Cairo University

Faculty of Engineering

Computer Engineering Department
2nd Year

CMP201-A Microprocessors Systems I

Project Description

Fall 2019

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 any two players 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

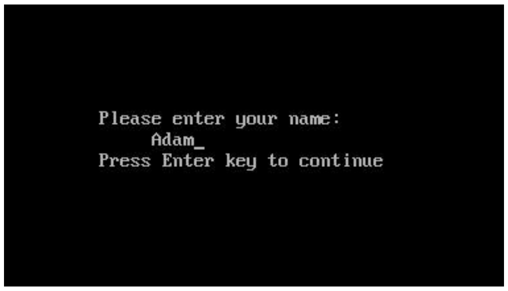
In this section, we divide the project requirements into a set of functionalities. Each function is described separately, and then we provide an overall system description that combines all functions together.

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 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:



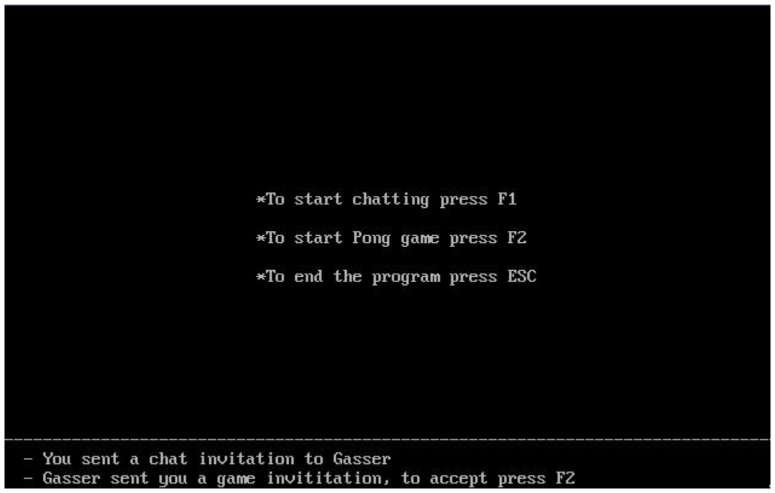
```
Please enter your name:
    Adam_
Press Enter key to continue
```

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.



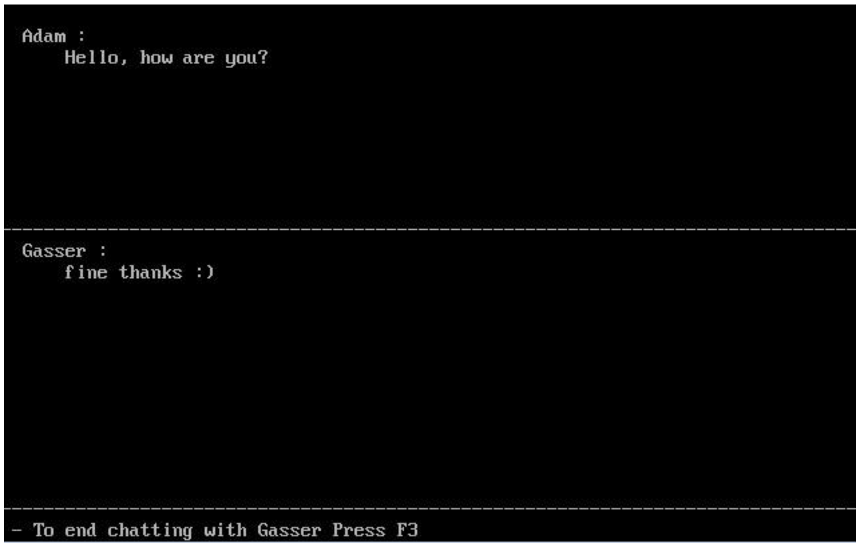
```
*To start chatting press F1

*To start Pong 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 Game (For example, Pong Game)

In this part, the users should be able to play a two players game. For example, Pong game. Pong is a two-dimensional sports game that simulates table tennis. The player controls a paddle by moving it vertically across the left side of the screen, and can compete against another player controlling a second paddle on the opposing side. Players use the paddles to hit a ball back and forth. The aim is for each player to reach five points before the opponent; a point is earned when the other user fails to return the ball.

The game should be in the whole screen leaving the last 1/5 part of screen for inline chatting, usernames and scores. See Figure 4. The main scenario and a detailed description of the game are provided in the following sections.

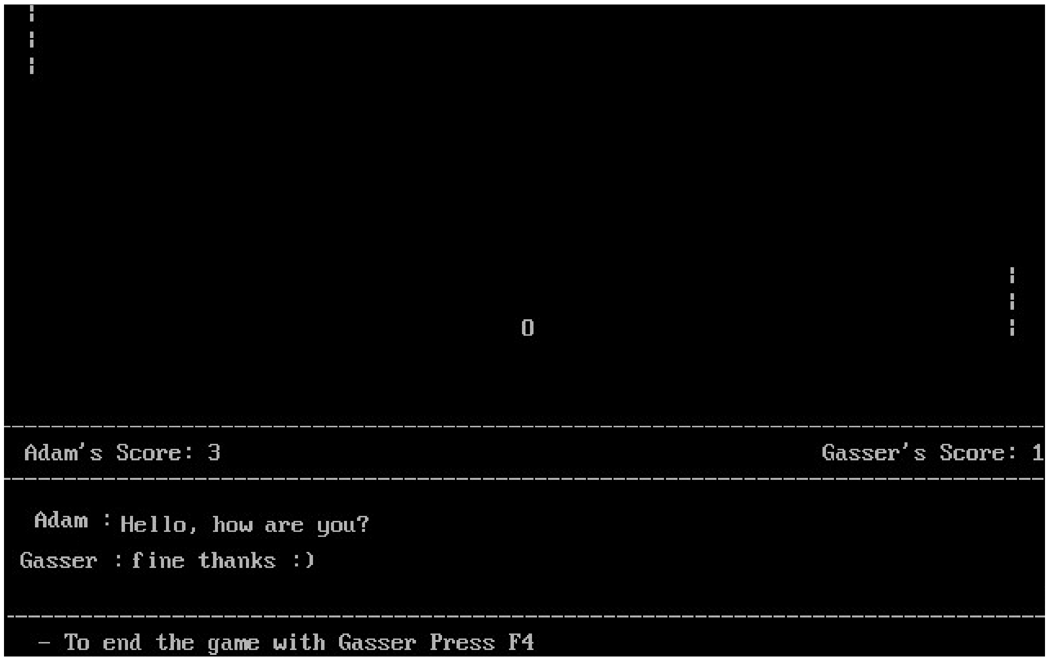

