CSE213

Microcontroller Programming

I/O Interface

Introduction

- Basic I/O interface
- Handshaking process
- Serial and Parallel communication
- I/O interface examples

Chapter Objectives

Upon completion of this chapter, you will be able to:

- Explain the operation of the basic input and output interfaces.
- Use I/O instructions.
- Define handshaking and explain how to use it with I/O devices.

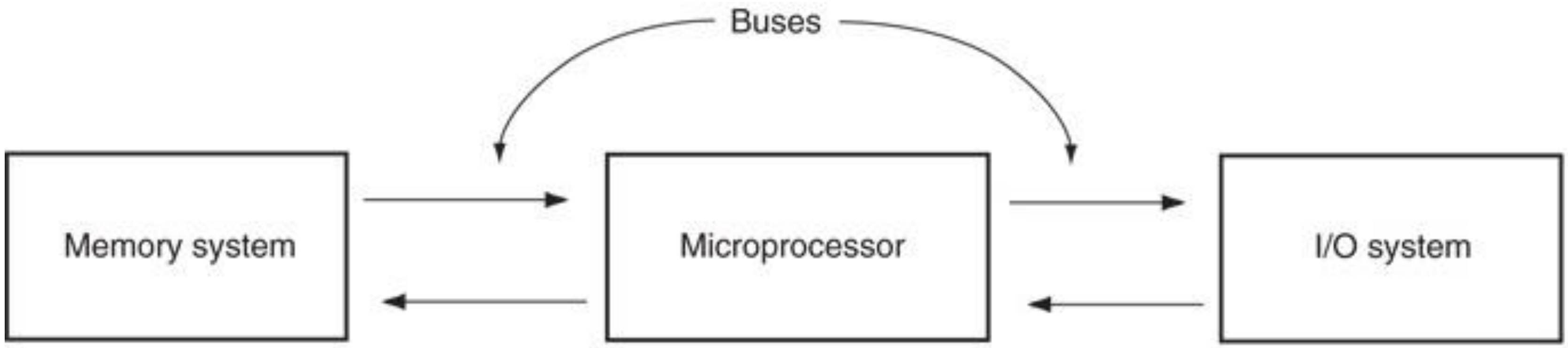
Chapter Objectives

Upon completion of this chapter, you will be able to:

- Learn the differences between serial and parallel communication.
- Understand how to interface external devices to the microprocessor via I/O ports.
- Use virtual devices provided with the emulator.

INTRO TO I/O INTERFACE

- A microprocessor is great at solving problems.
- But, if it can not communicate with the outside world, it is of little worth.
- Therefore, I/O interface is of great importance.



INTRO TO I/O INTERFACE

- I/O instructions (IN, INS, OUT, and OUTS) are explained in this lecture.
- Also isolated (direct or I/O mapped I/O) and memory-mapped I/O, the basic input and output interfaces, and handshaking.
- Knowledge of these topics makes it easier to understand the connection and operation of the programmable interface components and I/O techniques.

The I/O Instructions

- One type of instruction transfers information to an I/O device (OUT).
- Another reads from an I/O device (IN).
- Instructions are also provided to transfer strings of data between memory and I/O.
  - INS and OUTS, found except the 8086/8088

- Instructions that transfer data between an I/O device and the microprocessor’s accumulator (AL, AX, or EAX) are called **IN** and **OUT**.
- The I/O address is stored in register DX as a 16-bit address or in the byte (p8) immediately following the opcode as an 8-bit address.
- The 16-bit address is called a **variable address** because it is stored in DX, and then used to address the I/O device.

- Other instructions that use DX to address I/O are the INS and OUTS instructions.
- I/O ports are 8 bits in width.
  - a 16-bit port is actually two consecutive 8-bit ports being addressed
  - a 32-bit I/O port is actually four 8-bit ports

- When data are transferred using IN or OUT, the I/O address, (**port number** or simply port), appears on the address bus.
- External I/O interface decodes the port number in the same manner as a memory address.
  - the 8-bit fixed port number (p8) appears on address bus connections A_7–A_0 with bits A_{15}–A_8 equal to 00000000_2
  - connections above A_{15} are undefined for I/O instruction

- The 16-bit variable port number (DX) appears on address connections A_{15}–A_0.
- The first 256 I/O port addresses (00H–FFH) are accessed by both fixed and variable I/O instructions.
  - any I/O address from 0100H to FFFFH is only accessed by the variable I/O address (via DX)
- In a PC computer, all 16 address bus bits are decoded with locations 0000H–03FFH.
  - used for I/O inside the PC on the ISA (**industry standard architecture**) bus

- INS and OUTS instructions address an I/O device using the DX register.
  - but do not transfer data between accumulator and I/O device as do the IN/OUT instructions
  - Instead, they transfer data between memory and the I/O device
- Pentium 4 and Core2 operating in the 64-bit mode have the same I/O instructions.
- There are no 64-bit I/O instructions in the 64-bit mode.
  - most I/O is still 8 bits and likely will remain so

Summary of I/O instructions

- IN AL, p8
- IN AX, p8
- IN AL, DX
- IN AX, DX

- OUT p8, AL
- OUT p8, AX
- OUT DX, AL
- OUT DX, AX

p8: A port number between 0 and 255
For other port numbers, use DX register.

