Green University of Bangladesh
Department of Computer Science and Engineering (CSE)
Faculty of Sciences and Engineering
Semester: (Spring, Year:2023), B.Sc. in CSE (Day)

Course Title: Microprocessors & Microcontrollers Lab

Course Code: CSE 304 **Section:** 212 D5

Lab Project Name: Digital Clock (displays the current time)

Student Details

Name		ID
1.	Sajid Rahman Rifan	212902017

Submission Date:
Course Teacher's Name: Md. Nasif Osman Khansur

[For Teachers use only: Don't Write Anything inside this box]

Lab Project Status

Marks: …………………………………… **Signature:** ....

Comments: .... **Date:** ....

Chapter 1

Introduction

1.1 Introduction

In the fast-paced digital age, time management is crucial, and a digital clock stands as a reliable companion in keeping track of the ever-elusive seconds, minutes, and hours. Unlike traditional analog clocks, which rely on the movement of hands around a dial, digital clocks present time in a straightforward numerical format, making it easily readable and accessible.

A digital clock typically features a digital display that showcases the current time in digits. The digits represent hours and minutes, and sometimes seconds, offering a precise and instant readout. This simplicity and clarity make digital clocks popular in various settings, from homes and offices to public spaces and beyond.

1.2 Design Goals/Objective

- **Real-time Updates:** The digital clock should update in real-time, displaying the current time accurately.
- **Resizable Window:** Allow users to resize the window to accommodate their preferences.
- Timezone Support: Optionally, include the ability to set and display the time in different time zones.
- **Get Current Time:**
    1) Read the current time from the RTC.
    2) The time could be stored in registers or memory locations.
- **End Program:**
    1. Add any necessary cleanup code.
    2. Halt the program or return control to the operating system.

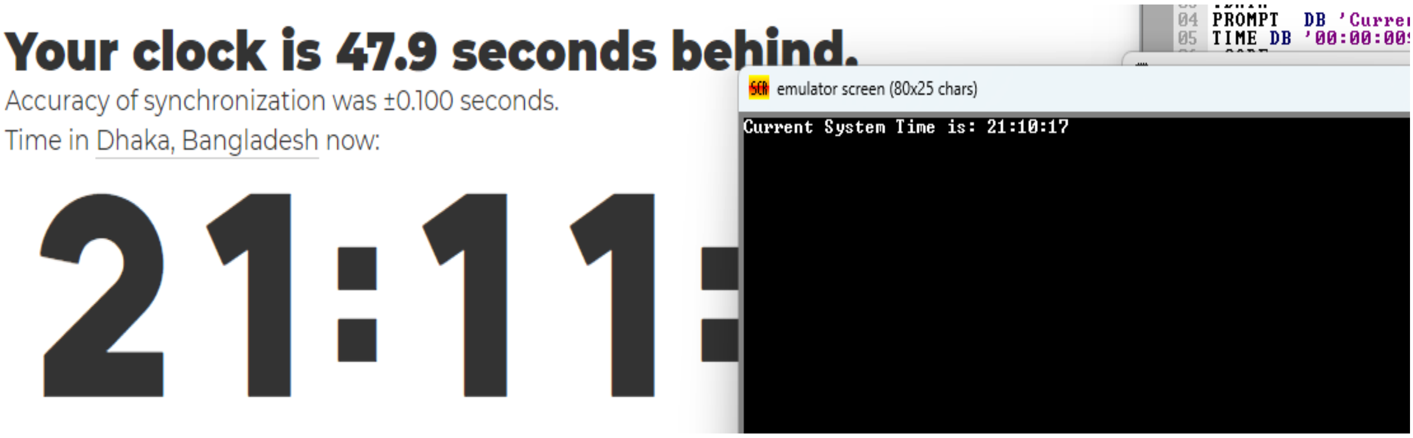
Chapter 2

Design/Development/Implementation of the Project

2.1 Implementation

```
.MODEL SMALL
.STACK 100H
.DATA
PROMPT  DB 'Current System Time is : $'
TIME DB '00:00:00$'
.CODE
   MAIN PROC
     MOV AX, @DATA
     MOV DS, AX
     LEA BX, TIME

     CALL GET_TIME

     LEA DX, PROMPT
     MOV AH, 09H
     INT 21H
     LEA DX, TIME
     MOV AH, 09H
     INT 21H
     MOV AH, 4CH
     INT 21H

   MAIN ENDP
GET_TIME PROC

    PUSH AX
    PUSH CX
    MOV AH, 2CH
    INT 21H
    MOV AL, CH

    CALL CONVERT

    MOV [BX], AX
    MOV AL, CL
    CALL CONVERT

    MOV [BX + 3], AX
    MOV AL, DH

    CALL CONVERT

    MOV [BX + 6], AX
    POP CX
    POP AX
    RET

GET_TIME ENDP
   CONVERT PROC

    PUSH DX
    MOV AH, 0
    MOV DL, 10
    DIV DL
    OR AX, 3030H
    POP DX
    RET

   CONVERT ENDP
 END MAIN
```

2.2 OUTPUT:



Your clock is 47.9 seconds behind.

Accuracy of synchronization was ±0.100 seconds.

Time in Dhaka, Bangladesh now:

21:11:

```
PROMPT   DB 'Curre
TIME DB '00:00:00
```

```
Current System Time is: 21:10:17
```

FIG:01

Chapter 3

Performance Evaluation

3.1 Simulation Environment/ Simulation Procedure

Data Section:

a. PROMPT DB 'Current System Time is : $': This is a string that will be displayed before showing the time.
b. TIME DB '00:00:00$': This is the initial placeholder for the time, set to '00:00:00'. The '$' at the end is a string termination character.

