Cairo University

Faculty of Engineering

Computer Engineering Department

(CMP2010) 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 players' processor simulation 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 coloring), and special ASCII characters (for organizing the screen). But note that a part of the grading is: how you present your results?

The rest of this document describes exact details of the functional requirements and some guidelines to 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

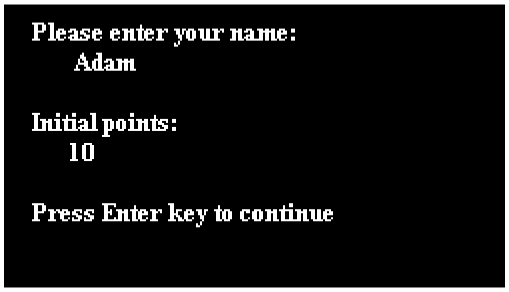
In this section, we divide the project requirements into a set of functionalities. Each function is described separately, and then we provide an overall system description that combines all functions together.

Connection Mechanism

Two PCs are connected together 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 as 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 it 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 the something like:



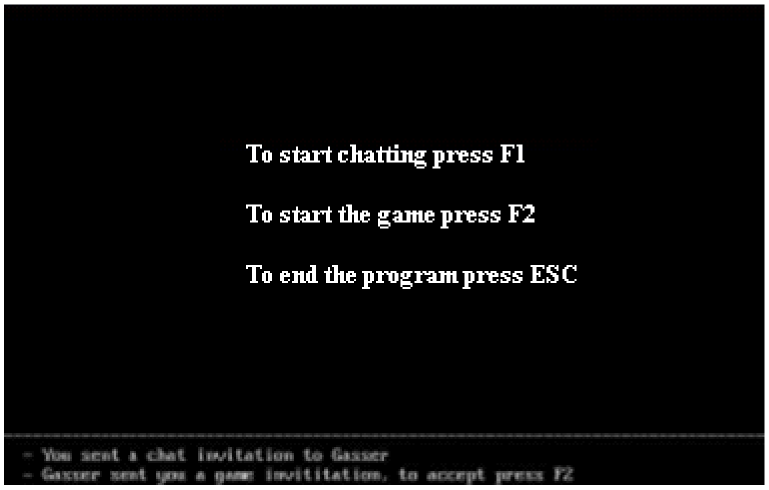
```
Please enter your name:
     Adam

Initial points:
    10

Press Enter key to continue
```

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.



```
To start chatting press F1

To start the game press F2

To end the program press ESC

--------------------------------------------------------------------------------
- You sent a chat invitation to Gasser
- Gasser sent you a game invititation, to accept press F2
```

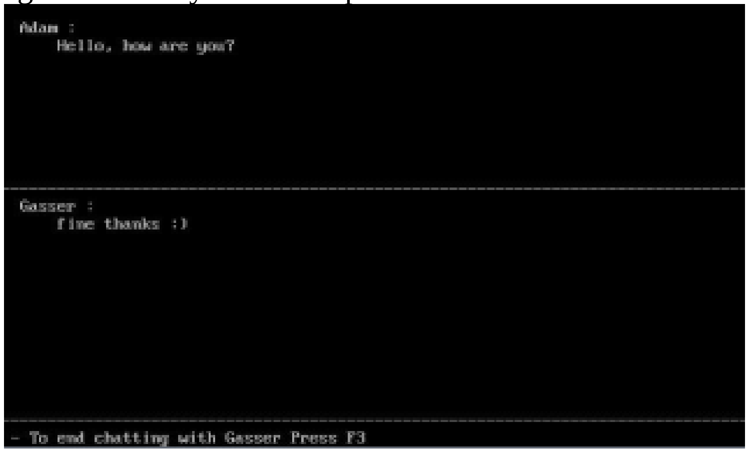
\

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 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.



```
Adam :
    Hello, how are you?

--------------------------------------------------------------------------------
Gasser :
    fine thanks :)

--------------------------------------------------------------------------------
- To end chatting with Gasser Press F3
```

Figure 3: Chatting Window

Chatting Mode Scenario

This section provides a simple description to 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 in the main screen, below the main menu.
2. Until the other user accepts the chat invitation, both users should remain in 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 the 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 by now from the notification bar. Any other notifications should be restored.

The Car Racing Game

Develop a two-player car racing game in Assembly 8086 where players compete to reach the end of the track before their opponent. The game should allow each player to control their car using arrow keys, navigate through a track with mystical obstacles, and strategically utilize special power-ups to gain an edge in the race.

The game should be in the whole screen leaving the lower part of screen for inline chatting, usernames, scores (percentage of track) and available power ups. The main scenario and a detailed description of the game are provided in the following sections.

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 in 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:
   - Generate a racing track randomly, ensuring it features multiple right and left turns. The track width should be a multiple of three times the car width.
   - Randomly place obstacles and power-ups throughout the track.
   - Both players should commence from an identical starting point, positioned side by side. The endpoint of the track should be marked with a distinct color.
   - Each player starts with the same speed and without any power-ups.
   - Initiate the game with a 2-minute timer.
5. During the game:
   - Players control their cars using the four arrow keys.
   - Both players must remain visible on the screen, regardless of their positions.

- Additional power-ups should randomly appear on the track during the game.
- Upon obtaining a power-up, it should be displayed in the status bar. Players can activate the power-up using a designated key. Various power-up types include:
  - Increase the speed of the player for the next 5 seconds.
  - Decrease the speed of the other player for the next 5 seconds.
  - Generate an obstacle behind the player.
  - Enable the player to pass through one obstacle.
- The game concludes under the following conditions:
  - One player (winner) reaches the goal (end of the track).
  - The game timer reaches zero (both players lose).

Feel free to enhance the details or add any additional features that would enhance the gaming experience.

5. Both users remain in the game mode unless one of the users presses F4; in that case, a screen of the scores should appear for 5 seconds and 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 three phases:

Phase 1: Two Players Game at one PC

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

Phase 2: Full project delivery

A complete version of your projects must be submitted by email. **Email title should be in the form [CMP201A- Group Number**] with brackets**.** Project discussion will be scheduled later.

Guidelines

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

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

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

State machines are one of the best counters in designing program of this type. Try designing your program as a state machine.

Try to think of the program as a C/C++ program, and then convert all high-level language 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 own 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**. This project is a teamwork project, but at the end it is an academic material, hence all team members are expected to receive similar grades, but on the basis of the member of the least knowledge. On the project delivery day, any question could be asked to any team member at random.