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 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

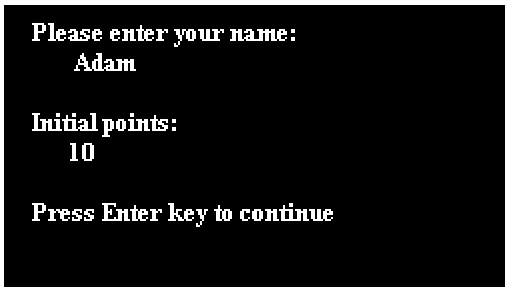
In this section, we divide the project requirements into a set of functionalities. Each function is described separately, and then we provide an overall system description that combines all functions together.

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:



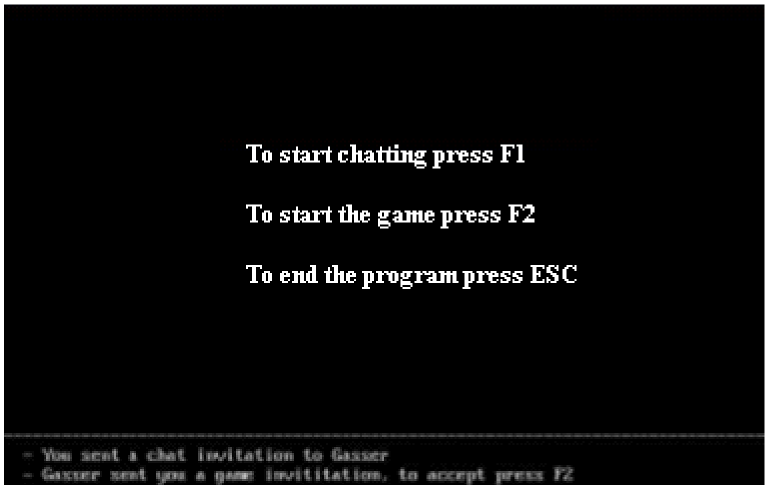
```
Please enter your name:
     Adam

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
```

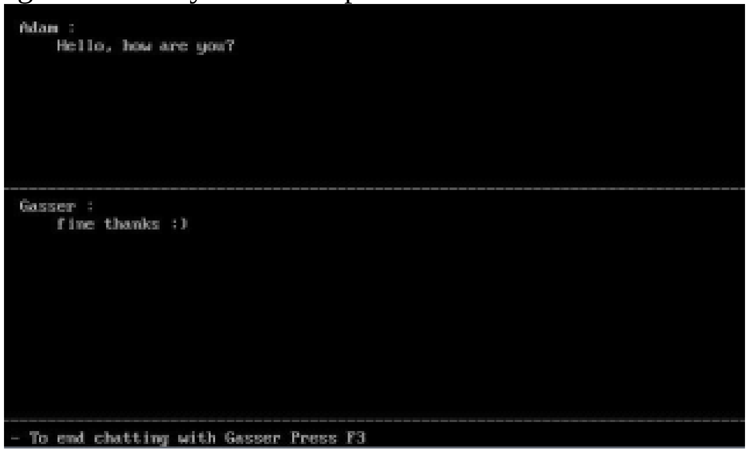
\

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

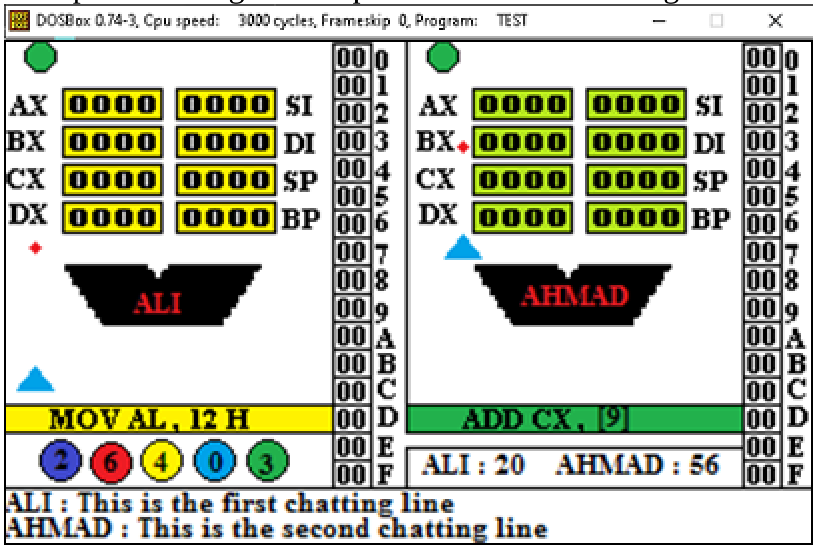
This section provides a simple description to the main scenario that should be followed in the chatting mode.

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 Processor Simulation Game

In this part, the users should be able to play a two player's processor simulation game. The final target for each player is to put certain value in one of the opponent registers. Each player writes a command that executes on the opponent processor and could use special types of power up to prevent the opponent from reaching to the required value.

