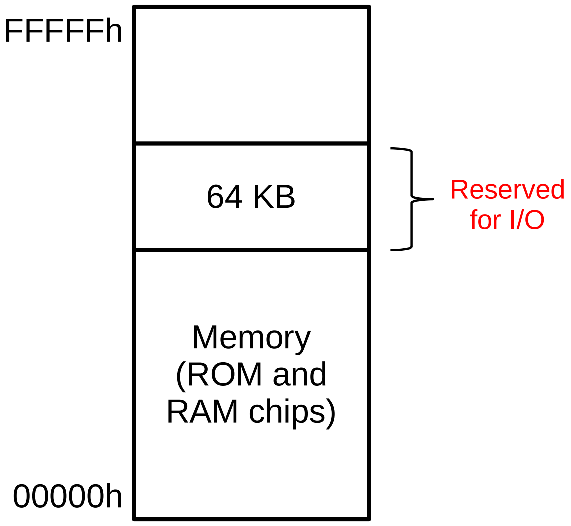


Memory-Mapped
I/O Interfacing Method

- The M / $\overline{\text{IO}}$ control signal is not used in Memory-Mapped interfacing.
- Memory and I/O addresses share common address space.
- There are no specific input/output instructions.
- Usual memory load/store instructions are used.
- Example 8086 Assembly language instruction : **MOV**

Address Spaces in
Memory-Mapped I/O

- There is a reserved address range in the memory space for the I/O chips.
- I/O addresses are part of the main memory addresses.



FFFFFh

64 KB

Reserved
for I/O

Memory
(ROM and
RAM chips)

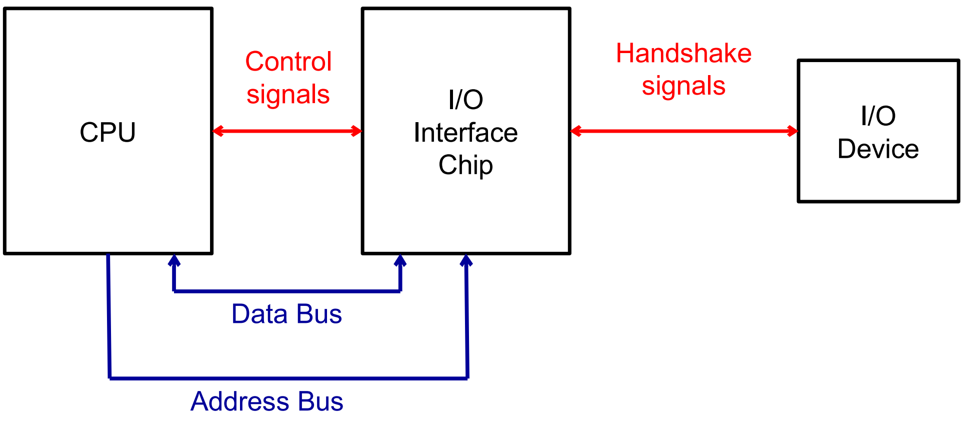
00000h

Differences Between Isolated I/O and Memory-mapped I/O

Isolated I/O	Memory-mapped I/O
I/O devices are treated separate from memory.	I/O devices are treated as part of memory.
Entire 1 MB address space is available for use as memory.	Entire 1 MB cannot be used as memory, since I/O devices use some addresses of memory.
Separate instructions (IN and OUT) are used for I/O.	No separate instructions are needed. The MOV instruction is used for I/O.
Data transfer takes place between I/O port and AL or AX register only.	Data transfer can take place between I/O port and any CPU register.

Input/Output Transfer Synchronization

- The I/O interface chip has two set of signals.
- **Control signals:** Synchronize data transfer between CPU and Interface chip.
- **Handshaking signals:** Synchronize data transfer between Interface chip and I/O device.



CPU and I/O Interfacing Methods

- **Programmed I/O (Polling method)**
  - CPU checks device status in a loop.
  - CPU waits for I/O module to complete operation.
  - Wastes CPU time.
- **Interrupt Driven I/O**
  - Overcomes CPU unnecessary waiting.
  - No repeated CPU checking of device.
  - I/O module interrupts when ready.
- **Direct Memory Access (DMA)**
  - Requires additional hardware module on system bus.
  - DMA controller takes over from CPU, for I/O operations.

Programmed I/O
(Polling method)

- Input Operations :
  CPU checks a status bit of the interface chip, to find out whether the interface has received new data from input device.
- Output Operations :
  CPU checks a status bit of the interface chip, to find out whether it can send new data to interface chip.