Isolated (direct) and Memory-Mapped I/O

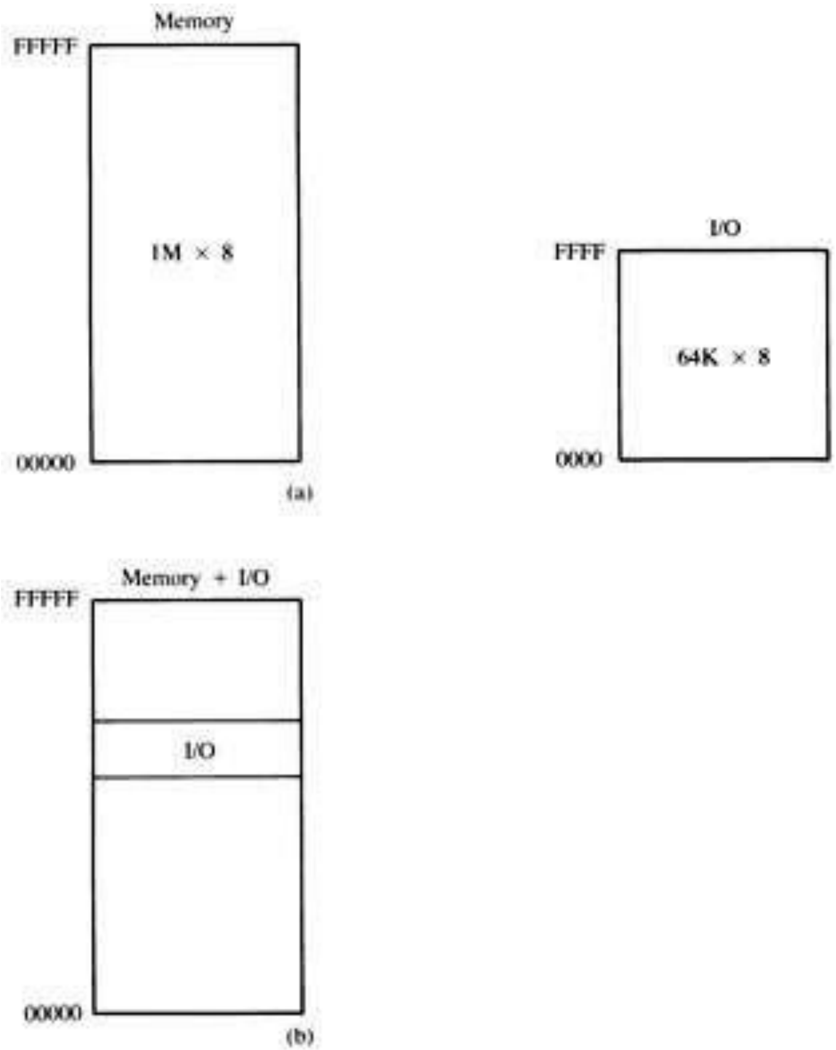
- Two different methods of interfacing I/O: **isolated I/O** and **memory-mapped I/O**.
- In isolated I/O, the IN, INS, OUT, and OUTS transfer data between the microprocessor's accumulator or memory and the I/O device.
- In memory-mapped I/O, any instruction that references memory can accomplish the transfer.
- The PC <u>does not</u> use memory-mapped I/O.

Isolated I/O

- The most common I/O transfer technique used in the Intel-based system is isolated I/O.
  - *isolated* describes how I/O locations are isolated from memory in a separate I/O address space
- Addresses for isolated I/O devices, called ports, are separate from memory.
- Because the ports are separate, the user can expand the memory to its full size without using any of memory space for I/O devices.

- A disadvantage of isolated I/O is that data transferred between I/O and microprocessor must be accessed by the IN, INS, OUT, and OUTS instructions.
- Separate control signals for the I/O space are developed (using M/$\overline{\text{IO}}$ and W/$\overline{\text{R}}$), which indicate an I/O read ($\overline{\text{IORC}}$) or an I/O write ($\overline{\text{RD}}$) operation.
- These signals indicate an I/O port address, which appears on the address bus, is used to select the I/O device.