```
0

Adam's Score: 3                                    Gasser's Score: 1

 Adam : Hello, how are you?
Gasser : fine thanks :)

 - To end the game with Gasser Press F4
```

Figure 4: Pong Game

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 accept the game invitation, both programs should remain in 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, only the player who initiated the game should be asked to decide the level of the game. Only two levels should be available. For level 1, user should press 1 and for level 2, user should press 2. While the first player is picking the game level, the other player should see a static screen of the new game. For the pong game, paddles are placed at their initial positions and the player should not be able to move his paddle.

   After selecting the level of the game, the other player should be notified with the chosen level. After that, the game starts.

   Game starting positions are based on who initiated the game. For example, the user who initiated the game will have his paddle at the left side of the screen. The other user will have his paddle at the right side of the screen. The initial positions of the paddles rows are 10, 11, 12, each paddle occupy vertically three rows. The left and the right paddles are positioned at columns 2 and 77 respectively. The initial position of the ball is at (11, 3).

Different levels should be handled by controlling the speed of the game or the game complexity (based on your game). This can be done by controlling the rate of updating the game frame. The rate of updating the game frame can be controlled by either adding loops to slow down the update rate, or updating the game every ***t*** seconds.

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.

Pong Game Flow

The game flow is described as follows:

1. The player who initiated the game gets to serve first.
2. To serve the ball for the first time, the player should press the space bar.
3. The ball starts in front the paddle of the player who serves first.
4. The ball moves toward the next player.
5. To bounce the ball with the paddle, the player can move the paddle up and down using the Up/Down arrows.
6. The ball bounces with the paddle of each player until one of the players score.
7. To score a point, a player should hit the ball against the opponent's wall.
8. If a player scores a point, his/her score, at the bottom of the screen, should be updated. The player, who did not score, serves next.
9. The player wins when he/she scores 5 points.

Game Controls

This section lists the controls that can be used in game mode.

1. Press space bar to serve a ball for the first time.
2. Up and down arrows are used to move paddle up and down.
3. Press F4 to return from game mode.
4. Typing any character is used for in-game chat while in game mode. Based on your game, it may need to be paused. Pong game for example, need the paddles to be paused while chatting.

The Players and the Ball Shapes

1. Each player is a 3 parts paddle, upper, middle and lower. Each part is drawn to the screen using the character **'|'.**
2. The ball is drawn to the screen using the capital character **'O'**.

Players Communication

This section describes how the two users communicate with each other, which one gets to control the flow, and what the exchanged data is.

1. The player serving the ball should be the one responsible for updating the whole game. The player should receive the opponent's position and use it along with both his own position and the ball position to do the following logic.
   a. The player should calculate the ball position each frame (this requires calculating and knowing the incidence and reflection angles of the bouncing ball).