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


DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: TEST
AX
BX
CX
DX
SI
DI
SP
BP
0000
ALI
AHMAD
00
0 1 2 3 4 5 6 7 8 9 A B C D E F
2 6 4 0 3

MOV AL, 12 H

ADD CX, [9]

ALI : 20 AHMAD : 56

ALI : This is the first chatting line
AHMAD : This is the second chatting line

Figure 4: Ali's main 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 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, each player should send its initial points to the opponent. The lower number will be used for both of them as initial points and 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, the 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.

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

   - Each player should choose a forbidden alphanumeric character (0 – 9 A-Z) that the opponent can't use for any command and press F4 when ready to start the game.
   - The player who sent the invitation starts with the first command. The command written by the first player executes on the other player processor. The winner is the one who is able to put the value "105e" in any of the opponent registers.
   - The opponent should prevent this by the initial forbidden character or special power ups.

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.

Level 2: hidden forbidden character, could select own processor / opponent, initial values for registers

Game Flow

Connection

The game flow is described as follows:

1. Assuming the two players are Ahmed and Ali, Ahmed initiated the game.

Game Config.

2. Initial points screen appears asking both players to choose the initial points.
3. Ahmed choses the initial point to be 70 and Ali chooses it to be 60. Initial points for both players start at 60 because it is the lower number.
4. Level selection screen appears to Ahmed only. Ahmed selected level 1.
5. The forbidden character selection screen appears for both players.
6. Ahmed chooses 'M' as a forbidden character while Ali chooses '0' as a forbidden character. The character appears on both screens.

GAME GUI

7. As both of them enter their forbidden character, the main game screen appears for both players as shown in figure 4 with all registered initialized by zeros.

GAME LOGIC

8. Ahmed wants to start by writing "Mov ax, 105e". He will not be able to write this command because Ali forbids him from using the character "0". Ahmed wrote the command "Mov AX, FFF". For simplicity, all numbers are in hexadecimal format without "H".
9. The command executed at Ali's processor and Ali's AX register contains the value

"0FFF".

10. In Ali's turn, He wants to write Mov Ax,105e but he is forbidden from the character "M". In addition, He knows that if Ahmed writes "add ax,5F" as his next instruction he will be the winner. Therefore, he must use one of the available special power ups.
11. Five types of power ups are available:
    a. Executing a command on your own processor (consumes 5 points)
    b. Executing a command on your processor and your opponent processor at the same time (consumes 3 points)
    c. Changing the forbidden character only once (consumes 8 points)
    d. Making one of the data lines stuck at zero or at one for a single instruction (consumes 2 points)
    e. Clearing all registers at once. (Consumes 30 points and could be used only once). For a team of five, five power ups should be implemented. Otherwise, any team could choose only four power ups.
12. Ali chose the fourth power up and entered 3 as the line number and zero as the sticking value to be zero. In this case, if Ahmed wrote the command "add ax,5F" it will be converted to "Add ax,57" Note that "F" converted to "7" because "F" = 1111 while the line number 3 is stuck at zero it will be equal to 0111 = 7. In addition to choosing the power up, Ali entered the following command "Mov BX,AL". This command is invalid. Therefore, Ali loses one point, the command is not executed on Ahmed's processor and Ali loses his turn.
13. Five types of errors should be detected for a team of five members. Otherwise, any team should choose any four types of errors from the following list
    a. Size mismatch
    b. Memory to memory operation
    c. Invalid register name
    d. Pushing 8 bits
    e. Incorrect addressing mode like "mov ax, [CX]"
14. At least five addressing modes should be allowed for a team of five members, otherwise any four addressing modes are required for any team.

Twenty different commands should be allowed for a team of five members, otherwise any 16 commands are required for any team. Commands like
[ADD-ADC-SUB-SBB-DIV-MUL-MOV-IDIV-IMUL-XOR-AND-OR-NOP-SHR-SHL-SAR-CLC-ROR-RCL-RCR-ROL-PUSH-POP-INC –DEC]

15. If any player's points reach zero s/he loses.

Flying Objects

16. Flying objects appear randomly at both screens at the same time. Each player tries to shoot it using a gun. Based on the object color it gives the players a certain number of points i.e. green object gives the player one point, blue object gives the player two points … etc. The gun is moved using arrows and shooting using a space bar. Colored circles with a number inside it determines the numbers of hit flying objects.

Second Level

17. For the second level, all rules of the first level are applied in addition to the following rules
    a. The forbidden character is hidden. Any command including the forbidden character will not be executed. Each player should deduce the forbidden character through non executed commands.
    b. For each command, each player is able to decide whether to execute it on his/her processor or the opponent processor.
    c. Each player should be able to put initial values for any register at the beginning of

the level.

d. An extra power up is introduced, letting each player be able to change the target value only once to any value other than values currently existing in any register for the two players.

18. Note: each team has the freedom to design the game graphics and screen organization. **DO NOT USE THE SAME INTERFACE MENTIONED IN FIGURE 4.** Each team should use their own design for the user interface.

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 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.