Blinking an LED Using PC Parallel Port (NASM + FreeDOS)

Project Goal

In this project, we make a small LED light turn **ON and OFF** using an old PC **parallel port (LPT port)** and **assembly language**.

Normally, Windows blocks direct hardware control. So we use **FreeDOS**, which lets us control hardware directly.

You will learn:

- How parallel port pins work
- How to connect LED safely
- How to write simple assembly code
- How to run it in FreeDOS

Why Not Windows?

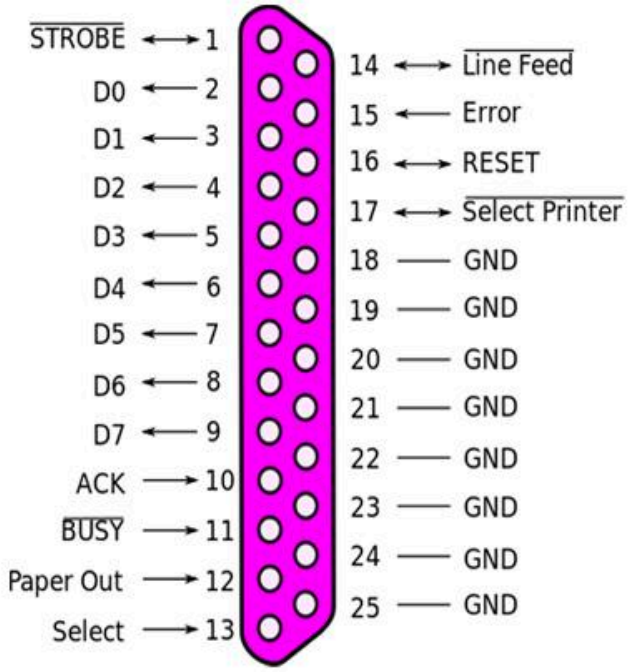
Modern Windows (XP → Windows 11):

- Blocks direct hardware port access
- Needs special drivers
- Sometimes unstable

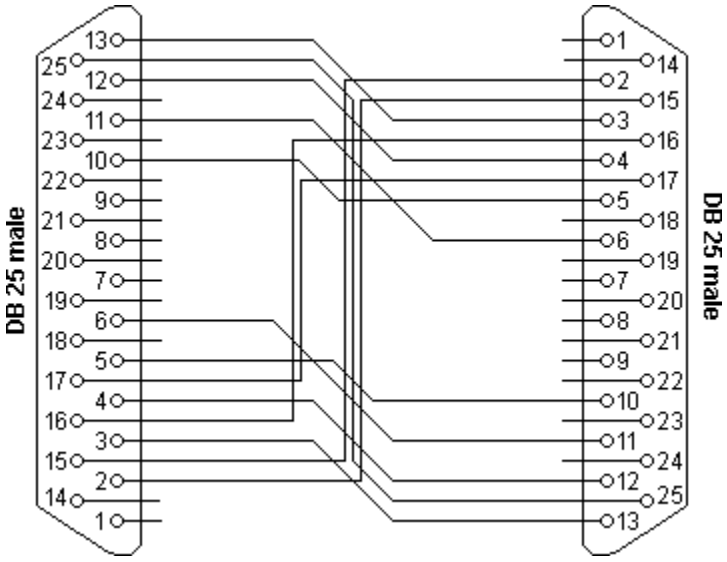
FreeDOS:

- Allows direct hardware control
- No drivers needed
- Best for learning low-level hardware

Understanding the Parallel Port



Pin Number	Pin Name	Pin Description
1	*STROBE*	Data Valadition Signal
2	D0	Data Bit 0
3	D1	Data Bit 1
4	D2	Data Bit 2
5	D3	Data Bit 3
6	D4	Data Bit 4
7	D5	Data Bit 5
8	D6	Data Bit 6
9	D7	Data Bit 7
10	ACK	Acknowledge Status Line
11	*BUSY*	Busy Status
12	Paper Out	Paper Out Status
13	Select	Printer Select Status
14	*Line Feed*	Auto Feed Status
15	Error	Error Signal Status
16	RESET	Reset Line Control
17	*Select Printer*	Print Select Control Line
18	GND	Ground Line
19	GND	Ground Line
'20	GND	Ground Line
21	GND	Ground Line
22	GND	Ground Line
23	GND	Ground Line
24	GND	Ground Line
25	GND	Ground Line



4

Important Points

Parallel port has:

- **8 Data Pins → D0 to D7 (Pins 2–9)**
- **Ground Pins → Example Pin 25**

Computer sends 0 or 1 (OFF or ON voltage).