Code Section:

MAIN PROC:

i. MOV AX, @DATA and MOV DS, AX: Set up the data segment.
ii. LEA BX, TIME: Load the effective address of the TIME string into BX.
iii. CALL GET_TIME: Call the GET_TIME procedure to retrieve the current systemtime.
iv. Display the prompt and the current time using DOS interrupts (INT 21H).
v. MOV AH, 4CH and INT 21H: Terminate the program.

GET_TIME PROC:

vi. PUSH AX and PUSH CX: Save the values of AX and CX on the stack.
vii. MOV AH, 2CH and INT 21H: Get the system time into CX:DX.
viii. Extract the hours, minutes, and seconds from the CX:DX values and convert them to ASCII before storing them in the TIME string.
ix. POP CX and POP AX: Restore the values of AX and CX from the stack.
x. RET: Return from the procedure.

CONVERT PROC:

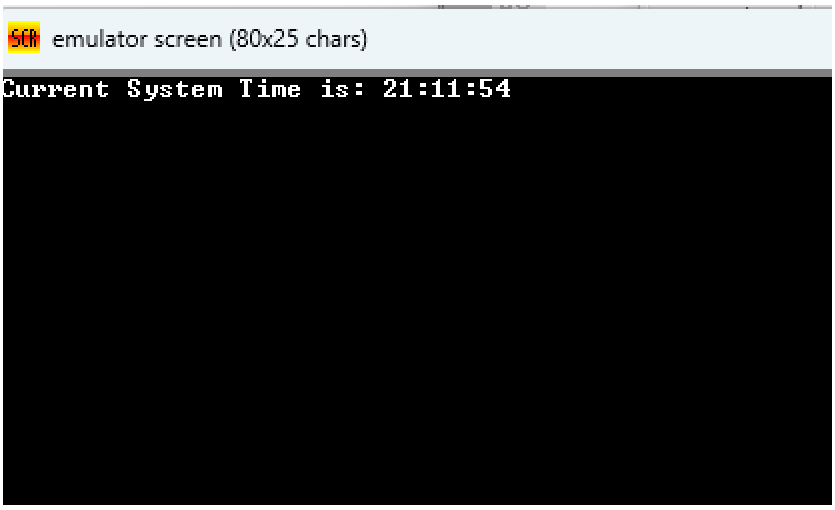
xi. PUSH DX: Save the value of DX on the stack.
xii. MOV AH, 0 and MOV DL, 10: Set up for division by 10.
xiii. DIV DL: Divide AX by 10, result in AL (quotient), and AH (remainder).
xiv. OR AX, 3030H: Convert the quotient to ASCII.
xv. POP DX: Restore the original value of DX.
xvi. RET: Return from the procedure.

END MAIN: End of the program.

3.2 Results and Discussions

To write an assembly language program to display current system time.(DOS PROGRAMMING)

3.2.1 Results



```
Current System Time is: 21:11:54
```

FIG:02

3.2.2 Analysis and Outcome

- The program initializes the data segment, calls a procedure to obtain the current system time, converts the time to ASCII, and displays it along with a prompt message.
- The time is displayed in the format "HH:MM:SS".
- The program terminates after displaying the time.
- The GET_TIME procedure interacts with the DOS interrupt 21H to obtain the system time.
- The CONVERT procedure is used to convert binary digits to ASCII characters.

Chapter 4

Conclusion

4.1 Introduction

This assembly program initializes a data segment, obtains the current system time, converts the time components to ASCII, and then displays the formatted time along with a prompt message. The time is displayed in the HH:MM:SS format. The CONVERT procedure is used for converting numeric values to ASCII characters.

4.1 Practical Implications

Real-Time Clock (RTC) Interaction:
The program uses interrupt 21h, function 2Ch to get the current time from the Real-Time Clock (RTC) in the BIOS. Practical implication: Understanding and interacting with system interrupts is crucial for low-level programming tasks.
ASCII Conversion:
The CONVERT procedure converts decimal values to ASCII characters. Practical implication: This is a common operation in systems programming for displaying numerical information.
Memory Model and Segmentation:
The program uses the small memory model and handles data in segments. Practical implication: Managing memory efficiently is critical in assembly programming, especially in environments with limited resources.
Interrupt-Based I/O:
The program uses interrupt 21h for displaying messages. Practical implication: Understanding and utilizing interrupt-driven I/O is fundamental for interacting with the operating system in assembly language.

4.2 Scope of Future Work

User Interface Improvements:

- Allow the user to input a specific format or customize the display format.
- Implement a graphical user interface (GUI) instead of using the console for a more interactive experience.

Error Handling:

- Add error handling for cases where the time retrieval fails or the format is not as expected.
- Check for invalid input or unexpected values and handle them appropriately.

Real-Time Clock (RTC) Interfacing:

- Interface with the real-time clock (RTC) hardware directly for more accurate and real-time clock values.

Time Zone Handling:

- Implement functionality to display the time in different time zones.

References

https://vikramlearning.com/jntuh/notes/microprocessors-and-microcontrollers/program-for-digital-clock-design-using-8086/250?fbclid=IwAR3rTdKTFNci1SpbQJJWD12W0k8qN6zxrWDmNZywS6GSvdwjP4zoJ3v6rYE

GOOGLE