CSCI516: Program 1 - October 02, 2008
The Program is due:
October 15, 2008 in the beginning of the class

For your first program, you are to write an assembly program that will display **on a clear screen** as follows:

Hello - I am xxxxxxxxx (insert your name and your list ID#)

To view the current Time please press "CT",

To view the current day of the week press "DW",

To view the current date press "CD",

The program must end if the user press "EP" and write the following message: Thank you – Have a nice day!

Prompt the user with the above questions. You are then to read the user's response from the keyboard. If the user's response is CT, then display *"The current time is:* **XX:XX:XX"**.

If the user's response is DW, then display *"Today is:* **the X day of the week".**

If the user's response is CD, then display *"The current date is:* **mm.dd.yy"**

You may use the string output function (int 21h function 09h – *See page 466,* or the Lecture_Interrups on my Web site) to write to the screen. You may use buffered input to read from the keyboard (int 21h function 0Ah – *See page 469*, or the Lecture_Interrups on my Web site).

Use good structured methods to design your program. Use meaningful labels. Align your fields and use comments to explain the meaning of your code. Neatness counts in your grade. Any additional feature will be bring to you additional points.

You may use the MASM or the Turbo Assembler to assemble and link your program. Your Source Program (Prg1_ID.ASM) must be located on a labeled floppy disk with your name. Submit all files generated by your compiler, along with your source file. After the program has been assembled <u>with no errors</u>, execute the program. If necessary, use the Turbo Debugger to find any problems. Appendix D in your book discusses the Turbo Debugger.
Submit the following files on a floppy disk (drive A): **Prg1_ID.ASM, Prg1_ID.LST, Prg1_ID.OBJ, Prg1_ID.EXE. The notation ID means your class list number.**

Some hints in forms of Pseudo-code and charts are given below.
The Lecture which discusses the matter is Lecture_Interrups on my web page.

An Example Pseudo-code for Program 1

Main Program

> Call Clear_Screen
> Prompt all questions
> Call Read_Key_Board
> *If* key=CT *Then*
> > Call Print_MessageCT
> > > *Else*
> > Call Print_End
> *End If*

Second Question: *If* key=CW *Then*
> > Call Print_MessageCW
> *End If*
> Call Print_End

Third Question: *If* key=CD *Then*
> > Call Print_MessageCD
> *End If*
> Call Print_End

End Main

Clear_Screen
> Save all Registers
> Write 25 Blank Lines to the Screen
> Reset Cursor to Line 1 Column 1
> Restore all Registers
End Clear_Screen

Print_MessageT
> Write the answer to the Screen
> Call Get_Time
> Write the time to the Screen
End Print_Message

Print_MessageW
> Write the answer to the Screen
> Call Get_Date
> Write the day of the week to the Screen
End Print_Message

Print_MessageD
> Write the answer to the Screen
> Call Get_Date
> Write the date to the Screen
End Print_Message

```
Read_Key_Board
        Write Repeat Line to the Screen
        Read Reply from the Key Board
End Read_Key_Board
Get_Date
        Get current Date from the Operating System
        Move 0 to ah
        Call To_ASCII
        Move converted Day of the week to Message
        Move Year to AX Register
        Call To_ASCII
        Move converted Year to Message
        Move Month to AX Register
        Call To_ASCII
        Move converted Month to Message
        Move Day to AX Register
        Call To_ASCII
        Move converted Day to Message
End Get_Date
Print_End
        Write Ending Line to the Screen
End Print_End

Get_Time
        Get current Time from the Operating System
        Set AM_PM = 'PM'
        If Hour > 12    Then
                Hour = Hour - 12
        Else
                Set AM_PM = 'AM'
        End IF
        Move Hour to AX Register
        Call To_ASCII
        Move converted Hour to Message
        Move Minute to AX Register
        Call To_ASCII
        Move converted Minute to Message
        Move Second to AX Register
        Call To_ASCII
        Move converted Second to Message
End Get_Time

To_ASCII
        Set Count to 5
        Set Index to 4
        Do While Count is > 0
                Divide AX by 10 - Quotient to AX
```

```
                    Remainder to DX
            Add 30h to DX to Convert to ASCII
            Move DL to Ascii_Out [Index]
            Decrement Index
            Decrement Count
        End Do
End To_ASCII

End  Main Program
```

An Example Block Diagrams

PrtMsg

```
┌─────────────────────┐
│  Print the Message 1 │
│    to the Screen     │
└─────────────────────┘
          │
          ▼
┌─┬─────────────────┬─┐
│ │      Call       │ │
│ │     GetTime     │ │
└─┴─────────────────┴─┘
          │
          ▼
┌─────────────────────┐
│  Print the Message 2 │
│    to the Screen     │
└─────────────────────┘
          │
          ▼
┌─┬─────────────────┬─┐
│ │      Call       │ │
│ │     GetDate     │ │
└─┴─────────────────┴─┘
          │
          ▼
       Return
```

GetDate

```
┌─────────────────────┐
│ Get the Current Date │
│ from the Operating   │
│       System         │
└─────────────────────┘
          │
          ▼
┌─┬─────────────────┬─┐
│ │      Call       │ │
│ │     To_ASCII    │ │
└─┴─────────────────┴─┘
          │
          ▼
┌─────────────────────┐
│ Format the Date into │
│   Printable Format   │
└─────────────────────┘
          │
          ▼
       Return
```

GetTime

```
┌─────────────────────┐
│ Get the Current Time │
│ from the Operating   │
│       System         │
└─────────────────────┘
          │
          ▼
┌─┬─────────────────┬─┐
│ │      Call       │ │
│ │     To_ASCII    │ │
└─┴─────────────────┴─┘
          │
          ▼
┌─────────────────────┐
│ Format the Time into │
│   Printable Format   │
└─────────────────────┘
          │
          ▼
       Return
```

```
      ┌─────────────┐                              ┌─────────────┐
      │   To_ASCII  │                              │    PrtEnd   │
      └─────────────┘                              └─────────────┘
             │                                            │
             ▼                                            ▼
   ┌───────────────────┐                      ┌───────────────────┐
   │ Set Loop Count to 5│                      │                   │
   │ Set Index Value to 4│                     │ Print the Message 1│
   │ Initialize BX to 10│                      │  to the Screen    │
   └───────────────────┘                      └───────────────────┘
             │                                            │
             ▼◄────────────────────────────┐             ▼
   ┌───────────────────┐                    │    ┌───────────────────┐
   │  Divide AX by 10  │                    │    │      Return       │
   │   Quotion to AX   │                    │    └───────────────────┘
   │ Remainder to DX   │                    │
   └───────────────────┘                    │
             │                              │
             ▼                              │
   ┌───────────────────┐                    │
   │     Add 30h to    │                    │
   │     Remainder     │                    │
   │ (Convert Remainder│              No    │
   │    to an ASCII    │                    │
   │    Character)     │                    │
   └───────────────────┘                    │
             │                              │
             ▼                              │
   ┌───────────────────┐                    │
   │ Store Remainder in│                    │
   │ Output Area[Index]│                    │
   └───────────────────┘                    │
             │                              │
             ▼                              │
   ┌───────────────────┐          ◇─────────┘
   │ Decrement Index   │─────────►◇  Count  ◇
   │ and Count by 1    │          ◇  = 0 ?  ◇
   └───────────────────┘          ◇─────────◇
                                       │
                                      Yes
                                       │
                                       ▼
                                ┌─────────────┐
                                │    Return   │
                                └─────────────┘
```