1. Introduction
Serial ports can be used to communicate with devices and with other computers. In this lab you will learn serial port communication and how to program serial port controller. In addition to that you will learn how to use dos interrupts.

1.1. Materials and Equipment
You will work on 80x86 pc that has an 8250 UART. You will be given a null modem cable. In addition to that you will need a floppy disk. You have to copy MS-DOS system files and Borland TASM assembler on the disk, which is available as an image file on the course web page.

1.2 Objectives
At the end of the following experiments you will be able to program 8250 UART, transmit and receive data from the serial port. Use MS-DOS software interrupts to do system programming.

2. Preliminary Work
In the following sections you will be assigned some preliminary work that you should perform before lab sections. You should submit the source codes of the programs that you have written and also a document that contains the steps of the algorithms that you have used for 2.1, 2.2 and 2.3. The details of the submission will be announced on the newsgroup. What you submit must be definitely your own work. You cannot use any library for the x86 assembly language, also you cannot use bios or MS-DOS software interrupts for the serial port (you can use and should use MS-DOS interrupts for other operations). Also it is forbidden to use the source codes in the book “Art of Assembly programming”, since they are the codes of the standard library developed by Randall Hyde.

2.1. Initialization of the 8250 UART
Write an assembly program for initializing COM 1 port 14400 baud, 7 data bits, 2 stop bits and odd parity.

2.2. Implementing Serial Communications
Write an assembly program which sends the character read from keyboard to COM1 and displays any character received from the serial port using interrupts, polling is not allowed and will be given no credits.
2.3. Tic-Tac-Toe

In this part you will implement the game tic-tac-toe (known as “X-O” in Turkey). On the text screen you will draw the tic-tac-toe screen and the users will play the game on the command line you supply. One of the users will play on one computer and the other user will play on the connected computer. The board situation should be updated on both of the computers.

In the beginning you will clean the screen and draw the board in the center of the screen and at the bottom line you will supply a command line saying “Command ->”. The numbering of the board positions is given in figure 1.

```
 1 2 3
 4 5 6
 7 8 9
```

Figure 1. Cell numbering on the board

Each user can play his/her move by writing his/her sign and the position on the board. For example, “X1” command puts the “X” sign on the position “1” on the board and side the message “#X1#” to the other side. When a user wants to play a move when it is not his/her turn you should give the message, “It is not your turn!” on the screen. When a user enters an illegal command you should give the message “Illegal command!” on the screen. The users can exit the game by typing “bye”. An exit message “#exit#” should be sent to the other side.

When a user wins the game the message, “Congratulations!” should be printed on the winner side, “<sign> has won the game!” (X or O should be printed with respect to the winner sign for the <sign>), on the other side. The game may end with draw (when there is no place to play and no winner), in this case you should print “Draw!” . After the game ends you should ask the user if he/she wants to play again by the message, “Do you want to play again? (y/n)”. According to the response of the user the program ends (if one of the users enters ‘n’), in this case an exit message should be sent to the others side (with the message “#exit#”). If the users want to play again (both of them should respond with ‘y’), the screen is cleared and the game begins again.

The sign of the users are selected at the beginning, the first user playing determines the sign, thus, if the first user plays with ‘X’ at the beginning that user is assigned ‘X’, if he/she begins with ‘O’ that user is assigned ‘O’.

You should do all the communication through the ‘COM 2’ port with the parameters that are given in 2.1. The interrupt method should be used, polling is not allowed and will be given no credits.

The address of the screen memory of the first text screen starts from the address B800:0000.

3. Experimental Work

Each student must bring a floppy disk that contains an MS-DOS system, an editor, and Borland TASM assembler. The source codes of the programs that you have written should also be included in the disk. You must also bring a printed copy of the document about what you have done with you.
3.1. Part 1
You should run the program that you have implemented in 2.1.

3.2. Part 2
You should run the program that you have implemented in 2.2.

3.3. Part 3
You should run the program that you have implemented in 2.3

4. Lab Rules
- Those without any preliminary work will not be allowed to attend the lab.
- Those who have cheated in preliminary, quiz or other lab work get zero for both labs.
- Those who are more than 20 minutes late will not be allowed to attend the lab session.
- Make-up labs require written official excuse. (e.g. from Health Center)
- Collaboration between lab groups is forbidden.
- The assistant may ask questions to each individual about the details of the experiment, which are also graded.
- At the beginning of the lab session, there will be an open-book-and-notes lab quiz, which is strictly fifteen (15) minutes long.
- Eating, drinking and smoking are forbidden.

5. Resources
- Microprocessors and Interfacing, Hall: Chapter 14.
- Art of Assembly Programming, Randall Hyde (Chapter 13, 22, and 23 is very useful), available at: http://webster.cs.ucr.edu/AoA/DOS/index.html
- Some MS-DOS software interrupts that will be available on the course page.