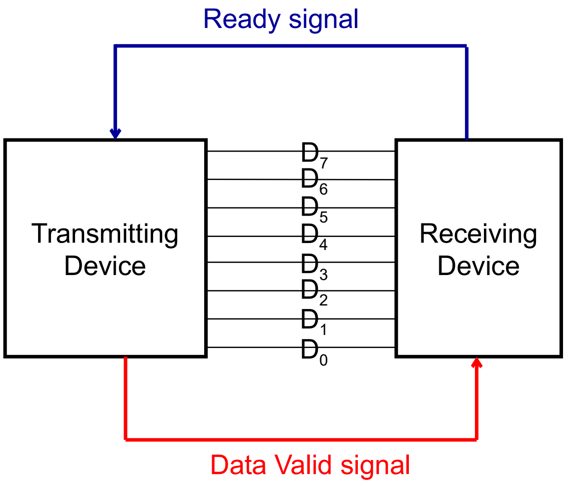
Interrupt Driven I/O

- Input Operations :
  Interface chip interrupts the CPU, whenever it has received new data from input device.

- Output Operations :
  Interface chip interrupts the CPU, whenever it can accept new data from microprocessor.

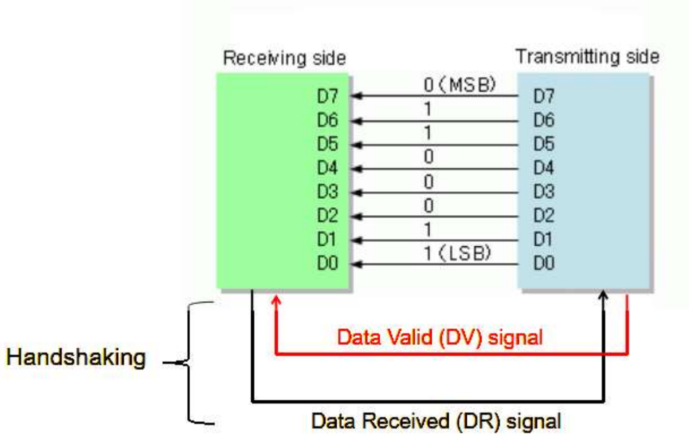
Parallel Communication

- In parallel communication, all bits are transferred at the same time.
- Each bit is transferred along its own line.
- In addition to eight parallel data lines, other lines are used to read status information and send control signals.



Example: Synchronization of two parallel devices

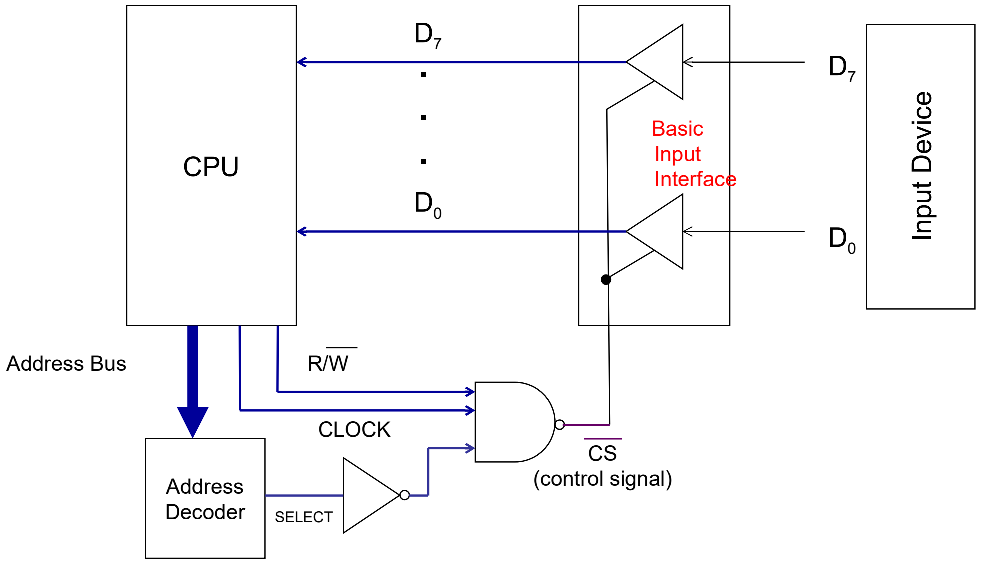
- **Transmitter** and **receiver** interfaces use handshaking protocol.
- **Transmitter** initiates 2-wire handshaking
  - DV low indicates new data is available.
  - DR low indicates that receiver has read the data.