Figure 11–1 The memory and I/O maps for the 8086/8088 microprocessors. (a) Isolated I/O. (b) Memory-mapped I/O.

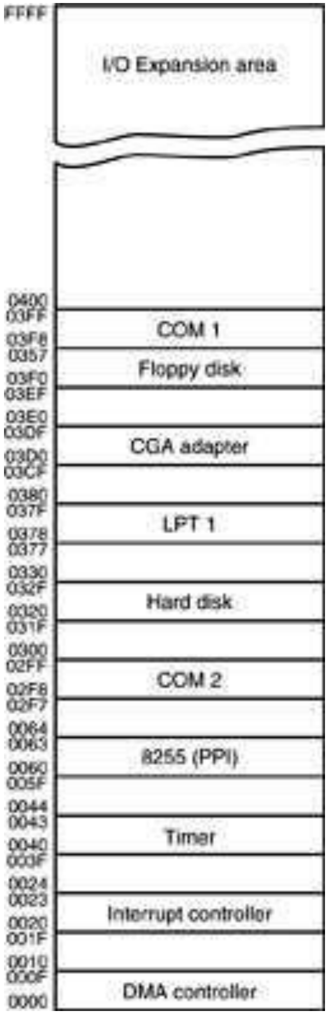


– in the PC, isolated I/O ports are used to control peripheral devices
– an 8-bit port address is used to access devices located on the system board, such as the timer and keyboard interface
– a 16-bit port is used to access serial and parallel ports, video and disk drive systems

Memory-Mapped I/O

- Memory-mapped I/O does not use the IN, INS, OUT, or OUTS instructions.
- It uses any instruction that transfers data between the microprocessor and memory.
  - treated as a memory location in memory map
- Advantage is any memory transfer instruction can access the I/O device.
- Disadvantage is a portion of memory system is used as the I/O map.
  - reduces memory available to applications

Personal Computer I/O Map



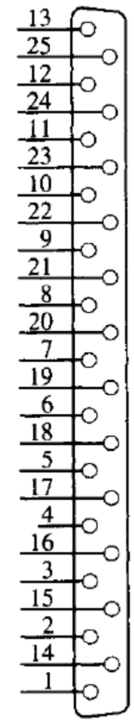
- the PC uses part of I/O map for dedicated functions, as shown here
- I/O space between ports 0000H and 03FFH is normally reserved for the system and ISA bus
- ports at 0400H–FFFFH are generally available for user applications, main-board functions, and the PCI bus
- 80287 coprocessor uses 00F8H–00FFH, so Intel reserves I/O ports 00F0H–00FFH

Figure 11–2 I/O map of a personal computer illustrating many of the fixed I/O areas.

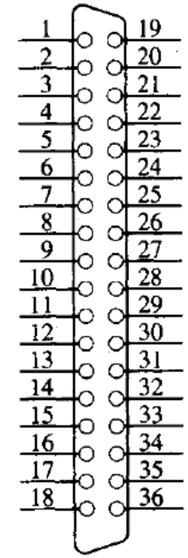
Handshaking

- Many I/O devices accept or release information slower than the microprocessor.
- A method of I/O control called **handshaking** or **polling**, synchronizes the I/O device with the microprocessor.
- An example is a parallel printer that prints a few hundred characters per second (CPS).
- The processor can send data much faster.
  - a way to slow the microprocessor down to match speeds with the printer must be developed

- Fig 11–5 illustrates typical input output connections found printer.
  - data transfers via data connections
- ASCII data are placed on D_7–D_0, pulse is then applied to $\overline{\text{STB}}$ connection.
  - BUSY indicates the printer is busy
  - $\overline{\text{STB}}$ is a clock pulse used to send data to printer
- The strobe signal sends or clocks the data into the printer so that they can be printed.
  - as the printer receives data, it places logic 1 on the BUSY pin, indicating it is printing data

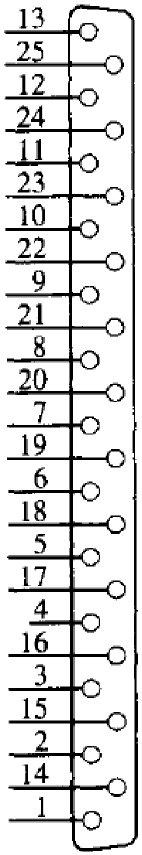


Connector DB25

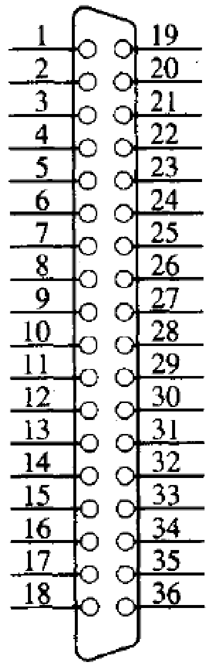


Connector CENT36

Figure 11–5 The DB25 connector found on computers and the Centronics 36-pin connector found on printers for the Centronics parallel printer interface.



