Cairo University

Faculty of Engineering

Computer Engineering Department

(CMP2010/ CMPN201) Microprocessors Systems I

Project Description

Introduction

This section presents an overview of the project requirements and constraints. Specific details are discussed later. It is required to connect 2 PCs through a Simple network, using serial communication. Two functions are to be implemented: chatting, and a two-player real-time chess game.

This is an assembly language project; hence you are only allowed to use the console window for your application. You are allowed to use text/graphics mode GUI. To enhance the graphical presentation, you must make use of the video-RAM functionalities. You can use text mode attributes, such as character attributes (for foreground and background colouring), and special ASCII characters (for organising the screen). But note that a part of the grading is: how you present your results.

The rest of this document describes the exact details of the functional requirements and some guidelines for their implementations. In addition, grading criteria hints are given to help you get the highest possible mark in case of not completing the full list of requirements. Please read it very carefully. For any inquiries, please return to the TAs through their emails.

Functional Requirements

This section divides the project requirements into a set of functionalities. Each function is described separately, and then we provide an overall system description that combines all functions.

Connection Mechanism

Two PCs are connected through the serial port. You will probably need to use only two signals of the serial port: Transmit (Tx) and Receive (Rx), in addition to the ground. Each Transmitter of one PC is connected to the receiver of the other PC and vice versa. The required cable will be like the one you used in the serial communication lab.

Defining Usernames

The program should ask the user for suggested initial points and his/her username to use while chatting or playing with the other user. The username should not exceed 15 characters and start with a letter (*No digits or special characters*). This should be done at the beginning of the program. In other words, the first screen at each terminal should display something like this:

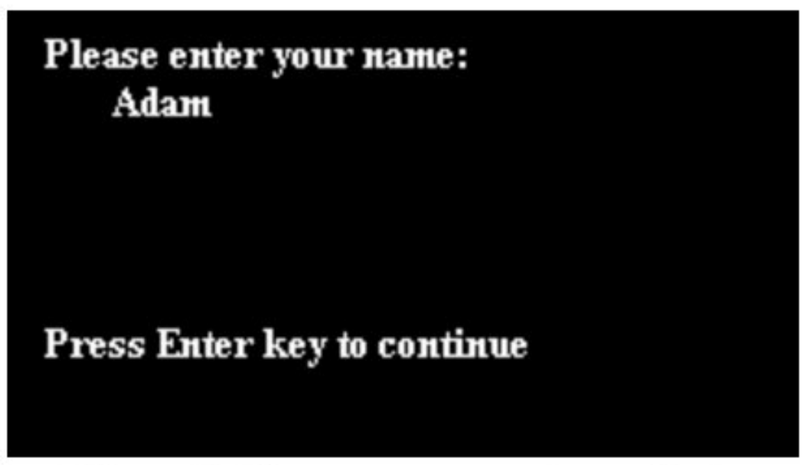

Figure 1: First screen at each terminal.

After both users enter their names, users should exchange names so that each user could know the other user's name.

Main screen

After allowing each user to enter his/her username, the main screen should appear with a list of available functionalities and how to navigate to each of them. Also, after exiting any of the functionalities, this screen should appear to wait for the next action of the user. Figure 2 is a simple example of the main screen.

