Digital Clock in Assembly Language

Abstract

There are primarily two types of clocks. One being analog and the other being digital. The analog clock was the only type of clock in the past when there were no digital technology available. But with the dawn of digital era, digital clocks have become one of our daily necessities. We use digital clock in different forms like hand watch, cell phones, personal computers etc.

We see digital clocks in our personal computers everyday. It is a prime component of these devices. In this project we have implemented a digital clock using the graphics mode of assembly language programming. Here we show how this can be done and walk you through the whole process.

Introduction

Assembly language is one the earliest means of computer programming. It is the lowest level of programming above machine language programming. Assembly programming requires deep understanding about the hardware we are dealing with. The hardware components can be directly manipulated using this. Since the programmer can play with the components as he wishes, using this programming method can get us rid of a lot of redundancy and thus create faster programs.

A digital clock is a type of clock that displays the time digitally, as opposed to an analog clock, where the time is indicated by the positions of rotating hands. Digital clocks are often associated with electronic drives, but the “digital” description refers only to the display, not the drive mechanism.

To represent the time, most digital clocks use a seven-segment display. It shows hours, minutes and seconds and can be set to show in both 24 hours and 12 hours format.

Features

The main and significant features of this program is described below.

1. The clock shows the time of the computer system.
2. It shows hours, minutes and seconds.
3. It uses 12 hour format.
4. It is developed in VGA graphics with 320x200 resolution.
5. It is displayed in a classy, stylish and attractive font.
6. The interface is simplistic with white text on black background.

Key concepts

Interrupts

In systems programming, an interrupt is a signal to the processor emitted by hardware or software indicating an event that needs immediate attention. An interrupt alerts the processor to a high-priority condition requiring the interruption of the current code the processor is executing. The processor responds by suspending its current activities, saving its state, and executing a function called an interrupt handler (or an interrupt service routine, ISR) to deal with the event. This interruption is temporary, and, after the interrupt handler finishes, the processor resumes normal activities.

Graphics modes

There is the text mode and the graphics mode. The text mode is used to display text only whereas the graphics mode can be used to display both text and other graphical elements. There are 3 basic types of graphics display mode.

Color Graphics Adapter (CGA): It has three modes.

1. 320 x 200 4 color
2. 320 x 200 4 color (color burst off)
3. 640 x 200 2 color

Enhanced Graphics Adapter (EGA): It has four modes.

1. 320 x 200 16 color
2. 640 x 200 16 color
3. 640 x 350 monochrome
4. 640 x 350 16 color

Visual Graphics Array (VGA): It has three modes.

1. 640 x 480 2 color
2. 640 x 480 16 color
3. 320 x 200 256 color

Here in this project we have used the 320 x 200 256 color mode of the VGA mode. The VGA adapter has higher resolution than the EGA. There are also more colors since it can generate 64 levels of red, green and blue.

Interrupts used in the program

INT 10H: This interrupt is used to execute graphics based routines.

```
AH = 00H; set graphics mode
AL = 13H; graphics mode VGA 320 x 200 256 color
INT 10H
```

```
AH = 0CH; draw pixel
AL = color
BH = page
CX = column
DX = row
INT 10H
```

INT 16H: The bios interrupt 16H routine is the key-board driver. In order to take an input from the keyboard we have used this interrupt where the function value 00H was stored in AH.

```
AH = 00H; wait for a key press
INT 16H
```

INT 21H: This is a DOS interrupt. It is used to request different DOS function. The function value is stored in AH. In our program we have used three functions.

```
AL = interrupt number
AH = 35H; gets interrupt vector stored in AL
INT 21H; returns the address to ES:BX
```

```
AL = interrupt number
AH = 25H; sets interrupt vector stored in AL
INT 21H; sets the vector to DS:DX
```

```
AH = 2CH; gets the current time, CH = hour, CL = minute, DH = second
INT 21H;
```

```
AH = 4CH; returns program to DOS
INT 21H;
```

INT 1CH: The 1CH interrupt is automatically generated by an internal clock circuit. It is generated about 18.2 times per second on an average. We can define a routine for this interrupt which will be executed every time the interrupt is generated. Since we need to continuously refresh the display, we simply define a routine to refresh the display.

Procedure

1. First we define two macros for later uses. One for drawing rows and another for drawing columns. We send the row, column and color values as parameters for the macros.
2. We then set the video mode for our display operation.
3. Since we are going to use 1CH in order to continuously refresh the screen, we set the interrupt vector for 1CH.
4. The routine which will be executed each time the 1CH interrupt is generated, is defined.
5. In the routine, we first generate interrupt to get the current time from the system.
6. As the time is returned into the registers, we use those values to convert that time to a string for ease in later tasks.
7. We define ten functions each for displaying each digit from 0 to 9.
8. Each digit is created using the macros defined earlier. We draw rows and columns for the digits accordingly.
9. Before a digit is printed on the screen, we first clear the screen since it will have the previously displayed digit on the screen.
10. As the screen keeps getting refreshed continuously, the program waits for a key to be pressed in order to terminate the program.
11. As a key is pressed the display is set back to text mode and the program terminates and returns control to DOS.

Limitations

Since this is a very basic level program it has some limitations.

1. The program can’t show time in milliseconds.
2. It can only show time. There is no feature for setting up alarms.
3. Due to refreshing continuously, sometimes the digits flicker.

Discussion

This project has worked as a real life working experience. Developing a graphical program from scratch in assembly was a tough job. There are way too many limitations. It is also cumbersome due to the low level nature of the language. Though assembly language is mainly used from programming different system tasks, developing GUI is also a fun thing to do.

We could've added a date feature in the program. There could've been an alarm system. Adding feature for a stop watch would've been great. Due to lack of time and lots of limitations in assembly language, we couldn't do that.

References

1. "Assembly Language Programming and Organization of the IBM PC", Authors: Ytha Yu and Charles Marut, Publisher: Mitchell McGraw-Hill.
2. "Microprocessors and Interfacing", Author: Douglas V Hall, Publisher: Tata McGraw-Hill Publishing Company Ltd.
3. "Wikipedia", Link: en.wikipedia.com
4. "Anatomy of the standard VGA 256-color palette", Link: www.youtube.com/watch?v=-9QnckzyYvs