Connector DB25



Connector CENT36

DB25 Pin number	CENT36 Pin number	Function	DB25 Pin number	CENT36 Pin number	Function
1	1	$\overline{\text{Data Strobe}}$	12	12	Paper empty
2	2	Data 0 (D0)	13	13	Select
3	3	Data 1 (D1)	14	14	Afd
4	4	Data 2 (D2)	15	32	$\overline{\text{Error}}$
5	5	Data 3 (D3)	16	—	$\overline{\text{RESET}}$
6	6	Data 4 (D4)	17	31	Select in
7	7	Data 5 (D5)	18—25	19—30	Ground
8	8	Data 6 (D6)	—	17	Frame ground
9	9	Data 7 (D7)	—	16	Ground
10	10	$\overline{\text{Ack}}$	—	33	Ground
11	11	Busy			

- The software polls or tests the BUSY pin to decide whether the printer is busy.
  - If the printer is busy, the processor waits
  - if not, the next ASCII character goes to the printer
- This process of interrogating the printer, or any asynchronous device like a printer, is called handshaking or polling.

I/O COMMUNICATION METHODS

1. Serial communication
2. Parallel communication

- Serial communication is the process of sending data one bit at one time, sequentially, over a communication channel or computer bus.
- In parallel communication, on the other hand, several bits are sent together on a link comprising of several wired channels in parallel.

Examples of serial communication architectures

- RS-232
- I²C (Inter Integrated Circuit)
- SPI (Serial Peripheral Interface)
- USB (Universal Serial Bus)
- FireWire
- Ethernet
- Fibre Channel
- SATA (Serial Advanced Technology Attachment)
- PCI (Peripheral Component Interconnect) Express

Examples of parallel communication architectures

- Printer port
- ISA (Industry Standard Architecture)
- ATA (Advanced Technology Attachment)
- SCSI (Small Computer System Interface)
- PCI

Serial vs Parallel

- Serial circuitry is simple and cheap.
- Parallel circuitry is complex and expensive.
- In parallel communication, you can send many bits at once; however, receive 'at once' is not a true statement.
- Each bit travels a slightly different path down the cable and to a different pin on the chip .
- The small differences in path length that each bit travels becomes more and more significant as speed increases.
- Therefore, some sort of buffering and synchronization is necessary.
- In serial communication, no such a problem exists.
- So, after a certain speed and distance threshold, serial communication is superior.

Interfacing processors to serial and parallel devices

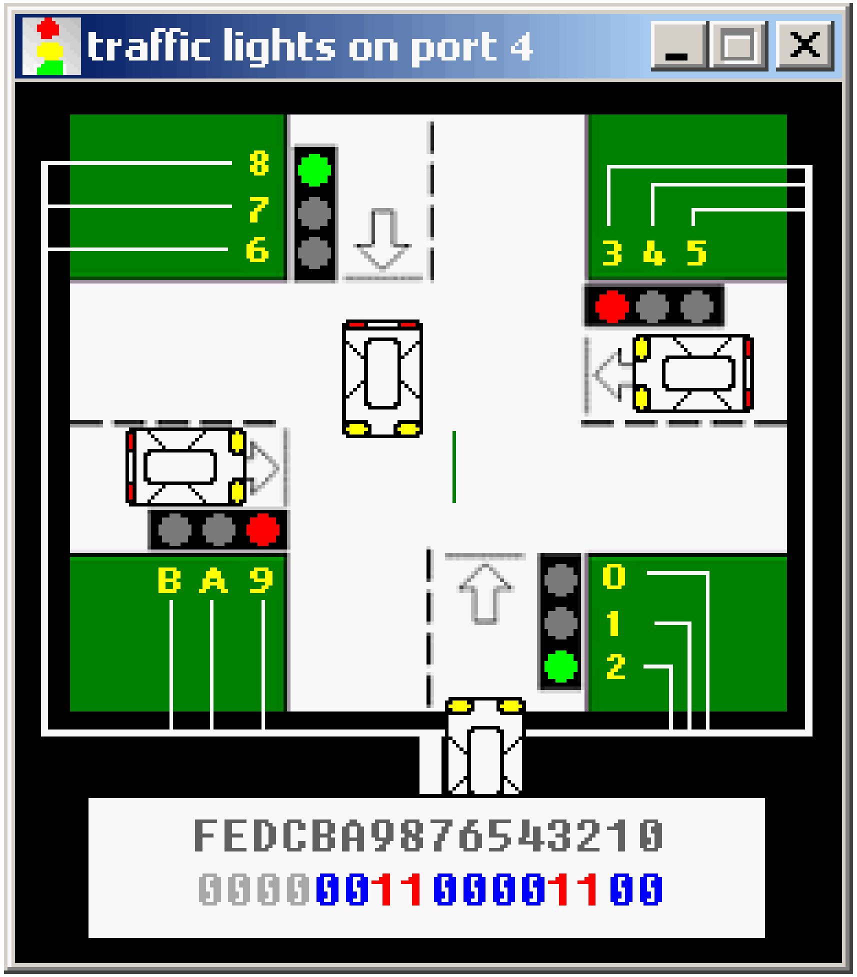
- Processors use peripheral interface circuits to interface to serial and parallel devices.
- These circuits are either provided as separate integrated circuits or built inside the processor.