Basic Parallel Input Interface

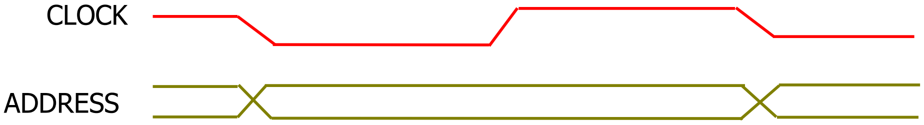
- The basic input device (to the microprocessor) is a set of tri-state buffers.
- When data is coming in by way of a data bus (either from port or memory) it must come in through a three-state buffer.
- **Tri-state buffer** chips can be used as input interface. (Example chip: 74244)
- Output pins of **interface** are connected to **Data Bus** of CPU (as the inputs of CPU).
- Input pins of **interface** are connected to an input device.

Basic Parallel Input Interface

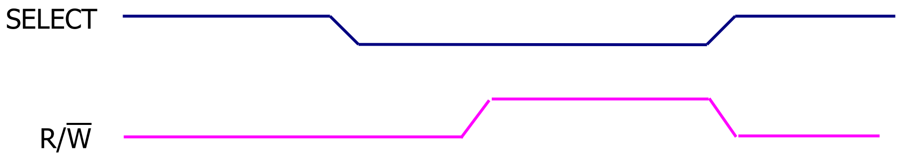


Basic Input Interface Timing Diagram

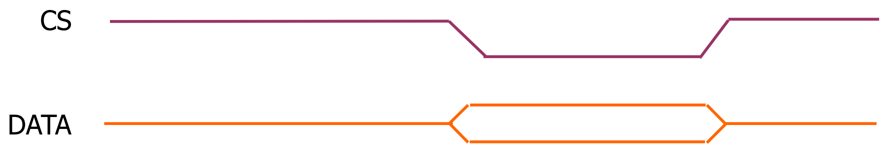
1) Address is placed on address bus at the falling edge of clock.



2) Address is decoded, in order to form the CS (Chip Select) (Enable) signal.



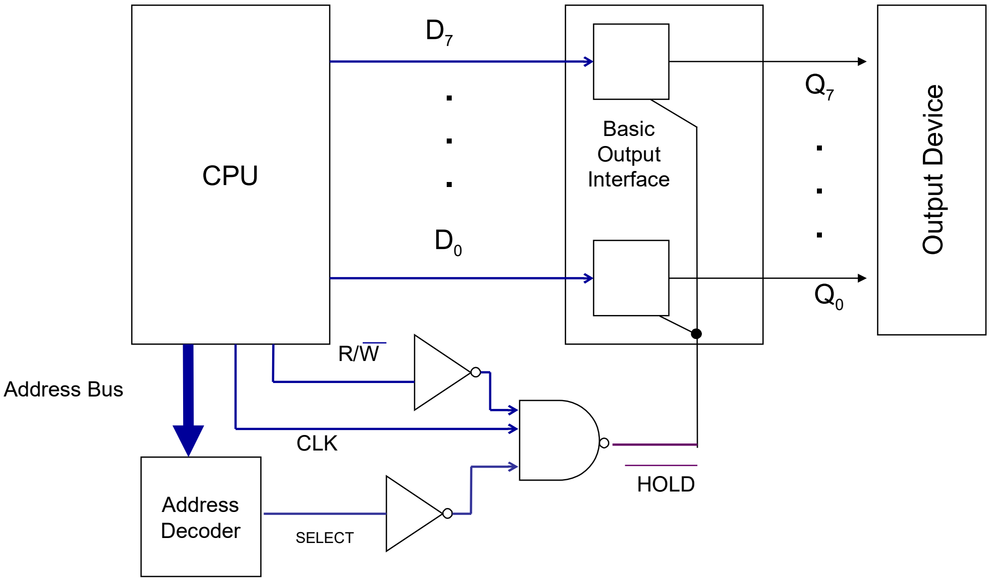
3) NAND {CLK, M/$\overline{IO}$, $\overline{READ}$} is applied, in order to form CS signal.



Basic Parallel Output Interface

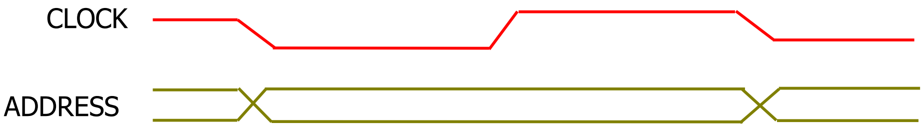
- The basic output device (from the microprocessor) is a set of latches.
- Since the data provided by the CPU to an output port is on the system data bus for a limited amount of time (nano seconds), it must be latched before it is lost.
- **D-Latches** (Data Latches) can be used as output interface. (Example chip: 74374)
- Inputs of **interface** are connected to the **Data Bus** of CPU.
- Outputs of **interface** are connected to an output device.

