**Lab 12: Input / Output - Parallel Port Operation**

**OBJECTIVE:**

* Getting introduced to parallel port, introduction to pin configuration of the port
* Learning how to address parallel port of computer through assembly and how to write on parallel port and how to read data from any external source
* This lab is being designed to make the students enable to interface the microprocessor to external world and making them enable to control externally interfaced devices by microprocessor

**IN/OUT Instruction:**

|  |  |  |
| --- | --- | --- |
| **IN** | AL, im.byte  AL, DX  AX, im.byte  AX, DX | Input from port into **AL** or **AX**.  Second operand is a port number. If required to access port  number over 255 - **DX** register should be used.  Example:  IN AX, 4 ; get status of traffic lights.  IN AL, 7 ; get status of stepper-motor. |
| **OUT** | im.byte, AL  im.byte, AX  DX, AL  DX, AX | Output from **AL** or **AX** to port.  First operand is a port number. If required to access port  number over 255 - **DX** register should be used.  Example:  MOV AX, 0FFFh ; Turn on all  OUT 4, AX ; traffic lights.  MOV AL, 100b ; Turn on the third  OUT 7, AL ; magnet of the stepper-motor. |

**EQUIPMENT:-**

* SOFTWARE:

1. Turbo assembler (TASM) or Microsoft Assembler(MASM)

* HARDWARE:

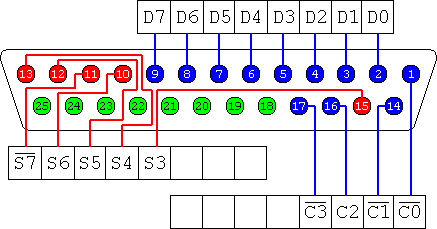
1. Computer with parallel port
2. Parallel cable
3. 8 LEDs
4. 8x1KΩ Resistances

**INTRODUCTION TO PARALLEL PORT:**

PC parallel pot is 25 pin D shaped female connector usually used for connecting printer and other devices. The pins are classified into following categories:

1. 8 output pins accessed via the DATA PORT
2. 5 input pins one inverted accessed via status port
3. 4 output pins (3 inverted) accessed via control port
4. Other 8 pins are grounded

This is the basic description of the parallel port which specifies the basic pin configuration of the parallel port which gives an idea about how to use the parallel port. You can see the pin configuration in the figure below:



The output of the parallel port is normally TTL level. The currents the pins can sink is different for different ports which can vary from 12mA to 20 mA.

**THE OUT COMMAND:**

The OUT command is used to write the data on parallel port. This command has the following syntax:

OUT dx, ax

The dx should contain the address of parallel port data register. The address of data pins of parallel port is **378h,** so you should movethis address todx. The above command sends the contents of AX to the address contained in DX. So you should use this pair of commands to write data on parallel port.

Mov dx, 378h

Out dx, ax

**ROUTINE TO GENERATE A DELAY:**

When generating different patterns you need to have a specific time delay in between different patterns. To generate a small delay you can write the function:

delay proc

mov dx,15000

l1:

dec dx

mov bx,0ffffh

l2:

dec bx

cmp bx,0

jne l2

cmp dx,0

jne l1

ret

delay endp

**THE CALL AND RET STATEMENT:**

The return command here is used to transfer back the control back to the calling procedure. Every procedure (except main procedure should have a RET statement at some place (usually the last statement). To invoke a procedure CALL instruction is used. The syntax is

CALL NAME

Executing CALL statement causes the following to happen:

1. The return address to the calling program is saved on the stack. This is the offset address of the next instruction after call statement.
2. IP gets the offset address of first instruction of the procedure which transfers the control to the procedure
3. To return the control back RET statement is used

**ROTATE INSTRUCTION:**

The ROL (Rotate left) and ROR (Rotate right) shift the bits to left and right respectively.

You can see in figure how does ROR works.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |

Syntax is

ROR destination,cl

The value of cl would determine how much time the ROR would be executed.

**PROCEDURE:-**

**Writing on parallel port:**

Before starting the procedure here is a sample code to blink all the LEDs.

.model small

.code

main proc

mov cx,5

start1:

mov ax,0ffffh

mov dx,0378h

out dx,ax

call delay

mov ax,0h

mov dx,0378h

out dx,ax

call delay

jmp start1

mov ah,4ch

int 21h

main endp

delay proc

mov dx,15000

l1:

dec dx

mov bx,0ffffh

l2:

dec bx

cmp bx,0

jne l2

cmp dx,0

jne l1

ret

delay endp

end main

* Connect the positive ends of LEDs to data pins (pin 2 to 9) and negative ends to resistances
* Ground the other end of resistances i.e. connect to any other pin (from pin 18 to 25)
* Assemble the code and verify the pattern you sent on the port

**EXERCISES:**

1. Turn on all the LEDs
2. Create an LED pattern by turning on LEDs towards left
3. Create an LED pattern by turning on LEDs towards right
4. Create an LED pattern by just blinking odd LEDs
5. Write the codes in specified blank space

Insert a proper delay between the patterns. Don’t connect more than one LED with a single pin of parallel port, because it can sink only a current up to 20mA.