Examples:

- A UART (Universal asynchronous receiver/transmitter) is used for **serial** communications over a computer or peripheral device serial port.
- Intel 8255 chip is used to give the CPU access to programmable **parallel** I/O.

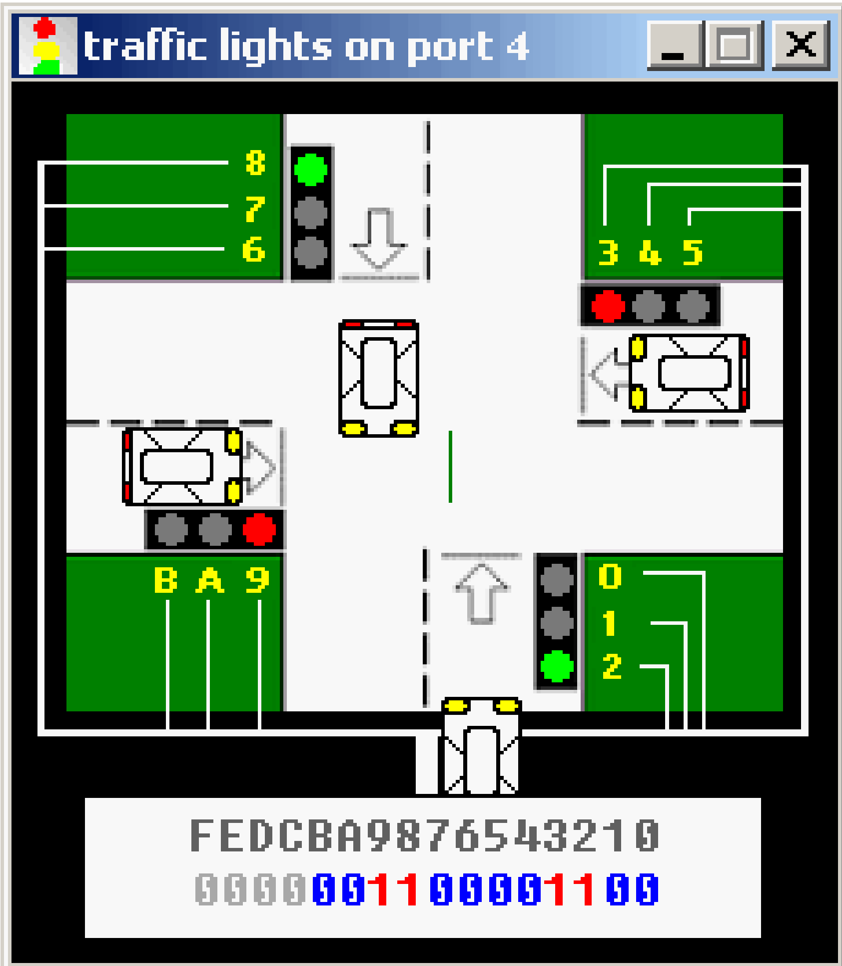
I/O INTERFACE EXAMPLES

Traffic Lights Emulation



- Traffic lights are controlled by sending data to I/O port 4.
- There are 12 lamps:
  - 4 green
  - 4 yellow
  - 4 red.