Basic Parallel Output Interface

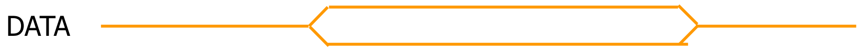


Basic Output Interface Timing Diagram

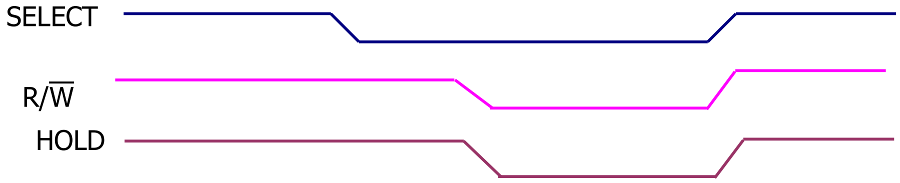
1) Address is placed on address bus at the falling edge of clock.



2) Data is placed on the data bus.



3) Address is decoded, in order to form the Select signal.
4) NAND {CLK, SELECT, R/W} is applied, in order to form HOLD.



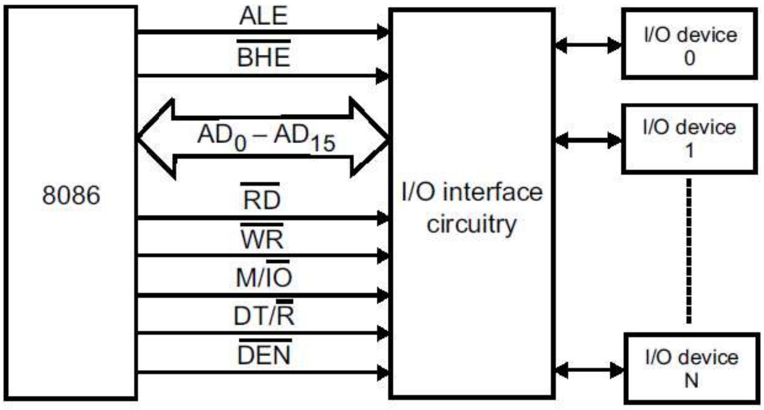
Topics

- Parallel Input/Output Interfacing
- 8086 Basic I/O Interfaces

8086 Minimum Mode I/O Interface

The I/O interface circuitry performs the following tasks.

- Selecting the particular I/O port
- Synchronizing data transfer
- Writing the output data
- Reading the input data



Parallel I/O Interface Chips used with 8086 CPU

Chip Number	Chip Name	I/O Direction
74373, 74374	Latch	Output only
74244, 74245	Buffer	Input / Output
8255	Programmable Peripheral Interface	Input / Output

8086 Assembly Instructions for I/O

- The 8086 CPU can access data from I/O ports as well as from the memory.
- 8086 can be used with isolated I/O, or memory-mapped I/O interface methods.
- In isolated method, the IN and OUT instructions are used between CPU and I/O interface for data transfers.
- Any I/O port in the interface chip can be used for either 8-bit or 16-bit data transfer.

Addressing Method	Port Address	Input Instruction	Output Instruction	Description
Direct	Fixed	IN AL, port	OUT port, AL	Port addresses between 00-FF are written with instruction.
		IN AX, port	OUT port, AX	
Indirect	Variable	IN AL, DX	OUT DX, AL	Port addresses between 100h-FFFFh should be loaded to DX register first.
		IN AX, DX	OUT DX, AX	

Input / Output Ports

I/O devices are connect to the CPU through the I/O interface chip ports.

Ports are:

- Registers (part of the I/O interface chip)
- Can be 8, 16, or 32 bits
- Addressed in the range 0000-FFFFh
- Accessed with Assembly instructions IN, OUT

IN Instruction

- The IN instruction copies data from a peripheral device port to the AL or AX register.
- If an 8-bit port is read, the data will go to AL.
- If a 16-bit port is read, the data will go to AX.
- IN instruction does not change any flag.
- IN instruction has two formats, fixed port and variable port.

- **FIXED PORT FORMAT:** The 8-bit address of a port is specified directly in the instruction. Up to 256 possible ports can be addressed.

