CSCI516: Program 1 - October 11, 2010
The Program is due:
October 25, 2010 in the beginning of the class

For Late Submissions 10 out of 100 points will be taken off.

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

I am xxxxxxxxx (insert your name and your Class 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",

For all students with an odd Class ID
To find the sum of the time components (hh+min+sec) press +;

For all students with an even Class ID
To find the sum of the date components (mm+day+year) press +;

The program must end if the user press "EP". The program respond is: 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"**

For all students with an odd Class ID
The sum of the time components (hh+min+sec) is: *give the sum*

For all students with an even Class ID
The sum of the date components (mm+day+year is: *give the sum*

Start the work on a clear screen.
Write every question or respond on a separate row.
5 points will be taken of for every error.

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 with no errors, 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.**
5 points off for every mistake.
In case of copied programs or substantial pieces of the program a mark of 0 or 50 will be given.

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 of a program for finding sum of integer numbers is give on P76 Ed.5.

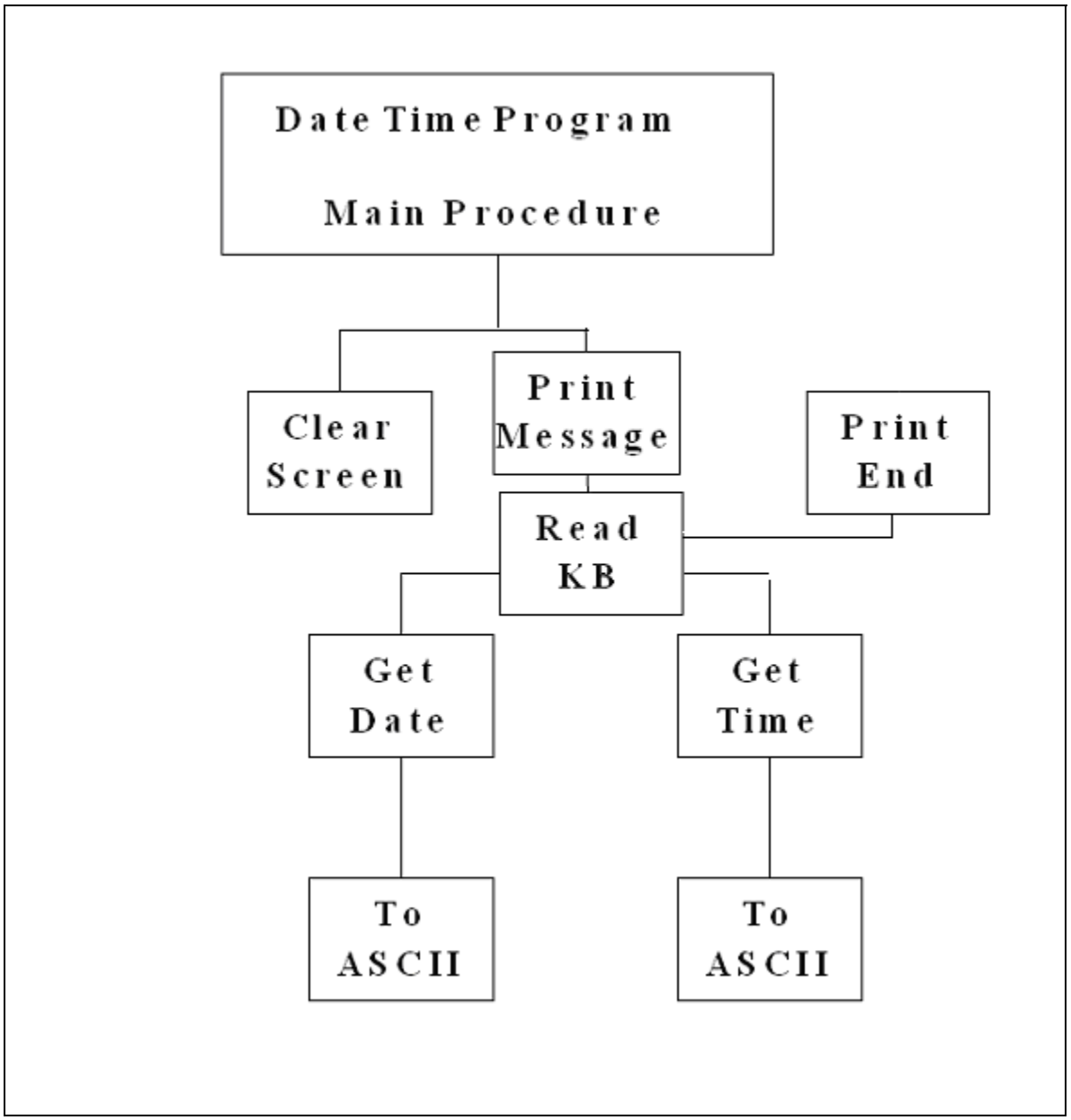
An Example Pseudo-code for Program 1

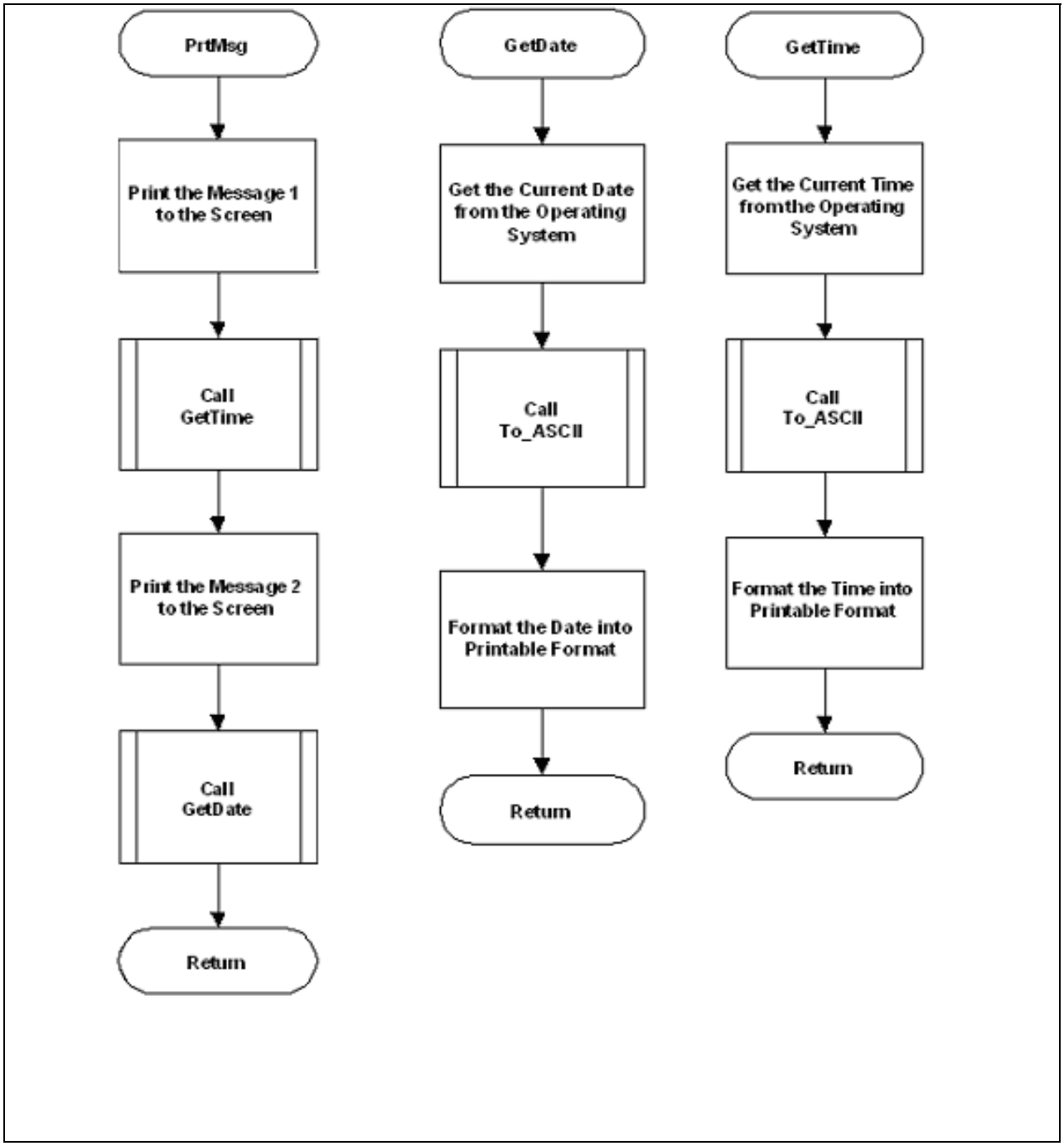
```
Main Program
                    Call Clear_Screen
                    Prompt all questions
                    Call Read_Key_Board
                    If key=CT Then
                            Call Print_MessageCT
                                Else
                            Call Print_End
                    End If
Second Question:    If key=CW Then
                            Call Print_MessageCW
                    End If
                    Call Print_End
Third Question:     If key=CD Then
```