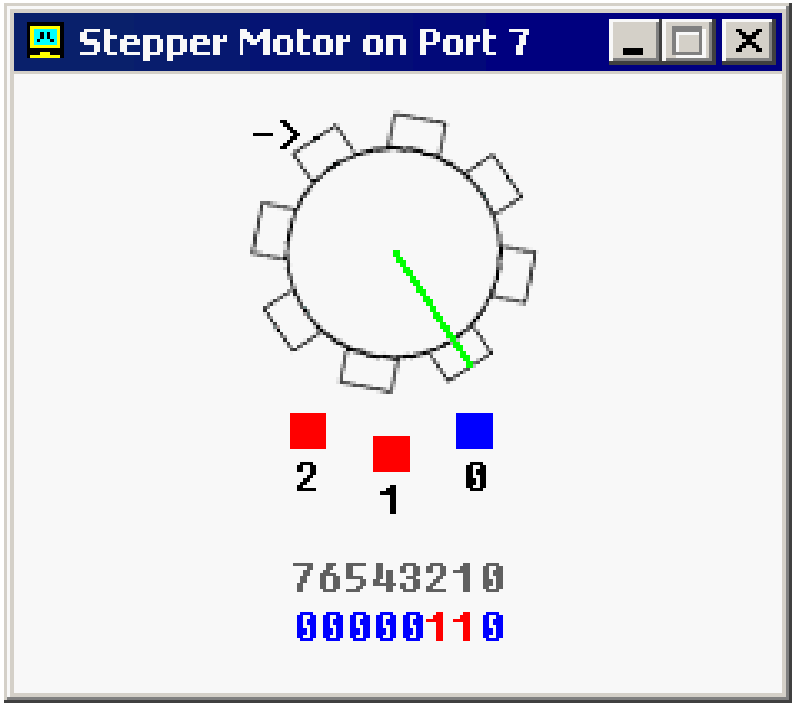
Traffic Lights Emulation



- You can set the state of each lamp by setting its bit:
  1 - the lamp is turned on.
  0 - the lamp is turned off.
- Bits (12 to 15) are unused.
- For example, all lights will be red when following code is executed:

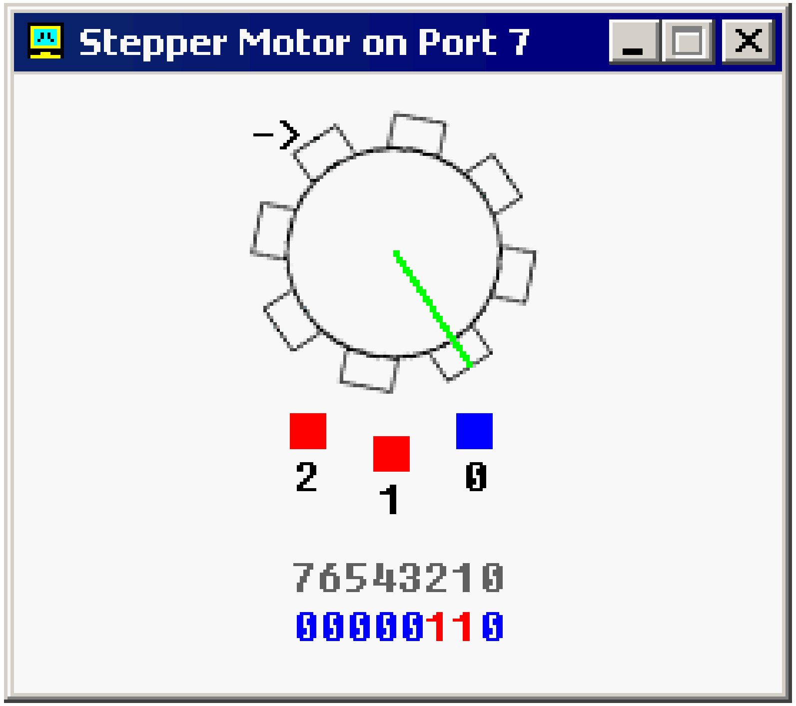
```
MOV AX, 0000001001001001b
OUT 4, AX
```

Stepper Motor



- The stepper motor is controlled by sending data to I/O port 7.
- Stepper motor is electric motor that can be precisely controlled by signals from a computer.
- The motor turns through a precise angle each time it receives a signal.