General Syntax

```
IN Accumulator, PortNumber
```

Examples: 8 bit fixed port address

`IN AL, OC8h`	Input a byte from port C8h to AL

`IN AX, 34h`	Input a word from port 34h to AX

IN Instruction

- **VARIABLE-PORT FORMAT:** The port address is loaded into the DX register (Indirect Addressing) before the IN instruction.
- Since DX is a 16-bit register, the port address can be any number between 0000H and FFFFH.
- Therefore, up to 65,536 ports are addressable.
- The variable-port IN instruction has advantage that the port address can be computed, or dynamically determined in the program.

Examples: 16 bit variable port address

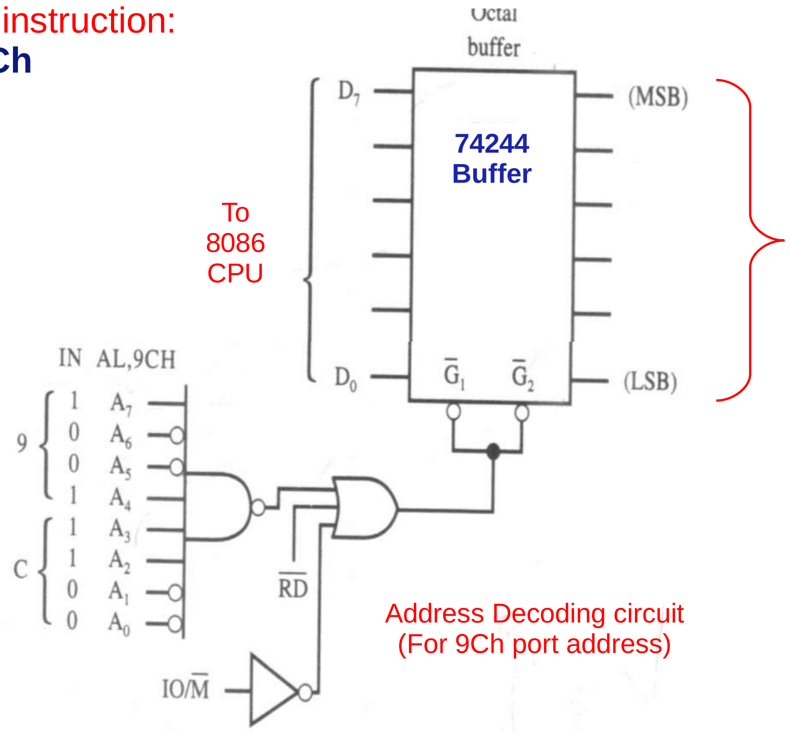
`MOV DX, OFF78h`	Initialize DX to point to port
`IN AL, DX`	Input a byte from 8-bit port FF78h to AL
`IN AX, DX`	Input a word from 16-bit port FF78h to AX

Input Buffering

- Input devices must be isolated from the system data bus.
- Otherwise unwanted data may be transferred on to the data bus.
- When data comes in from a port or memory, data must be input through a tri-state buffer.
- Tri-state buffers are used which provide isolation as well as strengthen the signal.
- Example: Interfacing the input devices such as switches, or buttons require buffers.

Example: Address Decoding Circuit for Input Port 9Ch

Example instruction:
IN AL, 9Ch



74244 Buffer

To 8086 CPU

From Input Device

Address Decoding circuit (For 9Ch port address)

OUT Instruction

- The OUT instruction copies a byte from AL or a word from AX to the specified port.
- OUT instruction does not change any flag.
- Similarly to IN instruction, the OUT instruction has two forms, fixed port and variable port.
- **FIXED PORT FORMAT:** The 8-bit port address is specified directly in the instruction.
- Up to 256 possible ports can be addressed.

General Syntax

```
OUT PortNumber, Accumulator
```

Examples: 8 bit fixed port address

`OUT 3Bh, AL`	Copy the content of AL to port 3Bh

`OUT 2Ch, AX`	Copy the content of AX to port 2Ch

OUT Instruction

- **VARIABLE PORT FORMAT:** The OUT instruction copies , content (data) of AL or AX to the port at an address contained in DX.
- The DX register must be loaded with the desired port address, before this form of the OUT instruction is used.