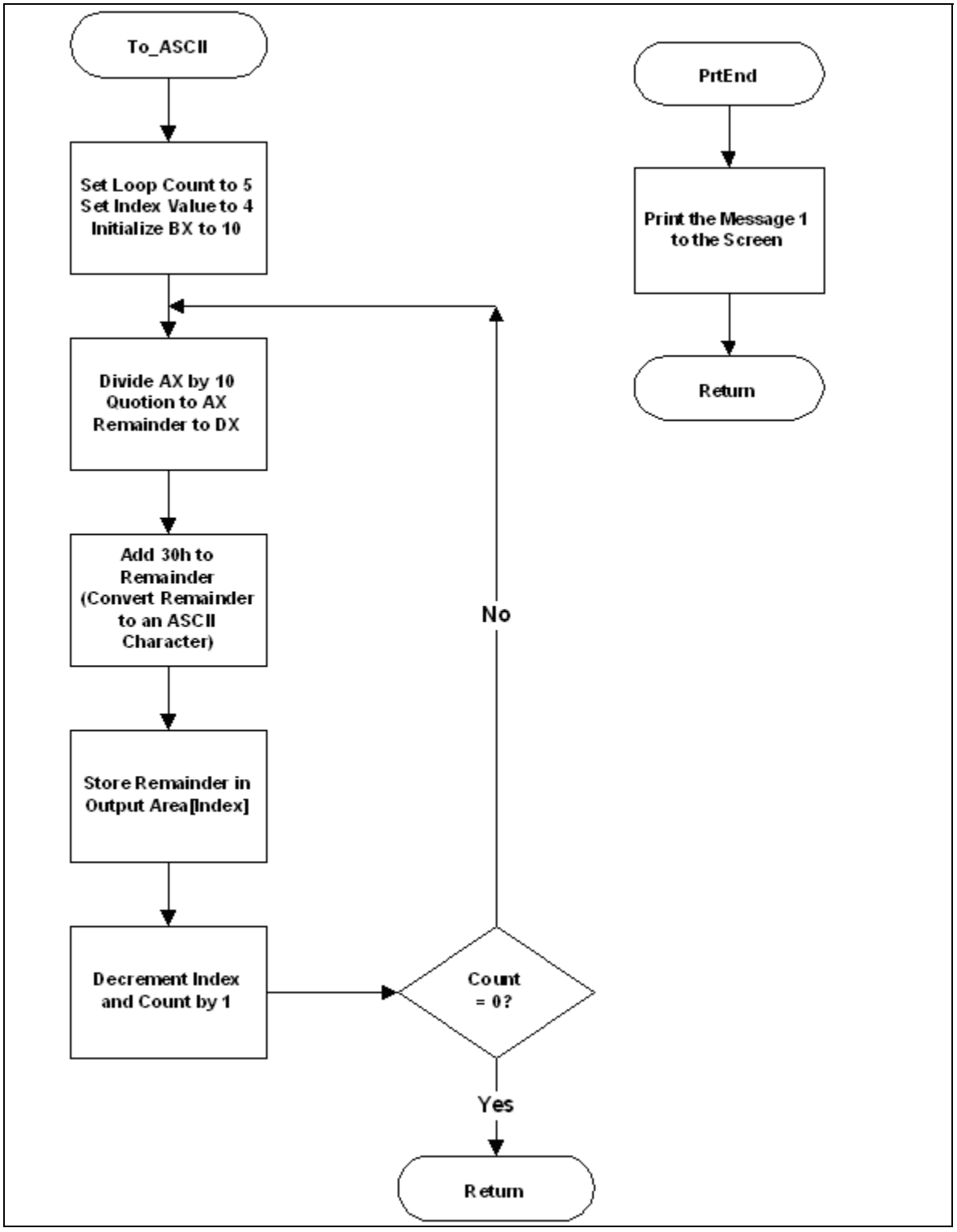
```
                    Call Print_MessageCD
            End If
            Call Print_End

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
Set Loop Count to 5
Set Index Value to 4
Initialize BX to 10
Divide AX by 10
Quotion to AX
Remainder to DX
Add 30h to
Remainder
(Convert Remainder
to an ASCII
Character)
Store Remainder in
Output Area[Index]
Decrement Index
and Count by 1
Count
= 0?
No
Yes
Return
PrtEnd
Print the Message 1
to the Screen
Return

HERE ARE SOME EXAMPLE PIECES OF CODE and NOTES TO HELP YOU DEVELOP YOUR PROGRAM1

1)