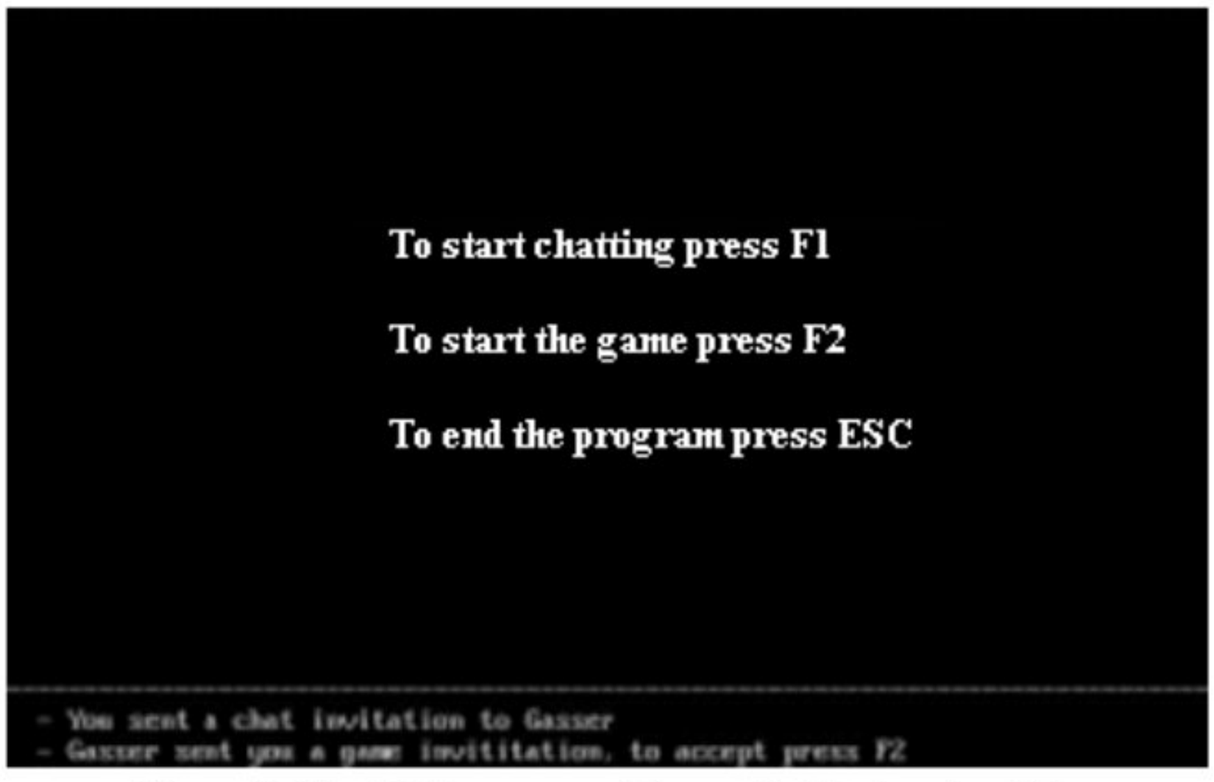

Figure 2: The Main screen of the available functionalities

The lower two or three lines should be dedicated to the notification bar. The notification bar should view any notifications for the user, i.e. game invitation is received from the other user, or a chat invitation is sent to the other user.

Chatting

In this part, the users should be able to chat with each other. The screen should be divided into two halves. For example, as in Figure 3, the first half is for showing data written by the current user, and the other is for showing data sent by the second user across the network. Scrolling functionality should be provided.

Figure 3: Chatting Window

Chatting Mode Scenario

This section provides a simple description of the main scenario that should be followed in the chatting mode.

1. If a user wants to chat with the other user, (s) he should press F1 to send a chat invitation. This invitation should appear on both machines in the notification bar on the main screen, below the main menu.

2. Until the other user accepts the chat invitation, both users should remain on the main screen.

3. To accept the invitation, the other user should press F1. In that case, both users should enter the chatting mode.

4. Both users remain in chatting mode till one of the users presses F3. In that case, both programs should return to the main screen waiting for another choice from the users. The chat invitations should disappear now from the notification bar. Any other notifications should be restored.

The Real-time Chess Game

In this part, the users should be able to play a two-player real-time chess game. The final target for each player is to win the chess game. The game is a popular Android game titled "Chezz: Play Fast Chess". In this project, you will implement a version like it. Each player can move his pieces to win the game. The goal is to kill your opponent's king.

The game is like any standard chess, each piece moves exactly like standard chess. The main difference is that there are no turns, which means one player can move multiple pieces while the other player didn't move any pieces. There is a countdown for each piece to be moved again, for example, if I moved the queen now, I can't move the same queen again for 2 seconds although I can move any other piece at this time.

The game should be on the whole screen leaving the lower part of the screen for inline chatting, usernames, and game time. The main scenario and a detailed description of the game are provided in the following sections.

Figure 4: part of the game screen

Game Mode Scenario

1. If a user wants to play with the other user across the network, (s) he should press F2 to send a game invitation. This invitation should appear on both machines in the notification bar on the main screen, below the main menu.