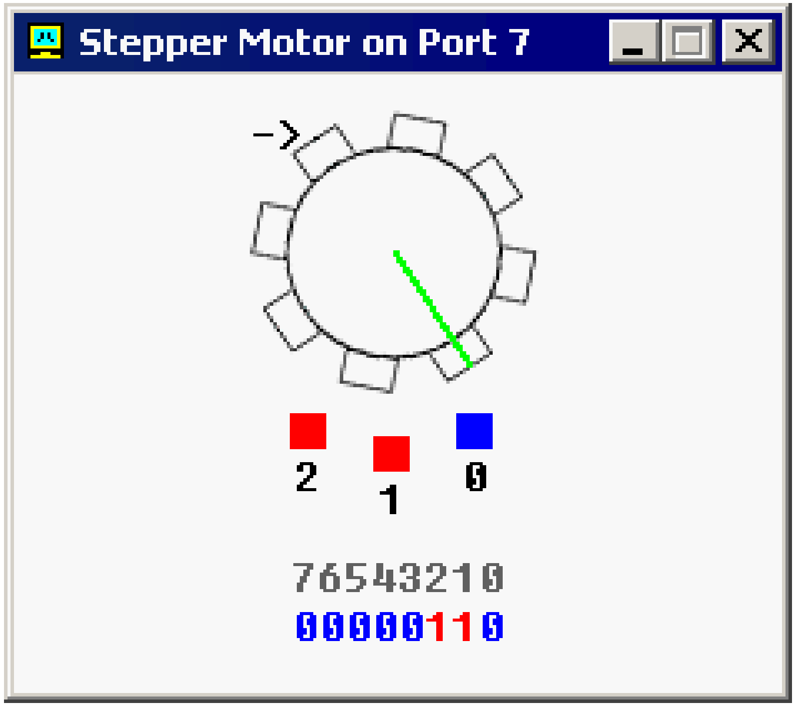
Stepper Motor



- By varying the rate at which signal pulses are produced, the motor can be run at different speeds or turned through an exact angle and then stopped.

- This is a basic 3-phase stepper motor, it has 3 magnets controlled by bits **0, 1 and 2**. other bits (3..7) are unused.

Stepper Motor



- When magnet is working it becomes red.
- The arrow in the left upper corner shows the direction of the last motor move.
- Green line is here just to see that it is really rotating.

Stepper Motor

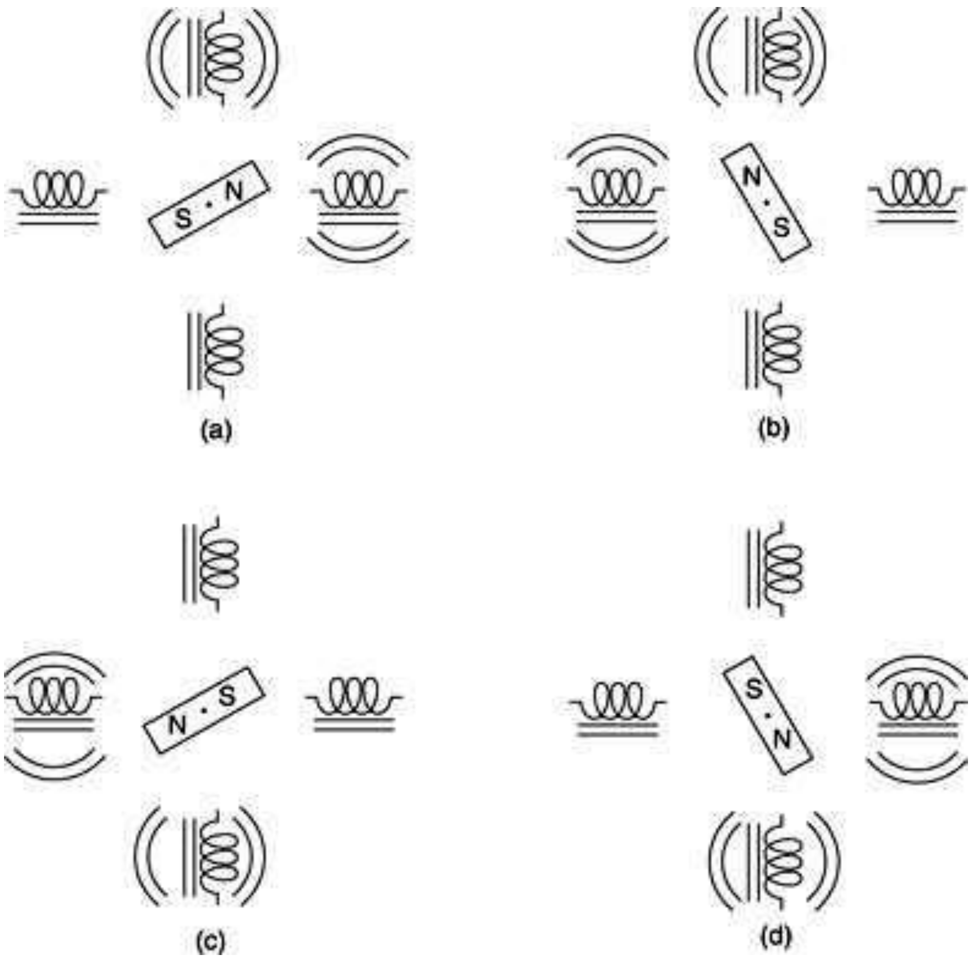
- If you ever played with magnets you will understand how it works.
- Just activate the magnets in a precise order.