```
;       Keyboard Input Buffer

KbBuff   DB      2           ; max no of input chars
                             ; ( 1 + the enter key)
KbNoRead DB      0           ; No of chars actualy read
                             ; This field is returned
                             ; ( the enter key don't count)
KbIn1    DB      ' '         ; room for char 1
KbEntKey DB      ' '         ; room for the enter key (0Dh)
```

2) Main Program - look at the main block diagram given in the assignment.

```
.code
main PROC
                mov ax,@data
                mov ds,ax
                CALL ClrScreen
L1:             CALL Print_Msg
                CALL GotoNextLine
                mov ah,0Ah
                mov dx,OFFSET kybdData
                int 21h
                mov ah, [kybdData.buffer]
                mov al, [kybdData.buffer+1]
                CALL ClrScreen
                cmp al,'D'
                JNZ L2
                cmp ah,'C'
                JNZ L2
                mov dx, OFFSET date
                CALL PrintMSg
                CALL Get_Date
                CALL GotoNextLine
                JMP ENDL1
L2:     CALL ClrScreen
                cmp al,'T'
                JNZ L3
                cmp ah,'C'
                JNZ L3
                mov dx, OFFSET time
                CALL PrintMSg
                CALL Get_Time
                CALL GotoNextLine
                JMP ENDL1
```

```
L3:             CALL ClrScreen
                cmp al,'W'
                JNZ L4
                cmp ah,'D'
                JNZ L4
                CALL Get_Day_Of_Week
                mov dx, OFFSET day
                CALL PrintMSg
                CALL GotoNextLine
                JMP ENDL1
L4:             CALL ClrScreen
                cmp al,'P'
                JNZ ENDL1
                cmp ah,'E'
                JNZ ENDL1
                JMP ENDL2
ENDL1:          JMP L1
ENDL2:          CALL GotoNextLine
                mov dx, OFFSET endMsg
                CALL PrintMSg
                CALL GotoNextLine

                exit
main ENDP


3)ClrScreen   Proc              ; Procedure to clear the screen

        push    ax          ; Save all regesters
        push    bx
        push    cx
        push    dx