Examples: 16 bit variable port address

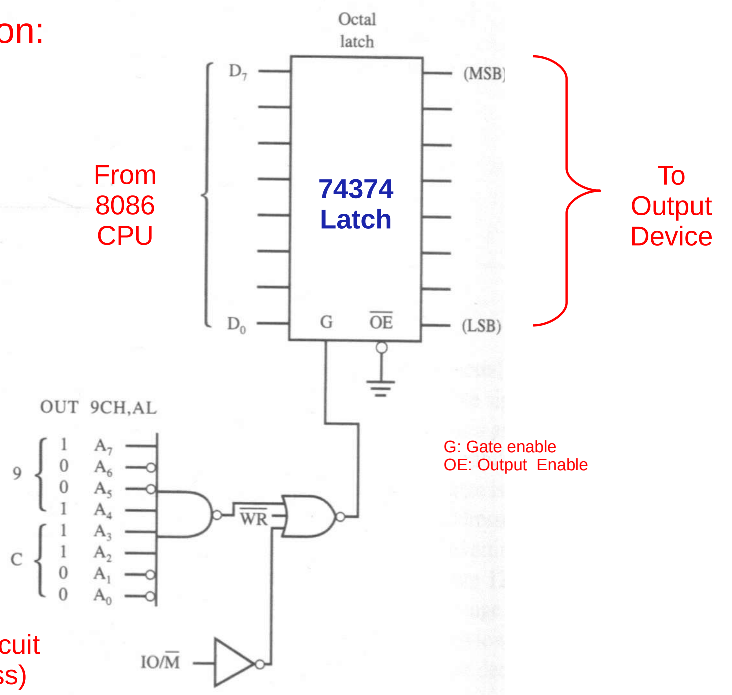
`MOV DX, 0FFF8h`	Load a port address (16 bit) in DX
`OUT DX, AL`	Copy content of AL to port FFF8h
`OUT DX, AX`	Copy content of AX to port FFF8h

Output Latching

- When data is sent out by CPU, the data on the data bus must be latched.
- Memory chips have internal latches to store data.
- A latching system must be designed for I/O ports.
- Data provided by the processor is available only for short period of time.
- Data must be latched, otherwise it will be lost.
- Example: Interfacing output devices like LEDs require latches.

Example: Address Decoding Circuit for Output Port 9Ch

Example instruction:
OUT 9Ch, AL



From 8086 CPU

74374 Latch

To Output Device

G: Gate enable
OE: Output Enable

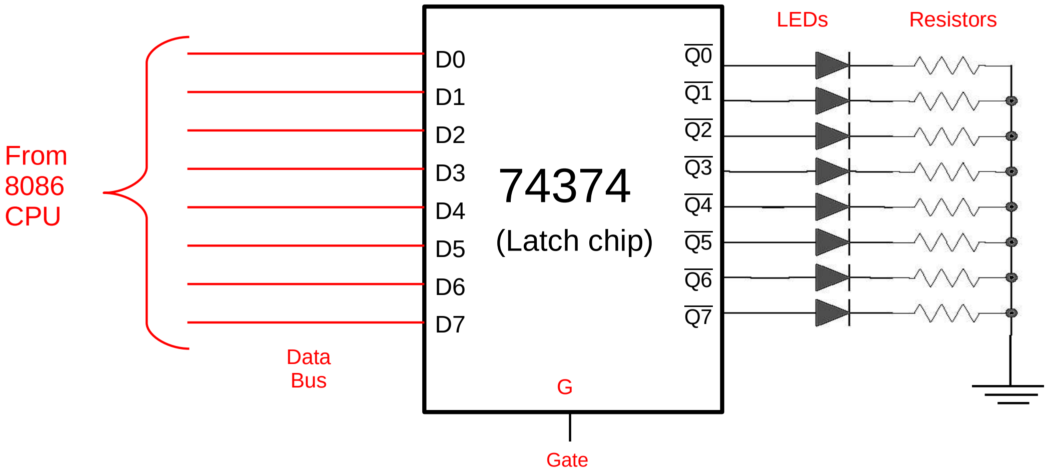
Address Decoding circuit
(For 9Ch port address)

Example: Output Interface with 74374 Latch chip

- The 74374 chip is a three-state Octal (8 bit) D-type Latch.
- It can be used for output.
- Suppose 8 LEDs (Light Emitting Diodes) are connected to the latch chip.
- The latch chip is connected to the 8086 CPU.
- Isolated I/O interface method will be used.
- Suppose the fixed port address of the 74374 latch chip is 00004h.
- Block diagram of 74374 chip and the LEDs are shown below.

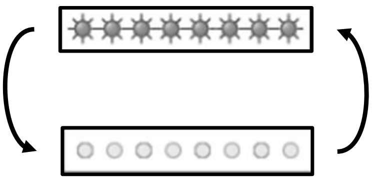
Application Example1: LEDs Blinking Program

- 74374 Latch chip is used to drive 8 LEDs (supplying the +5V).
- Resistors are required for limiting the electric current on each LED.