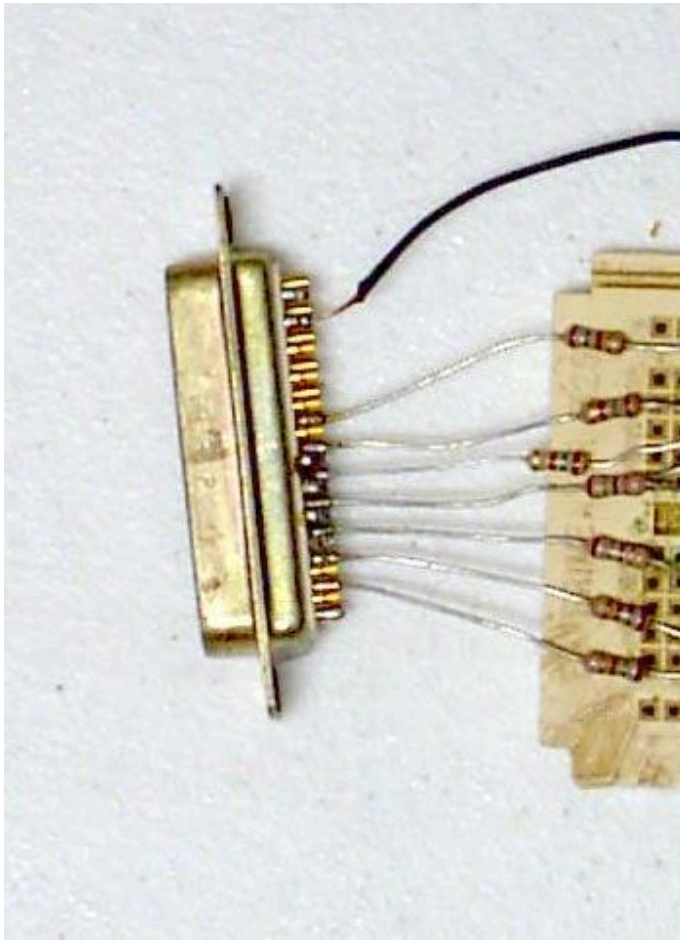
Example:

- Send `1` → Pin gives 5V → LED ON
- Send `0` → Pin gives 0V → LED OFF

Default Address:

```
LPT1 = 0x378
```

LED Circuit (Very Important – Safety)



Parts Needed

- 1 LED
- 1 Resistor (220Ω – 330Ω recommended)
- Parallel port cable / DB25 connector

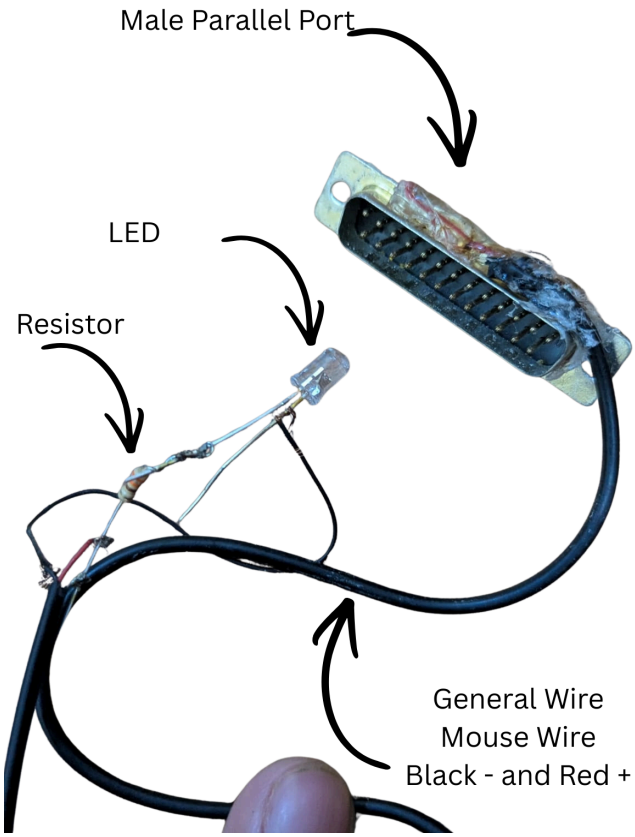
Connection

LED Leg	Connect To
Long Leg (Anode)	Resistor → Pin 2
Short Leg (Cathode)	Ground (Pin 25)

Why Resistor?