;       call BIOS to Clear the Screen

        mov     ah,6        ; Clear the screen command
        mov     al,0        ; clear the whole screen
        mov     ch,0        ; Start X Cord.
        mov     cl,0        ; Start Y Cord.
        mov     dh,24       ; End X Cord.
        mov     dl,79       ; End Y Cord.
        mov     bh,7        ; Clear to normal Attributes
        int     10h         ; BIOS interrupt

;       Set the cursor to line 1 position 1

        mov     ah,2
        mov     dh,1        ; row 1
        mov     dl,1        ; column 1
```

```
        mov     bh,0        ; page 0
        int     10h         ; BIOS interrupt

        pop     dx          ; Restore all regesters
        pop     cx
        pop     bx
        pop     ax
        ret
ClrScreen  ENDP
```

4)

```
PrtMsg   PROC
        push    ax          ; Save all regesters
        push    bx
        push    cx
        push    dx

;       call OS to Print the Message
        mov     dx,OFFSET Msg1
        mov     ah,9h       ; Function code for display string
        int     21h         ; call OS to do it

        call    GetDate     ; Read and Format the Date
        call    GetTime     ; Get the current time of Day.

        mov     dx,OFFSET Msg2
        mov     ah,9h       ; Function code for display string
        int     21h         ; call OS to do it
        pop     dx          ; Restore all registers
        pop     cx
        pop     bx
        pop     ax
        ret
PrtMsg   ENDP
```

5) An Example program to show how to get the system time and date as well as how to display them is given in the text book section “MS-DOS Function Calls (INT 21h) P438, 5th Edition”.
Example programs are also given below:

```
GetTime  PROC

;       GetTime will read the current time from the system and
;       convert it to Ascii characters.  If the Hour is 00 to 11,
;       it will insert "AM" in AmPm. If the hour = 12 it will
;       insert "PM". If the Hour is 13 to 23 it will insert "PM"
;       and subtract 12 from Hour.

;       It will move the Hour to MsgHour (2 digits) and the
;       minutes to MsgMinutes (2 digits).
```

```
        push    ax          ; Save all regesters
        push    bx
        push    cx
        push    dx

        mov     ah,2Ch      ; get the current time
        int     21h

;       Set up the AM/PM Message
              mov     AmPm,'P'    ; move PM to the message
        mov     AmPm+1,'M'
        cmp     ch,12       ; compare the hour to 12
        je      AmPm_Ok     ; hour = 12 - just print it
        ja      IsPM        ; hour > 12 - print PM and sub 12 from the hour
        mov     AmPm,'A'    ; hour < 12 - move in AM
        jmp     AmPm_Ok

IsPM:

            sub     ch,12               ; sub 12 hours

AmPm_Ok:
;           Convert the hours to Ascii

        mov     al,ch       ; move the hour to AX
        mov     ah,0        ; clear the high order of the reg
        call    To_Ascii    ; convert it to charactrs

;       Hours will be in AsciiOut in the form 000HH.

        mov     al,AsciiOut+3  ; get the fourth digit
        mov     MsgHr,al    ; save the first digit
        mov     al,AsciiOut+4  ; get the fifth digit
        mov     MsgHr+1,al  ; save the second digit

;       Convert the minutes to Ascii
        mov     al,cl       ; move the minutes to AX
        mov     ah,0        ; clear the high order of the reg
        call    To_Ascii    ; convert it to charactrs

;       Minutes will be in AsciiOut in the form 000MM.
        mov     al,AsciiOut+3  ; get the fourth digit
        mov     MsgMin,al   ; save the first digit
        mov     al,AsciiOut+4  ; get the fifth digit
        mov     MsgMin+1,al ; save the second digit

        mov     al,dh       ; move the seconds to AX
        mov     ah,0        ; clear the high order of the reg
```