LEDs Blinking (Flashing) Program

- Write an Assembly program that turns all LEDS on and off continuously in endless loop.
- There should be some waiting time between each on and off operation.



All LEDs are on

All LEDs are off

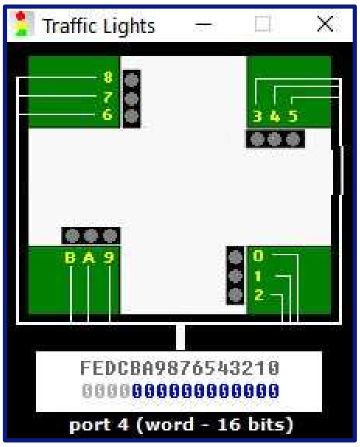
LEDs Blinking Program

```
; Program for Blinking (Flashing) all LEDs.
; (With 74374 Latch chip)
; In Emu8086, all 12 traffic lights are turned on and off.

.model small
.code
MainDongu:         ; Endless main loop
mov ax, 0FFFh    ; Turn on all 12 LEDs
out 4, ax              ; Fixed address of output port is 4 (in Emu8086)

; Wait for 1 second.
mov cx, 255
bekle1: loop bekle1  ; CX is implicit counter operand

mov ax, 0                ; Turn off all 12 LEDs
out 4, ax                ; Output port address is 4

; Wait for 1 second again.
mov cx, 255
bekle2: loop bekle2   ; CX is implicit counter operand

JMP MainDongu      ; Goto endless loop
END                          ; End of file
```

Testing the LEDs Blinking Program
(Using Emu8086 Emulator)

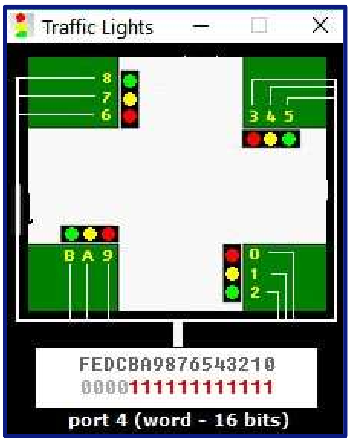
- Run the LEDs program in Emu8086 emulator.
- Open the Virtual Devices --> Traffic_Lights.exe program.
- By default, Emu8086 uses the port address at 4 for LED outputs.
- Emu8086 enumarates the 12 LEDs as 0,1,2,3,...,A,B.
- The following is the emulation screen for the LED binking application, by using the 12 traffic ligths as LEDs.

All LEDs are turned off



```
FEDCBA9876543210
0000000000000000
port 4 (word - 16 bits)
```

All LEDs are turned on



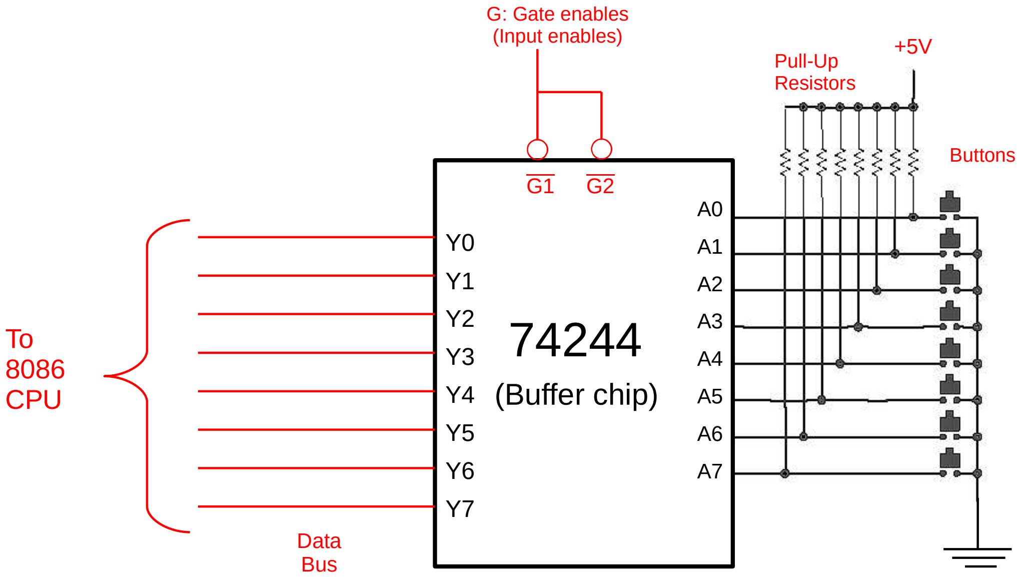
```
FEDCBA9876543210
0000111111111111
port 4 (word - 16 bits)
```

Example: Input Interface with 74244 Data Buffer chip

- The 74244 chip is a Octal (8 bit) 3-state Buffer (Line Driver / Line Receiver).
- The block diagram below shows an example design with the connection signals and the address decoding circuit.
- Suppose 8 Buttons are connected to the 74244 data buffer chip.
- The buffer chip is connected to 8086 CPU.
- Suppose the port address of the buffer chip will be stored in DX register as 0001h.
- Block diagram of 74244 chip and the buttons are shown below.