Without resistor:

- Too much current
- Can damage port
- Can burn LED



Setting Up FreeDOS + NASM

Step 1 — Install FreeDOS

- Download FreeDOS
- Make bootable USB
- Boot PC into FreeDOS

Step 2 — Install NASM

Copy NASM files into FreeDOS drive:

```
C:\NASM
```

Main file:

```
NASM.EXE
```

Writing Assembly Code

Important Rule

DOS programs start at:

```
org 0x100
```

Simple LED Blink Code

```
org 0x100

mov dx, 0x378     ; LPT1 port address

mov al, 1         ; Turn LED ON
out dx, al
```

```
mov cx, 0FFFFh    ; Delay
delay:
loop delay

mov al, 0         ; Turn LED OFF
out dx, al

mov ah, 4Ch       ; Exit program
int 21h
```

How Code Works (Super Simple)

Code	Meaning
mov dx,378h	Select parallel port
mov al,1	Send ON signal
out dx,al	Send signal to pin
delay loop	Wait some time
mov al,0	Send OFF signal

Compile the Program

In FreeDOS:

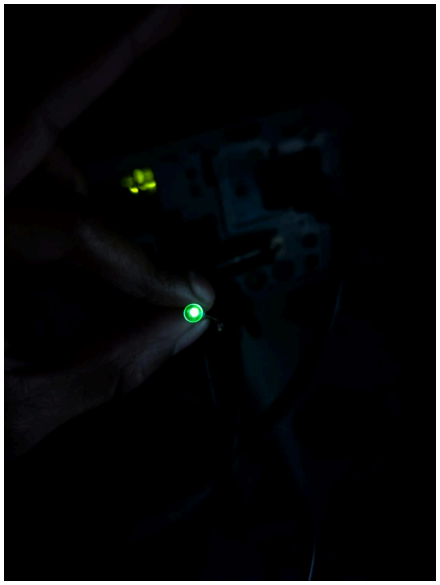
```
nasm -f bin blink.asm -o blink.com
```

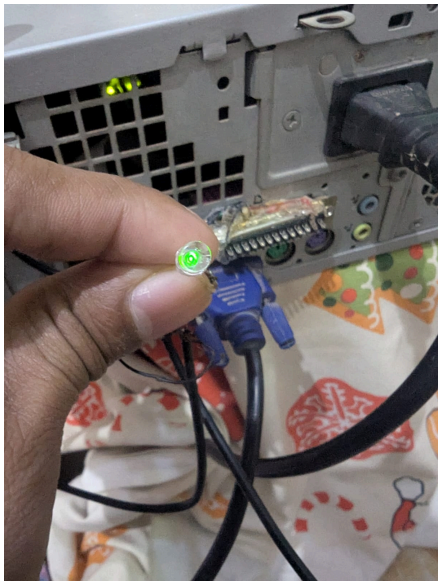
Run Program

```
blink
```

LED should:

- Turn ON
- Wait
- Turn OFF





If LED Not Working

Check:

- Correct pin wiring
- Resistor connected
- Correct port address (0x378)
- FreeDOS booted (not Windows)

What You Learned

You learned:

- Hardware control using assembly
- Parallel port basics
- Real I/O port programming
- DOS assembly program structure

Real Life Use Cases

Same idea used in:

- Industrial control
- Embedded systems
- Microcontrollers (Arduino works same way)
- Old hardware automation

Future Improvements

You can:

- Blink LED continuously
- Control multiple LEDs
- Control relays
- Build small hardware projects

Final Simple Summary

If you remember only 4 things:

1. Use FreeDOS (not Windows)
2. Use port address 0x378
3. Always use resistor with LED
4. Use OUT instruction to control pins