   b. The player should detect if one of the players scores a point, update the scores and send the updated score to the opponent. The player who did not score gets to serve the ball next. Whenever a player serves a ball, he/she becomes responsible for updating the game.
   c. The player should detect if one of the players won the game after scoring a point and notify the opponent to view the results and end the game mode.
2. With every update step, the starting player should send his own paddle position and the ball position. The other player should send back his position only.
3. Every game frame should be bounded by an update packet; the opponent player should not update his position until he receives the new update packet.

Ball Bouncing

This section illustrates details about how to handle bouncing the ball in its different states.

1. The ball starts with a random angle.

   You can use this simple function to get a random angle:

   ```
   R = current seconds % 3
    if (R== 0) then angle = 45°
   else if (R== 1) then angle = 0°
   else angle = -45°
   ```
2. When the ball hits *the middle of <u>the paddle</u>*, the velocity in the X direction is inverted to be –X.
3. When the ball hits *the upper or the lower walls of <u>the screen</u>*, the velocity in the Y direction is inverted to be –Y.
4. The game deals with only 5 angles which are 30°, 45°, 0°, -45°, -30°
5. If the ball hits *the upper or the lower parts of <u>the paddle</u>*, the reflection angle depends on the angle of incidence as follows.
   a. If it is 45° or -45°, it should be reflected as -30° or 30°, respectively.
   b. If it is 30° or -30°, it should be reflected as 0°.
   c. If it is 0°, it should be reflected as 45° if the ball bounces off the upper part or as -45° if the ball bounces off the lower part.

See Figure 5 for more illustration, note that the angle between the solid arrow and the horizontal line is the angle of incidence, and the angle between the dashed line and the horizontal line is the angle of reflection.

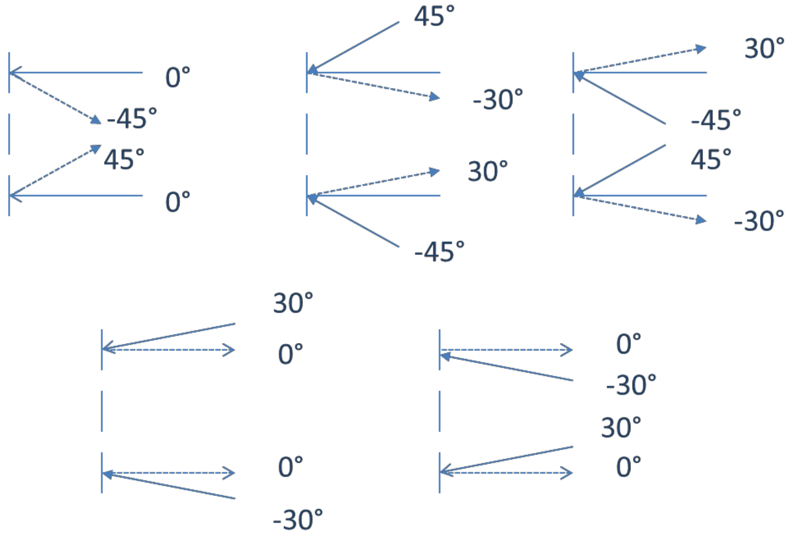


Figure 5: Incidence Reflection angles relationship

Reflection Angles Mapping

This section illustrates how the previously mentioned reflection angles are mapped into movement on the screen. To simulate the effect of reflecting the ball with different reflection angles, we need to translate this reflection into movement in X and Y directions.

So here is a list of those mappings.

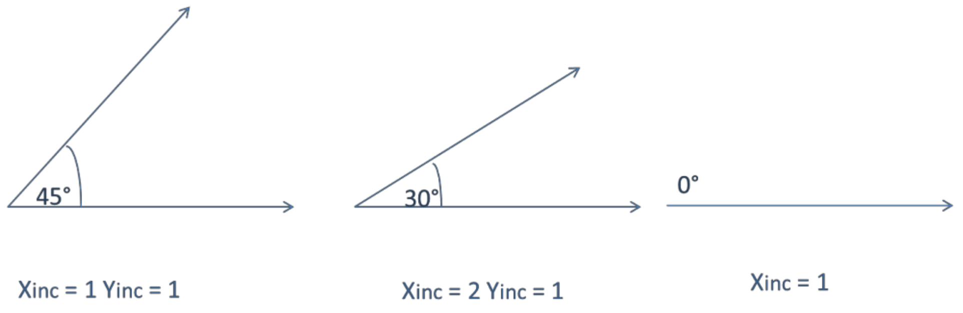


Figure 6: Reflection Angles Mapping

1. If the reflection angle is 45° then increment the ball positions by 1 in both X and Y directions.
2. If the reflection angle is -45° then increment the ball positions by 1 in X direction and decrement the ball position by 1 in Y directions.
3. If the reflection angle is 30° then increment the ball positions by 2 in X direction and by 1 in Y direction.

4. If the reflection angle is -30° then increment the ball positions by 2 in X direction and decrement it by 1 in Y direction.
5. If the reflection angle is 0° then increment the ball positions by 1 in both X direction.

The previous mappings are used when the ball hits the left paddle. For bouncing the ball off the right paddle mappings can be easily derived from the previous mappings.

Those mappings are illustrated in Figure 6.

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: Team forming and game selection

Project groupsconsists of 4 members. Each group should choose a two players game. Send your teams and your selected games. You are allowed to choose any game, and wait for acceptance. Groups with the same game will assumed to be competitors.

Phase 2: Two Players Game at one PC

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

Phase 3: Full project delivery

A complete version of your projects must be submitted by email
Email title should be in the form [CMP201A- Group Number – Game title] 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.