- For example, the code below will do three clock-wise half-steps:

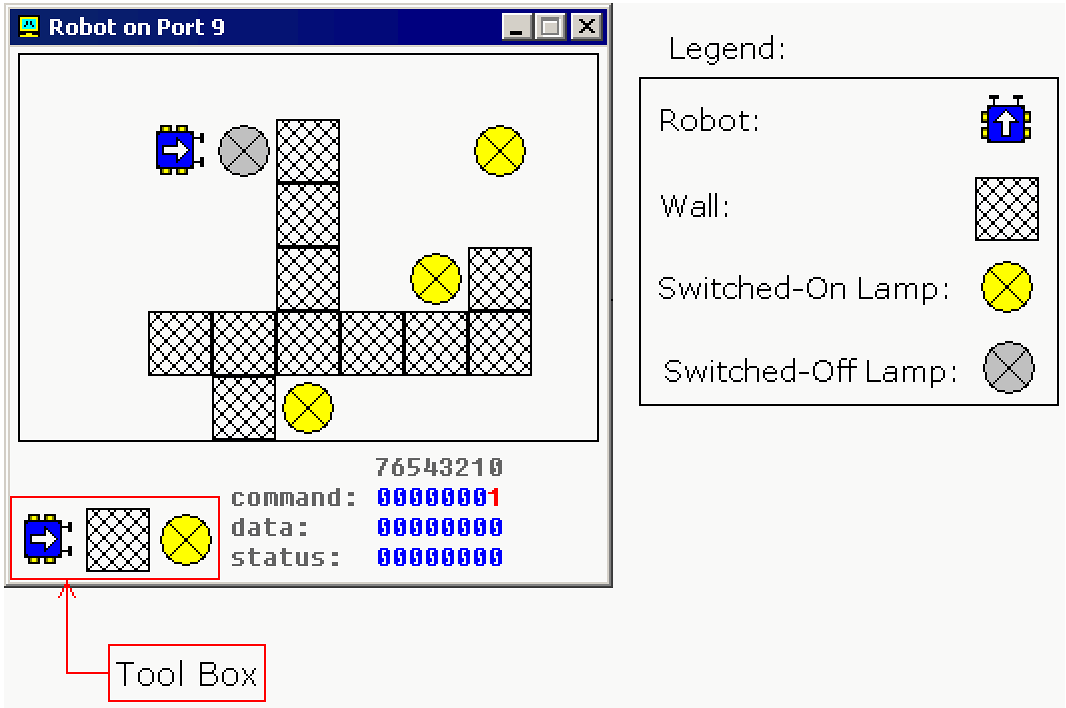
```
MOV AL, 001b  ; initialize
OUT 7, AL
MOV AL, 011b  ; half step 1
OUT 7, AL
MOV AL, 010b  ; half step 2
OUT 7, AL
MOV AL, 110b  ; half step 3
OUT 7, AL
```

Stepper Motor

Four-Coil Stepper Motor that uses an Armature with a Single Pole



Robot



The robot is controlled by sending data to I/O port 9.

Robot

- The first byte (port 9) is a **command register**.
- Send data to this port to make the robot do something.

decimal value	binary value	action
0	00000000	do nothing.
1	00000001	move forward.
2	00000010	turn left.
3	00000011	turn right.
4	00000100	examine. examines an object in front using sensor. when robot completes the task, result is set to **data register** and **bit #0** of **status register** is set to **1**.
5	00000101	switch on a lamp.
6	00000110	switch off a lamp.

Robot

- The second byte (port 10) is a **data register**.
- This register is set after robot completes the **examine** command:

decimal value	binary value	meaning
255	11111111	wall
0	00000000	nothing
7	00000111	switched-on lamp
8	00001000	switched-off lamp

Robot

- The third byte (port 11) is a **status register**.
- Read values from this port to determine the state of the robot.
- Each bit has a specific property:

bit number	description
bit #0	**zero** when there is no new data in **data register**, **one** when there is new data in **data register**.
bit #1	**zero** when robot is ready for next command, **one** when robot is busy doing some task.
bit #2	**zero** when there is no error on last command execution, **one** when there is an error on command execution (when robot cannot complete the task: move, turn, examine, switch on/off lamp).

Robot

- Example:

```
MOV AL, 1 ; move forward.
OUT 9, AL ;
MOV AL, 3 ; turn right.
OUT 9, AL ;
MOV AL, 1 ; move forward.
OUT 9, AL ;
MOV AL, 2 ; turn left.
OUT 9, AL ;
MOV AL, 1 ; move forward.
OUT 9, AL ;
```

Robot

- Keep in mind that robot is a mechanical creature and it takes some time for it to complete a task.
- You should always check bit#1 of **status register** before sending data to port 9.
- Otherwise the robot will reject your command and "**busy!**" will be shown.
- Remember handshaking process!

The Intel Microprocessors: 8086/8088, 80186/80188, 80286, 80386, 80486 Pentium, Pentium Pro Processor, Pentium II, Pentium, 4, and Core2 with 64-bit Extensions Architecture, Programming, and Interfacing, Eighth Edition
Barry B. Brey