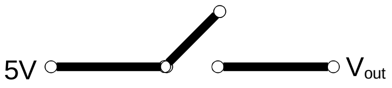
Application Example2: Buttons Input Program

- The Buffer chip has 8 inputs (A1-A8), and 8 outputs (Y1-Y8).
- Each of the $\overline{\text{G1}}$ and $\overline{\text{G2}}$ pins controls 4 input bits.

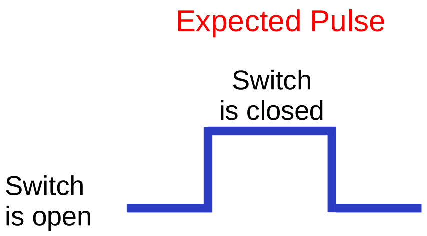


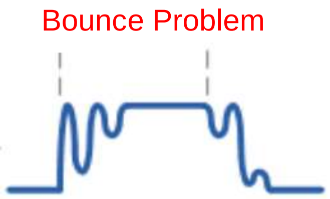
Switch Bounce Problem

- When a switch (or button) is closed, it may cause multiple ON and OFF signals lasting several milliseconds.
- The multiple switch closing/opening problem is called as Switch Bounce problem. (Also called as Contact Noise problem)
- Software solution: Detect the first key-press, wait for a short time, then read again.
- Hardware solution: Use a Pull-Up or Pull-Down resistor with the switch (or button).



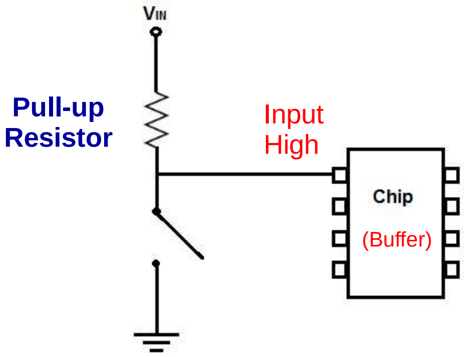
If switch is open, the output is 0V (Low, Logic 0)
If switch is closed, the output is 5V (High, Logic 1)





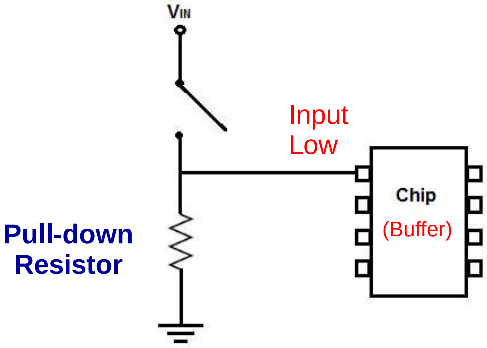
Switch with
Pull-Up Resistor

- The pull-up resistor pulls a pin of a chip up to a HIGH state.
- Input of the buffer chip is normally HIGH.
- When the switch is closed, input of the chip drops to LOW.



Switch with
Pull-Down Resistor

- It is the opposite of Pull-Up resistor.
- The pull-down resistor pulls a pin of a chip down to a LOW state.
- Input of the buffer chip is normally LOW.
- When the switch is closed, input of the chip becomes HIGH.



Buttons Input Program

- Write an Assembly program to continuously read the 8 buttons connected to the 74244 buffer chip.
- Program should stop when user presses any of the buttons.

```
; Program for reading Buttons.
; ( With 74244 Buffer chip )
.model small
.code

mov dx, 0001h        ; Address of input port is 01h.

Dongu:                ; Endless loop
in AL, dx             ; Read all 8 button inputs
cmp AL, 0FFh          ; FF means all inputs are high (No button is pressed).
                      ; When a button is pressed, its input drops to low.
JNE Son               ; Goto label if not equal (Any button is pressed)
JMP Dongu             ; Go to endless loop (No button is pressed).

Son:
.EXIT                 ;Program stops
END                   ;End of file
```