```
        call    To_Ascii        ; convert it to charactrs

        mov     al,AsciiOut+3   ; get the fourth digit
        mov     MsgSec,al       ; save the first digit
        mov     al,AsciiOut+4   ; get the fifth digit
        mov     MsgSec+1,al     ; save the second digit

        pop     dx              ; Restore all regesters
        pop     cx
        pop     bx
        pop     ax
        ret
GetTime  ENDP

GetDate  PROC
;       GetDate will read the current date from the system and
;       convert it to Ascii characters.  It will then move the
;       Year to MsgYear (4 digits), the Month to MsgMonth (2 digits)
;       and the Day to MsgDay (2 digits).

        push    ax              ; Save all regesters
        push    bx
        push    cx
        push    dx

        mov     ah,2Ah          ; get the current date
        int     21h

          mov   ah, 0
        call    To_Ascii        ; convert it to charactrs
          mov   al,AsciiOut+4   ; get the fifth digit
          mov   MsgDW,al        ; get the current day of the week

;       Convert the Year from Binary to ASCII
        mov     ax,cx           ; move the year to AX
        call    To_Ascii        ; convert it to charactrs

;       Year will be in AsciiOut in the form 0YYYY.
        mov     al,AsciiOut+1   ; get the 1st digit
        mov     MsgYear,al      ; move it to the message area
        mov     al,AsciiOut+2   ; get the second digit
        mov     MsgYear+1,al    ; save the second digit
        mov     al,AsciiOut+3   ; get the third digit
        mov     MsgYear+2,al    ; save the third digit
        mov     al,AsciiOut+4   ; get the fourth digit
        mov     MsgYear+3,al    ; save the fourth digit

;       Convert the Month from Binary to ASCII
        mov     al,dh           ; move the month to AX
```

```
        mov     ah,0          ; clear the high order of the reg
        call    To_Ascii      ; convert it to charactrs

;       Month will be in AsciiOut in the form 000MM.

        mov     al,AsciiOut+3 ; get the fourth digit
        mov     MsgMon,al     ; save the first digit
        mov     al,AsciiOut+4 ; get the fifth digit
        mov     MsgMon+1,al   ; save the second digit

;       Convert the Day from Binary to ASCII

        mov     al,dl         ; move the day to AX
        mov     ah,0          ; clear the high order of the reg
        call    To_Ascii      ; convert it to charactrs

;       Day will be in AsciiOut in the form 000DD.

        mov     al,AsciiOut+3 ; get the fourth digit
        mov     MsgDay,al     ; save the first digit
        mov     al,AsciiOut+4 ; get the fifth digit
        mov     MsgDay+1,al   ; save the second digit

        pop     dx            ; Restore all regesters
        pop     cx
        pop     bx
        pop     ax
        ret
GetDate  ENDP
```

6)ReadKeyBoard -Use "int 21h, function 0Ah", read keyboard, and

"int 21h, function 9h", to display string.

7) To_Ascii PROC

```
;       To_Ascii - Proc to take an unsigned binary value located in the
;       AX Reg and convert it to a string of printable ASCII characters.
;       The results are placed in AsciiOut (5 bytes in length).

        push    ax        ; Save all regesters
        push    bx
        push    cx
        push    dx
        push    si

        mov     cx,5      ; number of out put characters
        mov     si,4      ; Index value to ASCIIOut
        mov     bx,10     ; value to divide by
```

```
Ascii_Loop:
        mov     dx,0       ; clear dx for divide
        div     bx         ; divide by 10
        add     dx,30h     ; convert remainder to ascii
        mov     AsciiOut[si],dl   ; move the char to the output
        dec     si         ; sub 1 from the index
        loop    Ascii_Loop ; do it 5 times

        pop     si         ; Restore all regesters
        pop     dx
        pop     cx
        pop     bx
        pop     ax
        ret
To_Ascii ENDP
```