2. Until the other user accepts the game invitation, both programs should remain on the main screen.

3. To accept the invitation, the other user should press F2. In this case, both users should enter a new game.

4. At the beginning of the game:
   - The chessboard will be set and each piece in its place like the standard chess
   - The first player (who sent the invitation) should take the white colour, and the other player the black colour
   - Start the game timer

5. During the game:
   - To move a piece, each player will have five keys. (for example, A, S, D, and W for move Left, Down, Right, and Up, and Q for click)
     - If the player presses left then the next piece on the left should be highlighted, same for Down, Right, and Up. All of them are used to navigate through the player pieces.
     - If the player presses Q (click), then the player wants to move the highlighted piece so you should highlight all the available moves for this piece.
       - Then the player should use navigation arrows to navigate through the available moves of

the selected (clicked) piece
    - Then the player should press Q again to move the selected piece to the highlighted positions
    - Then this piece should have a countdown (starts from 3 seconds), while the countdown this piece should not be able to move, but of course, any other piece could be moved
    - If the new position has an opponent piece, then this piece is killed (removed from the board)
- Unlike standard chess games, the winner in this game should kill the king of the other player not checkmate it.
- [Bonus] Implement a powerup that appears randomly on the chessboard, any player who takes this powerup (by moving one of his pieces to this position), should have a smaller countdown for each of his pieces (will start from 2 seconds instead of 3)
- [Bonus] Implement the batch mode. In this mode, any player could write a batch of moves to be executed in sequence. The batch could be entered by entering the block number of the a peice location followed by the number of target location. For example, 0123456230 the first 0 is to enter writing a batch. The first move is to move the piece at row 1 and column 2 to the block at row 3 and column 4. This is followed by moving the piece at (5,6) to the block (2,3). The last 0 is used to end the batch and execute it. If the batch contains moving a piece to an invalid location then this piece is killed(removed) . If the batch contains an invalid move that tries to move from an empty block or the opponent piece then a random piece except the king is removed.
- The previous two bonus features are obligatory for teams of five members.

6. Both users remain in the game mode unless the game ends (one king has been killed), or one of the users presses F4; in that case, both users should return to the main screen waiting for another action from the users.

Summary

This section tries to connect all the previous components into one fully integrated system in a group of points:

Users have to define their names to other users.

When a user decides to start a chatting session or a new game, (s) he presses F1 or F2. Thus, the system operates according to the scenarios mentioned in each section.

If a user wishes to quit the program, he/she could press ESC. A quit is only accepted when the user is in the Main screen mode. When one user quits, the program must send the ESC to the other user.

Project Phases

The project is divided into two phases:

Phase 1: Two Players Game on one PC

An initial version of the game without a communication module should be delivered

Phase 2: Full project delivery

A complete version of your projects must be submitted to the classroom.

Guidelines

In this section, we present some helpful guidelines for the implementation. You are not obligated to follow it; you are free to design your own alternatives.

For good and easy message handling, design all your messages to have one fixed size.

You must very carefully organise your program using procedures and labels, as necessary, for easier code maintenance and bug tracking.

State machines are one of the best counters in designing programs of this type. Try designing your program as a state machine.
Try to think of the program as a C/C++ program, then convert all high-level languages to their corresponding low-level language.

When dealing with serial communication, it is never a good trend to start working with your program on an emulator. Emulators have their assumptions, which do not map to reality, and thus may lead to incorrect programs when they are compiled and linked using the MS assembler and linker.

Try delivering complete atomic functions. For example, you could start by implementing the main screen and the complete user interface function only, without the game functionality. When you are 100% sure it is working fine, take a snapshot of that program, then move on to implementing the game module. Delivering one function correctly working will be graded higher than not delivering any function working correctly!

Delivering concrete requirements is awarded more than partially incomplete ones. You can consider the set concrete requirements (and corresponding grading criteria) as follows:

- Chatting module
- The game itself
- Inline game chatting
- User interface
- Code organization (ease or code reading)

Honor System

READ CAREFULLY. Any identified cheating of any type is simply graded a big **ZERO. Taking part or full project from another team or someone else is graded with the negative full mark**. This project is a teamwork project, but in the end, it is academic material, hence all team members are expected to receive similar grades, based on the member with the least knowledge. On the project delivery day, any question could be asked to any team member at random.